Supporting Argumentation Systems by Graph Representation and Computation

Jérôme Fortin, Rallou Thomopoulos, Jean-Rémi Bourguet, Marie-Laure Mugnier

HAL Id: lirmm-00768328

https://hal-lirmm.ccsd.cnrs.fr/lirmm-00768328

Submitted on 21 Dec 2012

Supporting Argumentation Systems by Graph Representation and Computation

Jérôme Fortin
Université Montpellier II
fortin@supagro.inra.fr

Rallou Thomopoulos
INRA
rallou.thomopoulos@supagro.inra.fr

Jean-Rémi Bourguet
Université Montpellier III
bourguet@lirmm.fr

Marie–Laure Mugnier
Université Montpellier II
mugnier@lirmm.fr

Abstract

Argumentation is a reasoning model based on arguments and on attacks between arguments. It consists in evaluating the acceptability of arguments, according to a given semantics. Due to its generality, Dung's framework for abstract argumentation systems, proposed in 1995, is a reference in the domain. Argumentation systems are commonly represented by graph structures, where nodes represent arguments and edges attacks between arguments. However beyond this graphical support, graph operations have not been considered as reasoning tools in argumentation systems. This paper proposes a conceptual graph representation of an argumentation system and a computation of argument acceptability relying on conceptual graph defaults.

1 Introduction

Argumentative reasoning is based on the construction and the evaluation of interacting arguments. Most of the existing argumentation models are grounded on the abstract argumentation framework proposed by Dung in [Dung, 1995]. It consists of a set of arguments and a binary relation on this set, expressing conflicts among arguments. Argumentation systems are commonly represented by graph structures, where nodes represent arguments and edges attacks between arguments.

Two main tools are currently available for argument visualization: Araucaria [Araucaria, website] and Carneades [Carneades, website]. These tools facilitate the visual display of arguments and in particular their structure (e.g. premises and conclusion), however they are not reasoning tools. This paper deals with the reasoning issue, moreover it regards a different level, since it does not focus on argument structure, but on the representation of a whole argument base and on the computation of maximal acceptable sets of arguments from it.

The chosen formalism is the conceptual graph (CG) model [Chein and Mugnier, 2009]. Indeed, it is the only AI formalism that combines a graphical representation and graph-based operations, together with an equivalent logical interpretation, providing a well-founded graphical and logical reasoning model. A recent study [de Moor *et al.*, 2009] has considered the representation of "argument maps" in the conceptual graph model. It designs an architecture to map argument maps and conceptual graphs, to allow visualizing the arguments put forward by different actors and examining them through queries. The authors present the advantages of using conceptual graphs, and highlight that, beyond this first step, further reasoning features remain to be explored.

In this paper, we focus on this reasoning issue. Nonmonotonic reasoning is used as a computational tool to determine maximal acceptable sets of arguments. It is instantiated by conceptual graph defaults [Baget *et al.*, 2009; Baget and Fortin, 2010]. The paper is organized as follows. Section 2 presents the background on argumentation systems and conceptual graphs. Section 3 introduces a representation of argumentation systems in the conceptual graph formalism. Section 4 proposes a way of computing four kinds of acceptable sets of arguments, namely the naive, the admissible, the preferred, and the stable sets, according to the semantics of argumentation systems.

2 Background

This section introduces fundamental notions of argumentation systems and the basic conceptual graph formalism as well as one of its extensions, namely CG defaults.

2.1 Argumentation systems

Three main steps can be distinguished in an argumentation process: 1) constructing *arguments* and counter-arguments, 2) evaluating the collective *acceptability* of sets of arguments, and 3) drawing *justified conclusions*. In [Dung, 1995], an abstract argumentation framework is defined as follows.

Definition 1 (Dung's argumentation framework). *An argumentation framework is a pair $AF = \langle A, R \rangle$ where A is a set of arguments and $R \subseteq A \times A$ is an attack relation. An argument α attacks an argument β iff $(\alpha, \beta) \in R$.*

In the above definition, arguments are abstract entities, whose origin and structure are left unknown. With each argumentation system is naturally associated a directed graph whose nodes are the arguments and edges represent the attack relation between them.

Definition 2 (Conflict-free, Defense). *Let $B \subseteq A$.*

- *B is conflict-free iff $\nexists \, \alpha, \, \beta \in B$ such that $(\alpha, \beta) \in R$.*

- *B defends an argument $\alpha \in B$ iff for each argument $\beta \in A$, if $(\beta, \alpha) \in R$, then $\exists \gamma \in B$ such that $(\gamma, \beta) \in R$.*

Among all the conflicting arguments, one has to select acceptable subsets of arguments for inferring conclusions and making decision. In [Dung, 1995; Bondarenko *et al.*, 1997], several semantics for the notion of acceptability have been proposed. For the purpose of this paper, we only recall naive, admissible, preferred, and stable semantics. Other semantics (e.g. complete, grounded) are not presented here.

Definition 3 (Acceptability semantics). *Let $B \subseteq A$.*

- *B is a* naive *set iff it is a maximal (w.r.t. set-inclusion) conflict-free set.*

- *B is an* admissible *set iff it is a conflict-free set that defends all its elements.*

- *B is a* preferred *set iff it is a maximal (w.r.t. set-inclusion) admissible set.*

- *B is a* stable *set iff it is a maximal (w.r.t. set-inclusion) conflict-free set such that every element of $(A \setminus B)$ is attacked by an element in B.*
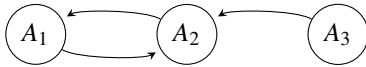
Figure 1: Graph of attacks

Figure 1 presents a graph of attacks, where A_1 and A_2 attack each other and argument A_3 attacks A_2. There are two naive sets $S_1 = \{A_1, A_3\}$ and $S_2 = \{A_2\}$. S_1 is admissible, preferred and stable, while S_2 has none of these properties.

Note that a stable set is *a fortiori* a preferred set (which is itself an admissible set). Given an argumentation framework, there is always a naive (resp. admissible, preferred) set of arguments, possibly equal to the empty set. This is not true for the stable semantic. For example, consider an argumentation framework with three arguments A_1, A_2, A_3, such that A_1 attacks A_2, A_2 attacks A_3 and A_3 attacks A_1 (hence a cycle of attacks). It does not have any stable set of arguments because, for any conflict-free set, there is at least one argument outside this set that is not attacked by any argument of the set.

2.2 The Conceptual Graph Formalism

The conceptual graph (CG) formalism [Sowa, 1984; Chein and Mugnier, 2009] is a knowledge representation and reasoning formalism based on labelled graphs. In its simplest form, a CG knowledge base is composed of two parts: the *support*, which encodes terminological knowledge –and constitutes a part of the represented domain ontology– and *basic conceptual graphs* built on this support, which express assertional knowledge, or *facts*. The knowledge base can be further enriched by other kinds of knowledge built on the support. Here, we will consider two kinds of *rules*: "usual rules" and CG defaults, which lead to non-monotonic reasoning.

The support. It provides the ground vocabulary used to build the knowledge base, i.e., the types of concepts, denoted

by $\mathcal{T}_C$ and the types of relations that can link concept instances, denoted by $\mathcal{T}_R$. $\mathcal{T}_C$ is partially ordered by a *kind of* relation, with $\top$ being its greatest element. $\mathcal{T}_R$ is also partially ordered by a *kind of* relation, with any two comparable relation types having necessarily the same arity (i.e., number of arguments). Each relation has a signature that specifies its arity and the maximal concept type of each of its arguments.

Figure 2 shows the sets of concept types and of relation types (each with their signature) used in the application. $\mathcal{T}_R$ is partitioned into two sets, $\mathcal{T}_{R_1}$ and $\mathcal{T}_{R_2}$. $\mathcal{T}_{R_1}$ is a set of unary relations representing properties of sets (of arguments): being a naive set (Acc^{na}), an admissible set (Acc^{ad}), a preferred set (Acc^{pr}), a stable set (Acc^{st}) or a non-stable set ($nonAcc^{pr}$). $\mathcal{T}_{R_2}$ is a set of binary relations, that contains for instance the attack relation $\mathcal{R}$ (with signature (Arg, Arg)).
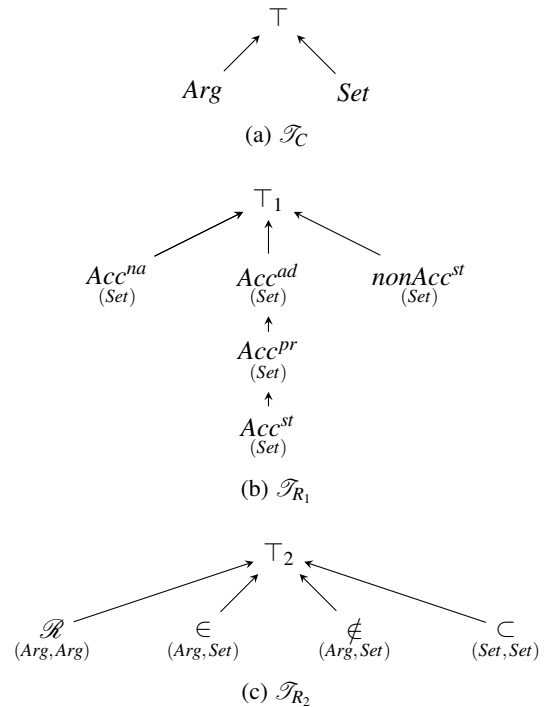
Figure 2: CG Support with concept and relation types for representing argumentation systems

Basic conceptual graphs. A basic CG is a bipartite graph composed of: (i) a set of *concept nodes* (pictured as rectangles), which represent entities, attributes, states or events; (ii) a set of *relation nodes* (pictured as ovals), which express the nature of relationships between concept nodes; (iii) a set of *edges* linking relation nodes to concept nodes; (iv) a *labelling* function, which labels each node or edge: the label of a concept node is a pair $t : m$, where t is a concept type and m is a marker, the label of a relation node is a relation type, and the label of an edge is its rank in the total order on the arguments of the incident relation node; furthermore, the relation type signatures have to be satisfied: the number of edges incident to a relation node is equal to the arity of its type r, and the
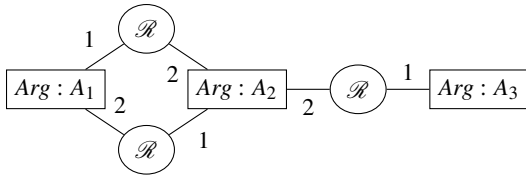
Figure 3: Basic conceptual graph

concept type assigned to its neighbor by an edge labelled i is less or equal to the ith element of the signature of r. The marker of a concept node can be either an identifier referring to a specific individual (for instance A_1 of type *Arg* in Figure 3) or the generic marker (noted $*$) referring to an unspecified instance of the associated concept type (see for instance in Figure 4). A basic CG without occurrence of the generic marker is said to be totally instantiated. Figure 3 shows a totally instantiated basic CG built on the support of Figure 2, which encodes the attack graph of Figure 1.

Logical translation. Conceptual graphs have a logical translation in first-order logic, which is given by a mapping classically denoted by ϕ. ϕ assigns a formula $\phi(S)$ to a support S, and a formula $\phi(G)$ to any basic CG G on this support. First, each concept or relation type is translated into a predicate (a unary predicate for a concept type, and a predicate with the same arity for a relation type) and each individual marker occurring on the graphs is translated by a constant. Then, the *kind of* relation between types of the support is translated by logical implications. For example, the fact that Acc^{st} is a specialization of Acc^{pr} (Figure 2b) is translated by: $\forall x,\ Acc^{st}(x) \rightarrow Acc^{pr}(x)$

Then, given a basic conceptual graph G on S, $\phi(G)$ is built as follows. A distinct variable is assigned to each node with a generic marker. An atom of the form $t(e)$ is assigned to each concept node with label $t : m$, where e is the variable assigned to this node if $m = *$, otherwise $e = m$. An atom of the form $r(e_1, \ldots, e_k)$ is assigned to each relation node with label r, where e_i is the variable or the constant corresponding to the ith neighbor of the relation. $\phi(G)$ is then the existential closure of the conjunction of all atoms assigned to its nodes. For instance, the logical translation of the conceptual graph represented in Figure 3 is the following: $Arg(A_1) \wedge Arg(A_2) \wedge Arg(A_2) \wedge \mathscr{R}(A_1, A_2) \wedge \mathscr{R}(A_2, A_1) \wedge \mathscr{R}(A_3, A_2)$
Note that in this case, the graph is totally instantiated, thus its logical translation has no variable.

Specialization relation, homomorphism. Any set of conceptual graphs is partially preordered by a *specialization relation*, which can be computed by a *homomorphism* (allowing the restriction of the node labels) also called projection in the conceptual graph community. The specialization relation, and thus homomorphism, between two graphs, corresponds to the logical entailment between the corresponding formulas, i.e., there is a homomorphism from G to H (both built on a support S) if and only if $\phi(G)$ is entailed by $\phi(H)$ and $\phi(S)$

(e.g. [Chein and Mugnier, 2009])[1].

Basic CG rules. Basic CG rules [Salvat and Mugnier, 1996] are an extension of basic CGs. A CG rule (notation: $R = (H, C)$) is of the form "if H then C", where H and C are two basic CG (respectively called the *hypothesis* and the *conclusion* of the rule), which may share some concept nodes. Formally, it can be defined as a single bicolored basic CG, as illustrated in Figure 4: the hypothesis is composed of the white nodes; the conclusion is induced by the black nodes and the white concept nodes that are linked to a black relation node; these latter nodes that belong both to the hypothesis and the conclusion are called *frontier* nodes. Intuitively, this rule says that "if an argument x attacks an argument y that attacks an argument z, then x defends z".
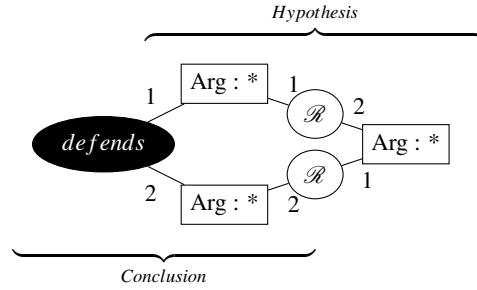

Figure 4: CG rule

A rule R is applicable to a basic CG G if there is a homomorphism from its hypothesis to G. Let π be such a homomorphism. Then, the *application of R on G according to π* produces a basic CG obtained from G by adding the conclusion of R according to π, i.e., merging each frontier node c of the added conclusion with the node of G that is image of c by the homomorphism π. For instance, the rule in Figure 4 can be applied twice to the basic CG of Figure 3, which allows to infer that A_3 defends A_1 and that A_1 defends itself.

The mapping ϕ to first-order logic is extended to CG rules. Let $R = (H, C)$ be a CG rule, and let $\phi'(R)$ denotes the conjunction of atoms associated with the basic CG underlying R (all variables are kept free). Then, $\phi(R) = \forall x_1 \ldots \forall x_k (\phi'(H) \rightarrow (\exists y_1 \ldots \exists y_q \phi'(C)))$, where $\phi'(H)$ and $\phi'(C)$ are the restrictions of $\phi'(R)$ to the nodes of H and C respectively, $x_1, \ldots, x_k$ are all the variables appearing in $\phi(H)$ and $y_1, \ldots, y_q$ are all the variables appearing in $\phi(C)$ but not in $\phi(H)$. For example, the rule of Figure 4 is translated as follows: $\forall x\ \forall y \forall z\ Arg(x) \wedge Arg(y) \wedge Arg(z) \wedge \mathscr{R}(x, y) \wedge \mathscr{R}(y, z) \rightarrow defends(x, z)$.

The rule application mechanism is logically sound and complete: given a set of rules $\mathscr{R}$, basic CGs G and H (representing for instance a query and a set of facts), G is entailed by the logical formulas assigned to $\mathscr{R}$, H and S (the support) if and only if there is a sequence of rule applications with rules of $\mathscr{R}$ leading from H to a basic CG H' such that there is

---

[1]Note that, for the completeness part, H has to be in normal form: each *individual* marker appears at most once in it.

a homomorphism from G to H' (in other words, by applying rules to H, it is possible to obtain H' which entails G).

When a rule is applied, it may create new individuals (one for each generic concept node in its conclusion, i.e., one for each existential variable y_i in the logical translation of the rule). In the following, we will assume that all facts (represented as basic CGs) are completely instantiated. Then, when a rule is applied, we will instantiate each new generic concept node created, by replacing its generic marker with a new individual marker (which can be seen as a skolem function, moreover without variable in this case). This will allow us to present CG defaults in a simpler way.

2.3 Conceptual Graph Defaults

A Brief Introduction to Reiter's Default logic

Let us recall some basic definitions of Reiter's default logics. For a more precise description and examples, the reader should refer to [Reiter, 1980; Brewka and Eiter, 1999].

Definition 4 (Reiter's Default theory). *A Reiter's* default theory *is a pair* (Δ, W) *where W is a set of FOL formulae and* Δ *is a set of* defaults *of form* $\delta = \frac{\alpha(\overrightarrow{x}):\beta_1(\overrightarrow{x}),\cdots,\beta_n(\overrightarrow{x})}{\gamma(\overrightarrow{x})}$, $n \geq 0$, *where* $\alpha(\overrightarrow{x})$, $\beta_i(\overrightarrow{x})$ *and* $\gamma(\overrightarrow{x})$ *are FOL formulae for which each free variable is in the tuple of variable* $\overrightarrow{x} = (x_1, \cdots, x_k)$.

The intuitive meaning of a default δ is "For all individuals $\overrightarrow{x} = (x_1, \cdots, x_k)$, if $\alpha(\overrightarrow{x})$ is believed and each of $\beta_1(\overrightarrow{x}), \cdots, \beta_n(\overrightarrow{x})$ can be consistently believed, then one is allowed to believe $\gamma(\overrightarrow{x})$". $\alpha(\overrightarrow{x})$ is called the *prerequisite*, the $\beta_i(\overrightarrow{x})$ are called the *justifications* and $\gamma(\overrightarrow{x})$ is called the *consequent*. A default is said *closed* if $\alpha(\overrightarrow{x})$, $\beta_i(\overrightarrow{x})$ and $\gamma(\overrightarrow{x})$ are all closed FOL formulae. A default theory (Δ, W) is said to be *closed* if all its defaults are closed.

Intuitively, an *extension* of a default theory (Δ, W) is a set of formulae that can be obtained from (Δ, W) while being consistently believed. More formally, an extension E of (Δ, W) is a minimal deductively closed set of formulae containing W such that for any $\frac{\alpha:\beta}{\gamma} \in \Delta$, if $\alpha \in E$ and $\neg\beta \notin E$, then $\gamma \in E$. The following theorem [Reiter, 1980] provides an equivalent characterization of extensions that we use here as a formal definition.

Theorem 1 (Extension). *Let* (Δ, W) *be a closed default theory and E be a set of FOL formulae. We inductively define* $E_0 = W$ *and for all* $i \geq 0$, $E_{i+1} = Th(E_i) \cup \{\gamma \mid \frac{\alpha:\beta_1\cdots,\beta_n}{\gamma} \in \Delta, \alpha \in E_i \text{ and } \neg\beta_1, \cdots, \neg\beta_n \notin E\}$, *where* $Th(E_i)$ *is the deductive closure of* E_i. *Then E is an extension of* (Δ, W) *iff* $E = \cup_{i=0}^{\infty} E_i$.

Note that this characterization is not effective for computational purposes since both E_i and $E = \cup_{i=0}^{\infty} E_i$ are required for computing E_{i+1} (for more details on generating extensions, see in [Reiter, 1980; G. Brewka, 2007]). Moreover, Theorem 1 is only useful to deal with a closed default theory, which is less expressive than the general case, since no variable can be shared between the hypothesis, the conclusion or a justification. When we want to apply some non-closed default, we first have to instantiate each free variable by all the constants that may appear in the knowledge base, which yields a set of closed defaults.

Default Rules in the Conceptual Graph Formalism

We now present an extension of CG rules, which has been introduced in [Baget *et al.*, 2009; Baget and Fortin, 2010] and allows for default reasoning. It can be seen as a graphical implementation of a subset of Reiter's default logic: indeed, we restrict the kind of formulae that can be used in the three components of a default. On the other hand, we can deal directly with non–closed defaults, i.e., without instantiating free variables before processing the defaults. In Reiter's logic, the application of a default is subject to a consistency check with respect to current knowledge: each justification J has to be consistent with the current knowledge, i.e., $\neg J$ should not be entailed by it. In CG defaults, justifications are replaced by graphs called *constraints*; a constraint C can be seen as the negation of a justification: C should not be entailed by current knowledge.

Definition 5 (CG defaults). *A CG default is a tuple* $D = (H, C, C_1, \ldots, C_k)$ *where H is called the hypothesis, C the conclusion and $C_1, \ldots,$ and C_k are called the constraints of the default; all components of D are themselves basic CGs and may share some concept nodes.*

Briefly said, H, C and each C_i respectively correspond to the prerequisite, the consequent and the negation of a justification in a Reiter's default.

In this paper, we will represent CG defaults by a multi-colored basic CG. As in a CG rule, the hypothesis is represented by the white nodes. The conclusion is represented by the black nodes and frontier nodes. Each constraint is represented by a different level of gray.

The intuitive meaning of a CG default is rather simple: "for all individuals $x_1 \ldots x_k$, if $H[x_1 \ldots x_k]$ holds true, then $C[x_1 \ldots x_k]$ can be inferred provided that no $C_i[x_1 \ldots x_k]$ holds true". If we can map by homomorphism the hypothesis H to a fact graph G, then we can add the conclusion of the default according to this homomorphism (as in a standard rule application), unless this homomorphism can be extended to map one of the constraints. As already pointed out, while the negation of a justification in a Reiter's default should not be entailed, in a CG default the constraint itself should not be entailed.

The entailment mechanism is based on the construction of a default derivation tree.

Let $\mathscr{K} = ((\mathscr{S}, G, \mathscr{R}), \mathscr{D})$ be a knowledge base, where G is a basic CG, $\mathscr{R}$ is a set of CG rules, and $\mathscr{D}$ is a set of CG defaults, all defined over the support $\mathscr{S}$. As previously mentioned, we will assume that G is completely instantiated. Then the rule application mechanism ensures that all derived facts are also completely instantiated.

A node of the default derivation tree $\text{DDT}(\mathscr{K})$ is labelled by a basic CG called fact and a set of basic CGs called constraints. A node of $\text{DDT}(\mathscr{K})$ with label $(G, \mathscr{C})$ is said to be *valid* if there is no homomorphism from a constraint in $\mathscr{C}$ or a constraint occurring in the label of one of its ancestors to G. We now define inductively the tree $\text{DDT}(\mathscr{K})$:

- the root is labelled by $(G, \emptyset)$

- if x is a valid node of $\text{DDT}(\mathscr{K})$ with label $(F, \mathscr{C})$

for every default $D = (H,C,C_1,\ldots,C_k)$ in $\mathscr{D}$, for every homomorphism π from H to a basic CG F' $\mathscr{R}$-derived from F,

x has a successor whose fact is obtained by the application of D as a classical rule $R = (H,C)$ without considering it as a default, and whose constraints are the $\pi(C_i)$, iff that successor is valid.

In the above definition, $\pi(C_i)$ is obtained from C_i by replacing the labels of concept nodes that belong to the domain of π (thus are shared with H) with their image by π. This allows to bind some nodes of C_i to nodes of the current fact. Let us consider for instance the DDT in Figure 6, obtained with the set of rules in Figure 5 and the initial fact G in Figure 3. The successor of the root is obtained by applying the rule R_N, which has an empty hypothesis and no constraints: hence, the conclusion is simply added to G, after instantiation of the generic concept node [Set:*] with a new individual marker (denoted by E in the figure). The left most successor of this tree node (let us note it n_3) is obtained by applying the rule D_N with the concept nodes [Set:*] and [Arg:*] from the rule hypothesis being respectively mapped to [Set:E] and [Arg:A_1]. Both constraints of D_N are instantiated accordingly. It is checked that n_3 is valid: indeed, the instantiated constraints do not map to the newly built fact. Thus n_3 is actually added to the tree. Note that all descendants of n_3 will have to satisfy the constraints labelling n_3.

The leaves of $DDT(\mathscr{K})$ exactly encode extensions of a default theory (see [Baget *et al.*, 2009; Baget and Fortin, 2010]).

3 Argumentation System Modelling

This section proposes a representation of an abstract argumentation system in the CG formalism, as previously introduced. The associated reasoning mechanisms that computes acceptable sets of arguments, will be presented in Section 4.

3.1 Support Description

To encode an argumentation system, independently from a given application domain, the elements that constitute an argumentation framework (arguments, sets of arguments), their properties (admissibility semantics) and the relations between them (attack relation, membership and inclusion relations) have to be introduced in the support of the CG model.

Figure 2 shows a support that represents these elements. The fact that stable sets of acceptable arguments are specializations of preferred sets, which are themselves specializations of admissible sets, is directly encoded in the support through the "kind of" relation.

3.2 Graph of Attacks

Based on the generic support of Figure 2, a graph of attacks is defined as follows.

Definition 6 (Graph of attacks). *A graph of attacks is a basic CG such that:*

- *concept nodes are labelled by pairs Arg : i, where i denotes an instance of an argument;*

- *relation nodes are labelled by R, which represents the attack relation.*

Figure 3 shows an example of a graph of attacks in which A_1 attacks A_2, A_2 attacks A_1 and A_3 attacks A_2.

4 Computing acceptable sets of arguments using CG defaults

In this section, we use CG defaults to compute acceptable sets of arguments.

4.1 Computing naive sets of acceptable arguments

We will show how to compute all the naive sets of arguments using two rules (a "classic" CG rule and a CG default, see Figure 5). The way we compute the naive sets is thus purely declarative. Using these rules, the default computation mechanism calculates the default extensions. Each of these extensions encodes in a graphical manner a naive set of arguments.
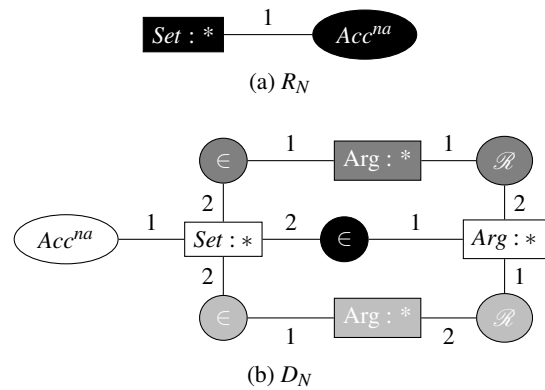

Figure 5: Rules generating naive sets of acceptable arguments

The first rule encodes the following information: "A naive set of arguments exists". Indeed, as mentioned in section 2.1, for any graph of attacks, a naive set of arguments always exists. This rule (denoted by R_N) is a classic CG rule (see Figure 5(a)).

Given a conflict-free set of arguments E, a naive set of arguments (which has to be maximal) can be built by iteratively adding some arguments. An argument a may be added to E if $E \cup \{a\}$ is still conflict-free. The CG default D_N given in Figure 5(b) is designed in such a way that the argument a is added to the set E only if it is not in conflict with any argument of E. This is guaranteed by the two constraints of D_N:

- the first one (in dark gray) ensures that a does not attack any argument of E;

- the second one (in light gray) ensures that a is not attacked by any argument of E.

Therefore, applying this rule to a graph of attacks preserves the property that the group of arguments linked to the set E by the relation $\in$ is conflict-free.

The deduction mechanism based on the default deviation tree ensures that when a default extension is computed, the rule D_N cannot be applied any more. Hence the extension

node labelled by E is a maximal (w.r.t. set inclusion) conflict-free set of arguments, i.e. a naive set.

Figure 6 shows the default derivation tree that is computed to obtain the default extensions encoding the naive sets. The default extensions are encoded in the leaves of the tree. Note that two of the leaves of the tree are identical, so we obtain only two different default extensions: $Acc_1^{na} = \{A_2\}$ $Acc_1^{na} = \{A_1, A_2\}$.

4.2 Computing preferred sets of acceptable arguments

One way to obtain the preferred sets from the naive ones is thus to iteratively remove the non-defended arguments. For that, we use the two CG defaults shown in Figure 7. Given a set of arguments, if there is an argument in this set that is not defended, the CG default D_{CP1} creates a new set, and the non-defended argument is declared as not belonging to this set. The CG default D_{CP2} adds all the arguments to the new set of arguments, unless they have been declared as not belonging to this set. By applying these two rules until obtaining of the default extensions, the resulting sets of arguments have the property of being:

- conflict-free, since they are subsets of naive sets;

- without non-defended arguments.

Among them, the maximal ones (w.r.t. set-inclusion) are the preferred sets of acceptable arguments. Note that, to select maximal sets, one would have to compare extensions, which is not possible in our framework. Thus, tagging such sets by the Acc^{pr} relation, indicating they are preferred sets, is done outside the framework.
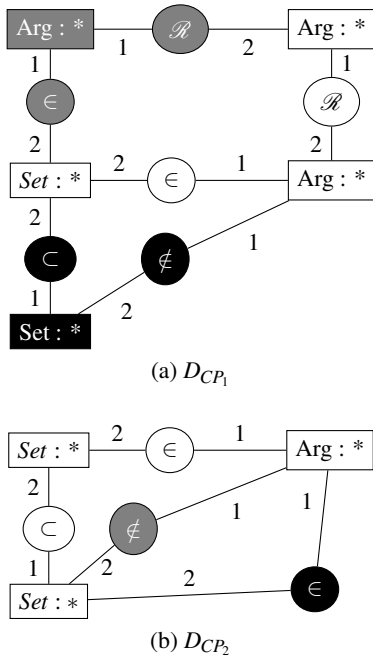

(a) D_{CP_1}

(b) D_{CP_2}

Figure 7: Rules generating preferred sets of acceptable arguments
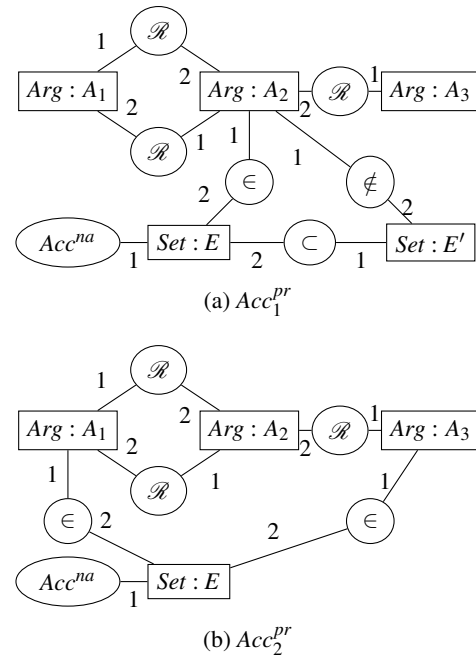

(a) Acc_1^{pr}

(b) Acc_2^{pr}

Figure 8: Extensions computed from the leaves of Figure 6, with the CG defaults of Figure 7

Figure 8 represents the preferred sets of our running example (subject to maximality test). The first set, derived from Acc_1^{na}, is the empty set (since there is no $\in$ relation linking E' to any argument). As it is included in the set $Acc_2^{pr} = \{A_1, A_3\}$, derived from Acc_2^{na}, it is not maximal and thus not tagged as a preferred set.

4.3 Computing stable sets of acceptable arguments

A stable set of arguments is *a fortiori* a preferred set. To be stable, a preferred set of arguments has to attack all the arguments that are not in the set. It turns out that once a set of preferred arguments is computed, it is easy to check whether it is also a stable set of arguments. This can be done using the CG default represented in Figure 9, which starts from a preferred set and expresses a condition to reject it if it is not a stable set; it is tagged as "not stable" (denoted by $notAcc^{st}$) unless each argument belongs to it (dark gray constraint) or is attacked by an argument that belongs to it (light gray constraint). Then, when a default extension is computed, a preferred extension is identified as a stable extension iff it is not tagged as "not stable".

In our running example, the unique preferred set of arguments $Acc_2^{pr} = \{A_1, A_3\}$ is also a stable set, since the only argument that is not in the set, A_2, is attacked by A_1 and A_3.

5 Conclusion

In this paper, we have shown how an argumentation framework can be represented in the CG formalism. This formalism also allows to compute different kinds of acceptable sets of arguments. However, it does not capture the notion of maximality in the definition of preferred sets. Therefore, in order
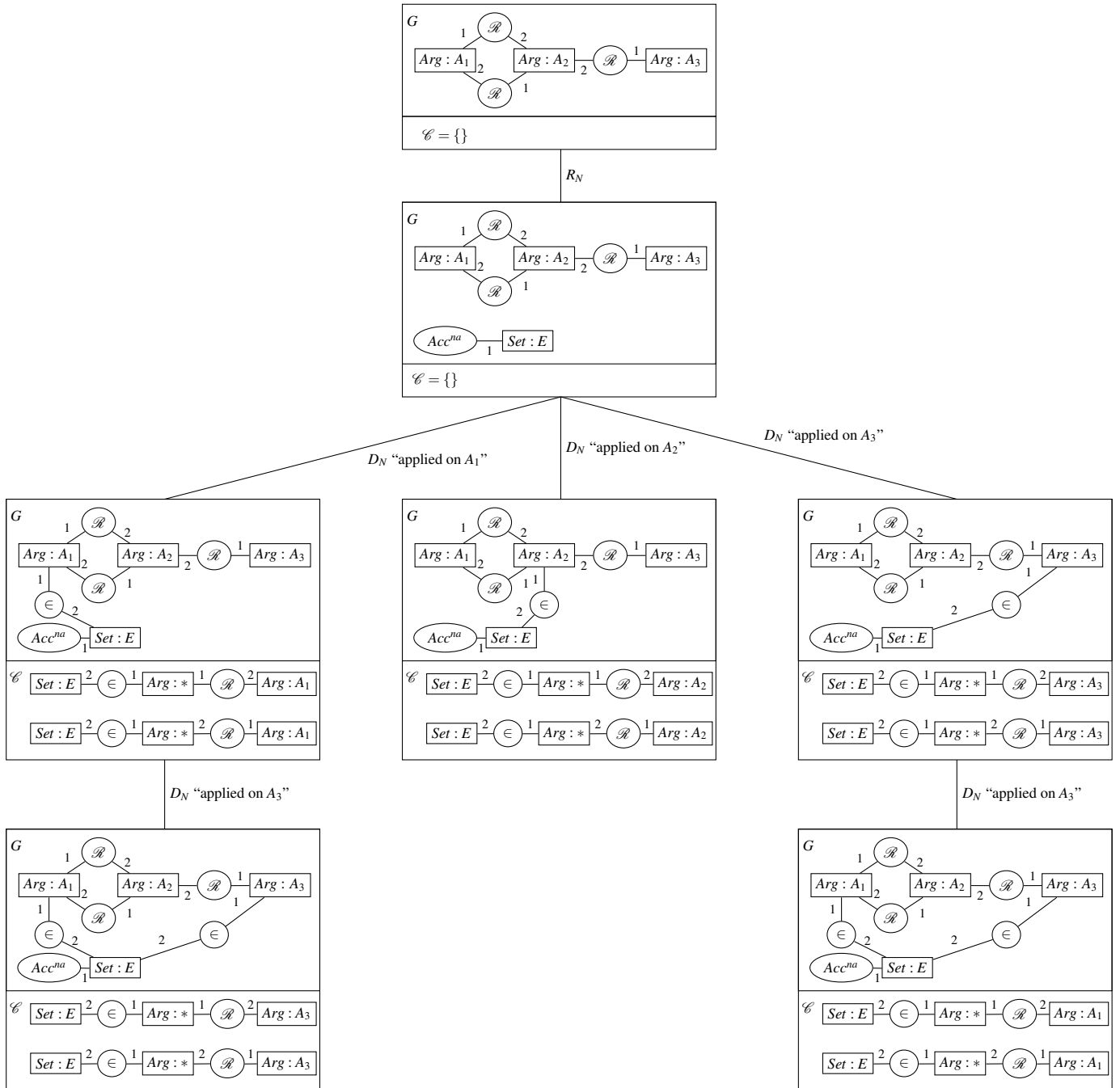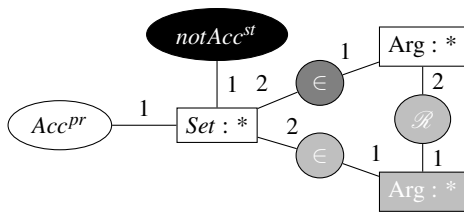
Figure 6: DDT

Figure 9: Rule that tags preferred sets of arguments as "not stable"

to be properly used in an argumentation context, this formalism still needs to be extended.

Another interest of using conceptual graphs in this context would consist in representing not only the relationships between arguments, but also the *internal structure* of arguments. Indeed, basic CGs can be extended to nested CGs, in which concept nodes not only have a type and a marker, but also a description, which is itself a nested CG. The basic homomorphism notion can be easily extended to nested CG. Generally speaking, this allows for a hierarchical representation of knowledge and reasoning by taking this structuring into account. In our application case, the first level would correspond to arguments seen as "black boxes". Then, by "zooming" on arguments, one would have access to the internal description of arguments. Since the internal structure of an argument is a CG, it benefits from the graph mechanisms for reasoning. In the literature, there has been several proposals to represent this internal structure (e.g. [Bentahar *et al.*, 2010]). In a preliminary study [Bourguet, 2010], whose aim was to represent the viewpoints of different actors and the associated arguments in a health policy case, we chose to represent an argument with several parts, one of them being the *action* advocated by the argument. Any two different actions were either in a specialization relation (i.e., one action is more specific than the other) or incompatible. The attack relation between arguments was computed from the action parts of the arguments: an argument *a* attacks an argument *b* if the action of *a* is not entailed by the action of *b*, i.e., either *a* and *b* are incompatible, or *a* is a strict specialization of *b*. In a CG framework, an argument can be represented as a concept node with a nested description, which is itself partitioned into several CGs, one of them corresponding to the action advocated by the argument. A set of arguments is then a nested CG, in which each concept node is an argument. The attack relation can be computed automatically by comparing the action graphs of arguments. For the above attack relation, this can be done with a simple homomorphism check: if the action of *a* does not map by homomorphism to the action of *b*, then *a* attacks *b*. This is only one simple example of how the internal structure of arguments can be represented, and the attack relation generated, in the CG framework. As for further work, we want to study the adequacy of this framework with the proposals in the argumentation literature.

References

[Araucaria, website] Araucaria.
http://araucaria.computing.dundee.ac.uk/, website.

[Baget and Fortin, 2010] J.F. Baget and J. Fortin. Default conceptual graph rules, atomic negation and Tic–Tac–Toe. In *Proc. of ICCS'10*, volume 6208 of *LNAI*, pages 42–55, 2010.

[Baget *et al.*, 2009] J.F. Baget, M. Croitoru, J. Fortin, and R. Thomopoulos. Default conceptual graph rules : preliminary results for an agronomy application. In *Proc. of ICCS'09*, volume 5662 of *LNAI*, pages 86–99, 2009.

[Bentahar *et al.*, 2010] Jamal Bentahar, Bernard Moulin, and Micheline Bélanger. A taxonomy of argumentation models used for knowledge representation. *Artif. Intell. Rev.*, 33(3):211–259, 2010.

[Bondarenko *et al.*, 1997] A Bondarenko, P M Dung, R A Kowalski, and F Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence Journal*, 93:63–101, 1997.

[Bourguet, 2010] Jean-Rémi Bourguet. *Contribution aux méthodes d'argumentation pour la prise de décision. Application à l'arbitrage au sein de la filière céréalière.* PhD thesis, Université Montpellier II, 2010.

[Brewka and Eiter, 1999] Gerhard Brewka and Thomas Eiter. Prioritizing default logic: Abridged report. In *In Festschrift on the occasion of Prof.Dr. W. Bibel's 60th birthday*. Kluwer, 1999.

[Carneades, website] Carneades.
http://carneades.berlios.de/, website.

[Chein and Mugnier, 2009] Michel Chein and Marie-Laure Mugnier. *Graph-based Knowledge Representation and Reasoning. Computational Foundations of Conceptual Graphs*. Springer, Advanced Information and Knowledge Processing Series, London, 2009.

[de Moor *et al.*, 2009] Aldo de Moor, Jack Park, and Madalina Croitoru. Argumentation Map Generation with Conceptual Graphs: the Case for ESSENCE. In *Proc. of the 4th ICCS Conceptual Structures Tool Interoperability Workshop (CS-TIW 2009)*, pages 58–69, Russia, 2009.

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

[G. Brewka, 2007] M. Truszczynski G. Brewka, I. Niemel. Nonmonotonic reasoning. In F. van Harmelen V. Lifschitz, B. Porter, editor, *Handbook of Knowledge Representation*, pages 239–284. Elsevier, 2007.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Salvat and Mugnier, 1996] E. Salvat and M-L. Mugnier. Sound and complete forward and backward chaining of graph rules. In *Proc of ICCS 1996: Conceptual Structures: Knowledge Representation as Interlingua*, volume 1115, 1996.

[Sowa, 1984] J. F. Sowa. *Conceptual Structures: Information Proc. in Mind and Machine*. Addison–Wesley, 1984.