Computing the justification status of literals in polynomial time

Daphne Odekerken[1,2], Floris Bex[1,3], AnneMarie Borg[1], and Bas Testerink[2]

[1]Department of Information and Computing Sciences, Utrecht University
[2]National Police Lab AI, Netherlands Police
[3]Tilburg Institute for Law, Technology and Society, Tilburg University

1 Preliminaries

We will study the problem of labelling justification status in the context of abstract 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.

1.1 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.

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

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

- $\mathcal{L}$ is a finite logical language consisting of propositional literals.
- $\mathcal{R}$ is a finite set of defeasible rules of the form $a_1, \ldots, a_m \Rightarrow c$ such that $\{a_1, \ldots, a_m, c\} \subseteq \mathcal{L}$, 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.
- $\overline{}$ is a contradiction function from $\mathcal{L}$ to $2^{\mathcal{L}}$. l is a *contradictory* of m iff $m \in \overline{l}$ and $l \in \overline{m}$. Each $l \in \mathcal{L}$ has at least one contradictory. For each $l \in \mathcal{L} : l \notin \overline{l}$.

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

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

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

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

- c if $c \in \mathcal{K}$. Then A is an **observation-based argument**.
  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 $\{c\}$.
  The set of direct subarguments $\texttt{dirsub}(A)$ of A is $\emptyset$.
  The height $\texttt{h}(A)$ of A is 0.

- $A_1, \ldots, A_m \Rightarrow c$ if for each $i \in [1 \mathrel{..} m]$: 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 $\mathcal{R}$. Then A is a **rule-based argument**.
  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 $\texttt{top-rule}(A)$ is r.
  The set of subarguments $\texttt{sub}(A)$ of A is $\{A_1, \ldots, A_m\}$.
  The height $\texttt{h}(A)$ of A is $\texttt{h}(A) = 1 + \max(\texttt{h}(A_1), \ldots, \texttt{h}(A_m))$.

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".

Apart from the standard properties of arguments as defined in ASPIC⁺, the definition above also defines the notions of argument height and direct subarguments. Intuitively, the *height* of an argument is the maximum number of inferences between a premise and the conclusion of the argument. Given a rule-based argument, the arguments for the antecedents of the rule on which the arguments is based are the *direct subarguments*. 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.

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 $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$. 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 $B''_1, \ldots, B''_n \Rightarrow l$.

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

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

1.2 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 $AF = \langle \mathcal{A}, \mathcal{C} \rangle$ be an argumentation framework and $S \subseteq \mathcal{A}$. Then:

- S is **conflict free** iff for each $X, Y \in S : (X, Y) \notin \mathcal{C}$.

- $X \in \mathcal{A}$ is **acceptable with respect to** S iff for each $Y \in \mathcal{A}$ such that $(Y, X) \in \mathcal{C}$, there is a $Z \in S$ such that $(Z, Y) \in \mathcal{C}$.

- 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 $X \in \mathcal{A}$ 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 $G(AT)$.

1.3 Justification status of statements

Based on the presence or absence of arguments for a literal in the grounded extension, we distinguish four justification statuses for literals, 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 $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$ and let $AF = \langle \mathcal{A}, \mathcal{C} \rangle$ be the argumentation framework defined by AT. Then the justification status of $l \in \mathcal{L}$ in AT is:

- **unsatisfiable** iff there is no argument for l in $\mathcal{A}$;
- **defended** iff there exists an argument for l in $\mathcal{A}$, which is also in the grounded extension $G(AT)$;
- **out** iff there exists an argument for l in $\mathcal{A}$, but each argument for l in $\mathcal{A}$ is attacked by an argument in the grounded extension $G(AT)$;
- **blocked** iff there exists an argument for l in $\mathcal{A}$, 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].

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

2 Labelling algorithm

In this section, we propose our algorithm for computing the justification status of literals. Subsequently, we show soundness and completeness and study the computational complexity of this algorithm.

2.1 Justification status algorithm

2.1.1 Preprocessing

Our algorithm starts with a preprocessing step (Algorithm 1. In this step, the literals l for which there is an argument in *Arg*(AT) are labelled $L[l] = \langle 0, 1, 1, 1 \rangle$, while all other literals are labelled $\langle 1, 0, 0, 0 \rangle$ by L. The algorithm has a time complexity of $\mathcal{O}(|\mathcal{L}| \cdot |\mathcal{R}|^2)$ (Lemma 1) and is sound and complete (Lemma 2 and 3). In this section, we only provide proof sketches; for full proofs, we refer to Section 3.2.

Lemma 1 (Time complexity PREPROCESS)**.** *The time complexity of* PREPROCESS *is* $\mathcal{O}(|\mathcal{L}| \cdot |\mathcal{R}|^2)$.

Proof sketch. The runtime of PREPROCESS is dominated by line 11, which needs to check all (at most $|\mathcal{L}|$) antecedents in each of the ($|\mathcal{R}|$) iterations of the for loop in lines 10–14 in each of the iterations of the while loop in lines 8–14. Thanks to the *Change* condition in line 8, the while loop has at most $|\mathcal{R}|$ iterations, since each rule label can change at most once from $\langle 1, 0, 0, 0 \rangle$ to $\langle 0, 1, 1, 1 \rangle$. □

Algorithm 1 Preprocessing step

```
1:  procedure PREPROCESS($\mathcal{L}, \mathcal{R}, ^{-}, \mathcal{K}$)
2:      for Literal $l$ in $\mathcal{L}$ do
3:          if $l \in \mathcal{K}$ then $L[l] = \langle 0,1,1,1 \rangle$
4:          else $L[l] = \langle 1,0,0,0 \rangle$
5:      for Rule $r$ in $\mathcal{R}$ do
6:          $L[r] = \langle 1,0,0,0 \rangle$
7:      $Change$ = True
8:      while $Change$ do
9:          $Change$ = False
10:         for Rule $r$ in $\mathcal{R}$ do
11:             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
12:                 $L[r] = \langle 0,1,1,1 \rangle$
13:                 $L[\texttt{cons}(r)] = \langle 0,1,1,1 \rangle$
14:                 $Change$ = True
15:     return $L$
```

Lemma 2 (Soundness preprocessing step)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, ^{-})$. Furthermore let L_p be the labelling obtained by* PREPROCESS *(Algorithm 1) on $\mathcal{L}$, $\mathcal{R}$, $^{-}$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if $L_p[l] = \langle 1,0,0,0 \rangle$, then there is no argument for l in AT. For each $r \in \mathcal{R}$: if $L_p[r] = \langle 1,0,0,0 \rangle$, then there is no argument based on r in AT.*

Proof sketch. This can be shown by contraposition: if there is some argument A for some $l \in \mathcal{L}$ or based on some $r \in \mathcal{R}$ in AT, then l or r is labelled $\langle 0,1,1,1 \rangle$. If A is observation-based, then $L_p[l] = \langle 0,1,1,1 \rangle$ 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 $L_p[a] = \langle 0,1,1,1 \rangle$, hence the label $L_p[r] = \langle 0,1,1,1 \rangle$ is assigned by line 13 and $L_p[l] = \langle 0,1,1,1 \rangle$ is assigned by line 14. In both cases, l or r is not labelled $\langle 1,0,0,0 \rangle$ by L_p. □

Lemma 3 (Completeness preprocessing step)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, ^{-})$. Furthermore let L_p be the labelling obtained by* PREPROCESS *(Algorithm 1) on $\mathcal{L}$, $\mathcal{R}$, $^{-}$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if there is no argument for l in AT, then $L_p[l] = \langle 1,0,0,0 \rangle$. For each $r \in \mathcal{R}$: if there is no argument based on r in AT, then $L_p[r] = \langle 1,0,0,0 \rangle$.*

Proof sketch. This can be shown by contraposition: if some $l \in \mathcal{L}$ is labelled $L_p[l] = \langle 0,1,1,1 \rangle$ then there is an argument for l in $Arg(AT)$; if some $r \in \mathcal{R}$ is labelled $L_p[r] = \langle 0,1,1,1 \rangle$ then there is an argument based on r in $Arg(AT)$. For those $l \in \mathcal{L}$ that are labelled $\langle 0,1,1,1 \rangle$ before the first iteration of the for loop, hence in line 3, there is an observation-based argument for l since $l \in \mathcal{K}$. For all $r \in \mathcal{R}$ labelled $\langle 0,1,1,1 \rangle$, each $a \in \texttt{ants}(r)$ must be labelled $L_p[a] = \langle 0,1,1,1 \rangle$ in an earlier iteration of the for loop (line 10–14), so there is an argument for each $a \in \texttt{ants}(r)$, so there is an argument based on r in $Arg(AT)$. For those $l \in \mathcal{L}$ that are labelled $\langle 0,1,1,1 \rangle$ in the for loop, there is some rule r with $\texttt{cons}(r) = l$ such that there is an argument based on r, so there is an argument for l in $Arg(AT)$. □

2.1.2 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 Algorithm 2 and Algorithm 3.

Proposition 1 (Time complexity JUSTIFICATION-LABEL)**.** *The time complexity of* JUSTIFICATION-LABEL *is $\mathcal{O}(|\mathcal{L}|^3 \cdot |\mathcal{R}| + |\mathcal{L}|^2 \cdot |\mathcal{R}|^2)$.*

Proof sketch. The runtime of JUSTIFICATION-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

Algorithm 2 RELABEL-LITERAL procedure

```
1: procedure RELABEL-LITERAL($\mathcal{L}, \mathcal{R}, \overline{\phantom{x}}, \mathcal{K}, L, l$)
2:     ▷ Labelling rules turning $L[l].d$ to False
3:     if some $l' \in \bar{l}$ is in $\mathcal{K}$ then $L[l].d$ = False                                          ▷ L-D-a
4:     if $l \notin \mathcal{K}$ then
5:         if for each rule $r$ for $l$: $\neg L[r].d$ then $L[l].d$ = False                                   ▷ L-D-b
6:         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

7:     ▷ Labelling rules turning $L[l].o$ to False
8:     if $l \in \mathcal{K}$ then $L[l].o$ = False                                                         ▷ L-O-a
9:     if for each $l' \in \bar{l}$, $l \notin \mathcal{K}$ then
10:        if for each rule $r$ for $l$: $\neg L[r].o$ then $L[l].o$ = False                                  ▷ L-O-b
11:        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

12:    ▷ Labelling rules turning $L[l].b$ to False
13:    if $l \in \mathcal{K}$ then $L[l].b$ = False                                                          ▷ L-B-a
14:    if some $l' \in \bar{l}$ is in $\mathcal{K}$ then $L[l].b$ = False                                     ▷ L-B-b
15:    if for each rule $r$ for $l$: $\neg L[r].d$ and $\neg L[r].b$ then $L[l].b$ = False                   ▷ L-B-c
16:    if for each $l' \in \bar{l}$: for each rule $r'$ for $l'$: $\neg L[r'].d$ and $\neg L[r'].b$ then
17:        if for each rule $r$ for $l$: $\neg L[r].b$ then $L[l].b$ = False                                 ▷ L-B-d
18:        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-e
19:    return $L$
```

Algorithm 3 RELABEL-RULE procedure

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

in the worst case requires checking the presence of that literal and all of its (at most $|\mathcal{L}|$) contradictories in $\mathcal{K}$, as well as the labels of all (max $|\mathcal{R}|$) rules for that literal or any of its contradictories. Line 15 is executed at most $|\mathcal{L}|$ 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 three times (i.e. at most three booleans can be turned to False), each rule $r \in \mathcal{R}$ is recolored at most $4 \cdot |\mathtt{ants}(r)|$ times. This means that line 15 is executed at most $4 \cdot |\mathcal{L}|^2 \cdot |\mathcal{R}|$ times. Then the total time required for all iterations of line 15 is at most $4 \cdot c \cdot (|\mathcal{L}|^3 \cdot |\mathcal{R}| + |\mathcal{L}|^2 \cdot |\mathcal{R}|^2)$, where c is a positive constant. □

Given that the preprocessing step is sound, we now need to show that the remainder of the JUSTIFICATION-LABEL algorithm is sound as well.

Proposition 2 (Soundness justification labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b \rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if $L[l] = \langle 1, 0, 0, 0 \rangle$ (resp. $\langle 0, 1, 0, 0 \rangle$, $\langle 0, 0, 1, 0 \rangle$, $\langle 0, 0, 0, 1 \rangle$) then l is unsatisfiable (resp. defended, out, blocked) in AT.*

Proof sketch. The following items will be used in the soundness proof sketch and can be shown by induction:

1. For each $r \in \mathcal{R}$ labelled $\neg L[r].d$ and $\neg L[r].b$: each argument based on r in $Arg(AT)$ is attacked by some argument in $G(AT)$ (cf. Lemma 8);

Algorithm 4 Labelling procedure JUSTIFICATION-LABEL

```
procedure JUSTIFICATION-LABEL($\mathcal{L}, \mathcal{R}, \overline{\phantom{x}}, \mathcal{K}$)
    $L$ = PREPROCESS($\mathcal{L}, \mathcal{R}, \overline{\phantom{x}}, \mathcal{K}$)
    TODO-SET = empty set
    for Literal $l$ in $\mathcal{L}$ do
        $L$ = RELABEL-LITERAL($\mathcal{L}, \mathcal{R}, \overline{\phantom{x}}, \mathcal{K}, L, l$)
        Add all rules having $l$ as antecedent to TODO-SET
    while TODO-SET is not empty do
        Pop a rule $r$ from TODO-SET
        $L$ = RELABEL-RULE($\mathcal{L}, \mathcal{R}, \overline{\phantom{x}}, \mathcal{Q}, \mathcal{K}, L, r$)
        if $r$'s label changed then
            $L$ = RELABEL-LITERAL($\mathcal{L}, \mathcal{R}, \overline{\phantom{x}}, \mathcal{K}, L, \texttt{cons}(r)$)
            if $\texttt{cons}(r)$'s label changed then
                Add all rules having $\texttt{cons}(r)$ as antecedent to TODO-SET
            for $l' \in \overline{\texttt{cons}(r)}$ do
                $L$ = RELABEL-LITERAL($\mathcal{L}, \mathcal{R}, \overline{\phantom{x}}, \mathcal{K}, L, l'$)
                if $l'$'s label changed then
                    Add all rules having $l'$ as antecedent to TODO-SET
    return $L$
```

2. For each $r \in \mathcal{R}$ labelled $\neg L[r].u$ and $\neg L[r].o$: there is an argument based on r in *Arg*(AT) that is not attacked by any argument in $G(AT)$ (cf. Lemma 7);

3. For each $l \in \mathcal{L}$ labelled $\neg L[l].d$: there is no argument for l in $G(AT)$ (cf. Lemma 12).

We consider all four justification statuses:

- If $\boldsymbol{L[l] = \langle 1, 0, 0, 0 \rangle}$ then it was already labelled as such by PREPROCESS ($L_p[l] = \langle 1, 0, 0, 0 \rangle$) given $L[l].u$; then by Lemma 2 ***l* is unsatisfiable in *AT*** (cf. Lemma 9).

- 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 defended in AT. Furthermore, by Item 1 above, 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 $G(AT)$. So ***l* is defended in *AT*** (cf. Lemma 10).

- Suppose $\boldsymbol{L[l] = \langle 0, 0, 1, 0 \rangle}$. Given that $\neg L[l].u$, there is some argument for l in *Arg*(AT). We distinguish two cases. If some $l' \in \bar{l}$ is in $\mathcal{K}$ then each argument for l is attacked on its conclusion by an argument in $G(AT)$. Alternatively, all rules r for l are labelled $\neg L[r].d$ and $\neg L[r].b$. By Item 1 above all arguments based on rules for l must be attacked by an argument in $G(AT)$. To conclude, ***l* is out in *AT*** (cf. Lemma 11).

- If $\boldsymbol{L[l] = \langle 0, 0, 0, 1 \rangle}$ then $l \notin \mathcal{K}$ and there is some rule r for l that is labelled $\neg L[r].u$ and $\neg L[r].o$. By Item 2 above, there is an argument based on r in *Arg*(AT) that is not attacked by any argument in $G(AT)$. Furthermore, by Item 3 above there is no argument for l in $G(AT)$. As a result, ***l* is blocked in *AT*** (cf. Lemma 13). □

Finally, JUSTIFICATION-LABEL is complete, as we show in the next proposition.

Proposition 3 (Completeness justification labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b \rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if $L[l] = \langle 1, 0, 0, 0 \rangle$ (resp. $\langle 0, 1, 0, 0 \rangle$, $\langle 0, 0, 1, 0 \rangle$, $\langle 0, 0, 0, 1 \rangle$) then l is unsatisfiable (resp. defended, out, blocked) in AT.*

Proof sketch. The following items will be used in the soundness proof sketch and can be shown by induction:

1. Each $r \in \mathcal{R}$ such that each argument based on r in *Arg*(AT) is attacked on a subargument by some argument in $G(AT)$ is labelled $\neg L[r].d$ and $\neg L[r].b$ (cf. Lemma 16);

2. Each $r \in \mathcal{R}$ based on which there is an argument in $Arg(AT)$ that is not attacked on a subargument by any argument in $G(AT)$ is labelled $\neg L[r].u$ and $\neg L[r].o$ (cf. Lemma 15);

3. Each $l \in \mathcal{L}$ for which there is no argument in $G(AT)$ is labelled $\neg L[l].d$ (cf. Lemma 19).

We consider all four justification statuses:

- If $\boldsymbol{l}$ **is unsatisfiable in** $\boldsymbol{AT}$, then there is no argument for l in $Arg(AT)$, so $L_p[l] = \langle 1, 0, 0, 0 \rangle$ (Lemma 3) so $\boldsymbol{L[l] = \langle 1, 0, 0, 0 \rangle}$ (cf. Lemma 14).
- Alternatively, there is an argument for l in $Arg(AT)$, so by Lemma 2, $L_p[l] = \langle 0, 1, 1, 1 \rangle$.
  - If $\boldsymbol{l}$ **is defended in** $\boldsymbol{AT}$, then there is an argument for l in $Arg(AT)$ that is not attacked by any observation-based argument in $Arg(AT)$. Either $l \in \mathcal{K}$ (then $L[l] = \langle 0, 1, 0, 0 \rangle$) or there is a rule r for l such that there is an argument based on r that is not attacked by any observation-based argument in $Arg(AT)$. In the latter case, for each $l' \in \bar{l}$, $l' \notin \mathcal{K}$ and for each rule r' for l', each argument based on r' in $Arg(AT)$ is attacked by some argument in $G(AT)$; additionally, for each $a \in \texttt{ants}(r)$, a is defended in AT. Then by induction, for each $a \in \texttt{ants}(r)$: $L[a] = \langle 0, 1, 0, 0 \rangle$. By Item 1 above, for each $l' \in \bar{l}$, each r' for l' is labelled $\neg L[r'].d$ and $\neg L[r'].b$. Then $L[l] = \langle 0, 1, 0, 0 \rangle$ by case L-O-c and L-B-e. So in both cases, $\boldsymbol{L[l] = \langle 0, 1, 0, 0 \rangle}$ (cf. Lemma 17).
  - If $\boldsymbol{l}$ **is out in** $\boldsymbol{AT}$, then each argument for l is attacked by an observation-based argument in $Arg(AT)$. Either some $l' \in \bar{l}$ is in $\mathcal{K}$, which means that $L[l] = \langle 0, 0, 1, 0 \rangle$ by case L-D-a and L-B-b. Otherwise, for each r for l, each argument for l in $Arg(AT)$ is attacked on a subargument by an observation-based argument in $Arg(AT)$, so by Item 1 above, each rule r for l is labelled $\neg L[r].d$ and $\neg L[r].b$. Then $L[l] = \langle 0, 0, 1, 0 \rangle$ by case L-D-b and L-B-c. So in both cases, $\boldsymbol{L[l] = \langle 0, 0, 1, 0 \rangle}$ (cf. Lemma 18).
  - If $\boldsymbol{l}$ **is blocked in** $\boldsymbol{AT}$, then no argument for l is in $G(AT)$, so by Item 3, l is labelled $\neg L[l].d$ and $l \notin \mathcal{K}$. In addition, at least one argument for l is not attacked by an argument in $G(AT)$. Since this argument must be rule-based, there is some r for l such that some argument based on r in $Arg(AT)$ is not attacked by an argument in $G(AT)$; neither on its conclusion (so no $l' \in \bar{l}$ is in $\mathcal{K}$), nor on a subargument. By Item 2 above, this rule must be labelled $\neg L[r].u$ and $\neg L[r].o$. Then by case L-O-c, l is labelled $\neg L[l].o$, which implies that $\boldsymbol{L[l] = \langle 0, 0, 0, 1 \rangle}$ (cf. Lemma 20). □

3 Proofs

In this section, we prove the lemmas required for completing the proofs of Propositions 1, 2 and 3 on time complexity, soundness and completeness of JUSTIFICATION-LABEL. In order to do this, 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 1, in Section 3.1. Then we give proofs of time complexity, soundness and completeness of PREPROCESS in Section 3.2. The proofs for the JUSTIFICATION-LABEL algorithm follow in Section 3.3.

3.1 Specification "in" and "out" the grounded extension

In Lemmas 4 and 5, we give a more precise specification of which arguments are either in the grounded extension (Lemma 4) or attacked by an argument in the grounded extension (Lemma 5). These specifications will be useful for many proofs in this paper.

Lemma 4 (Specification "in" grounded extension)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory with argumentation system $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$. 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 defended by $G(AT)$, 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: $\boldsymbol{S = \{C \in G(AT) \mid}$ **there is no** $\boldsymbol{C' \in \mathsf{sub}(C)}$ **s.t.** $\boldsymbol{C'}$ **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 $A' \in \mathsf{sub}(A)$ such that $\mathsf{conc}(B) \in \overline{\mathsf{conc}(A')}$ and $\mathsf{conc}(A') \notin \mathcal{K}$. Given that B cannot be observation-based (otherwise B would be in $G(AT)$, which contradicts the conflict-freeness of $G(AT)$) and $\mathsf{conc}(A') \in \overline{\mathsf{conc}(B)}$ (by the symmetry of $^{-}$), A' attacks B. This implies that A cannot be in S, hence $\boldsymbol{S \subset G(AT)}$. Next, we prove that S is complete.

- $S \subset G(AT)$ and $G(AT)$ is conflict-free, so $\boldsymbol{S}$ **is conflict-free**.
- $\boldsymbol{S}$ **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 $D' \in \mathsf{sub}(D)$ such that $\mathsf{conc}(E) \in \overline{\mathsf{conc}(D')}$. Since $D \in G(AT)$, which is complete, there must be some argument in $Arg(AT)$ that attacks E, which means that $\mathsf{conc}(E) \notin \mathcal{K}$. Furthermore, since the contradiction is symmetric, $\mathsf{conc}(D') \in \overline{\mathsf{conc}(E)}$. 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.** $\boldsymbol{S}$ **is in** $\boldsymbol{S}$: 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 $D' \in \overline{\mathsf{sub}(D)}$ such that D' attacks B. Let $B' \in \mathsf{sub}(B)$ be the subargument on which D' attacks B: $\mathsf{conc}(D') \in \overline{\mathsf{conc}(B')}$. By an earlier assumption, B is not attacked by an observation-based argument, so $\mathsf{conc}(D') \notin \mathcal{K}$. Furthermore, by symmetry of contradiction, $\mathsf{conc}(B') \in \overline{\mathsf{conc}(D')}$, 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 5 (Specification "out" arguments)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory with argumentation system $AS = (\mathcal{L}, \mathcal{R}, ^{-})$. 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 $B \in G(AT)$, since B cannot be attacked. 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 $A' \in \mathsf{sub}(A)$ such that $\mathsf{conc}(B) \in \overline{\mathsf{conc}(A')}$ and therefore, by the symmetry of $^{-}$, $\mathsf{conc}(A') \in \overline{\mathsf{conc}(B)}$, $\mathsf{conc}(A') \notin \mathcal{K}$ and $\mathsf{conc}(B) \notin \mathcal{K}$, hence A' attacks B. $B \in G(AT)$, so by Lemma 4, 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. □

3.2 Proofs w.r.t. preprocessing

In this section, we prove that PREPROCESS (Algorithm 1 has a time complexity of $\mathcal{O}(|\mathcal{L}| \cdot |\mathcal{R}|^2)$ (Lemma 1) and is sound and complete (Lemma 2 and 3).

Lemma 1 (Time complexity PREPROCESS)**.** *The time complexity of* PREPROCESS *is* $\mathcal{O}(|\mathcal{L}| \cdot |\mathcal{R}|^2)$.

Proof. In the following, $c_2 \ldots c_{15}$ are positive constants. We consider each iteration of each line of Algorithm 1:

- Line 2–4 take constant time $(c_2 + c_3 + c_4)$ for each of the $|\mathcal{L}|$ iterations, so this requires at most $(c_2 + c_3 + c_4) \cdot |\mathcal{L}|$ operations in total;
- Line 5–6 require $(c_5 + c_6) \cdot |\mathcal{R}|$ iterations;

- Line 7 can be done in constant time c_7;
- The while loop (line 8–14) iterates until no label changed in the previous loop. Thanks to the check $L[r] = \langle 1,0,0,0\rangle$ in line 11, each rule can change label at most once, hence the while-loop iterates at most $|\mathcal{R}|$ times.
  - All iterations of line 8 and 9 in total take at most $(c_8 + c_9) \cdot |\mathcal{R}|$ time;
  - The for loop iterates $|\mathcal{R}|$ times for each iteration of the while loop, so lines 10–14 are executed at most $|\mathcal{R}|^2$ times in total.
    * A single execution of line 10 takes constant time c_{10}, hence all executions take at most $c_{10} \cdot |\mathcal{R}|^2$ time;
    * Line 11 checks all antecedents of each rule, which requires at most $c_{11} \cdot |\mathcal{L}|$ checks per execution – or $c_{11} \cdot |\mathcal{L}| \cdot |\mathcal{R}|^2$ operations in total;
    * All iterations of line 12–14 take at most $(c_{12} + c_{13} + c_{14}) \cdot |\mathcal{R}|^2$ operations in total;
- Line 15 takes constant time c_{15}.

The total time required for Algorithm 1 is $(c_7 + c_{15}) + (c_2 + c_3 + c_4) \cdot |\mathcal{L}| + (c_5 + c_6 + c_8 + c_9) \cdot |\mathcal{R}| + (c_{10} + c_{12} + c_{13} + c_{14}) \cdot |\mathcal{R}|^2 + c_{11} \cdot |\mathcal{L}| \cdot |\mathcal{R}|^2$ time. As a result, the time complexity of the preprocessing step must be $\mathcal{O}(|\mathcal{L}| \cdot |\mathcal{R}|^2)$. □

Lemma 2 (Soundness preprocessing step)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$. Furthermore let L_p be the labelling obtained by* PREPROCESS *(Algorithm 1) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if $L_p[l] = \langle 1,0,0,0\rangle$, then there is no argument for l in AT. For each $r \in \mathcal{R}$: if $L_p[r] = \langle 1,0,0,0\rangle$, then there is no argument based on r in AT.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let L_p be the labelling after the preprocessing step. We prove this lemma by contraposition and induction on argument height.

Proposition ($P(n)$): For each $l \in \mathcal{L}$: if there is an argument A for l in $Arg(AT)$ with $\mathtt{h}(A) \leq n$ then $L^p[l] = \langle 0,1,1,1\rangle$; and for each $r \in \mathcal{R}$: if there is an argument A based on r in $Arg(AT)$ with $\mathtt{h}(A) \leq n+1$ then $L^p[r] = \langle 0,1,1,1\rangle$.

Base case ($P(0)$):

- For each $l \in \mathcal{L}$ such that there is an argument A for l in $Arg(AT)$ with $\mathtt{h}(A) = 0$, then $l \in \mathcal{K}$, so l is labelled $\langle 0,1,1,1\rangle$ in Algorithm 1 line 3. Since no operation in Algorithm 1 labels literals from $\langle 0,1,1,1\rangle$ to $\langle 1,0,0,0\rangle$, we have that $L_p[l] = \langle 0,1,1,1\rangle$.
- For each $l \in \mathcal{R}$ such that there is an argument A based on r in $Arg(AT)$ with $\mathtt{h}(A) = 1$, there must be some argument A' with $\mathtt{h}(A')$ for each of the antecedents of r, which are labelled $L_p[l] = \langle 0,1,1,1\rangle$ in Algorithm 1 line 3 (see above). This means that line 11 applies for r, so line 12 was executed: $L_p[r] = \langle 0,1,1,1\rangle$.

Induction hypothesis ($P(k)$): For each $l \in \mathcal{L}$: if there is an argument A for l in $Arg(AT)$ with $\mathtt{h}(A) \leq k$ then $L^p[l] = \langle 0,1,1,1\rangle$; and for each $r \in \mathcal{R}$: if there is an argument A based on r in $Arg(AT)$ with $\mathtt{h}(A) \leq k+1$ then $L^p[r] = \langle 0,1,1,1\rangle$.

Induction step:

- For each $l \in \mathcal{L}$ such that there is an argument A for l in $Arg(AT)$ with $\mathtt{h}(A) = k+1$, A must be based on some rule r which is labelled $L^p[r] = \langle 0,1,1,1\rangle$ by the induction hypothesis. This must have happened in line 12, after which l was labelled $L_p[l] = \langle 0,1,1,1\rangle$ in line 13.
- For each $l \in \mathcal{R}$ such that there is an argument A based on r in $Arg(AT)$ with $\mathtt{h}(A) = k+1$, there must be some argument A' with $\mathtt{h}(A')$ for each of the antecedents of r, which are labelled $L_p[l] = \langle 0,1,1,1\rangle$ in Algorithm 1 line 3 or 13 (see above). This means that line 11 applies for r, so line 12 was executed: $L_p[r] = \langle 0,1,1,1\rangle$.

Since each argument A in $Arg(AT)$ has a finite $\mathtt{h}(A)$, we can generalise this to each $l \in \mathcal{L}$ and each $r \in \mathcal{R}$. As there is no operation that labels literals or rules from $\langle 0,1,1,1\rangle$ to $\langle 1,0,0,0\rangle$, we have that for each $x \in \mathcal{L} \cup \mathcal{R}$: if $L_p[x] = \langle 0,1,1,1\rangle$ then $L_p[x] \neq \langle 1,0,0,0\rangle$. By contraposition: if $L_p[x] = \langle 1,0,0,0\rangle$, then there is no argument for/based on x in $Arg(AT)$. □

Lemma 3 (Completeness preprocessing step)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$. Furthermore let L_p be the labelling obtained by* PREPROCESS *(Algorithm 1) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if there is no argument for l in AT, then $L_p[l] = \langle 1, 0, 0, 0\rangle$. For each $r \in \mathcal{R}$: if there is no argument based on r in AT, then $L_p[r] = \langle 1, 0, 0, 0\rangle$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let L_p be the labelling after the preprocessing step. First, we introduce some notation. Let $x \in \mathcal{L} \cup \mathcal{R}$ 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 $l \in \mathcal{L}$ such that $L_n^p[l] = \langle 0, 1, 1, 1\rangle$, there is an argument for l in *Arg*(AT); for each $r \in \mathcal{R}$ such that $L_n^p[r] = \langle 0, 1, 1, 1\rangle$, there is an argument based on r in *Arg*(AT).

Base case ($P(0)$):

- For each $l \in \mathcal{L}$ such that $L_0^p[l] = \langle 0, 1, 1, 1\rangle$, the condition in Algorithm 1 line 2 must have applied ($l \in \mathcal{K}$), so by Definition 4, there is an observation-based argument for l in *Arg*(AT).
- No rule $r \in \mathcal{R}$ is labelled $L_n^p[r] = \langle 0, 1, 1, 1\rangle$, so for each $r \in \mathcal{R}$ such that $L_n^p[r] = \langle 0, 1, 1, 1\rangle$, there is an argument based on r in *Arg*(AT).

Induction hypothesis ($P(k)$): For each $l \in \mathcal{L}$ such that $L_k^p[l] = \langle 0, 1, 1, 1\rangle$, there is an argument for l in *Arg*(AT); for each $r \in \mathcal{R}$ such that $L_k^p[r] = \langle 0, 1, 1, 1\rangle$, there is an argument based on r in *Arg*(AT).

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

- For each $r \in \mathcal{R}$ such that $L_{k+1}^p[r] = \langle 0, 1, 1, 1\rangle$, it must be that for each $a \in \mathtt{ants}(r)$: $L_k^p[a] = \langle 0, 1, 1, 1\rangle$. Then by the induction hypothesis, for each $a \in \mathtt{ants}(r)$, there is an argument for a in *Arg*(AT), which by Definition 4 implies that there is an argument based on r in *Arg*(AT).
- For each $l \in \mathcal{L}$ such that $L_{k+1}^p[l] = \langle 0, 1, 1, 1\rangle$, it must be that there is some rule based on which there is an argument in *Arg*(AT). Consequently, there is an argument for l in *Arg*(AT).

Finally note that Algorithm 1 terminates, given the running time is polynomial in the input (Lemma 1) and the language and rule set are finite. Consequently, for each $l \in \mathcal{L}$: if $L_p[l] = \langle 0, 1, 1, 1\rangle$ then there is an argument for l in *Arg*(AT). Then by contraposition: if there is no argument for l in *Arg*(AT), then $L_p[l] \neq \langle 0, 1, 1, 1\rangle$, so $L_p[l] = \langle 1, 0, 0, 0\rangle$. Similarly, each $r \in \mathcal{R}$ based on which there is no argument in *Arg*(AT) is labelled $L_p[r] = \langle 1, 0, 0, 0\rangle$. □

3.3 Proofs w.r.t. JUSTIFICATION-LABEL algorithm

In this section, we prove the lemmas and propositions concerning JUSTIFICATION-LABEL. We discuss the time complexity (Section 3.3.1), soundness (Section 3.3.3) and completeness (Section 3.3.4) of our approximation algorithm. For the soundness proofs, we need the notions of interim labelling and last boolean flip iteration, which we will introduce in Section 3.3.2.

3.3.1 Time complexity

Proposition 1 (Time complexity JUSTIFICATION-LABEL)**.** *The time complexity of* JUSTIFICATION-LABEL *is* $\mathcal{O}(|\mathcal{L}|^3 \cdot |\mathcal{R}| + |\mathcal{L}|^2 \cdot |\mathcal{R}|^2)$.

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).

Line	Time single execution	Max nr. of executions	Total time
2	$(c_7 + c_{15}) + (c_2 + c_3 + c_4) \cdot \lvert\mathcal{L}\rvert + (c_5 + c_6 + c_8 + c_9) \cdot \lvert\mathcal{R}\rvert + (c_{10} + c_{12} + c_{13} + c_{14}) \cdot \lvert\mathcal{R}\rvert^2 + c_{11} \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert^2$	1	$(c_7 + c_{15}) + (c_2 + c_3 + c_4) \cdot \lvert\mathcal{L}\rvert + (c_5 + c_6 + c_8 + c_9) \cdot \lvert\mathcal{R}\rvert + (c_{10} + c_{12} + c_{13} + c_{14}) \cdot \lvert\mathcal{R}\rvert^2 + c_{11} \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert^2$
3	c_{16}	1	c_{16}
4	c_{17}	$\lvert\mathcal{L}\rvert$	$c_{17} \cdot \lvert\mathcal{L}\rvert$
5	$c_{18} \cdot (\lvert\mathcal{L}\rvert + \lvert\mathcal{R}\rvert)$	$\lvert\mathcal{L}\rvert$	$c_{18} \cdot (\lvert\mathcal{L}\rvert^2 + \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert)$
6	$c_{19} \cdot \lvert\mathcal{R}\rvert$	$\lvert\mathcal{L}\rvert$	$c_{19} \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$
7	c_{20}	$3 \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$	$3 \cdot c_{20} \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$
8	c_{21}	$3 \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$	$3 \cdot c_{21} \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$
9	$c_{22} \cdot \lvert\mathcal{L}\rvert$	$3 \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$	$3 \cdot c_{22} \cdot \lvert\mathcal{L}\rvert^2 \cdot \lvert\mathcal{R}\rvert$
10	c_{23}	$3 \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$	$3 \cdot c_{23} \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$
11	$c_{24} \cdot (\lvert\mathcal{L}\rvert + \lvert\mathcal{R}\rvert)$	$2 \cdot \lvert\mathcal{R}\rvert$	$2 \cdot c_{24} \cdot (\lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert + \lvert\mathcal{R}\rvert^2)$
12	c_{25}	$2 \cdot \lvert\mathcal{R}\rvert$	$2 \cdot c_{25} \cdot \lvert\mathcal{R}\rvert$
13	$c_{26} \cdot \lvert\mathcal{R}\rvert$	$2 \cdot \lvert\mathcal{L}\rvert$	$2 \cdot c_{26} \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$
14	c_{27}	$3 \cdot \lvert\mathcal{L}\rvert^2 \cdot \lvert\mathcal{R}\rvert$	$3 \cdot c_{27} \cdot \lvert\mathcal{L}\rvert^2 \cdot \lvert\mathcal{R}\rvert$
15	$c_{28} \cdot (\lvert\mathcal{L}\rvert + \lvert\mathcal{R}\rvert)$	$3 \cdot \lvert\mathcal{L}\rvert^2 \cdot \lvert\mathcal{R}\rvert$	$3 \cdot c_{28} \cdot (\lvert\mathcal{L}\rvert^3 \cdot \lvert\mathcal{R}\rvert + \lvert\mathcal{L}\rvert^2 \cdot \lvert\mathcal{R}\rvert^2)$
16	c_{29}	$3 \cdot \lvert\mathcal{L}\rvert^2 \cdot \lvert\mathcal{R}\rvert$	$3 \cdot c_{29} \cdot \lvert\mathcal{L}\rvert^2 \cdot \lvert\mathcal{R}\rvert$
17	$c_{30} \cdot \lvert\mathcal{R}\rvert$	$2 \cdot \lvert\mathcal{L}\rvert$	$2 \cdot c_{30} \cdot \lvert\mathcal{L}\rvert \cdot \lvert\mathcal{R}\rvert$
18	c_{31}	1	c_{31}
Total time required for all lines			$\mathcal{O}(\lvert\mathcal{L}\rvert^3 \cdot \lvert\mathcal{R}\rvert + \lvert\mathcal{L}\rvert^2 \cdot \lvert\mathcal{R}\rvert^2)$

In the following, we will denote positive constants by c_i (with $i \in [1 \,..\, 31]$). Also, we assume that for each literal $l \in \mathcal{L}$, 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 $l \in \mathcal{L}$, we can check in constant time if $l \in \mathcal{K}$.

First we show **which amount of time is required for a single execution of a given line**.

- Line 2 requires running PREPROCESS$(\mathcal{L}, \mathcal{R}, \overline{}, \mathcal{K})$, which takes $(c_7 + c_{15}) + (c_2 + c_3 + c_4) \cdot |\mathcal{L}| + (c_5 + c_6 + c_8 + c_9) \cdot |\mathcal{R}| + (c_{10} + c_{12} + c_{13} + c_{14}) \cdot |\mathcal{R}|^2 + c_{11} \cdot |\mathcal{L}| \cdot |\mathcal{R}|^2$ steps, as shown in Lemma 1.
- Line 3 takes constant time c_{16}.
- Line 4 takes a new literal l from $\mathcal{L}$, which takes constant time c_{17}.
- A single execution of line 5 requires labelling a literal, which can be done in $c_{18} \cdot (|\mathcal{L}| + |\mathcal{R}|)$ time: in the worst case, it requires checking the presence of l and all its contradictories ($|\bar{l}| \leq |\mathcal{L}|$) in $\mathcal{K}$, as well as the labels of all (max $|\mathcal{R}|$) rules for that literal or a contradictory. Therefore, line 5 requires $c_{18} \cdot (|\mathcal{L}| + |\mathcal{R}|)$ 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 $\mathcal{R}$ rules having l as an antecedent, so this takes at most $c_{19} \cdot |\mathcal{R}|$ time.
- Line 7 only needs to check if a set is empty, which can be done in constant time c_{20}. 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_{21} time.
- Line 9 relabels a rule, which requires checking the labels of all antecedents of this rule. Since a rule has at most $|\mathcal{L}|$ antecedents, this takes at most $c_{22} \cdot |\mathcal{L}|$ 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_{23}, c_{25} and c_{29} 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 contraries in $\mathcal{K}$, as well as checking the labels of all rules for that literal or one of its contraries. A single execution therefore takes $c_{24} \cdot (|\mathcal{L}| + |\mathcal{R}|)$ time for line 11 and $c_{28} \cdot (|\mathcal{L}| + |\mathcal{R}|)$ time for line 15.
- Lines 13 and 17 both add at most $|\mathcal{R}|$ rules to TODO-SET, so a single execution of line 13 needs at most $c_{26} \cdot |\mathcal{R}|$ time and a single execution of line 17 needs at most $c_{30} \cdot |\mathcal{R}|$ time.

- A single execution of line 14 takes a literal from a list of contraries, which can be done in constant time c_{27}.
- Finally, a single execution of line 18 takes constant time c_{31}.

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 $\mathcal{L}$. This implies that lines 4–6 are executed at most $|\mathcal{L}|$ 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 two times (i.e. at most four booleans can be turned to False, see Lemma 6), each rule $r \in \mathcal{R}$ is recolored at most $3 \cdot |\mathtt{ants}(r)|$ times. There are $|\mathcal{R}|$ rules, so lines 7–10 are executed at most $3 \cdot |\mathcal{L}| \cdot |\mathcal{R}|$ 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 two times. There are $|\mathcal{R}|$ rules in total, so lines 11 and 12 are executed at most $2 \cdot |\mathcal{R}|$ times.
- Lines 13 and 17 are only executed just after the label of a literal changed. This can happen at most two times for each literal, because at most four booleans can be turned to False. There are $|\mathcal{L}|$ literals in total; therefore lines 13 and 17 are executed at most $2 \cdot |\mathcal{L}|$ times.
- Finally, lines 14–16 are executed at most $|\mathcal{L}|$ times (i.e. once for each $l' \in \overline{\mathtt{conc}(r)}$) for each of the maximal $3 \cdot |\mathcal{L}| \cdot |\mathcal{R}|$ iterations of the while loop. This means that lines 14–16 are executed at most $3 \cdot |\mathcal{L}|^2 \cdot |\mathcal{R}|$ 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 JUSTIFICATION-LABEL (Algorithm 4) is $\mathcal{O}(|\mathcal{L}|^3 \cdot |\mathcal{R}| + |\mathcal{L}|^2 \cdot |\mathcal{R}|^2)$. □

3.3.2 Interim labelling

In order to prove the soundness of the JUSTIFICATION-LABEL algorithm (Section 3.3.3), we will repeatedly use the notion of interim label, that is: the label at some point before the labelling is finished.

Definition 9 (Interim labelling)**.** Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b \rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For any literal or rule $x \in \mathcal{L} \cup \mathcal{R}$, 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 are relabelled for the first time.

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 $\neg L_i[l].d$, $\neg L_{i+1}[l].d$, 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 $\neg L[r].d$ at that point, because we know that l is an antecedent of r and l is labelled $\neg L_j[l].d$.

Definition 10 (Last boolean flip iteration)**.** Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $\mathcal{J} = 2^{\{u,d,o,b\}}$ be the set of all subsets of the labelling booleans u, d, o and b. For each $J \in \mathcal{J}$ and for each literal or rule $x \in \mathcal{L} \cup \mathcal{R}$, we denote by $c_J(x)$ the number of the iteration of the while loop in Algorithm 4 (executed on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$) 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 $L = \langle u, d, o, b \rangle$. In case there is some $j \in J$ such that j is True in the final labelling L, $c_J(x) = \infty$.

3.3.3 Soundness of the algorithm

In this section, we will prove that JUSTIFICATION-LABEL is sound. As we have shown in the proof sketch of Proposition 2, this requires proving Lemmas 7–13. In addition, we need to apply Lemma 6, which states that no literal or rule can have the label $\langle 0,0,0,0\rangle$ and is proven next. Since this lemma is required very frequently in the proofs that follow, we sometimes use it implicitly.

Lemma 6 (No 4x False label)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $x \in \mathcal{L} \cup \mathcal{R}$: $L[x] \neq \langle 0,0,0,0\rangle$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$. Let L_p be the labelling obtained by PREPROCESS (Algorithm 1) and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. We proceed by induction on the interim labelling of literals and rules.

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

Base case ($P(1)$): For each $x \in \mathcal{L} \cup \mathcal{R}$: if $L_p[x] = \langle 1,0,0,0\rangle$, then $L_0(x) \neq \langle 0,0,0,0\rangle$ because there is no operation that labels the u-boolean to False.

- For each $r \in \mathcal{R}$ such that $L_p[r] = \langle 0,1,1,1\rangle$, $L_0(r) = L_p[r] = \langle 0,1,1,1\rangle$ because rules are not relabelled between preprocessing and the first iteration of the while loop, hence $\boldsymbol{L_0(r) \neq \langle 0,0,0,0\rangle}$.
- For each $l \in \mathcal{L}$ such that $L_p[l] = \langle 0,1,1,1\rangle$, it must be that $l \in \mathcal{K}$ or there is a rule r for l that is labelled $L_p[r] = L_0(r) = \langle 0,1,1,1\rangle$. If $l \in \mathcal{K}$ then none of the defended-cases of Algorithm 2 applies. Alternatively, $l \notin \mathcal{K}$, so there is a rule r for l that is labelled $L_p[r] = L_0(r) = \langle 0,1,1,1\rangle$. If some $l' \in \bar{l}$ is in $\mathcal{K}$, then none of the out-cases of Algorithm 2 applies; if there is no $l' \in \bar{l}$ in $\mathcal{K}$, then none of the blocked-cases applies. This means that in all cases, $\boldsymbol{L_0(l) \neq \langle 0,0,0,0\rangle}$.

Induction hypothesis ($P(k)$): For each $x \in \mathcal{L} \cup \mathcal{R}$ 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 $x \in \mathcal{L} \cup \mathcal{R}$: $L_k(x) \neq \langle 0,0,0,0\rangle$. Again, we separately consider rules and literals.

- Let $r \in \mathcal{R}$ be an arbitrary rule. Suppose, towards a contradiction, that $L_k(r) = \langle 0,0,0,0\rangle$. Then $L_p[r] = \langle 0,1,1,1\rangle$ (because of $\neg L[r].u$), so for each $a \in \texttt{ants}(r)$: $L_p[a] = \langle 0,1,1,1\rangle$. The fact that $\neg L[r].d$ 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 $\neg L[a].b$. In addition, a is labelled $\neg L[a].u$ because $L_p[a] = \langle 0,1,1,1\rangle$ This contradicts the induction hypothesis, so $L_k(r) \neq \langle 0,0,0,0\rangle$. Since we chose r arbitrarily from $\mathcal{R}$, **there is no rule $\boldsymbol{r \in \mathcal{R}}$ such that $\boldsymbol{L_k(r) = \langle 0,0,0,0\rangle}$**.
- Now let $l \in \mathcal{L}$ be an arbitrary literal. Suppose, towards a contradiction, that $L_k(l) = \langle 0,0,0,0\rangle$.

  First note that $l \notin \mathcal{K}$: if $l \in \mathcal{K}$, then none of the defended-cases from Algorithm 2 would apply. Given that $L_k(l) = \langle 0,0,0,0\rangle$, it must be that $L_p[l] = \langle 0,1,1,1\rangle$, so there is a rule r for l such that $L_p[r] = \langle 0,1,1,1\rangle$, which means that $\neg L[r].u$. The label $\neg L[l].o$ must be caused by L-O-b or L-O-c, but in both cases for each $l' \in \bar{l}$: $l' \notin \mathcal{K}$ and 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-d or L-B-e. 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 we need to retract our assumption and conclude that $L_k(l) \neq \langle 0,0,0,0\rangle$.

  Recall that we picked l arbitrarily from $\mathcal{L}$; therefore we have that **for each $\boldsymbol{l \in \mathcal{L}}$: $\boldsymbol{L_k(l) \neq \langle 0,0,0,0\rangle}$**.

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

Lemma 7 (Labelled not unsatisfiable or out)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by* Justification-label *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $r \in \mathcal{R}$: if r is labelled $\neg L[r].u$ and $\neg L[r].o$, then 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 $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by Justification-label (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. We proceed by induction on $c_{\{u,o\}}$.

Proposition ($P(n)$): For each rule $r \in \mathcal{R}$ such that $c_{\{u,o\}}(r) \leq n$: if r is labelled $\neg L[r].u$ and $\neg L[r].o$, then 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 $\mathcal{R}$ such that $c_{\{u,o\}}(r) \leq 1$ and $\neg L[r].u$ and $\neg L[r].o$. Given $\neg L[r].u$ it must be that $L_p[r] = \langle 0,1,1,1\rangle$, so for each $a \in \texttt{ants}(r)$: $L_p[a] = \langle 0,1,1,1\rangle$. $\neg L[r].o$ must have been caused by case R-O-a: each $a \in \texttt{ants}(r)$ is labelled $\neg L[a].o$ and $c_{\{u,o\}}(a) = 0$. Then each a in $\texttt{ants}(r)$ must be in $\mathcal{K}$: if there would be some $a \in \texttt{ants}(r)$ that is not in $\mathcal{K}$, then there is some rule r' for a labelled $L_p[r'] = L_0(r') = \langle 0,1,1,1\rangle$ and there is no rule r'' for a that is labelled $\neg L[r''].u$ and $\neg L[r''].o$ by L_p or L_0, so none of the out-cases of Algorithm 2 would apply. Consequently, there is an observation-based argument for each of r's antecedents, hence there is an argument based on r in $Arg(AT)$ that, by Definition 5 and Lemma 4, is not attacked on a subargument by any argument in $G(AT)$.

Induction hypothesis ($P(k)$): For each $r \in \mathcal{R}$ such that $c_{\{u,o\}}(r) \leq k$: if r is labelled $\neg L[r].u$ and $\neg L[r].o$, then 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 $\mathcal{R}$ such that $c_{\{u,o\}}(r) \leq k+1$ and $\neg L[r].u$ and $\neg L[r].o$. $\neg L[r].u$ implies that $L_p[r] = \langle 0,1,1,1\rangle$, so by Lemma 3 there is an argument based on r in $Arg(AT)$. By Algorithm 3 case R-O-a, for each $\boldsymbol{a \in \texttt{ants}(r)}$: $\boldsymbol{\neg L[a].o}$ and $\boldsymbol{c_{\{u,o\}}(a) \leq k}$. Now let a be an arbitrary antecedent in $\texttt{ants}(r)$. We consider two cases:

- If $a \in \mathcal{K}$, then there is an observation-based argument for a in $Arg(AT)$ that, by Lemma 5, is not attacked by any argument in $G(AT)$.
- Alternatively, $a \notin \mathcal{K}$. Then there must be some rule r' for a that was labelled $L_p[r] = \langle 0,1,1,1\rangle$ by Algorithm 1. This implies that the label $\neg L[a].o$ can only be caused by case L-O-b or L-O-c. 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) no $a' \in \overline{a}$ is in $\mathcal{K}$. Given that $c_{\{u,o\}}(a) \leq k$, it follows that $c_{\{u,o\}}(r') \leq k$. So by the induction hypothesis and (1): 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 observation-based argument for any $a' \in \overline{a}$ in $Arg(AT)$, so by Lemma 5 no argument for a in $Arg(AT)$ can be attacked *on its conclusion* by an argument in $G(AT)$.

Since we chose a arbitrarily, for each $a \in \texttt{ants}(r)$ there is an argument for a in $Arg(AT)$ that is not attacked by any argument in $G(AT)$. Then we can construct from these arguments for a 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 $r \in \mathcal{R}$ that is labelled $\neg L[r].u$ and $\neg L[r].o$ (Definition 10), which means that the proposition is valid for each $r \in \mathcal{R}$ in general. □

Lemma 8 (Labelled not defended or blocked)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by* Justification-label *(Algorithm 4) and L_p be the labelling obtained by* Preprocess *(Algorithm 1) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. If $r \in \mathcal{R}$ is labelled $\neg L[r].d$ and $\neg L[r].b$ then each argument based on r in $Arg(AT)$ is attacked by an argument in $G(AT)$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by Justification-label (Algorithm 4) and L_p be the labelling obtained by Preprocess (Algorithm 1) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. We proceed by induction on $c_{\{d,b\}}$.

Proposition ($P(n)$): For each $r \in \mathcal{R}$ such that $c_{\{d,b\}}(r) \leq n$: if r is labelled $\neg L[r].d$ and $\neg L[r].b$, then 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 suppose that r is labelled $\neg L[r].d$ and $\neg L[r].b$.

If $L_p[r] = \langle 1,0,0,0\rangle$ then by Lemma 2 there is no argument based on r in $Arg(AT)$ so each argument based on r in $Arg(AT)$ is attacked by an argument in $G(AT)$. Alternatively, $L_p[r] = \langle 0,1,1,1\rangle$, so r must be labelled $\neg L[r].d$ and $\neg L[r].b$ by Algorithm 4 and $\boldsymbol{c_{\{d,b\}}(r) = 1}$.

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 $\boldsymbol{\neg L[a].d}$ and $\boldsymbol{\neg L[a].b}$ and $\boldsymbol{c_{\{d,b\}}(a) = 0}$.

Given that $c_{\{d,b\}}(a) = 0$, a must have been relabelled in Algorithm 4 line 5: 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 7 would not have applied for r), which would contradict our assumption that $L_p[r] = \langle 0,1,1,1\rangle$.

Given that a is labelled $\neg L[a].d$ by one of the defended-cases of Algorithm 2, $a \notin \mathcal{K}$. Then there is a rule r for a labelled $L_p[r'] = L_0(a') = \langle 0,1,1,1\rangle$ and there is no rule $r'' \in \mathcal{R}$ that is labelled $\neg L[r''].u$ and $\neg L[r''].o$ by L_p or L_0. This means that case L-D-a must have applied: there is some $a' \in \overline{a}$ in $\mathcal{K}$. So by Lemma 5, each argument for a in $Arg(AT)$ is attacked by an argument in $G(AT)$. Given that a is an antecedent of r, **each argument based on $\boldsymbol{r}$ in $\boldsymbol{Arg(AT)}$ is attacked by an argument in $\boldsymbol{G(AT)}$.**

Induction hypothesis ($P(k)$): For each $r \in \mathcal{R}$ such that $c_{\{d,b\}}(r) \leq k$: if r is labelled $\neg L[r].d$ and $\neg L[r].b$, then 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 $\mathcal{R}$ such that $c_{\{d,b\}}(r) = k+1$ and r is labelled $\neg L[r].d$ and $\neg L[r].b$; then $L_p[r] = \langle 0,1,1,1\rangle$.

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:

- If there is some $a' \in \overline{a}$ such that $a' \in \mathcal{K}$, then there is an observation-based argument for a' in $Arg(AT)$, so by Lemma 4, each argument for a in $Arg(AT)$ is attacked by an argument in $G(AT)$.
- Alternatively, each rule r' for a is labelled $\neg L[r'].d$ and $\neg L[r'].b$ and $c_{\{d,b\}}(r') \leq k$: if L-D-c caused $\neg L[a].d$, then by Lemma 6, a must have been labelled $\neg L[a].b$ by case L-B-c; if, alternatively, L-D-b caused $\neg L[a].d$, then either L-B-c or L-B-d caused $\neg L[a].b$.

  Consider an arbitrary rule r' for a. Either $L_p[r'] = \langle 1,0,0,0\rangle$ or $L_p[r'] = \langle 0,1,1,1\rangle$. If $L_p[r'] = \langle 1,0,0,0\rangle$, then by Lemma 2 there is no argument based on r' in any $Arg(AT)$. If, alternatively, $L_p[r'] = \langle 0,1,1,1\rangle$ (which must be the case for at least one rule for a, since $L_p[a] = \langle 0,1,1,1\rangle$), we apply the induction hypothesis: each argument based on r' in $Arg(AT)$ is attacked by an argument in $G(AT)$. Since $a \notin \mathcal{K}$ (by case L-D-b or L-D-c), there are no observation-based arguments for a either, so: **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 $r \in \mathcal{R}$ such that $c_{\{d,b\}} \leq k+1$: if r is labelled $\neg L[r].d$ and $\neg L[r].b$, then each argument based on r in $Arg(AT)$ is attacked by an argument in $G(AT)$. Finally, remember that for each rule $r \in \mathcal{R}$ that is labelled $\neg L[r].d$ and $\neg L[r].b$, $c_{\{d,b\}}(r)$ is finite (Definition 10). Therefore **each argument based on $\boldsymbol{r}$ in $\boldsymbol{Arg(AT)}$ is attacked by an argument in $\boldsymbol{G(AT)}$ for each $\boldsymbol{r \in \mathcal{R}}$ that is labelled $\boldsymbol{\neg L[r].d}$ and $\boldsymbol{\neg L[r].b}$.** □

Lemma 9 (Soundness unsatisfiable labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if $L[l] = \langle 1,0,0,0\rangle$ then l is unsatisfiable in AT.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $l \in \mathcal{L}$ such that $L[l] = \langle 1,0,0,0\rangle$, the fact that $L[l].u$ implies that l was already labelled unsatisfiable in the preprocessing step: $L_p[l] = \langle 1,0,0,0\rangle$, so l is unsatisfiable in AT (Lemma 2). □

Lemma 10 (Soundness defended labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each literal $l \in \mathcal{L}$: if $L[l] = \langle 0,1,0,0\rangle$ then l is defended in AT.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. We proceed by induction on $c_{\{u,o,b\}}(l)$.

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

Base case ($P(0)$): Let l be an arbitrary literal in $\mathcal{L}$ such that $c_{\{u,o,b\}}(l) \leq 0$ and suppose that $L[l] = \langle 0,1,0,0\rangle$. Note that this implies that $L_p[l] = \langle 0,1,1,1\rangle$. It follows that l must have been relabelled to $\langle 0,1,0,0\rangle$ by Algorithm 4 line 5. This implies that $l \in \mathcal{K}$: if $l \notin \mathcal{K}$, then there is some rule r for l that is labelled $L_p[r] = L_0(r) = \langle 0,1,1,1\rangle$

(since $L_p[l] = \langle 0,1,1,1\rangle$) and there is no rule r'' for l that is labelled $\neg L[r''].u$ and $\neg L[r''].o$ by L_p or L_0, so none of the out-cases of Algorithm 2 would apply. Given that $l \in \mathcal{K}$, there is an observation-based argument for l in each *Arg*(AT) that cannot be attacked and therefore is in the grounded extension. Consequently, l is defended in AT.

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

Induction step $(P(k+1))$: Now consider an arbitrary literal $l \in \mathcal{L}$ such that $c_{\{u,o,b\}}(l) = k+1$ and suppose that $L[l] = \langle 0,1,0,0\rangle$. If $l \in \mathcal{K}$, then l would be labelled defended before the start of the while loop, which contradicts our assumption that $c_{\{u,o,b\}}(l) = k+1$ for positive k. Consequently, $\boldsymbol{l \notin \mathcal{K}}$.

Given that $l \notin \mathcal{K}$ and $L_p[l] = \langle 0,1,1,1\rangle$, there is some rule r for l that was labelled $L_p[r] = \langle 0,1,1,1\rangle$ in the preprocessing step, so $\neg L[r].u$. Then $\neg L[r].o$ must have been caused by Algorithm 3 case L-O-b or L-O-c, but in both cases (1) for each $l' \in \bar{l}$: $l' \notin \mathcal{K}$ and (2) there is a rule r for l labelled $\neg L[r].u$ and $\neg L[r].o$. Then by Lemma 6, $\neg L[r].b$ must have been caused by case L-B-d or L-B-e, but in both cases (3) there is a rule r for l labelled $L[r] = \langle 0,1,0,0\rangle$ and (4) each rule r' for some $l' \in \bar{l}$ is labelled $\neg L[r'].d$ and $\neg L[r'].b$. Then by (3) $L_p[r]$ was $\langle 0,1,1,1\rangle$, which means that for each $a \in \mathtt{ants}(r)$: $L_p[a] = \langle 0,1,1,1\rangle$. Furthermore, $\neg L[r].o$ and $\neg L[r].b$ must have been caused by Algorithm 3 case R-O-a and R-B-a, so for each $a \in \mathtt{ants}(r)$: $L[a] = \langle 0,1,0,0\rangle$. By the induction hypothesis, there is an argument in $G(AT)$ for each $a \in \mathtt{ants}(r)$, so by Lemma 4, there is an argument A based on r in *Arg*(AT) such that each argument B attacking A *on a subargument* in AT is attacked by an observation-based argument in $G(AT)$.

Finally recall (4): each rule r' for some $l' \in \bar{l}$ is labelled $\neg L[r'].d$ and $\neg L[r'].b$. By Lemma 8, this implies that for each rule r' for some $l' \in \bar{l}$, each argument based on r' in *Arg*(AT) is attacked by an argument in $G(AT)$, hence each argument attacking an argument for l *on its conclusion* in AT is attacked by an argument in $G(AT)$.

To summarize, there is some argument A (based on r, for l) in *Arg*(AT) such that each argument in *Arg*(AT) attacking A (either on a subargument or on its conclusion l) is attacked by an argument in $G(AT)$. Then by Definitions 7 and 8, $\boldsymbol{l}$ **is defended in** $\boldsymbol{AT}$. 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 $l \in \mathcal{L}$: if $c_{\{u,o,b\}}(l) \leq n$ and $L[l] = \langle 0,1,0,0\rangle$ then l is defended in AT. Given that for each $l \in \mathcal{L}$ such that $L[l] = \langle 0,1,0,0\rangle$, $c_{\{u,o,b\}}(l)$ is finite (Definition 10), we derive: **for each** $\boldsymbol{l \in \mathcal{L}}$**: if** $\boldsymbol{L[l] = \langle 0,1,0,0\rangle}$ **then** $\boldsymbol{l}$ **is defended in** $\boldsymbol{AT}$. □

Lemma 11 (Soundness out labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$ and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\bar{\ }$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if $L[l] = \langle 0,0,1,0\rangle$ then l is out in AT.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$ and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\bar{\ }$ and $\mathcal{K}$. Suppose that $l \in \mathcal{L}$ is labelled $L[l] = \langle 0,0,1,0\rangle$ by JUSTIFICATION-LABEL (Algorithm 4). Then $L_p[l] = \langle 0,1,1,1\rangle$, so by Lemma 3 there is an argument for l in *Arg*(AT). Given that $\neg L[l].d$, $l \notin \mathcal{K}$, so there is a rule r for l that is labelled $L_p[r] = \langle 0,1,1,1\rangle$.

- If there is some $l' \in \bar{l}$ such that $l' \in \mathcal{K}$; then each argument for l in *Arg*(AT) is attacked by the observation-based argument for l'. Hence, by Lemma 5, $\boldsymbol{l}$ **is out in** $\boldsymbol{AT}$.
- Alternatively, for each rule r for l: $\neg L[r].d$ and $\neg L[r].b$: if $\neg L[l].d$ is caused by case L-D-b then this follows from case L-B-c/L-B-d and Lemma 6; alternatively, it follows from case L-D-c and case L-B-c together with Lemma 6. Now, by Lemma 8: each argument for l in *Arg*(AT) is attacked by some argument in $G(AT)$. Therefore, by Definition 8, $\boldsymbol{l}$ **is out in** $\boldsymbol{AT}$. □

Lemma 12 (Labelled not defended)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$ and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\bar{\ }$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if l is labelled $\neg L[l].d$ then there is no argument for l in $G(AT)$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$ and let $L = \langle u,d,o,b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\bar{\ }$ and $\mathcal{K}$.

For literals $l \in \mathcal{L}$ such that $L_p[l] = \langle 1,0,0,0\rangle$, there is no argument for l in *Arg*(AT) by Lemma 2. For literals $l \in \mathcal{L}$ such that $L_p[l] = \langle 0,1,1,1\rangle$, there is an argument for l in *Arg*(AT). Given that $\neg L[l].d$, $l \notin \mathcal{K}$, so there must be a rule-based argument for l in *Arg*(AT): there is some rule r for l such that $L_p[r] = \langle 0,1,1,1\rangle$ (Lemma 2). We proceed by induction on $c_{\{d\}}(l)$.

Proposition $(P(n))$: For each $l \in \mathcal{L}$ such that $c_{\{d\}}(l) \leq n$: if l is labelled $\neg L[l].d$ and $L_p[l] = \langle 0,1,1,1\rangle$, then there is no argument for l in $G(AT)$.

Base case ($P(0)$): Let l be an arbitrary literal from $\mathcal{L}$ such that $c_{\{d\}}(l) \leq 0$. At that point, there is a rule r for l that is labelled $L_p[r] = L_0(r) = \langle 0, 1, 1, 1 \rangle$ and there is no rule $r'' \in \mathcal{R}$ that is labelled $\neg L[r''].u$ and $\neg L[r''].o$ by L_p or L_0. This implies that $\neg L[l].d$ must have been caused by Algorithm 2 L-D-a: there is some $l' \in \bar{l}$ such that $l' \in \mathcal{K}$. This means that there is an observation-based argument for l' in $Arg(AT)$ which, by Definitions 5 and 7, is in $G(AT)$. Consequently, there is no argument for l in $G(AT)$.

Induction hypothesis ($P(k)$): For each $l \in \mathcal{L}$ such that $c_{\{d\}}(l) \leq k$: if l is labelled $\neg L[l].d$ and $L_p[l] = \langle 0, 1, 1, 1 \rangle$, then there is no argument for l in $G(AT)$.

Induction step: Let l be an arbitrary literal in $\mathcal{L}$ such that $c_{\{d\}}(l) = k + 1$ and $L_p[l] = \langle 0, 1, 1, 1 \rangle$. Then there is no $l' \in \bar{l}$ in $\mathcal{K}$: that would imply that $c_{\{d\}}(l) = 0$, which contradicts our assumption that $c_{\{d\}}(l) = k + 1$ for positive k.
Then $\neg L[l].d$ 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). Then, by R-D-a, each rule r for l has some $a \in \mathtt{ants}(r)$ with $\neg L[a].d$ and $c_{\{d\}}(a) \leq k$. By the induction hypothesis, there is no argument for a in the grounded extension $G(AT)$. As a result, 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 $G(AT)$. Consequently, there is no argument based on any rule for l in $G(AT)$. Furthermore, the fact that $l \notin \mathcal{K}$ implies that there is no observation-based argument either, so **there is no argument for $\boldsymbol{l}$ in $\boldsymbol{G(AT)}$**.

- Now suppose that there is a rule r for l that is labelled $L[r].d$ (L-D-c is applied). Then 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 7, there exists an argument for l' based on r' that is not attacked by any argument in $G(AT)$. So **no argument for $\boldsymbol{l}$ is in $\boldsymbol{G(AT)}$**.

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

Lemma 13 (Soundness blocked labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$ and let $L = \langle u, d, o, b \rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\bar{\ }$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if $L[l] = \langle 0, 0, 0, 1 \rangle$ then l is blocked in AT.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$ and let $L = \langle u, d, o, b \rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\bar{\ }$ and $\mathcal{K}$. Let $l \in \mathcal{L}$ be an arbitrary literal that is labelled $L[l] = \langle 0, 0, 0, 1 \rangle$ by JUSTIFICATION-LABEL (Algorithm 4). Note that this implies that $L_p[l] = \langle 0, 1, 1, 1 \rangle$, 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 4, based on Algorithm 2.
The fact that l is labelled $L[l].b$, although l is considered for labelling by Algorithm 2, implies that $l \notin \mathcal{K}$ (case L-B-a did not apply). Then fact that $L_p[l] = \langle 0, 1, 1, 1 \rangle$ must have been caused by some rule r for l such that $L_p[r] = \langle 0, 1, 1, 1 \rangle$, so $\neg L[r].u$ The fact that l is labelled $\neg L[l].o$ must be caused by either L-O-b or L-O-c. In both cases, there is a rule r for l that is labelled $\neg L[r].u$ and $\neg L[r].o$. By Lemma 7, 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 there is no argument for l in $G(AT)$ (Lemma 12). To conclude, l is blocked in AT (Definition 8). □

3.3.4 Completeness of the algorithm

In this section, we will prove that JUSTIFICATION-LABEL is sound. As we have shown in the proof sketch of Proposition 3, this requires proving Lemmas 14–20.

Lemma 14 (Completeness unsatisfiable labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$ and let $L = \langle u, d, o, b \rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\bar{\ }$ and $\mathcal{K}$. Each literal $l \in \mathcal{L}$ that is unsatisfiable in AT is labelled $L[l] = \langle 1, 0, 0, 0 \rangle$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \bar{\ })$ and let $L = \langle u, d, o, b \rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\bar{\ }$ and $\mathcal{K}$. Each literal $l \in \mathcal{L}$ that is unsatisfiable in AT is labelled $L_p[l] = \langle 1, 0, 0, 0 \rangle$ by Lemma 3, which by Lemma 6 implies that $L[l] = \langle 1, 0, 0, 0 \rangle$. □

Lemma 15 (Labelled not unsatisfiable or out)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $r \in \mathcal{R}$: if there is an argument based on r in $Arg(AT)$ that is not attacked on a subargument by any argument in $G(AT)$ then r is labelled $\neg L[r].u$ and $\neg L[r].o$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. We proceed by induction on the height of arguments based on a rule.

Proposition ($P(n)$): If there is an argument A based on r in $Arg(AT)$ with $\mathtt{h}(A) \leq n$ that is not attacked on a subargument by any argument in $G(AT)$ then r is labelled $\neg L[r].u$ and $\neg L[r].o$.

Base case ($P(0)$): Suppose that there is an argument A based on r in $Arg(AT)$ such that $\mathtt{h}(A) \leq 1$. Then by Lemma 2, $L_p[r] = \langle 0, 1, 1, 1\rangle$, so $\neg L[r].u$. Furthermore, for each $a \in \mathtt{ants}(r)$: $a \in \mathcal{K}$, so a is relabelled in Algorithm 4 line 5 as $\neg L[l].o$ by Algorithm 2 case L-O-a. Subsequently, r is added to TODO-SET and, when popped, relabelled in line 9 as $\neg L[r].o$ by case R-O-a.

Induction hypothesis ($P(k)$): If there is an argument A based on r in $Arg(AT)$ with $\mathtt{h}(A) \leq k$ that is not attacked on a subargument by any argument in $G(AT)$ then r is labelled $\neg L[r].u$ and $\neg L[r].o$.

Induction step ($P(k+1)$): Now suppose that there is an argument A based on r in $Arg(AT)$ with $\mathtt{h}(A) = k+1$ that is not attacked on a subargument by any argument in $G(AT)$. Then $\neg L[r].u$ by Lemma 2.

Furthermore, 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 attacked by any observation-based argument in $Arg(AT)$ and $\mathtt{h}(A_i) \leq k$.

- If there is an observation-based argument for a in $Arg(AT)$, then $a \in \mathcal{K}$, which means that a relabelled in Algorithm 4 as $\neg L[l].o$ by case L-O-a;
- Alternatively, there is a rule-based argument A' for a based on some rule r' for a in $Arg(AT)$ with $\mathtt{h}(A') \leq k$ that is not attacked (1) on a subargument or (2) on its conclusion. By the induction hypothesis and (1), some rule r' for a is labelled $\neg L[r'].u$ and $\neg L[r'].o$. By (2), no $a' \in \overline{a}$ can be in $\mathcal{K}$. The label $\neg L[r'].o$ must have been assigned in Algorithm 4 line 9, after which a is considered for relabelling in line 11 and labelled $\neg L[a].o$ by case L-O-c.

After relabelling the last antecedent $a \in \mathtt{ants}(r)$ as $\neg L[a].u$ and $\neg L[a].o$, r is added to TODO-SET (line 6/13) and considered for relabelling in line 9 of a later iteration of the while loop, where it is labelled $\neg L[r].o$ by case R-O-a.

We have proved $P(n)$ for each $n \in \mathbb{N}$. Since each argument has a finite height, this concludes the proof. □

Lemma 16 (Labelled not defended or blocked)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) and L_p be the labelling obtained by* PREPROCESS *(Algorithm 1) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. If each argument based on $r \in \mathcal{R}$ in $Arg(AT)$ is attacked by an argument in $G(AT)$ then r is labelled $\neg L[r].d$ and $\neg L[r].b$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) and L_p be the labelling obtained by PREPROCESS (Algorithm 1) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. We proceed by induction on the height of arguments.

Proposition ($P(n)$): For each $r \in \mathcal{R}$: if each argument A based on r in $Arg(AT)$ has $\mathtt{h}(A) \leq n$ and is attacked on a subargument by some argument in $G(AT)$ then r is labelled $\neg L[r].d$ and $\neg L[r].b$.

Base case ($P(2)$): Let $r \in \mathcal{R}$ be an arbitrary rule such that argument A based on r in $Arg(AT)$ has $\mathtt{h}(A) \leq 2$ and is attacked on a subargument by some argument in $G(AT)$.

Note that there is no argument A based on r in $Arg(AT)$ such that $\mathtt{h}(A) = 1$: if such an argument would exist, each of r's antecedents would be in $\mathcal{K}$, which means that it cannot be attacked by an observation-based argument in $Arg(AT)$, hence cannot be attacked by an argument in $G(AT)$ (Lemma 5).

This implies that there is a rule-based argument for each of the antecedents of r in $Arg(AT)$. Furthermore, there is at least one $a \in \mathtt{ants}(r)$ such that each argument for a is attacked by an argument in $G(AT)$ (otherwise we could reconstruct an argument based on r that is not attacked by any argument in $G(AT)$ on a subargument). Given that $\mathtt{h}(A) = 2$, each argument A' for a has $\mathtt{h}(A') \leq 1$, so by Lemma 5, A' cannot be attacked *on a subargument* by an argument in $G(AT)$. Consequently, it must be attacked on its conclusion: some $a' \in \overline{a}$ must be in $\mathcal{K}$. Then a is considered for relabelling in Algorithm 4 line 5 and labelled $\neg L[a].d$ and $\neg L[a].b$ by case L-D-a and L-B-b. After this, r is added to TODO-SET and considered for relabelling in a later iteration of line 9, where it is labelled $\neg L[r].d$ and $\neg L[r].b$ by case R-D-a and R-B-b.

Induction hypothesis ($P(k)$): For each $r \in \mathcal{R}$: if each argument A based on r in $Arg(AT)$ has $\mathtt{h}(A) \leq k$ and is attacked on a subargument by some argument in $G(AT)$ then r is labelled $\neg L[r].d$ and $\neg L[r].b$.

Induction step ($P(k+1)$): Now let $r \in \mathcal{R}$ be an arbitrary rule such that each argument A for l has $\mathtt{h}(A) \leq k+1$ and is attacked on a subargument by some argument in $G(AT)$. Then there is at least one $a \in \mathtt{ants}(r)$ such that each argument for a in $Arg(AT)$ is attacked by an argument in $G(AT)$ (otherwise we could reconstruct an argument based on r that is not attacked by any argument in $G(AT)$ on a subargument). By Lemma 5, each argument for a in $Arg(AT)$ is attacked by an observation-based argument in $Arg(AT)$.

- If this is an attack on its conclusion, then some $a' \in \overline{a}$ is in $\mathcal{K}$, which would imply that a is considered for relabelling in Algorithm 4 line 5 and labelled $\neg L[a].d$ and $\neg L[a].b$ by case L-D-a and L-B-b.
- Alternatively, $a \notin \mathcal{K}$ and for each rule r' for a, each argument A' based on r' in $Arg(AT)$ has $\mathtt{h}(A') \leq k$ and is attacked on a subargument by some argument in $G(AT)$. Then by the induction hypothesis, each rule r' for a is labelled $\neg L[r'].d$ and $\neg L[r'].b$. This must have happened in Algorithm 4 line 9, after which a was reconsidered for relabelling in line 11 and labelled $\neg L[a].d$ and $\neg L[a].b$ by Algorithm 2 case L-D-b and L-B-c.

After relabelling a in line 5 or 11, r is added to TODO-SET in line 6 or 13. In a later iteration of the while loop, it will be popped and relabelled in line 9 as $\neg L[r].d$ and $\neg L[r].b$ by case R-D-a and R-B-b. Since each argument (based on r) has finite height, this concludes our proof. □

Lemma 17 (Completeness defended labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. Each literal $l \in \mathcal{L}$ that is defended in AT is labelled $L[l] = \langle 0, 1, 0, 0\rangle$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. We proceed by induction.

Proposition ($P(n)$): For each $l \in \mathcal{L}$ for which there is an argument $A \in G(AT)$ with $\mathtt{h}(A) \leq n$, $L[l] = \langle 0, 1, 0, 0\rangle$.

Base case ($P(0)$): For each $l \in \mathcal{L}$ for which there is an argument $A \in G(AT)$ with $\mathtt{h}(A) \leq 0$, $l \in \mathcal{K}$. By Lemma 2, $L_p[l] = \langle 0, 1, 1, 1\rangle$. After preprocessing, l is labelled in Algorithm 4 line 5 as $L[l] = \langle 0, 1, 0, 0\rangle$ by Algorithm 2 case L-O-a and L-B-a.

Induction hypothesis ($P(k)$): For each $l \in \mathcal{L}$ for which there is an argument $A \in G(AT)$ with $\mathtt{h}(A) \leq k$, $L[l] = \langle 0, 1, 0, 0\rangle$.

Induction step ($P(k+1)$): Let $l \in \mathcal{L}$ be an arbitrary literal such that there is an argument $A \in G(AT)$ with $\mathtt{h}(A) \leq k+1$. Then l is labelled $L_p[l] = \langle 0, 1, 1, 1\rangle$ in preprocessing by Lemma 2.

If $l \in \mathcal{K}$, then l is labelled in Algorithm 4 line 5 as $L[l] = \langle 0, 1, 0, 0\rangle$ by Algorithm 2 case L-O-a and L-B-a.

Alternatively, A must be rule-based; let $r = \mathtt{top\text{-}rule}(A)$. By Lemma 2, $L_p[r] = \langle 0, 1, 1, 1\rangle$. By Lemma 4, the fact that $A \in G(AT)$ implies that each argument attacking A is attacked by an observation-based argument in $Arg(AT)$. So for each $A' \in \mathtt{dirsub}(A)$: each argument attacking A' is attacked by an observation-based argument in $Arg(AT)$ (hence $A' \in G(AT)$) and $\mathtt{h}(A') \leq k$. 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 any case, r is added to TODO-SET afterwards (in line 6, 13 or 17). In a later iteration of the while loop, r is popped from TODO-SET and labelled $L[r] = \langle 0, 1, 0, 0\rangle$ by Algorithm 3 case R-O-a and R-B-a.

Given that each argument attacking A is attacked by an observation-based argument in $Arg(AT)$, no $l' \in \overline{l}$ can be in $\mathcal{K}$ (since $\mathcal{K}$ is consistent) and each argument based on some rule r' for some $l' \in \overline{l}$ must be attacked by an argument in $G(AT)$. Then by Lemma 16 each rule r' for each $l' \in \overline{l}$ is labelled $\neg L[r'].d$ and $\neg L[r'].b$.

After labelling one rule r for l as $L[r] = \langle 0, 1, 0, 0\rangle$ and all rules r' for all $l' \in \overline{l}$ as $\neg L[r'].d$ and $\neg L[r'].b$, l is considered for relabelling in either line 11 or 15. At that point, l is labelled $L[l] = \langle 0, 1, 0, 0\rangle$ by case L-O-c and L-B-e.

At this point, we have proved $P(n)$ for each non-negative integer $n \in \mathbb{N}$. Finally, note that each argument has a finite height, which concludes our proof. □

Lemma 18 (Completeness out labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. Each literal $l \in \mathcal{L}$ that is out in AT is labelled $L[l] = \langle 0, 0, 1, 0\rangle$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. Let $l \in \mathcal{L}$ be an arbitrary literal that is out in AT: there is an argument for l in $Arg(AT)$ but each argument is attacked by an argument in the grounded extension. Given that each argument for l is attacked, $l \notin \mathcal{K}$, so there must be a rule-based argument for l based on some $r \in \mathcal{R}$. By Lemma 2, $L_p[r] = \langle 0, 1, 1, 1\rangle$.

- If some $l' \in \overline{l}$ is in $\mathcal{K}$, then l is relabelled in Algorithm 4 line 5 as $L[l] = \langle 0, 0, 1, 0\rangle$ by Algorithm 2 case L-D-a and L-B-b.
- Alternatively, $l \notin \mathcal{K}$ and for each rule r for l, each argument based on r in $Arg(AT)$ is attacked by an argument in $G(AT)$. Then by Lemma 16, each rule r for l is labelled $\neg L[r].d$ and $\neg L[r].b$. This must have happened in Algorithm 4 line 9, after which l is relabelled in line 11 as $L[l] = \langle 0, 0, 1, 0\rangle$ by case L-D-b and R-B-b. □

Lemma 19 (Labelled not defended)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. For each $l \in \mathcal{L}$: if there is no argument for l in $G(AT)$ then l is labelled $\neg L[l].d$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$.

For each $l \in \mathcal{L}$ such that there is no argument for l *at all* in $Arg(AT)$, l is labelled $L_p[l] = L[l] = \langle 1, 0, 0, 0\rangle$ (hence $\neg L[l].d$) by Lemma 3.

Alternatively, suppose that there is an argument for l in $Arg(AT)$: $L_p[l] = \langle 0, 1, 1, 1\rangle$. However, no argument for l is in $G(AT)$, so by Lemma 4, each argument for l in $Arg(AT)$ is attacked by an argument that is not attacked by an observation-based argument. We proceed by induction on the height of arguments.

Proposition ($P(n)$): For each $l \in \mathcal{L}$: if there is an argument for l in $Arg(AT)$, but each argument A for l has a height $\mathtt{h}(A) \leq n$ and is attacked by an argument in $Arg(AT)$ that is not attacked by any observation-based argument in $Arg(AT)$, then l is labelled $\neg L[l].d$.

Base case ($P(1)$): Let l be an arbitrary literal in $\mathcal{L}$ such that there is an argument for l in $Arg(AT)$, but each argument A for l has a height $\mathtt{h}(A) \leq 1$ and is attacked by an argument in $Arg(AT)$ that is not attacked by any observation-based argument in $Arg(AT)$. Note that there is no observation-based argument for l ($l \notin \mathcal{K}$), since these cannot be attacked. There must hence be some rule-based argument that is attacked on its conclusion by an argument for some $l' \in \overline{l}$. If this argument is rule-based, then $l' \in \mathcal{K}$, l is considered for relabelling in Algorithm 4 line 9 and labelled $\neg L[l].d$ by Algorithm 2 case L-D-a. Alternatively, the argument for l' is rule-based: there is some rule r' for l' in $\mathcal{R}$ such that there is an argument based on r' that is not attacked by an observation-based argument. Then by Lemma 15, r' is labelled $\neg L[r'].u$ and $\neg L[r'].o$. This must have happened in Algorithm 4 line 9, after which l is considered for relabelling in line 11 and labelled $\neg L[l].d$ by Algorithm 2 case L-D-c.

Induction hypothesis ($P(k)$): For each $l \in \mathcal{L}$: if there is an argument for l in $Arg(AT)$, but each argument A for l has a height $\mathtt{h}(A) \leq k$ and is attacked by an argument in $Arg(AT)$ that is not attacked by any observation-based argument in $Arg(AT)$, then l is labelled $\neg L[l].d$.

Induction step ($P(k+1)$): Let $l \in \mathcal{L}$ be an arbitrary literal such that each argument A for l has a height $\mathtt{h}(A) \leq k+1$ and is attacked by an argument in $Arg(AT)$ that is not attacked by any observation-based argument in $Arg(AT)$.

- If there is some $l' \in \overline{l}$ for which there is an argument in $Arg(AT)$ that is not-attacked by an observation-based argument, then l is labelled $\neg L[l].d$ by L-D-a or L-D-c (see base case).
- Alternatively, $l \notin \mathcal{K}$ and each rule-based argument for l in $Arg(AT)$ is attacked *on a subargument* by some argument that is not attacked by an argument that is not attacked by an observation-based argument. So for each rule r for l in $\mathcal{R}$, there must be some antecedent $a \in \mathtt{ants}(r)$ such that each argument A' for a has a height $\mathtt{h}(A) \leq k$ and is attacked by an argument in $Arg(AT)$ that is not attacked by any observation-based argument in $Arg(AT)$. By the induction hypothesis, a is labelled $\neg L[a].d$; this must have happened in Algorithm 4 line 5, 11 or 15; in any case, r is added to TODO-SET afterwards (in line 6, 13 or 17). When popped from TODO-SET, r is relabelled in line 9 as $\neg L[r].d$ by case R-D-a. After all rules for l have been relabelled, l is labelled $\neg L[l].d$ in line 11 by case L-D-b. □

Lemma 20 (Completeness blocked labelling)**.** *Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by* JUSTIFICATION-LABEL *(Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. Each $l \in \mathcal{L}$ that is blocked in AT is labelled $L[l] = \langle 0, 0, 0, 1\rangle$.*

Proof. Let $AT = (AS, \mathcal{K})$ be an argumentation theory where $AS = (\mathcal{L}, \mathcal{R}, \overline{})$ and let $L = \langle u, d, o, b\rangle$ be the labelling obtained by JUSTIFICATION-LABEL (Algorithm 4) on $\mathcal{L}$, $\mathcal{R}$, $\overline{}$ and $\mathcal{K}$. Let $l \in \mathcal{L}$ be an arbitrary literal that is blocked in AT. There is an A argument for l in $Arg(AT)$ not in $G(AT)$ (so $l \notin \mathcal{K}$), but not attacked by an argument in $G(AT)$ either. Then A is based on some rule r, no $l' \in \bar{l}$ is in $\mathcal{K}$ and r is not attacked by an argument in $G(AT)$. By Lemma 15, r is labelled $\neg L[r].u$ and $\neg L[r].o$. This must have happened in Algorithm 4 line 9, after which l is considered for relabelling in line 11 and labelled $\neg L[l].o$ by case L-O-c. In addition, l is labelled $\neg L[l].d$ because there is no argument for l in $G(AT)$ (Lemma 19). To conclude, $L[l] = \langle 0, 0, 0, 1\rangle$. □

References

Pietro Baroni, Martin Caminada, and Massimiliano Giacomin. An introduction to argumentation semantics. *The knowledge engineering review*, 26(4):365–410, 2011.

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

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

Sanjay Modgil and H Prakken. Abstract rule-based argumentation. *Handbook of Formal Argumentation*, 1:286, 2018.

Sanjay Modgil and Henry Prakken. A general account of argumentation with preferences. *Artificial Intelligence*, 195: 361–397, 2013.

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

Henry Prakken and Gerard Vreeswijk. Logics for defeasible argumentation. In *Handbook of philosophical logic*, pages 219–318. Springer, 2001.

Yining Wu and Martin Caminada. A labelling-based justification status of arguments. *Studies in Logic*, 3(4):12–29, 2010.