Arrow Update Synthesis

Hans van Ditmarsch[*,†], Wiebe van der Hoek[‡], Barteld Kooi[§], Louwe B. Kuijer[¶]

Abstract

In this contribution we present *arbitrary arrow update model logic* (AAUML). This is a *dynamic epistemic logic* or *update logic*. In update logics, static/basic modalities are interpreted on a given relational model whereas dynamic/update modalities induce transformations (updates) of relational models. In AAUML the update modalities formalize the execution of *arrow update models*, and there is also a modality for quantification over arrow update models. Arrow update models are an alternative to the well-known action models. We provide an *axiomatization* of AAUML. The axiomatization is a rewrite system allowing to eliminate arrow update modalities from any given formula, while preserving truth. Thus, AAUML is decidable and equally expressive as the base multi-agent modal logic. Our main result is to establish *arrow update synthesis*: if there is an arrow update model after which φ, we can *construct* (synthesize) that model from φ. We also point out some pregnant differences in *update expressivity* between arrow update logics, action model logics, and refinement modal logic, and we provide a novel axiomatization for refinement modal logic.

Keywords: modal logic, synthesis, dynamic epistemic logic, expressivity

1 Introduction

Modal logic In modal logic we formalize that propositions may not be merely true or false, but that they are necessarily or possibly true or false, or that they may be desirable, or forbidden, or true later, or never, or that they are *known*. A common setting is for such modal propositions to be interpreted in *relational models*, also known as *Kripke models*. They consist of a domain of abstract objects, called states or worlds; then, given a set of labels, often representing agents, for each such agent a binary relation between those states; and, finally, a valuation of atomic propositions on the domain, typically seen as a

[*]ReLax/LORIA, CNRS, France/India, `hans.van-ditmarsch@loria.fr`

[†]Hans van Ditmarsch is corresponding author. We acknowledge support from ERC project EPS 313360.

[‡]Computer Science, University of Liverpool, United Kingdom, `Wiebe.Van-Der-Hoek@liverpool.ac.uk`

[§]Department of Philosophy, University of Groningen, the Netherlands, `b.p.kooi@rug.nl`

[¶]Computer Science, University of Liverpool, United Kingdom, `Louwe.Kuijer@liverpool.ac.uk` unary relation, i.e., a property satisfied on a subset of the domain. The truth of a modal proposition is relative to a state in the relational model, called the actual state or the *point* of the model. The unit of interpretation is thus a *pointed model*: a pair consisting of a relational model and an actual state.

If a pair (s, s') in the relation for a this can mean that after executing action a in state s the resulting state is s'. But it can also mean that agent a considers state s' desirable in case she is in state s. The interpretation that we focus on, is that of *information*. That is, it is consistent with a's information in state s that the state would be s'. In state s it is true that agent a *knows* φ (or *believes* φ, depending on the properties of the relation), notation $\Box_a\varphi$, if the formula φ is true in all states s' accessible from s, i.e., for all s' with (s, s') in the relation for a. The modal logics using that kind of interpretation of modalities are called epistemic logics [6, 24].

As an example, consider two agents a, b (commonly known to be) uncertain about the truth of a propositional variable p. The uncertainty of a and b can be pictured as follows. We 'name' the states with the value of the variable p. The actual state is framed. Pairs in the accessibility relation are visualized as labelled *arrows*. In the actual state: p is true, agent a does not know p because she considers a state possible wherein p is false (formally $\neg\Box_a p$), agent a also does not know $\neg p$ because she considers a state possible wherein p is true (formally $\neg\Box_a\neg p$, also written as $\Diamond_a p$), and similarly for agent b. Agent a also knows that she is ignorant about p, as this is true in both states that she considers possible. The accessibility relations for a and b are both equivalence relations. This is always the case if the modalities represent knowledge.

$$\begin{tikzcd}
\neg p \arrow[loop above, "ab"] \arrow[r, leftrightarrow, "ab"] & \boxed{p} \arrow[loop above, "ab"]
\end{tikzcd}$$

Update logic In this work we focus on modal logics that are *update logics*. Apart from the modalities that are interpreted *in* a relational model, they have other modalities that are interpreted by *transforming* a relational model (and by then interpreting the formulas bound by that modality in the transformed model). If the modal logic is an epistemic logic, update logics are called *dynamic epistemic logics*. To distinguish them we call the former *static* and we call the latter *dynamic*.

The *updates* X that we consider can be defined as transformers of relational models. This transformation induces a binary update relation between pointed models. To an update relation corresponds an *update modality* (often also called update) that is interpreted with this relation, so we can see those as $[X]$ or $\langle X\rangle$, where $[X]\varphi$ means that φ is true in all pointed models transformed according to the X relation, and $\langle X\rangle\varphi$ that there is a pair of pointed models in the relation. Given a relational model we can change its domain of states, the relations between the states, or the valuations of atomic propositions, or two or more of those at the same time. There are therefore many options for change. Change the valuation of a model is also known as *factual change* [26, 21]. Update involving factual change is an interesting topic, but it is outside the scope of the current paper.

Public announcement logic The basic update for states is the model restriction, and the basic update operation interpreted as a model restriction is a *public announcement.* The logic with epistemic modalities and public announcements is *public announcement logic* (PAL) [18, 5]. A public announcement of φ restricts the domain to all states where the announced formula φ is true, thereby decreasing the uncertainty of the agents. As a result of the domain restriction, the relations and the valuation are adjusted in the obvious way. A condition for the transformation is that the actual state is in the restriction. This means that the announcement formula is true when announced.

As an example, after the public announcement of p, both a and b know that p:

$$\begin{tikzcd}
\neg p \arrow[loop above, "ab"] \arrow[r, leftrightarrow, "ab"] & \boxed{p} \arrow[loop above, "ab"] & \Rightarrow & \boxed{p} \arrow[loop above, "ab"]
\end{tikzcd}$$

Arrow update logic The basic update for relations is the relational restriction, i.e., a restriction of the *arrows*: a pair in the relation is called an ‘arrow’. This leaves all states intact, although some may have become unreachable. In *arrow update logic* (AUL), proposed in [15] we specify which arrows we wish to preserve, by way of specifiying what formulas should be satisfied at the source (state) of the arrow and the target (state) of the arrow. This determines the model transformation. Such a specification is called an *arrow update.* The logic AUL contains modalities for arrow updates.

Given initial uncertainty about p with both agents, a typical arrow update is the action wherein Anne (a) opens an envelope containing the truth about p while Bill (b) observes Anne reading the contents of the letter. We preserve all arrrows satisfying one of $p \to_a p, \neg p \to_a \neg p$, and $\top \to_b \top$. Therefore, only two arrows disappear, $\neg p \to_a p$ and $p \to_a \neg p$.

$$\begin{tikzcd}
\neg p \arrow[loop above, "ab"] \arrow[r, leftrightarrow, "ab"] & \boxed{p} \arrow[loop above, "ab"] & \Rightarrow & \neg p \arrow[loop above, "ab"] \arrow[r, leftrightarrow, "b"] & \boxed{p} \arrow[loop above, "ab"]
\end{tikzcd}$$

The boundary between state elimination and arrow elimination is vague. If p is true, the following arrow update with $\top \to_a p, \top \to_b p$ is the same update as a public announcement of p. This is because there is no arrow from the p state to the $\neg p$ state after the update. Therefore, if p is true, the $\neg p$ state does not matter. In another formalism this arrow update is known as the arrow elimination semantics of public announcement [11, 14].

$$\begin{tikzcd}
\neg p \arrow[loop above, "ab"] \arrow[r, leftrightarrow, "ab"] & \boxed{p} \arrow[loop above, "ab"] & \Rightarrow & \neg p \arrow[r, "b"] & \boxed{p} \arrow[loop above, "ab"]
\end{tikzcd}$$

Generalizations In PAL and AUL the complexity (the number of states) of the relational model cannot increase. By generalizing the mechanism underlying state elimination and arrow elimination we can achieve that, and thus express more model transformations. This increases their *update expressivity.* From the perspective of information change, this adds uncertainty about what is happening. We obtain *action models* [5] as a generalization of public announcements, and *arrow update models* [16] as a generalization of arrow updates.

Action model logic *Action model logic* (AML) was proposed by Baltag, Moss and Solecki in [5]. An *action model* is like a relational model but the elements of the domain are called *actions* instead of states, and instead of a valuation a *precondition* is assigned to each domain element. The transformed relational model is then the modal product of the relational model and the action model, restricted to (state,action) pairs where the action can be executed in that state. We refer to Section 6 for a formal introduction.

An example is the action as above wherein Anne reads the contents of a letter containing p or $\neg p$, but now with the increasing uncertainty that Bill is uncertain whether Anne has read the letter (and that they are both aware of these circumstances). The action model is not depicted (details are in Section 6). The model transformation is as follows. In the resulting framed state, a knows that p, but b considers it possible that a is uncertain about p, i.e., $\Box_a p \wedge \Diamond_b \neg(\Box_a p \vee \Box_a \neg p)$. In the figure we assume transitivity of the relation for b.

$$\begin{tikzcd}
 & & & \neg p \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "b"] \arrow[d, leftrightarrow, "b"'] & \boxed{p} \arrow[loop right, "ab"] \arrow[d, leftrightarrow, "b"] \\
\neg p \arrow[loop above, "ab"] \arrow[r, leftrightarrow, "ab"] & \boxed{p} \arrow[loop above, "ab"] & \Rightarrow & \neg p \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "ab"] & p \arrow[loop right, "ab"]
\end{tikzcd}$$

Similar logics (or semantics) for action composition are found in [22, 13, 21, 2, 32]. Action model logic is often referred to as (the) dynamic epistemic logic. As said, we use the latter more generally, namely to denote any update logic with an epistemic interpretation.

Arrow update model logic *Generalized arrow update logic* [16] is a (indeed) generalization of arrow update logic where the dynamic modalities for information change formalize execution of (pointed) *arrow update models*, structures akin to the *action models* of action model logic. In this contribution, instead of generalized arrow update logic we call it *arrow update model logic* (AUML). The arrow updates of [15] correspond to singleton arrow update models. The next Section 2 formally introduces them. The above is also an example of arrow update model execution — Section 6 explains how to get action models from arrow update models and vice versa, and to what extent they define the same update.

Quantification over information change Another extension of update logics is with quantification over updates. *Arbitrary public announcement logic* (APAL) adds quantification over public announcements to PAL [4]. *Arbitrary arrow update logic* (AAUL) [28] extends arrow update logic with quantifiers over information change induced by arrow updates: it contains dynamic modalities formalizing that *there is an arrow update after which* φ. *Arbitary action model logic* (AAML) by Hales [12] add quantifiers over action models to AML. In *arbitrary arrow update model logic* (AAUML), the topic of this paper, we add quantifiers over arrow update models to the logical language. It is like Hales' arbitrary action model logic. *Refinement modal logic* (RML) [8] has a modality representing quantification over updates, but does not have (deterministic/concrete) update modalities in the object language to quantify over. We will show that the AAML and AAUML quantifier behaves much (but not quite) like the refinement quantifier in RML. Section 7 is devoted to it.

Figure 1 gives an overview of the different logics discussed in the paper, in their relation to AAUML. The four logics in the left square are based on state manipulation, the four logics in the right square are based on arrow manipulation. Entirely on the left we find the base modal logic ML and the logic RML, that is also arrow manipulating.

All these logics are equally expressive as ML and are decidable, which can be shown by truth-preserving rewriting procedures to eliminate updates (for AAUML this is one of the results of the paper), except for APAL and AAUL, which are more expressive and undecidable [4, 9, 28, 29]. However, the logics greatly differ in update expressivity, as the typical examples above already demonstrated. See also Sections 5 — 7.

There are many other updates and update logics that we do not consider in this paper. In particular we do not consider updates X that can only be defined as *pointed* model transformers (that is, they cannot be *globally* defined on the entire model; they are defined *locally*: how they transform the model depends on the actual state). If such were the definition of an update, even the interpretation of a static modality can be seen as an update, namely transforming the model with point s into the model with point s', where the point has shifted given a pair (s, s') in the relation for an agent. Such local update logics are often more expressive than modal logic, are often undecidable, and may lack axiomatizations. Examples are [20, 3, 1]. In [1] not only relational *restriction* is considered but also relational *expansion* ('bridge') and relational *change* that is neither restriction nor expansion, such as reversing the direction of arrows ('swap'). It should finally be noted that the distinction between static modalities, interpreted in a model, and dynamic modalities, interpreted as updates, is not rigid: unifying perspectives include [21].

Synthesis For these update logics we can ask whether there is an update that achieves a certain goal. For the logics without quantification this question cannot be asked in the object language but only meta-logically. That is, we can ask whether there is an update X such that $\langle X \rangle \varphi$ is true. For the update logics with quantification this question can be asked in the object language. Let $\langle ? \rangle$ be (the existential version of) that quantifier. Then $\langle ? \rangle \varphi$ asks whether there is an update X that makes φ true.

$$\begin{tikzcd}
\mathrm{ML} \arrow[r] \arrow[d] \arrow[rrr, bend left] & \mathrm{PAL} \arrow[r] \arrow[d] & \mathrm{AML} \arrow[d] & \mathrm{AUL} \arrow[r] \arrow[d] & \mathrm{AUML} \arrow[d] \\
\mathrm{RML} & \mathrm{APAL} \arrow[r] & \mathrm{AAML} & \mathrm{AAUL} \arrow[r] & \mathrm{AAUML}
\end{tikzcd}$$

Figure 1: An overview of update logics discussed in the paper. Horizontal arrows informally represent *more complex* updates. Vertical arrows informally represent *quantification over* updates. The arrows can be interpreted as syntactic extensions (modulo the names of quantifiers) or as semantic generalizations. Assume transitive closure.

Only knowing *whether* there is an update that achieves a goal is not very satisfying; we would also like to know *which* update, if any, achieves the goal. So we would like to know not only whether the goal is achievable but also how it can be achieved. The process of constructing this update that achieves the goal is known as *synthesis*.

Formally, the synthesis problem for a given type of update takes as input a formula φ, and gives as output an update X of that type such that, whenever φ can be achieved, then X achieves φ. In symbols, this is the validity of $\langle ? \rangle\varphi \to \langle X \rangle\varphi$.

This is a rather strong goal. We do not consider it sufficient to find, for every pointed model (M, s), an update $X_{(M,s)}$ such that (M, s) satisfies $\langle ? \rangle\varphi \to \langle X_{(M,s)} \rangle\varphi$. We want one single update X_φ that achieves φ in every model where φ is achievable. Because this goal is so strong, there is, in general, no guarantee that synthesis is possible.

For PAL this strong kind of synthesis is impossible. If (M_1, s_1) satisfies $\langle \psi_1 \rangle\varphi$ and (M_2, s_2) satisfies $\langle \psi_2 \rangle\varphi$, so if φ can be achieved in two different situations using two different public announcements, then there is typically no unifying public announcement ψ such that (M_1, s_2) satisfies $\langle \psi \rangle\varphi$ and (M_2, s_2) satisfies $\langle \psi \rangle\varphi$.

For example, consider the four-state model below; $\overline{p}$ means that p is false in that state, etc. Both states where p, q, r are all true satisfy that $\langle ? \rangle(\Box_a p \wedge \neg\Box_b p)$. In the top-left pqr-state this is true because $\langle q \rangle(\Box_a p \wedge \neg\Box_b p)$ is true, whereas in the bottom-right pqr-state this is true because $\langle r \rangle(\Box_a p \wedge \neg\Box_b p)$ is true. However, there is no announcement φ such that $\langle \varphi \rangle(\Box_a p \wedge \neg\Box_b p)$ is truth in both pqr-states. Assuming that there were such an announcement easily leads to a contradiction.

$$\begin{tikzcd}
pqr \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "a"] \arrow[d, leftrightarrow, "b"'] & \overline{p}\overline{q}r \arrow[loop right, "ab"] \arrow[d, leftrightarrow, "b"] \\
\overline{p}q\overline{r} \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "a"] & pqr \arrow[loop right, "ab"]
\end{tikzcd}$$

For AUL this strong kind of synthesis is also not possible. But, somewhat surprisingly, in [12], Hales showed that this synthesis is possible for AML. This result was surprising for the following reason. Hales obtained his synthesis result with refinement modalities as quantifiers. It was already known that finite action model execution results in a refinement of the current relational model, but also that the other direction does not hold: there are refinements (i.e., updates) that can only be achieved by executing an infinite action model [25]. However, as the synthesis is with respect to making a given formula φ true, a finite syntactic object, synthesis for AML was after all possible.

In this contribution we show that synthesis is also possible for AUML. That is, for a given goal formula φ, we can construct an arrow update model X such that

> For all (M, s): there is an arrow update model Y such that (M, s) satisfies $\langle Y \rangle \varphi$, if and only if (M, s) satisfies $\langle X \rangle \varphi$.

In AAUML we also have a quantifier over arrow update models. Therefore, in that logic the synthesis translates to the above-mentioned validity $\langle ? \rangle \varphi \rightarrow \langle X \rangle \varphi$. In AUML / AAUML we synthesize a (single-)pointed arrow update model, whereas for AAUML Hales synthesizes a *multi*-pointed action model, and it can be easily shown that this cannot be single-pointed.

Results in the paper In this contribution we present *arbitrary arrow update model logic* (AAUML), that further extends arrow update model logic AUML, namely with dynamic modalities formalizing that *there is an arrow update* **model** *after which* φ. For this logic AAUML we obtain various results. We provide an *axiomatization* of AAUML. The axiomatization is a rewrite system allowing to eliminate dynamic modalities from any given formula, while preserving truth. Thus, unlike AAUL, AAUML is decidable, and equally expressive as multi-agent modal logic. We establish *arrow update model synthesis*: if there is an arrow update model after which φ, we can *construct* (synthesize) that model from φ. We define a notion of *update expressivity* and we determine the relative update expressivity of AAUML and other arrow update logics and action model logics. Finally, we also compare AAUML to RML, where we provide a novel axiomatization for RML.

Overview of content Section 2 presents the syntax and semantics of arbitrary arrow update model logic, AAUML, and elementary structural notions. In Section 3 we describe the procedure for synthesizing arrow update models. In that section we also introduce a number of validities that are useful when introducing an axiomatization for AAUML, which we do in the subsequent Section 4. Section 5 introduces the notion of *update expressivity*. Section 6 compares AAUML and AAML, and in particular their update expressivity. This comparison also includes examples of arrow update models that have exponentially larger corresponding action models. Section 7 compares AAUML to RML.

2 Arbitrary arrow update model logic

Throughout this contribution, let A be a finite set of *agents* and let P be a disjoint countably infinite set of *propositional variables* (or *atoms*).

2.1 Structures

A *relational model* is a triple $M = (S, R, V)$ with S a non-empty *domain* of *states*, R a function assigning to each agent $a \in A$ an *accessibility relation* $R_a \subseteq S \times S$, and $V : P \to S$ a *valuation function* assigning to each propositional variable $p \in P$ the subset $V(p) \subseteq S$ where the variable is true. For $s \in S$, the pair (M, s) is called a *pointed relational model*, and for $T \subseteq S$, the pair (M, T) is called a *multi-pointed relational model*.

Let two relational models $M = (S, R, V)$ and $M' = (S', R', V')$ be given. A non-empty relation $\mathfrak{R} \subseteq S \times S'$ is a *bisimulation* if for all $(s, s') \in \mathfrak{R}$ and $a \in A$:

atoms $s \in V(p)$ iff $s' \in V'(p)$ for all $p \in P$;

forth for all $t \in S$, if $R_a(s,t)$, then there is a $t' \in S'$ such that $R'_a(s', t')$ and $(t, t') \in \mathfrak{R}$;

back for all $t' \in S'$, if $R'_a(s', t')$, then there is a $t \in S$ such that $R_a(s, t)$ and $(t, t') \in \mathfrak{R}$.

We write $M \underline{\leftrightarrow} M'$ (*M and M' are bisimilar*) iff there is a bisimulation between M and M', and we write $(M, s) \underline{\leftrightarrow} (M', s')$ ((M, s) and (M', s') are bisimilar) iff there is a bisimulation between M and M' linking s and s'. Similarly, $(M, T) \underline{\leftrightarrow} (M', T')$ iff for all $s \in T$ there is $s' \in T'$ such that $(M, s) \underline{\leftrightarrow} (M', s')$, and vice versa. A *total bisimulation* is a bisimulation such that all states in the domain and codomain occur in a pair of the relation.

We will now define arrow update models. We can think of them as follows. If you remove the valuation from a relational model you get a *relational frame*. We now decorate each arrow (pair in the accessibility relation for an agent) with two formulas in some logical language $\mathcal{L}$: one for a condition that should hold in the source (state) of the arrow and the other that should hold in the target (state) of the arrow. The result is called an *arrow update model*.

Definition 1 (Arrow update model) Given a logical language $\mathcal{L}$, an *arrow update model* U is a pair (O, RR) where O is the set of *outcomes* and where RR is an *arrow relation* $RR : A \to \mathcal{P}((O \times \mathcal{L}) \times (O \times \mathcal{L}))$. $\dashv$

For each agent a, the arrow relation assigns to a pair (o, o') of outcomes in the domain a pair (φ, φ') of formulas. We write RR_a for $RR(a)$, and we write $(o, \varphi) \to_a (o', \varphi')$ for $((o, \varphi), (o', \varphi')) \in RR_a$, or even, if the outcomes are unambiguous, $\varphi \to_a \varphi'$. Formula φ is the *source condition* and formula φ' is the *target condition* of the a-labelled *arrow* from *source* o to *target* o'. A pointed arrow update model, or *arrow update*, is a pair (U, o) where $o \in O$. Similarly, we define the *multi-pointed arrow update model* (U, Q), where $Q \subseteq O$, known as well as arrow update. There is no confusion with the arrow updates of AUL [15], as those correspond to singleton pointed arrow update models.

Arrow update models are rather similar to the *action models* by Baltag *et al.* [5]. They are compared in Section 6.

2.2 Syntax

We proceed with the language and semantics of *arbitrary arrow update model logic* (AAUML).

Definition 2 (Syntax) The language of AAUML is inductively defined as

$$\mathcal{L} \ \ni \quad \varphi ::= p \mid \neg\varphi \mid (\varphi \land \varphi) \mid \Box_a \varphi \mid [U, o]\varphi \mid [\uparrow]\varphi$$

where $p \in P$, $a \in A$, where U is a *finite* arrow update model with source and target conditions of type φ, and $o \in \mathcal{D}(U)$. $\dashv$

The source and target conditions in U are thus lower in the inductive hierarchy: think of $[U, o]\varphi$ as an n-ary operator where not only the formula bound by $[U, o]$ is of type φ but also all the source and target conditions in U. We read $[U, o]\varphi$ as 'after executing arrow update (U, o), φ (holds), and $[\uparrow]\varphi$ as 'after an arbitrary arrow update, φ (holds)'. Other propositional connectives and dual diamond versions of modalities can be defined as usual by abbrevation: $\Diamond_a\varphi := \neg\Box_a\neg\varphi$, $\langle U, o\rangle\varphi := \neg[U, o]\neg\varphi$, and $\langle\uparrow\rangle\varphi := \neg[\uparrow]\neg\varphi$. Expression $\varphi[\psi/p]$ stands for uniform substitution of all occurrences of p in φ for ψ.

The propositional sublanguage is called $\mathcal{L}_{pl}$ and the language with additionally the modal construct $\Box_a\varphi$ is $\mathcal{L}_{ml}$ (the language of modal logic). In the language $\mathcal{L}$ of AAUML, (modalities for) multi-pointed arrow update models are defined by abbreviation as $[U, Q]\varphi := \bigwedge_{o \in Q}[U, o]\varphi$, and we informally allow them as primitive modalities in the logical language. However, in later sections we wish to compare the logic with only modalities for single-pointed arrow update models to the logic with modalities for multi-pointed arrow update models. Let us therefore call the former AAUML$_1$.

When doing synthesis, we will put formulas in disjunctive normal form. Since we are working in a modal logic, the definition has to be adapted from the usual one in propositional logic. We say that a formula $\varphi \in \mathcal{L}$ is in *disjunctive normal form* (DNF) if every subformula of φ is a disjunction of conjunctions of formulas $\psi_1, \dots, \psi_n$, where each ψ_i is an atom, or the negation of an atom, or has one of $\Box_a, \Diamond_a, [U, o], \langle U, o\rangle, [\uparrow]$ or $\langle\uparrow\rangle$ as main connective. Note that the requirement is that all subformulas are such disjunctions of conjunctions. In particular, this means that formulas have to be in DNF at every modal depth. So, for example, $p \lor \Box(q \lor (\Diamond p \land \neg q))$ is in DNF, while $p \lor \Box(q \lor \neg(\neg\Diamond p \lor q))$ is not. It is easy to see that every formula is equivalent to a formula in DNF.

2.3 Semantics

We continue with the semantics. The semantics is defined by induction on $\varphi \in \mathcal{L}$, and simultaneously with the execution of arrow update models.

Definition 3 (Semantics) Let a relational model $M = (S, R, V)$, a state $s \in S$, an arrow update model $U = (O, RR)$, an outcome $o \in O$, and a formula $\varphi \in \mathcal{L}$ be given.

$$\begin{array}{lll}
M, s \models p & \text{iff} & s \in V(p) \\
M, s \models \neg\varphi & \text{iff} & M, s \not\models \varphi \\
M, s \models \varphi \wedge \psi & \text{iff} & M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models \Box_a \varphi & \text{iff} & M, t \models \varphi \text{ for all } (s, t) \in R_a \\
M, s \models [U, o]\varphi & \text{iff} & M * U, (s, o) \models \varphi \text{ where } M * U \text{ is defined in } (\sharp) \\
M, s \models [\uparrow]\varphi & \text{iff} & M, s \models [U, o]\varphi \text{ for all } (U, o) \text{ satisfying } (\sharp\sharp)
\end{array}$$

$(\sharp)$: $M * U = (S', R', V')$ is defined as

$$\begin{array}{lll}
S' & = & S \times O \\
((s, o), (s', o')) \in R'_a & \text{iff} & (s, s') \in R_a, (o, \varphi) \to_a (o', \varphi'), M, s \models \varphi, \text{ and } M, s' \models \varphi' \\
V'(p) & = & V(p) \times O
\end{array}$$

$(\sharp\sharp)$: (U, o) is an arrow update model with all source and target conditions in $\mathcal{L}_{ml}$. $\dashv$

Validity in a model and validity are defined as usual.

The restriction of arrow formulas to $\mathcal{L}_{ml}$ in the semantics of $[\uparrow]\varphi$ is to avoid circularity of the semantics, as $[\uparrow]\varphi$ could otherwise itself be one of those arrow formulas. However, because we will prove that AAUML is as expressive as basic modal logic, we also have

$$M, s \models [\uparrow]\varphi \quad \text{iff} \quad M, s \models [U, o]\varphi \text{ for all } (U, o)$$

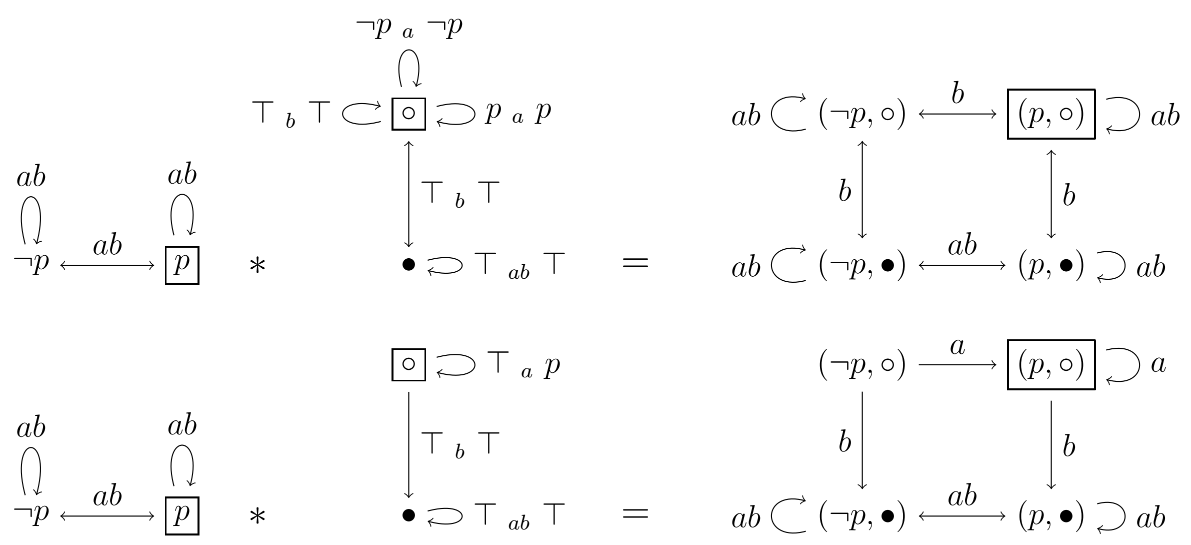
without any restriction on the source and target conditions of U. We will prove this property in Proposition 13, later.

2.4 Example

First consider the action of the introductory section of Anne reading a letter containing the truth about p while Bill remains uncertain whether she performs that action. The arrow update model producing the resulting information state is depicted below. In the figure, an arrow $\to$ labelled with $\varphi\ {}_i\ \varphi'$ and linking outcomes o, o' stands for the arrow $\varphi \to_i \varphi'$ between these outcomes, i.e., $((o, \varphi), (o', \varphi')) \in RR_i$; $\varphi\ {}_{ij}\ \varphi'$ stands for $\varphi \to_i \varphi'$ and $\varphi \to_j \varphi'$.

In the resulting model Bill considers it possible that Anne knows p, that she knows $\neg p$, and that she still is uncertain about p: $\Diamond_b(\Box_a p \wedge \Box_a \neg p \wedge \neg(\Box_a p \vee \Box_a \neg p))$.

Next, consider the action of Anne privately learning that p while Bill remains unaware of her doing so. The arrow update model achieving that and the resulting relational model are depicted further below. In the resulting model it is true that, for example, Bill incorrectly believes that Anne is uncertain about p: $\Box_a p \wedge \Box_b \neg(\Box_a p \vee \Box_a \neg p)$.



3 Arrow update synthesis

The goal of synthesis is to find, given a goal formula φ, an arrow update model (U, o) that makes φ true. There are at least three ways in which we could interpret this goal, however.

Definition 4 (Synthesis)

- The *local synthesis problem* takes as input a pointed model (M, s) and a goal formula φ. The output is an arrow update model (U, o) such that $M, s \models \langle U, o\rangle\varphi$, or "NO" if no such arrow update model exists.
- The *valid synthesis problem* takes as input a goal formula φ. The output is an arrow update model (U, o) such that $\models \langle U, o\rangle\varphi$, or "NO" if no such arrow update model exists.
- The *global synthesis problem* takes as input a goal formula φ. The output is an arrow update model (U, o) such that for every pointed model (M, s), if there is some (U', o') such that $M, s \models \langle U', o'\rangle\varphi$, then $M, s \models \langle U, o\rangle\varphi$. $\dashv$

We recall from the introduction that we take the third approach: when we say synthesis we mean global synthesis. An alternative, equivalent characterization of the global synthesis problem is that, for given φ, we want to find (U, o) such that $M, s \models \langle\Uparrow\rangle\varphi \leftrightarrow \langle U, o\rangle\varphi$. We further recall that synthesis is impossible for PAL and for AUL, but possible for AML [12]. We now show that synthesis for AUML is also possible. Because our version of synthesis is global, it cannot depend on any specific model. So our synthesis process is purely syntactic.

In our synthesis, we make use of so-called *reduction axioms*. These reduction axioms are a set of validities that, when taken together, show that AAUML has the same expressive power as modal logic.

3.1 Reduction axioms for arrow update models

We start by considering the reduction axioms for the $[U, o]$ operator.

Lemma 5 ([16]) Let (U, o) be an arrow update model, $p \in P$, $a \in A$ and $\varphi, \psi \in \mathcal{L}$. Then the following validities hold:

$$\begin{aligned}
&\models [U,o]p \leftrightarrow p\\
&\models [U,o]\neg\varphi \leftrightarrow \neg[U,o]\varphi\\
&\models [U,o](\varphi \wedge \psi) \leftrightarrow ([U,o]\varphi \wedge [U,o]\psi)\\
&\models [U,o]\Box_a\varphi \leftrightarrow \bigwedge_{(o,\psi)\to_a(o',\psi')} (\psi \to \Box_a(\psi' \to [U,o']\varphi))
\end{aligned}$$

Proof The first three validities follow immediately from the semantics of $[U, o]$. The fourth validity also follows from the semantics, in the following way

$$\begin{aligned}
M, w \models [U,o]\Box_a\varphi \Leftrightarrow & M * U, (w,o) \models \Box_a\varphi\\
\Leftrightarrow & \forall (w,o)R(w',o) : M * U, (w',o') \models \varphi\\
\Leftrightarrow & \forall (o,\psi) \to_a (o,\psi')\forall wRw' : \text{ if } M, w \models \psi, M, w' \models \psi'\\
& \qquad \text{then } M * U, (w',o') \models \varphi\\
\Leftrightarrow & \forall (o,\psi) \to_a (o,\psi') : \text{ if } M, w \models \psi\\
& \qquad \text{then } M, w \models \Box_a(\psi' \to [U,o']\varphi)\\
\Leftrightarrow & M, w \models \bigwedge_{(o,\psi)\to_a(o',\psi')} (\psi \to \Box_a(\psi' \to [U,o']\varphi))
\end{aligned}$$

□

Note that, in particular, $\models [U,o]\neg\varphi \leftrightarrow \neg[U,o]\varphi$ implies that $[U,o]$ is self-dual: we have $\models [U,o]\varphi \leftrightarrow \langle U,o\rangle\varphi$. This, of course, does not extend to the arbitrary arrow update operator: $\not\models [\uparrow]\varphi \leftrightarrow \langle\uparrow\rangle\varphi$.

3.2 Reduction axioms for the arrow update model quantifier

We can also write similar reduction axioms for $[\uparrow]$. In practice, however, it turns out to be slightly more convenient to write them for the dual operator $\langle\uparrow\rangle$.

Lemma 6 For every $\varphi \in \mathcal{L}$ and every $a \in A$, we have

$$\models \langle\uparrow\rangle\Diamond_a\varphi \leftrightarrow \Diamond_a\langle\uparrow\rangle\varphi$$

Proof Let M, w be any pointed model, and suppose that $M, w \models \langle\uparrow\rangle\Diamond_a\varphi$. Then there is some (U, o) such that $M * U, (w, o) \models \Diamond_a\varphi$. So (w, o) has an a-successor (w', o') such that $M * U, (w', o') \models \varphi$.

This implies that $M, w' \models \langle U, o'\rangle\varphi$ and therefore $M, w' \models \langle\uparrow\rangle\varphi$. Since w' is an a-successor of w, we obtain $M, w \models \Diamond_a\langle\uparrow\rangle\varphi$.

Now, suppose that $M, w \models \Diamond_a\langle\uparrow\rangle\varphi$. Then there is an a-successor w' of w such that $M, w' \models \langle\uparrow\rangle\varphi$. As witness for this $\langle\uparrow\rangle$ statement there must be some U', o' such that $M, w' \models \langle U', o'\rangle\varphi$.

Let (U, o) be the arrow update model obtained by adding one extra world o to U', and a transitions $(o, \top) \to_a (o', \top)$. Note that $(M * U', (w', o'))$ is bisimilar to $(M * U, (w', o'))$, and therefore $M * U, (w', o') \models \varphi$. Finally, note that (w', o') is an a-successor of (w, o), so we have $M * U, (w, o) \models \Diamond_a\varphi$ and therefore $M, w \models \langle\uparrow\rangle\Diamond_a\varphi$. □

Note that the proof is constructive. That is, if we find (U', o') such that $M, w \models \Diamond_a\langle U', o'\rangle\varphi$ then not only do we know that $M, w \models \langle\uparrow\rangle\Diamond_a\varphi$, we can also find a specific (U, o) such that $M, w \models \langle U, o\rangle\Diamond_a\varphi$.

Next, we consider a slightly stronger lemma.

Lemma 7 For every $\varphi_1, \cdots, \varphi_n \in \mathcal{L}$ and every $a \in A$ we have

$$\models \langle\uparrow\rangle \bigwedge_{1\leq i\leq n} \Diamond_a\varphi_i \leftrightarrow \bigwedge_{1\leq i\leq n} \Diamond_a\langle\uparrow\rangle\varphi_i$$

Proof The left to right direction is obvious, so we show only the right to left direction. So suppose that $M, w \models \bigwedge \Diamond_a\langle\uparrow\rangle\varphi_i$. Then there are a-successors $w_1, \cdots, w_n$ of w and pointed arrow update models $(U_1, o_1), \cdots, (U_n, o_n)$ such that $M, w_i \models \langle U_i, o_i\rangle\varphi_i$ for all i.

Now, let (U, o) be the arrow update model obtained by taking the union of all U_i and adding one extra outcome o, and adding arrows $(o, \top) \to_a (o_i, \top)$ for every o_i.

For every i, $(M * U_i, (w_i, o_i))$ is bisimilar to $(M * U, (w_i, o_i))$, so we have $M * U, (w_i, o_i) \models \varphi_i$. Finally, (w_i, o_i) is an a-successor of (w, o) for every i. As such, we have $M, w \models \langle U, o\rangle \bigwedge \Diamond_a\varphi_i$ and therefore $M, w \models \langle\uparrow\rangle \bigwedge \Diamond_a\varphi_i$. □

Again, the proof is constructive, so given (U_i, o_i) for all i, we can find the model (U, o). Note also that the φ_i need not be consistent with each other

Some reflection may be in order as to why Lemma 7 holds. Suppose that $M, w \models \bigwedge \Diamond_a\langle\uparrow\rangle\varphi_i$. So for every i, there are some world w_i that a considers possible as well as some event U_i and outcome o_i such that of (U_i, o_i) were to happen in w_i, then φ_i would become true.

Now let us look at the arrow update model (U, o) that we constructed. Effectively, this arrow update model represents us telling a that "we are performing one of the actions U_i, o_i, but we are not telling you which one." Now, for every i agent a considers it possible that w_i is the actual world, and that (U_i, o_i) is the event that happened. As such, after we execute our event we are in a situation where every φ_i is held possible by a.

So far, we have only considered diamonds. Now, let us add a box modality.

Lemma 8 For every $\varphi_1, \cdots, \varphi_n, \psi \in \mathcal{L}$ and every $a \in A$, we have

$$\models \langle\uparrow\rangle(\bigwedge_{1\leq i\leq n} \Diamond_a\varphi_i \land \Box_a\psi) \leftrightarrow \bigwedge_{1\leq i\leq n} \Diamond_a\langle\uparrow\rangle(\varphi_i \land \psi)$$

Proof The left to right direction is obvious, so we only show right to left. So suppose that $M, w \models \bigwedge \Diamond_a\langle\uparrow\rangle(\varphi_i \land \psi)$. Then for every $1 \leq i \leq n$, there are U_i, o_i and an a-successor w_i of w such that $M, w_i \models \langle U_,o_i\rangle(\varphi_i \land \psi)$.

Now, let (U, o) be the model obtained by taking the union of all U_i, and adding a single outcome o with arrows $(o, \top) \rightarrow_a (o_i, \langle U_i, o_i\rangle\psi)$ for every i.

Now, consider $(M * U, (w, o))$. By assumption, $M, w_i \models \langle U_i, o_i\rangle\psi$, so (w_i, o_i) is an a-successor of (w, o) in $(M * U)$. Furthermore, $M * U, (w_i, o_i) \models \varphi_i$. It follows that $M * U, (w, o) \models \Diamond_a\varphi_i$.

Additionally, note that for every outgoing a-arrow in (U, o) the postcondition of the arrow is $\langle U_i, o_i\rangle\psi$, and that $(M * U, (w_i, o_i))$ is bisimilar to $(M * U_i, (w_i, o_i))$. For every (w_i, o_i) that is an a-successor of (w, o), we therefore have $M * U, (w_i, o_i) \models \psi$ for all i. It follows that $M * U, (w, o) \models \Box_a\psi$.

Taken together, the above shows that $M, w \models \langle U, o\rangle(\bigwedge \Diamond_a\varphi_i \land \Box_a\psi)$, and therefore $M, w \models \langle\uparrow\rangle(\bigwedge \Diamond_a\varphi_1 \land \Box_a\psi)$ as was to be shown. $\Box$

Once again, the proof is constructive. Note that on the right-hand side we have eliminated the $\Box_a$ connective. This is a consequence of the fact that as the designer of the arrow update model U, we have the freedom to inform a that certain worlds, which she might previously have considered possible, are not in fact the actual world. This results in the removal of the a-arrows to these worlds. So if we want to make $\Box_a\psi$ true after the application of U, then we can simply have a eliminate all successors where ψ would otherwise become false. In the construction used in the lemma, we do this using the postcondition $\langle U_i, o_i\rangle\psi$.

In the preceding three lemmas, we only considered $\Diamond_a$ and $\Box_a$ operators for one single agent a. However, when constructing (U, o) we can place arrows for different agents independently, so the same construction works for multiple agents at the same time. This yields the following lemma.

Lemma 9 For every $a \in A$, let $\Phi_a \subseteq \mathcal{L}$ be a finite set of formulas, and let $\psi_a \in \mathcal{L}$ be a formula. Then

$$\models \langle\uparrow\rangle \bigwedge_{a\in A}(\bigwedge_{\varphi_a\in\Phi_a} \Diamond_a\varphi_a \land \Box_a\psi_a) \leftrightarrow \bigwedge_{a\in A} \bigwedge_{\varphi_a\in\Phi_a} \Diamond_a\langle\uparrow\rangle(\varphi_a \land \psi_a)$$

Lemma 9 is the most important reduction axiom for AAUML. However, not every formula is of a form such that the lemma can be applied. We therefore need two validities that allow us to put formulas in the correct form.

Lemma 10 For very $\varphi_1, \varphi_2 \in \mathcal{L}$ and every $\varphi_0 \in \mathcal{L}_{pl}$, we have

$$\models \langle\uparrow\rangle(\varphi_1 \vee \varphi_2) \leftrightarrow (\langle\uparrow\rangle\varphi_1 \vee \langle\uparrow\rangle\varphi_2)$$

and

$$\models \langle\uparrow\rangle(\varphi_0 \wedge \varphi_1) \leftrightarrow (\varphi_0 \wedge \langle\uparrow\rangle\varphi_1).$$

The proof of this lemma is rather trivial, so we omit it. It is, however, important and non-trivial to note that the disjunction case can be made constructive.

Suppose that we have already synthesized (U_1, o_1) and (U_2, o_2) such that $\models \langle\uparrow\rangle\varphi_1 \leftrightarrow \langle U_1, o_1\rangle\varphi_1$ and $\models \langle\uparrow\rangle\varphi_1 \leftrightarrow \langle U_2, o_2\rangle\varphi_2$. So we have two arrow update models that make φ_1 and φ_2 true whenever possible. This does not, however, immediately give us a *single*-pointed arrow update model (U, o) that guarantees $\varphi = \varphi_1 \vee \varphi_2$ whenever possible.[1] In order to find this (U, o), we have to combine (U_1, o_1) and (U_2, o_2). We do this in the following way.

First, we take the set of outcomes of U to be the union of the sets of outcomes of U_1 and U_2, plus one extra outcome o. Then, we add arrows as follows to U.

> For every $(o_1, \psi) \to_a (o', \psi')$ of U_1, add an arrow $(o, \psi \wedge \langle\uparrow\rangle\varphi_1) \to_a (o', \psi')$. Then, for every $(o_2, \psi) \to_a (o', \psi)$ of U_2, add an arrow $(o, \psi \wedge \neg\langle\uparrow\rangle\varphi_1) \to_a (o', \psi)$.

When executed in a $\langle\uparrow\rangle\varphi_1$ world, this arrow update model (U, o) will act as (U_1, o_1), since we added all arrows from o_1 with an extra $\langle\uparrow\rangle\varphi_1$ precondition. When executed in any $\neg\langle\uparrow\rangle\varphi_1$ world, (U, o) acts as (U_2, o_2). As long as either $\langle\uparrow\rangle\varphi_1$ or $\langle\uparrow\rangle\varphi_2$ holds, we therefore have $\langle U, o\rangle(\varphi_1 \vee \varphi_2)$.

More formally, we have the following lemma.

Lemma 11 Let $\varphi_1, \varphi_2 \in \mathcal{L}$, and for $i = 1, 2$ let (U_i, o_i) be such that $\models \langle\uparrow\rangle\varphi_i \leftrightarrow \langle U_i, o_i\rangle\varphi_i$, where $U_i = (O_i, RR_i)$. Furthermore, let $U = (O, RR)$ be given as follows:

- $O = \{o\} \cup O_1 \cup O_2$,
- RR contains exactly the following arrows:
    1. $(o', \psi') \to_a (o'', \psi'') \in RR_i$, for $i = 1, 2$,
    2. $(o, \psi \wedge \langle\uparrow\rangle\varphi_1) \to_a (o', \psi')$ where $(o_1, \psi) \to_a (o', \psi') \in RR_1$,
    3. $(o, \psi \wedge \neg\langle\uparrow\rangle\varphi_1) \to_a (o', \psi')$ where $(o_2, \psi) \to_a (o', \psi') \in RR_2$.

Then $\models \langle\uparrow\rangle(\varphi_1 \vee \varphi_2) \leftrightarrow \langle U, o\rangle(\varphi_1 \vee \varphi_2)$. $\dashv$

Proof The right to left direction is obvious. We show the left to right direction. Suppose therefore that $M, w \models \langle\uparrow\rangle(\varphi_1 \vee \varphi_2)$. Then, by Lemma 10, we have $M, w \models \langle\uparrow\rangle\varphi_1 \vee \langle\uparrow\rangle\varphi_2$. We continue by a case distinction.

[1] An alternative technique to synthesize an arrow update model for the disjunction is to take the double-pointed direct sum of (U_1, o_1) and (U_2, o_2); its points are $\{o_1, o_2\}$. This method is followed in [12].

First, suppose that $M, w \models \langle\Uparrow\rangle\varphi_1$. Then none of the $(o, \psi \wedge \neg\langle\Uparrow\rangle\varphi_1) \to_a (o', \psi')$ arrows are applicable in w. The $(o, \psi \wedge \langle\Uparrow\rangle\varphi_1) \to_a (o', \psi')$ arrows, on the other hand, are applicable if and only if $(o_1, \psi) \to_a (o', \psi')$ applies. It follows that $M * U, (w, o)$ is bisimilar to $M * U_1, (w, o_1)$, so $M, w \models \langle U, o\rangle\varphi_1$.

Suppose then that $M, w \not\models \langle\Uparrow\rangle\varphi_1$. Then we must have $M, w \models \langle\Uparrow\rangle\varphi_2$. In this case, the $(o, \psi \wedge \langle\Uparrow\rangle\varphi_1) \to_a (o', \psi')$ are inapplicable while the $(o, \psi \wedge \neg\langle\Uparrow\rangle\varphi_1) \to_a (o', \psi')$ arrows apply if and only if $(o_2, \psi) \to_a (o', \psi')$ does. So $M * U, (w, o)$ is bisimilar to $M * U_2, (w, o_2)$, and therefore $M, w \models \langle U, o\rangle\varphi_2$. $\square$

3.3 Reduction

In Section 3.1, we showed that $[U, o]$ commutes with $\neg$, distributes over $\wedge$ and, in a somewhat complicated way, commutes with $\Box_a$. Finally, whenever we encounter a formula of the form $[U, o]p$, we can simply remove the part $[U, o]$. So if φ is a formula of modal logic (and therefore doesn't contain $[U', o]$ or $[\Uparrow]$ operators), then we can transform $[U, o]\varphi$ into an equivalent formula φ' of modal logic.

Additionally, in Section 3.2, we showed that $\langle\Uparrow\rangle$ commutes, in a very complicated way, with Boolean combinations of modal formulas. Also, like $[U, o]$, a $\langle\Uparrow\rangle$ operator disappears once it encounters a propositional atom. So if φ is a formula of modal logic, then we can transform $\langle\Uparrow\rangle\varphi$ into an equivalent formula φ' of modal logic.

By successively eliminating the innermost $[U, o]$ or $\langle\Uparrow\rangle$ operators, we can transform any formula of AAUML into an equivalent formula of modal logic. In other words, we have the following theorem.

Theorem 12 For every $\varphi \in \mathcal{L}$ there is a formula $\varphi' \in \mathcal{L}_{ml}$ such that $\models \varphi \leftrightarrow \varphi'$. $\dashv$

For the next section, it is important to keep in mind that the reduction axioms not only guarantee the existence of φ', but also enable us to find it.

Theorem 12 also allows us to prove a claim that we made in Section 2.3. There, we defined $M, s \models [\Uparrow]\varphi$ by

> $M, s \models [\Uparrow]\varphi$ iff ($M, s \models [U, o]\varphi$ for every arrow update model (U, o) that has source and target conditions only in $\mathcal{L}_{ml}$).

Now that we have shown that every formula of $\mathcal{L}$ is equivalent to a formula of $\mathcal{L}_{ml}$, it follows immediately that the requirement of the source and target conditions being in $\mathcal{L}_{ml}$ is unnecessary.

Proposition 13 For every $\varphi \in \mathcal{L}$ and every pointed model (M, s), we have

$$M, s \models [\Uparrow]\varphi \quad \text{iff} \quad M, s \models [U, o]\varphi \text{ for every arrow update model } (U, o). \quad \dashv$$

Procedure $\mathit{Synth}(\varphi)$

Input: $\varphi \in \mathcal{L}$.
Output: (U_φ, o_φ) such that $\models \langle\uparrow\rangle\varphi \leftrightarrow \langle U_\varphi, o_\varphi\rangle\varphi$.

```
1. If $\varphi \notin \mathcal{L}_{ml}$, then use the reduction axioms to find a formula $\varphi_{modal} \in \mathcal{L}_{ml}$
   such that $\models \varphi \leftrightarrow \varphi_{modal}$, and return $\mathit{Synth}(\varphi_{modal})$. Otherwise, continue.
2. If $\varphi$ is not in disjunction normal form, compute the DNF $\varphi_{DNF}$ of $\varphi$ and
   return $\mathit{Synth}(\varphi_{DNF})$. Otherwise, continue.
3. If $\varphi = \varphi_1 \vee \varphi_2$, then compute $\mathit{Synth}(\varphi_1)$ and $\mathit{Synth}(\varphi_2)$, and combine
   the two arrow update models as in Lemma 11. Return the combined
   arrow update model.
4. If $\varphi$ is not a disjunction, then since it is in DNF it must be a conjunction,
   where each conjunct is (i) a literal, (ii) of the form $\Diamond_a\psi$, or (iii) of the form $\Box_a\chi$.
   Assume w.l.o.g. that for every $a$ there is exactly one conjunct $\Box_a\chi_a$.
   For every $\Diamond_a\psi$, compute $\mathit{Synth}(\psi \wedge \chi_a)$. Then use Lemma 8 to combine
   the arrow update models, and return the result. If there are no $\Diamond_a$ operators for any
   agent $a$, return the trivial arrow update model with one outcome and no arrows.
```

Table 1: Synthesis procedure

3.4 Synthesis

Recall that our goal, when performing synthesis, is to find, for given $\varphi \in \mathcal{L}$, an arrow update model (U_φ, o_φ) such that $\models \langle\uparrow\rangle\varphi \leftrightarrow \langle U_\varphi, o_\varphi\rangle\varphi$. Using Theorem 12, we can transform φ into an equivalent formula of modal logic. Then, using the procedure outlined in Section 3.2, we can find (U_φ, o_φ). The procedure is found in detail in Table 1.

3.5 Example

In order to gain better understanding of $\mathit{Synth}(\varphi)$, let us consider an example. Suppose $\varphi = \Diamond_a\Box_b p \wedge \Diamond_b(\Diamond_a q \wedge \Diamond_a r) \wedge \Box_b p)$. We want to perform synthesis for this φ.

Goal: find $\mathit{Synth}(\Diamond_a\Box_b p \wedge \Diamond_b(\Diamond_a q \vee \Diamond_a r) \wedge \Box_b p)$**.**

Because $\varphi \in \mathcal{L}_{ml}$, φ is in DNF and φ is not a disjunction, we continue past steps 1, 2 and 3. In step 4, it is assumed that for every agent there is exactly one $\Box$ conjunct. This means we need to add a trivial conjunct $\Box_a\top$.

Of the conjuncts of φ, two have $\Diamond$ as primary operator. So we need to perform synthesis for two more formulas; because of $\Diamond_a\Box_b p$ and $\Box_a\top$ we need to find $\mathit{Synth}(\Box_b p \wedge \top)$, and because of $\Diamond_b(\Diamond_a q \vee \Diamond_a r)$ and $\Box_b p$ we need to find $\mathit{Synth}((\Diamond_a q \vee \Diamond_a r) \wedge p)$.

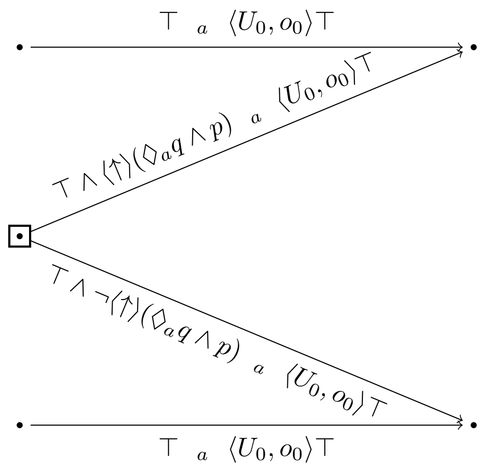
Subgoal 1: find $\mathit{Synth}(\Box_b p \wedge \top)$**.**
Since we are doing synthesis for a conjunction, we continue in steps 1, 2 and 3. Because there are no $\Diamond$ operators in $\Box_b p \wedge \top$, we return the trivial arrow update

model in step 4.

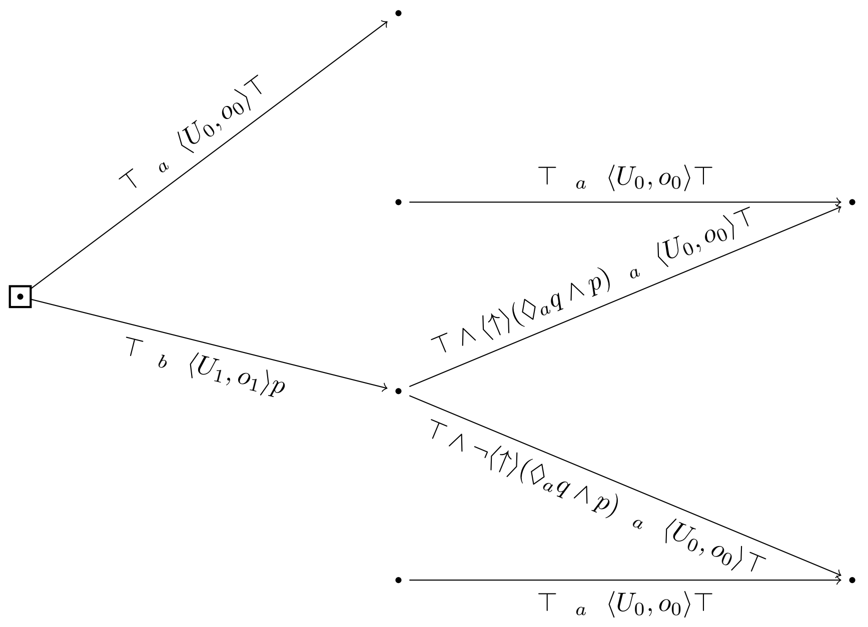
Subgoal 2: find *Synth*$((\Diamond_a q \vee \Diamond_a r) \wedge p)$**.** In step 2, we need to put the formula in DNF. We therefore continue with $(\Diamond_a q \wedge p) \vee (\Diamond_a r \wedge p)$. In step 3 we are then instructed to perform synthesis for the two disjuncts.

> **Sub-subgoal 2.1: find** *Synth*$(\Diamond_a q \wedge p)$**.**
> We continue up to step 4. There, we first add a a trivial conjunct $\Box_a \top$. Then, we are instructed to find *Synth*$(q \wedge \top)$.
>
> > **Sub-sub-subgoal 2.1.1: find** *Synth*$(q \wedge \top)$**.**
> > We proceed to step 4. There, since there are no $\Diamond$ operators in $q \wedge \top$, we return the trivial arrow update model (U_0, o_0).
>
> Now, in order to find *Synth*$(\Diamond_a q \wedge p)$, we take the trivial arrow update model found in sub-sub-subgoal 2.1.1, and add one extra outcome. Then, we connect this extra outcome to the trivial model by a $\top \rightarrow_a \langle U_0, o_0\rangle\top$ arrow. The source condition of this arrow is $\top$ because step 4 uses the construction from Lemma 8, and that construction always gives precondition $\top$. The arrow is for agent a, because we started with a $\Diamond_a$ operator. Finally, the postcondition is $\langle U_0, o_0\rangle\top$ because the arrow update model that the arrow points to is (U_0, o_0) and the $\Box_a$ conjunct was $\Box_a \top$.
>
> In other words, we obtain the following arrow update model, where the framed state indicates the designated outcome:
>
> $$\begin{tikzcd}\boxed{\cdot} \arrow[r, "\top \quad a \quad \langle U_0, o_0\rangle\top"] & \bullet\end{tikzcd}$$
>
> **Sub-subgoal 2.2: find** *Synth*$(\Diamond_a r \wedge p)$**.**
> Replacing the q of $\Diamond_a q \wedge p$ for an r does not change the arrow update model that we end up with. So in this sub-subgoal we find the same model as in sub-subgoal 2.1.

Now, in order to find *Synth*$((\Diamond_a q \wedge p) \vee (\Diamond_a r \wedge p))$, we need to combine the models found in sub-subgoals 2.1 and 2.2. Since we are working with a disjunction, we combine them as described in Lemma 11. That, is, we take copies of the two (identical) models and add one extra outcome. Then, we add two more arrows: every world that is reachable from the origin world of the model from sub-subgoal 2.1 by $\psi_1 \rightarrow_a \psi_2$, becomes reachable from the extra world by a $\psi_1 \wedge \langle\uparrow\rangle(\Diamond_a q \wedge p) \rightarrow_a \psi_2$ arrow. Likewise, every world reachable by $\psi_1 \rightarrow_a \psi_2$ from the origin of the model from sub-subgoal 2.2 becomes reachable from the extra world by $\psi_1 \wedge \neg\langle\uparrow\rangle(\Diamond_a q \wedge p) \rightarrow_a \psi_2$.


$\top \;_a\; \langle U_0, o_0\rangle\top$
$\top \wedge \langle\uparrow\rangle(\Diamond_a q \wedge p) \;_a\; \langle U_0, o_0\rangle\top$
$\top \wedge \neg\langle\uparrow\rangle(\Diamond_a q \wedge p) \;_a\; \langle U_0, o_0\rangle\top$
$\top \;_a\; \langle U_0, o_0\rangle\top$

Now, all that is left to do is to combine the arrow update models found in subgoals 1 and 2. The model we obtain is


$\top \;_a\; \langle U_0, o_0\rangle\top$
$\top \;_a\; \langle U_0, o_0\rangle\top$
$\top \wedge \langle\uparrow\rangle(\Diamond_a q \wedge p) \;_a\; \langle U_0, o_0\rangle\top$
$\top \;_b\; \langle U_1, o_1\rangle p$
$\top \wedge \neg\langle\uparrow\rangle(\Diamond_a q \wedge p) \;_a\; \langle U_0, o_0\rangle\top$
$\top \;_a\; \langle U_0, o_0\rangle\top$

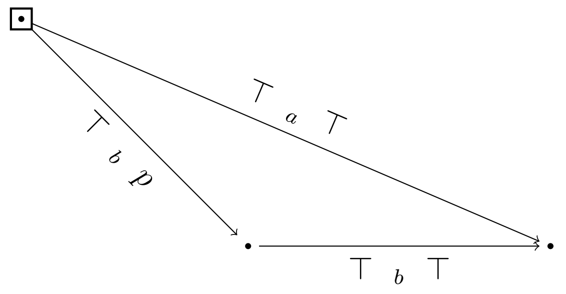
where (U_1, o_1) is the model that we found in subgoal 2. The root of the model is the leftmost outcome. Note that the depth (i.e., the maximum path length) of this arrow update model is 2, just like the depth (i.e., the maximum number of nested $\Box$ or $\Diamond$ operators) of φ. In general, the depth of the synthesized arrow update model is bounded by that of the formula for which synthesis is performed.

Also, note that the arrow update model that we obtained can quite easily be modified to obtain a smaller model that is still sufficient. In particular:

- the two outcomes that are not reachable from the root can be eliminated,
- the formulas $\langle U_0, o_0\rangle\top$, $\langle U_1, o_1\rangle p$, $\top \wedge \langle\uparrow\rangle(\Diamond_a q \wedge p)$ and $\top \wedge \neg\langle\uparrow\rangle(\Diamond_a q \wedge p)$ can be replaced by the equivalent formulas $\top$, p, $\Diamond_a q \wedge p$ and $\neg(\Diamond_a q \wedge p)$, respectively,

- the three leaf outcomes can be merged into one,
- and $\Diamond_a q \wedge p \to_a \top$ and $\neg(\Diamond_a q \wedge p) \to_a \top$ can be merged into one $\top \to_a \top$ arrow.

With these optimizations, we get the more aesthetically pleasing arrow update model



4 Axiomatization

Using the reduction axioms introduced before, we can find an axiomatization for AAUML. Let **AAUML** be the axiomatization shown in Table 2. In this section we show that the axiomatization **AAUML** is sound and complete, and we give some derivable (well-known) axiom schemata.

Prop	All tautologies of propositional logic	
K	$\Box_a(\varphi \to \psi) \to (\Box_a\varphi \to \Box_a\psi)$	
U1	$[U, o]p \leftrightarrow p$	
U2	$[U, o]\neg\varphi \leftrightarrow \neg[U, o]\varphi$	
U3	$[U, o](\varphi \wedge \psi) \leftrightarrow ([U, o]\varphi \wedge [U, o]\psi)$	
U4	$[U, o]\Box_a\varphi \leftrightarrow \bigwedge_{(o,\psi)\to_a(o',\psi')}(\psi \to \Box_a(\psi' \to [U, o']\varphi))$	
A1	$\langle\uparrow\rangle\varphi_0 \leftrightarrow \varphi_0$	where $\varphi_0 \in \mathcal{L}_{pl}$
A2	$\langle\uparrow\rangle(\varphi \vee \psi) \leftrightarrow (\langle\uparrow\rangle\varphi \vee \langle\uparrow\rangle\psi)$	
A3	$\langle\uparrow\rangle(\varphi_0 \wedge \varphi) \leftrightarrow (\varphi_0 \wedge \langle\uparrow\rangle\varphi)$	where $\varphi_0 \in \mathcal{L}_{pl}$
A4	$\langle\uparrow\rangle \bigwedge_{a\in A}(\bigwedge_{\varphi_a\in\Phi_a} \Diamond_a\varphi_a \wedge \Box_a\psi_a) \leftrightarrow \bigwedge_{a\in B} \bigwedge_{\varphi_a\in\Phi_a} \Diamond_a\langle\uparrow\rangle(\varphi_a \wedge \psi_a)$	
MP	from $\varphi \to \psi$ and φ infer ψ	
NecK	from φ infer $\Box_a\varphi$	
RE	from $\chi \leftrightarrow \psi$ infer $\varphi[\chi/p] \leftrightarrow \varphi[\psi/p]$	

Table 2: The axiomatization **AAUML** of the logic AAUML

Lemma 14 Axiomatization **AAUML** is sound for the logic AAUML. $\dashv$

Proof **Prop**, **K**, **MP**, **NecK**, **RE** are known from modal logic, **U1**—**U4** were demonstrated in Section 3.1 and originate in [16], **A1**—**A4** were shown to be valid in Section 3.2. $\Box$

It is important to note that **U1**—**U4** and **A1**—**A4** are so-called *reduction axioms* for the operators $[U, o]$ and $\langle\uparrow\rangle$, respectively, as mentioned in the previous section. This means that they are equivalences, where the formula inside the scope of the $[U, o]$ or $\langle\uparrow\rangle$ operator on the left-hand side is more complex than the formulas inside the scope of that operator on the right-hand side.

The derivation rule **RE** is important as our reductions are inside-out, not outside-in. Without it, for example, the validity $[U, o][U, o](p \vee \neg p)$ would not be derivable.

The axioms **A1**—**A4** could just as well have been formulated with the $[\uparrow]$ dual of the modality $\langle\uparrow\rangle$, e.g., **A2′** $[\uparrow](\varphi \wedge \psi) \leftrightarrow ([\uparrow]\varphi \wedge [\uparrow]\psi)$. We prefer the $\langle\uparrow\rangle$ versions as they match our usage of these axioms in synthesis. Further note that there is no reduction of shape $\langle\uparrow\rangle\neg\varphi \leftrightarrow \dots$. We assume that subformulas bound by $\langle\uparrow\rangle$ are first massaged into disjunctive normal form before a further reduction can take place (and again, for this, the derivation rule **RE** is essential).

Lemma 15 Axiomatization **AAUML** is complete for the logic AAUML. $\dashv$

Proof Let $\varphi \in \mathcal{L}$ be valid. Using an induction argument, we can eliminate all $[U, o]$ and $\langle\uparrow\rangle$ operators from it: φ must be provably equivalent to a formula $\varphi' \in \mathcal{L}_{ml}$. As φ' must also be valid (Theorem 12), φ' is provable in modal logic. From the provable equivalence between φ and φ' and the derivation of φ' we construct a derivation of φ in **AAUML**. $\Box$

We have now shown that:

Theorem 16 Axiomatization **AAUML** is sound and complete for the logic AAUML. $\dashv$

In the proof system **AAUML**, we do not have necessitation for the $[U, o]$ and $[\uparrow]$ operators. Such necessitation rules are derivable, however.

Proposition 17 The following two rules are derivable in **AAUML**.

- **NecU**: from φ infer $[U, o]\varphi$;
- **NecA**: from φ infer $[\uparrow]\varphi$. $\dashv$

Proof First, note that the rule

$$\textbf{U1}' \quad [U, o]\varphi_0 \leftrightarrow \varphi_0 \qquad\qquad \text{where } \varphi_0 \in \mathcal{L}_{pl}$$

is derivable, using **Prop**, **U1**–**U3** and **MP**. It is also convenient to use a variant of **MP** directly on bi-implications, instead of first converting the bi-implication to a single implication.

$$\textbf{MP}' \quad \text{from } \varphi \leftrightarrow \psi \text{ and } \varphi \text{ infer } \psi\text{, and from } \varphi \leftrightarrow \psi \text{ and } \psi \text{ infer } \varphi.$$

This **MP′** is, of course, also easily derived. Using **U1′** and **MP′**, we can derive **NecU** in a reasonable number of steps:

1.	φ	premise
2.	$\varphi \to (\varphi \leftrightarrow \top)$	**Prop**
3.	$\varphi \leftrightarrow \top$	**MP**(1,2)
4.	$[U, o]\top \leftrightarrow \top$	**U1′**
5.	$\top$	**Prop**
6.	$[U, o]\top$	**MP′**(5,4)
7.	$[U, o]\top \leftrightarrow [U, o]\varphi$	**RE**(3)
8.	$[U, o]\varphi$	**MP′**(6,7)

A derivation of **NecA** can be found in a similar way. Here, too, it is convenient to first derive an auxiliary axiom.

$$\mathbf{A1'} \quad [\uparrow]\varphi_0 \leftrightarrow \varphi_0 \qquad\qquad \text{where } \varphi_0 \in \mathcal{L}_{pl}$$

This $[\uparrow]$-version of **A1** is of course derivable. We can then derive **NecA** analogously to how we derived **NecU**, with the application of **U1′** replaced by **A1′**. □

5 Update expressivity

5.1 Expressivity

Recall that we are considering the basic modal logic ML and the update logics PAL, APAL, AML, AAML, AUL, AAUL, AUML, AAUML, and RML, as shown in Figure 1 on page 6. One natural thing to do with such related logics is to compare their power. The most straightforward way to make such a comparison is to compare their expressivity.

Formally, a language $\mathcal{L}_1$ is *at least as expressive* as a language $\mathcal{L}_2$ if for every formula of $\mathcal{L}_2$ there is an equivalent formula of $\mathcal{L}_1$. Having *equal expressivity* or *higher expressivity* (by which we always mean strictly higher expressivity) can be defined from the "at least as expressive" relation in the usual way. If neither language is at least as expressive as the other, we say that they are *incomparable* in expressivity.

In [18] that introduced PAL it was shown that PAL is equally expressive as ML (on the class of relational models where all accessibility relations are equivalence relations, but this does not matter for the reduction), and in [5] that introduced AML it was also shown that AML is equally expressive as ML. It is trivial to show that PAL and AML are at least as expressive as ML, as they extend the logical language. That every formula of PAL or AML is equivalent to a formula in ML, was shown by reduction axioms and rules. Similarly, AUL [15], AUML [16] and AAML [12] were shown to be equally expressive as ML, and therefore also equally expressive as PAL and AML. In [8] it was shown that RML is equally expressive as ML. Here, in Theorem 12 in Section 3, we showed that AAUML is also equally expressive as ML.

The two remaining logics are APAL and AAUL. The logic APAL was shown to be more expressive than ML in [4] and AAUL was shown to be more expressive than ML in [28], wherein it was also shown that APAL and AAUL are incomparable. This means that the expressivity landscape is as shown in Figure 2.

$$\begin{tikzcd}
& \text{ML, PAL, AML, AAML, AUL, AUML, AAUML, RML} \arrow[dl] \arrow[dr] & \\
\text{APAL} & & \text{AAUL}
\end{tikzcd}$$

Figure 2: The relative expressivity of the update logics discussed in the paper. Arrows indicate increasing expressivity. Absence of arrows indicates incomparable expressivity.

5.2 Update expressivity hierarchy

There is something a bit strange about this comparison, however. Although AML and PAL have the same expressivity, AML is clearly in some sense more powerful, since action models represent a far larger class of updates than public announcements. In order to capture the sense in which AML is more powerful than PAL, we use the term *update expressivity*. This concept was introduced as *action equivalence* in [31] (and its precursors) and also subsequently used in that sense in [16]. The definitions from [31, 16] do not deal very well with multi-pointed update modalities and with arbitrary update modalities, however, so we use a slightly adapted version.

The updates X we consider are relational model transformers and such transformations are defined by pairwise relating pointed models, so an update X should be seen as a relation between pointed relational models. In fact, three different things are called update: the *update relation* between pointed models, the *update modality* in a logical language, and, in some sense, the *update object*, often a name, that can be associated with the modality or the relation, such as an arrow update (U, o). To simplify the presentation in this section we call the relations $X, Y, \dots$ and the modalities $[X], [Y], \dots$ and we do not consider the update objects separately: we identify them with the update relations.

A relation between pointed relational models can be a one-to-one relation, i.e., a function or a partial function, a one-to-finitely-many relation, and a one-to-infinitely-many relation. For example, it is a function for a pointed arrow update model, a partial function for a public announcement, a one-to-many relation for a multi-pointed arrow update model, and a one-to-infinitely-many relation for an arrow update quantifier. In the first place, one would now like to say that update relations X and Y are the same (are equivalent) if they define the same relation between pointed models, modulo bisimulation. In the second place, we also want to compare an update X that is a partial function to an update Y that is a total function (or similarly for relations with restricted domains). In that kind of situation one would maybe like to say that updates X and Y are the same if X and Y define the same relation *on the domain of* X: we will then say that X is conditionally equivalent to Y (this relation is asymmetric). Such a requirement seems common practice in dynamic epistemic logic, and it is also respected in [16]. We recall from Section 1 the 'state eliminating' public announcement of p (i) and the 'arrow eliminating' public announcement of p (ii), originating with [10]: whenever p can be truthfully announced, the pointed relational models resulting from executing (i) and (ii) are bisimilar, as in the example. But when p is false, (ii) can be executed but not (i). So (i) and (ii) are equivalent on condition of the truth of the announcement. In view of these considerations, we propose the following definition. For $X(M,s)$ read $\{(M',s') \mid ((M,s),(M',s')) \in X\}$, and let $\mathcal{D}(X)$ be the domain of X, i.e., $\mathcal{D}(X) = \{(M,s) \mid X(M,s) \neq \emptyset\}$.

Definition 18 (Update equivalence, update expressivity) Given updates X and Y, X *is conditionally update equivalent to* Y, if for all (M,s) such that $s \in \mathcal{D}(X)$, $X(M,s) \leftrightarroweq Y(M,s)$. Further, X *is update equivalent to* Y, if X is conditionally update equivalent to Y, and Y is conditionally update equivalent to X. Update modalities $[X]$ and $[Y]$ are (conditionally) update equivalent, if X and Y are (conditionally) update equivalent.

A language $\mathcal{L}$ is *at least as update expressive as* $\mathcal{L}'$ if for every update modality $[X]$ of $\mathcal{L}'$ there is an update modality $[Y]$ of $\mathcal{L}$ such that X is conditionally update equivalent to Y; $\mathcal{L}$ is *equally update expressive as* $\mathcal{L}'$ (or 'as update expressive as') if $\mathcal{L}$ is at least as update expressive as $\mathcal{L}'$ and $\mathcal{L}'$ is at least as update expressive as $\mathcal{L}$. $\dashv$

We define '(strictly) more update expressive' and 'incomparable in update expressivity' as usual. We also extend the usage of 'update expressive' to the logics for the languages that we compare. Instead of 'update equivalent' we may use 'equivalent' if the context is clear. If updates X and Y are update equivalent, then $[X]\varphi \leftrightarrow [Y]\varphi$ is valid. This may not hold in the other direction! In Section 7 we give a counterexample.

We should stress that we do not claim that our definition is appropriate for all situations, merely that it gives an accurate view of the strengths of the different logics that we consider in this particular paper.

Let us now fill in the expressivity hierarchy for our target logics. The update expressivity of AUL is higher than that of PAL, and lower than that of AML [15]. The comparison between AML and AUML that we will address in Section 6 is less straightforward than that. In [16] it was shown that AML and AUML have the same update expressivity. That result does not distinguish between single-pointed and multi-pointed action models and arrow update models, however. Here, we therefore provide an alternative proof of their results that makes that distinction. Specifically, we show that the result from [16] only applies to the multi-pointed case, but that single-pointed arrow update models are more update expressive than single-pointed action models.

Adding quantification increases update expressivity, since the non-quantified logics cannot simulate a one-to-infinity relation. So, for example, APAL is more update expressive than PAL, and AAUML is more update expressive than AUML. When comparing the quantified logics among themselves, most pairs are incomparable. These incomparability results are all rather trivial, so we do not prove them here. The only comparable pair is AAUML and AAML, which have the same update expressivity because their underlying updates have the same update expressivity (Section 6). In Section 7 we will show that RML is incomparable to the other quantified logics, and in particular that the AAUML and AAML quantifiers are contained in the RML quantifier.

The landscape of update expressivity is therefore as shown in Figure 3.

$$\begin{tikzcd}
 & \text{APAL} & \text{AAUL} \\
\text{ML} \arrow[r] \arrow[d] & \text{PAL} \arrow[r] \arrow[u] \arrow[d] & \text{AUL} \arrow[u] \arrow[d] \\
\text{RML} & \text{AML}_1 \arrow[r] \arrow[d] & \text{AUML}_1 \arrow[d] \\
 & \text{AML} \arrow[r, leftrightarrow] \arrow[d] & \text{AUML} \arrow[d] \\
 & \text{AAML} \arrow[r, leftrightarrow] & \text{AAUML}
\end{tikzcd}$$

Figure 3: The relative update expressivity of the update logics discussed in the paper. We assume transitive closure of arrows. AML_1 and AUML_1 are the single-pointed versions.

6 Arrow updates versus action models

6.1 Action model logic

Arrow update model logic AUML is equally expressive as action model logic AML and their update expressivity relates in interesting ways. We build upon the results known from [16] but our constructions and proofs are slightly different. First we need to define action models and their execution in relational models. An action model [5] is a structure like a relational model but with a precondition function instead of a valuation function. Executing an action model into a relational model means computing what is known as their restricted modal product. This product encodes the new state of information, after action execution. These are the technicalities:

An *action model* $E = (\mathsf{S}, \mathsf{R}, \mathsf{pre})$ consists of a *domain* S of *actions* $e, f, \dots$, an *accessibility function* $\mathsf{R} : A \to \mathcal{P}(\mathsf{S} \times \mathsf{S})$, where each R_a is an accessibility relation, and a *precondition function* $\mathsf{pre} : \mathsf{S} \to \mathcal{L}$, where $\mathcal{L}$ is a logical language.

Let additional to a pointed action model (E, e) as above a pointed relational model (M, s) be given where $M = (S, R, V)$. Let $M, s \models \mathsf{pre}(\mathsf{s})$. The update $(M \otimes E, (s, e))$ is the pointed relational model where $M \otimes E = (S', R', V')$ such that

$$\begin{array}{lcl}
S' & = & \{(t, f) \mid M, t \models \mathsf{pre}(f)\} \\
((t, f), (t', f')) \in R'_a & \text{iff} & (t, t') \in R_a \text{ and } (f, f') \in \mathsf{R}_a \\
(t, f) \in V'(p) & \text{iff} & t \in V(p)
\end{array}$$

Action model modalities $[E, e]$ are interpreted similarly to arrow update modalities but unlike arrow update modalities are partial and not functional. Their execution depends on the truth of the precondition of the actual action (point) e in the actual state s:

$$M, s \models [E, e]\varphi \quad \text{iff} \quad M, s \models \mathsf{pre}(e) \text{ implies } M \otimes E, (s, e) \models \varphi$$

Similarly to arrow update modalities we can conceive a modal logical language with $[E, e]\varphi$ as an inductive language construct, for action models E with finite domains. The logic is called *action model logic* AML. And also similarly we can define multi-pointed action model by notational abbreviation, or if we wish to rule that out call the logic AML_1. Also similarly to AUML ([16], and Section 4) there is a complete axiomatization, that is a rewrite system allowing to eliminate dynamic modalities [5, 27]. If we further extend the logical language with a quantifier $[\otimes]$ over action models, such that

$$M, s \models [\otimes]\varphi \quad \text{iff} \quad M, s \models [E, e]\varphi \text{ for all action models } (E, e) \text{ satisfying } (*)$$

where (*) requires all preconditions of actions in E to be in $\mathcal{L}_{ml}$, we get the language and logic of *arbitrary action model logic* AAML. Hales showed in [12] that the (*) requirement can be relaxed, similarly to our Proposition 13.

Example action models that are update equivalent to the example arrow update models of Section 2 are as follows. We also depict their execution. The actions are given their preconditions as names. Note that the pointed relational model resulting from the (second) action of Anne privately learning that p is bisimilar to the four-state model in Section 2.

$$\begin{tikzcd}
& & & \neg p \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "b"] \arrow[dr, leftrightarrow, "b"'] & \boxed{p} \arrow[loop right, "ab"] \arrow[d, leftrightarrow, "b"] & & (\neg p, \neg p) \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "b"] \arrow[d, leftrightarrow, "b"'] & \boxed{(p, p)} \arrow[loop right, "ab"] \arrow[d, leftrightarrow, "b"] \\
\neg p \arrow[loop above, "ab"] \arrow[r, leftrightarrow, "ab"] & \boxed{p} \arrow[loop above, "ab"] & \otimes & & \top \arrow[loop right, "ab"] & = & (\neg p, \top) \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "ab"] & (p, \top) \arrow[loop right, "ab"]
\end{tikzcd}$$

$$\begin{tikzcd}
& & & & \boxed{p} \arrow[loop right, "a"] \arrow[d, "b"] & & & \boxed{(p, p)} \arrow[loop right, "a"] \arrow[dl, "b"'] \arrow[d, "b"] \\
\neg p \arrow[loop above, "ab"] \arrow[r, leftrightarrow, "ab"] & \boxed{p} \arrow[loop above, "ab"] & \otimes & & \top \arrow[loop right, "ab"] & = & (\neg p, \top) \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "ab"] & (p, \top) \arrow[loop right, "ab"]
\end{tikzcd}$$

6.2 From action models to arrow updates

A given action model can be transformed into an arrow update model by decorating each arrow in the action model with a source condition that is the precondition of the source action and with a target condition that is the precondition of the target action. That is all. Technically:

Let $E = (\mathsf{S}, \mathsf{R}, \mathsf{pre})$ be given. Arrow update model $U(E) = (O, RR)$ is defined as: $O = \mathsf{S}$, and for all agents a and actions e, e', $(e, \mathsf{pre}(e)) \rightarrow_a (e', \mathsf{pre}(e'))$ iff $(e, e') \in \mathsf{R}_a$.[2] We can now show that (E, e) is update equivalent to $(U(E), e)$, on condition of the executability of the action e, i.e., restricted to the denotation of $\mathsf{pre}(e)$.

Proposition 19 ([16]) (E, e) is conditionally update equivalent to $(U(E), e)$. $\dashv$

Proof Let $M = (S, R, V)$. To distinguish E from $U(E)$ in the proof, let us 'prime' the actions in the domain of $U(E)$: $e', f', \dots$ instead of $e, f, \dots$. Define relation $\mathfrak{R} : \mathcal{D}(M \otimes E) \rightarrow \mathcal{D}(M * U(E))$ as $\mathfrak{R} : (s, e) \mapsto (s, e')$, for all $s \in S$ and $e \in \mathsf{S}$. The full modal product $M * U(E)$ will typically have a larger domain than the restricted modal product $M \otimes E$ (only states wherein actions in S can be executed, survive). We will now show that this does not matter, as the surplus of (state,action) pairs are unreachable.

Let $M, s \models \mathsf{pre}(e)$. Then $\mathfrak{R} : (M * E, (s, e)) \underline{\leftrightarrow} (M \otimes U(E), (s, e'))$.

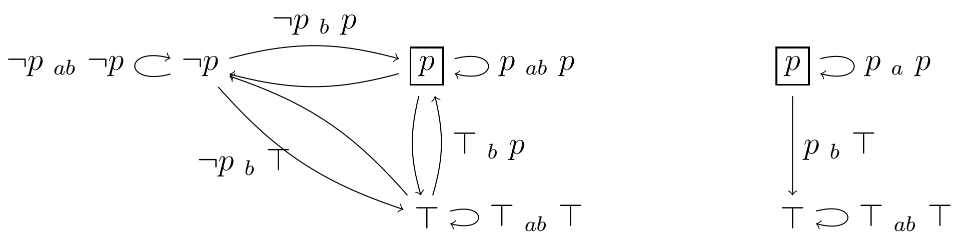
Let $((s, e), (s, e')) \in \mathfrak{R}$. The **atoms** clause is trivially satisfied as the states s match (and as update do not change facts). We now consider **forth**. Let $((s, e), (t, f)) \in R'_a$, where R'_a is the accessibility relation for a in $M \otimes E$. Given the definition of action model execution, $((s, e), (t, f)) \in R'_a$ iff $(s, t) \in R_a$, $(e, f) \in \mathsf{R}_a$, $M, s \models \mathsf{pre}(e)$, and $M, t \models \mathsf{pre}(f)$. We claim that pair (t, f') in $M * U(E)$ satisfies the requirements of the bisimulation. Firstly, $((s, e'), (t, f')) \in R''_a$ (in $M * U(E)$) because $(e', \mathsf{pre}(e')) \rightarrow_a (f', \mathsf{pre}(f'))$. Secondly, $(t, f), (t, f')) \in \mathfrak{R}$ by the definition of $\mathfrak{R}$. The proof of **back** is like **forth**, with the main important difference that the assumption in the **back** step that a pair (t, f') in $M * U(E)$ is a-accessible, means that we are in the 'good' part of the domain of $M * U(E)$, as this implies that $(e', \mathsf{pre}(e')) \rightarrow_a (f', \mathsf{pre}(f'))$, so that in the construction of $M \otimes E$, action e can be executed in s and action f in t. So (t, f) exists in $M \otimes E$ and this obviously satisfies the requirement for **back**. $\square$

Corollary 20 Let $F \subseteq \mathcal{D}(E)$. Then (E, F) is conditionally update equivalent to $(U(E), F)$. $\dashv$

From Proposition 19 follows that, for all $\varphi \in \mathcal{L}_{ml}$, $[E, e]\varphi$ is equivalent to $\mathsf{pre}(e) \rightarrow [U(E), e]\varphi$. See also [16][Cor. 3.9]. This can be used as a clause in an inductively defined translation from the language of AUML to the language of AML. In Corollary 20 the condition for update equivalence is $\bigvee_{e \in F} \mathsf{pre}(e)$.

The arrow update models constructed by the above procedure from the action model for Anne reading the letter containing p while Bill may notice her doing so, and for Anne privately learning p, are as follows. Note that they are update equivalent (namely on their domain of execution) to the arrow update models for these actions presented in Section 2. In the figure, by $\varphi \;_{ab}\; \varphi'$ we mean the two arrows $\varphi \rightarrow_a \varphi', \varphi \rightarrow_b \varphi'$. On the left, the dual arrows for b between outcomes have not been labelled. They are as expected: $\top \rightarrow_b \neg p$, $p \rightarrow_b \neg p$, $p \rightarrow_b \top$. (They are update equivalent to the example arrow update models in Section 2.4.)

[2]In [16], arrows $(e, \top) \rightarrow_a (e', \varphi')$ instead of $(e, \varphi) \rightarrow_a (e', \varphi')$ are stipulated. Both constructions deliver the desired update equivalence.



6.3 From arrow updates to action models

A given arrow update model can be transformed into an update equivalent action model by conditionalizing in each outcome over any possible 'valuation' of (any subset of) the source and target conditions of all outcomes. This leads to an exponential blowup. (See [16, Theorem 3.7]. Our construction and subsequent proof are different.) We proceed with the construction.

Let $U = (O, RR)$ be given. Let Φ be the collection of all source and target conditions occurring in U:

$$\Phi = \{\varphi \mid \text{there are } a \in A, \varphi' \in \mathcal{L}, o, o' \in O \text{ s.t. } (o,\varphi) \to_a (o',\varphi') \text{ or } (o,\varphi') \to_a (o',\varphi)\}.$$

We consider the formulas in Φ as 'atomic constituents' over which we consider 'valuations' $v \in 2^\Phi$ (lower case, to distinguish it from the relational model valuation V, upper case). The characteristic formula of a valuation is $\delta_v := \bigwedge_{\varphi\in\Phi} \overline{\varphi}$, where $\overline{\varphi} = \varphi$ if $v(\varphi) = 1$ and $\overline{\varphi} = \neg\varphi$ if $v(\varphi) = 0$. Action model $E(U) = (\mathsf{S}, \mathsf{R}, \mathsf{pre})$ is now such that:

$$\begin{array}{lcl} \mathsf{S} & = & O \times 2^\Phi \\ ((o,v),(o',v')) \in \mathsf{R}_a & \text{iff} & \exists \varphi,\varphi' : (o,\varphi) \to_a (o',\varphi'), v(\varphi) = 1, v'(\varphi') = 1 \\ \mathsf{pre}(o,v) & = & \delta_v \end{array}$$

Further, given (U, o), its single point o becomes a set of actions $E(o) := \{o\} \times 2^\Phi$. The corresponding action model $(E(U), E(o))$ is therefore multi-pointed (unless $\Phi = \{\top\}$ or $\Phi = \{\bot\}$). We note that the preconditions of actions need not be consistent formulas, just as source and target conditions of arrows need not be consistent formulas. Our construction is therefore different from that in [16], wherein only $v \in 2^\Phi$ are considered for which δ_v is consistent. That construction is more economical, but computational efficiency is not our goal. We can now show that (U, o) is update equivalent to $(E(U), E(o))$. Note that they both can be executed on the entire domain.

Proposition 21 (U, o) is update equivalent to $(E(U), E(o))$. $\dashv$

Proof Let (M, s) be given. We show that for some $(o, v) \in E(o)$, $(M * U, (s, o)) \leftrightarroweq (M \otimes E(U), (s, o, v))$. Define relation $\mathfrak{R}$ as follows:

$$\mathfrak{R} : (s,o) \mapsto (s,o,v) \quad \text{iff} \quad M, s \models \delta_v$$

We show that $\mathfrak{R} : (M{*}U, (s,o)) \leftrightarroweq (M{\otimes}E(U), (s,o,v))$. For **forth**, assume $((s,o),(s,o,v)) \in \mathfrak{R}$ and $((s,o),(s',o')) \in R'_a$ (in $M * U$). The latter implies that there are $\varphi, \varphi' \in \mathcal{L}$ such that $(o,\varphi) \rightarrow_a (o',\varphi')$, and that $M,s \models \varphi$ and $M,s' \models \varphi'$. Choose v' such that $M,s' \models \delta_{v'}$ (note that v' exists and is unique). As $M,s \models \varphi$ and $M,s' \models \varphi'$ respectively $M,s \models \delta_v$ and $M,s' \models \delta_{v'}$, we have that $v(\varphi) = 1$ and $v'(\varphi') = 1$. We claim that (s',o',v') is the requested witness to close the **forth** argument. Firstly, $((s,o,v),(s',o',v')) \in R''_a$ (in $M \otimes E(U)$) because $(s,s') \in R_a$ (in M) and $((o,v),(o',v')) \in \mathsf{R}_a$ (in $E(U)$), where the latter follows from $(o,\varphi) \rightarrow_a (o',\varphi')$, $v(\varphi) = 1$ and $v'(\varphi') = 1$ (see the definition of $E(U)$ above). Secondly, $((s',o'),(s',o',v') \in \mathfrak{R}$. Step **back** is very similar; now use that $\mathsf{pre}(o,v) = \delta_v$ and that $\mathsf{pre}(o',v') = \delta_{v'}$, and observe that $(E(U),E(o))$ can always be executed, it has precondition $\top$ ($\bigvee_{v \in 2^\Phi} \delta_v$ is equivalent to $\top$). $\square$

Corollary 22 Let now $Q \subseteq \mathcal{D}(U)$. Then (U,Q) is update equivalent to $(E(U),E(Q))$. $\dashv$

As an example we now show the action models constructed by the above procedure from the two example arrow update models of Section 2. The set of source and target condition formulas is $\{p, \neg p, \top\}$. Of their 8 valuations the two non-trivial (and different) valuations are characterized by p and $\neg p$. These formulas are also the action preconditions in the action points of the resulting action models. The reader may observe that these action models are, again, update equivalent to their 'original' action models at the start of this section.

$$\begin{tikzcd}
\boxed{(\circ,\neg p)} \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "b"] \arrow[d, leftrightarrow, "b"'] & \boxed{(\circ,p)} \arrow[loop right, "ab"] \arrow[d, leftrightarrow, "b"] \\
(\bullet,\neg p) \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "ab"] & (\bullet,p) \arrow[loop right, "ab"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\boxed{(\circ,\neg p)} \arrow[r, "a"] \arrow[d, "b"'] & \boxed{(\circ,p)} \arrow[loop right, "a"] \arrow[d, "b"] \\
(\bullet,\neg p) \arrow[loop left, "ab"] \arrow[r, leftrightarrow, "ab"] & (\bullet,p) \arrow[loop right, "ab"]
\end{tikzcd}$$

6.4 Relative update expressivity

We are now prepared to harvest the update expressivity results. First, let us show that AUML_1 is more update expressive than AML_1.

Proposition 23 AUML_1 is more update expressive than AML_1. $\dashv$

Proof From Proposition 19 follows that AUML_1 is at least as update expressive as AML_1. To show that the inclusion is strict, we need to show that for some (U,o), there is no (E,e) that induces the same relation.

Let U be the arrow update model with a single outcome o, and a single arrow $(o,p) \rightarrow_a (o,\top)$. Suppose towards a contradiction that there is a single-pointed action model (E,e) such that $(M*U,(s,o)) \leftrightarroweq (M \otimes E,(s,e))$ for every (M,s). Then, in particular, (E,e) must be executable everywhere, so $\mathsf{pre}(e)$ is equivalent to $\top$. Furthermore, if $M,s \models p \wedge \Diamond_a \top$, then $(M*U,(s,o))$ has at least one a-successor (s',o'). By assumption, $(M \otimes E,(s,e))$ is bisimilar to $(M * U, (s, o))$, so it has an a-successor (s', e'). This implies that $e\mathsf{R}e'$. Now, consider a different model (M', s) such that (i) $M', s \models \neg p$ and (ii) s has an a-successors s' such that $M', s' \models \mathsf{pre}(e')$. Then $(M' \otimes E, (s, e))$ has an a-successor, but $(M' * U, (s, o))$ has no a-successor. This contradicts the assumption that $(M * U, (s, o)) \leftrightarrows (M \otimes E, (s, e))$ for every (M, s). $\Box$

Proposition 24 AUML is equally update expressive as AML. $\dashv$

Proof From Corollary 20 follows that AUML is at least as update expressive as AML. From Corollary 22 follows that AML is at least as update expressive as AUML. $\Box$

It is obvious that AUML is more update expressive than AUML$_1$, and that AUML is more update expressive than AUML$_1$.

6.5 Applications illustrating the succinctness of arrow updates

In this section we give some application areas for the modelling of information change, where arrow updates are more succinct that corresponding (i.e., update equivalent) action models.

Lying You are lying if you say that something is true while you believe that it is false — and with the intention for the addressee(s) to believe that it is true. In the setting of public announcement logic a lie is a public announcement that is false. This is then contrasted to the (usual) public announcement that is true. Both are combined in the announcement that has no relation to its truth. This is known as the conscious update ([10], see the introduction) or, in a setting where lying is also distinguished, as the manipulative update [30]. The arrow update for the conscious/manipulative update of φ is the singleton arrow update model with arrows

$$(o, \top) \to_a (o, \varphi)$$

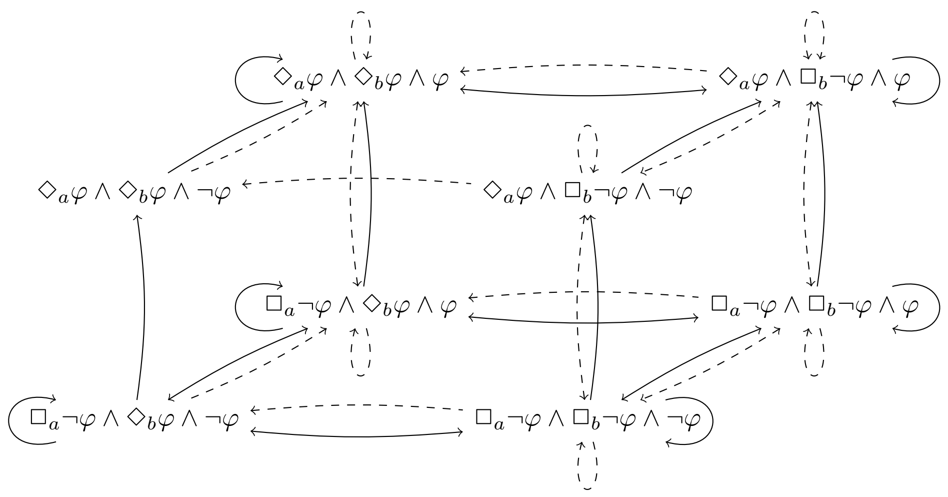
for all agents. Lying as such, wherein φ is required to be false, is not an arrow update as arrow update models have no preconditions, but such executability preconditions can be simulated as antecedents of logical implications.

A problem with the manipulative update is that an agent who already believes the opposite of the lie, believes everything after incorporating the lie into her beliefs (believing a contradiction comes at that price). This is because the accessibility of that agent becomes empty as a result of the update. A solution to that is the *cautious update* that is also known as lying to *sceptical* agents [19, 15, 23]: the agent only updates her beliefs if the new information is consistent with her current beliefs. The arrow update for the *sceptical update of* φ (again, we cannot model sceptical *lying* as this requires φ to be false) is a singleton arrow update model with arrows

$$\begin{array}{lcl} (o, \Diamond_a \varphi) & \to_a & (o, \varphi) \\ (o, \Box_a \neg\varphi) & \to_a & (o, \top) \end{array}$$

for all agents [15].

Given a group of agents, some may believe the announcement, and others not. The arrow update modelling allows for this. However, if we make an action model for announcements to sceptical agents, we need to distinguish all combinations explicitly (we used a similar construction to get action model $E(U)$ from arrow update U, above, in order to prove Prop. 21). For example, for two agents a and b, the action model consists of eight actions, with preconditions and accessibility relations as follows. In the picture we 'name' the actions with their preconditions. To simplify the visualization, we do not label arrows with a and b: solid arrows are for a and dashed arrows for b. We also assume transitive closure of accessibility.


$\Diamond_a\varphi \wedge \Diamond_b\varphi \wedge \varphi$
$\Diamond_a\varphi \wedge \Box_b\neg\varphi \wedge \varphi$
$\Diamond_a\varphi \wedge \Diamond_b\varphi \wedge \neg\varphi$
$\Diamond_a\varphi \wedge \Box_b\neg\varphi \wedge \neg\varphi$
$\Box_a\neg\varphi \wedge \Diamond_b\varphi \wedge \varphi$
$\Box_a\neg\varphi \wedge \Box_b\neg\varphi \wedge \varphi$
$\Box_a\neg\varphi \wedge \Diamond_b\varphi \wedge \neg\varphi$
$\Box_a\neg\varphi \wedge \Box_b\neg\varphi \wedge \neg\varphi$

For n agents there are $O(2^n)$ actions in the action model. In [15] this example is treated in greater detail, and also other, similar, examples are shown for which arrow updates are shown to be more succinct (exponentially smaller).

Attentive announcements Another example where action models are exponentially bigger than arrow updates is that of the *attention-based announcements* of [7]. This work presents a logic of announcements that are only 'heard' (received) by agents paying attention to it, paying attention to the announcer, so to speak. Such announcements are modelled employing an auxiliary set of designated 'attention (propositional) variables' h_a expressing that agent a pays attention. The corresponding arrow update model has domain $\{o, o'\}$, both outcomes designated, and with arrows

$$\begin{array}{lll} (o, h_a) & \to_a & (o, \varphi) \\ (o, \neg h_a) & \to_a & (o', \top) \\ (o', \top) & \to_a & (o', \top) \end{array}$$

for all agents. It cannot be modelled with a (singleton) [15] arrow update, the resulting relational model is typically larger than then model before the update, as the agents not paying attention believe that no announcement was made, and thus reason about the structure of the entire initial model. Incorporating the announcements depends on h_a being true or false, just as for sceptical announcements it depends on $\Diamond_a\varphi$ being true or false. So, this is similar. But not entirely so, because an agent not paying attention is, so to speak, inconscious of the announcement, and thus believes that the (entire) original model still encodes her beliefs), whereas an agent believing the opposite of the announcement 'knows' that if she where to have found the announcement believable, she would have changed her beliefs. So these are different parts of the same model, it is a mere restriction of the accessibility relation. Again, for attentive announcements, a corresponding action model is of exponential size, as any subset of agents may or may not be paying attention. See [7].

Comparative size of action models and arrow updates In general, if the observational powers of *all* agents are commonly known to be partial, then we can expect arrow updates for such dynamic phenomena to be exponentially smaller than corresponding action models. This was the case for announcements to sceptical lying and for attention-based announcements, and also for: agents making broadcasts (to all agents), agents seeing each other depending on their orientation, partial networks representing agents with neighbours or friends, etc. On the other hand, dynamic phenomena where all agents observe (some, few) designated agents have similarly-sized arrow updates and action models, such as: the private announcement to an individual agent or a subgroup of agents, and gossip scenarios where two agents call each other in order to exchange secrets, and where this call may be partially observed by all other agents. We do not know of scenarios where action models are more succinct than arrow updates.

7 Arbitrary arrow updates versus refinements

7.1 Refinement modal logic

We now compare the arbitrary arrow update modality of AAUML to the *refinement* quantifier of refinement modal logic RML [8]. Let us first be precise about its syntax and semantics.

We recall the definition of bisimulation in Section 2.1. If **atoms** and **back** hold, we call the relation a *refinement* (and dually, if **atoms** and **forth** hold, we call the relation a *simulation*). In [8] such a refinement relation is considered for any subset of the set of agents and defined as follows:

A relation $\mathfrak{R}_B$ that satisfies **atoms**, **back-**a, and **forth-**a for every $a \in A \setminus B$, and that satisfies **atoms**, and **back-**b for every $b \in B$, is a B-*refinement*, we say that (M', s') *refines* (M, s) for group of agents B, and we write $(M, s) \succeq_B (M', s')$. An A-*refinement* we call a *refinement* (clearly any B-refinement is also an A-refinement and thus a 'refinement' plain and simple), and $(M, s) \succeq_A (M', s')$ is denoted $(M, s) \succeq (M', s')$. With this relation comes a corresponding modality in the obvious way. Let $B \subseteq A$, and (M, s) and φ given, then

$$M, s \models [\succeq]_B \varphi \quad \text{iff} \quad M', s' \models \varphi \text{ for all } (M', s') \text{ such that } (M, s) \succeq_B (M', s').$$

Comparison of three quantifiers We now first focus on the refinement relation $\succeq$ for the set of all agents, to which corresponds the $[\succeq]$ modality. Consider three different ways to define quantification in information changing modal logics. We formulate them suggestively so that their correspondences stand out, where we recall Proposition 13 that the restrictions on source and target conditions need not be met when interpreting $[\uparrow]\varphi$, and similarly, [12] showed that the restrictions on action preconditions need not be met when interpreting $[\otimes]$.

$$\begin{array}{lll} M, s \models [\uparrow]\varphi & \text{iff} & M, s \models [U, o]\varphi \text{ for all arrow update models } (U, o) \\ M, s \models [\succeq]\varphi & \text{iff} & M', s' \models \varphi \text{ for all refinements } (M', s') \\ M, s \models [\otimes]\varphi & \text{iff} & M', s' \models [E, e]\varphi \text{ for all action models } (E, e) \end{array}$$

Theorem 25 Let $\varphi \in \mathcal{L}_{ml}$. Then $[\uparrow]\varphi$, $[\succeq]\varphi$, $[\otimes]\varphi$ are pairwise equivalent. ⊣

Proof

- The validity of $[\succeq]\varphi \leftrightarrow [\otimes]\varphi$ was shown in [12].
- To show that $[\otimes]\varphi \leftrightarrow [\uparrow]\varphi$, we use the semantics of these modalities. Let us do this for the diamond version. Both directions of the equivalence need to be shown.

$$\begin{array}{lr} M, s \models \langle\otimes\rangle\varphi & \\ \Leftrightarrow & \\ \exists(E, e) : M, s \models \langle E, e\rangle\varphi & \\ \Leftrightarrow & \text{Proposition 19} \\ \exists(U(E), e) : M, s \models \mathsf{pre}(e) \ \& \ M, s \models \langle U(E), e\rangle\varphi & \\ \Rightarrow & \\ \exists(U(E), e) : M, s \models \langle U(E), e\rangle\varphi & \\ \Leftrightarrow & \\ M, s \models \langle\uparrow\rangle\varphi & \end{array}$$

For the other direction we get this:

$$\begin{array}{lr} M, s \models \langle\uparrow\rangle\varphi & \\ \Leftrightarrow & \\ \exists(U, o) : M, s \models \langle U, o\rangle\varphi & \\ \Leftrightarrow & \text{Proposition 21} \\ \exists(E(U), E(o)) : M, s \models \langle E(U), E(o)\rangle\varphi & \\ \Rightarrow & \text{where } e \in E(o) \\ \exists(E(U), e) : M, s \models \langle E(U), e\rangle\varphi & \\ \Leftrightarrow & \\ M, s \models \langle\otimes\rangle\varphi & \end{array}$$

- To show that $[\uparrow]\varphi \leftrightarrow [\succeq]\varphi$ we use the previous two equivalences. $\square$

The theorem is formulated to make the correspondence between the three quantifiers stand out. Alternatively, we can have a inductively defined translation between the language $\mathcal{L}$ (of AAUML) and the language of arbitrary action model logic AAML that is compositional to the extent that arrow update quantifiers are translated into action model quantifiers (Theorem 25) and arrow update models into action models (Proposition 21), and vice versa (Proposition 19).

7.2 Update expressivity

Considering that $[\otimes]\varphi$, $[\uparrow]\varphi$ and $[\succeq]\varphi$ are equivalent (Theorem 25), and that $[\otimes]$ and $[\uparrow]$ have the same update expressivity (Section 5), one might expect all three to have the same update expressivity. This, however, is not so, because $[\otimes]$ and $[\uparrow]$ are finitary quantifiers — they quantify over, respectively, *finite* action models and over *finite* arrow update models — whereas refinements can be infinitary.

For one example, consider the relational model N consisting of all valuations, with the universal relation on that domain for all agents, and any state t in that domain. Clearly, the restriction of N to the singleton model consisting of t (wherein the agents have common knowledge of the valuation in t) is a refinement of (N, t). It can be obtained by successively announcing the value of each of the infinite number of atoms. However, it cannot be obtained by a single announcement (or, equivalently, by any finite sequence of those).

For another example, consider the following model M, with as single state s_0:

$$\begin{tikzcd}
\bullet \arrow[loop below, "a"]
\end{tikzcd}$$

Now, consider the following model M', with s_0 as its leftmost state:

$$\begin{tikzcd}
 & & & \bullet \\
 & & \bullet & \bullet \arrow[u, "a"'] \\
 & \bullet & \bullet \arrow[u, "a"'] & \bullet \arrow[u, "a"'] \\
\bullet \arrow[r, "a"'] & \bullet \arrow[u, "a"'] \arrow[r, "a"'] & \bullet \arrow[u, "a"'] \arrow[r, "a"'] & \bullet \arrow[u, "a"'] \arrow[r, "a"'] & \cdots
\end{tikzcd}$$

The pointed model (M', s_0) is a refinement of (M, s_0). But M' contains infinitely many states that are not bisimilar to one another. Furthermore, every arrow update model U is finite, so every product of U with M is finite (and therefore contains finitely many non-bisimilar states). As a result, there is no (U, o) such that (M', s_0) is bisimilar to $(M * U, (s, o))$.

It follows that arbitrary arrow updates are *not* at least as update expressive as refinements. But it also follows that refinements are not at least as update expressive as arbitrary arrow updates, since you cannot choose to exclude the above model (M', s_0) when performing a refinement in (M, s_0).

Proposition 26 RML and AAUML are incomparable in update expressivity. $\dashv$

And therefore RML and AAML are also incomparable. The reason that $[\uparrow]\varphi$ and $[\succeq]\varphi$ (and $[\otimes]\varphi$) are nonetheless equivalent is that while there is no (U, o) such that $(M * U, (s_0, o))$ is bisimilar to (M', s_0), it is the case that for every n there is an (U_n, o_n) such that $(M * U_n, (s_0, o_n))$ is n-bisimilar to (M', s_0). Since every formula in the languages under consideration is of finite depth, such finite approximations of M' suffice.

Finally, we should note that the incomparability already applies to the language for RML with only the $[\succeq]$ modality. The language above, as in [8], has $[\succeq]_B$ modalities for any subgroup $B \subseteq A$, meaning that, modulo bisimulation, only arrows in B are removed from a relational model. Similarly to the argument above it follows that this would only further increase expressivity.

7.3 Comparing the axiomatizations of AAUML and RML

We have seen that the refinement modality corresponds to the arrow update modality in the sense that $[\succeq]\varphi$ is equivalent to $[\uparrow]\varphi$. Given this identification, the language and semantics of AAUML thus extends that of RML. A comparison between the axiomatization **RML** of refinement modal logic RML (Table 3) and the axiomatization **AAUML** of AAUML (Table 2 on page 20) seems in order. In **RML**, in Table 3, ∇ is the *coalgebraic cover modality* of [17], defined as $\nabla\Phi := \bigwedge_{\varphi\in\Phi} \Diamond\varphi \land \Box \bigvee_{\varphi\in\Phi} \varphi$. From now on we abbreviate the right-hand term, and similar expressions, as $\nabla\Phi := \bigwedge \Diamond\Phi \land \Box \bigvee \Phi$.

Prop	all tautologies of propositional logic	
K	$\Box_a(\varphi \to \psi) \to \Box_a\varphi \to \Box_a\psi$	
R	$[\succeq]_a(\varphi \to \psi) \to [\succeq]_a\varphi \to [\succeq]_a\psi$	
RProp	$[\succeq]_a p \leftrightarrow p$ and $[\succeq]_a \neg p \leftrightarrow \neg p$	
RK	$\langle\succeq\rangle_a \nabla_a \Phi \leftrightarrow \bigwedge \Diamond_a \langle\succeq\rangle_a \Phi$	
RKmulti	$\langle\succeq\rangle_a \nabla_b \Phi \leftrightarrow \nabla_b \langle\succeq\rangle_a \Phi$	where $a \neq b$
RKconj	$\langle\succeq\rangle_a \bigwedge_{b\in B} \nabla_b \Phi^b \leftrightarrow \bigwedge_{b\in B} \langle\succeq\rangle_a \nabla_b \Phi^b$	
MP	from $\varphi \to \psi$ and φ infer ψ	
NecK	from φ infer $\Box_a\varphi$	
NecR	from φ infer $[\succeq]_a\varphi$	

Table 3: The axiomatization **RML** of RML

If we replace $\langle\uparrow\rangle$ by $\langle\succeq\rangle$ in axiom **A4** of Table 2 we get this principle **A4**$^{\succeq}$:

$$\mathbf{A4}^{\succeq} \quad \langle\succeq\rangle \bigwedge_{a\in A} (\bigwedge \Diamond_a \Phi_a \land \Box_a \psi_a) \leftrightarrow \bigwedge_{a\in B} \bigwedge \Diamond_a \langle\succeq\rangle (\Phi_a \land \psi_a)$$

Obviously, given Theorem 25, $\mathbf{A4}^{\succeq}$ is valid in RML. More interesting is to derive it in **RML** from the axioms **RK**, **RKmulti**, and **RKconj** relating the refinement modality to basic modalities.

Let us first demonstrate this for the single agent versions of the logics (where w.l.o.g. we assume that the language is arrow update model free, to simplify the proof). In other words, we compare:

$$\begin{array}{ll}\mathbf{A4}^{\succeq}_1 & \langle\succeq\rangle(\bigwedge\Diamond\Phi\wedge\Box\psi)\leftrightarrow\bigwedge\Diamond\langle\succeq\rangle(\Phi\wedge\psi)\\ \mathbf{RK}_1 & \langle\succeq\rangle\nabla\Phi\leftrightarrow\bigwedge\Diamond\langle\succeq\rangle\Phi\end{array}$$

Proposition 27 Axioms $\mathbf{RK}_1$ and $\mathbf{A4}^{\succeq}_1$ are interchangeable in $\mathbf{RML}_1$. ⊣

Proof Let $\mathbf{Ax}\vdash\varphi$ denote that φ is a theorem of system $\mathbf{Ax}$. We first show that $(\mathbf{RML}_1-\mathbf{RK}_1+\mathbf{A4}^{\succeq}_1)\vdash\mathbf{RK}_1$. Below, the equivalences either spell out definitions or correspond to provable equivalences.

$\langle\succeq\rangle\nabla\Phi$

$\Leftrightarrow$ by definition of ∇

$\langle\succeq\rangle(\bigwedge\Diamond\Phi\wedge\Box\bigvee\Phi)$

$\Leftrightarrow$ $\mathbf{A4}^{\succeq}_1$, where $\psi=\bigvee\Phi$

$\bigwedge\langle\succeq\rangle(\Phi\wedge\bigvee\Phi)$

$\Leftrightarrow$ use equivalence $\varphi\leftrightarrow(\varphi\wedge\bigvee\Phi)$ for all $\varphi\in\Phi$

$\bigwedge\langle\succeq\rangle\Phi$

We now show that $\mathbf{RML}_1\vdash\mathbf{A4}^{\succeq}_1$

$\langle\succeq\rangle(\bigwedge\Diamond\Phi\wedge\Box\psi)$

$\Leftrightarrow$ propositional logic

$\langle\succeq\rangle(\bigwedge\Diamond(\Phi\wedge\psi)\wedge\Box\bigvee(\Phi\wedge\psi))$

$\Leftrightarrow$ $\mathbf{RK}_1$, for the set $\{\varphi\wedge\psi\mid\varphi\in\Phi\}$

$\bigwedge\Diamond\langle\succeq\rangle(\Phi\wedge\psi)$ □

To compare **RK** and $\mathbf{A4}^{\succeq}$ for the multi-agent version we need to use the axioms **RKmulti** and **RMconj** as well. We also use the following validity: let the set of agents A be $\{b_1,\dots,b_n\}$, then $\langle\succeq\rangle\varphi$ (i.e., $\langle\succeq\rangle_A\varphi$) is equivalent to $\langle\succeq\rangle_{b_1}\dots\langle\succeq\rangle_{b_n}\varphi$ in any order of these agents. We may therefore additionally assume that $b_n=a$. If the language has $[\succeq]_B$ as primitives, in the axiomatization **RML** we may therefore so to speak have an additional axiom **RG**: $[\succeq]_B\varphi\leftrightarrow[\succeq]_{b_1}\dots[\succeq]_{b_n}\varphi$. Finally, below, note that Lemmas 6 (page 6) and 7 (page 13) can be assumed provable equivalences.

Proposition 28 Axiom $\mathbf{A4}^{\succeq}$ is derivable in **RML**. ⊣

Proof

$\langle\succeq\rangle\bigwedge_{b\in A}(\bigwedge\Diamond_a\Phi_a\wedge\Box_a\psi_a)$

$\Leftrightarrow$ **RG**

$\langle\succeq\rangle_{b_1} \dots \langle\succeq\rangle_a \bigwedge_{a\in A}(\bigwedge \Diamond_a \Phi_a \wedge \Box_a \psi_a)$

$\Leftrightarrow$ n applications of **RKconj**

$\bigwedge_{a\in A}\langle\succeq\rangle_{b_1} \dots \langle\succeq\rangle_a(\bigwedge \Diamond_a \Phi_a \wedge \Box_a \psi_a)$

$\Leftrightarrow$ **RK**$_1$ (Prop. 27)

$\bigwedge_{b\in A}\langle\succeq\rangle_{b_1} \cdots \bigwedge \Diamond_a \langle\succeq\rangle_a(\Phi_a \wedge \psi_a)$

$\Leftrightarrow$ repeated applications of Lemmas 6 and 7

$\bigwedge_{b\in A} \bigwedge \langle\succeq\rangle_{b_1} \dots \Diamond_a \langle\succeq\rangle_a(\Phi_a \wedge \psi_a)$

$\Leftrightarrow$ repeated applications of **RKmulti**, as $b_i \neq a$ for $i < n$

$\bigwedge_{b\in A} \bigwedge \Diamond_a \langle\succeq\rangle_{b_1} \dots \langle\succeq\rangle_a(\Phi_a \wedge \psi_a)$

$\Leftrightarrow$ **RG**

$\bigwedge_{a\in A} \bigwedge \Diamond_a \langle\succeq\rangle(\Phi_a \wedge \psi_a)$ $\Box$

Alternative axiomatization for RML Although we have now shown that **A4**$^\succeq$ is derivable from **RML**, it is of course impossible to show that **RK** is derivable from **RML** − **RK**+**A4**$^\succeq$, because the axiomatization **RML** uses individual refinements $[\succeq]_a$ as primitives and not $[\succeq]$. As both logics are equally expressive as the base (multi-agent) modal logic, at some level there is a correspondence but this is not very interesting.

The axiomatization **AAUML** provides us with an alternative axiomatization for RML, for the language with $[\succeq]$ as the unique update modality. Given the system **AAUML** of Table 2, remove axioms **U1** — **U4**, and replace in the axioms **A1** — **A4** $\langle\uparrow\rangle$ by $\langle\succeq\rangle$ and call the result **A1**$^\succeq$ — **A4**$^\succeq$. The resulting proof system is an alternative axiomatization for refinement modal logic. Let us call it **RMLalt**. It is displayed in Table 4. The correspondence between **RKProp** and **A1**$^\succeq$ is trivial. Other differences may be considered of interest. For example, **RMLalt** contains **RE** (replacement of equivalents), whereas **RML** contains **NecR** (necessitation for the refinement quantifier — we recall that necessitation for the arbitrary update quantifier is indeed derivable using **RE**).

Prop	all tautologies of propositional logic	
K	$\Box_a(\varphi \to \psi) \to (\Box_a\varphi \to \Box_a\psi)$	
A1$^\succeq$	$\langle\succeq\rangle\varphi_0 \leftrightarrow \varphi_0$	where $\varphi_0 \in \mathcal{L}_{pl}$
A2$^\succeq$	$\langle\succeq\rangle(\varphi \vee \psi) \leftrightarrow (\langle\succeq\rangle\varphi \vee \langle\succeq\rangle\psi)$	
A3$^\succeq$	$\langle\succeq\rangle(\varphi_0 \wedge \varphi) \leftrightarrow (\varphi_0 \wedge \langle\succeq\rangle\varphi)$	where $\varphi_0 \in \mathcal{L}_{pl}$
A4$^\succeq$	$\langle\succeq\rangle \bigwedge_{a\in A}(\bigwedge \Diamond_a\Phi_a \wedge \Box_a\psi_a) \leftrightarrow \bigwedge_{a\in A} \bigwedge \Diamond_a \langle\succeq\rangle(\Phi_a \wedge \psi_a)$	
MP	from $\varphi \to \psi$ and φ infer ψ	
NecK	from φ infer $\Box_a\varphi$	
RE	from $\chi \leftrightarrow \psi$ infer $\varphi[\chi/p] \leftrightarrow \varphi[\psi/p]$	

Table 4: The alternative axiomatization **RMLalt** of RML

8 Conclusions and further research

Conclusions We presented *arbitrary arrow update model logic* (AAUML). We provided an axiomatization of AAUML, which also demonstrates that AAUML is decidable and equally expressive as multi-agent modal logic. We established arrow update model synthesis for AAUML. We determined the update expressivity hierarchy including AAUML and many other update logics, including other arrow update logics, action model logics, and refinement modal logic. We also provided a novel axiomatization for refinement modal logic.

Further research: B-restricted arrow update synthesis Let B be any subset of the set of all agents. Building upon the B-refinements of [8] and motivated by a similar approach used in [12], a variant of the synthesis problem for AAUML is to consider *B-restricted* arrow update models. Roughly speaking, a B-restricted arrow update model represents an event where only the agents in B can gain more factual information, while the agents outside B remain at least as uncertain as they were before the event. The B-restricted synthesis problem can be solved in a very similar way to the unrestricted problem that we presented in this paper.

Similarly to how arrow update models have the same update expressivity as action models and refinements, B-restricted arrow update models have the same update expressivity as B-restricted action models, and B-refinements have larger update expressivity.

Formally introducing B-restricted arrow update models, and showing that the results apply there as well, would require a lot of complicate notation and several complex definitions. So for the sake of readability we did not include them in this paper.

Further research: complexity of synthesis We have shown that it is possible to perform synthesis for AAUML, and described an algorithm that does this synthesis. We have not, however, discussed the *computational complexity* of that algorithm. In fact, it is non-elementary. We suspect that this is unavoidable, i.e., that the difficulty of the synthesis problem is non-elementary. We do not, for now, have a hardness proof, however. For the related problem of B-restricted arrow update model synthesis we do have a hardness proof, and can show that, in the worst case, it takes non-elementary time.

References

[1] C. Areces, R. Fervari, and G. Hoffmann. Moving arrows and four model checking results. In *Proc. of 19th WoLLIC*, pages 142–153. Springer, 2012. LNCS 7456.

[2] G. Aucher. Characterizing updates in dynamic epistemic logic. In *Proceedings of Twelfth KR*. AAAI Press, 2010.

[3] G. Aucher, P. Balbiani, L. Fariñas del Cerro, and A. Herzig. Global and local graph modifiers. *Electr. Notes Theor. Comput. Sci.*, 231:293–307, 2009.

[4] P. Balbiani, A. Baltag, H. van Ditmarsch, A. Herzig, T. Hoshi, and T. De Lima. 'Knowable' as 'known after an announcement'. *Review of Symbolic Logic*, 1(3):305–334, 2008.

[5] A. Baltag, L.S. Moss, and S. Solecki. The logic of public announcements, common knowledge, and private suspicions. In *Proc. of 7th TARK*, pages 43–56. Morgan Kaufmann, 1998.

[6] P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*. Elsevier, 2006.

[7] T. Bolander, H. van Ditmarsch, A. Herzig, E. Lorini, P. Pardo, and F. Schwarzentruber. Announcements to attentive agents. *Journal of Logic, Language and Information*, 25(1):1–35, 2016.

[8] L. Bozzelli, H. van Ditmarsch, T. French, J. Hales, and S. Pinchinat. Refinement modal logic. *Information and Computation*, 239:303–339, 2014.

[9] T. French and H. van Ditmarsch. Undecidability for arbitrary public announcement logic. In C. Areces and R. Goldblatt, editors, *Advances in Modal Logic 7*, pages 23–42, London, 2008. College Publications.

[10] J.D. Gerbrandy. *Bisimulations on Planet Kripke*. PhD thesis, University of Amsterdam, 1999. ILLC Dissertation Series DS-1999-01.

[11] J.D. Gerbrandy and W. Groeneveld. Reasoning about information change. *Journal of Logic, Language, and Information*, 6:147–169, 1997.

[12] J. Hales. Arbitrary action model logic and action model synthesis. In *Proc. of 28th LICS*, pages 253–262. IEEE, 2013.

[13] B. Kooi. *Knowledge, Chance, and Change*. PhD thesis, University of Groningen, 2003. ILLC Dissertation Series DS-2003-01.

[14] B. Kooi. Expressivity and completeness for public update logics via reduction axioms. *Journal of Applied Non-Classical Logics*, 17(2):231–254, 2007.

[15] B. Kooi and B. Renne. Arrow update logic. *Review of Symbolic Logic*, 4(4):536–559, 2011.

[16] B. Kooi and B. Renne. Generalized arrow update logic. In *Proc. of 13th TARK*, pages 205–211, 2011. Poster presentation.

[17] L.S. Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96(1-3):277–317, 1999.

[18] J.A. Plaza. Logics of public communications. In *Proc. of the 4th ISMIS*, pages 201–216. Oak Ridge National Laboratory, 1989.

[19] D. Steiner. A system for consistency preserving belief change. In *Proc. of the ESSLLI Workshop on Rationality and Knowledge*, pages 133–144, 2006.

[20] J. van Benthem. An essay on sabotage and obstruction. In *Mechanizing Mathematical Reasoning*, volume 2605 of *LNCS 2605*, pages 268–276. Springer, 2005.

[21] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.

[22] H. van Ditmarsch. *Knowledge games.* PhD thesis, University of Groningen, 2000. ILLC Dissertation Series DS-2000-06.

[23] H. van Ditmarsch. Dynamics of lying. *Synthese*, 191(5):745–777, 2014.

[24] H. van Ditmarsch, J.Y. Halpern, W. van der Hoek, and B. Kooi, editors. *Handbook of epistemic logic.* College Publications, 2015.

[25] H. van Ditmarsch and B. Kooi. Semantic results for ontic and epistemic change. In *Proc. of 7th LOFT*, Texts in Logic and Games 3, pages 87–117. Amsterdam University Press, 2008.

[26] H. van Ditmarsch, W. van der Hoek, and B. Kooi. Dynamic epistemic logic with assignment. In *Proc. of 4th AAMAS*, pages 141–148. ACM, 2005.

[27] H. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library.* Springer, 2007.

[28] H. van Ditmarsch, W. van der Hoek, B. Kooi, and L.B. Kuijer. Arbitrary arrow update logic. *Artif. Intell.*, 242:80–106, 2017.

[29] H. van Ditmarsch, W. van der Hoek, and L.B. Kuijer. The undecidability of arbitrary arrow update logic. *Theor. Comput. Sci.*, 693:1–12, 2017.

[30] H. van Ditmarsch, J. van Eijck, F. Sietsma, and Y. Wang. On the logic of lying. In *Games, Actions and Social Software*, LNCS 7010, pages 41–72. Springer, 2012.

[31] J. van Eijck, J. Ruan, and T. Sadzik. Action emulation. *Synthese*, 185(1):131–151, 2012.

[32] J. van Eijck, F. Sietsma, and Y. Wang. Composing models. *Journal of Applied Non-Classical Logics*, 21(3-4):397–425, 2011.