Postulates for Paraconsistent Reasoning and Fault Tolerant Logic Programming

Martin Caminada and Jonathan Ben-Naim

Department of Information and Computing Sciences, Utrecht University

Technical Report UU-CS-2007-004

www.cs.uu.nl

ISSN: 0924-3275

Postulates for Paraconsistent Reasoning and Fault Tolerant Logic Programming*

Martin Caminada
Utrecht University

Jonathan Ben-Naim
University of Luxembourg

January 10, 2007

Abstract

In this paper we examine, from an abstract point of view, a number of properties to be satisfied not only by formalisms for paraconsistent reasoning, but also for formalisms that aim to improve on logic programming by warranting the existence of models.

1 Introduction

Paraconsistent logic can be described as a series of approaches to meaningful formal reasoning in the presence of possibly conflicting information [3, 2, 1]. An important though often neglected question is what it exactly is that one aims to achieve with a formalism for paraconsistent reasoning. That is, what are the important properties a formalism for paraconsistent reasoning should satisfy? An abstract treatment of this issue is important not only for the existing field of propositional and first order paraconsistent reasoning, but also for the field of logic programming and answer set programming, where similar problems occur.

This paper is structured as follows. First, in section 2, we provide three general postulates that we feel each formalism for paraconsistent reasoning should minimally support. Then, in section 3, an analysis is given of which existing paraconsistent logics support which of the postulates in section 2. In section 4, we broaden the discussion to include semantics for logic programming. It will be shown that some of the issues in paraconsistent logic also play a role in logic programming. In section 5, we round off the discussion with a summary of the main results.

2 The Postulates

In order for the postulates to be applicable to a broad range of formalisms, we define the basic concepts in a very general way.

Definition 1. *Let $\mathcal{L}$ be the language of a logical formalism, composed using a countably infinite set of atoms and a finite number of logical operators. Let $\mathcal{F}$ be the set of all well-formed formulas in $\mathcal{L}$. Let $\vdash$ be the entailment operator of this formalism. Let $S \subseteq \mathcal{F}$ We define $Cn(S)$ to be $\{c \mid S \vdash c\}$.*

Examples of logical formalism that satisfy Definition 1 are:

- classical propositional logic

*This work was partly supported by the EU ASPIC project

- propositional default logic, using the sceptical or credulous approach
- grounded normal logic programs under the well-founded semantics
- grounded normal logic programs under the stable model semantics, using the sceptical or credulous approach
- grounded extended logic programs under answer set semantics, using the sceptical or credulous approach

In the following definitions, we write $atoms(S)$ for the atoms that occur in a set of formulas S. For instance: $atoms(p \wedge q;\ r \vee p) = \{p, q, r\}$ and $atoms(p \leftarrow q;\ r \leftarrow s, t)) = \{p, q, r, s, t\}$. Furthermore, if A is a set of atoms and S a set of formulas then we write $S_{|A}$ for those formulas in S that contain only atoms from A. For instance: $\{p \wedge q;\ q \supset r;\ s \vee t;\ q\}_{|\{p,q\}} = \{p \wedge q;\ q\}$.

The first property to be stated is that of *non-interference.* Non-interference roughly means that, for two completely independent knowledge bases S_1 and S_2, S_1 does not influence the outcome with respect to the language of S_2.

Definition 2 (non-interference). *We say that a logical formalism satisfies* non-interference *iff for every two sets of formulas S_1 and S_2 such that $atoms(S_1) \cap atoms(S_2) = \emptyset$ it holds that $Cn(S_1)_{|atoms(S_1)} = Cn(S_1 \cup S_2)_{|atoms(S_1)}$ and $Cn(S_2)_{|atoms(S_2)} = Cn(S_1 \cup S_2)_{|atoms(S_2)}$.*

A property closely related to non-interference is that of *contamination.* Informally, a set of formulas is said to be contaminating iff it yields the same outcome when merged with a totally unrelated set of formulas. That is, a contaminating set of formulas makes all other unrelated sets of formulas irrelevant when being merged with it.

Definition 3 (contamination). *A set of formulas S_1, with $atoms(S_1) \neq atoms(\mathcal{F})$, is called* contaminating *iff for any set of formulas S_2 such that $atoms(S_1) \cap atoms(S_2) = \emptyset$ it holds that $Cn(S_1) = Cn(S_1 \cup S_2)$.*

Based on the concept of contamination, it is then possible to define the property of crash resistancy.

Definition 4 (crash resistancy). *We say that a logical formalism satisfies* crash resistancy *iff there does not exist a set of formulas S that is contaminating.*

The property of crash resistancy is perhaps best understood by examining what crashes commonly mean in software engineering and operating systems. As any experienced computer user knows, it sometimes can occur that a program misbehaves. Under some operating systems, however, the fact that one program misbehaves (like executing illegal instructions) causes the entire operating system to collapse, which then also has consequences for all other programs that were running, even if they are totally unrelated to the program that caused the original problem. The main point here is that one wants to avoid local problems having global effects, and rendering all other things irrelevant. It is this property that is expressed in the above definition of crash resistancy.

The last property to be discussed is that of backwards compatibility. The idea is, roughly, that a formalism (like paraconsistent logic) is to improve on an existing, possibly non crash resisting formalism (like propositional logic) it should yield the same outcome for all non-contaminating input in the latter formalism.

Definition 5 (backwards compatibility). *Let LF_1 and LF_2 be two logical formalisms that use the same language. We say that LF_2 is* backwards compatible *with LF_1 iff for every finite set S that is not contaminating in S_1, it holds that $Cn_{LF_1}(S) = Cn_{LF_2}(S)$.*

3 Logics for Paraconsistent Reasoning

Paraconsistent logics have been introduced to overcome a weakness of classical logic, namely there is no classical model for certain sets of formulas (called inconsistent sets) and therefore everything can be concluded from them. Note that this is more a convention to keep an elegant definition than an intuitively motivated choice. Many directions of research have been explored to remedy this problem.

Some approaches are based on possible worlds semantics, for example, the non-adjunctive logics and the relevance logics. Some, called non truth-functional logics, still define an interpretation as a function v from the language considered to "true" and "false", like classical logic. But, the constraints on v are weakened, in particular the truth value of $\neg\alpha$ is not fully determined by that of α. Last but not least, some define an interpretation as a function from the language to many epistemic states (e.g. "informed that true", "informed that false", "informed of both", "no information"). They are called the many-valued logics.

All of the above approaches aim at avoiding contaminating sets of formulas. However, as far as the authors know, only the many-valued community proposed some logics that are backward compatible with classical logic. In our opinion, one of the most interesting solution is the preferential version of the logic of Belnap [2, 3] proposed by [1].
Let us first recall very briefly some elements of classical logic. Let $\mathcal{L}$ be the language consisting of a set of propositional symbols (or atoms) $\mathcal{A}$ and the usual connectives $\neg$ and $\vee$. Let $\mathcal{F}$ be the set of all well-formed formulas of $\mathcal{L}$.
A *classical valuation* of $\mathcal{L}$ is a function v from $\mathcal{F}$ to $\{0,1\}$ (i.e. "false" and "true") such that $\forall\, \alpha, \beta \in \mathcal{F}$ the following holds: $v(\alpha) = 1$ iff $v(\neg\alpha) = 0$; $v(\alpha \vee \beta) = 1$ iff $v(\alpha) = 1$ or $v(\beta) = 1$.
A *classical model* of a set of formulas Γ is a valuation v such that $\forall\, \alpha \in \Gamma$, $v(\alpha) = 1$.
The *classical consequence relation* for $\mathcal{L}$ is the relation $\vdash_c$ on $\mathcal{P}(\mathcal{F}) \times \mathcal{F}$ such that $\forall \Gamma \subseteq \mathcal{F}$, $\forall\, \alpha \in \mathcal{F}$, $\Gamma \vdash_c \alpha$ iff every model for Γ is a model for α.

It turns out that $\vdash_c$ is not crash resistant because every inconsistent set of formulas is contaminating. As said previously, paraconsistent reasoning has been developed to overcome this problem. The logic of Belnap [3, 2] plays an important role in this field. It uses four epistemic states denoted by $\{1\}$, $\{0\}$, $\emptyset$, $\{0,1\}$. There are many possible meanings for them. Some authors see them as respectively "true", "false", "neither true nor false", "both true and false". Some others see them as the elements of a certain bilattice representing degrees of knowledge and degrees of truth. But, we prefer by far the intuitive meaning that follows.

Imagine some sources of information talking about the atoms (not the compound formulas). For each atom p, a source can tell that p is true, or that it is false, or nothing about p. We consider that an atom is *globally* told to be true (false) iff some source tells that. Then, we *develop* this global information in the following way: if a formula α is told to be true (false), then $\neg\alpha$ is implicitly told to be false (true), and vice versa. In addition, if at least one of α and β is told to be true, then so is $\alpha \vee \beta$. Note that the converse is not required as it would be counter-intuitive, however we automatically get it as the sources can talk only about the atoms. Finally, if both α and β are told to be false, then so is $\alpha \vee \beta$, and vice versa. This global and developed information constitutes a valuation in the logic of Belnap.

More formally, a *Belnap valuation* is a function v from $\mathcal{F}$ to $\mathcal{P}(\{0,1\})$ such that $\forall \alpha, \beta \in \mathcal{F}$,
$1 \in v(\neg\alpha)$ iff $0 \in v(\alpha)$
$0 \in v(\neg\alpha)$ iff $1 \in v(\alpha)$
$1 \in v(\alpha \vee \beta)$ iff $1 \in v(\alpha)$ or $1 \in v(\beta)$
$0 \in v(\alpha \vee \beta)$ iff $0 \in v(\alpha)$ and $0 \in v(\beta)$

Intuitively, $1 \in v(\alpha)$ (resp. $0 \in v(\alpha)$) means that α is told to be true (resp. false). In fact, a Belnap valuation represents, for some sources of information, the global and developed information obtained from what they say by the manner explained previously.
A *Belnap model* of a set of formulas Γ is a Belnap valuation v such that $\forall\, \alpha \in \Gamma$, $1 \in v(\alpha)$. The consequence relation of Belnap, denoted by $\vdash_B$, is defined in the following usual way: $\forall\, \Gamma \subseteq \mathcal{F}$, $\forall\, \alpha \in \mathcal{F}$, $\Gamma \vdash_B \alpha$ iff every Belnap model of Γ is a Belnap model of α.

It is easy to see that this logic is crash resistant for our simple language $\mathcal{L}$. However, it is not backward compatible with classical logic. Indeed, for instance, $\{\alpha, \neg\alpha \vee \beta\} \vdash_c \beta$ whilst $\{\alpha, \neg\alpha \vee \beta\} \nvdash_B \beta$. That is why Arieli and Avron proposed a preferential version. The idea is to consider that the epistemic states $\{0\}$ and $\{1\}$ are preferred to $\emptyset$ and $\{0, 1\}$. Consequently, [1] introduced the preferential relation $\prec$ such that for any two Belnap valuations v and w:
$v \prec w$ (v is preferred to w) iff $\{p \in \mathcal{A} : w(p)$ is $\{0\}$ or $\{1\}\} \subset \{p \in \mathcal{A} : v(p)$ is $\{0\}$ or $\{1\}\}$.
Then, given a set of Belnap valuations V, an element v of V is said to be preferred (in V) iff there is no element of V which is preferred to v.

This leads to the preferential consequence relation $\vdash_p$ such that $\forall\, \Gamma \subseteq \mathcal{F}$, $\forall\, \alpha \in \mathcal{F}$, $\Gamma \vdash_p \alpha$ iff every preferred Belnap model of Γ is a Belnap model of α. It can be checked that $\vdash_p$ is both crash resistant and backwards compatible with classical logic. The same kind of construction can be done with the standard three-valued logic [5]. More generally, the efforts devoted to the study of such logics emphasize the importance of those properties in the field on non-monotonic reasoning.

4 Logic Programming

It is interesting to observe that in the field of logic programming, similar problems play a role as in the field of paraconsistent logic. Under the stable model semantics [7] the program $S_1 = \{a \leftarrow \texttt{not}\, a\}$ is contaminating, since it yields the same outcome when it is merged with an arbitrary syntactically disjunct program S_2 (for instance $S_2 = \{b \leftarrow \texttt{not}\, c\}$). S_1 as well as $S_1 \cup S_2$ do not have any stable models. Therefore, the set of credulously accepted statements is $\emptyset$ and the set of sceptically accepted statements is the set of all literals.

Some alternative approaches to logic programming, such as the well-founded semantics [10] actually satisfy non-interference and crash resistancy. At the same time, it must be mentioned that the well-founded semantics, as well as approaches like [9, 11], are not backwards compatible with the now well-established stable model semantics. For instance, the program
$a \leftarrow \texttt{not}\, b$
$b \leftarrow \texttt{not}\, a$
$c \leftarrow \texttt{not}\, b, \texttt{not}\, c$
has a unique stable model $\{b\}$ whereas its well-founded model is the empty set.

We know of no existing formal semantics for logic programming that satisfies crash resistancy and non-interference, while at the same time being backwards compatible with stable model semantics. Nevertheless, such a semantics would be far from impossible. Caminada, for instance, proposes the semi-stable semantics for abstract argumentation [4]. Semi-stable semantics satisfies crash resistancy and non-interference, and is backwards compatible with stable semantics. As logic programming can be seen as an instantiated form of abstract argumentation [6], this means that we can directly apply semi-stable semantics as a foundation for logic programming. This will then yield a semantics that satisfies non-interference and crash resistancy and is backwards compatible with the stable model semantics. How such could exactly be done is a topic for further research.

5 Summary

In this paper, we have stated three postulates that can be applied to paraconsistent reasoning as well as to logic programming. It was observed that the original paraconsistent logic of Belnap [3, 2] satisfies crash-resistancy but not backwards compatibility. The approach of Arieli and Avron [1], at the other hand, satisfies both properties.

As for logic programming, it was observed that the stable model semantics does not satisfy crash-resistancy. Approaches like the well-founded semantics [8] and that of [9, 11] do satisfy crash resistancy but are not backwards compatible with the stable model semantics. However, it appears that a relatively new semantics (semi-stable, [4]) could be applied to logic programming to yield an approach that is not only crash resistant, but is also backwards compatible with the stable model semantics. This would make semi-stable semantics a very suitable candidate for fault-tolerant Answer Set Programming.

References

[1] O. Arieli and A. Avron. The value of the four values. *Artificial Intelligence*, 102:97–141, 1998.

[2] N. D. Belnap. How computers should think. In G. Ryle, editor, *Contemporary Aspects of Philosophy*, pages 30–56. Oriel Press, 1977.

[3] N. D. Belnap. A useful four-valued logic. In J.M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 7–37. Oriel Press, 1977.

[4] M.W.A. Caminada. Semi-stable semantics. In P.E. Dunne and TJ.M. Bench-Capon, editors, *Computational Models of Argument; Proceedings of COMMA 2006*, pages 121–130. IOS Press, 2006.

[5] I.M.L. D'Ottaviano and N.C.A. da Costa. Sur un problème de Jaśkowski, Sciences. In *Comptes Rendus de l'Académie des Sciences de Paris*, volume 270, pages 1349–1353, 1970.

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

[7] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference/Symposium on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[8] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3/4):365–385, 1991.

[9] Teodor C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13(4):445–463, 1990.

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

[11] Jia-Huai You and Li-Yan Yuan. A three-valued semantics for deductive databases and logic programs. *Journal of Computer System Sciences*, 49(2):334–361, 1994.