Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Graphical Representation of Assumption-Based Argumentation

Claudia Schulz
claudia.schulz@imperial.ac.uk
Department of Computing
Imperial College London
London SW7 2AZ, UK

Abstract

Since Assumption-Based Argumentation (ABA) was introduced in the nineties, the structure and semantics of an ABA framework have been studied exclusively in logical terms without any graphical representation. Here, we show how an ABA framework and its complete semantics can be displayed in a graph, clarifying the structure of the ABA framework as well as the resulting complete assumption labellings. Furthermore, we show that such an ABA graph can be used to represent the structure and semantics of a logic program (LP), based on the correspondence between the semantics of a LP and an ABA framework encoding this LP.

1 Introduction and Background

Assumption-Based Argumentation (ABA) (Bondarenko et al. 1997) provides a way to represent knowledge in a machine-readable form and to reason about it in a human-understandable way, namely in argumentative terms. In an ABA framework, knowledge is represented as a set of rules made of atoms and default elements called *assumptions*, which are assumed to be true as long as the contrary cannot be proven to hold.

Example 1. Consider the ABA framework ABA with $\mathcal{R}$ the set of ABA rules, $\mathcal{A}$ the set of assumptions, and $\overline{\alpha}$ denoting the contrary of an assumption $\alpha \in \mathcal{A}$:

- $\mathcal{R} = \{k \leftarrow \pi;\ p \leftarrow \kappa;\ r \leftarrow \rho;\ r \leftarrow \rho, \kappa\}$
- $\mathcal{A} = \{\kappa, \pi, \rho\};\ \ \overline{\kappa} = k;\ \overline{\pi} = p;\ \overline{\rho} = r$

Reasoning in ABA is based on the concepts of *derivation* of a conclusion from a set of assumptions by applying the rules, and *attack* of an assumption α by a set of assumptions deriving the conclusion $\overline{\alpha}$, i.e. deriving the contrary of α.

Example 2. In ABA, $\{\pi\}$ (and any superset thereof) attacks κ because there is a derivation for k from $\{\pi\}$ and k is the contrary of κ. Furthermore, $\{\kappa\}$ (and any superset thereof) attacks π, and $\{\rho\}$ (and any superset thereof) attacks ρ.

The semantics of an ABA framework can then be determined by assigning one of the labels IN, OUT, or UNDEC to every assumption, where the label of an assumption depends on the label of the assumptions in attacking sets of assumptions (Schulz and Toni 2014). We are here focussing on the complete semantics: A labelling $LabAsm$ is a *complete assumption labelling* iff for each assumption α it holds that:

- if $LabAsm(\alpha) = \text{IN}$ then each set of assumptions attacking α contains some β such that $LabAsm(\beta) = \text{OUT}$;
- if $LabAsm(\alpha) = \text{OUT}$ then there exists a set of assumptions AP attacking α such that $AP \subseteq \text{IN}(LabAsm)$;
- if $LabAsm(\alpha) = \text{UNDEC}$ then each set of assumptions attacking α contains some β such that $LabAsm(\beta) \neq \text{IN}$, and there exists a set of assumptions AP attacking α such that $AP \cap \text{OUT}(LabAsm) = \emptyset$.

Example 3. ABA has three complete assumption labellings:

- $\text{IN}(LabAsm_1) = \{\kappa\},\ \text{OUT}(LabAsm_1) = \{\pi\},\ \text{UNDEC}(LabAsm_1) = \{\rho\}$;
- $\text{IN}(LabAsm_2) = \{\pi\},\ \text{OUT}(LabAsm_2) = \{\kappa\},\ \text{UNDEC}(LabAsm_2) = \{\rho\}$;
- $\text{IN}(LabAsm_3) = \emptyset,\ \text{OUT}(LabAsm_3) = \emptyset,\ \text{UNDEC}(LabAsm_3) = \{\kappa, \pi, \rho\}$.

Determining the complete assumption labellings of an ABA framework is not easy for humans as it involves considering all sets of assumptions and all attacks between them. We show how an ABA framework can be represented graphically, which facilitates to understand the structure of an ABA framework and to determine its complete assumption labellings.

2 ABA graphs

Given that the ABA semantics are based on considering all attacks between all sets of assumptions, the most intuitive representation of an ABA framework is a graph with all sets of assumptions as nodes, and edges between them indicating attacks, as illustrated in Fig. 1. However, the large amount of sets of assumptions and attacks makes this graph rather complicated and unclear.

Thus, a more useful representation only displays *argument-supporting* sets of assumptions, i.e. sets containing only those assumptions necessary for the derivation of a conclusion, and attacks between them. This simplified representation is equivalent to the previous one, since it has been proven that determining complete assumption labellings can be equivalently done considering all or only

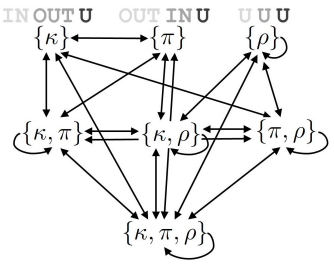

Figure 1: Attacks between all sets of assumption in ABA along with the three complete assumption labellings (see Example 3), indicated by the three differently coloured letters above the singleton sets, where "U" is shorthand for UNDEC.

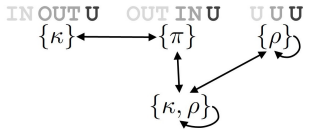

Figure 2: Attacks between argument-supporting sets of assumption in ABA and its complete assumption labellings.

argument-supporting sets of assumptions (Schulz and Toni 2015). As shown in Fig. 2, the simplified graph is considerably clearer than the previous one (compare Fig. 1).

Fig. 2 illustrates that, for example, in order to determine the label of π only $\{\kappa\}$ and $\{\kappa, \rho\}$ have to be examined, instead of also considering $\{\kappa, \pi\}$ and $\{\kappa, \pi, \rho\}$ (see Fig. 1).

3 ABA graphs for Logic Programs

Another technique for representing knowledge and reasoning about it in an efficient but arguably less human-understandable way is logic programming. Similarly to ABA a *logic program* (LP) is a set of rules made of atoms and default elements called *negation-as-failure* (NAF) literals, which are assumed to be true as long as their complementary atom cannot be proven to hold. The semantics of a LP are defined as tuples of true ($\mathcal{T}$) and false ($\mathcal{F}$) atoms which form a fixpoint of a function altering the initial LP (Baral and Gelfond 1994). Since this fixpoint definition can be difficult to retrace for humans, it is helpful to have a graphical representation of the structure of the LP. Two existing approaches are Extended Dependency Graphs, where every node represents the head of one rule, and Rule Graphs, where every node represents an atom (Costantini and Provetti 2010). We suggest a new approach where every node represents a possible derivation of an atom, displayed as the set of NAF literals necessary for this derivation and based on our ABA graphs.

Due to the similarity of representing knowledge in ABA and logic programming, it is straightforward to encode a LP in an ABA framework, where the ABA rules are formed by the LP and the ABA assumptions are formed by NAF literals (Bondarenko et al. 1997). Despite the different definitions of the semantics, it turns out that, for example, the 3-valued stable models of a LP (Przymusinski 1989) coincide with the complete assumption labellings of the encoding ABA framework (Schulz and Toni 2015). More precisely, an atom a is true (false) in a 3-valued stable model iff the NAF literal $not\ a$ is labelled OUT (IN resp.) in the corresponding complete assumption labelling.

Example 4. Let $\mathcal{P}$ be the following logic program:
$k \leftarrow not\ p\,;\ \ p \leftarrow not\ k\,;\ \ r \leftarrow not\ r\,;\ \ r \leftarrow not\ r, not\ k.$
$\mathcal{P}$ has three 3-valued stable models: $\langle \mathcal{T}_1 = \{p\}, \mathcal{F}_1 = \{k\}\rangle$, $\langle \mathcal{T}_2 = \{k\}, \mathcal{F}_2 = \{p\}\rangle$, $\langle \mathcal{T}_3 = \emptyset, \mathcal{F}_3 = \emptyset\rangle$. $\mathcal{P}$ has the same structure as $\mathcal{R}$ in ABA (Example 1), where $not\,k$ substitutes κ, $not\ p$ substitutes π, and $not\ r$ substitutes ρ, resulting in a direct correspondence between the 3-valued stable models of $\mathcal{P}$ and the complete assumption labellings of ABA.

Due to this correspondence, an ABA graph can represent the structure of a LP and its 3-valued stable models in terms of the encoding ABA framework. Fig. 3 displays the graphical representation of $\mathcal{P}$ in terms of the encoding ABA framework (structurally equivalent to ABA). Such a graph clarifies the structure of the respective LP, in particular regarding dependencies between literals. The graph can also be helpful to determine the 3-valued stable models of a LP in terms of complete assumption labellings.

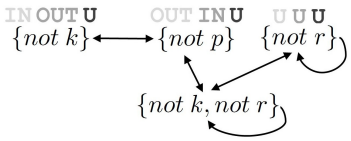

Figure 3: Graphical representation of $\mathcal{P}$ and its three 3-valued stable models (Example 4).

4 Conclusion

We introduced a graphical representation of the structure of ABA frameworks, which can help humans determine the complete assumption labellings of this framework. Furthermore, these graphs can be used as a representation of LPs by encoding a LP in an ABA framework and can help humans to determine the 3-valued stable models of a LP.

References

Baral, C., and Gelfond, M. 1994. Logic programming and knowledge representation. *The Journal of LP*.

Bondarenko, A.; Dung, P.; Kowalski, R.; and Toni, F. 1997. An abstract, argumentation-theoretic approach to default reasoning. *AI*.

Costantini, S., and Provetti, A. 2010. Graph representations of logic programs: properties and comparison. In *LANMR'10*.

Przymusinski, T. 1989. Every logic program has a natural stratification and an iterated least fixed point model. In *PODS'89*.

Schulz, C., and Toni, F. 2014. Complete assumption labellings. In *COMMA'14*.

Schulz, C., and Toni, F. 2015. Logic programming in assumption-based argumentation revisited – semantics and graphical representation. In *AAAI'15*.