Encoding Logic Programs in Assumption-Based Argumentation

Claudia Schulz and Francesca Toni
Imperial College London

Introduction

In a (normal) Logic Program (LP) knowledge is encoded as a set of *inference rules* made of *atoms* and *negation-as-failure (NAF) literals*, which are default elements assumed to hold as long as their complementary atom cannot be proven to hold. For example, *not a* denotes a NAF literal whose complementary atom is a. Similarly, in an *Assumption-Based Argumentation (ABA)* framework [1, 4, 15] knowledge is encoded as a set of inference rules, but made of *sentences* in some given underlying logical language. This language includes default elements, called *assumptions*, and each assumption is equipped with a *contrary* sentence. For example, $M(a \wedge b)$ may be an assumption whose contrary is $\neg(a \wedge b)$, with the underlying language that of classical logic extended with sentences of the form $M(s)$ where s is a sentence in classical logic ($M(s)$ stands for 's is consistent' and is used to capture default logic as an instance of ABA [1]). LPs are special instances of ABA frameworks [1], where sentences are atoms or NAF literals, assumptions are NAF literals, and the contrary of an assumption (NAF literal) is its complementary atom.

Even though the representation of knowledge in ABA frameworks resembles and generalises that of LPs and other non-monotonic reasoning formalism [1], the semantics of ABA frameworks is traditionally given in a completely different way from that of LPs, namely in dialectical terms, using a notion of *attack* between sets of assumptions (or between arguments that can be obtained from them, using the inference rules) [1, 4, 15]. When LPs are encoded as ABA frameworks, the dialectical semantics of the latter can be used to justify presence or absence of atoms and NAF literals in stable models of the encoded LP [13].

Alternatively, the semantics of ABA frameworks can be characterised in terms of labellings of assumptions as IN, OUT, or UNDEC [12, 2]. When LPs are encoded as ABA frameworks, various types of ABA labellings correspond to well-known semantics of LPs [2]. As a by-product of this correspondence, graphical representations of the structure and labellings of an ABA framework encoding a LP also visualises how the models of the encoded LP emerge from its structure [14].

In this short article we briefly explain and exemplify 1) how LPs are encoded as ABA frameworks and how their semantics correspond, (2) how the encoding can be used to provide justifications for (non-)membership in stable models via the dialectical formulation of ABA semantics, and (3) how the encoding can be used to provide a graphical representation of LPs and their semantics applying

the ABA semantics in terms of labellings.

(1) Encoding of LPs as ABA frameworks

We illustrate this encoding with an example. The following simple LP $\mathcal{P}_1$:

$$k \leftarrow not\ p \qquad\qquad p \leftarrow not\ k \qquad\qquad r \leftarrow not\ k$$

can be encoded as an ABA framework with

- inference rules: $\mathcal{P}_1$
- assumptions: $not\ p, not\ k, not\ r$
- p contrary of $not\ p$, k contrary of $not\ k$, r contrary of $not\ r$
- underlying language: $p,\ k,\ r,\ not\ p,\ not\ k,\ not\ r$

Note that in this and any ABA framework resulting from encoding a LP, contrary is "asymmetric", for example p is the contrary of $not\ p$ but $not\ p$ is not the contrary of p (because only assumptions have contraries, and p is not an assumption).

Here, the set of assumptions $\{not\ p\}$ attacks the assumptions $not\ k$, since the contrary k of the attacked assumption $not\ k$ can be derived from the attacking set $\{not\ p\}$ using the first inference rule in $\mathcal{P}_1$ (this derivation is denoted $\{not\ p\} \vdash k$ and called an *argument*). For the same reason $\{not\ p\}$, and any superset thereof, attacks all sets of assumptions containing $not\ k$, such as $\{not\ k\}$, $\{not\ k, not\ p\}$, and so on. Similarly, $\{not\ p\}$ attacks $not\ k$ (and any set containing it), $\{not\ k\}$ attacks $not\ r$ etc.

Given the encoding of LPs into ABA frameworks, several correspondence results hold between semantics for LPs and semantics for ABA frameworks, both in dialectical and labelling terms, as illustrated next separately for 2-valued and 3-valued semantics for LPs.

2-valued (Stable Model) Semantics

$\mathcal{P}_1$ has two stable models [8]: $\{k\}$ and $\{p, r\}$. Since stable models are 2-valued, the meaning of the first stable model, for instance, is clearly that k is true and both p and r are false.

In the encoding of $\mathcal{P}_1$ as an ABA framework, there are two *stable sets of assumptions*, $\{not\ p, not\ r\}$ and $\{not\ k\}$, where a stable set of assumptions does not attack itself and it attacks every assumption it does not contain [1]. For instance, $\{not\ p, not\ r\}$ is stable as it does not attack itself and it attacks $not\ k$, but $\{not\ p\}$ is not stable because it does not attack all assumptions not contained in it, in particular $\{not\ p\}$ does not attack $not\ r$. The first stable set of assumptions corresponds to the first stable model, in the sense that all elements of the stable model (just k in this case) can be derived from $\{not\ p, not\ r\}$ using inference rules in $\mathcal{P}_1$. An analogous correspondence holds between the second stable set of assumptions and the second stable model.

In labelling terms, there are two *stable labellings* [12, 2] of the ABA framework encoding $\mathcal{P}_1$, which are 2-valued in the sense that they only assign the labels IN and OUT: $\{(not\ k, \text{OUT}), (not\ p, \text{IN}), (not\ r, \text{IN})\}$ and $\{(not\ k, \text{IN}), (not\ p, \text{OUT}),$

(*not* r, OUT)}. In a stable labelling an assumption is labelled IN if and only if every set of assumption attacking it contains an assumption labelled OUT. So in the first stable labelling *not* p is labelled IN because all sets of assumptions attacking *not* p, i.e. {*not* k} and any superset thereof, contain an assumption labelled OUT (*not* k). Conversely, an assumption is labelled OUT if and only if some set of assumptions attacking it contains only assumptions labelled IN. So *not* k is labelled OUT because it is attacked by the set {*not* p} which contains only assumptions labelled IN. The first stable labelling corresponds to the first stable model in the sense that the assumptions labelled IN (*not* p and *not* r) coincide with the false atoms in the stable model (p and r); conversely, the assumptions labelled OUT (*not* k) coincide with the true atoms in the stable model (k). The same correspondence holds between the second stable labelling and the second stable model.

3-valued Semantics

Correspondences do not only hold between 2-valued LP and ABA semantics, but also between 3-valued ones, for example between *3-valued stable models* [9] of a LP and *complete sets/labellings* [1, 12] of the encoding ABA framework. To illustrate this correspondence, consider the LP $\mathcal{P}_2$:

$$k \leftarrow not\ p \qquad p \leftarrow not\ k \qquad r \leftarrow not\ r \qquad r \leftarrow not\ r, not\ k$$

which has three 3-valued stable models: $\langle\{p\},\{k\}\rangle$, $\langle\{k\},\{p\}\rangle$, and $\langle\emptyset,\emptyset\rangle$. Encoding $\mathcal{P}_2$ in an ABA framework yields three *complete sets of assumptions* (see [1] for the definition): {*not* k}, {*not* p}, and $\emptyset$. As in the case of the 2-valued semantics, all true atoms in a 3-valued stable model can be derived from the corresponding complete set of assumptions, for example p in the first 3-valued stable model can be derived from the first complete set of assumptions {*not* k}. In addition, the false atoms in a 3-valued stable model (k) coincide with the assumptions in the corresponding complete set of assumptions (*not* k).

In labelling terms, the ABA framework encoding $\mathcal{P}_2$ has three complete labellings [12]: {(*not* k, IN), (*not* p, OUT), (*not* r, UNDEC)}, {(*not* k, OUT), (*not* p, IN), (*not* r, UNDEC)}, and {(*not* k, UNDEC), (*not* p, UNDEC), (*not* r, UNDEC)}. Here, there is not only a correspondence between true and false atoms in the models of the LP and IN and OUT assumptions in the labellings of the encoding ABA framework, but also between undefined atoms in the models and UNDEC assumptions in the labellings. The previously observed correspondence between 2-valued semantics is thus extended to the 3-valued case as follows [14]:

- a is true in the (3-valued stable) model iff *not* a is labelled OUT by the (complete) labelling;
- a is false in the (3-valued stable) model iff *not* a is labelled IN by the (complete) labelling; and
- a is undefined in the (3-valued stable) model iff *not* a is labelled UNDEC by the (complete) labelling.

The same correspondence also holds between L-stable models [6] of a LP and semi-stable sets/labellings of the encoding ABA framework [14], well-founded

models [7] and grounded sets/labellings [2], and regular models [16] (equivalently preferred extension [3] and maximal stable models [10]) and preferred sets/labellings [2].

(2) Justification of LPs under stable semantics

The correspondence of semantics between LPs and their encoding ABA frameworks allows us to apply techniques developed for ABA frameworks to LPs. For example, a dialectical proof procedure for ABA frameworks has recently been adapted to explain why a literal is or is not contained in a stable model of a LP [13]. Considering again $\mathcal{P}_1$ and its stable models, the aforementioned procedure can be used to construct two different types of justifications as to why, for instance, r is contained in the second stable model $\{p, r\}$ of $\mathcal{P}_1$, one based on the interaction between derivations (*arguments* in ABA) as in Figure 1 (left) and the other one on the interaction between atoms and NAF literals as in Figure 1 (right). These justifications exhibit an important property, namely they explain a literal in terms of an admissible fragment [5] of the respective stable model.

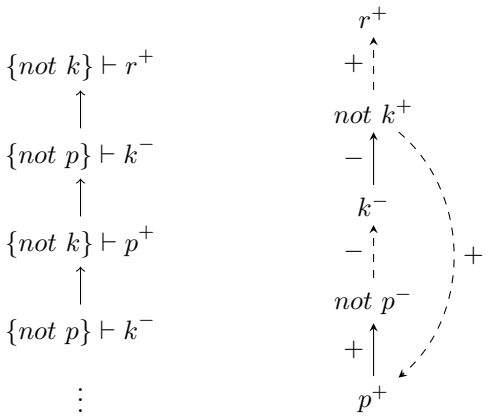

Figure 1: Justifications as to why r is contained in the second stable model of $\mathcal{P}_1$ based on interactions between arguments (left) and literals (right).

The explanation on the left of Figure 1 expresses that r is in the stable model, indicated by the $+$, because there is an argument $\{not\ k\} \vdash r$. This argument is *attacked* by an argument $\{not\ p\} \vdash k$, but k is not in the stable model, indicated by the $-$, so we can think of the attack as "not succeeding". The reason that k is not in the stable model is that the argument for k is again attacked, namely by an argument $\{not\ k\} \vdash p$ where p is in the stable model (indicated by $+$), so this attack "succeeds". The reason that p is in the stable model is that the argument for k attacks the argument for p, and so on.

The explanation on the right of Figure 1 expresses that r is in the stable model (r^+) because it is *supported* (dashed arrow) by the NAF literal $not\ k$ which is true with respect to the stable model ($not\ k^+$), so the support "succeeds" ($+$ on the dashed arrow). The reason that $not\ k$ is true is that even though it is *in conflict* (solid arrow) with its complementary atom k, k is not in

the stable model (k^-) and therefore the conflict "does not succeed" ($-$ on solid arrow). The rest of the justification can be read analogously.

(3) Visualising LPs and their semantics

The attack relation of an ABA framework can be displayed as a graph providing valuable information about the structure of the ABA framework. Since LPs can be encoded in ABA frameworks, the same graphical representation can also be used to visualise important structural features of LPs, in particular conflicts between literals [14].

Figure 2 shows the graphical representation of the ABA framework encoding $\mathcal{P}_2$. The three differently coloured labels above the assumptions indicate the labels of these assumptions in the three complete labellings of the ABA framework.

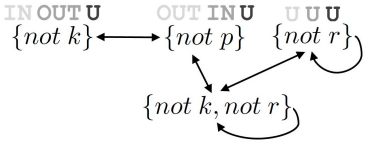

Figure 2: Attacks between sets of assumption in the ABA framework encoding $\mathcal{P}_2$ and its three complete labellings.

This representation provides some important insights about the structure of $\mathcal{P}_2$: Firstly, *not* k and *not* p are in a semantic conflict, meaning that they can never both be true in a 3-valued stable model. In other words, if k is true then p is false, and vice versa. Thus, the graph provides an intuitive explanation as to why k (respectively p) is true in one of the 3-valued stable models, but not in the other. Secondly, *not* r is in conflict with itself. Applying the same intuitive meaning as before, this means that if r is true then r has to be false, and vice versa, which of course is a contradiction. Consequently, r can only be undefined with respect to any 3-valued stable model. Thus, the graph again provides an explanation for the truth value of an atom.

Conclusion

In this article we demonstrated the correspondence between various semantics of a LP and the semantics of an ABA framework encoding the LP. This correspondence has proven useful to apply techniques developed for ABA frameworks to LPs, such as justifying (non)membership of literals in stable models and visualising LPs and their semantics. Linked to the justification of stable models is also the debugging of inconsistent LPs under the stable model or answer set semantics. Recently, different types of inconsistencies in LPs have been characterised [11] as a first step towards explaining why an inconsistency arises. We hope that based on that, explanations may be constructed using ABA techniques similar to the ones used for justifications. In future work it will also be interesting to see whether the correspondence between LPs and ABA frameworks goes beyond

normal LPs, i.e. whether the correspondence persists when dealing with LPs containing constraints, disjunction, or aggregates.

References

[1] Andrei Bondarenko, Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 1997.

[2] Martin Caminada and Claudia Schulz. On the equivalence between assumption-based argumentation and logic programming. In *ArgLP'15*, 2015.

[3] Phan Minh Dung. Negations as hypotheses: An abductive foundation for logic programming. In *ICLP'91*, 1991.

[4] Phan Minh Dung, Robert A. Kowalski, and Francesca Toni. Assumption-based argumentation. In *Argumentation in Artificial Intelligence*. Springer US, 2009.

[5] Phan Minh Dung and Phaiboon Ruamviboonsuk. Well-founded reasoning with classical negation. In *LPNMR'91*, 1991.

[6] Thomas Eiter, Nicola Leone, and Domenico Saccà. On the partial semantics for disjunctive deductive databases. *Annals of Mathematics and Artificial Intelligence*, 19(1-2):59–96, 1997.

[7] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.

[8] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP'88/SLP'88*, 1988.

[9] Teodor C. Przymusinski. Every logic program has a natural stratification and an iterated least fixed point model. In *PODS'89*, 1989.

[10] Domenico Saccà and Carlo Zaniolo. Stable models and non-determinism in logic programs with negation. In *PODS'90*, 1990.

[11] Claudia Schulz, Ken Satoh, and Francesca Toni. Characterising and explaining inconsistency in logic programs. In *LPNMR'15*, 2015.

[12] Claudia Schulz and Francesca Toni. Complete assumption labellings. In *COMMA'14*. IOS Press, 2014.

[13] Claudia Schulz and Francesca Toni. Justifying answer sets using argumentation. *Theory and Practice of Logic Programming*, FirstView, 2 2015.

[14] Claudia Schulz and Francesca Toni. Logic programming in assumption-based argumentation revisited - semantics and graphical representation. In *AAAI'15*. AAAI Press, 2015.

[15] Francesca Toni. A tutorial on assumption-based argumentation. *Argument & Computation, Special Issue: Tutorials on Structured Argumentation*, 5(1):89–117, 2014.

[16] Jia-Huai You and Li-Yan Yuan. Three-valued formalization of logic programming: Is it needed? In *PODS'90*, 1990.