

Relational Acceptability Semantics of Abstract Argumentation ^{*}

Ryuta Arisaka  and Takayuki Ito

Kyoto University, 606-8317 Kyoto, Japan

 ryutaarisaka@gmail.com, ito@i.kyoto-u.ac.jp

Abstract. We introduce *argumentation tuple relational calculus*, adapting tuple relational calculus for acceptability semantics, and derive *relational acceptability semantics* of abstract argumentation. It serves as a theoretical framework to link different types of acceptability semantics relationally. It also allows for refining existing acceptability semantics.

1 Introduction

Abstract argumentation frameworks by Dung [8] capture defeasible/conflicting information as a graph structure. Each node is an argument (information) and each edge from an argument to another argument is an attack from the source argument to the target argument. For the semantics of argumentation graphs, Dung defines several *acceptability semantics* deciding which sets of arguments are acceptable. Each of them can be derived from some of them via (1) set-theoretical operations on it and possibly also (2) the count of attacked arguments. In that sense, they are all linked in these two measures. The relationship among them is thus easy to see. However, various types of acceptability semantics exist by now and the two measures are no longer sufficient for linking them. For example, *multi-agent semantics*, e.g. [5, 2, 3], assigns a subgraph of a given argumentation graph to each agent, obtains (often) Dung acceptability semantics of each of the subgraphs (one per agent), and aggregates them in some way. This procedure is too involved to be emulated by the two measures alone. *Can we obtain a theoretical framework for recovering the linkability?* We present a *formal language for the linking* to address this question. Specifically, noting that much of the difficulty with handling multi-agent semantics is relational, we adapt tuple relational calculus [10] for acceptability semantics to derive argumentation tuple relational calculus, and formulate *relational acceptability semantics* which will be shown to extend the linkability to multi-agent semantics. It allows for easily specialising or generalising existing acceptability semantics, too. A fuller work is in [4].

Related work. Identification of common constraints among different acceptability semantics is popular [6, 1]. However, the constraint identification research focuses on identifying a specific set of formal constraints, and through them, a specific set of acceptability semantics. Thus, even provided there is no dependency among the constraints, they can identify at most 2^N different acceptability semantics with N being the number of them. By contrast, our proposal provides a proper formal language (a fragment of predicate logic) whose semantics is not

^{*} Supported by JSPS KAKENHI Grant Number 21K12028.

only defined at some points but instead defined for any expression allowed in the language. Hence, by capturing acceptability semantics in the language, it becomes possible to overcome the discreteness limitation. Further, the language allows the derivation of acceptability semantics through database queries which, unlike the full predicate logic expressions, can be handled directly in SQL.

2 Technical Preliminaries

$\mathcal{A}$ is an uncountable set of entities, *arguments*. An abstract argumentation framework is normally a finite graph (A, R) with $A \subseteq \mathcal{A}$ and $R \subseteq A \times A$. Given (A, R) , for any $a_1, a_2 \in A$, we say a_1 attacks a_2 just when $(a_1, a_2) \in R$. Given (A, R) , $A' \subseteq A$ and $a \in A$, A' is *conflict-free* iff there is no attack (edge) in $(A', R \cap (A' \times A'))$; A' *defends* a iff, for any argument $a_1 \in A$, if a_1 attacks a , then some argument $a' \in A'$ attacks a_1 ; and A' is *admissible* iff it defends every member of A' and is conflict-free. We obtain 4 different - but well-linked - acceptability semantics of (A, R) . The *complete semantics* of (A, R) comprises every admissible $A' \subseteq A$ that includes every argument it defends. Let Γ denote the complete semantics of (A, R) , then the set comprising: every set-inclusion-wise maximal member A' of Γ is the *preferred semantics* of (A, R) ; every member A' of Γ such that for each $a \in (A \setminus A')$ there is some $a' \in A'$ that attacks a is the *stable semantics* of (A, R) ; the minimum member of Γ is the *grounded semantics* of (A, R) . For each $\mathbf{x} \in \{\text{complete, preferred, stable, grounded}\}$, every member A' of the $\mathbf{x}$ semantics of (A, R) is *acceptable* under the $\mathbf{x}$ semantics. Now, as per [9, 7], we can represent each member $A' \subseteq A$ of $\mathbf{x}$ semantics of (A, R) with a labelling function. Let L denote $\{\text{in, out, und}\}$ and let Λ denote the class of all partial functions $\mathcal{A} \rightarrow L$ as labelling functions. Any member A' of the $\mathbf{x}$ semantics of (A, R) corresponds to some labelling function $\lambda \in \Lambda$ satisfying: (1) $\text{dom}(\lambda) = A$ ($\text{dom}(\lambda)$ is the domain of λ); (2) $\lambda(a) = \text{in}$ just when $a \in A'$; and (3) $\lambda(a) = \text{out}$ just when there is some $a' \in A'$ attacking a . Clearly, the *labelling-based $\mathbf{x}$ semantics* of (A, R) is the set of all labelling functions some member of the $\mathbf{x}$ semantics of (A, R) corresponds to.

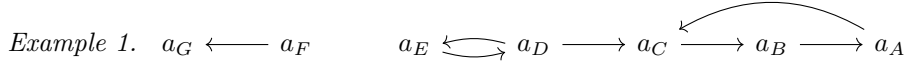
One typical idea of multi-agent semantics [5, 2, 3] is: given an (A, R) , partition A into $A_1, \dots, A_n$ ($\bigcup_{1 \leq i \leq n} A_i = A$), each of which represents arguments put forward by an agent; each agent i ($1 \leq i \leq n$) knows some (A_x, R_x) such that $A_i \subseteq A_x \subseteq A$ and $R_x = R \cap (A_x \times A_x)$, and derives some labelling-based acceptability semantics A_i of (A_x, R_x) . An external observer with the god's perspective aggregates these local labelling-based acceptability semantics $A_1, \dots, A_n$ into a global labelling-based acceptability semantics A_g of (A, R) (where each $\lambda_g \in A_g$ satisfies $\text{dom}(\lambda_g) = A$). Typically, each $\lambda_g \in A_g$ is such that, for each partition A_i , λ_g assigns the same labels to A_i as some $\lambda_i \in A_i$.

tbl ₁	a_G	a_F	a_E	a_D	a_C	a_B	a_A
λ_1	out	in	und	und	und	und	und
λ_2	out	in	in	out	und	und	und
λ_3	out	in	out	in	out	in	out

tbl ₂	a_G	a_F	a_E	a_D
λ_4	out	in	und	und
λ_5	out	in	in	out
λ_6	out	in	out	in

tbl ₃	a_E	a_D	a_C	a_B	a_A
λ_7	out	in	out	in	out

Fig. 1. The tables referred to by tbl₁, tbl₂ and tbl₃ list labelling functions ($\lambda_1(a_G) = \text{out}$, $\lambda_1(a_F) = \text{in}$ and so on in tbl₁, similiary for the others).



This is an argumentation graph. The labelling-based complete semantics of it comprises 3 labelling functions, as shown in (the table referred to by) tbl_1 in Fig. 1. Let us suppose 2 agents. Suppose agent 1 (resp. agent 2) puts forward $a_G, \dots, a_E$ (resp. $a_D, \dots, a_A$) and knows $a_G, \dots, a_D$ (resp. $a_E, \dots, a_A$) as well as attacks among them. Suppose agent 1 (resp. agent 2) uses the labelling-based complete semantics (resp. labelling-based stable semantics). Then, agent 1's (resp. agent 2's) local labelling-based complete (resp. stable) semantics comprises 3 (resp. 1) labelling functions as shown in tbl_2 (resp. tbl_3) in Fig. 1. Suppose the external observer uses the labelling-based complete semantics (tbl_1). Then the multi-agent semantics is a singleton set $\{\lambda_3\}$, since λ_3 is the only labelling function among $\{\lambda_1, \lambda_2, \lambda_3\}$ that can be matched by one of $\lambda_4, \lambda_5, \lambda_6$ for the labels of a_G, a_F and a_E and by λ_7 for the labels of a_D, a_C, a_B and a_A . ♣

3 Argumentation Tuple Relational Calculus and Relational Acceptability Semantics

We now formally develop *argumentation tuple relational calculus* which is a tuple relational calculus for manipulating labelling functions and formulate *relational acceptability semantics*.

Definition 1 (Arg-labelling table). An *arg-labelling table* is a tuple (A, Λ_1) with (1) $A \subseteq_{\text{fin}} \mathcal{A}$, called *header* of (A, Λ_1) , and $\Lambda_1 \subseteq \Lambda$ (*body* of (A, Λ_1)) with any of its member λ satisfying $\text{dom}(\lambda) = A$. ♠

There is no duplicate in the body of an arg-labelling table.

Definition 2 (Relational arg-labelling database). Let $\mathcal{T}$ be a set of table names, let tbl denote its member, and let TBL denote its subset. A *relational arg-labelling database schema* is a tuple $(L, \text{TBL}, \text{header})$ with $\text{header} : \text{TBL} \rightarrow 2^A$ associating a set of arguments to each table name. We denote the set of all relational arg-labelling database schemata by SCHM , and refer to its member by schm . A *relational arg-labelling database for* $\text{schm} \equiv (L, \text{TBL}, \text{header})$ is $db : \text{TBL} \rightarrow 2^A$ satisfying the following condition: for every $\text{tbl} \in \text{TBL}$ and every $\lambda \in db(\text{tbl})$, it holds that $\text{dom}(\lambda) = \text{header}(\text{tbl})$. ♠

Example 2 (Relational arg-labelling database). Observe in Fig. 1 that there are 3 arg-labelling tables in total. The arg-labelling database holding them is expressed with the following relational arg-labelling database schema and relational arg-labelling database with respect to it. Assume $A_1 = \{a_G, \dots, a_A\}$, $A_2 = \{a_G, \dots, a_D\}$, $A_3 = \{a_E, \dots, a_A\}$, $\Lambda_1 = \{\lambda_1, \lambda_2, \lambda_3\}$, $\Lambda_2 = \{\lambda_4, \lambda_5, \lambda_6\}$, and $\Lambda_3 = \{\lambda_7\}$. Then: schm is $(L, \{\text{tbl}_1, \text{tbl}_2, \text{tbl}_3\}, \text{header} \equiv \{\text{tbl}_1 \mapsto A_1, \dots, \text{tbl}_3 \mapsto A_3\})$; and db is $\{\text{tbl}_1 \mapsto \Lambda_1, \dots, \text{tbl}_3 \mapsto \Lambda_3\}$. It holds that $\text{dom}(\lambda_1) = A_1 = \text{header}(\text{tbl}_1)$, and similarly for all the others. ♣

Syntax and semantics of formal query language. The syntax of the query language is defined almost as tuple relational calculus [10], save we introduce a dyadic function count to count the number of a certain label assigned to the header arguments. This additional function helps keep formal expressions concise.

Definition 3 (Query formulas). V is an uncountable set of variables. An *atomic query formula* with respect to $\text{schm} \equiv (L, \text{TBL}, \text{header})$ is any below. (1) $v_1.a_1 \doteq v_2.a_2$ for $v_1, v_2 \in V$ and $a_1, a_2 \in \mathcal{A}$. (2) $v_1.a_1 \doteq l$ for $v_1 \in V$, $a_1 \in \mathcal{A}$ and $l \in L$. (3) $\text{tbl}[v_1]$ for $v_1 \in V$ and $\text{tbl} \in \text{TBL}$. (4) $\text{count}(v_1, l_1) \dot{\leq} \text{count}(v_2, l_2)$ for $v_1, v_2 \in V$ and $l_1, l_2 \in L$. (5) $\text{count}(v_1, l_1) \dot{\leq} n$ for $v_1 \in V$, $l_1 \in L$ and $n \in \mathbb{N}$. (6) $n \dot{\leq} \text{count}(v_1, l_1)$ for $v_1 \in V$, $l_1 \in L$ and $n \in \mathbb{N}$. Then, any below is a *query formula* with respect to schm . We may refer to a query formula by $\mathbf{F}$. (1) an atomic query formula with respect to schm . (2) $\neg \mathbf{F}_1$ if $\mathbf{F}_1$ is a query formula. (3) $\mathbf{F}_1 \wedge \mathbf{F}_2$ if $\mathbf{F}_1$ and $\mathbf{F}_2$ are query formulas. (4) $\mathbf{F}_1 \vee \mathbf{F}_2$ if $\mathbf{F}_1$ and $\mathbf{F}_2$ are query formulas. (5) $\exists v : A[\mathbf{F}_1]$ if v is in V , A is a set of arguments and $\mathbf{F}_1$ is a query formula. (6) $\forall v : A[\mathbf{F}_1]$ if v is in V , A is a set of arguments and $\mathbf{F}_1$ is a query formula. ♠

The following semantics of the language is fairly standard to the database theory.

Definition 4 (Semantics). Let $\text{eval} : V \rightarrow A$ be an interpretation function such that $\text{eval}(v) \in A$, and let a ‘semantic structure’ be a tuple $(\text{schm}, db, \text{eval})$ for schm and db for schm . We inductively define $(\text{schm}, db, \text{eval}) \models \mathbf{F}$ for any $(\text{schm}, db, \text{eval})$ and any $\mathbf{F}$ as follows.

- $(\text{schm}, db, \text{eval}) \models v_1.a_1 \doteq v_2.a_2$ iff $a_1 \in \text{dom}(\text{eval}(v_1))$ and $a_2 \in \text{dom}(\text{eval}(v_2))$ and $\text{eval}(v_1)(a_1) = \text{eval}(v_2)(a_2)$.
- $(\text{schm}, db, \text{eval}) \models v_1.a_1 \doteq l$ iff $a_1 \in \text{dom}(\text{eval}(v_1))$ and $\text{eval}(v_1)(a_1) = l$.
- $(\text{schm}, db, \text{eval}) \models \text{tbl}[v_1]$ iff $\text{eval}(v_1) \in db(\text{tbl})$.
- $(\text{schm}, db, \text{eval}) \models \text{count}(v_1, l_1) \dot{\leq} \text{count}(v_2, l_2)$ iff $|\{a \in \text{dom}(\text{eval}(v_1)) \mid \text{eval}(v_1)(a) = l_1\}| \leq |\{a \in \text{dom}(\text{eval}(v_2)) \mid \text{eval}(v_2)(a) = l_2\}|$
- $(\text{schm}, db, \text{eval}) \models \text{count}(v_1, l_1) \dot{\leq} n$ iff $|\{a \in \text{dom}(\text{eval}(v_1)) \mid \text{eval}(v_1)(a) = l_1\}| \leq n$.
- $(\text{schm}, db, \text{eval}) \models n \dot{\leq} \text{count}(v_1, l_1)$ iff $n \leq |\{a \in \text{dom}(\text{eval}(v_1)) \mid \text{eval}(v_1)(a) = l_1\}|$.
- $(\text{schm}, db, \text{eval}) \models \exists v : A[\mathbf{F}]$ iff there is some λ such that $\text{dom}(\lambda) = A$ and that $(\text{schm}, db, \text{eval}') \models \mathbf{F}$ where eval' is almost exactly eval except $\text{eval}'(v) = \lambda$.
- $(\text{schm}, db, \text{eval}) \models \forall v : A[\mathbf{F}]$ iff, for every λ , if $\text{dom}(\lambda) = A$, then $(\text{schm}, db, \text{eval}') \models \mathbf{F}$ where eval' is almost exactly eval except $\text{eval}'(v) = \lambda$.

Due to space, we omit the cases of $\neg \mathbf{F}$, $\mathbf{F}_1 \wedge \mathbf{F}_2$ and $\mathbf{F}_1 \vee \mathbf{F}_2$ which are standard. We say that $(\text{schm}, db, \text{eval})$ models $\mathbf{F}$ iff $(\text{schm}, db, \text{eval}) \models \mathbf{F}$. ♠

For atomic formulas, $v_1.a_1 \doteq v_2.a_2$ tests whether the label of a_1 assigned by $\lambda_x \equiv \text{eval}(v_1)$ and that of a_2 assigned by $\lambda_y \equiv \text{eval}(v_2)$ matches. The first two attached conditions force $a_1 \in \text{dom}(\lambda_x)$ and $a_2 \in \text{dom}(\lambda_y)$. $v_1.a_2 \doteq l$ tests whether the label of a_1 assigned by $\lambda_x \equiv \text{eval}(v_1)$ is l . $\text{tbl}[v_1]$ tests whether $\lambda_x \equiv \text{eval}(v_1)$ is in the body of the arg-labelling table $db(\text{tbl})$. $\text{count}(v_1, l_1) \dot{\leq} \text{count}(v_2, l_2)$ compares the number of arguments assigned l_1 by $\lambda_1 \equiv \text{eval}(v_1)$ and that of arguments assigned l_2 by $\lambda_2 \equiv \text{eval}(v_2)$. Similarly for the other two atomic formulas.

Definition 5 (Relational acceptability semantics). An *arg-labelling query* is an expression $\{v : A \mid \mathbf{F}\}$, whereby the only free variable in $\mathbf{F}$ is v . The semantics of a query $\{v : A \mid \mathbf{F}\}$ with respect to some schm and db is the set of all λ satisfying $(\text{schm}, db, \text{eval}) \models \mathbf{F}$ with $\text{eval}(v) = \lambda$. Let $\|\{v : A \mid \mathbf{F}\}\|$ denote the set. A *relational acceptability semantics* with respect to schm and db is some $\|\{v : A \mid \mathbf{F}\}\|$ with respect to schm and db . ♠

Basic database queries such as selecting rows of a table, selecting columns of a table, and joining two tables, produce certain effects on arg-labelling database with associated arg-labelling tables.

Example 3 (Selecting columns). In Fig. 1, tbl_2 (or the arg-labelling table referred to by tbl_2) is the result of selecting 4 columns a_G, a_F, a_E and a_D of tbl_1 . The corresponding relational acceptability semantics is $\|\{v : \{a_G, a_F, a_E, a_D\} \mid \text{tbl}_1[v]\}\|$ with respect to the same schm and db in Example 2. To see to it, we firstly enumerate all λ with $(\text{schm}, db, \text{eval}) \models \text{tbl}_1[v]$ for $\text{eval}(v) = \lambda$, which are λ_1, λ_2 and λ_3 ; for each of them, we force the domain to $\{a_G, \dots, a_D\}$, to obtain λ_4, λ_5 and λ_6 . We saw in Section 2 that multi-agent semantics has the step of restricting attention to a subset of arguments. This process is *explainable* as column selection operations. The time complexity of this query is $O(n)$ for the number of rows n . ♣

Example 4 (Selecting rows). The labelling-based complete/preferred/stable/grounded semantics is explained through row selection on the labelling-based complete semantics. With tbl_1 , $\|\{v : \{a_G, \dots, a_A\} \mid \text{tbl}_1[v] \wedge \neg 1 \leq \text{count}(v, \text{und})\}\|$ with respect to the same schm and db is the labelling-based stable semantics. The time complexity of this query is $O(n)$ for the number of rows n . ♣

The process of aggregation of local labelling-based acceptability semantics into a multi-agent semantics is explainable with condition join. We let $\bigwedge_{i \in \{G, \dots, A\}} v.a_i \doteq v_1.a_i$ abbreviate $v.a_G \doteq v_1.a_G \wedge \dots \wedge v.a_A \doteq v_1.a_A$. Similarly for others.

Example 5 (Condition join). Let us join tbl_2 and tbl_3 in such a way that for any λ in the body of tbl_1 , λ is in the resulting table's body iff there is some $\lambda_x \in db(\text{tbl}_2)$ and some $\lambda_y \in db(\text{tbl}_3)$ such that $\lambda_x(a_i) = \lambda(a_i)$ and $\lambda_y(a_j) = \lambda(a_j)$ hold for every $a_i \in \{a_G, a_F, a_E\}$ and every $a_j \in \{a_D, \dots, a_A\}$. The relational acceptability semantics of this operation is $\|\{v : \{a_G, \dots, a_A\} \mid \text{tbl}_1[v] \wedge \exists v_1 : \{a_G, \dots, a_D\}[\text{tbl}_2[v_1] \wedge (\bigwedge_{i \in \{G, \dots, E\}} v.a_i \doteq v_1.a_i) \wedge \exists v_2 : \{a_E, \dots, a_A\}[\text{tbl}_3[v_2] \wedge \bigwedge_{i \in \{D, \dots, A\}} v.a_i \doteq v_2.a_i]]\}\|$ with respect to the same schm and db . This, incidentally, forms the multi-agent semantics we saw in Example 1. With the sequential two joins, the time complexity of this query is capped by $O(n^2)$ for the largest number of rows n in the 3 tables. ♣

To unify multi-agent semantics and the traditional labelling-based semantics as a relational acceptability semantics with respect to some schm and db , it suffices to let them cover the labelling-based complete semantics of (A, R) as well as $(A_i, R \cap (A_i \times A_i))$ ($1 \leq i \leq n$) where A_i is the arguments agent i knows.

Theorem 1. *Given (A, R) and n partitions of A into $A_1, \dots, A_n$ (representing the arguments each of the agents expressed), suppose for every $1 \leq i \leq n$ that there is some A'_i (representing the arguments each of the agents knows) such that $A_i \subseteq A'_i \subseteq A$. Let TBL denote $\{\text{tbl}_{ad}, \text{tbl}_{1ad}, \dots, \text{tbl}_{nad}\}$ and let $\Lambda_{ad}, \Lambda_{1ad}, \dots, \Lambda_{nad}$ be the labelling-based complete semantics of: $(A, R), (A'_1, R \cap (A'_1 \times A'_1)), \dots, (A'_n, R \cap (A'_n \times A'_n))$. Now, let $\text{schm} \equiv (L, TBL, \text{header})$ and db be such that $\text{header}(\text{tbl}_{ad}) = A$, $\text{header}(\text{tbl}_{iad}) = A'_i$ ($1 \leq i \leq n$), $db(\text{tbl}_{ad}) = \Lambda_{ad}$ and $db(\text{tbl}_{iad}) = \Lambda_{iad}$ ($1 \leq i \leq n$). Then, the labelling-based complete/preferred/stable/grounded semantics of (A, R) and the multi-agent semantics of (A, R) are a relational acceptability semantics with respect to schm and db .*

This shows that the argumentation tuple relational calculus makes the acceptability semantics linkable in the single relational perspective. The relational perspective helps fine-tune existing acceptability semantics, too. As we saw, multi-

agent semantics derives a local view (or local views) covering a part of an argumentation graph in order to compute an output. We can see it as a specialisation of the following *partial semantics*. In Fig. 1, tbl_2 is a partial semantics of tbl_1 .

Proposition 1 (Partial semantics). *Given (A, R) , assume some labelling-based acceptability semantics of it Λ^{sem} such that each $\lambda \in \Lambda^{\text{sem}}$ satisfies $\text{dom}(\lambda) = A$. For an $A_1 \subseteq A$, we say A_1 is a partial semantics of Λ^{sem} iff there is some $A_1 \subseteq A$ such that all the following conditions hold. (1) For every $\lambda_x \in A_1$, $\text{dom}(\lambda_x) = A_1$ holds. (2) For every $\lambda_x \in A_1$ and for every $a_1 \in A_1$, there is some $\lambda \in \Lambda^{\text{sem}}$ such that $\lambda_x(a_1) = \lambda(a_1)$. (3) For every $\lambda \in \Lambda^{\text{sem}}$ and for every $a_1 \in A_1$, there is some $\lambda_x \in A_1$ such that $\lambda_x(a_1) = \lambda(a_1)$. Assume TBL is some set of table names, then every partial semantics of Λ^{sem} is a relational acceptability semantics with respect to $\text{schm} \equiv (L, \{\text{tbl}_{\text{sem}}\} \cup \text{TBL}, \text{header})$ and db where tbl_{sem} and db satisfy the following conditions: $\text{header}(\text{tbl}_{\text{sem}}) = A$ and $\text{db}(\text{tbl}_{\text{sem}}) = \Lambda^{\text{sem}}$. Given the schm and db , it is polynomial-time computable.*

4 Conclusions

Given a plethora of acceptability semantics being proposed in various ways, there was always a question of how they may link in what way. As we showed, it is reasonable to let a formal language take care of the linking. We formulated the relational acceptability semantics by formulating argumentation tuple relational calculus. We also showed how it assists find more general acceptability semantics from an existing one, which should complement the recent endeavour [6, 1] to try to find reasonable formal constraints for characterising acceptability semantics for specialised abstract argumentations. A fuller work is in [4].

References

1. L. Amgoud and J. Ben-Naim. Weighted Bipolar Argumentation Graphs: Axioms and Semantics. In *IJCAI*, pages 5194–5198, 2018.
2. R. Arisaka and S. Bistarelli. Defence Outsourcing in Argumentation. In *COMMA*, pages 353–360, 2018.
3. R. Arisaka, J. Dauphin, K. Satoh, and L. van der Torre. Multi-agent Argumentation and Dialogue. *IfCoLog Journal of Logics and their Applications*, 9(4):921–954, 2022.
4. R. Arisaka and T. Ito. Relational Argumentation Semantics. *arXiv:2104.12386*, 2021.
5. R. Arisaka, K. Satoh, and L. van der Torre. Anything you say may be used against you in a court of law: Abstract Agent Argumentation (Triple-A). In *AICOL*, pages 427–442, 2017.
6. P. Baroni, A. Rago, and F. Toni. How Many Properties Do We Need for Gradual Argumentation? In *AAAI*, pages 1736–1743, 2018.
7. M. Caminada. On the Issue of Reinstatement in Argumentation. In *JELIA*, pages 111–123, 2006.
8. P. M. Dung. On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming, and n-Person Games. *Artificial Intelligence*, 77(2):321–357, 1995.
9. H. Jakobovits and D. Vermeir. Robust Semantics for Argumentation Frameworks. *Journal of Logic and Computation*, 9:215–261, 1999.
10. R. Ramakrishnan and J. Gehrke. *Database Management Systems (3rd edition)*. McGraw-Hill, 2002.