Constrained Assumption-Based Argumentation Frameworks

Emanuele De Angelis
CNR-IASI
Rome, Italy
emanuele.deangelis@iasi.cnr.it

Fabio Fioravanti
University of Chieti - Pescara
Pescara, Italy
fabio.fioravanti@unich.it

Maria Chiara Meo
University of Chieti - Pescara
Pescara, Italy
mariachiara.meo@unich.it

Alberto Pettorossi
University of 'Tor Vergata'
Rome, Italy
pettorossi@info.uniroma2.it

Maurizio Proietti
CNR-IASI
Rome, Italy
maurizio.proietti@iasi.cnr.it

Francesca Toni
Imperial
London, United Kingdom
ft@imperial.ac.uk

ABSTRACT

Assumption-based Argumentation (ABA) is a well-established form of structured argumentation. ABA frameworks with an underlying atomic language are widely studied, but their applicability is limited by a representational restriction to ground (variable-free) arguments and attacks built from propositional atoms. In this paper, we lift this restriction and propose a novel notion of *constrained ABA* (*CABA*), whose components, as well as arguments built from them, may include constrained variables, ranging over possibly infinite domains. We define non-ground semantics for CABA, in terms of various notions of non-ground attacks. We show that the new semantics conservatively generalise standard ABA semantics.

KEYWORDS

Assumption-based Argumentation; Non-ground Argumentation; Constraints

ACM Reference Format:
Emanuele De Angelis, Fabio Fioravanti, Maria Chiara Meo, Alberto Pettorossi, Maurizio Proietti, and Francesca Toni. 2026. Constrained Assumption-Based Argumentation Frameworks. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/KRAP9309

1 INTRODUCTION

Assumption-based Argumentation (ABA) [5, 8, 12, 27] is a well-established form of structured argumentation, whereby arguments are deductions for claims, supported by rules and assumptions, and attacks between arguments are generated whenever the claim of the attacking argument is contrary to an assumption supporting the attacked argument. The semantics of ABA frameworks is defined in terms of various notions of *acceptable extensions*, amounting to sets of assumptions/arguments that can be deemed to defend themselves against attacks, in some sense.

ABA has been advocated as a unifying framework for various forms of non-monotonic reasoning [5], including logic programming with negation as failure under several semantics (where assumptions are negation as failure literals of the form *not p*, and the contrary of *not p* is *p*). The logic programming instance of ABA finds application in several domains, including legal reasoning [13], planning [14], healthcare [9, 24, 29], and causal discovery [23]. However, while other instances of ABA, such as the one to capture default logic, rely upon first-order logic as the underlying deductive system [5], in the logic programming instance sentences making up rules, assumptions and their contraries need to be *propositional* atoms.

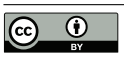
In this paper we propose *Constrained ABA* (*CABA*), dropping this restriction, as illustrated next in a legal setting.

Example 1.1. Let us consider the following CABA rules modelling an AI system designed to support a tax department employee to decide whether or not a person should pay tax.

R1. $must\_pay\_tax(P) \leftarrow income(P, I),\ I \geq 0,\ nonexempt(P)$
R2. $exempt(P) \leftarrow income(P, I),\ I \geq 0,\ I \leq 16,000,\ salary\_income(P)$
R3. $other\_incomes(P) \leftarrow foreign\_income(P, F),\ F \geq 10,000$

These rules serve as templates, with variables P, I, F standing as placeholders for any values, in the spirit of argument schemes with critical questions [20, 28] and logic programming (where variables are implicitly universally quantified at the front of each rule). Rule R1 states that anyone is required to pay taxes if he/she has an income and has no exemption (formalised with assumption $nonexempt(P)$, with contrary $exempt(P)$). The exemption applies to people having as income only their salary (formalised with assumption $salary\_income(P)$, with contrary $other\_incomes(P)$), provided that it is not above the threshold of 16,000 (in some currency). Having other sources of income is then specified by Rule R3, which states that any income from a foreign country above 10,000 should be considered additional income. To represent facts about individuals in ABA, the rules, assumptions, and contraries should be grounded. Note that this may not be possible if the individuals to whom the rules apply are not known up-front. Without knowing those individuals, while grounding may be practicable (yet inefficient) for Rule R2, it would require fixing an upper bound on both the income I in Rule R1 and the foreign income F in the Rule R3, which may not be feasible or desirable in practice. Constraints, such as $I \geq 0$, in CABA rules make it possible to avoid considering all ground instances altogether.

As illustrated by Example 1.1, CABA frameworks may include constrained variables, ranging over possibly infinite domains. From a practical perspective, CABA frameworks enable the use of specific constraint domains and corresponding solvers within the declarative paradigm of ABA. From a theoretical perspective, this approach



Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/KRAP9309

offers a unified framework for capturing and analysing the use of constraints alongside ABA.

Contributions.
(1) We define the novel concept of CABA frameworks (Sect. 4) and notions of non-ground constrained arguments (Sect. 5), attacks (Sect. 6) and extension-based semantics for CABA (Sect. 7), focusing on conflict-free, admissible, and stable extensions.
(2) We show (Sect. 7.1) that the new notions amount to the standard ABA notions after grounding (as well as conservatively generalising those in standard ABA).
(3) We define (Sect. 7.2) novel notions of extensions for CABA, without explicit reference to grounding of standard ABA, and we identify conditions on the theory of constraints for which they are equivalent to the ones obtained by grounding. These new notions allow us to effectively construct finite non-ground CABA extensions also in cases where their ground counterpart are indeed infinite.

The proofs of all results not included in the paper are in the extended version available at http://arxiv.org/abs/2602.13135.

2 RELATED WORK

Various ABA instances have been considered, to capture existing non-monotonic formalisms [5]. Some of these instances, notably corresponding to default logic and circumscription, rely upon first-order logic as the underlying deductive system and can thus be deemed non-ground/non-propositional. Rather than identifying a new instance of ABA which integrates reasoning with constrained variables, we define a novel framework, CABA, singling out reasoning with variable instances and constraints. CABA restricts attention to atomic languages, in the spirit of logic programming instances of ABA [5], and integrates a stand-alone constraint solver in the spirit of Constraint Logic Programming (CLP) [16, 17].

Our CABA frameworks share some representational characteristics with CLP with negation as failure [7, 17]. However, our focus is on the definition of argumentative semantics rather than top-down procedural machinery (e.g. SLDNF-variants with constraints) as in much CLP research [16, 25, 26].

Of particular relevance to our work is Answer Set Programming (ASP) with constraints, s(CASP) [3], given its use of stable models semantics related to stable extensions for CABA. But s(CASP) too has a procedural, top-down spin. Other proposals for integrating reasoning with constraints and ASP, notably [6], focus on a semantic angle via the use of the Here and There logic, but are restricted to stable models whereas we can accommodate other semantics (admissible extensions in this paper).

Frameworks for Abductive Logic Programming with constraints [1, 21] are also related to our work, but they focus on the completion semantics for logic programming [7, 18] rather than variants of the stable and admissible extension semantics for ABA [5] as we do.

Within argumentation, argument schemes [20, 28] are templates for arguments and may thus be seen as a form of non-ground argumentation, but lack a semantics. Also, non-ground variants of ABA are used in ABA Learning [10], but as templates, in the spirit of argument schemes. Also, ABA Learning cannot accommodate constraints. Finally, DeLP [15] is another form of argumentation relying upon logic programming notions, and equipped with a procedural semantics accommodating, like in CABA, rule schemata and constraints therein. However, differently from DeLP, our CABA adopts extension-based semantics in the spirit of [11].

3 BACKGROUND

First-order languages. A first order language is a set of first order formulas constructed, as usual [22], from given sets of variables, function symbols, and predicate symbols. *Atomic languages* are first order languages whose formulas are *atomic formulas* only. Those atomic formulas, also called *atoms*, are made out of predicate symbols applied to terms.

We use capital letters, e.g. X, to denote variables, lower-case letters, e.g. t, to denote terms, and sans-serif letters, e.g. X and t, to denote tuples of variables and tuples of terms, respectively.

A *substitution* ϑ is a total function mapping n (≥ 0) distinct variables $X_1, \ldots, X_n$ to n terms $t_1, \ldots, t_n$, and it is denoted by $\{X_1/t_1, \ldots, X_n/t_n\}$ or simply $\{\mathsf{X}/\mathsf{t}\}$. We assume that each t_i is distinct from X_i. A *variable renaming* (or *renaming*, for short) is a substitution $\{\mathsf{X}/\mathsf{Y}\}$, where X and Y are tuples of variables and Y is a permutation of X [2]. Given a term t, we denote by $vars(t)$ the set of variables occurring in t. Given a substitution ϑ and a term t, an *instance of* t (*via* ϑ), denoted $t\vartheta$, is the term obtained by replacing each variable occurrence X in t by $\vartheta(X)$. If ϑ is a renaming, then $t\vartheta$ is a *variable renaming* (or *renaming*, for short) *of* t (*via* ϑ). A term t is *ground* if $vars(t) = \emptyset$ and, if $vars(t\vartheta) = \emptyset$, then the substitution ϑ is said to be *grounding*.

We lift the function *vars* and the notions of 'being an instance of', 'being a renaming of', and 'being ground' to tuples, sets, and all the other syntactic constructs used in this paper.

Given a set $V = \{X_1, \ldots, X_n\}$ of variables and a formula B, $\exists_V B$ denotes the formula $\exists X_1 \ldots \exists X_n\, B$. The *existential closure of* B, denoted $\exists(B)$, is $\exists_V B$, where V is the set of the variables in B outside the scope of a quantifier. The notation $\exists_{-V} B$ is a shorthand for $\exists_{vars(B)\setminus V} B$. Similar terminology and notations are used for $\forall$.

We say that an atomic language $\mathcal{L}$ is *predicate closed* if, for every predicate p of arity n in $\mathcal{L}$ and n-tuple t of terms in $\mathcal{L}$, $p(\mathsf{t})$ is in $\mathcal{L}$.

Deductive systems. ABA (as well as our constrained version thereof) uses a *deductive system* $(\mathcal{L}, \mathcal{R})$ consisting of a language $\mathcal{L}$ (not necessarily first-order, in general) and a set $\mathcal{R}$ of (inference) rules of the form $s_0 \leftarrow s_1, \ldots, s_m$ (with $m \geq 0$ and, for all i, with $0 \leq i \leq m$, $s_i \in \mathcal{L}$); s_0 and the sequence $s_1, \ldots, s_m$ are called the *head* and the *body* of the rule, respectively. If $m = 0$ then the rule is represented as $s_0 \leftarrow$ and it is called a *fact*. With an abuse of terminology, we state that every rule $s_0 \leftarrow s_1, \ldots, s_m$ in $\mathcal{R}$ belongs to $\mathcal{L}$.

Theories of constraints. We consider a *theory of constraints*, denoted $\mathcal{CT}$, which is based on a first order language $\mathcal{K}$ whose atomic formulas C are called *atomic constraints* (or *constraints*, for short). Constraints are over an algebraic structure with domain D. $\mathcal{CT}$ is equipped with a notion of validity: we write $\mathcal{CT} \models \varphi$ to denote that the formula φ of $\mathcal{K}$ is *valid* in $\mathcal{CT}$. A set $\{c_1, \ldots, c_n\} \subseteq C$ of constraints is said to be *consistent* if $\mathcal{CT} \models \exists(c_1 \wedge \ldots \wedge c_n)$. We assume that $\mathcal{CT}$ is a first order theory with equality, which is interpreted as the identity on D and includes the *Clark Equality Theory* axiomatising the identity between ground terms [19]. Examples of $\mathcal{CT}$'s are the linear integer arithmetic (LIA) and the linear rational arithmetic (LRA).

Assumption-based argumentation (*ABA*). An *ABA framework* [5] (presented here following more recent papers [8, 12, 27]) is a 4-tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$, where:
- $\langle \mathcal{L}, \mathcal{R} \rangle$ is a deductive system;
- $\mathcal{A} \subseteq \mathcal{L}$ is a non-empty set of *assumptions*;[1]
- $\overline{\ }$ is a total mapping from $\mathcal{A}$ into $\mathcal{L}$, where $\overline{a}$ is called the *contrary* of a, for $a \in \mathcal{A}$.

An ABA framework is *flat* if assumptions are not heads of rules.

Example 3.1. $\langle \mathcal{L}_B, \mathcal{R}_B, \mathcal{A}_B, \overline{\ }^B \rangle$ is an ABA framework, call it B:
- $\mathcal{L}_B = \{p(i),\ q(i),\ r(i),\ a(i),\ b(i) \mid i = 1, 2\}$;
- $\mathcal{R}_B = \{p(1) \leftarrow a(1), q(1) \leftarrow b(1), r(1) \leftarrow,\ p(2) \leftarrow a(2), q(2) \leftarrow b(2)\}$;
- $\mathcal{A}_B = \{a(1),\ b(1),\ a(2),\ b(2)\}$;
- $\overline{a(1)}^B = q(1), \quad \overline{a(2)}^B = q(2), \quad \overline{b(1)}^B = r(1), \quad \overline{b(2)}^B = r(2)$.

In flat ABA frameworks, *arguments* are deductions of claims supported by rules and assumptions, and *attacks* are directed at the assumptions which support arguments. Formally, following [12]:
- *An argument for a claim $s \in \mathcal{L}$ supported by $R \subseteq \mathcal{R}$ and $A \subseteq \mathcal{A}$*, denoted $A \vdash_R s$, is a finite tree such that: (i) the root is s, (ii) every non-leaf node s' has as children all and only the sentences of the body of exactly one rule Ri in R whose head is s', or, if Ri is a fact, s' has the only child *true* (representing the empty body), and (iii) every leaf is either an assumption in A or *true*. A is the set of all assumptions in the tree and R is the set of all rules used to construct the tree.

  Note that, for every assumption $\alpha \in \mathcal{A}$, there is an argument $\{\alpha\} \vdash_\emptyset \alpha$, which is a tree consisting of the single node α.
- Argument $A_1 \vdash_{R_1} s_1$ *attacks* argument $A_2 \cup \{a\} \vdash_{R_2} s_2$ iff $s_1 = \overline{a}$.

In the remainder, we will often omit to specify the supporting rules in arguments (as attacks do not depend on them.)

Example 3.2. In the ABA framework B of Example 3.1 we have the following set Arg_B of arguments: $\big\{\{a(1)\} \vdash p(1),\ \{b(1)\} \vdash q(1),\ \emptyset \vdash r(1), \{a(2)\} \vdash p(2),\ \{b(2)\} \vdash q(2),\ \{a(1)\} \vdash a(1),\ \{a(2)\} \vdash a(2),\ \{b(1)\} \vdash b(1),\ \{b(2)\} \vdash b(2)\big\}$. Among other attacks, we have that $\emptyset \vdash r(1)$ attacks $\{b(1)\} \vdash q(1)$, as $\overline{b(1)}^B = r(1)$.

Given a flat ABA framework $F = \langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$, let Arg be the set of all arguments of F and Att be the set of all attacks of F, i.e. $Att = \{(\alpha,\beta) \in Arg \times Arg \mid \alpha \text{ attacks } \beta\}$. Then (Arg, Att) is an *Abstract Argumentation* (AA) framework [11] and the standard semantics for the latter can be used to determine the semantics for the ABA framework [27].[2] For example, we say that: for all $\Gamma \subseteq Arg$,
- Γ is a *conflict-free extension* iff $\nexists\, \alpha, \beta \in \Gamma$ s.t. $(\alpha, \beta) \in Att$;
- Γ is an *admissible extension* iff (i) Γ is conflict-free, and (ii) $\forall \alpha \in \Gamma$, $\forall \beta \in Arg$ if $(\beta, \alpha) \in Att$, then $\exists\, \gamma \in \Gamma$ s. t. $(\gamma, \beta) \in Att$ (i.e. Γ "attacks" every argument that attacks any of its arguments);
- Γ is a *stable extension* iff (i) Γ is conflict-free, and (ii) $\forall \beta \in Arg \setminus \Gamma$, $\exists\, \alpha \in \Gamma$ s.t. $(\alpha, \beta) \in Att$ (i.e. Γ "attacks" all arguments it does not contain).

Example 3.3. The AA framework of the ABA framework B of Example 3.1 is (Arg_B, Att_B) with Arg_B as given in Example 3.2 and $Att_B = \big\{\big(\emptyset \vdash r(1), \{b(1)\} \vdash q(1)\big), \big(\emptyset \vdash r(1), \{b(1)\} \vdash b(1)\big)\big\} \cup \big\{\big(\{b(X)\} \vdash q(X), \{a(X)\} \vdash p(X)\big), \big(\{b(X)\} \vdash q(X), \{a(X)\} \vdash a(X)\big) \mid X = 1, 2\big\}$. Then, the set $\big\{\emptyset \vdash r(1),\ \{a(1)\} \vdash p(1),\ \{a(1)\} \vdash a(1),\ \{b(2)\} \vdash q(2), \{b(2)\} \vdash b(2)\big\}$ of arguments is the only stable extension of B. That extension is also admissible. The claims in that stable extension make out the "model" $\{r(1), p(1), a(1), q(2), b(2)\}$ [27].

[1] The non-emptiness requirement can always be satisfied by including in $\mathcal{A}$ a *bogus assumption*, with its own contrary, neither occurring elsewhere in the ABA framework. For conciseness, we will not write this assumption and its contrary explicitly.

[2] ABA semantics were originally defined in terms of sets of assumptions and attacks between them [5], but can be reformulated, for flat ABA frameworks, in terms of sets of arguments and attacks between them (see [27]), as given here.

4 CABA FRAMEWORKS

This section introduces a novel formalism that integrates a generic constraint-based computational mechanism into ABA frameworks.

Definition 4.1. A *Constrained Assumption-Based Argumentation* (*CABA*) *framework* is a 6-tuple $\langle \mathcal{L}_c, C, \mathcal{R}, \mathcal{CT}, \mathcal{A}, \overline{\ } \rangle$, where:
- $\mathcal{L}_c$ is an atomic language;
- $C \subseteq \mathcal{L}_c$ is a set of atomic constraints;
- $\mathcal{R}$ is a set of (*inference*) *rules* of the form $s_0 \leftarrow s_1, \ldots, s_m$, where $s_0 \in \mathcal{L}_c \setminus C$, $m \geq 0$, and, for all i, with $1 \leq i \leq m$, $s_i \in \mathcal{L}_c$;
- $\mathcal{CT}$ is a theory of constraints whose set of atomic constraints is C;
- $\mathcal{A} \subseteq \mathcal{L}_c \setminus C$ is a non-empty set of *assumptions*;
- $\overline{\ }$ is a total mapping from $\mathcal{A}$ into $\mathcal{L}_c \setminus C$, s.t. if $p(\mathrm{t}_1), p(\mathrm{t}_2) \in \mathcal{A}$, then there exists a predicate symbol, say cp, with $\overline{p(\mathrm{t}_1)} = cp(\mathrm{t}_1)$ and $\overline{p(\mathrm{t}_2)} = cp(\mathrm{t}_2)$; for all $a \in \mathcal{A}$, $\overline{a}$ is called the *contrary* of a.

The atomic languages $\mathcal{L}_c$, C, and $\mathcal{A}$ are all predicate closed.

In the remainder, unless otherwise specified, we assume we are given a CABA framework $F_c = \langle \mathcal{L}_c, C, \mathcal{R}, \mathcal{CT}, \mathcal{A}, \overline{\ } \rangle$.

Note that, differently from standard ABA, in CABA the rules may be non-ground. When rules have variables, they represent rule schemata whose variables can be instantiated by terms. For example, given rule R: $p(X) \leftarrow X < 3, a(X)$ and the substitution $\vartheta = \{X/2\}$, we get the ground rule $R\vartheta$: $p(2) \leftarrow 2 < 3, a(2)$.

Here, we focus on *flat* CABA frameworks, that is, we stipulate that no assumptions occur in heads of rules.

Example 4.2. Let $V = \{X, Y, \ldots\}$ be a countable set of variables, $\mathbb{Q}$ be the set of rationals, and T be the set of terms built out of $V \cup \mathbb{Q}$ and binary function symbols + and − only. Then, the following 6-tuple $\langle \mathcal{L}_c, C, \mathcal{R}, \mathcal{CT}, \mathcal{A}, \overline{\ } \rangle$ is a CABA framework, call it FA:
- $\mathcal{L}_c = \{p(t), s(t), a(t_1, t_2), b(t), ca(t_1, t_2), cb(t) \mid t, t_1, t_2 \in T\} \cup C$
- $C = \{t_1 < t_2,\ t_1 \leq t_2,\ t_1 = t_2,\ t_1 \neq t_2,\ t_1 \geq t_2,\ t_1 > t_2 \mid t_1, t_2 \in T\}$;
- $\mathcal{R} = \{R1, R2, R3, R4, R5\}$, where:

  $R1.\ p(X) \leftarrow X < 1,\ a(X, Y),\ b(X),\ s(Y)$

  $R2.\ p(X) \leftarrow a(Y, X),\ cb(Y)$ $\qquad$ $R3.\ s(Y) \leftarrow Y > 0$

  $R4.\ ca(X, Y) \leftarrow X < 5,\ Y > 3$ $\qquad$ $R5.\ cb(Y) \leftarrow Y < 10$
- $\mathcal{CT}$ is the theory of linear rational arithmetic (LRA);
- $\mathcal{A} = \{a(t_1, t_2) \mid t_1, t_2 \in T\} \cup \{b(t) \mid t \in T\}$;
- for all $t, t_1, t_2 \in T$, $\overline{a(t_1, t_2)} = ca(t_1, t_2)$ and $\overline{b(t)} = cb(t)$.

With abuse of notation, we will write as $s_0 \leftarrow C, s_1, \ldots, s_m$ any rule of the form $s_0 \leftarrow c_1, \ldots, c_k, s_1, \ldots, s_m$, where $k \geq 0$, $m \geq 0$, $C = \{c_1, \ldots, c_k\} \subseteq C$, and $\{s_1, \ldots, s_m\} \subseteq \mathcal{L}_c \setminus C$. Without loss of generality, we assume that every rule in $\mathcal{R}$ is written in the *normalised form* $p(\mathrm{X}_0) \leftarrow C, p_1(\mathrm{X}_1), \ldots, p_m(\mathrm{X}_m)$ where each tuple X_i is made of distinct variables, at the price of adding equality constraints. For example, we write $p(X, X, 4+1) \leftarrow X < 3,\ a(7)$ as $p(X, Y, Z) \leftarrow X = Y,\ Z = 4+1,\ X < 3,\ U = 7,\ a(U)$.

4.1 Relating CABA and standard ABA

Theorem 4.4 below shows how to view (flat) CABA frameworks as standard (flat) ABA frameworks. Indeed, we can define a grounding procedure, called *Ground*, for transforming any CABA framework F_c (see Definition 4.1) into an equivalent ABA framework.

Definition 4.3. $Ground(F_c)$ is the 4-tuple $\langle \mathcal{L}_{cg}, \mathcal{R}_g, \mathcal{A}_g, \overline{\ }^g \rangle$ where ϑ is a grounding substitution:
- $\mathcal{L}_{cg} = \{s \in \mathcal{L}_c \mid s \text{ is } ground\}$;
- $\mathcal{R}_g = \{R\vartheta \in \mathcal{L}_{cg} \mid R \in \mathcal{R}\} \cup \{c \leftarrow \mid c \in \mathcal{L}_{cg} \cap \mathcal{C},\ \mathcal{CT} \models c\}$;
- $\mathcal{A}_g = \{a \in \mathcal{A} \mid a \text{ is } ground\}$;
- $\overline{a}^g = \overline{a}$, for $a \in \mathcal{A}_g$

We denote by Arg_c the set of all arguments in $Ground(F_c)$.

Theorem 4.4. *$Ground(F_c)$ is an ABA framework.*

This result enables us to use the ABA semantics to reason about CABA frameworks by translating their components into the ground counterparts. However, this may require an expensive, possibly infinite, grounding. *Constrained arguments*, given next, may avoid this.

Trivially, the converse of Theorem 4.4 holds, in that every standard ABA framework $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{\ } \rangle$ where $\mathcal{L}$ and $\mathcal{A}$ are predicate closed can be seen as the CABA framework $\langle \mathcal{L}, \emptyset, \mathcal{R}, \emptyset, \mathcal{A}, \overline{\ } \rangle$. Thus, CABA frameworks are a conservative generalisation of ABA frameworks. We will refer to these CABA frameworks, corresponding to ABA frameworks, as *ABA-as-CABA frameworks*.

5 CONSTRAINED ARGUMENTS IN CABA

In this section, we introduce the notion of a *constrained argument* in a CABA framework F_c. Within that general notion, we identify the special case of a *most general constrained argument*. During the construction of arguments, we use rules in normalized form that have been renamed apart (that is, renamed using fresh new variables), thus preventing undesirable clashes of names.

Definition 5.1. Let $C \subseteq \mathcal{C}$ be a *consistent* set of constraints, $A \subseteq \mathcal{A}$ a set of assumptions, and $R \subseteq \mathcal{R}$ a set of rules.
- A *tight constrained argument* for claim $s \in \mathcal{L}_c \setminus \mathcal{C}$ *supported by* $C \cup A$ *and* R, denoted $C \cup A \vdash_R s$, is a finite tree such that: (i) the root is s, (ii) every non-leaf node $s' = p(\mathrm{t})$ has as children all and only the instantiated atoms $s_1\vartheta, \ldots, s_m\vartheta$ exactly one renamed apart rule Ri: $p(\mathrm{X}) \leftarrow s_1, \ldots, s_m$ in R with $\vartheta = \{\mathrm{X}/\mathrm{t}\}$, or, if Ri is a fact, s' has the only child *true*, and (iii) every leaf is either a constraint in C or an assumption in A or *true*. C and A are the sets of all constraints and assumptions, respectively, occurring in the tree, and R is the set of all rules used to construct the tree.
- A *most general constrained argument* is a tight constrained argument whose claim is an atom of the form $p(\mathrm{X})$.

We denote by *TCArg* and *MGCArg* the set of all tight and most general constrained arguments, respectively, of F_c.

From Definition 5.1, it follows that $MGCArg \subseteq TCArg$. Actually, we have that $MGCArg \subset TCArg$, as some tight constrained arguments are not most general, as illustrated next.

Example 5.2. Let us consider the CABA framework *FA* of Example 4.2. The following are tight constrained arguments: (i) $\{Y < 10, a(Y,0)\} \vdash_{\{R2,R5\}} p(0)$, (ii) $\{0 < 1, Y > 0, a(0,Y), b(0)\} \vdash_{\{R1,R3\}} p(0)$, (iii) $\{4 > 0\} \vdash_{\{R3\}} s(4)$, and (iv) $\{X < 10\} \vdash_{\{R5\}} cb(X)$. Argument (iv) is a most general constrained argument; the others are not.

The following definitions introduce notions of *constrained arguments* and *constrained instances* thereof.

Definition 5.3. $C' \cup A' \vdash_R s'$ is a *constrained argument* in F_c if there exist a *tight* constrained argument $C \cup A \vdash_R s$ in F_c, a substitution ϑ, and a set $D \subseteq \mathcal{C}$ of constraints such that: (i) $C' = (C\vartheta) \cup D$, $A' = A\vartheta$, $s' = s\vartheta$, and (ii) C' is consistent. We denote by *CArg* the set of all constrained arguments of F_c.

From Definitions 5.1 and 5.3, we get that $TCArg \subseteq CArg$. Actually, we have that $TCArg \subset CArg$, as some constrained arguments are not tight (see Example 5.5 below).

Definition 5.4. The constrained argument $\alpha' = C' \cup A' \vdash_R s'$ is a *constrained instance* (*via ϑ and D*) of the constrained argument $\alpha = C \cup A \vdash_R s$ if: (i) $C' = (C\vartheta) \cup D$, $A' = A\vartheta$, $s' = s\vartheta$, and (ii) C' is consistent. We say that α' is an *instance* (*via ϑ*) of α if α' is a constrained instance (via ϑ and D) of α with $D = \emptyset$.

In what follows, as for arguments of ABA frameworks, also for constrained arguments of CABA frameworks we will often omit to specify the set R of rules involved. For instance, we will write $\{Y < 10, a(Y,X)\} \vdash p(X)$ instead of $\{Y < 10, a(Y,X)\} \vdash_{\{R2,R5\}} p(X)$.

Example 5.5. For the CABA framework *FA* of Example 4.2, $\{8 < 10, 8 < 12, a(8,0)\} \vdash p(0)$ is a constrained instance (via $\vartheta = \{Y/8\}$ and $D = \{8 < 12\}$) of the tight constrained argument $\alpha = \{Y < 10, a(Y,0)\} \vdash p(0)$. Also, $\{Y < 10, Y = 7, Y > 5, a(Y,0)\} \vdash p(0)$ is a constrained instance of the non-tight constrained argument $\{Y < 10, Y = 7, a(Y,0)\} \vdash p(0)$ which is, in turn, an instance of α.

Example 5.6. Let M be the set $MGCArg(FA)$ of all most general constrained arguments of the CABA framework *FA* of Example 4.2. M is obtained by considering all variable renamings of the constrained arguments in the set $\widetilde{M} = \{\alpha_1, \ldots, \alpha_7\}$, where:

α_1. $\{X < 1, Y > 0,\ a(X,Y),\ b(X)\} \vdash_{\{R1,R3\}} p(X)$;
α_2. $\{Y < 10, a(Y,X)\} \vdash_{\{R2,R5\}} p(X)$;
α_3. $\{Y > 0\} \vdash_{\{R3\}} s(Y)$;
α_4. $\{X < 5,\ Y > 3\} \vdash_{\{R4\}} ca(X,Y)$;
α_5. $\{X < 10\} \vdash_{\{R5\}} cb(X)$;
α_6. $\{a(X,Y)\} \vdash_{\emptyset} a(X,Y)$;
α_7. $\{b(X)\} \vdash_{\emptyset} b(X)$.

Note that, in particular, the most general constrained argument $\{U < 1, Z > 0, a(U,Z), b(U)\} \vdash_{\{R1,R3\}} p(U)$ is an element of M and it is derived from α_1 by the variable renaming $\{X/U, Y/Z, U/X, Z/Y\}$.

For ABA-as-CABA frameworks, constrained arguments are standard arguments in ABA, and thus, again, CABA is a conservative generalisation of ABA. Next, we explore the relation between CABA and ABA arguments for generic CABA frameworks.

5.1 Mapping Arguments in CABA to ABA

The following proposition establishes that every tight constrained argument can be obtained as an instance of a most general constrained argument. As a consequence, the most general constrained arguments can provide a foundation for reconstructing the entire argumentative structure of a CABA framework.

Proposition 5.7. *$C \cup A \vdash_R p(\mathrm{t}) \in TCArg$ iff there exists $C' \cup A' \vdash_R p(\mathrm{X}) \in MGCArg$ such that for $\vartheta = \{\mathrm{X}/\mathrm{t}\}$, we have that $C = C'\vartheta$, $A = A'\vartheta$, and $C'\vartheta$ is consistent.*

Definition 5.8. For any $\alpha \in CArg$, $GrCInst(\alpha)$ is the set of all constrained instances of α that are ground. For any set $\Gamma \subseteq CArg$ of constrained arguments, $GrCInst(\Gamma)$ is the set $\bigcup_{\alpha \in \Gamma} GrCInst(\alpha)$.

The following corollary relates the sets of ground instances of constrained arguments and most general constrained arguments.

Corollary 5.9. *GrCInst*(*CArg*) = *GrCInst*(*MGCArg*).

The following definition is based on the fact that, for any ground constrained argument $\{c_1,\ldots,c_m\}\cup A\vdash s$ (with $m\geq 0$), we have that $\mathcal{CT}\models(c_1\wedge\ldots\wedge c_m)$. By abuse of notation, for all ground atoms $p(\mathrm{t})$, $p(\mathrm{t}')$ in $\mathcal{L}_c$, we write $\mathcal{CT}\models p(\mathrm{t})\leftrightarrow p(\mathrm{t}')$ iff $\mathcal{CT}\models \mathrm{t}=\mathrm{t}'$. Given two ground conjunctions C_1 and C_2 of atoms, we write $\mathcal{CT}\models C_1\leftrightarrow C_2$ iff (i) $\forall a_1\in C_1$, $\exists a_2\in C_2$ such that $\mathcal{CT}\models a_1\leftrightarrow a_2$, and (ii) the same as (i) by interchanging C_1 and C_2.

Definition 5.10. The ground constrained arguments $\alpha=C\cup A\vdash s$ and $\alpha'=C'\cup A'\vdash s'$ are *equal* (written $\alpha=\alpha'$) *modulo ground constraints* if (i) $\mathcal{CT}\models\bigwedge\{a\mid a\in A\}\leftrightarrow\bigwedge\{a\mid a\in A'\}$, and (ii) $\mathcal{CT}\models s\leftrightarrow s'$.

For instance, $\{0<10, a(0,2+1), a(0,1+2)\}\vdash p(4-1)=\{a(0,3)\}\vdash p(3)$ modulo ground constraints. In the sequel, a ground constrained argument $C\cup A\vdash s$ will also be written as $A'\vdash s'$, where A' and s' satisfy Conditions (i) and (ii), respectively, of Definition 5.10. Indeed, the elimination of consistent, ground constraints preserves equality modulo ground constraints.

Example 5.11. Let M be the set *MGCArg*(*FA*) of the most general constrained arguments of *FA* of Example 4.2. The set *GrCInst*(M) of the ground constrained instances of M is the following, modulo ground constraints. The subscripts α_i's relate the associated sets to the arguments of $\widetilde{M}$ shown in Example 5.6. We assume that $n, m\in\mathbb{Q}$.

$$\begin{array}{lll} \mathit{GrCInst}(M)=\{\{a(n,m),b(n)\}\vdash p(n)\mid n<1,\ m>0\}_{\alpha_1} & \cup & \\ \{\{a(m,n)\}\vdash p(n)\mid m<10\}_{\alpha_2} & \cup & \{\{\}\vdash s(n)\mid n>0\}_{\alpha_3} \quad \cup \\ \{\{\}\vdash ca(n,m)\mid n<5,\ m>3\}_{\alpha_4} & \cup & \{\{\}\vdash cb(n)\mid n<10\}_{\alpha_5} \quad \cup \\ \{\{a(n,m)\}\vdash a(n,m)\}_{\alpha_6} & \cup & \{\{b(n)\}\vdash b(n)\}_{\alpha_7}. \end{array}$$

The following result establishes a correspondence between the arguments in the CABA framework F_c and those in its ground ABA counterpart $\mathit{Ground}(F_c)$ (see Definition 4.3).

Theorem 5.12. *GrCInst*(*CArg*) = Arg_c *modulo ground constraints.*

We can lift the correspondence stated by this theorem to sets of arguments (and extensions, as defined below), by first introducing the following equivalence relation which extends the equivalence based on equality modulo ground constraints.

Definition 5.13. The *equivalence relation* $\equiv$ over pairs of subsets of *CArg* is the smallest equivalence relation satisfying the following:

(1) *Ground Constraints.* For any two ground constrained arguments α_1 and α_2, if $\alpha_1=\alpha_2$ modulo ground constraints, then $\{\alpha_1\}\equiv\{\alpha_2\}$.
(2) *Ground Instances.* For every $\alpha\in\mathit{CArg}$, $\{\alpha\}\equiv\mathit{GrCInst}(\alpha)$.
(3) *Congruence for Union.* For all sets $\Gamma,\Delta_1,\Delta_2\subseteq\mathit{CArg}$, if $\Delta_1\equiv\Delta_2$, then $(\Gamma\cup\Delta_1)\equiv(\Gamma\cup\Delta_2)$.

The following theorem establishes a key correspondence between the equivalence of sets of constrained arguments and the equality of the sets of their ground instances.

Theorem 5.14. *For every* $\Gamma,\Delta\subseteq\mathit{CArg}$, $\Gamma\equiv\Delta$ *iff* *GrCInst*(Γ) = *GrCInst*(Δ) *modulo ground constraints.*

From Theorem 5.12 and Theorem 5.14, we get the following.

Corollary 5.15. *Let* $\Gamma,\Delta\subseteq\mathit{CArg}$ *and let* $\Gamma_c,\Delta_c\subseteq\mathit{Arg}_c$ *be the sets of arguments in* $\mathit{Ground}(F_c)$ *(see Definition 4.3) which correspond to* *GrCInst*(Γ) *and* *GrCInst*(Δ)*, respectively, as established by Theorem 5.12. Then* $\Gamma\equiv\Delta$ *iff* $\Gamma_c=\Delta_c$ *modulo ground constraints.*

As a consequence of Corollaries 5.9 and 5.15, for any $\Gamma\subseteq\mathit{CArg}$, we may use *GrCInst*($\Gamma$) to refer both to: (i) the set of ground constrained instances of Γ, and (ii) the corresponding set of arguments in $\mathit{Ground}(F_c)$.

Now we present some properties of the equivalence relation $\equiv$.

Proposition 5.16. (1) Renaming. *For any constrained argument* α *and variable renaming* ρ, $\{\alpha\}\equiv\{\alpha\rho\}$.

(2) Generalization of Claims and Assumptions. *Let* $\alpha=C\cup A\vdash p(\mathrm{t})$ *be a constrained argument and let* X *be a tuple of new distinct variables. Then* $\{\alpha\}\equiv\{C\cup\{\mathrm{X}=\mathrm{t}\}\cup A\vdash p(\mathrm{X})\}$. *Similarly,* $\{C\cup A\cup\{p(\mathrm{t})\}\vdash s\}\equiv\{C\cup\{\mathrm{X}=\mathrm{t}\}\cup A\cup\{p(\mathrm{X})\}\vdash s\}$.

(3) Constraint Equivalence. *Let* C, D_1, *and* D_2 *be finite sets of consistent constraints. If* $\mathcal{CT}\models\forall\big(\exists_{-V}(\bigwedge\{c\mid c\in C\})\leftrightarrow(\exists_{-V}(\bigwedge\{c\mid c\in D_1\})\vee\exists_{-V}(\bigwedge\{c\mid c\in D_2\}))\big)$, *where* $V=\mathit{vars}(\{A,s\})$, *then* $\{C\cup A\vdash s\}\equiv\{D_1\cup A\vdash s, D_2\cup A\vdash s\}$.

By Point (3) of Proposition 5.16, for $D_1=D_2=D$ we have that, for all finite sets C and D of constraints, if $\mathcal{CT}\models\forall\big(\bigwedge\{c\mid c\in C\}\leftrightarrow\bigwedge\{c\mid c\in D\}\big)$, then $\{C\cup A\vdash s\}\equiv\{D\cup A\vdash s\}$.

In Examples 5.6 and 5.11, (i) by Points (1) and (2) of Definition 5.13, we have that $\{\alpha_1\}\equiv\{\{a(n,m),b(n)\}\vdash p(n)\mid n<1, m>0\}$, and (ii) by Definition 5.13, we have that $M\equiv\widetilde{M}\equiv\mathit{GrCInst}(M)$.

Example 5.17 (Ex. 5.6 contd.). Let $\widetilde{N}$ be the set $\widetilde{M}$ where α_2 has been replaced by:

$\alpha_{2,1}$: $\{Y<10,\ Y\geq 5,\ a(Y,X)\}\ \vdash\ p(X)$;
$\alpha_{2,2,1}$: $\{Y<10,\ Y<5,\ X>3,\ a(Y,X)\}\ \vdash\ p(X)$;
$\alpha_{2,2,2}$: $\{Y<10,\ Y<5,\ X\leq 3,\ a(Y,X)\}\ \vdash\ p(X)$

Now, by Point (3) of Proposition 5.16 with $V=\{X,Y\}$, and Point (3) of Definition 5.13, from $\mathcal{CT}\models\forall(Y\geq 5\vee Y<5)$ and $\mathcal{CT}\models\forall(X>3\vee X\leq 3)$, we get $\widetilde{M}\equiv\widetilde{N}$. Thus, we also get $M\equiv\widetilde{N}$.

Overall, these results enable the use of ABA semantics to reason about CABA frameworks by translating constrained arguments into standard ABA arguments. However, again, this may require an expensive, possibly infinite, grounding. Attacks between constrained arguments, given next, may avoid this.

6 ATTACKS IN CABA

In this section we introduce attacks between constrained arguments. We distinguish between *full* and *partial attacks*, defined in terms of different logical relations between the constraints involved.

Definition 6.1. Let $\alpha=\{c_1,\ldots,c_m\}\cup A_1\vdash s_1$ and $\beta=\{d_1,\ldots,d_n\}\cup A_2\cup\{a\}\vdash s_2$ be constrained arguments, with $s_1=\overline{a}$. Then,

- α *fully attacks* β if $\mathcal{CT}\models\forall\,((d_1\wedge\ldots\wedge d_n)\rightarrow\exists_{-\mathit{vars}(s_1)}(c_1\wedge\ldots\wedge c_m))$, and
- α *partially attacks* β if $\mathcal{CT}\models\exists\,(\exists_{-\mathit{vars}(s_1)}(c_1\wedge\ldots\wedge c_m)\wedge\exists_{-\mathit{vars}(s_1)}(d_1\wedge\ldots\wedge d_n)))$.

Moreover, let $\alpha'=\{c_1,\ldots,c_m\}\cup A_1\vdash\overline{p(\mathrm{t})}$ and $\beta'=\{d_1,\ldots,d_n\}\cup A_2\cup\{p(\mathrm{u})\}\vdash s_2$ with $\mathrm{t}\neq\mathrm{u}$. Let X be a tuple of new distinct variables. Then, α' *fully* (or *partially*) *attacks* β' if $\{c_1,\ldots,c_m,\mathrm{X}=\mathrm{t}\}\cup A_1\vdash\overline{p(\mathrm{X})}$ *fully* (or *partially*, resp.) *attacks* $\{d_1,\ldots,d_n,\mathrm{X}=\mathrm{u}\}\cup A_2\cup\{p(\mathrm{X})\}\vdash s_2$.

Example 6.2. $\{X>0,\ Y<2,\ q(X,Y)\}\vdash\overline{p(X)}$ fully attacks $\{X>10,\ Z<3,\ p(X)\}\vdash r(Z)$, because $\exists_{-\{X\}}(X>0\wedge Y<2)$ is $\exists Y(X>0\wedge Y<2)$ and $\mathcal{CT}\models\forall X,Z\,\big((X>10\wedge Z<3)\rightarrow\exists Y(X>0\wedge Y<2)\big)$.

In the sequel we use the following property of partial attacks.

Proposition 6.3. *Let α, β be constrained arguments. Then, α partially attacks β iff there exist $\alpha' = C \cup A_1 \vdash s_1$ and $\beta' = D \cup A_2 \vdash s_2$ such that: (i) $\{\alpha\} \equiv \{\alpha'\}$, (ii) $\{\beta\} \equiv \{\beta'\}$, (iii) $s_1 = \overline{a}$, for some $a \in A_2$, (iv) $vars(C) \cap vars(D) \subseteq vars(s_1)$, and (v) $C \cup D$ is consistent.*

Example 6.4. Referring to Example 5.6, α_5 fully attacks α_1 as $\mathcal{CT} \models \forall X, Y\,((X<1 \wedge Y>0) \rightarrow X<10)$. Instead, α_4 does not fully attack α_1 as $\mathcal{CT} \not\models \forall X, Y((X<1 \wedge Y>0) \rightarrow (X<5 \wedge Y>3))$. However, α_4 partially attacks α_2 as $\{X<5,\ Y>3,\ X_1=X,\ X_2=Y\} \vdash ca(X_1, X_2)$ partially attacks $\{Y<10, X_1=Y, X_2=X,\ a(X_1, X_2)\} \vdash p(X)$. Indeed, $\mathcal{CT} \models \exists X_1, X_2\ (X_1<10 \wedge X_1<5 \wedge X_2>3)$. By Proposition 6.3, α_4 partially attacks $\{X<10, a(X, Y)\} \vdash p(Y)$ (a renaming of α_2).

The following result relates our different notions of attack.

Proposition 6.5. *Let α and $\beta \in CArg$. If α fully attacks β, then α partially attacks β.*

For ABA-as-CABA frameworks, partial and full attacks coincide and are standard attacks in ABA, again pointing to CABA being a conservative generalisation of ABA. Next, we explore the relation between CABA and ABA attacks for generic CABA frameworks.

6.1 Mapping Attacks in CABA to ABA

The following result establishes a correspondence between full and partial attacks in CABA and attacks in standard ABA [27].

Theorem 6.6. *For $\alpha, \beta \in CArg$, $\Gamma = GrCInst(\alpha)$, and $\Delta = GrCInst(\beta)$: (i) α fully attacks β iff for each $\beta' \in \Delta$ there exists $\alpha' \in \Gamma$ such that α' attacks β' in $Ground(F_c)$, and (ii) α partially attacks β iff there exist $\alpha' \in \Gamma$ and $\beta' \in \Delta$ such that α' attacks β' in $Ground(F_c)$.*

Note that, when the notion of attack is considered for ground instances of constrained arguments in CABA frameworks, the distinction between full and partial attack collapses. In such a case, we refer to them generically as 'attack'.

7 EXTENSION-BASED SEMANTICS FOR CABA

In this section we explore two ways to equip CABA with a semantics. First, we leverage on the mapping, by grounding, of arguments and attacks in CABA onto corresponding standard ABA notions (Section 7.1). Then, we define novel notions of extensions for CABA, without resorting to up-front mapping to standard ABA (Section 7.2). Throughout, we focus on conflict-free, admissible, and stable extension semantics.

7.1 CABA Semantics via ABA Semantics

We can define the extensions of a CABA framework F_c as extensions of the ABA framework $Ground(F_c)$.

Definition 7.1. A set $\Gamma \subseteq CArg$ of constrained arguments is a conflict-free, or admissible, or stable extension in F_c if $GrCInst(\Gamma)$ is a conflict-free, or admissible, or stable extension, resp., $Ground(F_c)$.

In order to determine the extensions in $Ground(F_c)$, we can leverage on the mapping to AA, as for standard ABA [27].

Let $GrCInst(CArg)$ be as introduced in Definition 5.8 (which, by Theorem 5.12, is equal to Arg_c from Definition 4.3 modulo ground constraints). Let $Att = \{(\alpha, \beta) \in GrCInst(CArg) \times GrCInst(CArg) \mid \alpha$ attacks $\beta\}$ be the set of all attacks in standard ABA (see Section 3). Then, $\Gamma \subseteq CArg$ is a conflict-free (or admissible, or stable) extension of F_c iff Γ is a conflict-free (or admissible, or stable, resp.) extension of $(GrCInst(CArg), Att)$.

Example 7.2. Let $GrCInst(M)$ be as in Example 5.11. The AA framework is $(GrCInst(M), Att)$, where the set Att is as follows, modulo ground constraints, with $n, m \in \mathbb{Q}$:

$\{\ (\{\} \vdash ca(n,m),\ \{a(n,m), b(n)\} \vdash p(n)) \mid n<1,\ m>3\ \} \cup$
$\{\ (\{\} \vdash cb(n),\ \{a(n,m), b(n)\} \vdash p(n)) \mid n<1,\ m>0\ \} \cup$
$\{\ (\{\} \vdash ca(n,m),\ \{a(n,m)\} \vdash p(m)) \mid n<5,\ m>3\ \} \cup$
$\{\ (\{\} \vdash ca(n,m),\ \{a(n,m)\} \vdash a(n,m)) \mid n<5,\ m>3\ \} \cup$
$\{\ (\{\} \vdash cb(n),\ \{b(n)\} \vdash b(n)) \mid n<10\ \}$.

Then, let us consider the following set $\Delta \subseteq GrCInst(M)$, modulo ground constraints, where $n, m \in \mathbb{Q}$:

$\Delta = \{\{a(n,m)\} \vdash p(m) \mid n \geq 5,\ n<10\} \cup$
$\{\{a(n,m)\} \vdash p(m) \mid m \leq 3,\ n<5\} \cup \{\{\} \vdash s(n) \mid n>0\} \cup$
$\{\{\} \vdash ca(n,m) \mid n<5,\ m>3\} \cup \{\{\} \vdash cb(n) \mid n<10\} \cup$
$\{\{a(n,m)\} \vdash a(n,m) \mid n \geq 5\} \cup \{\{a(n,m)\} \vdash a(n,m) \mid m \leq 3\} \cup$
$\{\{b(n)\} \vdash b(n) \mid n \geq 10\}$.

We have that Δ is conflict-free, admissible, and the unique stable extension of $(GrCInst(M), Att)$. Note that every ground argument in $\{\{a(n,m), b(n)\} \vdash p(n) \mid n<1, m>0\}$ is attacked by an argument in $\{\{\} \vdash cb(n) \mid n<10\}$ $(\subseteq \Delta)$ (see Example 5.11).

Now, let us consider the set $\Gamma \subseteq CArg$ defined as follows:

$\Gamma = \{\ (\alpha_{2,1})\ \{Y \geq 5,\ Y<10,\ a(Y,X)\} \vdash p(X),$
$(\alpha_{2,2,2})\ \{X \leq 3,\ Y<5,\ a(Y,X)\} \vdash p(X),\quad (\alpha_3)\ \{Y>0\} \vdash s(Y),$
$(\alpha_4)\ \ \{X<5,\ Y>3\} \vdash ca(X,Y),\quad (\alpha_5)\ \ \{X<10\} \vdash cb(X),$
$(\alpha_{6,1})\ \{X \geq 5, a(X,Y)\} \vdash a(X,Y),\ \ (\alpha_{6,2})\ \{Y \leq 3, a(X,Y)\} \vdash a(X,Y),$
$(\alpha_{7,1})\ \{X \geq 10,\ b(X)\} \vdash b(X)\ \}$.

The labels (α_i)'s relate the associated constrained arguments to the elements of $\widetilde{M}$ and $\widetilde{N}$, if any (see Examples 5.6 and 5.17).

Trivially, $GrCInst(\Gamma) = \Delta$ modulo ground constraints. Then, by Definition 7.1, Γ is a conflict-free, admissible, and stable extension of the CABA framework *FA* of Example 4.2. This extension is the unique stable one, modulo $\equiv$.

Note that, for ABA-as-CABA frameworks the notions of conflict-free, admissible, and stable extensions from Definition 7.1 are exactly as in standard ABA, allowing us to conclude that CABA is a conservative generalisation of ABA.

7.2 Native CABA Semantics

Now, we naturally characterise conflict-free and stable extensions directly in terms of the notions of full and partial attacks in CABA.

Definition 7.3. A set $\Sigma \subseteq CArg$ is *non-ground conflict-free* (*NGCF*) iff there are no $\alpha, \beta \in \Sigma$ such that α partially attacks β. By $FAtt(\Sigma)$ we denote the set $\{\beta \in CArg \mid \exists \alpha \in \Sigma$ such that α fully attacks $\beta\}$.

Theorem 7.4. *For any $\Sigma \subseteq CArg$ we have that:*
(1) Σ is conflict-free iff Σ is NGCF; and
(2) Σ is stable iff Σ is NGCF and $\Sigma \cup FAtt(\Sigma) \equiv CArg$.

Example 7.5. For the CABA framework of Example 4.2, we have that the set Γ of arguments (see Example 7.2) is conflict-free. Moreover, α_1, $\alpha_{2,2,1}$, $\alpha_{6,3}$: $\{X<5, Y>3, a(X,Y)\} \vdash a(X,Y)$, and $\alpha_{7,2}$: $\{X<10,\ b(X)\} \vdash b(X)$ belong to $FAtt(\Gamma)$. Indeed, $\alpha_1, \alpha_{2,2,1}, \alpha_{6,3}$, and $\alpha_{7,2}$ are fully attacked by α_5, α_4, α_4, and α_5, respectively. Since $\{\alpha_2\} \equiv \{\alpha_{2,1}, \alpha_{2,2,1}, \alpha_{2,2,2}\}$, $\{\alpha_6\} \equiv \{\alpha_{6,1}, \alpha_{6,2}, \alpha_{6,3}\}$, and $\{\alpha_7\} \equiv \{\alpha_{7,1}, \alpha_{7,2}\}$, we have that $\Gamma \cup FAtt(\Gamma) \equiv CArg$. Thus, Γ is a stable extension. Actually, Γ is the unique stable extension, modulo $\equiv$.

However, the characterisation of stable extensions for CABA by Theorem 7.4 still refers to the set $CArg$ of all (ground and non-ground) constrained arguments. Moreover, it neglects admissible

extensions, as these cannot be naturally characterised in terms of NGCF and *FAtt*. Now, we propose a characterisation, not requiring ground instances, and including admissible extensions.

We start off by introducing the following notion.

Definition 7.6. Two constrained arguments α and β are said to have *common constrained instances* if $GrCInst(\alpha) \cap GrCInst(\beta) \neq \emptyset$.

Example 7.7. $\{X{>}3, Y{<}0, a(X)\} \vdash p(X)$ and $\{Z{>}0, a(Z)\} \vdash p(Z)$ have the common constrained instance $\{X{>}3, Y{<}0, X{>}0, a(X)\} \vdash p(X)$.

The following result allows us to check whether or not constrained arguments have common constrained instances.

Proposition 7.8. *Two constrained arguments α and β have common constrained instances iff there exist $\alpha' = C' \cup A' \vdash s$ and $\beta' = D' \cup A' \vdash s$ such that: (i) $\{\alpha\} \equiv \{\alpha'\}$, (ii) $\{\beta\} \equiv \{\beta'\}$, (iii) $vars(C') \cap vars(D') \subseteq vars(\{A', s\})$, and (iv) $C' \cup D'$ is consistent.*

Then, we can apply the notion of absence of common constrained instances to characterise *instance-disjoint* sets. We can also introduce a way to compare instances of constrained arguments based on attacks between them, by assessing whether or not they are *non-overlapping* in that every partial attack is a full attack. This notion will help identify non-ground instances of most general constrained arguments to choose extensions from.

Definition 7.9. Let $\Delta \subseteq CArg$. (i) Δ is *instance-disjoint* if $\forall \alpha, \beta \in \Delta$, α and β have no common constrained instances; (ii) Δ is *non-overlapping* if $\forall \alpha, \beta \in \Delta$, α partially attacks β iff α fully attacks β.

Theorem 7.10. *Let $\Delta \subseteq CArg$ be non-overlapping, and $\Sigma \subseteq \Delta \equiv CArg$.*
(1) *Σ is conflict-free iff $\nexists \alpha, \beta \in \Sigma$, such that α fully attacks β;*
(2) *Σ is admissible iff Σ is conflict-free and $\forall \alpha \in \Sigma$, if $\exists \beta \in \Delta$ such that β fully attacks α, then $\exists \gamma \in \Sigma$ such that γ fully attacks β;*
(3) (3.1) *Σ is stable if Σ is conflict-free and $\forall \beta \in \Delta \setminus \Sigma$, $\exists \alpha \in \Sigma$ such that α fully attacks β.*
(3.2) *If Δ is instance-disjoint, then the converse of (3.1) holds.*

Theorem 7.10 provides native characterisations of CABA semantics in terms of full attacks. Points (1) and (3) give a native characterisation of conflict-free and stable extensions not requiring ground instances (unlike Theorem 7.4). Point (2) gives a native characterisation of admissible extensions. The following example shows the need of the instance-disjoint condition on Δ in Point (3.2).

Example 7.11. Let $\Delta = \{\alpha_1{:}\{X>0\} \vdash p(X), \alpha_2{:}\{X>3\} \vdash p(X)\} \equiv CArg$. Δ is non-overlapping, but not instance-disjoint. By Definition 7.1, $\Sigma = \{\alpha_1\}$ is stable. However, α_1 does not attack α_2.

Finally, we present a procedure, called *Argument Splitting*, that, given a set of constrained arguments, constructs an equivalent instance-disjoint, non-overlapping set. The procedure requires the constraint theory $\mathcal{CT}$ to satisfies the conditions we now specify.

Definition 7.12. $\mathcal{CT}$ is *closed under negation* if, for every $c \in \mathcal{C}$, there exist $d_1, \ldots, d_m \in \mathcal{C}$ such that $\mathcal{CT} \models \forall(\neg c \leftrightarrow (d_1 \vee \ldots \vee d_m))$, and, for $i, j = 1, \ldots, m$ with $i \neq j$, $\mathcal{CT} \models \neg\exists(d_i \wedge d_j)$. $\mathcal{CT}$ is *closed under existential quantification* if, for every $\{c_1, \ldots, c_m\} \subseteq \mathcal{C}$ and set V of variables, there exist $d_1, \ldots, d_i, \ldots, d_j, \ldots, d_n \in \mathcal{C}$ such that $\mathcal{CT} \models \forall(\exists_{-V}(c_1 \wedge \ldots \wedge c_m) \leftrightarrow ((d_1 \wedge \ldots \wedge d_i) \vee \ldots \vee (d_j \wedge \ldots \wedge d_n)))$.

When $\mathcal{CT}$ is closed under negation and existential quantification, we can define two splitting operations: (1) $split_{ci}$, which is applied to pairs of constrained arguments with common constrained instances, and (2) $split_{pa}$, which is applied to pairs of constrained arguments such that one partially attacks, and does not fully attack, the other. Now we need the following notion.

Definition 7.13. Let $C = \{c_1, \ldots, c_m\}$ and $D = \{d_1, \ldots, d_n\}$ be sets of constraints. By *constraint split* of C and D, we denote a set $cs(C, D) = \{E_1, \ldots, E_r\}$ of sets of constraints where, for $i = 1, \ldots, r$, e_i is the conjunction of the constraints in E_i, such that: (1) $\mathcal{CT} \models \forall((\neg\exists_{-V}(c_1 \wedge \ldots \wedge c_m) \wedge d_1 \wedge \ldots \wedge d_n) \leftrightarrow (e_1 \vee \ldots \vee e_r))$, with $V = vars(C) \cap vars(D)$; (2) $\mathcal{CT} \models \exists(e_i)$, i.e., e_i is consistent; (3) for $j = 1, \ldots, r$ with $i \neq j$, $\mathcal{CT} \models \neg\exists(e_i \wedge e_j)$, i.e., e_i and e_j are *mutually exclusive*.

The next proposition shows that, if $\mathcal{CT}$ is closed under negation and existential quantification, constraint split is indeed defined for all pairs of finite sets of constraints.

Proposition 7.14. *If $\mathcal{CT}$ is closed under negation and existential quantification, then for all finite sets $C, D \subseteq \mathcal{C}$ such that $C \cup D$ is consistent, there exist $E_1, \ldots, E_r$ such that $cs(C, D) = \{E_1, \ldots, E_r\}$.*

Now, based on constraint split, we define $split_{ci}$ and $split_{pa}$.

Definition 7.15. Let $\{\alpha\} \equiv \{C \cup A \vdash s\}$ and $\{\beta\} \equiv \{D \cup A \vdash s\}$ such that: (i) $vars(C) \cap vars(D) \subseteq vars(\{A, s\})$, and (ii) $C \cup D$ is consistent. Assume that α and β have common constrained instances. Then, $split_{ci}(\alpha, \beta) = \{\beta_1, \ldots, \beta_r\}$, where $cs(C, D) = \{E_1, \ldots, E_r\}$ and, for $i = 1, \ldots, r$, $\beta_i = E_i \cup A \vdash s$.

Due to Proposition 7.8, in $split_{ci}(\alpha, \beta)$ it is not restrictive to require that $\{\alpha\} \equiv \{C \cup A \vdash s\}$ and $\{\beta\} \equiv \{D \cup A \vdash s\}$. $split_{ci}$ is well-defined, as, for $i = 1, \ldots, r$, E_i is a consistent set of constraints.

Definition 7.16. Let $\{\alpha\} \equiv \{C \cup A_1 \vdash s_1\}$ and $\{\beta\} \equiv \{D \cup A_2 \vdash s_2\}$ such that: (i) $s_1 = \overline{a}$, for some $a \in A_2$, (ii) $vars(C) \cap vars(D) \subseteq vars(s_1)$, and (iii) $C \cup D$ is consistent. Assume that α partially attacks β. Then, $split_{pa}(\alpha, \beta) = \{\beta_0, \beta_1, \ldots, \beta_r\}$, where (1) $E_0 = C \cup D$, (2) $cs(C, D) = \{E_1, \ldots, E_r\}$ and, (3) for $i = 0, \ldots, r$, $\beta_i = E_i \cup A_2 \vdash s_2$.

Similarly to the case of $split_{ci}$, due to Proposition 6.3, in the above definition it is not restrictive to assume that $\{\alpha\} \equiv \{C \cup A_1 \vdash s_1\}$ and $\{\beta\} \equiv \{D \cup A_2 \vdash s_2\}$. $split_{pa}$ is well-defined, as for $i = 0, \ldots, r$, E_i is a consistent set of constraints. It can also be seen that the outputs of the two splitting operations are defined up to equivalence. That is, if in Definition 7.15 we consider $\{\alpha\} \equiv \{C' \cup A' \vdash s'\}$ and $\{\beta\} \equiv \{D' \cup A' \vdash s'\}$ such that Conditions (i)-(ii) hold for C', A', s', D' instead of C, A, s, D, then $split_{ci}(\alpha, \beta) = \{\beta'_1, \ldots, \beta'_r\}$ with $\{\beta_1\} \equiv \{\beta'_1\}, \ldots, \{\beta_r\} \equiv \{\beta'_r\}$. A similar property holds for $split_{pa}$.

Now we show that the two operations $split_{ci}$ and $split_{pa}$ preserve equivalence.

Proposition 7.17. *Let $\mathcal{CT}$ be closed under negation and existential quantification and let $\alpha, \beta \in \Delta \subseteq CArg$.*
(1) *Let $split_{ci}(\alpha, \beta) = \{\beta_1, \ldots, \beta_r\}$. Then, (i) there is no pair in $\{\alpha, \beta_1, \ldots, \beta_r\}$ with common constrained instances, and (ii) $(\Delta \setminus \{\beta\}) \cup split_{ci}(\alpha, \beta) \equiv \Delta$.*
(2) *Let $split_{pa}(\alpha, \beta) = \{\beta_0, \beta_1, \ldots, \beta_r\}$. Then, (i) α fully attacks β_0, (ii) α does not partially attack any argument in $\{\beta_1, \ldots, \beta_r\}$, and (iii) $(\Delta \setminus \{\beta\}) \cup split_{pa}(\alpha, \beta) \equiv \Delta$.*

Example 7.18 (Ex. 7.11 contd.). Δ is transformed into the equivalent, instance-disjoint set $\Delta \setminus \{\alpha_2\}$, as $split_{ci}(\alpha_2, \alpha_1) = \{\}$.

Example 7.19. Let us consider the CABA framework with rules:

$cp(X) \leftarrow r(X),\ q(X)$ $\quad$ $cq(X) \leftarrow s(X),\ p(X)$
$r(X) \leftarrow X \geq Y, Y \geq 0$ $\quad$ $s(X) \leftarrow X \leq 0$

where $q(X), p(X)$ are assumptions with contraries $cq(X), cp(X)$, respectively. The most general arguments are:

α_1: $\{X \geq Y,\ Y \geq 0,\ q(X)\} \vdash cp(X)$ $\quad$ α_2: $\{X \leq 0,\ p(X)\} \vdash cq(X)$
α_3: $\{X \geq Y,\ Y \geq 0\} \vdash r(X)$ $\quad$ α_4: $\{X \leq 0\} \vdash s(X)$
α_5: $\{p(X)\} \vdash p(X)$ $\quad$ α_6: $\{q(X)\} \vdash q(X)$.

This set of arguments is instance-disjoint, but not non-overlapping: α_1 and α_2 partially attack each other, and they also partially attack α_5 and α_6, respectively. Now, $split_{pa}(\alpha_1, \alpha_2)$ gets:

$\alpha_{2,1}$: $\{X = 0,\ p(X)\} \vdash cq(X)$ $\quad$ $\alpha_{2,2}$: $\{X < 0,\ p(X)\} \vdash cq(X)$.

The constraint of $\alpha_{2,1}$ is obtained: (i) by adding the constraints of α_1 (i.e. $X \geq Y, Y \geq 0$) to the constraint of α_2 (i.e. $X \leq 0$), thereby getting $X \geq Y, Y \geq 0, X \leq 0$, and then, (ii) by eliminating Y, which occurs neither in the claim nor in the assumption of α_2, thereby getting $X = 0$ by Proposition 5.16 (3). The constraint of $\alpha_{2,2}$ is obtained by adding the formula $\neg\exists Y(X \geq Y \wedge Y \geq 0)$ to the constraint of α_2, thereby getting, modulo equivalence in $\mathcal{CT}$, the constraint $X < 0$. Similarly, by $split_{pa}$ and Proposition 5.16, we replace $\alpha_1, \alpha_5, \alpha_6$ by:

$\alpha_{1,1}$: $\{X = 0,\ q(X)\} \vdash cp(X)$ $\quad$ $\alpha_{1,2}$: $\{X > 0,\ q(X)\} \vdash cp(X)$
$\alpha_{5,1}$: $\{X < 0, p(X)\} \vdash p(X)$ $\quad$ $\alpha_{6,1}$: $\{X < 0, q(X)\} \vdash q(X)$
$\alpha_{5,2}$: $\{X = 0, p(X)\} \vdash p(X)$ $\quad$ $\alpha_{6,2}$: $\{X = 0, q(X)\} \vdash q(X)$
$\alpha_{5,3}$. $\{X > 0, p(X)\} \vdash p(X)$ $\quad$ $\alpha_{6,3}$: $\{X > 0, q(X)\} \vdash q(X)$.

Then, the extension $\{\alpha_{1,1}, \alpha_{1,2}, \alpha_{2,1}, \alpha_{2,2}, \alpha_3, \alpha_4, \alpha_{5,1}, \alpha_{5,2}, \alpha_{5,3}, \alpha_{6,1}, \alpha_{6,2}, \alpha_{6,3}\}$ is instance-disjoint and non-overlapping.

The following *Argument Splitting* procedure repeatedly applies the two splitting operations to get an instance-disjoint, non-overlapping set of constrained arguments.

PROCEDURE *Argument Splitting.* *Input*: $\Delta \subseteq CArg$.
REPEAT
  IF $\exists \alpha, \beta \in \Delta$ with common constrained instances
    THEN $\Delta := (\Delta \setminus \{\beta\}) \cup split_{ci}(\alpha, \beta)$;
  IF $\exists \alpha, \beta \in \Delta$ s.t. α partially attacks and does not fully attack β
    THEN $\Delta := (\Delta \setminus \{\beta\}) \cup split_{pa}(\alpha, \beta)$;
UNTIL Δ is instance-disjoint and non-overlapping.

From Proposition 7.17, we immediately get the following result.

THEOREM 7.20. *If the Argument Splitting procedure terminates for input Δ and its output is Δ', then (i) $\Delta \equiv \Delta'$ and (ii) Δ' is instance-disjoint and non-overlapping.*

Now, by combining Theorem 7.10 and Argument Splitting, we get a computational method to construct admissible or stable extensions, if the given CABA framework admits any. First, we construct *MGCArg*, which, by Corollary 5.9, is equivalent to *CArg*. Then, by Argument Splitting, we produce $\Delta' \subseteq CArg$ which is a non-overlapping, instance-disjoint set of constrained arguments equivalent to *MGCArg*. Finally, we can identify admissible and stable extensions by a direct application of Theorem 7.10 on Δ'.

However, in order to construct admissible and stable extensions, Δ' should be a *finite* set of constrained arguments. In general, the problem of knowing whether or not a given CABA framework has a finite Δ' is undecidable. We leave it for future work to characterize classes of CABA frameworks for which it is possible to construct finite, instance-disjoint, non-overlapping sets of constrained arguments equivalent to *MGCArg*.

The following example shows how admissible and stable extensions can be computed using the output of Argument Splitting.

Example 7.21 (Ex. 7.19 contd.). The CABA framework has the following two stable, and hence admissible, extensions: $E_1 = \{\alpha_{1,1}, \alpha_{1,2}, \alpha_{2,2}, \alpha_3, \alpha_4, \alpha_{5,1}, \alpha_{6,2}, \alpha_{6,3}\}$ and $E_2 = \{\alpha_{1,2}, \alpha_{2,1}, \alpha_{2,2}, \alpha_3, \alpha_4, \alpha_{5,1}, \alpha_{5,2}, \alpha_{6,3}\}$. We have that: $\alpha_{1,1} \in E_1$ attacks $\alpha_{2,1} \notin E_1$, and $\alpha_{2,1} \in E_2$ attacks $\alpha_{1,1} \notin E_2$. Additional admissible, non-stable extensions are, among others: $\{\alpha_{1,1}, \alpha_3, \alpha_4\}$ and $\{\alpha_{2,1}, \alpha_3, \alpha_4\}$.

Thus, by our method we can effectively construct finite non-ground CABA (admissible or stable) extensions that are equivalent to ground ABA extensions consisting of infinite sets.

8 CONCLUSIONS

We define CABA frameworks, allowing us to use argumentative reasoning in cases that require constraints over infinite domains. The semantics for CABA is defined in terms of constrained arguments and attacks between them. Each constrained argument represents a possibly infinite set of ground instances obtained by substituting the variables therein. This semantics does not require an expensive, potentially infinite, grounding.

This work opens many avenues for future research, including the following ones.

We have focused on conflict-free, admissible, and stable extensions only, but it would be interesting to study other extension-based semantics, e.g. preferred extensions, complete extensions, and grounded extensions. We have focused on the flat case, but it would be interesting to define non-flat CABA, when assumptions may occur in the head of rules. Similarly, it would be interesting to consider variants of CABA frameworks, in the spirit of variants of ABA frameworks, e.g. with preferences between assumptions [8], or probability distributions over assumptions [13].

We have defined CABA frameworks as ABA frameworks with additional components and notions. It would be interesting to explore whether our CABA could be obtained as an instance of standard ABA, for an appropriate choice of deductive system, assumptions and contraries, e.g. in the spirit of default logic [5]. Given that, for the definition of CABA, we drew inspiration from CLP, one could study relations between various forms thereof and CABA.

CABA addresses a very specific computational issue with deploying ABA without expensive, potentially infinite, grounding. It would be interesting to study its computational complexity as well as providing computational machinery therefor, either via mapping to ASP-based CLP systems [4] for stable extensions, or via dispute derivations [8] for admissible extensions. It would also be useful to consider classes of constraint domains where the reasoning tasks are decidable.

Finally, it would be interesting to explore applications of CABA in settings where constraints naturally emerge, as in the case of legal reasoning, in the spirit of the motivating Example 1.1.

ACKNOWLEDGMENTS

Toni was funded by the ERC (ADIX, grant no.101020934). De Angelis and Proietti were supported by MUR PRIN 2022 Project DOMAIN funded by the EU–NextGenEU, M4.C2.1.1, CUP B53D23013220006. De Angelis, Fioravanti, Proietti, and Meo acknowledge the support of PNRR MUR project PE0000013-FAIR. De Angelis, Fioravanti, Meo, Pettorossi, and Proietti are members of INdAM-GNCS.

REFERENCES

[1] Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni. 2008. Verifiable agent interaction in abductive logic programming: The SCIFF framework. *ACM Trans. Comput. Logic* 9, 4, Article 29 (Aug. 2008), 43 pages. https://doi.org/10.1145/1380572.1380578

[2] Krzysztof R. Apt. 1990. Introduction to Logic Programming. In *Handbook of Theoretical Computer Science*, J. van Leeuwen (Ed.). Elsevier, 493–576.

[3] Joaquín Arias, Manuel Carro, Elmer Salazar, Kyle Marple, and Gopal Gupta. 2018. Constraint Answer Set Programming without Grounding. *Theory and Practice of Logic Programming* 18, 3–4 (2018), 337–354. https://doi.org/10.1017/S1471068418000285

[4] Joaquín Arias, Gopal Gupta, and Manuel Carro. 2021. A Short Tutorial on s(CASP), a Goal-directed Execution of Constraint Answer Set Programs. In *ICLP Workshops (CEUR Workshop Proceedings, Vol. 2970)*. CEUR-WS.org. https://ceur-ws.org/Vol-2970/gdepaper1.pdf

[5] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. 1997. An Abstract, Argumentation-Theoretic Approach to Default Reasoning. *Artif. Intell.* 93 (1997), 63–101. https://doi.org/10.1016/S0004-3702(97)00015-5

[6] Pedro Cabalar, Roland Kaminski, Max Ostrowski, and Torsten Schaub. 2016. An ASP Semantics for Default Reasoning with Constraints. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA*, S. Kambhampati (Ed.). IJCAI/AAAI Press, 1015–1021. http://www.ijcai.org/Abstract/16/148

[7] Keith L. Clark. 1977. Negation as Failure. In *Logic and Data Bases, Symposium on Logic and Data Bases, 1977 (Advances in Data Base Theory)*, H. Gallaire and J. Minker (Eds.). Plenum Press, New York, 293–322. https://doi.org/10.1007/978-1-4684-3384-5_11

[8] Kristijonas Cyras, Xiuyi Fan, Claudia Schulz, and Francesca Toni. 2017. Assumption-based Argumentation: Disputes, Explanations, Preferences. *FLAP* 4, 8 (2017). http://www.collegepublications.co.uk/downloads/ifcolog00017.pdf

[9] Kristijonas Čyras, Tiago Oliveira, Amin Karamlou, and Francesca Toni. 2021. Assumption-based argumentation with preferences and goals for patient-centric reasoning with interacting clinical guidelines. *Argument Comput.* 12, 2 (2021), 149–189. https://doi.org/10.3233/AAC-200523

[10] Emanuele De Angelis, Maurizio Proietti, and Francesca Toni. 2024. Learning Brave Assumption-Based Argumentation Frameworks via ASP. In *ECAI 2024 - 27th European Conference on Artificial Intelligence (Frontiers in Artificial Intelligence and Applications, Vol. 392)*, U. Endriss, F. S. Melo, K. Bach, A. J. Bugarín Diz, J. M. Alonso-Moral, S. Barro, and F. Heintz (Eds.). IOS Press, 3445–3452. https://doi.org/10.3233/FAIA240896

[11] Phan Minh Dung. 1995. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artif. Intell.* 77, 2 (1995), 321–358. https://doi.org/10.1016/0004-3702(94)00041-X

[12] Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. 2009. Assumption-Based Argumentation. In *Arg. in AI*. Springer, 199–218.

[13] Phan Minh Dung and Phan Minh Thang. 2010. Towards (Probabilistic) Argumentation for Jury-based Dispute Resolution. In *COMMA*. 171–182. https://doi.org/10.3233/978-1-60750-619-5-171

[14] Xiuyi Fan. 2018. A Temporal Planning Example with Assumption-Based Argumentation. In *PRIMA 2018: Principles and Practice of Multi-Agent Systems*. 362–370. https://doi.org/10.1007/978-3-030-03098-8_22

[15] Alejandro J. García and Guillermo R. Simari. 2014. Defeasible logic programming: DeLP-servers, contextual queries, and explanations for answers. *Arg. & Comp* 5, 1 (2014), 63–88. https://doi.org/10.1080/19462166.2013.869767

[16] Joxan Jaffar and Jean Louis Lassez. 1987. Constraint Logic Programming. In *Conference Record of the Fourteenth Annual ACM Symposium on Principles of Programming Languages, Munich, Germany, 1987*. ACM Press, 111–119. https://doi.org/10.1145/41625.41635

[17] Joxan Jaffar and Michael Maher. 1994. Constraint Logic Programming: A Survey. *Journal of Logic Programming* 19/20 (1994), 503–581. https://doi.org/10.1016/0743-1066(94)90033-7

[18] Kenneth Kunen. 1987. Negation in Logic Programming. *Journal of Logic Programming* 4, 4 (1987), 289–308. https://doi.org/10.1016/0743-1066(87)90007-0

[19] John W. Lloyd. 1987. *Foundations of Logic Programming*. Springer-Verlag, Berlin. Second Edition.

[20] Fabrizio Macagno, Douglas Walton, and Chris Reed. 2017. Argumentation Schemes. History, Classifications, and Computational Applications. *FLAP* 4, 8 (2017). http://www.collegepublications.co.uk/downloads/ifcolog00017.pdf

[21] Paolo Mancarella, Giacomo Terreni, Fariba Sadri, Francesca Toni, and Ulle Endriss. 2009. The CIFF proof procedure for abductive logic programming with constraints: Theory, implementation and experiments. *Theory Pract. Log. Program.* 9, 6 (2009), 691–750. https://doi.org/10.1017/S1471068409990093

[22] Elliott Mendelson. 1997. *Introduction to Mathematical Logic*. Chapman&Hall. 4th Edition.

[23] Fabrizio Russo, Anna Rapberger, and Francesca Toni. 2024. Argumentative Causal Discovery. In *KR*. 938–949. https://doi.org/10.24963/KR.2024/88

[24] Kenneth Skiba, Matthias Thimm, and Johannes Peter Wallner. 2024. Ranking Transition-Based Medical Recommendations Using Assumption-Based Argumentation. In *RATIO*. 202–220. https://doi.org/10.1007/978-3-031-63536-6_12

[25] Peter Stuckey. 1991. Constructive Negation for Constraint Logic Programming. In *Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands*. IEEE Computer Society, 328–339. https://doi.org/10.1109/LICS.1991.151657

[26] Peter Stuckey. 1995. Negation and Constraint Logic Programming. *Inf. Comput.* 118, 1 (1995), 12–33. https://doi.org/10.1006/INCO.1995.1048

[27] Francesca Toni. 2014. A tutorial on Assumption-based Argumentation. *Arg. & Comp* 5, 1 (2014), 89–117.

[28] Douglas Walton, Chris Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press. http://www.cambridge.org/us/academic/subjects/philosophy/logic/argumentation-schemes

[29] Zhiwei Zeng, Zhiqi Shen, Jing Jih Chin, Cyril Leung, Yu Wang, Ying Chi, and Chunyan Miao. 2020. Explainable and Contextual Preferences based Decision Making with Assumption-based Argumentation for Diagnostics and Prognostics of Alzheimer's Disease. In *AAMAS*. 2071–2073. https://doi.org/10.5555/3398761.3399078