Intelligent Systems with Applications 16 (2022) 200110

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Approximating stability for applied argument-based inquiry

Daphne Odekerken [*,a,b], Floris Bex [a,c], AnneMarie Borg [a], Bas Testerink [b]

[a] Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, Utrecht, 3584 CC, the Netherlands
[b] National Police Lab AI, Netherlands Police, Driebergen, the Netherlands
[c] Tilburg Institute for Law, Technology and Society, Tilburg University, Warandelaan 2, Tilburg, 5037 AB, the Netherlands

ARTICLE INFO

Keywords:
Dynamic argumentation
Structured argumentation
Inquiry
Law enforcement

ABSTRACT

In argument-based inquiry, agents jointly construct arguments supporting or attacking a topic claim to find out if the claim can be accepted given the agents' knowledge bases. While such inquiry systems can be used for various forms of automated information intake, several efficiency issues have so far prevented widespread application. In this paper, we aim to tackle these efficiency issues by exploring the notion of stability: can additional information change the justification status of the claim under discussion? Detecting stability is not tractable for every input, since the problem is CoNP-complete, yet in practical applications it is essential to guarantee efficient computation. This makes approximation a viable alternative. We present a sound approximation algorithm that recognises stability for many inputs in polynomial time and discuss several of its properties. In particular, we show that the algorithm is sound and identify constraints on the input under which it is complete. As a final contribution of this paper, we describe how the proposed algorithm is used in three different case studies at the Netherlands Police.

1. Introduction

Human or artificial agents that cooperate in collecting information in order to find out if a given (topic) claim should be accepted, engage in an inquiry dialogue (Walton and Krabbe, 1995). A domain that is especially suitable for artificial intelligence (AI) systems that perform automated inquiry is law enforcement. For example, the Netherlands Police every day receives 200–300 reports of online trade fraud (Schraagen et al., 2019), as well as 800–900 requests for assistance or information from international law enforcement agencies (Testerink et al., 2019a). In such cases, an AI-based support system that automatically handles at least part of the inquiry would be of much help. However, AI systems in law enforcement will have to fully comply with requirements on, e.g., transparency and proportionality, law enforcement having been identified as a high-risk domain in the European Commission's recent proposal for a regulatory framework on AI (European Commission, 2021). For example, with respect to proportionality, no more information than is needed for making a decision should be collected. With respect to transparency, it makes sense to use more interpretable symbolic AI methods rather than data-driven machine learning methods (Rudin, 2019). In this paper, we focus on the symbolic AI technique of *computational argumentation*.

Computational argumentation is a subfield of AI concerning reasoning with incomplete or inconsistent information (Atkinson et al., 2017). Two central concepts in computational argumentation are abstract argumentation frameworks (Dung, 1995) – sets of arguments and the attack relations between them – and structured argumentation frameworks – e.g., ASPIC$^+$ (Prakken, 2010), where arguments are constructed from an argumentation theory, containing a knowledge base and a set of rules, and the attack relation is based on the individual elements in the arguments. For both abstract and structured argumentation frameworks, one can determine sets of arguments that can collectively be considered as acceptable. This notion of acceptability in computational argumentation can be used in argument-based inquiry, where the participating agents construct arguments that influence the acceptability status of some central topic claim(s). An advantage of argument-based inquiry is that the outcome – whether a topic claim should be accepted or not – can be explained by arguments related to this claim. If these arguments are then based on legal rules, the outcome can thus be related to an explicit representation of the legal background.

Whereas argument-based inquiry has been studied before (Black and Hunter, 2009; Fan and Toni, 2012; Parsons et al., 2002), efficiency – in terms of both the length of the dialogue and computational efficiency – remains an issue. For example, the dialogue system for (warrant) inquiry

* Corresponding author at: Department of Information and Computing Sciences, Utrecht University, Princetonplein 5, Utrecht 3584 CC, the Netherlands.
E-mail addresses: d.odekerken@uu.nl (D. Odekerken), f.j.bex@uu.nl (F. Bex), a.borg@uu.nl (A. Borg), bas.testerink@politie.nl (B. Testerink).

https://doi.org/10.1016/j.iswa.2022.200110
Received 21 April 2022; Received in revised form 13 July 2022; Accepted 8 August 2022
Available online 12 August 2022

proposed by Black and Hunter (2009) exhaustively constructs all the arguments for or against a particular topic claim to determine the claim's acceptability status, which typically leads to redundant interactions in the dialogue, conflicting with the principle of proportionality (i.e. no more information should be collected than is needed for making a decision). In addition, computational argumentation, particularly using structured argumentation frameworks, is largely based on reasoning in logic (cf. Besnard et al. (2014)). As a consequence, many problems in argumentation are situated in high complexity classes (see e.g. Dvořák and Dunne, 2017; Parsons et al., 2002) and are therefore intractable if P $\neq$ NP, a common assumption that we also make in this paper. We consider this to be an important reason why argument-based systems for inquiry are not yet widely used in actual applications.

In order to resolve the issue of redundant interactions, we explore the notion of *stability*: can any additional information still change the acceptability status of the topic claims? If not, it does not make sense to inquire into additional arguments related to this claim. Consequently, the notion of stability is a natural termination criterion for argument-based inquiry. In this paper, we study the task of detecting stability in argumentation frameworks based on a variant of the structured argumentation framework ASPIC$^+$ (Prakken, 2010), which is one of the main approaches to structured argumentation (Besnard et al., 2014).

Studying stability does not solve the problem of computational efficiency, since the problem of detecting stability is in a high complexity class: we will prove that it is CoNP-complete. Problems in this complexity class are generally considered intractable: there is no exact polynomial-time algorithm for any CoNP-complete problem. This means that each algorithm that accurately detects stability for every input needs exponential time. A naive but accurate stability detection algorithm would for example generate all argumentation theories that can be obtained by adding information and subsequently compare the acceptability statuses of topic claims. We will experimentally show that such an algorithm has an impractically large running time even for very small inputs. Since practical applications require fast computation for arbitrary inputs, we propose a polynomial-time approximation algorithm for estimating stability. By proving the algorithm's worst-case time complexity, we guarantee an upper limit on the running time in relation to the size of the input. Having a polynomial algorithm paves the road to efficient stability detection that is scalable to large argumentation theories. We will validate this by an empirical analysis on synthesised argumentation theories as well as examples from the law enforcement domain.

The efficiency of our algorithm comes at a cost: it is not exact, which is an inevitable property of applying polynomial algorithms for CoNP-complete problems. We will extensively evaluate the performance of the algorithm, both empirically and theoretically. The empirical analysis indicates that the performance is satisfactory for the inputs that we tested. Thanks to our theoretical analysis, we can give more general guarantees on the algorithm's performance. First, we will show that the algorithm is sound: if the algorithm determines that a claim is stable, then indeed no additional information can change the acceptability of this claim. When applied as a termination criterion in inquiry, this ensures that the inquiry dialogue is not terminated too early. In addition, we will show that the algorithm is complete given specific conditions on the input. If these conditions are met, the algorithm is able to recognize all stable situations and consequently makes sure that the inquiry is terminated exactly in time.

As a final contribution, we describe how our algorithm is applied at the Netherlands Police in argument-based inquiry systems for three different processes with a legal reasoning subtask, including two intake processes and a specific human-in-the-loop classification task. These systems operate in various domains, namely fraud intake, handling international messages and the classification of fraudulent web shops. The fraud intake system has been used by hundreds of users every day since September 2019. Although various domains and applications require some variance in implementation, our algorithm for detecting stability could be directly used in all three use cases. This demonstrates that our proposed algorithm is generally applicable for real-life inquiry, even in high-risk domains such as law enforcement.

Outline Before we formally define the stability problem in Section 3, we discuss the preliminaries in Section 2. In Section 4, we propose an approximation algorithm for detecting stability and prove properties of soundness, conditional completeness and time complexity. In addition, we report on experiments measuring the algorithm's running time. In Section 5, we discuss the three case studies at the Netherlands Police. Finally, we discuss related work in Section 6 and conclude in Section 7.

Contributions This work unites and extends our earlier research in Testerink et al. (2019b) and Odekerken et al. (2020). Our main new contributions include a generalisation of the notion of conflict from classical negation to the use of a contradiction function; a more precise analysis of (in)completeness cases; full proofs for all complexity, soundness and conditional completeness results; empirical analyses of algorithms for detecting stability; and an extensive description of three case studies at the Netherlands Police. In addition, we discuss our design choices in more detail and connect our approach to related work in conversational AI and dynamic argumentation. In order to enable reproducibility, we made the implementations of the algorithms, data set generators and visualisation tool presented in this paper available at https://github.com/DaphneO/StabilityLabelAlgorithm.

2. Preliminaries

We will study the problem of stability in the context of argumentation frameworks (Dung, 1995) where arguments are constructed by an instantiation of the ASPIC$^+$ framework (Prakken, 2010). In this section, we first recall ASPIC$^+$ definitions and specify how arguments are constructed from an argumentation system and a knowledge base. Subsequently, we define abstract argumentation frameworks based on ASPIC$^+$ arguments and the attacks between them. Based on these abstract argumentation frameworks, we can derive which arguments should be accepted under grounded semantics (Dung, 1995). Finally, we define a justification status for statements based on the existence and/or acceptability status of arguments for and against them.

2.1. ASPIC+

ASPIC$^+$ is a general framework for structured argumentation. As a result of various revisions and extensions in the development of the framework over the years, it is not a single framework, but rather a family of frameworks varying on several elements (Modgil and Prakken, 2018). In this paper, we define a light-weight ASPIC$^+$ instantiation that suffices for our purpose. We will motivate our choices at the end of this section.

The basic notion of ASPIC$^+$ is that of an argumentation system, which consists of a logical language $\mathscr{L}$, a set of rules $\mathscr{R}$ and a contradiction function $\bar{}$. An argumentation system is defined as follows.

Definition 1. (*Argumentation system*) An **argumentation system** is a tuple $AS = (\mathscr{L}, \mathscr{R}, \bar{})$ where:

- $\mathscr{L}$ is a finite logical language consisting of propositional literals.
- $\mathscr{R}$ is a finite set of defeasible rules of the form $a_1, \ldots, a_m \Rightarrow c$ such that $\{a_1, \ldots, a_m, c\} \subseteq \mathscr{L}$, where $\{a_1, \ldots, a_m\}$ are the *antecedents* and c is the *consequent* of the rule. For any rule r, the antecedents and consequent are denoted by $\texttt{ants}(r)$ and $\texttt{cons}(r)$, respectively.
- $\bar{}$ is a contradiction function from $\mathscr{L}$ to $2^{\mathscr{L}}$. l is a *contradictory* of m iff $m \in \bar{l}$ and $l \in \bar{m}$. Each $l \in \mathscr{L}$ has at least one contradictory. For each $l \in \mathscr{L} : l \notin \bar{l}$.

In our examples we often use classical negation ($\neg$) as contradiction function: for each $l \in \mathscr{L} : \bar{l} = \{\neg l\}$ and $\overline{\neg l} = \{l\}$. An argumentation theory is a combination of an argumentation system AS and a knowledge

base $\mathscr{K} \subseteq \mathscr{L}$.

Definition 2. (*Knowledge base*) A **knowledge base** $\mathscr{K} \subseteq \mathscr{L}$ over an argumentation system $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ is a set of literals that is consistent (i.e., for each pair $l, m \in \mathscr{K} : l \notin \overline{m}$).

Definition 3. (*Argumentation theory*) An **argumentation theory** $AT = (AS, \mathscr{K})$ is a pair consisting of an argumentation system AS and a knowledge base $\mathscr{K}$.

Given an argumentation theory, we can derive two types of arguments: observation-based arguments are based on elements from the knowledge base, whereas rule-based arguments are constructed by chaining applications of defeasible rules.

Definition 4. (*Arguments*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory. An **argument** A on the basis of the argumentation theory AT is a structure obtainable by applying one or more of the following steps finitely many times:

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

We denote by $Arg(AT)$ the set of **arguments** on the basis of AT. An argument with conclusion c is referred to as "an argument for c" and an argument with top rule r by "an argument based on r".

In order to clarify the notion of arguments, we introduce an example of an argumentation theory in the domain of online trade fraud, visualised in Fig. 1. Online trade fraud is a type of high-volume crime that concerns fake web shops and malicious second-hand traders on platforms such as eBay. We will use this argumentation theory as a running example in this section. Note that this is a simplified example: the actual argumentation theory that is used by the Netherlands Police is more complex. For more details on the argumentation theories used in actual applications, we refer to Section 5.

Example 1. (*Argument*) Let $AT = (AS, \mathscr{K})$ with $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ be an argumentation theory in the domain of online trade fraud, visualised in Fig. 1. $\mathscr{L}$ consists of the following literals and their negations:

- 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: suspect url;
- s: screenshot of payment;
- t: trusted web shop;
- cd: citizen delivered;
- rd: citizen received delivery;
- d: deception;
- f: fraud.

Let $\mathscr{K} = \{b, sm, \neg rp, u, t\}$. Then there are, for example, observation-based arguments for sm and b, and a rule-based argument for cd based on the rule $sm, b \Rightarrow cd$, having these observation-based arguments and itself as subarguments. Similarly, there are rule-based arguments for $\neg rd$, d, $\neg f$ and f.

Arguments can be in conflict. In ASPIC$^+$, attacks between arguments are based on the arguments' structure. In this paper, we only consider rebuttal attacks, where arguments attack each other on the conclusion of a defeasible inference, as defined next.

Definition 5. (*Attack*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$. For two arguments $A, B \in Arg(AT)$ we say that A **attacks** B (on B') iff $\texttt{conc}(A) \in \bar{l}$ for some $B' \in \texttt{sub}(B)$ of the form $B''_1, \ldots, B''_n \Rightarrow l$.

Example 2. (*Online trade fraud*) In our running example on online trade fraud (Fig. 1), $Arg(AT)$ includes an argument for f based on the rule $cd, \neg rd, d \Rightarrow f$ and an argument for $\neg f$ based on $b, t \Rightarrow \neg f$. These arguments attack each other.

Given an argumentation theory $AT = (AS, \mathscr{K})$ we can define an *argumentation framework* (Dung, 1995) consisting of a set of arguments $\mathscr{A}$ and a set of attacks $\mathscr{C}$.

Definition 6. (*Argumentation framework*) Let AT be an argumentation theory $AT = (AS, \mathscr{K})$. An **argumentation framework** AF defined by AT, is a pair $AF = \langle \mathscr{A}, \mathscr{C} \rangle$ where $\mathscr{A} = Arg(AT)$ and $(X, Y) \in \mathscr{C}$ iff X attacks Y in AT.

At this point, we have defined our instantiation of the ASPIC$^+$ framework. Readers familiar with ASPIC$^+$ will notice that our instantiation is simplified in that we only consider axiom premises, a finite set of defeasible rules, rebuttal attacks and no preferences. The simplifications provide a minimal number of formal concepts that maintains the theoretical underpinnings of computational argumentation whilst being sufficiently expressive for real-world applications, such as the application of inquiry dialogue at the Netherlands Police, as we motivate below.

First, full ASPIC$^+$ makes a distinction between ordinary premises (which are defeasible and therefore can be attacked) and axiom premises that cannot be attacked. We only consider axioms in the knowledge base, because in inquiry dialogues the reasoning is always based on observations – or evidence – that can be considered to have been

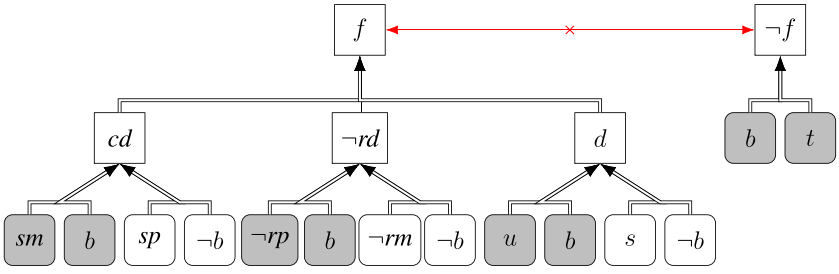


Fig. 1. Example of an argumentation theory AT from the law enforcement domain. (Rounded) squares represent literals from $\mathscr{L}$ and literals in $\mathscr{K}$ are shaded, where rounded squares are queryable literals, to be discussed in Section 3.1. Rules are represented by double-lined arrows and attacks as single-lined arrows. Note that the literals b and $\neg b$ are visualised multiple times and and the contrariness relation between b and $\neg b$ is omitted for clarity.

established with certainty in the context under consideration (Black and Hunter, 2009). For instance, in the context of online trade fraud, the reasoning is based on basic observations provided by citizens or gathered from a police database, and we assume that the fact that such observations have been made can be established with certainty. Of course, it can be argued that we can never be completely certain about an observation: for example, the observation may have been automatically extracted from (noisy) text by a less-than-perfect classifier (cf. Section 5.1.2). However, we solve such issues with establishing observations as part of the information extraction step, which differs per use case. For example, in the fraud intake application, we included an observation validation step in which the citizen can check and correct the extracted observations, as we will explain in Section 5.1.2.

Second, we only consider defeasible rules without preferences. If we would also consider strict rules and/or a preference relation between defeasible rules, we would need to train police employees to construct argumentation theories in such a way that the rationality postulates, i.e. desirable properties for structured argumentation, are satisfied. Caminada and Amgoud (2007) proposed the rationality postulates of sub-argument closure, closure under strict rules, direct consistency and indirect consistency. These four rationality postulates are satisfied if the argumentation theory is well-formed (Prakken, 2010). Since we do not consider strict rules or preferences between arguments and the knowledge base is required to be consistent, each argumentation theory as defined in Definition 3 is well-formed. This makes it more feasible for police employees without background in computational argumentation to adapt or create rule sets.

Third, we define the language and rules of the argumentation system as finite sets. This enables us to assign a label to each literal and rule in the polynomial algorithm proposed later in this paper. For the application in e.g. police practice this is not restricting, since it is quite natural to use a limited number of rules and literals are used to capture domain-specific information.

Fourth, we chose to restrict the attack types to rebuttal attacks. In full ASPIC$^+$, arguments can attack each other in three ways: rebuttal, undermining and undercutting attacks (Prakken, 2010). Since our knowledge base only contains axiom premises, undermining attacks on premises cannot occur and therefore do not need to be considered. Furthermore, all notions of conflict in our practical use-cases can be modelled using rebuttal attacks, as we illustrate with examples in Section 5.

To conclude, our instantiation of ASPIC$^+$, in which we only consider a finite set of axiom premises, defeasible rules without preferences and rebuttal attacks, makes it more feasible for police employees to construct or edit argumentation systems, but still suffices to create meaningful inquiry agents for our use cases at the police. For future use cases, it might be interesting to allow rule preferences and undercutting attacks as well, provided that preferences are defined in such a way that the argument ordering is reasonable. However, these extensions would complicate the definitions, solutions and proofs; therefore, we leave this for future work.

2.2. Argumentation semantics

The evaluation of arguments is done using the semantics of Dung (1995). In this paper, we focus on grounded semantics, where the grounded extension is the set of arguments that should be accepted according to the grounded semantics.

Definition 7. (*Grounded extension*) Let $AF = \langle \mathscr{A}, \mathscr{C} \rangle$ be an argumentation framework and $S \subseteq \mathscr{A}$. Then:

- S is **conflict free** iff for each $X, Y \in S : (X, Y) \notin \mathscr{C}$.
- $X \in \mathscr{A}$ is **acceptable with respect to** S iff for each $Y \in \mathscr{A}$ such that $(Y, X) \in \mathscr{C}$, there is a $Z \in S$ such that $(Z, Y) \in \mathscr{C}$.
- S is an **admissible set** iff S is conflict free and $X \in S$ implies that X is acceptable with respect to S.
- S is a **complete extension** iff S is admissible and for each X: if $X \in \mathscr{A}$ is acceptable with respect to S then $X \in S$.
- S is the **grounded extension** of AF iff it is the set inclusion minimal complete extension.

For an argumentation theory AT that defines an argumentation framework AF, we refer to the grounded extension of AF with $G(AT)$.

Apart from the grounded semantics, various other argumentation semantics exist; see Baroni et al. (2011) for an overview of some well-known options. In this paper, we concentrate on the grounded semantics, for two reasons. First, the grounded semantics is the most skeptical complete semantics, which implies that as few arguments as possible are accepted, while still being a complete extension. This fits the application in police investigation: in our case studies, we only accept a statement if, given current information, for every reasonable position (extension) that one can take, an argument for the statement is in that extension. This requirement is only met by the grounded extension, which is the intersection of all complete extensions of the argumentation framework (Dung, 1995). Second, Cerutti et al. (2020) show that using a grounded reasoner is already an almost perfect approximation algorithm for many semantics.

2.3. Justification status of statements

Since we study inquiry dialogues, where the goal is to find out if a statement is justified (Walton and Krabbe, 1995), we are interested in the justification status of statements rather than arguments. In ASPIC$^+$, a statement is justified under grounded semantics if and only if there exists an argument for that statement that is justified; otherwise, the statement is not justified (Modgil and Prakken, 2013, Definition 15). Our application in police practice however demands a distinction between more than two statuses, which gives a more fine-grained notion of justification in the argumentation theory. This enables us, for example, to differentiate between exculpatory evidence and a lack of evidence in police inquiry.

In earlier work, Prakken and Vreeswijk (2001, Definition 4.28) distinguish three possible statuses that can be assigned to conclusions of arguments; Wu and Caminada (2010) distinguish six statuses to be assigned to literals; and Hecham et al. (2018) propose six statement labels for different variations of defeasible reasoning.

We distinguish four justification statuses, in which we have a special status *unsatisfiable* for literals for which there is no argument, in contrast to Prakken and Vreeswijk (2001) and Wu and Caminada (2010). Our four justification statuses are similar to the UNSUP, IN$_{def}$, OUT$_{def}$ and AMBIG statement labels by Hecham et al. (2018).

Definition 8. (*Multi-valued statement justification status*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \_)$ and let $AF = \langle \mathscr{A}, \mathscr{C} \rangle$ be the argumentation framework defined by AT. Then the justification status of $l \in \mathscr{L}$ in AT is:

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

The *defended* status that we defined here corresponds to the justified status of conclusions of arguments in Modgil and Prakken (2013, Definition 15). Conclusions of arguments that are not justified can be either *out* or *blocked*, where intuitively a literal that is *blocked* can be accepted

by a credulous reasoner, under different semantics (Baroni et al., 2011). The distinction between *out* and *blocked* is also convenient for the stability algorithm in Section 4 – in Example 5 we will demonstrate this.

Example 3. (*Example 2 continued*) In the argumentation theory AT from Fig. 1, $G(AT)$ contains (unattacked) arguments for sm, b, $\neg rp$, u, t, cd, $\neg rd$ and d, so these literals are defended in AT. There are arguments for f and $\neg f$ in $Arg(AT)$ that attack each other, but these are not attacked by any argument in $G(AT)$ or acceptable w.r.t. $G(AT)$, so f and $\neg f$ are blocked in AT. Each other literal $l \in \mathscr{L}$ is unsatisfiable in AT: there is no argument for l in $Arg(AT)$.

Note that the four statement justification statuses are mutually exclusive and complementary, so each literal in each argumentation theory has exactly one justification status.

Lemma 1. *The statement justification statuses unsatisfiable, defended, out and blocked from Definition 8 are mutually exclusive and complementary.*

Proofs for all lemmas and propositions in this paper can be found in the appendix; the proof of Lemma 1 is available in Appendix B.3.

3. Stability

In this section, we introduce the problem of stability and discuss its complexity. Stability can be seen as a dynamic variant on the justification status defined in the previous section: the justification status determines if a literal l is justified *given current information*. However, in an inquiry dialogue, more information can be added by, for instance, querying a citizen or data base, which possibly results in a change of l's justification status. If additional information cannot influence l's justification status, then we say that l is *stable*.

3.1. Defining stability

First, we define the stability problem, which is a problem in dynamic argumentation (Doutre and Mailly, 2018). Informally, a literal is stable in some argumentation theory if its justification status cannot change by adding more literals to the knowledge base. We impose some restrictions on the allowed additions on the knowledge base, for two reasons.

The first reason is that, in any dynamic argumentation setting, allowing any change to the argumentation framework effectively makes the problem trivial; a similar problem was identified by Baumann and Brewka (2010). In our case, allowing any literal to be added to the knowledge base makes the stability problem trivial, as then any literal can be made stable and defended simply by adding it to the knowledge base $\mathscr{K}$ (provided that none of its contradictories is already in $\mathscr{K}$).

The second reason that prevents us from simply adding any literal to the knowledge base is more practical. Since our use cases are within the law enforcement domain, we base our argumentation theories on laws and literals that represent certain legal concepts. In our running example on online trade fraud, 'fraud' is such a concept, which is defined precisely in Article 326 of the Dutch Criminal Code. Because we cannot expect citizens to know the exact legal definition of 'fraud', we do not want to directly ask them if they have been a victim of such fraud. Instead, we want to ask citizens whether they observed certain basic, non-legal facts, such as whether they sent money or received a product, and then use legal rules captured in the rule base $\mathscr{R}$ to derive legal conclusions about fraud.[1] We therefore distinguish between queryable and non-queryable literals, where queryables are a specific set of literals that can be obtained (i.e. added to the knowledge base) by querying the environment (e.g. the citizen or a database).

Definition 9. (*Queryables*) Given an argumentation theory $AT = (AS, \mathscr{K})$ with $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, a set of queryables $\mathscr{Q}$ is a set of literals $\mathscr{L}$ such that $\mathscr{K} \subseteq \mathscr{Q} \subseteq \mathscr{L}$ and if $q \in \mathscr{Q}$ then for each $q' \in \overline{q} : q' \in \mathscr{Q}$.

The set of queryables restricts the literals that can be added to the knowledge base. Note that Definition 9 requires that all contradictories of each literal in $\mathscr{Q}$ are also in $\mathscr{Q}$. The reason for this is that contradictories can be seen as alternative answers to a given query: when querying the environment (e.g., q?), both the literal (q) and its contradictory ($\in \overline{q}$, e.g., $\neg q$) can be given as an answer and added to the knowledge base. Adding a queryable literal q to the knowledge base of an argumentation theory $AT = (AS, \mathscr{K})$ (where $\overline{q} \cap \mathscr{K} = \emptyset$) results in a new argumentation theory $AT' = (AS, \mathscr{K} \cup \{q\})$. The set of all argumentation theories that can be obtained by adding queryables to the knowledge base is the set of future argumentation theories.

Definition 10. (*Future argumentation theories*) The set of **future argumentation theories** $F_{\mathscr{Q}}(AT)$ of an argumentation theory $AT = (AS, \mathscr{K})$ given a set of queryables $\mathscr{Q}$ consists of all argumentation theories $AT' = (AS, \mathscr{K}')$ with $\mathscr{K} \subseteq \mathscr{K}' \subseteq \mathscr{Q}$.

Note that the argumentation theory AT always belongs to the set of future argumentation theories $F_{\mathscr{Q}}(AT)$. Further note that, since all future argumentation theories in $F_{\mathscr{Q}}(AT)$ are argumentation theories in the sense of Definition 3, their knowledge base must be consistent.

Next, we formally define stability based on the notions of future argumentation theories and the justification status of statements. We distinguish four types of stability, relative to the four justification statuses from Definition 8.

Definition 11. (*Stability*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and $\mathscr{Q}$ is a set of queryables. Given a literal $l \in \mathscr{L}$:

- l is **stable-unsatisfiable** in AT w.r.t. $\mathscr{Q}$ iff for each $AT' \in F_{\mathscr{Q}}(AT)$, l is unsatisfiable in AT';
- l is **stable-defended** in AT w.r.t. $\mathscr{Q}$ iff for each $AT' \in F_{\mathscr{Q}}(AT)$, l is defended in AT';
- l is **stable-out** in AT w.r.t. $\mathscr{Q}$ iff for each $AT' \in F_{\mathscr{Q}}(AT)$, l is out in AT';
- l is **stable-blocked** in AT w.r.t. $\mathscr{Q}$ iff for each $AT' \in F_{\mathscr{Q}}(AT)$, l is blocked in AT'.

A literal $l \in \mathscr{L}$ is **stable** in AT w.r.t. $\mathscr{Q}$ iff any of the above cases applies.

Example 4. (*Example 3 continued*) In our running example, the rounded squares in Fig. 1 represent queryables: the set of queryables $\mathscr{Q} = \{sm, \neg sm, sp, \neg sp, rp, \neg rp, rm, \neg rm, u, \neg u, s, \neg s, t, \neg t, b, \neg b\}$.

By querying the client agent, we could obtain more information; $F_{\mathscr{Q}}(AT)$ for example contains an argumentation theory with knowledge base $\mathscr{K}' = \mathscr{K} \cup \{\neg sp\} = \{sm, b, \neg rp, u, t, \neg sp\}$. However, adding information does not influence f's justification status: for each AT' in $F_{\mathscr{Q}}(AT)$, f is blocked in AT'. Therefore, f is stable-blocked in AT w.r.t. $\mathscr{Q}$.

3.2. A naive algorithm

The first solution that may come to one's mind in order to detect if a literal is stable in some argumentation theory w.r.t. the set of queryables could be an algorithm that (1) generates all future argumentation theories for the given argumentation theory; (2) computes the current justification status of the literal and stores this as a justification label; and (3) assigns a stability label based on the justification labels for the future argumentation theories. Such an algorithm, let us call it STABILITY-

[1] See e.g. Bex and Verheij (2013) for the interplay between facts and law in a formal argumentation setting.

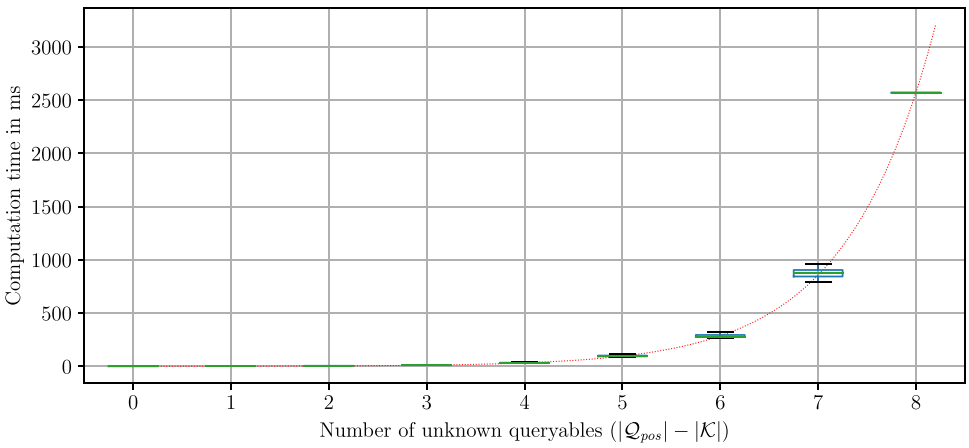


Fig. 2. Computation time of STABILITY-NAIVE, applied to the 3^8 argumentation theories that are in $F_{\mathscr{C}}((AS,\emptyset))$ for the literal f, where AS and $\mathscr{C}$ are the argumentation system and set of queryables from our running example in Fig. 1. The results are shown as boxplots grouped by the number of unknown queryables. The red dotted line is the formula $g(x) = 0.392{\cdot}x^3$ and indicates that the computation time grows exponentially with the number of unknown queryables. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

NAIVE,[2] is sound and complete, but also exponential: assuming that the contradiction function of the argumentation system is instantiated as classical negation, each argumentation theory $AT = (AS, \mathscr{K})$ has $3^{\frac{1}{2}|\mathscr{C}|-|\mathscr{K}|}$ future argumentation theories. This is highly problematic for the runtime, as we demonstrate with an experiment on the running example on online trade fraud (Fig. 1). Recall that this argumentation system has 24 literals (including negation), 16 of which are queryable, and 8 rules. For this experiment, we created a data set of all $3^8 = 6561$ argumentation theories that are in $F_{\mathscr{C}}((AS,\emptyset))$. For each of these argumentation theories, we executed STABILITY-NAIVE and measured the computation time.[3] The results are shown as boxplots grouped by the number of unknown queryables (i.e. $|\mathscr{C}_{pos}| - |\mathscr{K}|$ where $\mathscr{C}_{\text{pos}}$ is the set of all non-negated queryables) in Fig. 2.

From the figure it becomes clear that even for a simple argumentation system such as our toy example, the exponential algorithm STABILITY-NAIVE cannot be used for real-time inquiry, given that the computation would take multiple seconds for each step in the inquiry dialogue. For the argumentation theory in which the knowledge base is empty, so eight queryables are unknown, the computation time is over 2500 ms. To explicate the exponential computation time of STABILITY-NAIVE, we plotted the line $0.392{\cdot}3^{|\mathscr{C}_{pos}|-|\mathscr{K}|}$, which fits nicely with the measured computation times, in red. Assuming that this formula can be used to extrapolate the computation time to inputs with a larger number of unknown queryables, the expected computation time would be 1 h and 33 min in an application where $|\mathscr{C}_{pos}| - |\mathscr{K}| = 15$; if $|\mathscr{C}_{pos}| - |\mathscr{K}| = 20$, it would take over two weeks. As will become clear in Section 5 on case studies, this number of queryables is common for realistic applications; for example, the argumentation system that we use for actual trade fraud intake has a $\mathscr{C}_{pos}$ of 15 positive queryables.

Note that we do not claim that each sound and complete algorithm for computing stability is as slow as STABILITY-NAIVE. We can imagine several improvements on the algorithm, such as only considering those unknown queryables that influence a particular literal for which we want to know the stability status (see Alfano et al., 2021 for a comparable approach on a related problem) and/or more efficient searching by SAT-based approaches (similar to Niskanen and Järvisalo, 2020). However, algorithms based on these ideas do not yet exist for ASPIC$^+$ and implementing these improvements is beyond the scope of this paper.

[2] Defining the algorithm would be outside the scope of this paper, but we provide the source code in our GitHub repository at https://github.com/DaphneO/StabilityLabelAlgorithm.

[3] All experiments in this paper (including the ones in Sections 4.2.5 and 5.1.3) were run on a Intel(R) Core(TM) i7-7820HQ CPU 2.90 GHz 16 GB RAM machine.

Moreover, speeding up the computation this way does not solve the issue of exponentially growing computation time. In the next section, we will show that the problem of deciding if a literal is stable is CoNP-complete, which means that it is unlikely that an exact polynomial-time algorithm exists.

3.3. The complexity of the stability problem

In this section, we discuss the complexity of the stability problem. Consider an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and $\mathscr{C}$ is the set of queryable literals; we define STABILITY as the problem of deciding if a literal $l \in \mathscr{L}$ is stable in AT w.r.t. $\mathscr{C}$.

Proposition 1. (*Complexity of STABILITY problem*) *Given an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and $\mathscr{C}$ is a set of queryables, the STABILITY problem is CoNP-complete.*

Proof sketch. CoNP-hardness can be shown by a polynomial-time reduction from the CoNP-complete problem UNSAT. Given a CNF formula $\phi = (l_{11} \vee \ldots \vee l_{1k}) \wedge \ldots \wedge (l_{n1} \vee \ldots \vee l_{nm})$, we could transform ϕ into the argumentation theory $AT = (AS, \mathscr{K})$ and the set of queryables $\mathscr{C}$ shown in Fig. 3. In this reduction, ϕ is unsatisfiable iff t is stable-unsatisfiable in AT w.r.t. $\mathscr{C}$.

Furthermore, STABILITY is in CoNP: a certificate would be an argumentation theory $AT' \in F_{\mathscr{C}}(AT)$ such that the justification status of t in AT' differs from t's justification status in AT. We can check in polynomial time if $AT' \in F_{\mathscr{C}}(AT)$; furthermore, the justification status of a literal in a given argumentation theory can be checked in polynomial time.□

The full proof can be found in Appendix B.1.

3.4. Handling complexity with approximation algorithms

In the previous section, we have shown that the STABILITY problem is CoNP-complete. Under the assumption that P ≠ NP, problems in this

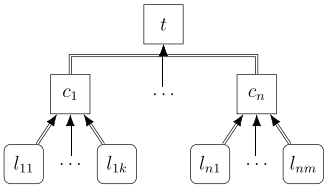


Fig. 3. Reduction UNSAT.

complexity class are considered intractable: each sound and complete algorithm for computing stability has some exponential component that will eventually result in intolerably high runtimes for some inputs. This means that an exact algorithm for stability would need exponential time. Practical applications, such as our inquiry agents at the police, require fast computation for arbitrary argumentation theories. Common approaches to deal with hard problems in argumentation are either exact algorithms based on SAT-solvers or approximation algorithms. SAT-based approaches, such as Niskanen and Järvisalo (2020), are exact (sound and complete) and can solve many instances efficiently, although the worst-case time complexity is still exponential. An alternative to sound and complete algorithms is to use an approximation that is sound, complete or neither. Popular approaches in this area are to apply data-driven techniques or devise direct approximation algorithms. Data-driven techniques such as neural networks have the benefit of predictable computational cost but provide no soundness guarantee (see e.g. Craandijk and Bex (2020)). For the police this is problematic; often it is better to accidentally collect too much information than to make a wrong decision. We therefore prefer direct approximation algorithms over data-driven techniques.

Our contribution is to specify a polynomial algorithm for stability approximation that is sound, but not complete. This way, we can guarantee fast computation for arbitrary argumentation theories and ensure that more information cannot change the justification status of any literal that is labelled stable.

4. Approximating stability

In this section, we propose our algorithm for approximating stability. Subsequently, we show soundness and conditional completeness and study the computational complexity of this algorithm.

4.1. Stability approximation algorithm

Our algorithm for approximating stability iteratively constructs a labelling that assigns a label to each literal and rule. Before explaining how the algorithm achieves such a labelling, we will first illustrate how such a labelling should look like by showing the correct labelling for an example argumentation theory in Section 4.1.1. Having observed some properties of this labelling, we subsequently explain our proposed procedure to approximate this labelling in Sections 4.1.2 and 4.1.3.

4.1.1. Desired stability labelling

In order to decide if a literal is stable, our algorithm uses a labelling L that assigns a quadruple of four booleans $\langle u, d, o, b\rangle$ (i.e. a label) to each literal and rule. Each of these booleans corresponds to a justification status. Intuitively, the truth value of a boolean belonging to a literal represents the possibility that this literal can still become stable-unsatisfiable (u), stable-defended (d), stable-out (o) or stable-blocked (b) in a future argumentation theory. Rules are assigned a label as well, because this helps in efficiently computing the labels of their conclusion literals. The label assigned to a given rule aggregates information from labels assigned to its antecedents. If only one of the booleans in a label of some literal or rule is True, then that literal or rule is labelled stable, as formally defined next.

Definition 12. (*Labelled stable*) Given a label $L[x]$ for some literal or rule $x \in \mathscr{L} \cup \mathscr{R}$:

- x is **labelled stable-unsatisfiable** by L iff $L[x] = \langle 1,0,0,0\rangle$;
- x is **labelled stable-defended** by L iff $L[x] = \langle 0,1,0,0\rangle$;
- x is **labelled stable-out** by L iff $L[x] = \langle 0,0,1,0\rangle$;
- x is **labelled stable-blocked** by L iff $L[x] = \langle 0,0,0,1\rangle$.

A literal or rule $x \in \mathscr{L} \cup \mathscr{R}$ is **labelled stable** by L iff any of the above applies.

In Example 5, we will clarify which labelling our algorithm tries to obtain by showing the correct labels for an example argumentation theory. Note that this example does not yet describe the labelling procedure, but rather motivates the literal labels that it tries to achieve.

Example 5. (*Stability labelling*) In Fig. 4, we give an example of an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and $\langle u, d, o, b\rangle$ is the stability labelling that *should* be found by a sound and complete stability labelling algorithm. The set of queryables $\mathscr{Q}$ is $\{q_1,\ldots,q_5\} \cup \{\neg q_1,\ldots,\neg q_5\}$. From these queryables, q_2, q_3 and $\neg q_4$ are observed in AT (i.e., in $\mathscr{K}$), while q_1, $\neg q_1$, q_5 and $\neg q_5$ can still be observed in a future argumentation theory $AT' \in F_{\mathscr{Q}}(AT)$.

Some literals are already stable:

- each literal l in $\{\neg q_2, \neg q_3, x, \neg x, y, \neg y, z, \neg z\}$ is stable-unsatisfiable (label: $\langle 1,0,0,0\rangle$) because there is no rule for l and either $l \notin \mathscr{Q}$ or there is some contradictory $l' \in \bar{l}$ such that $l' \in \mathscr{K}$;
- q_2, q_3, $\neg q_4$ and s are stable-defended (label: $\langle 0,1,0,0\rangle$) because there is an argument for these literals that cannot be attacked; and
- t and $\neg t$ are stable-blocked (label: $\langle 0,0,0,1\rangle$) because the arguments for these literals attack each other, but neither is in the grounded extension.

Other literals are not yet stable, but can become stable in a future argumentation theory. Consider for example the argumentation theory $AT' = (AS, \mathscr{K} \cup \{q_1\})$ in $F_{\mathscr{Q}}(AT)$. In AT', the literal q_1 would be stable-defended; $\neg q_1$ would be stable-unsatisfiable and both $\neg p$ and p become stable-blocked in AT' w.r.t. $\mathscr{Q}$. An alternative argumentation theory in $F_{\mathscr{Q}}(AT)$ is $AT'' = (AS, \mathscr{K} \cup \{\neg q_1\})$, where $\neg q_1$ and p become stable-defended, while q_1 and $\neg p$ are stable-unsatisfiable in AT'' w.r.t. $\mathscr{Q}$.

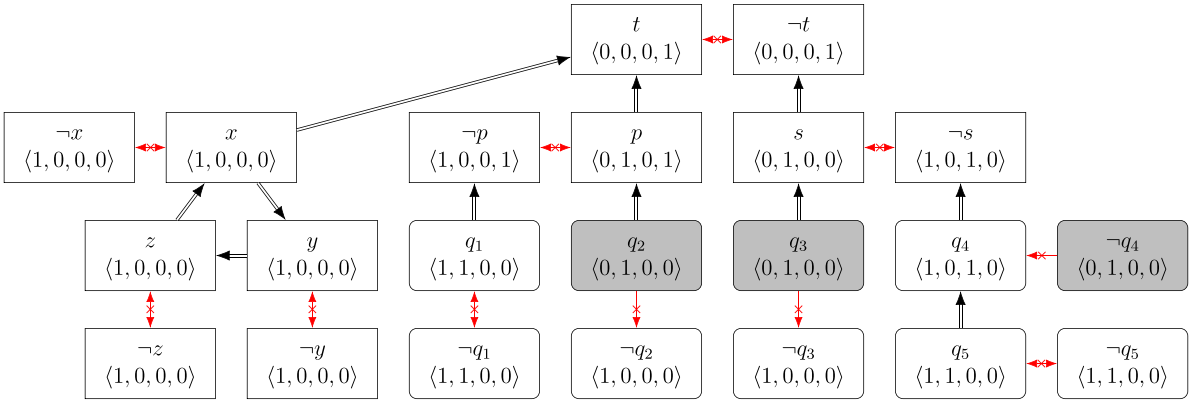


Fig. 4. Argumentation theory illustrating the ground-truth $\langle u, d, o, b\rangle$-labelling of literals.

procedure PREPROCESS($\mathcal{L}, \mathcal{R}, ^-, \mathcal{Q}, \mathcal{K}$)
  for Literal l in $\mathcal{L}$ **do**
    if $l \in \mathcal{Q}$ and for each $l' \in \bar{l}$: $l' \notin \mathcal{K}$ **then** $L[l] = \langle 1,1,1,1 \rangle$
    else $L[l] = \langle 1,0,0,0 \rangle$
  for Rule r in $\mathcal{R}$ **do**
    $L[r] = \langle 1,0,0,0 \rangle$
  Change = True
  while *Change* **do**
    Change = False
    for Rule r in $\mathcal{R}$ **do**
      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**
        $L[r] = \langle 1,1,1,1 \rangle$
        $L[\texttt{cons}(r)] = \langle 1,1,1,1 \rangle$
        Change = True
  return L

Algorithm 1. Preprocessing step.

Similarly, adding either q_5 or $\neg q_5$ would make each literal in $\{q_5, \neg q_5, q_4, \neg s\}$ stable.

Finally, recall from Section 2.3 that the distinction between the *out* and *blocked* justification status is convenient for our algorithm. This can be seen by considering the difference in statuses for p and s. Both literals have a contradictory that cannot become stable-defended in any future argumentation theory ($\neg p$ and $\neg s$, respectively). However, $\neg s$ cannot be stable-blocked in any future argumentation theory (given that each argument for $\neg s$ is attacked by the observation-based argument for $\neg q_4$), which means that s is stable-defended. On the other hand, $\neg p$ can still become stable-blocked (by adding q_1 to the knowledge base), so p is not stable-defended but can still become stable-blocked.

Based on the example above, we can make three general remarks about the desired labelling:

Remark 1. The labels of literals depend on the labels of antecedents of rules for those literals, as well as of antecedents of rules for contradictories of those literals. For example, the status of t depends on the status of p and the status of s. In addition, the labels of queryable literals also depend on their presence (or one of their contradictories' presence) in the knowledge base: the status of q_4 depends on the status of q_5, but also on the presence of $\neg q_4$ in the knowledge base.

Remark 2. It is not always required that each of these antecedents is stable for the conclusion literal to be stable. For example: t and $\neg t$ are stable although p is not stable. s is stable although q_4 is not stable.

Remark 3. The rule set of an argumentation theory can contain cycles of support relations, such as the cycle $x \Rightarrow y$, $y \Rightarrow z$, $z \Rightarrow x$ in the example.

We take these three remarks into account in our proposed labelling procedure. By Remark 1, the labels of rules depend on the labels of their antecedent literals, while labels of literals depend on the labels of rules for that literal or for one of its contradictories. Literals and rules are labelled bottom-up, 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. By Remark 2, the information contained in the labelling should be more precise than a single label indicating if the literal is stable or not. For example, if we only had one label for literals that are not stable, it would not be possible to label s in Example 5 as stable, since it depends on the literal q_4 that would just be labelled as not stable. Therefore, we propose to assign quadruple labels $\langle u, d, o, b \rangle$ to each literal and rule. By Remark 3, it is necessary to correctly handle cycles of support relations. We deal with this in a preprocessing step.

In the following two subsections, we will describe the algorithm in two steps: a preprocessing step and the main labelling procedure.

4.1.2. Preprocessing

Our algorithm starts with a preprocessing step. This enables the algorithm to properly deal with argumentation theories containing *support cycles*, i.e. cycles of inference relations based on which no argument can be constructed. For example, consider the inference relation between the literals x, y and z in Fig. 4. None of these literals are queryable, so there is no future argumentation theory in which there is an observation-based argument for any of them. Since there are no other rules for any of these literals, other than the three rules $x \Rightarrow y$, $y \Rightarrow z$ and $z \Rightarrow x$ that form a cycle, these three rules form a support cycle. Support cycles can be problematic for defeasible reasoning algorithms, when not handled properly. In Odekerken et al. (2020), we showed that an initial version of our algorithm for stability, proposed in Testerink et al. (2019b), does not label literals and rules that are dependent on support cycles: in a bottom-up labelling procedure, there is no place to start labelling.

The preprocessing step is specified in Algorithm 1. The idea of this algorithm is that initially, all literals that cannot be in the knowledge base in a future argumentation theory and all rules are labelled $\langle 1,0,0,0 \rangle$ (i.e. stable-unsatisfiable). Then, the algorithm incrementally labels those rules for which all antecedents are not labelled $\langle 1,0,0,0 \rangle$ and their consequents as $\langle 1,1,1,1 \rangle$ (i.e. may still become unsatisfiable, defended, out or blocked) based on the intuition that there may be an argument based on these rules in a future argumentation theory.

Example 6. (*Preprocessing step*) We return to the argumentation theory $AT = (AS, \mathcal{K})$ where $AS = (\mathcal{L}, \mathcal{R}, \bar{})$, with queryables $\mathcal{Q}$, from Example 5 and Fig. 4.

Let L_p = PREPROCESS($\mathcal{L}, \mathcal{R}, \mathcal{Q}, \mathcal{K}$). The labelling L_p is illustrated in Fig. 5. The literals in $\{\neg q_2, \neg q_3, x, \neg x, y, \neg y, z, \neg z\}$ are labelled $\langle 1,0,0,0 \rangle$ by L_p. For each other literal l in $\mathcal{L}$: $L_p[l] = \langle 1,1,1,1 \rangle$. Further note that the rules in and from the support cycle, i.e. $x \Rightarrow y$, $y \Rightarrow z$, $z \Rightarrow x$ and $x \Rightarrow t$, are

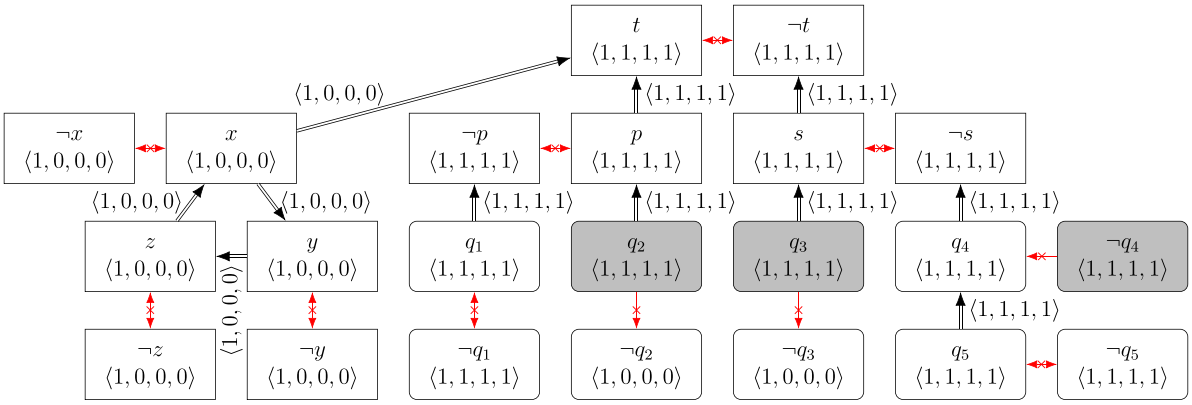


Fig. 5. Argumentation theory illustrating the $\langle u, d, o, b\rangle$-labelling of literals and rules after the preprocessing step.

labelled $\langle 1,0,0,0\rangle$. The other rules are labelled $\langle 1,1,1,1\rangle$ by L_p.

4.1.3. Quadruple labelling procedure

The result of the preprocessing procedure is an initial labelling L_p for each of the literals and rules in our argumentation system. After preprocessing, we apply a bottom-up labelling procedure that updates the quadruple of four booleans $\langle u, d, o, b\rangle$ for each literal and rule, resulting in the final labelling L. Algorithm 4 specifies how literals and rules are visited in the labelling procedure, where literals and rules are labelled according to the labelling rules specified in Algorithms 2 and 3. We will discuss both the labelling rules and the labelling procedure, starting with the labelling rules.

In Section 4.1.1 we observed that there are cases in which a literal should be labelled stable, although it is dependent on literals that are not yet labelled stable (2). Because of this, a labelling that assigns a single label (e.g. stable-unsatisfiable / stable-defended / stable-out / stable-blocked / unstable) to literals and rules does not suffice. This is one of the reasons why our initial algorithm for stability (Testerink et al., 2019b), which assigns a single label, detects fewer stable situations than our more recent proposal in Odekerken et al. (2020), which assigns a quadruple label $\langle u, d, o, b\rangle$ consisting of four booleans to each label and rule. The labelling rules in these algorithms require some additional notation for indexing single booleans of labels:

Notation 1. (*Indexing of label parts*) Given a label $L[x] = \langle u, d, o, b\rangle$ for some literal or rule $x \in \mathscr{L} \cup \mathscr{R}$, we refer to the u- (resp. d-, o-, b-) boolean

procedure RELABEL-LITERAL($\mathcal{L}, \mathcal{R}, {}^-, \mathcal{Q}, \mathcal{K}, L, l$)
  ▷ ***Labelling rules turning*** $L[l].u$ **to False**
  if $l \in \mathcal{K}$ **then** $L[l].u$ = False ▷ L-U-a
  if there is a rule r for l with $\neg L[r].u$ **then** $L[l].u$ = False ▷ L-U-b

  ▷ ***Labelling rules turning*** $L[l].d$ **to False**
  if some $l' \in \bar{l}$ is in $\mathcal{K}$ **then** $L[l].d$ = False ▷ L-D-a
  if $l \notin \mathcal{Q}$ **then**
    if for each rule r for l: $\neg L[r].d$ **then** $L[l].d$ = False ▷ L-D-b
    if there is some $l' \in \bar{l}$ for which there is a rule r' with $\neg L[r'].u$ and $\neg L[r'].o$ **then** $L[l].d$ = False ▷ L-D-c

  ▷ ***Labelling rules turning*** $L[l].o$ **to False**
  if $l \in \mathcal{K}$ **then** $L[l].o$ = False ▷ L-O-a
  if $l \in \mathcal{Q}$ and for each $l' \in \bar{l}$, some $l'' \in \bar{l}'$ is in $\mathcal{K}$ **then**
    if for each rule r for l: $\neg L[r].o$ **then** $L[l].o$ = False ▷ L-O-b
    if there is a rule r for l with $\neg L[r].u$ and $\neg L[r].o$ **then** $L[l].o$ = False ▷ L-O-c
  if $l \notin \mathcal{Q}$ **then**
    if for each rule r for l: $\neg L[r].o$ **then** $L[l].o$ = False ▷ L-O-d
    if there is a rule r for l with $\neg L[r].u$ and $\neg L[r].o$ **then** $L[l].o$ = False ▷ L-O-e
  if for each rule r for l: $\neg L[r].d$ and $\neg L[r].o$ and $\neg L[r].b$ **then** $L[l].o$ = False ▷ L-O-f

  ▷ ***Labelling rules turning*** $L[l].b$ **to False**
  if $l \in \mathcal{Q}$ **then** $L[l].b$ = False ▷ L-B-a
  if for each rule r for l: $\neg L[r].d$ and $\neg L[r].b$ **then** $L[l].b$ = False ▷ L-B-b
  if for each $l' \in \bar{l}$: for each rule r' for l': $\neg L[r'].d$ and $\neg L[r'].b$ **then**
    if for each rule r for l: $\neg L[r].b$ **then** $L[l].b$ = False ▷ L-B-c
    if there is a rule r for l with $\neg L[r].u$ and $\neg L[r].o$ and $\neg L[r].b$ **then** $L[l].b$ = False ▷ L-B-d
  return L

Algorithm 2. RELABEL-LITERAL procedure.

1: **procedure** RELABEL-RULE($\mathcal{L}, \mathcal{R}, L, r$)
2: **if** for each antecedent l of r: $\neg L[l].u$ **then** $L[r].u$ = False ▷ R-U-a
3: **if** there is an antecedent l of r with $\neg L[l].d$ **then** $L[r].d$ = False ▷ R-D-a
4: **if** for each antecedent l of r: $\neg L[l].o$ **then** $L[r].o$ = False ▷ R-O-a
5: **if** for each antecedent l of r: $\neg L[l].b$ **then** $L[r].b$ = False ▷ R-B-a
6: **if** there is an antecedent l of r with $\neg L[l].d$ and $\neg L[l].b$ **then** $L[r].b$ = False ▷ R-B-b
7: **return** L

Algorithm 3. RELABEL-RULE procedure.

of $L[x]$ with $L[x].u$ (resp. $L[x].d, L[x].o, L[x].b$).

Our labelling rules are an extension of the labelling rules in Odekerken et al. (2020) in the sense that we account for a general contradiction function rather than classical negation. In RELABEL-LITERAL (Algorithm 2) and RELABEL-RULE (Algorithm 3) we specify how the labels of literals and rules are updated. Note that each of these labelling rules is able to turn a boolean (u, d, o or b) to False, but none of them is able to turn a boolean to True.

Example 7. (*Quadruple labelling rules*) We return to the argumentation theory from Fig. 4 and Examples 5 and 6. Some labelling rules for literals in RELABEL-LITERAL (Algorithm 2) are not dependent on rules for (a contradictory of) that literal. For example, case L-B-a already labels a literal $l \in \mathscr{L}$ as $\neg L[l].b$ if $l \in \mathscr{C}$. This labelling rule applies for the literals in $\{q_1, \ldots, q_5\} \cup \{\neg q_1, \ldots, \neg q_5\}$, because they are queryable. Case L-U-a and L-O-a apply if a literal is in the knowledge base $\mathscr{K}$. Thanks to these labelling rules, the literals $l \in \{q_2, q_3, \neg q_4\}$ in the example will be labelled $\neg L[l].u$ and $\neg L[l].o$. Finally, case L-D-a does not depend on rules either: for a given literal $l \in \mathscr{L}$, it applies if there is some $l' \in \bar{l}$ such that $l' \in \mathscr{K}$. This case applies for the literals $\neg q_2$, $\neg q_3$ and q_4. Note that the absence of rules for a literal is informative for the justification status as well: for example, the literal q_1 is labelled $\neg L[q_1].o$ by case L-O-f because there is no rule for q_1 in $\mathscr{R}$.

Other labels are based on the rules for (a contradictory of) a literal and propagate properties of (attacks on) subarguments. For example, the rule $q_3 \Rightarrow s$ is labelled $\langle 0, 1, 0, 0\rangle$ by RELABEL-RULE's cases R-U-a, R-O-a and R-B-a (and the fact that its only antecedent q_3 is labelled $\langle 0, 1, 0, 0\rangle$). Similarly, the rule $q_4 \Rightarrow \neg s$ is labelled $\langle 1, 0, 1, 0\rangle$ by the cases R-D-a and R-B-a (or R-B-b). Note that this rule cannot be labelled stable. Still, although the literal s is dependent on a rule that is not labelled stable, we can label s as stable: $L[s] = \langle 0, 1, 0, 0\rangle$ by RELABEL-LITERAL's cases L-U-b, L-O-d and L-B-c (or L-B-d). Other literals, such as p, cannot be labelled stable, but still we can exclude some stability statuses: p can never be stable-unsatisfiable (case L-U-b) or stable-out (case L-O-d) w.r.t. $\mathscr{C}$ in any future argumentation theory.

Finally, we discuss the algorithm that visits the literals and rules, repeatedly applying Algorithms 2 and 3. As identified in Remark 1 in Section 4.1.1, the labels of literals depend on the antecedents of rules for those literals and for contradictories of those literals; labels of queryable literals also depend on the knowledge base. Therefore, STABILITY-LABEL (Algorithm 4) starts by labelling those literals that are queryable or for which there is no rule, see lines 4–6. Then, literals and rules are labelled incrementally: after considering a literal for the first time (line 6) or changing a literal (line 13 and 16), the label of each rule that has this literal as an antecedent is added to the set TODO-SET and therefore at some point considered for relabelling. The algorithm ends when the TODO-SET is empty (line 7): at this moment, the labelling has reached a fixed point and cannot change any more. We give some intuition by labelling the example from the beginning of this section.

Example 8. (*Labelling step*) We return to the argumentation theory from Fig. 4 and Examples 5–7 and show how literals and rules are considered for (re)labelling by Algorithm 4.

First, note that the literals x, y and z are never reconsidered for relabelling after the preprocessing step, since they are not queryable and there is no rule for those literals that can change its label. Therefore, the labels in L assigned to these literals equal the labels in L_p, that is: $\langle 1, 0, 0, 0\rangle$ (as shown in Example 6).

All queryables in Q, as well as the literals $\neg x$, $\neg y$ and $\neg z$, for which there is no rule, are considered for relabelling in Algorithm 4 line 5; see Fig. 6. For the queryable literals, RELABEL-LITERAL case L-B-a applies. As a result, the b-boolean is turned to False: there is no future argumentation

1: **procedure** STABILITY-LABEL($\mathcal{L}, \mathcal{R}, ^-, Q, \mathcal{K}$)
2: L = PREPROCESS($\mathcal{L}, \mathcal{R}, ^-, Q, \mathcal{K}$)
3: TODO-SET = empty set
4: **for** Literal l in $\mathcal{L}$ such that $l \in Q$ or there is no rule for l in $\mathcal{R}$ **do**
5: L = RELABEL-LITERAL($\mathcal{L}, \mathcal{R}, ^-, Q, \mathcal{K}, L, l$)
6: Add all rules having l as antecedent to TODO-SET
7: **while** TODO-SET is not empty **do**
8: Pop a rule r from TODO-SET
9: L = RELABEL-RULE($\mathcal{L}, \mathcal{R}, L, r$)
10: **if** r's label changed **then**
11: L = RELABEL-LITERAL($\mathcal{L}, \mathcal{R}, ^-, Q, \mathcal{K}, L, \mathtt{cons}(r)$)
12: **if** $\mathtt{cons}(r)$'s label changed **then**
13: Add all rules having $\mathtt{cons}(r)$ as antecedent to TODO-SET
14: **for** $l' \in \mathtt{cons}(r)$ **do**
15: L = RELABEL-LITERAL($\mathcal{L}, \mathcal{R}, ^-, Q, \mathcal{K}, L, l'$)
16: **if** l''s label changed **then**
17: Add all rules having l' as antecedent to TODO-SET
18: **return** L

Algorithm 4. Labelling procedure STABILITY-LABEL.

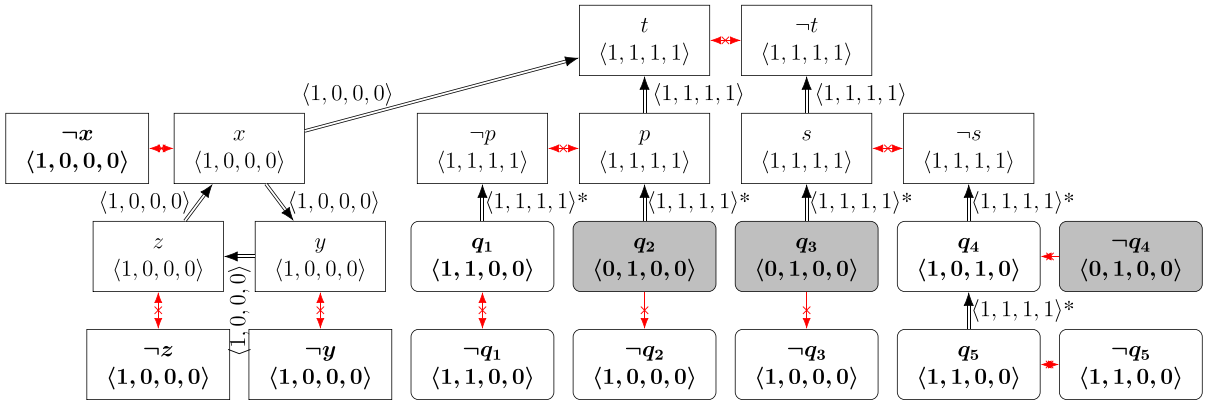


Fig. 6. Argumentation theory illustrating the $\langle u, d, o, b\rangle$-labelling of Stability-Label between lines 6 and 7. All literals that have been considered for relabelling by Relabel-Literal are illustrated in boldface. The rules in todo-set are indicated with an asterisk.

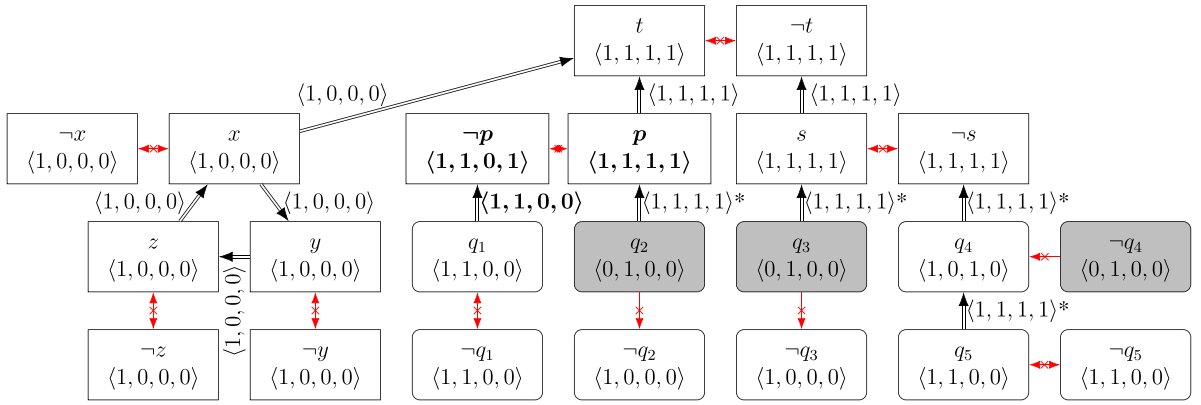


Fig. 7. Argumentation theory illustrating the $\langle u, d, o, b\rangle$-labelling of Stability-Label after the first iteration of the while-loop, where the rule $q_1 \Rightarrow \neg p$ has been selected from todo-set. All literals that have been considered for relabelling by Relabel-Literal are illustrated in boldface. The rules in todo-set are indicated with an asterisk.

theory AT' such that queryable literals are stable-blocked in AT' w.r.t. $\mathscr{E}$. Furthermore, q_2, q_3 and $\neg q_4$ are in the knowledge base. Therefore, the u- and o-booleans for these literals are turned to False by case L-U-a and L-O-a. For $\neg q_2$, $\neg q_3$ and q_4, case L-D-a applies, since they have a contradictory in the knowledge base. As a result, the d-boolean is turned to False. Finally, the o-booleans of q_1, $\neg q_1$, $\neg q_2$, $\neg q_3$, q_5 and $\neg q_5$ are turned to False by case L-O-f.

After relabelling of aforementioned literals, the rules $q_1 \Rightarrow \neg p$, $q_2 \Rightarrow p$, $q_3 \Rightarrow s$, $q_5 \Rightarrow q_4$ and $q_4 \Rightarrow \neg s$ are added to todo-set. Suppose that the rule $q_1 \Rightarrow \neg p$ is selected first from todo-set (this order is arbitrary). This rule is labelled $\langle 1, 1, 0, 0\rangle$ by case R-O-a and R-B-a, as illustrated in Fig. 7. Since this rule's label changed, the literals $\neg p$ and p are considered for relabelling by Relabel-Literal as well. Then the label of $\neg p$ changes into $\langle 1, 1, 0, 1\rangle$ (by case L-O-d). The label of p does not change, so the rule $p \Rightarrow t$ is not yet added to todo-set.

In the next iteration of the while loop, the rule $q_2 \Rightarrow p$ is selected from todo-set and relabelled as $\langle 0, 1, 0, 0\rangle$ by case R-U-a, R-O-a and R-B-a of Relabel-Rule; see Fig. 8. After this, both p and $\neg p$ are reconsidered for relabelling. Since the label of p changes, the rule $p \Rightarrow t$ is added to todo-set.

In additional iterations of the while loop the rules $q_3 \Rightarrow s$, $q_4 \Rightarrow \neg s$ and $q_5 \Rightarrow q_4$ are selected from todo-set, so that these rules, their consequents

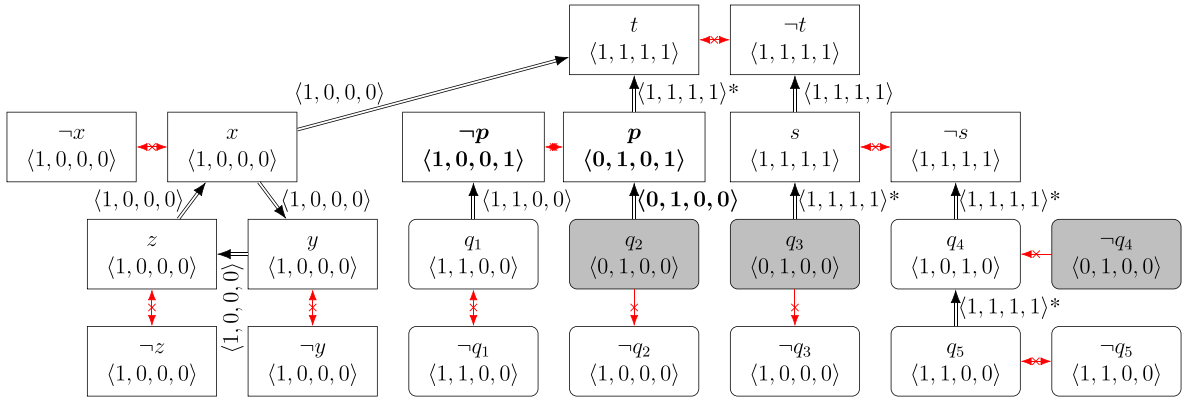


Fig. 8. Argumentation theory illustrating the $\langle u, d, o, b\rangle$-labelling of Stability-Label after the second iteration of the while-loop, where the rule $q_2 \Rightarrow p$ has been selected from todo-set. All literals that have been considered for relabelling by Relabel-Literal are illustrated in boldface. The rules in todo-set are indicated with an asterisk.

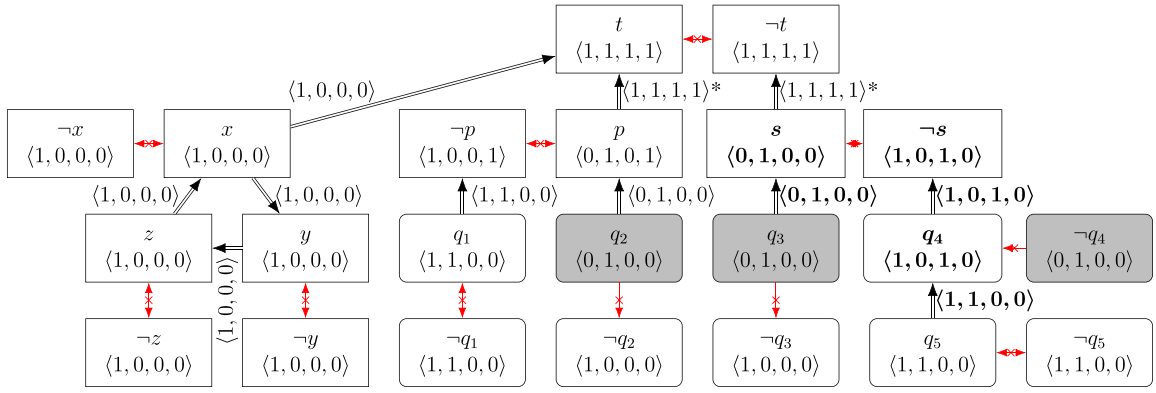


Fig. 9. Argumentation theory illustrating the $\langle u,d,o,b\rangle$-labelling of Stability-Label after the fifth iteration of the while-loop, where the rules $q_3 \Rightarrow s$, $q_5 \Rightarrow q_4$ and $q_4 \Rightarrow \neg s$ have been selected from todo-set. All literals that have been considered for relabelling by Relabel-Literal are illustrated in boldface. The rules in todo-set are indicated with an asterisk.

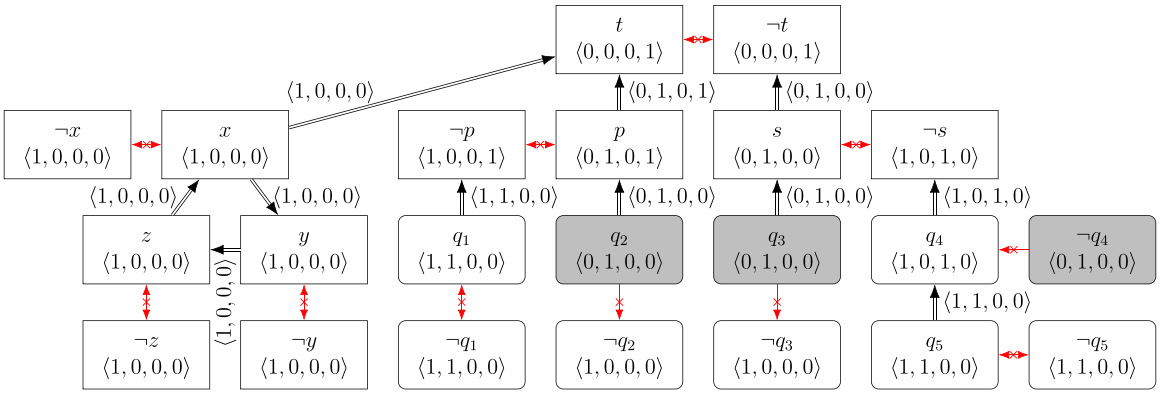


Fig. 10. Argumentation theory illustrating the $\langle u,d,o,b\rangle$-labelling of Stability-Label after the final iteration of the while-loop, where the rules $p \Rightarrow t$ and $s \Rightarrow \neg t$ have been selected from todo-set. Note that the resulting labels for literals correspond to the desired labels as illustrated in Fig. 4.

and the contradictories of those contradictories can be relabelled. The result after this iteration is shown in Fig. 9.

Subsequently, the rules $p \Rightarrow t$ and $s \Rightarrow \neg t$ are considered for relabelling by Relabel-Rule. For both rules, the u-booleans and o-booleans are turned to False by case R-U-a and R-O-a, respectively. Furthermore, $s \Rightarrow \neg t$ is labelled $\neg L[s \Rightarrow \neg t].b$ by case R-B-a. Finally, t and $\neg t$ are relabelled: $L[t] = L[\neg t] = \langle 0,0,0,1\rangle$ by case L-U-b, L-D-c, L-O-d. Note that the resulting labelling of literals, illustrated in Fig. 10, exactly matches the desired stability labelling shown in Fig. 4.

In our running example, we have seen that our algorithm Stability-Label is able to find the accurate stability labelling for a specific argumentation theory. Keep in mind that this is not the case for every possible argumentation theory: Stability-Label is an approximation algorithm, in which we sacrificed perfect accuracy for fast computation. In the next subsection, we will prove properties of the proposed algorithm: we will show that it is sound, identify conditions under which it is complete and prove that it runs in polynomial time.

4.2. Properties of the proposed algorithm

In this subsection, we present various properties of Stability-Label. We start by conducting an accuracy analysis experiment on the argumentation system of our running example on online trade fraud in Section 4.2.1. We subsequently consider Stability-Label's soundness in Section 4.2.2, discuss conditional completeness in Section 4.2.3 and finally analyze the algorithm's time complexity and computation time[4] in Sections 4.2.4 and 4.2.5.

As before, the proofs of the lemmas and propositions in this section can be found in Appendix B. Many of these lemmas depend on additional lemmas in the appendix that were too specific to include in the paper, but may be interesting for the reader who wants to know more about the properties of Stability-Label. An example is Lemma 9, which guarantees that no literal or rule will have the label $\langle 0,0,0,0\rangle$.

4.2.1. Experimental accuracy analysis

The performance of Stability-Label in terms of accuracy can be assessed empirically for a given argumentation system $AS = (\mathscr{L}, \mathscr{R}, \_)$, topic literal $l \in \mathscr{L}$ and set of queryables $\mathscr{Q}$ as follows: first, compute the accuracy of stability-label by generating all future argumentation theories $F_{\mathscr{Q}}((AS,\varnothing))$; then verify for each $AT' \in F_{\mathscr{Q}}((AS,\varnothing))$ if the stability label estimated by Stability-Label matches the ground truth stability label, that can be obtained by for example the stability-naive algorithm from Section 3.2.

The confusion matrix in Table 1 shows the performance of the stability algorithm on the toy example on fraud of Fig. 1, where the topic literal is *fraud*. For each of the 6551 possible argumentation theories that can be constructed from the argumentation system and set of queryables, the label obtained by Stability-Label is compared to the ground truth label.

The accuracy of Stability-Label for a given argumentation system and set of queryables can be computed by computing the fraction of future argumentation theories in $F_{\mathscr{Q}}((AS,\varnothing))$ in which the topic literal is correctly labelled as (not) stable.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{All}} = \frac{3864 + 2265}{6551} = 93.4\%$$

[4] In addition, we discuss empirical results on our case studies in Section 5.

Table 1
Confusion matrix showing the performance of Stability-label on the toy example on fraud of Fig. 1. The topic literal is *fraud*.

		Ground truth						
	$\langle 0,0,0,1\rangle$	$\langle 0,1,0,0\rangle$	$\langle 0,1,0,1\rangle$	$\langle 1,0,0,0\rangle$	$\langle 1,0,0,1\rangle$	$\langle 1,1,0,0\rangle$	$\langle 1,1,0,1\rangle$	Total
Stability algorithm								
$\langle 0,0,0,1\rangle$	27	0	0	0	0	0	0	27
$\langle 0,1,0,0\rangle$	0	108	0	0	0	0	0	108
$\langle 0,1,0,1\rangle$	0	0	27	0	0	0	0	27
$\langle 1,0,0,0\rangle$	0	0	0	3729	0	0	0	3729
$\langle 1,0,0,1\rangle$	0	0	0	0	189	0	0	189
$\langle 1,0,1,0\rangle$	0	0	0	144	0	1124	0	1268
$\langle 1,0,1,1\rangle$	0	0	0	288	152	304	469	1213
Total	27	108	27	4161	341	1428	469	6551

Note that the number of false positives is 0, so for each argumentation theory based on the argumentation system: if the literal *fraud* is not stable in that argumentation theory, then it is not labelled as stable by the stability algorithm – or, by contraposition: if *fraud* is labelled as stable by Stability-label, then it is stable. This means that our algorithm is sound for *fraud* in the argumentation theory of Fig. 1. In the next section, we will show that the algorithm is sound in general.

When analysing the errors, it becomes clear that for each of the knowledge bases $\mathscr{K}$ such that the estimated stability label of *fraud* in $(AS,\mathscr{K})$ is incorrect both $b\notin\mathscr{K}$ and $\neg b\notin\mathscr{K}$. So for each argumentation theory $(AS,\mathscr{K})$ such that $b\in\mathscr{K}$ or $\neg b\in\mathscr{K}$, the accuracy is 100%. This means that it would be a good idea to ask b at an early stage of the dialogue. On a more general note: the order of asking questions influences the performance of the stability algorithm in an inquiry dialogue. Furthermore, the performance of the stability algorithm depends on the argumentation system – for example, the accuracy of Stability-label on the argumentation system from Fig. 4 is 100%.

It would be possible to extend our experiments with more argumentation systems, but that would require obtaining the ground truth; at this point we only have the stability-naive algorithm to do that, which for large sets of queryables takes more time than is feasible, as we have argued in Section 3.2. Furthermore, accuracy percentages do not reveal which aspects of an argumentation theory are problematic for stability estimation by Stability-label. Therefore, we extend our empirical analysis with theoretical evidence for the soundness and conditional completeness of our proposed algorithm. In the next sections, we show that Stability-label is sound for every possible argumentation system and identify conditions under which the algorithm is complete. For argumentation systems satisfying these conditions, like that of Fig. 4, Stability-label is sound and complete, hence 100% accurate.

4.2.2. Soundness

In this section, we show that Stability-label is sound: given an argumentation theory AT and a set of queryables $\mathscr{Q}$, if the algorithm labels l as stable in AT w.r.t. $\mathscr{Q}$, then l is stable in AT w.r.t. $\mathscr{Q}$. Soundness is a valuable property in general, but specifically in our application in inquiry dialogue, because it ensures that the inquiry dialogue is only terminated if no additional information can change the conclusion. As a first step, we show that the preprocessing algorithm is sound in the sense that each literal that is labelled $\langle 1,0,0,0\rangle$ is stable-unsatisfiable in AT w.r.t. $\mathscr{Q}$.

Lemma 2. (*Soundness preprocessing step*) *Let $AT=(AS,\mathscr{K})$ be an argumentation theory where $AS=(\mathscr{L},\mathscr{R},\text{-})$ and let $\mathscr{Q}$ be a set of queryables. Furthermore let L_p be the labelling obtained by Preprocess (Algorithm 1) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. For each $l\in\mathscr{L}$: if $L_p[l]=\langle 1,0,0,0\rangle$, then l is stable-unsatisfiable in AT w.r.t. $\mathscr{Q}$.*

Proof sketch. This can be shown by contraposition: if l is not stable-unsatisfiable in AT w.r.t. $\mathscr{Q}$, then there is some $AT'\in F_{\mathscr{Q}}(AT)$ such that there is an argument A for l in $Arg(AT')$. If A is observation-based, then $L_p[l]=\langle 1,1,1,1\rangle$ is assigned in Algorithm 1 line 3; if A is rule-based, then there is some rule r for l such that all $a\in\texttt{ants}(r)$ are labelled $L_p[a]=\langle 1,1,1,1\rangle$, hence the label $L_p[r]=\langle 1,1,1,1\rangle$ is assigned by line 13. In both cases, $L_p[l]\neq\langle 1,0,0,0\rangle$.□

The full proof can be consulted in Appendix B.6. Given that the preprocessing step is sound, we now need to show that the remainder of the Stability-label algorithm is sound as well. Below we give a proof sketch; for the full proof, we refer to Appendix B.7.

Proposition 2. (*Soundness stability labelling*) *Let $AT=(AS,\mathscr{K})$ be an argumentation theory where $AS=(\mathscr{L},\mathscr{R},\text{-})$, let $\mathscr{Q}$ be the set of queryable literals and let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. If L labels a literal $l\in\mathscr{L}$ stable-unsatisfiable in AT w.r.t. $\mathscr{Q}$, then l is stable-unsatisfiable in AT w.r.t. $\mathscr{Q}$; if L labels a literal $l\in\mathscr{L}$ stable-defended in AT w.r.t. $\mathscr{Q}$, then l is stable-defended in AT w.r.t. $\mathscr{Q}$; if L labels a literal $l\in\mathscr{L}$ stable-out in AT w.r.t. $\mathscr{Q}$, then l is stable-out in AT w.r.t. $\mathscr{Q}$; and if L labels a literal $l\in\mathscr{L}$ stable-blocked in AT w.r.t. $\mathscr{Q}$, then l is stable-blocked in AT w.r.t. $\mathscr{Q}$.*

Proof sketch. The following items will be used in the soundness proof sketch and can be shown by induction (cf. Lemmas 8, 11, 12 and 17 in the appendix):

1. For each $r\in\mathscr{R}$ labelled $\neg L[r].d$ and $\neg L[r].b$: for each $AT'\in F_{\mathscr{Q}}(AT)$, each argument based on r is attacked by some argument in $G(AT')$;
2. For each $l\in\mathscr{L}$ labelled $\neg L[l].u$: there is an argument for l in $Arg(AT')$ for each $AT'\in F_{\mathscr{Q}}(AT)$;
3. For each $r\in\mathscr{R}$ labelled $\neg L[r].u$ and $\neg L[r].o$: if $\texttt{cons}(r)\notin\mathscr{Q}$ then for each $AT'\in F_{\mathscr{Q}}(AT)$ there is an argument based on r in $Arg(AT')$ that is not attacked by any argument in $G(AT')$;
4. For each $l\in\mathscr{L}$ labelled $\neg L[l].d$: there is no argument for l in $G(AT')$ for any $AT'\in F_{\mathscr{Q}}(AT)$.

There are four cases in which a literal $l\in\mathscr{L}$ is labelled stable by L (see Lemmas 14, 15, 16 and 18 in the appendix).

- If $\boldsymbol{L[l]=\langle 1,0,0,0\rangle}$ then it was already labelled as such by Preprocess ($L_p[l]=\langle 1,0,0,0\rangle$), because each rule r for l must have been labelled $L[r]=\langle 1,0,0,0\rangle$ and either $l\notin\mathscr{Q}$ or there is an $l'\in\bar{l}$ such that $l'\in\mathscr{K}$. By Lemma 2 $\boldsymbol{l}$ is stable-unsatisfiable in $\boldsymbol{AT}$ w.r.t. $\mathscr{Q}$.
- Suppose $\boldsymbol{L[l]=\langle 0,1,0,0\rangle}$; if l's label was assigned in the $(k+1)$'th iteration then there is a rule r for l that is labelled $L[r]=\langle 0,1,0,0\rangle$ and for each r' for l' where $l'\in\bar{l}$, r' is labelled $\neg L[r'].d$ and $\neg L[r'].b$. Each $a\in\texttt{ants}(r)$ is labelled $\langle 0,1,0,0\rangle$ in or before the k'th iteration; hence by induction each $a\in\texttt{ants}(r)$ is stable-defended in AT w.r.t. $\mathscr{Q}$. Furthermore, by Item 1 above: for each $AT'\in F_{\mathscr{Q}}(AT')$, each rule-based argument for each $l'\in\bar{l}$ is attacked by an argument in $G(AT')$. Consequently, there is an argument for l, based on r, in $G(AT')$ of each $AT'\in F_{\mathscr{Q}}(AT)$. So $\boldsymbol{l}$ is stable-defended in $\boldsymbol{AT}$ w.r.t. $\mathscr{Q}$.
- Suppose $\boldsymbol{L[l]=\langle 0,0,1,0\rangle}$. We distinguish two cases. If $l\in\mathscr{Q}$ then some $l'\in\bar{l}$ is in $\mathscr{K}$. Alternatively, $l\notin\mathscr{Q}$; then all rules r for l are labelled ¬

$L[r].d$ and $\neg L[r].b$. By Item 1 above all arguments based on rules for l must be attacked by an argument in $G(AT')$ for each $AT' \in F_{\mathscr{C}}(AT)$. Furthermore, by Item 2 above there is an argument for l in each $AT' \in F_{\mathscr{C}}(AT)$. To conclude, $\boldsymbol{l}$ is stable-out in $\boldsymbol{AT}$ w.r.t. $\mathcal{Q}$.

- If $\boldsymbol{L}[l] = \langle \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{1} \rangle$ then $l \notin \mathscr{C}$ and there is some rule r for l that is labelled $\neg L[r].u$ and $\neg L[r].o$. By Item 3 above, there is an argument based on r in each $AT' \in F_{\mathscr{C}}(AT)$ that is not attacked by any argument in $G(AT')$. Furthermore, by Item 4 above there is no argument for l in $G(AT')$ for any $AT' \in F_{\mathscr{C}}(AT)$. As a result, $\boldsymbol{l}$ is stable-blocked in $\boldsymbol{AT}$ w.r.t. $\mathcal{Q}$.

□

4.2.3. Conditional completeness

Next, we consider the completeness of our algorithm. As we illustrate in Example 9, Stability-label is not complete for all argumentation theories.

Example 9. (*Example 4 continued*) We return to our running example on fraud. Consider the argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, -)$ where $\mathscr{L}$, $\mathscr{R}$ and $\mathscr{C}$ correspond to the language, rules and queryables in Fig. 1, but $\mathscr{K} = \{\neg sm, rm\}$. Stability-label does not label f stable: it expects a future argument for f based on $cd, \neg rd, d \Rightarrow f$, where the argument for cd is based on $sp, \neg b \Rightarrow cd$ 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 AT' in $F_{\mathscr{C}}(AT)$ there is no argument for f in $Arg(AT')$, so f should be labelled $\langle 1, 0, 0, 0 \rangle$.

Example 9 shows that there are argumentation theories where the Stability-label algorithm wrongfully takes the possibility into account that there exists an argument for a literal in a future argumentation theory. Specifically, this issue is caused by an *inconsistent potential argument*, which we define next.

Definition 13. (*Potential argument*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, -)$. A **potential argument** A^p on the basis of an argumentation theory AT given a set of queryables $\mathscr{C}$ is a structure obtainable by applying one or more of the following steps finitely many times:

- c is an **observation-based potential argument** if $c \in \mathscr{C}$ and for each $c' \in \overline{c}$: $c' \notin \mathscr{K}$.
  $\texttt{prem}(A^p) = \{c\}$; $\texttt{conc}(A^p) = c$; $\texttt{sub}(A^p) = \{c\}$.
- $A_1, \ldots, A_m \Rightarrow c$ is a **rule-based potential argument** if there is a rule $c_1, \ldots, c_m \Rightarrow c$ in $\mathscr{R}$ and for each $i \in [1..m]$: A_i is a potential argument on the basis of AT given $\mathscr{C}$ and $\texttt{conc}(A_i) = c_i$.
  $\texttt{prem}(A^p) = \texttt{prem}(A_1) \cup \ldots \cup \texttt{prem}(A_m)$; $\texttt{conc}(A^p) = c$; $\texttt{sub}(A^p) = \texttt{sub}(A_1) \cup \ldots \cup \texttt{sub}(A_m) \cup \{A^p\}$; $\texttt{top}-\texttt{rule}(A^p) = r$.

We denote the set of potential arguments on the basis of AT given $\mathscr{C}$ as $P_{\mathscr{C}}(AT)$ and refer to a potential argument with conclusion c as "a potential argument for c". A potential argument with top rule r is "a potential argument based on r". Given some $A^p \in P_{\mathscr{C}}(AT)$, A^p is **inconsistent** iff there exist $p_1, p_2 \in \texttt{prem}(A^p)$ such that $p_1 \in \overline{p_2}$. Given some set of potential arguments $S \subseteq P_{\mathscr{C}}(AT)$, S is **inconsistent** iff there is some $a \in \mathscr{L}$ such that there is some $a' \in \overline{a}$ and $\{a, a'\} \in \{\texttt{prem}(A^p) | A^p \in S\}$.

Just like arguments, potential arguments can be in conflict. Next, we define p-attacks, which are rebuttal attacks between potential arguments.

Definition 14. (*P-attack*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, -)$. For two potential arguments $A^p, B^p \in P_{\mathscr{C}}(AT)$ we say that A^p **p-attacks** B^p iff A^p's conclusion is c and:

attack on conclusion : there is some $c' \in \overline{c}$ such that c' is the conclusion of B^p and $c' \notin \mathscr{K}$.

attack on subargument : there is some $c' \in \overline{c}$ such that c' is the conclusion of a subargument B' of B^p such that $B' \neq B^p$ and $c' \notin \mathscr{K}$.

We write A^p *p-attacks* B^p *on* B' if A^p attacks B^p, $B' \in \texttt{sub}(B^p)$ and $\texttt{conc}(A^p) \in \overline{\texttt{conc}(B')}$.

Note that, for a given argumentation theory AT, each argument on the basis of AT or some future argumentation theory is a potential argument in $P_{\mathscr{C}}(AT)$ since $\mathscr{K} \subseteq \mathscr{C}$ and for each $(AS, \mathscr{K}')$ in $F_{\mathscr{C}}(AT) : \mathscr{K}' \subseteq \mathscr{C}$. On the other hand, there may be a potential argument $A^p \in P_{\mathscr{C}}(AT)$ such that there is no $AT' \in F_{\mathscr{C}}(AT)$ with $A^p \in Arg(AT')$, but then A^p must be inconsistent; we encountered this situation in Example 9, where the only potential argument for f in $P_{\mathscr{C}}(AT)$ was inconsistent and therefore not an argument in any future argumentation theory.

Being an approximation algorithm, Stability-label heuristically reasons with potential arguments rather than with arguments in future argumentation theories. We explicate this in the next lemma, which lists the conditions under which one or more booleans of the stability label are labelled negative. These six items can be shown by induction. For the proofs, we refer to Appendix B.8.

Lemma 3. (*Conditions for labelling*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, -)$ and let L be the labelling after executing the Stability-label algorithm on $\mathscr{L}$, $\mathscr{R}$, $-$, $\mathscr{C}$ and $\mathscr{K}$. Let L_p be the labelling after executing Preprocess on $\mathscr{L}$, $\mathscr{R}$, $-$, $\mathscr{C}$ and $\mathscr{K}$. Let $l \in \mathscr{L}$ be a literal. Then:*

1. *$L_p[l] = \langle 1, 0, 0, 0 \rangle$ iff there is no potential argument A^p for l in $P_{\mathscr{C}}(AT)$.*
2. *If there is an argument for l in $Arg(AT)$, then $\neg L[l].u$.*
3. *If each potential argument for l in $P_{\mathscr{C}}(AT)$ is p-attacked by an observation-based argument in $Arg(AT)$, then $\neg L[l].d$ and $\neg L[l].b$.*
4. *There is an argument A for l in $Arg(AT)$ and there is no observation-based potential argument in $P_{\mathscr{C}}(AT)$ that p-attacks A iff $\neg L[l].u$ and $\neg L[l].o$.*
5. *If each potential argument A^p for l in $P_{\mathscr{C}}(AT)$ is p-attacked by an argument B in $Arg(AT)$ and there is no observation-based potential argument in $P_{\mathscr{C}}(AT)$ that p-attacks B, then $\neg L[l].d$.*
6. *If there is an argument A for l in $Arg(AT)$ and each potential argument B^p in $P_{\mathscr{C}}(AT)$ that p-attacks A is p-attacked by an observation-based argument in $Arg(AT)$, then $L[l] = \langle 0, 1, 0, 0 \rangle$.*

Because of this heuristic of reasoning with potential arguments, Stability-label efficiently recognises many, though not all situations in which a literal is stable. From Lemma 3 we can now derive in which situations literals are stable in the argumentation theory, but not labelled as such. Before formally specifying this in Proposition 3, we give some intuitions on the three complications of using potential arguments instead of arguments in future argumentation theories:

1. Potential arguments are not required to be consistent, while inconsistent potential arguments cannot be derived in any future argumentation theory – recall Example 9.
2. Consistent potential arguments do exist as arguments in some future argumentation theory, but in some situations Stability-label needs to reason with combinations of potential arguments, which may not be derivable from the same future argumentation theory. We will see this in Example 10.
3. The existence of a potential argument in a future argumentation theory may cause other potential arguments to become arguments in the theory as well. This is illustrated in Example 11.

Example 10. (*Incomplete because of observation-unattackable literal*) In

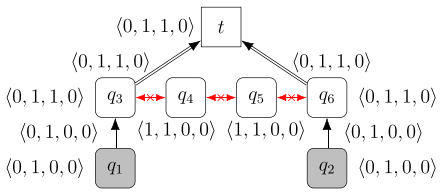


Fig. 11. t is stable-defended in AT, but it is not labelled as such because each argument for t in $Arg(AT)$ is p-attacked by an observation-based potential argument in $P_{\mathscr{O}}(AT)$. However, there is no $AT' \in F_{\mathscr{O}}(AT)$ in which all arguments for t are attacked by an observation-based argument in $Arg(AT')$. (Proposition 3 Case 2a) Note that this inconsistency can be between more than two arguments; for example consider the argumentation theory obtained by adding to $\mathscr{L}$ and $\mathscr{O}$: q_7, q_8, q_9, q_{10}, adding q_7 to $\mathscr{K}$, contradictories: $\overline{q_6} = \{q_5, q_8\}$, $\overline{q_8} = \{q_6, q_9\}$, $\overline{q_9} = \{q_8, q_{10}\}$, $\overline{q_{10}} = \{q_8\}$ and add the rules $q_7 \Rightarrow q_{10}$ and $q_{10} \Rightarrow t$ to $\mathscr{R}$.

the argumentation theory $AT = (AS, \mathscr{K})$ of Fig. 11, there are two arguments for t in $Arg(AT)$, i.e. $A : [q_1 \Rightarrow q_3] \Rightarrow t$ and $B : [q_2 \Rightarrow q_6] \Rightarrow t$. These arguments are not attacked by any argument in $Arg(AT)$, so t is defended in AT. There are potential arguments for q_4, which p-attacks A, and q_5, which p-attacks B. As a consequence, t is not labelled stable-defended by Stability-label, as the algorithm assumes that these potential arguments can become actual arguments in a future argumentation theory. There is however no $AT' \in F_{\mathscr{O}}(AT)$ in which both q_4 and q_5 are in $Arg(AT')$. Therefore, in each $AT' \in F_{\mathscr{O}}(AT)$, some argument for t is not attacked by any argument in $Arg(AT')$, so t is stable-defended in AT w.r.t. $\mathscr{O}$.

In the example above, t is stable-defended in AT w.r.t. $\mathscr{O}$ but not labelled as such because the attackers of arguments for t are not derivable from the same future argumentation theory. Formally, t is observation-unattackable in AT w.r.t. $\mathscr{O}$, as defined next.

Definition 15. (*Observation-unattackable*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $l \in \mathscr{L}$ be a literal. l is **unattackable** in AT w.r.t. $\mathscr{O}$ iff there is no consistent set of potential arguments $T^p \subseteq P_{\mathscr{O}}(AT)$ such that each argument for l in $Arg(AT)$ is p-attacked by some potential argument in T^p. l is **observation-unattackable** in AT w.r.t. $\mathscr{O}$ iff there is no consistent set of observation-based potential arguments $T^p \subseteq P_{\mathscr{O}}(AT)$ such that each argument for l in $Arg(AT)$ is p-attacked by some potential argument in T^p.

A final cause of Stability-label's incompleteness is related to the fact that updating a knowledge base so as to add a potential argument to the set of arguments may introduce additional arguments.

Example 11. (*Incomplete because of forcing argument*) Fig. 12 visualises an argumentation theory $AT = (AS, \mathscr{K})$ where $\mathscr{O} = \{q_1, \neg q_1, q_2, \neg q_2, q_3, \neg q_3\}$ and $\mathscr{K} = \{q_1, q_3\}$. Both in the current argumentation theory and in the two future argumentation theories $(AS, \mathscr{K} \cup \{q_2\})$ and $(AS, \mathscr{K} \cup \{\neg q_2\})$, t is blocked because there are rule-based arguments for

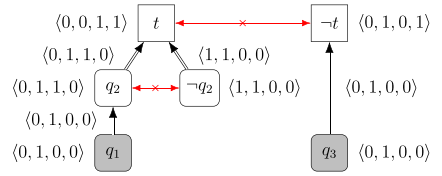


Fig. 12. For each $AT' \in F_{\mathscr{O}}(AT)$: t and $\neg t$ are blocked in AT', but not labelled as such. t is labelled $L[l].o$ because the argument $[q_1 \Rightarrow q_2] \Rightarrow t$ is p-attacked by the observation-based potential argument $\neg q_2$ of which the introduction in AT forces a new argument for t (Proposition 3 Case 2b). $\neg t$ is labelled $L[\neg t].d$ because there is an argument $q_3 \Rightarrow \neg t$ for $\neg t$ that is attacked by the argument $[q_1 \Rightarrow q_2] \Rightarrow t$, which is in turn p-attacked by the observation-based potential argument $\neg q_2$, but the introduction of $\neg q_2$ in AT forces $\neg q_2 \Rightarrow t$, which attacks $q_3 \Rightarrow \neg t$ (Proposition 3 Case 5c).

both t and $\neg t$. However, Stability-label does not label t as stable because it reckons with the possibility that $[q_1 \Rightarrow q_2] \Rightarrow t$, the only argument for t in $Arg(AT)$, is attacked by $\neg q_2$ in a future argumentation theory. The algorithm does not take into account the side effect of adding $\neg q_2$ to the knowledge base, that is: the inclusion of the argument $\neg q_2 \Rightarrow t$, which attacks and is attacked by the argument $q_3 \Rightarrow \neg t$.

Next, we formally define this phenomenon as potential arguments forcing arguments.

Definition 16. (*Forcing arguments*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$. Given two potential arguments A^p, B^p in $P_{\mathscr{O}}(AT)$, the introduction of A^p in AT **forces** B^p iff $B^p \in Arg((AS, \mathscr{K} \cup \texttt{prem}(A^p)))$. We say that the introduction of A^p in AT **forces a new argument** for l iff there is some B^p for l in $P_{\mathscr{O}}(AT)$ such that the introduction of A^p in AT forces B^p and $B^p \notin Arg(AT)$.

At this point, we have addressed the three causes of Stability-label's incompleteness: inconsistent potential arguments, observation-unattackable literals and potential arguments forcing arguments. We use these notions to formally specify the situations in which a literal is stable, but not labelled as such.[5] The full proof of Proposition 3, as well as additional examples illustrating the cases of incompleteness, can be found in B.8.

Proposition 3. (*Conditional completeness stability labelling*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let L be the labelling after executing the Stability-label algorithm on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{O}$ and $\mathscr{K}$. Given a literal $l \in \mathscr{L}$, if l is stable in AT but l is not labelled stable by L, then some of the following five cases applies:*

1. *l is stable-unsatisfiable in AT and each potential argument for l in $P_{\mathscr{O}}(AT)$ is inconsistent.*
2. *l is stable-defended or stable-blocked in AT and:*
   (a) *l is observation-unattackable in AT w.r.t. $\mathscr{O}$;*[6] *or*
   (b) *some argument A for l in Arg(AT) is p-attacked by an observation-based potential argument B^p such that the introduction of B^p in AT forces a new argument for l.*
3. *l is stable-defended in AT and there is an argument A for l in Arg(AT) that is p-attacked by a potential argument in $P_{\mathscr{O}}(AT)$ that is not p-attacked by an argument in Arg(AT) and:*
   (a) *each potential argument in $P_{\mathscr{O}}(AT)$ p-attacking A that is not p-attacked by an argument in Arg(AT) is inconsistent; or*
   (b) *there is a consistent potential argument B^p in $P_{\mathscr{O}}(AT)$ p-attacking A that is not p-attacked by an argument in Arg(AT), but the introduction of B^p in AT forces a new argument for l; or*
   (c) *l is unattackable in AT w.r.t. $\mathscr{O}$.*
4. *l is stable-out in AT and each potential argument for l in $P_{\mathscr{O}}(AT)$ that is not p-attacked by an observation-based argument in Arg(AT) has inconsistent premises; or*
5. *l is stable-blocked in AT and there is a potential argument A^p for l in $P_{\mathscr{O}}(AT)$ such that each argument in Arg(AT) p-attacking A^p is p-attacked by an observation-based potential argument C^p in $P_{\mathscr{O}}(AT)$ and:*
   (a) *A^p is inconsistent; or*
   (b) *the introduction of A^p in AT forces an argument attacking A^p; or*
   (c) *the introduction of C^p in AT forces an argument that p-attacks A^p.*

Proof sketch. There are four cases in which a literal $l \in \mathscr{L}$ is stable, but not labelled as such by L.

[5] Note that these conditions for incompleteness cannot be checked in polynomial time. In fact, under the assumption that P ≠ NP it is generally impossible to check any exact incompleteness conditions for any stability approximation algorithm in polynomial time: if that would be possible, then this could be used to create an exact polynomial algorithm for the stability problem.

[6] This condition is specifically required for argumentation systems that do not have classical negation as a contrariness function.

- If l is **stable-unsatisfiable** in AT but $\boldsymbol{L}[l] \neq \langle \boldsymbol{1},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$ then by Lemma 3 Item 1, each potential argument for l in $P_{\mathscr{C}}(AT)$ is inconsistent **(Case 1)**.
- If l is **stable-defended** in AT but $\boldsymbol{L}[l] \neq \langle \boldsymbol{0},\boldsymbol{1},\boldsymbol{0},\boldsymbol{0}\rangle$ then:
  - If $L[l].u$ or $L[l].o$ (although l is not unsatisfiable or out in any $AT' \in F_{\mathscr{C}}(AT)$) then either l is observation-unattackable in AT **(Case 2a)** or there is an argument $A \in Arg(AT)$ for l that is p-attacked by some potential argument of which the introduction in AT forces a new argument for l **(Case 2b)**.
  - If $\neg L[l].u$ and $\neg L[l].o$ then there must be some rule-based argument A for l in $Arg(AT)$ that is not p-attacked by any observation-based argument in $P_{\mathscr{C}}(AT)$ (Lemma 3 Item 4). Still, l is not labelled stable-defended by L, so there must be at least one potential argument p-attacking A. Then either each potential argument p-attacking A is inconsistent **(Case 3a)**; there is some consistent B^p p-attacking A, but the introduction of this potential argument in AT forces some new argument for l **(Case 3b)**; or it is not possible to attack all arguments for l in $Arg(AT)$ at the same time **(Case 3c)**.
- If l is **stable-out** in AT but $\boldsymbol{L}[l] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{1},\boldsymbol{0}\rangle$ then there is an argument for l in $Arg(AT)$, so by Lemma 3 Item 2 $\neg L[l].u$ which implies $L[l].d$ or $L[l].b$. Then by Lemma 3 Item 3, each potential argument for l in $P_{\mathscr{C}}(AT)$ that is not p-attacked by an observation-based argument in $Arg(AT)$ has inconsistent premises **(Case 4)**.
- If l is **stable-blocked** in AT but $\boldsymbol{L}[l] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{1}\rangle$ then:
  - If $L[l].u$ or $L[l].o$ (although l is not unsatisfiable or out in any $AT' \in F_{\mathscr{C}}(AT)$) then either l is observation-unattackable in AT **(Case 2a)** or there is an argument $A \in Arg(AT)$ for l that is p-attacked by some potential argument of which the introduction in AT forces a new argument for l **(Case 2b)**.
  - If $\neg L[l].u$ and $\neg L[l].o$ then $L[l].d$, so by Lemma 3 Item 5 there is a potential argument A^p for l such that each argument in $Arg(AT)$ p-attacking A^p is p-attacked by some observation-based potential argument C^p in $P_{\mathscr{C}}(AT)$. Then either A^p is inconsistent **(Case 5a)**, or A^p is consistent, so there is a future argumentation theory AT' from which A^p can be derived. However, given that l is blocked in AT', either the introduction of A^p or C^p in AT forces an argument that attacks A^p **(Case 5b/5c)**.

□

4.2.4. Time complexity

In the next two sections, we discuss the computation time of our proposed algorithm. Here we discuss the worst-case time complexity of Stability-label by a formal complexity analysis; in Section 4.2.5 (and in Section 5 on case studies) we will complement this analysis with a series of experiments.

The Stability-label algorithm consists of a preprocessing part (Algorithm 1) and a final labelling part (Algorithm 4 line 2–18). In the next two lemmas, we analyse the individual parts and give proof sketches. Full proofs are available in Appendix B.5.

Lemma 4. (*Time complexity Preprocess*) *The time complexity of Preprocess is* $\mathscr{O}(|\mathscr{L}|^2 + |\mathscr{L}|\cdot|\mathscr{R}|^2)$.

Proof sketch. The time complexity of Preprocess is bounded polynomially by the size of the language and rule set of the argumentation system. The complexity is mainly caused by line 3 (which checks the at most $|\mathscr{L}|$ contradictories of each of the $|\mathscr{L}|$ literals) and line 11 (which checks the label of each of the at most $|\mathscr{L}|$ antecedents of each rule $r \in \mathscr{R}$ in each of the maximal $|\mathscr{R}|$ iterations of the while loop).□

Proposition 4. (*Time complexity Stability-label*) *The time complexity of Stability-label is* $\mathscr{O}(|\mathscr{L}|^3\cdot|\mathscr{R}| + |\mathscr{L}|^2\cdot|\mathscr{R}|^2)$.

Proof sketch. The runtime of Stability-label is particularly dominated by line 15, which relabels all contradictories of the conclusion of a rule that is relabelled. A single execution of this line requires labelling a literal, which in the worst case requires checking the presence of that literal and all of its (at most $|\mathscr{L}|$) contradictories in $\mathscr{C}$ and $\mathscr{K}$, as well as the labels of all (max $|\mathscr{R}|$) rules for that literal or any of its contradictories. line 15 is executed at most $|\mathscr{L}|$ times for each iteration of the while loop. The total number of iterations of the while loop equals the number of times a rule is added to todo-set. A rule is only added to todo-set if it was not yet visited (line 6) or if the label of one of its antecedents changed after a relabelling (line 13 or line 17). Since the label of a literal can change at most four times (i.e. at most four booleans can be turned to False), each rule $r \in \mathscr{R}$ is relabelled at most $5\cdot|\texttt{ants}(r)|$ times. This means that line 15 is executed at most $5\cdot|\mathscr{L}|^2\cdot|\mathscr{R}|$ times. Then the total time required for all iterations of line 15 is at most $5\cdot c\cdot(|\mathscr{L}|^3\cdot|\mathscr{R}| + |\mathscr{L}|^2\cdot|\mathscr{R}|^2)$, where c is a positive constant.□

4.2.5. Computation time

The worst-case complexity analysis shows that Stability-label is polynomial, having a worst-case time complexity of $\mathscr{O}(|\mathscr{L}|^3\cdot|\mathscr{R}| + |\mathscr{L}|^2\cdot|\mathscr{R}|^2)$ (Proposition 4), which guarantees that the impact of the size of the argumentation system on the computation time is limited. Considering that Stability-label is sound and in many cases complete (Sections 4.2.2 and 4.2.3), this polynomial algorithm is preferable to an exact exponential algorithm for our application in argument-based inquiry. Still, the fact that an algorithm is polynomial states that its runtime on inputs of size n is at most $c\cdot n^k$ for some positive constants k,c, but

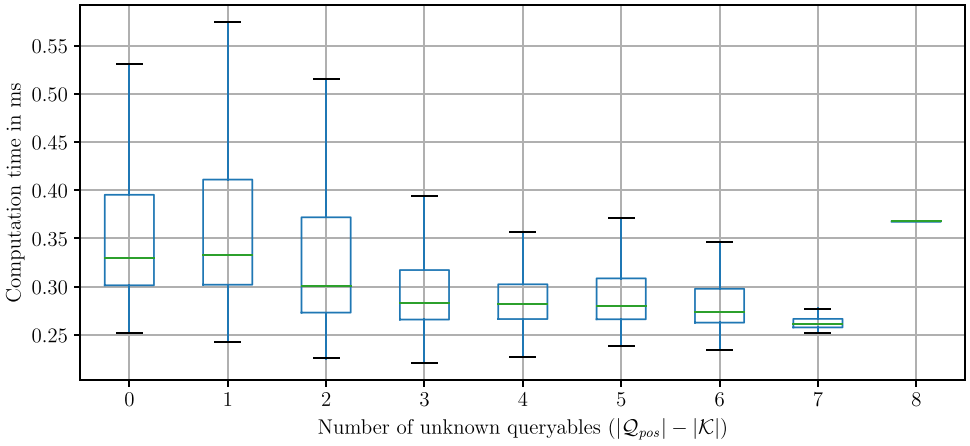


Fig. 13. Computation time of Stability-label on the argumentation theories for the running example on online trade fraud (see Fig. 1), grouped by the number of unknown queryables.

does not necessarily imply that the algorithm runs in, say, a few milliseconds. If the constants k and/or c are very large, the runtime may still exceed the limit of what is acceptable. A natural question that might arise at this point is: is Stability-label fast enough for practical applications?

Experiment 1: running example As a first experiment, we measure Stability-label's computation time on all 6551 possible argumentation theories for our running example on online trade fraud. Fig. 13 shows the results as boxplots, grouped by the number of unknown queryables. In contrast to Stability-naive, the size of the unknown queryable set does not play a significant role in the runtime of Stability-label. The stability labels are typically estimated in less than a millisecond. This is well within the limits of acceptable runtime for (real-time) applications at the police.

Experiment 1 shows that our algorithm runs fast on the small running example. However, from its theoretical upper bound it may seem that Stability-label's runtime will highly (and potentially problematically) increase when the algorithm is executed on algorithms with large languages and/or rule sets. In order to rebut this presumption, we conducted two experiments in which we measure Stability-label's runtime of randomly generated argumentation theories parametrised by language and rule size.

Experiment 2: randomly generated argumentation systems As a second experiment, we timed Stability-label on randomly generated argumentation theories with varying language and rule set sizes. A detailed description of our argumentation theory generation procedure can be found in Appendix C. The results are shown in Fig. 14. In general, for a fixed number of literals the computation time increases as the number of rules increases. This is in line with our expectations: more rules result in more iterations of the while loop in Algorithm 4 line 7–17. Note that the

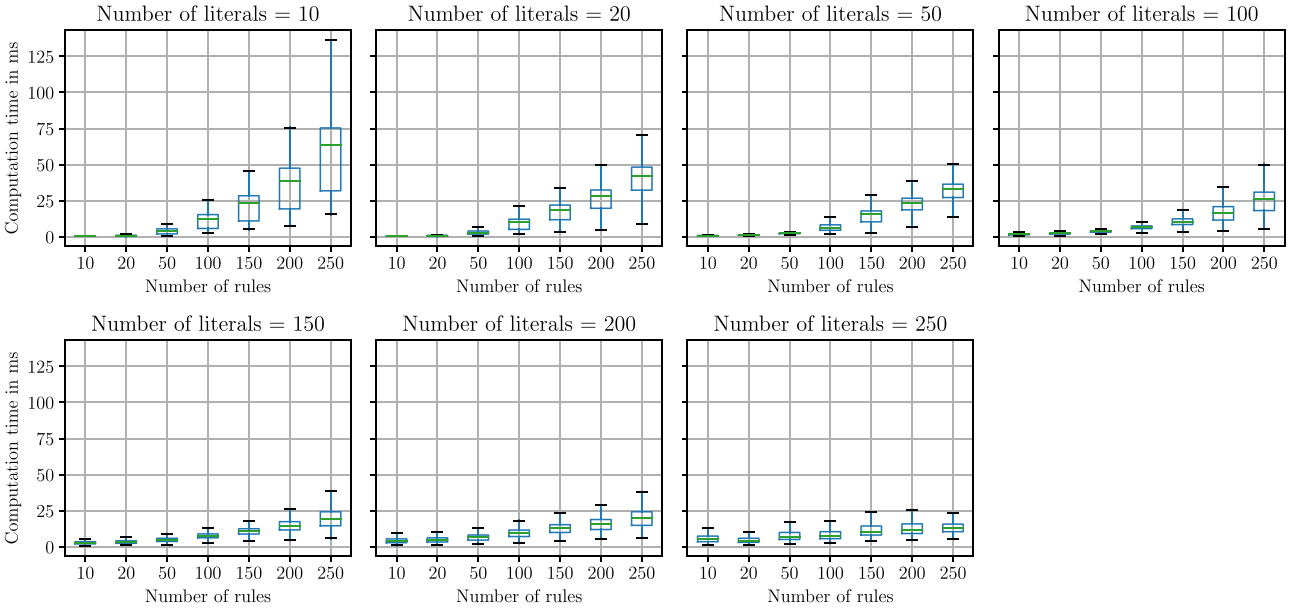


Fig. 14. Computation time of randomly generated data set parametrised by language and rule sizes.

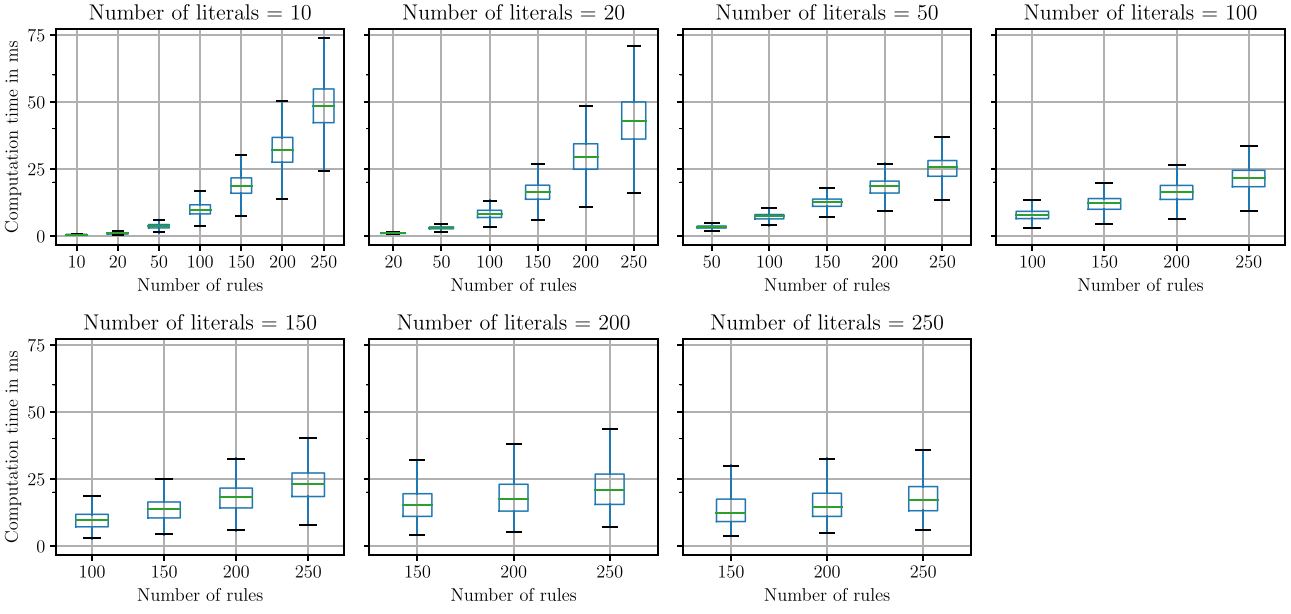


Fig. 15. Computation time of randomly generated layered data set parametrised by language and rule sizes. Note that some combinations of literals and rules were not possible, e.g. it is not possible to make an argumentation system with 200 literals and 100 rules, given that we initialised the generator in such a way that it makes argumentation systems of three layers.

Table 2
Overview of the case studies .

Application	Description	Topic literals	Section
Fraud Intake	An agent that handles the intake of citizens' complaints on online trade fraud.	*The complainant was allegedly a victim of fraud (based on Article 326)*. Additional topic literals are possible actions, such as passing on the complaint to a specialised team or referring the citizen to the Legal Services Counter.	5.1
International Messages	An agent that supports international communication between law-enforcement organizations.	Possible actions (e.g. respond to the message or ignore it).	5.2
Web Shop Classification	An agent that assists in the classification of fraudulent web shops.	Factors contributing to an advice on the trustworthiness of the web shop	5.3

computation time decreases as the number of rules is kept constant and the number of literals increases. Our explanation for this is as follows: an argumentation system that has relatively many literals and few rules must have many literals such that there is no rule for that literal or one of its contradictories. For each of these literals l, only three labels are possible: $L[l] = \langle 1,0,0,0\rangle$ iff $l \notin \mathscr{C}$ or $l \in \mathscr{K}$; $L[l] = \langle 1,1,0,0\rangle$ iff $l \in \mathscr{C}$ and $l \notin \mathscr{K}$ and for each $l' \in \bar{l}$: $l' \notin \mathscr{K}$; and $\langle 0,1,0,0\rangle$ iff $l \in \mathscr{K}$. Those labels are assigned at the latest by Algorithm 4 line 6. On the other hand, in argumentation systems with few literals and many rules, there will be some literals that are considered for relabelling many times because there are many rules for (a contradictory of) that literal. This is reflected in the relatively long computation times of argumentation systems with 10 literals and 250 rules. Still, even for these argumentation systems Stability-Label is reasonably fast with a computation time in the order of tenths of a second.

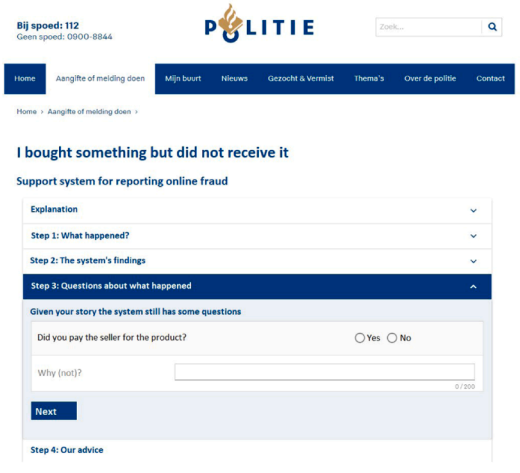
The data sets generated in the previous experiment provide some insight in the influence of the language and rule set size on the computation time, but it should be noted that the structure of these randomly generated argumentation systems differs from the argumentation systems used in practice, for example in our case studies (Section 5). The argumentation systems used in our case studies have a layered structure with the following four properties: first, all literals that are not queryable are either the conclusion of some rule or have contradictory for which a rule exists; other literals would be unsatisfiable in any argumentation theory and therefore not interesting for an inquiry dialogue. Second, for each rule $r \in \mathscr{R}$, the set of its antecedents and consequent is consistent. This means that rules such as $a, b, c \Rightarrow \neg a$ or $\neg a$, $a \Rightarrow b$ would not exist. Third, for each rule $r \in \mathscr{R}$, each literal occurs at most once in the antecedents or consequent, which excludes rules such as $a, b, b, c \Rightarrow d$ or $a, b \Rightarrow a$. Fourth, in applications where the distinction between the unsatisfiable and out statuses is not important, $\mathscr{R}$ does not contain rules for queryable literals.

Experiment 3: layered argumentation systems In order to assess Stability-label's computation time for more realistic argumentation theories, we conducted another experiment on a data set generated by an alternative procedure. The resulting argumentation theories have a layered structure, as concisely described above; for more details, we refer to Appendix C. The resulting computation times are shown in Fig. 15. Note that not all combinations of literal and rule set sizes are possible. In general, the results are similar to the results of Experiment 2, though slightly less computation time is needed for argumentation theories with many rules and few literals.

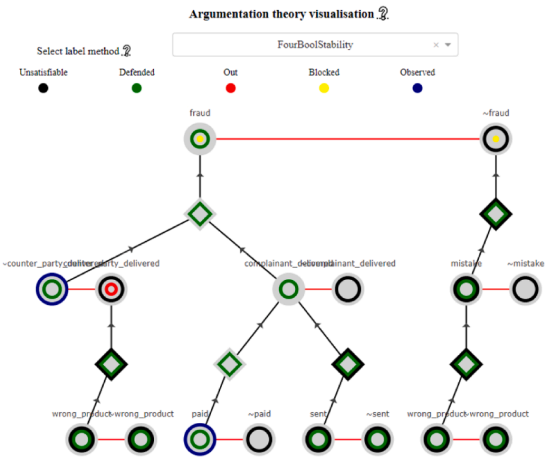
From the experiments above we derive that Stability-label is sufficiently fast for argumentation systems that have at most a few hundred literals and/or rules. As we will see in Section 5, argumentation systems for our applications at the Netherlands Police are an order of magnitude smaller. To conclude, Stability-label is clearly fast enough for practical applications.

5. Case studies at the netherlands police

One of the challenges faced by the Netherlands Police is the vast amount of incoming information that needs to be acted upon. Inquiry systems can offer a solution, provided that they are designed and used in accordance with democratic principles and fundamental rights (European Commission, 2021). Specifically, inquiry systems are able to process data, trying to reach some (domain-specific) legal conclusion and investigate if any additional information is required to guarantee that this conclusion will not change any more (or: is stable). By minimizing the information that is collected in order to make a decision, they respect the notion of proportionality. Another important requirement is transparency, which is satisfied by using interpretable symbolic AI methods.



(a) The agent on the web site of the Netherlands Police.



(b) Visualisation of the stability labels.

Fig. 16. Screenshots of the actual and demo versions of the agent for fraud intake. The English translation is ours; the agent on the police web site is only in Dutch.

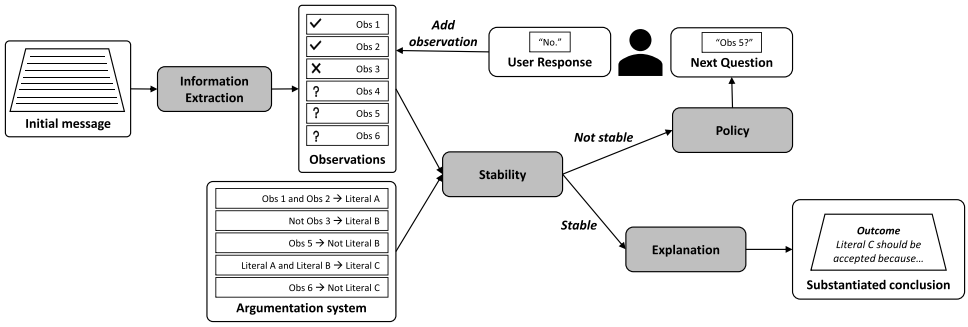


Fig. 17. Architecture of the hybrid inquiry agent. The agents for fraud intake (Section 5.1) and handling international messages (Section 5.2) are instantiations of this architecture.

In this section, we share our experience with the application of argumentation-based inquiry agents at the Netherlands Police in three use cases. This way, we not only show how our algorithm for estimating stability can be applied in practice, but also provide experiences with applying argumentation-based solutions in realistic settings.

All three use cases, summarised in Table 2, concern inquiry agents in a context where unstructured data is processed in order to reach some legal conclusion, in the domains of fraud intake, handling of international messages and classification of fraudulent web shops. The projects corresponding to the presented use cases are in different phases. The system for fraud intake has been in production since September 2019 (see Fig. 16a) while the other projects are still in a proof of concept phase.

5.1. Case study: fraud intake

The first implementation of our architecture in an agent at the Netherlands Police concerns a system that assists in the intake of online trade fraud. This involves cases such as fake web shops and malicious second-hand traders on trading platforms such as eBay. Every year, the police receives tens of thousands of complaints on online trade fraud, which are typically sent by citizens via an online form, and this number is increasing. Traditional online forms for intake have proven to be insufficient. Citizens tend to have a hard time identifying which facts are relevant or not from a legal standpoint. It is also not always clear for a citizen whether a case is a criminal or civil case. The inquiry agent therefore helps the citizen to provide all relevant legal facts and then proceeds to advise the citizen on the best course of action given these facts and the legal context.

Example 12. (*Trade fraud complaint example*) This is a fictitious example of the free text in a complaint.

> I was looking for a playstation and contacted fred fraudster via eBay. he said that he lives in aberdeen which is far away so we agreed that he would send it to me. But after two weeks waiting I still have not recieved anything.

Except for the fact that this text is written in English and not in Dutch, this is a typical example of the free text submitted in a complaint on online trade fraud. In this complaint, some crucial information is missing: the complainant in this example does not say if he/she paid for the product.

5.1.1. Architecture

The agent for fraud intake is implemented according to the architecture illustrated in Fig. 17. The architecture implements a sense-reason-act cycle. The steps consist of sensing the truth value of observable statements, reasoning about whether the inquiry is finished and if so, about the relevant actions and acting upon the relevant actions in order to gather relevant observations. The *information extraction* component uses natural language processing techniques to automatically extract initial observations, i.e. low-level statements, from the available (free text) input. These observations are then combined with the (case study specific) argumentation system to build arguments for and against the topic literal. The *stability* component uses Stability-Label to decide if any additional observations that the user could possibly add in the future can change the justification status of the topic literal. If there are still relevant potential observations, then the *policy* module returns the best query given current observations. Otherwise, the dialogue terminates and the *explanation* module generates an explanation of the outcome.

5.1.2. Information extraction

For the information extraction component, observations are extracted from the initial free-text user input. Observations in this domain are for example “the complainant paid” or “the product was not delivered”. After experimenting with various methods (Schraagen and Bex, 2019; Schraagen et al., 2019; 2017), including machine learning approaches and hand-made classifiers, we decided to use document-level regular expressions (regexes) for information extraction. We furthermore enable the citizen to correct any classification mistakes in a separate screen, which is shown immediately after the text input step. From this point, the observations are used as axiom literals in the initial knowledge base.

5.1.3. Argumentation system

The argumentation system was developed in collaboration with domain experts at the national centre for counteracting online trade fraud and the Netherlands Public Prosecution Service. Table 3 reports on some specifications. The argumentation system has a layered structure, in the sense that there is a high-level rule, based on Article 326 of the Dutch Criminal Code, specifying whether there is evidence of fraud. We use seven different topic literals, which correspond to the follow-up actions. These actions include e.g. accepting the complaint, consulting a human expert, rejecting the complaint but accepting it as a civil case or completely rejecting it and are dependent on the rule on fraud. Other rules relate general legal terms to more specific literals, where the most specific literals are concrete enough to ask to the citizen using the system. These queryable literals make up half of the language. There are no

Table 3
Specification of the argumentation system used for fraud inquiry at the police .

Number of literals	30 (and negations)
Number of queryable literals	15 (and negations)
Number of topic literals	7
Number of rules	43
Number of future argumentation theories from $(AS, \emptyset)$	5.443.200
Number of potential arguments	126
Number of arguments in at least one future argumentation theory	124

rules for queryable literals.

For contradiction, we use a function in which not only each literal and its negation are contradictories, but we add an additional contradictory relation between some literals. For example, the observation $\neg$*package_ordered* and the observation *delivery_proof_fake* are contradictory, because alleged victims of online trade fraud that did not order a package will not receive a fake delivery proof. This way, we restrict the number of questions in some dialogues. Note that the number of future argumentation theories from $(AS, \mathscr{C})$ is just 5.443.200; if we would not have these additional contradictories, we would have $3^{|\mathscr{C}_{\text{pos}}|} = 3^{15} =$ 14.348.907 future argumentation theories (where $\mathscr{C}_{\text{pos}}$ is the set of all non-negated queryables).

Whereas the process of identifying rules *for* fraud was relatively straightforward, it was more challenging to construct rules on which counterarguments could be based. For domain experts without a background in logic or knowledge representation, it is not trivial to fully understand the consequences of an attack between arguments. In particular, we experienced difficulties in identifying on which level a counterargument should be placed. To give an example: we recently extended the rule set so as to accept not only cases of fraud, but also fraud attempts. In case of a fraud attempt in the context of online trade fraud, the complainant realised in time that the counterparty would not deliver goods as promised and therefore did not pay. For fraud, the police maintains a directive that the complainant should have waited at least five days after the promised date of delivery. The complainant not having waited long enough is a reason for the police not to accept a complaint on trade fraud. However, the question if it would be a reason not to accept a complaint on fraud attempt turned out to be hard to answer. We tried to simplify this process in two ways.

First, we developed a user interface that visualises the argumentation system and directly shows the effect of adding or removing a literal to the knowledge base on the literal labelling, see Fig. 16.[7] This gave not only domain experts but also the software developers implementing the intake agent, insight in the working of the labelling algorithm. Second, we iteratively developed the rules by providing the domain experts various example observation sets and asking them if this would be a case of (attempted) fraud or if more information would be required.

Given the size of the argumentation system and our experiments in Section 4.2.5, we would expect that running stability takes no more than a few milliseconds. We verified this with the following experiment. First, we generated a data set in which we sampled 10.000 knowledge bases, i.e. consistent subsets of $\mathscr{C}$, for each size between 0 and 15. This resulted in 160.000 argumentation theories based on the fraud intake argumentation system. For each of these argumentation theories, we measured the time required for running Stability-label ten times. The average computation time is 1.7 ms and is not dependent on the number of unknown literals. This shows that our algorithm can be used in real time for estimating stability in applied argumentation-based inquiry.

5.1.4. Policy

In argumentation theories where not every topic literal is stable, it is necessary to query the citizen using the system for more information. This is handled by the policy module. In the fraud intake agent, the policy module consists of two parts: a relevance listing step that identifies which questions are still relevant and a strategy for choosing the next question of this relevant set.

The relevance listing step is argumentation-based and uses an approximation algorithm that is similar to Stability-label in order to find out which queryable literals are still relevant, in the sense that observing them in the future would lead to a stable situation. The details of this algorithm are outside the scope of this paper.[8] Once we have a set of relevant literals for each of the topics, the inquiry agent should select the next question from these relevant literals. In the current system, the agent asks for the literal in the relevant set with the highest priority.

5.1.5. Explaining stability

The main target audience of explanations in this use case are the citizens submitting a complaint. An explanation is particularly interesting in those situations where the intake agent advises not to submit the complaint, because the citizen does not seem to be a victim of fraud. An initial experimental study on the effect of various types of explanation on the trust of citizens shows that a substantive explanation that gives reasons for its advice (e.g. "The advice is not to submit a complaint. This is probably not a case of fraud since you tried to buy a product from a trusted web shop") can help in improving citizens' trust in the fraud intake agent (Nieuwenhuizen, 2020). Being argumentation-based, our inquiry agent architecture has high potential for explainability (see e.g. Borg and Bex, 2021a; Borg and Bex, 2021c; Cyras et al., 2021; Vassiliades et al., 2021). For our use case, we selected the smallest sets of queryable literals for which a topic literal is labelled stable-defended and mapped them to a natural-language explanation, similar to the one provided at the beginning of this paragraph.

5.1.6. Discussion

The inquiry intake agent has been running on the police website since September 2019 and has been used over 240.000 times in three years. We found that such argumentation-based system is useful for inquiry, but that an approximation-based algorithm is required to guarantee short computation times. We also did a general evaluation of the system. For those complaints that are actually submitted, we compare the system's advice to the human domain expert's assessment. In around 85% of the cases, the advice of the system agrees with the human expert's advice. In addition, we ask citizens about their experience in using the system in a poll that pops up immediately after usage of the system. In this poll, citizens give a score of 4.3 out of five, which suggests that they are quite satisfied with the system.

5.2. Case study: international messages

The next use case concerns the support of international communication between law-enforcement organizations. An earlier description of this use case can be found in Testerink et al. (2019a). The communication concerns mutual requests for legal assistance and informative messages regarding basically anything related to crime. The police has to judge per message what the right course of action is. Some messages can simply be ignored, whilst others may require a thorough follow-up. It is not uncommon that an employee has to consult multiple administrative systems in order to properly process a message. This digital inquiry is exactly what the algorithm discussed in this paper supports.

Example 13. (*Interpol message*) Consider the following fictional but realistic example of an unstructured message:

> Subject: Piet Jan PUK, dob 12.03.1945, r/o Netherlands, 123 Stad, Eenlaan 1 - loss of identification document
> Dear colleagues,
> Please be informed that the a/m subject reported a loss of his identification document - driving licence to our Local Police Station in East Arkham. Please supply us with details about the driving licence (date and place of issue, validity) because these data are necessary for inserting the document into the database of stolen and lost documents.

[7] See https://www.uu.nl/en/onderzoek/ai-labs/nationaal-politielab-ai/approximating-stability-applied-argument-based-inquiry for a demo of this interface.

[8] See Odekerken et al. (2022) for a discussion on the complexity of this problem.

Thank you for your cooperation. Best regards, End. NCB Canberra

To process this message, one first has to observe that this message is a request which warrants a response and that the existence of a document is to be determined. Next, it has to be checked whether sufficient information is available for processing the request. If not, then a follow-up question ought to be formulated. Otherwise, it has to be observed whether or not the document indeed occurs in the administration. Depending on whether this is the case, the response would be either the requested information or a notification that the document is unknown. Although this example is fairly straightforward, requests can be complex cases with various legal subtleties.

The agent that we designed for handling these international messages is implemented according to the same architecture as the fraud intake agent (see Fig. 17). Whereas the implementation of most components of this architecture depends on the specific use case, the stability component is the same for both use cases and uses the Stability-label algorithm described in Section 4.

First, we discuss some properties of the argumentation system. The inquiry agent of this use case is supposed to give an advice upon an action, which may be to ignore, forward or respond to a message, and to assign it to a low, medium or high importance. Hence we take these upper level actions as the topic literals of the inquiry dialogue. The rules and queryable literals are designed in such a way that they reflect the reasoning behind any current choice of users to execute a certain action or not. The rule set has a fairly modular structure. Roughly speaking, there is a set of rules for different types of actions that the agent may perform. The antecedents of these rules are either queryables or the consequents of other rules. The queryables represent possible outcomes of particular data base queries.

Example 14. (*Excerpt from rule set*) Consider for instance this simplified rule set:

1. *request*, ¬*only_on_hit* ⇒ *respond*
2. *request*, *only_on_hit* ⇒ *ignore*
3. *request*, *only_on_hit*, *query_hit* ⇒ ¬*ignore*
4. *request*, *only_on_hit*, *query_hit* ⇒ *respond*
5. *query_hit_person_data_base* ⇒ *query_hit*
6. *person_query_required*, *person_found* ⇒ *query_hit_person_data_base*

This set represents the reasoning that requests are ignored if only positive hits are requested (rule 2), unless there is a hit on the requested query; in that case, the message should be responded to (rules 3 and 4). In case not only positive hits are requested, a response is required in general (rule 1). Rules 3 and 4 have a common antecedent *query_hit*, which is not queryable, but arguments from this literal can be instantiated by literals specific for a given data base, for instance a data base with personal records (see rule 5). Finally, for rules related to database queries, such as rule 6, usually one of the queryables is the actual query result (*person_found*), which is irrelevant if the message does not call for this query. Suppose for example that ¬*person_query_required* is observed; then *person_query_required* and *query_hit_person_data_base* are labelled $\langle 1, 0, 0, 0\rangle$ by Stability-Label. This indicates that the observation *person_found* is not relevant for deciding if a message should be ignored or responded to and the agent will not attempt to query the database. If, alternatively, *person_query_required* is observed, then the *person_found* literal remains relevant. Consequently, the agent may still choose to unveil its valuation by executing the actual database query.

For selecting the next question, the policy is optimised by minimising the number of questions, using reinforcement learning (Schraagen et al., 2019).

At the end of the inquiry process, when all topic literals are labelled stable, the inquiry agent is required to explain why it suggests a specific course of action. For this use case, the explanation is generated automatically by recursively determining the cause of labels for rules and literals.

5.3. Case study: web shop classification

Our third and final use case concerns a human-in-the loop classifier that distinguishes bona fide from mala fide web shops. A large part of the complaints on online trade fraud received by the Netherlands Police concerns reports on web shops that do not deliver goods. In order to decide if a web shop is bona fide or mala fide, human analysts at the national centre for counteracting online trade fraud used to manually check suspect web shops. However, this is a lot of work given the large number of suspect web shops. Therefore, the police experiments with using AI to speed up the process. Creating a classifier for mala fide web shops is not trivial, since properties of these web shops change over time and some of them cannot be assessed automatically and accurately; see Odekerken and Bex (2020) for a more elaborate description of this use case. It is therefore important that the analyst can influence the final decision taken by the classifier. In particular, our algorithm Stability-label plays a central role in validating that the factors that are used as input for the classification procedure are correct. These factors are typically based on observations obtained by scraping the web site, extracting information using natural language processing techniques and interaction with APIs. Note however that an observation obtained this way may not be perfect. Incorrect observations will probably lead to incorrect factors, which in some cases leads to an incorrect advice. We address this by introducing an inference step, in which arguments for factor literals are constructed based on the extracted observations and an argumentation system. In classification, only those factors are considered for which there is an argument in the grounded extension. The rules of this argumentation system are devised in such a way that the experts at the police can always correct the observations found by the feature extractor; see Example 15. Those topics that are not labelled stable-defended by Stability-label are not used for classification without asking the analyst to check and possibly correct the underlying observations.

Example 15. (*Rules for factors*) Fig. 18 illustrates the rules related to the factors *b_trusted* (stating that the web shop is on the trusted list of some trustmark company) and *m_fake_logo* (stating that the web shop misuses a trustmark logo). In this example, *b_trusted* is unsatisfiable in the current argumentation theory, as well as each argumentation theory that can be obtained in the future. The reason for this is that its antecedent *a_registered* is unsatisfiable, given that a contradictory of this queryable literal is observed. The mala fide factor *m_fake_logo* is defended in the current argumentation theory, since there is an argument for this literal that is not attacked. However, in the future argumentation theory $AT' = (AS, \mathscr{K} \cup \{\neg e\_logo\})$, this argument is attacked by an observation-based argument for ¬*e_logo*, which means that *m_fake_logo* is

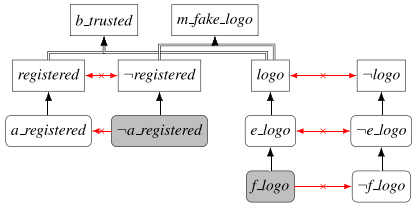


Fig. 18. Excerpt from the rule set. If a feature extractor found a trustmark logo at the web site (*f_logo*), but an API call returns that this site is not registered at the trustmark company (¬*a_registered*), there is an argument for the mala fide factor that the web shop uses a fake trustmark logo (*m_fake_logo*). However, if the analyst would search for the logo on the web shop, the image found by the feature extractor may not be a trustmark logo (¬*e_logo*). In that case, each argument for *m_fake_logo* would be attacked by the observation-based argument for ¬*e_logo*. Consequently, *m_fake_logo* would be out in this future argumentation theory, hence would not be considered as a factor in the (best-case) factor set.

out in AT'.

6. Related work

The notion of stability is related to research in dynamic argumentation, as we further describe in Section 6.1. As we have seen previously, stability is a natural termination criterion for argument-based inquiry. In Section 6.2, we relate our inquiry systems to other approaches in computational argumentation and conversational AI.

6.1. Stability and dynamic argumentation

Dynamic argumentation (Doutre and Mailly, 2018) is a research direction within computational argumentation that assumes that the argumentation framework is not static, but can be altered by e.g. the addition or removal of arguments and/or attacks, possibly resulting in a changed acceptability status of some argument(s). Research on dynamic argumentation includes work on the impact of a change operation (Alfano et al., 2020; 2017; 2019; 2021; Cayrol et al., 2010), enforcement (Baumann and Brewka, 2010; Borg and Bex, 2021b; Doutre and Mailly, 2018), resolution (Modgil and Prakken, 2012) and the relation with belief revision (Falappa et al., 2009; Snaith and Reed, 2016). Since the task of estimating stability is to decide if any further information (resulting in additional arguments) can change the acceptability status of some claim, it can be placed in the field of dynamic argumentation as well.

Most research on dynamic argumentation, e.g. Alfano et al. (2017, 2019); Baumann and Brewka (2010); Cayrol et al. (2010); Doutre and Mailly (2018), only considers the effect of changes in the *abstract* framework, such as adding an argument. In doing so, these approaches abstract away from dependencies between arguments. This is problematic, especially when these approaches reckon with argumentation frameworks for which no structured instantiation exists. For example, adding a new argument A to the arguments of an argumentation framework may introduce new attacks based on its structure in relation to the structure of arguments that are already present. Furthermore, adding a new argument A to the arguments of an argumentation framework also introduces all subarguments of A, which may not have been present in the original framework, possibly in combination with other arguments of which A is a subargument. In addition, adding one argument A may prevent the introduction of an additional argument B, for example because their premises contain mutually inconsistent axioms. Finally, work on abstract argumentation frameworks can only show the effects of adding arguments and/or attacks to the acceptability of *arguments*, but not on the acceptability of *claims*.

Some related work on dynamic argumentation introduces extensions to abstract argumentation frameworks that capture some of these dependencies. For instance, in argument-incomplete argumentation frameworks (Baumeister et al., 2018), the introduction of an argument forces the introduction of all attacks such that both arguments involved in the attack are present in the resulting argumentation theory. Recently, stability was defined for these argument-incomplete argumentation frameworks (Mailly and Rossit, 2020). Still, it is impossible to represent the other aforementioned dependencies between arguments and claims in incomplete argumentation frameworks. Consequently, even an algorithm that would be perfectly accurate in detecting stability on an abstract, argument-centric level cannot be used directly for sound and complete stability detection on a more structured, claim-centric level.

For this reason, we define the *problem* of stability on structured argumentation frameworks. This does not mean that each (approximation) *algorithm* for estimating stability on a structured level captures all aforementioned dependencies between arguments and claims. In fact, our algorithm Stability-label ignores some of these dependencies so as to guarantee polynomial runtime; recall the cases of incompleteness from Section 4.2.3. Still, we consider it to be preferable to present a sound but incomplete algorithm for estimating stability on a structured level, as we do in this paper, than to propose an exact algorithm for detecting stability defined on abstract argumentation frameworks. The reason for this is that the former approach is more transparent, revealing cases of incompleteness rather than concealing them in abstraction. This allows for a fair comparison between stability detection algorithms, such as Stability-label, Stability-naive or any other algorithm to be designed in the future, in terms of soundness, completeness and computational complexity.

At present, none of the existing work in dynamic argumentation specifically studies stability on the structured level. In general, research on dynamics in structured argumentation frameworks is scarce. We briefly discuss some related research. Modgil and Prakken (2012) 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 and prove that results from resolutions on an abstract level cannot always be transferred to the structured level. In Borg and Bex (2021b), enforcement is studied in a structured argumentation setting. Enforcement is related to, yet different from stability. Some literal can be enforced if it is possible to alter the knowledge base in such a way that it is justified, whereas it is stable(-defended) if each possible addition to the knowledge base results in an argumentation theory in which the literal is justified. Another related study (Snaith and Reed, 2016) 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, nor do they provide an efficient (approximation) algorithm. Two related papers that do take computational complexity into account are Alfano et al. (2020, 2021). The authors propose efficient algorithms to minimise re-computations after a change in an Attack-Support Argumentation Framework or 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.

6.2. Inquiry dialogue in computational argumentation and conversational AI

Our approach can also be related to other approaches for inquiry dialogue. In this section, we consider both argumentation-based inquiry and research on conversational AI. Within the argumentation community, inquiry has been studied mainly in the context of dialogue protocols and strategies, which formally define the legal or best move(s) for an agent at each stage of the dialogue (Black and Hunter, 2009; Fan and Toni, 2012; Parsons et al., 2002). Rather than focusing on the procedural context in which the inquiry dialogue takes place, we specifically concentrate on efficiency.

Task-oriented systems in the field of conversational AI can also be considered as inquiry systems. This prototypical pipeline of this kind of system consists of a natural language understanding component, a dialogue management component and a natural language generation component (Chen et al., 2017). Our approach is compatible with this pipeline, in which the stability and policy modules together fulfill the role of the dialogue manager. A natural question that might arise here is why we designed an argumentation-based solution for dialogue management, rather than a hand-written dialogue flow, which is a more common approach for this component of the pipeline. A dialogue flow is based on a finite state machine (Jurafsky and Martin, 2009) that explicitly specifies which question should be asked in which situation. Although this method is conceptually simple at first sight, it has some serious drawbacks compared to our approach. First, in the legal domain it is common to have many exceptions to general rules. Hence answers to questions typically influence whether inquiring upon other questions is

still relevant. For real-world applications, this result in a high number of branches in dialogue flows. It is quite difficult to manually design such a dialogue flow, since a domain expert needs to anticipate for any possible user inputs, see e.g. Paek and Pieraccini (2008). In contrast, our approach does not require manually defining a dialogue flow; legal rules can be applied in the rule set of the argumentation system. Furthermore, an agent based on dialogue flow cannot automatically generate an explanation for its conclusion, while our argumentation-based agent can relate its decisions to an explicit representation of the legal background. This enhances the transparency of the inquiry process.

7. Conclusion

We have studied the task of detecting stability: given a specific structured argumentation theory, based on an instance of ASPIC$^+$, can adding information result in a changed justification status of a specific literal? We have shown that the task is CoNP-complete, which is problematic in practical applications, such as identifying the termination criterion in human-computer inquiry dialogue. To resolve this issue, we proposed a polynomial-time algorithm Stability-label for estimating stability that improves on the algorithms in Testerink et al. (2019b) and Odekerken et al. (2020). 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 three inquiry agents at the Netherlands Police with a legal reasoning subtask.

In addition, we have identified conditions under which Stability-label is complete. Given that the algorithm is not for all argumentation theories complete, there are examples of argumentation theories for which the algorithm does not detect that a literal is stable. In our use cases, this can result in the agent asking unnecessary questions. Still, for our applications, asking a redundant question in some specific situations is preferable to using an exact algorithm for stability that runs in exponential time, such as Stability-naive in Section 3.2.

In this work on stability, we have shown that an inherently complex task in dynamic argumentation can be performed accurately in real time – provided that the input satisfies some specific conditions – using an approximation algorithm. Thanks to this approximation approach, we were able to present one of the first polynomial algorithms for a dynamic argumentation task defined on structured argumentation frameworks. We therefore conclude that the design and (theoretical) analysis of approximation algorithms for complex tasks in computational argumentation is an interesting research direction, which opens up possibilities for real-world applications of computational argumentation.

In future research related to stability, our work can be extended in various ways. A first extension would be to consider more expressive instantiations of ASPIC$^+$, for example an instantiation in which preferences between rules can be specified. This would allow for more natural argumentation theories, but the approximation algorithm would require more complex labelling rules than the ones in Algorithms 2 and 3 and may have additional cases of incompleteness.

A second extension would be to study the stability problem when considering other changes than axiom addition; for example: would the justification status of a literal change if we would add or remove some rules? Although these extensions would not be of particular interest for our use cases at the police, where the legal rules cannot be changed by users, they could be interesting for other applications. However, when adding those, there are some subtleties to keep in mind. First, the application of a change operation to an argumentation theory should result in a valid argumentation theory. If, for example, a rule is added to the argumentation system, then its antecedents as well as its consequent should be in the language. In addition, it is desirable that the resulting argumentation theory (still) satisfies the rationality postulates identified by Caminada and Amgoud (2007), which is not trivial when changes in strict rules, axiom premises and preferences are allowed (see Modgil and Prakken (2013)). Lastly, restrictions on changes (such as only adding axioms from $\mathscr{L}$ to the knowledge base) should be chosen with care, since they prevent the stability problem from becoming trivial.

A third and final extension would be to further explore the notions of relevance and explainability in relation to stability. As we mentioned in Section 5.1.4, we already use an approximation algorithm to select relevant questions, based on the labelling found by Stability-label. Similarly, this labelling can be used to automatically generate explanations (Section 5.2). In order to assess the quality of these algorithms, we first need to formally define notions of relevance of explainability. Subsequently, we could evaluate them by a theoretical and/or empirical analysis. This would give more insight in the quality of these algorithms, and thereby make our argument-based inquiry solution more applicable for general use.

Funding

This research has been partly funded by the Dutch Ministry of Justice and the Netherlands Police.

CRediT authorship contribution statement

Daphne Odekerken: Conceptualization, Methodology, Formal analysis, Investigation, Software, Visualization, Writing – original draft. **Floris Bex:** Conceptualization, Methodology, Supervision, Visualization, Writing – review & editing, Project administration, Funding acquisition. **AnneMarie Borg:** Supervision, Formal analysis, Writing – review & editing. **Bas Testerink:** Conceptualization, Software, Writing – review & editing.

Appendix A. Computing the justification status of a literal

In our experiments and proofs, we use a sound and complete algorithm, to decide upon the justification status of a literal in the (current) argumentation theory. We will refer to this algorithm as Justification-label. Just like Stability-label, it is based on a labelling of literals and rules and is polynomial. Justification-label is sound, complete and runs in $\mathscr{O}(\left|\mathscr{L}\right|^3\cdot\left|\mathscr{R}\right| + \left|\mathscr{L}\right|^2\cdot\left|\mathscr{R}\right|^2)$. For full proofs of soundness, completeness and time complexity, we refer to https://www.uu.nl/en/onderzoek/ai-labs/nationaal-politielab-ai/approximating-stability-applied-argument-based-inquiry.

Lemma 5. (*Justification status in polynomial time*) *Given an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, \_)$, the justification status of l in AT can be found in polynomial time.*

Appendix B. Proofs

Since the full proofs for lemmas and propositions in the paper are quite extensive, we collect them here in the appendix. In Section B.1, we give the full proof of Proposition 1 (the stability problem is CoNP-complete). In Section B.2 we give a more precise specification of which arguments are either in the grounded extension or attacked by an argument in the grounded extension in our ASPIC$^+$ instantiation, as presented in Section 2. These specifications will be useful for the soundness and completeness proofs of Stability-label. The proof of Lemma 1, showing that the four justification

statuses introduced in Section 2.3 are mutually exclusive and complementary, can be found in Section B.3. In Sections B.6 and B.7, we give the full proves of the soundness of the algorithms PREPROCESS and STABILITY-LABEL, respectively. Finally, we give the conditional completeness proof in Section B.8.

B1. Complexity of the stability problem

Proposition 1. (*Complexity of STABILITY problem*) *Given an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, -)$ and $\mathscr{C}$ is a set of queryables, the STABILITY problem is CoNP-complete.*

Proof. Consider an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, -)$ and $\mathscr{C}$ is the set of queryable literals.

We will show that STABILITY is CoNP-complete by a CoNP-hardness proof and a proof that the STABILITY problem is in CoNP. In order to show that STABILITY is CoNP-hard, we will give a polynomial-time computable reduction f from the known CoNP-hard problem UNSAT, which is the decision problem of determining whether a given boolean expression in conjunctive normal form (CNF) has a truth assignment of its variables that makes the output true.

Formally: consider a boolean expression $\phi = (l_{11} \vee \ldots \vee l_{1k}) \wedge \ldots \wedge (l_{n1} \vee \ldots \vee l_{nm})$ in CNF; let $\mathscr{C}$ be the set containing all literals in ϕ and let v_I be the valuation function for propositional classical logic given the truth assignment function $I : \mathscr{C} \rightarrow \{\text{True}, \text{False}\}$. The goal of the UNSAT**problem** is to decide if $v_I(\phi)$ is False for every truth assignment I.

Next, we give a **reduction function** from UNSAT to the STABILITY problem. Let f be the function that converts an arbitrary formula $\phi = (l_{11} \vee \ldots \vee l_{1k}) \wedge \ldots \wedge (l_{n1} \vee \ldots \vee l_{nm})$ in CNF to an argumentation theory $AT = (AS, \mathscr{K})$ with an argumentation system $AS = (\mathscr{L}, \mathscr{R}, -)$ and a set of queryables $\mathscr{C}$ (illustrated in Fig. 3) such that:

- $\mathscr{L}$ consists of:
  (1) For each literal in $\mathscr{C}$: the literal and its negation; and
  (2) For each clause in ϕ: a clause-specific literal and its negation; and
  (3) A topic literal and its negation.
  The following formula expresses this formally, using numbers in superscript to refer to the three items identified above: $\mathscr{L} = \mathscr{C} \cup \{-l_{ij} | l_{ij} \in \mathscr{C}\}^{(1)} \cup \{c_i | \exists j : l_{ij} \in \mathscr{C}\} \cup \{-c_i | \exists j : l_{ij} \in \mathscr{C}\}^{(2)} \cup \{t, \neg t\}^{(3)}$;
- $\mathscr{R}$ consists of:
  (1) For each clause i and for each literal l_{ij} in this clause: $l_{ij} \Rightarrow l_i$; and
  (2) $(c_1, \ldots, c_n) \Rightarrow t$.
  Formally: $\mathscr{R} = \{l_{ij} \Rightarrow c_i | l_{ij} \in \mathscr{C}\}^{(1)} \cup \{(c_1, \ldots, c_n) \Rightarrow t\}^{(2)}$.
- $-$ corresponds to classical negation.
- $\mathscr{C}$ contains all literals occurring in the CNF ϕ and their negations: $\mathscr{C} = \{l_{ij} | l_{ij} \in \mathscr{C} \vee -l_{ij} \in \mathscr{C}\}$; and
- $\mathscr{K}$ is empty: $\mathscr{K} = \varnothing$.

We now examine the time needed for computing the reduction $f(\phi)$ for an arbitrary formula ϕ in CNF. For each literal l_{ij} occurring in ϕ, two literals are added to the language $\mathscr{L}$, one rule is added to $\mathscr{R}$ and two literals are added to $\mathscr{C}$. Furthermore, for each clause two literals are added to $\mathscr{L}$. Finally, the topic literal and its negation are added to $\mathscr{L}$ and a single rule for the topic literal is added to $\mathscr{R}$. Since the number of clauses cannot exceed the number of literals occurring in ϕ, this reduction can be computed $\mathcal{O}(|\mathscr{C}|)$ operations; hence the reduction f can be computed in polynomial time.

Next, we prove that there is no truth assignment $\boldsymbol{I}$ such that $v_I(\boldsymbol{\phi}) = \textbf{True}$ iff $\boldsymbol{t}$ is stable in $\boldsymbol{AT}$ w.r.t. $\mathcal{Q}$.

- **Right to left**: we prove this by contraposition. Suppose that there exists a truth assignment I such that $v_I(\phi) = \text{True}$. $\mathscr{K}$ is empty, so $Arg(AT) = \varnothing$, hence there is no argument for t in $Arg(AT)$. Now let $\mathscr{K}' = \{l_{ij} \in \mathscr{C} | v_I(l_{ij}) = \text{True}\}$ and let AT' be $(AS, \mathscr{K}')$. $v_I(\phi) = \text{True}$, hence for each $i \in [1..n]$, $v_I(l_{i1} \vee \ldots \vee l_{ik}) = \text{True}$. Consider an arbitrary $i \in [1..n]$. $v_I(l_{i1} \vee \ldots \vee c_{ik}) = \text{True}$, so there is a $j \in [1..k]$ such that $v_I(l_{ij}) = \text{True}$. Then by definition of $\mathscr{K}'$, $l_{ij} \in \mathscr{K}'$, which implies that there is an observation-based argument for c_{ij}. Furthermore, there is a rule $l_{ij} \Rightarrow c_i$ in $\mathscr{R}$; therefore there is a rule-based argument for c_i in $Arg(AT')$. Since we chose i arbitrary, for each $i \in [1..n]$, there is a rule-based argument for c_i in $Arg(AT')$. Then there is an argument for t based on the rule $(c_1, \ldots, c_n) \Rightarrow t$ in $Arg(AT')$. There is no argument for t in $Arg(AT)$ and $AT \in F_{\mathscr{C}}(AT)$, so by Definition 8, t is unsatisfiable in AT. There is an argument for t in $Arg(AT')$; this implies that t is not unsatisfiable in AT'. Then by Definition 11, t is not stable in AT w.r.t. $\mathscr{C}$.
- **Left to right**: we prove this part by contraposition as well. Suppose that t is not stable in AT w.r.t. $\mathscr{C}$. Since there is no argument (for t) in $Arg(AT)$ ($\mathscr{K} = \varnothing$), we have that t is unsatisfiable in AT. By Definition 11, this implies that there exists an $AT' = (AS, \mathscr{K}')$ in $F_{\mathscr{C}}(AT)$ such that there is an argument for t in $Arg(AT')$. $t \notin \mathscr{C}$, so the argument for t must be rule-based. The only rule for t in $\mathscr{R}$ is $(c_1, \ldots, c_n) \Rightarrow t$, so there is an argument for t based on this rule in $Arg(AT')$. Then for each $i \in [1..n]$, there is an argument for c_i in $Arg(AT')$. Consider an arbitrary $i \in [1..n]$. $c_i \notin \mathscr{C}$, so there must be a rule-based argument for c_i. Let $\{r_{i1}, \ldots, r_{ik}\}$ be the rules for l_i in $\mathscr{R}$. Let $r_{ij} : l_{ij} \Rightarrow c_i$ (with $i \in [1..k]$) be an arbitrary rule such that there is an argument based on r_{ij} in $Arg(AT')$. Then there must be an argument for l_{ij} in $Arg(AT')$. This argument must be observation-based, hence $l_{ij} \in \mathscr{K}'$. Let I be a truth assignment such that:

$$I(l_{ij}) = \begin{cases} \text{True} & \text{if } l_{ij} \in \mathscr{K}' \\ \text{False} & \text{if } l_{ij} \notin \mathscr{K}'. \end{cases}$$

Given that $l_{ij} \in \mathscr{K}'$, we have that $v_I(l_{ij}) = \text{True}$ and therefore $v_I(l_{i1} \vee \ldots \vee l_{ik}) = \text{True}$ as well. Since we chose i arbitrary, this is the case for every i in $[1..n]$. Then $v_I((l_{11} \vee \ldots \vee l_{1k}) \wedge \ldots \wedge (l_{n1} \vee \ldots \vee l_{nm})) = v_I(\phi) = \text{True}$. So there exists a truth assignment I such that $v_I(\phi) = \text{True}$.

We have shown that there exists a reduction function f from UNSAT to the STABILITY problem that can be computed in polynomial time, such that $v_I(\phi)$ is False for every truth assignment I iff $t \in \mathscr{L}$ is stable in $f(\phi) = AT = (AS, \mathscr{K})$ w.r.t. $\mathscr{C}$. This implies that UNSAT $\leq_p$ STABILITY. UNSAT is CoNP-hard, so **STABILITY is CoNP-hard**.

Now we prove that **STABILITY is in CoNP**. A decision problem is in CoNP iff for any no-instance x there is a "certificate" that a polynomial-time algorithm can use to verify that x is a no-instance. So STABILITY is in CoNP iff for any argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, for any set of queryables $\mathscr{C}$ and for any $l \in \mathscr{L}$ such that l is not stable in AT w.r.t. $\mathscr{C}$, there is a polynomial-size certificate that a polynomial-time algorithm can use to verify that l is not stable in AT w.r.t. $\mathscr{C}$. A suitable certificate would be some knowledge base $\mathscr{K}' \subseteqq \mathscr{C}$ such that the justification status of l in $(AS, \mathscr{K}')$ differs from the justification status of l in AT. Obviously $\mathscr{K}'$ is polynomial-size in the input since $\mathscr{K}' \subseteqq \mathscr{C}$. Furthermore, l is not stable in AT w.r.t. $\mathscr{C}$ iff there is some $AT' \in F_{\mathscr{C}}(AT)$ such that the justification status of l in AT' differs from the justification status of l in AT. Given the certificate $\mathscr{K}'$, the following procedure can be used to verify that l is not stable in AT w.r.t. $\mathscr{C}$:

1. Verify that $\mathscr{K}'$ is consistent and that $\mathscr{K} \subseteqq \mathscr{K}' \subseteqq \mathscr{C}$. If this is the case, then $(AS, \mathscr{K}) \in F_{\mathscr{C}}(AT)$. This step can be done in polynomial time.
2. Find the justification status of l in AT. By Lemma 5, this can be done in polynomial time.
3. Find the justification status of l in AT'. This can be done in polynomial time (Lemma 5).
4. Verify that the justification status of l in AT differs from the justification status of l in AT'. This step can be done in polynomial time as well.

To conclude, we have shown that the STABILITY problem is CoNP-hard and that it is in CoNP. This implies that **STABILITY is CoNP-complete**.□

B2. Specification "in" and "out" the grounded extension

In Lemmas 6 and 7, we give a more precise specification of which arguments are either in the grounded extension (Lemma 6) or attacked by an argument in the grounded extension (Lemma 7). These lemmas will be used repeatedly in the soundness and conditional completeness proofs in Sections B.7 and B.8

Lemma 6. (*Specification "in" grounded extension*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory with argumentation system $AS = (\mathscr{L}, \mathscr{R}, \text{-})$. An argument $A \in Arg(AT)$ is in the grounded extension $G(AT)$ iff each argument attacking A is attacked by an observation-based argument.*

Proof. The proof from **right to left** is trivial: observation-based arguments cannot be attacked (Definition 5), so each observation-based argument is in $G(AT)$. If each argument attacking A is attacked by an observation-based argument, then A is acceptable w.r.t. $G(AT)$, so $A \in G(AT)$.

We now prove the **left to right** part by contradiction. Suppose that $A \in G(AT)$ and that there is an argument B attacking A, and B is not attacked by an observation-based argument. We will prove that there is a strict subset of $G(AT)$ that is complete, which contradicts the assumption that $G(AT)$ is a grounded extension. We construct this set S as follows: $\boldsymbol{S} = \{\boldsymbol{C} \in \boldsymbol{G(AT)} | \textbf{there is no } \boldsymbol{C'} \in \mathtt{sub}(\boldsymbol{C}) \textbf{ s.t. } \boldsymbol{C'} \textbf{ attacks } \boldsymbol{B}\}$.

First, we show that S is a strict subset of $G(AT)$. Since A is attacked by B, there must be some $A' \in \mathtt{sub}(A)$ such that $\mathtt{conc}(B) \in \overline{\mathtt{conc}(A')}$ and $\mathtt{conc}(A') \notin \mathscr{K}$. Given that B cannot be observation-based (otherwise B would be in $G(AT)$, which contradicts the conflict-freeness of $G(AT)$) and $\mathtt{conc}(A') \in \overline{\mathtt{conc}(B)}$ (by the symmetry of -), A' attacks B. This implies that A cannot be in S, hence $S \subset \boldsymbol{G(AT)}$. Next, we prove that S is complete.

- $S \subset G(AT)$ and $G(AT)$ is conflict-free, so S **is conflict-free**.
- S **is an admissible set**: suppose, towards a contradiction, that there is some $D \in S$ such that some argument E attacks D and each argument attacking E is not in S. E attacks D, so there is an argument $D' \in \mathtt{sub}(D)$ such that $\mathtt{conc}(E) \in \overline{\mathtt{conc}(D')}$. Since $D \in G(AT)$, which is complete, there must be some argument in $Arg(AT)$ that attacks E, which means that $\mathtt{conc}(E) \notin \mathscr{K}$. Furthermore, since the contradiction function is symmetric, $\mathtt{conc}(D') \in \overline{\mathtt{conc}(E)}$. This implies that D' attacks E and therefore $D' \notin S$. However note that $D' \in G(AT)$: $D \in G(AT)$, so each argument attacking D is attacked by some argument in $G(AT)$; each argument attacking D' also attacks D and therefore must be attacked by some argument in $G(AT)$ so $D' \in G(AT)$. By definition of S, D' has a subargument that attacks B. But then D must have the same subargument attacking B, so $D \notin S$; a contradiction. As a consequence, S must be an admissible set.
- **Each argument that is acceptable w.r.t.** S **is in** S: suppose, towards a contradiction, that there exists an argument $D \in Arg(AT)$ that is acceptable w.r.t. S and $D \notin S$. If D is acceptable w.r.t. S, then D is acceptable w.r.t. $G(AT)$; therefore $D \in G(AT)$. $D \notin S$, so by definition of S there is a subargument $D' \in \mathtt{sub}(D)$ such that D' attacks B. Let $B' \in \mathtt{sub}(B)$ be the subargument on which D' attacks B: $\mathtt{conc}(D') \in \overline{\mathtt{conc}(B')}$. By an earlier assumption, B is not attacked by an observation-based argument, so $\mathtt{conc}(D') \notin \mathscr{K}$. Furthermore, by symmetry of contradiction, $\mathtt{conc}(B') \in \overline{\mathtt{conc}(D')}$, which means that B' attacks D' and therefore also attacks D. Since D is acceptable w.r.t. S, there must be an argument E in S attacking B'. But then E would attack B as well, hence $E \notin S$, a contradiction. As a result, each argument that is acceptable w.r.t. S is in S.

To conclude, there is a set $S \subset G(AT)$ such that S is complete. This contradicts our assumption that $G(AT)$ is the grounded extension since the grounded extension is minimal w.r.t. set inclusion. So if $A \in G(AT)$, then each argument attacking A is attacked by an observation-based argument.□

Lemma 7. (*Specification "out" arguments*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory with argumentation system $AS = (\mathscr{L}, \mathscr{R}, \text{-})$. An argument $A \in Arg(AT)$ is attacked by an argument in the grounded extension $G(AT)$ (A is "out"), iff A is attacked by an observation-based argument.*

Proof. **Right to left**: if an argument $A \in Arg(AT)$ is attacked by an observation-based argument B, then B cannot be attacked, hence must be in $G(AT)$. So A is attacked by an argument in the grounded extension.

Left to right: let $A \in Arg(AT)$ be an argument that is attacked by some $B \in G(AT)$ and suppose, towards a contradiction, that A is not attacked by any observation-based argument. Then there is a subargument $A' \in \mathtt{sub}(A)$ such that $\mathtt{conc}(B) \in \overline{\mathtt{conc}(A')}$ and therefore, by the symmetry of -, $\mathtt{conc}(A') \in \overline{\mathtt{conc}(B)}$, $\mathtt{conc}(A') \notin \mathscr{K}$ and $\mathtt{conc}(B) \notin \mathscr{K}$, hence A' attacks B. $B \in G(AT)$, so by Lemma 6, A' must be attacked by an observation-based

argument. However, this observation-based argument would attack A as well; a contradiction. To conclude, A is attacked by an observation-based argument.□

B3. Justification statuses are exclusive and complementary

Next, we give the full proof of Lemma 1, which was introduced in Section 2.3 of the main paper.

Lemma 1. *The statement justification statuses unsatisfiable, defended, out and blocked from Definition 8 are mutually exclusive and complementary.*

Proof. Consider an arbitrary argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $l \in \mathscr{L}$ be an arbitrary literal. We consider all possibilities.

- If there is an argument for l in $Arg(AT)$, then l cannot be unsatisfiable in AT.
  - If there is an argument for l in $G(AT)$, then l is defended and cannot be blocked in AT. This also means that l cannot be out in AT: that would require that each argument for l in $\mathscr{A}$ is attacked by an argument in $G(AT)$, which would contradict conflict-freeness of $G(AT)$.
  - Alternatively, there is an argument for l in $Arg(AT)$, but not in $G(AT)$. Then l cannot be defended in AT.
    - If there is an argument for l in $Arg(AT)$ but each argument for l in $Arg(AT)$ is attacked by an argument in the grounded extension $G(AT)$, then l is out and not blocked in AT.
    - Alternatively, there is an argument for l in $Arg(AT)$, but not in $G(AT)$ and at least one argument for l in $Arg(AT)$ is not attacked by any argument in $G(AT)$. Then l is blocked and cannot be out in AT.
- If there is no argument for l in $Arg(AT)$, then l is unsatisfiable and not defended, out or blocked in AT.

□

B4. Soundness, completeness and time complexity of STABILITY-LABEL

In Sections B.5–B.8, we prove the time complexity, soundness and conditional completeness of our algorithm STABILITY-LABEL.

We chose to present them here in a different order than in the paper: Proposition 4 on time complexity is proven before the soundness and conditional completeness propositions in the paper: in order to prove soundness, we need to show that STABILITY-LABEL terminates, which directly follows from the facts that the algorithm runs in polynomial time and the argumentation theory's language and rule set are finite. Therefore, we will prove Propositions 2–4 before Proposition 1.

B5. Time complexity of the stability algorithm

In this section, we specify the time complexity of STABILITY-LABEL in terms of its input (Proposition 4). Since the first step of STABILITY-LABEL is running the PREPROCESS algorithm, we start with the time complexity of this algorithm in a separate lemma.

Lemma 4. (*Time complexity PREPROCESS*) *The time complexity of PREPROCESS is* $\mathscr{O}(\left|\mathscr{L}\right|^2 + \left|\mathscr{L}\right| \cdot \left|\mathscr{R}\right|^2)$.

Proof. In the following proof, $\{c_1, \ldots, c_{14}\}$ is a set of positive constants.

- The PREPROCESS algorithm starts with a for loop in line 2–4, which is iterated $|\mathscr{L}|$ times. A single iteration of line 2 takes constant time c_1. A single iteration of line 3 requires for each literal $l \in \mathscr{C}$ a check that no contradictory literal $\bar{l}' \in \bar{l}$ is in $\mathscr{K}$. No literal can have more than $|\mathscr{L}|$ contradictories, so a single iteration of line 3 takes at most $c_2 \cdot |\mathscr{L}|$ operations. An iteration of line 4 can be done in constant time c_3. Then all iterations of line 2–4 require at most $(c_1 + |\mathscr{L}| \cdot c_2 + c_3) \cdot |\mathscr{L}|$ operations.
- lines 5 and 6 are repeated $|\mathscr{R}|$ times and take constant time per iteration, which means that these lines take $(c_4 + c_5) \cdot |\mathscr{R}|$ steps in total.
- Line 7 takes constant time c_6 and is executed just once.
- Next we consider the while-loop (lines 8–14), which iterates until no label changed in the previous loop. Thanks to the check $L_p[r] = \langle 1, 0, 0, 0 \rangle$ in line 11, each rule can change label at most once, hence the while-loop iterates at most $|\mathscr{R}|$ times.
  - Line 8 only requires a label check, which can be done in constant time c_7; the total time required for all iterations of line 8 is at most $c_7 \cdot |\mathscr{R}|$.
  - Similarly, the total time needed for all iterations of line 9 does not exceed $c_8 \cdot |\mathscr{R}|$.
  - The for-loop iterates $|\mathscr{R}|$ times for each iteration of the while-loop, so lines 10–13 are executed at most $|\mathscr{R}|^2$ times in total. A single execution of line 10 takes constant time c_9. Line 11 checks all antecedents of each rule, which requires at most $c_{10} \cdot |\mathscr{L}|$ checks per execution; a single execution of line 12, 13 and 14 take constant time $c_{11} + c_{12} + c_{13}$. So the total time required for all executions of lines 10–14 is at most $(c_9 + c_{10} \cdot |\mathscr{L}| + c_{11} + c_{12} + c_{13}) \cdot |\mathscr{R}|^2$.
- Finally, line 15 is executed just once and takes constant time c_{14}.

The total time required for Algorithm 1 is at most $(c_1 + |\mathscr{L}| \cdot c_2 + c_3) \cdot |\mathscr{L}| + (c_4 + c_5) \cdot |\mathscr{R}| + c_6 + (c_7 + c_8) \cdot |\mathscr{R}| + (c_9 + c_{10} \cdot |\mathscr{L}| + c_{11} + c_{12} + c_{13}) \cdot |\mathscr{R}|^2 + c_{14}$. As a result, the time complexity of the preprocessing step must be $\mathscr{O}(\left|\mathscr{L}\right|^2 + \left|\mathscr{L}\right| \cdot \left|\mathscr{R}\right|^2)$.□

Proposition 4. (*Time complexity STABILITY-LABEL*) *The time complexity of STABILITY-LABEL is* $\mathscr{O}(\left|\mathscr{L}\right|^3 \cdot \left|\mathscr{R}\right| + \left|\mathscr{L}\right|^2 \cdot \left|\mathscr{R}\right|^2)$.

Proof. We will prove this by first showing the amount of time that is required for a *single* execution of a given line (also given in the second column of the table below). Next, we will consider the number of iterations of each line (third column), multiply them to get the total time required for each line (fourth column) and combine this into the big-O notation (final row). In the following, we will denote positive constants by c_i (with $i \in [1..29]$).

Line	Time single execution	Max iterations	Total time
2	$(c_1 + \lvert\mathscr{L}\rvert \cdot c_2 + c_3) \cdot \lvert\mathscr{L}\rvert + (c_4 + c_5) \cdot \lvert\mathscr{R}\rvert + c_6 + (c_7 + c_8) \cdot \lvert\mathscr{R}\rvert + (c_9 + c_{10} \cdot \lvert\mathscr{L}\rvert + c_{11} + c_{12} + c_{13}) \cdot \lvert\mathscr{R}\rvert^2 + c_{14}$	1	$(c_1 + \lvert\mathscr{L}\rvert \cdot c_2 + c_3) \cdot \lvert\mathscr{L}\rvert + (c_4 + c_5) \cdot \lvert\mathscr{R}\rvert + c_6 + (c_7 + c_8) \cdot \lvert\mathscr{R}\rvert + (c_9 + c_{10} \cdot \lvert\mathscr{L}\rvert + c_{11} + c_{12} + c_{13}) \cdot \lvert\mathscr{R}\rvert^2 + c_{14}$
3	c_{15}	1	c_{15}
4	c_{16}	$\lvert\mathscr{L}\rvert$	$c_{16} \cdot \lvert\mathscr{L}\rvert$
5	$c_{17} \cdot (\lvert\mathscr{L}\rvert + \lvert\mathscr{R}\rvert)$	$\lvert\mathscr{L}\rvert$	$c_{17} \cdot (\lvert\mathscr{L}\rvert^2 + \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert)$
6	$c_{18} \cdot \lvert\mathscr{R}\rvert$	$\lvert\mathscr{L}\rvert$	$c_{18} \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$
7	c_{19}	$5 \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$	$5 c_{19} \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$
8	c_{20}	$5 \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$	$5 c_{20} \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$
9	$c_{21} \cdot \lvert\mathscr{L}\rvert$	$5 \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$	$5 c_{21} \cdot \lvert\mathscr{L}\rvert^2 \cdot \lvert\mathscr{R}\rvert$
10	c_{22}	$5 \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$	$5 c_{22} \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$
11	$c_{23} \cdot (\lvert\mathscr{L}\rvert + \lvert\mathscr{R}\rvert)$	$4 \cdot \lvert\mathscr{R}\rvert$	$4 c_{23} \cdot (\lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert + \lvert\mathscr{R}\rvert^2)$
12	c_{24}	$4 \cdot \lvert\mathscr{R}\rvert$	$4 c_{24} \cdot \lvert\mathscr{R}\rvert$
13	$c_{25} \cdot \lvert\mathscr{R}\rvert$	$4 \cdot \lvert\mathscr{L}\rvert$	$4 c_{25} \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$
14	c_{26}	$5 \cdot \lvert\mathscr{L}\rvert^2 \cdot \lvert\mathscr{R}\rvert$	$5 c_{26} \cdot \lvert\mathscr{L}\rvert^2 \cdot \lvert\mathscr{R}\rvert$
15	$c_{27} \cdot (\lvert\mathscr{L}\rvert + \lvert\mathscr{R}\rvert)$	$5 \cdot \lvert\mathscr{L}\rvert^2 \cdot \lvert\mathscr{R}\rvert$	$5 c_{27} \cdot (\lvert\mathscr{L}\rvert^3 \cdot \lvert\mathscr{R}\rvert + \lvert\mathscr{L}\rvert^2 \cdot \lvert\mathscr{R}\rvert^2)$
16	c_{28}	$5 \cdot \lvert\mathscr{L}\rvert^2 \cdot \lvert\mathscr{R}\rvert$	$5 c_{28} \cdot \lvert\mathscr{L}\rvert^2 \cdot \lvert\mathscr{R}\rvert$
17	$c_{29} \cdot \lvert\mathscr{R}\rvert$	$4 \cdot \lvert\mathscr{L}\rvert$	$4 c_{29} \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$
18	c_{30}	1	c_{30}
Total time required for all lines			$\mathscr{O}(\lvert\mathscr{L}\rvert^3 \cdot \lvert\mathscr{R}\rvert + \lvert\mathscr{L}\rvert^2 \cdot \lvert\mathscr{R}\rvert^2)$

We assume that for each literal $l \in \mathscr{L}$, the list of rules for that literal can be obtained in constant time, as well as the list of rules having that literal as an antecedent. As a result, checking if there are rules for this literal can be done in constant time, since we can check in constant time if the list of rules for the literal is empty. For any literal $l \in \mathscr{L}$, we can check in constant time if $l \in \mathscr{C}$.

First we show the amount of time that is required for a single execution of a given line.

- Line 2 requires running PREPROCESS$(\mathscr{L}, \mathscr{R}, _, \mathscr{C}, \mathscr{K})$, which takes $(c_1 + \lvert\mathscr{L}\rvert \cdot c_2 + c_3) \cdot \lvert\mathscr{L}\rvert + (c_4 + c_5) \cdot \lvert\mathscr{R}\rvert + c_6 + (c_7 + c_8) \cdot \lvert\mathscr{R}\rvert + (c_9 + c_{10} \cdot \lvert\mathscr{L}\rvert + c_{11} + c_{12} + c_{13}) \cdot \lvert\mathscr{R}\rvert^2 + c_{14}$ steps, as shown in Lemma 4.
- Line 3 takes constant time c_{15}.
- Line 4 takes a new literal l from $\mathscr{L}$; a single execution of this line requires a check if $l \in \mathscr{C}$, which, by assumption, takes constant time and, if $l \notin \mathscr{C}$, requires an additional check if there is a rule for l in $\mathscr{R}$, which takes constant time as well. To conclude, a single execution of line 4 takes constant time c_{16}.
- A single execution of line 5 requires labelling a literal, which can be done in $c_{17} \cdot (\lvert\mathscr{L}\rvert + \lvert\mathscr{R}\rvert)$ time: in the worst case, it requires checking the presence of l and all its contradictories ($\lvert\bar{l}\rvert \leq \lvert\mathscr{L}\rvert$) in $\mathscr{C}$ and $\mathscr{K}$, as well as the labels of all (max $\lvert\mathscr{R}\rvert$) rules for that literal or a contradictory. Therefore, line 5 requires $c_{17} \cdot (\lvert\mathscr{L}\rvert + \lvert\mathscr{R}\rvert)$ time per execution.
- In a single execution of line 6, all rules having l as an antecedent are added to TODO-SET. There are at most $\mathscr{R}$ rules having l as an antecedent, so this takes at most $c_{18} \cdot \lvert\mathscr{R}\rvert$ time.
- Line 7 only needs to check if a set is empty, which can be done in constant time c_{19}. The next line only needs to pop an element from a set, which can be done in constant time as well, so line 8 needs c_{20} time.
- Line 9 relabels a rule, which requires checking the labels of all antecedents of this rule. Since a rule has at most $\lvert\mathscr{L}\rvert$ antecedents, this takes at most $c_{21} \cdot \lvert\mathscr{L}\rvert$ time per execution of line 9.
- lines 10, 12 and 16 only check if the label of a rule or literal changed; this can be done in constant time c_{22}, c_{24} and c_{28}, respectively.
- lines 11 and 15 relabel a literal. As explained before (for line 5), this requires checking the presence of a literal and all its contradictories in $\mathscr{C}$ and $\mathscr{K}$, as well as checking the labels of all rules for that literal or one of its contradictories. A single execution therefore takes $c_{23} \cdot (\lvert\mathscr{L}\rvert + \lvert\mathscr{R}\rvert)$ time for line 11 and $c_{27} \cdot (\lvert\mathscr{L}\rvert + \lvert\mathscr{R}\rvert)$ time for line 15.
- Lines 13 and 17 both add at most $\lvert\mathscr{R}\rvert$ rules to TODO-SET, so a single execution of line 13 needs at most $c_{25} \cdot \lvert\mathscr{R}\rvert$ time and a single execution of line 17 needs at most $c_{29} \cdot \lvert\mathscr{R}\rvert$ time.
- A single execution of line 14 takes a literal from a list of contradictories, which can be done in constant time c_{26}.
- Finally, a single execution of line 18 takes constant time c_{30}.

Now we consider the number of iterations of each line; this is also represented in the third column of the table above.

- lines 2, 3 and 18 are executed just once.
- lines 4–6 are repeated for each literal in $\mathscr{C}$ or for which there is no rule in $\mathscr{R}$. This implies that lines 4–6 are executed at most $\lvert\mathscr{L}\rvert$ times.
- The lines 7–10 are executed once in every iteration of the while loop. The total number of iterations of the while loop equals the number of times a rule is added to TODO-SET. A rule is only added to TODO-SET if it was not yet visited (line 6) or if the label of one of its antecedents changed after a relabelling (line 13 or line 17). Since the label of a literal can change at most four times (i.e. at most four booleans can be turned to False), each rule $r \in \mathscr{R}$ is relabelled at most $5 \cdot \lvert\texttt{ants}(r)\rvert$ times. There are $\lvert\mathscr{R}\rvert$ rules, so lines 7–10 are executed at most $5 \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$ times. Besides, we will see in Lemma 9 that in practice, the maximum of changes is three, which means that these lines are executed at most $4 \cdot \lvert\mathscr{L}\rvert \cdot \lvert\mathscr{R}\rvert$ times.
- Next, we consider lines 11 and 12. These lines are only executed if the label of a rule changed. A label can only be changed by turning one of the four booleans to False. Therefore, for each rule, its label can be changed at most four times (again, we will see in Lemma 9 that in practice, the maximum of changes is three). There are $\lvert\mathscr{R}\rvert$ rules in total, so lines 11 and 12 are executed at most $4 \cdot \lvert\mathscr{R}\rvert$ times.
- Lines 13 and 17 are only executed just after the label of a literal changed. This can happen at most four times for each literal, because at most four booleans can be turned to False (again, we will see in Lemma 9 that the maximum of changes per label is in practice 3). There are $\lvert\mathscr{L}\rvert$ literals in total; therefore lines 13 and 17 are executed at most $4 \cdot \lvert\mathscr{L}\rvert$ times.

- Finally, lines 14–16 are executed at most $|\mathscr{L}|$ times (i.e. once for each $l' \in \overline{\texttt{conc}(r)}$) for each of the maximal $5{\cdot}|\mathscr{L}|{\cdot}|\mathscr{R}|$ iterations of the while loop. This means that lines 14–16 are executed at most $5{\cdot}|\mathscr{L}|^2{\cdot}|\mathscr{R}|$ times.

An upper bound on the total amount of time that is needed for all executions of a single line can now be obtained by multiplying the maximum time required for a single execution by the number of executions of each line. We do this in the fourth column of our table. From these results, it becomes clear that the total running time of Algorithm 4 is dominated by the lines for relabeling, in particular of line 15. To conclude, the total time complexity of Stability-label (Algorithm 4) is $\mathscr{O}(\left|\mathscr{L}\right|^3{\cdot}\left|\mathscr{R}\right| + \left|\mathscr{L}\right|^2{\cdot}\left|\mathscr{R}\right|^2)$.□

B6. Soundness of the preprocessing algorithm

In this section, we show that Preprocess (Algorithm 1) is sound: if a literal is labelled $\langle 1,0,0,0\rangle$, then there is no future argumentation theory in which there is an argument for that literal. Before we can prove this in Section B.6.2, we first need some additional definitions in Section B.6.1.

B6.1. Height of arguments

In this section, we introduce the notions of argument height and direct subarguments, since we will use them repeatedly in our induction proofs for the soundness of the preprocessing step (Section B.6.2) and the conditional completeness of the Stability-label algorithm (Section B.8). Intuitively, the height of an argument is the maximum number of inferences between a premise and the conclusion of the argument.

Definition 17. (*Height of arguments*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory and let $A \in Arg(AT)$ be an argument. The height $\texttt{h}(A)$ of A is:

- if A is an **observation-based argument** then $\texttt{h}(A) = 0$;
- if A is a **rule-based argument** of the form $A_1,\ldots,A_m \Rightarrow c$, then $\texttt{h}(A) = 1 + \max(\texttt{h}(A_1),\ldots,\texttt{h}(A_m))$.

Given a rule-based argument, the arguments for the antecedents of the rule on which the arguments is based are the direct subarguments.

Definition 18. (*Direct subarguments*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory and let $A \in Arg(AT)$ be an argument. The direct subarguments $\texttt{dirsub}(A)$ of A are:

- if A is an **observation-based argument** then $\texttt{dirsub}(A) = \varnothing$.
- if A is a **rule-based argument** of the form $A_1,\ldots,A_m \Rightarrow c$ then the set of direct subarguments $\texttt{dirsub}(A)$ of A is $\{A_1,\ldots,A_m\}$.

In the proofs that follow, we will repeatedly use the notion of argument height and direct subarguments to prove a certain property of an argument by induction, in the following way. As a base case, we prove the property for arguments with height of 0 or 1; consequently, we assume that the property holds for (direct) subarguments A' with $\texttt{h}(A') \leq k$ and prove the property for argument A with height $\texttt{h}(A) = k+1$. Note that Definition 4 on arguments enforces that all arguments have finite height, since the number of steps for constructing an argument is finite.

B6.2. Soundness proof

In the next lemma, we prove the soundness of the preprocessing step: if a literal l is labelled $L_p[l] = \langle 1,0,0,0\rangle$ by Preprocess, then l is stable-unsatisfiable in that argumentation theory with respect to the set of queryables. This property will also be useful in the soundness proofs for Stability-label in Section B.7.

Lemma 2. (*Soundness preprocessing step*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \neg)$ and let $\mathscr{C}$ be a set of queryables. Furthermore let L_p be the labelling obtained by Preprocess (Algorithm 1) on $\mathscr{L}$, $\mathscr{R}$, $\neg$, $\mathscr{C}$ and $\mathscr{K}$. For each $l \in \mathscr{L}$: if $L_p[l] = \langle 1,0,0,0\rangle$, then l is stable-unsatisfiable in AT w.r.t. $\mathscr{C}$.*

Proof. We prove this by contraposition. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \neg)$ and let $\mathscr{C}$ be a set of queryables. Further let $l \in \mathscr{L}$ be a literal and let L_p be the labelling after the preprocessing step.

Now suppose that l is not stable-unsatisfiable in AT w.r.t. $\mathscr{C}$. Then by Definition 11, there exists some argumentation theory $AT' = (AS, \mathscr{K}')$ in $F_{\mathscr{C}}(AT)$ such that there is an argument A for l in $Arg(AT')$. We now prove by induction on the height of A that l is labelled $\langle 1,1,1,1\rangle$ by Algorithm 1.

Base case: Suppose that $\texttt{h}(A) = 0$. Then $l \in \mathscr{K}'$. This implies that $l \in \mathscr{C}$ and for each $l' \in \bar{l}$: $l' \notin \mathscr{K}$, so l is labelled $\langle 1,1,1,1\rangle$ in Algorithm 1 line 3. Since there is no operation in Algorithm 1 that labels literals from $\langle 1,1,1,1\rangle$ to $\langle 1,0,0,0\rangle$, we have that $L_p[l] = \langle 1,1,1,1\rangle$.

Induction hypothesis: If A is an argument for l in $Arg(AT')$ and $\texttt{h}(A) \leq k$, then $L_p[l] = \langle 1,1,1,1\rangle$.

Induction step: Now suppose that $\texttt{h}(A) = k+1$. Then A must be based on some rule r and for each $A_i \in \texttt{dirsub}(A)$: A_i in $Arg(AT')$ and $\texttt{h}(A_i) \leq k$. By the induction hypothesis, for each $a \in \texttt{ants}(r)$: $L_p[a] = \langle 1,1,1,1\rangle$, so $L_p[l]$ is labelled $L_p[l] = \langle 1,1,1,1\rangle$ by Algorithm 1 line 13.

Since each argument A in $Arg(AT')$ has a finite $\texttt{h}(A)$, $L_p[l] = \langle 1,1,1,1\rangle$. As there is no operation that labels literals from $\langle 1,1,1,1\rangle$ to $\langle 1,0,0,0\rangle$, we have that $L_p[l] = \langle 1,1,1,1\rangle$ (so $L_p[l] \neq \langle 1,0,0,0\rangle$). By contraposition: if $L_p[l] = \langle 1,0,0,0\rangle$, then for each $AT' \in F_{\mathscr{C}}(AT)$: l is unsatisfiable in AT'. In other words: l is stable-unsatisfiable in AT w.r.t. $\mathscr{C}$.□

B7. Soundness of the stability algorithm

In this section, we show that Stability-label (Algorithm 4) is sound: if a literal $l \in \mathscr{L}$ is labelled stable in AT w.r.t. $\mathscr{C}$, then l is stable in AT w.r.t. $\mathscr{C}$. Before we can prove this in Section B.7.2, we first need some additional definitions in Section B.7.1.

B7.1. Interim labelling

In order to prove the soundness of the Stability-label algorithm, we will repeatedly use the notion of interim label, that is: the label at some point before the labelling is finished.

Definition 19. (*Interim labelling*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \neg)$ and let $\mathscr{C}$ be a set of queryables. Furthermore

let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For any literal or rule $x \in \mathscr{L} \cup \mathscr{R}$, we denote the **interim labelling** state of x immediately after the i'th iteration of the while loop as $L_i[x]$.

Given that the interim labelling L_i is the labelling state immediately after the i'th iteration of the while loop, L_0 is the labelling state just before the start of the first iteration of the while loop; to be precise: between line 6 and 7 of Algorithm 4. At this point, the preprocessing step has finished (line 2) and literals that are queryable or that are not the consequent of any rule are relabelled (line 5). Note that the labels of rules at this point are unchanged compared to the label obtained from preprocessing.

Remark 4. (*No rule label change between preprocessing and while loop*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, -)$ and let $\mathscr{C}$ be a set of queryables. Let L_p be the labelling obtained by Preprocess (Algorithm 1) and let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each rule $r \in \mathscr{R}$, $L_0[r] = L_p[r]$.

Proof. By Definition 19, $L_0[r]$ is the interim labelling state of r immediately after the 0'th iteration of the while loop. Between the end of the preprocessing step (after Algorithm 4 line 2) and the start of the while loop (from line 7), there is no operation that changes the labelling state of any rule. Therefore, for each rule $r \in \mathscr{R}$, $L_p[r] = L_0[r]$.□

Another new concept that is related to the interim labelling is the number of the iteration of the while loop in which one or more booleans from u, d, o and b is turned from True to False. We will repeatedly use this in our induction proofs; for example, if we know that the d-boolean of literal l was relabelled for the last time in iteration i, then we know that l is labelled $\neg L_i[l].d$, but also $\neg L_{i+1}[l].d$, $\neg L_{i+2}[l].d$, etc. This means for example that each rule r that has l as an antecedent will be considered for relabelling in some iteration $j > i$ and can be labelled $\neg L_j[r].d$ at that point, because we know that l is an antecedent of r and l is labelled $\neg L_{j-1}[l].d$.

Definition 20. (*Last boolean flip iteration*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, -)$ and let $\mathscr{C}$ be a set of queryables. Let $\mathscr{J} = 2^{\{u,d,o,b\}}$ be the set of all subsets of the labelling booleans u, d, o and b. For each $J \in \mathscr{J}$ and for each literal or rule $x \in \mathscr{L} \cup \mathscr{R}$, we denote by $c_J(x)$ the number of the iteration of the while loop in Algorithm 4 (executed on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$) in which the last boolean $j \in J$ was turned from True to False – provided that each boolean $j \in J$ is False in the final labelling L. In case there is some $j \in J$ such that j is True in the final labelling L, $c_J(x) = \infty$.

Example 16. (*Last boolean flip iteration*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, -)$. $\mathscr{L} = \{q, \neg q, a, \neg a\}$, $\mathscr{R} = \{q \Rightarrow a\}$ and - corresponds to classical negation. The set of queryables is $\{q, \neg q\}$ and the knowledge base is $\mathscr{K} = \{q\}$.

Then $L_p[q] = L_p[a] = \langle 1,1,1,1 \rangle$ and $L_p[\neg q] = L_p[\neg a] = \langle 1,0,0,0 \rangle$. Before the first iteration of the while loop (i.e. Algorithm 4 line 7–17), q and $\neg q$ are considered for relabelling. q's label is changed into $L_0[q] = \langle 0,1,0,0 \rangle$, while $L_0[\neg q] = L_p[\neg q] = \langle 1,0,0,0 \rangle$. No rule in $\mathscr{R}$ is relabelled between the preprocessing step and the start of the while-loop, so $L_0[q \Rightarrow a] = L_p[q \Rightarrow a] = \langle 1,1,1,1 \rangle$. Then the rule $q \Rightarrow a$ is added to todo-set. In the first iteration of the while loop, this rule is popped from the todo-set and relabelled so that $L_1[q \Rightarrow a] = \langle 0,1,0,0 \rangle$. Since $L_1[q \Rightarrow a] \neq L_0[q \Rightarrow a]$, the consequent a is relabelled, so that $L_1[a] = \langle 0,1,0,0 \rangle$. After that, $\neg a$ is considered for relabelling as well, but its label does not change: $L_1[\neg a] = \langle 1,0,0,0 \rangle$. At that point, the todo-set is empty and the algorithm terminates – which implies that for each $x \in \mathscr{L} \cup \mathscr{R} : L[x] = L_1[x]$.

$c_{\{u,o,b\}}(a) = 1$ because $L[a] = L_1[a] = \langle 0,1,0,0 \rangle$ and in $L_0[a]$ not all booleans in $\{u, o, b\}$ are turned false yet. $c_{\{d,o,b\}}(\neg a) = c_{\{d,o\}}(\neg a) = c_{\{d,b\}}(\neg a) = c_{\{o,b\}}(\neg a) = c_{\{d\}}(\neg a) = c_{\{o\}}(\neg a) = c_{\{b\}}(\neg a) = 0$: none of these (combinations of) booleans changes in any iteration of the while loop. Finally, note that $c_{\{d\}}(a) = \infty$, because a is labelled $L[a].d$ in the final labelling L.

B7.2. Soundness proof

We start this section by some general remarks that we will use throughout the proofs, sometimes implicitly.

First, note that Preprocess makes sure that all literals or rules have a true u-boolean, see the next remark. The reason for this is that all literals and rules initially get a true u label in line 3, 4 or 6, and there is no operation in Preprocess that turns these booleans to False.

Remark 5. (*Each u-boolean is True after Preprocess*) Given an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, -)$ and $\mathscr{C}$ is a set of queryables, let L_p be the labelling obtained by Preprocess (Algorithm 1) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each literal or rule in $x \in \mathscr{L} \cup \mathscr{R}$ the u-boolean in $L_p[x]$ is True.

Second, note that there is no operation before the first iteration of the while loop in Stability-label that changes the labels of rules, so these labels remain unaffected. From the previous remark and Remark 4, it directly follows that for each rule, its u-label before the first iteration of the while loop is True.

Remark 6. (*Each u boolean for rules is True before Stability-label line 7*) Given an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, -)$ and $\mathscr{C}$ is a set of queryables, let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each $r \in \mathscr{R} : L_0[r].u$ is True.

These remarks will be used to specify implications of a given labelling for (arguments in) future argumentation theories. For starters, an implication of the labelling $\neg L[r].u$ to a given rule r implies that there is an argument for that rule in the current argumentation theory.

Lemma 8. (*Argument existence labelling*) *Given an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, -)$ and $\mathscr{C}$ is a set of queryables, let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each $r \in \mathscr{R}$: if r is labelled $\neg L[r].u$ then there is an argument based on r in $Arg(AT)$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, -)$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by Algorithm 4 on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. We proceed by induction on $c_{\{u\}}$.

Proposition ($P(n)$): For each rule $r \in \mathscr{R}$ such that r is labelled $\neg L[r].u$ and $c_{\{u\}}(r) \leq n$, there is an argument based on r in $Arg(AT)$.

Base case ($P(1)$): Let r be an arbitrary rule in $\mathscr{R}$ such that $c_{\{u\}}(r) \leq 1$. Note that r cannot be labelled $\neg L[r].u$ by the Preprocess algorithm (see Remark 5). So this must have happened in the first iteration of the while loop ($c_{\{u\}}(r) = 1$) by Algorithm 4 (Stability-label) line 9, in which Algorithm 3 is applied. Then the fact that r is labelled $\neg L[r].u$ can only be caused by case R-U-a, so for each $a \in \texttt{ants}(r) : \neg L[a].u$ and $c_{\{u\}}(a) = 0$. Then each $a \in \texttt{ants}(r)$ must have been labelled $\neg L[a].u$ in Algorithm 4 line 5, by Algorithm 2. Case L-U-b does not apply (see Remark 6), so by case L-U-a, each $a \in \texttt{ants}(r)$ is in $\mathscr{K}$. Then there is an argument in $Arg(AT)$ for each $a \in \texttt{ants}(r)$; therefore, there is an argument based on r in $Arg(AT)$.

Induction hypothesis ($P(k)$): For each rule $r \in \mathscr{R}$ such that $c_{\{u\}}(r) \leq k$, there is an argument based on r in $Arg(AT)$.

Induction step ($P(k+1)$): Now let r be an arbitrary rule in $\mathscr{R}$ such that $c_{\{u\}}(r) = k+1$. Then r must have been relabelled to $\neg L[r].u$ by Algorithm 4 line 9, by case R-U-a: for each $a \in \texttt{ants}(r) : \neg L[a].u$ and $c_{\{u\}} \leq k$. Consider an arbitrary antecedent $a \in \texttt{ants}(r)$. Given that $\neg L[a].u$, either $a \in \mathscr{K}$ (case L-U-a) or for each r' for a: r' is labelled $\neg L[r'].u$ in or before the k'th iteration of the while loop (case L-U-b). If $a \in \mathscr{K}$, then there is an observation-based argument for a in $Arg(AT)$. Otherwise, by the induction hypothesis, there is an argument based on r', for a, in $Arg(AT)$. Because we picked a arbitrarily, for each $a \in \texttt{ants}(r)$, there is an argument for a in $Arg(AT)$. And hence, there is an argument based on r in $Arg(AT)$.

At this point, we have proven $P(n)$ for all natural numbers $n \in \mathbb{N}$. Since Algorithm 4 has a polynomial runtime (Proposition 4) in terms of finite $\mathscr{L}$ and $\mathscr{R}$, each $r \in \mathscr{R}$ such that $\neg L[r].u$ must have some non-negative $i < n$ such that $c_{\{u\}}(r) \leq i$ – hence there is an argument based on r in $Arg(AT)$.□

The following lemma guarantees that no literal or rule is labelled $\langle 0,0,0,0\rangle$ by Stability-label. This is an important property that we will use in many proofs related to soundness and conditional completeness, because it enables us to make statements like: "given that some rule r for l is labelled $\neg L[r].u$ and $\neg L[r].o$, it is impossible that each rule r for l is labelled $\neg L[r].d$ and $\neg L[r].b$".

Lemma 9. (*No 4x False label*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, let $\mathscr{C}$ be a set of queryables and let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each $x \in \mathscr{L} \cup \mathscr{R}$: $L[x] \neq \langle 0,0,0,0\rangle$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. We proceed by induction on the interim labelling of literals and rules.

Proposition ($P(n)$): For each $x \in \mathscr{L} \cup \mathscr{R}$ and for each non-negative integer $i < n$: $L_i[l] \neq \langle 0,0,0,0\rangle$.

Base case ($P(1)$): We make a distinction between rules and literals.

- Let $r \in \mathscr{R}$ be an arbitrary rule. There are two options: $L_p[r] = \langle 1,0,0,0\rangle$ or $L_p[r] = \langle 1,1,1,1\rangle$.
  - First suppose that $L_p[r] = \langle 1,0,0,0\rangle$. This implies that there is an antecedent $a \in \texttt{ants}(r)$ such that $L_p[a] = \langle 1,0,0,0\rangle$: otherwise, the L_p label of r would have been changed to $\langle 1,1,1,1\rangle$ by Algorithm 1 line 12. So by Lemma 2, there is an antecedent $a \in \texttt{ants}(r)$ such that for each $AT' \in F_{\mathscr{C}}(AT)$: a is unsatisfiable in AT'. Consequently, there is no argument for a in $Arg(AT)$, so there is no argument based on r in $Arg(AT)$. Then, by Lemma 8, r is not labelled $\neg L[r].u$ and hence $\boldsymbol{L_0}[r] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.
  - Alternatively, suppose that $L_p[r] = \langle 1,1,1,1\rangle$. Then, by Remark 4, $L_0[r] = L_p[r] = \langle 1,1,1,1\rangle$, which means that $\boldsymbol{L_0}[r] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.

  Recall that we picked r as an arbitrary rule in $\mathscr{R}$. Therefore, **for each** $\boldsymbol{r \in \mathcal{R}}$: $\boldsymbol{L_0}[r] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.
- Now suppose that l is an arbitrary literal. Again, we consider both $L_p[l] = \langle 1,0,0,0\rangle$ and $L_p[l] = \langle 1,1,1,1\rangle$.
  - First suppose that $L_p[l] = \langle 1,0,0,0\rangle$. Then $l \notin \mathscr{K}$: if $l \in \mathscr{K}$, there would be an argument for l in $Arg(AT)$, which would imply that l is not stable-unsatisfiable in AT (Definitions 10 and 11), a contradiction to the soundness of Preprocess (Lemma 2). Suppose, towards a contradiction, that $L_0 = \langle 0,0,0,0\rangle$. By Algorithm 2 case L-U-b, there is a rule r for l that is labelled $\neg L[r].u$ before the start of the while loop. This however contradicts Remark 6. Therefore, $\boldsymbol{L_0}[l] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.
  - Alternatively, $L_p[l] = \langle 1,1,1,1\rangle$. We consider two cases.
    - First assume that $l \in \mathscr{C}$ and for each $l' \in \bar{l} : l' \notin \mathscr{K}$. This implies that l is considered for relabelling in Algorithm 4 line 5. In this relabelling step, none of the labelling rules L-D-a, L-D-b or L-D-c from Algorithm 2 applies, so l is labelled $L[l].d$. This means that $\boldsymbol{L_0}[l] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.
    - Alternatively, either $l \notin \mathscr{C}$ or there is an $l' \in \bar{l}$ such that $l' \in \mathscr{K}$. In both cases, the fact that $L_p[l] = \langle 1,1,1,1\rangle$ must have been caused by the fact that there is a rule r for l such that $L_p[r] = \langle 1,1,1,1\rangle$ (Algorithm 4 line 9). Rules are not relabelled before the first iteration of the while loop (Remark 4), hence there is a rule r for l such that $L_0[r] = L_p[r] = \langle 1,1,1,1\rangle$. As a result, neither L-U-a nor L-U-b from Algorithm 2 applies; therefore l is labelled $L[l].u$. So $\boldsymbol{L_0}[l] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.

    Since we chose l arbitrarily from $\mathscr{L}$, we conclude: **for each** $\boldsymbol{l \in \mathcal{L}}$: $\boldsymbol{L_0}[l] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.

Induction hypothesis ($P(k)$): For each $x \in \mathscr{L} \cup \mathscr{R}$ and for each non-negative integer $i < k$: $L_i \neq \langle 0,0,0,0\rangle$.

Induction step ($P(k+1)$): We will show that for each $x \in \mathscr{L} \cup \mathscr{R}$: $L_k[x] \neq \langle 0,0,0,0\rangle$.

- Let $r \in \mathscr{R}$ be an arbitrary rule. Suppose, towards a contradiction, that $L_k[r] = \langle 0,0,0,0\rangle$. Then $\neg L[r].d$, which must be caused by Algorithm 3 case R-D-a, so there is an antecedent $a \in \texttt{ants}(r)$ that is labelled $\neg L[a].d$. Furthermore, $\neg L[r].b$ can be caused by either R-B-a or R-B-b, but in both cases, there is an antecedent $a \in \texttt{ants}(r)$ such that $\neg L[a].d$ and $\neg L[a].b$. Since $\neg L[r].o$ must be caused by R-O-a, there is an antecedent $a \in \texttt{ants}(r)$ such that $\neg L[a].d$ and $\neg L[a].o$ and $\neg L[a].b$. Finally, $\neg L[r].u$ implies that $\neg L[a].u$ for each antecedent a of r. But then there is an antecedent $a \in \texttt{ants}(r)$ such that $L_{k-1}[a] = \langle 0,0,0,0\rangle$. This contradicts the induction hypothesis, **there is no rule** $\boldsymbol{r \in \mathcal{R}}$ **such that** $\boldsymbol{L_k}[r] = \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.
- Now let $l \in \mathscr{L}$ be an arbitrary literal. Suppose, towards a contradiction, that $L_k[l] = \langle 0,0,0,0\rangle$.

  First we show that $l \notin \mathscr{C}$: suppose that $l \in \mathscr{C}$. This implies that $\neg L[l].d$ must have been caused by Algorithm 2 case L-D-a, hence there is an $l' \in \bar{l}$ that is in $\mathscr{K}$, which by consistency of $\mathscr{K}$ implies that $l \notin \mathscr{K}$, so $\neg L[l].u$ must have been caused by L-U-b. Furthermore, there is some $l' \in \bar{l}$ such that no $l'' \in \overline{l'}$ is in $\mathscr{K}$, so $\neg L[l].o$ must have been caused by L-O-f. But that implies that there exists a rule r for l such that $L_k[r] = \langle 0,0,0,0\rangle$, which contradicts our earlier conclusion that such a rule does not exist. Therefore $\boldsymbol{l \notin \mathcal{Q}}$.

  Given that $l \notin \mathscr{C}$, $\neg L[l].u$ must have been caused by case L-U-b. Then the fact that l is labelled $\neg L[l].o$ must be caused by either L-O-d or L-O-e; in both cases, there is a rule r for l with $\neg L[r].u$ and $\neg L[r].o$. Since there is no rule labelled $\langle 0,0,0,0\rangle$ by L_k, $\neg L[l].b$ must have been caused by L-B-c or L-B-d. In both cases, there is a rule r for l with $\neg L[r].u$, $\neg L[r].o$ and $\neg L[r].b$ and for each $l' \in \bar{l}$: for each rule r' for l': $\neg L[r'].d$ and $\neg L[r'].b$. Finally, $\neg L[l].d$ can be caused by either L-D-b or L-D-c. However, both cases contradict our earlier conclusion that there is no rule labelled $\langle 0,0,0,0\rangle$ by L_k. So **for each** $\boldsymbol{l \in \mathcal{L}}$: $\boldsymbol{L_k}[l] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.

At this point we have proven our proposition $P(n)$ for all natural numbers $n \in \mathbb{N}$: for each $x \in \mathscr{L} \cup \mathscr{R}$ and for each non-negative integer $i < n$: $L_i[x] \neq \langle 0,0,0,0\rangle$. Given that Algorithm 4 terminates (see Proposition 4), there is some finite i such that $L_i[x] = L[x]$ for all $x \in \mathscr{L} \cup \mathscr{R}$. Therefore **for each** $\boldsymbol{x \in \mathcal{L} \cup \mathcal{R}}$: $\mathcal{L}[x] \neq \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0}\rangle$.□

The next lemma shows that any argument in the current argumentation theory is retained in each future argumentation theory. We will repeatedly use this lemma in our soundness proofs.

Lemma 10. (*Argument persistence in future argumentation theories*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory with argumentation system $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let A be an argument in $Arg(AT)$. Then for each set of queryables $\mathscr{Q}$, for each $AT' \in F_{\mathscr{Q}}(AT)$: $A \in Arg(AT')$.*

Proof. Suppose that $AT = (AS, \mathscr{K})$ is an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{Q}$ be a set of queryables. Let $AT' = (AS, \mathscr{K}')$ be an arbitrary future argumentation theory in $F_{\mathscr{Q}}(AT)$. By Definition 10 of future argumentation theories, $\mathscr{K} \subseteq \mathscr{K}'$. From the definition of arguments (Definition 4), it follows that arguments are constructed only based on their knowledge base and rule set, so given that A can be constructed from $\mathscr{K}$ and $\mathscr{R}$, A can also be constructed from $\mathscr{K}'$ and $\mathscr{R}$. This implies that $A \in Arg(AT')$. Since AT' is an arbitrary argumentation theory in $F_{\mathscr{Q}}(AT)$, this generalises to all future argumentation theories: for each $AT' \in F_{\mathscr{Q}}(AT)$: $A \in Arg(AT')$.□

Next, we discuss the situations in which the booleans representing the stable-unsatisfiable, stable-defended, stable-out and stable-blocked statuses for rules are turned to False. Due to the way the labelling rules are defined, there are many situations in which either the *u*- and *o*-boolean or the *d*- and *b*-boolean are turned False together. Lemma 11 analyses the situation that both the unsatisfiable and the out boolean for a rule are turned to False. In this situation, there must be some argument based on that rule in each future argumentation theory AT' that is not attacked *on a subargument* by an argument in the grounded extension $G(AT')$. In the next definition, we formally define attacks on a subargument and their counterpart: attacks on a conclusion.

Definition 21. (*Attack on conclusion or on a subargument*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let A and B be two arguments in $Arg(AT)$. If A attacks B on B, we say that A attacks B **on its conclusion**; otherwise (A attacks B on B' and $B' \neq B$), we say that A attacks B **on a subargument**.

Using Definition 21 we can now prove Lemma 11:

Lemma 11. (*Labelled not unsatisfiable or out*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and $\mathscr{Q}$ is a set of queryables; furthermore let L be the labelling obtained by* Stability-label *(Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. For each $r \in \mathscr{R}$: if r is labelled $\neg L[r].u$ and $\neg L[r].o$, then for each $AT' \in F_{\mathscr{Q}}(AT)$, there is an argument based on r in $Arg(AT')$ that is not attacked on a subargument by any argument in $G(AT')$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{Q}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. We proceed by induction on $c_{\{u,o\}}$.

Proposition ($P(n)$): For each rule $r \in \mathscr{R}$ such that $c_{\{u,o\}}(r) \leq n$: if r is labelled $\neg L[r].u$ and $\neg L[r].o$, then for each $AT' \in F_{\mathscr{Q}}(AT)$, there is an argument based on r in $Arg(AT')$ that is not attacked on a subargument by any argument in $G(AT')$.

Base case ($P(1)$): Let r be an arbitrary rule in $\mathscr{R}$ such that $c_{\{u,o\}}(r) \leq 1$ and $\neg L[r].u$ and $\neg L[r].o$. Since $\neg L[r].u$ can neither be assigned by Preprocess nor before the first iteration of the while loop (Remark 6), $c_{\{u,o\}}(r) = 1$. Then $\neg L[r].u$ and $\neg L[r].o$ must have been caused by Algorithm 3 case R-U-a and R-O-a: each $a \in \texttt{ants}(r)$ is labelled $\neg L[a].u$ and $\neg L[a].o$ and $c_{\{u,o\}}(a) = 0$. Then each $a \in \texttt{ants}(r)$ is in $\mathscr{K}$: the $\neg L[a].u$ label before the first iteration of the while loop cannot be caused by Algorithm 2 case L-U-b (see Remark 6), and must hence have been caused by case L-U-a. Consequently, there is an observation-based argument for each of r's antecedents, hence there is an argument A based on r in $Arg(AT)$. Then, by Definition 5 and Lemma 10, for each $AT' \in F_{\mathscr{Q}}(AT)$, A is not attacked on a subargument by any argument in $G(AT')$.

Induction hypothesis ($P(k)$): For each $r \in \mathscr{R}$ such that $c_{\{u,o\}}(r) \leq k$: if r is labelled $\neg L[r].u$ and $\neg L[r].o$, then for each $AT' \in F_{\mathscr{Q}}(AT)$, there is an argument based on r in $Arg(AT')$ that is not attacked on a subargument by any argument in $G(AT')$.

Induction step ($P(k+1)$): Now let r be an arbitrary rule in $\mathscr{R}$ such that $c_{\{u,o\}}(r) \leq k+1$ and $\neg L[r].u$ and $\neg L[r].o$. By Algorithm 3 case R-U-a and R-O-a, for each $a \in \texttt{ants}(r)$: $\neg L[a].u$, $\neg L[a].o$ and $c_{\{u,o\}}(a) \leq k$. Now let a be an arbitrary antecedent in $\texttt{ants}(r)$. We consider two cases:

- If $a \in \mathscr{K}$, then for each $AT' \in F_{\mathscr{Q}}(AT)$ there is an observation-based argument for a in $Arg(AT')$ that, by Lemma 7, is not attacked by any argument in $G(AT')$.
- Alternatively, $a \notin \mathscr{K}$. Then the label $\neg L[a].u$ must have been labelled by case L-U-b. This implies that the label $\neg L[a].o$ must have been by case L-O-b, L-O-c, L-O-d or L-O-e. In any case: (1) there is a rule r' for a that is labelled $\neg L[r'].u$ and $\neg L[r'].o$ and (2) if $a \in \mathscr{Q}$ then for each $a' \in \overline{a}$ there is some $a'' \in \overline{a}$ that is in $\mathscr{K}$. Given that $c_{\{u,o\}}(a) \leq k$, it follows that $c_{\{u,o\}}(r') \leq k$. So by the induction hypothesis and (1): for each $AT' \in F_{\mathscr{Q}}(AT)$, there is an argument for a (based on r') in $Arg(AT')$ that is not attacked *on a subargument* by an argument in $G(AT')$. Furthermore, by (2) there is no $AT' \in F_{\mathscr{Q}}(AT)$ such that there is an observation-based argument for any $a' \in \overline{a}$ in $Arg(AT')$, so by Lemma 7 no argument for a in $Arg(AT')$ for any $AT' \in F_{\mathscr{Q}}(AT)$ can be attacked *on its conclusion* by an argument in $G(AT')$.

Given that for each $AT' \in F_{\mathscr{Q}}(AT)$, for each $a \in \texttt{ants}(r)$, there is an argument for a in $Arg(AT')$ that is not attacked by any argument in $G(AT')$, we can construct an argument based on r in $Arg(AT')$ that is not attacked *on a subargument* by any argument in $G(AT')$.

Finally, recall that $c_{\{u,o\}}(r)$ is finite for each $r \in \mathscr{R}$ that is labelled $\neg L[r].u$ and $\neg L[r].o$ (Definition 20), which means that the proposition is valid for each $r \in \mathscr{R}$ in general.□

Lemma 12 analyses the situation that a rule $r \in \mathscr{R}$ for l is labelled $\neg L[r].d$ and $\neg L[r].b$ and shows that l cannot be defended or blocked in any future argumentation theory of *AT thanks to that rule*, that is: there is no future argumentation theory AT' in which there is an argument based on r that is not attacked by an argument $G(AT)$.

Lemma 12. (*Labelled not defended or blocked*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{Q}$ be a set of queryables. Let L be the labelling obtained by* Stability-label *(Algorithm 4) and L_p be the labelling obtained by* Preprocess *(Algorithm 1) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. If $r \in \mathscr{R}$ such that $L_p[r] = \langle 1,1,1,1 \rangle$ and r is labelled $\neg L[r].d$ and $\neg L[r].b$, then for each $AT' \in F_{\mathscr{Q}}(AT)$, any argument based on r is attacked by an argument in $G(AT')$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{Q}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) and L_p be the labelling obtained by Preprocess (Algorithm 1) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. We proceed by induction on $c_{\{d,b\}}$.

Proposition ($P(n)$): For each $r \in \mathscr{R}$ such that $c_{\{d,b\}}(r) \leq n$: if $L_p[r] = \langle 1,1,1,1 \rangle$ and r is labelled $\neg L[r].d$ and $\neg L[r].b$, then for each $AT' \in F_{\mathscr{Q}}(AT)$, each

argument based on r in $Arg(AT')$ is attacked by an argument in $G(AT')$.

Base case ($P(1)$): Let r be an arbitrary rule such that $c_{\{d,b\}}(r) \leq 1$ and $L_p[r] = \langle 1,1,1,1\rangle$ and suppose that r is labelled $\neg L[r].d$ and $\neg L[r].b$. Since $L_p[r] = \langle 1,1,1,1\rangle$, r must be labelled $\neg L[r].d$ and $\neg L[r].b$ by Algorithm 4 and by Remark 4 $\boldsymbol{c_{\{d,b\}}(r) = 1}$.

The labelling of r as $\neg L[r].d$ and $\neg L[r].b$ must have happened in line 9, caused by Algorithm 3. Then by case R-D-a and case R-B-a or R-B-b, some $a \in \texttt{ants}(r)$ must have been labelled $\neg \boldsymbol{L}[a].\boldsymbol{d}$ and $\neg \boldsymbol{L}[a].\boldsymbol{b}$ and $\boldsymbol{c_{\{d,b\}}(a) = 0}$.

Given that $c_{\{d,b\}}(a) = 0$, a must have been relabelled either in the preprocessing step or in Algorithm 4 line 5. However, if a was labelled $\neg L[a].d$ and $\neg L[a].b$ in the preprocessing step, then $L_p[a] = \langle 1,0,0,0\rangle$, which means that $L_p[r] = \langle 1,0,0,0\rangle$ (since Algorithm 1 line 11 would not have applied for r), which would contradict our assumption that $L_p[r] = \langle 1,1,1,1\rangle$. So a must have been labelled $\neg L[a].d$ and $\neg L[a].b$ in Algorithm 4 line 5. Then the condition in line 4 must have been true: either $a \in \mathscr{C}$ or there is no rule for a in $\mathscr{R}$.

Next, we show by contradiction that $a \in \mathscr{C}$: suppose that $a \notin \mathscr{C}$. Then by the condition in Algorithm 4 line 4, there is no rule for a in $\mathscr{R}$. But in that case, a would have been labelled as $L_p[a] = \langle 1,0,0,0\rangle$ in the preprocessing step (line 4) and never relabelled to $\langle 1,1,1,1\rangle$ because the condition of line 7 would never apply (given that there is no rule for a in $\mathscr{R}$). This contradicts our earlier conclusion that $L_p[a] = \langle 1,1,1,1\rangle$. Therefore, $a \in \mathscr{C}$.

Given that $a \in \mathscr{C}$ and a is relabelled in Algorithm 4 line 5, by case L-D-a, there is some $a' \in \overline{a}$ such that $a' \in \mathscr{K}$. Then there is an observation-based argument for a' in $Arg(AT)$ – and by Lemma 10, for each $AT' \in F_{\mathscr{C}}(AT)$: $a' \in Arg(AT')$. So by Lemma 7, for each $AT' \in F_{\mathscr{C}}(AT)$, each argument for a in $Arg(AT')$ is attacked by an argument in $G(AT')$.

Given that a is an antecedent of r, we can derive that **for each $\boldsymbol{AT'}$ in $\boldsymbol{F_{\mathcal{Q}}(AT)}$, each argument based on $\boldsymbol{r}$ in $\boldsymbol{Arg(AT')}$ is attacked by an argument in $\boldsymbol{G(AT')}$**. Since we chose r as an arbitrary rule in $\mathscr{R}$ such that $c_{\{d,b\}}(r) \leq 1$, we can generalise this to all rules that are labelled $\langle 1,1,1,1\rangle$ by L_p.

Induction hypothesis ($P(k)$): For each $r \in \mathscr{R}$ such that $c_{\{d,b\}}(r) \leq k$, if $L_p[r] = \langle 1,1,1,1\rangle$ and r is labelled $\neg L[r].d$ and $\neg L[r].b$ then for each $AT' \in F_{\mathscr{C}}(AT)$, each argument based on $r \in Arg(AT')$ is attacked by an argument in $G(AT')$.

Induction step ($P(k+1)$): Let r be an arbitrary rule in $\mathscr{R}$ such that $c_{\{d,b\}}(r) = k+1$ and assume that $L_p[r] = \langle 1,1,1,1\rangle$ and r is labelled $\neg L[r].d$ and $\neg L[r].b$. By Algorithm 3 case R-D-a and either R-B-a or R-B-b, there is an antecedent a of r that is labelled $\neg L[a].d$ and $\neg L[a].b$ and $c_{\{d,b\}}(a) \leq k$. We distinguish two cases: $a \in \mathscr{C}$ and $a \notin \mathscr{C}$.

- If $a \in \mathscr{C}$, then by case L-D-a there is some $a' \in \overline{a}$ such that $a' \in \mathscr{K}$. So there is an observation-based argument A for a' that is in $Arg(AT')$ for every $AT' \in F_{\mathscr{C}}(AT)$. Being observation-based, A cannot be attacked and therefore $A \in G(AT')$ for every $AT' \in F_{\mathscr{C}}(AT)$. So **for each $\boldsymbol{AT'} \in \boldsymbol{F_{\mathcal{Q}}(AT)}$: each argument for $\boldsymbol{a}$ in $\boldsymbol{Arg(AT')}$ is attacked by an argument in $\boldsymbol{G(AT')}$**.
- Now suppose that $a \notin \mathscr{C}$. Then by the labelling rules of Algorithm 3, **each rule $\boldsymbol{r'}$ for $\boldsymbol{a}$ is labelled $\boldsymbol{\neg L[r'].d}$ and $\boldsymbol{\neg L[r'].b}$ and $\boldsymbol{c_{\{d,b\}}(r') \leq k}$**: if L-D-c caused $\neg L[a].d$, then by Lemma 9, a must have been labelled $\neg L[a].b$ by case L-B-b; if, alternatively, L-D-b caused $\neg L[a].d$, then either L-B-b or L-B-c caused $\neg L[a].b$.

  Consider an arbitrary rule r' for a. Either $L_p[r'] = \langle 1,0,0,0\rangle$ or $L_p[r'] = \langle 1,1,1,1\rangle$. If $L_p[r'] = \langle 1,0,0,0\rangle$, some $a' \in \texttt{ants}(r')$ is labelled $L_p[a'] = \langle 1,0,0,0\rangle$, so by Lemma 2 there is no argument based on r' in any $Arg(AT')$ where $AT' \in F_{\mathscr{C}}(AT)$. If, alternatively, $L_p[r'] = \langle 1,1,1,1\rangle$ (which must be the case for at least one rule for a, since $L_p[a] = \langle 1,1,1,1\rangle$), we apply the induction hypothesis: for each $AT' \in F_{\mathscr{C}}(AT)$: each argument based on r' in $Arg(AT')$ is attacked by an argument in $G(AT')$. Since $\boldsymbol{a \notin \mathcal{Q}}$, there are no observation-based arguments for a either, so: **for each $\boldsymbol{AT'} \in \boldsymbol{F_{\mathcal{Q}}(AT)}$, each argument for $\boldsymbol{a}$ in $\boldsymbol{Arg(AT')}$ is attacked by an argument in $\boldsymbol{G(AT')}$**.

By generalising r, we can now derive that for each rule $r \in \mathscr{R}$ such that $c_{\{d,b\}} \leq k+1$: if r is labelled $\neg L[r].d$ and $\neg L[r].b$, then for each $AT' \in F_{\mathscr{C}}(AT)$, each argument based on r in $Arg(AT')$ is attacked by an argument in $G(AT')$. Finally, remember that for each rule $r \in \mathscr{R}$ that is labelled $\neg L[r].d$ and $\neg L[r].b$, $c_{\{d,b\}}(r)$ is finite (Definition 20), which means that the proposition is valid for each $r \in \mathscr{R}$ in general.□

In the next lemma, we show that no literal $l \in \mathscr{L}$ can be labelled $L[l] = \langle 1,0,0,0\rangle$ *after* the preprocessing step, that is: if $L_p[l] \neq \langle 1,0,0,0\rangle$. We will use this lemma for proving soundness of stable-unsatisfiable labelling.

Lemma 13. (*Stable-unsatisfiable labelling only in preprocessing*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, let $\mathscr{C}$ be a set of queryable literals and let L be the labelling obtained by* Stability-label *(Algorithm 4) and L_p the labelling obtained by* Preprocess *(Algorithm 1) on $\mathscr{L}$, $\mathscr{R}$, $\neg$, $\mathscr{C}$ and $\mathscr{K}$. For each literal $l \in \mathscr{L}$: if $L[l] = \langle 1,0,0,0\rangle$ then $L_p[l] = \langle 1,0,0,0\rangle$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. We will, towards a proof by contraposition, first show that for each $l \in \mathscr{L}$ such that $L_p[l] \neq \langle 1,0,0,0\rangle$ it must be that $L[l] \neq \langle 1,0,0,0\rangle$. We prove this by induction on the interim labelling of literals.

Proposition $P(n)$: For each $l \in \mathscr{L}$: if $L_p[l] = \langle 1,1,1,1\rangle$ then for each non-negative integer $i < n$: $L_i[l] \neq \langle 1,0,0,0\rangle$.

Base case ($P(1)$): Let $l \in \mathscr{L}$ be an arbitrary literal such that $L_p[l] = \langle 1,1,1,1\rangle$ and suppose that $L_0[l] = \langle 1,0,0,0\rangle$.

Then l must have been relabelled by Algorithm 4 line 5, using Algorithm 2. Given that l is labelled $\neg L[l].d$, either $l \notin \mathscr{C}$ or there is some $l' \in \overline{l}$ such that $l' \in \mathscr{K}$. Therefore, $L_p[l] = \langle 1,1,1,1\rangle$ cannot be caused by Algorithm 1 line 3 and hence must be caused by Algorithm 1 line 13. This means that Algorithm 1 line 12 must have been executed as well, so there is some rule r for l such that $L_p[r] = \langle 1,1,1,1\rangle$. Then by Remark 4 **there is some rule $\boldsymbol{r}$ for $\boldsymbol{l}$ such that $\boldsymbol{L_0[r] = \langle 1,1,1,1\rangle}$**.

Furthermore, given that l is relabelled by Algorithm 4 line 5, the condition in Algorithm 4 line 4 must have applied, which implies that $l \in \mathscr{C}$. Consequently, there is an $l' \in \overline{l}$ such that $l' \in \mathscr{K}$. But then no labelling rule in Algorithm 2 derives $\neg L[l].o$. This contradicts our assumption that $L_0[l] = \langle 1,0,0,0\rangle$. Therefore, **for each $\boldsymbol{l \in \mathcal{L}}$ such that $\boldsymbol{L_p[l] = \langle 1,1,1,1\rangle}$, for each non-negative integer $\boldsymbol{i < 1}$: $\boldsymbol{L_i[l] \neq \langle 1,0,0,0\rangle}$**.

Induction hypothesis ($P(k)$): For each $l \in \mathscr{L}$: if $L_p[l] = \langle 1,1,1,1\rangle$ then for each non-negative integer $i < k$: $L_i[l] \neq \langle 1,0,0,0\rangle$.

Induction step ($P(k+1)$): Let l be an arbitrary literal such that $L_p[l] = \langle 1,1,1,1\rangle$. Suppose, towards a contradiction, that $L_k[l] = \langle 1,0,0,0\rangle$. Then each rule r for l is labelled $L_i[r] = \langle 1,0,0,0\rangle$, as we will show next.

- First suppose that $l \in \mathscr{C}$. Then there is some $l' \in \bar{l}$ such that $l' \in \mathscr{K}$ by case L-D-a, so $l \notin \mathscr{K}$. Then case L-O-f must have applied, so **each rule r for l is labelled $L_k[r] = \langle 1,0,0,0\rangle$**.
- Alternatively, suppose that $l \notin \mathscr{C}$. We next show by contradiction that each rule r for l is labelled $\neg L[r].d$: suppose that there is some rule r for l that is labelled $L[r].d$. Then the fact that l is labelled $\neg L[l].d$ must have been caused by case L-D-c and, by Lemma 9, $\neg L[l].b$ must be caused by case L-B-b: each rule r for l is labelled $\neg L[r].d$ and $\neg L[r].b$. But that contradicts our assumption, so each rule r for l is labelled $\neg L[r].d$.

  Given that each rule r for l is labelled $\neg L[r].d$, the fact that l is labelled $\neg L[l].b$ must be caused by case L-B-b or L-B-c; as a result, the fact that l is labelled $\neg L[l].o$ must be caused by case L-O-f or L-O-d. In both cases, by Lemma 9, **each rule r for l is labelled $L_k[r] = \langle 1,0,0,0\rangle$**.

Now let r be an arbitrary rule for l; we know that $L_k[r] = \langle 1,0,0,0\rangle$. The fact that r is labelled $\neg L[r].d$, $\neg L[r].o$ and $\neg L[r].b$ must have been caused by Algorithm 3 case R-D-a, case R-B-a or R-B-b and case R-O-a: there is some $a \in \texttt{ants}(r)$ that is labelled $\neg L[a].d$ and $\neg L[a].o$ and $\neg L[a].b$. Note that a must have been labelled as such before the $(k-1)$'th iteration of the while loop. By Lemma 9, $L_{k-1}[a] = \langle 1,0,0,0\rangle$.

But then by the induction hypothesis, a was already labelled stable-unsatisfiable by L_p. Given that $L_p[a] = \langle 1,0,0,0\rangle$, the condition in Algorithm 1 line 11 would not have applied for r, so r would have been labelled $L_p[r] = \langle 1,0,0,0\rangle$. Since we chose r arbitrary, we know that $L_p[r] = \langle 1,0,0,0\rangle$ for each rule r for l. But that implies that l is labelled stable-unsatisfiable in Algorithm 1 line 4 and is not re-labelled to $\langle 1,1,1,1\rangle$ in Algorithm 1 line 13. So $L_p[l] = \langle 1,0,0,0\rangle$. This contradicts our assumption that $L_p[l] = \langle 1,1,1,1\rangle$. Hence **for each $l \in \mathcal{L}$ such that $L_p[l] = \langle 1,1,1,1\rangle$, for each non-negative integer $i < k+1$: $L_i[l] \neq \langle 1,0,0,0\rangle$**.

At this point, we have proven $P(i)$ for each finite non-negative $i < n$ and for each $l \in \mathscr{L}$. Since Algorithm 4 has a polynomial runtime (Proposition 4), there must be some finite non-negative $i < n$ such that for each $l \in \mathscr{L}$: $L[l] = L_i[l]$. By contraposition, this implies for each $l \in \mathscr{L}$: **if $L[l] = \langle 1,0,0,0\rangle$ then $L_p[l] \neq \langle 1,1,1,1\rangle$** and therefore $L_p[l] = \langle 1,0,0,0\rangle$.□

Having shown that literals can only be labelled as unsatisfiable in the prepocessing step, it is a small step to prove that Stability-label is sound for the stable-unsatisfiable cases.

Lemma 14. (*Soundness stable-unsatisfiable labelling*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\ })$, let $\mathscr{C}$ be the set of queryable literals and let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, $\bar{\ }$, $\mathscr{C}$ and $\mathscr{K}$. For each $l \in \mathscr{L}$: if $L[l] = \langle 1,0,0,0\rangle$ then l is stable-unsatisfiable in AT w.r.t. $\mathscr{C}$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\ })$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by Stability-label on $\mathscr{L}$, $\mathscr{R}$, $\bar{\ }$, $\mathscr{C}$ and $\mathscr{K}$. Each literal $l \in \mathscr{L}$ that is labelled $L[l] = \langle 1,0,0,0\rangle$ was already labelled $L_p[l] = \langle 1,0,0,0\rangle$ by Lemma 13, which by Lemma 2 implies that l is stable-unsatisfiable in AT w.r.t. $\mathscr{C}$.□

Next, we show that Stability-label is sound for literals that are labelled stable-defended.

Lemma 15. (*Soundness stable-defended labelling*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\ })$, let $\mathscr{C}$ be the set of queryable literals and let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, $\bar{\ }$, $\mathscr{C}$ and $\mathscr{K}$. For each literal $l \in \mathscr{L}$: if $L[l] = \langle 0,1,0,0\rangle$ then l is stable-defended in AT w.r.t. $\mathscr{C}$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\ })$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, $\bar{\ }$, $\mathscr{C}$ and $\mathscr{K}$. We proceed by induction on $c_{\{u,o,b\}}(l)$.

Proposition ($P(n)$): For each $l \in \mathscr{L}$ such that $c_{\{u,o,b\}}(l) \leq n$: if $L[l] = \langle 0,1,0,0\rangle$ then l is stable-defended in AT w.r.t. $\mathscr{C}$.

Base case ($P(0)$): Let l be an arbitrary literal in $\mathscr{L}$ such that $c_{\{u,o,b\}}(l) \leq 0$ and suppose that $L[l] = \langle 0,1,0,0\rangle$. Note that $L[l].d$ implies that $L_p[l] = \langle 1,1,1,1\rangle$. It follows that l must have been relabelled to $\langle 0,1,0,0\rangle$ by Algorithm 4 line 5. At that moment, all rules $r \in \mathscr{R}$ are labelled $L[r].u$ (Remark 6) hence the $\neg L[l].u$ label cannot be caused by Algorithm 2 L-U-b. So case L-U-a must have applied: $l \in \mathscr{K}$. Then there is an observation-based argument for l in each $AT' \in F_{\mathscr{C}}(AT)$ (Definition 10) that cannot be attacked and therefore is in the grounded extension. Consequently, l is stable-defended in AT w.r.t. $\mathscr{C}$.

Induction hypothesis ($P(k)$): For each $l \in \mathscr{L}$ such that $c_{\{u,o,b\}}(l) \leq k$: if $L[l] = \langle 0,1,0,0\rangle$ then l is stable-defended in AT w.r.t. $\mathscr{C}$.

Induction step ($P(k+1)$): Now consider an arbitrary literal $l \in \mathscr{L}$ such that $c_{\{u,o,b\}}(l) = k+1$ and suppose that $L[l] = \langle 0,1,0,0\rangle$.

First suppose towards a contradiction that $l \in \mathscr{C}$. Note that $l \in \mathscr{K}$, since otherwise there would be a rule r for l with $\neg L[r].u$ (L-U-b) and for each rule r for l: $\neg L[r].d$, $\neg L[r].o$ and $\neg L[r].b$ (L-O-f), a contradiction with Lemma 9. However, this means that l would be labelled stable-defended before the start of the while loop, which contradicts our assumption that $c_{\{u,o,b\}}(l) = k+1$ for positive k. Consequently, $l \notin \mathcal{Q}$.

Given that $l \notin \mathscr{C}$, the label $\neg L[l].u$ must have been caused by Algorithm 2 case L-U-b. Then $\neg L[l].o$ must have been caused by either L-O-d or L-O-e and $\neg L[l].b$ must have been caused by either L-B-c or L-B-d (L-B-a contradicts $l \notin \mathscr{C}$; L-B-b contradicts Lemma 9). It follows that there is **a rule r for l with $\neg L[r].u$, $\neg L[r].o$ and $\neg L[r].b$** and **for each $l' \in \bar{l}$, for each rule r' for l': $\neg L[r'].d$ and $\neg L[r'].b$**. Consider both conclusions:

- Let r be the rule for l such that $\neg L[r].u$, $\neg L[r].o$ and $\neg L[r].b$. The fact that r is labelled $\neg L[r].u$ must have been caused by R-U-a; then $\neg L[r].o$ has to be caused by R-O-a and $\neg L[r].b$ must have been caused by R-B-a, so each antecedent a of r is labelled $\neg L[a].u$, $\neg L[a].o$ and $\neg L[a].b$. By Lemma 9, the d-boolean must have been positive: for each $a \in \texttt{ants}(r) : L_k[a] = \langle 0,1,0,0\rangle$. Then by the induction hypothesis, each antecedent $a \in \texttt{ants}(r)$ is stable-defended in AT w.r.t. $\mathscr{C}$. So by Definition 11, for each $a \in \texttt{ants}(r)$, for each AT' in $F_{\mathscr{C}}(AT)$, there is an argument for a in $G(AT')$. By Lemma 6, each argument attacking this argument is attacked by an observation-based argument (which is unattacked and therefore in the grounded extension). To conclude, for each $AT' \in F_{\mathscr{C}}(AT)$, there is an argument A based on r in $Arg(AT')$ such that each argument B attacking A *on a sub-argument* in AT' is attacked by an observation-based argument in $G(AT')$.
- Now consider that each rule r' for some $l' \in \bar{l}$ is labelled $\neg L[r'].d$ and $\neg L[r'].b$. By Lemma 12, this implies that for each $AT' \in F_{\mathscr{C}}(AT)$, for each $l' \in \bar{l}$, for each rule r' for l', each argument based on r in $Arg(AT')$ is attacked by an argument in $G(AT')$. Thus for each $AT' \in F_{\mathscr{C}}(AT)$, each argument attacking an argument for l *on its conclusion* in AT' is attacked by an argument in the grounded extension $Arg(AT')$.

To summarize, for each $AT' \in F_{\mathscr{C}}(AT)$, there is some argument A (based on r, for l) in $Arg(AT')$ such that each argument B in $Arg(AT')$ attacking A (either on a subargument or on its conclusion l) is attacked by an argument in the grounded extension $Arg(AT)$. Then by Definitions 7, 8 and 11, **l is**

stable-defended in AT w.r.t. $\mathcal{Q}$. Since we chose l arbitrarily, this concludes the induction step.

At this point, we have shown that for each $n \in \mathbb{N}$: for each $l \in \mathscr{L}$: if $c_{\{u,o,b\}}(l) \leq n$ and $L[l] = \langle 0,1,0,0 \rangle$ then l is stable-defended in AT w.r.t. $\mathscr{C}$. Given that for each $l \in \mathscr{L}$ such that $L[l] = \langle 0,1,0,0 \rangle$, $c_{\{u,o,b\}}(l)$ is finite (Definition 20), we derive: **for each $l \in \mathcal{L}$: if $L[l] = \langle 0,1,0,0 \rangle$ then l is stable-defended in AT w.r.t. $\mathcal{Q}$.**□

We continue with showing soundness for literals that are labelled stable-out by Stability-label.

Lemma 16. (*Soundness stable-out labelling*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, let $\mathscr{C}$ be the set of queryable literals and let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each $l \in \mathscr{L}$: if $L[l] = \langle 0,0,1,0 \rangle$ then l is stable-out in AT w.r.t. $\mathscr{C}$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. Suppose that $l \in \mathscr{L}$ is labelled $L[l] = \langle 0,0,1,0 \rangle$ by Stability-label (Algorithm 4). Note that this implies that $\boldsymbol{L_p}[l] = \langle \boldsymbol{1},\boldsymbol{1},\boldsymbol{1},\boldsymbol{1} \rangle$, so the fact that l is labelled $\neg L[l].u$, $\neg L[l].d$ and $\neg L[l].b$ must have been caused by Algorithm 2. We consider two cases: $l \in \mathscr{C}$ and $l \notin \mathscr{C}$.

- First suppose that $l \in \mathscr{C}$. Then the fact that l is labelled $\neg L[l].d$ must have been caused by Algorithm 2 case L-D-a, so there is some $l' \in \bar{l}$ such that $l' \in \mathscr{K}$. Knowledge bases are consistent, therefore $l \notin \mathscr{K}$. This means that $\neg L[l].u$ cannot be caused by Algorithm 2 case L-U-a, hence must have been caused by case L-U-b: there is a rule r for l with $\neg L[r].u$. Then by Lemma 8, there is an argument for l based on r in $Arg(AT)$. However, since there is some $l' \in \bar{l}$ such that $l' \in \mathscr{K}$, each argument for l is attacked by the observation-based argument for l'. Hence, by Lemmas 10 and 7, **for each $AT' \in \boldsymbol{F}_{\mathcal{Q}}(\boldsymbol{AT})$, $\boldsymbol{l}$ is out in $\boldsymbol{AT'}$.**
- Now suppose that $l \notin \mathscr{C}$. Then $l \notin \mathscr{K}$, so label $\neg L[l].u$ must have been caused by case L-U-b: there is a rule r for l with $\neg L[r].u$. Then by Lemma 8, there is an argument based on r in $Arg(AT)$. Furthermore, since l is labelled $L[l].o$, we know that there is no rule r for l with $\neg L[r].u$ and $\neg L[r].o$ (otherwise Algorithm 2 case L-O-e would apply). As a result, L-B-d cannot apply, so the fact that l is labelled $\neg L[l].b$ must be caused by either L-B-b or L-B-c. In both cases, for each rule r for l: $\neg L[r].d$ and $\neg L[r].b$. Now, by Lemma 12: for each AT' in $F_{\mathscr{C}}(AT)$, each argument for l is attacked by the grounded extension. Therefore, by Definition 8, **$\boldsymbol{l}$ is out in $\boldsymbol{AT'}$ for each $\boldsymbol{AT'}$ in $\boldsymbol{F}_{\mathcal{Q}}(\boldsymbol{AT})$.**

To conclude, **for each $\boldsymbol{l} \in \mathcal{L}$ that is labelled $\boldsymbol{L}[l] = \langle \boldsymbol{0},\boldsymbol{0},\boldsymbol{1},\boldsymbol{0} \rangle$: $\boldsymbol{l}$ is stable-out in $\boldsymbol{AT}$ w.r.t. $\mathcal{Q}$.**□

In order to prove soundness for stable-blocked labeled literals, we first need the following additional lemma: if a literal l is labelled $\neg L[l].d$ by Stability-label, then there is no future argumentation theory such that there is an argument for l in the grounded extension.

Lemma 17. (*Labelled not defended*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, let $\mathscr{C}$ be the set of queryable literals and let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each $l \in \mathscr{L}$: if l is labelled $\neg L[l].d$ then there is no argument for l in $G(AT')$ for any AT' in $F_{\mathscr{C}}(AT)$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. We proceed by induction on $c_{\{d\}}$.

Proposition ($P(n)$): For each $l \in \mathscr{L}$ such that $c_{\{d\}}(l) \leq n$: if l is labelled $\neg L[l].d$, then there is no argument for l in $G(AT')$ for any AT' in $F_{\mathscr{C}}(AT)$.

Base case ($P(0)$): Let l be an arbitrary literal from $\mathscr{L}$ such that $c_{\{d\}}(l) \leq 0$. We consider two cases:

- First suppose that $L_p[l] = \langle 1,0,0,0 \rangle$. By Lemma 2, for each $AT' \in F_{\mathscr{C}}(AT)$, there is no argument for l in $Arg(AT')$. As a consequence, **there is no argument for $\boldsymbol{l}$ in $\boldsymbol{G}(\boldsymbol{AT'})$ for any $\boldsymbol{AT'}$ in $\boldsymbol{F}_{\mathcal{Q}}(\boldsymbol{AT})$.**
- Alternatively, suppose that $L_p[l] = \langle 1,1,1,1 \rangle$. We will first show that $l \in \mathscr{C}$: suppose that $l \notin \mathscr{C}$. We know that l was labelled $\neg L[l].d$ in Algorithm 4 line 5, so the condition in line 4 must have applied. By assumption $l \notin \mathscr{C}$, hence we have that there is no rule for l in $\mathscr{R}$. However, that implies that l would have been labelled $\langle 1,0,0,0 \rangle$ in the preprocessing step (Algorithm 1 line 4), and this label would not change: Algorithm 1 line 13 would not be executed for l since the condition in line 11 does not apply. A contradiction with our assumption that $L_p[l] = \langle 1,1,1,1 \rangle$. Therefore, $\boldsymbol{l} \in \mathcal{Q}$.

  Given that $l \in \mathscr{C}$, the label $\neg L[l].d$ must be caused by Algorithm 2 case L-D-a: there is some $l' \in \bar{l}$ such that $l' \in \mathscr{K}$. This means that there is an observation-based argument for l' in $Arg(AT)$ which, by Lemma 10 and Definitions 5 and 7, is in the grounded extension of every $AT' \in F_{\mathscr{C}}(AT)$. Therefore **there is no argument for $\boldsymbol{l}$ in $\boldsymbol{G}(\boldsymbol{AT'})$ for any $\boldsymbol{AT'}$ in $\boldsymbol{F}_{\mathcal{Q}}(\boldsymbol{AT})$.**

Induction hypothesis ($P(k)$): For each $l \in \mathscr{L}$ such that $c_{\{d\}}(l) \leq k$: if l is labelled $\neg L[l].d$, then there is no argument for l in $G(AT')$ for any future argumentation theory AT' in $F_{\mathscr{C}}(AT)$.

Induction step: Let l be an arbitrary literal in $\mathscr{L}$ such that $c_{\{d\}}(l) = k+1$. First, we show that $\boldsymbol{l} \notin \mathcal{Q}$: if l would be in $\mathscr{C}$, then the label $\neg L[l].d$ must have been caused by Algorithm 2 case L-D-a: there is an $l' \in \bar{l}$ such that $l' \in \mathscr{K}$. However, that would imply that $c_{\{d\}}(l) = 0$, which contradicts our assumption that $c_{\{d\}}(l) = k+1$ for positive k.

Given that $l \notin \mathscr{C}$, $\neg L[l].d$ must have been caused by case L-D-b or L-D-c. Next, we consider these two cases:

- Suppose that each rule r for l is labelled $\neg L[r].d$ (L-D-b is applied). $c_{\{d\}}(l) = k+1$, so there exists at least one rule for l. We take an arbitrary rule r for l. Then, by R-D-a, there is an $a \in \texttt{ants}(r)$ with $\neg L[a].d$ and $c_{\{d\}}(a) \leq k$. By the induction hypothesis, for each $AT' \in F_{\mathscr{C}}(AT)$, there is no argument for a in $G(AT')$. As a result, for each $AT' \in F_{\mathscr{C}}(AT)$, each argument for a would be attacked by some argument that is not attacked by any argument in $G(AT')$, which implies that each argument based on r would be attacked by some argument that is not attacked by any argument in $G(AT')$. Since we chose r arbitrarily, we know that there is no rule-based argument for l in $G(AT')$ for any AT' in $F_{\mathscr{C}}(AT)$. Furthermore, the fact that $l \notin \mathscr{C}$ implies that there is no observation-based argument either, so **there is no argument for $\boldsymbol{l}$ in $\boldsymbol{G}(\boldsymbol{AT'})$ for any $\boldsymbol{AT'} \in \boldsymbol{F}_{\mathcal{Q}}(\boldsymbol{AT})$.**
- Now suppose that there is a rule r for l that is labelled $L[r].d$ (L-D-c is applied). Then $l \notin \mathscr{C}$ and there is some $l' \in \bar{l}$ such that there is a rule r' for l' with $\neg L[r'].u$ and $\neg L[r'].o$. By Lemma 11, for each future argumentation theory AT' in $F_{\mathscr{C}}(AT)$ there exists an argument for l' based on r' that is not

attacked by any argument in $G(AT')$. So for every $AT' \in F_{\mathscr{Q}}(AT)$, **if an argument for l exists, then it cannot be in $G(AT')$.**

At this point, we have shown for each non-negative integer $n \in \mathbb{N}$: for each $l \in \mathscr{L}$ such that $c_{\{d\}}(l) \leq k$: if l is labelled $\neg L[l].d$, then there is no argument for l in $G(AT')$ for any $AT' \in F_{\mathscr{Q}}(AT)$. Given that for each $l \in \mathscr{L}$ such that l is labelled $\neg L[l].d$: $c_{\{d\}}(l)$ is finite (Definition 20), we derive: **for each $l \in \mathcal{L}$ that is labelled $\neg L[l].d$, then there is no argument for l in $G(AT')$ for any $AT' \in F_{\mathcal{Q}}(AT)$.**□

We now use Lemma 17 to prove Lemma 18, which is the soundness proof for the fourth and final justification status: blocked.

Lemma 18. (*Soundness stable-blocked labelling*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, let $\mathscr{Q}$ be the set of queryable literals and let L be the labelling obtained by* Stability-label *(Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. For each $l \in \mathscr{L}$: if $L[l] = \langle 0,0,0,1 \rangle$ then l is stable-blocked in AT w.r.t. $\mathscr{Q}$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{Q}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. Let $l \in \mathscr{L}$ be an arbitrary literal that is labelled $L[l] = \langle 0,0,0,1 \rangle$ by Stability-label (Algorithm 4). Note that $L[l].b$ implies that $L_p[l] = \langle 1,1,1,1 \rangle$, so the fact that l is labelled $\neg L[l].u$, $\neg L[l].d$ and $\neg L[l].o$ must have been caused by Algorithm 2.

The fact that l is labelled $L[l].b$, although l is considered for labelling by Algorithm 2, implies that $l \notin \mathscr{Q}$ (case L-B-a did not apply). Then $l \notin \mathscr{K}$, so label $\neg L[l].u$ must have been caused by case L-U-b. Then the label $\neg L[l].o$ must be caused by either L-O-d or L-O-e. In both cases, some rule r for l is labelled $\neg L[r].u$ and $\neg L[r].o$. By Lemma 11, for each $AT' \in F_{\mathscr{Q}}(AT)$, there is an argument based on r in $Arg(AT')$ that is not attacked by an argument in $G(AT')$. Furthermore, the fact that l is labelled $\neg L[l].d$ implies that for every future argumentation theory AT' in $F_{\mathscr{Q}}(AT)$ there is no argument for l in the grounded extension $G(AT')$ (Lemma 17). To conclude, for each $AT' \in F_{\mathscr{Q}}(AT)$, l is blocked in AT' (Definition 8), so l is stable-blocked in AT w.r.t. $\mathscr{Q}$ (Definition 11).□

Finally, we combine these lemmas to show that Stability-label is sound: if some literal l is labelled as stable in some argumentation theory AT w.r.t. the set of queryables $\mathscr{Q}$, then l is stable in AT w.r.t. $\mathscr{Q}$.

Proposition 2. (*Soundness stability labelling*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, let $\mathscr{Q}$ be the set of queryable literals and let L be the labelling obtained by* Stability-label *(Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. If L labels a literal $l \in \mathscr{L}$ stable-unsatisfiable in AT w.r.t. $\mathscr{Q}$, then l is stable-unsatisfiable in AT w.r.t. $\mathscr{Q}$; if L labels a literal $l \in \mathscr{L}$ stable-defended in AT w.r.t. $\mathscr{Q}$, then l is stable-defended in AT w.r.t. $\mathscr{Q}$; if L labels a literal $l \in \mathscr{L}$ stable-out in AT w.r.t. $\mathscr{Q}$, then l is stable-out in AT w.r.t. $\mathscr{Q}$; and if L labels a literal $l \in \mathscr{L}$ stable-blocked in AT w.r.t. $\mathscr{Q}$, then l is stable-blocked in AT w.r.t. $\mathscr{Q}$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{Q}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{Q}$ and $\mathscr{K}$. Suppose that a literal $l \in \mathscr{L}$ is labelled stable by Stability-label (Algorithm 4) and let L be the obtained labelling. Then $L[l] = \langle 1,0,0,0 \rangle$, $L[l] = \langle 0,1,0,0 \rangle$, $L[l] = \langle 0,0,1,0 \rangle$ or $L[l] = \langle 0,0,0,1 \rangle$. That l is stable follows from:

- If $L[l] = \langle 1,0,0,0 \rangle$ then l is stable-unsatisfiable in AT w.r.t. $\mathscr{Q}$ by Lemma 14.
- If $L[l] = \langle 0,1,0,0 \rangle$ then l is stable-defended in AT w.r.t. $\mathscr{Q}$ by Lemma 15.
- If $L[l] = \langle 0,0,1,0 \rangle$ then l is stable-out in AT w.r.t. $\mathscr{Q}$ by Lemma 16.
- If $L[l] = \langle 0,0,0,1 \rangle$ then l is stable-blocked in AT w.r.t. $\mathscr{Q}$ by Lemma 18.

□

B8. Conditional completeness of the stability labelling algorithm

In this section, we prove Proposition 3, which specifies conditions under which Stability-label is complete. Similar to Section B.6, we first define height of potential arguments in Section B.8.1 before proceeding to the actual proofs in Section B.8.2.

B8.1. Height of potential arguments

Recall that the completeness cases from Proposition 3 are defined in terms of potential arguments (Definition 13). In order to prove this proposition by induction, we need the notion of height of potential arguments. Similar to the height of arguments, the height of a potential argument is the maximum number of inferences between a premise and the conclusion of the potential argument.

Definition 22. (*Height of potential arguments*) Let $AT = (AS, \mathscr{K})$ be an argumentation theory and let $A^p \in P_{\mathscr{Q}}(AT)$ be an argument. The height $\mathtt{h}(A^p)$ of A^p is:

- if A^p is an **observation-based potential argument** then $\mathtt{h}(A^p) = 0$;
- if A is a **rule-based argument** of the form $A_1, \ldots, A_m \Rightarrow c$, then $\mathtt{h}(A) = 1 + \max(\mathtt{h}(A_1), \ldots, \mathtt{h}(A_m))$.

Note that Definition 13 on potential arguments enforces that all potential arguments have finite height, since the number of steps for constructing a potential argument is finite.

B8.2. Conditional completeness proof

In this section, we prove Proposition 3, which specifies conditions under which Stability-label is complete. In order to prove this, we first need Lemma 3 (see Section 4.2.3 in the paper), which consists of six items and shows in which situations Stability-label turns one or more booleans (u, d, o and/or b) of the labelling L to False. In the following lemmas, we will prove each of these items. First, we prove that Preprocess (Algorithm 1) labels literals/rules as $\langle 1,0,0,0 \rangle$ if and only if there exists no potential argument for/based on them. This is Item 1 of Lemma 3.

Lemma 19. (*Soundness and completeness Algorithm 1 for potential arguments*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let L_p be the labelling after executing Algorithm 1. For each rule $r \in \mathscr{R}$: $L_p[r] = \langle 1,0,0,0 \rangle$ iff there is no $A^p \in P_{\mathscr{Q}}(AT)$ such that A^p is based on r. For each literal $l \in \mathscr{L}$: $L_p[l] = \langle 1,0,0,0 \rangle$ iff there is no $A^p \in P_{\mathscr{Q}}(AT)$ such that A^p is a potential argument for l.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{C}$ be a set of queryables. Let L_p be the labelling obtained by Preprocess (Algorithm 1) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$.

- **Left to right (soundness)**: We proceed by induction:
  Proposition ($P(n)$): Each $l \in \mathscr{L}$ for which there is a potential argument A^p in $P_{\mathscr{C}}(AT)$ with $\mathtt{h}(A) \leq n$ is labelled $L_p[l] = \langle 1,1,1,1\rangle$; each $r \in \mathscr{R}$ such that there is a potential argument A^p based on r in $P_{\mathscr{C}}(AT)$ with $\mathtt{h}(A^p) \leq n+1$ is labelled $L_p[r] = \langle 1,1,1,1\rangle$.
  Base case ($P(0)$):
  - For each l in $\mathscr{L}$ such that there is some A^p for l in $P_{\mathscr{C}}(AT)$ and $\mathtt{h}(A^p) = 0$, then by Definition 22 $l \in \mathscr{C}$ and for each $l' \in \bar{l}$: $l' \notin \mathscr{K}$, so by Algorithm 1 line 3, $L_0^p[l] = \langle 1,1,1,1\rangle$.
  - For each $r \in \mathscr{R}$ such that there is some A^p based on r in $P_{\mathscr{C}}(AT)$ with $\mathtt{h}(A^p) = 0$, there must be an observation-based potential argument for each $a \in \mathtt{ants}(r)$, so for each $a \in \mathtt{ants}(r)$: $L_p[a] = \langle 1,1,1,1\rangle$. This label must have been assigned by Algorithm 1 line 3, after which r was labelled $L_p[r] = \langle 1,1,1,1\rangle$ in Algorithm 1 line 12.
  Induction hypothesis ($P(k)$): Each $l \in \mathscr{L}$ for which there is a potential argument A^p in $P_{\mathscr{C}}(AT)$ with $\mathtt{h}(A) \leq k$ is labelled $L_p[l] = \langle 1,1,1,1\rangle$; each $r \in \mathscr{R}$ such that there is a potential argument A^p based on r in $P_{\mathscr{C}}(AT)$ with $\mathtt{h}(A^p) \leq k+1$ is labelled $L_p[r] = \langle 1,1,1,1\rangle$.
  Induction step ($P(k+1)$):
  - For each l in $\mathscr{L}$ such that there is some A^p for l in $P_{\mathscr{C}}(AT)$ and $\mathtt{h}(A^p) = k+1$, there must be some rule r on which A^p is based; by the induction hypothesis, $L_p[r] = \langle 1,1,1,1\rangle$. This must have happened in line 12 of Algorithm 1, after which l was labelled $L_p[l] = \langle 1,1,1,1\rangle$ in line 13.
  - For each $r \in \mathscr{R}$ such that there is some A^p based on r in $P_{\mathscr{C}}(AT)$ with $\mathtt{h}(A^p) = k+2$, there must be potential arguments for each $a \in \mathtt{ants}(r)$ with a height of at most $k+1$, so by the induction hypothesis, each $a \in \mathtt{ants}(r)$: $L_p[a] = \langle 1,1,1,1\rangle$. At least one of these labels must have been assigned by Algorithm 1 line 13, after which r was labelled $L_p[r] = \langle 1,1,1,1\rangle$ in Algorithm 1 line 12 in a later iteration of the while loop.
  Finally note that all potential arguments have finite height, so $P(n)$ can be generalised to all l in $\mathscr{L}$. If $L_p[l] = \langle 1,0,0,0\rangle$ then $L_p[l] = \langle 1,1,1,1\rangle$, so there is no A^p for l in $P_{\mathscr{C}}(AT)$. Similarly, for each $r \in \mathscr{R}$ labelled $L_p[r] = \langle 1,0,0,0\rangle$, there is no A^p based on r in $P_{\mathscr{C}}(AT)$.
- **Right to left (completeness)**: First, we introduce some notation, which is similar to the interim labelling from Definition 19. Let $x \in \mathscr{L} \cup \mathscr{R}$ be a literal or rule; we denote by $L_0^p[x]$ the label given to x by the preprocessing algorithm (Algorithm 1) between line 7 and 8. Furthermore, $L_k^p[x]$ is the label given to x by the preprocessing algorithm just after the k'th iteration of the for loop (line 10–14). Using this notation, we proceed by induction.
  Proposition ($P(n)$): For each $l \in \mathscr{L}$ such that $L_n^p[l] = \langle 1,1,1,1\rangle$, there is a potential argument for l in $P_{\mathscr{C}}(AT)$; for each $r \in \mathscr{R}$ such that $L_n^p[r] = \langle 1,1,1,1\rangle$, there is a potential argument based on r in $P_{\mathscr{C}}(AT)$.
  Base case ($P(0)$):
  - For each $l \in \mathscr{L}$ such that $L_0^p[l] = \langle 1,1,1,1\rangle$, the condition in Algorithm 1 line 2 must have applied ($l \in \mathscr{C}$ and for each $l' \in \bar{l}$: $l' \notin \mathscr{K}$), so by Definition 13, there is an observation-based potential argument for l in $P_{\mathscr{C}}(AT)$.
  - No rule $r \in \mathscr{R}$ is labelled $L_n^p[r] = \langle 1,1,1,1\rangle$, so for each $r \in \mathscr{R}$ such that $L_n^p[r] = \langle 1,1,1,1\rangle$, there is a potential argument based on r in $P_{\mathscr{C}}(AT)$.
  Induction hypothesis ($P(k)$): For each $l \in \mathscr{L}$ such that $L_k^p[l] = \langle 1,1,1,1\rangle$, there is a potential argument for l in $P_{\mathscr{C}}(AT)$; for each $r \in \mathscr{R}$ such that $L_k^p[r] = \langle 1,1,1,1\rangle$, there is a potential argument based on r in $P_{\mathscr{C}}(AT)$.
  Induction step ($P(k+1)$):
  - For each $r \in \mathscr{R}$ such that $L_{k+1}^p[r] = \langle 1,1,1,1\rangle$, it must be that for each $a \in \mathtt{ants}(r)$: $L_k^p[a] = \langle 1,1,1,1\rangle$. Then by the induction hypothesis, for each $a \in \mathtt{ants}(r)$, there is a potential argument for a in $P_{\mathscr{C}}(AT)$, which by Definition 13 implies that there is a potential argument based on r in $P_{\mathscr{C}}(AT)$.
  - For each $l \in \mathscr{L}$ such that $L_{k+1}^p[l] = \langle 1,1,1,1\rangle$, it must be that there is some rule based on which there is a potential argument in $P_{\mathscr{C}}(AT)$. Consequently, there is a potential argument for l in $P_{\mathscr{C}}(AT)$.
  Finally note that Algorithm 1 terminates, given the running time is polynomial in the input (Lemma 4) and the language and rule set are finite. Consequently, for each $l \in \mathscr{L}$: if $L_p[l] = \langle 1,1,1,1\rangle$ then there is a potential argument for l in $P_{\mathscr{C}}(AT)$. Then by contraposition: if there is no potential argument for l in $P_{\mathscr{C}}(AT)$, then $L_p[l] \neq \langle 1,1,1,1\rangle$, so $L_p[l] = \langle 1,0,0,0\rangle$. Similarly, each $r \in \mathscr{R}$ based on which there is no potential argument in $P_{\mathscr{C}}(AT)$ is labelled $L_p[r] = \langle 1,0,0,0\rangle$.

□

Next, we specify the condition under which a literal l is labelled $\neg L[l].u$ or a rule r is labelled $\neg L[r].u$. This is Item 2 of Lemma 3. Note that this is the reversed version of Lemma 8.

Lemma 20. (*Argument existence labelling*) *Given an argumentation theory $AT = (AS, \mathscr{K})$ where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and $\mathscr{C}$ is a set of queryables, let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each literal $l \in \mathscr{L}$: if there is an argument for l in $Arg(AT)$, then $\neg L[l].u$. For each $r \in \mathscr{R}$: if there is an argument based on r in $Arg(AT)$ then $\neg L[r].u$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. We proceed by induction on the height of arguments for l/based on r.

Proposition ($P(n)$): Each $l \in \mathscr{L}$ for which there is an argument A with $\mathtt{h}(A) \leq n$ is labelled $\neg L[a].u$ and each $r \in \mathscr{R}$ based on which there is an argument A with $\mathtt{h}(A) \leq n+1$ is labelled $\neg L[r].u$.

Base case ($P(0)$): For each $l \in \mathscr{L}$ such that there is an argument A for l with $\mathtt{h}(A) = 0$, it must be that $l \in \mathscr{K}$, hence $l \in \mathscr{C}$. This means that the condition for the if-statement in Algorithm 4 line 4 applies and l is labelled $\neg \boldsymbol{L}[l].\boldsymbol{u}$ in Algorithm 4 line 5 by Algorithm 2 case L-U-a.

For each $r \in \mathscr{R}$ such that there is an argument A based on r with $\mathtt{h}(A) = 1$, it must be that for each $a \in \mathtt{ants}(r)$ there is an argument A' for a with $\mathtt{h}(A) = 0$, so a is labelled $\neg L[a].u$ in Algorithm 4 line 5; consequently r is added to todo-set in line 6. When popped from this set, r is labelled $\neg \boldsymbol{L}[r].\boldsymbol{u}$ by case R-U-a.

Induction hypothesis ($P(k)$): Each $l \in \mathscr{L}$ for which there is an argument A with $\mathtt{h}(A) \leq k$ is labelled $\neg L[l].u$ and each $r \in \mathscr{R}$ based on which there is an argument A with $\mathtt{h}(A) \leq k+1$ is labelled $\neg L[r].u$.

Induction step ($P(k+1)$): For each $l \in \mathscr{L}$ such that there is an argument A for l with $\mathtt{h}(A) = k+1$, it must be that $\mathtt{top-rule}(A)$ is labelled $\neg L[\mathtt{top-rule}(A)].u$ (induction hypothesis). After the top rule's label change, which must have happened in Algorithm 4 line 9, its consequent l is relabelled in line 11 as $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{u}$ (by case L-U-b).

For each $r \in \mathscr{R}$ such that there is an argument A based on r with $\mathtt{h}(A) = k+2$, it must be that for each $a \in \mathtt{ants}(r)$ there is an argument A' for a with $\mathtt{h}(A) \leq k+1$, so by the induction hypothesis (applicable if $\mathtt{h}(A') \leq k$) and above conclusion (applicable if $\mathtt{h}(A') = k+1$), a is labelled $\neg L[a].u$. The last of these antecedents must have been relabelled in Algorithm 4 line 11 or line 15; in both cases, r is added to TODO-SET immediately afterwards and, when popped, labelled $\neg \boldsymbol{L}[r].\boldsymbol{u}$ by case R-U-a.

Finally, recall from Definition 17 that all arguments have finite height. This concludes the proof.□

The next lemma shows under which conditions a literal l is labelled $\neg L[l].d$ and $\neg L[l].b$. This corresponds to Lemma 3 Item 3.

Lemma 21. (*Conditions for $\neg L[l].d$ and $\neg L[l].b$*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, let $\mathscr{C}$ be the set of queryable literals and let L be the labelling obtained by* STABILITY-LABEL *(Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each literal $l \in \mathscr{L}$: if each potential argument for l in $P_{\mathscr{C}}(AT)$ is p-attacked by an observation-based argument in $Arg(AT)$, then $\neg L[l].d$ and $\neg L[l].b$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by STABILITY-LABEL (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$.

Let $l \in \mathscr{L}$ be a literal such that each potential argument for l in $P_{\mathscr{C}}(AT)$ is p-attacked by an observation-based argument in $Arg(AT)$. We proceed by induction on the height of potential arguments for a literal.

Proposition ($P(n)$): For each $l \in \mathscr{L}$: if for each potential argument A^p for l in $P_{\mathscr{C}}(AT)$: [$\mathtt{h}(A^p) \leq n$ and A^p is p-attacked by an observation-based argument in $Arg(AT)$] then l is labelled $\neg L[l].d$ and $\neg L[l].b$.

Base case ($P(1)$): Let $l \in \mathscr{L}$ be an arbitrary literal such that each potential argument A^p for l in $P_{\mathscr{C}}(AT)$ has $\mathtt{h}(A^p) \leq 1$ and A^p is p-attacked by an observation-based argument in $Arg(AT)$. Note the following:

- **There is no potential argument A^p for l in $P_{\mathcal{Q}}(AT)$ such that $\mathtt{h}(A^p) = 0$**: if such a potential argument would exist, then it would be observation-based (by Definition 13: $l \in \mathscr{C}$ and for each $l' \in \bar{l}$: $l' \notin \mathscr{K}$), which means that it could not be p-attacked by an observation-based argument in $Arg(AT)$.
- $l \in \mathcal{Q}$: otherwise, the attack on A^p by some observation-based argument in $Arg(AT)$ must have been *on a subargument*, but that is impossible: since $A^p \in P_{\mathscr{C}}(AT)$ and $\mathtt{h}(A^p) = 1$, it must be that for each $a \in \mathtt{ants}(\mathtt{top-rule}(A^p))$, there is no $a' \in \bar{a}$ such that $a' \in \mathscr{K}$.
- **There is some $l' \in \bar{l}$ such that $l' \in \mathcal{K}$**: if none of $\bar{l}$ would be in $\mathscr{K}$, then there would be an observation-based potential argument for l in $P_{\mathscr{C}}(AT)$ that is not p-attacked by any observation-based argument in $Arg(AT)$, a contradiction.

Given that $l \in \mathscr{C}$, l is considered for relabelling by Algorithm 4 line 5 and labelled $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{d}$ **and** $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{b}$ by Algorithm 2 case L-D-a and L-B-a. As l is chosen arbitrarily, we can generalise this result.

Induction hypothesis ($P(k)$): For each $l \in \mathscr{L}$: if for each potential argument A^p for l in $P_{\mathscr{C}}(AT)$: [$\mathtt{h}(A^p) \leq k$ and A^p is p-attacked by an observation-based argument in $Arg(AT)$] then l is labelled $\neg L[l].d$ and $\neg L[l].b$.

Induction step ($P(k+1)$): Now let $l \in \mathscr{L}$ be an arbitrary literal such that each potential argument A^p for l has $\mathtt{h}(A^p) \leq k+1$ and is p-attacked by an observation-based argument in $Arg(AT)$. We consider two cases: either $l \in \mathscr{C}$ or $l \notin \mathscr{C}$.

- First suppose that $l \in \mathscr{C}$. Then **there is some $l' \in \bar{l}$ such that $l' \in \mathcal{K}$**: if for each $l' \in \bar{l}$: $l' \notin \mathscr{K}$, then there would be an observation-based potential argument for l in $P_{\mathscr{C}}(AT)$ that is not p-attacked by any observation-based argument in $Arg(AT)$ (Definition 13); a contradiction. Given that $l \in \mathscr{C}$, l is considered for relabelling in Algorithm 4 line 5 and labelled $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{d}$ and $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{b}$ by Algorithm 2 case L-D-a and L-B-a.
- Alternatively suppose that $l \notin \mathscr{C}$. Then each potential argument for l in $P_{\mathscr{C}}(AT)$ must be rule-based. Let $r \in \mathscr{R}$ be an arbitrary rule for l. We consider two cases and show that $\neg L[r].d$ and $\neg L[r].b$:
  - If $L_p[r] = \langle 1,0,0,0 \rangle$ then $L[r] = \langle 1,0,0,0 \rangle$ since there is no operation in Algorithm 4 that can turn booleans from False to True. This implies that r is labelled $\neg \boldsymbol{L}[r].\boldsymbol{d}$ **and** $\neg \boldsymbol{L}[r].\boldsymbol{b}$.
  - Alternatively, $L_p[r] = \langle 1,1,1,1 \rangle$, so by Lemma 19, there is an A^p based on r in $P_{\mathscr{C}}(AT)$. Since A^p is p-attacked by an observation-based argument in $Arg(AT)$ and $l \notin \mathscr{C}$, it must be p-attacked *on a subargument*.

    Next we prove that **there is some $a \in \mathtt{ants}(r)$ such that each A^p_i for a is p-attacked by an observation-based argument in $Arg(AT)$**: if this would not be the case, we could construct a potential argument $A^p_l = A^p_1, \ldots, A^p_n \Rightarrow l$ for l that is not p-attacked by an observation-based argument in $Arg(AT)$; a contradiction.

    Then by the induction hypothesis $\neg \boldsymbol{L}[\boldsymbol{a}].\boldsymbol{d}$ **and** $\neg \boldsymbol{L}[\boldsymbol{a}].\boldsymbol{b}$. This label cannot be caused by the preprocessing step: if $L_p[a]$ would be $\langle 1,0,0,0 \rangle$, then r would be labelled $\langle 1,0,0,0 \rangle$ in Algorithm 1 line 6, and never be relabelled to $\langle 1,1,1,1 \rangle$ by line 12; a contradiction. So $\boldsymbol{L_p}[\boldsymbol{a}] = \langle \boldsymbol{1},\boldsymbol{1},\boldsymbol{1},\boldsymbol{1} \rangle$. As a consequence, a must have been labelled $\neg L[a].d$ and $\neg L[a].b$ in Algorithm 4 line 5, line 11 or line 15. In all cases, r is added to TODO-SET immediately afterwards and relabelled in a later iteration of the while loop (Algorithm 4 line 9) as $\neg \boldsymbol{L}[r].\boldsymbol{d}$ **and** $\neg \boldsymbol{L}[r].\boldsymbol{b}$ by case R-D-a and R-B-b.

    Thus in both cases, **r is labelled $\neg \boldsymbol{L}[r].\boldsymbol{d}$ and $\neg \boldsymbol{L}[r].\boldsymbol{b}$**. Since r is an arbitrary rule for l, we can generalise this to all rules for l in $\mathscr{R}$. Finally, to show that l is labelled $\neg L[l].d$ and $\neg L[l].b$, we consider two possibilities:
  - If each rule r for l is labelled $L_p[r] = \langle 1,0,0,0 \rangle$ then l is initially labelled $\langle 1,0,0,0 \rangle$ by Algorithm 1 line 4. Since line 12 was never reached for any rule r for l, line 13 was not reached for l, hence $L_p[l]$ remains $\langle 1,0,0,0 \rangle$. No operation in Algorithm 4 turns booleans to True, so $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{d}$ **and** $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{b}$.
  - Alternatively, there is a rule r for l that is labelled $L_p[r] = \langle 1,1,1,1 \rangle$. Since r is a rule for l, it must be labelled $\neg L[r].d$ and $\neg L[r].b$. This must have been caused by Algorithm 4 line 9; after this change, l has been considered for relabelling by line 11 and labelled $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{d}$ **and** $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{b}$ by case L-D-b and L-B-b.

At this point, we have proven $P(n)$ for each non-negative integer $n \in \mathbb{N}$. Given that all potential arguments have finite height, we have shown for each $l \in \mathscr{L}$: if each potential argument for l in $P_{\mathscr{C}}(AT)$ is p-attacked by an observation-based argument in $Arg(AT)$, then l is labelled $\neg L[l].d$ and $\neg L[l].b$.□

Lemma 22 shows under which conditions a literal l is labelled $\neg L[l].u$ and $\neg L[l].o$. This corresponds to the left-to-right part of Lemma 3 Item 4.

Lemma 22 (Conditions for $\neg L[l].u$ and $\neg L[l].o$). *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\cdot})$, let $\mathscr{Q}$ be the set of queryable literals and let L be the labelling obtained by* Stability-label *(Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, $\bar{\cdot}$, $\mathscr{Q}$ and $\mathscr{K}$. For each literal $l \in \mathscr{L}$: if there is an argument A for l in $Arg(AT)$ and there is no observation-based potential argument in $P_{\mathscr{Q}}(AT)$ that p-attacks A, then $\neg L[l].u$ and $\neg L[l].o$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\cdot})$, let $\mathscr{Q}$ be a set of queryables and let L be the labelling obtained by Stability-label on $\mathscr{L}$, $\mathscr{R}$, $\bar{\cdot}$, $\mathscr{Q}$ and $\mathscr{K}$. We proceed by induction:

Proposition ($P(n)$): If there is an argument A for l in $Arg(AT)$ such that $\mathtt{h}(A) \leq n$ and there is no observation-based potential argument in $P_{\mathscr{Q}}(AT)$ that p-attacks A, then l is labelled $\neg L[l].u$ and $\neg L[l].o$.

Base case ($P(0)$): Suppose that there is an argument A for l in $Arg(AT)$ such that $\mathtt{h}(A) \leq 0$. Then $l \in \mathscr{K}$ (Definition 17), so l is relabelled in Algorithm 4 line 5 as $\neg L[l].u$ and $\neg L[l].o$ by Algorithm 2 case L-U-a and L-O-a.

Induction hypothesis ($P(k)$): If there is an argument A for l in $Arg(AT)$ such that $\mathtt{h}(A) \leq k$ and there is no observation-based potential argument in $P_{\mathscr{Q}}(AT)$ that p-attacks A, then l is labelled $\neg L[l].u$ and $\neg L[l].o$.

Induction step ($P(k+1)$): Now suppose that there is an argument A for l in $Arg(AT)$ such that $\mathtt{h}(A) = k+1$ and there is no observation-based potential argument in $P_{\mathscr{Q}}(AT)$ that p-attacks A. Then by Definition 17 there is a rule r for l such that for each antecedent $a \in \mathtt{ants}(r)$, there is an argument A_i for a in $Arg(AT)$, such that A_i is not p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$ and $\mathtt{h}(A_i) \leq k$. Then by the induction hypothesis, each $a \in \mathtt{ants}(r)$ is labelled $\neg L[a].u$ and $\neg L[a].o$. This label change must have happened in line 5, 11 or 15 of Algorithm 4, but in all cases, r is added to todo-set immediately afterwards (line 6/13/17) and considered for relabelling in line 9 of a later iteration of the while loop. By case R-U-a and R-O-a, r is labelled $\neg L[r].u$ and $\neg L[r].o$. Subsequently, l is considered for relabelling in line 11 and labelled $\neg L[l].u$ by case L-U-b. Furthermore, l is labelled $\neg L[l].o$ by either case L-O-e (if $l \notin \mathscr{Q}$) or L-O-c (if $l \in Q$, because then no observation-based potential argument for some $l' \in \bar{l}$ would p-attack A, which means that for each $l' \in \bar{l}$, some $l'' \in \overline{l'}$ must be in $\mathscr{K}$).

We have proven $P(n)$ for each $n \in \mathbb{N}$. Recall that all arguments have a finite height. This means that for each $l \in \mathscr{L}$: if there is an argument A for l in $Arg(AT)$ and there is no observation-based potential argument in $P_{\mathscr{Q}}(AT)$ that p-attacks A, then l is labelled $\neg L[l].u$ and $\neg L[l].o$.□

Lemma 23 shows the consequences of the label $\neg L[l].u$ and $\neg L[l].o$ of a literal l: if it is labelled as such, we can derive that there is an argument for l in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$. This corresponds to the right-to-left part of Lemma 3 Item 4.

Lemma 23 (Consequences of $\neg L[l].u$ and $\neg L[l].o$). *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\cdot})$, let $\mathscr{Q}$ be the set of queryable literals and let L be the labelling obtained by* Stability-label *(Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, $\bar{\cdot}$, $\mathscr{Q}$ and $\mathscr{K}$. For each literal $l \in \mathscr{L}$: if $\neg L[l].u$ and $\neg L[l].o$ then there is an argument A for l in $Arg(AT)$ and there is no observation-based potential argument in $P_{\mathscr{Q}}(AT)$ that p-attacks A.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\cdot})$ and let $\mathscr{Q}$ be a set of queryables. Let L be the labelling obtained by Stability-label on $\mathscr{L}$, $\mathscr{R}$, $\bar{\cdot}$, $\mathscr{Q}$ and $\mathscr{K}$. We proceed by induction.

Proposition ($P(n)$): For each $l \in \mathscr{L}$ that is labelled $\neg L[l].u$ and $\neg L[l].o$ such that $c_{\{u,o\}}(l) \leq n$, there is an argument for l in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$.

Base case ($P(0)$): Let $l \in \mathscr{L}$ be an arbitrary literal labelled $\neg L[l].u$ and $\neg L[l].o$ such that $c_{\{u,o\}}(l) \leq 0$. The label $\neg L[l].u$ cannot be assigned by Pre-process, so l must have been relabelled in Algorithm 4 line 5. At that point, all rules for l are still labelled $L[r].u$, so $\neg L[l].u$ cannot be caused by Algorithm 2 case L-U-b hence must be caused by L-U-a: $l \in \mathscr{K}$. Then there is an observation-based argument for l in $Arg(AT)$, which by Definition 14 cannot be p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$.

Induction hypothesis ($P(k)$): For each $l \in \mathscr{L}$ that is labelled $\neg L[l].u$ and $\neg L[l].o$ such that $c_{\{u,o\}}(l) \leq k$, there is an argument for l in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$.

Induction step ($P(k+1)$): Let $l \in \mathscr{L}$ be an arbitrary literal labelled $\neg L[l].u$ and $\neg L[l].o$ such that $c_{\{u,o\}}(l) = k+1$. This means that $l \notin \mathscr{K}$ (if $l \in \mathscr{K}$ then $c_{\{u,o\}}(l) = 0$). So the label $\neg L[l].u$ must have been caused by case L-U-b: there is a rule r for l labelled $\neg L[r].u$. Then by Lemma 9, the label $\neg L[l].o$ cannot be caused by case L-O-f and therefore must be caused by case L-O-b, L-O-c, L-O-d or L-O-e. In any of these four cases: there is a rule r for l labelled $\neg L[r].u$ and $\neg L[r].o$ ($c_{\{u,o\}}(r) = k+1$) and if $l \in \mathscr{Q}$ then for each $l' \in \bar{l}$ there is an $l'' \in \bar{l}$ such that $l'' \in \mathscr{K}$.

Given that there is a rule r for l labelled $\neg L[r].u$ and $\neg L[r].o$, each antecedent a of l must be labelled $\neg L[a].u$ and $\neg L[a].o$ by case R-U-a and R-O-a and the fact that $c_{\{u,o\}}(a) \leq k$. Then by the induction hypothesis, for each $a \in \mathtt{ants}(r)$ there is an argument A for a in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$ *on a subargument*.

Finally, since either $l \notin \mathscr{Q}$ or for each $l' \in \bar{l}$ there is some $l'' \in \overline{l'}$ such that $l'' \in \mathscr{K}$, there is no observation-based potential argument for any $l' \in \bar{l}$, p-attacking A *on its conclusion*. To conclude, there is an argument for l that is not p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$.

We have proven $P(i)$ for each non-negative integer $i < n$. Since Algorithm 4 has a polynomial runtime (Proposition 4), there must be some finite non-negative $i < n$ such that for each $l \in \mathscr{L}$: if $\neg L[l].u$ and $\neg L[l].o$ then there is an argument for l in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$.□

The next lemma shows in which situations a literal l is labelled $\neg L[l].d$ by Stability-label. This corresponds to Lemma 3 Item 5.

Lemma 24 (Conditions for $\neg L[l].d$). *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\cdot})$, let $\mathscr{Q}$ be the set of queryable literals and let L be the labelling obtained by* Stability-label *(Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, $\bar{\cdot}$, $\mathscr{Q}$ and $\mathscr{K}$. For each literal $l \in \mathscr{L}$: if each potential argument A^p for l in $P_{\mathscr{Q}}(AT)$ is p-attacked by an argument B in $Arg(AT)$ and there is no observation-based potential argument in $P_{\mathscr{Q}}(AT)$ that p-attacks B, then $\neg L[l].d$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \bar{\cdot})$ and let $\mathscr{Q}$ be a set of queryables. Let L be the labelling obtained by Stability-label on $\mathscr{L}$, $\mathscr{R}$, $\bar{\cdot}$, $\mathscr{Q}$ and $\mathscr{K}$. Let l be an arbitrary literal in $\mathscr{L}$ and suppose that each potential argument A^p for l in $P_{\mathscr{Q}}(AT)$ is p-attacked by an argument B in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$. We proceed by induction on the height of potential arguments.

Proposition ($P(n)$):

- For each $l \in \mathscr{L}$, if each potential argument A^p for l has a height $\mathtt{h}(A^p) \leq n$ and is p-attacked by an argument in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{Q}}(AT)$, then l is labelled $\neg L[l].d$; and

- For each $r \in \mathscr{R}$, if each potential argument A^p based on r has a height $\mathtt{h}(A^p) \leq n+1$ and is p-attacked on a subargument by an argument in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{C}}(AT)$, then r is labelled $\neg L[r].d$.

Base case ($P(0)$):

- Let $l \in \mathscr{L}$ be an arbitrary literal such that each potential argument for l has a height of 0 and is p-attacked by an argument that is not p-attacked by any observation-based potential argument in $P_{\mathscr{C}}(AT)$.
  - If $\boldsymbol{l} \in \mathcal{Q}$ then there is some $l' \in \bar{l}$ that is in $\mathscr{K}$: otherwise, by Definition 13 there would be some observation-based potential argument for l that p-attacks each argument in $Arg(AT)$ by which it is p-attacked; a contradiction. Then l is labelled in Algorithm 4 line 5 as $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{d}$ by Algorithm 2 case L-D-a.
  - If $\boldsymbol{l} \notin \mathcal{Q}$ then there is no potential argument for l in $P_{\mathscr{C}}(AT)$, so by Lemma 19 $L_p[l] = \langle 1,0,0,0\rangle$, hence $\neg L[l].d$.
- Let $r \in \mathscr{R}$ be an arbitrary rule such that each potential argument based on r has a height of at most 1 and is p-attacked on a subargument by an argument that is not p-attacked any observation-based potential argument in $P_{\mathscr{C}}(AT)$.

  Then **there is no potential argument based on r in $P_{\mathcal{Q}}(\boldsymbol{AT})$**: suppose, towards a contradiction, that such a potential argument A^p would exist, then A^p would be p-attacked on a subargument A' by an argument B in $Arg(AT)$. Given that $\mathtt{h}(A^p) = 1$, A' must be observation-based, so there is some $a \in \mathtt{ants}(r)$ such that $a \in \mathscr{C}$ and for each $a' \in \bar{a}$: $a' \notin \mathscr{K}$. However, that would mean that B's conclusion is not in $\mathscr{K}$, so (the observation-based potential argument) A' p-attacks B; a contradiction.

  Consequently, r is labelled $L_p[r] = \langle 1,0,0,0\rangle$ (Lemma 19), which implies $\neg \boldsymbol{L}[\boldsymbol{r}].\boldsymbol{d}$.

Induction hypothesis ($P(k)$):

- For each $l \in \mathscr{L}$, if each potential argument A^p for l has a height $\mathtt{h}(A^p) \leq k$ and is p-attacked by an argument in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{C}}(AT)$, then l is labelled $\neg L[l].d$; and
- For each $r \in \mathscr{R}$, if each potential argument A^p based on r has a height $\mathtt{h}(A^p) \leq k+1$ and is p-attacked on a subargument by an argument in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{C}}(AT)$, then r is labelled $\neg L[r].d$.

Induction step ($P(k+1)$):

- Let $l \in \mathscr{L}$ be an arbitrary literal such that each potential argument for l has a height of at most $(k+1)$ and is p-attacked by an argument that is not p-attacked by any observation-based potential argument in $P_{\mathscr{C}}(AT)$.
  - If $\boldsymbol{l} \in \mathcal{Q}$ then there is some $l' \in \bar{l}$ such that $l' \in \mathscr{K}$ (see base case) so l is labelled $\neg L[l].d$ by L-D-a.
  - Alternatively, $\boldsymbol{l} \notin \mathcal{Q}$. We distinguish two options:
    - First suppose that **there is an argument $\boldsymbol{B}$ for some $\boldsymbol{l'} \in \bar{l}$ in $\boldsymbol{Arg(AT)}$ that is not p-attacked by any observation-based potential argument in $\boldsymbol{P_{\mathcal{Q}}(AT)}$**. Given $l' \notin \mathscr{C}$, B must be rule-based: there is a rule r' for l' such that for each $a' \in \mathtt{ants}(r')$, there is an argument for a' in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{C}}(AT)$. Then by Lemma 22, each $a' \in \mathtt{ants}(r')$ is labelled $\neg \boldsymbol{L}[\boldsymbol{a'}].\boldsymbol{u}$ and $\neg \boldsymbol{L}[\boldsymbol{a'}].\boldsymbol{o}$. Given that $L_p[a'] = \langle 1,1,1,1\rangle$ for each $a' \in \mathtt{ants}(r)$ (Lemma 9), these labels must have been assigned in Algorithm 4 line 5, 11 or 15. In all cases, r' is added to TODO-SET immediately afterwards, considered for relabelling in a later iteration of the while loop (Algorithm 4 line 9) and labelled $\neg \boldsymbol{L}[\boldsymbol{r'}].\boldsymbol{u}$ and $\neg \boldsymbol{L}[\boldsymbol{r'}].\boldsymbol{o}$ by case R-U-a and R-O-a. Subsequently, l is relabelled in Algorithm 4 line 15 (recall that $l' \in \bar{l}$ implies $l \in \bar{l'}$), as $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{d}$ by case L-D-c.
    - Alternatively, **for each $\boldsymbol{l'} \in \bar{l}$**, each argument for l' in $\boldsymbol{Arg(AT)}$ is p-attacked by some observation-based potential argument in $\boldsymbol{P_{\mathcal{Q}}(AT)}$. Then each potential argument for l in $P_{\mathscr{C}}(AT)$ must be rule-based. So for each rule r for l in $\mathscr{R}$: each potential argument based on r has $\mathtt{h}(A^p) \leq k+1$ and is p-attacked by an argument in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{C}}(AT)$. By the induction hypothesis, each rule r for l in $\mathscr{R}$ is labelled $\neg L[r].d$.
      - If this happened in preprocessing, Algorithm 1 line 12 was never reached for any r for l, so line 13 was not reached either, so $L_p[l] = \langle 1,0,0,0\rangle$, which implies $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{d}$.
      - Alternatively, some rule for l that was relabelled, must have been considered by Algorithm 4 line 9, which means that l was relabelled in line 11 as $\neg \boldsymbol{L}[\boldsymbol{l}].\boldsymbol{d}$ by case L-D-b.
- Let $r \in \mathscr{R}$ be an arbitrary rule such that each potential argument A^p based on r has $\mathtt{h}(A^p) \leq (k+2)$ and is p-attacked on a subargument by an argument B in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{C}}(AT)$. Then there is at least one $a \in \mathtt{ants}(r)$ such that each potential argument A' for a has $\mathtt{h}(A') \leq k+1$ and is p-attacked by an argument B in $Arg(AT)$ and there is no observation-based potential argument in $P_{\mathscr{C}}(AT)$ that p-attacks B, which implies that a is labelled $\neg L[a].d$ (see the item above).
  - If this happened in preprocessing, then Algorithm 1 line 11 never applied for r so line 8 was never reached, hence $L_p[r] = \langle 1,0,0,0\rangle$, therefore, $\neg \boldsymbol{L}[\boldsymbol{r}].\boldsymbol{d}$.
  - Alternatively, a was labelled $\neg L[a].d$ by Algorithm 4 line 5, 11 or 15; in all cases, r was added to TODO-SET immediately afterwards and, when being popped from this set, labelled $\neg \boldsymbol{L}[\boldsymbol{r}].\boldsymbol{d}$ by R-D-a in line 9.

Finally, recall that each potential argument has finite height, so from $P(n)$ we can generalise to each $l \in \mathscr{L}$.□

Finally, Lemma 25 specifies in which situations $L[l] = \langle 0,1,0,0\rangle$. This corresponds to Item 6 of Lemma 3.

Lemma 25 (Conditions for $L[l] = \langle 0,1,0,0\rangle$). *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$, let $\mathscr{C}$ be the set of queryable literals and let L be the labelling obtained by* STABILITY-LABEL *(Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. For each literal $l \in \mathscr{L}$: if there is an argument A for l in $Arg(AT)$ and each potential argument B^p in $P_{\mathscr{C}}(AT)$ that p-attacks A is p-attacked by an observation-based argument in $Arg(AT)$, then $L[l] = \langle 0,1,0,0\rangle$.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by STABILITY-LABEL (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. We proceed by induction; in order to prove the lemma for literals, we will also prove an auxiliary statement for rules.

Proposition ($P(n)$):

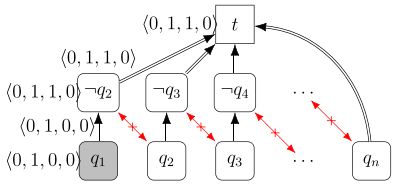


Fig. B.19. t is stable-defended in AT. However, t is labelled $L[t].o$ because the argument $[q_1 \Rightarrow \neg q_2] \Rightarrow t$ is p-attacked by the observation-based potential argument q_2. The introduction of this potential argument in AT forces a new argument for t, i.e. $[q_2 \Rightarrow \neg q_3] \Rightarrow t$. **(Case 2b).**

- For each $l \in \mathscr{L}$ for which there is an argument $A \in Arg(AT)$ for l such that $\mathtt{h}(A) \leq n$ and each potential argument in $P_{\mathscr{C}}(AT)$ attacking it is attacked by an observation-based argument in $Arg(AT)$, $L[l] = \langle 0,1,0,0 \rangle$.
- For each $r \in \mathscr{R}$ based on which there is an argument $A \in Arg(AT)$ such that $\mathtt{h}(A) \leq n+1$ and each potential argument in $P_{\mathscr{C}}(AT)$ attacking it on a subargument is attacked by an observation-based argument in $Arg(AT)$, $L[r] = \langle 0,1,0,0 \rangle$.

Base case ($P(0)$):

- For each $l \in \mathscr{L}$ for which there is an argument $A \in Arg(AT)$ for l such that $\mathtt{h}(A) = 0$, $l \in \mathscr{K}$. Then l is labelled in Algorithm 4 line 5 as $L[l] = \langle 0,1,0,0 \rangle$ by Algorithm 2 case L-U-a, L-O-a and L-B-a.
- For each $r \in \mathscr{R}$ based on which there is an argument $A \in Arg(AT)$ such that $\mathtt{h}(A) \leq 1$, each $a \in \mathtt{ants}(r)$ is in $\mathscr{K}$ and therefore labelled $L[a] = \langle 0,1,0,0 \rangle$. Afterwards, r is added to TODO-SET in line 6 and relabelled in line 9 as $L[r] = \langle 0,1,0,0 \rangle$ by case R-U-a, R-O-a and R-B-a.

Induction hypothesis ($P(k)$):

- For each $l \in \mathscr{L}$ for which there is an argument $A \in Arg(AT)$ for l such that $\mathtt{h}(A) \leq k$ and each potential argument in $P_{\mathscr{C}}(AT)$ attacking it is attacked by an observation-based argument in $Arg(AT)$, $L[l] = \langle 0,1,0,0 \rangle$.
- For each $r \in \mathscr{R}$ based on which there is an argument $A \in Arg(AT)$ such that $\mathtt{h}(A) \leq k+1$ and each potential argument in $P_{\mathscr{C}}(AT)$ attacking it on a subargument is attacked by an observation-based argument in $Arg(AT)$, $L[r] = \langle 0,1,0,0 \rangle$.

Induction step ($P(k+1)$):

- Let l be an arbitrary literal from $\mathscr{L}$ for which there is an argument $A \in Arg(AT)$ such that $\mathtt{h}(A) = k+1$ and each potential argument in $P_{\mathscr{C}}(AT)$ attacking it is attacked by an observation-based argument in $Arg(AT)$.
  - If $l \in \mathscr{K}$ then l is labelled $\boldsymbol{L[l] = \langle 0,1,0,0 \rangle}$ in line 5 by case L-U-a, L-O-a and L-B-a.
  - Alternatively, suppose that $l \notin \mathscr{K}$.

    First we show that each rule r' for each $l' \in \bar{l}$ (if any) is labelled $\neg L[r'].d$ and $\neg L[r'].b$. Let $r' \in \mathscr{R}$ be an arbitrary rule for some $l' \in \bar{l}$.
    - If there is a potential argument based on r' in $P_{\mathscr{C}}(AT)$, then there is a potential argument for each $a' \in \mathtt{ants}(r')$ in $P_{\mathscr{C}}(AT)$ (Definition 13).

      Note that there must be some $a' \in \mathtt{ants}(r')$ such that each potential argument for a' is p-attacked by an observation-based argument in $Arg(AT)$: suppose, towards a contradiction, that this is not the case. Then we could construct a potential argument B^p based on r' (hence p-attacking A on its conclusion) from the potential arguments for $a' \in \mathtt{ants}(r)$ that is not p-attacked *on a subargument* by any observation-based argument in $Arg(AT)$. Since $l' \notin \mathscr{C}$, B^p cannot be p-attacked *on its conclusion* either by any observation-based argument in $Arg(AT)$; a contradiction.

      Then by Lemma 21, there must be some antecedent $a' \in \mathtt{ants}(r')$ that is labelled $\neg L[a'].d$ and $\neg L[a'].b$. This implies that r' is labelled $\neg L[r'].d$ and $\neg L[r'].b$ (Algorithm 3 case R-D-a and R-B-b).
    - Otherwise $L[r'] = \langle 1,0,0,0 \rangle$ by Lemma 19, which implies that $\neg L[r'].d$ and $\neg L[r'].b$.

      Since r' was chosen arbitrarily, each rule r' for each $l' \in \bar{l}$ must be labelled $\neg L[r'].d$ and $\neg L[r'].b$.

      Recall that A is rule-based because $\mathtt{h}(A) = k+1$. Let r be the top rule of A. By the induction hypothesis, $L[r] = \langle 0,1,0,0 \rangle$.

      Also note that either $l \in \mathscr{C}$ or for each $l' \in \bar{l}$, there is some $l'' \in \overline{l'}$ in $\mathscr{K}$: if this would not be the case, then there would be an (observation-based) potential argument for some $l' \in \bar{l}$ in $P_{\mathscr{C}}(()AT)$ (p-attacking A) that is not attacked by any observation-based argument in $Arg(AT)$.

      After relabelling each r' for each $l' \in \bar{l}$ and r this way in Algorithm 4 line 9, l is relabelled in line 11 or 15 as $\boldsymbol{L[l] = \langle 0,1,0,0 \rangle}$ by case L-U-b, L-O-c or L-O-e and L-B-d.
- Let r be an arbitrary rule from $\mathscr{R}$ based on which there is an argument $A \in Arg(AT)$ such that $\mathtt{h}(A) \leq k+2$ and each potential argument in $P_{\mathscr{C}}(AT)$ attacking it is attacked by an observation-based argument in $Arg(AT)$. Then by the induction hypothesis, each $a \in \mathtt{ants}(r)$ is labelled $L[a] = \langle 0,1,0,0 \rangle$; this must have happened in Algorithm 4 line 5, 11 or 15; in all cases, r is added to TODO-SET immediately afterwards and labelled in a later iteration of line 9 as $L[r] = \langle 0,1,0,0 \rangle$ by case R-U-a, R-O-a and R-B-a.

At this point, we have proven $P(n)$ for each non-negative integer $n \in \mathbb{N}$. Since each argument has finite height, we can generalise to each $l \in \mathscr{L}$ and each $r \in \mathscr{R}$, which concludes our proof.□

We have now formally proven all items in Lemma 3:

Lemma 3. (*Conditions for labelling*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \_)$ and let L be the labelling after executing the* STABILITY-LABEL *algorithm on $\mathscr{L}$, $\mathscr{R}$, $\_$, $\mathscr{C}$ and $\mathscr{K}$. Let L_p be the labelling after executing* PREPROCESS *on $\mathscr{L}$, $\mathscr{R}$, $\_$, $\mathscr{C}$ and $\mathscr{K}$. Let $l \in \mathscr{L}$ be a literal. Then:*

1. *$L_p[l] = \langle 1,0,0,0 \rangle$ iff there is no potential argument A^p for l in $P_{\mathscr{C}}(AT)$.*
2. *If there is an argument for l in $Arg(AT)$, then $\neg L[l].u$.*
3. *If each potential argument for l in $P_{\mathscr{C}}(AT)$ is p-attacked by an observation-based argument in $Arg(AT)$, then $\neg L[l].d$ and $\neg L[l].b$.*
4. *There is an argument A for l in $Arg(AT)$ and there is no observation-based potential argument in $P_{\mathscr{C}}(AT)$ that p-attacks A iff $\neg L[l].u$ and $\neg L[l].o$.*

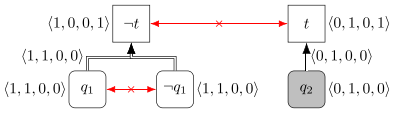


Fig. B.20. Although t is stable-defended in AT, it is labelled $L[t].b$ because the only argument for t in $Arg(AT)$ (i.e. $q_2 \Rightarrow t$) is p-attacked by the inconsistent potential argument $q_1, \neg q_1 \Rightarrow t$. **(Case 3a).**

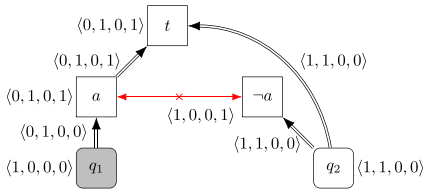


Fig. B.21. t is defended in AT. However, t is labelled $L[t].b$. The reason for this is that for each argument A for t in $Arg(AT)$ (in this case only $[q_1 \Rightarrow a] \Rightarrow t$), there is some potential argument B^p (in this case $q_2 \Rightarrow \neg a$) p-attacking A that is not p-attacked by any observation-based argument. However, the introduction of B^p in AT forces a new argument ($q_2 \Rightarrow t$). **(Case 3b).**

5. *If each potential argument A^p for l in $P_{\mathscr{C}}(AT)$ is p-attacked by an argument B in $Arg(AT)$ and there is no observation-based potential argument in $P_{\mathscr{C}}(AT)$ that p-attacks B, then $\neg L[l].d$.*
6. *If there is an argument A for l in $Arg(AT)$ and each potential argument B^p in $P_{\mathscr{C}}(AT)$ that p-attacks A is p-attacked by an observation-based argument in $Arg(AT)$, then $L[l] = \langle 0, 1, 0, 0 \rangle$.*

Proof. We have proven these items in the Lemmas 19, 20, 21, 22, 24 and 25, respectively.□

From the above result, we can derive in which situations literals are stable in the argumentation theory, but not labelled as such. First, Lemma 26 specifies the situations in which a literal l is labelled $L[l].u$ or $L[l].o$, while l is not unsatisfiable or out in any future argumentation theory (hence the label $\neg L[l].u$ and $\neg L[l].o$ would be correct).

Lemma 26 (Incorrect labelling). *$L[l].u$ or $L[l].o$) Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let L be the labelling after executing the* Stability-label *algorithm on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. Given a literal $l \in \mathscr{L}$, if for each $AT' \in F_{\mathscr{C}}(AT)$, l is not unsatisfiable or out in AT', but l is labelled $L[l].u$ or $L[l].o$, then either l is observation-unattackable in AT w.r.t. $\mathscr{C}$ or some argument for l in $Arg(AT)$ is p-attacked by an observation-based potential argument B^p such that the introduction of B^p in AT forces a new argument for l.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let $\mathscr{C}$ be a set of queryables. Let L be the labelling obtained by Stability-label (Algorithm 4) on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. Let $l \in \mathscr{L}$ be a literal such that for each $AT' \in F_{\mathscr{C}}(AT)$, l is not unsatisfiable or out in AT' and l is labelled $L[l].u$ or $L[l].o$.

Given that l is labelled $L[l].u$ or $L[l].o$, each argument for l in $Arg(AT)$ is p-attacked by an observation-based potential argument in $P_{\mathscr{C}}(AT)$ (Lemma 22). Since l is not unsatisfiable or out in any $AT' \in F_{\mathscr{C}}(AT)$, for each $AT' \in F_{\mathscr{C}}(AT)$ there is at least one argument for l in $Arg(AT')$ that is not p-attacked by an observation-based argument in $Arg(AT')$ (Lemma 7).

So each argument A for l in $Arg(AT)$ that is not attacked by an observation-based argument in $Arg(AT)$ must be p-attacked by some observation-based potential argument B^p in $P_{\mathscr{C}}(AT)$. Let S be the set of all arguments for l in $Arg(AT)$ that are not attacked by any observation-based argument in $Arg(AT)$. We consider two possibilities:

- If there exists some *consistent* $\mathscr{K}'$ such that each argument in S is attacked by an observation-based argument in $Arg((AS, \mathscr{K} \cup \mathscr{K}'))$,[9] then $AT' = (AS, \mathscr{K} \cup \mathscr{K}')$ must be in $F_{\mathscr{C}}(AT)$. Let $\mathscr{K}'$ be a minimal set with this property; this implies that each $k \in \mathscr{K}'$ is the conclusion of some observation-based potential argument in $P_{\mathscr{C}}(AT)$ that p-attacks an argument in S. Given that l is not unsatisfiable or out in AT', there must be some argument B for l in $Arg(AT')$ that is not attacked by an observation-based argument and therefore cannot be in S, so $B \notin Arg(AT)$. Then by Definition 16, some argument $A \in Arg(AT)$ for l is p-attacked by at least one observation-based potential argument B^p such that the introduction of B^p in AT **forces a new argument** for l.[10]
- Alternatively, there is no consistent $\mathscr{K}'$ such that each argument in S is attacked by an observation-based argument in $Arg((AS, \mathscr{K} \cup \mathscr{K}'))$; then there is no consistent set of observation-based potential arguments $T^p \subseteq P_{\mathscr{C}}(AT)$ such that each argument for l in $Arg(AT)$ is p-attacked by some potential argument in T^p.[11] Then by Definition 15, l is **observation-unattackable** in AT w.r.t. $\mathscr{C}$.

□

Finally, we use Lemmas 3 and 26 to specify the other situations in which Stability-label does not detect stability in Proposition 3.

Proposition 3. (*Conditional completeness stability labelling*) *Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, \text{-})$ and let L be the labelling*

[9] See Fig. 12 in the paper for an example: let S be all arguments for t in $Arg(AT)$: $S = \{[q_1 \Rightarrow q_2] \Rightarrow t\}$. It is possible to attack all these arguments by adding $\mathscr{K}' = \{\neg q_2\}$ to the knowledge base.

[10] Note that this new argument is not attacked by an observation-based argument in $Arg(AT')$, but may be attacked by an observation-based argument in some $Arg(AT'')$ where $AT'' \in F_{\mathscr{C}}(AT')$. Fig. B.19 is an example of this situation: adding q_2 to the knowledge base results in a new argument $[q_2 \Rightarrow \neg q_3] \Rightarrow t$ for t, but this argument can be attacked in a future argumentation theory by adding q_3 to the knowledge base.

[11] See Fig. 11 in the paper for an example.

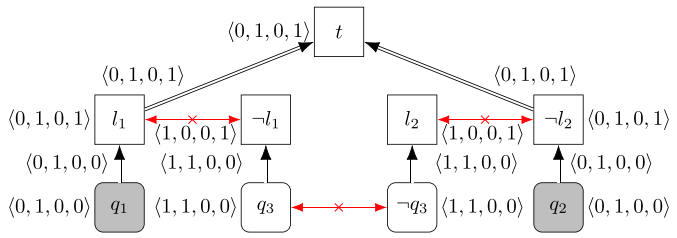


Fig. B.22. t is stable-defended in AT, but it is labelled $L[t].b$ because each argument for t in $Arg(AT)$ is p-attacked by a potential argument in $P_{\mathscr{C}}(AT)$ that is not p-attacked by an observation-based argument in $Arg(AT)$. However, there is no $AT' \in F_{\mathscr{C}}(AT)$ in which all arguments for t are attacked by an argument in $Arg(AT')$ that is not attacked by an observation-based argument. **(Case 3c)**.

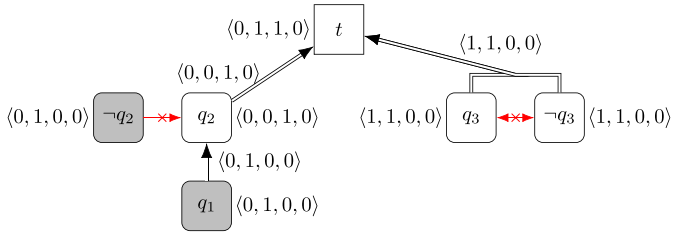


Fig. B.23. t is stable-out in AT. However, t is labelled $L[l].d$ because each potential argument for l in $P_{\mathscr{C}}(AT)$ that is not p-attacked by an observation-based argument in $Arg(AT)$ (in this case only $q_3, \neg q_3 \Rightarrow t$) is inconsistent. **(Case 4)**.

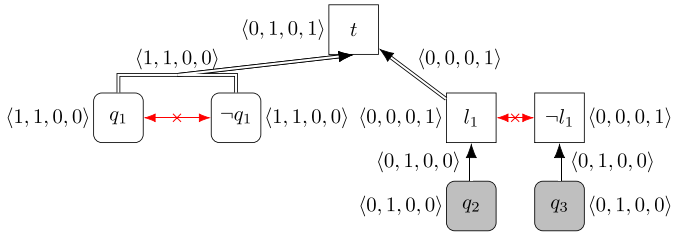


Fig. B.24. Although t is stable-blocked in AT, it is labelled $L[t].d$ because there is an inconsistent potential argument $q_1, \neg q_1 \Rightarrow t$ for t in $P_{\mathscr{C}}(AT)$ that is not p-attacked by any argument in $Arg(AT)$. **(Case 5a)**.

after executing the Stability-label *algorithm on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. Given a literal $l \in \mathscr{L}$, if l is stable in AT but l is not labelled stable by L, then some of the following five cases applies:*

1. *l is stable-unsatisfiable in AT and each potential argument for l in $P_{\mathscr{C}}(AT)$ is inconsistent.*
2. *l is stable-defended or stable-blocked in AT and:*
   (a) *l is observation-unattackable in AT w.r.t. $\mathscr{C}$,*[6] *or*
   (b) *some argument A for l in $Arg(AT)$ is p-attacked by an observation-based potential argument B^p such that the introduction of B^p in AT forces a new argument for l.*
3. *l is stable-defended in AT and there is an argument A for l in $Arg(AT)$ that is p-attacked by a potential argument in $P_{\mathscr{C}}(AT)$ that is not p-attacked by an argument in $Arg(AT)$ and:*
   (a) *each potential argument in $P_{\mathscr{C}}(AT)$ p-attacking A that is not p-attacked by an argument in $Arg(AT)$ is inconsistent; or*
   (b) *there is a consistent potential argument B^p in $P_{\mathscr{C}}(AT)$ p-attacking A that is not p-attacked by an argument in $Arg(AT)$, but the introduction of B^p in AT forces a new argument for l; or*
   (c) *l is unattackable in AT w.r.t. $\mathscr{C}$.*
4. *l is stable-out in AT and each potential argument for l in $P_{\mathscr{C}}(AT)$ that is not p-attacked by an observation-based argument in $Arg(AT)$ has inconsistent premises; or*
5. *l is stable-blocked in AT and there is a potential argument A^p for l in $P_{\mathscr{C}}(AT)$ such that each argument in $Arg(AT)$ p-attacking A^p is p-attacked by an observation-based potential argument C^p in $P_{\mathscr{C}}(AT)$ and:*
   (a) *A^p is inconsistent; or*
   (b) *the introduction of A^p in AT forces an argument attacking A^p; or*
   (c) *the introduction of C^p in AT forces an argument that p-attacks A^p.*

Proof. Let $AT = (AS, \mathscr{K})$ be an argumentation theory where $AS = (\mathscr{L}, \mathscr{R}, -)$ and $\mathscr{C}$ is a set of queryables and let L be the labelling after executing the Stability-label algorithm on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$. Let L_p be the labelling after executing Preprocess on $\mathscr{L}$, $\mathscr{R}$, -, $\mathscr{C}$ and $\mathscr{K}$ and let $l \in \mathscr{L}$ be a literal.

[12] This condition is specifically required for argumentation systems that do not have classical negation as a contrariness function.

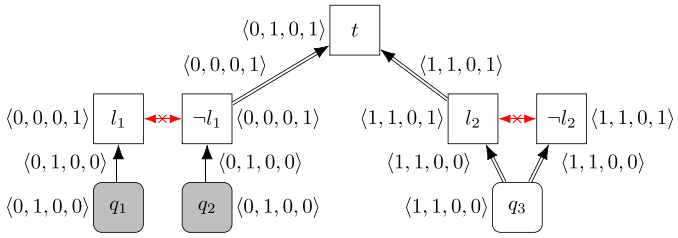


Fig. B.25. Although t is stable-blocked in AT, it is labelled $L[t].d$ because there is a potential argument $A^p : [q_3 \Rightarrow l_2] \Rightarrow t$ for t in $P_{\mathscr{C}}(AT)$ of which the introduction forces an argument attacking A^p. **(Case 5b)**.

UnsatisfiableSuppose that l is stable-unsatisfiable in AT and $L[l] \neq \langle 1,0,0,0\rangle$.[13]Then $L_p[l] \neq \langle 1,0,0,0\rangle$, so by Lemma 19, there is a potential argument for l in $P_{\mathscr{C}}(AT)$. Each potential argument for l in $P_{\mathscr{C}}(AT)$ is inconsistent, as we show next by contradiction. If there would be some consistent A^p for l in $P_{\mathscr{C}}(AT)$, then $\mathscr{K}' = \mathscr{K} \cup \mathtt{prem}(A^p)$ is consistent, since $\mathtt{prem}(A^p)$ is required to be consistent with $\mathscr{K}$ by Definition 13. Then $AT' = (AS, \mathscr{K}')$ is an argumentation theory in $F_{\mathscr{C}}(AT)$ and $A^p \in Arg(AT')$; contradiction. Thus there is a potential argument for l in $P_{\mathscr{C}}(AT)$ but each potential argument for l in $P_{\mathscr{C}}(AT)$ is inconsistent **(Case 1)**.

OutSuppose that l is stable-out in AT and $L[l] \neq \langle 0,0,1,0\rangle$.[14]$AT \in F_{\mathscr{C}}(AT)$, so l is out and therefore not unsatisfiable in AT. Then by Lemma 20, l is labelled $\neg L[l].u$. So l must be labelled $L[l].d$ or $L[l].b$, while for each $AT' \in F_{\mathscr{C}}(AT)$, l is not defended or blocked in AT'. Then by Lemma 21, there is a potential argument for l in $P_{\mathscr{C}}(AT)$ that is not p-attacked by any observation-based argument in $Arg(AT)$. Let A^p be an arbitrary potential argument with this property. Next, we show by contradiction that A^p is inconsistent.

Suppose that A^p is consistent; then $\mathscr{K} \cup \mathtt{prem}(A^p)$ is consistent. This means that $AT' = (AS, \mathscr{K} \cup \mathtt{prem}(A^p))$ is an argumentation theory in $F_{\mathscr{C}}(AT)$ and that $A \in Arg(AT')$. Since l is not defended and blocked in any future argumentation theory of AT, it must be unsatisfiable or out in AT', which by Definition 8 and Lemma 7 means that each argument for l in $Arg(AT')$ must be attacked by an observation-based argument in $Arg(AT')$. Given that A^p is not p-attacked by any observation-based argument in $Arg(AT)$, A^p cannot be attacked by any observation-based argument in $Arg(AT)$. Hence A^p must be attacked by an observation-based argument in $Arg(AT')\backslash Arg(AT)$, which means that it has its only premise in $\mathtt{prem}(A^p)$. This however contradicts with the assumed consistency of $\mathtt{prem}(A^p)$.

To conclude, each potential argument for l in $P_{\mathscr{C}}(AT)$ that is not p-attacked by an observation-based argument in $Arg(AT)$ is inconsistent. **(Case 4)**

Defended Suppose that l is stable-defended in AT and $L[l] \neq \langle 0,1,0,0\rangle$. We consider two cases.

First suppose that l is labelled $L[l].u$ or $L[l].o$. Since l is defended in AT' for each $AT' \in F_{\mathscr{C}}(AT)$, we know that for each $AT' \in F_{\mathscr{C}}(AT)$, l is not unsatisfiable or out in AT'. Then by Lemma 26, either l is observation-unattackable in AT w.r.t. $\mathscr{C}$ or some argument for l in $Arg(AT)$ is p-attacked by an observation-based potential argument B^p such that the introduction of B^p in AT forces a new argument for l **(Case 2)**.

Alternatively, l is labelled $\neg L[l].u$ and $\neg L[l].o$. By Lemma 25, the fact that $L[l] \neq \langle 0,1,0,0\rangle$ implies that each argument for l in $Arg(AT)$ is p-attacked by some potential argument in $P_{\mathscr{C}}(AT)$ that is not p-attacked by any observation-based argument in $Arg(AT)$. Given that l is labelled $\neg L[l].u$ and $\neg L[l].o$, by Lemma 23 there is an argument A for l in $Arg(AT)$ that is not p-attacked by any observation-based potential argument in $P_{\mathscr{C}}(AT)$. A is however p-attacked by some potential argument in $P_{\mathscr{C}}(AT)$ (that is not p-attacked by any observation-based argument in $Arg(AT)$), which must then be rule-based. We consider two possibilities:

One possibility would be that each potential argument in $P_{\mathscr{C}}(AT)$ p-attacking A that is not p-attacked by any observation-based argument in $Arg(AT)$ is inconsistent[15] **(Case 3a)**.

Alternatively, there exists some *consistent* potential argument B^p in $P_{\mathscr{C}}(AT)$ that p-attacks A and is not p-attacked by any observation-based argument in $Arg(AT)$. Let $b = \mathtt{conc}(B^p)$ and consider $AT' = (AS, \mathscr{K} \cup \{b\})$. There are two possibilities:

The introduction of B^p in AT forces a new argument for l[16]This corresponds to **Case 3b**.

Alternatively, the arguments for t in $Arg(AT')$ are exactly the same as the arguments for t in $Arg(AT)$. Then it cannot be possible to attack all arguments for t at the same time by arguments that are not attacked by an observation-based argument. So by Definition 15, l is unattackable in AT w.r.t. $\mathscr{C}$[17] **(Case 3c)**.

- **Blocked** Suppose that for each $AT' \in F_{\mathscr{C}}(AT)$, l is blocked in AT' and $L[l] \neq \langle 0,0,0,1\rangle$. We consider two cases.
  - First suppose that l is labelled $L[l].u$ or $L[l].o$. Since l is blocked in each $AT' \in F_{\mathscr{C}}(AT)$, it cannot be unsatisfiable or out. Then by Lemma 26, either l is observation-unattackable in AT w.r.t. $\mathscr{C}$ or some argument for l in $Arg(AT)$ is p-attacked by an observation-based potential argument B^p such that the introduction of B^p in AT forces a new argument for l **(Case 2)**.
  - Alternatively, l is labelled $\neg L[l].u$ and $\neg L[l].o$ but still $L \neq \langle 0,0,0,1\rangle$. Then l should be labelled $L[l].d$. By Lemma 24 there is a potential argument $A^p \in P_{\mathscr{C}}(AT)$ for l such that each argument B in $Arg(AT)$ that p-attacks A^p is p-attacked by some observation-based potential argument $C^p \in P_{\mathscr{C}}(AT)$. Given that l is stable-blocked in AT, $l \notin \mathscr{C}$ (otherwise there would be some future argumentation theory in which either l or some of its contradictories would be in the knowledge base). So each potential argument for l in $P_{\mathscr{C}}(AT)$ must be rule-based.
    - One possibility is that $\mathtt{prem}(A^p)$ is inconsistent.[18]This corresponds to **Case 5a**.

[13] See f in Example 9.
[14] See for example Fig. B.23: t is stable-out in AT but not labelled as such.
[15] See Fig. B.20 for an example. The only argument for t in $Arg(AT)$ is p-attacked by an inconsistent potential argument.
[16] See Fig. B.21: the introduction of each potential argument p-attacking the argument $[q_1 \Rightarrow a] \Rightarrow t$ in AT forces a new argument for t.
[17] See Fig. B.22 for an example: there are two arguments for t in $Arg(AT)$, but it is not possible to attack them both because the set of potential attackers attacking arguments for t in $Arg(AT)$ is inconsistent.
[18] In Fig. B.24 there is an inconsistent potential argument for t that is not p-attacked by any argument in $Arg(AT)$.

- Alternatively, suppose that $\texttt{prem}(A^p)$ is consistent. Then A^p is an argument in $Arg(AT')$ where $AT' = (AS, \mathscr{K} \cup \texttt{prem}(A^p))$. Given that l is stable-blocked in AT, it must be blocked in AT', so there must be some argument B in $Arg(AT')$ that attacks A^p.
  - First suppose that $B \notin Arg(AT)$. Then there is some $b \in \texttt{prem}(B)$ such that $b \in \texttt{prem}(A^p)$.[19] This implies that the introduction of A^p in AT forces an argument attacking A^p, which corresponds to **Case 5b**.
  - Alternatively, $B \in Arg(AT)$. However, given that l is labelled $L[l].d$, there must be some observation-based potential argument C^p in $P_{\mathscr{C}}(AT)$ that p-attacks B.[20] Still, $C^p \notin Arg(AT'')$ for any $AT'' \in F_{\mathscr{C}}(AT')$ (since l is blocked and not defended in AT'). Let $c = \texttt{conc}(C^p)$. So for each $AT'' = (AS, \mathscr{K}'')$ in $F_{\mathscr{C}}(AT')$ there is some $c' \in \overline{c}$ in $\mathscr{K}''$. This implies that there is some $c' \in \overline{c}$ in $\mathscr{K}' = \mathscr{K} \cup \texttt{prem}(A^p)$. Given that $C^p \in P_{\mathscr{C}}(AT)$, no $c' \in \overline{c}$ was in $\mathscr{K}$. So there is some $c' \in \overline{c}$ in $\texttt{prem}(A^p)$. By Definition 16, this implies that the introduction of C^p in AT forces an argument that attacks A^p **(Case 5c)**.

To conclude, each situation in which l is stable in AT but not labelled as such by L matches one of the five cases.□

Appendix C. Data set generation procedure

C1. Random data set

For our experiment on randomly generated data, we implemented an argumentation theory generator that can be parametrised by the language size $|\mathscr{L}|$, rule size $|\mathscr{R}|$, rule antecedent distribution (i.e. the number of rules that should have a specific number of antecedents) and queryable size $|\mathscr{C}|$. The generator can be found in our GitHub repository at https://github.com/DaphneO/StabilityLabelAlgorithm.

First, the language is generated as $[l_0..l_{n-1}] + [\neg l_0..\neg l_{n-1}]$ where $n = 0.5{\cdot}|\mathscr{L}|$. In this experiment, we only consider classical negation as contradiction function and accordingly add the contradictories $(l_i, \{\neg l_i\})$ and $(\neg l_i, \{l_i\})$ for each $i \in [0..n-1]$. Next, we sample $0.5{\cdot}|\mathscr{C}|$ literals from the positive literals of the language and add these literals, as well as their negations, to the set of queryables. After that, rules are generated one-by-one by randomly selecting a conclusion literal and a number of antecedent literals (that is dependent on the rule antecedent distribution) from the language.

In this experiment, we initialised the data set generator with a language size from $[10, 20, 50, 100, 150, 200, 250]$, a rule set size from $[10, 20, 50, 100, 150, 200, 250]$, a queryable set size of 45% of the language size (rounded to the nearest integer if necessary) and a rule antecedent distribution such that 40% of the rules (rounded up) have one antecedent, 40% (rounded up) of the rules have two antecedents and the remaining rules have three antecedents. For each of these 49 configurations, we generated 50 argumentation systems.

Given an argumentation system AS, we did not generate *all* argumentation theories in $F_{\mathscr{C}}((AS, \emptyset))$, since that would take way too long for data points with large $\mathscr{C}$, where $|F_{\mathscr{C}}((AS, \emptyset))| = 3^{0.5|\mathscr{C}|}$. Instead, we generated 5 argumentation theories for each of the $0.5{\cdot}|\mathscr{C}|$ possible knowledge base sizes. Note that this means that some argumentation theories (in particular those with a knowledge base of 0 items) will occur multiple times, while other argumentation theories are absent from the data set. For our experiment, this resulted in $5{\cdot}50{\cdot}0.5{\cdot}|\mathscr{C}|$ argumentation theories. We ran the Stability-label algorithm on each of these argumentation theories and measured the computation time.

C2. Layered data set

Our "layered" data set generator can be parametrised by the size of the language $|\mathscr{L}|$, rule set size $|\mathscr{R}|$, rule antecedent distribution and literal layer distribution (i.e. the number of literals such that the potential argument for that literal with the largest height should be at most a specific number). For example, a data set generated with the parameters: a language size of 8, rule size of 3, rule antecedent distribution of $[(1,2),(2,1)]$ and literal layer distribution of $[(0,8),(1,1),(2,1)]$ could contain an argumentation theory with the rules $l_0 \Rightarrow l_1$, $l_1 \Rightarrow l_2$ and $l_3, l_4 \Rightarrow l_2$. In this argumentation theory, the "highest" potential argument for l_2 has a height of 2; the potential argument for l_1 has a height of 1; for all other literals, only observation-based potential arguments (with a height of 0) are possible.

For this experiment, we initialised the data set generator with a language size varying from $[10, 20, 50, 100, 150, 200, 250]$; a rule set size varying from $[10, 20, 50, 100, 150, 200, 250]$; a rule antecedent distribution such that 40% of the rules (rounded up) has a single antecedent, 40% (rounded up) of the rules has two antecedents and the remaining rules have three antecedents; a literal layer distribution such that 40% (rounded up) of the literals have layer 0; 40% of the literals (rounded up) have layer 1 and the remaining literals have layer have layer 2.

References

Alfano, G., Cohen, A., Gottifredi, S., Greco, S., Parisi, F., & Simari, G. (2020). Dynamics in abstract argumentation frameworks with recursive attack and support relations. *24th European conference on artificial intelligence, ECAI 2020* (pp. 577–584).

Alfano, G., Greco, S., & Parisi, F. (2017). Efficient computation of extensions for dynamic abstract argumentation frameworks: An incremental approach. *Proceedings of the 26th international joint conference on artificial intelligence* (pp. 49–55).

Alfano, G., Greco, S., & Parisi, F. (2019). An efficient algorithm for skeptical preferred acceptance in dynamic argumentation frameworks. *Proceedings of the 28th international joint conference on artificial intelligence* (pp. 18–24).

Alfano, G., Greco, S., Parisi, F., Simari, G. I., & Simari, G. R. (2021). Incremental computation for structured argumentation over dynamic DeLP knowledge bases. *Artificial Intelligence, 300*, 103553. https://doi.org/10.1016/j.artint.2021.103553

Atkinson, K., Baroni, P., Giacomin, M., Hunter, A., Prakken, H., Reed, C., Simari, G., Thimm, M., & Villata, S. (2017). Towards artificial argumentation. *AI Magazine, 38*(3), 25–36.

Baroni, P., Caminada, M., & Giacomin, M. (2011). An introduction to argumentation semantics. *The Knowledge Engineering Review, 26*(4), 365–410.

Baumann, R., & Brewka, G. (2010). Expanding argumentation frameworks: Enforcing and monotonicity results. *Computational models of argument. Proceedings of COMMA 2010* (pp. 75–86). IOS Press.

Baumeister, D., Neugebauer, D., Rothe, J., & Schadrack, H. (2018). Verification in incomplete argumentation frameworks. *Artificial Intelligence, 264*, 1–26.

Besnard, P., Garcia, A., Hunter, A., Modgil, S., Prakken, H., Simari, G., & Toni, F. (2014). Introduction to structured argumentation. *Argument and Computation, 5*(1), 1–4.

Bex, F., & Verheij, B. (2013). Legal stories and the process of proof. *Artificial Intelligence and Law, 21*(3), 253–278.

Black, E., & Hunter, A. (2009). An inquiry dialogue system. *Autonomous Agents and Multiagent Systems, 19*(2), 173–209.

[19] See Fig. B.25 for an example. Whereas t is stable-blocked in AT, it is labelled $L[t].d$ because there is a potential argument for t of which the introduction in AT forces an argument attacking itself.

[20] See $\neg t$ in Fig. 12 for an example.

Borg, A., & Bex, F. (2021a). A basic framework for explanations in argumentation. *IEEE Intelligent Systems, 36*(2), 25–35.

Borg, A., & Bex, F. (2021b). Enforcing sets of formulas in structured argumentation. In M. Bienvenu, G. Lakemeyer, & E. Erdem (Eds.), *Proceedings of the 18th international conference on principles of knowledge representation and reasoning (KR'21)* (pp. 130–140). https://doi.org/10.24963/kr.2021/13

Borg, A., & Bex, F. (2021c). Necessary and suent explanations for argumentation-based conclusions. In J. Vejnarová, & N. Wilson (Eds.), *Lecture notes in artificial intelligence: vol. 12897. Proceedings of the 16th European conference on symbolic and quantitative approaches to reasoning with uncertainty (ECSQARU'21)* (pp. 45–58). Springer. https://doi.org/10.1007/978-3-030-86772-0_4.

Caminada, M., & Amgoud, L. (2007). On the evaluation of argumentation formalisms. *Artificial Intelligence, 171*(5–6), 286–310.

Cayrol, C., de Saint-Cyr, F. D., & Lagasquie-Schiex, M.-C. (2010). Change in abstract argumentation frameworks: Adding an argument. *Journal of Artificial Intelligence Research, 38*, 49–84.

Cerutti, F., Thimm, M., & Vallati, M. (2020). An experimental analysis on the similarity of argumentation semantics. *Argument and Computation, 11*(3), 1–36.

Chen, H., Liu, X., Yin, D., & Tang, J. (2017). A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter, 19*(2), 25–35.

Craandijk, D., & Bex, F. (2020). Deep learning for abstract argumentation semantics. *Proceedings of the twenty-ninth international joint conference on artificial intelligence* (pp. 1667–1673). https://doi.org/10.24963/ijcai.2020/231

Cyras, K., Rago, A., Albini, E., Baroni, P., & Toni, F. (2021). Argumentative XAI: A survey. CoRR abs/2105.11266https://arxiv.org/abs/2105.11266.

Doutre, S., & Mailly, J.-G. (2018). Constraints and changes: A survey of abstract argumentation dynamics. *Argument and Computation, 9*, 223–248. https://doi.org/10.3233/AAC-180425

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence, 77*, 321–357.

Dvořák, W., & Dunne, P. E. (2017). Computational problems in formal argumentation and their complexity. *IfCoLog Journal of Logic and its Applications, 4*, 2557–2622.

European Commission (2021). Proposal for a regulation of the European Parliament and of the council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain union legislative acts. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206, [Online; accessed 17 September 2021].

Falappa, M., Kern-Isberner, G., & Simari, G. (2009). Belief revision and argumentation theory. In G. Simari, & I. Rahwan (Eds.), *Argumentation in artificial intelligence* (pp. 341–360). Springer. https://doi.org/10.1007/978-0-387-98197-0_17.

Fan, X., & Toni, F. (2012). Agent strategies for ABA-based information-seeking and inquiry dialogues. *Proceedings of the 20th european conference on artificial intelligence* (pp. 324–329).

Hecham, A., Bisquert, P., & Croitoru, M. (2018). On a flexible representation for defeasible reasoning variants. *Proceedings of the 17th international conference on autonomous agents and multiagent systems* (pp. 1123–1131).

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition, 2nd edition*. Pearson.

Mailly, J.-G., & Rossit, J. (2020). Stability in abstract argumentation. *NMR 2020 workshop notes* (pp. 93–99).

Modgil, S., & Prakken, H. (2012). Resolutions in structured argumentation. *Computational models of argument. Proceedings of COMMA 2012* (pp. 310–321).

Modgil, S., & Prakken, H. (2013). A general account of argumentation with preferences. *Artificial Intelligence, 195*, 361–397.

Modgil, S., & Prakken, H. (2018). Abstract rule-based argumentation. In P. Baroni, D. Gabbay, M. Giacomin, & L. van der Torre (Eds.), *vol. 1. Handbook of formal argumentation* (pp. 286–361). College Publications.

Nieuwenhuizen, E. (2020). Artificiële intelligentie, is dat wel te vertrouwen? een experimentele studie naar het effect van uitleg over beslissingen van het intelligente aangiftesysteem van de politie op het vertrouwen van burgers in deze beslissingen. (*Can we trust Artificial Intelligence? An experimental study into the effect of explaining decisions of the police's intelligent intake system on citizen trust.*). Master's thesis. Utrecht University.

Niskanen, A., & Järvisalo, M. (2020). Algorithms for dynamic argumentation frameworks: An incremental sat-based approach. *Proceedings of the 24th European conference on artificial intelligence* (pp. 849–856). IOS Press.

Odekerken, D., & Bex, F. (2020). Towards transparent human-in-the-loop classification of fraudulent web shops. In S. Villata, J. Harašta, & P. Křemen (Eds.), *Frontiers in artificial intelligence and applications: vol. 334. Proceedings of the 33rd international conference on legal knowledge and information systems (JURIX'20)* (pp. 239–242). IOS Press.

Odekerken, D., Borg, A., & Bex, F. (2020). Estimating stability for efficient argument-based inquiry. In H. Prakken, S. Bistarelli, F. Santini, & C. Taticchi (Eds.), *Frontiers in artificial intelligence and applications: vol. 326. Computational models of argument. Proceedings of COMMA 2020* (pp. 307–318). IOS Press.

Odekerken, D., Borg, A., & Bex, F. (2022). Stability and relevance in incomplete argumentation frameworks (forthcoming). *Computational models of argument. Proceedings of COMMA 2022*.

Paek, T., & Pieraccini, R. (2008). Automating spoken dialogue management design using machine learning: An industry perspective. *Speech Communication, 50*(8–9), 716–729.

Parsons, S., Wooldridge, M., & Amgoud, L. (2002). An analysis of formal inter-agent dialogues. *Proceedings of the first international joint conference on autonomous agents and multiagent systems* (pp. 394–401). ACM.

Prakken, H. (2010). An abstract framework for argumentation with structured arguments. *Argument and Computation, 1*(2), 93–124.

Prakken, H., & Vreeswijk, G. (2001). Logics for defeasible argumentation. In D. Gabbay, & F. Guenthner (Eds.), *vol. 4. Handbook of philosophical logic* (pp. 219–318). Springer.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*(5), 206–215.

Schraagen, M., & Bex, F. (2019). Extraction of semantic relations in noisy user-generated law enforcement data. *13th international conference on semantic computing* (pp. 79–86). IEEE.

Schraagen, M., Bex, F., Odekerken, D., & Testerink, B. (2019). Argumentation-driven information extraction for online crime reports. *Proceedings of the international workshop on legal data analysis and mining (LEDAM'18)* (pp. 20–25). CEUR-WS.

Schraagen, M., Brinkhuis, M., & Bex, F. (2017). Evaluation of named entity recognition in Dutch online criminal complaints. *Computational Linguistics in The Netherlands Journal, 7*, 3–16.

Snaith, M., & Reed, C. (2016). Argument revision. *Journal of Logic and Computation, 27*(7), 2089–2134.

Testerink, B., Odekerken, D., & Bex, F. (2019a). Ai-assisted message processing for the netherlands national police. In K. Branting (Ed.), *Proceedings of the ICAIL 2019 workshop on AI and the administrative state (AIAS'19)* (pp. 10–13). CEUR-WS.

Testerink, B., Odekerken, D., & Bex, F. (2019b). A method for efficient argument-based inquiry. In A. Cuzzocrea, S. Greco, H. L. Larsen, D. Saccà, T. Andreasen, & H. Christiansen (Eds.), *Proceedings of the 13th international conference on flexible query answering systems (FQAS'19)* (pp. 114–125). Springer.

Vassiliades, A., Bassiliades, N., & Patkos, T. (2021). Argumentation and explainable artificial intelligence: A survey. *The Knowledge Engineering Review, 36*, e5. https://doi.org/10.1017/S0269888921000011

Walton, D., & Krabbe, E. C. (1995). *Commitment in dialogue: Basic concepts of interpersonal reasoning*. State University of New York Press.

Wu, Y., & Caminada, M. (2010). A labelling-based justification status of arguments. *Studies in Logic, 3*(4), 12–29.