# Making inconsistency respectable 1: A logical framework for inconsistency in reasoning

Dov Gabbay and Anthony Hunter
Department of Computing
Imperial College
London SW7 2BZ, UK
{dg,abh}@doc.ic.ac.uk

**Abstract**

We claim there is a fundamental difference between the way humans handle inconsistency and the way it is currently handled in formal logical systems: To a human, resolving inconsistencies is *not necessarily* done by "restoring" consistency but by supplying rules telling one how to act when the inconsistency arises. For artificial intelligence there is an urgent need to revise the view that inconsistency is a "bad" thing, and instead view it as mostly a "good" thing. Inconsistencies can be read as signals to take external action, such as "ask the user", or invoke a "truth maintenance system", or as signals for internal actions that activate some rules and deactivate other rules. There is a need to develop a framework in which inconsistency can be viewed according to context, as a vital trigger for actions, for learning, and as an important source of direction in argumentation.

## 1 Our position

Unfortunately the consensus of opinion in the logic community is that inconsistency is undesirable. Many believe that databases should be completely free of inconsistency, and try to eradicate inconsistency from databases by any means possible. Others address inconsistency by isolating ir, and perhaps resolving it locally. All seem to agree, however, that data of the form $q$ and $\neg q$, for any proposition $q$ cannot exist together, an that the conflict must be resolved somehow.

This view is too simplistic for capturing common-sense reasoning, and furthermore, fails to use the benefits of inconsistency in modelling cognition. Inconsistecy in information is the norm, and we should feel happy to be able

to formalize it. There are cases where $q$ and $\neg q$ can be perfectly acceptable together and hence need not be resolved. In other cases, $q$ and $\neg q$ serve as a welcome trigger for various logical actions. We see inconsistency as useful in directing reasoning, and instigating the natural processes of learning. We must put forward a framework for handling inconsistencies. We need to classify the various aspects of inconsistency and present logical systems which capture various strategies for reasoning with inconsistency.

To summarize, inconsistency in a logical system should be dealt with in a fashion that is more akin to that of human reasoning, namely:

**INCONSISTENCY IMPLIES ACTION**

Having identified an inconsistency, we need to know whether to act on it, and if so, what kind of action to take.

In this series of papers, we attempt to clarify these points, and discuss some relevant work that has been undertaken to address aspects of these issues. We feel that by taking a more appropriate view on inconsistency, we can develop an intuitive framework for diverse logics for common-sense reasoning. In this, paper we present our position.

# 2 Introduction

Classical logic, and intuitionistic logic, take the view that anything follows from an inconsistency. Effectively, when inconsistency occurs in a database, it explodes. The semantics reinforces the nature of inconsistency — there is no classical or intuitionisitic model for inconsistent data.

Yet, from a proof-theoretic perspective, inconsistency is a powerful theorem proving technique for classical and intuitionistic logics — for any formula, if we show an inconsistency holds in teh database, then the formula holds. Indeed for classical logic it is even stronger — if we assume the negation of a formula, and show inconsistency holds, then the formula holds.

It is well known, that for practical reasoning, such proof theoretical power is not compatible with human reasoning. The following will illustrate the point:

> The Americans are very much woried about computer information and technology falling into the wrong hands. As part of their security measures, they have instructed the CIA and the FBI to keep detailed files on computer scientists. The CIA and FBI co-operate in amalgamating their databases.
>
> In the case of Professor Nobody, an unfortunate discrepency had occurred. The FBI had his address as Stanford University, and the CIA had his address as Imperial College. When he two databases were joined together, the union contained contradictory information.

> The computer using classical logic, inferred that Professor Nobody was none other than Public Enemy No 1.

There are several points of pronciple herre. First, classical and intuitionisitic logics do not deal with contradictions correctly. Although, it is logically correct to infer from the above contradiction that Professor Nobody is Public Enemy No. 1, it is clearly an incorrect step in terms of human reasoning. More importantly, Professor Nobody's address is completely irrelevant to the question of whether he is Public Enemy No. 1 or not.

Examples of this nature prompted the logic community to study such logics as relevant logics, and paraconsistent logics, which do indeed isolate inconsistency by various means. But these logics do not offer strategies for dealing with inconsistency. There remains the question of what to do when we have two contradictory items of information in the database. Do we choose one of them? How do we make the choice? Do we leave them in and find a way "around" them? Other logics, such as certain non-monotonic logics, resolve some forms of inconsistncy, but do not allow the representation of certian forms of inconsistent data, or give no answer when present.

A second point is more practical. The CIA agent may investigate Professor Nobosy and find the charge the computer made to be rediculoous. They may suspect that there is a contradiction in the database but they may not know how to locate it. Generally, the contradiction may involve several steps of reasoning and may not be as blatant as in the case of Professor Nobody. We may haveseveral simple and innocent looking data items and some very reasonable rules which together give the wrong answers, but no single item is to blame. How do we debug our system in this case? Systems have been proposed that address aspects of these questions, including truth maintenance systems, but their underlying philosphy is still to eradicate inconsistency at all costs.

From this second point, we can see that inconsistency is actually a spur to act on the basis of an inconsistency. if we are making a decision and we find an inconsistency in our reasoning, we seek to identify further information to resolve it. The following is an example (based on an example from [Nut88].

> Professor Nobody had purchased a tropical fish from the pet shop, together with a guide on tropical fish. The book stated that if the fish is an etropline, it comes from sea-water, and if the fish is cichlid, it comes from fresh-water. When he bought the fish he was told by the shop-owner that the fish was a cichlid, but unfortunately, when he got home he realized that the bag containing the fish stated it was an etropline. Immediately, he rang the shop-owner who clarified the situation by informing him that etroplines are a form of cichlid. Relieved, he put the fish in sea-water.

This example also shows how strategies can be used to resolve inconsistency. In this case, it became apparent to Professor Nobody that since etroplines are a

form of cichlid, the contradiction arising from the two results from the fact with regard to water — etroplines are one exception to the general rule for ciclid. This is an example of a strategy based on using the more specialized rules when making decisions.

So far in our discussion, we have seen the need to isolate, remove, or override inconsistencies in data in some way, or alternatively use them as a spur to acquire more information. Yet, for some kinds of reasoning, it seems desirable to maintain the inconsistency:

> Professor Nobody was 55 years old and wanted an early retirement. He could in fact retire with a full pension if he were ill. So Professor Nobody presented his Head of Department, Professor Somebody, with a letter certifying he had a heart condition. He was thus able to retire. His wife, Mrs Faith Nobody, however, heard of this letter and Professor Nobody told her that he was actually quite healthy and the letter was a trick to get an early retirement. Mrs Nobody was relieved. unfortunately, Professor Somebody overheard the conversation, and very angrily confronted Professor Nobody. Professor Nobody was undisturbed, he explained to Professor Somebody that he had to tell his wife what he had told her, in order to stop her worrying. This may have been the end of the story except that, unfortuantely, Mrs Nobody overheard the conversation with Professor Somebody and was worried again. Professor Nobody assured his wife that he was quite healthy and that he had to tell Professor Somebody what he had told him in order not to get his pension revoked.

There is a basic inconsistency here, but there is no need to "restore' consistency. In fact, to restore consistency in this case, is to cause disaster. if Professor Somebody meets Mrs Nobody in a party, he will pretend that her husband is healthy in order not to worry her, or least avoid the subject. Mrs Nobody will have to pretend that her husband is not healthy, in order to keep the pension. Professor Nobody himself will pretend one way or the other depending on the occasion. The database as we described it *does not care* about the inconsistency. There are no means to resolve it and it makes no difference what "the truth" is.

There are many situations where inconsistency can be used to bst advantage in arguments. Take the following example:

> Professor Nobody was attending an ESPRIT project meeting in Brussels until Saturday. He had booked a Saturday afternoon flight back to London and had arranged for his wife to pick him up at the airport. However, on the Friday afternoon, he had a row with his project partners and decided to fly back to London that evening.

> Without telling his wife of his altered travel plans, he returned home in hte early hours of Saturday morning intending to give a pleasant suprise. He tiptoed to his bedroom intendeding to quietly slip into bed. but to his suprise, there was *another man* in his bed with his wife. Both were fast asleep. he was shocked and angry. But being a logician, he excercised self-restrint and paused to think. Then he left the house quietly, and went back to the airport. He caught a morning flight back to Brussels, and then returned to London on the flight his wife was expecting to see him to be on.

We know that Professor Nobody is a logician, but his behaviour does seem to be inconsistent. Most would have expected that perhaps he should have hit another man, or perhaps he should have made a row. Indeed, we have inconistencies at the object-level and the meta-level. the object-level inconsistencies include all expectations about his wife, Faith, being in the bed on her own, etc, and his new knowledge about his wife being unfaithful.

At the meta-level, if we view Professor Nobody as a classical database management system, we should expect, by comparison with the database system, that he resolve the conflict by adopting some strategy, such as killing the other man (i.e. database deletion to maintain consistency), or alternatively he should update the database by say leaving his wife. However, his action seems to be contrary to the rules of a classical database management system — he seems to be refusing to accept the new input. By going back to Brussels, and following his original plans, he is acting as if the new input was never provided. The refusal of input is therefore inconsistent with the meta-level rules of a classical database management system.

However, instead of making a decision of what action to take when he saw his wife in bed with another man, he chose to pretend that he did not even know her. by not taking action at the time, he could choose to raise the issue with his wife when it suited him best. Essentially, he chose to adopt a strategy of "keeping his options open". Such a strategy therefore requires a mode sophisticated formalization of the meta-level rules of the "database management systems" which are defined for capturing aspects of human reasoning. Thus the database management system, which is more human-like, can delay or refuse input.

Indeed the above example shows how argumentation is based on more sphisticated meta-level theoreies for using inconsistencies. to highlight this we show the same use in another example:

> Professor Nobody had a research assistant called Dr Incompetent. One Monday Professor Nobody asked why Dr Incompetent hadn't come to work the previous week. Dr Incompetent said that he had been ill in bed all of hte last weeek. Professor Nobody expressed sympathy at Dr Incompetent's ill health, but unfortunately for Dr Incompetent, Professor Nobody knew otherwise. For Professor Nobody had seen Dr Incompetent sunbathing in the Park every day last

week. However, he didn't tell Dr Incompetent that he know. Over the course of the summer, this scenario was repeated four times. Upon the fifth occurence, Professor Nobody sacked Dr Incompetent.

As in the previous example, Professor Nobody, knew information to which he didn't admit. Indeed he acted as if he knew the contrary. For the above example, he talked with Dr Incompetent, as if he believed him, but he later expressed his lack of belief in him. In order to support such reasoning, he uses essentially inconsistent information in a way that benefits his overall argumentation strategies. Similarly in the earlier examples, he pretended that he didn't know his wife had been unfaithful, and then used the contrary information to his advantage.

When reasoning with these examples, we are happy with the inconsistencies bcause we read them as signals to take action or signals which trigger some rules and deactivate other rules. We do not preceive ourselves as living in some platonic world contemplating for eternity which of teo contradictory items to disregard. Rather, we constantly face inconsistencies in the real-world, most of which we deal with without any problems.

We believe a solution for better hanling of contradictions can be found by looking closely at the ways humans deal with them. in dveloping a new framework, the following points need to be considered:

- We do not share the view that contradictions in a database are necessarily a bad thing, and that they should be avoided at all costs. Contradictory information seems to be part of our lives and sometimes we even prefer ambiguities and irreconcilable views. We must therefore seek logical principles that allow for contradictions in the same way that we humans allow for them, and even make them useful.

- Human seem to intuitively grasp that some information is more relevant that other information. The notion of relevance should be developed and used.

- There seems to be a hierarchy of rules involved in rationalizing inconsistency. Rules of the form "When contradictory information is received about A, then do B" seemed to be used constantly. These are meta-rules, i.e. rules about rules. Full exploitation of these rules require our database language to be able to talk about itself.

- There is a need to be careful about throwing things out of the database. Always keep an open mind, that although A is rejected now (because of new infomration contradicting it), it may become useful again. Perhaps some uncertainty values may be attached to all data items.

- Learning is a process that is directed by inconsistency. For example, if we have a hypothesis to explain a system, we check our hypothesis by

making a series of observations of the system: If the hypothesis predicts the obsservations, then we have increased confidence in the hypothesis, but if hte hypothesis is the negation of the observation, then we have inconsistency. this inconsistency should initaite further learning about the system.

- Argumentation proceeds via inconsistency. If two people, say Jack and Joe, undertake a dialogue, the inconsistencies between Jack's and Joe's views constitute foci for the dialogue. If the objective of hte dialogue is for one of the participants, say Jack, to exert a particular view upon Joe, then jack will use inconsistencies in Joe's argument as support for Jack's point of view.
- Despite everthing we do, although we may give variuos heuristic rules dealing with contradictory items of equal reliability and relevance, there will be no way of deciding which of the two items is the correct one. in this case, we can only wait and be suspicious of any computation involving them.

These poitns indicate that a wide range of cognitive activities are involved in reasoning with inconsistent information including uncertainty reasoning, meta-reasoning, learning, and argumentation. Yet central to our position is that we should suspend the axiom of absurdity (*ex falso quodlibet*) for many kinds of reasoning. Furthermore, the cognitive activities involved in reasoning with inconsistent information seem to be directly related to the kind of inconsistency. In the next section, we consider issues formalizations of inconsistent information, and categories of inconsistent information.

# 3 A case study

It is valuable to illustrate our ideas through a simple case study. We take a database with a few simple clauses.

$$t_1 : \forall bird(X) \rightarrow fly(X)$$

$$t_2 : \forall bird(X) \wedge big(X) \rightarrow fly(X)$$

$$t_3 : bird(a)$$

$$t_4 : big(a)$$

$$t_5 : hungry(b)$$

$$t_6 : hungry(a)$$

Conisder the query $?bird(b)$. Classical and intuitionisitc logic will derive a contradiction from $(t_1)$ — $(t_4)$ and then deduce the goal. Relevance logic will notice that the contradiction involves another part of the database, namely the part dealing with "a" and will not infer the goal. A paraconsistent logic will allow the inference of both consequents, though some such as IDL [PB91] will annotate the consequents to show they are defeasible.

A truth maintenance system will recognize the inconsistency, and possibly take out some of the data and arrive at a new consistent database. For example, it could take out $t_3$ or even modiy $t_1$ to $t_1^*$:

$$t_1^* : \forall X bird(X) \wedge \neg big(X) \rightarrow fly(X)$$

A defeasible logic will resolve the inconsistency by some means. For example, in this case the rules are ordered according to "specificity". The rule $t_2$ has a more specific antecedent than $t_1$. In a logic such as LDR [Nut88], this ordering allows for a resolution of the conflict by prefering the conclusion from the more specific rule. However, a defeasible system cannot always resolve inconsistency. If we add the following rule,

$$t_7 : \forall X hungry(X) \rightarrow \neg fly(X)$$

then $t_7$ and $t_1$ are in conflict, but neither is stronger than the other. Thus the query $?fly(a)$ will get no answer.

There are some other approaches which use multiple truth values, such as the values $\{T, F, N, B\}$, where $T$ denotes "true", $F$ denotes "false", $N$ denotes "neither" true nor false, and $B$ denotes "both" true and false [Bel77]. For the example, $fly(a)$ would be assigned $B$. Such as assignment captures the situation of both $fly(a)$ and $\neg fly(a)$ following from the database.

Our framework generalizes on all of the above solutions, both technically and conceptually. First we conceptually allow for both $fly(a)$ and $\neg fly(a)$ to be derivable, or even explicitly reside in the database. We regard this favourably. Second, we need to know what *action* to take when facing any inconsistency, and that this action leads to desirable results.

What do we mean by action? The notions and notation for actions is external to the database. We can invoke some form of truth maintenance system to restore consistency, or we can refer to a user, or we can attempt to obtain more data. In each case the action and the results of the action are expressed in a special language supplementary to the language of the database.

Such a language is not entirely new. Planning systems exhibit aspects of this behaviour. Consider the blocks world. The predicates $on(X, Y)$ and $free(Y)$ are database predicates, and the predicate $move(X, Y)$ is an action predicate. A mixed language can state what happens when we execute $move(X, Y)$. For example the following is a mixed langauge statement.

$$move(X, Y) \rightarrow (one(X, Y) \wedge \neg free(Y))$$

We are thus proposing that inconsistencies shoudl be dealt with in a framework whose data and actions are mixed. The paradigm we require can be captured by the following. This is a similar idea to planning, but is different in the way it is developed.

INCONSISTENCY IMPLIES ACTION

Action statements are also used elsewhere in real practice to augment declarative information such as in relational database technology and logic programming. For example, in a warehouse database, if certain stock level parameters fall below a predetermined level, then the database program can automatically generate an order for more supplies for the warehouse. Such an example indicates how declarative information in the database can be linked to actions.

In our framework, the new concept of a database is a pair, the ordinary data (corresponding to the old notion of a database), and the supplementary database, describing the actions. Given the same ordinary data, we get different databases if we vary the actions of the supplementary part, and different logics result from different diciplines for supplementary databases. The supplementary actions capture how we can use and change an object-level database when inconsistencies arise. For example, according to our framework, the database $t_1$,...,$t_6$ given above is the object-level data. We need to supplement this with the action data of the form below, where $\Gamma$ is the inconsistent subset of the database, and QUERY-USER($\Gamma$) is an action predicate that queries the user about the inconsistency:

$$\text{INCONSISTENCY}(\Gamma) \rightarrow \text{QUERY-USER}(\Gamma)$$

Alternatively, we could supplement $t_1$,...,$t_6$ with the following action data, where INVOKE-TMS is an action predicate that invokes some form of truth maintenance system:

$$\text{INCONSISTENCY}(\Gamma) \rightarrow \text{INVOKE-TMS}(\Gamma)$$

In both cases, the action predicate can cause a new object-level database to be formed, and that the inconsistency has been resolved in the new database.

As another example, we consider the above example of Professor Nobody seeking early retirement. For this each agent has a supplementary database that provides information from the object-level database according to who that agent is dealing with. For Professor Nobody we could represent it as follows, where $\Gamma$ is the object-level database, QUERY($X$) is a query $X$ of the object-level database, WHO-IS-ASKING is an action to ascertain who is making the query, ASKING represents the reply to the action, and USEF-FOR-QUERY is the subset of the database that can be used to answer the query.

$$\Gamma = \{\neg \textit{professor-nobody-ill}, \textit{professor-nobody-ill}\}$$

$$\text{QUERY}(\Gamma, X) \rightarrow \text{WHO-IS-ASKING}(X)$$

$$\text{ASKING}(\textit{mrs-nobody}) \rightarrow \text{USE-FOR-QUERY}(\{\neg \textit{professor-nobody-ill}\})$$

$$\text{ASKING}(\textit{professor-somebody}) \rightarrow \text{USE-FOR-QUERY}(\{\textit{professor-nobody-ill}\})$$

In this way, any query of the database causes the supplementary database to check the identity of the querier, and then use the appropriate subset of the database to answer the original query. This is not resolving the inconsistency, but instead uses the action language as a way of dealing with the inconsistency.

As a final example, we consider the above example of Professor Nobody and his unfaithful wife Faith. In the supplementary database for our model of Professor Nobody, we represent the following information, where UPDATE is a request to update the object-level database $\Delta$ with the data $X$, INCONSISTENT holds if $\Delta \cup \{X\}$ is inconsistent, SERIOUS-IMPLICATIONS caputres the situations where $\Delta \cup \{X\}$ has particular consequences, PRETEND-NO-UPDATE supports the daul reasoning that allow the agent to act on an update by $X$, and also as if there has been no update by $X$.

$$\begin{array}{l}(\text{UPDATE}(\Delta, X) \\ \wedge\ \text{inconsistent}(\Delta \cup \{X\}) \\ \wedge\ \text{serious-implications}(\Delta \cup \{X\})) \\ \qquad \rightarrow \text{pretend-no-update}(\Delta, X)\end{array}$$

This inconsistency causes the supplementary database to invoke an updating of the database that would allow "keeping options open".

These examples are intended to illustrate how a supllementary database can be used to support a more human-oriented approach to reasoning with inconsistent information. Even thought some of the likely examples of inconsistent information can be dealt with by existing approaches, such as truth maintenance systems, paraconsistent logics, etc. the intention of the framework is to identify general pronciples of reasoning with inconsistent information. SOme strategies may be quite specifc, others might be quite generic. However, we believe that it is necessary o develop a framework in which the variety of approaches can be formalized ad developed.

As we have seem form thse examples, how we view inconsistency depends on various factors. We need to be able to differentiate between trivial inconsistencies that we can ignore, and significant inconsistencies that we need to act upon. Furhtermore, we need to classify inconsistency according to the appropriate action that is required. We list possible categories of action that might be adopted in a supplementary database.

**Learning action** This should result in some form of revision of information. We consider inconsistencies that cause such actions as useful, since they constitute part of the natural learning processes.

**Information acquisition action** This type of action involves seeking further infomration to reconcile an inconsistency. We consider inconsistencies that cause such actions as useful, since theydirect the process of acquiring knowledge for decision making.

**Inconsistency removal action** This type of action adopts strategies for resolving an inconsistency such as localizing the inconsistency, or adopting some belief revision, or truth maintenance, heuristic to resolve the inconsistency. We consider inconsistencies that cause such actions as an integral part of non-monmotonmic reasoning.

**inference preference action** Some inconsistencies can be resolved if we have a strategy for preferring some inferences over others. Examples include preferring inferences from more specialized rules. We consider inconsistencies that cause such actions as an integral part of non-monotonic reasoning.

**Argumentation action** Inconsitencies that occur during the course of a dialogue between two agents can serve to direct further dialogue. An inconsitency can be a focus for dialogue. We consider such inconsitencies as useful, since they constitute the basis of natural argumentation processes.

# 4 Discussion

It is inconsistency that, at least in part, makes us human. We couldn't conduct our normal ("consistent") lives if we did not ignore the majority of inconsistencies around. There are too many inconsistencies in the real-world for us to be able to resolve them all. Our view on inconsistencies depends on the kind of inconsistency. However, we do stress that inconsistency is important for cognitive activities and as such should be considered to be a desirable notion in logics for artificial intelligence. Our view is different from teh current view that contradictions are "bad". We think that they are an essential part of life and some cane even be made useful. In this paper we have outlined a serie sof actions. We now require a clearer logic formalization coupling inconsistency with action or meta-action.

We propsoe that reasoning with inconsistency can be studied as a Labelled Deductive System [Gab89] where deduction is done both on labels and on formulae. The framework of Labelled Deductive Systems (LDS) is that a basic unit of information is a labelled formula, and that a logic can then be defined in terms of allowed operations on the labelled formulae. Hence, for example, logical consequence can be defined on labelled formulae, where $l_i$ are labels and $A_i$ are formulae.

$$l_1 : A_1, ..., l_n : A_n \vdash l : B$$

The actions of the supplementary database will be linked to sets of labels. Thus very briefly, if we can prove $l_1 : A_1, ..., l_n : \neg A_n$ with different labels, then the set $(\{l_1, .., l_n\}, A)$ will trigger an action. the action could be internal, namely some further non-monotonic mechanism such as choose $l_1 : A$, or some external action, such as query the user. The labels give us the tight control over a derivation that is necessary in deciding which action to take. In this framework, it is possible to trigger action even when there is no inconsistency, such as for formalizing the warehouse database system whcih generate actions according to cetain stock level parameters. We will pursue these issues in subsequent work.

## Acknowledgements

This work is supported by SERC Rule-based Systems Project (GR/G29861). Special thanks are due to Lydia Rivlin, Ruth Kempson and Mark Reynolds for reading earlier drafts of this paper at short notice.

## References

[Bel77] N Belnap. A useful four-valued logic. In G Epstein, editor, *Modern Uses of Multiple-valued Logic*, pages 8–37. Reidel, 1977.

[Gab89] D Gabbay. Lablled deductive systems. Technical report, Department of Computing, Imperial College, London, 1989.

[Nut88] D Nute. Defeasible reasoning and decision-support systems. *Decision Support Systems*, 4:97–110, 1988.

[PB91] T Pequeno and A Buchsbaum. The logic of epistemic inconsistency. In *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning*, pages 453–460. Morgan Kaufmann, 1991.