Contradiction in Argumentation Frameworks

H. Jakobovits
Dept. of Computer Science
Free University of Brussels, VUB
hadassa@star1.vub.ac.be

D. Vermeir
Dept. of Computer Science
Free University of Brussels, VUB
dvermeir@vub.ac.be

Abstract

We present a theory of argumentation that can deal with contradiction within an argumentation framework, thus solving a problem posed in [1]. By representing logic programs as sets of interacting arguments, we can apply our results for general argumentation frameworks to logic-programming semantics. This yields a new semantics for logic programs that properly extends traditional approaches such as stable [2] and well-founded [3] models.

Keywords: Argumentation framework, logic program, semantics

1 Introduction

The challenge of understanding argumentation and its role in human reasoning has been addressed by many researchers in different fields including philosophy, logic and AI.

A promising formal theory of argumentation has recently been proposed in a seminal article by Dung[1]. His theory is based on the idea that a statement (argument) is believable if it can be argued successfully against contesting arguments. Thus, the theory considers so-called argumentation frameworks that model the *interactions* between different arguments, while abstracting from the meaning or internal structure of an argument.

Within a given framework of interacting arguments, there might be one, or more, sets of conclusions, called *extensions*, that are deemed acceptable. The selected set(s) must satisfy certain criteria, such as consistency, coherence, etc. These criteria have been successfully formalized in a rigorous manner, in [1]. However, as the author points out, the theory lacks facilities to satisfactorily deal with contradictory arguments.

In this paper, we present an alternative theory of acceptability in argumentation frameworks, which solves this problem. In addition, our approach unifies several concepts from [1] (notably grounded and stable extensions), using the single simple concept of "labelling".

In [1], the power of argumentation frameworks was demonstrated by showing that several systems for nonmonotonic reasoning (including logic programming) can be mapped to argumentation frameworks in a natural way. Such a mapping is obtained by interpreting arguments, and their interactions, in terms of the target system.

In this paper, we define such a mapping for logic programs. By representing logic programs as sets of interacting arguments, we can apply our results for general argumentation frameworks to logic-programming semantics. This yields a new semantics for logic programs that properly extends traditional approaches such as stable [2] and well-founded [3] models.

The proofs of all theorems in this paper can be found in [4] and [5].

2 Argumentation Frameworks

According to [1], in order for an extension to be acceptable, it must be able to defend itself against any criticisms, as follows:

Definition 1 *An* **argumentation framework** *is a pair, $AF = (A, \rightsquigarrow)$, where A is a set of arguments, and $\rightsquigarrow$ is a binary relation on A, i.e. $\rightsquigarrow \subseteq A \times A$. An argument a is said to* **attack** *an argument b, denoted $a \rightsquigarrow b$, if $(a, b) \in \rightsquigarrow$. We say that a set S of arguments attacks another set T, denoted $S \rightsquigarrow T$, if there is an argument in S which attacks an argument in T. $a \not\rightsquigarrow b$ means that a does not attack b. A set S of arguments is* **consistent** *iff there are no arguments $a, b \in S$ such that $a \rightsquigarrow b$. An argument a is* **defended** *by a set S of arguments iff any argument $b \in A$ which attacks a, is attacked by S. A consistent set S of arguments is* **admissible** *iff each argument in S is defended by S.*

The only admissible extension of the framework in Figure 1 is the empty set. Indeed, since the argument a attacks itself, the set $\{a\}$ is inconsistent. Any admissible set, by definition, must be consistent; therefore, $\{a\}$ cannot be a subset of an admissible set. Similarly, the set $\{b, c\}$ cannot be a subset, since it is inconsistent. Neither of the singletons $\{b\}$ or $\{c\}$ are admissible since they each have an attacker which they do not counterattack. In other words, $\{b\}$ is

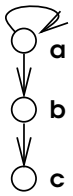


Figure 1: An argumentation framework

invalidated by its attack from a, and $\{c\}$ is invalidated by its attack from b.

We have the following criticism of the above reasoning: if a invalidates $\{b\}$, why does it not invalidate the attack of b on c? In our opinion, the set $\{c\}$ should be a "legal" extension, or at least an extension that receives some recognition. We feel that there is a limitation in previous approaches which stems from a bias in the definition of admissible sets. According to Definition 1, either an argument is included in an admissible set S, in which case its attacks are valid, and so can be used to defend elements of S, or it is not included in the admissible set and then its attacks cannot be used as a defence at all.

We feel that a theory of argumentation should recognize the existence of the attack on b by a, by adding an extension in which b *and* its attack on c are invalidated. A theory of argumentation should be general enough to include both extensions of the argumentation framework in Figure 1, the empty set and the set $\{c\}$. Once both of these extensions are admitted, one can discuss which extension should be chosen under various circumstances.

With this motivation, we relax the notion of a defence, by allowing a defending set T to be inconsistent, as follows:

Definition 2 *A set T of arguments is a* **supporting defence** *of a consistent set of arguments S iff the following conditions hold:*

1. $\forall a \in A$ *if* $a \rightsquigarrow T \cup S$ *then* $T \cup S \rightsquigarrow a$.
2. $S \not\rightsquigarrow T$.
3. $T \not\rightsquigarrow S$.
4. $S \cap T = \emptyset$.

A consistent set S of arguments in AF is **defendable** *iff there is a set T which is a supporting defence of S.*

Example 1 In the argumentation framework of Figure 1, the set $\{a\}$ is a supporting defence of the set $\{c\}$. Thus, the set $\{c\}$ is defendable. Notice that the defence attacks itself, since the argument a is self-contradicting.

3 Labellings

We now introduce the basic definition of our theory of argumentation, in which we associate a symbol to each argument in the framework. This mapping defines a three-valued extension: those arguments that are assigned the symbol "+" are accepted, those that are assigned the symbol "-" are rejected, and those that are assigned the symbol "$\pm$" are left undecided.

Definition 3 *Let $(A, \rightsquigarrow)$ be an argumentation framework. A* **labelling** *is a mapping*

$$l : A \rightarrow 2^{\{+,-\}}$$

that satisfies the following conditions:

1. *if* $- \in l(a)$ *then* $\exists b \rightsquigarrow a \ : \ + \in l(b)$
2. *if* $+ \in l(a)$ *then* $\forall b \rightsquigarrow a \ : \ - \in l(b)$
3. *if* $+ \in l(a)$ *then* $\forall c : a \rightsquigarrow c \Rightarrow - \in l(c)$
4. $\forall a \in A \ : \ l(a) \neq \emptyset$

Labellings of argumentation frameworks are not necessarily unique, as is demonstrated by the following example:

Example 2 Figure 2 gives the labellings of the argumentation framework shown in Figure 1.

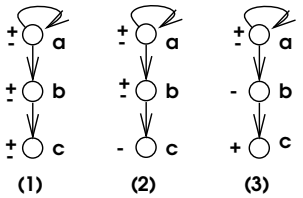


Figure 2

In our theory of argumentation, the acceptability of extensions is determined by labellings, as follows:

Definition 4 *A set S of arguments is an* **acceptable extension** *iff there is a labelling l such that $S = \{a \mid l(a) = +\}$. In this case the set $\{a \mid l(a) = \pm\}$ is called an* **undecided set with respect to** S.

The correspondence between acceptable extensions and defendable sets is given in the following theorem:

Theorem 1 *Let AF be an argumentation framework.*

- *If a set S of arguments is an acceptable extension then S is defendable, and an undecided set with respect to S is a supporting defence of S.*
- *If a set S of arguments is maximal defendable (i.e. S is defendable and there is no defendable set S' such that $S \subset S'$) then S is an acceptable extension. In this case the maximal supporting defence of S is an undecided set with respect to S.*

Example 3 Consider the labellings of example 2. The labelling (3) shows that the set $\{c\}$ is defendable. The labellings (1) and (2) both correspond to the defendable set $\emptyset$.

It is clear in the above argumentation framework that there are some arguments which must be undecided, due to the cycle with an odd number of nodes. The labelling (1) is the one that propagates this uncertainty as much as possible, since it allows a problem of inconsistency which lies in one part of the framework, to affect the information available in the whole framework. This labelling captures the fact that there is a special kind of inconsistency in the framework, and corresponds to the only admissible set (according to definition 1) for this argumentation framework, the set $\emptyset$. The labelling (3) limits the uncertainty propagation as much as possible, thus providing information about the truth value of the argument *c*.

4 Robust Labellings

In this section we present a criterion by which to choose those labellings that respect the stability of the decided arguments. When an extension is proposed, the so-called "restricted argumentation framework" consists of the arguments which have not yet been decided (i.e. the arguments a such that $l(a) \neq \pm$), with their interactions. The information contained in this restricted argumentation framework may permit further conclusions to be added to the decided arguments (i.e. the arguments a such that $l(a) = \pm$). In this case, the set of arguments in the original extension may or may not be compatible with the new added conclusions, in that the union of their decided arguments may or may not correspond to a labelling. We shall call a labelling l **robust** if the decided arguments remain unchanged, even if some undecided arguments become decided. This monotonicity of the deciding process is captured in the following recursive definition:

Definition 5 *Let l be a labelling of $(A, \rightsquigarrow)$, and $(A\,|_{\{a|l(a)=\pm\}}, \rightsquigarrow|_{\{a|l(a)=\pm\}})$ be the argumentation framework restricted to $\{a \mid l(a) = \pm\}$. The labelling l is* **robust** *iff for any robust labelling l' of $(A\,|_{\{a|l(a)=\pm\}}, \rightsquigarrow|_{\{a|l(a)=\pm\}})$, $l \dagger l'$ is a labelling of $(A, \rightsquigarrow)$, where $l \dagger l'$ is defined as follows:*

$$l \dagger l'(a) = \begin{cases} l'(a) & \text{if } l(a) = \pm \\ l(a) & \text{otherwise} \end{cases}$$

Example 4 The labelling (3) in Figure 2 is robust, since the only labelling of the argumentation framework restricted to the undecided set $\{a\}$ is the one given. Similarly, the labelling of the set $\{a, b\}$ shown in Figure 3 is robust. The

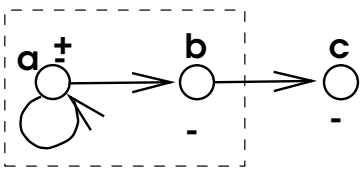


Figure 3

labelling (2) in Figure 2, however, is not robust since it is not compatible with the robust labeling of the undecided set $\{a, b\}$ shown in Figure 3. Indeed, c violates condition (1) in definition 3, so the mapping of Figure 3 is not a labelling.

We have shown that an argumentation framework AF has a defendable set of arguments only if there is a labelling of AF. In order for our argumentation theory to be reasonable, it should always allow the empty set to be defendable. The following theorem states a further result, and will be useful in the next section:

Theorem 2 *Every argumentation framework has a robust labelling.*

5 Logic Programs as Argumentation Frameworks

In this section we show how to associate an argumentation framework to a logic program. We then use the labellings of the argumentation framework to determine a semantics for the program.

Definition 6 *A* **logic program** *is a set of rules of the form*

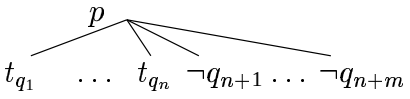
$$p \leftarrow q_1, ..., q_m$$

where p is an atom and $q_1, ..., q_m$ are literals, i.e. atoms or negated atoms. The conclusion, p, is called the head and the subgoals $q_1, ..., q_m$ are called the body. The arrow may be read "if".

The following definition is necessary in order to be able to define the associated argumentation framework:

Definition 7 *Let P be a logic program and let p be a positive literal in the Herbrand base $\mathcal{B}_P$ [7]. Then T_p is the set of proof trees for p, where a* **proof tree** *t_p is defined recursively as follows:*

1. *If there is a rule $p \leftarrow$ in P then the following tree is a proof tree for p:*
$$p$$
2. *If $\neg q$ is a negative literal in $\neg\mathcal{B}_P$ then the following tree is a proof tree for $\neg q$:*
$$\neg q$$
3. *Suppose there is a rule*
$$p \leftarrow q_1, \ldots, q_n, \neg q_{n+1}, \ldots, \neg q_{n+m}$$
in P, where
$$q_1, \ldots, q_n \in \mathcal{B}_P$$
$$\neg q_{n+1}, \ldots, \neg q_{n+m} \in \neg\mathcal{B}_P$$
and let $t_{q_1}, \ldots, t_{q_n}$ be proof trees for $q_1, \ldots, q_n$. Then the following is a proof tree for p :



Definition 8 *Let P be a logic program. The* **argumentation framework associated to** *P, denoted AF_P, is defined as follows:*

$$AF_P = (\bigcup_{p \in \mathcal{B}_P} T_p, \rightsquigarrow),$$

where an argument t_p attacks an argument t_q iff $\neg p \in t_q$ (i.e. iff t_q contains a node $\neg p$).

The idea of this definition is that each argument in the framework corresponds to a successful derivation of a literal p. A proof tree t_p attacks a proof tree t_q when the corresponding proof for q relies on $\neg p$. This is reasonable since, if p is true, a proof for q that uses $\neg p$ is useless.

We define the semantics resulting from labellings as follows:

Definition 9 *Let P be a logic program and $AF_P = (A, \rightsquigarrow)$ its associated argumentation framework. A set $S \subset \mathcal{B}_P \cup \neg\mathcal{B}_P$ is a* **labelling-model** *iff there is a labelling ℓ of AF_P such that:*

1. *The set of positive literals in S is the set $\{ p \mid \exists t_p \in A \ : \ \ell(t_p) = + \}$.*
2. *The set of negative literals in S is the set $\{\neg q \mid \forall t_q \in A, \ell(t_q) = - \}$.*

A labelling-model is **robust** *iff there is a corresponding labelling which is robust.*

The idea behind the definition of a labelling-model is the following: we accept literals that have a successful proof which is defendable; we negate the literals that have no defendable proof; we do not decide on the truth of literals for which the validity of the proofs are "undecided", i.e. those literals whose proof trees are all labelled $\pm$. Robust labelling-models are those that respect the stability of the decided literals.

Example 5 Figure 4 shows the labellings of the argumentation framework associated to the program

$$\begin{array}{lcl} p & \leftarrow & \neg q \\ q & \leftarrow & \neg p \end{array}$$

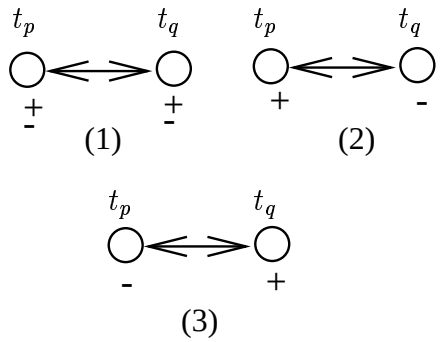


Figure 4

The model resulting from the labelling (1) is $\emptyset$, which is the well-founded model. The models resulting from (2) and (3) are $\{p, \neg q\}$ and $\{\neg p, q\}$ respectively, which are the stable models. All of these models are robust.

Example 6 Figure 5 shows the labellings of the argumentation framework associated to the program

$$\begin{array}{lcl} p & \leftarrow & \neg p \\ q & \leftarrow & \neg p \\ r & \leftarrow & \neg q \end{array}$$

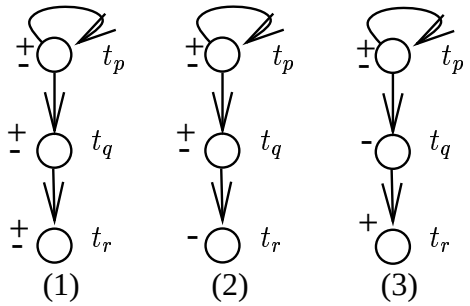


Figure 5

The model resulting from the labelling (1) is $\emptyset$. This model is robust, and is the well-founded model. The model resulting from the labelling (2), $\{\neg r\}$, is not robust. The model resulting from the labelling (3), $\{r, \neg q\}$, is robust.

In our opinion, the most reasonable model for this program is $\{\neg q, r\}$. This model captures the notion of negation as failure (NAF), better than the well-founded model. The idea of the NAF-principle is that the negation of a literal holds iff the literal cannot be shown to hold. We agree that negation cannot be applied to the literal p since, although p cannot be proven, we cannot negate it without causing an inconsistency. The literal q, however, cannot be proven either, since any proof of q would require $\neg p$ to hold which, in turn, would imply p and, thus, an inconsistency. The negation of q would make r true, thus resulting in the model $\{\neg q, r\}$.

Example 7 Figure 6 shows the labellings of the argumentation framework associated to the program

$$\begin{array}{lcl} p & \leftarrow & q \\ q & \leftarrow & \neg p \\ r & \leftarrow & \neg q \\ s & \leftarrow & \neg r \\ s & \leftarrow & \neg s \end{array}$$

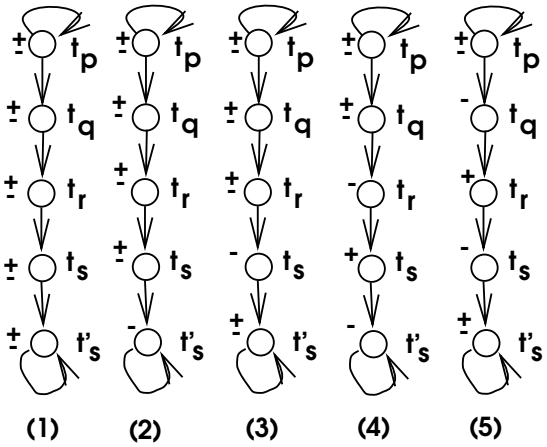


Figure 6

The labellings (1), (2), and (3) all correspond to the same model $\emptyset$. This model is robust, and is the well-founded

model. The model resulting from the labelling (4), $\{\neg r, s\}$, is not robust. The model resulting from the labelling (5), $\{\neg q, r\}$, is robust.

Theorem 3 *Let P be a program. All stable models [2], stable partial models [9], the well-founded (partial) model [3], the three-valued stable models [8], and regular models [10] of P are robust labelling-models of P.*

Thus, we have shown that the traditional models are included in the models produced by our method, so that our theory unifies previous approaches. In addition, examples 6 and 7 show that for programs involving contradiction, there are reasonable models that are captured by labellings and that are not included in traditional semantics.

6 Other approaches

An interesting approach to negation as failure in logic-programming semantics has been suggested in [6], which offers a criterion for acceptability of models. It can be shown that if a model is acceptable (in the sense of [6]), then it is robust iff it is a labelling-model. Conversely, if a model is a labelling-model then it is robust iff it is acceptable. However, there are some acceptable models which are not labelling-models and which, in our opinion, are not reasonable, as in the following example:

Example 8 Figure 7 shows the labellings of the argumentation framework associated to the program

$$\begin{array}{lcl} p & \leftarrow & \neg p \\ q & \leftarrow & \neg p \\ r & \leftarrow & \neg q, \neg s \\ s & \leftarrow & \end{array}$$

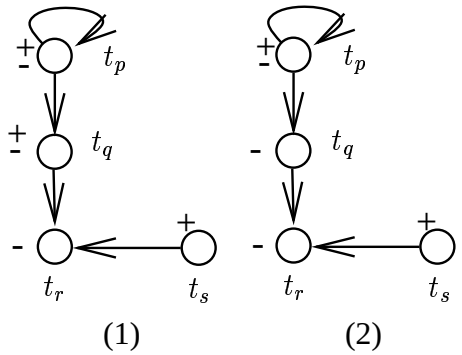


Figure 7

The model resulting from the labelling (1) is $\{s, \neg r\}$, which is the well-founded model. The model resulting from the labelling (2) is $\{\neg q, s, \neg r\}$, which captures the fact that there is no way to prove that q holds. Both of these models are robust. The acceptable models, according to [6], are $\{s, \neg r\}$, $\{\neg q, s, \neg r\}$, and $\{s\}$. The latter, which is not a labelling-model, does not capture negation as failure; indeed, the set $\{r\}$ is unfounded with respect to this model and yet $\neg r$ is not included in the model.

7 Further Research

Example 6 showed that our theory offers a range of choices for the effect of contradictions on the acceptable extensions. The robust model $\emptyset$ is the model whose labelling is maximally undecided and corresponds to the well-founded model. The undecided part of the robust model $\{r, \neg q\}$ is minimal and thus limits the effects of this contradiction, by negating the "first" literal that cannot be proven. In Example 5 the maximal robust labelling corresponds to the well-founded model $\emptyset$, while the two minimal robust labellings give the stable models. In [5] we generalize these remarks by defining an order on robust models which permits us to characterize the various classical models. In addition, we give an alternative formulation of the semantics resulting from robust labellings, thus providing a basis for these models in logic-program semantics theory. A characterization of the semantics is achieved by giving a definition which is equivalent to definition 9, but which is independent of labellings of the associated argumentation framework.

Acknowledgements

The first author acknowledges the support of the National Science Foundation, NFWO. Thanks are extended to Daniel Lewin for useful comments.

References

[1] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[2] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert A. Kowalski and Kenneth A. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium on Logic Programming*, pages 1081–1086, Seattle, 1988. ALP, IEEE, The MIT Press.

[3] A. Van Gelder, K. Ross, and J. S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings of the Seventh ACM Symposium on Principles of Database Systems*, pages 221–230, Austin, Texas, 1988. Association for Computing Machinery.

[4] H. Jakobovits. A theory of argumentation and its application to semantics for logic programs. Msc thesis, Free University of Brussels, VUB, 1995.

[5] H. Jakobovits and D. Vermeir. Logic programming semantics revisited. in preparation, 1995.

[6] A. C. Kakas, P. Mancarella, and Phan Minh Dung. The acceptability semantics for logic programs. In P. Van Hentenrijck, editor, *Proceedings of the 11th International Conference on Logic Programming*, pages 504–519. MIT Press, 1994.

[7] J. W. Lloyd. *Foundations of Logic Programming, second edition*. Springer Verlag, 1987.

[8] T. Przymusinski. Well-founded semantics coincides with three-valued stable semantics. *Fundamenta Informaticae*, 13:445–463, 1990.

[9] D. Sacca and C. Zaniolo. Stable models and nondeterminism for logic programs with negation. In *Proceedings of the 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. Association for Computing Machinery, 1990.

[10] Jia-Huai You and Li Yan Yuan. Three-valued formalization of logic programming: Is it needed? In *Proc. of the PODS'90 conference*, pages 172–182. 1990.