Semantic Argumentation using Rewriting Systems

Esteban Guerrero
Department of Computing Science, Umeå University, Sweden
esteban.guerrero@umu.se

Juan Carlos Nieves
Department of Computing Science, Umeå University, Sweden
juan.carlos.nieves@umu.se

In this article, we introduce a general framework for *structured argumentation* providing consistent and well-defined *justification* for conclusions that *can* and *cannot* be inferred and there is certainty about them, which we call semantic and NAF-arguments, respectively. We propose the so-called *semantic argumentation* guaranteeing well-known principles for quality in structured argumentation, with the ability to generate semantic and NAF-arguments, those where the *conclusion* atoms are semantically interpreted as *true*, and those where the conclusion is assumed to be *false*. This framework is defined on the set of all logic programs in terms of *rewriting systems* based on a *confluent set* of *transformation rules*, the so-called *Confluent Logic Programming Systems*, making this approach a general framework. We implement our framework named *semantic argumentation solver* available open source.

1 Motivation

We propose a general argument construction based on the partial interpretation of programs using different *families* of *logic programming semantics* induced by *rewriting systems functions* [6]. *Rewriting rules* are used to replace parts of a logic program based on the concept of a *normal form*, which is the least expression of a program that cannot be rewritten any further [9]. For example, having a program with the only rule: $innocent(x) \leftarrow not\ guilty(x)$, current structured argumentation approaches [10] generate the only consistent argument: $\langle \underbrace{\{innocent(X) \leftarrow\ not\ guilty(x)\}}_{Support}, \underbrace{innocent(x)}_{Conclusion} \rangle$, expressing that person x is innocent if x can not be proved guilty. However, in domain applications that need the generation of argument-based *reason explanations*, providing structured and well-defined reasons why x is not guilty ($not\ guilty(x)$) are needed. We emphasize the role of investigating such computational mechanisms that can also build arguments justifying conclusions based on the atoms that are inferred as *false*, *e.g.*, to state that *there is* <u>*certainty*</u> *in affirming that the guiltiness of x is false (there is no evidence), therefore the x must be innocent*, *i.e.*, $\langle \underbrace{\{innocent(X) \leftarrow\ not\ guilty(x)\}}_{Support}, \underbrace{not\ guilty(x)}_{Conclusion} \rangle$. These types of arguments have been less explored in the formal argumentation theory, except for *assumption-based argumentation* (ABA) [8].

2 Syntax and semantics

We use propositional logic with the following connectives $\wedge, \leftarrow$, *not*, and $\top$ where $\wedge$, and $\leftarrow$ are 2-place connectives, *not* and $\top$. The negation symbol *not* is regarded as the so-called *negation as failure* (NAF).

We follow standard logic programming syntax, *e.g.*, [5], for lack of space we do not include some basic and well-established syntax.

An *interpretation* of the signature $\mathscr{L}_P$ is a function from $\mathscr{L}_P$ to {false,true}. A *partial interpretation* of $\mathscr{L}_P$, are the sets $\langle I_1, I_2\rangle$ where $I_1 \cup I_2 \subseteq \mathscr{L}_P$. We use $\mathtt{SEM}(P) = \langle P^{true}, P^{false}\rangle$, where $P^{true} := \{p |\ p \leftarrow \top \in P\}$ and $P^{false} := \{p |\ p \in \mathscr{L}_P \backslash \mathtt{HEAD}(P)\}$. $\mathtt{SEM}(P)$ is also called model of P [6]. We use three value semantics that are characterized by ***rewriting systems*** following a set of **Basic Transformation Rules** for **Rewriting Systems** (see details in [6]), those rules are named: **RED+**, **RED-**, **Success**, **Failure**, **Loop**, **SUB**, and **TAUT**. Then, two rewriting systems ($\mathscr{CS}$) can be defined based on the previous basic transformations: $CS_0 = \{RED^+, RED^-, Success, Failure, Loop\}$, induces the WFS [3]. $CS_1 = CS_0 \cup \{SUB, TAUT, LC\}$, induces WFS+ [6]. The *normal form* of a normal logic program P with respect to a rewriting system $\mathscr{CS}$ is denoted by $norm_{\mathscr{CS}}(P)$. Every rewriting system $\mathscr{CS}$ induces a 3-valued logic semantics $\mathtt{SEM}_{\mathscr{CS}}$ as $\mathtt{SEM}_{\mathscr{CS}}(P) := \mathtt{SEM}(norm_{\mathscr{CS}}(P))$. To simplify the presentation, we use the entailment $\models_{\mathtt{SEM}_{\mathscr{CS}}}$ applied to a logic program P is defined by $\mathtt{SEM}_{\mathscr{CS}}(P) = \langle T, F\rangle$ in which $P \models_{\mathtt{SEM}^T_{\mathscr{CS}}} a$ if and only if $a \in T$, similarly, if $P \models_{\mathtt{SEM}^F_{\mathscr{CS}}} a$ if and only if $a \in F$. We use the entailment $\models_{\mathtt{SEM}_{\mathscr{CS}_0}}$ and $\models_{\mathtt{SEM}_{\mathscr{CS}_1}}$ for confluent rewriting system CS_0 and CS_1 respectively; and the form $\models_{\mathtt{SEM}_{\mathscr{CS}}}$ to indicate that any rewriting system can be used.

3 Semantic and NAF-arguments

Let us introduce a formal definition of semantic arguments.

Definition 1 (Semantic argument) *Given a normal logic program P and $S \subseteq P$. $Arg_P = \langle S,\ g\rangle$ is a* ***semantic argument*** *under $\mathtt{SEM}_{\mathscr{CS}}$ w.r.t. P, if the following conditions hold true:*

1. $S \models_{\mathtt{SEM}^T_{\mathscr{CS}}} g$
2. *S is minimal w.r.t. the set inclusion satisfying 1.*

We simplify the notation of these semantic arguments as $\mathscr{A}rg^+$. Condition 1 states that the interpretation of conclusion g is *true w.r.t.* T in $\mathtt{SEM}_{\mathscr{CS}}(S)$. Condition 2 in Definition 1 guarantees the support minimality.

Now, let us define NAF-arguments as follows:

Definition 2 (NAF-arguments) *Given a normal logic program P and $S \subseteq P$. $Arg_P = \langle S,\ not\ g\rangle$ is a* ***NAF-argument*** *under the $\mathtt{SEM}_{\mathscr{CS}}$ w.r.t. P, if the following conditions hold true:*

1. $S \models_{\mathtt{SEM}^F_{\mathscr{CS}}} g$,
2. *S is minimal w.r.t. the set inclusion satisfying 1.*

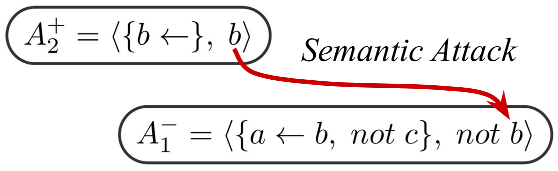
Condition 1 in Definition 2 is the interpretation of the conclusion *w.r.t.* $\models_{\mathtt{SEM}^F_{\mathscr{CS}}}$, with the set of all the NAF-arguments noted as $\mathscr{A}rg^-$. The addition of *not* in the conclusion of a NAF-argument stresses that such an atom is interpreted as false by $\mathtt{SEM}_{\mathscr{CS}}$.

Example 1 *Let us consider a program P_3 for building semantic and NAF-arguments considering $\mathscr{CS}_0$ and $\mathscr{CS}_1$.*

We build semantic and NAF-arguments as follows: 1) get related clauses of atoms (S_i); 2) for every related clause compute $\mathtt{SEM}_{\mathscr{CS}_0}(S_i)$ and $\mathtt{SEM}_{\mathscr{CS}_1}(S_i)$; 3) the support (every S_i) is joined to the conclusion[1]*. Then, the following sets of arguments are built considering $\mathscr{CS}_0$ and $\mathscr{CS}_1$:*

[1]We implemented this procedure and the sources are open, then can be found in `https://people.cs.umu.se/~esteban/argumentation/`

$$P_3 := \left\{ \begin{array}{l} a \leftarrow b, not\ c \\ b \leftarrow \\ d \leftarrow not\ d \\ e \leftarrow e \end{array} \right\}$$


$A_2^+ = \langle\{b \leftarrow\},\ b\rangle$
Semantic Attack
$A_1^- = \langle\{a \leftarrow b,\ not\ c\},\ not\ b\rangle$

Case $\mathscr{CS}_0$: $\mathscr{A}rg_{P_3} = \{A_1^+, A_2^+, A_3^+, A_1^-, A_2^-, A_3^-, A_4^-, A_6^-\}$.
Case $\mathscr{CS}_1$: $\mathscr{A}rg_{P_3} = \{A_1^+, A_2^+, A_3^+, A_5^+, A_1^-, A_2^-, A_3^-, A_4^-, A_6^-\}$.

An effect of interpreting argument supports under $\mathtt{SEM}_{\mathscr{CS}}$ is that some atoms (or sets of them) are evaluated in *opposition* to other arguments (*e.g.*, $A_1^+ = \langle S_2, a\rangle$ and $A_1^- = \langle S_1,\ not\ a\rangle$ in Example 1), suggesting a *semantic attack* relationship.

Definition 3 (Semantic attack) *Let* $A = \langle S_A, a\rangle \in \mathscr{A}rg^+$, $B = \langle S_B, not\ b\rangle \in \mathscr{A}rg^-$ *be two semantic arguments where* $\mathtt{SEM}_{\mathscr{CS}}(S_A) = \langle T_A, F_A\rangle$ *and* $\mathtt{SEM}_{\mathscr{CS}}(S_B) = \langle T_B, F_B\rangle$. *We say that A attacks B if* $x \in T_A$ *and* $x \in F_B$, *denoted* $\mathtt{attacks}(x,y)$.

Lemma 1 *Semantic and NAF-arguments built from any normal logic program are always consistent.*

Definition 4 (Semantic Argumentation Framework (SAF)) *Let P be a normal program. Then, a **semantic argumentation framework** is the tuple:* $SAF_P = \langle \mathscr{A}rg_P, \mathscr{A}tt\rangle$

We can straightforward extend the definitions of *argumentation semantics* in [7] as follows:

Definition 5 *Let* $SAF_P = \langle \mathscr{A}rg_P, \mathscr{A}tt\rangle$ *be a semantic argumentation framework. An admissible set of arguments* $S \subseteq AR$ *is:*

- stable *if and only if S attacks each argument which does not belong to S.*
- preferred *if and only if S is a maximal (*w.r.t. *inclusion) admissible set of AF.*
- complete *if and only if each argument, which is acceptable with respect to S, belongs to S.*
- grounded *if and only if S is the minimal (*w.r.t. *inclusion)* complete *extension of AF*[2].

Example 2 *Let us consider* $P_5 = \{a \leftarrow not\ b;\ b \leftarrow not\ a;\ c \leftarrow not\ c, not\ a;\ d \leftarrow not\ d, not\ b;\}$. $\mathtt{SEM}_{\mathscr{CS}}$ *will remove rules involving atoms c and d. Then, applying Definition 4, we have the framework:* $SAF_{P_5} = \langle\{A_6^- = \langle\{a \leftarrow not\ b\},\ not\ a\rangle, A_6^+ = \langle\{b \leftarrow not\ a\},\ b\rangle,\ A_5^- = \langle\{b \leftarrow not\ a\},\ not\ b\rangle,\ A_5^+ = \langle\{a \leftarrow not\ b\},\ a\rangle\},\ \mathtt{attacks}(A_5^+, A_6^+),\ \mathtt{attacks}(A_5^+, A_6^-),\ \mathtt{attacks}(A_6^+, A_5^+),\ \mathtt{attacks}(A_6^+, A_5^-)\rangle$. *When we apply Definition 5 to* SAF_{P_5} *we obtained the following extensions:*

- *Stable = preferred:* $\{\{A_5^+, A_5^-\},\ \{A_6^+, A_6^-\}\}$
- *Complete:* $\{\{A_5^+, A_5^-\},\ \{A_6^+, A_6^-\},\ \{\}\}$
- *Grounded:* $\{\}$

4 Conclusions

The main contributions are: 1) *Semantic Argumentation Frameworks* (SAF) can be used for justifying *true* and *false* interpreted conclusions. 2) SAF is based on *families* of rewriting confluent systems. 3) Satisfies all the well-known argumentation postulates [1, 4]. Future work will involve the exploration of our framework under other Confluent Logic Programming Systems, the satisfaction of other argumentation principles, and the investigation of commonalities between ABA and semantic argumentation.

[2]In [2] it is shown that grounded can be characterized in terms of complete extensions.

References

[1] Leila Amgoud (2014): *Postulates for logic-based argumentation systems*. International Journal of Approximate Reasoning 55(9), pp. 2028–2048, doi:10.1016/j.ijar.2013.10.004.

[2] Pietro Baroni, Martin Caminada & Massimiliano Giacomin (2011): *An introduction to argumentation semantics*. The knowledge engineering review 26(4), pp. 365–410, doi:10.1017/S0269888911000166.

[3] Stefan Brass, Ulrich Zukowski & Burkhard Freitag (1997): *Transformation-based bottom-up computation of the well-founded model*. In: Non-Monotonic Extensions of Logic Programming, Springer, pp. 171–201, doi:10.1007/BFb0023807.

[4] Martin Caminada & Leila Amgoud (2007): *On the evaluation of argumentation formalisms*. Artificial Intelligence 171(5-6), pp. 286–310, doi:10.1016/j.artint.2007.02.003.

[5] Jürgen Dix (1995): *A Classification Theory of Semantics of Normal Logic Programs: I. Strong Properties*. Fundam. Inform. 22(3), pp. 227–255, doi:10.3233/FI-1995-2234.

[6] Jürgen Dix, Mauricio Osorio & Claudia Zepeda (2001): *A general theory of confluent rewriting systems for logic programming and its applications*. Annals of Pure and Applied Logic 108(1-3), pp. 153–188, doi:10.1016/S0168-0072(00)00044-0.

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

[8] Phan Minh Dung, Robert A. Kowalski & Francesca Toni (2009): *Assumption-Based Argumentation*. In: Argumentation in Artificial Intelligence, Springer, pp. 199–218, doi:10.1007/978-0-387-98197-0_10.

[9] Juan Carlos Nieves & Mauricio Osorio (2016): *Ideal extensions as logical programming models*. Journal of Logic and Computation 26(5), pp. 1361–1393, doi:10.1093/logcom/exu014.

[10] Henry Prakken (2010): *An abstract framework for argumentation with structured arguments*. Argument and Computation 1(2), pp. 93–124, doi:10.1080/19462160903564592.