July 2020

Estimating Stability for Efficient Argument-based Inquiry

Daphne ODEKERKEN [a,b], AnneMarie BORG [a] and Floris BEX [a,c]
[a] *Department of Information and Computing Sciences, Utrecht University* [1]
[b] *National Police Lab AI, Netherlands Police*
[c] *Tilburg Institute for Law, Technology and Society, Tilburg University*

Abstract. We study the dynamic argumentation task of detecting stability: given a specific structured argumentation setting, can adding information change the acceptability status of some propositional formula? Detecting stability is not tractable for every input, but efficient computation is essential in practical applications. We present a sound approximation algorithm that recognises stability for many inputs in polynomial time and we discuss several of its properties. In particular, we show under which constraints on the input our algorithm is complete. The proposed algorithm is currently applied for fraud inquiry at the Dutch National Police - we provide an English demo version that also visualises the output of the algorithm.

Keywords. dynamic argumentation, structured argumentation, inquiry

1. Introduction

One task of the police is the intake of citizens' reports on crimes: the citizen tells the police what happened; subsequently, additional questions can be asked to determine if the citizen has been the victim or witness of a crime. Certain high-volume crimes can be reported online. This can be as simple as filling out a web form, but can also be a more involved online dialogue with a (possibly artificial) agent. One specific high volume crime that can be reported online at the Dutch National Police is internet trade fraud. This concerns fake web shops and malicious second-hand traders on platforms such as eBay. In [3], an initial sketch was given for an artificial agent handling the intake of internet trade fraud by combining natural language processing with symbolic techniques for reasoning about crime reports. During the subsequent development of the intake agent, we regarded intake as *argument-based inquiry* [4]. In this inquiry, defeasible rules representing the laws and practices surrounding trade fraud are combined with the citizen's knowledge of the specific situation they observed, to build arguments for and against the main claim made by the citizen: that they have been the victim of trade fraud.

The first contribution of this paper is to present the implemented version of the intake agent. It has been released on the web site of the Dutch Police[2] where it handles the intake of hundreds of fraud reports every day. Because the police web site only shows

[1] This research has been partly funded by the Dutch Ministry of Justice and the Dutch National Police.
[2] https://aangifte.politie.nl/iaai-preintake

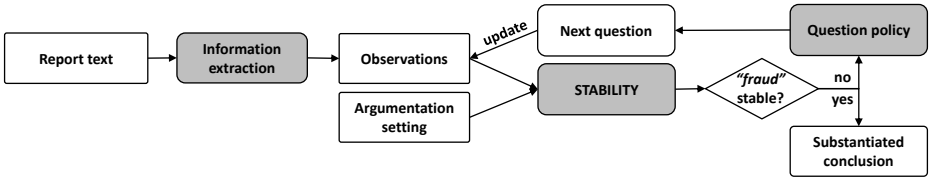


Figure 1. *Overview of the hybrid inquiry agent for the intake of fraud complaints.*

the Dutch user interface, we provide a demo[3] of an English version that gives more insight in the underlying reasoning. The agent's architecture is illustrated in Figure 1. The *information extraction* component uses natural language processing techniques to automatically extract the initial observations from the free text user input [12]. These observations are then combined with rules concerning trade fraud in the argumentation setting to build arguments for and against the claim "fraud". The *stability* component decides if any additional observations that the citizen could possibly add in the future can change the acceptability status of the "fraud" claim. If not, the dialogue terminates; otherwise a *question policy* component finds the best question to ask given current observations. The stability component is thus an important part of the agent's architecture: it provides a termination criterion that prevents the agent from asking unnecessary questions. If, for example, it is already clear from the initial observations that we are not dealing with fraud because the citizen simply received a product they did not like, the agent will not continue to exhaustively inquire [4] about further details of the situation.

The rest of this paper focuses on a more theoretical study of the stability component of the intake agent. Stability in structured argumentation is a form of dynamic argumentation that was introduced in [14]. Informally, a claim is stable if more information cannot change the acceptability of the claim, where this acceptability depends on the acceptability of arguments for this claim in terms of Dung's grounded semantics [7]. Detecting stability is complex: a brute-force approach would involve generating and evaluating all possible future argumentation setups given new observations, which would require far too much time in an applied setting. In this paper, we provide some new insights on the complexity of the stability problem by showing that it is CoNP-hard.

A sound approximation algorithm for stability was provided in [14]. However, the conditions under which the algorithm is complete were not studied in-depth. When investigating these conditions, we identified the nontrivial issues of *irrelevant labels* and *support cycles*, in the presence of which the algorithm from [14] does not detect a stable situation. In this paper, we solve these issues by proposing a new approximation algorithm consisting of an alternative labelling and a preprocessing step. We prove[4] that the refined algorithm has polynomial time complexity and that it is sound. Furthermore, we specify constraints on the input under which the new algorithm is complete.

Section 2 below specifies the structured argumentation setup for which we formally define the stability problem in Section 3. In Section 4 we then identify issues with the algorithm of [14], propose our refined algorithm and study its properties. Section 5 discusses related work in dynamic argumentation and Section 6 concludes the paper.

[3] https://nationaal-politielab.sites.uu.nl/estimating-stability-for-efficient-argument-based-inquiry/

[4] Due to space restrictions, proofs are omitted in this paper. The proofs are available at https://nationaal-politielab.sites.uu.nl/estimating-stability-for-efficient-argument-based-inquiry/

2. Preliminaries

Our argumentation setup is a variation on ASPIC$^+$ [11], albeit simplified in that we only consider axiom premises, defeasible rules and no preferences. From a theoretical perspective, this can be considered to be a limitation; however, from a practical point of view this simplification makes it more feasible for police employees without background in formal argumentation to adapt or create rule sets. We add the notion of queryable literals $\mathcal{Q}$. These literals can be obtained (i.e. added to the knowledge base) by querying the citizen, thus restricting the possibilities of updating the knowledge base.

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

- $\mathcal{L}$ is a finite propositional language, closed under classical negation ($\neg$). The literals will be denoted by lower-case letters. We write $a = -b$ iff $a = \neg b$ or $b = \neg a$.
- $\mathcal{R}$ is a finite set of defeasible rules $a_1, \ldots, a_m \Rightarrow c$ such that $\{a_1, \ldots, a_m, c\} \subseteq \mathcal{L}$. Where $r \in \mathcal{R}$, $\texttt{ants}(r) = \{a_1, \ldots, a_m\}$ are the antecedents of r and $\texttt{cons}(r) = c$ is its consequent. We refer to a rule with consequent c as "a rule for c".
- $\mathcal{Q} \subseteq \mathcal{L}$ is a set of queryable literals, s.t. $l \in \mathcal{Q}$ iff $-l \in \mathcal{Q}$.
- $\mathcal{K} \subseteq \mathcal{Q}$ is the knowledge base, which must be consistent: if $l \in \mathcal{K}$ then $-l \notin \mathcal{K}$.

Based on an argumentation setup, arguments can be constructed from $\mathcal{R}$ and $\mathcal{K}$.

Definition 2 (Argument)**.** Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ be an argumentation setup. We denote by $Arg(AS)$ the set of **arguments** inferred from AS. An argument $A \in Arg(AS)$ is:

- an **observation-based argument** c iff $c \in \mathcal{K}$.
  The conclusion $\texttt{conc}(A)$ of A is c. The set of subarguments $\texttt{sub}(A)$ of A is $\{c\}$.
- a **rule-based argument** $A_1, \ldots, A_m \Rightarrow c$ iff for each $i \in [1 \,..\, m]$: A_i is in $Arg(AS)$ with conclusion c_i and there is a rule $r : c_1, \ldots, c_m \Rightarrow c$ in $\mathcal{R}$.
  The conclusion $\texttt{conc}(A)$ of A is c. The set of subarguments $\texttt{sub}(A)$ of A is $\texttt{sub}(A_1) \cup \ldots \cup \texttt{sub}(A_m) \cup \{A\}$. The top rule $\texttt{top-rule}(A)$ of A is r.

We refer to an argument with conclusion c as "an argument for c". We refer to a rule-based argument with top rule r as "an argument based on r".

Definition 3 (Attack)**.** Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ be an argumentation setup. For two arguments $A, B \in Arg(AS)$ we say that A **attacks** B on B' iff A's conclusion is c, there is a subargument $B' \in \texttt{sub}(B)$ such that $\texttt{conc}(B') = -c$ and $-c \notin \mathcal{K}$.

Our definition of attack corresponds to rebuttal in ASPIC$^+$ [11]. From Definition 3 it follows directly that observation-based arguments cannot be attacked.

Example 1 (Online trade fraud)**.** Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$, visualised in Figure 2, be an argumentation setup in the domain of online trade fraud. $\mathcal{L}$ consists of the literals $\{b, sm, sp, rp, rm, u, s, t, sd, rd, d, f\}$ and their negations. Squares represent literals from $\mathcal{L}$, rounded squares are queryable literals (from $\mathcal{Q}$) and literals in $\mathcal{K}$ are shaded. Rules are represented by double-lined arrows and attacks as single-lined arrows. $Arg(AS)$ includes an argument for f based on the rule $sd, \neg rd, d \Rightarrow f$ and an argument for $\neg f$ based on $b, t \Rightarrow \neg f$. These arguments attack each other.

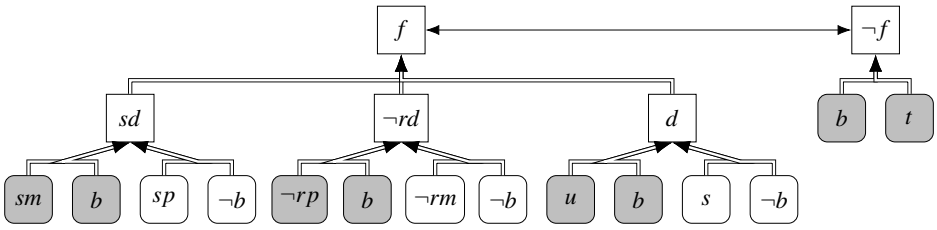


Figure 2. Example of an argumentation setup AS from the law enforcement domain. b: citizen tried to buy a product (as opposed to selling a product); sm: citizen sent money; sp: citizen sent product; rp: citizen received product; rm: citizen received money; u: suspicious url; s: screenshot of payment; t: trusted web shop; sd: citizen delivered; rd: citizen received delivery; d: deception; f: fraud. Note that the literals b and $\neg b$ are visualised multiple times and attacks between them are omitted for clarity.

Like in ASPIC$^+$, the evaluation of arguments is done using the semantics of [7]. We choose grounded semantics since it is the most skeptical semantics, which fits the application in police investigation. We subsequently use the grounded extension to define the acceptability of literals in an argumentation setup.

Definition 4 (Grounded Extension)**.** Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ be an argumentation setup and let $S \subseteq Arg(AS)$. S is said to be **conflict-free** iff there are no $A, B \in S$ such that A attacks B. S **defends** $A \in Arg(AS)$ iff for each $B \in Arg(AS)$ that attacks A there is a $C \in S$ that attacks B. S is **admissible** iff it is conflict-free and defends all its arguments. S is a **complete extension** iff it is admissible and contains all the arguments it defends. The **grounded extension** $G(AS)$ is the least (w.r.t. $\subseteq$) complete extension.

Definition 5 (Acceptability)**.** Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ be an argumentation setup. The acceptability of literal $l \in \mathcal{L}$ given AS is:

- **unsatisfiable** iff there is no argument for l in $Arg(AS)$;
- **defended** iff there exists an argument for l in $Arg(AS)$ that is also in the grounded extension $G(AS)$;
- **out** iff there exists an argument for l in $Arg(AS)$, but each argument for l in $Arg(AS)$ is attacked by an argument in the grounded extension $G(AS)$;
- **blocked** iff there exists an argument for l in $Arg(AS)$, but no argument for l is in the grounded extension $G(AS)$ and at least one argument for l is not attacked by an argument in $G(AS)$.

Note that these acceptability statuses are complementary: e.g. if l is not unsatisfiable, defended or out, then it is blocked. This follows directly from the definition.

Example 2 (Example 1 continued)**.** In argumentation setup AS from Figure 2, $G(AS)$ contains (unattacked) arguments for sm, b, $\neg rp$, u, t, sd, $\neg rd$ and d, so these literals are defended in AS. There are arguments for f and $\neg f$ in $Arg(AS)$ that attack each other, but these are not attacked or defended by any argument in $G(AS)$, so f and $\neg f$ are blocked in AS. Each other literal $l \in \mathcal{L}$ is unsatisfiable in AS: there is no argument for l in $Arg(AS)$.

3. Stability

Using Definition 5, we can determine the acceptability status of a literal $l \in \mathcal{L}$ in a given argumentation setup $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$. However, by adding more information, l's ac-

ceptability status may change. Informally, l is *stable* in AS if its acceptability status cannot change by adding any combination of queryables to the knowledge base - provided that the resulting knowledge base is consistent. Note that we restrict the changes on the argumentation setup to adding knowledge, since we expect the citizen to attribute only facts on his/her situation. Next, we define future setups, which specify how information can be added to AS.

Definition 6 (Future setups)**.** The set of **future setups** $F(AS)$ of an argumentation setup $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ consists of all argumentation setups $AS' = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}')$ with $\mathcal{K} \subseteq \mathcal{K}'$.

Note that the argumentation setup AS always belongs to the set of future setups $F(AS)$. Further recall from Definition 1 that $\mathcal{K}'$ must be consistent since AS' is an argumentation setup. Using the notion of future setups, we now define stability.

Definition 7 (Stability)**.** Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ be an argumentation setup. A literal $l \in \mathcal{L}$ is **stable** in AS iff there is an acceptability status $acc \in \{$unsatisfiable, defended, out, blocked$\}$ such that for each $AS' \in F(AS)$, l is acc in AS'.

Example 3 (Example 2 continued)**.** In our running example, the literal f is stable. By querying the client agent, we could obtain more information; $F(AS)$ for example contains an argumentation setup with knowledge base $\mathcal{K}' = \mathcal{K} \cup \{\neg sp\} = \{sm, b, \neg rp, u, t, \neg sp\}$. However, adding information does not influence f's acceptability status: for each AS' in $F(AS)$, f is blocked in AS'. Therefore, f is stable in AS.

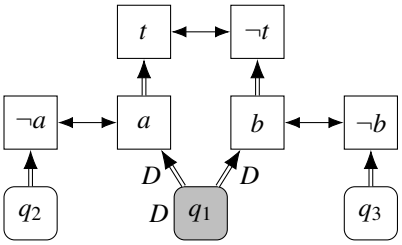
Proposition 1. *Determining stability is CoNP-hard.*

This can be shown by a polynomial-time reduction from the CoNP-complete problem UNSAT. The full proof is available on the website with additional material.

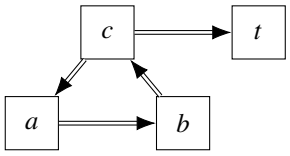
CoNP-hard problems are generally considered intractable (unless P = NP). Given the above results and assuming that P $\neq$ NP, there is no exact polynomial-time algorithm that determines for an arbitrary argumentation setup AS if a literal is stable in AS. This means that an exact algorithm would need exponential time. Since practical applications require fast computation for arbitrary argumentation setups, we consider a sound polynomial-time approximation algorithm in the next section.

4. Approximating stability

A first approximation algorithm for determining stability in formal argumentation was proposed in [14]. This algorithm assigns a label to literals and rules that it considers to be stable. Each label relates to one of the four cases of stability: U (unsatisfiable); D (defended); O (out); or B (blocked). However, the algorithm is not complete: there exist argumentation setups that are stable but are not labelled as such by the approximation algorithm. In [14] we gave an example, but no precise specification of argumentation setups for which the algorithm does not recognise stability. In the next subsection, we give two additional examples which reveal different issues of the method described in [14]. In Sections 4.2 and 4.3, we present a refined algorithm to solve these issues. Subsequently, we will show soundness and conditional completeness and study the computational complexity of this refinement in Section 4.4.



(a) D/B is irrelevant in [14] *Case B lit. A*.



(b) Support Cycle. $\mathcal{Q} = \emptyset$, so $F(AS) = \{AS\}$. $\mathcal{K} = \emptyset$, so there is no argument for t in $Arg(AS)$. However, L does not label t.

Figure 3. Examples of incompleteness of the basic algorithm from [14].

4.1. Examples of incompleteness basic algorithm

Figures 3a and 3b illustrate two different issues of the algorithm from [14].

Example 4 (Irrelevant label problem)**.** Figure 3a represents an argumentation setup AS in which q_1, q_2 and q_3 are queryable. q_1 is in the knowledge base. There is an argument for t based on $a \Rightarrow t$ and an argument for $\neg t$ based on $b \Rightarrow \neg t$ in $Arg(AS)$. So for each $AS' \in F(AS)$, t is blocked in AS'. However, t is not recognised as being stable by the algorithm in [14]. The literal q_1 and rules $q_1 \Rightarrow a$ and $q_1 \Rightarrow b$ are correctly labelled D, but the other literals and rules are not labelled by the algorithm. a and b are not labelled because they may become either defended (if $\neg q_2$ resp. $\neg q_3 \in \mathcal{K}'$) or blocked (if q_2 resp. $q_3 \in \mathcal{K}'$). As a result, the rules $a \Rightarrow t$ and $b \Rightarrow \neg t$ are not labelled because they may become either defended or blocked. In all future setups in $F(AS)$, the argument for a is either defended (if $q_2 \notin \mathcal{K}'$) or blocked (if $q_2 \in \mathcal{K}'$). Similarly, in every future setup, the argument for b is either defended (if $q_3 \notin \mathcal{K}'$) or blocked (if $q_3 \in \mathcal{K}'$). The algorithm in [14] has a labelling rule *Case B literal A* stating that "$l \in \mathcal{L}$ is labelled B iff $l \in \mathcal{Q}$ and a rule for l and a rule for $-l$ are labelled D or B". However, this rule does not apply: although $a \Rightarrow t$ and $b \Rightarrow \neg t$ will be labelled D or B in a future setup in which we have information about q_2 and q_3, we do not know the exact label - which is here irrelevant.

We will refer to the issue illustrated in Figure 3a as the *irrelevant label problem*. It is caused by the fact that L only assigns a label if there is exactly one possible acceptance status for all future setups, but does not take into account that some acceptability statuses are *impossible* in a future setup. The next example reveals another issue of the basic algorithm, which we will refer to as the *support cycle problem*.

Example 5 (Support cycle problem)**.** Figure 3b represents an argumentation setup AS in which a, b, c and t are literals that are not queryable. As a result, there is no other future argumentation setup than the current setup: $F(AS) = \{AS\}$. There is no argument for t in $Arg(AS)$, hence t is unsatisfiable for every $AS' \in F(AS)$. However, no rule or literal is labelled U since the algorithm in [14] only labels a non-queryable literal U if all rules for this literal are labelled U; a rule only gets labelled U if at least one antecedent of that rule is labelled U. Because of this support cycle, there is no place to start labelling.

Due to the irrelevant label problem and the support cycle problem, the algorithm from [14] fails to recognise the stability of some argumentation setups. We present a solution to these problems in Sections 4.2 and 4.3.

4.2. Reasoning with possible future labels

In this section, we present an alternative labelling method that bypasses the irrelevant label problem by reasoning with possible future labels. Whereas the approximation algorithm presented in [14] relies on a partial labelling function L that assigns at most one label to each literal in $\mathcal{L}$ and rule in $\mathcal{R}$ ($L : \mathcal{L} \cup \mathcal{R} \nrightarrow \{U, D, O, B\}$ where $\nrightarrow$ denotes a partial function), we propose a labelling L' that assigns a quadruple of four booleans $\langle u, d, o, b \rangle$ to each literal and rule. Each boolean corresponds to an acceptability status. Intuitively, the truth value of a boolean belonging to a literal or rule represents the possibility that this literal or rule may become unsatisfiable (u), defended (d), out (o) or blocked (b) in a future argumentation setup. Similar to the approach in [14], labels of rules depend on the labels of their antecedent literals and labels of literals depend on the labels of rules for that literal. Literals and rules are labelled incrementally, starting from queryable literals and literals for which there is no rule and relabelling literals and rules based on the resulting new labels, until no new label can be added.

Definition 8 (Quadruple labelling L')**.** Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ be an argumentation setup. The **labelling function** $L' : \mathcal{L} \cup \mathcal{R} \to \{0,1\} \times \{0,1\} \times \{0,1\} \times \{0,1\}$ assigns a label $\langle u, d, o, b \rangle$ to each literal or rule in $\mathcal{L} \cup \mathcal{R}$. Given a literal or rule $x \in \mathcal{L} \cup \mathcal{R}$, we write $\neg u(x)$ [resp. $\neg d(x), \neg o(x), \neg b(x)$] iff the u- [resp. d-, o-, b-] boolean of x's label is False and $u(x)$ [resp. $d(x), o(x), b(x)$] iff the u- [resp. d-, o-, b-] boolean of x's label is True. We say that a rule or literal x is **labelled stable** by L' iff exactly one of the booleans is True: $L'(x)$ is $\langle 1,0,0,0 \rangle$, $\langle 0,1,0,0 \rangle$, $\langle 0,0,1,0 \rangle$ or $\langle 0,0,0,1 \rangle$.
Given a literal $l \in \mathcal{L}$, $L'(l) = \langle u, d, o, b \rangle$ where:
literal cannot become unsatisfiable: $\neg u(l)$ iff:
- L-U-a) $l \in \mathcal{K}$; or
- L-U-b) there is a rule r for l with $\neg u(r)$.

literal cannot become defended: $\neg d(l)$ iff:
- L-D-a) $-l \in \mathcal{K}$; or
- L-D-b) $l \notin \mathcal{Q}$ and for each rule r for l: $\neg d(r)$; or
- L-D-c) $l \notin \mathcal{Q}$ and there is a rule r' for $-l$ with $\neg u(r')$ and $\neg o(r')$.

literal cannot become out: $\neg o(l)$ iff:
- L-O-a) $l \in \mathcal{K}$; or
- L-O-b) for each rule r for l: $\neg d(r)$, $\neg o(r)$ and $\neg b(r)$; or
- L-O-c) $l \notin \mathcal{Q}$ and for each rule r for l: $\neg o(r)$; or
- L-O-d) $l \notin \mathcal{Q}$ and there is a rule r for l with $\neg u(r)$ and $\neg o(r)$.

literal cannot become blocked: $\neg b(l)$ iff:
- L-B-a) $l \in \mathcal{Q}$; or
- L-B-b) for each rule r for l: $\neg d(r)$ and $\neg b(r)$; or
- L-B-c) for each rule r for l: $\neg b(r)$ and for each rule r' for $-l$: $\neg d(r')$ and $\neg b(r')$.
- L-B-d) there is a rule r for l with $\neg u(r)$, $\neg o(r)$ and $\neg b(r)$ and for each rule r' for $-l$: $\neg d(r')$ and $\neg b(r')$.

Given a rule $r \in \mathcal{R}$, $L'(r) = \langle u, d, o, b \rangle$ where:
rule cannot become unsatisfiable: $\neg u(r)$ iff:
- R-U-a) for each antecedent l of r: $\neg u(l)$.

rule cannot become defended: $\neg d(r)$ iff:
- R-D-a) there is an antecedent l of r with $\neg d(l)$.

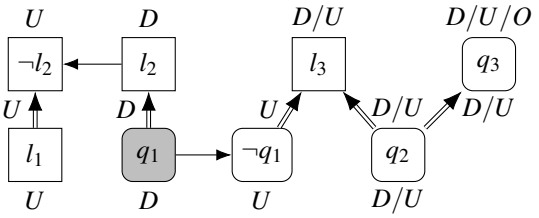


Figure 4. Quadruple labelling example.

rule cannot become out: $\neg o(r)$ iff:
- R-O-a) for each antecedent l of r: $\neg o(l)$; or
- R-O-b) there is an antecedent l of r with $\neg d(l)$ and $\neg o(l)$ and $\neg b(l)$.

rule cannot become blocked: $\neg b(r)$ iff:
- R-B-a) for each antecedent l of r: $\neg b(l)$; or
- R-B-b) there is an antecedent l of r with $\neg d(l)$ and $\neg b(l)$.

Example 6. We give some intuition by labelling the *AS* from Figure 4. Some rules apply if (the negation of) a literal is in $\mathcal{K}$ or $\mathcal{Q}$, e.g. q_1 is labelled $\langle 0,1,0,0\rangle$ by Definition 8 case L-U-a, L-O-b and L-B-a: there is an observation-based argument for q_1 that cannot be attacked in any future setup. The absence of rules for a literal is informative for the acceptability status as well: e.g. l_1 is labelled $\langle 1,0,0,0\rangle$ by L-D-b, L-O-b/c and L-B-b/c.

Other labels are based on the rules for (the negation of) a literal and propagate properties of (attacks on) subarguments. For example, $q_1 \Rightarrow l_2$ is labelled $\langle 0,1,0,0\rangle$ by R-U-a, R-O-a and R-B-a and l_2 is labelled $\langle 0,1,0,0\rangle$ by L-U-b, L-O-c/d and L-B-c/d. Some literals and rules cannot be labelled stable, but still some acceptability status(es) can be excluded: e.g. the rule $q_2 \Rightarrow q_3$ is labelled $\langle 1,1,0,0\rangle$ by case R-O-a and R-B-a.

Example 7 (Alternative labelling Figure 3a)**.** Consider the L' labelling for the argumentation setup from Figure 3a. q_1 is in the knowledge base, so by Definition 8, $L'(q_1) = \langle 0,1,0,0\rangle$. Then $L'(q_1 \Rightarrow a) = L'(q_1 \Rightarrow b) = \langle 0,1,0,0\rangle$ by R-U-a, R-O-a and R-B-a. q_2 and q_3 are queryable but not in the knowledge base and there are no rules for q_2 or q_3, so by Case L-O-b and L-B-a: $L'(q_2) = L'(q_3) = \langle 1,1,0,0\rangle$. For the rules $q_2 \Rightarrow \neg a$ and $q_3 \Rightarrow \neg b$, only the d- and u-booleans are True by R-O-a and R-B-a. As a result, for the literals a and b only the d- and b-booleans are True by L-U-b and L-O-c, which implies by R-U-a and R-O-a that $L'(b \Rightarrow t) = L'(a \Rightarrow t) = \langle 0,1,0,1\rangle$. Finally, t is labelled $L'(t) = \langle 0,0,0,1\rangle$ (by L-U-b, L-D-c and L-O-c/d), so t is labelled stable by L'.

In Example 7 we saw that t is labelled stable by our labelling function L', but its stability was not detected by [14]'s labelling function L. In general, each literal or rule labelled stable by L is also labelled stable by L', but L' covers more stable setups than L.

4.3. Preprocessing

The new labelling proposed in the previous section does not solve the support cycle problem: if we would apply the labelling L' from Definition 8 to the argumentation setup from Figure 3b, all literals l (including literal t) would be labelled $\langle 1,1,1,1\rangle$. In order to solve this issue, we add a preprocessing step, which is specified in Algorithm 1. The

Algorithm 1 Preprocessing step

1: **procedure** PREPROCESS($\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}$)
2: Label each literal l s.t. $l \in \mathcal{Q} \wedge -l \notin \mathcal{K}$ as $\langle 1,1,1,1 \rangle$
3: Label all other literals as $\langle 1,0,0,0 \rangle$
4: Label each $r \in \mathcal{R}$ as $\langle 1,0,0,0 \rangle$
5: **while** a label changed in the previous loop **do**
6: **for** Rule r in $\mathcal{R}$ **do**
7: **if** $L(r) = \langle 1,0,0,0 \rangle$ and for each $l \in \texttt{ants}(r)$: $L(l) \neq \langle 1,0,0,0 \rangle$ **then**
8: Label r as $\langle 1,1,1,1 \rangle$
9: Label $\texttt{cons}(r)$ as $\langle 1,1,1,1 \rangle$

idea of this algorithm is that initially, all literals that cannot be in the knowledge base in a future setup and all rules are labelled $\langle 1,0,0,0 \rangle$ (i.e. unsatisfiable). Then, the algorithm incrementally removes unsatisfiable labels of rules for which all antecedents are not labelled $\langle 1,0,0,0 \rangle$, and of the consequents of these rules, based on the intuition that there may be an argument based on these rules in a future setup.

Example 8 (Alternative labelling Example 5)**.** We reconsider Figure 3b, assuming that the preprocessing step has been executed. In Line 3, all literals (a, b, c and t) are labelled $\langle 1,0,0,0 \rangle$. Since the if-statement in Line 7 never returns true, no rule or literal gets another label, so the while loop is executed only once. After termination of Algorithm 1, all literals are still (correctly) labelled $\langle 1,0,0,0 \rangle$.

4.4. Properties of the proposed algorithm

In this subsection, we present properties of STABILITY, our proposed algorithm, which runs PREPROCESS on the argumentation setup and then labels all literals and rules by repeatedly applying Definition 8. First, we consider STABILITY's soundness.

Proposition 2 (Soundness stability labelling)**.** *Given an argumentation setup $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ and labelling L' after executing the* STABILITY *algorithm, if a literal $l \in \mathcal{L}$ is labelled stable in AS, then l is stable in AS.*

Soundness can be proven by systematically analysing all argumentation setups in which a literal l is labelled stable (e.g. $l \in \mathcal{K}$ or [$l \notin \mathcal{Q}$ and there is no rule for l in $\mathcal{R}$]) and proving that l is stable in each of them. Next, we consider completeness. As illustrated in Example 9, STABILITY is not complete for all argumentation setups.

Example 9 (Example 3 continued)**.** Consider the argumentation setup $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ where $\mathcal{L}$, $\mathcal{R}$ and $\mathcal{Q}$ are as in Figure 2, but $\mathcal{K} = \{\neg sm, rm\}$. STABILITY does not label f stable: it expects a future argument for f based on $sd, \neg rd, d \Rightarrow f$, where the argument for sd is based on $sp, \neg b \Rightarrow sd$ and the argument for $\neg rd$ is based on $\neg rp, b \Rightarrow rd$. However, this argument would require both b and $\neg b$ to be in the knowledge base, which violates the consistency criterion. In fact, for each AS' in $F(AS)$ there is no argument for f in $Arg(AS')$, so f should be labelled $\langle 1,0,0,0 \rangle$.

Example 9 shows that there are argumentation setups where the STABILITY algorithm wrongfully takes the possibility into account that there exists an argument for a

literal in a future argumentation setup. Specifically, this issue is caused by *inconsistent potential arguments*, which we define next.

Definition 9 (Potential argument)**.** Let $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ be an argumentation setup. A **potential argument** A^p inferred from AS is:

- c iff $c \in \mathcal{Q}$ and $-c \notin \mathcal{K}$. $\texttt{prem}(A^p) = \{c\}$; $\texttt{conc}(A^p) = c$; and $\texttt{sub}(A^p) = \{c\}$.
- $A_1, \ldots, A_m \Rightarrow c$ iff there is a rule $c_1, \ldots, c_m \Rightarrow c$ in $\mathcal{R}$ and for each $i \in [1 \,..\, m]$: A_i is a potential argument inferred from AS and $\texttt{conc}(A_i) = c_i$. $\texttt{prem}(A^p) = \texttt{prem}(A_1) \cup \ldots \cup \texttt{prem}(A_m)$; $\texttt{conc}(A^p) = c$; and $\texttt{sub}(A^p) = \texttt{sub}(A_1) \cup \ldots \cup \texttt{sub}(A_m) \cup \{A\}$.

We denote the set of potential arguments by $P(AS)$. Given some $A^p, B^p \in P(AS)$, A^p **p-attacks** B^p iff there is a $B' \in \texttt{sub}(B^p)$ s.t. $\texttt{conc}(A^p) = -\texttt{conc}(B')$ and $-\texttt{conc}(B') \notin \mathcal{K}$; A^p is **inconsistent** with B^p iff $\{a, -a\} \in \texttt{prem}(A^p) \cup \texttt{prem}(B^p)$ for some $a \in \mathcal{L}$.

Note that, for any argumentation setup AS, each argument inferred from AS or some future setup is a potential argument in $P(AS)$. However, there may be a potential argument $A^p \in P(AS)$ such that there is no $AS' \in F(AS)$ with $A^p \in Arg(AS')$, but then A^p must be inconsistent *with itself*, like in Example 9. Example 10 reveals another issue, where stability is not detected due to an inconsistency of two *different* potential arguments.

Example 10 (Mutual inconsistency issues)**.** Given the argumentation setup AS illustrated in Figure 5, for each $AS' = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}')$ in $F(AS)$, l_1 is blocked in AS': if $\neg q_2 \notin \mathcal{K}'$ then there is an argument for l_1 based on $q_2 \Rightarrow l_1$; otherwise there is an argument for l_1 based on $\neg q_2 \Rightarrow l_1$. However, l_1 is not labelled $\langle 0,0,0,1 \rangle$ because STABILITY wrongfully anticipates a future setup AS' in which each argument for l_1 is attacked by an argument in $G(AS')$ (thus $o(l_1)$). For the same reason, $\neg l_1$ is labelled $d(\neg l_1)$ and therefore $L'(\neg l_1) \neq \langle 0,0,0,1 \rangle$, although $\neg l_1$ is blocked in each $AS' \in F(AS)$.

The issues illustrated in Examples 9 and 10 can be generalised to the following two situations. Given an argumentation setup $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ and a literal $l \in \mathcal{L}$:

- l is **inconsistently supported** in AS iff there are $A^p, B^p \in P(AS)$ such that $\texttt{conc}(A^p) = \texttt{conc}(B^p) = l$ and A^p is inconsistent with B^p.
- l is **inconsistently attacked** in AS iff there is a $C^p \in P(AS)$ such that $\texttt{conc}(C^p) = l$ and there are $A^p, B^p \in P(AS)$ such that A^p p-attacks C^p, B^p p-attacks C^p and A^p is inconsistent with B^p.

If a potential argument is inconsistent with itself, its conclusion l can be incorrectly labelled $d(l)$ or $b(l)$ (e.g. f in Example 9). Similarly, if two potential arguments with the same conclusion are inconsistent, their conclusion l can be incorrectly labelled $o(l)$ (e.g. l_1 in Example 10). Moreover, if l is inconsistently attacked, l may be incorrectly labelled $d(l)$ (e.g. $\neg l_1$ in Example 10) or $b(l)$. Otherwise, l is labelled stable if it is stable in AS.

Proposition 3 (Conditional completeness stability labelling)**.** *Given an argumentation setup $AS = (\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K})$ and a labelling L' after executing* STABILITY*. If $l \in \mathcal{L}$ is stable in AS and l is not inconsistently supported or attacked, then l is labelled stable by L'.*

Finally, the proposed algorithm runs in polynomial time, which makes it suitable for practical applications such as human-computer inquiry dialogues.

Proposition 4. *The time complexity of* STABILITY *is $\mathcal{O}(|\mathcal{L}|^2 \cdot |\mathcal{R}| + |\mathcal{L}| \cdot |\mathcal{R}|^2)$.*

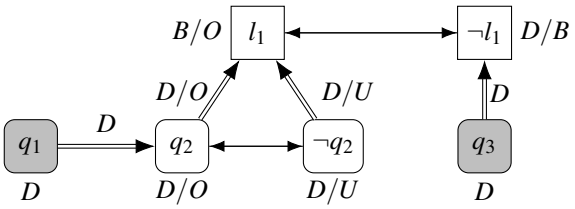


Figure 5. For each $AS' \in F(AS)$: l_1 and $\neg l_1$ are blocked in AS', but not labelled as such due to inconsistent support (l_1) and attack ($\neg l_1$).

5. Related work

We study stability [14]: given a specific structured argumentation setting, can adding information change the acceptability status of some propositional formula? This is a relatively new task within dynamic argumentation, which studies the acceptability of (sets of) arguments or their conclusions in relation to changes on the argumentation framework. Research on dynamic argumentation includes work on the impact of a change operation [5,1], enforcement [2,6], resolution [10] and the relation with belief revision [8,13].

Most research on dynamic argumentation, e.g. [2,5,6], only considers the effect of changes in the *abstract* framework, such as adding an argument. This approach does not take into account dependencies between arguments. For example, adding a new argument A often introduces more arguments having A as a subargument. Conversely, we study the effect of changes in the underlying *structured* argumentation framework.

None of the existing work in structured dynamic argumentation specifically studies stability. We briefly discuss some related research. [10] study resolutions in structured argumentation: they show how the acceptability of *arguments* changes due to a change of *preferences* in the underlying structured argumentation framework. Another related study [13] shows how the acceptability of a specific set of *arguments* can be altered by a minimal number of changes on the premises and/or rules in an argumentation framework, relating dynamic argumentation to belief revision. However, they do not focus on a specific task, such as stability; they do not consider *computational complexity* or provide an efficient (approximation) algorithm. One of the few papers that take computational complexity into account is [1]. The authors propose an efficient algorithm to minimise re-computations after a change in a DeLP program. However, whereas they study acceptability status after a *specific change*, we study the status after *any possible change* in the structured argumentation framework.

Finally, [9] apply a similar strategy to ours for efficiently determining the acceptability status of literals: they create a graph representing (the relation between) literals and rules and incrementally label its nodes and edges. However, they only determine the *current* acceptability status without considering changes.

6. Discussion and conclusion

We have studied the task of detecting stability: given a specific structured argumentation setup, based on a variation on ASPIC$^+$, can adding information result in a changed acceptability status of a specific literal? We have shown that the task is CoNP-hard. This

is problematic in practical applications, such as identifying the termination criterion in human-computer inquiry dialogue. We proposed an algorithm for estimating stability that improves on the algorithm in [14]. We have shown that the refined algorithm is sound and runs in polynomial time. Thanks to these properties, the algorithm has been taken into use as part of an agent handling intake of fraud reports at the Dutch National Police – we provide an English demo that also visualises the agent's stability component.

There are examples of argumentation setups for which the algorithm does not detect that a literal is stable; in our application, this can result in the agent asking unnecessary questions. This issue could be resolved by a further refinement of the algorithm, which lists the knowledge bases $\mathcal{K}'$ of all future setups and checks that each $\mathcal{K}'$ is consistent. However, such an algorithm would have exponential time complexity.

In future work, we plan to extend the argumentation framework and the allowed updates. Furthermore, our demo applies a heuristic to select relevant questions; we plan to specify this formally. Finally, we will extensively evaluate the fraud intake agent.

References

[1] Gianvincenzo Alfano, Sergio Greco, Francesco Parisi, Gerardo Ignacio Simari, and Guillermo Ricardo Simari. An incremental approach to structured argumentation over dynamic knowledge bases. In *Proceedings of the 16th International Conference on Principles of Knowledge Representation and Reasoning*, pages 78–87, 2018.

[2] Ringo Baumann and Gerhard Brewka. Expanding argumentation frameworks: Enforcing and monotonicity results. In *Proceedings of the 2010 conference on Computational Models of Argument: Proceedings of COMMA 2010*, pages 75–86. IOS Press, 2010.

[3] Floris Bex, Joeri Peters, and Bas Testerink. AI for online criminal complaints: From natural dialogues to structured scenarios. In *Artificial Intelligence for Justice Workshop (ECAI 2016)*, pages 22–29, 2016.

[4] Elizabeth Black and Anthony Hunter. An inquiry dialogue system. *Autonomous Agents and Multiagent Systems*, 19(2):173–209, 2009.

[5] Claudette Cayrol, Florence Dupin de Saint-Cyr, and Marie-Christine Lagasquie-Schiex. Change in abstract argumentation frameworks: Adding an argument. *Journal of Artificial Intelligence Research*, 38:49–84, 2010.

[6] Sylvie Doutre and Jean-Guy Mailly. Constraints and changes: A survey of abstract argumentation dynamics. *Argument and Computation*, 9:223–248, 11 2018.

[7] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77:321–357, 1995.

[8] Marcelo Alejandro Falappa, Gabriele Kern-Isberner, and Guillermo Ricardo Simari. Belief revision and argumentation theory. *Argumentation in Artificial Intelligence*, pages 341–360, 2009.

[9] Abdelraouf Hecham, Pierre Bisquert, and Madalina Croitoru. On a flexible representation for defeasible reasoning variants. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, pages 1123–1131, 2018.

[10] Sanjay Modgil and Henry Prakken. Resolutions in structured argumentation. In *Proceedings of the 4th International Conference on Computational Models of Argument*, pages 310–321, 2012.

[11] Henry Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124, 2010.

[12] Marijn Schraagen and Floris Bex. Extraction of semantic relations in noisy user-generated law enforcement data. In *13th International Conference on Semantic Computing*, pages 79–86. IEEE, 2019.

[13] Mark Snaith and Chris Reed. Argument revision. *Journal of Logic and Computation*, 27(7):2089–2134, 2016.

[14] Bas Testerink, Daphne Odekerken, and Floris Bex. A method for efficient argument-based inquiry. In *Proceedings of the 13th International Conference on Flexible Query Answering Systems*, 2019.