M4M 2007

Aximo: automated axiomatic reasoning for information update

Simon Richards[1,2]

School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom

Mehrnoosh Sadrzadeh[3]

School of Electronics and Computer Science
University of Southampton
Southampton, United Kingdom

Abstract

Aximo is a software written in C^{++} that verifies epistemic properties of dynamic scenarios in multi-agent systems. The underlying logic of our tool is based on the algebraic axiomatics of Dynamic Epistemic Logic. We also present a new theoretical result: the worst case complexity of the verification problem of Aximo.

Keywords: Automated Verification, Algebraic Axiomatics, Dynamic Epistemic Logic.

1 Introduction

One of the applications of modal logic is reasoning about the information of agents in multi-agent systems in the context of epistemic logics [7]. This field of application has been extended to information update in the context of dynamic epistemic logics [1,3,5]. In these applications, one reasons about the information of *interacting* agents who communicate with each other and get their information updated as a result. Dynamic epistemic logic (DEL) reasons about the information of these agents, the communication actions between them, and the changes induced to the information by the actions. One of the novelties of DEL is its ability to reason about honest as well as dishonest agents and their actions. It does so by using a modality that stands for a *possibly wrong belief*, this is the belief that is caused by the cheating and lying actions of dishonest agents.

[1] Support from EPSRC grant EP/D000033/1 is acknowledged by both authors.

[2] Email: sgr104@ecs.soton.ac.uk

[3] Email: ms6@ecs.soton.ac.uk

Kripke structures provide intuitive relational semantics for modal logics. One advantage of these models, other than being intuitive, is that their frame conditions directly give rise to axioms of a Hilbert-style proof system, e.g. a transitive frame gives rise to axiom 4: the so called *positive introspection* axiom of knowledge. Although semantics of DEL is based on Kripke structures, not all of its modal axioms are obtained from its frame conditions. This is because the semantics of DEL involves higher level operations that act on the Kripke structures themselves. But if one thinks in the spirit of Stone duality and moves to an algebraic semantics, the axioms corresponding to these operations can be treated on the same level as the axioms corresponding to the modalities, which are also operations on the base algebra. As a result, one directly obtains an inequational machinery to reason about information flow in multi-agent systems. It was this line of thought that led to the algebraic axiomatics of DEL, developed in the thesis of the second author [9] and jointly with Baltag and Coecke [2]. The algebra is equipped with a complete sequent calculus and a theorem that shows how to construct the algebra from the relational model of DEL. In fact, and as discussed in [2,9], one does not need the full Boolean setting of DEL to get the algebra up and running: the propositions can just be elements of a complete lattice, with no necessary internal negation and implication. Another distinguished feature of the algebra is that it encodes the epistemic and dynamic modalities as adjoint pairs that go equipped with a mechanical unfolding method. The simplicity of the algebraic axiomatics, uniformity of reasoning about dynamics and epistemics, and the mechanical nature of the adjoint modalities motivated the development of `Aximo` [8].

`Aximo` came out of the Masters' project of the first author in Computer Software Engineering, under supervision of the second author. It implements the axioms of the algebra of DEL. These consist of the axioms of (1) a distributive lattice of propositions implementing a logic on propositions; (2) a quantale of actions implementing a linear logic on actions; (3) the action of the quantale on the propositions, enriching the propositional logic with updates; (4) dynamic modalities as adjoints to update; and (5) epistemic modalities as adjoints to the endomorphisms of the pair of quantale-module. Given the specification of a scenario, `Aximo` provides automatic proofs of the dynamic epistemic properties that hold in the algebra generated from the specification. We compute the worst case complexity of the program. The complexity is based on the epistemic properties of states and actions involved in a scenario, as well as the *kernel* of each action. `Aximo` is the first automated tool for the algebra of DEL[4], we have not yet compared its efficiency with that of `DEMO` [6], which is a model checker based on the Kripke semantics of DEL. We aim to apply `Aximo` to the verification of classical and quantum security protocols, these constitute the ongoing work of the second author. Since `Aximo` has been written in a modular way, we would hope that the relevant axioms of these applications (e.g. hashes and signatures, non-local quantum co-relations) can be added on top of the existing axioms of the algebra.

[4] Because of the algebraic nature of the implemented logic and as pointed out by one of our referees, we are not sure if `Aximo` is a theorem prover or a model checker, may be it is a bit of both!

2 The Algebra

The algebra consists of a triple $(M, Q, \{app_A\}_{A\in\mathcal{A}})$ where Q is a quantale, M is its right module, and $\{app_A\}_{A\in\mathcal{A}}$ is a family of endomorphisms of the pair, i.e. $app_A\colon (M, Q) \to (M, Q)$. The signature of the quantale is $(Q, \bigvee, \cdot, \tau, \top, \bot)$ and the signature of the module is $(M, \bigvee, \bigwedge, \top, \bot)$. Moreover, the quantale acts on the module via a binary operation $- \cdot - : M \times Q \to M$. This structure has been developed and referred to as a distributive *Epistemic System* in [2,9]. We briefly recall it here by stating its axioms. We only state the binary versions of the axioms to keep the presentation close to a logical view of the algebra[5].

- **Disjunction and conjunction.** The axioms for joins, meets, $\top$ and $\bot$ in both quantale and module are the usual axioms, that is the Tarski-style semantic axioms for disjunction, conjunction, true and false. For instance we have $m \leq \top, \bot \leq m$ and also

$$If \quad m \leq m'' \quad and \quad m' \leq m'' \quad then \quad m \vee m' \leq m''$$
$$If \quad m \leq m' \quad and \quad m \leq m'' \quad then \quad m \leq m' \wedge m''$$

- **Distributivity of composition over disjunction.** The quantale has a non-commutative resource sensitive binary operation[6] $- \cdot - \colon Q \times Q \to Q$. It preserves joins (in both arguments) and has a neutral element τ satisfying the following

$$(q_1 \vee q_2) \cdot q' = (q_1 \cdot q') \vee (q_2 \cdot q') \qquad q \cdot \tau = \tau \cdot q = q$$

- **Distributivity of the action over disjunctions.** The action of quantale on the module preserves the disjunctions of both the quantale and the module

$$(m_1 \vee m_2) \cdot q = (m_1 \cdot q) \vee (m_2 \cdot q), \qquad m \cdot (q_1 \vee q_2) = (m \cdot q_1) \vee (m \cdot q_2)$$

  It is associative over the composition of the quantale and preserves its unit

$$m \cdot (q \cdot q') = (m \cdot q) \cdot q', \qquad m \cdot \tau = m$$

- **Dynamic adjunction.** Since the action of the quantale preserves arbitrary joins, it has Galois right adjoints in both argument that preserve arbitrary meets and $\top$. We are interested in the adjoint in the first argument, which is denoted by $[q]m$ and satisfies the following axiom

$$m \cdot q \leq m' \quad iff \quad m \leq [q]m'$$

- Each element of the family of endomorphisms $\{app_A\}_{\mathcal{A}}$ is a pair of endomorphisms: one on the module and one on the quantale, that is $app_A = (app_A^M\colon M \to M, app_A^Q\colon Q \to Q)$. These satisfy the following
  · **Distributivity of app_A^M over disjunction.** The endomorphisms on the module preserve the disjunction and $\bot$

$$app_A^M(m_1 \vee m_2) = app_A^M(m_1) \vee app_A^M(m_2), \qquad app_A^M(\bot) = \bot$$

  · **Distributivity of app_A^Q over disjunction.** The endomorphisms on the quantale satisfy the disjunction and $\bot$,

$$app_A^Q(q_1 \vee q_2) = app_A^Q(q_1) \vee app_A^Q(q_2), \qquad app_A^Q(\bot) = \bot$$

[5] All the axioms also holds for arbitrary meets and joins.

[6] This is indeed the tensor $\otimes$ of non-commutative multiplicative linear logic. We denote it with the same notation as for the action of Q on M.

- **Weak distributivity of app_A^Q over composition.** The weak version means that we only ask for one direction of the usual distributivity as follows
$$\tau \leq app_A^Q(\tau), \qquad app_A^Q(q \cdot q') \leq app_A^Q(q) \cdot app_A^Q(q')$$
- **Weak distributivity of app_A over action of Q on M.** The pair (app_A^M, app_A^Q) satisfies the following
$$app_A^M(m \cdot q) \leq app_A^M(m) \cdot app_A^Q(q)$$
- **Epistemic Adjunction.** Each app_A (on the quantale and on the module) has a Galois right adjoint $\Box_A$ that is meet and $\top$ preserving
$$\Box_A(x_1 \wedge x_2) = \Box_A x_1 \wedge \Box_A x_2, \qquad \Box_A \top = \top$$
- The adjunction means that we have the following axiom for app_A
$$app_A(x) \leq x' \quad iff \quad x \leq \Box_A\, x'$$

- **Stabilizers and Kernels.**
  - We distinguish a special subset of the module $Stab(Q) \subseteq M$, whose elements satisfy the following
$$\phi \in Stab(Q) \quad iff \quad \phi \cdot q \leq \phi$$
  - Each element of the quantale has a kernel $ker(q)$ in the module for which we have the following axiom
$$m \in ker(q) \quad iff \quad m \cdot q = \bot$$

3 Interpretation of the Algebra

We think of the elements of the module as logical propositions of the partial order of the module as the logical entailment between propositions. The elements of the quantale are communication actions and the join on the quantale stands for the non-determinsitic choice of actions. So the order of the quantale is the order of non-determinism. The action of the quantale on the module $m \cdot q$ is the update of information in m by action q. Its right adjoint is indeed the *weakest precondition* of Hoare logic and the *dynamic modality* of PDL and DEL. Our interpretation is thus as follows

- $[q]m$ is all the propositions that become true after applying action q to proposition m. We read it as 'after doing action q proposition m holds'.

Each app_A denotes the *appearance* of agent A as follows

- On the module, $app_A^M(m)$ is all the propositions that *appear* to agent A as true where as in reality proposition m is true. We read it as 'the *appearance* of agent A of proposition m'.
- On the quantale, $app_A^Q(q)$ is all the actions that *appear* to agent A as happening where as in reality action q is happening. We read it as 'the *appearance* of agent A of action q'.

The Galois right adjoint in each case is the epistemic modality: not necessarily truthful knowledge or possibly wrong belief (in the context where no wrong knowledge is allowed). That is

- We read $\Box_A^M m$ as 'agent A knows or believes that proposition m holds'.
- We read $\Box_A^Q q$ as 'agent A knows or believes that action q is happening'.

The set $Stab(Q)$ is interpreted as the set of *facts*. These are elements of the module that are stable under any update. The reason for stability is that our actions are epistemic and cannot change the facts of the world. If a fact is true before an action, it will stay true afterwards. The kernel of each action can be seen as its *co-precondition* or *co-content*, that is the set of states to which it *cannot* apply. For details and examples see [2,9].

From Kripke Structures to the Algebra

Assume given a Kripke structure $(S, \{R_A\}_{A\in\mathcal{A}})$ with S a set of states and $R_A \subseteq S \times S$ a family of accessibility relations. We lift this relational structure to the algebraic structure of the powerset of its states, that is $\mathcal{P}(S)$. The app_A maps arise from the accessibility relations of the Kripke structure, since there is an isomorphism between relations on a set $R_A \subseteq S \times S$ and join preserving maps on the powerset of the set $app_A : \mathcal{P}(S) \to \mathcal{P}(S)$. The isomorphisms works as follows:

$$for\ x, y \in S, \quad y \in app_A(\{x\}) \quad iff \quad (x, y) \in R_A$$

The knowledge modality defined on the subsets of states as below

$$for\ T \subseteq S, \quad \Box_A T = \{s \in S \mid app_A(s) \subseteq T\}$$

is exactly the right adjoint to the app_A map on the $\mathcal{P}(S)$, that is

$$app_A(\{s\}) \subseteq T \quad iff \quad \{s\} \subseteq \Box_A T$$

Given the set of atomic actions Σ involved in a scenario, the quantale is generated by first closing Σ under sequential composition and obtain Σ^* (the free monoid or all the words over Σ) and then taking its powerset, that is $Q := \mathcal{P}(\Sigma^*)$. Following the approach introduced in DEL [1] and to be able to encode mis-information actions such as cheating and lying, we assume there is a Kripke structure on the atomic set of actions. In an action Kripke structure $(\Sigma, \{R'_A\}_{A\in\mathcal{A}})$ we read $(\sigma, \sigma') \in R'_A$ as whenever action σ is happening, agent A thinks that action σ' is happening. These accessibility relations are point wisely extended to Σ^* and in the same way as on the module, they give rise to the appearance maps $app_A\colon \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$. The effect of the actions on the states is modeled in DEL by taking the *update product* of state and action kripke structures. We have proved in [2,9] that $(\mathcal{P}(S^*), \mathcal{P}(\Sigma^*), \{app_A\}_{A\in\mathcal{A}})$ forms an *epistemic system* where S^* is the closure of states under the update product.

4 Design, Data structures, and Complexity

Once provided with the specification of an interactive multi-agent scenario, `Aximo` provides automatic proofs for the properties of the scenario. The following information are needed for the specification of a scenario (1) the states and actions involved in the scenario, (2) the appearances of each to the agents involved, (3) the facts that each state satisfies, and (4) the kernel of each action. `Aximo` treats these assumptions as (or assumes these assumptions are) inequalities on the epistemic system generated from the states and actions. One then poses questions to `Aximo` that whether a certain property holds in the epistemic systems of the specified scenario and `Aximo` replies with a yes-no answer and provides the details of a proof or failure. The answer is provided by applying to the inputted property the generic axioms of epistemic systems (enumerated in sec.2) and the specific axioms that encode the assumptions of the specified scenario, for example the kernel and fact inequalities.

In order to see this, we go through a simple example. The generic proof steps for all scenarios are depicted in the high level flow chart of the `Aximo` (for more details see appendix). Assume a coin tossing scenario with two agents where agent 1 tosses a coin and covers it. None of the agents know on what face the coin has landed. We encode this scenario in an epistemic system $(M, Q, \{app_A\}_{A \in \mathcal{A}})$ with states $s(0), s(1) \in M$, agents $1, 2 \in \mathcal{A}$, and facts $f(0), f(1) \in Stab(Q)$. State $s(0)$ is the state in which the coin has landed heads and fact $f(0)$ is the fact saying 'the coin is heads'. We thus have $s(0) \leq f(0)$ and similarly $s(1) \leq f(1)$ for the state in which the coin is tails and its corresponding fact. Since both of the agents are uncertain about the face of the coin, we have $app_1(s(0)) = app_1(s(1)) = s(0) \vee s(1)$ and similarly for 2. Assume now that 1 uncovers the coin and publicly announces that it is heads. So we have one action $a(0) \in Q$ that appears as it is to all the agents since it is public, so $app_1(a(0)) = app_2(a(0)) = a(0)$. Since $a(0)$ is the announcement of heads, it cannot apply to the states that satisfy tails, that is $s(1) \leq ker(a(0))$ We want to prove that after the announcement of heads, agent 2's uncertainty gets waived and he gets to know the fact that the coin is heads, that is $s(0) \leq [a(0)]\Box_2 f(0)$, or in the notation of `Aximo` $s(0) \leq d(0)e(2)f(0)$. In order to prove this, `Aximo` proceeds via the following steps:

(1) By dynamic adjunction this inequality holds iff we have

$$s(0) \cdot a(0) \leq e(2)f(0)$$

(2) By epistemic adjunction this is iff

$$app_2(s(0) \cdot a(0)) \leq f(0)$$

(3) By weak distributivity of appearance over update it is enough to prove

$$app_2(s(0)) \cdot app_2(a(0)) \leq f(0)$$

(4) By assumptions of the scenario this is equivalent to

$$(s(0) \vee s(1)) \cdot a(0) \leq f(0)$$

(5) By distributivity of update over disjunction this is equivalent to

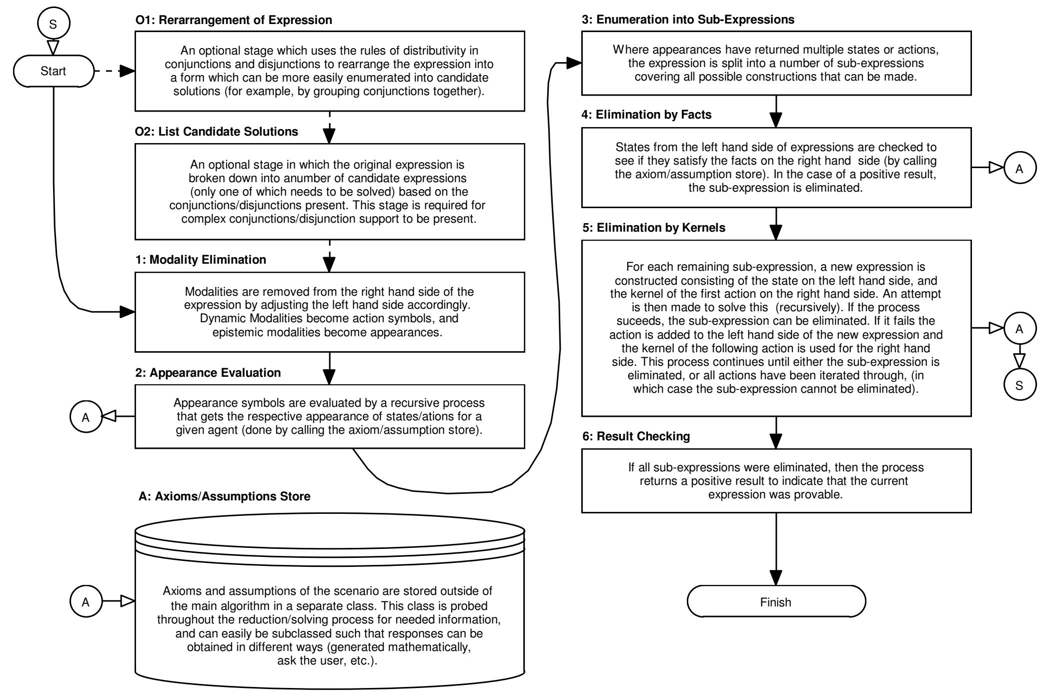
$$(s(0) \cdot a(0)) \vee (s(1) \cdot a(0)) \leq f(0)$$

(6) By the definition of disjunction, we have to prove both of these two cases

$$s(0) \cdot a(0) \leq f(0), \qquad s(1) \cdot a(0) \leq f(0)$$

(7) For the second one we have $s(1) \leq ker(a(0))$ and thus $s(1) \cdot a(0) = \bot \leq f(0)$.

(8) The first one follows since we have $s(0) \leq f(0)$ by assumptions, since update is order preserving we obtain from this $s(0) \cdot a(0) \leq f(0) \cdot a(0)$. Now since facts are stable under any update we get $f(0) \cdot a(0) \leq f(0)$.


S
Start
O1: Rerarrangement of Expression
An optional stage which uses the rules of distributivity in conjunctions and disjunctions to rearrange the expression into a form which can be more easily enumerated into candidate solutions (for example, by grouping conjunctions together).
O2: List Candidate Solutions
An optional stage in which the original expression is broken down into anumber of candidate expressions (only one of which needs to be solved) based on the conjunctions/disjunctions present. This stage is required for complex conjunctions/disjunction support to be present.
1: Modality Elimination
Modalities are removed from the right hand side of the expression by adjusting the left hand side accordingly. Dynamic Modalities become action symbols, and epistemic modalities become appearances.
2: Appearance Evaluation
Appearance symbols are evaluated by a recursive process that gets the respective appearance of states/ations for a given agent (done by calling the axiom/assumption store).
A
A: Axioms/Assumptions Store
Axioms and assumptions of the scenario are stored outside of the main algorithm in a separate class. This class is probed throughout the reduction/solving process for needed information, and can easily be subclassed such that responses can be obtained in different ways (generated mathematically, ask the user, etc.).
A
3: Enumeration into Sub-Expressions
Where appearances have returned multiple states or actions, the expression is split into a number of sub-expressions covering all possible constructions that can be made.
4: Elimination by Facts
States from the left hand side of expressions are checked to see if they satisfy the facts on the right hand side (by calling the axiom/assumption store). In the case of a positive result, the sub-expression is eliminated.
A
5: Elimination by Kernels
For each remaining sub-expression, a new expression is constructed consisting of the state on the left hand side, and the kernel of the first action on the right hand side. An attempt is then made to solve this (recursively). If the process suceeds, the sub-expression can be eliminated. If it fails the action is added to the left hand side of the new expression and the kernel of the following action is used for the right hand side. This process continues until either the sub-expression is eliminated, or all actions have been iterated through, (in which case the sub-expression cannot be eliminated).
A
S
6: Result Checking
If all sub-expressions were eliminated, then the process returns a positive result to indicate that the current expression was provable.
Finish

High Level Flow chart of `Aximo`

The assumptions of a scenario are either read from a text input file or asked from the user and assigned interactively. While computing, all the computation steps are broadcasted to the screen. If `Aximo` fails to verify the input inequality and thus produces a no answer, the user can find out where things went wrong by following the series of steps on the screen. Internally, inequalities are stored as a pair of classes each representing a list (for symbols on the left and right hand sides of the expression respectively). These are pointed to from an encapsulating class which is used to provide expression-wide functionality and handle the inequality from a single pointer. By using two seperate list classes, modifications and function calls can be made on each side of the inequality independantly from the other, reducing difficulty of implementation. The solving process uses a list of lists to keep track of it's progress. Each item in the first list represents the various candidate solutions, with the entry itself being a list to all the sub-inequalities present in that particular candidate. It is worth noting that the size of these lists can be quite dynamic, expanding as new candidates/sub-inequalities are created, and shrinking as sub-inequalities are successfully eliminated.

The complexity of the program is in direct relation with the number of cases it has to check to verify the original inequality. Assume an input inequality where we have m dynamic modalities for actions $a_1 \cdots a_m$ and n epistemic modalities for agents $1 \cdots n$, and one fact $f(l)$. After applying the dynamic and epistemic adjunctions in total $n \times m$ times, we have to verify the following

$$app_n(\cdots app_1(s)) \cdot app_n(\cdots app_1(a_1)) \cdot \cdots \cdot app_n(\cdots app_1(a_m)) \leq f(l)$$

If we take k to be the maximum number of choices in the agents' appearances of states and actions, then in the worst case each nested appearance expression, e.g. $app_n(\cdots app_1(s))$ provides us with k^n choices. Because of distributivity between disjunction and update, in the worst case one ends up to verify an exponential number of inequalities

$$(k^n)^{(1+m)}$$

If all these inequalities are true, then the original one is true. Otherwise, when the first false inequality is encountered, the program will stop and return a no answers for the original property. There are two steps to verify each choice: checking if the state satisfies the fact and using the axiom for facts. For example for an inequality of the form $s' \cdot a'_1 \cdot \cdots \cdot a'_m \leq f(l)$, the program tries to verify the following by questioning the user $s' \leq f(l)$. If this fails, that is the state does not satisfy the fact, then the program proceeds with asking the user about the kernel of the actions and then recursively calls itself to verify the kernel inequalities. The first recursive call is to verify $s' \leq ker(a'_1)$. If we assume $ker(a'_1) = e(j) \cdots e(n'-j+1)f'(l')$, then we get $k^{n'}$ cases. If one of these fails, so does the inequality $s' \leq ker(a'_1)$ and the program moves to the second expression $s' \cdot a'_1 \leq ker(a'_2)$ and so on. The worst case occurs when the program gets to the following inequality

$$s' \cdot a'_1 \cdot \cdots \cdot a'_{m-1} \leq ker(a'_m)$$

The total number of cases for verification of the kernel will be $\sum_{i=0}^{m}(k^{n'})^i$. Repeat this for all the inequalities obtained before, that is $(k^n)^{(m+1)}$ and we get

$$(k^n)^{(m+1)} \times \sum_{i=0}^{m}(k^{n'})^i = (k^n)^{(m+1)} \times \frac{1-(k^{n'})^{m+1}}{1-k^{n'}}$$

by the convergence of geometric series. We thus obtain the following

Proposition 4.1 *The complexity of* `Aximo` *is of the order $k^{2n''(m+1)}$ where m is the number of actions in the input equality, $n'' = max\{n, n'\}$ is the maximum of the number of epistemic modalities in the input inequality and in kernels of the actions of the input inequality, and k is the maximum of the number of choices in the appearance of the initial state and the actions of the input inequality.*

Proof. Directly follows from the above calculation of the number of inequalities to be verified in the worst case. □

The number of choices in the appearance of actions determines what kind of agents are involved in a scenario, for example honest, cheating, or suspicious. By fixing this number in each case, we obtain a better complexity bound:

Corollary 4.2 *In a scenario where either no one cheats or no one suspects the cheating, the complexity of* `Aximo` *becomes $k^{n''} \times m^2$. If the cheating action is suspected by at most w alternative actions by all the agents, the complexity becomes $k^{2n''} \times w^{mn''}$.*

Proof. If there is no cheating, all the actions are assumed to be public, so the appearance of them to all the agents is identity. If there is cheating but the outsiders do not suspect it, the appearance of the cheating action to the cheating agents is identity and to the outsiders is the τ action (they think nothing has happened). In both of these cases the number of cases to be verified is $k^{2n''} \times w^{mn''}$. But if the outsiders suspect the cheating, the number of choices in their appearance of the cheating action, that is our variable w, may vary. In this case the number of cases to be verified becomes $k^n \times w^m \times k^{n'} \times \sum_{i=0}^{m} (w^{n'})^i$. For instance, in the setting of security protocols the agents may suspect that either their messages were intercepted or not ($w = 2$), for more examples see [9]. $\square$

5 Test Cases

The first test case is our simple coin toss example with cheating and lying actions. The second test case is the milestone puzzle of muddy children and new versions of it with cheating and lying. These two represent the two ends of the spectra of scenarios of interest to us: in the coin toss we have one dynamic modality $m = 1$ and many epistemic modalities, where as in the muddy children, we deal with many dynamic modalities and only one epistemic modality $n = 1$.

Coin-Toss:
Consider again our above scenario and its encoding, recall that there are 2 agents, 1 throws a coin and covers it in his palm. Each agent thinks either the coin is heads $f(0)$ or tails $f(1)$. For instance the program returns a yes answer for the following inequalities

$$s(0) \leq e(1)(f(0) + f(1)), \quad s(0) \leq e(1)e(2)(f(0) + f(1))$$

In the same lines, the program will return a no answer for the following

$$s(0) \leq (e(2)f(0) \& e(1)f(1))$$

After 1 announces that the coin is heads (action 0) we have a yes to the following

$$s(0) \leq (d(0)e(2)f(0)) \& (d(0)e(1)f(0)), \qquad s(0) \leq d(0)e(2)e(1)e(2)e(1)f(0)$$

If 1 lies, we have a new action $a(2)$ in which 1 announces tails when he sees heads. The appearance of this action is identity to 1, that is $app_1(a(2)) = a(2)$, but others are not aware of the lying and think that the usual truthful announcement is happening, that is $app_2(a(2)) = a(0)$. The kernel of this action is where the coin has landed tails $s(1) \in ker(a(2))$. The program returns a yes answer to the following

$$s(0) \leq d(2)e(2)f(1), \qquad s(0) \leq d(2)e(1)e(2)f(1), \qquad s(0) \leq d(2)e(2)e(1)f(1)$$

According to the last inequality agent 2 has obtained a piece of *wrong* knowledge as a result of agent 1's lying: 2 knows that 1 knows that the coin is tails, where as 1 knows that it is heads, that is $s(0) \leq d(2)e(1)f(0)$. The output screen for the inequality $s(0) \leq d(2)e(1)f(0)$ is presented below

```
Enter expression:  s(0)<= d(2)e(1)f(0)
Solving- s(0)<= d(2)e(1)f(0)
Rearrangement/Optimisation-s(0)<= d(2)e(1)f(0)
Candidate Solutions- s(0)<= d(2)e(1)f(0)
Attempting to Solve Candidate- s(0)<= d(2)e(1)f(0)
Candidate Enumerated-s(0)<= d(2)e(1)f(0)
Dynamic Modalities Removed-s(0)a(2)<= e(1)f(0)
Epistemic Modalities Removed- app(1|s(0))app(1|a(2))<= f(0)
Apearances Evaluated-s(0,1)a(2)<= f(0)
Further Enumeration-s(0)a(2)<= f(0), s(1)a(2)<= f(0)
Parts Remaining After Elimination by Axioms-s(1)a(2)<= f(0)
Parts Remaining After Elimination by Known Solution- s(1)a(2)<= f(0)
Performing Elimination by Action Kernels Trying- s(1)<= kernel(a(2))
-as- s(1)<= f(1) - Solving s(1)<= f(1)
- Rearrangement/Optimisation- s(1)<= f(1)
- Candidate Solutions- s(1)<= f(1)
- Attempting to Solve Candidate- s(1)<= f(1)
- Candidate Enumerated- s(1)<= f(1)
- Dynamic Modalities Removed- s(1)<= f(1)
- Epistemic Modalities Removed- s(1)<= f(1)
- Apearances Evaluated- s(1)<= f(1)
- Further Enumeration-s(1)<= f(1)

- Parts Remaining After Elimination by Axioms- - *none*
- --Expression Passed--
Parts Remaining After Elimination by Action Kernels- *none*
--Expression Passed--
Try another expression?  Enter Y/N:
```

A sample output screen of Aximo

Assume we are interested in verifying the following general inequality

$$s \leq d(0)e(1)\cdots e(n)f(0)$$

In all these properties, we only have one action, that is $m = 1$ and the kernels have no epistemic modalities, that is $ker(0) = ker(2) = f(1)$, so we have $n' = 0$. By corollary 4.2, in this case one obtains a better complexity bound as shown below:

Agents	**App. of actions**	**No. cases**
honest	identity	2^n
cheating with no suspicion	singleton	2^n
cheating with suspicion	$\mid app_i(q) \mid = w$	$2^n \times w$

Muddy children with cheating and lying:

The puzzle goes like this: n' children are playing in the mud and $k' \geq 1$ of them have dirty foreheads. Each child sees other children's foreheads but cannot see his own. Their father announces 'at least one of you is dirty', and asks 'do you know if you are dirty?'. The children look around and simultaneously reply no! We prove that after $k' - 1$ rounds of no answers, all the dirty children get to know that they are dirty. This scenario is encoded in an epistemic system $(M, Q, \{app_A\}_{A \in \mathcal{A}})$ with children as agents $\{i \mid 1 \leq i \leq n'\} \subseteq \mathcal{A}$. The states of the system are denoted by s_β for $\beta \subseteq \mathcal{A}$ where each s_β represents the states in which the children in β are dirty and the children not in β are clean. The appearance of each child of each state is $app_i(s_\beta) = s_{\beta \cup \{i\}} \vee s_{\beta \setminus \{i\}}$, that is the choice of two states: in one he is dirty and in another one he is clean. The set of facts include $\{f(i) \mid 1 \leq i \leq n'\} \cup \{f(i') \mid n' + 1 \leq i' \leq 2n'\}$

where $f(i)$ stands for the fact 'child i is dirty' and $f(i')$ for the fact 'child i' is clean'. Each state satisfies its corresponding fact, that is $s_\beta \leq f(i)$ for all $i \in \beta$ and $s_\beta \leq f(i+n')$ for all $i \notin \beta$. The actions of the epistemic system are father's original announcement $a(0)$ and the children's simultaneous no answers (all encoded in the same action) $a(1)$. These actions are public announcements, so their appearances to each child is identity, that is $app_i(a(0)) = a(0)$ and $app_i(a(1)) = a(1)$. The kernel of $a(0)$ is $f(0)$, that is the fact standing for 'no child is dirty'. The kernel of the no answers $a(1)$ is $\bigvee_{i=1}^{n} e(i)f(i)$, that is the state in which the children know that they are dirty. After encoding the above assumptions in `Aximo`, it provides a yes answer to the following inequality for all $1 \leq i \leq k'$

$$s_{\{1,2,\cdots,k'\}} \leq d(0)\,\underbrace{d(1)\cdots d(1)}_{k'-1}\,e(i)f(i)$$

That is, after father's initial announcement and $k'-1$ rounds of no answers, all the dirty children get to know that they are dirty. The yes answer provided by `Aximo` means that the inequality holds in an epistemic system satisfying the above assumptions.

Now consider a cheating version of the above scenario in which just before the $k'-1$'th round of no answers, all but one of the dirty children (say, all except 1), cheat by secretly telling each other that they are dirty. This scenario is encoded in an epistemic system similar to the above, but where we also have a cheating action $a(2)$ that can only apply to states in which children 2 to k' are dirty, that is the states satisfying $\bigwedge_{j=2}^{k'} f(j)$. The cheating action appears as it is to all the cheaters, that is $app_j(a(2)) = a(2)$, and as 'nothing' to all the other children, that is $app_1(a(2)) = app_l(a(2)) = \tau$ for $k'+1 \leq l \leq n'$. As a result of the cheating, in the $k'-1$'th round, all the dirty cheating children will announce that they know they are dirty where as 1 answers as usual. We denote this round of mixed answers in the public announcement $a(3)$ whose kernel and appearance are easily determined. After encoding all these assumptions in `Aximo`, we obtain a yes answer to the following inequality

$$s_{\{1,2,\cdots,k'\}} \leq d(0)\,\underbrace{d(1)\cdots d(1)}_{k'-2}\,d(2)d(3)e(1)f(n'+1)$$

which says that child one will wrongly believe that he is clean.

For a version with lying children assume a scenario in which there is only one dirty child, that is $k'=1$ and he lies in his answer (by saying no instead of yes) in the first round. We encode this scenario in an epistemic system in a similar fashion as the above scenarios where moreover we have a lying action $a(4)$, which can only apply to the states that moreover satisfy $e(1)f(1)$. This action appears as identity to the lier, that is $app_1(a(4)) = a(4)$ where as all the other children think that it is an honest no answer, that is $app_j(a(4)) = a(1)$ for $2 \leq j \leq n'$. After encoding these assumptions in `Aximo`, we obtain a yes answer to the following

$$s_{\{1\}} \leq d(0)d(4)e(j)f(j)$$

This means that the clean children $2 \leq j \leq k'$ will wrongly believe that they are dirty. The complexity analysis for these scenarios are obtained in a similar way to the coin toss, that is by applying corollary 4.2.

6 Challenges and Future Work

Theoretical challenges: Distributivity, Negation, Preservation.
(1) For simplicity of implementation we work with distributive epistemic systems. As discussed in [2,9], distributivity is not necessary to prove properties of puzzles like muddy children. We would like to extend our software to the non-distributive case. (2) Following [2] , `Aximo` does not have negation in its language and it uses adjunction rather than De Morgan dualities to reason about modalities. However, it is easy to include negation: by adding the axioms to the appropriate setting, e.g. Boolean negation to a Boolean algebra and Intuitionistic negation to a Heyting Algebra module. It is nice to note that even in the presence of either of these, our current epistemic operators will not become De Morgan duals. (3) The recursive calls to the solver for verification of the kernel inequalities is one of the complexity bottlenecks of `Aximo`. This can be overcome by using *stability under update* properties (i.e. preservation theorems). The preservation of facts under any update is the simplest such result. In recent work [4] we have shown that any state is stable under any action with a positive content. Lack of negation in our setting means that we need a more refined version of this result, that is, one that distinguishes formulas in a positive fragment and in the presence of kernel rather than content. This is a nice theoretical challenge that came out of our implementation.

Practical challenges: Dynamic kernels, Memorizing, Garbage collection
(1) In order to stop `Aximo` from looping indefinitely, we only allow epistemic modalities in the kernels of actions. A natural generalization would be to relax this and use other methods of preventing an infinite loop. One possibility would be to keep track of sub-inequalities and return (with a no answer) from the loop if they overlap with a kernel inequality. Of course in this case the no answer will not mean that the inequality was false. (2) While solving the puzzles, we noticed that there is a good number of overlaps between the sub-inequalities generated from the choices in appearances, for example for the coin toss this is a 1/2 ratio. In order to avoid repetition, we aim to detect these overlaps. (3) The main algorithm in `Aximo` relies heavily on list manipulation and contains a lot of dynamic memory allocation, making it more suitable to a language such as Digital Mars D which supports list handling and garbage collection (for memory handling) at the compiler/language specification level, thus improving overall performance. As D is largely interoperable with C, porting the algorithm to D would not reduce its accessibility to developers.

Comparison.
`DEMO` is a model checker [6] based on the underlying Kripke semantics of DEL. It inputs the initial kripke structure of the scenario and the kripke structures of the actions involved. Its main task is computing the *update product* of these structures, as introduced in [1], and then browsing it to model check a dynamic epistemic property. We defer a formal comparison of `Aximo` and `DEMO` to future work and only hint to the fact that since `Aximo` is based on a non-Boolean propositional setting and thus does not have negation, it stores less information about states than `DEMO` whose propositional language is Boolean.

Acknowledgement

We thank C. Cîrstea for her useful comments and V. Sassone for his kind advice.

References

[1] A. Baltag and L.S. Moss, 'Logics for epistemic programs', *Synthese* **139**, 2004.

[2] A. Baltag, B. Coecke and M. Sadrzadeh, 'Epistemic actions as resources', *Journal of Logic and Computation* **17(3)**, 555-585, 2007.

[3] J. van Benthem, 'One is a Lonely Number', Technical Report PP-2002-27, ILLC, Amsterdam, 2002, to appear in P. Kopke, ed., *Colloquium Logicum*, Munster, 2001, AMS Publications.

[4] C. Cirstea and M. Sadrzadeh, 'Coalgebraic Epistemic Update without Change of Model', *Lecture Notes in Computer Science* **4624**, pp. 158-172, June 2007.

[5] W. van Der Hoek and M. Wooldridge, 'Time, Knowledge, and Cooperation: Alternating-Time Temporal Epistemic Logic', COORDINATION 2002.

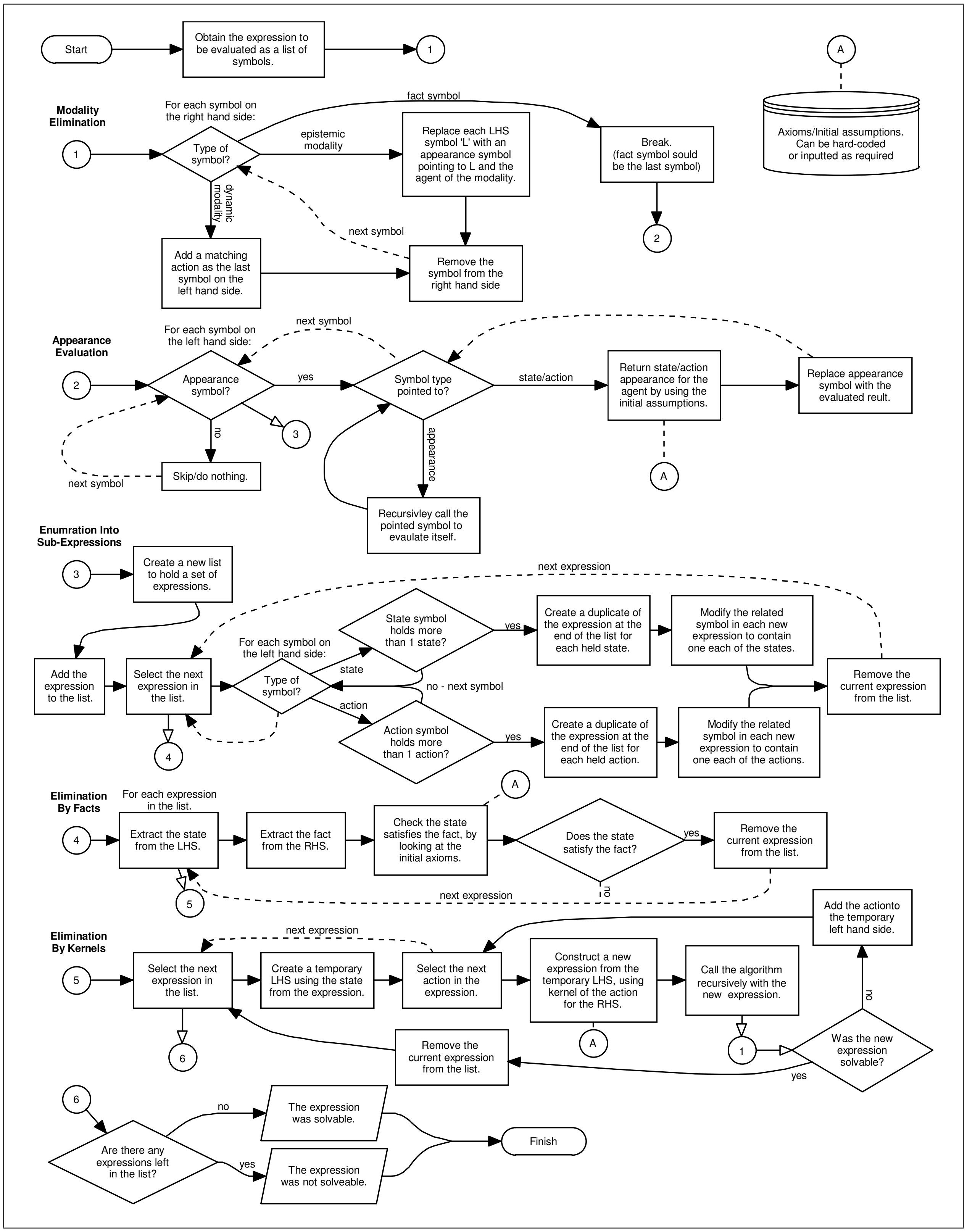
[6] Jan van Eijck, CWI, Amsterdam `http://homepages.cwi.nl/∼jve/demo/DEMO.pdf`.

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

[8] S. Richards and M. Sadrzadeh, `Aximo`, downloadable from `http://www.charcoalfeathers.net/research/projects/aximo`, August 2007.

[9] M. Sadrzadeh, 'Actions and Resources in Epistemic Logic', Ph.D. Thesis, University of Quebec at Montreal, 2005, `www.ecs.soton.ac.uk/∼ms6/all.pdf`.

7 Appendix


Start
Obtain the expression to be evaluated as a list of symbols.
1
A
Axioms/Initial assumptions. Can be hard-coded or inputted as required
Modality Elimination
For each symbol on the right hand side:
1
Type of symbol?
fact symbol
epistemic modality
dynamic modality
Replace each LHS symbol 'L' with an appearance symbol pointing to L and the agent of the modality.
Break. (fact symbol sould be the last symbol)
2
Add a matching action as the last symbol on the left hand side.
next symbol
Remove the symbol from the right hand side
Appearance Evaluation
For each symbol on the left hand side:
2
Appearance symbol?
next symbol
yes
Symbol type pointed to?
state/action
Return state/action appearance for the agent by using the initial assumptions.
Replace appearance symbol with the evaluated reult.
no
3
appearance
A
Skip/do nothing.
next symbol
Recursivley call the pointed symbol to evaulate itself.
Enumration Into Sub-Expressions
3
Create a new list to hold a set of expressions.
next expression
Add the expression to the list.
Select the next expression in the list.
For each symbol on the left hand side:
Type of symbol?
state
action
State symbol holds more than 1 state?
yes
Create a duplicate of the expression at the end of the list for each held state.
Modify the related symbol in each new expression to contain one each of the states.
no - next symbol
Remove the current expression from the list.
Action symbol holds more than 1 action?
yes
Create a duplicate of the expression at the end of the list for each held action.
Modify the related symbol in each new expression to contain one each of the actions.
4
Elimination By Facts
For each expression in the list.
4
Extract the state from the LHS.
Extract the fact from the RHS.
A
Check the state satisfies the fact, by looking at the initial axioms.
Does the state satisfy the fact?
yes
Remove the current expression from the list.
no
next expression
5
Elimination By Kernels
next expression
5
Select the next expression in the list.
Create a temporary LHS using the state from the expression.
Select the next action in the expression.
Construct a new expression from the temporary LHS, using kernel of the action for the RHS.
Call the algorithm recursively with the new expression.
Add the actionto the temporary left hand side.
no
A
1
Was the new expression solvable?
yes
Remove the current expression from the list.
6
6
Are there any expressions left in the list?
no
The expression was solvable.
yes
The expression was not solveable.
Finish