Arbitrary Arrow Update Logic with Common Knowledge is neither RE nor co-RE

Louwe B. Kuijer
Department of Computer Science, University of Liverpool, UK
LORIA, CNRS/Université de Lorraine, France
louwe.kuijer@liverpool.ac.uk

Arbitrary Arrow Update Logic with Common Knowledge (AAULC) is a dynamic epistemic logic with (i) an arrow update operator, which represents a particular type of information change and (ii) an arbitrary arrow update operator, which quantifies over arrow updates.

By encoding the execution of a Turing machine in AAULC, we show that neither the valid formulas nor the satisfiable formulas of AAULC are recursively enumerable. In particular, it follows that AAULC does not have a recursive axiomatization.

1 Introduction

One of the active areas of study in the field of Dynamic Epistemic Logic is that of *quantified update logics*. Examples of these quantified logics include Arbitrary Public Announcement Logic (APAL) [5], Group Announcement Logic (GAL) [1], Coalition Announcement Logic (CAL) [2], Arbitrary Arrow Update Logic (AAUL) [7], Refinement Modal Logic (RML) [6] and Arbitray Action Model Logic (AAML) [10].

All these logics have an operator that quantifies over all updates of a particular type. For example, in APAL the formula $[!]\varphi$ means "$[\psi]\varphi$ holds for every public announcement ψ" and in AAUL the formula $[\updownarrow]\varphi$ means "$[U]\varphi$ holds for every arrow update U."

One important question about these logics is the decidability of their satisfiability problems. The satisfiability problems of RML and AAML are known to be decidable [10, 6], whereas for the other logics the satisfiability problem is known to be undecidable [9, 3, 4, 8]. More precisely, the satisfiability problems for each of these undecidable logics was shown, by a reduction from the tiling problem, to be co-RE hard. But, so far, it has remained an open question whether they are co-RE.

In other words, while we know that we cannot generate a list of all satisfiable formulas of APAL, GAL, CAL and AAUL, we do not know whether it is possible to generate a list of all valid formulas of these logics.

The question of whether the valid formulas are RE is of particular interest, since a negative result would imply the non-existence of a recursive axiomatization of the logic in question.[1] After all, a recursive axiomatization would allow us to list all valid formulas.

Here, we study a variant AAULC of AAUL, which in addition to all the operators from AAUL also contains a common knowledge operator. We show that the valid formulas of AAULC are not recursively

[1] Finitary axiomatizations for APAL and GAL were proposed in [5] and [1], respectively, but these axiomatizations contain a flaw that renders them unsound.[2] See `http://personal.us.es/hvd/errors.html` for details and a proof of the unsoundness.

[2] We should stress that it is only the finitary axiomatizations that are unsound; the infinitary axiomatization presented in [5] is sound and complete (although the completeness proof contains an error; again, see `http://personal.us.es/hvd/errors.html` for details).

To appear in EPTCS.

enumerable, by a reduction from the non-halting problem. The proof from [8], which shows that the validities of AAUL are not co-RE also applies to AAULC. Still, the non-RE proof in this paper can be extended to a non-co-RE proof with little effort, so in this paper we prove that the validity problem of AAULC is neither RE nor co-RE.

We consider this result to be interesting in its own right. Additionally, and perhaps even more importantly, we also hope that the proof presented here can provide inspiration for proofs about the (non)existence of recursive axiomatizations for APAL, GAL, CAL and AAUL.

The structure of this paper is as follows. First, in Section 2, we define AAULC. Then, in Section 3 we discuss the notation that we will use to describe Turing machines. Finally, in Section 4, we show that both the halting problem and the non-halting problem can be reduced to the validity problem of AAULC.

2 AAULC

Here, we provide the definitions of Arbitrary Arrow Update Logic with Common Knowledge (AAULC). The logics AUL, AAUL and AAULC were designed to reason about information change, but they can also be applied to other domains, most notably that of Normative Systems. A brief overview of the epistemic interpretation of arrow updates is given after the formal definitions. See [11] and [7] for a more in-depth discussion of the applications of AUL and its variants.

Let $\mathscr{P}$ be a countable set of propositional atoms, and let $\mathscr{A}$ be a finite set of agents. We use five agents in our proof, so we assume that $|\mathscr{A}| \geq 5$. The proof can be modified to use only one agent, but such modification requires a lot of complicated notation so we do not so here.

Definition 1. The language $\mathscr{L}_{AAULC}$ of AAULC is given by the following normal forms.

$$\begin{aligned} \varphi &::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \Box_a \varphi \mid C\varphi \mid [U]\varphi \mid [\updownarrow]\varphi \\ u &:= (\varphi, a, \varphi) \\ U &:= \{u_1, \cdots, u_n\} \end{aligned}$$

Where $p \in \mathscr{P}$ and $a \in \mathscr{A}$. The language $\mathscr{L}_{AULC}$ of *Arrow Update Logic with Common Knowledge* is the fragment of $\mathscr{L}_{AAULC}$ that does not contain the $[\updownarrow]$ operator.

We use $\wedge, \Diamond$ and $\langle\updownarrow\rangle$ in the usual way as abbreviations. The formulas of $\mathscr{L}_{AAULC}$ are evaluated on standard multi-agent Kripke models.

Definition 2. A *model* is a triple $\mathscr{M} = (W, R, V)$, where W is a set of worlds, $R : \mathscr{A} \to 2^{W \times W}$ assigns to each agent an accessibility relation and $V : \mathscr{P} \to 2^W$ is a valuation.

Note that we use the class K of all Kripke models. Our reason for using K, as opposed to a smaller class such as $S5$, is that Arrow Update Logic is traditionally evaluated on K, see also [11] and [7]. For the results presented in this paper the choice of models is not very important; the proof that we use would, with some small modifications, also work on $S5$.

We also write $R_a(w)$ for $\{w' \mid (w, w') \in R(a)\}$. The semantics for most operators are as usual, so we omit their definitions. We do provide definitions for $[U]$ and $[\updownarrow]$, since these operators are not as well known as the others.

Definition 3. Let $\mathscr{M} = (W, R, V)$ be a model, and let $w \in W$. Then

$$\begin{aligned} \mathscr{M}, w \models [U]\varphi \quad &\Leftrightarrow \quad \mathscr{M} * U, w \models \varphi \\ \mathscr{M}, w \models [\updownarrow]\varphi \quad &\Leftrightarrow \quad \forall U \in \mathscr{L}_{AULC} : \mathscr{M}, w \models [U]\varphi \end{aligned}$$

where $\mathscr{M} * U$ is given by

$$\begin{aligned} \mathscr{M} * U &:= (W, R * U, V), \\ R * U(a) &:= \{(w_1, w_2) \in R(a) \mid \\ &\qquad \exists (\varphi_1, a, \varphi_2) \in U : \mathscr{M}, w_1 \models \varphi_1 \text{ and } \mathscr{M}, w_2 \models \varphi_2\} \end{aligned}$$

Definition 4. Let $\mathscr{M}_1 = (W_1, R_1, V_1)$ and $\mathscr{M}_2 = (W_2, R_2, V_2)$ be models and let $w_1 \in W_1$, $w_2 \in W_2$. We say that w_1 and w_2 are *AULC*-indistinguishable if for every $\varphi \in \mathscr{L}_{AULC}$, we have $\mathscr{M}, w_1 \models \varphi \Leftrightarrow \mathscr{M}, w_2 \models \varphi$.

An arrow update U represents an information-changing event, of a kind that is sometimes referred to as a *semi-private announcement*. Unlike with a public announcement, the information gained though a semi-private announcement is not common knowledge. It is, however, common knowledge what information is gained under which conditions.

A typical example is the following. Suppose that a and b are playing a game of cards. The cards have been dealt to them, face down. Now, a picks up her cards and looks at them. By doing this, agent a learns what cards she holds. This new information is not common knowledge, since b doesn't learn a's cards, so this event cannot be represented as a public announcement. It is common knowledge, however, under which conditions a gains which information: if a has the Ace of Spades, then she learns that she has the Ace of Spades, and so on. The event of a picking up her cards can therefore be considered a semi-private announcement, which can be represented as an arrow update.

A clause $(\varphi_1, a, \varphi_2) \in U$ says that in every world that satisfies φ_1, the new information gained by agent a is consistent with φ_2. If there are multiple clauses that apply to a single world, we consider them to apply disjunctively, i.e., the new information is consistent with both postconditions: if $(\varphi_1, a, \varphi_2), (\psi_1, a, \psi_2) \in U$ and a world satisfies both φ_1 and ψ_1, then a's new information is consistent with every world that satisfies either φ_2 or ψ_2. We assume that U provides a full description of the new information, so any world consistent with the new information satisfies the postcondition of at least one applicable clause.

Semantically, this means that a transition $(w_1, w_2) \in R(a)$ is retained by the update $[U]$ if and only there is at least one clause $(\varphi_1, a, \varphi_2) \in U$ such that w_1 satisfies φ_1 and w_2 satisfies φ_2. Every other transition is removed from the model.

The example discussed above, where a looks at her cards, is represented by the arrow update

$$U_{cards} := \{(\top, b, \top)\} \cup \{(card, a, card) \mid card \in deck\},$$

where *deck* is the deck from which the cards were dealt. The clause $(\top, b, \top)$ states that b doesn't directly learn anything new: every distribution of cards is consistent with b's new information. The clause $(card, a, card)$, for $card \in deck$ states that if a holds *card*, then by looking at her cards she learns that she holds *card*.

The arbitrary arrow update operator $[\updownarrow]$ quantifies over all arrow updates that do not themselves contain the $[\updownarrow]$ operator. So $\mathscr{M}, w \models [\updownarrow]\varphi$ if and only if $\mathscr{M}, w \models [U]\varphi$ for every $U \in \mathscr{L}_{AULC}$. This restriction to $[\updownarrow]$-free updates keeps the semantics from becoming circular.[3]

The operator $[\updownarrow]$ allows us to ask, inside the object language, whether there is a semi-private announcement that makes a formula true. So, for example, $\mathscr{M}, w \models [\updownarrow](\Box_a p \wedge \neg \Box_b p)$ asks whether, in the situation represented by the model $\mathscr{M}, s$, there is a semi-private announcement that informs a of the truth of p without letting b know that p is true. Recall that an event is a semi-private announcement if it is

[3] Similar restrictions exist in the other quantified update logics. In APAL, for example, the arbitrary arrow update operator $[!]$ quantifies over all public announcements $[\psi]$ where $\psi \in \mathscr{L}_{PAL}$.

common knowledge under what conditions which information is gained. So $[\updownarrow](\Box_a p \wedge \neg\Box_b p)$ is true if and only if there is a method to inform a of the truth of p without informing b, under the assumption that the method itself is common knowledge. Or, in other (and slightly trendier) words, $[\updownarrow](\Box_a p \wedge \neg\Box_b p)$ is true if and only if it is possible to inform a but not b of the truth of p, without relying on *security through obscurity*.

For more examples of the applications of arrow updates and arbitrary arrow updates, see [11] and [7].

Remark 1. In the semantics of AAULC, we let $[\updownarrow]$ quantify over arrow updates in $\mathscr{L}_{AULC}$, so these updates may contain the common knowledge operator C. This means that our $[\updownarrow]$ operator is slightly different from the one in AAUL [7], since the quantification in AAUL is over updates that do not contain C.

This difference is not important for the current paper. All the results presented here still hold if we let $[\updownarrow]$ quantify only over the updates that contain neither $[\updownarrow]$ nor C.

3 Turing Machines

A full discussion of Turing machines is outside the scope of this paper. We assume that the reader is familiar with the basic ideas of a Turing machine; here we only concern us with the notation that we use to represent Turing machines.

Definition 5. A *Turing machine* T is a tuple $T = (\Lambda, S, \Delta)$, where Λ is a finite alphabet such that $\alpha_0 \in \Lambda$, S is a finite set of states such that $s_0, s_{end} \in S$ and $\Delta : \Lambda \times S \to \Lambda \times S \times \{left, remain, right\}$ is a transition function.

We write Δ_1, Δ_2 and Δ_3 for the projections of Δ to its first, second and third components. So if α is the symbol currently under the read/write head and s is the current state, then the machine will write the symbol $\Delta_1(\alpha, s)$, go to the state $\Delta_2(\alpha, s)$ and move the read/write head in direction $\Delta_3(\alpha, s)$.

We assume, without loss of generality, that the state s_0 doesn't re-occur. Furthermore, note that we defined Δ to be a function with $\Lambda \times S$ as domain. So the machine T continues after reaching s_{end}. This is notationally more convenient than letting T terminate once it reaches s_{end}. We don't care about what happens after reaching s_{end}, though.

Definition 6. A Turing machine T *halts* if, when starting in state s_0 with a tape that contains only the symbol α_0, the system reaches the state s_{end}.

It is well known that the halting Turing machines are recursively enumerable, but the non-halting ones are not [12].

The Turing machines that we consider are deterministic, so the execution of a machine T on a tape that only contains α_0 happens in exactly one way. We call this the run of T. One straightforward way to represent this run of T is to consider it as a function $run^T : \mathbb{Z} \times \mathbb{N} \to \Lambda \times S \times \{0,1\}$, where $run^T(n,m) = (\alpha, s, x)$ means that at time m, the symbol in position n on the tape is α, the machine is in state s and the read/write head is at position n if and only if $x = 1$.

For notational reasons, it is convenient to extend this function to $run^T : \mathbb{Z} \times \mathbb{Z} \to \Lambda \times S \times \{0,1\}$, where $run^T(n,m) = (\alpha_0, s_{void}, 0)$ for all $m < 0$. Doing so allows us to avoid a number of special cases that we would otherwise have to consider for $m = 0$. Like with Δ, we use run^T_1, run^T_2 and run^T_3 to refer to the projections to the first, second and third coordinates.

4 The Reduction

For every Turing machine T, we want to represent the unique run run^T in AAULC. In order to do this, we start by encoding certain facts as propositional atoms. For every state $s \in S \cup \{s_{void}\}$ and every element $\alpha \in \Lambda$ of the alphabet, we assume that $s, \alpha \in \mathscr{P}$. We are free to do this, since $\mathscr{P}$ is countably infinite, while S and Λ are finite. As one might expect, we use the propositional atom s to represent the state of the Turing machine at a particular point in time being s, and we use the atom α to represent a particular position of the tape containing the symbol α at a particular point in time. Additionally, we assume that $pos, lpos, rpos \in \mathscr{P}$. These three atoms are used to indicate that a particular point on the tape is the current position of the read/write head, to the left of the current position of the read/write head and to the right of the current position of the read/write head, respectively.

We also assume that there are five agents named $a, right, left, up$ and $down$ in $\mathscr{A}$. Note that we can do this because we assumed that $|\mathscr{A}| \geq 5$. With these preliminaries out of the way, we can define the formula φ_T that represents the Turing machine T in AAULC.

Definition 7. Let $T = (\Lambda, S, \Delta)$ be a Turing machine. The formula φ_T is given by

$$\varphi_T := C\psi_{grid} \wedge C\psi_{sane} \wedge C\psi_T \wedge s_0 \wedge pos,$$

where $\psi grid$, ψ_{sane} and ψ_T are as shown in Tables 1–3.

Table 1: The Formula ψ_{grid}.

$$\begin{aligned}
\psi_{grid} &:= ref_a \wedge direction \wedge inverse \wedge commute \\
D &:= \{left, right, up, down\} \\
INV &:= \{(left, right), (right, left), (up, down), (down, up)\} \\
COMM &:= (\{up, down\} \times \{left, right\}) \cup (\{left, right\} \times \{up, down\}) \\
ref_a &:= \Diamond_a \top \wedge [\updownarrow] \Box_a \Diamond_a \top \\
no_other &:= \bigwedge_{x \in \mathscr{A} \setminus (D \cup \{a\})} \Box_x \bot \\
direction &:= \bigwedge_{x \in D} (\Diamond_x \top \wedge [\updownarrow](\Diamond_x \Diamond_a \top \rightarrow \Box_x \Diamond_a \top)) \\
inverse &:= [\updownarrow](\Diamond_a \top \rightarrow \bigwedge_{(x,y) \in INV} \Box_x \Box_y \Diamond_a \top) \\
commute &:= [\updownarrow] \bigwedge_{(x,y) \in COMM} (\Diamond_x \Diamond_y \Diamond_a \top \rightarrow \Box_y \Box_x \Diamond_a \top)
\end{aligned}$$

Table 2: The Formula ψ_{sane}.

$$\begin{aligned}
\psi_{sane} &:= position_1 \wedge position_2 \wedge one_state \wedge same_state \wedge \\
&\quad one_symbol \wedge void_state \wedge initial_symbol \wedge unchanged \\
position_1 &:= \neg(pos \wedge lpos) \wedge \neg(pos \wedge rpos) \wedge \neg(rpos \wedge lpos) \\
position_2 &:= ((pos \vee rpos) \rightarrow \Box_{right} rpos) \wedge ((pos \vee lpos) \rightarrow \Box_{left} lpos) \\
one_state &:= \bigvee_{s \in states} (s \wedge \bigwedge_{s' \in states \setminus \{s\}} \neg s) \\
same_state &:= \bigwedge_{s \in states} (s \rightarrow (\Box_{left} s \wedge \Box_{right} s)) \\
one_symbol &:= \bigvee_{\alpha \in symbols} (\alpha \wedge \bigwedge_{\beta \in symbols \setminus \{\alpha\}} \neg \beta) \\
void_state &:= (s_0 \vee s_{void}) \rightarrow \Box_{down} s_{void} \\
initial_symbol &:= s_0 \rightarrow \alpha_0 \\
unchanged &:= \bigwedge_{\alpha \in symbols} ((\neg pos \wedge \alpha) \rightarrow \Box_{up} \alpha)
\end{aligned}$$

Table 3: The Formula ψ_T.

$$
\begin{array}{rl}
\psi_T := & \mathit{position\_change}_T \wedge \mathit{state\_change}_T \wedge \mathit{symbol\_change}_T \\
\mathit{position\_change}_T := & \bigwedge_{\{(s,\alpha)|\Delta_3(s,\alpha)=\mathit{left}\}}((\mathit{pos} \wedge s \wedge \alpha) \rightarrow \Box_{\mathit{up}}\Box_{\mathit{left}}\mathit{pos}) \wedge \\
& \bigwedge_{\{(s,\alpha)|\Delta_3(s,\alpha)=\mathit{right}\}}((\mathit{pos} \wedge s \wedge \alpha) \rightarrow \Box_{\mathit{up}}\Box_{\mathit{right}}\mathit{pos}) \wedge \\
& \bigwedge_{\{(s,\alpha)|\Delta_3(s,\alpha)=\mathit{remain}\}}((\mathit{pos} \wedge s \wedge \alpha) \rightarrow \Box_{\mathit{up}}\mathit{pos}) \\
\mathit{state\_change}_T := & \bigwedge_{s' \in \mathit{states}} \bigwedge_{\{(s,\alpha)|\Delta_2(s,\alpha)=s'\}}((\mathit{pos} \wedge s \wedge \alpha) \rightarrow \Box_{\mathit{up}}s') \\
\mathit{symbol\_change}_T := & \bigwedge_{\beta \in \mathit{symbols}} \bigwedge_{\{(s,\alpha)|\Delta_1(s,\alpha)=\beta\}}((\mathit{pos} \wedge s \wedge \alpha) \rightarrow \Box_{\mathit{up}}\beta)
\end{array}
$$

This formula may look somewhat intimidating, but apart from ψ_{grid} all named formulas are very simple encodings of aspects of a Turing machine. The formula ψ_{grid}, as the name might suggest, encodes a $\mathbb{Z} \times \mathbb{Z}$ grid.

We first show that φ_T is satisfiable. After that, we show that any model that satisfies φ_T contains a representation of run_T.

Lemma 1. *For every Turing machine T, the formula φ_T is satisfiable.*

Proof. Let $\mathscr{M} = (W, R, V)$ be given as follows. Take $W = \mathbb{Z} \times \mathbb{Z}$. For every direction $x \in D$ let $(w, w') \in R(x)$ if and only if w' is immediately to the x of w, and let $R(a) = \{(w, w) \mid w \in W\}$. For $x \notin D \cup \{a\}$, let $R(x) = \emptyset$. Now, for any $\alpha \in \Lambda$ and $s \in S \cup \{s_{\mathit{void}}\}$, let $V(\alpha) = \{(n,m) \mid \mathit{run}_1^T(n,m) = \alpha\}$ and $V(s) = \{(n,m) \mid \mathit{run}_2^T(n,m)$. Furthermore, let $V(\mathit{pos}) = \{(n,m) \mid \mathit{run}_3^T(n,m) = 1\}$, $V(\mathit{lpos}) = \{(n,m) \mid \exists n' > n : \mathit{run}_3^T(n',m) = 1\}$ and $V(\mathit{rpos}) = \{(n,m) \mid \exists n' < n : \mathit{run}_3^T(n',m) = 1\}$. (In other words, *lpos* holds if you are to the left of *pos* and *rpos* holds if you are to the right of *pos*.)

We claim that $\mathscr{M}, (0,0) \models \varphi_T$. Since $\mathit{run}^T(0,0) = (\alpha_0, s_0, 1)$, we have $\mathscr{M}, (0,0) \models \mathit{pos} \wedge s_0$. This leaves the conjuncts $C\psi_{grid}, C\psi_{sane}$ and $C\psi_T$. We start by looking at $C\psi_{sane}$.

The conjuncts of ψ_{sane} hold under the following conditions.

- $\mathit{position}_1$ holds if being the position of the head, being to the right of the head and being to the left of the head are mutually exclusive.
- $\mathit{position}_2$ holds if all worlds to the left of the head satisfy *lhead* and all worlds to the right of the head satisfy *rhead*.
- *initial_symbol* holds if, at the initial state s_0, the entire tape contains the symbol α_0.
- *one_state* holds if at every (n,m), the system is in exactly one state.
- *same_state* holds if for every $n, m, k \in \mathbb{Z}$, the worlds (n,m) and (k,m) are in the same state. (So the state depends only on time, not on the tape position.)
- *one_symbol* holds if at every time m, every position n contains exactly one symbol.
- *void_state* holds if at every time before s_0, the system was in the dummy state s_{void}.
- *unchanged* holds if every symbol that is not under the read/write head remains unchanged.

All of these conditions are satisfied, because the valuation of $\mathscr{M}$ was derived from the run of a Turing machine. So $\mathscr{M}, (0,0) \models C\psi_{sane}$. Now, consider the conjuncts of ψ_T.

- $\mathit{position\_change}_T$ holds if the read/write head moves in the appropriate direction, as specified by Δ.
- $\mathit{state\_change}_T$ holds if the state changes as specified by Δ.

- *symbol_change*$_T$ holds if the symbol under the read/write head is written as specified by Δ.

These conditions are also satisfied, because the valuation of $\mathscr{M}$ was derived from *run*$_T$. So $\mathscr{M},(0,0) \models C\psi_T$. Left to show is that $\mathscr{M},(0,0) \models \psi_{grid}$.

So take any $w \in W$. The world w itself is the only a-successor of w. So we have $\mathscr{M},w \models \Diamond_a\top$. Furthermore, it is impossible for any arrow update to retain the a-arrow from w while removing the a-arrows from its successor, since they are the same a-arrow. It follows that $\mathscr{M},w \models [\updownarrow]\Box_a\Diamond_a\top$. So we have shown that $\mathscr{M},w \models \mathit{ref}_a$.

We have defined $R_x = \emptyset$ for all $x \notin D \cup \{a\}$, so we also have $\mathscr{M},w \models \mathit{no\_other}$.

Now, consider *direction*. Take any $x \in D$. There is a x-arrow from w to the world w' to its x, so $\mathscr{M},w \models \Diamond_x\top$. Furthermore, since this w' is the only x-successor of w, it follows that it is impossible to retain an a-arrow on one x-successor of w while removing all a-arrows from another. So $\mathscr{M},w \models [\updownarrow](\Diamond_x\Diamond_a\top \to \Box_x\Diamond_a\top)$. This holds or every $x \in D$, so $\mathscr{M},w \models \mathit{direction}$.

For opposite directions x and y, there is exactly one x-y-successor of w, namely w itself. It follows that it is impossible for any arrow update to retain the a-arrow on w while removing all a-arrows from its x-y-successor, so $\mathscr{M},w \models \mathit{inverse}$.

Finally, for perpendicular directions x and y there is exactly one x-y-successor w' of w, and this w' is also the unique y-x-successor of w. So it is impossible for an arrow update to retain the a-arrow from the x-y-successor while removing it from some y-x-successor. So $\mathscr{M},w \models \mathit{commute}$.

This completes the proof that $\mathscr{M},w \models \psi_{grid}$ and therefore the proof that $\mathscr{M},w \models \varphi_T$. So φ_T is satisfiable. □

Lemma 2. *If $\mathscr{M},w_0 \models \varphi_T$, then $\mathscr{M},w_0 \models C\neg s_{end}$ if and only if T is non-halting.*

Proof. Suppose $\mathscr{M},w_0 \models \varphi_T$. Then, by definition, $\mathscr{M},w_0 \models \psi_{grid} \wedge C\psi_{sane} \wedge C\psi_T \wedge pos \wedge s_0$. We will first show that $\mathscr{M},w_0 \models C\psi_{grid}$ implies that the model $\mathscr{M}$ is grid-like. Then, we will show that the remaining subformulas imply that the model $\mathscr{M}$ represents *run*^{T}.

By the $\Diamond_a\top$ conjunct of *ref*$_a$, every reachable world w has at least one a-successor. If any a-successor of w is *AULC*-distinguishable from w, then it would be possible for an arrow update to remove the a-arrow from this successor while retaining the a-arrow from w. This would contradict the $[\updownarrow]\Box_a\Diamond_a\top$ conjunct of *ref*$_a$.

Now, for any $x \in D$, consider the x-successors of w, of which there is at least one by the $\Diamond_x\top$ part of *direction*. If any of these successors were *AULC*-distinguishable, it would be possible to remove the a-arrow from one of them but not from the other. This would contradict the $[\updownarrow](\Diamond_x\Diamond_a\top \to \Box_x\Diamond_a\top)$ part of *direction*.

For any opposite directions x and y, consider any x-y-successor w' of w. If any *AULC* formula could distinguish between w' and w', it would be possible for an arrow update to retain the a-arrow from w while removing the a-arrow from w', which would contradict *inverse*.

Finally, for any perpendicular direction x and y, consider any x-y- and y-x-successors of w. If these successors were *AULC*-distinguishable, it would be possible for an arrow update to remove one while retaining the other, contradicting *commute*.

Taken together, the above facts imply that $\mathscr{M}$ contains a representation of a grid $\mathbb{Z} \times \mathbb{Z}$ where, for every $x \in D$, we have $w' \in R_x(w)$ if and only if w' is to the x of w. Furthermore, if w represents (n,m) then so does every a-successor of w. Every world (n,m) may be represented by multiple worlds in $\mathscr{M}$, but all the worlds that represent a single grid point are *AULC*-indistinguishable from one another. Furthermore, the formula *no_other* implies that we cannot escape this grid, every reachable world represents some grid point (n,m).

The remaining subformulas of φ_T guarantee that this grid encodes run_T. First, consider ψ_{sane}. This formula enforces a number of general sanity constraints.

- The formula $position_1$ says that being the current position of the read/write head, being to the right of the position of the head and being to the left of the position of the head are mutually exclusive.
- The formula $position_2$ says that if you are either at the current position of the read/write head or to the right of the current position, then if you go further to the right then you will be to the right of the current position. Similarly, it says that if you are either at the current position of the head or to the left of it and go further left, then you will be to the left of the head. Together with $position_1$, this guarantees that the read/write head is in at most one position at any time step. (Ensuring that the head is in at least one position at every time is done later).
- The formula *one_state* says that every world is in exactly one state.
- The formula *same_state* says that if a world is in state s, then the worlds to the left and right are also in state s. So all the worlds that represent a single time step satisfy the same state. Together with *one_state*, this implies that every time step is associated with exactly one state.
- The formula *one_symbol* says that every world satisfies exactly one symbol.
- The formula *void_state* says that every time before the initial state s_0 is in the dummy state s_{void}. So the worlds satisfying s_0 are where the computation starts.
- The formula *initial_symbol* says that the s_0 worlds satisfy α_0, so if the system is in the initial state s_0, then the tape is empty.
- Finally, the formula $symbol\_unchanged_T$ guarantees that the symbol remains unchanged everywhere other than under the read/write head.

Now, consider ψ_T, which forces the transitions to satisfy Δ.

- The formula $position\_change_T$ guarantees that the read/write head moves in the correct direction, depending on the current symbol under the head and the current state.
- The formula $state\_change_T$ guarantees that the next state is as specified by T.
- The formula $symbol\_change_T$ guarantees that the correct symbol is written to the tape, as specified by T.

The last two conjuncts of φ_T do not contain a common knowledge operator. They state that the world w_0 satisfies *pos* and s_0. So w_0 represents the point $(0,0)$. Because the rules of T always require the read/write head to stay in he same position or to move to the left or right, this also implies that at every time after w_0 the head is in at least one position.

Taken together, the above shows that the valuation on the grid represents run_T. So the grid contains a s_{end} state if and only if T is halting. Since every reachable state is part of the grid, it follows that $\mathscr{M}, w_0 \models C\neg s_{end}$ if and only if T is non-halting. □

Theorem 1. *The formula $\varphi_T \to C\neg s_{end}$ is valid if and only if T is non-halting. Furthermore, $\varphi_T \to \neg C\neg s_{end}$ is valid if and ony if T is halting.*

Proof. Suppose that T is non-halting. Then, by Lemma 2, we have $\models \varphi_T \to C\neg s_{end}$. Furthermore, since φ_T is satisfiable, this implies that $\not\models \varphi_T \to \neg C\neg s_{end}$.

Suppose, on the other hand, that T is halting. Then, by Lemma 2, we have $\models \varphi_T \to \neg C\neg s_{end}$. Furthermore, since φ_T is satisfiable, this implies that $\not\models \varphi_T \to C\neg s_{end}$. □

Corollary 1. *The set of valid formulas of AAULC is neither RE not co-RE.*
Corollary 2. *AAULC does not have a finitary axiomatization.*

5 Conclusion

The validity problems for the quantified update logics APAL, GAL, CAL and AAUL are known not to be co-RE. It is not currently known whether these problems are RE. This question is particularly relevant because if the validity problem of a logic is not RE, then that logics cannot have a recursive axiomatization.

The logic AAULC adds a common knowledge operator to AAUL. Here, we showed that the validity problem of AAULC is not RE, using a reduction from the non-halting problem of Turing machines. This reduction uses the common knowledge operator C, so it does not immediately follow that the validity problem of AAUL is not RE. Still, we believe that the proof presented here can be adapted for AAUL.

It is less clear whether our reduction could be adapted for APAL, GAL and CAL. Still, it seems worthwhile to attempt to modify this reduction for APAL, GAL and CAL. If such an attempt succeeds, it would show that these logics are nor recursively axiomatizable. Or if the attemp fails, then the way in which it fails might provide a hint about how to prove that the validity problems of these logics are RE.

References

[1] Thomas Ågotnes, Philippe Balbiani, Hans van Ditmarsch & Pablo Seban (2010): *Group announcement logic*. *Journal of Applied Logic* 8(1), pp. 62 – 81, doi:10.1016/j.jal.2008.12.002.

[2] Thomas Ågotnes & Hans van Ditmarsch (2008): *Coalitions and Announcements*. In Padgham, Parkes, Müller & Parsons, editors: *Proc. of 7th Int. Conf. on Autonomous Agents and Multi-agent Systems (AAMAS 2008)*, pp. 673–680.

[3] Thomas Ågotnes, Hans van Ditmarsch & Tim French (2014): *The Undecidability of Group Announcements*. In Ana Bazzan Michael Huhns Alessio Lomuscio, Paul Scerri, editor: *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, pp. 893–900.

[4] Thomas Ågotnes, Hans van Ditmarsch & Tim French (2016): *The Undecidability of Quantified Announcements*. *Studia Logica* 104(4), pp. 597–640, doi:10.1007/s11225-016-9657-0.

[5] Philippe Balbiani, Alexandru Baltag, Hans van Ditmarsch, Andreas Herzig, Tomohiro Hoshi & Tiago de Lima (2008): *'Knowable' as 'known after an announcement'*. *Review of Symbolic Logic* 1(3), pp. 205–334, doi:10.1017/S1755020308080210.

[6] Laura Bozzelli, Hans van Ditmarsch, Tim French, James Hales & Sophie Pinchinat (2014): *Refinement modal logic*. *Information and Computation* 239, pp. 303–339, doi:10.1016/j.ic.2014.07.013.

[7] Hans van Ditmarsch, Wiebe van der Hoek, Barteld Kooi & Louwe B. Kuijer (2017): *Arbitrary Arrow Update Logic*. *Artificial Intelligence* 242, pp. 80–106, doi:10.1016/j.artint.2016.10.003.

[8] Hans van Ditmarsch, Wiebe van der Hoek & Louwe B. Kuijer (2016): *The Undecidability of Arbitrary Arrow Update Logic*. ArXiv:1609.05686.

[9] Tim French & Hans van Ditmarsch (2008): *Undecidability for arbitrary public announcement logic*. In C. Areces & R. Goldblatt, editors: *Proceedings of the seventh conference 'Advances in Modal Logic'*, College Publications, London, pp. 23–42.

[10] James Hales (2013): *Arbitrary action model logic and action model synthesis*. In: *28th Annual ACM/IEEE Symposium on Logic in Computer Science*, IEEE, pp. 253–262, doi:10.1109/LICS.2013.31.

[11] Barteld Kooi & Bryan Renne (2011): *Arrow update logic*. *Review of Symbolic Logic* 4(4), pp. 536–559, doi:10.1017/S1755020311000189.

[12] Alan M. Turing (1937): *On Computable Numbers, with an Application to the Entscheidungsproblem*. *Proceedings of the London Mathematical Society* s2-42(1), pp. 230–265, doi:10.1112/plms/s2-42.1.230.