Electronic Notes in Theoretical Computer Science 82 No. 1 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume82.html 22 pages

A Coalgebraic Semantics for Epistemic Programs

Alexandru Baltag [1]

Computing Laboratory
University of Oxford
Oxford, UK

Abstract

This paper gives a fresh look at my previous work on "epistemic actions" and information updates in distributed systems, from a coalgebraic perspective. I show that the "relational" semantics of epistemic programs, given in [BMS2] in terms of epistemic updates, can be understood in terms of functors on the category of coalgebras and natural transformations associated to them. Then, I introduce a *new, alternative, more refined semantics* for epistemic programs: programs as "epistemic coalgebras". I argue for the advantages of this second semantics, from a semantic, heuristic, syntactical and proof-theoretic point of view. Finally, as a step towards a generalization, I show these concepts make sense for other functors, and that apparently unrelated concepts, such as Bayesian belief updates and process transformations, can be seen to arise in the same way as our "epistemic actions".

1 Introduction

Epistemic models have been used as models for partial information in multi-agent systems. But epistemic models are static: they are models that only take into account the *state of information (knowledge or belief)* of every agent involved at a given time. However, the most important issues have to do with *information flow*: the *changes* affecting the agent's information states, changes due to various actions (message-passing or other forms of information exchange, information updates or even disinformation). There are some notions of *knowledge programs* and *knowledge protocols* in the literature, but no general, robust one: most of these notions are *local*, as they live in a fixed epistemic-dynamic model: they do not describe *generic* kinds of changes, that might affect (in different ways) *any arbitrary epistemic situation*. Hence, the

[1] Email: baltag@comlab.ox.ac.uk

problem of offering a good *dynamic* counterpart of epistemic models can be considered as still open.

History of the Problem. Information flow was the central subject of [FHMV], and this book gives the standard approach to this subject. However, the authors do *not* introduce or study epistemic programs in a formal way, since they are committed to a *temporal* point of view: they use a combination of temporal and epistemic models, and use a temporal-epistemic logic to study them. [2]

The work of J. Gerbrandy is the first one that really addressed this problem in a general enough frame (although his work has been partially anticipated by J. Plaza, J. van Benthem, F. Veltman and others). He used non-wellfounded sets to provide a semantics for a restricted notion of epistemic programs; essentially, this is a particular case of our "Third Definition of Epistemic Updates" (in section 3 below), in terms of the final coalgebra. He did not study updates as formal semantic objects (relations on the final coalgebra), but simply described for each program modality, a specific such relation. His *syntax* for programs was very limited, and as a result he could describe only a very restricted class of programs.

In past work [3], we have introduced a general notion of "epistemic actions", subsuming many various information updates encountered in the literature (including Gerbrandy's programs), and we have studied the *logic* induced by such programs, in PDL style, providing a complete axiomatization. But we did *not* concentrate on the *semantic* aspect: epistemic actions were introduced there simply as syntactic notions, as a way to generate dynamic modalities. The semantics for these modalities was given (similarly to Gerbrandy's) by describing in detail a specific model-theoretic construction for each such modality; but these model-theoretic transformations were *not* considered and studied *as objects in their own respect*, and so there was *no really semantical notion* of epistemic actions. In recent [4] and current work [5], this gap is filled, by giving general semantic notions of "epistemic programs". In particular, in [BMS2] (still to appear), we introduced the notion of *epistemic update* (as defined below, in section 3, in the "First Definition" of update) and used it as our basic semantics for epistemic programs. Our work on completeness has unexpectedly lead us to endow our *syntactic* action expressions with a structure *similar to the one of epistemic models*: in effect, this is nothing but a *syntactic coalgebra*. But we have not considered it as an alternative semantics for epistemic programs; nor have we set our study (of epistemic models and programs) in coalgebraic terms.

The Contribution of This Paper. This is first of all a *coalgebraic recon-*

[2] The relation between their work and ours is similar to the one between the two main approaches to program verification: temporal logic versus Hoare-style (or PDL-style) logics.
[3] [BMS], joint work with L. Moss and S. Solecki.
[4] [B2], [B3]
[5] [BMS2], joint work with L. Moss and S. Solecki

struction of most of my previous work. It is easy to see that epistemic models are nothing but coalgebras for a simple functor. The first new contribution is showing that the notion of "epistemic updates" introduced in [BMS2] has a *natural category-theoretical meaning*: namely, as pairs of functors on the category of coalgebras and natural transformations related to the functors. This notion makes sense in general, for arbitrary functors, giving us an interesting notion of *coalgebraic transformation*. I give examples for other functors, showing e.g. that it covers process transformations.

Another new contribution is the introduction of a more refined semantics for epistemic programs: the *coalgebraic semantics*. Roughly speaking, epistemic programs are seen to be themselves coalgebras for a functor of the same type as the one which generates epistemic models (as coalgebras). This idea originates in the syntactic coalgebra mentioned above (as implicit in our previous work); but now it is made explicit and taken at face value, as an *alternative semantics*. I show this captures more features of epistemic programs than the first semantics, allowing us to define new useful operations. It also gives us new useful tools for defining and proving things about epistemic programs, by coinduction.

In on-going work, I try to generalize this second semantics to a large class of functors, obtaining something like a notion of "dynamic coalgebra" [6] . But for now, I only give other examples of functors for which this can be done; in particular, the Larson-Skou functor gives us a *probabilistic notion* of epistemic models and epistemic programs. Standard Bayesian belief update can now be seen as such a program, being the probabilistic analogue of the public announcement update. Our general notion provides an interesting generalization of Bayesian update, which could be called *Bayesian epistemic programs*.

Preliminaries: Epistemic Models and Epistemic Propositions, Coalgebraically. Given a set Φ, a *predicate on* Φ is a function $P : \Phi \to 2$, where $2 = \{0, 1\}$ is the set of truth values. We denote by $Pred(\Phi) =: \Phi \to 2$ the family of all predicates on Φ. For a predicate $P \in Pred(\Phi)$ and an object $p \in \Phi$, we write $P(p)$, and we say that P holds at p, or p satisfies P, whenever $P(p) = 1$. We sometimes identify a predicate P with the class $P_\Phi = \{p \in \Phi : P(p) = 1\}$.

For any endofunctor $T : Set \to Set$ and any set Φ, we define an endofunctor T_Φ on Set by putting:

$$T_\Phi =: Pred(\Phi) \times T,$$

where we have ambiguously denoted by $Pred(\Phi)$ the constant endofunctor associated to the set $Pred(\Phi)$ (mapping every set to the set $Pred(\Phi)$ and every function to identity on $Pred(\Phi)$).

For most of this paper, we shall consider only a particular choice for T. Let us Ag be a finite set of *agents*, denoted by $a, b, \ldots$. We shall use capital

[6] I owe thanks to one of the referees for suggesting this name.

letters $A, B, \ldots \subseteq Ag$ to denote *finite sets of agents*. Let κ be an inaccessible cardinal and let $\mathcal{P}_k : Set \to Set$ be the κ-bounded powerset endofunctor on the category of sets (for sets, this is defined by $\mathcal{P}_k(S) =: \{X \subseteq S : |X| < \kappa\}$). *From now on (unless specified otherwise), we fix* $T =: \mathcal{P}_\kappa^{Ag}$.

Epistemic Models as Coalgebras. Fix now a class Φ of objects, denoted by $p, q, \ldots$ and called *atomic sentences*. For this particular set Φ, we shall denote by $F =: T_\Phi = Pred(\Phi) \times \mathcal{P}_\kappa^{Ag}$ the functor T_Φ introduced before. An *epistemic model* (over a set Φ of atomic sentences) [7] is just an F-coalgebra; i.e. a structure $\mathbf{S} = (S, e)$, with $e : S \to F(S)$, for the above functor $F = T_\Phi$. Whenever the coalgebraic map e is understood, we use the notations $s_0 =: e(s)_0$ and $s_a =: e(s)_1(a)$; here, we denoted by $t_0 \in Pred(\Phi)$, and respectively $t_1 \in \mathcal{P}_\kappa(S)^{Ag}$, the first, and respectively the second, component of any object $t \in F(S) = Pred(\Phi) \times \mathcal{P}_\kappa S^{Ag}$. In an epistemic model $\mathbf{S}$, the members of its support S are called *possible worlds* or *states*. Observe that s_0 is a predicate on Φ called the *(factual) content* of the state s, and $s_a \subseteq S$ is a set of states, called the *appearance of state s to agent a*. Since the atomic sentences $p \in \Phi$ are supposed to describe "facts of the world", the factual content s_0 of a given state s will be interpreted as telling us which "facts" hold at each state: we say that *p holds at state s in model* $\mathbf{S}$, and write $p_{\mathbf{S}}(s)$, whenever $s_0(p)$ holds. The appearance $s_a \subseteq S$ of state s to agent a is intuitively the set of all the states that are "indistinguishable" from s to agent a: if the actual state were s then agent a would think any of the states $s' \in s_a$ might be the actual one. The worlds $s' \in w_a$ are the *epistemic alternatives of the world s* (for agent a). For every agent, one can define a binary relation $\to_a \subseteq S \times S$ of *(epistemic) indistinguishability for agent a*, by setting: $s \to_a s'$ iff $s' \in s_a$.

A *pointed model* (or pointed F-coalgebra) is just a pair $(\mathbf{S}, \mathbf{s})$ consisting of an epistemic model $\mathbf{S}$ together with a designated state $\mathbf{s}$, called the *"actual state"* (or the "real world").

Our models being F-coalgebras, they can be endowed with coalgebraic morphisms as usually to form a category, which will be denoted by $\mathbf{Mod} =: \mathbf{Coalg}(\mathbf{F})$. As expected, the corresponding (Aczel-Mendler) notion of *bisimulation* is in this case nothing but a binary relation $R \subseteq \mathbf{S} \times \mathbf{S}'$ between two models, such that whenever we have sRs' we also have: (1) $s_0 = s'_0$; (2) for every agent $a \in Ag$ and every $t \in s_a$ there exists some $t' \in s'_a$ such that $s'Rt'$; (3) for every agent $a \in Ag$ and every $t' \in s'_a$ there exists some $t \in s_a$ such that $s'Rt'$. We call *epistemic bisimilarity*, and denote by $\sim$, the notion of coalgebraic bisimilarity obtained in this way. Similarly, we denote by $\mathbf{pMod} =: \mathbf{pCoalg}(\mathbf{F})$ the category of pointed models, where the morphisms are coalgebraic morphisms mapping the designated (actual) of the first model to the designated state of the second model. The resulting notion of bisimulation is just a bisimulation relating the two designated states, and we also

[7] Observe that the notion of epistemic model is mathematically the same as the one of "transition system" with basic predicates in Φ and "action labels" in Ag; only the interpretation is different: labels in Ag are taken to represent agents, instead of actions.

denote the resulting bisimilarity relation by $\sim$.

As it is well known, the functor $F = Pred(\Phi) \times \mathcal{P}_{\kappa}^{Ag}$ is bounded and has a final coalgebra $\Omega = \Omega_F$. The final map induces a map $\omega_F : \mathbf{p}Mod \to \Omega_F$, having the property that two pointed models are bisimilar iff they have the same image via ω_F.

Epistemic Propositions. "Epistemic properties" are epistemically relevant properties of states in epistemic models. Epistemic properties are given by *epistemic propositions*. We define several different, but equivalent, versions of this notion.

First Definition of Propositions. An *epistemic proposition* is a "bisimulation-invariant predicate map" on epistemic models, i.e. a map $\mathbf{P} : Mod \to Set$, associating to each model $\mathbf{S}$ some *predicate* $\mathbf{P_S} \in Pred(S)$ on S, such that: if $\mathbf{P_S}(s)$ holds and $(\mathbf{S}, s) \sim (\mathbf{T}, t)$, then $\mathbf{P_T}(t)$ holds.

But now one can easily see that bisimulation-invariance is a *naturality* condition, which leads to our next version of the same concept:

Second (Category-Theoretical) Definition of Epistemic Propositions (A. Kurz): Let $U : \mathbf{M}od \to Set$ be the forgetful functor, taking each coalgebra (S, e) to its support S. An *epistemic proposition* [8] is a natural transformation $\mathbf{P} : U \to 2$.

By going to the final coalgebra, we can factor over bisimilarity and simplify this to the following:

Third Definition of Epistemic Propositions. An *epistemic proposition* is a predicate $\mathbf{P}_\Omega : \Omega \to 2$ on the final coalgebra $\Omega = \Omega_F$, i.e. a Function into the set $2 = \{0, 1\}$.

Indeed, we can just take, for every pointed model $(\mathbf{S}, s)$, $\mathbf{P}_\Omega(\omega(S, s)) =: \mathbf{P_S}(s)$, and check that this function is well-defined and is total function on Ω. It is in fact easy to see that the above three definitions of "epistemic propositions" are *equivalent*: there exist canonical bijective correspondences between the objects falling under any two of the above definitions. Using systematic ambiguity (as we have already done in stating the above definitions), we shall denote by the same names $\mathbf{P}$ the (corresponding) objects falling under the above definitions, and call them all epistemic propositions.

Operations with Epistemic Propositions. Let $Prop_F =: Pred(\Omega_F)$ be the family of epistemic propositions (seen as predicates on the final coalgebra). This collection is closed under various natural operations with predicates:

- The propositions $\top$ ("true") and $\bot$ ("false") are defined in the standard way: $\top(s)$ holds for all states $s \in \Omega$, while $\bot(s)$ holds for none.
- Epistemic propositions are closed under standard *Boolean operations* $\neg$, $\wedge$, $\vee$ with propositions/predicates: this is obvious is we think of them as predicates on the final coalgebra. We can even extend this to infinite conjunctions

[8] This is Kurz's notion of "modal proposition", introduced in the general context of Kripke models.

$\bigwedge$ and disjunctions $\bigvee$.

- For each atomic sentence $p \in \mathcal{C}$ we have an atomic proposition $\mathbf{p}$. On both arbitrary coalgebras and the final one, we can define $\mathbf{p}(s)$ to hold (at a state s) iff $s_0(p)$ holds. (Recall that $s_0 = e(s)_0 \in Pred(\Phi)$.)
- *Epistemic Modalities: Knowledge or Belief.* For any epistemic proposition $\mathbf{P}$ and agent a, we can define another proposition $\Box_a P$, by setting, for every state $s \in \Omega$: $(\Box_a \mathbf{P})(s)$ iff $\mathbf{P}(t)$ holds for all $t \in s_a$ We read this new proposition as saying that (at the current state s) "agent *a knows, or believes* that $\mathbf{P}$".
- *Iterated Epistemic Modalities: Common Knowledge.* For any proposition $\mathbf{P}$ and any subgroup $A \subset Ag$ of agents, we define a new proposition $\Box_A^* \mathbf{P}$ by using infinite conjunction: $\Box_A^* \mathbf{P} =: \bigwedge\{\Box_{a_1} \dots \Box_{a_n} \mathbf{P} : n \geq 0,\ a_1, \dots, a_n \in A\}$. This is the modality corresponding (in the standard Kripke semantics) to the reflexive-transitive closure of the union of all indistinguishability relations $\rightarrow_a$ with $a \in A$. We read $\Box_A^* \mathbf{P}$ as saying that proposition $\mathbf{P}$ is *common knowledge among the members of the group A*.

2 Epistemic Programs

Until now, all our models described only "static" epistemic situations: the state of a system and the information states of all the agents *at a given time*. We take now into account *possible informational changes* of the given system: epistemic programs.

An epistemic program is a program taking pointed epistemic models as both inputs and outputs. It represents an "epistemic computation" performed on the information states of the agents, computation corresponding to some specific way to induce an information update of all the agents: some form of *communication, announcement, discovery, act of introspection* etc.

Examples of Epistemic Programs

1. **Public Announcements P!** Given an epistemic proposition $\mathbf{P}$, we can consider the action $\mathbf{P}!$ of *publicly announcing that* $\mathbf{P}$ is true (at the current state). By this, we mean a *truthful, fully reliable, public* broadcasting of the message $\mathbf{P}$. This is an *impersonal* announcement: we are not interested for that moment in who does the announcing, but only consider it as an anonymous announcement. [9]

The effect of this announcement is that all the agents *publicly learn* that $\mathbf{P}$ *was* true (at the moment of its announcing). This means they all publicly update their states of information, eliminating from their sets of possible states the ones in which $\mathbf{P}$ *was not true before this action happened*. The fact that this update is public means that, while doing it, each agent knows the others do it as well, and he also knows the others know this etc.: in short, while this

[9] One can of course model more "personal" announcements in our frame, but we just want to keep it simple for now.

action is happening, it is *common knowledge that it is happening*. This is why it is called a public announcement.

However (as observed by J. Gerbrandy and others), it is not always the case that the act of publicly announcing **P** is necessarily producing common knowledge of **P**. This is because the truth-value of **P** might change due to its own announcement: consider the proposition "The war has just started, but John doesn't know it." By publicly announcing this (to everyone, including John), the proposition which was announced has become false: *now*, John knows.

2. **Private Announcements $\mathbf{P}!_A$ to subgroups.** This is similar to the previously mentioned action, except that the announcement is broadcasted only to a subgroup $A \subseteq Ag$ of "inside" agents, while it is assumed that none of the "outsiders" suspect that this is happening: they don't observe anything, so they just assume by default that nothing is happening (or, if you like, that the trivial action "skip" is happening...).

3.**Private Announcements with Suspicious Outsiders.** This is similar to the previous action, except that while the private announcement is broadcasted to the insiders, some outsider $b \notin A$ starts to *suspect* that this is happening.

4. **Private Announcements with Secret Interception by an Outsider (Wiretapping).** As before, but now the outsider b actually *finds out* what's going on: he knows about the announcement, say because he "secretly wiretaps" the insider's conversation (or intercepts the message in some other way). But the wiretapping is done in a completely secretive manner, so that the none of the insiders suspects it is happening.

5. **Tests P**? For technical reasons (having to deal with conditional, if-then-else, actions), we also need the standard (non-epistemic) notion of "test". The truth of a proposition **P** is "tested", without necessarily anybody knowing about it: this is an 'objective' test. Such an action can only happen if **P** does actually hold at the current state, and in this case the same state (in the same model) is returned as the output; otherwise, this program "diverges": it fails to produce any output-state. A particular case is the action $skip =: \top?$ (where $\top$ is the "true" proposition), which leaves everything unchanged.

6. **Questions P**?? This is the impersonal, public action of raising a question "Does **P** hold?" is raised. No new information about the world is added to the system: the output-state will be the same as the input-state. So, from the input-output perspective, this is exactly like the *skip* action. However, as we'll see, these actions are different from the point of their internal epistemic structure; in a questioning action, the agents have some information *about the action itself* (though not about the world): they learn that *an issue has been raised*.

Operations with Epistemic Programs. We shall consider various natural operations with epistemic programs:

- The standard *regular program constructors*: given two programs π, ρ, it is natural to consider their *sequential composition* $\pi \cdot \rho$ ("do first π then ρ"), their *non-deterministic sum* $\pi + \rho$ ('choice': "do either π or ρ") and the *iteration* (Kleene star) π^* of a program π ("do π some finite number of times", definable for instance as the infinite non-deterministic sum of all the possible sequential compositions of finitely many copies of π).
- It is also natural to consider *recursion* (fixed points): given equations (or systems of equations) involving epistemic programs, of the form $X = \pi(X)$, we may consider the solution $< X | X = \pi(X) >$ of this equation. We will of course need to restrict the shape of the equations, imposing an appropriate *guardedness* condition.
- *Private Updating.* For any agent a and program π, we consider the program π^a of "agent a privately updating her information state with π". This is an action in which agent a "learns" program π (without this program π necessarily happening): agent a comes to believe (or "suspect") that program π might be currently running. Hence, so far as a is concerned, everything goes on in this action *as if* π is actually running: a's state of information is changed according to π. But this is a 'fully' private action of "gratuitous suspicion": so far as all the other agents are concerned, nothing really happens (or, if you like, they believe that the program *skip* is happening).
- *Answering a Question.* We may consider an operation that takes any public question $\pi = \mathbf{P}??$ and returns the action of (truthful, impersonal) *public answering* $Ans!(\pi) =: \mathbf{P}!+\neg\mathbf{P}!$. This is the public announcement of the correct answer: indeed, the first action ($\mathbf{P}!$) can only happen when $\mathbf{P}$ (its precondition) is actually true, and then it consists of publicly announcing the truth of $\mathbf{P}$; similarly for the action $\neg\mathbf{P}!$. As a result, the non-deterministic sum $\mathbf{P}! + \neg\mathbf{P}!$ is in fact a deterministic action, which announces $\mathbf{P}$ if it is true, and else announces $\neg\mathbf{P}$. One can generalize this to a operation of *private answering inside a subgroup* A: put $Ans!_A(\pi) =: \mathbf{P}!_A + \neg\mathbf{P}!_A$.
- *"Interception of the secret message": Private Discovery of the "Real" Action.* While a (possibly non-deterministic) π is taking place, a group A uncovers what is really going on, including what is the actual deterministic resolution of π. The outsiders do not know about this, so they see everything exactly as if π was going on "undisturbed". This program, called $Ans!_A(\pi)$, generalizes the private answering of a question: while the non-deterministic program π is going on, the insiders are announced the answer to the question "what is really going on?". For them, there is no non-determinism or uncertainty left: they know precisely what's happening, and how it'll turn out. If we apply this to the questioning program $\pi = \mathbf{P}??$, we obtain the previous example. But this operation generalizes many others: private (and public) announcements $\mathbf{P}!_A$ are also particular cases of this: we have $\mathbf{P}!_A = Ans!_A(\mathbf{P}?)$. Also, "private announcement with secret interception (wiretapping) by outsiders" is another special case, obtainable as

$Ans!_B(\mathbf{P}!A)$, where B is the group of outsiders doing the wiretapping (and A, as before, is the group to whom the private announcement is addressed).

Basic Syntax for Epistemic Programs: We choose a few operations as our basic program constructors. We start with a countable list of *program variables* $X, Y, \ldots$ and, for any finite list $\boldsymbol{X}$ of variables, we construct *program expressions* π, σ in the free variables $\boldsymbol{X}$.

$$\pi ::= \; X \mid \mathbf{P}? \mid \pi + \sigma \mid \pi \cdot \sigma \mid \pi^* \mid \pi^a \mid < X | X = \pi >$$

where X is any variable, $a \in Ag$ is any agent, and in the expression $< X|X = \pi >$, the variable X is *epistemically guarded*, i.e. occurs only inside subexpressions of the form σ^a (for some agents a). A variable X is *closed* in an expression π if it occurs only inside subexpressions of the form $< X|X = \sigma >$ (for some other expressions σ). A program expression π is *closed* if all variables are closed in it. Closed program expressions are sometimes also called simply *programs*. Given an finite list $\pi_1, \ldots, \pi_n$ of programs, we use the abbreviations $\sum_{i=1}^{n} \pi_i =: \pi_1 + \cdots \pi_n$ and $\prod_{i=1}^{n} \pi_i =: \pi_1 \cdot \cdots \cdot \pi_n$. We also define the programs $skip =: \top?$ and $crash =: \bot?$ (where $\top$ is the true proposition and $\bot$ is the false one), and define a *questioning program* $\mathbf{P}?? =: \mathbf{P}? + \neg\mathbf{P}?$. In other words, the (impersonal, public) action of raising a question is taken to be the non-deterministic choice between testing $\mathbf{P}$ and testing $\neg\mathbf{P}$.

3 Relational Semantics: Epistemic Updates

The update semantics captures *only the "external", input-output behaviour* of epistemic programs: it tells us, for given program and a given input-state in an epistemic model, how will the program "update" the model, and what are its possible output-states (living in the updated model); it gives, for each program, a corresponding transition relation between any input epistemic model and some output model. In conclusion, it associates to each epistemic program π an *epistemic update* $\pi^\sharp$, which contains all the above information. I will give here three equivalent versions of this notion, playing roles which are roughly analogous to the three definitions of the notion of epistemic proposition.

Preliminary Notions. Given a relation $R \subseteq S \times T$, its *range function* $\hat{R} : S \to \mathcal{P}(T)$ is defined by: $\hat{R}(s) =: \{t \in T : (s,t) \in R\}$. Also, the *powerset-lifting of the relation* R is a relation $R^{\mathcal{P}} \subseteq \mathcal{P}(S) \times \mathcal{P}(T)$, defined by putting $(S', T') \in R^{\mathcal{P}}$ iff: $\forall s \in S' \exists t \in T'$ s. t. $(s,t) \in R$ and $\forall t \in T' \exists s \in S$ s. t. $(s,t) \in R$.

An *update operator* (or simply, an *update*) is a pair of operations $\mathbf{r} = (\mathbf{S} \mapsto \mathbf{S^r}, \mathbf{S} \mapsto \mathbf{r_S})$, the first being an *update map* taking any epistemic model $\mathbf{S}$ to an updated model $\mathbf{S^r}$, and the second giving an *update relation* $\mathbf{r_S} \subseteq \mathbf{S} \times \mathbf{S^r}$ between the two models. The update relation can be understood as a "transition relation", describing which states go to which states via a given program: but it is a transition relation *between models*. Using powerset-lifting,

we can see that an update operator can be alternatively described as a pair $\mathbf{r} = (\mathbf{S} \mapsto \mathbf{S^r}, \mathbf{S} \mapsto \hat{\mathbf{r}}_\mathbf{S})$, where $\mathbf{S^r}$ is as above and $\hat{\mathbf{r}}_\mathbf{S} : \mathbf{S} \to \mathcal{P}(\mathbf{S^r})$ is a non-deterministic "transition map" This is essentially the definition of "epistemic updates" in [BMS]. However, in order for an update to be *epistemically meaningful*, it needs to satisfy some extra condition, requiring that the given program behaves well with respect to (epistemic) bisimulation: An update operator $\mathbf{r} = (\mathbf{S} \mapsto \mathbf{S}(\mathbf{r}), \mathbf{S} \mapsto \mathbf{r_S})$ is said to *preserve epistemic bisimilarity* if it has the property that: if $(\mathbf{S}, s) \sim (\mathbf{T}, t)$ then, for every $s' \in \mathbf{S^r}$ such that $(s, s') \in \mathbf{r_S}$, there exists some $t' \in \mathbf{T^r}$ such that $(t, t') \in \mathbf{r_T}$ and $(\mathbf{S^r}, s') \sim (\mathbf{T^r}, t')$.

Proposition 3.1 *Given an update operator* $\mathbf{r}$*, the following are equivalent:*

- $\mathbf{r}$ *preserves epistemic bisimilarity*
- *for all models* $\mathbf{S}, \mathbf{T}$ *and states* $s \in S, t \in T$*, we have that*[10] *:* $s \sim t$ *implies* $\hat{\mathbf{r}_\mathbf{S}}(s) \sim^{\mathcal{P}} \hat{\mathbf{r}_\mathbf{T}}(t)$.

This leads to our

First Definition of Epistemic Updates: An *epistemic update* is defined as an update operator which preserves epistemic bisimilarity.

It is easy to see that bisimulation-preservation is equivalent to requiring *functoriality*, and respectively *naturality*, of the first, respectively second, components of an epistemic update. This leads to the

Second Definition[11] **of Epistemic Updates:** Let $U : \mathbf{Mod} \to Set$ be the forgetful functor from the category of models (=F-coalgebras) to the category of sets. An *epistemic update* is a pair $\mathbf{r} = (\uparrow^\mathbf{r}, \hat{\mathbf{r}})$, consisting of an *endofunctor* $\uparrow^\mathbf{r}$: $\mathbf{Mod} \to \mathbf{Mod}$, on the category of F-coalgebras, together with a *natural transformation* $\hat{\mathbf{r}} : U \to \mathcal{P} \circ U \circ \uparrow^\mathbf{r}$. (Here, $\mathcal{P}$ is the powerset endofunctor on Set and $\circ$ is the composition of two functors.)

Finally, we can provide an analogue for updates of the third definition of epistemic propositions. By going to the final coalgebra $\Omega = \Omega_F$, we can again factor over bisimilarity:

Third Definition of Epistemic Updates. An *epistemic update* is just a function $\hat{\mathbf{r}}_\Omega : \Omega \to \mathcal{P}(\Omega)$, mapping every state in the final coalgebra to a set of such states. In other words, it is a $\mathcal{P}$-Coalgebra for the powerset functor, having as support the final F-coalgebra $\Omega = \Omega_F$. Obviously, we can associate to any such function $\hat{\mathbf{r}}_\Omega$ a binary relation $\mathbf{r}_\Omega \subset \Omega \times \Omega$ on the final coalgebra, defined by $\mathbf{r}_\Omega = \{(x, y) \in \Omega \times \Omega : y \in \hat{\mathbf{r}}_\Omega(x)\}$. We interpret this relation as a *transition relation* between the states of the final coalgebra. Obviously, there is a bijective correspondence between functions from Ω to $\mathcal{P}(\Omega)$ and transition relations on Ω, so we can identify epistemic updates $\hat{\mathbf{r}}_\Omega$ with such

[10] Here, $\sim \subseteq \mathbf{S} \times \mathbf{T}$ is the relation of bisimilarity, i.e. the largest bisimulation, between the two models.

[11] This can be viewed as a generalization of the semantics of modal operators in terms of natural transformations given in [K2].

binary Relations $\mathbf{r}_\Omega$ on Ω.

Again, it is easy to see that the above three definitions of "epistemic updates" are *equivalent*: there exist canonical bijective correspondences between the objects falling under any two of the above definitions. As before, by systematic ambiguity, we use the same name ("updates") and same notation for all three notions. The point of having all three is that each can make sense in different contexts. When we want to see how a given epistemic program π does actually affect ("update") a specific situation, it is more convenient to use the first definition; when proving category-theoretical properties, the second definition is more useful. Finally, the third definition offers the simplest way to define (by corecursion) a relational semantics for various epistemic programs.

Example. The update semantics of a public announcement: Given an epistemic proposition $\mathbf{P}$, the epistemic update (in the sense of our first definition) $\mathbf{P}!^\sharp = (\mathbf{S} \mapsto \mathbf{S}^{\mathbf{P}!}, \mathbf{S} \mapsto \mathbf{r}_{\mathbf{S}}^{\mathbf{P}!})$ associated to a public announcement $\mathbf{P}!$ is defined by: $\mathbf{S}^{\mathbf{P}!} =: \mathbf{S}^{\mathbf{P}}$, where $\mathbf{S}^{\mathbf{P}}$ is the *relativization* of the model $\mathbf{S}$ to proposition $\mathbf{P}$, i.e. the *restriction* of the model $\mathbf{S}$ to the set $\mathbf{P}_{\mathbf{S}}$; [12] and $\mathbf{r}_{\mathbf{S}}^{\mathbf{P}} =: \{(s,t) \in \mathbf{S} \times \mathbf{S}^{\mathbf{P}} : s = t\} = \{(s,s) : s \in \mathbf{S}(\mathbf{P}!)\}$, seen as a binary relation between $\mathbf{S}$ and $\mathbf{S}^{\mathbf{P}}$.

The semantics can be made even simpler by using the third definition: in this version, the epistemic update induced by the public announcement $\mathbf{P}!$ is given by a *partial function* $f : \Omega \to \Omega$, having as domain the class $\mathbf{P}_\Omega =: \{s \in \Omega : \mathbf{P}_\Omega(s) = 1\} = \{\omega(\mathbf{S}, s) : s \in \mathbf{P}_{\mathbf{S}}\}$, where recall that $\omega =: \omega_F$ is the map from **pMod** to Ω_F induced by the final map; and whose values are defined by the *corecursive definition*: $f(s)_0 =: s_0$, $f(s)_a =: \{f(t) : t \in s_a \cap dom(f)\}$, for all states $x \in dom(f) = \mathbf{P}_\Omega$. Since the program $\mathbf{P}!$ of public announcement is *deterministic*, the update $\mathbf{P}!^\sharp_\Omega$ induced by this program could in fact be identified with the partial function f defined above. But, formally, they are distinct, and the update $\mathbf{P}!^\sharp_\Omega : \Omega \to \mathcal{P}\Omega$ is defined putting $\mathbf{P}!^\sharp_\Omega(s) =: \{f(s)\}$ if $\mathbf{P}_{\mathbf{S}}(s)$ holds, and $\mathbf{P}!^\sharp_\Omega(s) =: \emptyset$ otherwise.

Update Equivalence. Two epistemic updates $\mathbf{r} = (\mathbf{S} \mapsto \mathbf{S}^{\mathbf{r}}, \mathbf{S} \mapsto \mathbf{r}_{\mathbf{S}}$ and $\mathbf{r}' = (\mathbf{S} \mapsto \mathbf{S}^{\mathbf{r}'}, \mathbf{S} \mapsto \mathbf{r}'_{\mathbf{S}}$ are *equivalent* if, for every pointed model $(\mathbf{S}, s)$, we have: $\mathbf{S}^{\mathbf{r}} \sim \mathbf{S}^{\mathbf{r}'}$ and $\hat{\mathbf{r}_{\mathbf{S}}}(s) \sim^{\mathcal{P}} \hat{\mathbf{r}'_{\mathbf{S}}}(s)$. We write $\mathbf{r} \equiv \mathbf{r}'$ for equivalent updates. We say that two programs π, σ are *update equivalent*, and write $\pi \equiv \sigma$, if their updates are equivalent: $\pi^\sharp \equiv \sigma^\sharp$. Update equivalent programs are equivalent from a purely "external", input-output point of view.

If we use the third semantics and think of updates as Functions on Ω_F, then update equivalence is just *identity*: $\mathbf{r} \equiv \mathbf{r}'$ iff $\mathbf{r}_\Omega = \mathbf{r}'_\Omega$. So: *if our notion of equivalence for programs is update equivalence, then our third version of update semantics is fully abstract.*

Formal Semantics. For each closed program expression π we define an

[12] More explicitly, the *relativization* $\mathbf{S}^{\mathbf{P}} = (S^{\mathbf{P}}, \{\cdot^{\mathbf{P}}_a\}_{a \in Ag}, \cdot^{\mathbf{P}}_0)$ of a model $\mathbf{S} = (S, \{\cdot_a\}_{a \in Ag}, \cdot_0)$ to a proposition $\mathbf{P}$ is given by $S^{\mathbf{P}} =: \mathbf{P}_{\mathbf{S}}$, $s^{\mathbf{P}}_a =: s_a \cap S^{\mathbf{P}}$ and $s^{\mathbf{P}}_0 =: s_0$, for all $s \in S^{\mathbf{P}}$. This is a standard model-theoretic notion.

update $\pi^\sharp : \Omega \to \mathcal{P}(\Omega)$, and for each program expression $\pi(X_1, \ldots, X_n)$ in free variables $X_1, \ldots, X_n$, we define a function $\pi^\sharp$ of n variables, taking n-tuples of updates into an update. The definition is recursive (and compositional). We skip here the details concerning the semantics of free expressions, but only mention the clauses for closed expressions: $(\mathbf{P}?)^\sharp(s) =: \{s\}$, if $\mathbf{P}(s)$ holds, and $(\mathbf{P}?)^\sharp(s) =: \emptyset$ otherwise; $(\pi + \sigma)^\sharp(s) =: \pi^\sharp(s) \cup \sigma^\sharp(s)$, $(\pi \cdot \sigma)^\sharp =: \sigma^\sharp(\pi^\sharp(s))$, $(\pi^*)^\sharp(s) =: \bigcup\{(\pi^\sharp)^n(s) : n \geq 0\}$; $(\pi^a)^\sharp(s) =: \{t\}$, where t is given by: $t_0 = s_0$, $t_a = \bigcup\{\pi^\sharp(s') : s' \in s_a\}$ and $t_b = s_b$ for all $b \neq a$. Finally, if π is a expression in the free variable X only and if $\pi^\sharp$ is the corresponding unary function from updates to updates, then $< X|X = \pi >^\sharp$ is the unique fixed point [13] of the function $\pi^\sharp$.

Dynamic Modalities: from Programs to Propositions. Every epistemic update gives rise to a *modality*: a unary proposition operator. Given an update $\mathbf{r} = (\mathbf{S} \mapsto \mathbf{S}(\mathbf{r}), \mathbf{S} \mapsto \mathbf{r_S})$ and a proposition $\mathbf{P}$, we can construct a new proposition $[\mathbf{r}]\mathbf{P}$, saying that "after the update $\mathbf{r}$, $\mathbf{P}$ will surely hold" (at any possible output-state). This can be defined by saying that $([\mathbf{r}]\mathbf{P})_\mathbf{S}(s)$ holds iff: $\mathbf{P}_{\mathbf{S^r}}(t)$ holds for every t such that $(s,t) \in \mathbf{r_S}$. For programs π we can thus define dynamic modalities, by setting $[\pi]P =: [\pi^\sharp]P$, meaning that "after every execution of π (at the current state), $\mathbf{P}$ will hold". If we use the third semantics (thinking of propositions as predicates $\Omega \to 2$ on the final Coalgebra, and of updates as $\mathcal{P}$-coalgebras on Ω) then this can be simplified to: $[\pi](P)(s)$ holds iff $P(t)$ holds for every $t \in \pi^\sharp(s)$.

Proofs by Coinduction. We can prove many update identities, by a mixture of coinduction on Ω and functional reasoning (pointwise identity). For example, using the above coinductive definition of $\mathbf{P}!^\sharp$, we can check that:

$$\mathbf{P}! \equiv \mathbf{P}? \cdot \prod_{a \in Ag} \mathbf{P}!^a$$

This implies that we have the equivalence: $\mathbf{P}! \equiv < X|X = \mathbf{P}? \cdot \prod_{a \in Ag} X^a >$, which is why we chose to not have public announcements $\mathbf{P}!$ in our basic syntax: they are already definable in our syntax, up to update equivalence.

Another update identity is: $\mathbf{P}?? \equiv skip$, which shows that (impersonal, public) questions induce *no* informational changes on a given epistemic model.

Disadvantages of the Update Semantics. Identifying the meaning of an epistemic program with the epistemic update induced by it may be a convenient and economical choice in many cases, but in general it runs in a number of problems. As already mentioned, they only describe the informational features of an epistemic program from a *purely 'external', input-output point of view*. They do not explicitly tell us anything about the *inherent informational features* of the program itself: what do agents know/see/suspect about the program itself.

[13] i.e. the unique solution of the update equation $x \equiv \pi^\sharp(x)$. The "epistemic guardedness" condition guarantees the existence and uniqueness of the solution.

One consequence of this is that the update semantics is *not an adequate semantics for questions*: the last update identity mentioned above shows that update semantics will necessarily identify every question with *skip*. Although on one hand, we do feel indeed that questions do not necessarily add information about the state of the world, they carry information about themselves: they *publicly raise an issue*. This is absent in the update semantics, and as a result the operation of *answering a question* (as defined above) *does not make any sense* in update semantics: there simply is no operation on updates which will map $\mathbf{P}? + \neg\mathbf{P}?$ into $\mathbf{P}! + \neg\mathbf{P}!$. This is because the first program is always update equivalent to *skip*: the information about $\mathbf{P}$ (about which question was asked, which issue was raised) has disappeared. More generally, there is *no way* to define the operation $Ans!_A$ of *interception of the message*, or of "discovery" of the "real" action taking place. This operation is again inconsistent with update equivalence [14].

Another problem is that, for complicated actions, it is extremely hard *to get from the informal description of an epistemic action to its formal update semantics*. In general, we need for each such (informal description of) complicated epistemic action, to give an algorithm to compute, for *any* arbitrary epistemic model, the way this action will change the model. This is a complicated business. And it is in any case very *hard to visualize* epistemic programs in terms of their updates: these updates are huge objects, infinite Functions acting on $\mathbf{Coalg}(\mathbf{F})$.

Finally, update semantics doesn't give us a clue towards *finding the right syntax for epistemic programs*. The most important ingredients in that syntax are the *epistemic ones*, which do not arise very naturally as operations with updates: the operation of private updating π^a with a program and the operation of solving "epistemically guarded" systems of program equations. The intuitions underlying for these operations lead us to an *alternative, more refined* semantics for epistemic programs: the coalgebraic semantics.

4 Coalgebraic Semantics: Epistemic Programs as Coalgebras

To explicitly formalize the inherent informational/epistemic aspects of a program, we will now consider *epistemic models for programs*: the underlying coalgebraic structure of the program itself. For this, we need to consider, in addition to the functor $F = T_\Phi$, a different functor $G = T_\Psi$, obtained by changing our initial choice (fixed until now) of the set Φ. Namely, instead of the set Φ of atomic sentences, we take as our set $\Psi =: \Omega = \Omega_F$ the final coalgebra of the functor F. Then $Pred(\Psi) = Pred(\Omega_F) = Prop$ is nothing but the *set of all epistemic propositions*. Taking now $G = T_\Psi$, we consider as before G-coalgebras, as a new kind of epistemic models. But these are to be

[14] or more precisely, update equivalence is not a congruence with respect to this operation

thought as *dynamic models*: they are models for *actions* and programs.

An *epistemic action structure* is a G-coalgebra $\Sigma = (\Sigma, \epsilon)$, with $\epsilon : \Sigma \to G(\Sigma)$, for the above functor $G = T_\Psi$. As before, whenever ϵ is understood, we use the notations $\sigma_0 =: \epsilon(\sigma)_0$ and $\sigma_a =: \epsilon(\sigma)_1(a)$, for all agents a. In an action structure, the members of Σ are called *simple actions*, σ_0 is called the *precondition of action* σ, and the set σ_a is called the *appearance of* σ *to agent* a. Note that the precondition σ_0 is a *proposition*. Intuitively, this proposition gives the *condition of possibility* for action σ: this action can be executed at a state s in an epistemic model $\mathbf{S}$ iff the state satisfies the precondition of the action; i.e. iff $(\mathbf{S}, s) \models \sigma_0$, or equivalently if $(\sigma_0)_{\mathbf{S}}(s)$ holds. As for the appearance sets, they express the agents' state of knowledge (or belief) *about the action itself*. The actions $\sigma' \in \sigma_a$ are the *epistemic alternatives of the action* σ *to agent* a. As before we can write $\sigma \to_a \sigma'$, instead of $\sigma' \in \sigma_a$, and call this the (epistemic) *indistinguishability relation for agent* a: if the current action happening is σ, then the agent a would think that any of the actions $\sigma' \in \sigma_a$ might be happening.

This is just a variation of our initial setting obtained by changing T_Φ with $G = T_\Psi$, so all the notions carry over: we have essentially the same notion of *epistemic bisimilarity* (but now between actions) and a final coalgebra Ω_G for the category of action structures.

Definition (I) Semantically, an *epistemic program* (or a *program model*) is a pair $\pi = (\Sigma, \Gamma)$, consisting of an action structure (G-coalgebra Σ) and a *subset* $\Gamma \subseteq \Sigma$ of designated actions, called the *currently possible actions*.

Observe that program models are a kind of dynamic analogue of pointed (state) models, but having a *set* of designated 'points' instead of only one. The reason has to do with *non-determinism*; indeed, *deterministic actions will correspond precisely to pointed action structures*, i.e. program models having a singleton as designated state. So deterministic actions are in bijection with the members of Σ. But *non-deterministic* programs will have a maybe bigger set of designated actions: these correspond to all *deterministic resolutions* of program π. When executing π, any of the designated actions might be happening: they are all the *currently possible* actions [15].

We can once again factor over bisimilarity by going to the final coalgebra. Indeed, taking the final map $\omega_G : pCoalg(G) \to \Omega_G$, we can associate to each epistemic program $\pi = (\Sigma, \Gamma)$ a unique subset $|\pi| \subset \Omega_\Gamma$, given by: $|\pi| =: \{\omega_G(\Sigma, \sigma) : \sigma \in \Gamma\}$. This establishes a bijection between program models and members of $\mathcal{P}(\Omega_G)$, so that we can identify the two classes of objects. This gives a simpler version of the same notion:

Definition (II) An *epistemic program* is just a subset of the final coalgebra Ω_G.

We use letters α, β to denote the elements of Ω_G: they are the "univer-

[15] although not all *epistemically possible* actions: some agents could think that other actions in Σ may be happening).

sal simple actions". In this version, deterministic programs correspond to *singletons* $|\pi| = \{\alpha\} \subseteq \Omega_G$. We use systematic ambiguity to identify such deterministic programs with their unique currently possible action, writing in this case $\pi = \alpha$ and $|\pi| = \{\pi\}$.

Program Bisimilarity. Two programs $\pi = (\Sigma, \Gamma), \pi' = (\Sigma', \Gamma')$ are *program equivalent*, or *bisimilar*, if there exists a bisimulation $R \subseteq \Sigma \times \sigma'$ between the two action structures s.t. $(\Gamma, \Gamma') \in R^{\mathcal{P}}$ (where $R^{\mathcal{P}}$ is the powerset lifting of R). Equivalently, if $\sim \subseteq \Sigma \times \Sigma'$ is the relation of *bisimilarity* (the largest bisimulation) between the two structures, then the programs are bisimilar iff $\Gamma \sim^{\mathcal{P}} \Gamma'$. We write in this case $\pi \sim \pi'$.

One can easily check that, if we look at programs as members of $\mathcal{P}(\Omega_G)$, then program bisimilarity is just identity: $\pi \sim \pi'$ iff $|\pi| = |\pi'|$. This means that our second version of coalgebraic semantics for programs, in terms of $\mathcal{P}(\Omega_G)$, is *fully abstract for program bisimilarity.*

Program Bisimilarity is Stronger than Update Equivalence: It is easy to see that: $\pi \sim \pi' \Rightarrow \pi \equiv \pi'$, but the converse is false. The questioning program $\mathbf{P}?? = \mathbf{P}? + \neg\mathbf{P}?$ provides a counterexample: we saw it is update equivalent to *skip*, but we'll soon see the two are not bisimilar.

Examples and Pictures: from Informal Description to Coalgebraic Semantics

Unlike the update semantics, the coalgebraic semantics is really very close to the informal description of the epistemic program. Going from one to the other is rather straightforward. To represent it graphically, we draw simple actions as circles labelled with their own preconditions, and epistemic uncertainty relations as arrows; we use double circles for the designated ("currently possible") action(s).

1. **Public Announcements**. First, the program $\mathbf{P}!$ is deterministic, so only one action is currently possible: the set $|\mathbf{P}!|$ is a singleton. As announced, we write $|\mathbf{P}!| = \{\mathbf{P}!\}$. As for the action $\mathbf{P}!$ itself, the informal description of the program $\mathbf{P}!$ gives the main semantical features straight away: this is a *truthful, public announcement that* $\mathbf{P}$. Truthfulness means this action can only happen if $\mathbf{P}$ holds: so $\mathbf{P}$ is the *precondition* of this action. As for the appearance, *publicity* means that this actions "looks like itself" to all the agents: its only epistemic alternative is itself. This gives us:

$$|\mathbf{P}!| =: \{\mathbf{P}!\}$$
$$\mathbf{P}!_0 =: \mathbf{P}$$
$$\mathbf{P}!_a =: \{\mathbf{P}!\}$$

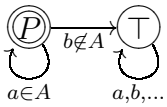
So the picture $\mathbf{P}!$ is:

Observe that the action *skip* can be considered as a particular case of this (as well as of test $\mathbf{P}?$), since we have: $skip = \top? \sim \top!$. Of course, *skip* can be defined directly by corecursion, setting: $|skip| =: \{skip\}$, $\text{skip}_0 = \top$, $\text{skip}_a = \{skip\}$.

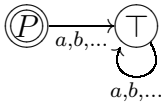
2. **Private Announcements to Subgroups**. The only difference from the previous one is that now the appearance of the action to the 'outsiders' is *skip*: for them, nothing happens:

$$\begin{aligned} |\mathbf{P}!_A| &=: \{\mathbf{P}!_A\} \\ (\mathbf{P}!_A)_0 &=: \mathbf{P} \\ (\mathbf{P}!_A)_a &=: \{\mathbf{P}!_A\} \quad \text{for } a \in A \\ (\mathbf{P}!_A)_b &=: \{skip\} \quad \text{for } b \notin A \end{aligned}$$

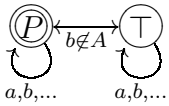
which gives the picture



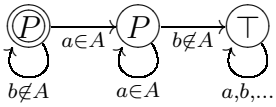
3. **Tests**. Observe that "objective" (non-epistemic, "invisible") tests $\mathbf{P}?$ are a particular case of the previous example: namely, when the set A of 'insiders' is empty. Everybody is an outsider:



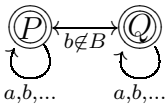
4. **Private Announcements with Suspicious Outsiders**. The same as the previous one, except that the outsiders b consider both *skip* and the secret announcement as possible, so the appearance to the outsiders consists of two actions: the secret announcement and the *skip* action.



5. **Private Announcements with (Secret) Interception by Outsider**. In this case, the outsider is in control, he "sees" the action as it is: so this action appears as itself; while this time looks to the 'insiders' as... fully private announcement to their subgroup: so it looks like the action in Example 2 above.



6. **Non-deterministic programs** The non-deterministic program in which the 'insiders' are privately announced either that **P** or that **Q**: maybe a coin is thrown, and depending on its outcome, one or the other is announced. These are the "rules of the game" and they are *public*: the outsiders (like everybody else) know that one of these two announcements is going on, but don't know which. But the 'insiders' hear the message, so they know.



Non-destructive Programs. Observe that all the above programs have the property that all agents stay "alive" throughout the program: i.e. $\alpha_a \neq \emptyset$ for any $\alpha \in \Sigma, a \in AG$. We call these kind of programs *non-destructive*. They correspond to updates in which no live agents are killed: i.e. if $s_a \neq \emptyset$ and $(s,t) \in \mathbf{r}$ then $t_a \neq \emptyset$. We are primarily interested in non-destructive programs; observe that all the programs which can be defined in our formal syntax are non-destructive.

Defining Program Operations by Corecursion:

Defining main operations by corecursion is also much simpler in coalgebraic semantics than in update semantics. But we have to assume that our original set Ψ of propositions is closed under Boolean operations and modalities (including dynamic modalities). Non-deterministic sum is just union (of subsets of Ω_G):

Sum: $|\sum_{i\in I} \pi_i| =: \bigcup_{i\in I} |\pi|$.

Private Updating. Given a program π and an action a, we define the program π^a by

$$\begin{aligned} |\pi^a| &=: \{\pi^a\} \\ (\pi^a)_0 &=: \top \\ (\pi^a)_a &=: |\pi| \\ (\pi^a)_b &=: \{skip\} \quad \text{for } b \neq a \end{aligned}$$

Sequential Composition. Similarly, we define the operation $\cdot : \Omega_G \times \Omega_G \to \Omega_G$ of sequential composition of simple actions α and β (which is again a simple, deterministic action!) and then the sequential composition $\pi \cdot \sigma$ of two programs.

$$
\begin{aligned}
(\alpha \cdot \beta)_0 &=: \quad \alpha_0 \wedge [\alpha]\beta_0 \\
(\alpha \cdot \beta)_a &=: \{\alpha' \cdot \beta' : \alpha' \in \alpha_a, \beta' \in \beta_a\} \\
|\pi \cdot \sigma| &=: \{\alpha \cdot \beta : \alpha \in |\pi|, \beta \in |\pi|\}
\end{aligned}
$$

Iteration: We define $\pi^* =: \sum_{n \geq 0} \pi^n$.

Answering Questions, Intercepting Messages, "Discovering What's Going On". Now we can finally define the operation $Ans!_A\pi$, by which, while π is happening, the agents in A find out what is going on (i.e. what is the real deterministic resolution of π). As we saw earlier, this generalizes the operations of answering a question, secretly intercepting a private message, discovering on which side has fallen etc. But we also saw that this operation could *not* be defined in update semantics, since *didn't make sense there*: it requires distinguishing between update equivalent, but non-bisimilar, programs. We need to define it for both programs π and simple actions α.

$$
\begin{aligned}
|Ans!_A\pi| &=: \{Ans!_A\alpha : \alpha \in |\pi|\} \\
(Ans!_A\alpha)_0 &=: \quad \alpha_0 \\
(Ans!_A\alpha)_a &=: \quad \{Ans!_A\alpha\} \qquad \text{for } a \in A \\
(Ans!_A\alpha)_b &=: \quad \alpha_b \qquad \text{for } b \notin A
\end{aligned}
$$

Solving Fixed-Point (Systems of) Equations. In the coalgebraic semantics, it is much easier (than in the update semantics) to solve (now, up to program bisimilarity!) systems of "epistemically guarded" equations. In fact, if for a second we look at the coalgebraic pictures of program as if they were processes, we can easily see the notion of "epistemic guardedness" is just the analogue of the standard notion of guardedness. So the existence of unique solutions up to bisimulation for such guarded equations follows from the corresponding classical results for processes.

Formal Semantics. Using the above-defined basic programs and operations with programs, it it straightforward to recursively define the coalgebraic semantics for our basic program syntax: closed program expressions are interpreted as programs, and expressions with n free variables as n-ary functions from programs to programs. The semantics is compositional, and *fully abstract with respect to program bisimilarity.*

Proofs by Coinduction. Proving by coinduction program identities (up to bisimilarity) is even more straightforward than in the update semantics. The following are easy exercises:

$$\mathbf{P}! \sim \mathbf{P}? \cdot \prod_{a \in Ag} \mathbf{P}!^a$$

$$Ans!_A(\mathbf{P}??) \sim \mathbf{P}!_A + \neg\mathbf{P}!_A$$

where $\mathbf{P}?? =: \mathbf{P}?+\neg\mathbf{P}?$; and finally, for all *non-destructive programs*, we have:

$$\pi \sim \sum_{a\in|\pi|} \alpha_0? \cdot \prod_{a\in Ag} (\alpha_a)^a$$

where $\alpha_a =: \sum\{\beta : \beta \in \alpha_a\}$ is the "apparent action" (the way action α appears [16] to a). This is another action, denoted by systematic ambiguity in the same way as the *set* α_a (the "appearance" of α to a). This last identity allows us in fact to prove a Representation Theorem, saying that our basic syntax is sufficient to present all non-destructive *finitary programs* (i.e. programs whose underlying simple action structure is finite):

Proposition 4.1 *Every non-destructive finitary program $\pi = (\Sigma, \Gamma)$ (with Σ of finite cardinality) is representable in our basic syntax up to bisimilarity, without using the operator of (Kleene) iteration *. The converse is also true: every program built in this way is finitary.*

Recovering Epistemic Updates: The "Product Update". Obviously, any semantics for programs must at least tell us the input-output behaviour of the program. In the coalgebraic semantics, this is done by associating to each epistemic program $\pi \subseteq \Omega_G$ an epistemic update $\pi^\sharp : \Omega_F \to \mathcal{P}(\Omega_F)$. This can be defined directly, by corecursion, but to make this more transparent, it is better to think of states as living in generic F-coalgebras $\mathbf{S}$ and of simple actions as living in G-coalgebras Σ. We first define a partial "update" operation $\otimes : \mathbf{Coalg}(\mathbf{F}) \times \mathbf{Coalg}(\mathbf{G}) \to \mathbf{Coalg}(\mathbf{F})$, called the *update product* $\mathbf{S} \otimes \Sigma$ of the two coalgebras:

$$\begin{aligned}
\mathbf{S} \otimes \Sigma =: &\quad \{(s,\sigma) \in S \times \Sigma : s_\in(\sigma_0)_\mathbf{S}\}\\
(s,\sigma)_0 =: &\quad s_0\\
(s,\sigma)_a =: &\quad \{(s',\sigma') \in S \otimes \Sigma : s' \in s_a, \sigma' \in \sigma\}
\end{aligned}$$

The intuition is that independent epistemic uncertainties must be 'multiplied' : if when the current state is s agent thinks that state s' might be the current state, and when the current action is α agent a thinks that α' might be the current action, then when in the output state $\cdot\alpha$ agent a thinks that $s' \cdot \alpha'$ might be the output state.

The update product is a *functor* between the product category $\mathbf{Coalg}(\mathbf{F}) \times \mathbf{Coalg}(\mathbf{G})$ and $\mathbf{Coalg}(\mathbf{F})$: for coalgebraic morphisms $f : \mathbf{S} \to \mathbf{S}'$, $g : \Sigma \to \Sigma'$, the morphism $f \otimes g : \mathbf{S} \otimes \Sigma \to \mathbf{S}' \otimes \Sigma'$ is given by $(f \otimes g)(s,\sigma) =: (f(s), g(\sigma))$. Thus, each G-coalgebra Σ will induce a *functor* $\uparrow^\Sigma$: $\mathbf{Coalg}(\mathbf{F}) \to \mathbf{Coalg}(\mathbf{F})$, given by $\uparrow^\Sigma(\mathbf{S}) = \mathbf{S} \otimes \Sigma$. If, in fact, we are given a *program* $\pi = (\Sigma, \Gamma)$ then Γ induces a

[16] Observe that a deterministic action α might "appear" to a to be non-deterministic: this is because, as far as appearances go, epistemic non-determinism *is* real determinism. If agent a has more than one epistemic alternative for α then this actions appears to him as non-deterministic.

natural transformation $\hat{\Gamma} : U \to \mathcal{P} \circ U \circ \uparrow^{\Sigma}$: indeed, for each F-coalgebra $\mathbf{S}$ we can define a $\hat{\Gamma}_{\mathbf{S}} : S \to \mathcal{P}(\mathbf{S} \otimes \Sigma)$ by putting: $\hat{\Gamma}_{\mathbf{S}}(s) =: \{(s, \sigma) \in \mathbf{S} \otimes \Sigma : \sigma \in \Gamma\}$. In this way, we recover the *epistemic update* $\pi^{\sharp} = (\uparrow^{\Sigma}, \hat{\Gamma})$ associated to our program $\pi = (\Sigma, \Gamma)$.

Towards a Generalization. Obviously, our first semantics makes sense, not just for functors of the form $F = T_{\Phi}$, but for arbitrary functors F having a final coalgebra. If for instance, we take $F = 1 + \mathcal{P}_{\kappa}(A \times Id)$, where A is an arbitrary set of "actions", 1 is the constant functor associated to a singleton set $1 = \{*\}$ and Id is the identity functor on Set, then F-coalgebras (S, E) can be interpreted as *process with actions in A and possibility of deadlock*. Any $s \in S$ with $e(s) = *$ stands for "deadlock" δ. Then updates can be interpreted as *process transformations*. An example of such transformation is *encapsulation*: blocking all actions in a set $H \subseteq A$, i.e. renaming them to deadlock δ. Other examples are *sequential, or parallel, composition with a fixed process*. The Larsen-Skou functor L is another example, which actually is closer to ours, since it is of the form T_{Φ}: for a set S put $T(S) = \{\mu : \mu : S \to [0, 1]$ is a finite measure with $\sum_{s \in S} \mu(s) = 1\}$, and for a map $f : S \to S'$, put $Tf : T(S) \to T(S')$, $(T(f)\mu)(s') =: \sum_{a \in f^{-1}(b)} \mu(a)$. This gives a functor, and if in addition we fix a set Φ of "basic predicates" and a set A of "actions", we can take the functor $L = T_{\Phi}$, according to our earlier definition. This gives a notion of *probabilistic process with actions in A and predicates in* Φ, for which we get a corresponding notion of process transformation.

However, in the last case, as many others, the *second, coalgebraic* semantics can also be generalized. This is natural, if we interpret labels $a \in A$ as *agents* instead of actions, and interpret T-coalgebras (S, e) as *probabilistic epistemic models*, instead of processes: we read $e(s)(a)(s') = p$ as saying that if the current state is s, agent a believes that the probability of the current state to be s' is p. We use similar notations as above, putting $s_0 = e(s)_0, s_a = e(s)_1(a)$ Then, if we take again the set $\Psi = Pred(\Omega_{T_{\Phi}}$, we *can naturally define a "probabilistic product update"*, $\otimes : \mathbf{Coalg}(\mathbf{T}_{\Phi}) \times \mathbf{Coalg}(\mathbf{T}_{\Psi}) \to \mathbf{Coalg}(\mathbf{T}_{\Phi})$. The support set $S \otimes \Sigma$ and $(s, \sigma)_0$ are defined as above, and we put

$$(s, \sigma)_a(s', \sigma') =: \frac{s_a(s') \cdot \sigma_a(\sigma')}{\sum\{s_a(t) \cdot \sigma_a(\tau) : t \in S, \tau \in \Sigma\}}$$

This is functorial (acting on morphisms as our product update functor does), and implements the same idea of multiplying independent uncertainties, but in a probabilistic setting, with renormalization. This combines the ideas of product update with Bayesian revision. In particular, it is easy to see that classical Bayesian revision corresponds in this setting to our program of "public announcement".

Conclusions. The coalgebraic semantics gives us a more refined view of epistemic programs, exhibiting the *internal informational features of an action*. It allows us to define useful operations (answering a question, interception) with programs which weren't definable in a purely input-output semantics.

It makes easy to go from the informal description of an informational exchange to its semantics as an epistemic program. It facilitates the definition of complicated operations by corecursion, and proving identities of programs by coinduction. It has suggested a good syntax for such programs. Finally, it suggests possible generalizations.

References

[B] A. Baltag *A Logic of Epistemic Actions*. In the *Proceedings of the Workshop on "Foundations and Applications of Collective Agent Based Systems"* 11th European Summer School on Logic, Language and Information (ESSLLI'99), Utrecht University, Utrecht 1999.

[B2] A. Baltag, *Logics for Insecure Communication*. In the Proccedings of TARK'01.

[B3] A. Baltag, *A Logic for Suspicious Players: Epistemic Actions and Belief Updates in Games"* Bulletin Of Economic Research, volume 54, nr.1, January 2002, pp. 1-46)

[BMS] A. Baltag, L.S. Moss, S. Solecki, *The Logic of Public Announcements, Common Knowledge and Private Suspicions* (Extended Abstract). In the *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge* (TARK'98), pp. 43-56. Morgan Kaufmann Publishers. 1998.

[BMS2] A. Baltag, L.S. Moss, S. Solecki, *The Logic of Public Announcements, Common Knowledge and Private Suspicions*. CWI Technical Report SEN-R9922, November 1999. Submitted for publication to APAL, November 1999. Electronic version available at http://www.cs.indiana.edu/cogsci/techreps/238.html

[FHMV] R. Fagin, J, Halpern, Y. Moses, M. Vardi, *Reasoning about Knowledge*, MIT Press, 1995

[GG] J. Gerbrandy, W. Groeneveld, *Reasoning about information change*, JLLI 6 (1197) 147-169

[G] J. Gerbrandy, *Bisimulations on Planet Kripke*, Ph. D. Dissertation, University of Amsterdam, 1999.

[K] A.Kurz, D.Pattinson, *Coalgebras and Modal Logic for Parameterised Endofunctors.* CWI Technical Report, SEN-R0040, December 2000.

[K2] A.Kurz, D.Pattinson, *Definability, Canonical Models, Compactness for Finitary Coalgebraic Modal Logic.* In Lawrence Moss, editor, Coalgebraic Methods in Computer Science (CMCS'02), volume 65.1 of Electronic Notes in Theoretical Computer Science, 2002.