A Method for Efficient Argument-based Inquiry

Bas Testerink[1], Daphne Odekerken[1,2], and Floris Bex[2]

[1] Police Lab AI
Netherlands National Police
{bas.testerink,daphne.odekerken}@politie.nl
[2] Police Lab AI
Utrecht University
f.j.bex@uu.nl

Abstract. In this paper we describe a method for efficient argument-based inquiry. In this method, an agent creates arguments for and against a particular topic by matching argumentation rules with observations gathered by querying the environment. To avoid making superfluous queries, the agent needs to determine if the acceptability status of the topic can change given more information. We define a notion of stability, where a structured argumentation setup is *stable* if no new arguments can be added, or if adding new arguments will not change the status of the topic. Because determining stability requires hypothesizing over all future argumentation setups, which is computationally very expensive, we define a less complex approximation algorithm and show that this is a sound approximation of stability. Finally, we show how stability (or our approximation of it) can be used in determining an optimal inquiry policy, and discuss how this policy can be used to, for example, determine a strategy in an argument-based inquiry dialogue.

Keywords: Computational Argumentation· Inquiry

1 Introduction

When performing inquiry or information seeking, an agent gathers information from the environment such that it can form an opinion on a particular topic. There are different strategies that one can consider for an agent [7, 3, 11]. We propose a method for capturing agent inquiry policies in a way that is efficient – both computationally and in terms of the length of the inquiry process. The knowledge of the agent is modelled as as a structured argumentation setup similar to ASPIC$^+$ [8]. A set of possible queryable literals (observations which can be made in the future) is also defined as part of the argumentation setup. Given these queryable literals and the arguments that follow from the current observations, it can then be determined whether the topic is an acceptable conclusion [4], and which future observations (i.e. answers to queries) could conceivably change the acceptability of this conclusion.

In order to avoid making superfluous queries we define a notion of *stability*: an argumentation setup is stable if given the possible queries no new arguments can be added or adding new arguments will not change the acceptability of the topic. Concretely: does an argument for the topic exist, is this argument in the grounded extension [4], and can future answers to available queries change these facts? It is computationally complex to generate all arguments given the current observations and then calculate the grounded extension. Extra complexity is added for inquiry because one has to hypothesize on the possible results of future queries. We therefore we propose a considerably less complex *approximation algorithm* for determining stability. We also show that this algorithm provides a sound approximation of stability: if the approximation algorithm, for example, determines that the topic is acceptable, it is guaranteed that the topic is in the grounded extension and further observations cannot change this.

The proposed inquiry method is currently applied in practice for crime investigation. As an example throughout the paper, we use a simplified version of the domain of internet trade fraud (e.g. scammers on eBay or fake online stores), and specifically the situation where a complainant files an official complaint with the police. Structured argumentation is an obvious way to model the practical and legal rules concerning a crime [10]. Crime investigation should also be performed efficiently, as investigative actions (questioning the complainant, requesting the counterparty's bank details) inevitably come with a cost. Furthermore, investigation is a stochastic process, as investigative actions are not guaranteed to yield new information – the complainant might, for instance, not know the requested information. Our method takes these aspects into account. The method shows how the argumentation setup can be used to construct a Markov-decision process that represents the inquiry task. Any suitable technique can be used to approximate the optimal policy given the MDP (e.g. dynamic programming or reinforcement learning). Roughly speaking, the argumentation aspect of the method determines what kind of information is still relevant, and the policy learning aspect determines which relevant information to inquire about next.

Due to limited space we had to abbreviate examples and proofs. For the interested reader we provide extended examples and full proofs[3]. The rest of this paper is structured as follows. In Section 2 we discuss our base argumentation formalism. In Section 3 we then describe stability, that is, how to hypothesize over possible future observations, and an algorithm that approximates stability, and give soundness and complexity results for the approximation algorithm. Section 4 discusses our inquiry policy. Section 5 discusses related work and Section 6 concludes the paper.

2 Base Formalism

The base formalism for argumentation draws upon ASPIC$^+$ [8] for structured argumentation and Dung's grounded semantics [4] for abstract argumentation. From ASPIC$^+$ the concepts of a topic language, knowledge base and defeasible rules are used. The concepts of queryable literals and a topic are added to these.

[3] Extended examples and proofs: https://preview.tinyurl.com/y656r3ek

The queryable literals are those literals in the topic language of which an observation might be made. The topic is a special literal of interest for which the agent aims to get a stable opinion. Together, these components are referred to as an argumentation setup as defined below. For notation convenience we use $-l$ to negate a literal l, i.e.: $-l = p$ if $l = \neg p$ and $-l = \neg p$ if $l = p$, for some propositional atom p.

Definition 1 (Argumentation Setup, AS). *An argumentation setup AS is a tuple $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ where:*

- *$\mathcal{L}$ is a logical language consisting of propositional literals, such that if $l \in \mathcal{L}$ then also $-l \in \mathcal{L}$.*
- *$\mathcal{R}$ is a set of defeasible rules $p_1 ... p_m \Rightarrow q$ s.t. $p_1, ..., p_m, q \in \mathcal{L}$. $p_1 ... p_m$ are called the antecedents of a rule and q the consequent. The antecedents of a rule are unordered. We refer to a rule $p_1 ... p_m \Rightarrow q$ as 'a rule for q'.*
- *$\mathcal{Q} \subseteq \{l \in \mathcal{L} | l \neq \neg p\}$ is a set of non-negated queryable literals.*
- *$\mathcal{K} \subseteq \mathcal{L}$, such that $\forall l \in \mathcal{K} : (-)l \in \mathcal{Q} \wedge -l \notin \mathcal{K}$, is a knowledge base of observations which is a consistent set of literals.*
- *$\tau \in \mathcal{L}$ is a topic.*

Example 1. As an example, let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup for a simplified fraud scenario. Figure 1 depicts the topic language and rules. We abbreviate in formal examples the literals in the graph to the parenthesized literals. In this example $\mathcal{L}$ consists of literals made of the atoms f, cp, c, p, s and w, where 'f' stands for 'this is a case of fraud', 'c' for 'the complainant delivered', 'cp' for 'the counterparty delivered', 'p' for 'the complainant paid', 's' for 'the complainant sent a product' and 'w' for 'the wrong product was delivered'. The rules are given by $\mathcal{R} = \{p \Rightarrow c,\ s \Rightarrow c,\ (\neg cp, c) \Rightarrow f,\ w \Rightarrow cp,\ w \Rightarrow \neg f\}$. In the graph we represent a rule with an '&' that points to its consequent and is undirectionally connected to its antecedents. The idea behind the rules is that if the complainant delivered in the trade but the counterparty did not, then defeasibly the setup is a case of fraud. If a wrong product was delivered, then it is defeasibly not a case of fraud. Finally, if a wrong product was delivered then arguably the counterparty delivered in the trade, but this could be overruled by the fact that the complainant considers the ordered product to not have been delivered. The queryable literals are given by $\mathcal{Q} = \{p,s,w,cp\}$. This means that for instance the complainant can be queried for whether he/she paid. As the topic we take $\tau = f$. We will consider different knowledge bases throughout the examples.

As per ASPIC$^+$'s formalism, an argument is an inference tree that is constructed through the application of rules. The starting points for constructing an argument are the observations in the knowledge base. They are arguments themselves and on top of them new arguments can be made. Cyclic arguments are forbidden to avoid an infinite number of arguments. This is enforced by requiring that the conclusion of an argument cannot occur in any of its subarguments. The inference function that gives the arguments for a given argumentation setup is defined next.

Definition 2 (Inference, I). *Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. An argument is an inference $A_1...A_m \mapsto c$ such that $A_1...A_m$ is an unordered set of arguments called its premises and $c \in \mathcal{L}$ is its conclusion. We refer to $A_1...A_m \mapsto c$ as 'an argument for c'. The arguments of AS are given by $I(AS)$:*

- *$\emptyset \mapsto c \in I(AS)$ iff $c \in \mathcal{K}$.*
- *$A_1...A_m \mapsto c \in I(AS)$ iff $A_1...A_m$ are in $I(AS)$ and their conclusions are $c_1...c_m$, and there is a rule $c_1...c_m \Rightarrow c \in \mathcal{R}$, and c does not occur in any of the arguments $A_1...A_m$.*

Example 2. Consider the previously defined argumentation setup and let the knowledge base be $\mathcal{K} = \{p, \neg cp\}$ (the complainant paid but the counterparty did not deliver). For this example $I(AS) = \{A_1 = (\emptyset \mapsto p), A_2 = (\emptyset \mapsto \neg cp), A_3 = (A_1 \mapsto c), A_4 = (A_2, A_3 \mapsto f)\}$. Hence, given this knowledge there is an argument for fraud.

Arguments may attack and/or defend each other. An argument A attacks an another argument B if A's conclusion negates some conclusion of a subargument of B (a premise attack) or B's own conclusion (a rebut). In the first case, the attack is one-sided (from A to B), in the other case it is two-sided. There is one exception; an argument cannot be attacked on a premise or its conclusion if that premise/conclusion is an observation. The reasoning behind this is that an observation is a low-level directly observed piece of evidence and not defeasibly inferred. An argument A can defend another argument B if A attacks attackers for B. The notion of attack and defense are defined next.

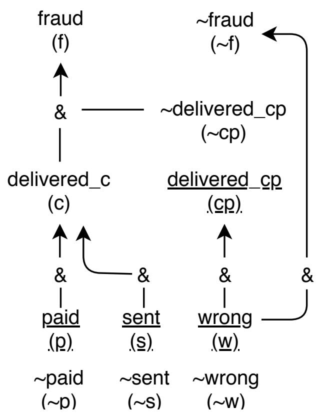

Fig. 1. A graphical representation of the example topic language and rules. Queryable literals are underlined.

Definition 3 (Attack, Defense). *Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. For two arguments $A, B \in I(AS)$ we say that A attacks B iff A's conclusion is c and $-c$ occurs in B and $-c \notin \mathcal{K}$. A set of arguments $X \subseteq I(AS)$ defends an argument $A \in I(AS)$ iff for each $B \in I(AS)$ that attacks A there is a $C \in X$ that attacks B.*

The acceptability of arguments is determined by Dung's grounded semantics for abstract argumentation [4].

Definition 4 (Grounded Extension, G). *Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. The grounded extension $G \subseteq I(AS)$ of AS is the smallest set of arguments (w.r.t. set inclusion) such that:*

- *There is no pair $A, B \in G$ such that A attacks B (conflict-free), and*
- *$G = \{A \in I(AS) | G$ defends $A\}$ (complete)*

Example 3. Consider the previously defined argumentation setup and let the knowledge base be $\mathcal{K} = \{p, \neg cp, w\}$ (the complainant paid and the counterparty delivered the wrong product). For this example $I(AS)$ are the arguments in Figure 2. The attack relation is also shown in Figure 2. The grounded extension for this example is $\{A_1, A_2, A_3, A_5\}$. Note that without the observation w, as in the previous example, the argument A_4 for fraud would be in the grounded extension. Hence, extra observations may change whether or not an argument is in the grounded extension.

3 Hypothesizing over Future Observations

An inquiry agent tries to form a stable opinion on its topic literal. An argument for a literal can be non-existent, in the grounded extension, or outside the grounded extension. In the latter case, the argument might be attacked from within the grounded extension or otherwise from outside the grounded extension. These four cases indicate different belief-statuses. If the status of the topic cannot change by executing more queries, then the argumentation setup is called stable. The possible future argumentation setups are all those setups where the queryable literals are put in a current knowledge base as either positive or negative. Stability of an argumentation setup is defined below.

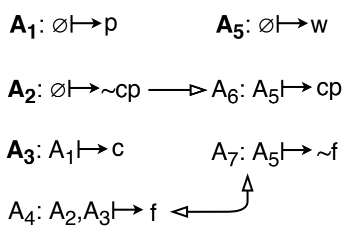

Fig. 2. A graphical representation arguments and their attack relation (the arrows). Boldface arguments are in the grounded extension.

Definition 5 (Future setups, Stability, F). *Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. The set of all future setups $F(AS)$ consists of all setups $(\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}', \tau)$ such that $\mathcal{K} \subseteq \mathcal{K}'$. AS is stable iff any of the following holds:*

- ***Unsatisfiable:*** *For each $AS' \in F(AS)$ there is no argument for τ in $I(AS')$*
- ***Defended:*** *For each $AS' \in F(AS)$ there is an argument for τ in the grounded extension of AS'*
- ***Out:*** *For each $AS' \in F(AS)$ there is an argument for τ in $I(AS')$ but all arguments for τ are attacked by an argument in the grounded extension of AS'*
- ***Blocked:*** *For each $AS' \in F(AS)$ there is an argument for τ in $I(AS')$ but not in the grounded extension of AS' and at least one argument for τ is not attacked by an argument from the grounded extension of AS'*

Example 4. Let us consider the previous example again where $\mathcal{K} = \{p, \neg cp, w\}$ and the arguments and attack relation are shown in Figure 2. The **blocked** case applies because there exist arguments for f and $\neg f$ (A_4 and A_7) and they are both outside the grounded extension. Furthermore, the only queryable literal that is left is s, which cannot influence this situation if s or $\neg s$ is observed. Consider also the setup where $\mathcal{K} = \{cp\}$. For this setup no argument can possibly exist for f because all potential arguments require $\neg cp$. Therefore, in that setup the **unsatisfiable** case applies. If s, $\neg w$ and $\neg cp$ are observed, then there exists an argument for f in the grounded extension and no further observations (i.e. p or $\neg p$) can change this. Therefore in that case the **defended** case applies. Finally, in this example the **out** case can only apply for the literal $\neg cp$. This happens when w and cp are observed. In that case w is a basis for an argument for cp whilst $\neg cp$'s observation unilaterally attacks that argument.

A brute-force method for determining stability would be to calculate all possible future setups and then for each setup calculate the grounded extension to see whether the topic is stable. This results in possibly $3^{|\mathcal{Q}|}$ different setups to calculate the grounded extension for. The number of arguments in the grounded extension given $n = |\mathcal{L}|$ is maximally $n \cdot g(n), g(n) = (1 + g(n))^{n-1}$. The runtime complexity of this approach would be unpractical. Therefore a less complex approach to this task is preferable. The following labelling is an approximation of the task. The idea behind it is that rules and literals are labelled, where labels relate to the cases of stability. The labelling is defined as follows. After the definition and an example we discuss the soundness and complexity of the labelling.

Definition 6 (Labelling, L). *Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. A labelling L is a partial function that assigns a label from $\{U, D, O, B\}$ to literals and rules. Literals that are in $\mathcal{Q}$ but not observed ($l, -l \notin \mathcal{K}$) are not labelled. For the other literals and the rules the labelling is defined as follows:*
***Case** U **literal:** $l \in \mathcal{L}$ is labelled U iff either: A) No rule exists for l and if $(-)l \in \mathcal{Q}$ then $-l \in \mathcal{K}$. B) There are rules for l and they are labelled U and $l \notin \mathcal{K}$.*
***Case** U **rule:** $r \in \mathcal{R}$ is labelled U iff any of its antecedents is labelled U.*
***Case** D **literal:** $l \in \mathcal{L}$ is labelled D iff either: A) $l \in \mathcal{K}$. B) There is a rule for l labelled D, $-l \notin \mathcal{K}$ and there is no rule for $-l$. C) There is a rule for l labelled D, $-l \notin \mathcal{K}$ and there are rules for $-l$ but they are all labelled U or O.*
***Case** D **rule:** $r \in \mathcal{R}$ is labelled D iff all its antecedents are labelled D.*
***Case** O **literal:** $l \in \mathcal{L}$ is labelled O iff either: A) There exists a rule for l labelled D, O or B and $-l$ is labelled D. B) There are rules for l of which at least one is labelled O and the rest is either labelled O or U.*
***Case** O **rule:** $r \in \mathcal{R}$ is labelled O iff at least one antecedent is labelled O and the rest is labeled D, B or O.*
***Case** B **literal:** $l \in \mathcal{L}$ is labelled B iff $l, -l \notin \mathcal{Q}$ and either: A) A rule for l and a rule for $-l$ is labelled D or B. B) There are rules for l of which one is labelled B and the rest is either labelled U, O or B.*
***Case** B **rule:** $r \in \mathcal{R}$ is labelled B iff at least one antecedent is labelled B and the rest is labeled B or D.*

Example 5. Figure 3. Shows the labelling for the example argumentation setup where $\mathcal{K} = \{p, \neg cp, w\}$. As expected given the previous example, the f literal is labelled B. Consider also the setup where $\mathcal{K} = \{cp\}$. The labelling for that setup labels $\neg cp$ U due to case 'U **literal** A'. Consequently, the rule $c, \neg cp \Rightarrow f$ is labelled U due to case 'U rule'. Finally f is labelled U because of case 'U **literal** B'. Hence, as discussed in the previous example, for the example argumentation setup where $\mathcal{K} = \{cp\}$ the topic f is unsatisfiable. In the 'extended examples and proofs' document we discuss more examples.

Note that alongside possible future observations, other literals can also be unlabelled. In particular, as long as the topic remains unlabelled it means that more information is required. The labelling of an argumentation setup is a sound approximation of stability. This means that if the topic is labelled, then the argumentation setup is stable. The following two propositions together show this. As the method is an approximation there are cases where a literal might be stable but unlabelled. The proof for proposition 2 in the 'extended examples and proofs' document[4] contains such an example.

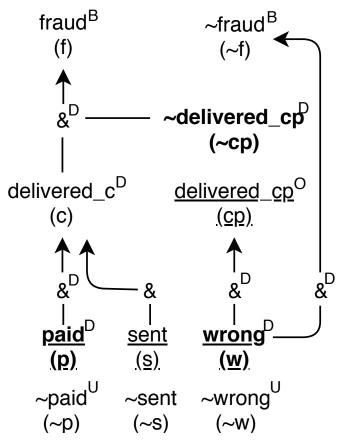

Fig. 3. Labelling of Figure 2 given $\mathcal{K} = \{p, w, \neg cp\}$. Boldface literals are in $\mathcal{K}$ and underlined literals are queryable.

Proposition 1. *Let L be the labelling of an argumentation setup $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$. If for a rule $r = (p_1 ... p_m \Rightarrow c) \in \mathcal{R}$: $L(r) \in \{D, B, O\}$ then there exists an argument for c in $I(AS)$.*

Proof sketch: By following the labels D, B and O from literals to rules we must end up in observed literals. Hence, if a rule is labelled D, B or O then we can follow the labelling until we end up with a set of observed literals. The observed literals, and the rules that were passed by following the labelling, can be used to construct the argument for c.

Proposition 2. *Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup and L be its labelling. If $L(\tau) = U$, D, O or B then AS is stable because of the* ***unsatisfiable****,* ***defended****,* ***out****, or* ***blocked*** *case of Definition 5, respectively.*

Proof sketch: The different labels are 'introduced' under the stated property and their propagation through the rules to other literals preserves this property. U is introduced if a literal l has no rules for it and is unobservable (possibly due to $-l$ being observed). Hence, no argument could exist for l. D is introduced for observed literals and hence for such literals an argument is guaranteed to exist

[4] Extended examples and proofs: https://preview.tinyurl.com/y656r3ek

that is in the grounded extension. O is introduced if a rule for a literal l can be applied to make an argument but $-l$ is observed. Hence every argument for l is unilaterally attacked by an argument in the grounded extension. B is introduced if for l and $-l$ there exists at least one argument that is not attacked by an argument from the grounded extension.

U is propagated through rules if for a rule at least one premise is labelled U. This indicates that the rule cannot be applied to construct an argument for its conclusion. If a literal has only U-labelled rules then therefore no rule can be applied to construct an argument for that literal. D is propagated if for a rule all premises are labelled D. This indicates that if the conclusion cannot have a rebutter, then this rule can be applied to construct an argument for in the grounded extension. So an unobservable literal becomes D if this holds. O is propagated if for a rule all premises are not unsatisfiable (so arguments can be made) but at least one is O, indicating that there will always be a unilateral premise attack from the grounded extension to arguments based on this rule. Which is why literals with only O-applicable rules are labelled O. B is propagated if for a rule if at least one of its premises is labelled B and the others are D or B-labelled. This indicates that every argument based on this rule will have a unilateral attacker on one of its premises. However, this attacker has a bi-lateral attacker that rebuts it. Hence arguments based on this rule will be attacked but not by arguments in the grounded extension. Therefore if a literal has only rules that are labelled U, O or B then the labelled becomes B.

Finally, we aim to improve upon the complexity of the brute-force method of determining stability. The following proposition discusses the big-Oh complexity of labelling an argumentation setup.

Proposition 3. *Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup. The labelling L of AS can be constructed in $O((|\mathcal{L}| + |\mathcal{R}|)^2)$.*

Proof sketch: A simple algorithm for the labelling works as follows. We can start with the set $\mathcal{L} \cup \mathcal{R}$ minus the literals from $\mathcal{L}$ that might be observed in the future. Then, we iterate through the cases of Definition 6 until no case applies anymore. Any time a case applies for a literal or rule, we remove it from the set. Worst-case, the set shrinks one-by-one until the empty set is reached (every literal/rule has a label) in which case a quadratic number of passes through the cases has been executed $(0.5 * (|\mathcal{L}| \cdot |\mathcal{R}|)^2)$.

4 Optimizing Inquiry Policies

An inquiry policy returns a query to execute given an argumentation setup and available queries. In the following, we show how to model the inquiry setting as a Markov Decision Process (MDP) for which the optimal policy can be obtained by standard methods. An MDP consists of actions, states, a transition function and a reward function. For the actions we take the queries that are available. A query can be executed once during a dialogue.[5]. A state in the MDP is a pair of an argumentation setup and a set of available queries. After executing a query it is removed from the available queries and the setup may change because new observations might be added to the knowledge base.

The transition function tells us what the probability of a transition from one state to the next is when executing some query. For example, observing that a wrong product was received increases the probability of observing in the future that the complainant paid. We cancel out illegal transitions by setting their probability to zero. As for the reward function; we generate positive reward when a stable argumentation setup is reached from an unstable setup. Any transitions among unstable setups are negative because a query was executed. The transitions among stable setups are considered neutral. The labelling of Section 3 is applied to approximate stability. Finally, the optimal policy immediately follows from the MDP. The best action to execute give a state is the query which maximizes the expected reward. By maximizing reward, the policy will minimize the expected number of executed queries before reaching a stable setup.

Definition 7 (Argumentation MDP, policy, M, π). *Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}, \tau)$ be an argumentation setup and $F(AS)$ be all its possible future setups. An argumentation MDP M for AS is a tuple (Q, S, δ, r), where:*

- *Q is a set of queries*
- *$S = F(AS) \times 2^Q$ is the state space*
- *$\delta : S \times Q \times S \to [0, 1]$ is the transition function which returns the probability of the next state being $s_2 \in S$ given some state $s_1 \in S$ and query $q \in Q$. Furthermore, $\delta(s_1, q, s_2) = 0$ if the knowledge base of AS_1 is not a subset of that of AS_2, or if q is not available ($q \notin Q_1$), or if $Q_2 \neq Q_1 \setminus \{q\}$.*
- *$r : S \times S \to \mathbb{I}$ is the reward for transitioning from $s_1 = (AS_1, Q_1) \in S$ to $s_2 = (AS_2, Q_2) \in S$ and is given by: $r(s_1, s_2) = |Q_1|$ if τ is labelled in the labelling of AS_2 but not AS_1, $r(s_1, s_2) = 0$ if τ was already labelled in the labelling of AS_1, else $r(s_1, s_2) = -1$.*

The policy $\pi : S \to Q$ is given by:
$\forall s_1 = (AS_1, Q_1) \in S : \pi(s_1) = argmax_{q \in Q_1} \Sigma_{s_2 \in S} \delta(s_1, q, s_2)(r(s_1, s_2) + V(s_2))$
where: $V(s_2) = \Sigma_{s_3 \in S} \delta(s_2, \pi(s_2), s_3)(r(s_2, s_3) + V(s_3))$.

Example 6. Consider a policy for the MDP that belongs to the argumentation setup of the previous examples. A query can be any action that potentially leads to some observations. For this example we take Boolean queries that only have single queryable literal for which they may lead to an observation. Let the queries be $Q = \{p?, s?, w?, cp?\}$. For a query $x?$ assume that its execution results in x or $-x$ being added to the knowledge base. The policy may then look like the one that is shown in Figure 4. Note that for simplicity's sake, we assume some (implicit) probabilities for the transitions. From the policy it can be read

[5] Note that, if desired, the same query can be 'copied' multiple times in the formal model to allow for repeated execution.

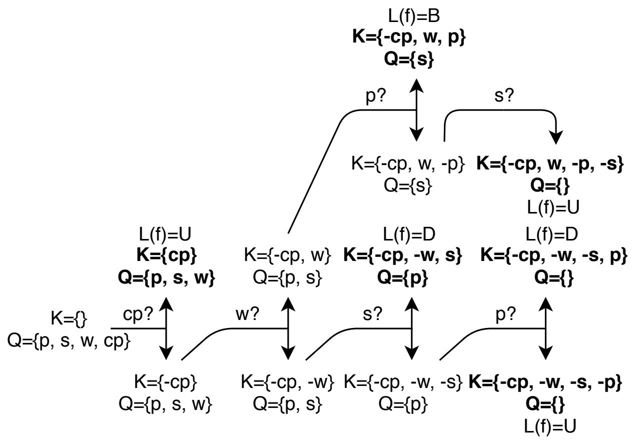

Fig. 4. Part of the example policy. These are the reachable states/actions from the empty observation set (i.e., zero probability transitions are omitted). States with stable setups are boldface and the label of the topic f is shown as well for such states.

that first it queried whether the counterpary delivered (partially) upon his/her promise. If not, then it is queried whether a wrong product was delivered. If so, then it is queried whether the complainant paid, or otherwise whether the complainant sent a product. In both situations, if the answer is negative, then the last query is executed. Note that in the example policy, after the user answered negative to 'cp?' and positive to 'w?', that the label for fraud cannot be D in the future. It depends on an application whether this might be a reason to halt the execution. Formally, the system still executes queries because the reason why the label is not 'D' is still unstable.

Roughly speaking, for our real-world applications the argumentation system provides information on what queries are relevant and the policy then chooses among the relevant queries. We apply various techniques to make the policy such as dynamic programming and q-learning. The policy itself can also be used in different ways. In one of our applications it guides a dialogue with the user and in another it prioritizes database queries. Similarly, the argumentation setup has different applications. It can be used to determine what information is relevant but also provide argumentation-based explanations for the agent's decisions.

5 Related Work

A typical argument-based inquiry is a so-called *inquiry dialogue* [7, 3], in which arguing agents collaborate to answer some question the answer to which is not known by the individual agents. Our setting is different in that the information source of our agent is not necessarily another arguing agent: a query can be implemented as an utterance in an argumentation dialogue, but it can also for instance be a query on a database. Thus, our optimal query policy can be used to determine an efficient strategy for inquiry dialogues, but it can also be used outside such dialogues. Furthermore, as already discussed in Section 3, in our setting the agent can only query observations, whereas in an argument inquiry dialogue the other agent (i.e. the agents that "answers" the query) can also provide an argument, making these dialogues slightly more flexible. However, in the existing work on inquiry dialogue strategies [3, 11] only exhaustive strategies are defined, which simply perform all queries, that is, the agent keeps asking questions even if it already has an argument for its topic, or if adding new arguments cannot change the status of the topic. Such exhaustive strategies are both inefficient (in terms of the length of the inquiry process) and computationally very expensive. While there is work on developing more optimal policies or strategies for argumentation dialogues (see [9] for a fairly recent overview), some of which shows some similarities with our work in their use of (in their case partially observable) MDPs for defining policies [6], this existing work all focuses on strategies for so-called persuasion dialogue, in which the agent tries to convince an opponent of some conclusion. Furthermore, this existing work is all based on abstract argumentation frameworks [4], in which only arguments as single entities (propositions, nodes) and their attacks are considered, and there is no argument structure that includes inferences based on a knowledge base. Work on inquiry that does not explicitly use arguments is [5], in which the authors propose a four-valued logic with the truth values true, false, inconsistent and unknown, which, broadly speaking, are similar to the current paper's Defended, Out, Blocked and Unsatisfiable stability cases. Like [3], the strategies for inquiry discussed in this paper are all exhaustive and hence not efficient.

With respect to the idea of stability, much of the work on argumentation is more concerned with the static situation: given the knowledge we have now, what are the acceptable arguments. A notable exception is by Ballnat and Gordon [1]. This work looks at determining which possible future additions to a knowledge base might change the acceptability of a conclusion. They do not use their basic ideas for determining optimal policies, however, nor do they discuss the complexity and approximation results we provide in this paper. The idea of stability also has clear links to what is called the enforcing problem [2]: given a set of arguments, can we modify this set by adding or removing arguments and conflicts so that some argument from the original set becomes acceptable? The work that exists on this topic, however, again only deals with abstract argumentation frameworks.

6 Conclusion and Future Work

In this paper we have described a policy for efficient argument-based inquiry – that is, a policy that minimizes the expected number of queries required to reach a stable setup in which the acceptability of some conclusion cannot change any more given a stochastic environment. Our approach is efficient in a number of ways, computationally as well as with respect to the inquiry process itself.

– By rewarding an agent that reasons towards a stable argumentation as quickly as possible we ensure that only the minimum required number of queries is executed to draw a stable conclusion.
– By approximating the notion of stability we make determining or learning the policy feasible in terms of computational complexity.
– Finally, by defining the policy by means of an MDP we allow the agent to avoid executing queries that are likely to be unsuccessful.

As for future work we want to further explore the notion of stability in structured argumentation, taking into account the possibility of rules, preferences and attacks being added through queries. We also intend to see how we can further embed the policy in argument dialogues between agents. Finally, our implementations of this method are leading to best-practices (e.g. to deal with noisy observations) which we aim to publish publicly.

References

1. Ballnat, S., Gordon, T.F.: Goal selection in argumentation processes. In: Computational models of argument: Proceedings of COMMA 2010. Frontiers in Artificial Intelligence and Applications, vol. 216, pp. 51–62. IOS Press (2010)
2. Baumann, R.: What does it take to enforce an argument? minimal change in abstract argumentation. In: Proceedings of the 20th European Conference on Artificial Intelligence. pp. 127–132. No. 242 in Frontiers in Artificial Intelligence and Applications, IOS Press (2012)
3. Black, E., Hunter, A.: An inquiry dialogue system. Autonomous Agents and Multi-Agent Systems **19**(2), 173–209 (2009)
4. Dung, P.M.: On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. Artificial intelligence **77**(2), 321–357 (1995)
5. Dunin-Keplicz, B., Strachocka, A.: Tractable inquiry in information-rich environments. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015). pp. 53–60. AAAI publishing (2015)
6. Hadoux, E., Beynier, A., Maudet, N., Weng, P., Hunter, A.: Optimization of probabilistic argumentation with markov decision models. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015). pp. 2004–2010. AAAI publishing (2015)
7. Parsons, S., McBurney, P., Wooldridge, M.: The mechanics of some formal inter-agent dialogues. In: Advances in Agent Communication, pp. 329–348. Springer (2004)
8. Prakken, H.: An abstract framework for argumentation with structured arguments. Argument and Computation **1**(2), 93–124 (2010)
9. Thimm, M.: Strategic argumentation in multi-agent systems. KI-Künstliche Intelligenz **28**(3), 159–168 (2014)
10. Verheij, B.: Dialectical argumentation with argumentation schemes: An approach to legal logic. Artificial intelligence and Law **11**(2-3), 167–195 (2003)
11. Yan, C., Lindgren, H., Nieves, J.C.: A dialogue-based approach for dealing with uncertain and conflicting information in medical diagnosis. Autonomous Agents and Multi-Agent Systems **32**(6), 861–885 (2018)