Logics for Epistemic Programs

Alexandru Baltag
Oxford University
Computing Laboratory
Oxford, OX1 3QD, UK
baltag@comlab.ox.ac.uk

Lawrence S. Moss
Mathematics Department
Indiana University
Bloomington, IN 47405 USA
lsm@cs.indiana.edu

Abstract

We construct logical languages which allow one to represent a variety of possible types of changes affecting the information states of agents in a multi-agent setting. We formalize these changes by defining a notion of *epistemic program.* The languages are two-sorted sets that contain not only sentences but also actions or programs. This is as in dynamic logic, and indeed our languages are not significantly more complicated than dynamic logics. But the semantics is more complicated. In general, the semantics of an epistemic program is what we call a *program model.* This is a Kripke model of 'actions', representing the agents' *uncertainty about the current action.* in a similar way that Kripke models of 'states' are commonly used in epistemic logic to represent the agents' *uncertainty about the current state* of the system. Program models induce changes affecting agents' information, which we represent as changes of the state model, called *epistemic updates.* Formally, an update consists of two operations: the first is called the update map, and it takes every state model to another state model, called the *updated model*; the second gives, for each input state model, a transition relation between the states of that model and the states of the updated model.

Each variety of epistemic actions, such as public announcements or completely private announcements to groups, gives what we call an *action signature*, and then each family of action signatures gives a logical language. The construction of these languages is the main topic of this paper. We also mention the systems that capture the valid sentences of our logics. But we defer to a separate paper the completeness proof.

The basic operation used in the semantics is called the *update product.* A version of this was introduced in [6], and the presentation here improves on the earlier one. The update product is used to obtain from any program model the corresponding epistemic update, thus allowing us to *compute* changes of information or belief. This point is of interest independently of our logical languages. We illustrate the update product and our logical languages with many examples throughout the paper.

1 Introduction

Traditional epistemic puzzles often deal with *changes of knowledge* that come about in various ways. Perhaps the most popular examples are the puzzles revolving around the fact that a declaration of ignorance of some sentence A may well lead to knowledge of A. We have in mind the scenarios that go by names such as the Muddy Children, the Cheating Spouses, the Three Wisemen, and the like. The standard treatment of these matters (a) introduces the Kripke semantics of modal logic so as to formalize the informal notions of *knowledge* and *common knowledge*; (b) formalizes one of the scenarios as a particular model; (c) and finally shows how the formalized notions of knowledge and common knowledge illuminate some key aspects of the overall scenario. The informal notion of knowledge which is closest to what is captured in the traditional semantics is probably *justified true belief*. But more generally, one can consider *justifiable beliefs*, regardless of whether or not they happen to be true or not; in many contexts, agents may be deceived by certain actions, without necessarily losing their rationality. Thus, such beliefs, and the *justifiable changes affecting these beliefs*, may be accepted as a proper subject for logical investigation. The successful treatment of a host of puzzles leads naturally to the following

Thesis I Let s be a social situation involving the intuitive concepts of knowledge, justifiable beliefs and common knowledge among a group of agents. Assume that s is presented in such a way that all the relevant features of s pertaining to knowledge, beliefs and common knowledge are completely determined. Then we may associate to s a mathematical model $\mathbf{S}$. ($\mathbf{S}$ is a multi-agent Kripke model; we call these *epistemic state models*.) The point of the association is that all intuitive judgements concerning s correspond to formal assertions concerning $\mathbf{S}$, and vice-versa.

We are not aware of any previous formulations of this thesis. Nevertheless, some version of this thesis is probably responsible for the appeal of epistemic logic.

We shall not be concerned in this paper with a defense of this thesis, but instead we return to our opening point related to *change*. Dynamic epistemic logic, dynamic doxastic logic, and related formalisms, attempt to incorporate *change from model to model* in the syntax and semantics of a logical language. We are especially concerned with changes that result from information-updating actions of various sorts. Our overall aim is to formally represent *epistemic actions*, and we associate to each of them a corresponding *update*. By "updates" we shall mean operations defined on the space of all state models, operations which are meant to represent well-defined, systematic changes in the information states of all agents. By an "epistemic action" (or program) we shall mean a representation of the way such a change "looks" to each agent.

Perhaps the paradigm case of an epistemic action is a public announcement. The first goal is to say in a general way what the effect of a (public) announcement should be on a model. It is natural to model such announcements by the logical notion of *relativization*: publicly announcing a sentence causes all agents to restrict attention to the worlds where the sentence was true (before the announcement). Note that the informal notion of announcement takes situations to situations, and the formal notion of relativization is an operation taking models to models.

In this paper, we wish to consider many types of epistemic actions that are more difficult to handle than public announcements. These include *half-transparent, half-opaque* types of actions, such as announcements to groups in a completely private way, announcements that include the possibility that outsiders suspect the announcement but this suspicion is lost on the insiders, private announcements which are secretely intercepted by outsiders etc. We may also consider types of actions exhibiting information-loss and misinformation, where agents are deceived by others or by themselves.

Thesis II Let σ be a *social "action"* involving and affecting the knowledge (beliefs, common knowledge) of agents. This naturally induces a *change of situation*; i.e., an operation o taking situations s into situations $o(s)$. Assume that o is presented by assertions concerning knowledge, beliefs and common knowledge facts about s and $o(s)$, and that o is completely determined by these assertions. Then

(a) We may associate to the action σ a mathematical model $\mathbf{\Sigma}$ which we call an *epistemic action model.* ($\mathbf{\Sigma}$ is also a multi-agent Kripke model.) The point again is that all the intuitive features of, and judgments about, σ correspond to formal properties of $\mathbf{\Sigma}$.

(b) There is an operation $\otimes$ taking a state model $\mathbf{S}$ and an action model $\mathbf{\Sigma}$ and returning a new state model $\mathbf{S} \otimes \mathbf{\Sigma}$. So each $\mathbf{\Sigma}$ induces an *update operation* O on state models: $O(\mathbf{S}) = \mathbf{S} \otimes \mathbf{\Sigma}$.

(c) The update O is a faithful model of the situation change o, in the sense that for all s: if s corresponds to $\mathbf{S}$ as in Thesis I, then again $o(s)$ corresponds to $O(\mathbf{S})$ in the same way; i.e. all intuitive judgements concerning $o(s)$ correspond to formal assertions concerning $O(\mathbf{S})$, and vice-versa.

Our aim in this paper is not to offer a full conceptual defense of these two theses. Instead, we will justify the intuitions behind them through examples and usage. We shall use them to build logical languages and models and show how these can be applied to the analysis of natural examples of "social situations" and "social actions". As in the case of standard possible-worlds semantics (for which a 'strong', ontological defense is hard, maybe even impossible, to give), the usefulness of these formal developments may provide a 'weak', implicit defense of the philosophical claims underlying our semantics.

Our method of defining updates is quite general and leads to *logics of epistemic programs*, extending standard systems of epistemic logic by adding *updates* as new operators. These logical languages also incorporate features of propositional dynamic logic. Special cases of our logic, dealing only with public or semi-public *announcements to mutually isolated groups*, have been considered in Plaza [18], Gerbrandy [10, 11], and Gerbrandy and Groeneveld [12]. But our overall setting is much more liberal, since it allows for all the above-mentioned types of actions. We feel it would be interesting to study further examples with an eye towards applications, but we leave this to other papers.

In our logical systems, we capture only the *epistemic aspect* of these real actions, disregarding other (intentional) aspects. In particular, to keep things simple we only deal with "'purely epistemic" actions; i.e., the ones that do not change the facts of the world, but affect only the agents' beliefs about the world. However, this is *not* an essential limitation, as our formal setting can be easily adapted to express fact-changing actions.

On the semantic side, the main original technical contribution of our paper lies in our decision to represent not only the epistemic states, but also (for the first time) the epistemic actions. For this, we use *action models*, which are epistemic Kripke models of "actions", similar to the standard Kripke structures of "states". While for states, these structures represent in the usual way the uncertainty of each agent concerning the current state of the system, we similarly use *action signatures* to represent the uncertainty of each agent concerning the current action taking place. For example, there will be a single action signature that represents public announcements. There will be a different action signature representing a completely private announcement to one specified agent, etc. The intuition is that we are dealing with potentially "half-opaque/half-transparent" actions, about which the agents may be incompletely informed, or even completely misinformed. The components ("possible worlds") of an action model are called *"simple" actions*, since they are meant to represent particularly simple kinds of actions, whose epistemic impact is *uniform* on states: the informational features of a simple action are intrinsic to the action, and thus are independent of the informational features of the states to which it can be applied. This independence is subject to only one restriction: the action's presuppositions or conditions of possibility, which a state must satisfy in order for the action to be executable. Thus, besides the epistemic structure, simple actions have *preconditions*, defining their domain of applicability: not every action is possible in every state. We model the update of a state by an action as a partial *update* operation, given by a restricted product of the two structures: the uncertainties present in the given state and the given action are multiplied, while the "impossible" combinations of states and actions are eliminated (by testing the actions' preconditions on the state). The underlying intuition is that, since the agent's uncertainties concerning the state and the ones concerning the simple action are mutually independent, the two uncertainties must be multiplied, except that we insist on applying an action only to those states which satisfy its precondition.

On the syntactic side, we use a mixture of dynamic and epistemic logic, with dynamic modalities associated to each action signature, and with common-knowledge modalities for various groups of agents (in addition to the usual individual-knowledge operators). In this paper, we present a sound system for this logic. The logic includes an Action-Knowledge Axiom that generalizes similar axioms found in other papers in the area; cf. [10, 11, 18]. The main original features of our system is an inference rule which we call the Action Rule. This allows one to infer sentences expressing common knowledge facts which hold after an epistemic action. From another point of view, the Action Rule expresses what might be called a notion of "epistemic (co)recursion". Overall, the Action-Knowledge Axiom and the Action Rule express fundamental formal features of the interaction between action and knowledge in multi-agent systems. The logic is studied further in our paper with S. Solecki [6]. There we present the completeness and decidability of the logic, and we prove various expressivity results.

For impatient readers The main logical systems of the paper are presented in Section 4.2, and to read that one would only have to read the definition in Section 4.1 first. To understand the semantics, one should read in addition Sections 2.1, 2.3, and 3.1–3.4. But we know that our systems would not be of any interest if the motivation were not so great. For this reason, we have included many examples and discussions, particularly in the sections of the paper preceding the introduction of the logics. Readers may read as much or as little of that material as they desire. Indeed, some readers may find our examples and discussion of more interest than the logical systems themselves. The main logical systems are presented in Section 5.

Technical results concerning our systems will appear in other papers. The completeness/decidability result for the main systems of this paper will appear in a paper [7] written with Sławomir Solecki; this paper also contains results on the expressive power of our systems. For stronger systems of interest, there are *undecidability* results; cf. [16].

Acknowledgements

We began work on this topic in 1998, and this set of papers was completed in 2003. So the work has gone through numerous editions (as have we). We have found the presentation of the work to be thorny: there are many details that must be done correctly, there are many options for the presentation. We fear that the mathematically easiest way to present the work would not be understandable to most readers. Since we really want to have results that are not only true but usable, we went "back to the drawing board" several times.

We are therefore grateful to many people for discussions at various points, and we express our gratitude knowing that most of them have not even seen the end product. We especially thank Jelle Gerbrandy, Rohit Parikh, and Johan van Benthem for extremely useful conversations. We also thank Balder ten Cate, Hans van Ditmarsch, Melvin Fitting, Joseph Miller, Jan Rutten, Yde Venema, and an anonymous referee who feels the work is "somewhat in bad taste" for their comments, questions, and advice. We have learned much from presenting this work in seminars, and we thank the Indiana University Logic Group, the CUNY Applied Logic Seminar, and students in a courses at the 2002 and 2003 North American Summer School in Logic, Language and Information.

A. Baltag thanks the Dutch National Institute for Mathematics and Computer Science (CWI, Amsterdam) and the Institute for Logic, Language and Information (ILLC) at the University of Amsterdam for their support. L. Moss thanks the Johns Hopkins University Cognitive Science Department and the University of Maryland Institute for Advanced Computer Studies for support as well.

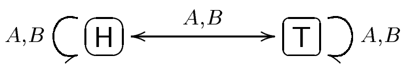
1.1 Scenarios

Probably the best way to enter our our overall subject is to consider some "epistemic scenarios." These give the reader some idea of what the general subject is about, and they also provide us with test problems at different points.

SCENARIO 1: THE CONCEALED COIN. A and B enter a large room containing a remote-control mechanical coin flipper. One presses a button, and the coin spins through the air, landing in a small box on a table. The box closes. The two people are much too far to see the coin.

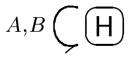
The main contents of any representation of the relevant knowledge states of A and B are that (a) there are two alternatives, heads and tails; (b) neither party knows which alternative holds; and (c) that (a) and (b) are common knowledge. The need for the notion of common knowledge in reasoning about multi-agent interaction is by now standard in applied epistemic logic, and so we take it as unproblematic that one would want (c) to come out in any representation of this scenario.

Perhaps the clearest way to represent this scenario is with a diagram:

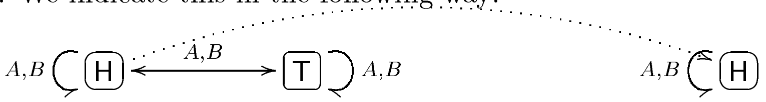


In more standard terms, we have a *set of two alternatives*, call them x and y. We also have some *atomic information*, that x represents the possible fact of the coin is lying heads up and that y represents the other possible fact. Finally, we also have some extra apparatus needed to indicate that, no matter the actual disposition of the coin, A and B think both alternatives are possible. Following the standard practice in epistemic logic, we take this apparatus to be *accessibility relations* between the alternatives. The diagram should be read as saying that were the actual state of the world to be x, say, then A and B would still entertain *both* alternatives.

SCENARIO 2: THE COIN REVEALED TO SHOW HEADS. A and B sit down. One opens the box and puts the coin on the table for both to see. It's heads. The result of this scenario is a model which again is easy to grasp. It consists of one state; call it s. Each agent *knows* the state in the sense that they think s is the only possible state.

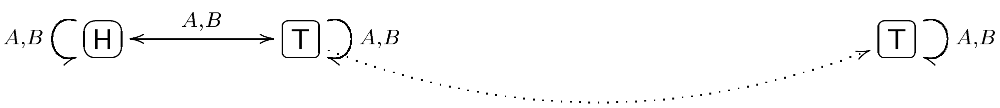


We also shall be interested in keeping track of the relation of the model in Scenario 1 and the model just above. We indicate this in the following way:

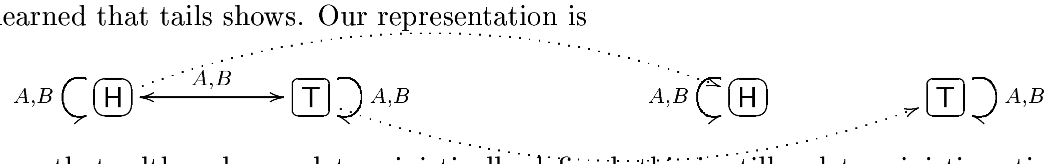


The first thing to note is that the dotted connection is a *partial function*; as we shall see later, this is the hallmark of a *deterministic* epistemic action. But we also will see quite soon that the before-after relation is not always a partial function; so there are epistemic actions which are not deterministic. Another thing to note at this point is that the dotted connection is *not* subscripted with an agent or a set of agents. It does not represent alternative possibility with respect to anyone, but instead stands for the before-after relation between two models: it is a *transition relation*, going from *input states* to the corresponding *output states*. In this example, the transition relation is in fact a partial function whose domain is the set of states which could possibly be subject to the action of revealing heads. This is possible in only one of the two starting states.

SCENARIO 2.1: THE COIN REVEALED TO SHOW TAILS. As a variation on Scenario 2, there is a different Scenario in which the coin is revealed in the same way to both A and B but with the change that tails shows. Our full representation is:

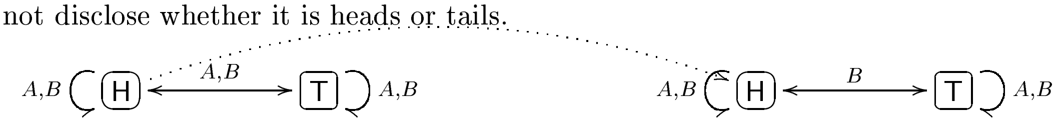


Scenario 2.2: the coin revealed. Finally, we can consider the *non-deterministic sum* of publicly revealing heads and publicly revealing tails. The coin is revealed to both A and B, but all that we as external modelers can say is that either they learned that heads shows, or that they learned that tails shows. Our representation is



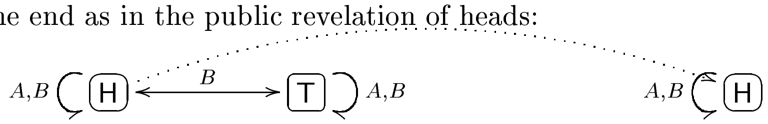
Observe that, although non-deterministically defined, this is still a deterministic action: the relation described by the dotted connection is still a function.

Scenario 3: a semi-private viewing of heads. The following is an alternative to the scenarios above in which there is a public revelation. After Scenario 1, A opens the box herself. The coin is lying heads up. B observes A open the box but does not see the coin. And A also does not disclose whether it is heads or tails.



No matter which alternative holds, B would consider both as possible, and A would be certain which was the case

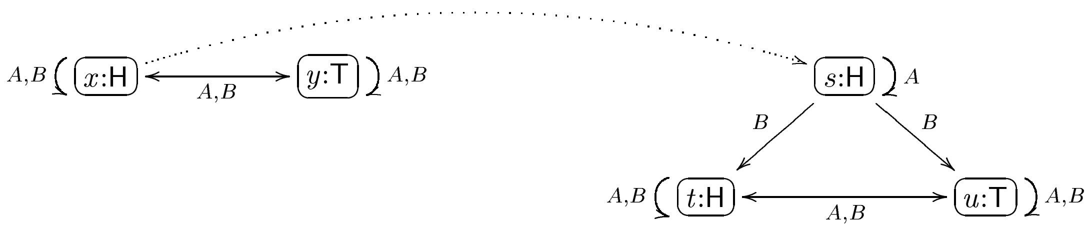
Scenario 3.1: B's turn. After Scenario 3, B takes a turn and opens the box the same way. We expect that after both have individually opened the boxes they see the same things; moreover, they know this will happen. This time, we begin with the end of Scenario 3, and we end with the same end as in the public revelation of heads:



Scenario 4: cheating. After Scenario 1, A secretly opens the box herself. The coin is lying Heads up. B does not observe A open the box, and indeed A is certain that B did not suspect that anything happened after they sat down. This is substantially more difficult conceptually, and the representation is accordingly more involved.

Such cheating is like an announcement that results in A's being sure that the coin lies heads up, and B learns nothing. But the problem is how to model the fact that, after the announcement, B knows nothing (new). We cannot just delete all arrows for B to represent such lack of knowledge: this would actually *increase* B's (false) 'knowledge', by adding to his set of beliefs new ones; for example, he'll believe it is not possible that the coin is lying heads up. *Deleting* arrows always corresponds to *increasing* 'information' (even if sometimes this is just by adding false information). But this seems wrong in our case, since B's possibilities should be unchanged by A's secret cheating. Instead, our representation of the informational change induced by such cheating should be:

(1)

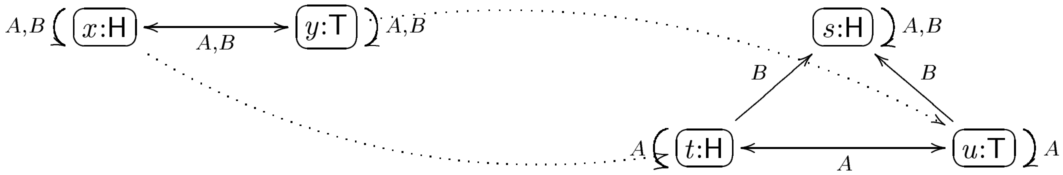


The main point is that after A cheats, B does not think that the actual state of the world is even possible. The states that B thinks are possible should only be two states which are the same as the "before" states, or states which are in some important way similar to those. In addition to the way the state actually is, where A knows the coin lies heads up *or rather as an aspect of that state*, we need to consider other states to represent the possibilities that B must entertain.

We have indicated names s, t, and u of the possibilities in (1). (We could have done this with the scenarios we already considered, but there was less of a reason.) State s has a different status from t and u: while t and u are the states those that B thinks could hold, s is the one that A *knows* to be the case. Note that the substructure determined by t and u is isomorphic to the "before" structure. This is the way that we formalize the intuition that the states available to B after cheating are essentially the same as the states before cheating.

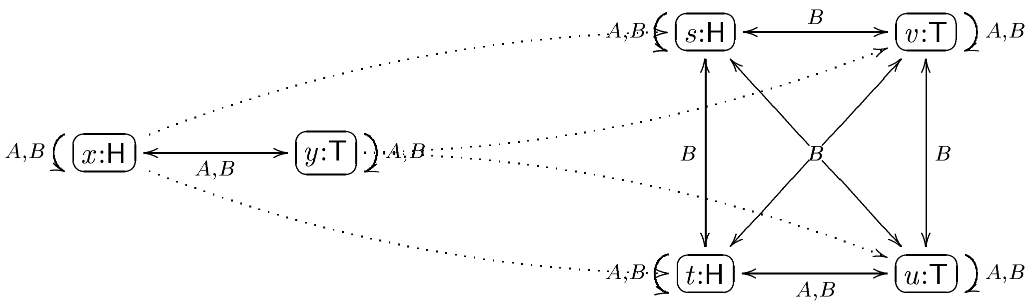
SCENARIO 5: MORE CHEATING. After Scenario 4, B does the very same thing. That is, B opens the box quite secretly. We leave the reader the instructive exercise of working out a representation. We shall return to this matter in Section 3.5, where we solve the problem based on our general machinery. We merely mention now that part of the goal of the paper is precisely to develop tools to build representations of complex epistemic examples such as this one.

SCENARIO 6: LYING. After Scenario 1, A convinces B that the coin lies heads up and that she knows this. In fact, she is lying. Here is our representation:



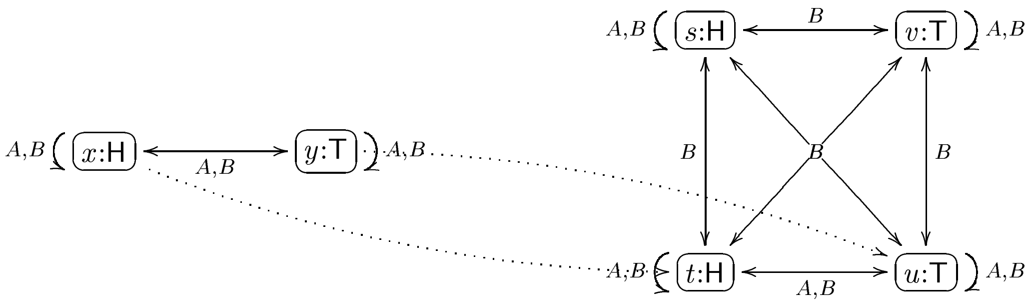
SCENARIO 7: PICK A CARD. As another alternative to Scenario 3, C walks in and tells both A and B at the same time that she has a card which either says H, T, or is blank. In the first two cases the card describes truly the state of the coin in the box, and in the last case the intention

is that no information is given. Then C gives the card to A in the presence of B. Here is our representation:



The idea here is that t and u represent states where the card was blank, s the state where the card showed H, and t the state where the card showed T.

Scenario 8: common knowledge of (unfounded) suspicion. As yet another alternative to Scenario 3, suppose that after A and B make their original entrance, A has not looked, but B has some suspicion concerning A's cheating; so B considers possible that she (A) has secretely opened the box (but B cannot be sure of this, so he also considers possible that nothing happened); moreover, we assume there is common knowledge of this (B's) suspicion. That is, we want to think of one single action (a knowing glance by A, for example) that results in all of this. The representation of "before" and "after" is as follows:



One should compare this with Scenario 7. The blank card there is a parallel to no looking here; card with H is parallel to A's looking and seeing H; and similarly for T. This accounts for the similarity in the models. The main difference is that Scenario 7 was described in such a way that we do not know what the card says; in this scenario we stipulate that A definitely did not look. This accounts for the difference in dotted lines between the two figures.

Scenario 8.1: private communication about the other. After Scenario 8, someone stands in the doorway behind A and raises a finger. This indicates to B that A has not cheated. The communication between B and the third party was completely private. Again, we leave the representation as an exercise to the reader.

One goal of the paper is to propose a theoretical understanding of these representations. It turns out that there are simple operations and general insights which allow one to construct them. One operation allows one to pass, for example, from the representation of Scenario 1 to that of Scenario 4; moreover, the operation is so applicable that it allows us to add private knowledge for one agent in a "private" way to *any* representation.

1.2 Further contents of this paper

Section 2 continues our introduction by both reviewing some of the main background concepts of epistemic logic, and also by situating our logical systems with respect to well-known systems. Section 2.1 presents a new "big picture" of the world of epistemic actions. While the work that we do could be understood without the conceptual part of the big picture, it probably would help the reader to work through our definitions.

Section 3 begins the technical part of the paper in earnest, and here we revisit some of the Scenarios of Section 1.1 and their pictures. The idea is to get a logical system for reasoning with, and about, these kinds of models. Section 4 gives the syntax and semantics of our logical systems. These are studied further in Sections 5. For example, we present sound and complete logical systems (with the proofs of soundness and completeness deferred to [7]).

Endnotes We gather at the ends of our sections some remarks about the literature and how our work fits in with that of others.

We mentioned other work in dynamic epistemic logic and dynamic doxastic logic. Much more on these logics and many other proposals in epistemic logic may be found in Gochet and Gribomont [13], and Meyer and van der Hoek [15]. Also, a survey of many topics the area of information update and communication may be found in van Benthem's papers [20, 21, 22].

The ideas behind several of the scenarios in Section 1.1 are to be found in several places: see, e.g., Plaza [18], Gerbrandy [10, 11], Gerbrandy and Groeneveld [12], and van Ditmarsch [23, 24]. We shall discuss these papers in more detail later. Our scenarios go beyond the work of these papers. Specifically, our treatment of the actions in Scenarios 6 and 8 seems new. Also, our use of the relation between "before" and "after" (given by the dotted arrows) is new.

2 Epistemic updates and our target logics

We fix a set AtSen of *atomic sentences*, and a set $\mathcal{A}$ of *agents*. All of our definitions depend on AtSen and $\mathcal{A}$, but for the most part we omit mention of these.

2.1 State models and epistemic propositions

A *state model* is a triple $\mathbf{S} = (S, \xrightarrow{A}_{\mathbf{S}}, \|\cdot\|_{\mathbf{S}})$ consisting of a set S of "states"; a family of binary accessibility relations $\xrightarrow{A}_{\mathbf{S}} \subseteq S \times S$, one for each agent $A \in \mathcal{A}$; and a "valuation" (or a "truth" map) $\|.\|_{\mathbf{S}} : \mathsf{AtSen} \to \mathcal{P}(S)$, assigning to each atomic sentence p a set $\|p\|_{\mathbf{S}}$ of states. When dealing with a single fixed state model $\mathbf{S}$, we often drop the subscript $\mathbf{S}$ from all the notation.

In a state model, atomic sentences are supposed to represent non-epistemic, "objective" *facts* of the world, which can be thought of as properties of states; the valuation tells us which facts hold at which states. The accessibility relations model the agents' epistemic uncertainty about the current state. That is, to say that $s \xrightarrow{A} t$ in $\mathbf{S}$ means that in the model, in state s, agent A considers it possible that the state is t.

Definition Let SMod be the collection of all state models. An *epistemic proposition* is an operation $\boldsymbol{\varphi}$ defined on SMod such that for all $\mathbf{S} \in \mathsf{SMod}$, $\boldsymbol{\varphi}_{\mathbf{S}} \subseteq S$.

The collection of epistemic propositions is closed in various ways.

1. For each atomic sentence p we have an atomic proposition $\boldsymbol{p}$ with $\boldsymbol{p}_{\mathbf{S}} = \|p\|_{\mathbf{S}}$.[1]
2. If $\boldsymbol{\varphi}$ is an epistemic proposition, then so is $\neg\boldsymbol{\varphi}$, where $(\neg\boldsymbol{\varphi})_{\mathbf{S}} = S \setminus \boldsymbol{\varphi}_{\mathbf{S}}$.
3. If C is a set or class of epistemic propositions, then so is $\bigwedge C$, with
$$(\bigwedge C)_{\mathbf{S}} \quad = \quad \bigcap\{\boldsymbol{\varphi}_{\mathbf{S}} : \boldsymbol{\varphi} \in C\}.$$
4. Taking C above to be empty, we have an *"always true"* epistemic proposition $\mathbf{tr}$, with $\mathbf{tr}_{\mathbf{S}} = S$.
5. We also may take C in part (3) to be a two-element set $\{\boldsymbol{\varphi}, \boldsymbol{\psi}\}$; here we write $\boldsymbol{\varphi} \wedge \boldsymbol{\psi}$ instead of $\bigwedge\{\boldsymbol{\varphi}, \boldsymbol{\psi}\}$. We see that if $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ are epistemic propositions, then so is $\boldsymbol{\varphi} \wedge \boldsymbol{\psi}$, with $(\boldsymbol{\varphi} \wedge \boldsymbol{\psi})_{\mathbf{S}} = \boldsymbol{\varphi}_{\mathbf{S}} \cap \boldsymbol{\psi}_{\mathbf{S}}$.
6. If $\boldsymbol{\varphi}$ is an epistemic proposition and $A \in \mathcal{A}$, then $\Box_A\boldsymbol{\varphi}$ is an epistemic proposition, with
$$(\Box_A\boldsymbol{\varphi})_{\mathbf{S}} \quad = \quad \{s \in S : \text{if } s \xrightarrow{A} t, \text{ then } t \in \boldsymbol{\varphi}_{\mathbf{S}}\}. \tag{2}$$
7. If $\boldsymbol{\varphi}$ is an epistemic proposition and $\mathcal{B} \subseteq \mathcal{A}$, then $\Box^*_{\mathcal{B}}\boldsymbol{\varphi}$ is an epistemic proposition, with
$$(\Box^*_{\mathcal{B}}\boldsymbol{\varphi})_{\mathbf{S}} \quad = \quad \{s \in S : \text{if } s \xrightarrow{\mathcal{B}^*} t, \text{ then } t \in \boldsymbol{\varphi}_{\mathbf{S}}\}.$$
Here $s \xrightarrow{\mathcal{B}^*} t$ iff there is a sequence
$$s = u_0 \quad \to_{A_1} \quad u_1 \quad \to_{A_2} \quad \cdots \to_{A_n} \quad u_{n+1} = t$$
where $A_1, \ldots, A_n \in \mathcal{B}$. In other words, there is a sequence of arrows from the set $\mathcal{B}$ taking s to t. We allow $n = 0$ here, so $\xrightarrow{\mathcal{B}^*}$ includes the identity relation on S. To see that $\Box^*_{\mathcal{B}}\boldsymbol{\varphi}$ is indeed an epistemic proposition, we use parts 3 and 6 above; we may also argue directly, of course.

2.2 Epistemic logic recalled

In this section, we review the basic definitions of modal logic. In terms of our work in Section 2.1 above, the *infinitary modal propositions* are the smallest collection of epistemic propositions containing the propositions $\boldsymbol{p}$ corresponding to atomic sentences p and closed under negation $\neg$, infinitary conjunction $\bigwedge$, and the agent modalities $\Box_A$. The *finitary* propositions are the smallest collection closed the same way, except that we replace $\bigwedge$ by its special cases $\mathbf{tr}$ and the binary conjunction operation.

[1]It is important for us that the *sentence* p be a syntactic object, while the *proposition* $\boldsymbol{p}$ be a semantic object. See Section 2.5 for further discussion.

Syntactic and semantic notions It will be important for us to make a sharp distinction between syntactic and semantic notions. For example, we speak of atomic *sentences* and atomic *propositions*. The difference for us is that atomic sentences are entirely syntactic objects: we won't treat an atomic sentence p as anything except an unanalyzed mathematical object. On the other hand, this atomic sentence p also has associated with it the atomic proposition $\boldsymbol{p}$. For us $\boldsymbol{p}$ will be a function whose domain is the (proper class of) state models, and it is defined by

$$\boldsymbol{p}_{\mathbf{S}} \quad = \quad \{s \in S : s \in \|p\|_{\mathbf{S}}\}. \tag{3}$$

This difference may seem pedantic at first, and surely there are times when it is sensible to blur it. But for various reasons that will hopefully become clear, we need to insist on it.

Up until now, the only syntactic objects have been the atomic sentences $p \in \mathsf{AtSen}$. But we can build the collections of *finitary* and *infinitary atomic sentences* by the same definitions that we have seen, and then the work of the past section is the *semantics* of our logical systems. For example, we have sentences $p \wedge q$, $\Box_A \neg p$, and $\Box^*_{\mathcal{B}} q$. These then have corresponding epistemic propositions as their semantics: $\boldsymbol{p} \wedge \boldsymbol{q}$, $\Box_A \neg \boldsymbol{p}$, and $\Box^*_{\mathcal{B}} \boldsymbol{q}$, respectively. Note that the latter is a properly infinitary proposition (and so $\Box^*_{\mathcal{B}} q$ is a properly infinitary sentence); it abbreviates an infinite conjunction.

The rest of this section studies examples of the semantics and it also makes the connection of the formal system with the informal notions of *knowledge*, *belief* and *common knowledge*. We shall study Scenario 3 of Section 1.1, where A opens the box herself to see heads in a semi-public way: B sees A open the box but not the result, A is aware of this, etc. We want to study the model after the opening. We represented this as

$$\begin{tikzcd} \boxed{\mathsf{H}} \arrow[loop left, "A,B"] \arrow[r, leftrightarrow, "B"] & \boxed{\mathsf{T}} \arrow[loop right, "A,B"] \end{tikzcd}$$

We first must represent this picture as a bona fide state model $\mathbf{S}_3$ in our sense[2]. The picture includes no explicit states, but we must fix some states to have a representation. We choose distinct objects s and t. Then we take as our state model $\mathbf{S}_3$, as defined by

$$\begin{array}{lcl@{\qquad\qquad}lcl}
S_3 & = & \{s,t\} & & & \\
\xrightarrow{A} & = & \{(s,s),(t,t)\} & \|\mathsf{H}\| & = & \{s\} \\
\xrightarrow{B} & = & \{(s,s),(s,t),(t,s),(t,t)\} & \|\mathsf{T}\| & = & \{t\}
\end{array}$$

In Figure 1, we list some sentences of English along with their translations into standard epistemic logic. We also have calculated the semantics of the formal sentences in the model $\mathbf{S}_3$. It should be stressed that the semantics is exactly the one defined in the previous section. For example,

$$\begin{array}{lcl}
\|\Box_A \mathsf{T}\|_{\mathbf{S}_3} & = & \{u \in S_3 : \text{if } u \xrightarrow{A} v, \text{ then } v \in \|\mathsf{T}\|_{\mathbf{S}_3}\} \\
& = & \{u \in S_3 : \text{if } u \xrightarrow{A} v, \text{ then } v = t\} \\
& = & \{t\}
\end{array}$$

We also emphasize two other points. First, the translation of English to the formal system is based on the rendering of "A knows" (or "A has a justified belief that") as $\Box_A$, and of "it is

[2]The subscript 3 comes from the number of the scenario; we shall speak of corresponding models $\mathbf{S}_1$, $\mathbf{S}_2$, etc., and each time the models will be the ones pictured in Section 1.1.

English	Formal rendering	Semantics
the coin shows heads	H	$\{t\}$
A knows the coin shows heads	$\Box_A \mathsf{H}$	$\{s\}$
A knows the coin shows tails	$\Box_A \mathsf{T}$	$\{t\}$
B knows that the coin shows head	$\Box_B \mathsf{H}$	$\emptyset$
A knows that B doesn't know it's heads	$\Box_A \neg \Box_B \mathsf{H}$	$\{s, t\}$
B knows that A knows that B doesn't know it's heads	$\Box_B \Box_A \neg \Box_B \mathsf{H}$	$\{s, t\}$
it is common knowledge that either A knows it's heads or A knows that it's tails	$\Box^*_{A,B}(\Box_A \mathsf{H} \vee \Box_A \mathsf{T})$	$\{s, t\}$
it is common knowledge that B doesn't know the state of the coin	$\Box^*_{A,B} \neg (\Box_B \mathsf{H} \vee \Box_B \mathsf{T})$	$\{s, t\}$

Figure 1: Examples of translations and semantics

common knowledge that" by $\Box^*_{A,B}$. Second, the chart bears a relation to intuitions that one naturally has about the states s and t. Recall that s is the state that obtains after A looks and sees that the coin is lying heads up. The state t, on the other hand, is a state that would have been the case had A seen tails when she looked.

2.3 Updates

A *transition relation* between state models $\mathbf{S}$ and $\mathbf{T}$ is a relation between the sets S and T. We write $r : \mathbf{S} \to \mathbf{T}$ for this. An *update* $\mathbf{r}$ is a pair of operations

$$\mathbf{r} \quad = \quad (\mathbf{S} \mapsto \mathbf{S}(\mathbf{r}), \mathbf{S} \mapsto \mathbf{r_S}),$$

where for each $\mathbf{S} \in \mathsf{SMod}$, $\mathbf{r_S} : \mathbf{S} \to \mathbf{S}(\mathbf{r})$ is a transition relation. We call $\mathbf{S} \mapsto \mathbf{S}(\mathbf{r})$ the *update map*, and $\mathbf{S} \mapsto \mathbf{r_S}$ the *update relation*.

Example 2.1 Let $\boldsymbol{\varphi}$ be an epistemic proposition. We get an update $Pub\ \boldsymbol{\varphi}$ which represents *the public announcement of* $\boldsymbol{\varphi}$. For each $\mathbf{S}$, $\mathbf{S}(Pub\ \boldsymbol{\varphi})$ is the sub-state-model of $\mathbf{S}$ determined by the states in $\boldsymbol{\varphi}_\mathbf{S}$. In this submodel, information about atomic sentences and accessibility relations is simply inherited from the larger model. The update relation $(Pub\ \boldsymbol{\varphi})_\mathbf{S}$ is the inverse of the inclusion relation of $\mathbf{S}(Pub\ \boldsymbol{\varphi})$ into $\mathbf{S}$.

Example 2.2 We also get a different update $?\boldsymbol{\varphi}$ which represents *testing whether φ is true.* Here we take $\mathbf{S}(?\boldsymbol{\varphi})$ to be the model whose state set is

$$(\{0\} \times \boldsymbol{\varphi}_\mathbf{S}) \ \cup \ (\{1\} \times \mathbf{S}).$$

The arrows are defined by $(i, s) \xrightarrow{A} (j, t)$ iff $s \xrightarrow{A} t$ in $\mathbf{S}$ and $j = 1$. (Note that states of the form $(0, s)$ are never the targets of arrows in the new model.) Finally, we set

$$\|p\|_{\mathbf{S}(?\boldsymbol{\varphi})} \quad = \quad \{(i, s) \in \mathbf{S}(?\boldsymbol{\varphi}) : s \in \|p\|_\mathbf{S}\}.$$

The relation $(?\boldsymbol{\varphi})_\mathbf{S}$ is the set of pairs $(s, (0, s))$ such that $s \in \boldsymbol{\varphi}_\mathbf{S}$.

We shall study these two examples (and many others) in the sequel, and in particular we shall justify the names "public announcement" and "test". For now, we continue our general discussion by noting that the collection of updates is closed in various ways.

1. *Skip*: there is an update $\mathbf{1}$ with $\mathbf{S}(\mathbf{1}) = \mathbf{S}$, and $\mathbf{1}_{\mathbf{S}}$ is the identity relation on $\mathbf{S}$.

2. *Sequential Composition*: if $\mathbf{r}$ and $\mathbf{s}$ are epistemic updates, then their composition $\mathbf{r};\mathbf{s}$ is again an epistemic update, where $\mathbf{S}(\mathbf{r};\mathbf{s}) = \mathbf{S}(\mathbf{r})(\mathbf{s})$, and $(\mathbf{r};\mathbf{s})_{\mathbf{S}} = \mathbf{r}_{\mathbf{S}};\mathbf{s}_{\mathbf{S}(\mathbf{r})}$. Here, we use on the right side the usual composition ; of relations[3].

3. *Disjoint Union* (or *Non-deterministic choice*): If X is any set of epistemic updates, then the disjoint union $\bigsqcup_X \mathbf{r}$ is an epistemic update, defined as follows. The set of states of the model $\bigsqcup_X \mathbf{r}$ is the disjoint union of all the sets of states in each model $\mathbf{S}(\mathbf{r})$:

$$\{(s, \mathbf{r}) : \mathbf{r} \in X \text{ and } s \in \mathbf{S}(\mathbf{r})\}.$$

   Similarly, each accessibility relation $\xrightarrow{A}$ is defined as the disjoint union of the corresponding accessibility relations in each model:

$$(t, \mathbf{r}) \xrightarrow{A} (u, \mathbf{s}) \quad \text{iff} \quad \text{if } \mathbf{r} = \mathbf{s} \text{ and } t \xrightarrow{A} u \text{ in } \mathbf{S}(\mathbf{r}).$$

   The valuation $\|p\|$ in $\mathbf{S}(\bigsqcup_X \mathbf{r})$ is the disjoint union of the valuations in each state model:

$$\|p\| \quad = \quad \{(s, \mathbf{r}) : \mathbf{r} \in X \text{ and } s \in \|p\|_{\mathbf{S}(\mathbf{r})}\}.$$

   Finally, the update relation $(\bigsqcup_X \mathbf{r})_{\mathbf{S}}$ between $\mathbf{S}$ and $\mathbf{S}(\bigsqcup_X \mathbf{r})$ is the union of all the update relations $\mathbf{r}_{\mathbf{S}}$:

$$t(\bigsqcup_X \mathbf{r})_{\mathbf{S}} \ (u, \mathbf{s}) \quad \text{iff} \quad t\mathbf{s}_{\mathbf{S}}\ u.$$

4. Special case: *Binary Union*. The (disjoint) union of two epistemic updates $\mathbf{r}$ and $\mathbf{s}$ is an update $\mathbf{r} \sqcup \mathbf{s}$, given by $\mathbf{r} \sqcup \mathbf{s} = \bigsqcup\{\mathbf{r}, \mathbf{s}\}$.

5. Another special case: *Kleene star* (*iteration*). We have the operation of Kleene star on updates:

$$\mathbf{r}^* \quad = \quad \bigsqcup\{\mathbf{1}, \mathbf{r}, \mathbf{r}\cdot\mathbf{r}, \ldots, \mathbf{r}^n, \ldots\}$$

   where $\mathbf{r}^n$ is recursively defined by $\mathbf{r}^0 = \mathbf{1}$, $\mathbf{r}^{n+1} = \mathbf{r}^n;\mathbf{r}$.

6. *Crash*: We can also take $X = \emptyset$ in part 3. This gives an update $\mathbf{0}$ such that $\mathbf{S}(\mathbf{0})$ is the empty model for each $\mathbf{S}$, and $\mathbf{0}_{\mathbf{S}}$ is the empty relation.

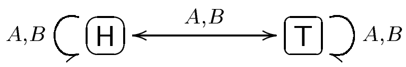
The operations $\mathbf{r};\mathbf{s}$, $\mathbf{r} \sqcup \mathbf{s}$ and $\mathbf{r}^*$ are the natural analogues of the operations of relational composition, union of relations and iteration of a relation, and also of the regular operations on programs in PDL. The intended meanings are: for $\mathbf{r};\mathbf{s}$, sequential composition (*do* $\mathbf{r}$*, then do* $\mathbf{s}$); for $\mathbf{r} \sqcup \mathbf{s}$, non-deterministic choice (*do either* $\mathbf{r}$ *or* $\mathbf{s}$); for $\mathbf{r}^*$, iteration (*repeat* $\mathbf{r}$ *some finite number of times*).

[3]We are writing relational composition in left-to-right order in this paper.

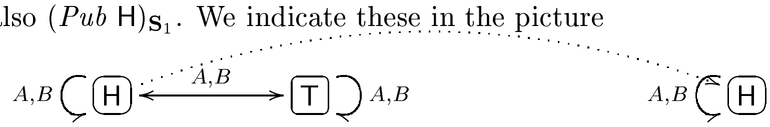
A new operation: dynamic modalities for updates. If φ is an epistemic proposition and $\mathbf{r}$ an update, then $[\mathbf{r}]\varphi$ is an epistemic proposition defined by

$$([\mathbf{r}]\varphi)_{\mathbf{S}} \quad = \quad \{s \in S : \text{if } s \; \mathbf{r}_{\mathbf{S}} \; t, \text{ then } t \in \varphi_{\mathbf{S}(\mathbf{r})}\}. \tag{4}$$

We should compare (4) and (2). The point is that we may treat updates in a similar manner to other box-like modalities; the structure given by an update allows us to do this. This point leads to the formal languages which we shall construct in due course. But we can illustrate the idea even now. Suppose we want to interpret the sentence $[Pub\ \mathsf{H}]\Box_A\mathsf{H}$ in our very first model, the model $\mathbf{S}_1$ pictured again below:



We shall call the two states s (where H holds) and t. Again, we want to determine $[\![[Pub\ \mathsf{H}]\Box_A\mathsf{H}]\!]_{\mathbf{S}_1}$. We already have in Example 2.1 a general definition of $Pub\ \mathsf{H}$ as an update, so we can calculate $\mathbf{S}_1(Pub\ \mathsf{H})$ and also $(Pub\ \mathsf{H})_{\mathbf{S}_1}$. We indicate these in the picture



The one-state model on the right is $\mathbf{S}_1(Pub\ \mathsf{H})$, and the dotted arrow shows $(Pub\ \mathsf{H})_{\mathbf{S}_1}$. So we calculate:

$$[\![[Pub\ \mathsf{H}]\Box_A\mathsf{H}]\!]_{\mathbf{S}_1} \quad = \quad \{s \in \mathbf{S}_1 : \text{whenever } s\ (Pub\ \mathsf{H})_{\mathbf{S}_1}\ t, \text{ then also } t \in [\![\Box_A\mathsf{H}]\!]_{\mathbf{S}_1(Pub\ \mathsf{H})}\}.$$

In $\mathbf{S}_1(Pub\ \mathsf{H})$, the state satisfies $\Box_A\mathsf{H}$. Thus $[\![[Pub\ \mathsf{H}]\Box_A\mathsf{H}]\!]_{\mathbf{S}_1} = \{s, t\}$. It might be more interesting to consider $\langle Pub\ \mathsf{H}\rangle\Box_A\mathsf{H}$; this is $\neg[Pub\ \mathsf{H}]\neg\Box_A\mathsf{H}$. Similar calculations show that

$$[\![\langle Pub\ \mathsf{H}\rangle\Box_A\mathsf{H}]\!]_{\mathbf{S}_1} \quad = \quad \{s \in \mathbf{S}_1 : \text{for some } t,\ s\ (Pub\ \mathsf{H})_{\mathbf{S}_1}\ t \text{ and } t \in [\![\Box_A\mathsf{H}]\!]_{\mathbf{S}_1(Pub\ \mathsf{H})}\}.$$

Hence $[\![\langle Pub\ \mathsf{H}\rangle\Box_A\mathsf{H}]\!]_{\mathbf{S}_1} = \{s\}$.

The point here is that we have a general semantics for sentences like $[Pub\ \mathsf{H}]\Box_A\mathsf{H}$, and this semantics crucially uses equation (4). That is, to determine the truth set of a sentence like $[Pub\ \mathsf{H}]\Box_A\mathsf{H}$ in a particular model $\mathbf{S}$, one *applies* the update map to $\mathbf{S}$ and works with the update relation between the $\mathbf{S}$ and the $\mathbf{S}([Pub\ \mathsf{H}])$; one also uses the semantics of $\Box_A\mathsf{H}$ in the new model. This overall point is one of the two leading features of our approach; the other is the update product which we introduce in Section 3.

2.4 The target logical systems

This paper presents a number of logical systems which contain *epistemic operators* of various types. These operators are closely related to aspects of the scenarios of Section 1.1. The logics themselves are presented formally in Section 4, but this work takes a fair amount of preparation. We delay this until after we motivate the subject, and so we turn to an informal presentation of the syntax and semantics of some logics. The overall idea is that the operators we study correspond roughly to the *shapes* of the action models which we shall see in Section 3.5.

THE LOGIC OF PUBLIC ANNOUNCEMENTS. We take a two-place sentential operator $[Pub\ -]-$. That is, we want an operation taking sentences φ and ψ to a new sentence $[Pub\ \varphi]\psi$, and we want a logic closed under this operation. The intended interpretation of $[Pub\ \varphi]\psi$ is: assuming that φ holds, then announcing it results in a state where ψ holds. The announcement here should be completely public. The semantics of every sentence χ will be an epistemic proposition $[\![\chi]\!]$ in the sense of Section 2.1.

Note that we refer to this operator as a *two-place* one. This just means that it takes two sentences and returns another sentence. We also think of $[Pub\ \varphi]$ as a modal operator in its own right, on a par with the knowledge operators $\Box_A$. And so we shall think of *Pub* as something which takes sentences into one-place modal operators.

We also consider a *dual* $\langle Pub\ \varphi\rangle$ to $[Pub\ \varphi]$. As one expects, the semantics will arrange that $\langle Pub\ \varphi\rangle\psi$ and $\neg[Pub\ \varphi]\neg\psi$ be logically equivalent. (That is, they will hold at the same states.) Thus, the intended interpretation of $\langle Pub\ \varphi\rangle\psi$ is: φ holds, and announcing it results in a state where ψ holds. As one indication of the difference, in $\mathbf{S}_1$, $y \models [Pub\ \mathsf{H}]\mathsf{true}$ (vacuously, as it were: it is not possible to make a true announcement of H in y). But we do have $y \models \neg\langle Pub\ \mathsf{H}\rangle\mathsf{true}$; this is how our semantics works out. Once again we only consider true announcements. Our semantics will arrange that $\varphi \to (\langle Pub\ \varphi\rangle\psi \leftrightarrow [Pub\ \varphi]\psi)$. So in working with the example scenarios, the difference between the two modalities will be small.

Further, we *iterate* the announcement operation, obtaining sentences such as $[Pub\ p][Pub\ q]r$. We also want to consider *announcements about announcements*, as in $\langle Pub\ \langle Pub\ \varphi\rangle\psi\rangle\chi$. This says that it is possible to announce publicly $\langle Pub\ \varphi\rangle\psi$, and as a result of a true announcement of this sentence, χ will hold.

THE LOGIC OF COMPLETELY PRIVATE ANNOUNCEMENTS TO GROUPS. This time, the syntax is more complicated. If φ and ψ are sentences and $\mathcal{B}$ is a set of agents, then $[Pri^{\mathcal{B}}\ \varphi]\psi$ is again a sentence. The intended interpretation of this is: assuming that φ holds, then announcing it publicly to the subgroup $\mathcal{B}$ in such a way that outsiders do not even suspect that the announcement happened results in a state where ψ holds. (The "*Pri*" in the notation stands for "private.") For example, this kind of announcement occurs in the passage from Scenario 1 to Scenario 4; that is, "cheating" is a kind of private announcement to the "group" $\{A\}$. We want it to be the case that in $\mathbf{S}_1$,

$$x \models \langle Pri^{\{A\}}\ \mathsf{H}\rangle(\Box_A\mathsf{H}\ \wedge\ \neg\Box_B\Box_A\mathsf{H}).$$

That is, in x, it is possible to announce H to A privately (since H is true in x), and by so doing we have a new state where A knows this fact, but B does not know that A knows it.

The logic of private announcements to groups allows as modal operators $[Pri^{\mathcal{B}}\ \varphi]$ for all sentences φ of the logic. We shall show that this logical system extends the logic of public announcements, the idea being that when we take $\mathcal{B}$ to be the full set $\mathring{A}$ of agents, $Pub\ \varphi$ and $Pri^{\mathcal{A}}\varphi$ should be equivalent in the appropriate sense.

THE LOGIC OF COMMON KNOWLEDGE OF ALTERNATIVES. If φ and ψ are sentences and $\mathcal{B}$ is a set of agents, then $[Cka^{\mathcal{B}}\ \vec{\varphi}]\psi$ is again a sentence. The intended interpretation of this is: assuming that φ_1 holds, then announcing it publicly to the subgroup $\mathcal{B}$ in such a way it is common knowledge to the set of all agents that the announcement was one of $\vec{\varphi}$ results in a state where ψ holds. For example, consider Scenario 3. In $\mathbf{S}_1$, we have

$$x \models \neg\Box_A\mathsf{H}\ \wedge\ \langle Cka^{\{A\}}\ \mathsf{H},\mathsf{T}\rangle(\Box_A(\mathsf{H}\ \wedge\ \neg\Box_B\Box_A\mathsf{H}) \wedge \Box_B(\Box_A\mathsf{H} \vee \Box_A\mathsf{T})).$$

Syntactic	Semantic
sentence φ	epistemic proposition $\boldsymbol{\varphi}$
language $\mathcal{L}$, $\mathcal{L}(\mathbf{\Sigma})$, etc.	state model $\mathbf{S}$
action signature $\mathbf{\Sigma}$	update $\mathbf{r}$
basic action expression $\sigma\vec{\psi}$	epistemic program model $(\mathbf{\Sigma}, \Gamma, \boldsymbol{\psi}_1, \ldots \boldsymbol{\psi}_n)$
program π, action α	canonical action model $\mathbf{\Omega}$

Figure 2: The main notions in this paper

The action here is A learning that either the coin lies heads up or that it lies tails up, and this is done in such a way that B knows that A learns one of the two alternatives but not which one. Before the action, A does not know that the coin lies heads up. As a result, A knows this, and knows that B does not know it.

At this point, we have the syntax of some logical languages and also examples of what we intend the semantics to be. We do not yet have the *formal* semantics of any of these languages, of course. However, even before we turn to this, we want to be clear that our goal in this paper is to study a very wide class of logical systems, including ones for the representation of all possible "announcement types." The reasons to be interested in this approach rather than to study a few separate logics are as follows: (1) it is more general and elegant to have a unified presentation; and (2) it gives a formal account of the notion of an "announcement type" which would be otherwise lacking and which should be of independent interest. So what we will really be concerned with is:

THE LOGIC OF ALL POSSIBLE EPISTEMIC ACTIONS. If α is an epistemic action and φ is a sentence, then $[\alpha]\varphi$ is again a sentence. Here α is some sort of epistemic action (which of course we shall define in due course); the point is that α might involve arbitrarily complex patterns of suspicion. In this way, we shall recover the four logical systems mentioned above as fragments of the larger logic of all possible epistemic actions.

Our claims in this paper Here are some of the claims of the paper about the logical languages we shall construct and about our notion of updates.

1. Each *type of epistemic action*, such as public announcements, completely private announcements, announcements with common knowledge of suspicion, etc, corresponds to a natural collection of updates as we have defined them above.
2. Each type also gives rise to a logical language with that type of action as a primitive construct. Moreover, it is possible to precisely define the syntax of the language to insure closure under the construct. That is, we should be able to formulate a language with announcements not just about atomic facts, but about announcements themselves, announcements about announcements, etc.

2.5 A guide to the concepts in this paper

In this section, we catalog the main notions that we shall introduce in due course. After this, we turn to a discussion of the particular languages that we study, and we situate them with

$$
\begin{tikzcd}
\mathcal{L}_0 \arrow[r, hook] \arrow[d, hook] & \mathcal{L}_1 \arrow[d, hook] & & & \\
\mathcal{L}_0(\mathcal{S}) \arrow[r, hook] & \mathcal{L}_1(\mathcal{S}) \arrow[r, hook] \arrow[dr] & \mathcal{L}(\mathcal{S}) \arrow[r, dashed] & \mathcal{L}_0^{\omega} \arrow[r, hook] & \mathcal{L}_0^{\infty} \\
& & \mathrm{PDL} \arrow[ur] & &
\end{tikzcd}
$$

Figure 3: Languages in this paper

existing logical languages.

Recall the we insist on a distinction of syntactic and semantic objects in this paper. We list in Figure 2 the main notions that we will need. We do this mostly to help readers as they explore the different notions. We mention now that the various notions are not developed in the order listed; indeed, we have tried hard to organize this paper in a way that will be easiest to read and understand. For example, one of our main goals is to present a set of languages (syntactic objects) and their semantics (utilizing semantic objects).

Languages This paper studies a number of languages, and to help the reader we list these in Figure 3. Actually, what we study are not individual languages, but rather *families* of languages parameterized by different choices of primitives. It is standard in modal logic to begin with a set of atomic propositions, and we do the same. The difference is that we shall call these atomic *sentences* in order to make a distinction between these essentially syntactic objects and the *semantic* propositions that we study beginning in Section 2.3. This is our first parameter, a set AtSen of *atomic sentences*. The second is a set $\mathcal{A}$ of *agents*.

Given these, $\mathcal{L}_0$ is ordinary modal logic with the elements of AtSen as atomic sentences and with agent-knowledge (or belief) modalities $\Box_A$ for $A \in \mathcal{A}$.

We add *common-knowledge operators* $\Box_{\mathcal{B}}^*$, for sets of agents $\mathcal{B} \subseteq \mathcal{A}$, to get a larger language $\mathcal{L}_1$. In Figure 3, we note the fact that $\mathcal{L}_0$ is literally a subset of $\mathcal{L}_1$ by using the inclusion arrow. The syntax and semantics of $\mathcal{L}_0$ and $\mathcal{L}_1$ are presented in Figure 4.

Another close neighbor of the system in this paper is Propositional Dynamic Logic (PDL). PDL was first formulated by Fischer and Ladner [9], following the introduction of dynamic logic in Pratt [19]. The syntax and the main clauses in the semantics of PDL are shown in Figure 5.

We may also take $\mathcal{L}_0$ and close under countable conjunctions (and hence also disjunctions). We call this language $\mathcal{L}_0^{\omega}$. Note that $\mathcal{L}_1$ is not literally a subset of $\mathcal{L}_0^{\omega}$, but there is a translation of $\mathcal{L}_1$ into $\mathcal{L}_0^{\omega}$ that preserves the semantics. We would indicate this in a chart with a dashed arrow. Going further, we may close under arbitrary (set-sized) boolean operations; this language is then called $\mathcal{L}_0^{\infty}$.

PDL is propositional dynamic logic, formulated with atomic programs a replacing the agents A that we have fixed. We might note that we can translate $\mathcal{L}_1$ into PDL. The main clauses in

Syntax	**Sentences** φ	$p_i \mid \neg\varphi \mid \varphi \wedge \psi \mid \Box_A \varphi \mid \Box^*_{\mathcal{B}} \varphi$
Semantics	**Main Clauses**	$[\![\Box_A\varphi]\!]\mathbf{s} = \Box_A[\![\varphi]\!]\mathbf{s}$ $[\![\Box^*_{\mathcal{B}}\varphi]\!]\mathbf{s} = \Box^*_{\mathcal{B}}[\![\varphi]\!]\mathbf{s}$

Figure 4: The languages $\mathcal{L}_0$ and $\mathcal{L}_1$. For $\mathcal{L}_0$, we drop the $\Box^*_{\mathcal{B}}$ construct.

Syntax	**Sentences** φ **Programs** π	$p_i \mid \neg\varphi \mid \varphi \wedge \psi \mid [\pi]\varphi$ $a \mid ?\varphi \mid \pi;\sigma \mid \pi \cup \sigma \mid \pi^*$
Semantics	**Main Clauses**	$[\![[\pi]\varphi]\!]\mathbf{s} = \{s : \text{if } s[\![\pi]\!]\mathbf{s}t, \text{ then } t \in [\![\varphi]\!]\mathbf{s}\}$ $[\![?\varphi]\!]\mathbf{s} = \{(s,s) : s \in [\![\varphi]\!]\mathbf{s}\}$ $[\![\pi;\sigma]\!]\mathbf{s} = [\![\pi]\!]\mathbf{s};[\![\sigma]\!]\mathbf{s}$ $[\![\pi \cup \sigma]\!]\mathbf{s} = [\![\pi]\!]\mathbf{s} \cup [\![\sigma]\!]\mathbf{s}$ $[\![\pi^*]\!]\mathbf{s} = ([\![\pi]\!]\mathbf{s})^*$

Figure 5: Propositional Dynamic Logic (PDL)

the translation are:

$$\begin{array}{lcl} (\Box_A\varphi)^t & = & [a]\varphi^t \\ (\Box^*_{\mathcal{B}}\varphi)^t & = & [(\bigcup_{b\in B} b)^*]\varphi^t \end{array}$$

Beginning in Section 4.2, we study some new languages. These will be based on a third parameter, an *action signature* $\mathbf{\Sigma}$. For each "type of epistemic action" there will be an action signature $\mathbf{\Sigma}$. For each $\mathbf{\Sigma}$, we'll have languages $\mathcal{L}_0(\mathbf{\Sigma})$, $\mathcal{L}_1(\mathbf{\Sigma})$, and $\mathcal{L}(\mathbf{\Sigma})$. More generally, for each *family of action signatures* $\mathcal{S}$, we have languages $\mathcal{L}_0(\mathcal{S})$, $\mathcal{L}_1(\mathcal{S})$, and $\mathcal{L}(\mathcal{S})$. These will be related to the other languages as indicated in the figure.

So we shall extend modal logic by adding one or another type of epistemic action. The idea is then we can generate a logic from an action signature $\mathbf{\Sigma}$ corresponding to an intuitive action. For example, corresponding to the notion of a public announcement is a particular action signature $\mathbf{\Sigma}_{\text{pub}}$, and then the languages $\mathcal{L}_0(\mathbf{\Sigma}_{\text{pub}})$, $\mathcal{L}_1(\mathbf{\Sigma}_{\text{pub}})$, and $\mathcal{L}(\mathbf{\Sigma}_{\text{pub}})$ will have something to do with our notion of a "logic of public announcements."

In [7], we compare the expressive power of the languages mentioned so far. It turns out that all of the arrows in Figure 3 correspond to proper increases in expressive power. (There is one exception: $\mathcal{L}_0$ turns out to equal $\mathcal{L}_0(\mathcal{S})$ in expressive power for all $\mathcal{S}$.) It is even more interesting to compare expressive power as we change the action signature. For example, we would like to compare logics of public announcement to logics of private announcement to groups. Most of the natural questions in this area are open as of this writing.

2.6 Reformulation of Test-only PDL

In PDL, there are two types of syntactic objects, sentences and programs. The programs in PDL are interpreted on a structure by relations on that structure. This is *not* the way our

Syntax	**Sentences** φ	$p_i \mid \neg\varphi \mid \varphi \wedge \psi \mid [\pi]\varphi$
	Programs π	$?\varphi \mid \mathit{skip} \mid \pi;\sigma \mid \pi \cup \sigma \mid \pi^*$
Semantics	**Main Clauses**	$[\![[\pi]\varphi]\!] = [[\![\pi]\!]][\![\varphi]\!]$
		$[\![\mathit{skip}]\!] = \mathbf{1}$ from Section 2.3
		$[\![?\varphi]\!] = ?[\![\varphi]\!]$ as in Example 2.2
		$[\![\pi;\sigma]\!] = [\![\pi]\!];[\![\sigma]\!]$
		$[\![\pi \cup \sigma]\!] = [\![\pi]\!] \sqcup [\![\sigma]\!]$
		$[\![\pi^*]\!] = [\![\pi]\!]^*$

Figure 6: Test-only PDL, with a semantics in our style

semantics works, and to make this point we compare the standard semantics of (a fragment of) PDL with a language closer to what we shall ultimately study.

To make this point, we consider the *test-only* fragment of PDL in our terms. This is the fragment built over the empty set of atomic programs. So the programs are *skip*, the tests $?\varphi$ and compositions, choices, or iterations of these; sentences are formed as in PDL We give a reformulation in Figure 6. The point is that in PDL the programs are interpreted by relations on a given model, and in our terms programs are interpreted by *updates*. We have discussed updates of the form $?\boldsymbol{\varphi}$ in Example 2.2. Given that we have an interpretation of the sentence φ as an epistemic proposition $[\![\varphi]\!]$, we then automatically have an update $?[\![\varphi]\!]$.

For the sentences, the main thing to look at is the semantics of sentences $[\pi]\varphi$; here we use the semantic notions from Section 2.3. The way the semantics works is that we have $[\![\pi]\!]$ and $[\![\varphi]\!]$; the former is an update and the latter is an epistemic proposition. Then we use both of these to get an overall semantics, using equation (4). In more explicit terms,

$$
\begin{array}{rcl}
[\![[\pi]\varphi]\!]_{\mathbf{S}} & = & ([[\![\pi]\!]][\![\varphi]\!])_{\mathbf{S}} \\
& = & \{s \in S : \text{if } s\ [\![\pi]\!]_{\mathbf{S}}\ t, \text{ then } t \in [\![\varphi]\!]_{\mathbf{S}}\}
\end{array}
$$

2.7 Background: bisimulation

We shall see the concept of bisimulation at various points in the paper, and this section reviews the concept and also develops some of the appropriate definitions concerning bisimulation and updates.

Definition Let $\mathbf{S}$ and $\mathbf{T}$ be state models. A *bisimulation* between $\mathbf{S}$ and $\mathbf{T}$ is a relation $R \subseteq S \times T$ such that whenever $s\ R\ t$, the following three properties hold:

1. $s \in \|p\|_{\mathbf{S}}$ iff $t \in \|p\|_{\mathbf{T}}$ for all atomic sentences p. That is, s and t agree on all atomic information.

2. For all agents A and states s' such that $s \xrightarrow{A} s'$, there is some state t' such that $t \xrightarrow{A} t'$ and $s'\ R\ t'$.

3. For all agents A and states t' such that $t \xrightarrow{A} t'$, there is some state s' such that $s \xrightarrow{A} s'$ and $s' \; R \; t'$.

Example 2.3 This example concerns the update operation $?\boldsymbol{\varphi}$ of Example 2.2. Fix an epistemic proposition $\boldsymbol{\varphi}$ and a state model $\mathbf{S}$. Recall that $(?\boldsymbol{\varphi})_{\mathbf{S}}$ relates each $s \in \boldsymbol{\varphi}$ to the corresponding pair $(0, s)$. We check that the following relation R is a bisimulation between $\mathbf{S}$ and $\mathbf{S}(?\boldsymbol{\varphi})$:

$$R \quad = \quad (?\boldsymbol{\varphi})_{\mathbf{S}} \cup \{(s, (1, s)) : s \in S\}.$$

The definition of $\mathbf{S}(?\boldsymbol{\varphi})$ insures that the interpretations of atomic sentences are preserved by this relation.

Next, suppose that $s \xrightarrow{A} t$ in $\mathbf{S}$ and $s \; R \; (i, s')$. Then we must have $s' = s$. Further, the definition of R tells us that $t \; R \; (1, t)$, and $(i, s) \xrightarrow{A} (1, t)$.

Finally, suppose that $s \; R \; (i, s')$ and $(i, s') \xrightarrow{A} (j, t)$. Then $s' = s$ and $j = 1$. In addition, the definition $\xrightarrow{A}$ in $\mathbf{S}(?\boldsymbol{\varphi})$ implies that $s \xrightarrow{A} t$ in $\mathbf{S}$. And as above, $t \; R \; (1, t)$. This concludes our verifications.

Recall that $\mathcal{L}_0$ is ordinary modal logic, formulated as always over our fixed sets of agents and atomic sentences. The next result concerns the language $\mathcal{L}_0^\infty$ of *infinitary* modal logic. In this language, one has conjunctions and disjunctions of arbitrary sets of sentences. We have the following well-known result:

Proposition 2.4 *If there is a bisimulation R such that $s \; R \; t$, then s and t agree on all sentences in $\mathcal{L}_0^\infty$: for all $\varphi \in \mathcal{L}_0^\infty$, $s \in [\![\varphi]\!]_{\mathbf{S}}$ iff $t \in [\![\varphi]\!]_{\mathbf{T}}$.*

A *pointed state model* is a pair $(\mathbf{S}, s)$ such that $s \in S$. The state s is called the *designated state* (or the "point") of our pointed model, and is meant to represent the *actual state* of the system. Two pointed models models are said to be *bisimilar* if there exists a bisimulation relation between them which relates their designated states. So, denoting by $\equiv$ the relation of bisimilarity, we have:

$$(\mathbf{S}, s) \equiv (\mathbf{T}, t) \quad \text{iff} \quad \text{there is a bisimulation } R \text{ between } \mathbf{S} \text{ and } \mathbf{T} \text{ such that } s \; R \; t.$$

This relation $\equiv$ is indeed an equivalence relation.

When $\mathbf{S}$ and $\mathbf{T}$ are clear from the context, we write $s \equiv t$ instead of $(\mathbf{S}, s) \equiv (\mathbf{T}, t)$.

We say that a proposition $\boldsymbol{\varphi}$ *preserves bisimulations* if whenever $(\mathbf{S}, s) \equiv (\mathbf{T}, t)$, then $s \in \boldsymbol{\varphi}_{\mathbf{S}}$ iff $t \in \boldsymbol{\varphi}_{\mathbf{T}}$. We also say that an update $\mathbf{r}$ *preserves bisimulations* if the following two conditions hold:

1. If $s \; \mathbf{r}_{\mathbf{S}} \; s'$ and $(\mathbf{S}, s) \equiv (\mathbf{T}, t)$, then there is some t' such that $t \; \mathbf{r}_{\mathbf{T}} \; t'$ and $(\mathbf{S}(\mathbf{r}), s') \equiv (\mathbf{T}(\mathbf{r}), t')$.
2. If $t \; \mathbf{r}_{\mathbf{T}} \; t'$ and $(\mathbf{S}, s) \equiv (\mathbf{T}, t)$, then there is some s' such that $s \; \mathbf{r}_{\mathbf{S}} \; s'$ and $(\mathbf{S}(\mathbf{r}), s') \equiv (\mathbf{T}(\mathbf{r}), t')$.

Proposition 2.5 *Concerning bisimulation preservation:*

1. *The bisimulation preserving propositions include the atomic propositions* $\boldsymbol{p}$*, and they are closed under all of the (infinitary) operations on propositions.*

2. *The bisimulation preserving updates are closed under composition and (infinitary) sums.*

3. *If* φ *and* $\mathbf{r}$ *preserve bisimulations, so does* $[\mathbf{r}]\varphi$*.*

Proof We show the last part. Suppose that $s \in ([\mathbf{r}]\varphi)_{\mathbf{S}}$, and let $(\mathbf{S}, s) \equiv (\mathbf{T}, t)$. To show that $t \in ([\mathbf{r}]\varphi)_{\mathbf{T}}$, let $t\ \mathbf{r}_{\mathbf{T}}\ t'$. Then by condition (2) above, there is some s' such that $s\ \mathbf{r}_{\mathbf{S}}\ s'$ and $(\mathbf{S}(\mathbf{r}), s') \equiv (\mathbf{T}(\mathbf{r}), t')$. Since $s \in ([\mathbf{r}]\varphi)_{\mathbf{S}}$, we have $s' \in \varphi_{\mathbf{S}(\mathbf{r})}$. And then $t' \in \varphi_{\mathbf{T}(\mathbf{r})}$, since φ too preserves bisimulation. $\dashv$

Endnotes As far as we know, the first paper to study the interaction of communication and knowledge in a formal setting is Plaza's paper "Logics of Public Communications" [18]. As the title suggests, the epistemic actions studied are public announcements. In essence, he formalized the logic of public announcements. (In addition to this, [18] contains a number of results special to the logic of announcements which we have not generalized, and it also studies an extension of the logic with non-rigid constants.) The same formalization was made in Gerbrandy [10, 11], and also Gerbrandy and Groeneveld [12]. These papers further formalize the logic of completely private announcements. (However, their semantics use non-wellfounded sets rather than arbitrary Kripke models. As pointed out in [17], this restriction was not necessary.) The logic of common knowledge of alternatives was formulated in [6] and also in van Ditmarsch's dissertation [23].

Our introduction of updates is new here, as are the observations on the test-only fragment of PDL.

For connections of the ideas here with coalgebra, see Baltag [4].

One very active arena for work on knowledge is distributed systems, and the main source of this work is the book **Reasoning About Knowledge** [8] by Fagin, Halpern, Moses, and Vardi. We depart from [8] by introducing operators whose semantics are updates as we have defined them, and by doing without temporal logic operators. In effect, our Kripke models are simpler, since they do not incorporate all of the runs of a system; the new operators can be viewed as a compensation for that.

Remark Our formulation of a program model uses a designated *set* of simple actions. There are other equivalent formulations. Another way would be to use a disjoint union of *pointed* program models; It would be possible to further reformulate some of our definitions below and thereby to give a semantics for our ultimate languages that differs from what we obtain in Section 4.2 below. However, the two semantics would be equivalent. The reason we prefer to work with designated sets is that it permits us to draw diagrams with fewer states. The cost of the simpler representations is the slightly more complicated definition, but we feel this cost is worth paying.

3 The update product operation

In this section, we present the centerpiece of the formulation of our logical systems by introducing *action models*, *program models*, and an *update product* operation. The leading idea is that epistemic actions, like state models, have the property that different agents think that different things are possible. To model the effect of an epistemic update on a state, we use a kind of *product of epistemic alternatives.*

3.1 Epistemic action models

Let Φ be the collection of all epistemic propositions. An *epistemic action model* is a triple $\mathbf{\Sigma} = (\Sigma, \stackrel{A}{\to}, \mathrm{pre})$, where Σ is a set of *simple actions*, $\stackrel{A}{\to}$ is an $\mathcal{A}$-indexed family of relations on Σ, and $\mathrm{pre} : \Sigma \to \Phi$.

An epistemic action model is similar to a state model. But we call the members of the set Σ "simple actions" (instead of states). We use different notation and terminology because of a technical difference and a bigger conceptual point. The technical difference is that $\mathrm{pre} : \Sigma \to \Phi$ (that is, the codomain is the collection of all epistemic propositions). The conceptual point is that we think of "simple" actions as being *deterministic* actions whose epistemic impact is *uniform on states* (in the sense explained in our Introduction). So we think of "simple" actions as particularly simple kinds of deterministic actions, whose appearance to agents is uniform: the agents' uncertainties concerning the current action are independent of their uncertainties concerning the current state. This allows us to abstract away the action uncertainties and represent them as a Kripke structure of actions, in effect forgetting the state uncertainties.

As announced in the Introduction, this uniformity of appearance is restricted only to the action's domain of applicability, defined by its preconditions. Thus, for a simple action $\sigma \in \Sigma$, we interpret $\mathrm{pre}(\sigma)$ as giving the *precondition of* σ; this is what needs to hold at a state (in a state model) in order for action σ to be "accepted" in that state. So σ will be executable in s iff its precondition $\mathrm{pre}(\sigma)$ holds at s.

At this point we have mentioned the ways in which action models and state models differ. What they have in common is that they use accessibility relations to express each agent's uncertainty concerning something. For state models, the uncertainty has to do with which state is the real one; for action models, it has to do with which action is taking place.

Usually we drop the word "epistemic" and therefore refer to "action models".

Example 3.1 Here is an action model:

$$\begin{tikzcd} \boxed{\sigma{:}\mathsf{H}} \arrow[loop left, "A"] \arrow[rr, "B"] & & \boxed{\tau{:}\mathbf{tr}} \arrow[loop right, "{A,B}"] \end{tikzcd}$$

Formally, $\Sigma = \{\sigma, \tau\}$; $\sigma \stackrel{A}{\to} \sigma$, $\sigma \stackrel{B}{\to} \tau$, $\tau \stackrel{A}{\to} \tau$, $\tau \stackrel{B}{\to} \tau$; $\mathrm{pre}(\sigma) = \mathsf{H}$, and $\mathrm{pre}(\tau) = \mathbf{tr}$, where recall that $\mathbf{tr}$ is the *"always true"* proposition.

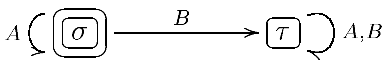
As we shall see, this action model will be used in the modeling of a *completely private announcement to A that the coin is lying heads up.* Further examples may be found later in this section.

3.2 Program models

To model non-deterministic actions and non-simple actions (whose appearances to agents are not uniform on states), we define *epistemic program models*. In effect, this means that we decompose complex actions ('programs') into "simple" ones: they correspond to sets of simple, deterministic actions from a given action model.

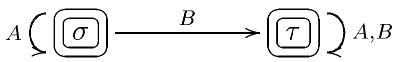
An *epistemic program model* is defined as a pair $\pi = (\mathbf{\Sigma}, \Gamma)$ consisting of an action model $\mathbf{\Sigma}$ and a set Γ of *designated simple actions*. Each of the simple actions $\gamma \in \Gamma$ can be thought as being a possible "deterministic resolution" of the non-deterministic action π. As announced above, the intuition about the map pre is that an action is executable in a given state only if all its preconditions hold at that state. We often spell out an epistemic program model as $(\Sigma, \xrightarrow{A}, \mathrm{pre}, \Gamma)$ rather than $((\Sigma, \xrightarrow{A}, \mathrm{pre}), \Gamma)$. When drawing the diagrams, we use doubled circles to indicate the designated actions in the set Γ. Finally, we usually drop the word "epistemic" and just refer to these as *program models*.

Example 3.2 Every action model $\mathbf{\Sigma}$ and every $\sigma \in \Sigma$ gives a program model by taking $\{\sigma\}$ as the set of designated simple actions. For instance, in connection with Example 3.1, we have



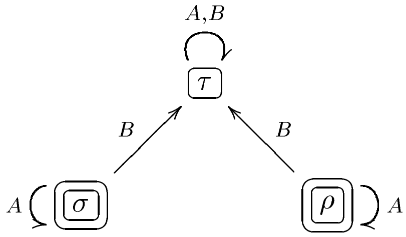
with $\mathrm{pre}(\sigma) = \mathsf{H}$, $\mathrm{pre}(\tau) = \mathbf{tr}$, as before. Program models of this type are very common in our treatment. But we need the extra ability to have sets of designated simple actions to deal with more complicated actions, as our next examples show.

Example 3.3 *A non-deterministic action:* Let us model the non-deterministic action of *either making a completely private announcement to A that the coin is lying heads up, or not making any announcement.* The action is completely private, so B doesn't suspect anything: he thinks no announcement is made. The program model is obtained by choosing $\Gamma = \{\sigma, \tau\}$ in the action model $\mathbf{\Sigma}$ from Example 3.1. The picture is



with $\mathrm{pre}(\sigma) = \mathsf{H}$, $\mathrm{pre}(\tau) = \mathbf{tr}$, as before. Alternatively, one could take the disjoint union of $\mathbf{\Sigma}$ with the one-action program model with precondition $\mathbf{tr}$.

Example 3.4 *A deterministic, but non-simple action:* Let us model the action of *(completely) privately announcing to A whether the coin is lying heads up or not.* Observe that this is a deterministic action; that is, the update relation is functional: at any state, the coin is either heads up or not. But the action is not simple: its appearance to A depends on the state. In states in which the coin is heads up, this action looks to A like a private announcement that H is the case; in the states in which the coin is not heads up, the action looks to A like a private announcement that $\neg\mathsf{H}$. (However, the appearance to B is uniform: at any state, the action appears to him as if no announcement is made.) The only way to model this deterministic, but non-simple, action in our setting is as a non-deterministic program model, having as its 'designated' actions two mutually exclusive simple actions: one corresponding to a (truthful) private announcement that H, and another one corresponding to a (truthful) private announcement that ¬H.



with $\mathrm{pre}(\sigma) = \mathsf{H}$, $\mathrm{pre}(\tau) = \mathbf{tr}$, and $\mathrm{pre}(\rho) = \neg\mathsf{H}$.

3.3 The update product of a state model with an epistemic action model

The following operation plays a central role in this paper.

Given a state model $\mathbf{S} = (S, \xrightarrow{A}_{\mathbf{S}}, \|\cdot\|_{\mathbf{S}})$ and an action model $\mathbf{\Sigma} = (\Sigma, \xrightarrow{A}, \mathrm{pre})$, we define their *update product* to be the state model

$$\mathbf{S} \otimes \mathbf{\Sigma} \quad = \quad (S \otimes \Sigma, \xrightarrow{A}, \|.\|),$$

given by the following: the new states are pairs of old states s and simple actions σ which are "consistent", in the sense that all preconditions of the action σ "hold" at the state s

$$S \otimes \Sigma \quad = \quad \{(s, \sigma) \in S \times \Sigma : s \in \mathrm{pre}(\sigma)_{\mathbf{S}}\}. \tag{5}$$

The new accessibility relations are taken to be the "products" of the corresponding accessibility relations in the two frames; i.e., for $(s, \sigma), (s', \sigma') \in \mathbf{S} \otimes \mathbf{\Sigma}$ we put

$$(s, \sigma) \xrightarrow{A} (s', \sigma') \quad \text{iff} \quad s \xrightarrow{A} s' \text{ and } \sigma \xrightarrow{A} \sigma', \tag{6}$$

and the new valuation map $\|.\|_{\mathbf{S}} : \mathsf{AtSen} \to \mathcal{P}(S \otimes \Sigma)$ is essentially given by the old valuation:

$$\|p\|_{\mathbf{S}\otimes\mathbf{\Sigma}} \quad = \quad \{(s, \sigma) \in S \otimes \Sigma : s \in \|p\|_{\mathbf{S}}\}. \tag{7}$$

Intended interpretation The update product restricts the full Cartesian product $S \times \Sigma$ to the smaller set $S \otimes \Sigma$ in order to insure that *states survive actions* in the appropriate sense.

For each agent A, the product arrows $\xrightarrow{A}$ on the output frame represent agent A's epistemic uncertainty about the output state. The intuition is that the components of our action models are "simple actions", so the uncertainty regarding the action is assumed to be independent of the uncertainty regarding the current (input) state. This independence allows us to "multiply" these two uncertainties in order to compute the uncertainty regarding the output state: if whenever the input state is s, agent A thinks the input might be some other state s', and if whenever the current action happening is σ, agent A thinks the current action might be some other action σ', and if s' survives σ', then whenever the output state (s, σ) is reached, agent A thinks the alternative output state (s', σ') might have been reached. Moreover, these all of the output states that A considers possible are of this form.

As for the valuation, we essentially take the same valuation as the one in the input model. If a state s survives an action, then the same facts p hold at the output state (s, σ) as at the input state s. This means that *our actions, if successful, do not change the facts.* This condition can of course be relaxed in various ways, to allow for fact-changing actions. But in this paper we are primarily concerned with purely epistemic actions, such as the earlier examples in this section.

3.4 Updates induced by program models

Recall that we defined updates and proper epistemic actions in Section 2.3. Right above, in Section 3.2, we defined epistemic action *models.* Note that there is a big difference: the updates are pairs of operations on the class of all state models, and the program models are typically finite structures. We think of *program models* as capturing specific mechanisms, or algorithms, for inducing updates. This connection is made precise in the following definition.

Definition Let $(\mathbf{\Sigma}, \Gamma)$ be a program model. We define an update which we also denote $(\mathbf{\Sigma}, \Gamma)$ as follows:

1. $\mathbf{S}(\mathbf{\Sigma}, \Gamma) = \mathbf{S} \otimes \mathbf{\Sigma}$.
2. $s\ (\mathbf{\Sigma}, \Gamma)_{\mathbf{S}}\ (t, \sigma)$ iff $s = t$ and $\sigma \in \Gamma$.

We call this *the update induced by* $(\mathbf{\Sigma}, \Gamma)$.

Bisimulation preservation Before moving to examples, we note a simple result that will be used later.

Proposition 3.5 *Let $\mathbf{\Sigma}$ be an action model in which every pre(σ) is a bisimulation preserving epistemic proposition. Let $\Gamma \subseteq \Sigma$ be arbitrary. Then the update induced by $(\mathbf{\Sigma}, \Gamma)$ preserves bisimulation.*

Proof Write $\mathbf{r}$ for the update induced by the program model $(\mathbf{\Sigma}, \Gamma)$. Fix $\mathbf{S}$ and $\mathbf{T}$, and suppose that $s \equiv t$ via the relation R_0. Suppose that $s\ \mathbf{r}_{\mathbf{S}}\ s'$, so $s' \in \mathbf{S}(\mathbf{r})$ is of the form (s, σ) for some $\sigma \in \Gamma$. Then $(t, \sigma) \in \mathbf{T}(\mathbf{r})$, and clearly $t\ \mathbf{r}_{\mathbf{T}}\ (t, \sigma)$. We need only show that $(s, \sigma) \equiv (t, \sigma)$. But the following relation R is a bisimulation between $\mathbf{S}(\mathbf{r})$ and $\mathbf{T}(\mathbf{r})$:

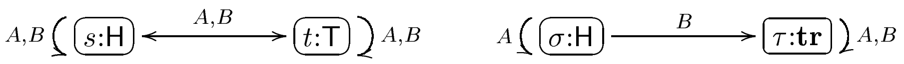
$$(s', \tau_1)\ R\ (t', \tau_2) \quad \text{iff} \quad s'\ R_0\ t' \text{ and } \tau_1 = \tau_2.$$

The verification of the bisimulation properties is easy. And R shows that $(s, \sigma) \equiv (t, \sigma)$, as desired. $\dashv$

3.5 The coin scenario models as examples of the update product

We return to the coin scenarios of Section 1.1. Our purpose is partly to indicate how one may obtain the models there as examples of the update product, and at the same time to exemplify the update product construction itself. In this section, the set $\mathcal{A}$ of agents is $\{A, B\}$, and the set AtSen of atomic propositions is $\{\mathsf{H}, \mathsf{T}\}$. We remind the reader that T represents the coin lying tails up, while $\mathbf{tr}$ is our notation for the true epistemic proposition.

Example 3.6 We begin with an example worked out with many details, and then the rest of our examples will omit many similar calculations. This example has to do with Scenario 4 from Section 1.1, where the coin lies heads up and A takes a look at the coin in such a way that she is certain that B does not suspect anything. We take as $\mathbf{S}_1$ and $\mathbf{\Sigma}_4$ the structures shown below:



(We remind the reader that T is the atomic sentence for "tails" and $\mathbf{tr}$ is for "true".) $\mathbf{S}_1$ is from Scenario 1. In $\mathbf{\Sigma}_4$, we take the set of distinguished states to be $\{\sigma\}$. $\mathbf{\Sigma}$ comes from Examples 3.1 and 3.2. To take the update product, we first form the cartesian product $S_1 \times \Sigma_4$:

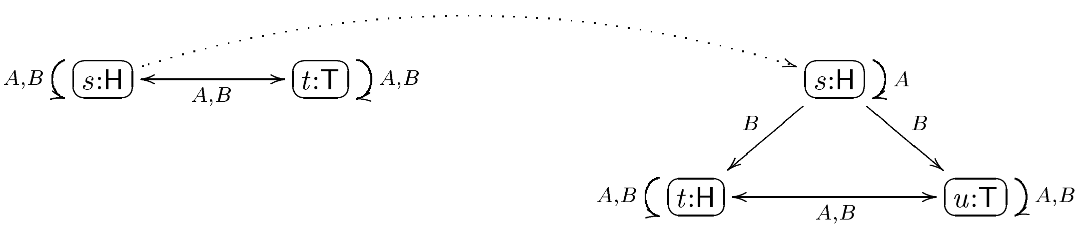
$$\{(s,\sigma),(s,\tau),(t,\sigma),(t,\tau)\}$$

Of these four pairs, we only want those whose first component satisfies (in $\mathbf{S}$) the precondition of the second component. We do not want (t,σ), since $\mathrm{pre}(\sigma) = \mathsf{H}$ and $t \notin [\![\mathsf{H}]\!]_{\mathbf{S}}$. But the other three pairs do satisfy our condition. So the state model $\mathbf{S}_1 \otimes \mathbf{\Sigma}_4$ will have three states: (s,σ), (s,τ), and (t,τ). The atomic information is inherited from the first component, so we have $[\![\mathsf{H}]\!]_{\mathbf{S}_1\otimes\mathbf{\Sigma}_4} = \{(s,\sigma),(s,\tau)\}$ and $[\![\mathsf{T}]\!]_{\mathbf{S}_1\otimes\mathbf{\Sigma}_4} = \{(t,\tau)\}$. The accessibility relations $\xrightarrow{A}$ and $\xrightarrow{B}$ are those of the product. For example, we have $(s,\sigma) \xrightarrow{B} (s,\tau)$, because $s \xrightarrow{B} s$ and $\sigma \xrightarrow{B} \tau$. But we do not have $(s,\sigma) \xrightarrow{A} (s,\tau)$, because $\sigma \xrightarrow{A} \tau$ is false.

Now, we rename the states as follows:

$$(s,\sigma) \;\rightsquigarrow\; s \qquad\qquad (s,\tau) \;\rightsquigarrow\; t \qquad\qquad (t,\tau) \;\rightsquigarrow\; u$$

And then we get a picture of this state model, the same one we had in Scenario 4:



The dotted line here shows the update relation between s and (s,σ). This is the only update relation. For example, we do not relate s and (s,τ) because $\tau \notin \Gamma = \{\sigma\}$. Let $\mathbf{S}_4 = \mathbf{S}_1 \otimes \mathbf{\Sigma}_4$.

Example 3.7 $\mathbf{\Sigma}_4$ from Example 3.6 represents the action where A cheats, learning heads or tails in a way which is oblivious to B. It is also natural to consider an action of A *privately learning the state of the coin.* (This action may take place even if the state is tails.) We could use the following program model $\mathbf{\Sigma}_4'$:

This program model consists of two disjoint components. We leave it to the reader to calculate $\mathbf{S}_1 \otimes \mathbf{\Sigma}_4'$, and also the relation between "before" and "after".

Example 3.8 Next we construct the model corresponding to Scenario 5, where B cheats after Scenario 4. We consider the update product of the state model $\mathbf{S}_4$ from Example 3.6 above with the program model $\mathbf{\Sigma}_5$ shown below:

$$\begin{tikzcd} \boxed{\sigma{:}\mathsf{H}} \arrow[loop left, "B"] \arrow[r, "A"] & \boxed{\tau{:}\mathbf{tr}} \arrow[loop right, "A,B"] \end{tikzcd}$$

It represents cheating by B. The update product $\mathbf{S}_4 \otimes \mathbf{\Sigma}_5$ has five states: (s, σ), (t, σ), (s, τ), (t, τ), and (u, τ). Notice that (u, σ) is omitted, since $u \notin [\![\mathsf{H}]\!]_{\mathbf{S}_4}$. We leave it to the reader to calculate the accessibility relations in $\mathbf{S}_4 \otimes \mathbf{\Sigma}_5$; and to draw the appropriate figure.

Incidentally, we posed in Section 1.1 the exercise of constructing this representation from first principles. Many people are able to construct the five state picture, and some others construct a related picture with seven states. The seven state picture is bisimilar to the one illustrated here.

Example 3.9 We next look back at Scenario 2. The simplest action structure is $\mathbf{\Sigma}_2$:

$$\begin{tikzcd} \boxed{\mathsf{H}} \arrow[loop right, "A,B"] \end{tikzcd} \tag{8}$$

It represents a public announcement to A and B that the coin is lying heads up. Here, the distinguished set Γ is the entire action structure. For the record, we formalize Σ_2 as a singleton set $\{Pub\}$. We have $Pub \xrightarrow{A} Pub$ and $Pub \xrightarrow{B} Pub$. Also, we set $\mathrm{pre}(Pub) = \mathsf{H}$. We did not put the name Pub in the representation in (8), but in situations where we want the name we would draw the same picture except that instead of H we would say $Pub : \mathsf{H}$. Let us call the same structure $\mathbf{S}_2$ when we view it as a state model; formally these are the same object.

Let $\mathbf{S}$ be any model with the property that every action has both a successor in $\xrightarrow{A}$ where H is true, and also a successor in $\xrightarrow{B}$ where H is true. Then $\mathbf{S} \otimes \mathbf{\Sigma}_2$ is bisimilar to $\mathbf{S}_2$. In particular, $\mathbf{S}_1 \otimes \mathbf{\Sigma}_2$ is bisimilar to $\mathbf{S}_2$.

Example 3.10 Let $\mathbf{\Sigma}_3$ be the following program model:

$$\begin{tikzcd} \boxed{\boxed{\mathsf{H}}} \arrow[loop left, "A,B"] \arrow[r, leftrightarrow, "B"] & \boxed{\mathsf{T}} \arrow[loop right, "A,B"] \end{tikzcd}$$

$\mathbf{\Sigma}_3$ represents an announcement to A of heads in the manner of Scenario 3. That is, B knows that A either sees heads or sees tails, but not which. Similarly, let let $\mathbf{\Sigma}'_3$ represent the same kind of announcement to B:

$$\begin{tikzcd} \boxed{\boxed{\mathsf{H}}} \arrow[loop left, "A,B"] \arrow[r, leftrightarrow, "A"] & \boxed{\mathsf{T}} \arrow[loop right, "A,B"] \end{tikzcd}$$

Then we have the following:

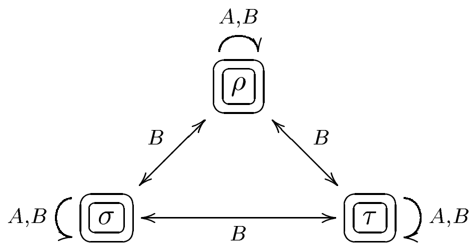
$\mathbf{S}_1 \otimes \mathbf{\Sigma}_3 \cong \mathbf{S}_3$, where $\mathbf{S}_3$ is the model in Scenario 3 and $\cong$ denotes isomorphism.

$\mathbf{S}_3 \otimes \mathbf{\Sigma}'_3 \cong \mathbf{S}_2$. This conforms with the intuition that successive semi-private viewings by the two parties of the concealed coin amount to a public viewing.

$\mathbf{S}_3 \otimes \mathbf{\Sigma}_3 \cong \mathbf{S}_3$. There is no point for A to look twice.

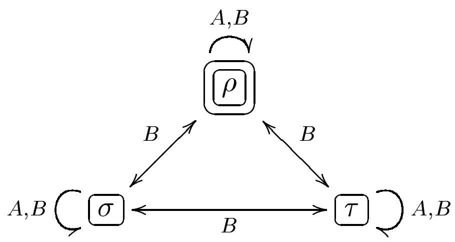
$\mathbf{S}_4 \otimes \mathbf{\Sigma}_3$ is a three-state model bisimilar to the model $\mathbf{S}_4$ from Scenario 4.

Example 3.11 To obtain the model in Scenario 7, we use the following program model $\mathbf{\Sigma}_7$:



with $\mathrm{pre}(\rho) = \mathbf{tr}$, $\mathrm{pre}(\sigma) = \mathsf{H}$, and $\mathrm{pre}(\tau) = \mathsf{T}$. As illustrated, the set Γ is the entire set Σ_7 of simple actions. More generally, the general shape of $\mathbf{\Sigma}_7$ is the frame for an action which has three possibilities. A learns which one happens, and B merely learns that one of the three happens. Further, these two aspects are common knowledge. We omit the calculation that shows that $\mathbf{S}_1 \otimes \mathbf{\Sigma}_7$ is the model drawn in Scenario 7.

Example 3.12 The program model employed in Scenario 8 is $\mathbf{\Sigma}_8$, shown below:



Again we take $\mathrm{pre}(\rho) = \mathbf{tr}$, $\mathrm{pre}(\sigma) = \mathsf{H}$, and $\mathrm{pre}(\tau) = \mathsf{T}$. The difference between this and $\mathbf{\Sigma}_7$ above, is that instead of $\Gamma = \Sigma_7$, we take Γ to be $\{s\}$. Then $\mathbf{S}_1 \otimes \mathbf{\Sigma}_8 \cong \mathbf{S}_8$.

3.6 Operations on program models

1 and 0 We define program models $\mathbf{1}$ and $\mathbf{0}$ as follows: $\mathbf{1}$ is a one-action set $\{\sigma\}$ with $\sigma \xrightarrow{A} \sigma$ for all A, $\mathrm{P{\scriptstyle RE}}(\sigma) = \mathbf{tr}$, and with distinguished set $\{\sigma\}$. The point here is that the update induced by this program model is exactly the update $\mathbf{1}$ from Section 2.3. We purposely use the same notation. Similarly, we let $\mathbf{0}$ be the empty program model. Then its induced update is what we called $\mathbf{0}$ in Section 2.3.

Sequential composition In all settings involving "actions" in some sense or other, sequential composition is a natural operation. In our setting, we would like to define a composition operation on program models, corresponding to the sequential composition of updates. Here is the relevant definition.

Let $\mathbf{\Sigma} = (\Sigma, \xrightarrow{A}, \mathrm{pre}_\Sigma, \Gamma_{\mathbf{\Sigma}})$ and $\mathbf{\Delta} = (\Delta, \xrightarrow{A}, \mathrm{pre}_\Delta, \Gamma_{\mathbf{\Delta}})$ be program models. We define the *composition*

$$\mathbf{\Sigma}; \mathbf{\Delta} \quad = \quad (\Sigma \times \Delta, \xrightarrow{A}, \mathrm{pre}_{\Sigma;\Delta}, \Gamma_{\Sigma;\Delta})$$

to be the following program model:

1. $\Sigma \times \Delta$ is the cartesian product of the sets Σ and Δ.

2. $\xrightarrow{A}$ in the composition $\mathbf{\Sigma}; \mathbf{\Delta}$ is the family of product relations, in the natural way:

$$(\sigma, \delta) \xrightarrow{A} (\sigma', \delta') \quad \text{iff} \quad \sigma \xrightarrow{A} \sigma' \text{ and } \delta \xrightarrow{A} \delta'.$$

3. $\mathrm{pre}_{\Sigma;\Delta}(\sigma, \delta) = \langle(\mathbf{\Sigma}, \sigma)\rangle \mathrm{pre}_{\Delta}(\delta)$.

4. $\Gamma_{\Sigma;\Delta} = \Gamma_{\mathbf{\Sigma}} \times \Gamma_{\mathbf{\Delta}}$.

In the definition of pre, $(\mathbf{\Sigma}, \sigma)$ is an abbreviation for the update $(\mathbf{\Sigma}, \{\sigma\})$, as defined in Section 3.4, i.e., the update induced by the program model $(\mathbf{\Sigma}, \{\sigma\})$.

Example 3.13 This example constructs a program model for lying, as we find in Scenario 6. Lying in our subject cannot be taken to be simply a case of private or public announcement: this will not work out. In our given situation, B simply knows that A doesn't know the side of the coin, and hence cannot accept any lying announcement that would claim such knowledge. One way to make sense of the action of A (successfully) lying to B is to assume that, *first, before the lying*, a suspicion was aroused in B that A might have privately learnt (e.g., by opening the box, or been told) which side of the coin was lying up; then second, that B subsequently receives an untruthful announcement that A knows the coin is lying heads up, an announcement which is known to be false by A herself (but which is believable, and now believed, by B). Obviously, we cannot express things about past actions in our logic, so we have to start right at the beginning, before the lying announcement is sent, and capture the whole action of successful lying as a sequential composition of two actions: B's suspicion of A's private learning, followed by B's receiving (and believing) the lying announcement. This is what we shall do here.[4] Let φ be $\neg\Box_A \mathsf{H}$ and let ψ be $\mathsf{H} \wedge \Box_A \mathsf{H}$. Let $\mathbf{\Sigma}_6$ be

$$\begin{tikzcd} \boxed{\boxed{\varphi}} \arrow[loop left, "A"] \arrow[r, "B"] & \boxed{\psi} \arrow[loop right, "A,B"] \end{tikzcd}$$

The idea is that B "learns" a false statement, namely that A knows the state of the coin. Further, we saw $\mathbf{\Sigma}_8$ in Example 3.12. We consider $\mathbf{\Sigma}_8; \mathbf{\Sigma}_6$. One can check using Proposition 3.14 that $\mathbf{S}_1 \otimes (\mathbf{\Sigma}_8; \mathbf{\Sigma}_6) \cong (\mathbf{S}_1 \otimes \mathbf{\Sigma}_8) \otimes \mathbf{\Sigma}_6 \cong \mathbf{S}_8 \otimes \mathbf{\Sigma}_6 \cong \mathbf{S}_6$. In addition, one can calculate $\mathbf{\Sigma}_8; \mathbf{\Sigma}_6$ explicitly to see what a reasonable one-step program for lying would be. The interesting point is that its preconditions are complex sentences having to do with actions in our language.

Disjoint union If $\mathbf{\Sigma} = (\Sigma, \xrightarrow{A}, \mathrm{pre}_{\Sigma}, \Gamma_{\mathbf{\Sigma}})$ and $\mathbf{\Delta} = (\Delta, \xrightarrow{A}, \mathrm{pre}_{\Delta}, \Gamma_{\mathbf{\Delta}})$, we take $\mathbf{\Sigma} \sqcup \mathbf{\Delta}$ to be the disjoint union of the models, with union of the distinguished actions. The intended meaning is the *non-deterministic choice* between the programs represented by $\mathbf{\Sigma}$ and $\mathbf{\Delta}$. Here is the definition in more detail, generalized to arbitrary (possibly infinite) disjoint unions: let $\{\mathbf{\Sigma}_i\}_{i \in I}$ be a family of program models, with $\mathbf{\Sigma}_i = (\Sigma_i, \xrightarrow{A}_i, \mathrm{pre}_i, \Gamma_i)$; we define their *(disjoint) union*

$$\bigsqcup_{i \in I} \mathbf{\Sigma}_i \quad = \quad \Big(\bigsqcup_{i \in I} \Sigma_i, \xrightarrow{A}, \mathrm{pre}, \Gamma\Big)$$

to be the model given by:

[4] In Section 4.5, we shall consider a slightly simpler model of lying.

1. $\bigsqcup_{i\in I}\Sigma_i$ is $\bigcup_{i\in I}(\Sigma_i\times\{i\})$, the disjoint union of the sets Σ_i.

2. $(\sigma,i)\xrightarrow{A}(\tau,j)$ iff $i=j$ and $\sigma\xrightarrow{A}{}_i\tau$.

3. $\mathrm{pre}(\sigma,i)=\mathrm{pre}_i(\sigma)$.

4. $\Gamma=\bigcup_{i\in I}(\Gamma_i\times\{i\})$.

Iteration Finally, we define an iteration operation by $\mathbf{\Sigma}^*=\bigsqcup\{\mathbf{\Sigma}^n : n\in N\}$. Here $\mathbf{\Sigma}^0=\mathbf{1}$, and $\mathbf{\Sigma}^{n+1}=\mathbf{\Sigma}^n;\mathbf{\Sigma}$.

We conclude by verifying that our definition of the operations on program models are correct in the sense that they are faithful to the corresponding operations on updates from Section 2.3.

Proposition 3.14 *The update induced by a composition of program models is the composition of the induced updates. Similarly for sums and iteration, mutatis mutandis.*

Proof Let $(\mathbf{\Sigma},\Gamma_{\mathbf{\Sigma}})$ and $(\mathbf{\Delta},\Gamma_{\mathbf{\Delta}})$ be program models. We denote by $\mathbf{r}$ the update induced by $(\mathbf{\Sigma},\Gamma_{\mathbf{\Sigma}})$, by $\mathbf{s}$ the update induced by $(\mathbf{\Delta},\Gamma_{\mathbf{\Delta}})$, and by $\mathbf{t}$ the update induced by $(\mathbf{\Sigma},\Gamma_{\mathbf{\Sigma}});(\mathbf{\Delta},\Gamma_{\mathbf{\Delta}})$. We need to prove that $\mathbf{r};\mathbf{s}=\mathbf{t}$. Let $\mathbf{S}=(S,\xrightarrow{A}_{\mathbf{S}},[\![.]\!]_{\mathbf{S}})$ be a state model. Recall that

$$\mathbf{S}(\mathbf{r};\mathbf{s}) \quad=\quad \mathbf{S}(\mathbf{r})(\mathbf{s}) \quad=\quad (\mathbf{S}\otimes(\mathbf{\Sigma},\Gamma_{\mathbf{\Sigma}}))\otimes(\mathbf{\Delta},\Gamma_{\mathbf{\Delta}}).$$

We claim that this is isomorphic to $\mathbf{S}\otimes(\mathbf{\Sigma};\mathbf{\Delta},\Gamma_{\mathbf{\Sigma};\mathbf{\Delta}})$, and indeed the isomorphism is $(s,(\sigma,\delta))\mapsto((s,\sigma),\delta)$. We check that $(s,(\sigma,\delta))\in\mathbf{S}\otimes(\mathbf{\Sigma};\mathbf{\Delta})$ iff $((s,\sigma),\delta)\in(\mathbf{S}\otimes\mathbf{\Sigma})\otimes\mathbf{\Delta}$. Indeed, the following are equivalent:

1. $(s,(\sigma,\delta))\in\mathbf{S}\otimes(\mathbf{\Sigma};\mathbf{\Delta})$.

2. $s\in\|\mathrm{pre}_{\Sigma;\Delta}(\sigma,\delta)\|_{\mathbf{S}}$.

3. $s\in\|\langle(\mathbf{\Sigma},\sigma)\rangle\mathrm{pre}_{\Delta}(\delta)\|_{\mathbf{S}}$.

4. $(s,\sigma)\in\mathbf{S}\otimes\mathbf{\Sigma}$ and $(s,\sigma)\in\|\mathrm{pre}_{\mathbf{\Delta}}(\delta)\|_{\mathbf{S}\otimes\mathbf{\Sigma}}$.

5. $((s,\sigma),\delta)\in(\mathbf{S}\otimes\mathbf{\Sigma})\otimes\mathbf{\Delta}$.

The rest of verification of isomorphism is fairly direct.

We also need to check that $\mathbf{t}_{\mathbf{S}}$ and $(\mathbf{r};\mathbf{s})_{\mathbf{S}}$ are related by the isomorphism. Now

$$\mathbf{t}_{\mathbf{S}} \quad=\quad \{(s,(s,(\sigma,\delta)))\in S\otimes(\Sigma;\Delta):\sigma\in\Gamma_{\mathbf{\Sigma}},\delta\in\Gamma_{\mathbf{\Delta}}\}.$$

Recall that $(\mathbf{r};\mathbf{s})_{\mathbf{S}}=\mathbf{r}_{\mathbf{S}};\mathbf{s}_{\mathbf{S}(\mathbf{r})}$ and that this is a relational composition in left-to-right order. And indeed,

$$\begin{array}{lcl}\mathbf{r}_{\mathbf{S}} & = & \{(s,(s,\sigma)):(s,\sigma)\in S\otimes\Sigma,\sigma\in\Gamma_{\mathbf{\Sigma}}\}\\ \mathbf{s}_{\mathbf{S}(\mathbf{r})} & = & \{((s,\sigma),((s,\sigma),\delta))\in\mathbf{S}(\mathbf{r})\otimes\Delta:\delta\in\Gamma_{\mathbf{\Delta}}\}.\end{array}$$

This completes the proof for composition. We omit the proofs for sums and iteration. $\dashv$

Endnotes The work of this section explains of how complex representations of naturally occurring scenarios may be computed from state models before the scenario and program models. Indeed, this is one of our main points. There are precursors to our work in special cases, most notably Hans van Ditmarsch's dissertation [23]. That work is about examples like $\mathbf{\Sigma}_7$, where some agent or set of agents knows that the current action belongs to some set, but does not know which action it is. But to our knowledge, the work in this section is the first time that anything like this has been obtained in general.

We have taught the material in this section in several courses at different levels. Our experience is that the material here is highly appealing to students and makes the case for formal representations in the first place (for students not familiar with formal logic) and for the kind of technical work that we pursue in this area (for those who are).

The idea of representing epistemic actions as Kripke models (or variants of them) was first presented in our earlier paper with S. Solecki [6]. However, the proposal of that paper was to employ Kripke models in the syntax of a logical language *directly.* Many readers have felt this to be confusing, since the resulting syntax looked as if it depended on the semantics. Proposals to improve the language were developed in several papers of Baltag [1, 2, 3, 5]. The logics of these papers were more natural than those of [6]. What is new in this paper is the overall apparatus of action signatures, epistemic actions, etc. We present what we hope is a natural syntax, and at the same time we have programs as syntactic entities in our logical languages (see Section 4 just below) linked to program models in the semantics.

4 Logical languages based on action signatures

This section presents the centerpiece of our formal work, the logical systems corresponding to *types of epistemic action.* Our overall goal is to abstract from the propositional language earlier, in a way which allows us to fix only the *epistemic structure* of the desired actions, and vary their preconditions.

4.1 Action signatures

We have presented many examples of the update product in Section 3.5. These allow us to represent many natural updates, and this is one of our goals in this paper. But the structures which we have seen so far are not sufficient to get logical languages incorporating updates. For example, we have in Example 3.7 a program model that represents a private announcement to the agent A that a proposition H happens, and this takes place in a way that B learns nothing. The picture of this was

$$
\begin{tikzcd}
\boxed{\boxed{\mathsf{H}}} \arrow[loop left, "A"] \arrow[r, "B"] & \boxed{\mathbf{tr}} \arrow[loop right, "A,B"]
\end{tikzcd}
$$

What we want to do now is to vary things a bit, and then to *abstract* them. For example, suppose we want to announce a different proposition to A, say ψ. We would use

$$
\begin{tikzcd}
\boxed{\boxed{\psi}} \arrow[loop left, "A"] \arrow[r, "B"] & \boxed{\mathbf{tr}} \arrow[loop right, "A,B"]
\end{tikzcd}
$$

Varying the proposition φ, all announcements of this kind could be thought as actions *of the same type*. We could then represent the *type* of the action by the following picture:

$$\begin{tikzcd} \boxed{\boxed{\ }} \arrow[loop left, "A"] \arrow[r, "B"] & \boxed{\mathbf{tr}} \arrow[loop right, "A,B"] \end{tikzcd}$$

And the previous representations include the information that what *actually* happens is what A hears. To vary this, we need only change which world is designated by the doubled border. We could switch things, or double neither or both worlds. So we obtain a structure consisting of two *action types*:

$$\begin{tikzcd} \boxed{\ } \arrow[loop left, "A"] \arrow[r, "B"] & \boxed{\mathbf{tr}} \arrow[loop right, "A,B"] \end{tikzcd}$$

The oval on the left represents the type Pri^A of *a fully private announcement to agent* A, while the oval on the right simply represents the type of a *skip* action (or of an empty, or trivial, public announcement). By inserting any given proposition ψ into the oval depicting the action type Pri^A, we can obtain specific private announcements $Pri^A\psi$, as depicted above. (There is no reason to insert any proposition into the right oval, since this comes already with its own precondition $\mathbf{tr}$: this means that the type of a *skip* action uniquely determines the corresponding action *skip*, since it contains all the information needed to specify this action.)

Another example would be the case in which we announce ψ to A in such a way that B is misled into believing $\neg\psi$ (and is also misled into believing that everyone learns $\neg\psi$). Now we use

$$\begin{tikzcd} \boxed{\boxed{\psi}} \arrow[loop left, "A"] \arrow[r, "B"] & \boxed{\neg\psi} \arrow[loop right, "A,B"] \end{tikzcd}$$

In itself, this is not general enough to give rise to an action type in our sense. But we can more generally consider the case in which ψ_1 is announced to A in such a way that B thinks that some other proposition ψ_2 is publicly announced:

$$\begin{tikzcd} \boxed{\boxed{\psi_1}} \arrow[loop left, "A"] \arrow[r, "B"] & \boxed{\psi_2} \arrow[loop right, "A,B"] \end{tikzcd}$$

By abstracting from the specific propositions, we obtain the following structure consisting of two action types:

$$\begin{tikzcd} \boxed{\ } \arrow[loop left, "A"] \arrow[r, "B"] & \boxed{\ } \arrow[loop right, "A,B"] \end{tikzcd}$$

Observe that if we are given now a sequence of two propositions (ψ_1, ψ_2), we could use them to fill in the oval with preconditions in two possible ways (depending on which proposition goes into the left oval). So, in order to uniquely determine how an action type will generate specific announcements, we need an *enumeration without repetition* of all the action types in the structure which do not come equipped with trivial preconditions (i.e., all the empty ovals in the diagram, since we assume the others have $\mathbf{tr}$ inside).

So at this point, we can see how to abstract things, leading to the following definition.

Definition An *action signature* is a structure

$$\mathbf{\Sigma} \quad = \quad (\Sigma, \overset{\circ}{\xrightarrow{A}}, (\sigma_1, \sigma_2, \ldots, \sigma_n))$$

where $\mathbf{\Sigma} = (\Sigma, \overset{\circ}{\xrightarrow{A}})$ is a finite Kripke frame, and $\sigma_1, \sigma_2, \ldots, \sigma_n$ is a designated *listing* of a subset of Σ without repetitions. We obviously must have $n \leq |\Sigma|$, but we allow the case $n = 0$ as well, which will produce an empty list. We call the elements of Σ *action types*, while the ones which are in the listing $(\sigma_1, \ldots, \sigma_n)$ will be called *non-trivial action types.*

The way this works for us is that an action signature together with an assignment of epistemic propositions to the non-trivial action types in Σ will give us a full-fledged action model. The trivial action types will describe actions which can always happen, so they will get assigned the trivial precondition **tr**, by default. And this is the exact sense in which the notion of an action signature is an abstraction of the notion of action model. We shall shortly use action signatures in constructing logical languages.

Examples 4.1 Here is a very simple action signature which we call $\mathbf{\Sigma}_{skip}$. Σ is a singleton $\{skip\}$, we put $skip \xrightarrow{A} skip$ for all agents A, and we take the *empty list* () of types as listing. So we have only one, trivial, type: *skip*. In a sense which we shall make clear later, this is an action in which "nothing happens", and moreover it is common knowledge that this is the case.

The type of *a public announcement* can simply be obtained by only changing our listing in $\mathbf{\Sigma}_{skip}$, to make the type non-trivial; we also change the name of the type to *Pub*, to distinguish it from the previous one. So the signature $\mathbf{\Sigma}_{\text{pub}}$ of *public announcements* has $\Sigma = \{Pub\}$, $Pub \xrightarrow{A} Pub$ for every agent A, and the listing is just (Pub). So *Pub* is the non-trivial type of a public announcement action. Note once again that we have not said *what* is being announced. We are purposely separating out the structure of the announcement from the content, and $\mathbf{\Sigma}_{Pub}$ is our model of the structure.

The next simplest action signature is the "test" signature $\mathbf{\Sigma}_?$. We take $\mathbf{\Sigma}_? = \{?, skip\}$, with the listing $(?)$. We also take $? \xrightarrow{A} skip$, and $skip \xrightarrow{A} skip$ for all A. So the test ? is the only non-trivial type of action here. This turns out to be a totally opaque form of test: φ is tested on the real world, but nobody knows this is happening: when it happening, all the agents think nothing (i.e., *skip*) is happening. The function of this type will be to generate tests $?\varphi$, which affect the states precisely in the way dynamic logic tests do.

For each set $\mathcal{B} \subseteq \mathcal{A}$ of agents, we define the action signature $Pri^{\mathcal{B}}$ of *completely private announcements to the group* $\mathcal{B}$. It has $\Sigma = \{Pri^{\mathcal{B}}, skip\}$; the listing is just $(Pri^{\mathcal{B}})$, which makes *skip* into a trivial type again; and we put $Pri^{\mathcal{B}} \xrightarrow{B} Pri^{\mathcal{B}}$ for all $B \in \mathcal{B}$, $Pri^{\mathcal{B}} \xrightarrow{C} skip$ for $C \notin \mathcal{B}$, and $skip \xrightarrow{A} skip$ for all agents A.

The action signature $Cka_k^{\mathcal{B}}$ is given by: $\Sigma = \{1, \ldots, k\}$; the listing is $(1, 2, \ldots, k)$, so all types are non-trivial. $i \xrightarrow{B} i$ for $i \leq k$ and $B \in \mathcal{B}$; and finally $i \xrightarrow{C} j$ for $i, j \leq k$ and $C \notin \mathcal{B}$. This action signature is called the signature of *common knowledge of alternatives for an announcement to the group* $\mathcal{B}$.

Signature-based program models Now that we have a general abstract notion, we introduce some notation to regain the earlier examples. Let $\mathbf{\Sigma}$ be a action signature, let $(\sigma_1, \ldots, \sigma_n$ be the corresponding listing of non-trivial types, let $\Gamma \subseteq \Sigma$, and let $\vec{\boldsymbol{\psi}} = \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n$ be a list of epistemic propositions. We obtain an epistemic program model $(\mathbf{\Sigma}, \Gamma)(\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n)$ in the following way:

1. The set of simple actions is Σ, and the accessibility relations are those given by the action signature.

sentences φ	true	p_i	$\neg\varphi$	$\varphi \wedge \psi$	$\Box_A \varphi$	$\Box^*_{\mathcal{B}}\varphi$	$[\pi]\varphi$
programs π	*skip*	*crash*	$\sigma\psi_1, \ldots, \psi_n$	$\pi \sqcup \rho$	$\pi ; \rho$	π^*	

$$
\begin{array}{lcl@{\qquad}lcl}
[\![\mathsf{true}]\!] & = & \mathbf{tr} & [\![\mathit{skip}]\!] & = & \mathbf{1} \\
[\![p]\!] & = & \boldsymbol{p} & [\![\mathit{crash}]\!] & = & \mathbf{0} \\
[\![\varphi \wedge \psi]\!] & = & [\![\varphi]\!] \wedge [\![\psi]\!] & [\![\sigma\psi_1, \ldots, \psi_n]\!] & = & (\boldsymbol{\Sigma}, \sigma, [\![\psi_1]\!], \ldots, [\![\psi_n]\!]) \\
[\![\neg\varphi]\!] & = & \neg[\![\varphi]\!] & [\![\pi ; \rho]\!] & = & [\![\pi]\!] ; [\![\rho]\!] \\
[\![\Box_A \varphi]\!] & = & \Box_A [\![\varphi]\!] & [\![\pi \sqcup \rho]\!] & = & [\![\pi]\!] \sqcup [\![\rho]\!] \\
[\![\Box^*_{\mathcal{B}} \varphi]\!] & = & \Box^*_{\mathcal{B}} [\![\varphi]\!] & [\![\pi^*]\!] & = & [\![\pi]\!]^* \\
[\![[\pi]\varphi]\!] & = & [[\![\pi]\!]][\![\varphi]\!] & & &
\end{array}
$$

Figure 7: The language $\mathcal{L}(\boldsymbol{\Sigma})$ and its semantics

2. For $j = 1, \ldots, n$, $\mathrm{pre}(\sigma_j) = \boldsymbol{\psi}_j$. We put $\mathrm{pre}(\sigma) = \mathbf{tr}$ for all the other (trivial) actions.

3. The set of distinguished actions is Γ.

In the special case that Γ is the singleton set $\{\sigma_i\}$, we write the resulting signature-based program model as $(\boldsymbol{\Sigma}, \sigma_i, \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n)$.

To summarize: *every action signature, set of distinguished action types in it, and corresponding tuple of epistemic propositions gives an epistemic program model in a canonical way.*

4.2 The syntax and semantics of $\mathcal{L}(\boldsymbol{\Sigma})$

Fix an action signature $\boldsymbol{\Sigma}$. We present in Figure 7 a logic $\mathcal{L}(\boldsymbol{\Sigma})$.

The first thing to notice about this is that for the first time in a great while, we have a genuine syntax. In this regard, note that n is fixed from $\boldsymbol{\Sigma}$; it is the length of the given listing $(\sigma_1, \ldots, \sigma_n)$. In the programs of the form $\sigma\psi_1, \ldots, \psi_n$ we have *sentences* $\vec{\psi}$ rather than *epistemic propositions* (which we had written using boldface letters $\boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n$ in Section 2.3). Also, the signature $\boldsymbol{\Sigma}$ figures into the semantics exactly in those programs $\sigma\psi_1, \ldots, \psi_n$; in those we require that $\sigma \in \Sigma$. The program model $(\boldsymbol{\Sigma}, \sigma, [\![\psi_1]\!], \ldots, [\![\psi_n]\!])$ is a signature-based program model as in the previous section.

The second thing to note is that, as in PDL, we have two sorts of syntactic objects: sentences and programs. We call programs of the form $\sigma\psi_1, \ldots, \psi_n$ *basic actions.* Note that they might not be "atomic" in the sense that the sentences ψ_j might themselves contain programs. The basic actions of the form $\sigma_i\psi_1, \ldots \psi_n$ (with $i \leq n$) are called *non-trivial*, since they are generated by non-trivial action types.

We use the following standard abbreviations: $\mathsf{false} = \neg\mathsf{true}$, $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$, $\Diamond_A \varphi = \neg\Box_A\neg\varphi$, $\Diamond^*_{\mathcal{B}}\varphi = \neg\Box^*_{\mathcal{B}}\varphi$, and $\langle\pi\rangle\varphi = \neg[\pi]\neg\varphi$.

The semantics defines two operations by simultaneous recursion on $\mathcal{L}(\boldsymbol{\Sigma})$:

1. $\varphi \mapsto [\![\varphi]\!]$, taking the sentences of $\mathcal{L}(\boldsymbol{\Sigma})$ into epistemic propositions; and

2. $\pi \mapsto [\![\pi]\!]$, taking the programs of $\mathcal{L}(\boldsymbol{\Sigma})$ into program models (and hence into induced updates).

The formal definition is given in Figure 7. The main thing to note is that with one key exception, the operations on the right-hand sides are immediate applications of our general definitions of the closure conditions on epistemic propositions from Section 2.1 and the operations on program models from Section 3.6. A good example to explain this is the clause for the semantics of sentences $[\alpha]\varphi$. Assuming that we have a program model $[\![\alpha]\!]$, we get an induced update in Section 3.4 which we again denote $[\![\alpha]\!]$. We also have an epistemic proposition $[\![\varphi]\!]$. We can therefore form the epistemic proposition $[[\![\alpha]\!]][\![\varphi]\!]$ (see equation (4) in Section 2.3). Note that we have overloaded the square bracket notation; this is intentional, and we have done the same with other notation as well.

Similarly, the semantics of *skip* and *crash* are the program models $\mathbf{1}$ and $\mathbf{0}$ of Section 3.6.

We also discuss the definition of the semantics for basic actions $\sigma\vec{\psi}$. This is precisely where the structure of the action signature $\mathbf{\Sigma}$ enters. For this, recall that we have general definition of a signature-based program model $(\mathbf{\Sigma}, \Gamma, \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_n)$, where $\Gamma \subseteq \Sigma$ and the $\boldsymbol{\psi}$'s are any epistemic propositions. What we have in the semantics of $\sigma\vec{\psi}$ is the special case of this where Γ is the singleton $\{\sigma\}$ and $\boldsymbol{\psi}_i$ is $[\![\psi_i]\!]$, a proposition which we will already have defined when we will come to define $[\![\sigma\vec{\psi}]\!]$.

At this point, it is probably good to go back to our earlier discussions in Section 2.1 of epistemic propositions and updates. What we have done overall is to give a fully syntactic presentation of languages of these epistemic propositions and updates. The constructions of the language correspond to the closure properties noted in Section 2.1. (To be sure, we have restricted to the finite case at several points because we are interested in a syntax, and at the same time we have re-introduced some infinitary aspects via the Kleene star.)

4.3 Epistemic program logics $\mathcal{L}(\mathcal{S})$

We generalize now our signature logics $\mathcal{L}(\mathbf{\Sigma})$ to *families* $\mathcal{S}$ of signatures, in order to define a general notion of epistemic program logics.

Logics generated by families of signatures Given a family $\mathcal{S}$ of signatures, we would like to combine all the logics $\{\mathcal{L}(\mathbf{\Sigma})\}_{\mathbf{\Sigma}\in\mathcal{S}}$ into a single logic. Let us assume the signatures $\mathbf{\Sigma} \in \mathcal{S}$ are mutually disjoint (otherwise, just choose mutually disjoint copies of these signatures). We define *the logic* $\mathcal{L}(\mathcal{S})$ *generated by the family* $\mathcal{S}$ in the following way: the syntax is defined by taking the same definition we had in Figure 7 for the syntax of $L(\mathbf{\Sigma})$, but in which on the side of the programs we take instead as *basic actions* all expressions of the form

$$\sigma\psi_1, \ldots, \psi_n$$

where $\sigma \in \Sigma$, for some arbitrary signature $\mathbf{\Sigma} \in \mathcal{S}$, and n is the length of the listing of non-trivial action types of $\mathbf{\Sigma}$. The semantics is again given by the same definition as in Figure 7, but in which the clause about $\sigma\psi_1, \ldots, \psi_n$ refers to the appropriate signature: for every $\mathbf{\Sigma} \in \mathcal{S}$, every $\sigma \in \Sigma$, if n is the length of the listing of $\mathbf{\Sigma}$, then

$$[\![\sigma\psi_1, \ldots, \psi_n]\!] \quad = \quad (\mathbf{\Sigma}, \sigma, [\![\psi_1]\!], \ldots, [\![\psi_n]\!]).$$

Example 4.2 *This example constructs the logic of all epistemic programs. Take the family*

$$\mathcal{S} \quad = \quad \{\mathbf{\Sigma} : \mathbf{\Sigma} \textit{ is a finite signature } \}$$

of all finite signatures[5]*. The logic* $\mathcal{L}(\mathcal{S})$ *will be called* the logic of all epistemic programs.

Preservation of bisimulation and atomic propositions We note two basic facts about the interpretations of sentences and programs in the logics $\mathcal{L}(\mathcal{S})$.

Proposition 4.3 *The interpretations of the sentences and programs of* $\mathcal{L}(\mathcal{S})$ *preserve bisimulation.*

Proof By induction on $\mathcal{L}(\mathcal{S})$, using Propositions 2.5 and 3.5. $\dashv$

Proposition 4.4 *The interpretations of programs of* $\mathcal{L}(\mathcal{S})$ *preserve atomic sentences in the sense that if* $s\ [\![\pi]\!]_{\mathbf{S}}\ t$*, then for all atomic sentences* p*,* $s \in \|p\|_{\mathbf{S}}$ *iff* $t \in \|p\|_{\mathbf{S}([\![\pi]\!])}$.

Proof By induction on π. $\dashv$

4.4 Formalization of the target logics in terms of signatures

We formalize now the target logics of Section 2.4 as epistemic program logics $\mathcal{L}(\mathcal{S})$. We use the action signatures of Examples 4.1 and the notation from there.

The logic of public announcements This is formalized as $\mathcal{L}(\mathbf{\Sigma}_{Pub})$. We have sentences $[Pub\ \varphi]\psi$, just as we described in Section 2.4. Note that $\mathcal{L}(\mathbf{\Sigma}_{Pub})$ allows announcements inside announcements. If φ, ψ, and χ are sentences, then so is $\langle Pub\ [Pub\ \varphi]\psi\rangle\chi$.

We check that $\mathcal{L}(\mathbf{\Sigma}_{Pub})$ is a good formalization of the logic of public announcements. Fix a sentence φ of $\mathcal{L}(\mathbf{\Sigma}_{Pub})$ and a state model $\mathbf{S}$. We calculate:

$$\begin{array}{rcl}\mathbf{S}([\![Pub\ \varphi]\!]) & = & \mathbf{S}(\mathbf{\Sigma}_{Pub}, Pub, [\![\varphi]\!]) \\ & = & \mathbf{S} \otimes (\mathbf{\Sigma}_{Pub}, Pub, [\![\varphi]\!]) \\ & = & \{(s, Pub) : s \in [\![\varphi]\!]_{\mathbf{S}}\}\end{array}$$

The state model has the structure given earlier in terms of the update product operation. The update relation $[\![Pub\ \varphi]\!]_{\mathbf{S}}$ relates s to (s, Pub) whenever the latter belongs to the state model $\mathbf{S}([\![Pub\ \varphi]\!])$. The model itself is isomorphic to the sub-state-model of $\mathbf{S}$ induced by $\{s \in S : s \in [\![\varphi]\!]_{\mathbf{S}}\}$. Under this isomorphism, the update relation is then the inverse of inclusion. This is just how the action of public announcement was described when we first encountered it, in Example 2.1 of Section 2.3.

Test-only PDL was introduced in Section 2.6. Recall that this is PDL built over the empty set of atomic actions. Although it was not one of the target languages of Section 2.4, it will be instructive to see how it is formalized in our setting. Recall that test-only PDL has actions of the form $?\varphi$. We want to use our action signature $\mathbf{\Sigma}_{?}$. The action types of it are ? and *skip*, only the first one being non-trivial: $n = 1$. So technically we have sentences of the following forms:

$$[?\ \varphi]\chi \quad \text{and } [skip\ \varphi]\chi \tag{9}$$

[5] To make this into a *set*, instead of a proper class, we really mean to take all finite signatures whose action types are natural numbers, and then take the disjoint union of this countable set of finite signatures.

Let us study the semantics of the basic actions ? φ and *skip* φ. Fix a state model $\mathbf{S}$. We calculate:

$$\begin{array}{lcl} \mathbf{S}([\![?\ \varphi]\!]) & = & \mathbf{S}(\mathbf{\Sigma}_?, ?, [\![\varphi]\!]) \\ & = & \mathbf{S} \otimes (\mathbf{\Sigma}_?, ?, [\![\varphi]\!]) \\ & = & \{(s, ?) : s \in [\![\varphi]\!]_{\mathbf{S}}\} \cup \{(s, \mathit{skip}) : s \in S\} \end{array}$$

The structure of $\mathbf{S}([\![?\ \varphi]\!])$ is that for each A, if $s \xrightarrow{A} t$ in $\mathbf{S}$, then $(s, ?) \xrightarrow{A} (t, \mathit{skip})$. Also $(s, \mathit{skip}) \xrightarrow{A} (t, \mathit{skip})$ under the same condition, and there are no other arrows. The update relation $[\![?\ \varphi]\!]_{\mathbf{S}}$ relates s to $(s, ?)$ whenever the latter belongs to the updated structure. Overall, this update is isomorphic to what we described in Example 2.2 of Section 2.3.

Turning to the update map of *skip* φ, we again fix $\mathbf{S}$. The model $\mathbf{S}([\![\mathit{skip}\ \varphi]\!])$ is again literally the same as what we calculated above. However, the update relation $[\![\mathit{skip}\ \varphi]\!]_{\mathbf{S}}$ now relates each $s \in S$ to the pair (s, skip). This relation is a bisimulation. We shall formulate a notion of action equivalence later, and it will turn out that the update $[\![\mathit{skip}\ \varphi]\!]$ is equivalent to $\mathbf{1}$. For now, we can also consider the semantics of sentences of the form $[\mathit{skip}\ \varphi]\psi$. We have

$$\begin{array}{lcl} [\![[\mathit{skip}\ \varphi]\psi]\!]_{\mathbf{S}} & = & \{s \in S : \text{if } s\ [\![\mathit{skip}\ \varphi]\!]_{\mathbf{S}}\ t, \text{ then } t \in [\![\psi]\!]_{\mathbf{S}([\![\mathit{skip}\ \varphi]\!])}\} \\ & = & \{s \in S : (s, \mathit{skip}) \in [\![\psi]\!]_{\mathbf{S}([\![\mathit{skip}\ \varphi]\!])}\} \\ & = & \{s \in S : s, \in [\![\psi]\!]_{\mathbf{S}}\} \end{array}$$

This last equality is by Proposition 4.3 on bisimulation preservation. The upshot is that $[\![[\mathit{skip}\ \varphi]\psi]\!]_{\mathbf{S}} = [\![\psi]\!]_{\mathbf{S}}$. So for this reason, we might as well identify the sentences $[\mathit{skip}\ \varphi]\psi$ and ψ. Or to put things differently, we might as well identify the basic action *skip* φ and (the constant of $\mathcal{L}(\mathbf{\Sigma}_?)$) *skip*. Since we are already modifying the syntax, we might also abbreviate $[\mathit{skip}\varphi]\psi$ to ψ. Doing all this leads to a language which is exactly test-only PDL as we had it in Section 2.6, and our semantics there agrees with what we have calculated in the present discussion.

In conclusion, *test-only PDL is equivalent to the logic* $\mathcal{L}_0(\mathbf{\Sigma}_?)$*; i.e., the* $\Box^*$*-free fragment of* $\mathcal{L}(\mathbf{\Sigma}_?)$.

The logic of totally private announcements Let *Pri* be the family

$$\mathit{Pri} \quad = \quad \{\mathit{Pri}^{\mathcal{B}} : \emptyset \neq \mathcal{B} \subseteq \mathcal{A}\}$$

of all signatures of totally private announcements to non-empty groups of agents (as introduced in Examples 4.1). Then $\mathcal{L}(\mathit{Pri})$ formalizes one of our target logics: the logic of totally private announcements. For example, in the case of $\mathcal{A} = \{A, B\}$, $\mathcal{L}(\mathit{Pri})$ will have basic actions of the forms: $\mathit{Pri}^A\ \varphi$, $\mathit{Pri}^B\ \varphi$, $\mathit{Pri}^{A,B}\ \varphi$, $\mathit{skip}^A\ \varphi$, $\mathit{skip}^B\ \varphi$, $\mathit{skip}^{A,B}\ \varphi$. As before, we may as well identify all the programs $\mathit{skip}^X\ \varphi$ with *skip*, and abbreviate $[\mathit{skip}^X]\psi$ to ψ.

The logic of common knowledge of alternatives can be formalized as $\mathcal{L}(\mathit{Cka})$, where

$$\mathit{Cka} \quad = \quad \{\mathit{Cka}_k^{\mathcal{B}} : \emptyset \neq \mathcal{B} \subseteq \mathcal{A}, 1 \leq k\}.$$

4.5 Other logics

Logics based on frame conditions Many other types of logics are easily representable in our framework. For example, consider the logic of all S4 programs. To formalize this, we need only consider the disjoint union of all finite action signatures whose underlying accessibility relations are preorders. For the logic of S5 programs, we change preorders to equivalence relations. Another important class is given by the K45 axioms of modal logic. These systems are particularly important because the K45 and S5 conditions are so prominent in epistemic logic.

Announcements by particular agents Our modeling of the notion of the *public announcement that* φ is impersonal in the sense that the announcement does not come from anywhere in particular. It might be best understood as coming externally, as if someone shouted φ into the room where the agents were standing.

We also want a notion of the *public announcement by* A *of* φ. We shall write this as $Pub^A\varphi$. For this, we identify $Pub^A\varphi$ with the (externally-made) public announcement that φ holds and A knows this. This identification does not represent the fact that A intends to inform the others of φ. But as we know, intentions are not modeled at all in our system. We claim, however, that on a purely propositional level, the identification works. And using it, we can represent announcements by A. One way to do this is via abbreviation: we take $\langle Pub^A\ \varphi\rangle\psi$ to be an abbreviation for $\langle Pub\ \varphi \wedge \Box_A\varphi\rangle\psi$. (A different way to formalize $Pub^A\ \varphi$ would be to use a special signature constructed just for that purpose. But the discussion here shows that there is no need to do this. One can use Σ_{Pub}.)

Lying We can also represent misleading epistemic actions, such as lying. Again, we want to fix an agent A and then represent the action of A *(successfully) lying that* φ to the other agents. To all those others, this action should be the same thing as an announcement of φ by A. But to say that A *lies* about φ, we want to assume that φ is actually false. Further, we want to assume that A moreover knows that φ is false. (For if φ just happened to be false when A said φ, we would not really want to call that "lying.")

The technical details on the representation go as follows. We take a signature Σ^A_{Lie} given as follows.

$$\Sigma^A_{Lie} \quad = \quad \{Secret^A, Pub^A\}.$$

We take $(Pub^A, Secret^A)$ as our non-repetitive list of types. The structure is given by taking $Secret^A \xrightarrow{A} Secret^A$; for $B \neq A$, $Secret^A \xrightarrow{B} Pub^A$; finally, for all B, $Pub^A \xrightarrow{B} Pub^A$.

$\mathcal{L}(\mathbf{\Sigma}^A_{Lie})$ contains sentences like $[Secret^A\ \varphi, \psi]\chi$. The extra argument ψ is a kind of secret condition. And we can use $[Lie^A\varphi]\chi$ as an abbreviation of

$$[Secret^A\ \varphi \wedge \Box_A\varphi, \neg\varphi \wedge \Box_A\neg\varphi]\chi.$$

That is, for A to lie about φ there is a condition that $\neg\varphi \wedge \Box_A\neg\varphi$. But the other agents neither need to know this ahead of time nor do they in any sense "learn" this from the announcement. Indeed, for the other agents, $Lie^A\varphi$ is just like a truthful public announcement by A.

As with private announcements, we take the family of signatures $\{\Sigma^A_{Lie} : A \in \mathcal{A}\}$. This family then generates a program logic. In this logic we have actions which represent lying on the part of any agent, not just one fixed agent.

Other effects: wiretapping, paranoia etc. It is possible to model scenarios where one player believes a communication to be private while in reality a second player intercepts the communication. We can also represent gratuitous suspicion ("paranoia"): maybe no "real" action has taken place, except that some people start suspecting some action (e.g., some private communication) has taken place. With these and other effects, the problem is not so much deciding how to model them. Once one has clear intuitions about a social scenario, it is not hard to do the modeling. The real issue in their application seems to be that in complex social situations, our intuitions are not always clear. There is no getting around this, and technical solutions are of limited value for conceptual problems.

Endnote This section is the centerpiece of this paper, and all of the work in it is new. We make a few points about the history of this approach. Our earlier paper with S. Solecki [6] worked with a syntax that employed structured objects as actions. In fact, the actions were Kripke models themselves. This type of syntax also is used in other work such as van Ditmarsch et al [25]. But the use of structured objects in syntax is felt by many readers to be awkward; see, e.g., remarks in chapter 4 of Kooi's thesis [14]. We disagree with the assertion that our earlier syntax blurs the distinction between syntax and semantics. But in order to make the subject more attractive, we have worked out other approaches to the syntax. Baltag [2, 3, 5] developed logical systems that streamline the syntax using a repertoire of program operations, such as *learning a program* and variable-binding operators. This paper is the first one to formulate program logics in terms of action signatures.

5 Logical systems

We write $\models \varphi$ to mean that for all state models $\mathbf{S}$ and all $s \in S$, $s \in [\![\varphi]\!]_{\mathbf{S}}$. In this case, we say that φ is *valid*. In this section, we fix an arbitrary family $\mathcal{S}$ of mutually disjoint action signatures, and we consider the generated logics. We present a *sound* proof system for the validities in $\mathcal{L}(\mathcal{S})$, and *sound and complete* proof systems for two important sublogics: the iteration-free fragment $\mathcal{L}_1(\mathcal{S})$ and the logic $\mathcal{L}_0(\mathcal{S})$ obtained by deleting both iteration and common knowledge operators. In particular, our results apply to the logics $\mathcal{L}(\mathbf{\Sigma})$, $\mathcal{L}_1(\mathbf{\Sigma})$ and $\mathcal{L}_0(\mathbf{\Sigma})$ given by only one signature. However, the soundness/completeness proofs will appear in [7]. So the point of this section is to just state clearly what the logical system is, and to illustrate its use.

Sublanguages We are of course interested in the languages $\mathcal{L}(\mathcal{S})$, but we also consider sublanguages $\mathcal{L}_0(\mathcal{S})$ and $\mathcal{L}_1(\mathcal{S})$. Recall that $\mathcal{L}_1(\mathcal{S})$ is the fragment without the action iteration construct π^*. $\mathcal{L}_0(\mathcal{S})$ is the fragment without π^* and $\Box^*_{\mathcal{B}}$. It turns out that $\mathcal{L}_0(\mathbf{\Sigma})$ is the easiest to study: it is of the same expressive power as ordinary multi-modal logic. On the other hand, the full logic $\mathcal{L}(\mathcal{S})$ is in general undecidable : indeed, even if we take a family consisting of only one signature, of public announcements *Pub*, the corresponding logic $\mathcal{L}(Pub)$ is undecidable (see [16]). $\mathcal{L}_1(\mathbf{\Sigma})$ is decidable and we have a complete axiom system for it (see [7]).

In Figure 8 below we present a logic for $\mathcal{L}(\mathcal{S})$. We write $\vdash \varphi$ if φ can be obtained from the axioms of the system using its inference rules. We often omit the turnstile $\vdash$ when it is clear from the context. Our presentation of the proof system uses the meta-syntactical notations associated with the notion of *canonical action model* $\mathbf{\Omega}$ (to be introduced below in Section 5.1).

This could have been avoided only at the cost of complicating the presentation. We chose the simplest version, and so our logical system, as given in Figure 8, can be fully understood only after reading Section 5.1.

Axioms Most of the system will be quite standard from modal logic. The Action Axioms are new, however. These include the Atomic Permanence axiom; note that in this axiom p is an atomic sentence. The axiom says that announcements do not change the brute fact of whether or not p holds. This axiom reflects the fact that our actions do not change any kind of local state.

The Action-Knowledge Axiom gives a criterion for knowledge after an action. For non-trivial basic actions $\sigma_i\vec{\psi}$ (induced by a non-trivial action type $\sigma_i \in \Sigma$, for some signature $\Sigma \in \mathcal{S}$), this axiom states that

$$\vdash [\sigma_i\vec{\psi}]\Box_A\varphi \leftrightarrow (\psi_i \to \bigwedge\{\Box_A[\sigma_j\vec{\psi}]\varphi : \sigma_i \xrightarrow{A} \sigma_j \text{ in } \mathbf{\Sigma}\}) \tag{10}$$

In words, two sentences are equivalent: first, an agent A will come to know φ after a basic action $\sigma_i\vec{\psi}$ is executed; and second, whenever this action $\sigma_i\vec{\psi}$ is executable (i.e., its precondition ψ_i holds), A knows (already before this action) that φ will come to be true after every action that A considers as possibly happening (whenever $\sigma_i\vec{\psi}$ is in fact happening). This axiom should be compared with the Ramsey axiom in conditional logic. One should also study the special case of it for the logic of public announcements in Section 5.3.

The Action Rule then gives a necessary criterion for *common* knowledge after a simple action. (Simple actions are defined below in Section 5.1. They include the actions of the form $\sigma\vec{\psi}$.) Since common knowledge is formalized by the $\Box^*_{\mathcal{B}}$ construct, this rule is a kind of induction rule. (The sentences χ_β play the role of strong induction hypotheses.) (For the induction rule for common knowledge assertions without actions, see Lemma 5.5.)

5.1 The canonical action model $\mathbf{\Omega}$

Recall that we defined action models and program models in Sections 3.1 and 3.2, respectively. At this point, we define a new action model called the *canonical action model* $\mathbf{\Omega}$ of a language $\mathcal{L}(\mathcal{S})$.

Definition Let $\mathcal{S}$ be a family of mutually disjoint signatures. Recall that a *basic action of* $\mathcal{L}(\mathcal{S})$ is a program of the form $\sigma\psi_1, \ldots, \psi_n$, where $\sigma \in \mathbf{\Sigma}$, for some signature $\mathbf{\Sigma} \in \mathcal{S}$, and n is the length of $\mathbf{\Sigma}$'s list of non-trivial action types. A *simple action of* $\mathcal{L}(\mathcal{S})$ is a program of $\mathcal{L}(\mathcal{S})$ in which neither the choice operation $\sqcup$ nor the iteration operation π^* occur. We use letters like α and β to denote simple actions *only*. A *simple sentence* is a sentence of $\mathcal{L}_1(\mathcal{S})$ in which neither action sum nor action iteration occur. So all programs in simple sentences are simple actions.

The canonical action model $\mathbf{\Omega}$ of $\mathcal{L}(\mathcal{S})$ We define a program model $\mathbf{\Omega}$ in several steps. The simple actions of $\mathbf{\Omega}$ are the simple actions of $\mathcal{L}(\mathcal{S})$ as defined just above. For all A, the accessibility relation $\xrightarrow{A}$ is the smallest relation such that

1. *skip* $\xrightarrow{A}$ *skip*.

	Basic Axioms	
	All sentential validities	
	($[\pi]$-normality)	$\vdash [\pi](\varphi \to \psi) \to ([\pi]\varphi \to [\pi]\psi)$
	($\Box_A$-normality)	$\vdash \Box_A(\varphi \to \psi) \to (\Box_A\varphi \to \Box_A\psi)$
$*$	($\Box^*_{\mathcal{C}}$-normality)	$\vdash \Box^*_{\mathcal{C}}(\varphi \to \psi) \to (\Box^*_{\mathcal{C}}\varphi \to \Box^*_{\mathcal{C}}\psi)$
	Action Axioms	For basic actions σ only:
	(Atomic Permanence)	$\vdash [\sigma]p \leftrightarrow (\textsc{Pre}(\sigma) \to p)$
	(Partial Functionality)	$\vdash [\sigma]\neg\chi \leftrightarrow (\textsc{Pre}(\sigma) \to \neg[\sigma]\chi)$
	(Action-Knowledge)	$\vdash [\sigma]\Box_A\varphi \leftrightarrow (\textsc{Pre}(\sigma) \to \bigwedge\{\Box_A[\sigma']\varphi : \sigma \xrightarrow{A} \sigma' \text{ in } \Omega\})$
$**$	**Action Mix Axiom**	$\vdash [\pi^*]\varphi \to \varphi \wedge [\pi][\pi^*]\varphi$
$*$	**Epistemic Mix Axiom**	$\vdash \Box^*_{\mathcal{C}}\varphi \to \varphi \wedge \bigwedge\{\Box_A\Box^*_{\mathcal{C}}\varphi : A \in \mathcal{C}\}$
	Skip Axiom	$\vdash [\mathit{skip}]\varphi \leftrightarrow \varphi$
	Crash Axiom	$\vdash [\mathit{crash}]\mathsf{false}$
	Composition Axiom	$\vdash [\pi;\rho]\varphi \leftrightarrow [\pi][\rho]\varphi$
	Choice Axiom	$\vdash [\pi \sqcup \rho]\varphi \leftrightarrow [\pi]\varphi \wedge [\rho]\varphi$

	Modal Rules	
	(Modus Ponens)	From $\vdash \varphi$ and $\vdash \varphi \to \psi$, infer $\vdash \psi$
	($[\pi]$-necessitation)	From $\vdash \psi$, infer $\vdash [\pi]\psi$
	($\Box_A$-necessitation)	From $\vdash \varphi$, infer $\vdash \Box_A\varphi$
$*$	($\Box^*_{\mathcal{C}}$-necessitation)	From $\vdash \varphi$, infer $\vdash \Box^*_{\mathcal{C}}\varphi$
$**$	**Program Induction Rule**	From $\vdash \chi \to \psi \wedge [\pi]\chi$, infer $\vdash \chi \to [\pi^*]\psi$
$*$	**Action Rule**	

Let ψ be a sentence, let α be a simple action, and let $\mathcal{C}$ be a set of agents. Let there be sentences χ_β for all β such that $\alpha \to^*_{\mathcal{C}} \beta$ (including α itself), and such that

1. $\vdash \chi_\beta \to [\beta]\psi$.
2. If $A \in \mathcal{C}$ and $\beta \xrightarrow{A} \gamma$, then $\vdash (\chi_\beta \wedge \textsc{Pre}(\beta)) \to \Box_A\chi_\gamma$.

From these assumptions, infer $\vdash \chi_\alpha \to [\alpha]\Box^*_{\mathcal{C}}\psi$.

Figure 8: The logical system for $\mathcal{L}(\mathcal{S})$. For $\mathcal{L}_1(\mathcal{S})$, we drop the $**$ axioms and rule; for $\mathcal{L}_0(\mathcal{S})$, we also drop the $*$ axioms and rules.

2. If $\sigma \xrightarrow{A} \sigma'$ in some signature $\mathbf{\Sigma} \in \mathcal{S}$ and $\vec{\varphi} = \vec{\psi}$, then $\sigma\vec{\varphi} \xrightarrow{A} \sigma'\vec{\psi}$.

3. If $\alpha \xrightarrow{A} \alpha'$ and $\beta \xrightarrow{A} \beta'$, then $\alpha;\beta \xrightarrow{A} \alpha';\beta'$.

Proposition 5.1 *As a frame,* $\mathbf{\Omega}$ *is* locally finite*: for each simple* α*, there are only finitely many* β *such that* $\alpha \xrightarrow{\mathcal{A}^*} \beta$.

Proof By induction on α; we use heavily the fact that the accessibility relations on $\mathbf{\Omega}$ are the smallest family with their defining property. For the simple action expressions $\sigma\vec{\psi}$, we use the assumption that all our signatures $\mathbf{\Sigma} \in \mathcal{S}$ are finite and mutually disjoint. $\dashv$

Next, we define $\textsc{Pre} : \Omega \to \mathcal{L}(\mathcal{S})$ by recursion so that

$$\begin{array}{lcl}
\textsc{Pre}(\mathit{skip}) & = & \mathsf{true} \\
\textsc{Pre}(\mathit{crash}) & = & \mathsf{false} \\
\textsc{Pre}(\sigma_i\vec{\psi}) & = & \psi_i \text{ for } \sigma_i \text{ in the given listing of } \mathbf{\Sigma} \\
\textsc{Pre}(\sigma\vec{\psi}) & = & \mathsf{true} \text{ for all trivial types } \sigma \\
\textsc{Pre}(\alpha;\beta) & = & \langle\alpha\rangle\textsc{Pre}(\beta)
\end{array}$$

Remark This function $\textsc{Pre}$ should not be confused with the function pre which is part of the structure of a program model. $\textsc{Pre}(\sigma)$ is a *sentence* in our language $\mathcal{L}(\mathcal{S})$, while pre was a purely semantic notion, associating *propositions* to simple actions in a program model. However, there is a connection: we are in the midst of defining the program model $\mathbf{\Omega}$, and its pre function is defined in terms of $\textsc{Pre}$. This is also perhaps a good place to remind the reader that neither $\textsc{Pre}$ nor pre is a first-class symbol in $\mathcal{L}(\mathcal{S})$; they are defined symbols.

Completing the definition of $\mathbf{\Omega}$ we set

$$\mathrm{pre}(\sigma) \quad = \quad [\![\textsc{Pre}(\sigma)]\!].$$

This action model $\mathbf{\Omega}$ is the *canonical (epistemic) action model*; it plays a somewhat similar role in our work to the canonical model in modal logic.

Proposition 5.2 (see [6] and [16]) *For every family* $\mathcal{S}$ *of action signatures, the logical systems for* $\mathcal{L}_0(\mathcal{S})$ *and* $\mathcal{L}_1(\mathcal{S})$ *presented in Figure 8 are sound, complete, and decidable. However, for every signature* $\mathbf{\Sigma}$ *which contains a (copy of the) "public announcement" action type* Pub*, the full logic* $\mathcal{L}(\mathbf{\Sigma})$ *(including iteration* π^**) is undecidable and the proof system for* $\mathcal{L}(\mathbf{\Sigma})$ *presented in Figure 8 is sound but incomplete. Indeed, validity in* $\mathcal{L}(\mathbf{\Sigma})$ *is* Π^1_1*-complete, so there are no recursive axiomatizations which are complete. (The same obviously applies to* $\mathcal{L}(\mathcal{S})$ *for any family of signatures* $\mathcal{S}$ *such that* $\mathbf{\Sigma} \in \mathcal{S}$*.)*

5.2 Some derivable principles

Lemma 5.3 *The Action-Knowledge Axiom is provable for all simple actions* α*:*

$$\vdash [\alpha]\Box_A\varphi \;\leftrightarrow\; (\textsc{Pre}(\alpha) \to \bigwedge\{\Box_A[\beta]\varphi : \alpha \xrightarrow{A} \beta \text{ in } \mathbf{\Omega}\}) \tag{11}$$

The proof is by induction on α and may be found in [7].

Lemma 5.4 *For all $A \in \mathcal{C}$, all simple α, and all β such that $\alpha \to_A \beta$,*

1. $\vdash [\alpha]\Box^*_{\mathcal{C}}\psi \to [\alpha]\psi$.
2. $\vdash [\alpha]\Box^*_{\mathcal{C}}\psi \wedge \textsc{Pre}(\alpha) \to \Box_A[\beta]\Box^*_{\mathcal{C}}\psi$.

Proof Part (1) follows easily from the Epistemic Mix Axiom and modal reasoning. For part (2), we start with a consequence of the Epistemic Mix Axiom: $\vdash \Box^*_{\mathcal{C}}\psi \to \Box_A\Box^*_{\mathcal{C}}\psi$. Then by modal reasoning, $\vdash [\alpha]\Box^*_{\mathcal{C}}\psi \to [\alpha]\Box_A\Box^*_{\mathcal{C}}\psi$. By the Action-Knowledge Axiom generalized as we have it in Lemma 5.3, we have $\vdash [\alpha]\Box^*_{\mathcal{C}}\psi \wedge \textsc{Pre}(\alpha) \to \Box_A[\beta]\Box^*_{\mathcal{C}}\psi$. $\dashv$

Lemma 5.5 (The Common Knowledge Induction Rule) *From $\vdash \chi \to \psi \wedge \Box_A\chi$ for all A, infer $\vdash \chi \to \Box^*_{\mathcal{A}}\psi$.*

Proof We apply the Action Rule to the simple action *skip*, recalling that $\textsc{Pre}(\mathit{skip}) = \mathsf{true}$, and $\mathit{skip} \xrightarrow{A} \mathit{skip}$ for all A. $\dashv$

5.3 Logical systems for the target logics

We presented a number of target logics in Section 2.4, and these were then formalized in Section 4.1. In particular, we have logics $\mathcal{L}_1(\mathbf{\Sigma})$ for a number of interesting action signatures $\mathbf{\Sigma}$. What we want to do here is to spell out what the axioms of $\mathcal{L}_1(\mathcal{S})$ come to when we specialize the general logic to the logics of special interest. In doing this, we find it convenient to adopt simpler notations tailored for the fragments.

The logic of public announcements is shown in Figure 9. We only included the axioms and rule of inference that specifically used the structure of the signature $\mathbf{\Sigma}_{\mathsf{pub}}$. So we did not include the sentential validities, the normality axiom for $\Box_A$, the composition axiom, modus ponens, etc. Also, we renamed the main axiom and rule to emphasize the "announcement" aspect of the system.

Our next logic is the logic of completely private announcements to groups. We discussed the formalization of this in Section 4.4. We have actions $\mathit{Pri}^B\varphi$ and (of course) *skip*. The axioms and rules are just as in the logic of public announcements, with a few changes. We must of course consider the relativized operators $[\mathit{Pri}^{\mathcal{B}}\varphi]$ instead of their simpler counterparts $[\mathit{Pub}\ \varphi]$.)

The actions *skip* all have true as their precondition, and since $(\mathsf{true} \to \psi)$ is logically equivalent to ψ, we certainly may omit these actions from the notation in the axioms and rules. The most substantive change which we need to make in Figure 9 concerns the Action-Knowledge Axiom. It splits into two axioms, noted below:

$$\begin{array}{ll} [\mathit{Pri}^{\mathcal{B}}\varphi]\Box_A\psi \leftrightarrow (\varphi \to \Box_A[\mathit{Pri}^{\mathcal{B}}\varphi]\psi) & \text{for } A \in \mathcal{B} \\ [\mathit{Pri}^{\mathcal{B}}\varphi]\Box_A\psi \leftrightarrow (\varphi \to \Box_A\psi) & \text{for } A \notin \mathcal{B} \end{array}$$

The last equivalence says: assuming that φ is true, then after a private announcement of φ to the members of $\mathcal{B}$, an outsider knows ψ just in case she knew ψ before the announcement.

Basic Axioms	
($[Pub\ \varphi]$-normality)	$\vdash [Pub\ \varphi](\psi \to \chi) \to ([Pub\ \varphi]\psi \to [Pub\ \varphi]\chi)$
Annoucement Axioms	
(Atomic Permanence)	$\vdash [Pub\ \varphi]p \leftrightarrow (\varphi \to p)$
(Partial Functionality)	$\vdash [Pub\ \varphi]\neg\psi \leftrightarrow (\varphi \to \neg[Pub\ \varphi]\psi)$
(Announcement-Knowledge)	$\vdash [Pub\ \varphi]\Box_A\psi \leftrightarrow (\varphi \to \Box_A[Pub\ \varphi]\psi)$
Modal Rules	
($[Pub\ \varphi]$-necessitation)	From $\vdash \psi$, infer $\vdash [Pub\ \varphi]\psi$
Announcement Rule	From $\vdash \chi \to [Pub\ \varphi]\psi$ and $\vdash \chi \wedge \varphi \to \Box_A\chi$ for all A,infer $\vdash \chi \to [Pub\ \varphi]\Box^*_{\mathcal{A}}\psi$

Figure 9: The main points of the logic of public announcements.

Finally, we study the logic of common knowledge of alternatives. This, too, was introduced in Section 2.4 and formalized in Section 4.1. The Action-Knowledge now becomes

$$\begin{array}{ll} [Cka^{\mathcal{B}}\vec{\varphi}]\Box_A\psi \leftrightarrow (\varphi_1 \to \Box_A[Cka^{\mathcal{B}}\vec{\varphi}]\psi) & \text{for } A \in \mathcal{B} \\ [Cka^{\mathcal{B}}\vec{\varphi}]\Box_A\psi \leftrightarrow (\varphi_1 \to \bigwedge_{0\leq i\leq k}\Box_A[Cka^{\mathcal{B}}\vec{\varphi}^i]\psi) & \text{for } A \notin \mathcal{B} \end{array}$$

where in the last clause, $(\varphi_1, \ldots, \varphi_n)^i$ is the sequence $\varphi_i, \varphi_1, \ldots, \varphi_{i-1}, \varphi_{i+1}, \ldots, \varphi_k$. (That is, we bring φ_i to the front of the sequence.)

5.4 Examples in the target logics

This section studies some examples of the logic at work.

We begin with an application of the Announcement Rule in the logic of public announcements. We again work with the atomic sentences H and T for heads and tails, and with the set $\{A, B\}$ of agents. We show

$$\vdash \Box^*_{A,B}(\mathsf{H} \leftrightarrow \neg\mathsf{T}) \to [Pub\ \mathsf{H}]\Box^*_{A,B}\neg\mathsf{T}.$$

That is, on the assumption that it is common knowledge that heads and tails are mutually exclusive, then as a result of a public announcement of heads it will be common knowledge that the state is not tails.

We give this application in detail. Recall that $\mathbf{\Sigma}_{\text{pub}}$ has one simple action which we call Pub. We take χ_{Pub} to be $\Box^*_{A,B}(\mathsf{H} \leftrightarrow \neg\mathsf{T})$. In addition $Pub \xrightarrow{A} Pub$ for all A, and there are no other arrows in $\mathbf{\Sigma}_{\text{pub}}$. We take α to be $Pub\ \mathsf{H}$; note that this is the only action accessible from itself in the canonical action model. To use the Announcement Rule, we must show that

1. $\vdash \Box^*_{A,B}(\mathsf{H} \leftrightarrow \neg\mathsf{T}) \to [Pub\ \mathsf{H}]\neg\mathsf{T}$.

2. $\vdash (\Box^*_{A,B}(\mathsf{H} \leftrightarrow \neg\mathsf{T}) \wedge \mathsf{H}) \rightarrow \Box_A\Box^*_{A,B}(\mathsf{H} \leftrightarrow \neg\mathsf{T})$, and the same with B replacing A.

From these assumptions, we may infer $\vdash \Box^*_{A,B}(\mathsf{H} \leftrightarrow \neg\mathsf{T}) \rightarrow [Pub\ \mathsf{H}]\Box^*_{A,B}\neg\mathsf{T}$.

For the first statement,

(a)	$\mathsf{T} \leftrightarrow [Pub\ \mathsf{H}]\mathsf{T}$	Atomic Permanence
(b)	$(\mathsf{H} \leftrightarrow \neg\mathsf{T}) \rightarrow (\mathsf{H} \rightarrow \neg[Pub\ \mathsf{H}]\mathsf{T})$	(a), propositional reasoning
(c)	$[Pub\ \mathsf{H}]\neg\mathsf{T} \leftrightarrow (\mathsf{H} \rightarrow \neg[Pub\ \mathsf{H}]\mathsf{T})$	Partial Functionality
(d)	$\Box^*_{A,B}(\mathsf{H} \leftrightarrow \neg\mathsf{T}) \rightarrow (\mathsf{H} \leftrightarrow \neg\mathsf{T})$	Epistemic Mix
(e)	$\Box^*_{A,B}(\mathsf{H} \leftrightarrow \neg\mathsf{T}) \rightarrow [Pub\ \mathsf{H}]\neg\mathsf{T}$	(d), (b), (c), propositional reasoning

And the second statement is an easy consequence of the Epistemic Mix Axiom.

What happens when a publicly known fact is announced? One intuition about public announcements and common knowledge is that if φ is common knowledge, then announcing φ publicly does not change anything. Formally, we express this by a scheme rather than a single equation:

$$\Box^*\varphi \rightarrow ([Pub\ \varphi]\psi \leftrightarrow \psi) \tag{12}$$

(In this line and in the rest of this section, we are omitting the subscripts on the $\Box^*$ operator. More formally, the subscript should be $\mathcal{A}$, since we are dealing with knowledge which is common to all agents.) What we would like to say is $\Box^*\varphi \rightarrow \bigwedge_\psi([Pub\ \varphi]\psi \leftrightarrow \psi)$, but of course this cannot be expressed in our language. So we consider only the sentences of the form (12), and we show that all of these are provable. We argue by induction on φ.

For an atomic sentence p, (12) follows from the Epistemic Mix and Atomic Permanence Axioms. The induction steps for $\wedge$ and $\neg$ are easy. Next, assume (12) for ψ. By necessitation and Epistemic Mix, we have

$$\Box^*\varphi \rightarrow (\Box_A[Pub\ \varphi]\psi \leftrightarrow \Box_A\psi)$$

Note also that by the Announcement-Knowledge Axiom

$$\Box^*\varphi \rightarrow ([Pub\ \varphi]\Box_A\psi \leftrightarrow \Box_A[Pub\ \varphi]\psi)$$

These two imply (12) for $\Box_A\psi$.

Finally, we assume (12) for ψ and prove it for $\Box^*_{\mathcal{B}}\psi$. We show first that $\Box^*\varphi \wedge \Box^*\psi \rightarrow [Pub\ \varphi]\Box^*\psi$. For this we use the Action Rule. We must show that

(a) $\Box^*\varphi \wedge \Box^*\psi \rightarrow [Pub\ \varphi]\psi$.

(b) $\Box^*\varphi \wedge \Box^*\psi \wedge \varphi \rightarrow \Box_A(\Box^*\varphi \wedge \Box^*\psi)$.

(Actually, since our common knowledge operators $\Box^*$ here are really of the form $\Box^*_{\mathcal{A}}$, we need (b) for all agents A.) Point (a) is easy from our induction hypothesis, and (b) is an easy consequence of Epistemic Mix.

To conclude, we show $\Box^*\varphi \wedge [Pub\ \varphi]\Box^*\psi \rightarrow \Box^*\psi$. For this, we use the Common Knowledge Induction Rule of Lemma 5.5; that is, we show

(c) $\Box^*\varphi \wedge [Pub\ \varphi]\Box^*\psi \rightarrow \psi$.

(d) $\Box^*\varphi \wedge [Pub\ \varphi]\Box^*\psi \to \Box_A(\Box^*\varphi \wedge [Pub\ \varphi]\Box^*\psi)$ for all A.

For (c), we use Lemma 5.4, part (1) to see that $[Pub\ \varphi]\Box^*\psi \to [Pub\ \varphi]\psi$; and now (c) follows from our induction hypothesis. For (d), it will be sufficient to show that

$$\varphi \wedge [Pub\ \varphi]\Box^*\psi \to \Box_A[Pub\ \varphi]\Box^*\psi$$

This follows from Lemma 5.4, part (2).

A commutativity principle for private announcements Suppose that $\mathcal{B}$ and $\mathcal{C}$ are disjoint sets of agents. Let φ_1, φ_2, and ψ be sentences. Then we claim that

$$\vdash [Pri^{\mathcal{B}}\varphi_1][Pri^{\mathcal{C}}\varphi_2]\psi \leftrightarrow [Pri^{\mathcal{C}}\varphi_2][Pri^{\mathcal{B}}\varphi_1]\psi.$$

That is, order does not matter with private announcements to disjoint groups.

Actions do not change common knowledge of non-epistemic sentences For yet another application, let ψ be any boolean combination of atomic sentences. Then for all actions α of any of our logics, $\vdash \psi \leftrightarrow [\alpha]\psi$. The proof is an easy induction on ψ. Even more, we have $\vdash \Box^*_{\mathcal{C}}\psi \leftrightarrow [\alpha]\Box^*_{\mathcal{C}}\psi$. In one direction, we use the Action Rule, and in the other, the Common Knowledge Induction Rule (Lemma 5.5).

Endnotes Although the general logical systems in this paper are new, there are important precursors for the target logics. Plaza [18] constructs what we would call $\mathcal{L}_0(\mathbf{\Sigma}_{Pub})$, that is, the logic of public announcements without common knowledge operators or program iteration. (He worked only on models where each accessibility relation is an equivalence relation, so his system includes the S5 axioms.) Gerbrandy [10, 11], and also Gerbrandy and Groeneveld [12] went a bit further. They studied the logic of completely private announcements (generalizing public announcements) and presented a logical system which included the common knowledge operators. That is, their system included the Epistemic Mix Axiom. They argued that all of the reasoning in the original Muddy Children scenario can be carried out in their system. This is important because it shows that in order to get a formal treatment of that problem and related ones, one need not posit models which maintain histories. Their system was not complete since it did not have anything like the Action Rule; this first appears in a slightly different form in [6].

5.5 Conclusion

We have been concerned with actions in the social world that affect the intuitive concepts of knowledge, (justifiable) beliefs, and common knowledge. This paper has shown how to define and study logical languages that contain constructs corresponding to such actions. The many examples in this paper show that the logics "work". Much more can be said about specific tricky examples, but we hope that the examples connected to our scenarios make the point that we are developing valuable tools.

The key steps in the development are the recognition that we can associate to a social action α a mathematical model $\mathbf{\Sigma}$. $\mathbf{\Sigma}$ is a program model. In particular, it is a multi-agent Kripke model, so it has features in common with the state models that underlie formal work in the entire area. There is a natural operation of update product at the heart of our work. This operation is surely of independent interest because it enables one to build complex and interesting state models. The logical languages that we introduce use the update product in their semantics, but the syntax is a small variation on propositional dynamic logic. The formalization of the target languages involved the signature-based languages $\mathcal{L}(\mathbf{\Sigma})$ and also their generalizations $\mathcal{L}(\mathcal{S})$. These latter languages are needed to formulate the logic of private announcements, for example. We feel that presenting the update product first (before the languages) will make this paper easier to read, and having a relatively standard syntax should also help.

Once we have our languages, the next natural step is to study them. This paper presented logical systems for validities, omitting many proofs due to the lack of space.

References

[1] Alexandru Baltag, A logic of epistemic actions, *(Electronic) Proceedings of the FACAS workshop, held at ESSLLI'99*, Utrecht University, Utrecht 1999.

[2] Alexandru Baltag, Logics for insecure communication, in J. van Bentham (ed) *Proceedings of the Eighth Conference on Rationality and Knowledge (TARK'01)*, Morgan Kaufmann, Los Altos, 111–122.

[3] Alexandru Baltag, A logic for suspicious players: epistemic actions and belief updates in games, *Bulletin Of Economic Research*, **54** (2002), no. 1, 1–46.

[4] Alexandru Baltag, A coalgebraic semantics for epistemic programs, In *Proceedings of CMCS'03*, *Electronic Notes in Theoretical Computer Science* **82** (2003), no.1 , 315–335.

[5] Alexandru Baltag, Logics for Communication: reasoning about information flow in dialogue games. Course presented at *NASSLLI'03*. Available at http://www.indiana.edu/∼nasslli.

[6] Alexandru Baltag, Lawrence S. Moss, and Sławomir Solecki, The logic of common knowledge, public announcements, and private suspicions, in I. Gilboa (ed), *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK'98)*, 1998, 43–56.

[7] Alexandru Baltag, Lawrence S. Moss, and Sławomir Solecki, The logic of epistemic actions: completeness, decidability, expressivity ms., 2003.

[8] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi, **Reasoning About Knowledge**, MIT Press, 1996.

[9] Michael J. Fischer and Richard E. Ladner, Propositional modal logic of programs. *J. Comput. System Sci.* 18 (1979), no. 2, 194–211.

[10] Jelle Gerbrandy, Dynamic epistemic logic, in Lawrence S. Moss, et al (eds.) **Logic, Language, and Information, vol. 2**, CSLI Publications, Stanford University 1999.

[11] Jelle Gerbrandy, *Bisimulations on Planet Kripke*, Ph.D. dissertation, University of Amsterdam, 1999.

[12] Jelle Gerbrandy and Willem Groeneveld, Reasoning about information change, *Journal of Logic, Language, and Information* **6** (1997) 147–169.

[13] P. Gochet and P. Gribomont, **Epistemic Logic**, ms., 2003.

[14] Barteld P. Kooi, *Knowledge, Chance, and Change*, Ph.D. dissertation, University of Groningen, 2003.

[15] J.- J. Meyer and W. van der Hoek, **Epistemic Logic for AI and Computer Science**, Cambridge University Press, Cambridge, 1995.

[16] Joseph S. Miller and Lawrence S. Moss, The undecidability of iterated modal relativization, Indiana University Computer Science Department Technical Report 586, May 2003.

[17] Lawrence S. Moss, From hypersets to Kripke models in logics of announcements, in J. Gerbrandy et al (eds.) **JFAK. Essays Dedicated to Johan van Benthem on the Occasion of his 60th Birthday**, Vossiuspers, Amsterdam University Press, 1999.

[18] Jan Plaza, Logics of public communications, *Proceedings, 4^{th} International Symposium on Methodologies for Intelligent Systems*, 1989.

[19] Vaughn R. Pratt, Semantical considerations on Floyd-Hoare logic, in *17th Annual Symposium on Foundations of Computer Science*, IEEE Comput. Soc., Long Beach, Calif., 1976, 109–121.

[20] Johan van Benthem, Update delights, ms., 2000.

[21] Johan van Benthem, Games in dynamic epistemic logic, *Bulletin of Economic Research*, 53 (2001) no. 4, 219–248.

[22] Johan van Benthem, Logic for information update, in J. van Bentham (ed) *Proceedings of the Eighth Conference on Rationality and Knowledge (TARK'01)*, Morgan Kaufmann, Los Altos, 51–68.

[23] Hans P. van Ditmarsch, *Knowledge Games*, Ph.D. dissertation, University of Groningen, 2000.

[24] Hans P. van Ditmarsch, Knowledge games, *Bulletin of Economic Research*, 53 (2001), no. 4, 249–273.

[25] Hans P. van Ditmarsch, W. van der Hoek, B. P. Kooi, Concurrent dynamic epistemic logic, V. F. Hendricks et al (eds), Synt. Lib. vol. 322, Kluwer Academic Publishers, 2003.