UNIVERSITY OF ALBERTA

Paraconsistency and Beyond: Studies in Reasoning with Inconsistency in Logic Programming

By

Suryanil Ghosh

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Doctor of Philosophy

Department of Computing Science

Edmonton, Alberta
Spring 1995

ISBN 0-612-01695-1

UNIVERSITY OF ALBERTA

LIBRARY RELEASE FORM

NAME OF AUTHOR: Suryanil Ghosh

TITLE OF THESIS: Paraconsistency and Beyond:
Studies in Reasoning with Inconsistency
in Logic Programming

DEGREE: Doctor of Philosophy

YEAR THIS DEGREE GRANTED: 1995

(Signed) Suryanil Ghosh

Permanent Address:
Rajbati, 125 - N. C. Ghosh Sarani,
Sheoraphuli, Hooghly
West Bengal, India
PIN - 712223

Date: 9th January 1995

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Paraconsistency and Beyond: Studies in Reasoning with Inconsistency in Logic Programming** submitted by *Suryanil Ghosh* in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Supervisor: Dr. Jia-Huai You

Examiner: Dr. R. G. Goebel (Computing Science)

Examiner: Dr. F. J. Pelletier (Computing Science)

Examiner: Dr. B. Linsky (Philosophy)

External: Dr. J. Han (Simon Fraser University)

Examiner: Dr. P. van Beek (Computing Science)

Date: Jan. 6, 1995

This thesis is dedicated to the ones I love.

Parents - Sushil & Jayashree

Brother - Indranil

Son - Akaash

Wife - Cornelia

Abstract

Existing classical logic frameworks either trivialize a theory having inconsistent information or eradicate contradictions from knowledge bases. Contradictions are the norm rather than exception in large knowledge bases, and in more realistic settings. So to *reason in the presence of inconsistency*, i.e. *paraconsistent reasoning*, would be a rational evolution from the ways of reasoning as embedded in the traditional logics. The avoidance of the computational overhead in eradicating the contradictions is also a pragmatic consideration for paraconsistent reasoning. Existing logics of inconsistency and paraconsistent logics allow us to reason in the presence of inconsistency. But they are restricted as they are unable to recognize inconsistency or check when reasoning is propagated from inconsistent information. The purpose of our work is to enhance the different existing logical formalisms to accommodate the ability to reason in the presence of inconsistency.

In this work we introduce fundamentally new ideas to handle inconsistency: *explicit paraconsistency* and *reasoning beyond paraconsistency*. Based on them we develop inconsistency handling strategies *Approach* $C-C_d$ which provide the foundations for logical frameworks akin to paraconsistent logics. We apply these new inconsistency handling strategies to the different evolving formalisms in the family of logic programming and deductive databases.

Acknowledgements

First and foremost I would like to thank my supervisor, Jia-huai You, for the technical and personal support he has provided me throughout my years in graduate school. There is no way I can thank Jia enough for all he has done for me. I would like to thank the other members of my committee, Randy Goebel, Jeffrey Pelletier, Bernhard Linsky, Peter van Beek and the external member Jiawei Han, for their valuable help in the development, presentation, evaluation and correction of my thesis research.

I would like to thank Aditya Ghose for the many discussions during the earlier stages of my thesis research and Padmanabhuni Srinivas for the many discussions during the later stages of my thesis research that led to important improvements in parts of the thesis results. I would also like to thank P. Srinivas for reading through my thesis.

Thanks to all the support staff in the department who provides a wonderful environment to carry on our academic endeavours. Particularly thanks to Edith Drummond who is always there for the graduate students.

I would like to thank my friends for their emotional support through good and not so good times, and for interesting discussions on life, the universe and everything. I would particularly like to thank Ramesh Sankaranarayana, Pavan Sikka, Charles Truscott, Reiner Loewan and Aditya Ghose.

Finally I would like to thank my parents and brother for waiting long years for my return home, and my wife and son who gave me the inspiration in the final years of my research work to finish my thesis.

from the Unreal lead us to the Real,
from Ignorance lead us to Knowledge,
from Life lead us to Eternal Life

— The Rig Veda.

Contents

List of Figures

Chapter 1

Introduction

1.1 Limitations of Classical Logic

Classical Logic (which we will denote as CL) corresponds to a very narrow and determinate conception of logic, and hence represents a very limited and restrictive view of the way in which the human mind works when it carries out deductive processes. One basic result of modern logico-philosophical investigation is that CL does not constitute an adequate formalization of the relation of classical consequence, as believed by the classicists. There are several reasons for this: In the first place, within CL there are theorems that deviate too far from certain extremely strong logical intuitions. One of these intuitions is that between the premises and the conclusion of an argument there must be some relation that is integral to the deductive step (e.g. relevance). According to CL, when two contradictory properties or propositions can be derived in a theory, the theory is trivialized owing to the *Principle of Non-contradiction.* The principle states that a theory will deduce everything in presence of contradiction such as a, $\neg a$, i.e. $\{a, \neg a\} \models b$. Because of this an inconsistent theory whose logic is CL becomes trivial. But, when one thinks seriously what logical consequence means, one can find no *reason* to deduce b from $\{a, \neg a\}$. There is no relation between the premises and the conclusion; there is nothing in common between them:

no connection can justify the deductive step between the former to the latter. In the next section we will discuss how a section of the logico-philosophical community has addressed this problem.

1.2 Paraconsistency: Expanding the horizon of logical rationality

Reason and inference should not break down in inconsistent situations. If an inconsistency is found in one's reasoning, one certainly does not invoke *ex falso quodlibet* and conclude that everything ought to be accepted, nor does one grind to a complete halt. Commonly on finding a contradiction one takes evasive action and changes the views until they are consistent. But common enough though this is, it is by no means rationally obligatory. The rational thing to do may well be to accept the contradiction, or at least to see what emerges from it. (We will further discuss the rationale of paraconsistency in a later section.) The important point for now is just that a theory of reason certainly must be *paraconsistent*, i.e. allow nontrivial reasoning in presence of contradiction. (Paraconsistent literally means *beyond consistent*). The attempt of recent literature to provide an account of rational human reasoning based on classical logic [27, 79] and probability theory, from which human inferential practice frequently deviates, is limited. With correct understanding Frege/Russel logic (CL) can be used in certain restricted domains, viz. consistent ones. This is in fact so, although obtaining a correct understanding of the matter is a sensitive business. Probabilistic logic on the other hand can capture inconsistent situations, by stating that p and $\sim p$ should be believed 0.5 respectively. But with the propagation of reasoning based on any one of the information p or $\sim p$, we lose the information that the information we derive is based on inconsistency. Moreover probablistic frameworks are restricted to only domains where one can determinately record an evidence of believing for different facts. So we see that paraconsistency expands the horizon of

logical rationality.

1.3 Paraconsistency: Characterization

Let $\models$ be a relation of logical consequence. $\models$ may be defined either model theoretically ($\Sigma \models a$ holds for some specified set of valuations, whenever all the formulas in Σ are true under an evaluation, so is a) or proof theoretically ($\Sigma \models a$ holds iff for some specified set of rules, there is a derivation of a, all of whose (undischarged) premises are in Σ), or in some other way. $\models$ is *explosive* iff for all a and b, $\{a, \neg a\} \models b$. It is *paraconsistent* iff it is not explosive. A logic is *paraconsistent* iff its logical consequence relation is. If a logic is defined in terms of a set of theses it may have more than one associated consequence relation. For example, $\{a_i, \ldots, a_n\} \models b$ iff $\vdash (a_1 \wedge \ldots \wedge a_n) \rightarrow b$ *or* $\vdash a_1 \rightarrow (\ldots \rightarrow (a_n \rightarrow b)\ldots)$ *or* $a_1, \ldots, a_n \rightarrow b$ (the last representing the theorem-preserving or *weak* inferential connection). In this case all its associated consequence relations should be paraconsistent.

Let Σ be a set of statements closed under logical consequence. Σ is *inconsistent* iff, for some a, $\{a, \neg a\} \subseteq \Sigma$. Σ is *trivial* iff for all b, $b \in \Sigma$. The important fact about paraconsistent logics is that they provide the basis for inconsistent but nontrivial theories. In other words, there are sets of statements closed under logical consequences which are inconsistent but nontrivial. This fact is sometimes taken as an alternative definition of *paraconsistent* and, given that logical consequence is transitive, it is equivalent to the original definition. The proof is this: If Σ is an inconsistent but nontrivial theory then obviously the consequence relation is paraconsistent. Conversely, suppose that $\{a, \neg a\} \not\models b$. Let Σ be the transitive closure of $\{a, \neg a\}$ under logical consequence. Then Σ is inconsistent but $b \notin \Sigma$. Because of the equivalence we also call any inconsistent but nontrivial theory *paraconsistent*, and derivatively, any position whose deductive closure provides a paraconsistent theory.

1.4 Paraconsistency: Motivation

Why should one be interested in paraconsistent logics? Among the many reasons are proof theoretic and semantic ones. In this section we will present the traditional motivations for paraconsistency.

1.4.1 The proof theoretic reason

The proof theoretic reason is that there are interesting theories T which are inconsistent but nontrivial. Clearly the underlying logic of such theories must be paraconsistent – hence the need to study paraconsistent logics. Examples of inconsistent but nontrivial theories are not uncommon. We present some of them below.

A first example is naive set theory, the theory of sets based on full abstraction axiom scheme, $\exists y \forall x(x \in y \leftrightarrow a)$. This, together with extensionality, characterizes the intuitive conception of set. The theory is inconsistent since it generates the set theoretic paradoxes (e.g. where R is the Russel set, defined as $\{x : \neg x \in x\}$, standard paradox arguments show that $R \in R$ and $\neg R \in R$). Yet it is nontrivial because there are many claims about sets which the intuitive notion rightly rejects (e.g. that $\{\emptyset\} \in \emptyset$, where $\emptyset$ is the null set).

Another group of examples of inconsistent but nontrivial theories derive from the history of science. Consider, for example, the Newton-Leibniz versions of the calculus. Let us concentrate on the Leibniz version. This was inconsistent since it required division by infinitesimal. Hence if α is any infinitesimal, $\alpha \neq 0$. Yet it also required infinitesimal and their products be neglected in the final value of their derivative. Thus $\alpha = 0$. (As much was pointed out by Berkeley [7] and then extended by Boyer [10] in their critique of the calculus.) Despite this the calculus was certainly nontrivial. None of Newton, Leibniz, the Benoullis, Euler, and so on, would have accepted that $\int_0^1 x dx = \pi$.

A very different but most interesting example of an inconsistent but nontrivial

theory in the history of the natural sciences is the Bohr theory of atom [48]. Many other examples of inconsistent but nontrivial theories from the history of science are be given in [22]. Indeed it could be persuasively argued that the whole state of scientific knowledge at any time is a paraconsistent theory [34].

A third group of examples of inconsistent but nontrivial theories comprises certain bodies of information which are theories only in a somewhat attenuated sense. What justifies their inclusion in the present setting is that inferences are made commonly from the information. Thus ideally they may be conceived of as deductively closed corpus or theories. Among the more interesting non-philosophical examples are certain bodies of law, such as bills of rights and constitutions. The following is a convenient hypothetical example which, however, makes the point clearly. The constitution of a certain country contains the clauses (a) "No person of the female sex shall have the right to vote," (b) "All property holders shall have the right to vote." We may also suppose that it is part of the common law that women may not legally be property holders. As enlightenment creeps over the country this part of the common law is cnanged to allow women to hold property. Patently the law is inconsistent. According to the law a woman does and does not have the right to vote. Someone who argued that her cat should be allowed to vote on the basis of (a) and (b) would not get very far. Actual historical examples of inconsistent legal situations are of course more complex and, therefore, more controversial. However two actual examples are the case of Riggs versus Palmer [19] and Lincoln's Proclamation of Emancipation [40]. In the former the clear legal right of inheritance was contradicted by the legal principle that no one shall acquire property by crime. The benefactor had, in fact, murdered the deceased. In the second, the freeing of slaves, who were undoubted legal property, with no compensation, contradicted the fifth amendment of the American Bill of Rights, which says that property shall not be taken without just compensation.

Other examples of inconsistent information from which inferences are drawn in-

clude: the data presented to a jury in a trial; the information fed into a knowledge base by different experts [55]; the information present in knowledge bases networked together [23]; a person's set of belief [78]. In each of these cases the information may obviously be inconsistent. Moreover, inferences are obviously made from this information. That there are inconsistent but nontrivial theories is thus well established.

1.4.2 The semantical reason

A second reason for interest in paraconsistent logics is the fact that there are true contradictions, that is, there are statements a and $\neg a$ such that both are true. Because of this, some inferences of the form $\{a, \neg a\} \models b$ must fail to be truth-preserving (let alone valid) since some statements (take one such for b) are not true. Persuasive examples of true contradictions are provided by the logical paradoxes. These are examples of arguments in set theory and semantics which appear to be perfectly sound arguments issuing in contradictory conclusions. If this is indeed the case then clearly the contradictory conclusions are true. For example consider the sentence:

(**) This is a false sentence in English.

This has two components, a subject 'this' and a predicate 'is a false sentence of English'. Each of the components has certain semantic conditions. Thus, the semantic condition of 'this' is its referring to a certain object – in this case (**), itself. The semantic condition of the predicate is that it applies truly to a certain class of objects, viz. those which are false English sentences. Now of course (**) is contradictory. In other words the semantic conditions of the components of (**) *overdetermine* its true value. They determine it to be both true and false. Of course, one can state dogmatically that this is incorrect and that we have got the semantic conditions of the components wrong. But this is to elevate consistency to an inviolable constraint on semantics. But there is no particular reason that we should suppose so. Semantic conditions are not eternal rules. They have grown up in a piecemeal and haphazard

way. It would, quite frankly, be amazing if they were consistent. Semantic conditions can be seen as determining a field of meaning. Overdetermining truth conditions produce singularities and other discontinuities in the field. But such conditions are to be expected, and in no way interfere with the rest of the field.

A somewhat more controversial example concerns the application of multi-criterial terms. For instance, to determine whether a phrase, such as 'below 0°C', correctly applies to a certain situation, we may observe the behavior of either a correctly functioning alcohol thermometer or a correctly functioning thermoelectric thermometer. These work on quite different principles, and there is no sense in which one is more basic to our determination of, or understanding of, temperature than the other. Certain behavior of either of these instruments provides a sufficient condition for the correct applicability of the term 'below 0°C' or its negation, and both have equal claim to determine the operational meaning of the phrase. Normally the world is such that these two conditions hold or fail together. However, in novel situation they may fall apart. In such a situation both the assertion that the phrase applies and the assertion that its negation does are true. By the symmetry of the situation neither claim can be truer than the other. Hence either both are true or both are false. To suppose that both are false would be to deny that they were criteria in the first place. Thus they must both be true.

Clearly there is a relationship between the proof-theoretic and the semantic motivations of paraconsistency. If the semantic rationale is correct, then so is the proof theoretic one. For if S is the set of things true in some domain containing true conditions then S is an inconsistent but nontrivial theory. However, it is possible to accept the proof-theoretic motivation without accepting the stronger semantic one outlined. For one can hold that there are inconsistent but nontrivial theories that are interesting, have important applications, useful properties, and so forth, without accepting that they are true. Instrumentalists and formalists would, of course, have no problem in accepting such a theme, though they might well find difficulties in clearly distin-

guishing the stronger, *dialethic* position from the weaker, more pragmatic, position. Whether either position is tenable on other grounds is another issue, which has been investigated in [67].

At this point it would be appropriate to state our motivation for paraconsistency. In the area of intelligent information processing, the problem of handling inconsistency has always been an important issue. Several approaches have been proposed to deal with the problem. The classical logic approach of trivialization is the default approach. Then there is the approach of revising existing information in a knowledge base such that the inconsistency no longer prevail. This may happen with the retraction of one or both of the complementary information responsible for the inconsistency. There are some other approaches towards handling inconsistency following traditional information processing ideas, like choosing the largest consistent set of information from the knowledge base and then reasoning from it. All of them have certain limitations. This we will discuss in a separate section later. For the time being we will state without clarifying, that the paraconsistent approach towards inconsistency handling is more pragmatic than the others. Paraconsistency embedded in a framework, enables the framework to hold inconsistent but nontrivial theories that are interesting, have important applications and useful properties, without accepting that they are true.

1.5 Classification of inconsistency in common-sense reasoning

As *common-sense reasoning* became an important area of research of the logic community in the bigger community of Artificial Intelligence, a few new concepts have evolved. The ability to infer from incomplete information and to retract if new evidence challenges the former position is understood to be a pivotal notion of common-sense reasoning. Thus knowledge grows *nonmonotonically.* A basic way of concluding

new information from incomplete information is by the *closed world assumption* proposed by Reiter [73]. *Negation as failure* proposed in [12] is a particular approach towards closed world assumption. The idea of it is as follows: "If α fails to be proved from a theory assume *not* α." The notation *not* denotes negation as failure.

This negation is limited, however, in the sense that *not* α does not refer to the presence of knowledge asserting the falsehood of the atom α but only to the lack of evidence about its truth. Indeed, some authors have translated *not* α as "α is not believed" [41], "α is not known" [28], and "there is no evidence that α is *true*" [21], in addition to the common interpretation "α is not provable from the program in question". So, from now on we will refer to *not* as *negation by default* as it is the inherent meaning that it carries, and as we understand from the different ways it is interpreted above.

An alternative to overcome some of the difficulties of dealing with negated information is to make use of *explicit negation* in addition to negation by default. In this way, the expressive power of a language is increased since the user is now allowed to state not only when an atom is *true* but also when a negated atom is *true* (without any ambiguity or default interpretation).

So now, we have two kinds of negation in the enhanced world of common-sense reasoning: explicit negation (which we denote as $\sim$) and negation by default (denoted by *not*). Explicit negation ($\sim$) is different from classical negation ($\neg$) (see [53]), though they are close in spirit. (This we will discuss later in Chapter 4.) Explicit negation has also been called strong negation (see e.g. [92, 93]) because of its closeness to the negation introduced by Nelson [56]. It has also been called classical negation by Gelfond and Lifschitz [30].

Existence of a thesis and its negation, together in a theory, causes inconsistency. As we have two kinds of negations we can have two kinds of inconsistency. The inconsistency arising on account of explicit negation can be classified as *ontological inconsistency* and the inconsistency arising on account of negation by default as

epistemic inconsistency.

Contradiction arising on account of insufficient knowledge in the reality to choose between contradictory information existing owing to the inconsistent behavior of the reality itself is *ontological inconsistency*[1]; e.g.

- contradictory information fed by different experts into the knowledge base of an intelligent system, or
- combining knowledge bases having mutually contradictory information.

Inconsistency in description of a state of affairs reflecting not an inconsistency in the state of affairs itself but a lack of knowledge about it is *epistemic inconsistency*[2]; e.g.

- assumption (closed world assumption) derived conclusions leading to contradictions, or
- when default conclusions are contradictory.

Ontological inconsistency is a direct consequence of explicit negation, i.e. the simultaneous presence of a fact and its explicit negation in a theory causes it. But epistemic inconsistency is an indirect consequence of negation by default. To be more precise we should say that it is the 'implicational consequence' of negation by default. Let us consider the following theory (which is in the extended logic programming language [34], a nonmonotonic framework, where *not* denotes negation by default and $\sim$ denotes explicit negation):

$$\{\sim a \leftarrow not\, b;\ a \leftarrow not\, c\}.$$

The contradiction in the thesis a arises because a and its negation are derived from the premises *not* b and *not* c respectively. Though the contradiction appears to be classical (i.e. ontological), as we have a and $\sim a$ as a consequence of the theory, its origin is

[1]This notion of ontological inconsistency has been used before roughly with the same meaning as here in [75, 58].

[2]This notion of epistemic inconsistency has been proposed before by [45, 58].

epistemic, i.e. *not b* and *not c*, which implies that the contradictory consequences are a result of close world assumptions.

1.6 Inconsistency handling in Logic Programming

The introduction of explicit negation[3] into a logic program as done in the language proposed by Gelfond and Lifschitz [30], gives rise to the possibility for the program to support inconsistencies. Recent investigations in the use of logic programming based formalisms to knowledge representation have not adequately solved the problem of coping with partially inconsistent information.

Three ways of handling such inconsistency has been proposed. First, one can allow inconsistency to trivialize the knowledge represented by the program by allowing derivation of arbitrary conclusions, as in classical logic. Most logic programming formalisms [30, 31, 29, 32] take this approach. We consider this approach of breaking down in presence of inconsistency to be cognitively inadequate. Second, one may adopt some techniques for belief revision. The argument behind it is that belief revision is particularly well-suited to nonmonotonic formalisms, since they derive conclusions on the basis of *incomplete* information using default assumptions, i.e. negation by default (*not*) as in the language of extended logic programs [30]. As a result, it is possible in these languages to derive inconsistencies which stem from assumptions, which we have classified as *epistemic inconsistency*. But it is also possible that an assumption giving rise to an inconsistency is no longer warranted because of knowledge base updates. It could also be that the information derived from an assumption causing inconsistency is of lower epistemicity than the complementary information causing inconsistency, which may be derived from facts. This situation should not warrant such assumption either. In both the situations epistemic incon-

[3]Explicit negation ($\sim$) does not truly correspond to classical negation ($\neg$) was also shown by Przymusinski [71].

sistencies should not not prevail. Pimentel etal [62] (w.r.t. to the first scenario) and Pereira etal [61] (w.r.t. the second scenario, also see [3, 18, 94]) uses some believe revision techniques for the purpose of deliberately removing the assumptions which contribute to such epistemic inconsistency. Their approaches apply to certain situations where epistemic inconsistency can be removed by retracting the responsible assumptions. But there are situations in which epistemic inconsistencies cannot be removed by retracting the responsible assumptions. Take the following program:

$$\{a \leftarrow notb, \sim a \leftarrow notc\}$$

The inconsistency though derived from assumptions cannot be resolved by retracting any one of the assumptions, because the assumptions are equally assumable. Neither of the above mentioned formalisms using belief revision techniques are able to resolve such epistemic inconsistencies intuitively as they take recourse to multiple extensions, putting a and $\sim a$ in different extensions.

Following the route of multiple extensions because of contradiction leads to the situation where in the line of reasoning following any of the extensions the reasoner is happily ignorant of the fact that its reasoning is based on contradiction. One may argue that the presence of multiple extensions should remind the reasoner of contradiction somewhere in the line of reasoning. But unfortunately, this argument would not hold as multiple extensions can have other reasons of origin too. For example:

$$\{a \leftarrow notb;\ b \leftarrow nota\}$$

or

$$\{a\ or\ b \leftarrow\}\ ,$$

where the language is enriched with the notion of disjunction (or). Moreover we lose the information about what is contradictory.

Furthermore some inconsistencies are not derived through the adoption of assumptions, and therefore cannot be removed by retracting assumptions. These inconsistencies arise on account of insufficient knowledge in the reality to choose between contradictory information existing owing to the inconsistent behaviour of the reality

itself. In more clearer terms: the inconsistency arising on account of explicitly negated fact or explicit negation directly or indirectly derived from indefeasible premises. A knowledge base system fed by different experts or combining knowledge bases are examples where such an inconsistency can arise. We have classified this type of inconsistency as *ontological inconsistency.* The formalisms of Pimentel etal and Pereira etal does not handle this situation either. A third approach to handle inconsistency, is the paraconsistent approach, i.e. when an inconsistency is allowed to be derived without trivialization and reasoning is carried out in presence of inconsistency. By this approach one can handle ontological inconsistency and also the cases of epistemic inconsistency which have not been handled by the first or second approach as we have discussed above. Quite a few paraconsistency based approaches have been proposed [55, 9, 92]. But none of them satisfactorily capture the scenario of reasoning in an inconsistent world. Paraconsistent semantics proposed in [55, 9, 23] fail to capture an inconsistent world as they follow the classical model theoretic approach of considering an interpretation (i.e of assigning truth-values to only positive propositions) and satisfaction by an interpretation at the same level. Thus a proposition gets both the truth-values **true** and **false** represented by **inconsistent** ($\top$) by their paraconsistent semantics. This portrays a confused view of the world. We will discuss in some details about the problems of these approaches in a later chapter.

1.7 Our contribution

In this work we introduce fundamentally new ideas of handling inconsistency. The ideas have some of the same basic characteristics as some of the traditional paraconsistent logics (which we will discuss in Chapter 2). They also have certain similarities to some of the new logics dealing with inconsistency, which we generally refer to as the *logics of inconsistency.* We view them separately from paraconsistent logics as their approach is different from the former. We also present them in Chapter 2.

In most of the existing logical frameworks where contradictions have been handled, they have been considered undesirable and the approach is to either trivialize the inconsistent theory or eradicate the contradictions from the knowledge base. This is the case in traditional classical logics and in more recent logics trying to expand their horizon to capture human common-sense reasoning. These second type of logics have been briefly discussed in Chapter 2. They follow the classical trend of abiding by the law of non-contradiction, *reductio ad absurdum* (which we denote as RAA) or *ex contradictione sequitur quodlibet* (which we denote as ECSQ).

We have strong objections against the ways the existing traditional logical frameworks handle inconsistency. We espouse the notion of *reasoning in presence of inconsistency* as against the traditional ways. To summarize, the following considerations motivate the need for logical formalisms capable of reasoning in the presence of inconsistency (i.e. paraconsistent reasoning):

- Contradictions are the norm rather than exception in large knowledge bases, e.g. networked knowledge bases, two experts feeding a large knowledge base. Knowledge bases may contain local inconsistencies that would make it contradictory and yet they may have a natural intended global meaning.

- Contradictions are common in realistic settings, and hence paraconsistency is a natural evolution towards rationality.

- A more pragmatic consideration: Inabilty to resolve contradiction due to lack of information and hence to be able to reason in presence of inconsistency.

Existing logics of inconsistency and paraconsistent logics allow us to reason in presence of inconsistency. But they are restricted as none of them are able to recognize inconsistency explicitly or check when reasoning is propagated from inconsistent information. Our work takes us a step forward. This restriction is overcome by recognizing the inconsistency, explicitly capturing it and then reasoning in its presence.

We can refer to this version of paraconsistency as *explicit paraconsistency*. We develop a formal strategy based on the concept of explicit paraconsistency. This we call **Approach** $\mathcal{C}$.

Moreover, we go beyond paraconsistency as we allow reasoning from inconsistent information. The status of this reasoning process becomes explicitly distinct as the elements involved in this reasoning process are explicitly marked as elements 'affected' by inconsistency. The elements involved in this reasoning process also have a different epistemic status relative to the elements involved in reasoning that are not affected by inconsistent information. This opens up a new horizon of reasoning, which we call *reasoning beyond paraconsistency*. We develop a formal strategy based on the concept of reasoning beyond paraconsistency, which we call **Approach** $\mathcal{C}_d$.

Now based on our inconsistency handling ideas we develop three different logical frameworks. We apply the combined inconsistency handling strategies **Approach** $\mathcal{C} - \mathcal{C}_d$ to the different formalisms in the family of logic programming and deductive databases for this purpose. We restrict ourself to the propositional framework, so as to focus on the complexities inherent in our approach.

We propose a model theoretic semantics for the first framework we develop based on positive logic programs with explicit negation. This model theoretic semantics has some fundamental difference from the traditional model theoretic semantics for classical logic. The difference is on account of treating inconsistency. We will elaborate on this in Chapter 4. We later provide a constructive semantics for the framework. The need for a constructive semantics is obvious. It provides a computational procedure. Then the question arises why do we need a model theoretic semantics. There are two reasons why we need it. First, it gives us insights into why the semantics provided by the constructive semantics is most appropriate. This we comprehend when the semantics provided by the constructive semantics corresponds with some specific model amongst the several models we get by model theoretic semantics. Furthermore, model theoretic semantics provides all kinds of possible interpretations (or

scenario) in which a formula may be understood (or evaluated). Some of the models may be useful for certain purposes. For example in classical logic we can talk about all possible ways in which a formula may be satisfied. But in logic programming we are interested in some models which are intended. For definite clauses there is a single least model. It does not mean that other models are not useful. In general beyond logic programming other models may be useful. Second, the model theoretic formalism for the first framework provides us with the foundation based on which we develop the more complex inconsistency handling logic programming frameworks.

1.8 Thesis outline

In this section we will give the outline of our research that we present in this thesis. In the next chapter we briefly survey the traditional paraconsistent logics, carry on to present a comprehensive survey of recent works in paraconsistent logics and logics of inconsistency, and finally discuss some of the new ways of handling inconsistency following the classical approach or the traditional *information processing*[4] approach. In Chapter 3 we discuss some of the existing literature in logic programming, based on which we build our logical systems that handle inconsistency. In Chapter 4 we present our basic ideas of handling inconsistency. We then investigate the inconsistency handling strategies *Approuch C* and *Approach* C_d based on the ideas in the skeleton framework of positive logic programs with explicit negation. In Chapter 5 we formally present *Paraconsistent Specifications (PS)*, i.e. *Approach* $C - C_d$ embedded positive logic programs with explicit negation. We give its model theoretic semantics and investigate its properties. In Chapter 6 we present a constructive semantics for PS. In Chapter 7 we venture into the realms of nonmonotonic systems and apply the inconsistency handling strategies *Approach C* and *Approach* C_d to extended logic pro-

[4]By this approach, both inconsistent information can be dropped, one of them can be chosen following a strategy or the reason for their origin is revised such that the contradictions disappear i.e. belief revision [2].

grams. We apply *Approach* $\mathcal{C}-\mathcal{C}_d$ to a more complex framework of *objective epistemic specifications* and develop the framework, in Chapter 8. In Chapter 9 we compare our approach to the existing approaches of handling inconsistency. Finally in the last chapter we summarize our research, highlight our contribution and discuss the future directions of our research. The bibliography is presented at the end.

Chapter 2

Survey: Inconsistency handling approaches

In this chapter we present the various logics that handle inconsistency. Some of them are paraconsistent in nature, some others have the information processing approach of dealing with inconsistency in an ad-hoc basis (e.g., throwing out the contradictory information pair or choosing one over the other).

2.1 A brief history of traditional paraconsistent logic

It is certainly possible to point to figures in history of philosophy who at least made allowance for nontrivial inconsistent theories or worlds, or who must have accepted the idea that an appropriate logical consequence relation is paraconsistent. Any dialetheist, such as Hegel, must have had to do so on pain of triviality of his philosophy. However formal paraconsistent logics are a creature of this century. Their design is, in a sense, a reaction to classical (i.e. Frege [27] / Russell [79, 80]) logic .

The dominant logical paradigm before this century was, of course, Aristotelian logic. The major part of this was the theory of the syllogism. Though Aristotelians

held that a contradiction cannot be true, Aristotelian syllogistic is not explosive. However, like a purely positive logic it is not paraconsistent either. The point is that the poverty of the forms of syllogistic inference and its associated grammatical forms makes it impossible to ask the question of what follows from a contradiction.

However it is quite possible to build onto Aristotelian syllogistic the machinery for expressing this problem. One way, used in the 19th. century, is by the theory of immediate inference inherited from the Stoics. Another is by adding a new class of judgments 's is p and p' and considering rules for the nontrivial consequences of a member of this class. This latter possibility was investigated by the Russian logician Vasil'ev about 1911 [88][1].

The paradigm that replaced Aristotelian logic, viz. classical logic was, of course, anything but paraconsistent. The Frege/Russel account of logical consequence was the legitimate descendant of certain medieval accounts of implication. What was more revolutionary than their logic itself was the methodology they brought to logic. The methodological techniques they used, such as separate analysis of the quantifier, axiomatization, and, in a rudimentary form, the syntactic/semantic distinction, revolutionized our conception of what a formal logic should be like.

The first person to conceive of the possibility of a paraconsistent formal logic, in the modern sense, was probably Lukasiewicz (1910) [52]. However, the first person to produce one was his pupil Jaskowski (1948) [42, 43]. Jaskowski's basic idea is to take **true** to be 'true according to position of some person (e.g. in a discussion)'. This we can represent logically as '**true** in some possible world (the world of that person's position)'. Then a pair of formulas a, $\neg a$, can be **true** without an arbitrary formula b being **true**.

In the 1950s work on paraconsistent logic began independently in South America. Asenjo (1954) [6] and da Costa [13, 14] started to study paraconsistent systems. Of these the most widely developed are those of da Costa. His approach was to graft on

[1] One can refer to [68] for a more elaborate discussion of the history of paraconsistent logic.

to ordinary positive logic a 'negation' operator that is not truth functional. If a takes the value **false**, then $\neg a$ takes the value **true**. But if a takes the value **true**, $\neg a$ may have value **true** or **false**.

A third, and again independent, approach can also be traced back to the late 1950s. At that time in North America Anderson and Belnap [4] taking off from the work of Ackermann [1], started to produce logical systems that were relevant i.e. that avoided the paradoxes of implication. For present purposes we can define a relevant propositional logic to be one in which if $\{a_1 \ldots a_n\} \vdash b$, b and $a_1 \wedge \ldots \wedge a_n$ share a propositional parameter. Anderson and Belnap's intention was not to produce a paraconsistent logic as such. However their logics were paraconsistent. The paraconsistent aspect of relevant logic was later taken up in Australia by Priest [63] and Routley [77, 76]. Priest and Routley have argued that the relevant approach to paraconsistency is the best [67]. We agree with them so far as the traditional paraconsistent approaches we have reviewed. But we will later show that *Approach* $C - C_d$ has many advantages over the relevant approach too. We will briefly present a relevant approach in the next section, so that we can later compare it with *Approach* $C - C_d$. For elaboration on the other two traditional approaches to paraconsistency one can look into the collection of essays on paraconsistent logics in Ch.5 of [68]. For more on paraconsistent logic in general we suggest reference to [5, 15, 17, 16].

2.1.1 Relevant approach to paraconsistency

There are several semantical ways to proceed for relevant approach. We will present the one by Priest [63], which is particularly simple to grasp. The relevant approach takes seriously the view that some statements are **true** and **false**. However, instead of insisting that every sentence take a unique truth value, it allows statements to have both.

Formally, let $V = \{\{1\}\{0\}\{1,0\}\}$. Here $\{1\}$ is (the classical) **true** and **true** only; $\{0\}$ is (the classical) **false** and **false** only; $\{1,0\}$ is (the paradoxical) **true** and **false**.

A valuation is a map v from the set of zero degree formulas (i.e. atoms a, b, ...) to V such that

1a) $1 \in v(\neg a)$ iff $0 \in v(a)$

1b) $0 \in v(\neg a)$ iff $1 \in v(a)$

2a) $1 \in v(a \wedge b)$ iff $1 \in v(a)$ and $1 \in v(b)$

2b) $0 \in v(a \wedge b)$ iff $0 \in v(a)$ or $0 \in v(b)$

3a) $1 \in v(a \vee b)$ iff $1 \in v(a)$ or $1 \in v(b)$

3b) $0 \in v(a \vee b)$ iff $0 \in v(a)$ and $0 \in v(b)$

Logical truth and consequence are defined in the obvious way.

$\Sigma \models_R a$ iff for all evaluations v either $1 \in v(a)$ or for some $b \in \Sigma$, $1 \notin v(b)$

$\models_R A$ iff for all evaluations v, $1 \in v(A)$.

Clearly these truth conditions are paraconsistent, i.e. $\{a, \neg a\} \not\models_R b$. Moreover, the truth conditions look very similar. Indeed they are just the classical ones. Of course in the classical case the second one of each pair (i.e. 1b, 2b and 3b) are redundant. However, this is no longer the case when we have grasped the paraconsistent insight that things may be both **true** and **false**.

A pleasing feature of the semantics is that the set of zero degree logical truths is exactly the set of classical tautologies. This shows that this is a particularly stable set of formulas valid in both classical and inconsistent contexts. Moreover in a sense it shows that relevant logic subsumes classical logic at its zero degree level.

Turning to deducibility relation, clearly this is a sub-relation of the classical one. The one relation of classical logic (CL) that is paraconsistently invalid is the principle of *disjunctive syllogism*

$$\{a, \neg a \vee b\} \models_{CL} b$$

and its cognates such as

$$\{a, \neg(a \wedge b)\} \models_{CL} \neg b$$

This is the only major principle of classical inference that is rejected on the relevant paraconsistent approach. The reason that disjunctive syllogism fails is that sentence a may be paradoxical. If a and $\neg a$ are **true**, then so are a and $\neg a \vee b$, whatever b is.

2.2 Recent and new approaches to paraconsistency: A complete survey

In the last few years their has been some interest in the research of reasoning with inconsistency in the logic particularly the logic community of Artificial Intelligence. There has been two approaches. One that of the traditional information processing approach, by which the effect of a contradiction is nullified by removing the cause from which it came about or by removing the models containing contradiction or by removing one in the complementary pair of contradictions. The other is of a renewed look into the paraconsistent approach and thus new ramifications. Multi-valued logics also added to this fray of reasoning in inconsistent situations as it has paraconsistent characteristics. In this section we will discuss some of the significant works that have facilitated reasoning in presence of contradictions and have taken the later approach, i.e. that of paraconsistency.

2.2.1 Belnap's 'Four-valued Logic'

Belnap [55] argued that a sophisticated question-answering machine that has the capability of making inferences from its database should employ a certain four-valued logic, the motivating consideration being that minor inconsistencies in its data should not be allowed to lead (as in classical logic) to irrelevant conclusions. The actual form

of the four-valued logic was 'deduced' from an interplay of this motivating consideration with certain ideas of Dana Scott [82, 83, 84] concerning *approximation lattices*.

A similar four-valued logic was introduced by [89]. More recently Ginsberg [37, 38] introduced the notion *bilattice*, with Belnap's logic as the simplest example. Fitting [25] used Belnap's four-valued logic, which he called $\mathcal{FOUR}$, as the semantical framework for a refutation mechanism (dual to the usual proof mechanism), which he introduced into logic programming. In [23], Fitting showed that a satisfactory logic programming language could be developed using any bilattice that met certain natural interlacing conditions.

We will present a brief sketch of results about Belnap's four-valued logic which we will also call $\mathcal{FOUR}$ as Fitting did. One can think of these truth-values as *sets* of ordinary truth values:{**true**}, which we will write as **true**; {**false**}, which we will write as **false**; $\emptyset$, which we will write as $\bot$ and read as *undefined*, and {**true**, **false**}, which we will write as $\top$ and read as *overdefined*.

Belnap noted that two natural partial orderings existed for these truth values. The subset relation is one; it can be thought of as ordering reflecting information content. Thus {**true**, **false**} contains more information than a {*true*} say. The other ordering expresses *degree of truth*; for example, $\emptyset$ has a higher degree of truth than {**false**}, precisely because it does not contain **false**. Thus we might call truth value t_1 *less-true-or-more-false* than t_2 if t_1 contains **false** but t_2 doesn't, t_2 contains **true** but t_1 doesn't. Both of these are natural orderings to consider.

Ginsberg had the insight to see there are intimate interconnections and to generalize them. Figure 2.1 is a double Hasse diagram displaying the two partial orderings of $\mathcal{FOUR}$ at once. The knowledge of information direction is vertical (labeled k), while the truth direction is horizontal (labeled t). Thus $a \preceq_k b$ if there is an upward path from a to b, and $a \preceq_t b$ if there is a rightward path from a to b.

Both partial orderings give a complete lattice. In particular, meets and joins exist in each direction. The notation $\wedge$ and $\vee$ is used for infinite meet and join, $\bigwedge$ and $\bigvee$

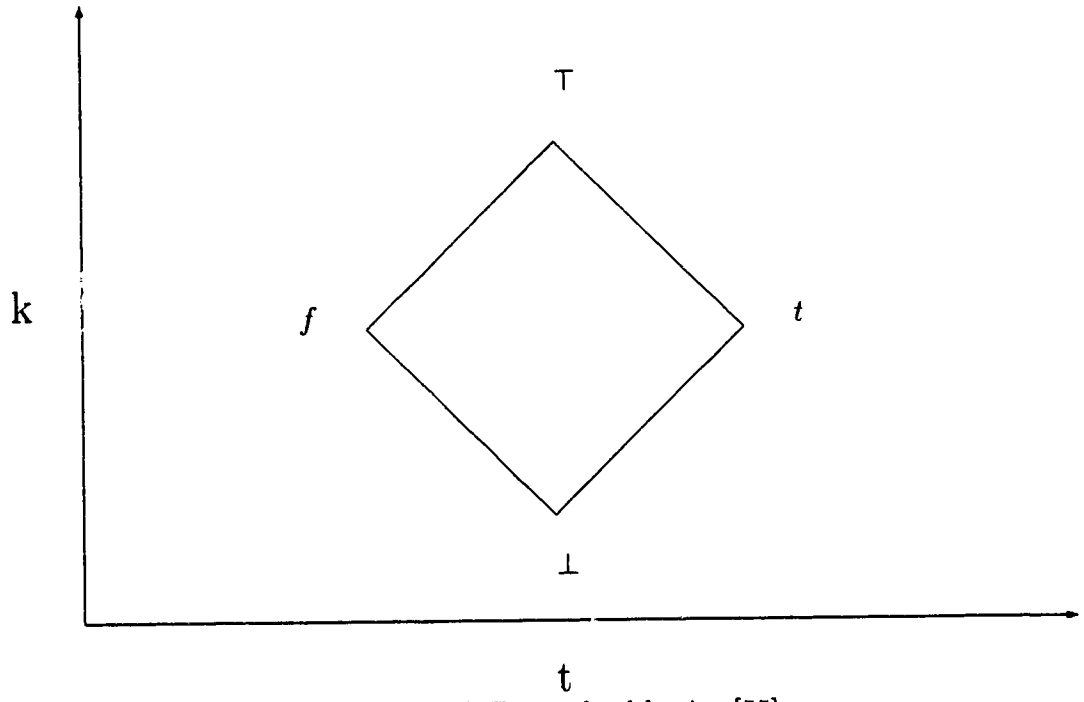

Figure 2.1: A Four-valued lattice [55]

for arbitrary meet and join, in the $\preceq_t$ ordering. $\otimes$ and $\oplus$ is used for finite meet and join, Π and Σ for arbitrary meet and join, in the $\preceq_k$ ordering. Negation is defined directly: $\neg$**true** = **false**; $\neg$**false** = **true**; $\neg\top = \top$; $\neg\bot = \bot$. Thinking of four-valued truth values as sets of ordinary truth values, this negation amounts to the application of ordinary negation to each member of the set.

Kleene's strong three-valued logic [46] and ordinary two-valued logic are present as sublattices. The operations of the $\preceq_t$ ordering and negation, when restricted to **false** and **true**, are the classical ones, and when restricted to **false**, **true** and $\bot$ are Kleene's. Thus $\mathcal{FOUR}$ retains all the mathematical machinery of the two and three-valued logics that have worked so well, and gives us complete lattices, thus simplifying the technical setting somewhat.

Note that the operations induced by the $\preceq_k$ ordering also have a natural interpretation. Combining truth values using $\otimes$ amounts to the consensus approach to conflicts mentioned above, while the use of $\oplus$ corresponds to the accept-anything version. This suggests that counterparts of these operations should be part of a logic-programming language designed for distributed implementation as in [23].

Now, we can discuss briefly how Fitting [23] applied $\mathcal{FOUR}$ to logic programming in a distributed environment. They considered a fixed logic programming language L (whose details can be omitted for our purpose here). In it clause heads are atomic, and clause bodies can contain negations and also conjunctions and disjunctions, explicit or implicit. $\mathcal{P}$ can be considered a program written in the language L. $\mathcal{P}$ is supposed to be distributed over several sites, with some provision for interaction. Some predicates may be local to the site and others may be global. If a query is issued at network sites, each site attempts to answer the query by the usual mechanism of replacing one question with a list of others. Subsequent queries involving local predicates are handled locally, while those involving global predicates are broadcast to the system in the same way that the initial query was. Same as the authors we will not consider the details of variable binding – indeed, for our purposes we will identify a program with the set of ground instances of its clauses. But even with this simple model a fundamental problem arises: what can be done with conflicting responses? If the system is asked $?q$, a site having clauses covering q may respond with "yes", while a site having no clauses for q will, via negation as failure, respond "no". The authors propose that several simpleminded solutions are possible, some of which will go beyond classical truth values. For instance, a consensus can be insisted. Then facing with conflicting answers, one can say that there is no information available. Or each site can be treated as an expert whose opinion is valued, and so when faced with the problem of two answers, one can accept both of them and simply record the existence of the conflict.

Fitting considered Belnap's four-valued logic $\mathcal{FOUR}$, as it provides the right setting for the situation. Now, the meaning assigned to a program $\mathcal{P}$ will be a model, but a four-valued one. As in classical logic, a domain D is considered. Logic programming is generally thought to have a fixed domain, in practice the Herbrand universe. So they considered D to identify with the set of ground atoms. With this understanding they characterized *valuations* as follows:

Definition 2.2.1 *A* valuation v *in $\mathcal{FOUR}$ is a mapping from the set of ground atomic formulas of L to $\mathcal{FOUR}$. Valuations are given two pointwise orderings as follows: $v_1 \preceq_k v_2$ if $v_1(a) \preceq_k v_2(a)$ for every ground atomic formula a; $v_1 \preceq_t v_2$ if $v_1(a) \preceq_t v_2(a)$ for every ground atomic formula a.* □

The set of valuations constitutes a complete lattice under each of the induced orderings $\preceq_k$ and $\preceq_t$, and inherits much of the structure of $\mathcal{FOUR}$ in a natural way. Valuations was extended to maps from the set of all ground formulas to $\mathcal{FOUR}$. A least fixed point semantics was also developed, but least in the $\preceq_k$ direction. Instead of in $\mathcal{FOUR}$ a broader setting of *bilattices* they established the general results. For the details refer to [23].

2.2.2 'RI ..' of Kifer and Lozinski

A logic, presented by Kifer and Lozinski in [45] and called *RI* (Reasoning with Inconsistency) is briefly described in this section.

Their approach is motivated by a view that the real world is inherently consistent, while inconsistency of its logical description occurs only in the mind of the beholder. So, the real world can tolerate inconsistency of the reasoner's beliefs, since the latter may not be grounded in reality. In other words, they distinguish between what the reasoner beliefs in (at the *epistemic* level), and what is actually **true** or **false** in the real world (at the *ontological* level). *RI* can faithfully imitate the standard predicate calculus at its ontological level, thus being intolerant to inconsistency, or it can interpret the calculus epistemically being able to tolerate inconsistency in full (or, it can mix the two interpretations).

Although *RI* allows one to reason about consistency of one's belief, it is worth noting that the logic itself is first-order. Some of the ideas about treatment of inconsistency used in *RI* appeared in [9, 8, 44]. However these works are not as general as Kifer's work, since they are restricted to a subset of logic and as explained later considers only monotonic kind of negation that Kifer refers to as "ontological".

We now present *RI* logic: A belief lattice *BL* is given, with a set of members such that for every element s in the set an ordering $\bot < s < \top$ holds, where $<$ is a reflective and transitive relation, and $\bot$ and $\top$ are lower and upper bounds of the lattice respectively. If ϕ is an atomic formula of predicate calculus, then $\phi : s$ is a literal of *RI*, where $s \in BL$. Composed formulae of *RI* are created in an obvious manner. *BL* is restricted to $\{t, f, \top, \bot\}$ in the following. Where t is **true**, f is **false**, $\top$ is both **true** and **false** (understood as *contradictory*) and $\bot$ is none of **true** and **false** (understood as *undefined or unknown*).

An Interpretation I is a triple (D, F, P), where D is the domain of I, F assigns to each function symbol a mapping from $D \times \ldots \times D$ to D, P assigns to each predicate symbol a mapping $D \times \ldots \times D \times BL$ to $\{1, 0\}$, such that the following conditions hold: for every predicate p and tuple τ from $D \times \ldots \times D$ there is an $r \in BL$ such that $P(p(\tau) : s) = 1$ iff $s < r$. (If $p(\tau)$ is believed to a degree r, it is impossible that it is not believed to a smaller degree.)

Concepts of valuation, satisfaction, truth, model, and logical entailment are defined in the usual manner. Classical implication ($\rightarrow$) (which is called ontological implication here), classical negation ($\neg$) and classical disjunction ($\vee$) have their usual meaning. Therefore

$$\phi \rightarrow \psi \text{ iff } \neg\phi \vee \psi.$$

Two new logical connectives are introduced, an epistemic negation (N) and an epistemic implication ($\sim>$). Epistemic implication is defined as:

$$\phi \sim> \psi \text{ iff } N\phi \vee \psi.$$

N is defined on a four-valued *BL* as $Nt = f$, $Nf = t$, $N\top = \top$, $N\bot = \bot$. It is extended to formulas as follows. $Np : s$ iff $p : Ns$, $N\neg p : s$ iff $\neg p : Ns$, the behavior of epistemic negation with respect to conjunction, disjunction, general quantifier and existential quantifier is the same as for classical negation.

Consider a theory $S = \{q : t \rightarrow p : t;\ q : t\}$. In *RI* logic $S \models_{RI} p : t$. Furthermore,

even if q were an inconsistent belief, $p : t$ would still follow: $\{q : t \rightarrow p : t;\ q : \top\} \models_{RI}$ $p : t$. Thus ontological implication allows to draw conclusions from inconsistent beliefs. This corresponds to the following way of reasoning: q is an inconsistent belief, but in the real world either q or $\neg q$ is **true**. So, to ensure that $q : t \rightarrow p : t$ is **true** in every case, $p : t$ is assumed.

Epistemic implication is overly cautious: from $\{q : t \sim> p : t;\ q : t\}$ $p : t$ does not follow. An interpretation $\{q : \top,\ p : \bot\}$ is a model of T, but it is not a model of $p : t$.

From this point of view it is preferable to consider only models with the least amount of inconsistency.

A model m is *e-inconsistent* iff there is a literal $l : \top$, which is **true** in m. This concept represents an epistemic inconsistency that is tolerated in *RI*. There are models of e-inconsistent sets of formulae, as we have seen. On the other hand, ontological inconsistency, is not tolerated. T is *o-inconsistent* in *RI* iff it contains both $\phi : t$ and $\neg\phi : t$.

An interpretation J is *more or equal e-consistent* as I iff for every atom $p(t_1, \ldots, t_n)$ the following holds: if $p(t_1, \ldots, t_n) : \top$ is **true** in J, then it is **true** in I. (All inconsistencies from J occur in I.) I is *most e-consistent* in a class of interpretations iff there is no I' such that I' is strictly more e-consistent than I. *Epistemic entailment* ($|\approx$) is restricted to the most e-consistent models: a formula ϕ is epistemically entailed by a set T iff every most e-consistent model of T is a model of ϕ.

An inference operator (Cn) implementing the epistemic entailment of *RI* is not a trivial one: from a e-inconsistent set of formulae $\{q : t \sim> p : t;\ q : b\}$, $p : t$ does not epistemically follow. On the other hand, $p : t$ epistemically follows from $\{q : t \sim> p : t;\ q : t\}$.

2.2.3 'Logic of Epistemic Inconsistency' of Pequeno and Buchsbaum

The Logic of Epistemic Inconsistency LEI, proposed by [58], is a paraconsistent logic that syntactically distinguishes between sentences expressing irrefutable (monotonically deduced) "knowledge" from those expressing defeasible (nonmonotonically inferred) ones. It was conceived to tolerate contradiction among this last kind of sentences and to enable meaningful reasoning on these circumstances. The main motivation for its construction has been its use for nonmonotonic reasoning in association to a special default logic, called IDL (for Inconsistent Default logic).

LEI was conceived to formalize and enable reasoning in the presence of *epistemic inconsistency* (refer to Chapter 1 Section 1.5). This kind of reasoning is applied to situations where the knowledge is necessarily incomplete, eventually inaccurate as well, and very often involving information giving evidence to contradictory conclusions. Unlike deduction, such a reasoning cannot be performed on local basis, without appealing to context. In the course of reasoning the arguments interfere with each other, generating conflicts and promoting the defeat of partial conclusions. Furthermore there is no guarantee that every arising conflict can be resolved. It may perfectly happen two opposite partial conclusions having equal rights to be achieved or, even if there is not such a perfect symmetry, it can happen anyway that the available knowledge does not enable a clear decision in favour of one of the alternatives. Thus contradiction arises. The authors argue that: In case of deduction this would carry out a revision of premises by the application of *reductio ad absurdum*, but this is not so for this case. There is no point in applying *reductio ad absurdum* to contradiction among defeasible conclusions.

In [58] it is suggested that these contradictory conclusions should be assimilated in a single theory and reasoned out just as any other. This would emphasize the need for a better understanding of this kind of situations in order to provide a purely logical analysis for them. LEI is a paraconsistent logic designed to cope with intuitions

concerning incompleteness of knowledge, such as the ones described above. Thus, its calculus is intended to reason out meaningfully the inconsistent theories arising in these situations. Its design aims to keep as many properties of classical logic as possible, without interfering with the properties required for the performance of this task. In fact, the calculus behaves classically for undoubting (monotonic, irrefutable) statements and paraconsistently for plausible (nonmonotonic, defeasible) ones. These two kinds of statements are distinguished in the language of the calculus by an interrogation mark (?) suffixing the formulas of the last kind. When used in association to IDL, these marks are supplied by its default rules.

Here we do not discuss the details about the calculus of LEI and its semantics. But we present an example to show its ability to reason in the presence of epistemic inconsistency and the advantages in doing so.

Pequeno argues against the splitting up of contradictory conclusions into multiple extensions, each one internally consistent as done in Reiter's default logic [74]. The splitting up of diverging default conclusions into multiple extensions has the effect of precluding the purely logical analysis of the whole situation. The contribution of extralogical mechanism to deal with extensions to perform reasoning, would be required. Furthermore, this approach has an undesirable side effect that prevents default logic to avoid unintended extensions (and conclusions) in situations such as the famous "Yale shooting problem", discussed in [39].

Example 2.2.1 *Consider the following example, taken from [54]:*

- *Animals usually do not fly;*
- *Winged animals are exception to this, they can fly;*
- *Birds are animals;*
- *Birds normally have wings.*

This can be axiomatized, using Reiter's default, as follows:

1. $\frac{an(x):\neg fly(x)\wedge\neg wing(x)}{\neg fly(x)}$
2. $wing(x) \rightarrow fly(x)$
3. $bird(x) \rightarrow an(x)$
4. $\frac{bird(x):wing(x)}{wing(x)}$

The following reasoning will then be possible: given that Tweety is a bird, it follows from (3) that it is an animal, and from this and the rule (1) that it cannot fly. By *modus tollens* on (2), if it cannot fly it is not winged. With Reiter's default logic this last conclusion prevents the application of the default rule and therefore, from the single fact that the poor Tweety is a bird, it comes out this bizarre conclusion that it is wingless. What happened wrong here is the splitting into two extensions, one in which Tweety is winged and another in which it is not. This did not allow the reasoner to see that, by being a bird, therefore winged, Tweety constitutes an exception to rule (1), which makes this rule not being applicable. This information belonged to another extension and thus could not be seen from the unintended extension. Thus, the dissolving of conflicts by the splitting into extensions prevented the consideration of a relevant piece of evidence, causing the trouble of not handling properly the exception condition.

In [57, 58] a logic with a tolerant disposition towards contradiction called *Inconsistent Default Logic* (IDL) is presented, which is able to solve this problem. A general IDL default rule reads as follows:

$$\frac{A : B : C}{B?}$$

A is the *antecedent* of the rule and B its *default condition.* C is a *proviso* (its negation is an *exception condition* for the application of the rule). Finally, $B?$ is the consequent. This rule is a modification of Reiter's rule in accordance with the following considerations:

- As a defeasible conclusion and an irrefutable one cannot have the same epistemic status, the former is distinguished from the later by the use of an interrogation mark (?) suffixing defeasible formulas.

- IDL implements the idea of accommodating conflicting views in a same extension. Therefore, in IDL the defeasible negation of a default condition $(\neg B)?$ (referred to as a *weak contradiction*) does not prevent the application of the default rule. In order to defeat a default application a *strong contradiction* $\neg B$ is required.

- The *seminormal* part of a default rule is frequently used to express an exception condition. In IDL, C is really taken as a *proviso* for the application of the rule, receiving a differentiate treatment. In order to defeat the application of an IDL default rule by its proviso, a weak contradiction, $(\neg C)?$, suffices.

We are now able to see how IDL works in Morris's example. The same argument could be constructed as before up to the temporary conclusion $\neg winged(Tweety)?$. But now this does not defeat the application of rule (4) and thus $winged(Tweety)?$ is also achieved. This last conclusion, even being defeasible, is able to prevent the application of rule (1) (by its proviso). Therefore $\neg fly(Tweety)?$ is withdrawn together with $\neg winged(Tweety)?$. So, with IDL, only the expected conclusions that by being a bird Tweety is winged and can fly are obtained.

LEI has been designed to serve as the monotonic basis for IDL and at the same time has an independent existence of its own. The ability to reason out contradictions without triviality characterizes LEI as a paraconsistent logic.

2.2.4 'Vivid Logic' of Wagner

In [90, 91] Wagner reported on a nonmonotonic system of partial logic with two kinds of negation, called *weak* and *strong*, respectively. Referring to Levesque's [49] idea of a *vivid knowledge* they call this system *vivid logic* (VL). Unlike classical logic,

VL offers several options for how to deal with inconsistency. The author discussed four of them in his paper [92] and then later extended them in connection with the semantics of extended deductive databases in his paper [93]. Each of the options lead to a particular version of VL which are called *liberal, credulous, conservative* and *skeptical*, respectively.[2]

Here, we will present VL in connection to extended databases (denoted as XDBs). The language of vivid logic corresponds to the language of XDBs where weak negation represents *not* (for negation as failure), and strong negation represents the second negation $\neg$ (for "classical" as in [30]). The authors used the concept of neutralization from directly skeptical inheritance [87, 85, 47] and applied it to the semantics of XDBs to solve the problem of conflicting rules. The principle of neutralization can also be viewed as emerging from the weakening of modus ponens: if modus ponens (i.e. rule application) is restricted to those cases where the conclusion is in some sense 'consistent' (as required with normal default rules), then the potential conclusions that are supported together with their complements are not inferable but neutralized.

The principle of neutralization does not only imply paraconsistency, i.e.

$$\{p, \neg p\} \not\vdash q$$

but also violates reflexivity by

$$\{p, \neg p\} \not\vdash p$$

that no longer holds for inconsistent formulas. The reward for giving up unrestricted reflexivity is the validity of the following non-standard principle:

$$\text{(Inherent Consistency) } X \vdash l \Rightarrow X \not\vdash \neg l$$

that holds in conservative and skeptical reasoning. Since Inherent Consistency holds for negation as failure,

[2]Here the notion of *credulous* reasoning is not related to the usual definition where a credulous conclusion is licensed by an arbitrary-choice extension.

$$X \vdash e \Rightarrow X \nvdash not\ e,$$

the following condition of *Coherence* (the name is adopted from [59]):

$$X \vdash l \Rightarrow X \vdash not \neg l,$$

implies Inherent Consistency.

An extended deductive database X consists of rules of the form $l \leftarrow E$ (read "l if E") where l is a proper literal and E is a set of extended literals (where extended literals are literals or weakly negated literals). The definition of four inference relations between XDB and a formula is given in the restricted language of XDBs: liberal, credulous, conservative and skeptical inference, denoted by $\vdash_l$, $\vdash_{cr}$, $\vdash_c$ and $\vdash_s$, respectively. All the inference relations have the following derivation rules in common:

$$X \vdash not\ not\ l \text{ if } X \vdash l$$
$$X \vdash E \text{ if } \forall e \in E : X \vdash e$$
$$X \vdash not\ E \text{ if } \exists e \in E : X \vdash not\ e$$

and each of them have their own own definition of derivability for literals. They are as follows:

Liberal Reasoning

A literal l can be liberally inferred if it is the conclusion of some rule and the premise of this rule can itself be liberally inferred, no matter whether its complement $\neg l$ is derivable as well:

$$X \vdash_l l \text{ if } \exists (l \leftarrow E) \in [X] : X \vdash_l E$$
$$X \vdash_l l \text{ if } \forall (l \leftarrow E) \in [X] : X \vdash_l not\ E$$

where, $[X]$ is the Herbrand expansion of X w.r.t. the Herbrand universe U_X induced by X, i.e. the set of all constant symbols occurring in X.

The following notions of credulous, conservative and skeptical reasoning are all based on a two-level inference architecture: a conclusion is only *accepted* if it is *supported* but not *doubted*. It is supported if there is a rule for it the premise for which is accepted, and it is doubted if there is a contradicting rule the premise of which is accepted. Differences arise with respect to the degree of skepticism: what counts as an argument in favour of some potential conclusion, and what counts as a neutralizing counterargument?

Credulous Reasoning

In credulous reasoning it suffices in order to establish a conclusion that it is liberally supported (i.e. the supporting argument may even be based on contradictory information), and not credulously doubted:

$$X \vdash_{cr} l \text{ iff } X \vdash_{l} l, \text{ and } \forall(\neg l \leftarrow E) \in [X] : X \vdash_{cr} not\ E$$
$$X \vdash_{cr} not\ l \text{ iff } X \vdash_{l} not\ l, \text{ or } \exists(\neg l \leftarrow E) \in [X] : X \vdash_{cr} E$$

Conservative Reasoning

In conservative reasoning support and doubt have the same weight: a conclusion holds if it is conservatively supported and not conservatively doubted:

- $X \vdash_{c} l$ iff $\exists(l \leftarrow E) \in [X] : X \vdash_{c} E$, and
  $\forall(\neg l \leftarrow F) \in [X] : X \vdash_{c} not\ F$
- $X \vdash_{c} not\ l$ iff $\forall(l \leftarrow E) \in [X] : X \vdash_{c} not\ E$, and
  $\exists(\neg l \leftarrow F) \in [X] : X \vdash_{c} F$

Skeptical Reasoning

In skeptical reasoning there must be no doubt in order to establish a conclusion: it has to be skeptically supported and must not be liberally doubted. In other words,

a sentence is not accepted if there is any counterargument even if it is based on contradictory information.

$$X \vdash_s l \text{ iff } \exists(l \leftarrow E) \in [X] : X \vdash_s E, \text{ and } X \vdash_l not\ \neg l$$
$$X \vdash_s not\ l \text{ iff } \forall(l \leftarrow E) \in [X] : X \vdash_s not\ E, \text{ or } X \vdash_l \neg l$$

These definitions however only works for 'well-behaved' XDBs that are called *well-founded* as defined in [93]. For non-wellfounded XDBs where in the loops (which is responsible for non-wellfoundedness) not involving weak negation the loop detection process as given in [92] seems possible to be used as seen by the author. On the other hand, it is suggested that loops involving weak negation could be assigned a declarative semantics in the style of stable model semantics [34].

The respective consequence operations by LC, CrC, CC and SC, i.e. $LC(X) = \{F : X \vdash_l F\}$, and corresponding for others.

Observation 2.2.1 *If X does not contain weak negation, then $S(X) \subseteq CC(X) \subseteq CrC(X) \subseteq LC(X)$.*

This observation expresses the decreasing degree of skepticism towards ambiguous information in the chain from skeptical to liberal inference. It is illustrated with the following example:

Example 2.2.2 *Let $X = \{p;\ \neg p;\ \neg q;\ q \leftarrow p;\ r \leftarrow p;\ \neg r \leftarrow q\}$ then:*

$$LC(X) = \{p, \neg p, \neg q, q, r, \neg r\}$$
$$CrC(X) = \{\neg q, r\}$$
$$CC(X) = \{\neg q\}$$
$$SC(X) = \emptyset$$

Although p is contradictory, it constitutes evidence for a counterargument against $\neg q$ in skeptical VL. This is not the case in conservative VL, where no counterargument against $\neg q$ is possible. In credulous VL it suffices for r to hold that it is supported by (the contradictory) p and not doubted by any counterargument. □

2.2.5 'Logic of Argumentation' of Fox, Krause and Ambler

In [26] a formal framework for reasoning with inconsistency based on concepts from a theory of 'argumentation' and 'practical reasoning' (with intended applications to decision making and multi-agent problem solving) is presented. Like the different VL reasoning modes, the proposed systems also have a two-level architecture: a formula can be only 'supported' or it is really 'confirmed' (in the stronger sense of VL logics 'accepted'). If a formula is supported or confirmed and its complement is supported, it is 'doubted'. If a formula is supported or confirmed and its complement is confirmed, it is 'rebutted'. In contrast to conservative and skeptical VL, both a formula and its complement can at the same time be confirmed. The philosophy of logic of argumentation, thus, is not based on the principle of mutual neutralization of contradictory information but rather on the principle that "the nature of the conflict should be made explicit to permit reasoning about it at the meta-level". Formulas are annotated by a sign representing their degree of inconsistency. However, one can usually not expect that users of a knowledge representation system specify such annotations when they enter information to the system, or ask queries to it. So, it seems unclear, whether the additional information of annotations can really be used in a practical system.

2.2.6 The 'Extended Well-founded Semantics for Paraconsistent Logic Programs' of Sakama

Sakama motivates his notion of an extended model by relating it to Ginsberg's lattice for default logic [37]. Recall that for the diagram I of an extended model $\mathcal{I}$, and an extended literal $e \in XLit$[3], $I \vdash e$ iff $e \in I$. Sakama presents a generalization of Przymusinski's [70, 72] fixpoint construction of the well-founded model of a program

[3]A literal l is of the form a or $\neg a$, where a is an atom. The set of all literals is Lit. An extended literal is of the form $not\, l$ or $not \neg l$, and the set of all extended literals is $XLit$.

based on two single-step inference operators X^+ and X^- taking an interpretation $I \subseteq XLit$ and a set of tentatively derived, resp. failed, literals $K \subseteq Lit$, and providing an improved set of tentatively derived, resp. failed, literals:

1. $X^+(I,K) := \{l \mid \exists l \leftarrow F \in [X] : I \cup K \vdash F\}$

2. $X^-(I,K) := \{l \mid \forall l \leftarrow F \in [X] : I \cup \bar{K} \vdash not\ F\}$

According to the paraconsistent extended well-founded semantics, if we consider the following example:

Example 2.2.3 *The information that Susan is married either to Peter or Tom, Peter is a bachelor, and a man is not married if he is a bachelor, is the following:*

$$X = \begin{cases} b(P) \\ m(P,S) \leftarrow not\ m(T,S) \\ m(T,S) \leftarrow not\ m(P,S) \\ \neg m(x,y) \leftarrow b(x) \end{cases}$$

□

from X does not follow that Susan is married to Tom, m(T,S), as opposed to $\mathcal{WFSX}$[4] [72], and to conservative and skeptical VL (refer to Section 2.2.4). On the other hand, for $X_1 = X \cup \{\neg m(T,S) \leftarrow not\ \neg b(T)\}$, a model $\mathcal{M}_1$ is provided:

$$< \{b(P)\}, \{b(P), b(S), b(T)\}, \{m(P,S), m(T,S)\}, \{b(T), b(S)\} >$$

Since the extended well-founded model does neither evaluate the implicit disjunctive information (that Susan is married to Peter or Tom) nor relate the validity of $not\ m(T,S)$ to the validity of $\neg m(T,S)$, $\mathcal{M}_1$ makes both $m(P,S)$ and $not\ m(T,S)$ **false**.

Sakama also presents a refinement of his X^+ and X^- operators in order to take account of the fact that some conclusions of an extended program may depend on

[4] Formally, a $\mathcal{WFSX}$ interpretation is given by $< I^t, I^{dt}, I^f, I^{df} >$, where I^t and I^f denotes **true** and **false** atoms respectively, I^{dt} and I^{df} denotes **true** and **false** atoms by defaults.

contradictory premises, and this should be recorded in some way. However, he does neither relate the validity of $\neg l$ nor the inconsistency of l to the failure of l, i.e. to the validity of *not* l, which seems to be a serious shortcoming.

2.3 New approaches in the classical way: A brief overview

In this section we will discuss a few approaches that have dealt with epistemic and ontological contradiction following the classical strategies of *Reductio ad Absurdum*, or *Ex Contradictione Sequitur Quodlibet* (ESCQ).

2.3.1 The 'Contradiction Removal Semantics' of Pereira, Alferes and Aparicio

The *Contradiction Removal Semantics* ($\mathcal{CRSX}$) of [60] extends the $\mathcal{WFSX}$ semantics [59] which is based on the extended stable model semantics of [72] and on the principle that $\neg l$ should imply *not* l, called *Coherence*. Formally, a $\mathcal{WFSX}$ interpretation $< I^t, I^{dt}, I^f, I^{df} >$ is required to satisfy

$$\text{(Coherence) } I^t \subseteq I^u \,\&\, I^f \supseteq I^{df}$$

implying that $I^t \cap I^f = \emptyset$.

We recall Example 2.2.3. The only $\mathcal{WFSX}$ model[5] of X is

$$\mathcal{M} = < \{b(P), m(T,S)\} + \{b(S), b(T)\}, \{m(P,S)\} + \{b(S), b(T)\} >$$

associated with the transformed program

[5] For the sake of brevity it is given $< I^t + (I^{dt} - I^t), I^f + (I^{df} - I^f) >$.

$$X^{\mathcal{M}} = \begin{cases} b(P) \\ m(T,S) \\ \neg m(P,S) \leftarrow b(P) \\ \neg m(T,S) \leftarrow b(T) \end{cases}$$

Considering

$$X' := X \cup \{\neg m(T,S) \leftarrow not \neg b(T)\}$$

that is, X is extended by adding the conditional fact that Tom and Susan are not married if it is true by default that Tom is a bachelor. X' has no $\mathcal{WFSX}$ model simply because all candidates are inconsistent. In order to be able to assign an intended model to programs where contradictions arise through default assumptions (i.e. by weakly negated premises), the $\mathcal{CRSX}$ semantics blocks those weakly negated premises that are responsible for contradictions. In other words, weakly negated sentences are revised if this avoids inconsistency. In the example of X' this means that one cannot conclude $not \neg b(T)$[6], and therefore X' has the following $\mathcal{CRSX}$ model:

$$\mathcal{M}' =< \{b(P), m(T,S)\} + \{b(S)\}, \{m(P,S)\} + \{b(S), b(T)\} >$$

2.3.2 'Semantics of Weighted Maximally-Consistent Subsets' of Lozinskii

Lozinskii in his paper [51] considers a knowledge system S whose purpose is to present the real world W faithfully. But if S turns out to be inconsistent containing contradictory data, he views the state as a result of information pollution with some wrong data. He argues that: One may reasonably assume that most of the system content still reflects the world truthfully, and therefore it would be a great loss to allow a small contradiction to depreciate or even destroy a large amount of correct knowledge. So,

[6]Technically this achieved by adding the 'inhibition rule' $\neg b(T) \leftarrow not \neg b(T)$ to X'.

despite the pollution, S must contain a meaningful subset, and so it is reasonable to assume that the semantics of a logic system is determined by that of its maximally consistent subsets, *mc-subsets*. The information contained in S allows deriving certain conclusions regarding the truth of a formula F in W. In this sense the author says that S contains a certain amount of *semantic information*, and provides an *evidence of* F. A close relationship is revealed between the evidence, the quantity of semantic information of the system, and the set of models of its mc-subsets. Based on these notions, he introduces the *semantics of weighted mc-subsets* as a way of reasoning in inconsistent systems. To show that this semantics indeed enables reconciling contradictions and deriving plausible beliefs about any statement including ambiguous ones, it is successfully applied to a series of justifying examples, such as chain proofs, rules with exceptions, and paradoxes. By an example we will illustrate the semantics of weighted mc-subsets below, but before that we will just mention some terminologies without going to the deeper mathematical definitions of them as given by the authors.

$E(S, F)$ means the *evidence* of a formula F being **true** in a set of formula S comprising to make a theory. s_is are the mc-subsets of S and $\mid s_i \mid$ is the size (number of formulas) of a mc-subset and $w(s_i)$ is a function of the size of s_i. $MC(S)$ is the set of all mc-subsets of S and $MOD(s)$ is the model of a set of formula s. Now we can define *weight of a mc-subset* and *evidence of a formula* with respect to a theory:

Definition 2.3.1 (Weight of an mc-subset) $w(s) = \frac{|S-s|!}{|S|^{|S-s|}}$ □

Definition 2.3.2 (Evidence of a formula F in a theory S)

$$E(S, F) = \frac{\Sigma_{s \in MC(S)} |MOD(s \cup \{F\})| \times w(s)}{\Sigma_{s \in MC(S)} |MOD(s)| \times w(s)}$$ □

Example 2.3.1 *Let us consider the following example of exception that has recently become a classical one.*

$$S = \begin{cases} (i) & bird(x) \rightarrow fly(x) \\ (ii) & penguin(x) \rightarrow bird(x) \\ (iii) & penguin(x) \rightarrow \neg fly(x) \\ (iv) & penguin(quacky) \ \square \end{cases}$$

Since the free variables of the clauses of S are supposed to be universally quantified, S is inconsistent. Indeed, by clauses (iv), (ii), (i), 'quacky' can fly, but owing to (iv), (ii), she cannot. However, is it possible to learn more about this matter from the system? Let us try to figure out by applying the semantics of weighted mc-subsets. S contains 3 mc-subsets:

$s_1 = \{(i),(ii),(iv)\}$, $\mid s_1 \mid = 3$, m_1 = {penguin(quacky),bird(quacky),fly(quacky)};

$s_2 = \{(ii),(iii),(iv)\}$, $\mid s_2 \mid = 3$, m_1 = {penguin(quacky),bird(quacky),¬fly(quacky)};

$s_3 = \{(i),(iii),(iv)\}$, $\mid s_3 \mid = 3$, m_1 = {penguin(quacky),¬bird(quacky),¬fly(quacky)};

So, the system of the above example provides the following evidence about the flying ability of 'quacky'.

$$E(S, fly(quacky)) = \tfrac{1}{3}, \quad E(S, \neg fly(quacky)) = \tfrac{2}{3}.$$

Hence it is most likely that 'quacky' cannot fly, which resolves the contradiction in S in full accord with the common knowledge about the flying ability of penguins.

2.4 Summary

In this chapter we presented a brief account of some of the traditional paraconsistent logics and carried on to present a comprehensive survey of new approaches to paraconsistency. In the last part we discussed some of the new approaches to handling inconsistency following the traditional classical approach of *ECSQ* (for ex contradiction sequitur quodlibet) and traditional information processing approach of *RAA* (for reductio ad absurdum). In Chapter 9 we will discuss the limitations of these work w.r.t. to our work.

Chapter 3

Preliminaries

In this chapter we review some of the preliminaries that we use throughout the thesis. We first review the general concepts involved in *model theory* and discuss the notions about *fixpoints* that are often used to define the declarative semantics of logic programs. We then review the *least model semantics* [20] for positive logic programs and the *answer set semantics* of *extended logic programs (ELP)* [30]. Finally we discuss the language of *objective epistemic specifications (OES)* a subset of the language of *epistemic specifications (ES)* [29].

3.1 Some Model Theoretic Concepts

We will define some terms involved in defining a propositional language. We start with the definition of *Herbrand base.*

Definition 3.1.1 *([50]) Let $\mathcal{L}$ be a propositional language. The Herbrand base $\mathcal{H}$ for $\mathcal{L}$ is the set of all propositional symbols from $\mathcal{L}$.* □

The definition of *Herbrand interpretation* is given as follows:

Definition 3.1.2 *([69]) Any Herbrand interpretation $I =< \bar{T}; \bar{F} >$ can be viewed as a function $I : \mathcal{H} \rightarrow \{0, \frac{1}{2}, 1\}$, from the Herbrand base $\mathcal{H}$ to the 3-element set $\mathcal{T} = \{0, \frac{1}{2}, 1\}$, defined by:*

$$I(\alpha) = \begin{cases} 0 & \text{if } \alpha \in \bar{F} \\ \frac{1}{2} & \text{if } \alpha \in \bar{U} \\ 1 & \text{if } \alpha \in \bar{T} \end{cases}$$

where $\bar{T}$ and $\bar{F}$ are disjoint subsets of the Herbrand base $\mathcal{H}$, $\bar{T}$ is the set of atoms that are true in I and $\bar{F}$ is the set of atoms that are false in I. $\bar{U} = \mathcal{H} - (\bar{T} \cup \bar{F})$ is the set of the remaining atoms in $\mathcal{H}$ which are neither **true** *or* **false** *but* **unknown** *(or* **undefined***).*

An interpretation I is two valued iff I maps $\mathcal{H}$ into the 2-element set $\{0, 1\}$. □

We now give the definition of truth valuation in an interpretation:

Definition 3.1.3 *([69]) If I is an interpretation, then the truth valuation $\hat{I}$ corresponding to I is a function $\hat{I} : \mathcal{F} \rightarrow \mathcal{T}$ from the set $\mathcal{F}$ of all (closed) formulae of the language to $\mathcal{T}$ recursively defi as follows:*

- *If α is an atom, then $\hat{I}$ $= I(\alpha)$.*
- *If F is a formula then $\hat{I}(\neg F) = 1 - \hat{I}(F)$.*
- *If F_1 and F_2 are formulae, then*

$$\hat{I}(F_1 \wedge F_2) = min\{\hat{I}(F_1), \hat{I}(F_2)\};$$
$$\hat{I}(F_1 \vee F_2) = max\{\hat{I}(F_1), \hat{I}(F_2)\};$$
$$\hat{I}(F_1 \leftarrow F_2) = \begin{cases} 1 & \text{if } \hat{I}(F_1) \geq \hat{I}(F_2); \\ 0 & \text{otherwise.} \end{cases}$$ □

Definition 3.1.4 *A theory over $\mathcal{L}$ is a (finite or infinite) set of closed formulae of $\mathcal{L}$. An interpretation I is a (2-valued or 3-valued) model of a theory R if $\hat{I}(F) = 1$, for all formulae F in R.* □

Definition 3.1.5 *A normal logic program P is a set of clauses of the form:*

$$\alpha \leftarrow \phi_1 \wedge \ldots \wedge \phi_m$$

where α is an atom, ϕ_is are literals of the form α or $\neg\alpha$ and $m \geq 0$. □

Clearly, every P is a theory. The following proposition is immediate.

Proposition 3.1.1 *An (Herbrand) interpretation M is a model of a program P if and only if for every instance*

$$\alpha \leftarrow \phi_1 \wedge \ldots \wedge \phi_m$$

of a program clause (where α is an atom and ϕ_is are literals of the form α or $\neg\alpha$) we have

$$\hat{M}(\alpha) \geq min\{\hat{M}(\phi_i) : i \leq m\}. \ \square$$

The definition of *standard ordering* ($\preceq$) is given as follows:

Definition 3.1.6 *([70]) If I and I' are two interpretations then we say that $I \preceq I'$ if $I(\alpha) \preceq I(\alpha)$ for any atom α. If $\mathcal{I}$ is a collection of interpretations, then an interpretation $I \in \mathcal{I}$ is called* minimal *in I if there is no interpretation $I' \in \mathcal{I}$ such that $I' \preceq I$ and $I' \neq I$. An interpretation I is called* least *in I if $I \preceq I'$, for any other interpretation $I' \in \mathcal{I}$. A model M of a theory P is called* minimal *(resp.* least*) if it is* minimal *(resp.* least*) among all models of P.* $\square$

The definition of *F-ordering* is given as follows:

Definition 3.1.7 *([24]) Let $I =< \bar{T}; \bar{F} >$ and $I' =< \bar{T}'; \bar{F}' >$ be two interpretations where $\bar{T}$, $\bar{F}$ and $\bar{T}'$, $\bar{F}'$ are two disjoint subsets of the* Herbrand base $\mathcal{H}$*, respectively. Then we say that $I \preceq_F I'$ iff $\bar{T} \subseteq \bar{T}'$ and $\bar{F}' \subseteq \bar{F}'$. We call this ordering an* F-ordering.

If $\mathcal{I}$ is a collection of interpretations, then an interpretation $I \in \mathcal{I}$ is called F-minimal *in $\mathcal{I}$ if there is no interpretation $I' \in \mathcal{I}$ such that $I' \preceq_F I$ and $I' \neq I$. An interpretation I is called* F-least *in $\mathcal{I}$ if $I \preceq_F I'$, for any other interpretation $I \in \mathcal{I}$. A model M of a theory P is called* F-minimal *(resp.* F-least*) if it is F-minimal (resp. F-least) among all models of P.* $\square$

3.2 Fixpoi ts

Declarative semantics of logic programs is often defined using fixpoints of some natural fixpoint operator O_p applied on *ordered sets* of interpretations. Suppose $\trianglelefteq$ is a partial ordering on the set $\mathcal{I}$ of interpretations of a given language $\mathcal{L}$, then $O_p : \mathcal{I} \rightarrow \mathcal{I}$ is a mapping.

Definition 3.2.1

(1) By the least upper bound $lub(\mathcal{I}')$ *of* $\mathcal{I}' \subseteq \mathcal{I}$ *(resp. the greatest lower bound* $glb(\mathcal{I}')$ *of* $\mathcal{I}' \subseteq \mathcal{I}'$*) we mean an interpretation* $I \in \mathcal{I}$ *such that* $I' \trianglelefteq I$*, for any* $I' \in \mathcal{I}'$ *and* $I' \trianglelefteq I''$ *for any other* I'' *with this property (resp.* $I \trianglelefteq I'$*, for any* $I' \in \mathcal{I}'$ *and* $I'' \trianglelefteq I'$ *for any other* I'' *with this property).*

(2) The partially ordered set $\mathcal{I}$ *is a* complete lattice *if* $lub(\mathcal{I}')$ *and* $glb(\mathcal{I}')$ *exists for every subset* $\mathcal{I}'$ *of* $\mathcal{I}$*.*

(3) Let $\mathcal{I}$ *be a complete lattice.* O_p *is monotonic if* $I \trianglelefteq I'$ *implies* $O_p(I) \trianglelefteq O_p(I')$*, for any* $\{I, I'\} \subseteq \mathcal{I}$*.*

(4) O_p *is nonmonotonic if* $I \trianglelefteq I'$ *does not imply* $O_p(I) \trianglelefteq O_p(I')$*, for some* $\{I, I'\} \subseteq \mathcal{I}$*.*

(5) Let $\mathcal{I}$ *be a complete lattice. Let* $\mathcal{I}' \in \mathcal{I}$*. e say* $\mathcal{I}'$ *is directed if every finite subset of* $\mathcal{I}'$ *has an upper bound in* $\mathcal{I}'$*.*

(6) Let $\mathcal{I}$ *be a complete lattice. We say* O_p *is continuous if* $O_p(lub(\mathcal{I}')) = lub(\mathcal{I}')$*, for every directed subset of* $\mathcal{I}'$ *of* $\mathcal{I}$*.*

(7) An interpretation $I \in \mathcal{I}$ *is a fixpoint of* O_p *if* $O_p(I) = I$*.*

(8) Let $\mathcal{I}$ *be a complete lattice.* $I \in \mathcal{I}$ *is the least fixpoint of* O_p *if* I *is a fixpoint and for all fixpoints* I' *of* O_p*, we have* $I \trianglelefteq I'$*.*

(9) By the smallest interpretation (under the given ordering) we mean an interpretation I_0 *such that* $I_0 \trianglelefteq I$*, for any other interpretation* I*.*

□

The next result is owing to a weak form of a theorem by Tarski [86], which generalizes an earlier result by Knaster and Tarski.

Proposition 3.2.1 ([50]) *Let $\mathcal{I}$ be a complete lattice and O_p be monotonic, O_p has a least fixpoint $lfp(O_p)$, and a greatest fixpoint $gfp(O_p)$. Furthermore, $lfp(O_p) = glb\{I : O_p(I) = I\} = glb\{I : O_p(I) \trianglelefteq I\}$ and $gfp(O_p) = lub\{I : O_p(I) = I\} = lub\{I : I \trianglelefteq O_p(I)\}$.* □

Least fixpoints of operator O_p are generated by *iterating* the operator O_p starting from the *smallest interpretation* I_0 and obtaining the (possibly transfinite) sequence:

$$O_p \uparrow^0 = I_0;$$
$$O_p \uparrow^{\delta+1} = O_p(O_p \uparrow^\delta);$$
$$O_p \uparrow^\lambda = lub\{O_p \uparrow^\delta : \delta < \lambda\};$$

where δ is an *ordinal*, $\delta + 1 = \delta \cup \{\delta\}$ which is the least ordinal greater than δ is the *successor ordinal*, and λ, which is not the successor of any ordinal is the *limit ordinal*.

The first finite ordinal is $\omega = \{0, 1, 2, \ldots\}$, the set of all non-negative integers. The smallest limit ordinal apart from 0 is ω.

Proposition 3.2.2 ([50]) *Let $\mathcal{I}$ be a complete lattice and $O_p : \mathcal{I} \to \mathcal{I}$ be continuous. Then $lfp(O_p) = O_p \uparrow^\omega$.* □

In the sequel we consider two principal orderings among interpretations, namely the *standard ordering* $\preceq$ (see Definition 3.1.6) and the *F-ordering* $\preceq_F$ (see Definition 3.1.7). Operators acting on sets of interpretations ordered by the standard ordering, is denoted by Θ, while those acting on sets of interpretations ordered by the F-ordering, is denoted by Ω. Recall that $I_0 =< \emptyset; \mathcal{H} >$ (resp. $I_0 =< \emptyset; \emptyset >$) is the smallest (resp. F-smallest) interpretation in the set of all interpretations ordered by $\preceq$ (resp. $\preceq_F$).

For a subset $\mathcal{I}'$ of $\mathcal{I}$, we denote by $lub(\mathcal{I}')$ (resp. $glb(\mathcal{I}')$) the least upper bound (resp. the greatest lower bound) of $\mathcal{I}'$ with respect to $\preceq$. Similarly, we denote by $lub_F(\mathcal{I}')$ (resp. $glb_F(\mathcal{I}')$) the least upper bound (resp. the greatest lower bound) of $\mathcal{I}'$ with respect to $\preceq_F$.

Observe, that if $\mathcal{I}' = \{I_\alpha : \alpha \in S$, the set of all atoms in the language$\}$, with $I_\alpha =< T_\alpha; F_\alpha >$, then:

$$lub(\mathcal{I}') = < \bigcup_{\alpha \in S} \bar{T}_\alpha; \bigcap_{s \in S} \bar{F}_\alpha >;$$

$$glb(\mathcal{I}') = < \bigcap_{\alpha \in S} \bar{T}_\alpha; \bigcup_{\alpha \in S} \bar{F}_\alpha >;$$

$$lub_F(\mathcal{I}') = < \bigcup_{\alpha \in S} \bar{T}_\alpha; \bigcup_{\alpha \in S} \bar{F}_\alpha >;$$

$$glb_F(\mathcal{I}') = < \bigcap_{\alpha \in S} \bar{T}_\alpha; \bigcap_{\alpha \in S} \bar{F}_\alpha >;$$

Although $lub(\mathcal{I}')$, $glb(\mathcal{I}')$ and $glb_F(\mathcal{I}')$ are always well-defined interpretations, $lub_F(\mathcal{I}') =< \bar{T}; \bar{F} >$ may not be an interpretation, because the sets $\bar{T}$ and $\bar{F}$ may not be disjoint. However, $lub_F(\mathcal{I}')$ is always an interpretation, if $\mathcal{I}'$ is an *F-directed* set of interpretations, i.e. such that for any $I', I'' \in \mathcal{I}'$ there is a $I''' \in \mathcal{I}'$ satisfying $I' \preceq_F I'''$ and $I'' \preceq_F I'''$.

3.3 Positive Logic Programs (PLP)

In the section we start with the definition of positive logic programs.

Definition 3.3.1 (Positive logic programs (PLP)) *A positive logic program is a set of clauses of the following form:*

$$\alpha_0 \leftarrow \alpha_1 \wedge \ldots \wedge \alpha_m$$

where α_is ($i = 0, \ldots, m$) are atoms and $m > 0$. □

The model-theoretic approach is particularly well-understood in the case of *positive logic programs*. In this section, we assume that all interpretations are 2-valued.

Example 3.3.1 *Suppose that our program P consists of the clauses:*

able-mathematician ← *physicist*

avoids-maths ← *businessman*

physicist.

This program has several different models, the largest of which is the model in which a person is both a physicist and a businessman and is thus an able mathematician and a person who avoids maths at the same time. This model hardly seems to describe the intended meaning of P correctly. Indeed there is nothing in the program to imply that the person is a businessman. We are inclined to believe that the lack of information implies we can assume the contrary. The program also has the unique least model *M_l:*

$$< \{physicist, able\text{-}mathematician\};\ \{businessman, avoids - math\} >$$

This model seems to reflect the semantics of P correctly. At the same time it incorporates the classical case of the closed-world assumption [73]: if no reason exists for some positive statement to be true, then we are allowed to infer that it is false. □

It turns out that the existence of the unique least model M_p^l is the property shared by all positive programs.

Theorem 3.3.1 ([20]) *Every positive logic program P has a unique least (Herbrand) model M_l.* □

This important result led to the definition of the so called least model semantics for positive logic programs.

Definition 3.3.2 (Least model semantics [20]) *By the least model semantics of a positive program P we mean the semantics determined by the least Herbrand model M_l of P.* □

The least Herbrand model semantics is very intuitive and it seems to reflect the intended meaning of positive logic programs properly. The motivation behind this approach is based on the idea that we should minimize positive information as much as possible, limiting it to facts explicitly implied by P, and making everything else false. In other words, the least model semantics is based on a natural form of the *closed world assumption.*

Least model semantics also has a natural fixpoint characterization. First we define the Van Emden-Kowalski immediate consequence operator $\Theta : \mathcal{I} \rightarrow \mathcal{I}$ on the set $\mathcal{I}$ of all interpretations of P (ordered by $\preceq$[1]).

Definition 3.3.3 (The Van Emden-Kowalski operator [20]) *Suppose that P is a positive logic program, $I \in \mathcal{I}$ is an interpretation of P and α is an atom. Then $\Theta(I)$ is an interpretation given by:*

(i) $\Theta(I)(\alpha) =$ **true** *if there is an instance of a clause $\alpha \leftarrow \alpha_1, \ldots, \alpha_m$ in P such that $I(\alpha_i) =$* **true**, *for all $i = 1, \ldots, m$;*

(ii) $\Theta(I)(\alpha) =$ **false**, *otherwise.* □

Theorem 3.3.2 ([20]) *The Van Emden-Kowalski operator Θ has the least fixpoint, which coincides with the least model M_l.* □

The least model M_l is obtained by iterating ω (which denotes the first infinite ordinal) times the operator Θ , starting with the smallest interpretation $I_0 =< \emptyset; \mathcal{H} >$. We get the sequence $\Theta \uparrow^n, n = 0, 1, 2, \ldots, \omega$, $(\Theta \uparrow^0 = I_0)$, of iterations with respect to the standard ordering $\preceq$ of interpretations. The sequence is monotonically increasing and it has a fixpoint

$$\Theta \uparrow^\omega = M_l.$$

[1] If $I =< \bar{T}; \bar{F} >$ and $I' =< \bar{T}'; \bar{F}' >$ are two interpretations, then $I \preceq I'$ iff $\bar{T} \subseteq \bar{T}'$ and $\bar{F} \subseteq \bar{F}'$. In particular, for 2-valued interpretations, $I \preceq I'$ iff $I \subseteq I'$. Here $\bar{T}$ (resp. $\bar{T}'$) and $\bar{F}$ (resp. $\bar{F}'$) are disjoint subsets of the Herbrand base $\mathcal{H}$, $\bar{T}$ (resp. $\bar{T}'$) is the set of atoms that is true in I (resp. I'), i.e. those that belong to I (resp. I') and $\bar{F}$ (resp. $\bar{F}'$) is the set of atoms that are false in I (resp. I') i.e., those that do not belong to I (resp. I'). Interpretations I (resp. I') above are called *2-valued* because they satisfy the condition $\mathcal{H} = \bar{T} \cup \bar{F}$ and thus assign to every ground atom either the value *true* or *false*.

3.4 Extended Logic Programs (ELP)

In this section we review the answer set semantics for extended logic programs [30]. But first we define an ELP.

An *extended logic program* (ELP) is a set of rules of the form

$$L_0 \leftarrow L_1 \wedge \ldots \wedge L_m, not\ L_{m+1} \wedge \ldots \wedge not\ L_n \quad ,$$

where $n \geq m \geq 0$, and each L_i is a literal.

Now we define the answer set semantics for ELP. First let P be an ELP that does not contain *not*. *Lit* stands for the set of literals in the language of P. An *answer set* of P is any minimal subset S of *Lit* such that

(i) for each rule $L_0 \leftarrow L_1 \wedge \ldots \wedge L_m$ from P, if $L_1, \ldots, L_m \in S$ then $L_0 \in S$;

(ii) if S contains a pair of complementary literals, then $S = Lit$.

We denote the answer set of a program P that does not contain negation as failure by $A(P)$.

If P is a positive program, i.e., a program containing neither *not* nor $\sim$, then condition (ii) is trivial, and $A(P)$ is simply the *minimal model* of P (refer to the last section).

Example 3.4.1 *Now let us consider an ELP P_1 without not:*

$$\{\sim p \leftarrow;\ p \leftarrow \sim q\}.$$

It has the answer set:

$$\{\sim p\}.$$

Let us consider another ELP P_2 without not:

$$\{\sim p,\ q \leftarrow \sim p\}.$$

It has the answer set:

$$\{\sim p,\ q\}.$$

□

Now let P be any ELP. For any set $S \subset Lit$, let P^S be the ELP obtained from P by deleting

(i) each rule that has a formula *not* L in its body with $L \in S$, and

(ii) all formulas of the form $not L$ in the bodies of the remaining rules.

Clearly, P^S does not contain *not*, so that its answer set is already defined. If this answer set coincides with S, then we say that S is an answer set of P. In other words, the answer sets of P are characterized by the equation

$$S = A(P^S) \tag{3.1}$$

Example 3.4.2 *Consider the program* P_3*:*

$$\sim q \leftarrow not\ p.$$

To check that $\{\sim q\}$ *is an answer set of the program* P_3*, we should construct the program* $P_3^{\{\sim q\}}$*. This program contains one rule*

$$\sim q \leftarrow$$

(the result of deleting not p from the only rule of P_3*. The answer set of this program is* $\{\sim q\}$*, the set that we started with. Consequently, this is indeed the answer set of* P_3*. It is easy to check that no other subset of literals has the same fixed point property.* □

An answer set of a *normal logic program* – a program without explicit negation, but with negation by default *not* – is a set of *atoms*. The definition of an answer set coincides here with the definition of stable models [34]. (Notice that the sign $\sim$ stands there for negation by default and thus corresponds to *not* in the notation of this chapter.) We conclude that the answer sets of a normal logic program are identical to its stable models. In this sense, the semantics of ELP, applied to normal logic programs, turns into the stable model semantics.

An ELP is *contradictory* if it has an inconsistent answer set (that is, an answer set containing a pair of complementary literals). For instance, the program P_4, consisting of the rules

$$\{\sim q \leftarrow not\ p;\ q \leftarrow not\ p\}$$

is contradictory as it has the answer set that contains both q, $\sim q$. Clearly a normal logic program cannot be contradictory.

Proposition 3.4.1 ([30]) *Every contradictory program has exactly one answer set - the set of all literals, Lit.* □

Being non-contradictory doesn't guarantee, of course, the existence of answer sets. This can be illustrated by the normal logic programs without stable models, such as $\{p \leftarrow not\ p\}$.

3.5 Objective Epistemic Specifications (OES)

In this section we present the language of OES, a sub-language of ES [29, 33]. The sub-language OES has not been defined before. But as it is a restricted version of the language of ES (the language of ES in propositional form without modal operators K and M), we make certain modifications to the definitions of the language of ES as given in [29] and present it here. We can also consider OES as an extension of the language of ELP (for extended logic programs) by inclusion of epistemic disjunction (which we shall soon elaborate on) and complex formulae.

Let us consider a language $\mathcal{L}_0$ consisting of propositional symbols $p, q, \ldots$, logical connectives $\wedge$ (and), $\sim$ (explicit negation), *not* (negation by default) and *or* (epistemic disjunction, different from classical disjunction $\vee$). Formulae of $\mathcal{L}_0$ are defined in the usual way. Formulas of the form p are called atoms. By literals we mean atoms α and their strong negations $\sim \alpha$. The set of all ground literals are denoted by *Lit*.

Let us consider a set of literals W. (W represents his/her current/working set of beliefs.) We inductively define the notion of truth ($\models$) and falsity ($=|$) of a formulae of $\mathcal{L}_0$ w.r.t. a pair W.

Definition 3.5.1

$W \models \phi$ *iff* $\phi \in W$

$W \models F_1 \wedge F_2$ *iff* $W \models F_1$ *and* $W \models F_2$

$W \models F_1 or F_2$ *iff* $W \models F_1$ *or* $W \models F_2$

$W \models \sim F$ *iff* $W =| F$

$W \models not\ F$ *iff* $W \not\models F$

$W =| \phi$ *iff* $\tilde{\phi} \in W$

$W =| F_1 \wedge F_2$ *iff* $W =| F_1$ *or* $W =| F_2$

$W =| F_1 or F_2$ *iff* $W =| F_1$ *and* $W =| F_2$

$W =|\sim F$ *iff* $W \models F$

$W =| not F$ *iff* $W \models F$

where, $\phi \in \Phi$*; if* ϕ *is an atom* α*,* $\tilde{\phi}$ *denotes* $\sim \alpha$ *and vice versa; and* F_i*s are arbitrary formulae constructed from the set* Φ *of all literals of our language by the connective* $\sim$*,* $\wedge$ *and or.* □

Remark 3.5.1 *The meaning of the connective or, called epistemic disjunction, is given by the semantics of disjunctive databases [31] and differ from that of* $\vee$ *(classical disjunction). The meaning of a formula* $a \vee b$ *is "a is true or b is true", while a or b is interpreted epistemically and means "believe a or believe b".*

Let us attempt some intuitive explanation of or. p or q is true w.r.t. a set of literals A if p is in A or q is in A. p or; q is false w.r.t. A if $\sim p$ *is in A or* $\sim q$ *is in A. Otherwise the truth of p or; q is unknown.*

Notice that A is a set of literals, not atoms. It represents an incomplete model of the world.

The language and the satisfiability relation described above together with the notion of a rule from logic programming are used to provide a specification of a reasoner with the desired properties. The formal notion of such a specification is captured by the following definitions:

Definition 3.5.2 *By an* objective epistemic specification *(denoted as OES) we mean a collection of rules of the form*

$$F \leftarrow F_1 \wedge \ldots \wedge F_m$$

where F *and* F_i*s are arbitrary formulae and* $m \geq 0$*.* □

Remark 3.5.2 *Our semantics is not "contrapositive" with respect to* $\leftarrow$ *and* $\sim$ *; it assigns different meaning to the rules* $p \leftarrow\sim q$ *and* $q \leftarrow\sim p$. *The reason is that it interprets expressions like these as* inference rules, *rather than conditionals. The language of OES includes explicit negation (which is different from classical negation but have some closeness too), but not classical implication.* $F \leftarrow G$ *means '*F *if* G*'.*

Now we define a set of literals satisfying an objective epistemic specification Π. We call such a set an *answer set* of T. The precise definitions of these notions are given in several steps:

Definition 3.5.3 *Let* Π_0 *be an OES consisting of rules of the form*

$$F \leftarrow$$

A set W *of literals is called a belief set of* T_0 *iff it is a minimal set with a property* $W \models F$ *for every rule from* T_0. *If* W *contains a pair of complementary literals then* $W = Lit$. □

Example 3.5.1 *Let* T *consist of the clauses:*

$$p \text{ or } q \leftarrow$$
$$\sim p \leftarrow$$
$$r \text{ or } s \text{ or } t \leftarrow$$
$$not\ s \leftarrow$$

Clearly T *has two belief sets:*

$$\{\sim p, q, r\}\ \{\sim p, q, t\}$$

□

Definition 3.5.4 *Let* Π *be an arbitrary OES and* W *be a set of literals in the language of* Π. *For every* W *by* Π_W *we denote the epistemic specification obtained from* Π *by:*

1. *Removing from the premises of rules of* T *all formulae* F_i *such that* $W \models F_i$.
2. *Removing all remaining rules with non-empty premises*

Then W *is called a world view of* Π *iff* W *is a belief set of* Π_W}. □

Example 3.5.2 *Let P consist of the formulae*

$$p \text{ or } q \leftarrow r \wedge not\ s$$

$$r \leftarrow$$

Clearly this specification has two answer sets: $\{p, r\}$ $\{q, r\}$. □

Example 3.5.3 *Let* II *consist of the formulae*

$$p \text{ or } q \leftarrow r \wedge not\ s$$

$$r \leftarrow$$

$$t \leftarrow not\ s$$

$$\sim t \leftarrow not\ s$$

It is easy to see that this specification is inconsistent and thus have a set of all literals Lit as an answer set, which pertains to having no answer set. □

Example 3.5.4 *Let P consist of the formulae*

$$p \leftarrow not\ q$$

$$q \leftarrow not\ p$$

This specification is satisfied by two answer sets: $\{p\}$ *and* $\{q\}$ □

Example 3.5.5 *Let P consist of the formulae*

$$p \leftarrow not\ p$$

This specification does not have an answer set. □

Example 3.5.6 *Let P consist of the formulae*

$$p \leftarrow q$$

$$q \leftarrow$$

$$r \leftarrow p$$

$$\sim q \leftarrow r$$

This specification does not have an answer set. □

Example 3.5.7 *Let P consist of the formulae*

$$\sim p \leftarrow$$

$p \leftarrow$

$q \leftarrow \sim (p \wedge \sim p)$

This specification does not have an answer set. □

Example 3.5.8 *Let P consist of the formulae*

$p \leftarrow q \text{ or } \sim q$

$q \leftarrow s$

$\sim q \leftarrow t$

$t \leftarrow not\ s$

$s \leftarrow not\ t$

This specification have two answer sets: $\{t, \sim q, p\}$ and $\{s, q, p\}$. □

An OES is *non-contradictory* if it has a world view with no inconsistent belief sets (i.e. belief sets containing a pair of complementary literals). Thus a OES is *contradictory* if it has a world view with inconsistent belief sets. For instance, the OES consisting of the rules

$$\{p \text{ or } q \leftarrow; \sim q \leftarrow\}$$

is contradictory as it has the world view with a belief set that contains both $q, \sim q$.

3.6 Summary

In this chapter we reviewed some of the existing notions in the realm of logic programming and some of the logic programming frameworks. Based on these we shall develop our inconsistency handling frameworks in the next few chapters.

In Chapter 6 we use the notion of fixpoints based on an extension of the Van Emden Kowalski operator Θ, which we discussed in this chapter, in developing a constructive semantics for an inconsistency handling framework based on positive logic programs. As the framework is akin to a positive logic program which has a fixpoint, without going into details of the properties of the specification we consider it to have a fixpoint.

Chapter 4

Approach $\mathcal{C} - \mathcal{C}_d$

In this chapter we propose some fundamentally new ideas of handling inconsistency based on which we develop several logical systems which we will present in the following chapters.

This chapter is organized as follows. In the next section we introduce the new ideas of handling inconsistency. Based on these ideas we propose inconsistency handling strategies that we present in Section 4.2. In Section 4.3 we discuss the rationale behind them and briefly compare them to the existing inconsistency handling approaches. In Section 4.4 we explore the originality in our ideas of handling inconsistency. We conclude with a summary and pointers to developments in the next chapters.

A preliminary version of the ideas presented in this chapter has appeared in [36].

4.1 The inconsistency handling ideas

In this section we propose some new ideas for treating knowledge in the presence of contradiction. In particular inconsistency handling strategies based on these ideas would, instead of providing automatic belief revision or trivialization (as in classical logic), suggest a way out of the dilemma imposed by the presence of contradiction. The concept central to our idea consists of considering inconsistent knowledge as

being as relevant as any other knowledge, capturing it explicitly, reporting it and producing new interesting knowledge from it. The concepts are referred to as *explicit paraconsistency* and *reasoning beyond paraconsistency.*

This cannot be attained via modal treatments, nonmonotonic logics or truth maintenance devices and can be attained only approximately via many-valued logics. This is because such approaches work by preventing contradictions or by remedying the situations instantly if inconsistency occurs. But we are interested in 'coexisting' with the contradiction and possibly taking profit of it as interesting knowledge. This is in the sense that understanding the cause of contradictions provides additional knowledge. It is also profitable in the sense that contradiction driven reasoning provides a new horizon of information which would have never been arrived at, following the existing approaches of handling inconsistency. This allows us a wider spectrum of information, some *irrefutable*, some *defeasible* (the ones inferred from assumptions) and the new ones, *paraconsistent* (which captures the contradiction implicitly without making the theory inconsistent) and *contradiction affected*, which have different epistemicity than the others. It is a matter of philosophical intuition where the epistemicity of these new types of information stand with respect to the irrefutable and defeasible ones.

In clearer terms, we intend to discuss the basis for a system in which:

(a) conflicting situations can be tolerated, explicitly captured and reported, without making the system inconsistent (this we call *explicit paraconsistency*), and

(b) intuitively relevant knowledge can be obtained from this conflicting situation, from the rest of the global theory that is not directly affected by contradictory information, from the contradictory information and from parts of the theory directly or indirectly affected by contradictory information (this we call *reasoning beyond paraconsistency*).

4.2 Introducing the new elements of *Approach C* – C_d: C and C_d

The question that arises often in the logic community is how to handle contradiction. The consensus in the logic community is that contradiction is undesirable. The approach has been to free knowledge bases of contradiction completely, and to try to eradicate contradiction from knowledge bases by any means possible or to trivialize the inconsistent theory. Many of them [2, 32, 61] seem to agree that contradiction should not exist in a knowledge base and must be resolved somehow. But some researchers in this area contend the prevailing notion amongst many of the researchers on the basis of the rationale that inconsistency in large knowledge bases is inevitable. So to be more pragmatic we must have some means to reason in the presence of inconsistency. This is just one motivation for reasoning in the presence of inconsistency. We have discussed several others in the introductory chapter.

We propose a new approach to handle inconsistency. We start with the basic language of positive logic programs [50, 20, 69] and introduce *explicit*[1] negation.

Earlier, positive logic programs have been extended by negation, but the semantics given to it is not that of explicit negation, but of 'negation by default' [12, 41, 28, 21]. By introducing in the language of positive logic programs the notion of explicit negation, only the existence of a negated fact in the knowledge-base will make the strong notion of negation manifest itself in the theory, instead of *closed world assumption* [73] that makes negation by default manifest in the theory. (For more on the rationale behind introducing explicit negation to logic programs see [30, 53].)

Existence of a thesis and its explicit negation, together in a theory, causes contradiction. Because we have two kinds of negations we also have two kinds of inconsistency. The inconsistency arising on account of explicit negation is called *ontological*

[1] Explicit negation ($\sim$) is different from classical negation ($\neg$). This we have mentioned in Chapter 1 and will discuss in Chapter 5, Section 5.4.

inconsistency and the inconsistency arising on account of negation by default is called *epistemic inconsistency.* This we have discussed in Chapter 1, Section 1.5.

In this chapter we will deal with ontological inconsistency by restricting our language to that of positive logic programs plus explicit negation. [2] To handle ontological inconsistency, we introduce a new connective $\mathcal{C}$ to the existing language of positive logic programs plus explicit negation (denoted as $\sim$). Let α be an atom. $\mathcal{C}\alpha$ (or $\mathcal{C} \sim \alpha$) means α *(resp.* $\sim \alpha$*) is inconsistent or* α *(resp.* $\sim \alpha$*) is in contradiction*, i.e. there is contradictory information available about α (resp. $\sim \alpha$) in a theory. By our semantics if α and $\sim \alpha$ are in the same theory , we replace them by $\mathcal{C}\alpha$. From now on we will have $\mathcal{C}\alpha$ to capture both $\mathcal{C}\alpha$ and $\mathcal{C} \sim \alpha$. When we say $\mathcal{C}\alpha$ is **true**, we mean both $\mathcal{C}\alpha$ and $\mathcal{C} \sim \alpha$ are **true**.

Unlike some of the other logics handling inconsistency (refer to Chapter 2), we do not allow contradictory information to prevail in the theory but still maintain paraconsistency. By an introspective method we check the theory for the presence of contradiction. If contradiction is present we replace it by a sentence depicting it. In this way we avoid all the repercussions[3] of having contradictions in a theory and at the same time maintain the privileges[4] of paraconsistency. We name this particular strategy of handling inconsistency based on the idea of *explicit paraconsistency*, as **Approach** $\mathcal{C}$.

Approach $\mathcal{C}$ based systems are paraconsistent without the presence of explicit inconsistency. Considering a traditional paraconsistent system, if a set of formulae $\Sigma = \{\sim a, a\}$ and $\models_{para}$ denotes logical consequence in the system, then $\Sigma \models_{para} a$ (resp. $\sim a$), but $\Sigma \not\models_{para} b$ (the theory is not trivialized by concluding everything from inconsistency). *Approach* $\mathcal{C}$ integrated paraconsistent systems behaves differently. If

[2] We handle both ontological and epistemic inconsistency in Chapter 7 in the context of extended logic programs.

[3] Contradictory pairs of information explicitly present in a theory leads to inferences being made from them. This may propagate conclusions affected by contradictions without the knowledge of the reasoner. This is not warranted for.

[4] Ability to carry out nontrivial reasoning in presence of inconsistency.

$\models_C$ denotes the logical consequence for the system, then $\Sigma \not\models_C a$ (resp. $\sim a$), $\Sigma \not\models_C b$ and $\Sigma \models_C Ca$. By this approach inconsistent pairs (such as a and $\sim a$) are not explicitly present in the theories but replaced by the new element Ca. We refer to formulas like Ca as 'paraconsistent elements' as they contribute to the paraconsistency of the system. There are many advantages to this approach which we shall discuss in the next section.

In most of the logics handling inconsistencies, none distinguishes between conclusions drawn from inconsistent and consistent information. Moreover there is no logical mechanism in these logics to allow or disallow conclusions from inconsistent information as the situation demands.

The philosophical question that we face here is whether we should at all distinguish between conclusions arrived from inconsistent information and conclusions arrived from consistent information. Existing paraconsistent logics have not dealt with this fundamental question. We argue that we should distinguish between conclusions from inconsistent and consistent information as we understand that the conclusions derived from inconsistency have different epistemic strength than the ones derived from consistent information. This argument is somewhat similar to [58] who distinguishes between irrefutable and defeasible conclusions arrived via irrefutable and defeasible rules respectively.

The inconsistency handling strategy we propose here based on our second idea, deals with the issue of the distinction of conclusions drawn from inconsistent and consistent information. The logical systems that we propose based on this strategy are compact systems where logical devices can be easily built in to give us the option to allow or disallow conclusions from inconsistent information as the situation arises. So based on our inconsistency handling strategies not only are we able to go beyond inconsistency by introducing paraconsistent techniques, but we are also able to develop enhanced logical frameworks that are capable of *reasoning beyond paraconsistency.*

To develop the inconsistency handling strategy based on the idea of reasoning

beyond paraconsistency we introduce a new connective $\mathcal{C}_d$. A formula $\mathcal{C}_d\alpha$ means that *α is the consequence of a premise of a rule[5] 'affected' by inconsistent information.* There are three possible ways in which the premise of a rule can be affected by contradictory information: (1) The premise of a rule can have a contradiction, i.e. can have a formula of the form $\mathcal{C}\alpha$ in it and also $\mathcal{C}\alpha$ as a consequence of the rest of the theory containing the rule; (2) the premise can have a formula of the form $\mathcal{C}_d\alpha$ and a consequence $\mathcal{C}_d\alpha$ of the rest of the theory; or (3) the premise can have a formula α and the rest of the theory has $\mathcal{C}\alpha$ or $\mathcal{C}_d\alpha$ as a consequence.

The three possibilities of getting $\mathcal{C}_d\alpha$ as a consequence, discussed above, can be illustrated by the following examples.

Example 4.2.1

(1) $\sim a$; a; $b \leftarrow \mathcal{C}a$. Here we get $\mathcal{C}a$ as a consequence of the theory, and also get $\mathcal{C}_d b$ as a consequence as the premise of the rule having b at the head has $\mathcal{C}a$.

(2) $\sim a$; a; $b \leftarrow \mathcal{C}a$; $d \leftarrow \mathcal{C}_d b$. Here we get $\mathcal{C}a$ as a consequence of the theory, and also get $\mathcal{C}_d b$ as a consequence as the premise of the rule having b at the head has $\mathcal{C}a$. Finally we get $\mathcal{C}_d d$ as a consequence as the premise of the rule with the head d has $\mathcal{C}_d b$ which is a consequence of the theory.

(3) $\sim a$; a; $b \leftarrow a$. Here we get $\mathcal{C}a$ as a consequence of the theory. We also get $\mathcal{C}_d b$ as a consequence, as the premise of the rule having b at the head has a in the premise. □

The formal strategy we proposed based on the idea of *reasoning beyond paraconsistency* is termed **Approach $\mathcal{C}_d$**.

We name the combination of *Approach $\mathcal{C}$* and *Approach $\mathcal{C}_d$*: **Approach $\mathcal{C} - \mathcal{C}_d$**.

Application of *Approach $\mathcal{C} - \mathcal{C}_d$* to the language of positive logic programs [20, 50] with explicit negation gives us a new language that we call *Paraconsistent Specification (PS)*. We will formally define this in Chapter 5, Section 5.1.

[5] An extended positive logic program rule, extended by explicit negation, the paraconsistent connective $\mathcal{C}$ and the connective handling reasoning beyond paraconsistency $\mathcal{C}_d$.

4.3 *Approach* $\mathcal{C} - \mathcal{C}_d$ and its rationale

Example 4.3.1 *Consider a PS comprised of a collection of rules:*

1. $republican \leftarrow$
2. $quaker \leftarrow$
3. $pacifist \leftarrow quaker$
4. $\sim pacifist \leftarrow republican$

The unique conclusion set we would expect to get from the above set of clauses by Approach $\mathcal{C}$ *is:*

$\{republican, quaker, \mathcal{C}pacifist\}$. □

Keeping the above example in mind we will briefly discuss the advantages and appropriateness of handling inconsistency by *Approach* $\mathcal{C} - \mathcal{C}_d$ in the light of the existing approaches of (a) paraconsistency, (b) traditional information processing and (c) multi-valued logics.

4.3.1 *Approach* $\mathcal{C}$ and its rationale

In the paraconsistent approaches [42, 13, 14, 17] and extensions of it [11, 58, 93] the view is too liberal as complementary contradictory information is allowed to prevail in the theory without registering that they are in contradiction. This allows to derive freely further information with the help of one or both the contradictory information. This may result in further complications in the reasoning process giving rise to more contradictory information.

By *Approach* $\mathcal{C}$ we have the option to allow or disallow derivation from contradiction, as contradiction is captured explicitly and reported. This is possible as the builtin logical device (which we will discuss in the next chapter) is able to explicitly recognize the contradiction by capturing it in a paraconsistent form. This was not possible in the paraconsistent approaches as the contradiction was not explicitly recognized but implicitly allowed to exist. In some cases [45] contradiction is only

recognized implicitly in the process of application of a consequence operator.

In the traditional information processing approach a choice is made between one of the contradictory information based on certain preference criteria or none of the contradictory information is chosen to play safe. Sometimes the belief is revised [2, 62, 61] to make the theory consistent. By our understanding this approach is too decisive and we may lose some important information in many of these ways. For example a knowledge base comprised of the rules $a \leftarrow notb$, $\sim a \leftarrow notc$ and $e \leftarrow notb$, $notb$ and $notc$ are equally assumable. We can retract our assumption in both $notb$ and $notc$ to avoid contradiction, but then we are unable to derive e and thus lose important information.

Approach C has the merits of both the approaches that we have discussed till now. This is in the sense that it is liberal enough to accept a contradiction in a theory, to let it be derived from a theory and also to reason with it. At the same time by capturing it by the connective $\mathcal{C}$, it is registering the fact that contradictory information exists and are keeping the option open to choose both or one of the contradictory information, or discard both of them. It leaves the option open to decide at a later stage when more information may be available to choose between the contradictory information. This saves us from taking a hasty decision to choose between one of them or to discard both to reestablish consistency as soon as it appears, which requires us to maintain a cumbersome process of revision, by which possibly valuable information can be lost.

There is another approach, where inconsistency has been handled by multiple-valued logics [55, 9, 23]. The commonly used two extra truth-valuations for these multi-valued logics are *unknown*, denoted by the symbol $\bot$ and *inconsistent* denoted by the symbol $\top$. The second truth-valuation captures the case where a thesis and its negation are present in a theory. One of the basic differences between these approaches and ours is that we stick to a two-valued semantics. Instead of truth-valuating an atom α to be inconsistent we affirm that both α and $\sim \alpha$ are **true** in a theory, by introducing the formula $\mathcal{C}\alpha$ in the theory.

In Belnap's four-valued logic [55] saying α is believed both **true** and **false** (i.e. truth-valuation of α is $\top$) is different from the meaning captured by $\mathcal{C}a$. Multi-valued logics provides an inconsistent system with an epistemological meaning in the sense that the system can reflect beliefs of a reasoner (human or machine) who may happen to hold conflicting beliefs. The multi-valued logic systems remain ontologically inconsistent as far as they agreed that reality complies with the *laws of excluded middle and noncontradiction*, which means that in every possible state of the real world every statement is either **true** or **false** exclusively. By it, a statement can only be 'believed' **true** and **false** simultaneously, (thus capturing a kind of epistemic inconsistency), but cannot be **true** and **false** in reality (i.e. ontologically).

An *Approach C* based reasoner takes an 'objective' introspective view of its knowledge base. It does not internalize the contradictory information as its own but considers its presence and reports it. Where as, a multi-valued logic based reasoner views its knowledge base 'subjectively' and considers the contradiction to be its own belief. So *Approach C* captures the scenario (of combining knowledge bases or different experts feeding a knowledge base, thus resulting in inconsistency that we define as ontological inconsistency) more appropriately by our intuition, than the multi-valued approach which in a way reports self-inconsistency. Moreover, we do not perceive self-inconsistency to be a rational characteristic of a reasoner.

Can we allow contradictions in a theory without introducing the $\mathcal{C}$ operator? Some paraconsistent logics (e.g., [13, 15, 93]) tolerate contradiction and reason in its presence. But the problem that occurs there is that a theory containing contradictions answers **true** to both the complements of the contradiction. The knowledge base in Example 4.3.1 answers *yes* to both the queries $?pacifist$ and $? \sim pacifist$ when asked at different times or at the same time. We comprehend the existence of contradiction in the knowledge base if we ask the complementary queries at the same time. But if we ask the complementary queries at different times then the querier would remain ignorant about the contradiction. We can also modify the query answering mechanism

such that it always answers both questions $pacifist$ and $\sim pacifist$ whenever one of them is asked. But the problem lies in the meaning of the answer "yes" given to both $pacifist$ and $\sim pacifist$. When asked the same query to a system based on *Approach C* having the knowledge base as given by Example 4.3.1, it will answer that 'it has contradictory information available about $pacifist$'. Here the meaning is very explicit.

In [55, 9, 45, 11, 23] (i.e. the multi-valued approaches), the aforementioned problem does not occur. But as we have pointed out before our strategy provides a different approach to solving the problem and captures the inconsistent scenario more appropriately than the multi-valued approaches.

By *Approach C*, we have the choice to reason more cautiously, by choosing to allow or disallow further reasoning from contradiction depending on the circumstances. So one can define different modes of reasoning based on the circumstances where we allow or disallow reasoning from contradiction. We won't be investigating this aspect of our system in our work here, but will leave it for future investigation.

4.3.2 *Approach* C_d and its rationale

For the paraconsistent logics that allow reasoning based on contradiction, the complementary pairs of a contradiction would lead to different paths in reasoning. This has a significant implication. We are following both extensions in the reasoning path considering both of them equally possible, but without the comprehension that they are in contradiction. It is the case for many realistic situations that, each individual element of a contradictory pair leads to different lines of reasoning. The conclusions in the two different reasoning paths are often mutually contradictory in realistic settings. Hence we get more contradictions by not explicitly registering contradiction at the beginning.

By *Approach C* different paths of reasoning are not followed based on the individual elements of a contradictory pair. Instead a single path of reasoning is followed. As the

contradictory pairs are not separate elements (α and $\sim \alpha$), but a single entity ($C'a$), any reasoning based on either α or $\sim \alpha$ is recognized to stem from contradiction. This we will understand better with understanding of the working of *Approach* C_d below.

Let us add to the set of clauses in Example 4.3.1 at the beginning of the section, another clause:

$$5.\ popular \leftarrow pacifist.$$

We will get a revised conclusion set with $C_d popular$ added to the conclusion set of Example 4.3.1. Most paraconsistent logics that we have mentioned so far do not address the question whether we should allow or disallow derivation from a contradiction. Usually in paraconsistent logics contradictions are accepted as a basis for further derivation. But the epistemic weakness (or we can say the epistemic confusion regarding the status of a theory affected by contradiction) is not propagated to the conclusions derived from contradiction. We have no way to recognize these conclusions from the ones derived from noncontradictory premises, thus giving them the same epistemicity. By our intuition, information concluded from contradictory and non-contradictory premises should have different epistemic status.

In *RI logic* [45] and in paraconsistent semantics for logic programs [9][6], $\{q : t \rightarrow p : t; q : \top\} \models p : t$. Here implication allows for conclusions from inconsistent beliefs. $p : t$ derived here will have the same epistemic status as $p : f$ derived from $\{r : t \rightarrow p : f; r : t\} \models p : f$ belonging to the same knowledge-base. Though, on our understanding we should be distinguishing between these two derived beliefs. We would like to give the user of the logical system a chance to decide what they would like to accept from the two beliefs derived from distinctly different premises, one affected by contradiction and the other not affected by it. Our system would distinguish between the two derived beliefs by prefixing the conclusion $p : t$ derived

[6] In [45, 9] a basic element of the language is not an atom α, but an 'annotated literal' $\phi : b_l$, where ϕ is a literal (i.e. an atom or its negation) and $b_l \in \{t$ (for **true**), f (for **false**), $\perp$ (for **unknown**), $\top$ (for **inconsistent**) $\}$.

from the first premise by C_d.

On the other hand, multi-valued approaches [55, 23] over commit the conclusions from contradictory premises. For a theory $T = \{a; \sim a; a \rightarrow b\}$, the model is $\{a = \top, b = \top\}$. Here b is just entailed from contradiction, but not contradictory by itself.

By *Approach* C_d, the new connective C_d introduced to capture the conclusions derived from contradiction, gives us the advantage to distinguish between conclusions that are affected by contradictions and the ones that are not. This distinction lets the user of the logical system determine the epistemic strength he/she would like to assign to conclusions derived from contradictions and ones that are not derived from contradictions. Thus the builtin logical device in the system is able to propagate the information from one derivation to another that a conclusion has been affected by contradiction sometime in the process of its derivation. This opens up a new horizon of information (i.e. information related to inference from contradiction) and a new dynamics of interaction is added to the system, interaction between irrefutable information and contradiction affected information. We also do not over commit the contradictory status of the conclusions, as in multi-valued approaches.

One more significant contribution of *Approach* $C - C_d$ is that it enhances the expressive power of the language. We can now express the notions 'contradictory' and 'contradiction affected' in our language. In a suitably formalized system of non-monotonic reasoning, this may be used to prevent a default assumption from being committed because it would lead to a contradiction. In addition, a knowledge engineer can use these language constructs for the purpose of maintaining a knowledge system.

Approach $C - C_d$ enables the expansion of the horizon of reasoning as we are able to reason with 'contradiction affected' information, i.e. information that is directly and indirectly (or propagatively[7]) affected by contradictory information. For example,

[7]i.e. when affected by 'contradiction affected' information.

given the following theory:

$$\{a; \sim a; b \leftarrow a; c \leftarrow b\}$$

the conclusion set we get is:

$$\{\mathcal{C}a, \mathcal{C}_d b, \mathcal{C}_d c\}.$$

The line of reasoning is as follows: $\mathcal{C}a$ is arrived from a and $\sim a$; $\mathcal{C}_d b$ is arrived from $\mathcal{C}a$ and $b \leftarrow a$; and $\mathcal{C}_d c$ is arrived from $\mathcal{C}_d b$ and $c \leftarrow b$. Here, b is directly affected by contradiction and c is propagatively affected by contradiction.

Existing logics handling inconsistency either block off the whole realm of 'contradiction affected' information, or deal with it in an unintuitive way. We have shown this for multi-valued approaches [55], which over commit the valuation of a conclusion derived from contradiction, and for other paraconsistent logics [13, 45, 9, 93], which allow conclusions from contradictory information, but give them the same epistemic status as irrefutable conclusions.

In Chapter 9 we elaborately discuss the merits of *Approach* $\mathcal{C} - \mathcal{C}_d$ w.r.t. some of the existing paraconsistent logics and other logics handling inconsistency.

4.4 Insights into some fundamental aspects of our approach

In a knowledge base inconsistent information (which we represent here as an atomic proposition α and its negation $\sim \alpha$) can coexist. This situation arises, say, when two knowledge bases are collapsed together, or when two experts feed inconsistent information to a knowledge base. To handle this situation, the crucial facet of the stance we take in our formalism is that, α and $\sim \alpha$ are not related to each other in the usual classical logic sense, i.e. if α is **true** we should not conclude that $\sim \alpha$ is **false**. The truth theoretic valuation ('truth' or 'falsity') of α and that of $\sim \alpha$ in an interpretation are independent of one another. The existential status of α is decoupled from that of $\sim \alpha$: the facticity of α is not to preclude that of $\sim \alpha$.

	α	$\sim \alpha$
Case (1)	**true**	**true**
Case (2)	**true**	**false**
Case (3)	**false**	**true**
Case (4)	**false**	**false**

Figure 4.1: Truth-value assignment table

Accordingly there will be four distinct interpretive possibilities (truth-value assignment) of the pair α and $\sim \alpha$ (as given by Figure 4.4):

Case (1) captures the anomalous circumstance where we can say that the world is *ontologically overdetermined.* This situation is considered *inconsistent* in standard classical logic and truth-valuated to $\top$ (for 'inconsistent') in many-valued logic. We include this anomalous scenario in our logic. Instead of taking the refuge of the truth-value $\top$[8], we will capture the notion of ontological overdetermination syntactically and keep to a two-valued semantics.

The fundamentally new approach in our idea in handling inconsistency is that we have two levels in interpretation of the world:

In the lower level of interpretation we allow inconsistency to prevail. Thus we get the *truth-value assignment in an interpretation* of complementary literals α and $\sim \alpha$ as given in Figure 4.4. We will refer to this lower level of *interpretation in an inconsistent context* as just *interpretation* or alternatively as *truth-valuation in an inconsistent context.* We only deal with the basic elements of our language (i.e. atomic propositions and their negations) representing the world in this level of interpretation.

In the higher level of interpretation we do not allow inconsistency but let it prevail implicitly, thus following *Approach C* based on the idea of *explicit paraconsistency.* If both α and $\sim \alpha$ have the truth-value assignment **true** in the lower level of interpretation, we capture it syntactically by a new symbol $C\alpha$ and truth-valuate it to

[8]We have discussed earlier that the many-valued logical systems do not capture the scenario of our concern.

true in the higher level of interpretation. At the same time we do not allow α or $\sim \alpha$ to be truth-valuated to **true** at this higher level of interpretation, though they are **true** in the lower level. Thus we are capable of handling inconsistency without becoming inconsistent, thus attaining paraconsistency. We will refer to this higher level of *interpretation in a paraconsistent context* as *satisfaction by an interpretation* or alternatively as *truth-valuation in a paraconsistent context.* Now that we have sorted out how to interpret the basic elements in our language in the higher level of interpretation, we will be able to handle the interpretation of more complex elements of our language. Interpretation of complex elements of the language will be based on the interpretation of the basic elements.

Case (2) and case (3) can now be easily semantically interpreted. These are the cases where either α or $\sim \alpha$ exists by itself in a knowledge base. For the case where α (resp. $\sim \alpha$) exists by itself in a knowledge base, the truth-value **true** is assigned to α (resp. $\sim \alpha$) in an interpretation. But the satisfaction of α (resp. $\sim \alpha$) by an interpretation will be determined by first checking whether $C\alpha$ is satisfied/not satisfied by the interpretation. Thus we define the satisfaction of α (resp. $\sim \alpha$) by an interpretation with respect to the satisfaction of $C\alpha$.

In the case, where α (resp. $\sim \alpha$) is satisfied by an interpretation, the relation between α and $\sim \alpha$ is in the usual sense of classical logic. $\sim \alpha$ (resp. α) cannot be satisfied by an interpretation when α (resp. $\sim \alpha$) is satisfied by the interpretation.

Case (4) captures the case of *ontological underdetermination.* This situation depicts a knowledge-base where neither α nor $\sim \alpha$ exists. By taking a two-valued approach we can still deal with this situation, instead of giving into a many-valued semantics. There the situation would have been captured as α being assigned the truth-value $\perp$ (for *unknown*). A world where this situation arises is referred to as the *schematic world* by Rescher etal [75] and it is incomplete in nature from point of view of α. Such schematic worlds are accordingly, ontologically fuzzy or blank or illegitimate in their make-up: certain of their features are lost in the fog of indeter-

minacy. Rescher etal [75] have discussed these anomalous scenarios in non-standard worlds.

In our logic we deal individually with the existence/non-existence (obtainability/non-obtainability) of α and $\sim \alpha$ in a knowledge base. The status of a literal α (or $\sim \alpha$) not present in a knowledge base, is syntactically captured by *not*, giving a new symbol $not\alpha$ (or $not \sim \alpha$) as an add-on to our conclusions. Semantically we define $not\alpha$ (or $not \sim \alpha$) as: α (or $\sim \alpha$) is not satisfied by an interpretation. Semantically *not* coincides with the *negation by default* used in nonmonotonic systems. The nonsatisfaction by an interpretation of α (resp. $\sim \alpha$) does not immediately imply the satisfaction of $\sim \alpha$ (resp. α) by the interpretation. We distinguish between these two in our semantics. For only cases(2) and (3), the nonsatisfaction of α (or $\sim \alpha$) by an interpretation implies the satisfaction of $\sim \alpha$ (or α) by the interpretation; the nonassignment of the truth-value **true** (or **false**) to α (or $\sim \alpha$) implies the assignment of the truth-value **false** (or **true**) to $\sim \alpha$ (or α).

Together $not\alpha$ and $not \sim \alpha$ captures the case where α is assigned the truth-value $\perp$ (**unknown**) in a many-valued logic interpretation. In case (4) the assignment of the truth-value **false** to both α and $\sim \alpha$, as given by the truth-table, has the intuitive meaning of ontological underdetermination, which is intuitively close to the meaning of **unknown** in many-valued logic.

We also consider the scenario depicted by case (4) in building our logic. We discuss this in a later chapter where we apply *Approach* $C - C_d$ to extended logic programs [30], which already have *not*.

It is crucial to note, however, that while α and $\sim \alpha$ both can be assigned the truth-value **true** by an interpretation, we shall certainly never have $\alpha \wedge \sim \alpha$ satisfied in an interpretation. Our perspective is that two mutually inconsistent states of affairs might well both be realized. Whereas a single *self-inconsistent* state of affairs can never be realized. Contradictions can be realized distributively but not collectively: *self-inconsistency* must be excluded. We shall always have $\alpha \wedge \sim \alpha$ unsatisfied by an

interpretation. This will have its affect on the application of the logical connective $\wedge$ over formulas.

4.5 Summary

In this chapter we introduced the main ideas involved in our work for this thesis. We introduced *Approach* C and *Approach* C_d individually. Then we elaborately discussed the rationale of the approaches in handling inconsistency. Finally we explored some fundamentally new aspects of our approach to handle inconsistency.

In the next two chapters we formally present the PS system with its model theoretic and constructive semantics. In Chapter 7 we build a nonmonotonic system integrated with *Approach* $C - C_d$, which we extend in Chapter 8.

Chapter 5

Paraconsistent Specifications (PS)

In this chapter we will formally define the language of Paraconsistent Specifications (PS), define its model theoretic semantics and investigate the properties of the formalism.

The chapter is organized as follows: In Section 5.1 we explain the syntax of PS and we present its semantics in Section 5.2. In Section 5.3 we explore the basic properties of the formalism. In Section 5.4 we investigate some of the properties of the logical system in the broader perspective of paraconsistent logics. In Section 5.5 we discuss how we answer queries about the status of a formula by a theory based on our framework. Finally we conclude this chapter with a summarization of our results.

Preliminary version of the results presented in this chapter has appeared in: [36].

5.1 Syntax

Let us consider a language $\mathcal{L}$ consisting of atomic sentences a, b, c, ..., p, q, ..., logical connectives $\wedge$ (and), $\sim$ (explicit negation), the new ones $\mathcal{C}$, $\mathcal{C}_d$ and the connective $\leftarrow$ which we call *inference implication*. Formulas of $\mathcal{L}$ will be defined in the following way. Formula of the form p will be called *atoms*. A *simple literal* is an atom α or a negated atom of the form $\sim \alpha$. We use Φ to denote the set of all simple literals. $\ddot{\phi}$

denotes the atomic part of a simple literal ϕ. *Paraconsistent literals* are formulas of the form $C\ddot{\phi}$ and $C_d\phi$, where $\phi \in \Phi$. We use Ψ to denote the set of all simple literals and paraconsistent literals. We call an element $\psi \in \Psi$ a *literal.*

We will restrict ourself to a subset of the language of $\mathcal{L}$, which we call *Paraconsistent Specifications.* We allow contradictions to prevail in a theory of a PS by cautiously capturing them with C. C_d captures the 'inferentially implicated' consequences (derivations) from premises which have contradictions or are affected by contradictions. The formal notion of such a specification is captured by the following definition:

Definition 5.1.1 (Paraconsistent Specification (PS)) *By a paraconsistent specification P we mean a collection of rules of the form*

$$\phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$$

where $\phi \in \Phi$, ψ_is are formulae belonging to the set of literals Ψ of the language $\mathcal{L}$ and $m \geq 0$. □

For a rule $R : \phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$ in a PS P, by $concl(R)$ we mean the simple literal in the head of R, i.e. ϕ, by $prem(R)$ we mean the set of literals in the body of R, i.e. $\{\psi_1, \ldots, \psi_m\}$.

We have discussed in Section 4.4 Chapter 4, that by the semantics of PS (which we formally present in the next section), self-inconsistency can never be realized, i.e. $\phi \wedge \tilde{\phi}$ ($\phi \in \Phi$) can never be satisfied by an interpretation. Therefore, any rule R in a PS P, where $\phi, \tilde{\phi} \in prem(R)$ (we call this a *self-inconsistent rule*), loses its role in semantics. So one can assume that a PS is free of self-inconsistent rules without loss of generality. We consider any specification we develop in the later chapters to be also free of self-inconsistent rules.

A *plain rule* in a PS is a rule R such that $prem(R) \subseteq \Phi$.

So, the subset of our language $\mathcal{L}$ we consider here is that of PS, syntactically an extended form of *positive logic programs* [50, 20, 69]. Semantically it covers positive

logic programs. We will discuss this in the next chapter, after defining the semantics for PS.

5.2 Model theoretic semantics

In this section we will propose a model theoretic semantics for paraconsistent specifications. The approach is fundamentally different from the existing model theoretic semantics of logic programs, as we allow the theories to be paraconsistent, i.e. allow for nontrivial reasoning in presence of contradiction. We define the semantics of our logic by an extension of the standard notion of *interpretations*, which we call *p-interpretations*. Here '*p*' stands for two notions: (1) PS-specific, i.e. the existence of the elements in a p-interpretation depends on the PS being considered and (2) paraconsistent, i.e. because in a p-interpretation we also allow paraconsistent literals along with simple literals. Here we adhere to a two-valued semantics.

In our semantic definitions, we introduce some revisions to the existing classical concepts of *interpretation* and *satisfaction by an interpretation*. We use them to represent different notions. We propose two levels in interpreting the world: one at an inconsistent level which we refer to here as *interpretation* and the other at a paraconsistent level, which we refer to here as *satisfaction by an interpretation*. We have discussed these concepts in the last chapter in Section 4.4.

The truth-value assignment to a propositional symbol is arbitrary. In our semantics, we have dissociated the assignment of truth values of the simple literals α and $\sim \alpha$. They can both get the value **true** or **false** simultaneously, as we see in the truth-value assignment table in Figure 4.4, Chapter 4. By differentiating between the concept of *interpretation* and *satisfaction by an interpretation*, we are able to make α and $\sim \alpha$ not *satisfied by an interpretation*, though we let them to be simultaneously **true** in the interpretation. This differentiation is achieved by restricting the *satisfaction* by attaching some conditions coupled with the truth-value assignment to

true.

In the two-tier approach of our semantics, in the first tier, corresponding to the assignment of truth-values in an interpretation, only simple literals pertaining to the PS being considered are arbitrarily assigned the truth-values **true** or **false**. In the second tier, satisfaction of simple literals by an interpretation are guided by certain conditions. Thus a simple literal which is **true** by the interpretation may not be satisfied by an interpretation. In the second tier, in addition to the simple literals already satisfied by an interpretation, paraconsistent literals are brought into the picture. We now have paraconsistent literals satisfied by an interpretation depending on the PS and the interpretation being considered. The satisfaction of paraconsistent literals of the form $\mathcal{C}_d\phi$ by an interpretation is guided by the rules of a PS. This is clear from the meaning of $\mathcal{C}_d\phi$.

Satisfaction by an interpretation is guided by the specific PS for which the interpretation is being considered. Thus we rename *satisfaction by an interpretation* as *satisfaction by an interpretation w.r.t. a PS*. We will define a *p-interpretation of a PS* as a set that contains all the simple literals and paraconsistent literals satisfied by the satisfiability relations (i.e. satisfiability conditions) of *satisfaction by an interpretation w.r.t. the PS*. Models are based on these p-interpretations. All these notions would be better understood with the formal definitions that follow.

Let us now define some notations that we use in the formal definitions below. Let $\phi \in \Phi$, where Φ is the set of all simple literals in $\mathcal{L}$. ϕ is of the form α or $\sim \alpha$, where α is an atom. If $\phi = \alpha$ then $\tilde{\phi} = \sim \alpha$ and if $\tilde{\phi} = \alpha$ then $\phi = \sim \alpha$. We call $\tilde{\phi}$ the *complement* of ϕ and vice versa.

Let Φ_p be the set of all the simple literals in a PS P. For example, the set $\Phi_p = \{a, b, \sim b\}$ for the PS $P = \{a \leftarrow b;\ \sim b \leftarrow a\}$. We will now define an *interpretation* and *satisfaction by an interpretation w.r.t. a PS P*.

Definition 5.2.1 (Interpretation)

An interpretation I for P is a mapping from the set Φ_p *to* {**true**,**false**},

which we denote as:

$$I : \Phi_p \rightarrow \{\textbf{true}, \textbf{false}\}$$

□

Remark 5.2.1 *We will follow the convention used commonly in literature. $I(\phi)$ denotes the truth value of ϕ ($\in \Phi_p$) in an interpretation I. By an interpretation I we will also mean the set of all the simple literals $\phi \in \Phi_p$ such that $I(\phi) =$* **true**.

Definition 5.2.2 (Satisfaction by an interpretation w.r.t. a PS)
Let $\phi, \varphi \in \Phi$ and $\psi_i s \in \Psi$. To say that a literal ψ is satisfied *by an interpretation I w.r.t. a PS P, we write $I \models_{C\text{-}C_d} \psi$, and $I \not\models_{C\text{-}C_d} \psi$ to say that a literal ψ is* not satisfied *by I w.r.t. P.*

(1) $I \models_{C\text{-}C_d} \ddot{C\phi}$ *iff* $I(\phi) =$**true** *and* $I(\tilde{\phi}) =$**true**.

(2) $I \models_{C\text{-}C_d} C_d\phi$ *iff*

there exists a rule $R : \phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$ *in* P, *such that*

a (i) $I \models_{C\text{-}C_d} \psi_i$ *for all* $i = 1, \ldots, m$, *and*

(ii) there exists some paraconsistent literal $\ddot{C\varphi}$ *(or* $C_d\varphi$*)* $\in prem(R)$

or

b (i) $I \models_{C\text{-}C_d} \ddot{C\psi_i}$ *(resp.* $C_d\psi_i$*) for a* $\psi_i \in \Phi$, *and*

(ii) $I \models_{C\text{-}C_d} \psi_j$ *for any* ψ_j *for which condition (2b)(i) does not hold.*

(3) $I \models_{C\text{-}C_d} \phi$ *iff there exists a plain rule* $R : \phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$ *in* P, *such that* $I(\phi) =$ **true**, $I \not\models_{C\text{-}C_d} \ddot{C\phi}$ *and*

(i) $I \not\models_{C\text{-}C_d} C_d\phi$

or

(ii) $I \models_{C\text{-}C_d} \psi_i$ *for all* $i = 1, \ldots, m$.

□

Remark 5.2.2 *Recall the meaning of a literal denoting contradiction from Chapter 4, Section 4.2. The convention we follow is that we do not imply $C\phi$ or $C\tilde{\phi}$ to capture the meaning that ϕ (resp. $\tilde{\phi}$) is in contradiction. Instead we imply a contradiction in ϕ or $\tilde{\phi}$ with its atomic part. Hence we imply $C\ddot{\phi}$, which captures the meaning that both ϕ and $\tilde{\phi}$ are in contradiction.*

In Definition 5.2.2(1) we state that a "contradiction" (which we denote by a paraconsistent literal of the form $C\ddot{\phi}$) is satisfied by an interpretation I w.r.t. a PS P if and only if both the complementary simple literals ϕ and $\tilde{\phi}$ (w.r.t. an atom $\ddot{\phi}$) are **true** in the interpretation.

We have discussed in Section 4.4 Chapter 4 the case where both $I(\phi) =$ **false** and $I(\tilde{\phi}) =$ **false**. This is the case of ontological underdetermination, where nothing is known about a proposition or its complement. We do not particularly capture this case here in the second level of interpretation, i.e. satisfaction by an interpretation, based on the following rationale: Satisfaction by an interpretation is a meta-reflection of what can be "consistently **true**" and "paraconsistently **true**", but not about what can be "consistently **false**" and "paraconsistently **false**".

In Definition 5.2.2(2) we state that a "contradiction-affected" literal (i.e. a paraconsistent literal of the form $C_d\phi$) is satisfied by I w.r.t. P if and only if there exists a rule $R \in P$ such that one or both of the following two conditions hold:
(a) All the elements in the premise of R are satisfied by I w.r.t. P and there exists some contradictory or contradiction-affected literals in the premise of R.
(b) There exists some simple literal ψ_i in the premise of R, such that a contradictory or a contradiction-affected literal in ψ_i (i.e. a paraconsistent literal $C\ddot{\psi}_i$ or $C_d\psi$) is satisfied by I w.r.t. P, and the rest of the elements in the premise of R are all satisfied by I w.r.t. P.

If we allowed self-inconsistent rules in a PS we would have had to handle it in the following way: In the above condition (b), we would have had to include the sub-condition that the simple literal $\tilde{\psi}_i$ is not in the premise of the rule R when the simple

literal ψ_i is in the premise of R. This blocks the chance of getting a contradiction-affected literal from a rule R which has a self-inconsistent formula (i.e. $\psi_i \wedge \grave{\psi}_i$) in its body.

In Definition 5.2.2(3) we state that a simple literal ϕ is satisfied by an interpretation I w.r.t. a PS P if and only if there exists a plain rule R in P such that $concl(R) = \phi$, a paraconsistent literal $C\ddot{\phi}$ is not satisfied by I w.r.t. to P, ϕ is **true** in I and one or both of the following two conditions prevail:

(a) A paraconsistent literal $C_d\phi$ is not satisfied by I w.r.t. to P.

(b) All the elements in the premise of R are satisfied by I w.r.t. P.

The satisfaction of a simple literal ϕ requires

(*) the existence of a plain rule R, such that $concl(R) = \phi$.

This apparently may seem to be a little strange. Given a PS $P = \{a \leftarrow b\}$, why should we have $I \not\models_{C\text{-}C_d} b$, even if $I(b) =$ **true** and $I(a) =$ **true**. The reason for this is: By our semantics we try to exclude some of the irrelevant models of a PS which are unintuitive, in the sense that a reasoner cannot justify its beliefs based on the given specification. Ultimately our aim is to generate models for a PS which provide the intended meaning of a PS. So by setting the requirement (*) in the semantic definitions, we are excluding some of the models which are not significant w.r.t. our aim.

Now we define a p-interpretation of a PS based on which a model of the PS will be defined.

Definition 5.2.3 (P-interpretation)

A p-interpretation I_p of a PS P is a set of literals satisfied by an interpretation I with respect to P. □

Now we define the notion of *negative satisfaction* of a simple literal by an interpretation w.r.t. a PS. This notion defines the status of satisfaction of a simple literal w.r.t. its complement.

Definition 5.2.4 (Negative satisfaction by an interpretation w.r.t. a PS) *Let $\phi \in \Phi$. To say that a simple literal is* negatively satisfied by an interpretation I w.r.t. a PS, *we write* $I =|_{\sim C_d} \phi$.

$$I =|_{\sim C_d} \tilde{\phi} \text{ iff } I \models_{C_d} \phi. \ \square$$

Definition 5.2.4 is self-explanatory. It states the definition of the negative satisfaction of a simple literal in terms of the satisfaction of its complement. This gives us an insight into the existential dynamics of an atom and its explicit negation w.r.t. a p-interpretation. So when a theory is consistent, our system behaves like a classical system. (We prove this in the next chapter.) In a classical system when a proposition is **true** its negation is **false** and vice versa. In our system when a proposition is satisfied its explicit negation is negatively satisfied and vice versa.

In the following examples we enumerate the p-interpretations for PS.

Example 5.2.1 *Consider the PS P given below:*

$$\{a \leftarrow b;\ \sim b \leftarrow a\}$$

Let $I_1 = \emptyset$ be an interpretation. The corresponding p-interpretation $I_{p1} = \emptyset$.

Let $I_2 = \{a\}$ be an interpretation. The corresponding p-interpretation $I_{p2} = \{a\}$.

Let $I_3 = \{b\}$ be an interpretation. The corresponding p-interpretation $I_{p3} = \emptyset$.

Let $I_4 = \{\sim b\}$ be an interpretation. The corresponding p-interpretation $I_{p4} = \{\sim b\}$.

Let $I_5 = \{a, b\}$ be an interpretation. The corresponding p-interpretation $I_{p5} = \{a\}$.

Let $I_6 = \{a, \sim b\}$ be an interpretation. The corresponding p-interpretation $I_{p6} = \{a, \sim b\}$.

Let $I_7 = \{b, \sim b\}$ be an interpretation. The corresponding p-interpretation $I_{p7} = \{Cb, C_d a, C_d \sim b\}$.

Let $I_8 = \{a, b, \sim b\}$ be an interpretation. The corresponding p-interpretation $I_{p8} = \{Cb, C_d a, C_d \sim b\}$.

These are all the p-interpretations of P. $\square$

Example 5.2.2 *Consider the PS P given below:*

$$\{p \leftarrow; \sim p \leftarrow\}$$

Let $I_1 = \emptyset$ *be an interpretation. The corresponding p-interpretation* $I_{p1} = \emptyset$.

Let $I_2 = \{p\}$ *be an interpretation. The corresponding p-interpretation* $I_{p2} = \{p\}$.

Let $I_3 = \{\sim p\}$ *be an interpretation. The corresponding p-interpretation* $I_{p3} = \{\sim p\}$.

Let $I_4 = \{p, \sim p\}$ *be an interpretation. The corresponding p-interpretation* $I_{p4} = \{\mathcal{C}p\}$.

These are all the p-interpretations of P. □

Definition 5.2.5 (Satisfaction by a p-interpretation of a rule in a PS)

A rule R in a PS P of the form:

$$\phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$$

is satisfied by a p-interpretation I_p *iff one or more of the following conditions hold*

(1) $\psi_i \notin I_p$, $\mathcal{C}\ddot{\psi}_i \notin I_p$ *and* $\mathcal{C}_d\psi_i \notin I_p$ *for some* $1 \leq i \leq m$;

(2) $\phi \in I_p$;

(3) R is a plain rule, $\mathcal{C}\ddot{\phi} \in I_p$, ψ_i *or* $\mathcal{C}_d\psi_i \in I_p$ *for all i and there is a plain rule* $R' : \tilde{\phi} \leftarrow \psi_1' \wedge \ldots \wedge \psi_m'$ *in P, such that* ψ_i' *or* $\mathcal{C}_d\psi_i' \in I_p$ *for all i;*

(4) $\mathcal{C}_d\phi \in I_p$, *there exists a* $\psi_i \in \Phi$ *such that* $\mathcal{C}\ddot{\psi}_i$ *or* $\mathcal{C}_d\psi_i \in I_p$, *and* $\psi_j \in I_p$ *for all* $\psi_j \neq \psi_i$;

(5) $\mathcal{C}_d\phi \in I_p$, $\psi_i \in I_p$ *for all i and there exists a* ψ_i *that is a paraconsistent literal.*

□

In Definition 5.2.5 we state that a rule $R : \phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$ is satisfied by a p-interpretation I_p iff one or more of the following conditions hold:

(1) Any literal ψ_i in the premise of R or its corresponding paraconsistent forms (i.e. $\mathcal{C}\psi_i$ *or* $\mathcal{C}_d\psi_i$, such that ψ_i is a simple literal) are not in I_p.

(2) The conclusion of the rule R (i.e. ϕ) is in I_p.

(3) The contradiction in the atomic part of the conclusion of R (i.e. $\mathcal{C}\ddot{\phi}$) is in I_p, all

the literals in the premise of R (i.e. v_i where $i = 1, \ldots, m$) or their contradiction affected forms (i.e. $\mathcal{C}_d v_i$, such that v_i is a simple literal) are in I_p and there is a rule $R' \in P$, such that $concl(R') = \dot{o}$ and for all the literals in the premise of R', the literal or its contradiction-affected form (if the literal is a simple literal) is present in I_p.

(4) The contradiction-affected literal corresponding to the conclusion of a (i.e. $\mathcal{C}_d o$) is in I_p, there are some simple literals like v_i in the premise of R whose paraconsistent counterparts are present in I_p and the rest of the literals which do not have their paraconsistent forms in I_p (i.e. literals $v_j \neq v_i$) are in I_p.

(5) The contradiction-affected literal on the conclusion of R (i.e. $\mathcal{C}_d o$) is in I_p, all the literals in the premise of R are in I_p and among them some are paraconsistent literals.

Remark 5.2.3 *In this work as in [9, 30], we give a non-classical interpretation to the '$\leftarrow$' symbol, i.e. we do not treat it as a classical material implication. $\leftarrow$ behaves like a rule implication. If the literals in the premise of the rule or their paraconsistent forms are satisfied, then the conclusion of the rule or its paraconsistent forms are satisfied.*

Definition 5.2.6 (Model of a PS)

We say that a p-interpretation I_p, is a model *M of a PS P iff every rule R in P is satisfied by I_p.* □

Definition 5.2.7 (Minimal model of a PS)

A model M of a PS P is minimal, if no proper subset of M is a model of P. □

Example 5.2.3 *Consider Example 5.2.1. The p-interpretations I_{p2} and I_{p5} are not models of P. I_{p1}, I_{p3}, I_{p4}, I_{p6}, I_{p7} and I_{p8} are all the models of P.*
Clearly the minimal model of P is $M_p = I_{p1} = I_{p3} = \emptyset$. □

Example 5.2.4 *Consider Example 5.2.2. The p-interpretations I_{p1}, I_{p2} and I_{p3} are not models of P. Only I_{p4} is a model of P.*
Thus the minimal model of P: $M_p = I_{p4} = \{\mathcal{C}p\}$. □

Example 5.2.5 *Consider the PS P given below:*

$$\{p \leftarrow;\ q \leftarrow;\ r \leftarrow q;\ \sim q \leftarrow r\}$$

The p-interpretations of P are.

$$\begin{array}{lll} I_{p1} = \emptyset & I_{p6} = \{p, q\} & I_{p11} = \{p, q, r\} \\ I_{p2} = \{p\} & I_{p7} = \{p, r\} & I_{p12} = \{p, r, \sim q\} \\ I_{p3} = \{q\} & I_{p8} = \{p, \sim q\} & I_{p13} = \{\mathcal{C}q, \mathcal{C}_d r, \mathcal{C}_d \sim q\} \\ I_{p4} = \{r\} & I_{p9} = \{q, r\} & I_{p14} = \{\mathcal{C}q, \mathcal{C}_d r, \mathcal{C}_d \sim q, p\} \\ I_{p5} = \{\sim q\} & I_{p10} = \{r, \sim q\} & \end{array}$$

Only the p-interpretation I_{p4} (induced from the interpretation $\{p, q, \sim q\}$ or $\{p, q, \sim q, r\}$) is a model of P.
Thus I_{p14} is the minimal model of P. □

Example 5.2.6 *Consider the PS P given below:*

$$\{a \leftarrow;\ \sim a \leftarrow c;\ b \leftarrow a;\ \sim b \leftarrow a;\ a \leftarrow b\}$$

The only p-interpretation I_p which is a model of P and its corresponding interpretation I are:

$$I_p = \{a, \mathcal{C}b, \mathcal{C}_d b, \mathcal{C}_d \sim b, \mathcal{C}_d a\} \qquad I = \{a, b, \sim b\}$$

Thus I_p is the minimal model of P. □

Example 5.2.7 *Consider the PS P given below:*

$$\{\sim a \leftarrow;\ a \leftarrow;\ \sim a \leftarrow c;\ b \leftarrow a;\ \sim b \leftarrow a;\ a \leftarrow b\}$$

The p-interpretations which are models of P and their corresponding interpretations are:

$$I_{p1} = \{\mathcal{C}a, \mathcal{C}b, \mathcal{C}_d b, \mathcal{C}_d \sim b, \mathcal{C}_d a\} \qquad I_1 = \{a, \sim a, b, \sim b\}$$

$$I_{p2} = \{\mathcal{C}a, \mathcal{C}_3 b, \mathcal{C}_i \sim b, \mathcal{C}_d a\}$$

$$I_{2a} = \{a, \sim a\},\ I_{2b} = \{a, \sim a, b\},\ I_{2c} = \{a, \sim a, \sim b\}$$

(The p-interpretation I_{p2} has three corresponding interpretations.) Clearly, I_{p2} is the minimal model of P. □

Definition 5.2.8 (Entailment of a literal from a PS)

A PS P entails *a literal ψ, denoted by $P \models_{\mathcal{C}\text{-}\mathcal{C}_a} \psi$, iff every model of P is also a model of ψ.* □

Example 5.2.8 *Consider Example 5.2.1 and Example 5.2.3. I_{p1}, I_{p3}, I_{p4}, I_{p6}, I_{p7} and I_{p8} are the models of P.*
Therefore, $P \models_{\mathcal{C}\text{-}\mathcal{C}_a} \emptyset$. □

5.3 Basic properties of the formalism

In this section we investigate some of the properties of our formalism.

We first make an interesting observation from our formalism.

Observation 5.3.1 *Any simple literal ϕ ($\in \Phi_p$)* **true** in an interpretation I of a PS P *is not necessarily* satisfied by I w.r.t. P. □

Observation 5.3.1 is evident from Definition 5.2.2. For example, when simple literals ϕ and $\sim \phi$ belong to an interpretation I of a PS P, then $I \not\models_{\mathcal{C}\text{-}\mathcal{C}_a} \phi$ (resp. $\sim \phi$).

The following proposition states that complementary elements of a contradiction cannot both be satisfied by an interpretation.

Proposition 5.3.1 *A simple literal ϕ and its complement $\tilde{\phi}$ cannot both be satisfied by an interpretation w.r.t. a PS.* □

Proof: Let us assume that both a simple literal ϕ and its complement $\sim \phi$ are satisfied by an interpretation w.r.t. a PS P. By Definition 5.2.2 (3) if a simple literal ϕ (resp. $\tilde{\phi}$) is satisfied by an interpretation I w.r.t. P it has to be **true** in the interpretation

I. Now by Definition 5.2.2 (1) if both ϕ and $\dot{\phi}$ are **true** in I then $I \models_{c-c_d} \mathcal{C}\ddot{\phi}$. Now by Definition 5.2.2 (3) neither ϕ nor $\dot{\phi}$ can be satisfied by I if $I \models_{c-c_d} \mathcal{C}\ddot{\phi}$. Hence our initial assumption that both a simple literal ϕ and its complement $\dot{\phi}$ can be satisfied by I cannot be true. Thus our proposition is proved. □

The following example demonstrates the above proposition.

Example 5.3.1 *Consider the PS given below:*

$$\{p \leftarrow;\ q \leftarrow;\ r \leftarrow q;\ \sim q \leftarrow r;\ r \leftarrow p\}$$

Let $I = \{p, q, r, \sim q\}$ *be an interpretation. The corresponding p-interpretation* $I_p = \{\mathcal{C}q, \mathcal{C}_d r, \mathcal{C}_d \sim q, p, r\}$. I_p *is a model of the PS P. There are no other models for the P which is a subset of* I_p. *Thus the minimal model of the PS P:* $M_p = \{\mathcal{C}q, \mathcal{C}_d r, \mathcal{C}_d \sim q, p, r\}$.

Here $I(q) =$ **true** *and* $I(\sim q) =$ **true**. *But neither* $I \models_{c-c_d} q$ *nor* $I \models_{c-c_d} \sim q$. *This is because* $I \models_{c-c_d} \mathcal{C}q$ *obstructs the satisfaction of both q and* $\sim q$ *by I.* □

In the following proposition we state that an implicit contradiction (i.e. a paraconsistent literal of the form $\mathcal{C}\ddot{\phi}$) and its originating causes (i.e. ϕ and $\dot{\phi}$) cannot be in the same theory.

Proposition 5.3.2 *Let* I_p *be a p-interpretation of a PS P. If a paraconsistent literal* $\mathcal{C}\ddot{\phi} \in I_p$, *neither* ϕ *nor* $\dot{\phi}$ *can belong to* I_p. □

Proof: If $\mathcal{C}\ddot{\phi} \in I_p$, a p-interpretation w.r.t. a PS P, then there is an interpretation I of P such that $I \models_{c-c_d} \mathcal{C}\ddot{\phi}$ (by Definition 5.2.3) . If $I \models_{c-c_d} \mathcal{C}\ddot{\phi}$ then by Definition 5.2.2 (1), $\phi \in I$ and $\dot{\phi} \in I$. Now by Definition 5.2.2 (3), $I \not\models_{c-c_d} \phi$ (resp. $\dot{\phi}$) as $I \models_{c-c_d} \mathcal{C}\ddot{\phi}$. Hence by Definition 5.2.3 neither ϕ nor $\dot{\phi}$ can belong to I_p. □

Proposition 5.3.2 is also well demonstrated by Example 5.3.1.

The following proposition states that 'contradiction-affected' literals owes its origin to other 'contradictory' or 'contradiction-affected' literals.

Proposition 5.3.3 *Let M be a model of a PS P. If $\mathcal{C}_d\phi \in M$ then there exists a rule $R : \phi \leftarrow \psi_1 \wedge \cdots \wedge \psi_m$ in P such that (1) there is at least one paraconsistent literal $\psi_i \in \{\psi_1, \cdots, \psi_m\}$ such that $\psi_i \in M$ or (2) there is at least one paraconsistent literal $\mathcal{C}\ddot{\varphi}$ (or $\mathcal{C}_d\varphi$) $\in M$ ($\varphi \in \Phi$), such that $\varphi \in \{\psi_1, \cdots, \psi_m\}$.* □

Proof: Let P be a PS and M be a model of P. Let $\mathcal{C}_d\phi \in M$. Then by Definitions 5.2.5 and 5.2.6 there exists a rule $R : \phi \leftarrow \psi_1 \wedge \cdots \wedge \psi_m$ in P, such that $I \models_{\mathcal{C}-\mathcal{C}_d} \mathcal{C}_d\phi$, where I is the interpretation corresponding to the model M. By Definition 5.2.2(2), if (1) $I \models_{\mathcal{C}-\mathcal{C}_d} \psi_i$ for all $i = 1, \cdots, m$ and any one of the ψ_i is a paraconsistent literal or (2) $I \models_{\mathcal{C}-\mathcal{C}_d} \mathcal{C}\ddot{\varphi}$ (or $\mathcal{C}_d\varphi$), where $\varphi \in \{\psi_1, \cdots, \psi_m\}$ and $I \models_{\mathcal{C}-\mathcal{C}_d} \psi$ for the rest of the ψ_is, then $I \models_{\mathcal{C}-\mathcal{C}_d} \mathcal{C}_d\phi$. Now by Definitions 5.2.5 and 5.2.6 as (1) $I \models_{\mathcal{C}-\mathcal{C}_d} \psi_i$ for some ψ_i or (2) $I \models_{\mathcal{C}-\mathcal{C}_d} \mathcal{C}\ddot{\varphi}$ (or $\mathcal{C}_d\varphi$), therefore, $\psi_i \in M$ or $\mathcal{C}\ddot{\varphi}$ (*or* $\mathcal{C}_d\varphi$) $\in M$. Hence proved. □.

Example 5.3.2 *Consider the PS given below:*

$$\{b \leftarrow;\ \sim b \leftarrow;\ \ a \leftarrow b;\ c \leftarrow \mathcal{C}_d a\}$$

Let $I = \{b, \sim b, a\}$ and $I' = \{b, \sim b\}$ be two interpretations. The p-interpretation corresponding both I and I' is $I_p = \{\mathcal{C}b, \mathcal{C}_d a, \mathcal{C}_d c\}$. I_p is a model of P. There are no other models for P which is a subset of I_p. Thus the minimal model of the PS P: $M_p = \{\mathcal{C}b, \mathcal{C}_d a, \mathcal{C}_d c\}$.

This example demonstrates Proposition 5.3.3. $\mathcal{C}_d a \in M_p$ by the rule $a \leftarrow b$ as $\mathcal{C}b \in M_p$. $\mathcal{C}_d c \in M_p$ by the rule $c \leftarrow \mathcal{C}_d a$ as $\mathcal{C}_d a \in M_p$. □

5.4 Properties of our formalism on a broader perspective

By adopting a two-valued semantics for our framework and dissociating the truth valuation of an atom α and an explicitly negated atom $\sim \alpha$, we have some interesting outcomes. As an atom α and its negation $\sim \alpha$ can both have the same truth-value

false simultaneously, we get the case where a PS P does not entail $\alpha \vee \sim \alpha$, where $\vee$ denotes classical disjunction; i.e.

$$P \not\models_{c\text{-}c_a} \alpha \vee \sim \alpha.$$

Thus we see that the use of the given semantics clearly implies that the *law of excluded middle* does not necessarily hold for explicit negation. Obviously then, explicit negation differs from classical negation ($\neg$) which, being defined for a two-valued logic (namely, classical logic), satisfies the mentioned law.

But interestingly, the formalism we have defined retains the *law of non-contradiction* that characterizes a consistent system, and naturally, also holds for classical logic, as consistency is an important criteria for it. The law of non-contradiction implies:

$$P \not\models_{c\text{-}c_a} \alpha \wedge \sim \alpha$$

This is what we expect from a *self-consistent* system. The system we have defined here is *paraconsistent*, that is, it allows reasoning in presence of inconsistent information but is self-consistent. A self-consistent system may have contradictory information in it coming from diverse sources, but should not by itself conclude a contradiction, such as the affirmation of "α and $\sim \alpha$" from two separate entities α and $\sim \alpha$.

So we see that the *adjunction principle*, i.e.

$$\{\alpha;\ \beta\} \models \alpha \wedge \beta$$

is restricted in our semantics. This is evident from the semantical definitions as given by Definition 5.2.2(1-2), by which either

$$I \models_{c\text{-}c_a} \alpha, \text{ or}$$

$$I \models_{c\text{-}c_a} \sim \alpha,$$

giving no scope for

$$I \models_{c\text{-}c_a} \alpha \wedge \sim \alpha.$$

5.5 Status answering based on entailment

Here we discuss how we answer a query regarding the *status* (i.e. the existential state by a theory based on what the theory entails) of a formula. This notion of status of a formula will become clearer as we proceed with the section. We allow queries (which we denote by q) of the following types:

- $q = \phi$, where ϕ is a simple literal
- $q = \mathcal{C}\phi$, where ϕ is a simple literal
- $q = \mathcal{C}_d\phi$, where ϕ is a simple literal
- $q = q_1 \wedge q_2$, where q_is are queries
- $q = q_1 \vee q_2$, where q_is are queries

We define the correct answers to a status query with respect to a PS as:

"yes", "no", "unknown", "in-contradiction" and "contradiction-affected".

If $q = \phi$,

$$answer(P,q) = \begin{cases} yes & P \models_{c\text{-}c_d} \phi \\ no & P \models_{c\text{-}c_d} \tilde{\phi} \\ unknown & P \not\models_{c\text{-}c_d} \phi,\ P \not\models_{c\text{-}c_d} \tilde{\phi}, \\ & P \not\models_{c\text{-}c_d} \mathcal{C}\tilde{\phi},\ P \not\models_{c\text{-}c_d} \mathcal{C}_d\phi \\ in-contradiction & P \models_{c\text{-}c_d} \mathcal{C}\tilde{\phi} \\ contradiction-affected & P \models_{c\text{-}c_d} \mathcal{C}_d\phi \end{cases}$$

We also get some combination answers:

$$(yes,\ contradiction-affected)$$

$$(no,\ contradiction-affected)$$

$$(in-contradiction,\ contradiction-affected)$$

In these cases it would be up to the querier to choose the appropriate answer depending on his/her philosophical conviction. If one believes that a strong "yes" (resp. "no") has a higher epistemic status than the answer "contradiction-affected", then the choice of either "yes" (resp. "no") is clear. Again, for queriers both "in-contradiction" and "contradiction-affected" may have equal epistemic status. Thus both will be considered by them.

Now we will give answers to the other kinds of queries:

If $q = \mathcal{C}\ddot{\phi}$,

$$answer(P, q) = \begin{cases} yes & P \models_{\mathcal{C}\text{-}\mathcal{C}_d} \mathcal{C}\ddot{\phi} \\ no & otherwise \end{cases}$$

If $q = \mathcal{C}_d\phi$,

$$answer(P, q) = \begin{cases} yes & P \models_{\mathcal{C}\text{-}\mathcal{C}_d} \mathcal{C}_d\phi \\ no & otherwise \end{cases}$$

If $q = q_1 \wedge q_2$,

$$answer(P, q) = \begin{cases} yes & answer(P, q_1) = yes \text{ and } answer(P, q_2) = yes \\ no & otherwise \end{cases}$$

If $q = q_1 \vee q_2$,

$$answer(P, q) = \begin{cases} yes & answer(P, q_1) = yes \text{ or } answer(P, q_2) = yes \\ no & answer(P, q_1) = no \text{ and } answer(P, q_2) = no \end{cases}$$

5.6 Summary

In this chapter we apply the inconsistency handling strategies, *Approach* $\mathcal{C}$ and *Approach* $\mathcal{C}_d$ to the language of positive logic programs with explicit negation. This

enhanced the language, enabling the expanded language PS to handle inconsistency in a pragmatic way. We also presented a model theoretic semantics and investigated the properties of our formalism.

In the next chapter we follow up with a constructive semantics for PS and investigate the correspondence between the model theoretic and constructive semantics.

Chapter 6

Paraconsistent Specifications: Constructive Semantics

In this chapter we present a constructive formulation to compute the minimal model of a PS. This constructive definition is based on an iterative process: an agent incrementally establishes its beliefs by iterating a composite function $\mathcal{R} \circ \mathcal{D}$, where $\mathcal{D}$ is a *deductive operator* that generates the literals in a PS that are derivable and $\mathcal{R}$ is a *deduction revision operator* that revises the deductions generated by $\mathcal{D}$ if they are *contradictory* themselves or are *affected by contradiction.*

This chapter is organized as follows. The next section presents the basic definitions needed to formulate the constructive framework. Section 6.2 presents the constructive formulation. Section 6.3 describes the properties of the formalization. In Section 6.5 we explore the relationship between the constructive and the model-theoretic semantics of PS. In Section 6.6 we relate the constructive semantics of PS to the fixpoint semantics of positive logic programs. We conclude in Section 6.7 with a summarization of the work which we report in this chapter.

6.1 Some definitions

In this section we give some definitions that enable us to formalize our constructive framework.

Our constructive formulation is based on iteratively using the composite function $\mathcal{R} \circ \mathcal{D}$ on a structure. We call this structure a *deduction graph* or *d-graph* as we use it to capture a deductive process. We will now formally define a d-graph.

Definition 6.1.1 (D-graph) *A* d-graph G *is a directed graph with each node containing one or more literals.* □

We will call a d-graph without any nodes as a *null deduction graph* or an *nd-graph*.

For defining a transformation function on d-graphs, we will use an operator T_p over sets of literals and paraconsistent literals. This is an extension of the standard *immediate consequence operator* (refer to Chapter 3). We will define it below.

Definition 6.1.2 *Let $\phi, \varphi \in \Phi$, P be a PS and S be a subset of Ψ. The consequence operator T_p is defined as follows:*

$$T_p(S) = \{\phi \mid \textit{there exists a rule } R \textit{ in } P \textit{ such that } concl(R) = \phi,\ prem(R) \subseteq S\}$$

□

Clearly T_p is monotonic. Hence it has a fixpoint.

Let $\mathcal{G}$ be the set of all d-graphs. Now we will define a transformation operator ($W_{S,T_p} : \mathcal{G} \rightarrow \mathcal{G}$) based on T_p and $S \subseteq \Psi$ as follows:

Definition 6.1.3 *Let P be a PS, R be a rule in P, G be a d-graph and S be a subset of Ψ. $W_{S,T_p}(G)$ is defined for each $\phi \in T_p(S)$ and each rule R with $\phi = concl(R)$ as follows:*

1. *If $prem(R) = \emptyset$,*
   (a) *if in G a node with ϕ does not already exist, create it, and draw an arc pointing to it from a null node;*

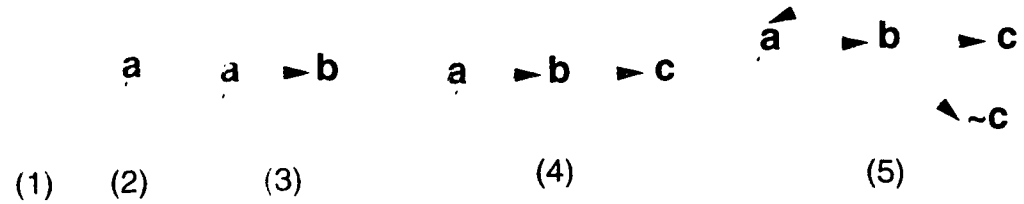

Figure 6.2: Operation W_{T_p} on d-graphs

(b) if in G a node with ϕ already exists, but does not have an arc pointing to it from a null node, draw such an arc.

2. *If $prem(R) \subseteq S$, create in G a new node with ϕ if it does not already exist, and draw an arc from each node having an element of $prem(R)$ to the node with ϕ, if such an arc does not already exist.*

3. *If there exists a literal $\varphi \in prem(R)$ such that $C\ddot{\varphi}$ (or $C_d\varphi$) $\in S$ and the remaining elements of $prem(R)$ $(\neq \varphi)$ belongs to S;*
create in G a new node with ϕ, if it does not already exist, and draw an arc from each node in G with the elements $C\ddot{\varphi}$ (or $C_d\varphi$) and also from each of the nodes in G with an element $\in prem(R)$ but $\neq \varphi$, to the node with ϕ, if such an arc does not already exist.

□

Remark 6.1.1 *From now on we denote the operator W_{S,T_p} as W_{T_p} because T_p is based on $S \subseteq \Psi$, and denoting W_{S,T_p} as W_{T_p} without the reference of S is implicit in the reference of T_p in W_{T_p}.*

W_{T_p} is based on T_P. Hence it is also monotonic. More precisely, $S_1 \subseteq S_2$, $G_1 \sqsubseteq G_2$ (where S_1, S_2 are subsets of Ψ, $G_1, G_2 \in \mathcal{G}$ and $\sqsubseteq$ denotes *less or equally complex*) implies $W_{T_p}(G_1) \sqsubseteq W_{T_p}(G_2)$. $W_{T_p}(G)$ $(G \in \mathcal{G})$ is no less complex than G. So W_{Tp} has a fixpoint. In the following example we will show the operation of W_{T_p} over d-graphs.

Example 6.1.1 *Let $P = \{b \leftarrow a;\ c \leftarrow b;\ \sim c \leftarrow b;\ a \leftarrow c;\ a \leftarrow\}$. S be an empty set and $G_\emptyset$ be the the corresponding nd-graph as given by Figure 6.2(1). The d-graphs which we get from P by applying W_{T_p} on the initial d-graph $G_\emptyset$ are given in the Figure 6.2 above. The d-graphs are in an incremental order of growth from left to right. The nodes of the d-graph in Figure 6.2 above have all the literals which are derivable from P by the T_p operator.* □

6.2 Constructive formulation

In this section, we define the operators $\mathcal{D}$ and $\mathcal{R}$. Then we define a composite function based on the two operators. This we use to generate a set of literals and paraconsistent literals w.r.t. a PS. We call this set *generated set* or *g-set*. We will later show that a g-set of a PS corresponds to the minimal model of the PS.

We define $\mathcal{D}$ based on W_{T_p}. The function $\mathcal{D}$ when applied to a d-graph w.r.t. a PS, exhausts all the rules in the PS to give a d-graph that cannot grow any more.

Definition 6.2.1 *Let P be a PS, G be a d-graph on P and S be the set of all the literals in the nodes in G. We define the function $\mathcal{D}$ on G as follows:*

$$\mathcal{D}(G) = W_{T_p} \uparrow^{\omega}$$

□

Conventionally, the notation $T_G \uparrow^{\omega}$ means the least fixpoint of T_G which is constructed from the smallest element in the domain. But here we change the meaning of the notation such that the construction can start from any element of the domain $\mathcal{G}$.

The example below illustrates the above definition.

Example 6.2.1 *Let us consider the PS P in Example 6.1.1. Let us consider an nd-graph $G_\emptyset$. Here $S = \emptyset$, the set of nodes of $G_\emptyset$. On applying $\mathcal{D}$ on $G_\emptyset$ w.r.t. P, by Definition 6.2.1, we get the d-graph G' in Figure 6.2(5). The intermediate steps from $G_\emptyset$ to G' which follows from Definition 6.1.3 are given by Figure 6.2(2-4).* □

Now we define the operator $\mathcal{R}$ over d-graphs. This function modifies a d-graph to produce another d-graph, if certain conditions prevail in the former d-graph.

Definition 6.2.2 *Let G be a d-graph and N_G be the set of all the elements in the nodes in G. $\mathcal{R}$ is a function on G, such that it gives a modified d-graph G_c by following the steps:*

step 1: If there are nodes with ϕ and $\dot{\phi}$ in the initial unmodified d-graph G, i.e. $\{\phi, \dot{\phi}\} \subseteq N_G$, and there are no directed paths to them from a node with paraconsistent literals or a node with a simple literal φ such that $\dot{\varphi} \notin N_G$, replace them by $\mathcal{C}\phi$ and $\mathcal{C}\dot{\phi}$ respectively.

step 2: If in G (possibly modified by step 1) there are nodes with ϕ pointed by an arc coming from a node with a paraconsistent literal

(a) add $\mathcal{C}_d\phi$ as another element to the node along with ϕ, if there is a directed path from null to the node with ϕ without intermediate nodes with paraconsistent literals;

(b) replace ϕ in the node with $\mathcal{C}_d\phi$, otherwise.

step 3: If in G (possibly modified by steps 1 or 2), there are nodes with $\mathcal{C}_d\phi$ pointing to nodes with $\mathcal{C}\varphi$, add a new element $\mathcal{C}_d\varphi$ to nodes with $\mathcal{C}\varphi$.

step 4: Repeat steps (2) and (3) until the prerequisites for the two steps are not fulfilled anymore.

step 5: Replace all nodes in G (possibly modified by steps 1, 2, 3 or 4) with formulas of the form $\mathcal{C}\phi$ or $\mathcal{C}\dot{\phi}$ by $\mathcal{C}\ddot{\phi}$.

□

Definition 6.2.3 (F-stable d-graphs) *A d-graph G is* F-stable *if for a function $\mathcal{F}$*

$$\mathcal{F}(G) = G$$

□

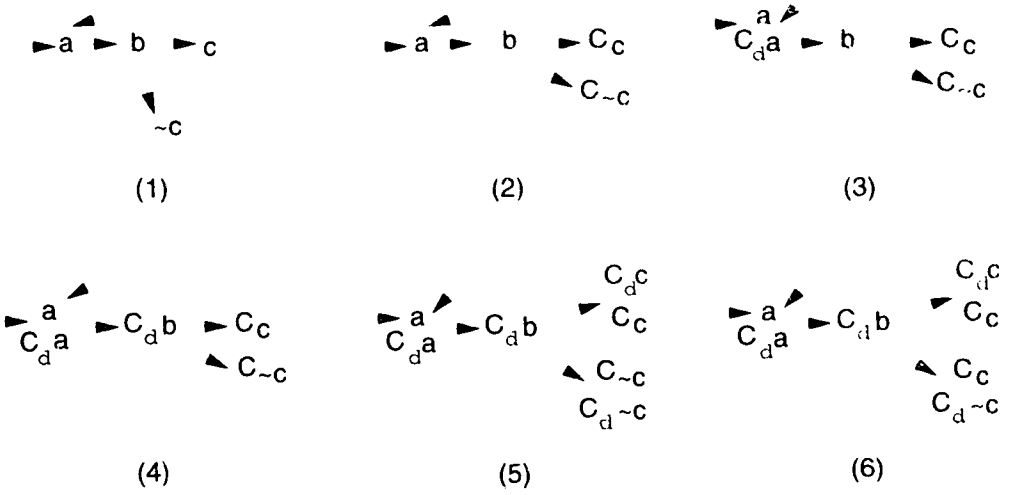

Figure 6.3: Operation $\mathcal{R}$ on d-graphs

It is obvious that $\mathcal{R}$ applied to a d-graph G results in a R-stable d-graph G_c. We will now demonstrate how this definition works with the following example.

Example 6.2.2 *Let us consider the PS P in Example 6.2.1 (which is the same as the one in Example 6.2.1). Let us also consider the graph G' in Example 6.2.1 given by Figure 6.3(1). We apply the function $\mathcal{R}$ on G' and get the d-graph G_c given by Figure 6.3(6) in the following steps:*

step 1: By Definition 6.2.2(1) we get the d-graph in Figure 6.3(2) starting from the d-graph G' in Figure 6.3(1).

step 2: By Definition 6.2.2(2a) we get the d-graph in Figure 6.3(3) starting from the d-graph in Figure 6.3(2).

step 3: By Definition 6.2.2(2b) we get the d-graph in Figure 6.3(4) starting from the d-graph in Figure 6.3(3).

step 4: By Definition 6.2.2(3) we get the d-graph in Figure 6.3(5) starting from the d-graph in Figure 6.3(4).

step 5: Definition 6.2.2(4) cannot be applicable to the d-graph in Figure 6.3(5). So we go to the next definitional step.

step 6: Definition 6.2.2(5) when applied to the d-graph in Figure 6.3(5) gives the d-graph G_c in Figure 6.3(6), which is the final product of the application of function $\mathcal{R}$ on G'.

□

We can now define a composite function $\mathcal{A}$ as follows:

$$\mathcal{A} = \mathcal{R} \circ \mathcal{D} \tag{6.1}$$

$\mathcal{A}$ applied repetitively on an nd-graph w.r.t. a PS culminates into a A-stable d-graph, because $\mathcal{D}$ is monotonic and $\mathcal{R}$ produces a R-stable d-graph. We call the set of literals we get w.r.t. the A-stable d-graph a *generated set* or *g-set*. We will show later that this set corresponds to the minimal model of the PS. We formalize the idea below:

Definition 6.2.4 (G-set of a PS) *Let $G_\emptyset$ be an nd-graph, G_is be d-graphs and P be a PS. A g-set is the set of all the literals in the nodes of a d-graph G_f such that*

$$G_1 = \mathcal{A}(G_\emptyset)$$
$$G_i = \mathcal{A}(G_{i-1})$$
$$G_f = \mathcal{A}(G_f).$$

□

Example 6.2.3 *Consider the PS P in Example 6.1.1. The corresponding g-set for P is given by the set $N_{G_c} = \{a, \mathcal{C}_d a, \mathcal{C}_d b, \mathcal{C}c, \mathcal{C}_d c, \mathcal{C}_d \sim c\}$ comprised of all the elements in the nodes of the d-graph G_c given in Figure 6.3(6).* □

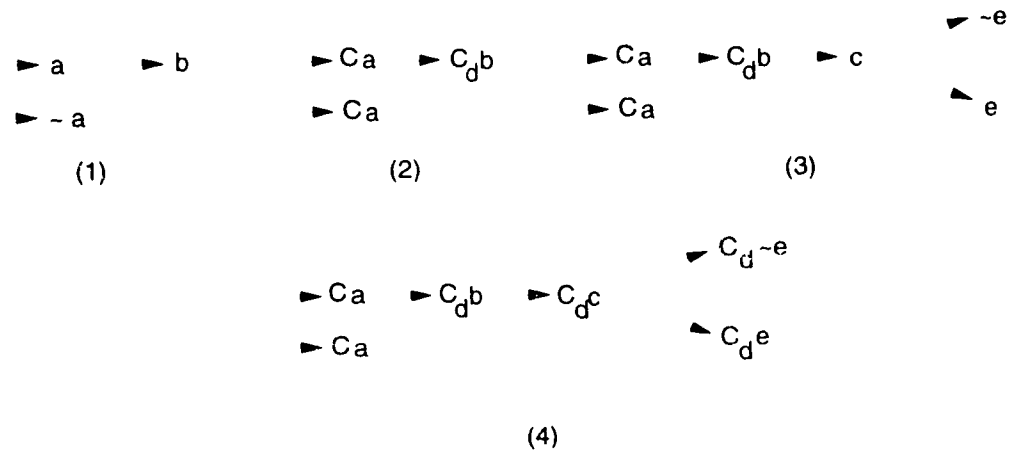

Figure 6.4: Operation $\mathcal{A}$ on d-graphs

Example 6.2.4 *Consider the PS* $P =$

$$\{a \leftarrow; \sim a \leftarrow; b \leftarrow a; c \leftarrow \mathcal{C}_d b; c \leftarrow c; \sim c \leftarrow c\}.$$

Let $G_\emptyset$ *be an nd-graph.*

The d-graph $\mathcal{D}(G_\emptyset)$ *towards the construction of* G_1 *is given by Figure 6.4(1).*

The d-graph $G_1 = \mathcal{A}(G_\emptyset)$ *is given by Figure 6.4(2).*

The d-graph $\mathcal{D}(G_1)$ *towards the construction* $\mathcal{A}G_1$ *is given by Figure 6.4(3).*

The d-graph $G_2 = \mathcal{A}(G_1)$ *is given by Figure 6.4(4).*

The d-graph $\mathcal{A}(G_2) = G_2$.

Hence the g-set of P is

$$\{\mathcal{C}a, \mathcal{C}_d b, \mathcal{C}_d c, \mathcal{C}_d c, \mathcal{C}_d \sim c\}$$

i.e. the set of all the elements in the nodes of the d-graph G_2. □

6.3 Properties of the Constructive Semantics

In this section we will investigate some properties of our constructive formulation.

We propose and prove the uniqueness of g-set of a PS.

Proposition 6.3.1 *A PS P has a unique g-set.*

Proof: Let a PS P have two g-sets N_G and $N_{G'}$. Let G and G' be the corresponding

d-graphs for the g-sets N_G and $N_{G'}$ respectively. G and G' must be arrived as two different A-stable d-graphs using the function $\mathcal{A}$ w.r.t. P. For this to happen the function $\mathcal{A}$ has to be non-deterministic, i.e. at a point of derivation of the sequence of d-graphs by the function $\mathcal{A}$, two different d-graphs should be derivable on applying $\mathcal{A}$ on the same d-graph. This would only happen if one of the composing functions of the composite function $\mathcal{A}$ is non-deterministic. But by definition both $\mathcal{D}$ and $\mathcal{R}$ are deterministic. Hence we get only one g-set of P. □

We state some propositions expressing some of the intrinsic properties of our formulation.

Proposition 6.3.2 *A simple literal ϕ and its complement $\tilde{\phi}$ cannot both be in the g-set of a PS.* □

Proposition 6.3.3 *Let N_G be the g-set of a PS P. If a paraconsistent literal $\mathcal{C}\ddot{\phi} \in N_G$, neither ϕ nor $\dot{\phi}$ belongs to N_G.* □

Proposition 6.3.4 *Let $R : \phi \leftarrow \psi_1 \wedge \cdots \wedge \psi_m$ be a rule in a PS P and N_G be the g-set of P. If $\mathcal{C}_d\phi \in N_G$ then (1) there is at least one paraconsistent literal $\psi_i \in \{\psi_1, \cdots, \psi_m\}$ such that $\psi_i \in N_G$ or (2) there is at least one simple literal ψ_i such that $\mathcal{C}\ddot{\psi}_i$ (or $\mathcal{C}_d\psi_i$) $\in N_G$.* □

The proofs of Propositions 6.3.2-6.3.4 are straight-forward from the construction of g-set.

6.4 Complexity and Scalability issues of the Constructive Semantics

The procedure involved in the constructive semantics is based on the repeated operation of the operator $\mathcal{D}$ and the operator R until a stable d-graph is reached. Let n be the total number of rules in a program. The maximum time needed for the

first operation $\mathcal{D}$ would be to the order of n. Now suppose the number of vertices in the d-graph after the completion of the first $\mathcal{D}$ operation is m. The operation $\mathcal{R}$ first involves the search of pairs of inconsistent complementary literals. Without applying any heuristics the time for the search would be to the order of m^2. We get this by the following way. For a simple literal in a node we search for the content of the rest of the nodes excluding the nodes for which we have already considered, for the complement of the simple literal. Thus the total number of comparisons is $(m-1)+(m-2)+(m-3)+\cdots+1$, which is approximately m^2. The repetitions of operation $\mathcal{D}$ and $\mathcal{R}$ will take significantly much less time compared to the first execution of the two operations unless there are many rules with contradictory or contradiction-affected literals in the body of the rule. This we do not forsee in a knowledge base. So we see that the procedure involved in the constructive semantics can be computed in polynomial time and thus tractable.

In [95], You and Ghosh have shown a constructive procedure for computing the intended model of a PS by program transformation. There in the worst case, a PS P' is obtained from a PS P by replacing all the occurrences of literals in P. The modification process is obviously tractable. Hence, the entire construction is tractable. We conjecture that the procedure by You and Ghosh and our constructive procedure are equivalent.

By our framework we try to model large knowledge bases. The problem of scalability is very important for large knowledge bases or combining knowledge bases. So it is important to explore the issues of scalability w.r.t. our approach. From the discussion of the complexity of our system we know that the computational process is tractable, however large the knowledge base is. Now if the knowledge base is increased with more rules and facts, the dynamics of the existing rules and facts in the knowledge base with the new ones does not allow the system to be cumulative, i.e. the existing g-set of the knowledge base may not be a subset of the new g-set of the knowledge base. So in that sense, the efficiency in scalability which comes with

cumulativity is not realizable by our system. But we can intuitively perceive that the framework can be made partially scalable. This we leave for future investigation.

6.5 Relating constructive and model-theoretic semantics of PS

In this section we explore the relationship between the constructive and the model-theoretic semantics of PS. As a result we get some interesting properties for the model-theoretic semantics of PS.

We present a theorem relating the two semantics of PS in the later part of the section. First we prove some lemmas which lead to the theorem.

Lemma 6.5.1 *Let P be a PS, W_{T_p} be the transformation operator and $\mathcal{D}$ be the deduction operator based on W_{T_p} as defined earlier. Let $G_\emptyset$ be an nd-graph. Let P_0 be the subset of P containing all the rules involved in the construction of $\mathcal{D}(G_\emptyset)$. Let M be any model of P_0. For any simple literal ϕ in $\mathcal{D}(G_\emptyset)$, ϕ, $C\ddot{\phi}$ or $C_d\phi$ belongs to M.* □

Proof: (By Induction)

Base Step:

For any element $\phi \in W_{T_p}(G_\emptyset)$, there is a rule $\phi \leftarrow$ belonging to P (by definition of W_{T_p}). Any model M_0 of the sub-PS P_0 of P containing only of rules of the form $\phi \leftarrow$, has to satisfy all such rules. To satisfy a rule $\phi \leftarrow$, ϕ or $C\ddot{\phi}$ or $C_d\phi$ belongs to M_0. Thus, for all $\phi \in M_0$, ϕ or $C\ddot{\phi}$ or $C_d\phi$ belongs to any model M_0 of the sub-PS P_0.

Induction Step:

Consider an operation by W_{T_p} on an intermediate d-graph $G_i = W_{T_p} \uparrow^i$, towards the construction of $\mathcal{D}(G_\emptyset)$. Let P_i represent the subset of rules of P used thus far in the construction of G_i. By inductive hypothesis for any $\phi \in N_{G_i}$ (where N_{G_i} is the set of all the literals in the nodes of G_i), ϕ, $C\ddot{\phi}$ or $C_d\varphi$ belongs to any model M_i of P_i.

Now, let P_{i+1} represent the set of rules in construction of $G_{i+1} = W_{T_p} \uparrow^{i+1} = W_{T_p}(G_i)$ (i.e. P_{i+1} includes the new rules used in construction of G_{i+1} from G_i). Any model M_{i+1} of P_{i+1} must satisfy every rule $\ldots \leftarrow c_1 \wedge \ldots \wedge c_m$, which is used in the construction of $W_{T_p} \uparrow^{i+1}$, such that $prem(R) \subseteq N_{G_i}$.

If $\phi \in N_{G_i}$, then already ϕ or $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi$ belongs to any model M_i of P_i and hence ϕ or $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi$ will also belong to any model M_{i+1} of P_{i+1}, since $P_{i+1} \supseteq P_i$.

If $\phi \notin N_{G_i}$, then for any model M_{i+1} in P_{i+1}, to satisfy rule R, ϕ or $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi$ belongs to M_{i+1} because $prem(R) \subseteq N_{G_i} \subseteq N_{G_{i+1}}$ (N_{G_i} the node set of G_{i+1}). Hence for any M_{i+1} of P_{i+1}, it is the case that for any $\phi \in N_{G_{i+1}}$, ϕ or $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi \in M_{i+1}$.

$\mathcal{D}(G_\emptyset)$ represents the d-graph based on which no further rules in P can be fired. So if P_0 represents the subset of rules fired thus far in construction of $\mathcal{D}(G_\emptyset)$, for any literal ϕ in $\mathcal{D}(G_\emptyset)$, ϕ, $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi$ belongs to any model of P_0.

Hence the lemma is proved. □

Example 6.5.1 *Let us consider the PS P of Example 6.2.4 and the corresponding Figure 6.4 for the construction of its g-set. The set of rules that are just involved in the construction of $\mathcal{D}(G_\emptyset)$ is $P_0 = \{a \leftarrow; \sim a \leftarrow; b \leftarrow a\}$. For each element ϕ (i.e. a simple literal) in the nodes of $\mathcal{D}(G_\emptyset)$ (which is given by the Figure 6.4(1)), there is a corresponding formula ϕ, $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi$ in any model of P_0.* □

Lemma 6.5.2 *Let P be a PS, $\mathcal{D}$ be the deduction operator and R be the deduction revision operator as defined earlier. Let $\mathcal{A}$ be the composite function as defined before, based on $\mathcal{D}$ and $\mathcal{R}$. Let $G_\emptyset$ be an nd-graph, P_0 be the subset of P containing all the rules involved in the construction of $\mathcal{D}(G_\emptyset)$ and M_0 be any model of P_0. Any literal ψ in $\mathcal{A}(G_\emptyset)$ also belongs to M_0.* □

Proof:

By Lemma 6.5.1, for every simple literal $\phi \in \mathcal{D}(G_\emptyset)$ w.r.t. P, ϕ, $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi$ belongs to any model M_0 of P_0.

$\mathcal{A}(G_\emptyset) = \mathcal{R} \circ \mathcal{D}(G_\emptyset)$. Hence we are concerned with the operation of $\mathcal{R}$ on the d-graph $G = \mathcal{D}(G_\emptyset)$.

Now for any pair of simple literals ϕ, $\tilde{\phi}$ in G we replace the nodes by $\mathcal{C}\ddot{\phi}$ by step (1) and (5) of operator $\mathcal{R}$ in construction of $\mathcal{R}(G)$. By Lemma 6.5.1, as ϕ and $\tilde{\phi} \in G$, ϕ, $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi \in M_0$ and $\tilde{\phi}$, $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\tilde{\phi} \in M_0$.

By the conditions of satisfiability of literals in a p-interpretation and hence in a model, the pairs $\{\phi, \tilde{\phi}\}$, $\{\phi, \mathcal{C}\ddot{\phi}\}$ and $\{\tilde{\phi}, \mathcal{C}\ddot{\phi}\}$ cannot belong to M_0. Moreover the literals ϕ and $\tilde{\phi}$ are on the head of rules with premises $\subseteq N_G$ (i.e. the set of all the literals in the nodes of G), where all the elements in N_G are simple literals not paraconsistent literals. Therefore we can neither have $\mathcal{C}_d\phi$ nor $\mathcal{C}_d\tilde{\phi}$ in M_0. Hence the only option left is that $\mathcal{C}\ddot{\phi} \in M_0$. Hence for any formula $\mathcal{C}\ddot{\phi}$ in $\mathcal{A}(G_\emptyset) = \mathcal{R}(G) = \mathcal{R} \circ \mathcal{D}(G_\emptyset)$ we will have $\mathcal{C}\ddot{\phi} \in M_0$.

Now, any node with a simple literal ϕ in G pointed by an arc coming from a node with $\mathcal{C}\ddot{\varphi}$ (φ is a simple literal), is replaced by the node $\mathcal{C}_d\phi$ by step (2b) of operator $\mathcal{R}$ in the construction of $\mathcal{R}(G)$. By Lemma 6.5.1 ϕ, $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi \in M_0$ as $\phi \in G$.

By the satisfiability relation of a formula by a model M_0 (i.e. a p-interpretation), if there is a rule R, such that $\mathcal{C}\ddot{\varphi} \in prem(R)$, $prem(R) \subseteq M_0$ and $\phi = concl(R)$, then $\mathcal{C}_d\phi \in M_0$. This exactly corresponds to the condition of $\mathcal{C}_d\phi$ to be in $\mathcal{R}(G)$. Hence for this case by Lemma 6.5.1 and the operation $\mathcal{R}$ on G, $\mathcal{C}_d\phi \in M_0$.

Similarly we can show that, the formulas like $\mathcal{C}_d\phi$ we obtain by step (3), (4) and (5) of the operator $\mathcal{R}$, in the construction of the d-graph $\mathcal{R}(G)$, are also in any model M_0 of P_0.

Thus any formula $\mathcal{C}_d\phi$ in $\mathcal{A}(G_\emptyset) = \mathcal{R}(G) = \mathcal{R} \circ \mathcal{D}(G_\emptyset)$ is also in M_0.

The remaining simple literals $\phi \in N_G$ for which by Lemma 6.5.1, ϕ or $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi \in M_0$. As none of the conditions for $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_d\phi$ to be in M_0 can be fulfilled. Hence $\phi \in M_0$.

Hence from all the cases we have enumerated above, we get that, every formula in $\mathcal{A}(G_\emptyset)$ is in any model M_0 of P_0. □

Example 6.5.2 *Let us consider the PS P of Example 6.2.4 and the corresponding Figure 6.4 for the construction of its g-set. The set of rules that are just involved in the construction of $\mathcal{D}(G_\emptyset)$ is $P_0 = \{a \leftarrow; \sim a \leftarrow; b \leftarrow a\}$. Each literal in the nodes of $\mathcal{A}(G_\emptyset)$ (which is given by the Figure 6.4(2)) is in any model of P_0.* □

Lemma 6.5.3 *Let the notations be as given in the lemmas before. Let P_f be the subset of a PS P containing all the rules involved in the construction of the $\mathcal{A}$-stable d-graph G_f, starting from an nd-graph applied upon by $\mathcal{A}$ w.r.t. P. Every literal $\psi \in G_f$ is in any model M_f of P_f.* □

Proof: (By Induction)

Base step:

Let P_0 be the subset of a PS P containing all the rules involved in the construction of $\mathcal{D}(G_\emptyset)$. By Lemma 6.5.2, any literal $\psi \in \mathcal{A}(G_\emptyset)$ w.r.t. P_0, is in any model M_0 of P_0.

Induction Step:

Consider any operation $\mathcal{A}$ on an intermediate d-graph G_i, towards the construction of G_f, where $\mathcal{A}(G_f) = G_f$. Let P_i represent the subset of rules used thus far in the construction of G_i. By inductive hypothesis for any literal $\psi \in N_{G_i}$ (where N_{G_i} is the set of all the literals in the nodes in G_i), ψ belongs to any model M_i of P_i.

Now, let P_{i+1} represent the set of rules in the construction of $G_{i+1} = \mathcal{A}(G_i)$ (i.e. P_{i+1} includes the new rules used in construction of G_{i+1} from G_i). Any model M_{i+1} of P_{i+1} must satisfy every rule R, which is used in the construction of $\mathcal{A}(G_i)$, such that each literal in $prem(R)$ or its paraconsistent forms $\in N_{G_i}$.

If a literal ψ $(= \phi, \mathcal{C}\ddot{\phi}$ *or* $\mathcal{C}_d\phi$, where $concl(R) = \phi) \in N_{G_i}$, then already ψ belongs to any model M_i of P_i and hence ψ will also belong to any model M_{i+1} of P_{i+1}, since $P_{i+1} \supseteq P_i$.

If $\psi \notin N_{G_i}$, then to satisfy any such rule R, ψ has to belong to each model M_{i+1} of P_{i+1}. Now since each literal in $prem(R)$ or its paraconsistent forms $\in N_{G_i}$, $\psi \in \mathcal{D}(G_i)$. Thus for any $\psi \in \mathcal{D}(G_i)$ ψ belongs to any arbitrary model M_{i+1} of P_{i+1}.

Now when we apply $\mathcal{R}$ to $\mathcal{D}(G_i)$, it is clear that any such rule $R : \phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$ which is involved in the construction of $\mathcal{A}(G_i)$, where $\phi, C\ddot{\phi}$ or $C_d\phi \notin N_{G_i}$, it is the case that $C_d\phi \in \mathcal{A}(G_i)$. This is because there is at least one simple literal $\psi_i \in prem(R)$, such that $C\ddot{\psi}_i$ or $C_d\psi_i \in N_{G_i}$ or there is a paraconsistent literal $\in prem(R)$. Hence only $C_d\phi$ will belong to any model M_{i+1} of P_{i+1}. Thus for every formula $\psi \in N_{G_{i+1}}$ ($N_{G_{i+1}}$ is the set of all the literals in the nodes in G_{i+1}) ψ is in any model M_{i+1} of P_{i+1}.

G_f represents the d-graph based on which no further rules in P can be fired. So if P_f represents the subset of rules fired thus far in the construction of G_f, for any literal ψ in G_f, ψ belongs to any model M_f of P_f.

Hence the lemma is proved. □

Example 6.5.3 *Let us consider the PS P of Example 6.2.4 and the corresponding Figure 6.4 for the construction of its g-set. The set of rules which are involved in the construction of G_f are all the rules in P. Each literal ψ in the nodes of G_f (which is given by the Figure 6.4(4)) is in any model of P.* □

Theorem 6.5.1 *The g-set of a PS P is a model of P.* □

Proof: Let the notations have their usual meanings as denoted before. Let P_f be the subset of a PS P containing all the rules involved in the construction of G_f, such that $G_1 = \mathcal{A}(G_\emptyset)$ and $G_f = \mathcal{A}(G_f)$. Every literal $\psi \in G_f$ is in any model M_f of P_f, by Lemma 6.5.3. P_f represents the subset of rules in P which are fired in the construction of G_f. No further rules in P can be fired in the construction of G_f. Hence for all the remaining rules R in P but $\notin P_f$, a literal in $prem(R)$ or its paraconsistent forms $\notin N_{G_f}$ (set of all the literals in the nodes in G_f). Hence literals in $prem(R)$ or its paraconsistent forms $\notin M_f$, if we assume that the model M_f has only the elements $\in N_{G_f}$, as those are all the elements it needs to have to satisfy all the rules in P_f. So, by the condition of satisfiability of a rule by a model (i.e. a p-interpretation), as

literals in $prem(R)$ or their paraconsistent forms $\notin M_f$, the rules R in P but $\notin P_f$ are all satisfied by the model M_f. Therefore M_f is a model of P.

Hence by Lemma 6.5.3:

$$\text{Every literal } \psi \in G_f \text{ is in any model } M \text{ of } P. \tag{6.2}$$

The g-set, which is the set of all the literals ψ in the nodes in G_f, also satisfies all he rules of a PS P. Hence the g-set is a model of P. Thus the theorem is proved. □

Corollary 6.5.1 *There always exists a model for any PS.* □

Proof: By the construction of a g-set, any PS P always has a g-set. A g-set is a model of P. Hence P has a model. Hence proved. □

Corollary 6.5.2 *The g-set of a PS P is a minimal model of P.* □

Proof: In the proof of Theorem 6.5.1 we have proved that: The g-set is a subset of any model M of a PS P (refer to statement 6.2). The g-set is also a model of P. Therefore it is a minimal model of P. Hence proved. □

Corollary 6.5.3 *There always exists a unique minimal model for a PS.* □

Proof: A PS P has a g-set. By Proposition 6.3.1 a PS has a unique g-set. Therefore P has a unique g-set, which is also its minimal model. Therefore P has a unique minimal model. Hence proved. □

Example 6.5.4 *Let us consider the PS P of Example 6.2.4 and the corresponding Figure 6.4 for the construction of its g-set. The g-set of P, i.e.* $\{Ca, C_d b, C_d c, C_d e, C_d \sim e\}$ *as given in Example 6.2.4 is the minimal model of P.* □

6.6 Relating to Positive Logic Programs

The purpose of our framework is to cope with inconsistency that arises when positive logic programs are extended to accommodate explicit negation. To handle inconsistency we introduce the connective $\mathcal{C}$. Furthermore, to capture the scenario of reasoning beyond paraconsistency we have introduced the connective $\mathcal{C}_d$. So, we would expect that, the semantics for paraconsistent specifications, will yield the same semantics for positive logic programs, as that of any existing semantics (i.e. least model semantics [20] reviewed in Chapter 3).

In this section we give a theorem that relates our semantics to least model semantics, for positive logic programs.

Theorem 6.6.1 *Let P be a paraconsistent specification consisting of clauses of the form*

$$\phi_0 \leftarrow \phi_1 \wedge \ldots \wedge \phi_m.$$

Then, if ϕ_is are atoms and $m > 0$, i.e. P is a positive logic program then:

The g-set N_G of P coincides with the **true** *set of the least fixpoint of the fixpoint operator Θ (given by Def 3.3.3 in Chapter 3) w.r.t. P.* □

Proof:

For a positive logic program P, the constructive definition for a g-set, i.e. the simple literals (they are only positive for this case) we get in the stable d-graph which we get when the operator $\mathcal{A} = \mathcal{R} \circ \mathcal{D}$ is applied on an nd-graph $G_\emptyset$ w.r.t P, reduces to that of the literals we get in $\mathcal{D}(G_\emptyset) = W_{T_p} \uparrow^w (G_\emptyset)$. This is because the operator R does not have any operational function here, as there does not arise any negative literal (as we are dealing with positive programs). Hence there won't arise any inconsistency, thus no explicit paraconsistency or reasoning beyond paraconsistency, which the $\mathcal{R}$ operator handles. W_{T_p} has a one to one correspondence with the operator T_p, which exactly coincides with the fixpoint operator Θ for the **true** atoms. Hence the g-set

N_G of P arrived as a fixpoint of the operator $\mathcal{A}$ coincides with the **true** set of the least fixpoint of the fixpoint operator Θ w.r.t. P. Hence the theorem is proved. □

Example 6.6.1 *If we consider Example 3.3.1 in Chapter 3, for program P, the g-set of P (which coincides with the minimal model of P) we get is*

$$\{ \textit{physicist, able-mathematician} \}.$$

This coincides with the **true** *set in the least model of P as given by Example 3.3.1 in Chapter 3.* □

6.7 Summary

In this chapter we have presented a constructive semantics for PS, investigated its properties, shown its correspondence to the model theoretic semantics of PS. We finally related the semantics of PS to the semantics of positive logic programs.

Chapter 7

Approach $\mathcal{C} - \mathcal{C}_d$ and Extended Logic Programs

In this chapter we expand the language of extended logic programs (which we denote as ELP) [30] to allow for 'reasoning in the presence of inconsistent information'. We apply the inconsistency handling strategy *Approach* $\mathcal{C}$ proposed in Chapter 4 to ELP, for this purpose. This extension increases the expressive power of the language of this nonmonotonic reasoning framework and allows one to reason explicitly in presence of contradictions without the trivialization (or explosion) approach of classical logic. The trivialization approach is certainly not adequate for the purpose of processing partially inconsistent information in a cognitively and computationally satisfactory way.

Furthermore, we apply the inconsistency handling strategy, *Approach* $\mathcal{C}_d$ (which we proposed in Chapter 4) to the language of ELP embedded with *Approach* $\mathcal{C}$, for *reasoning beyond paraconsistency*. It refers to the ability of reasoning from contradictory or contradiction-affected information, by the propagation of the fact that inference has been made based on contradictory or contradiction-affected information throughout the reasoning process. This extension also increases the expressive power of the language thus enabling the distinction between contradiction-affected

and indefeasible information (i.e. facts and information derived from facts).

The chapter is arranged in the following way. In the next section we present some background material regarding inconsistency and ELP. In Section 7.2 we present our inconsistency handling approaches, and their rationale in the context of ELP. In Section 7.2 we will introduce the extensions needed for the existing syntax of ELP to accommodate the inconsistency handling techniques and then we define the semantics. In Section 7.3 we explore some of properties of our formalism. In Section 7.4 we will explore the relationship of our extended answer set semantics to the existing answer set semantics of extended logic programs [30], to the stable model semantics of normal logic programs [34] and then to the model theoretic semantics of PS. We conclude with a summary of our contribution and a discussion of the scope for future work.

A preliminary version of the results presented in this chapter has appeared in [35].

7.1 Background

Extended logic programs [30] allow for negative conclusions in rules. So the question of dealing with contradictions in the database arises. The language of extended logic programs generates both *ontological* and *epistemic inconsistency*. (We have discussed them earlier in Chapter 1, but will revisit the concepts.) Ontological inconsistency owes its origin directly to 'explicit' negation. Epistemic inconsistency owes its origin directly to explicit negation and indirectly to negation by default, which is an added feature of extended logic programs.

Epistemic inconsistency manifests itself in the same way as ontological inconsistency. Both are expressed as an atom α and its explicit negation $\sim \alpha$. The difference is that, for ontological inconsistency the mutually contradicting elements owe their origin either (i) to their explicit presence in a knowledge base, or (ii) to derivation from facts. (By facts we mean indefeasible knowledge existing in the knowledge base.) For example in the language of ELP case (i) can be represented by the follow-

ing set of rules: $\{\sim a \leftarrow; a \quad\}$ comprising a knowledge base, and case (ii) as follows: $\{\sim a \leftarrow b; c \leftarrow; a \leftarrow c; b \leftarrow\}$. For epistemic inconsistency the mutually contradicting elements owes its origin to being derived (iii) partly or (iv) fully from assumptions, i.e. negation by default. In the language of ELP case (iii) can be represented by the following set of rules: $\{\sim a \leftarrow notb \wedge c; a \leftarrow d; d \leftarrow; c \leftarrow\}$ and case (iv) as follows: $\{\neg a \leftarrow notb; a \leftarrow notc\}$. But the difference in their origin does not express them differently in the theory. Hence, they are treated similarly.

We apply the inconsistency handling strategies to enhance the language of ELP to handle both ontological and epistemic inconsistency in a more satisfactory way, and empower it to reason from inconsistent information. We extend the answer set semantics [30] attached to extended logic programs (this we have reviewed in Chapter 3) to include our inconsistency handling strategies *Approach* $\mathcal{C}-\mathcal{C}_d$ and give it the ability to deal with contradictions in a more intuitive way.

7.2 *Approach* $\mathcal{C}-\mathcal{C}_d$ and its rationale in the context of ELP

Here, we apply *Approach* $\mathcal{C}-\mathcal{C}_d$ proposed in Chapter 4 to handle inconsistency. We introduce the new connectives $\mathcal{C}$ and $\mathcal{C}_d$ to the existing language of extended logic programs [30]. We allow formulas of the form $\mathcal{C}\alpha$ and $\mathcal{C}_d\alpha$ in the bodies of the rules of an ELP. We call this modified language of ELP, a *Paraconsistent Assumptive Specification* (which we denote by PAS). This is also an extension of the language of paraconsistent specification (PS), which we proposed in Chapter 4. Negation by default (*not*) is added to the language, giving the language the power of *closed world assumption* [73]. The terminology *assumptive* in PAS can be credited to its closed world assumptive capability.

We take the same approach as we have taken for definite (or positive) logic programs extended by explicit negation (refer to Chapter 4), but in the context of ex-

tended logic programs, and in the process reap similar benefits. Existing answer set semantics [30] for extended logic programs lacks the ability to handle inconsistency. According to it, an inconsistent answer set is trivialized to give the set of all literals of the language. But we would like our extended answer set semantics to be non-trivial. This is made clear in the following examples.

Example 7.2.1 *Consider an ELP as given below:*

$$\{p \leftarrow not\ q;\ \sim p \leftarrow not\ r;\ c \leftarrow a;\ a \leftarrow\}.$$

Answer set semantics of ELP [30] does not have an answer set for the above program. The contradictory information p and $\sim p$ sets off the explosive approach by which we get the set of all literals of the program. By applying **approach** C *to ELP, we will preserve the theory, and would expect to get as a conclusion the following set: $\{Cp, a, c\}$. This is more intuitive as we are able to conclude some information about the theory without trivializing.* □

Example 7.2.2 *We can imagine a scenario as given by the following database P:*

$$\{a \leftarrow not\ g;\ \sim a \leftarrow not\ f;\ e \leftarrow Ca;\ b \leftarrow a;\ c \leftarrow\sim a;\ \sim b \leftarrow c\}$$

arising in the real world, where the intention of the programmer is to have $\{c, Ca\}$ as the final set of conclusions from the database, as it is. And, if an f (resp. g) is added to the database, then the intended set of conclusions expected is $\{a, b\}$ (resp. $\{\sim a, c, \sim b\}$).

By the existing answer set semantics [30], we will get no answer set from P because a rule of the form $e \leftarrow Ca$ is not allowed in an ELP. If it were somehow allowed, we would still get an inconsistent set $\{a, \sim a, b, c, \sim b\}$, thus trivializing the theory. This wasn't the intention of the given database. It is unintuitive that just from a localized piece of inconsistent information the global theory will be contaminated and result in a collapse. By our answer set semantics for PAS embedded with Approach C *we would like to get the conclusion set which contains Ca.*

As we see from the above example, by capturing $a, \sim a$ by Ca we are blocking off the generation of unintuitive conclusions sometimes leading to an undesirable in-

consistency (if we allow $a, \sim a$ to be in our set of conclusions, it will facilitate the generation of $b, c, \sim b$) and thus more confusion.

It is reasonable to allow the derivation of c from the rule $c \leftarrow \mathcal{C}a$ if we get $\mathcal{C}a$ from the program after applying Approach $\mathcal{C}$ *to it. The problem that arises here is that if we infer $\sim c$ from some other part of the program, say by appending to P two more rules:*

$$\sim c \leftarrow r \text{ and } r \leftarrow$$

we get another inconsistency $\mathcal{C}c$. Getting this inconsistency is not intuitive as the inference of $\sim c$ is from a consistent piece of information r, whereas c is inferred from $\mathcal{C}a$, an inconsistent information. In our understanding it is rational to differentiate between conclusions derived from inconsistent information and consistent information. As in the inconsistent information there can be an intrinsic error. We consider the information derived from inconsistent information to have a lower epistemicity than information derived from a consistent information. So instead of arriving at the inconsistent information $\mathcal{C}e$, we would prefer the conclusion of $\sim e$ from the program P. This we will achieve by inferring $\mathcal{C}a$ and $\mathcal{C}_d e$ from the first set of rules. $\mathcal{C}_d e$ is concluded from the rule:

$$c \leftarrow \mathcal{C}a$$

based on the idea of Approach $\mathcal{C}_d$ *that if the premise of a rule is affected by a contradiction (here $\mathcal{C}a$), the head of the rule, which we conclude, is tagged with the information that it is inferred from a contradictory premise or a premise affected by contradictory information. Thus we are able to differentiate between conclusions inferred from consistent and inconsistent information. Now from all the above rules of the PAS P we get the conclusion set $\{\mathcal{C}a, r, \mathcal{C}_d e, \sim e, \mathcal{C}_d b, \mathcal{C}_d c, \mathcal{C}_d \sim b\}$. We do not derive the conclusion $\mathcal{C}e$ anymore.* □

Example 7.2.3 *Consider the PAS given below:*

$$\{q \leftarrow p;\ p;\ \sim p\}$$

This PAS has a unique answer set containing $\mathcal{C}p$ on applying Approach $\mathcal{C}$. *It is*

intuitively appropriate to block the inference of q and just derive the information that p (or $\sim p$) is inconsistent. But if we like our semantics to be more informative, it is rational to conclude something about q too. By applying Approach $\mathcal{C}_d$, *we are able to conclude* $\mathcal{C}_d q$. □

7.3 Formalizing Paraconsistent Assumptive Specifications (PAS)

In this section we will describe the language, define the semantics of the framework that we are proposing and shall investigate the properties of the formalism.

7.3.1 Syntax

Here we use the same notations we introduced in Chapter 5. We extend our language $\mathcal{L}$ with the connective *not* (negation by default). Formulas of $\mathcal{L}$ are defined similarly to what we defined for PS except that we introduce *default literals*, which are of the form $not\phi$, where $\phi \in \Phi$. We denote the set of all default literals by Φ^r. By *extended literals* we mean an element in the set $\mathcal{X} = \Phi \cup \Phi^r \cup \Psi$, the set of all simple literals, default literals and paraconsistent literals.

We will restrict ourself to a subset of the language of $\mathcal{L}$, which we call *Paraconsistent Assumptive Specifications* (PAS). The formal notion of such a specification is a syntactical extension of an *extended logic program* [30]. It is captured by the following definition:

Definition 7.3.1 (Paraconsistent Assumptive Specification (PAS))
A paraconsistent assumptive specification P is a collection of rules, where a rule R is of the form

$$\phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$$

where $\phi \in \Phi$, $\psi_i s$ *belong to* $\mathcal{X}$, $\psi_i, \tilde{\psi}_i\ (\in \Phi) \notin prem(R)$ *and* $m \geq 0$. □

We consider that any rule in a PAS is not self-inconsistent, for the same reasons as we considered for PS.

7.3.2 Semantics

The basics of the semantics that we are proposing here is the same as that of PS proposed in Chapter 4. Simple literals ϕ and $\sim\phi$ are not related to each other in the usual classical logic sense, i.e. if ϕ is **true** we should not conclude that $\sim\phi$ is **false** or vice-versa. The truth-theoretic valuation (truth or falsity) of ϕ and that of $\sim\phi$ in an interpretation are independent of one another. We also do not allow self-contradiction, i.e. a formula of the form $\phi \wedge \sim\phi$ does not have a truth-valuation **true** by our semantics.

We will give the truth-theoretic evaluation of sentences in our language in two steps. In the first step we will define the truth-valuation in an inconsistent context. That is when we allow both ϕ and $\sim\phi$ to be simultaneously **true**. Here we will initially just deal with simple literals. In the second step we will define the truth-valuation of extended literals and formulas formed from their conjunctions in a paraconsistent context in our language. Here the inconsistent literals are no longer **true**, but a new paraconsistent literal capturing the inconsistency is **true**.

We will now define the truth-valuation of literals in an inconsistent context. Let us consider a set W of literals. W represents the current (working) inconsistent set of beliefs of a reasoner. We will define the notion of truth in an inconsistent context of literals of $\mathcal{L}$ w.r.t. W.

Definition 7.3.2 (Truth-valuation in an inconsistent context)

A simple literal ϕ is **true** *in an inconsistent context iff $\phi \in W$.* □

We will now define truth-valuation of sentences in a paraconsistent context. Let us consider a set W^p of literals and paraconsistent literals and a set of literals W. W adheres to the truth-valuation in an inconsistent context. W^p represents the current

(working) paraconsistent set of beliefs, while W represents its current (working) inconsistent set of beliefs. We will inductively define the notion of truth ($\models_{para}$) in a paraconsistent context of formulae of $\mathcal{L}$ w.r.t. a pair $M^p =< W^p, W >$.

Definition 7.3.3 (Truth-valuation in a paraconsistent context)
Let $\phi \in \Phi$ and $\psi_i s \in \mathcal{X}$.

(1) $M^p \models_{para} \mathcal{C}\ddot{\phi}$ iff $\phi, \dot{\phi} \in W$ and $\mathcal{C}\ddot{\phi} \in W^p$.

(2) $M^p \models_{para} \phi$ iff $M^p \not\models_{para} \mathcal{C}\ddot{\phi}$, $\phi \in W$ and $\phi \in W^p$.

(3) $M^p \models_{para} not\phi$ iff $M^p \not\models_{para} \phi$ and $M^p \not\models_{para} \mathcal{C}\ddot{\phi}$.

(4) $M^p \models_{para} \mathcal{C}_d\phi$ iff $\mathcal{C}_d\phi \in W^p$.

(5) $M^p \models_{para} \psi_1 \wedge \ldots \wedge \psi_n$ iff $M^p \models_{para} \psi_i$ for $i = 1, \ldots, n$. □

Now we will define an *answer set* W^p, a set of extended literals satisfying a PAS P. The precise definition of this notion will be given in the following steps.

Definition 7.3.4 *Let P_0 be a paraconsistent assumptive specification consisting of rules of the form*

$$\phi \leftarrow .$$

Let $M^p =< W^p, W >$ be a pair, where W is a set of literals of P and W^p is a set of extended literals. W^p is called a belief set *of P_0 if it is a minimal set with the property that $M^p \models_{para} \phi$ (or $\mathcal{C}\ddot{\phi}$ or $\mathcal{C}_i\phi$) for every rule from P_0.* □

Example 7.3.1 *Let a PAS P_0 consists of rules:*

1. $q \leftarrow$
2. $\sim q \leftarrow$
3. $r \leftarrow$

Let $W = \{q, \sim q, r\}$. The corresponding W^p by Definition 7.3.3 and 7.3.4 is

$$\{\mathcal{C}q, r\}$$

So W^p is the belief set of P_0. □

Example 7.3.2 *Let PAS P consists of rules:*

1. $p \leftarrow$
2. $\sim q \leftarrow$
3. $q \leftarrow$

We can see that P has a belief set:

$$\{\mathcal{C}_d p, \mathcal{C} q\}.$$

□

Definition 7.3.5 *Let P be an arbitrary paraconsistent assumptive specification, W be a set of simple literals of P and W^p be a set of extended literals w.r.t. P. For every $M^p =< W^p, W >$, by P_{M^p} we will denote the paraconsistent assumptive specification obtained from P by:*

1. *Replacing all rules of the form:*

$$\varphi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m \quad ,$$

such that $\phi \in \Phi$, ψ_is are extended literals and $m \geq 0$,

by rules of the form:

$$\phi \leftarrow$$

iff

(a) (i) $M^p \models_{para} \mathcal{C}_d \phi$

(ii) $M^p \models_{para} \psi_1 \wedge \ldots \wedge \psi_m$ *and*

(iii) *there exists extended literals of the form $\mathcal{C}\ddot{\varphi}$ (resp. $\mathcal{C}_d \varphi$) $\in prem(R)$;*

or

(b) (i) $M^p \models_{para} \mathcal{C}_d \phi$

(ii) $M^p \models_{para} \mathcal{C}\ddot{\psi}_i$ *(resp. $\mathcal{C}_d \psi_i$) for some simple literal ψ_i and*

(iii) $M^p \models_{para} \psi_j$ *for all $\psi_j \neq \psi_i$.*

2. *Removing from the premises of the remaining rules of P all formulae F such that $M^p \models_{para} F$.*

3. Removing all remaining rules with non-empty premises.

Then W^P will be called an answer set of P iff W^P is a belief set *of P_{M^P}.* □

Example 7.3.3 *Let a PAS P consists of rules:*

1. $q \leftarrow not\, p$

2. $\sim q \leftarrow not \sim p$

3. $r \leftarrow$

Let $W = \{q, \sim q, r\}$ a set of simple literals of P and a corresponding $W^P = \{Cq, r\}$. By Definition 7.3.5 we obtain the PAS $P_{M^P} = P_0$, the PAS in Example 7.3.1. By Definition 7.3.4 W^P is the belief set of $P_{M^P} = P_0$. Hence W^P is the answer set of P.

□

Example 7.3.4 *Let a PAS P consists of rules:*

1. $p \leftarrow q$

2. $\sim q \leftarrow$

3. $q \leftarrow$

Let $W = \{q, \sim q\}$ or $\{q, \sim q, p\}$ and a corresponding $W^P = \{Cq, C_d p\}$. By Definition 7.3.5 we obtain the PAS $P_{M^P} = P_0$, the PAS in Example 7.3.2. By Definition 7.3.4 W^P is the belief set of $P_{M^P} = P_0$. Hence W^P is the answer set of P.

□

Example 7.3.5 *Consider the PAS given in Example 7.2.1. Let $W = \{p, \sim p, a, e\}$ and a corresponding $W^P = \{Cp, a, e\}$. By Definition 7.3.5 we obtain the PAS*

$$P_{M^P} = \{p \leftarrow; \sim p \leftarrow; e \leftarrow; a \leftarrow\}$$

By Definition 7.3.4 W^P is a belief set of P_{M^P}. Hence W^P is the answer set of P. □

Example 7.3.6 *Consider a PAS comprised of a collection of clauses:*

1. republican ←

2. quaker ←

3. pacifist ← quaker ∧ not ¬ pacifist

$$4.\ \sim pacifist \leftarrow republican \wedge not\, pacifist$$

We get two answer sets from the PAS:

$\{republican, \sim pacifist\}$ *and* $\{quaker, pacifist\}$. □

Example 7.3.7 *Let us consider the PAS P of Example 7.2.2:*

$$\{a \leftarrow not\, g;\ \sim a \leftarrow not\, f;\ c \leftarrow \mathcal{C}a;\ b \leftarrow a;\ c \leftarrow \sim a;\ \sim b \leftarrow c\}$$

appended by the rules:

$$\sim e \leftarrow r$$

$$r \leftarrow .$$

By the PAS transformation of Definition 7.3.5 we get the PAS P_{MP}*:*

$$\{a \leftarrow;\ \sim a \leftarrow;\ e \leftarrow;\ b \leftarrow;\ c \leftarrow;\ \sim b \leftarrow;\ \sim e \leftarrow;\ r \leftarrow\}$$

from P by considering W^p *to be the following set:*

$$\{\mathcal{C}a, \mathcal{C}_d e, \mathcal{C}_d b, \mathcal{C}_d c, \mathcal{C}_d \sim b, r, \sim e,\}.$$

Now, W^p *is the* belief set *(refer to Definition 7.3.4) of* P_{MP}*, hence it is the* answer set *of P.* □

7.4 Some Properties

Proposition 7.4.1 *Let* ϕ *be a simple literal.*

$$\phi \in W \text{ does not necessarily imply } M^p \models_{para} \phi.$$

□

Proof: Let $\phi \in \Phi$) and $\tilde{\phi}$ the complement of ϕ both be in W the current inconsistent set of beliefs of a reasoner and $\mathcal{C}\ddot{\phi} \in W^p$ the current paraconsistent set of beliefs of the reasoner.

Therefore by Definition 7.3.3(1) $\models_{para} \mathcal{C}\ddot{\phi}$.

Therefore by Definition 7.3.3(2) $\not\models_{para} \phi$ (or $\tilde{\phi}$).

Hence the proposition is proved. □

The above proposition is illustrated by the following example.

Example 7.4.1 *Let $P = \{p \leftarrow; \sim p \leftarrow\}$ be a PAS. If $W = \{p, \sim p\}$ be the inconsistent working set and $W^p = \{Cp\}$ is the corresponding paraconsistent working set, then we see $M^p \not\models_{para} p$ (or $\sim p$).* □.

Proposition 7.4.2 *A self-inconsistent formula (i.e. a formula of the form $\phi \wedge \dot{\phi}$, where $\phi \in \Phi$) is not true in a paraconsistent context.* □

Proof: Let us assume that $M^p (=< W^p, W >) \models_{para} \phi \wedge \dot{\phi}$.
Therefore $M^p \models_{para} \phi$ and $M^p \models_{para} \dot{\phi}$ by Definition 7.3.3(5).
Therefore (a) ϕ, $\dot{\phi} \in W$ and (b) $M^p \not\models_{para} C\ddot{\phi}$ by Definition 7.3.3(2).
Bby (a) and Definition 7.3.3(1) $\models_{para} C\ddot{\phi}$. This is in contrary to (b).
Therefore our assumption $\models_{para} \phi \wedge \dot{\phi}$ proved to be false. Hence the proposition is proved. □

The above proposition is illustrated by the following example.

Example 7.4.2 *Consider the PAS given below:*

$$\{q \leftarrow p \wedge \sim p;\ p;\ \sim p\}$$

This PAS has an unique answer set: $\{Cp\}$.
This is intuitively appropriate as we do not allow self-inconsistency, i.e. do not allow formulas like $p \wedge \sim p$. thus the rule $q \leftarrow p \wedge \sim p$ is removed as the premise of the rule does not truth-valuate to true in a paraconsistent context, i.e. $M^p \not\models_{para} p \wedge \sim p$. □

Proposition 7.4.3 *Let R be a rule in a PAS P and $M^p \models_{para} C\ddot{\phi}$. If ϕ, $\dot{\phi}$ or $C\ddot{\phi}$ belongs to the premise of R, conclusion of R will not belong to the answer set of P.*
□

Proof: Let us assume that R be a rule in a PAS P, $M^p \models_{para} C\ddot{\phi}$ and ϕ, $\dot{\phi}$ or $C\ddot{\phi}$ belongs to the premise of R. Let us also assume in contrary to the proposition that conclusion of R belongs to the answer set of P.

As conclusion of R belongs to W^p an answer set of P, it belongs to the belief set of the PAS P_{M^p} got by the PAS transformation by Definition 7.3.5.

Therefore by Definition 7.3.4 we can assert that:
(A): there is a body less rule R' in P_{M^P} such that the conclusion of R is equal to the conclusion of R'.

All the rules in P_{M^P} are arrived from P following the PAS transformation by Definition 7.3.5. But as $M^P \models_{para} C\ddot{\phi}$ and ϕ, $\tilde{\phi}$ or $C\ddot{\phi}$ belongs to the premise of R, by Definition 7.3.5 R should be removed from P in the process of transformation to P_{M^P}. This will not result with the rule R' in P_{M^P}. This is in contrary to the assertion (A). Hence we disproved our initial assumptions. Hence the proposition is proved. □

Proposition 7.4.4 *Let $R : \phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$ be a rule in a PAS P. If*

(a) (i) $M^P \models_{para} \psi_1 \wedge \ldots \wedge \psi_m$ and

(ii) there exists an extended literal of the form $C\ddot{\varphi}$ (resp. $C_d\varphi$) in the premise of R;

or

(b) (i) $M^P \models_{para} C\ddot{\psi}_i$ (resp. $C_d\psi_i$) for some simple literal $\psi_i \in prem(R)$, such that $\tilde{\psi}_i \notin prem(R)$ and

(ii) $M^P \models_{para} \psi_j$ for $\psi_j \neq \psi_i$;

then a contradiction-affected literal on conclusion of R (i.e. $C_d\phi$) is in an answer set of P. □

Proof: The proof is straight-forward from the semantical definitions of an answer set of a PAS. □

The following examples illustrate the above proposition.

Example 7.4.3 *Consider the PAS of Example 7.2.3 given below:*

$$\{q \leftarrow p;\ p;\ \sim p\}.$$

The answer set for PAS is: $\{Cp, C_dq\}$. □

Example 7.4.4 *Consider the PAS P below:*

$$\{p \leftarrow not \sim r;\ q \leftarrow not \sim p;\ r \leftarrow q;\ \sim q \leftarrow r\}$$

By the answer set semantics of PAS as given above, we will have the following answer set $\{p, \mathcal{C}q, \mathcal{C}_d r, \mathcal{C}_d \sim q\}$ for P. □

Example 7.4.5 *Let PAS P consist of the rules*

1. $p \leftarrow \mathcal{C}_d r \ \wedge \ not\ t$
2. $r \leftarrow q$
3. $q \leftarrow not\ s$
4. $\sim q \leftarrow not\ s$

We can see that P has a unique answer set: $\{\mathcal{C}q, \mathcal{C}_d r, \mathcal{C}_d p\}$. □

Example 7.4.6 *For the PAS $\{a \leftarrow not\ a\}$ we do not get an answer set.* □

7.5 Relating PAS to ELP and PS

In this section we show how the answer set semantics of PAS encapsulates the answer set semantics of ELP. We also relate the answer set semantics of PAS to the model theoretic semantics of PS.

The databases in the Example 3.4.1 and Example 3.4.2 can be considered as PAS. The answer sets given by the two PAS in the above mentioned examples by the answer set semantics of PAS are the two answer sets by the answer set semantics of ELP as given by the respective examples. The modified answer set semantics for PAS gives the same intended semantics as given by the answer set semantics for ELP if the ELP is non-contradictory.

The following proposition establishes the relationship of extended logic programs with paraconsistent assumptive specifications.

Proposition 7.5.1 *W^p is an answer set of a non-contradictory ELP P by the answer set semantics of ELP (as given in Chapter 3) iff it is an answer set of P by the answer set semantics of PAS.* □

Proof: A non-contradictory ELP P is contradiction free. Therefore the logical mechanisms to handle inconsistency embedded in the semantical definitions for a world view of an ELP (given in Chapter 3 Section 3.4) are redundant. Thus ELP P is also a PAS. Therefore the logical mechanisms to handle inconsistency embedded in the semantical definitions for an answer set of PAS are also redundant in computing an answer set for the non-contradictory ELP P. Thus effectively the semantical definitions for an answer set of ELP and the semantical definitions for an answer set of PAS has a one to one correspondence in computing an answer set of P. Therefore an answer set of P by the answer set semantics of ELP coincides with an answer set of P by the answer set semantics of PAS. Hence our proposition is proved. □

Example 7.5.1 *Consider the PAS P given in Example 3.4.1:*

$$\{\sim q \leftarrow;\ p \leftarrow \sim q\}$$

The answer set we get by the answer set semantics of PAS is : $\{\sim q, p\}$*, same as we get by the answer set semantics of ELP [30].* □

Example 7.5.2 *Consider the PAS P given in Example 3.4.2:*

$$\{\sim p \leftarrow not\ q\}$$

The answer set we get by the answer set semantics of PAS is : $\{\sim p\}$*, same as we get by the answer set semantics of ELP [30].* □

The following proposition establishes the relationship of normal logic programs with paraconsistent assumptive specifications.

Proposition 7.5.2 *Let P be a PAS consisting of rules of the form:*

$$\phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m.$$

such that, ϕ *is an atom,* ψ_i*s are atoms* α *or formulas of the form not* α *(i.e. P is a normal logic program) then* W^P *is an answer set of P iff* W^P *is a* stable model *(as defined in [34]) of P.* □

Proof: This proof is on similar lines to the proof of Proposition 7.5.1. □

Now we relate the answer set semantics of PAS to the model theoretic semantics of PS.

The following proposition relates the two semantics.

Proposition 7.5.3 *M is a minimal model of a PS P by the model theoretic semantics of PS (as given in Chapter 5) iff it is an answer set of P by the answer set semantics of PAS.* □

Proof: Let M be a minimal model of a PS P and I be the corresponding interpretation of M. Therefore M satisfies every rule $R : \phi \leftarrow \psi_1 \wedge \ldots \wedge \psi_m$ (such that $\phi \in \Phi$ and ψ_is $\in \mathcal{X}$) in P. Therefore by Definition 5.2.5 in Chapter 5 of satisfaction by a model (i.e. a p-interpretation) of a rule in a PS, one or more of the following conditions hold:

(a) $\psi_i \notin M$, $\mathcal{C}\ddot{\psi}_i \notin M$ and $\mathcal{C}_d\psi_i \notin M$ for some $1 \leq i \leq m$;

(b) $\phi \in M$;

(c) R is a plain rule, $\mathcal{C}\ddot{\phi} \in M$, ψ_i or $\mathcal{C}_d\psi_i \in M$ for all i and there is a plain rule $R' : \ddot{\phi} \leftarrow \psi'_1 \wedge \ldots \wedge \psi'_m$ in P, such that ψ'_i or $\mathcal{C}_d\psi'_i \in M$ for all i;

(d) $\mathcal{C}_d\phi \in M$, there exists a $\psi_i \in \Phi$ such that $\mathcal{C}\ddot{\psi}_i$ or $\mathcal{C}_d\psi_i \in M$, and $\psi_j \in M$ such that $\psi_j \neq \psi_i$;

(e) $\mathcal{C}_d\phi \in M$, $\psi_i \in M$ for all i and there exists a ψ_i that is a paraconsistent literal.

The language of PS is a subset of the language of PAS. PS does not have default literals in the body of its rules. So P is an PAS. Moreover for a PS, $\models_{para}$ (i.e. truth in paraconsistent context) guided by Proposition 7.4.4 (which states under what conditions a literal (i.e. a simple literal or a paraconsistent literal) is **true** in a paraconsistent context w.r.t. an EPAS) has a one to one correspondence with $\models_{\mathcal{C}\text{-}\mathcal{C}_d}$ (i.e. satisfaction by an interpretation w.r.t. a PS)

By Definition 7.3.5 every rule $R \in P$ is transformed by the conditions in it to get P_{M^p}. Now for any rule $R \in P$, (a), (b), (c), (d) or (e) holds. If (a) holds then strategy (3) in the PAS transformation definition is satisfied. If (b) (resp. (c), (d) or (e)) holds then strategy (2) (resp. (1)) in Definition 7.3.5 (i.e. the PAS transformation definition) is satisfied and thus we get the P_{M^p} with rules like $\phi \leftarrow$, such that $< M, I > \models_{para} \phi$ (*or* $C\ddot{\phi}$ *or* $C_d\phi$) (as $\models_{para}$ has a one to one correspondence with $\models_{C-C_d}$ for PS). Therefore M is an answer set of P by the answet set semantics of PAS.

Similarly we can prove that if M is an answer set of PS P by the answer set semantics of PAS, then it is a minimal model of P by the model theoretic semantics of PS.

Hence the proposition is proved. □

7.6 Summary

We have extended the language of extended logic programs to handle ontological and epistemic inconsistency. Instead of discarding the inconsistent answer sets by trivializing them we make them viable by capturing the inconsistent information in them explicitly. We have also shown the usefulness of capturing inconsistency, as in certain situations we may still use the inconsistent information for further reasoning.

Our semantics collapses to that of stable model semantics for normal logic programs. Stable model semantics of normal logic programs [34] and extensional semantics of default logic [74] are equivalent. Thus our semantics is related to default logic. So, we have a reason to believe that default logic can also benefit from our inconsistency handling approach. We should be able to extend default logic by our inconsistency handling approach $C - C_d$.

As future work we would like to extend the framework of PAS to introduce the notion of defeasible information inferred from assumptions (i.e. negation by default)

by the introduction of an *inferred from defeasible belief operator* $\mathcal{B}$. By this we can deal with epistemic inconsistency differently from ontological inconsistency. A similar approach has been taken by Pequeno and Buchsbaum [58] in the context of defeasible default logic. We find this approach a more natural way of resolving epistemic inconsistencies where one of the pair of the responsible inconsistent information stems from defeasible information and the other from indefeasible information, e.g. $\{a \leftarrow;\ \sim a \leftarrow not\ b\}$. By this approach we would not get any epistemic inconsistency as envisioned and resolved by Pereira etal [61] and Pimentel and Rodi [62].

We can think of an alternate way of resolving this. If our approach is embedded into the *contradiction removal semantics* [61], we can expect to have a but not $\sim a$ nor $\mathcal{C}a$. In other words, they become *unknown*. In this way we can complement and greatly enhance the contradiction handling capability of their system.

Chapter 8

Approach $\mathcal{C} - \mathcal{C}_d$ and Objective Epistemic Specifications

Epistemic Specifications (which we denote as ES) [29, 33] are a more general form of extended logic programs (ELP) [31] with the notions of *knowing* and *believing* added to it. *Objective Epistemic Specifications* (which we denote as OES), form a subset of the language of ES, i.e. ES without the notions of *knowing* and *believing*. In this chapter we expand the language of OES to allow for *reasoning in the presence of inconsistent information* (by *explicit paraconsistency*) and *reasoning beyond paraconsistent information*.

The language of OES is expressive than that of ELP. But the expressive power is hindered because of its inability to handle inconsistency. We apply the inconsistency handling strategies **Approach** $\mathcal{C}$ and **Approach** $\mathcal{C}_d$ (proposed in Chapter 4) to OES to handle inconsistency effectively. We call this extended language of OES *Paraconsistent Objective Epistemic Specifications* (POES). This extension allows one to reason explicitly in presence of contradictions without the trivialization (or explosion) approach of classical logic, which is certainly not adequate for processing partially inconsistent information in an intuitive way. The expansion also gives more expressive power to the language of OES.

The chapter is organized as follows: In Section 8.1 we present our inconsistency handling approach, and the rationale behind the new approach in the context of OES. (We have presented OES in Chapter 3.) In Section 8.2 we introduce the extensions needed for the existing syntax of OES to accommodate the inconsistency handling techniques. Then we define its semantics and investigate its properties. In Section 8.3 we explore the relationship of the semantics of POES to the semantics of OES [31, 29] and then to the semantics of PAS (as given in Chapter 7). In the last section we conclude with a summary of our contribution and directions to future work.

8.1 Background

OES allows for negative sentences in the head of a rule. This leads to contradiction. So, the question of dealing with contradictions in the OES databases arises. The language of OES generates both ontological and epistemic inconsistencies. We apply the inconsistency handling strategies **Approach** $\mathcal{C}$ and **Approach** $\mathcal{C}_d$ (i.e. the combined **Approach** $\mathcal{C}-\mathcal{C}_d$) to the language of OES to handle both *ontological and epistemic inconsistency*[1] in a satisfactory way. POES embedded with the inconsistency handling strategies is able to deal with contradictions in a more intuitive way than OES.

8.1.1 *Approach* $\mathcal{C}-\mathcal{C}_d$ and its Rationale in the context of OES

Here, we apply *Approach* $\mathcal{C}-\mathcal{C}_d$ (proposed in Chapter 4) to OES to handle inconsistency. We introduce the new connectives $\mathcal{C}$ and $\mathcal{C}_d$ to the existing language of OES [29]. Let $\ddot{F}$ denote the positive part of a simple formula F, e.g. if $F = \sim (p \wedge q)$ or $F = p \wedge q$, then $\ddot{F} = p \wedge q$. Now, by $\mathcal{C}\ddot{F}$ we mean *F is inconsistent (or F is in contradiction)*, i.e. there is contradictory information available about F in the theory. By our semantics if F and $\sim F$ are in the same theory, we replace them by $\mathcal{C}\ddot{F}$. $\mathcal{C}_d F$

[1] refer to Chapter 4

means *F is inferred from a premise (of a rule in a theory) affected by an inconsistent information.* We allow formulas of the form $C\ddot{F}$ and C_dF in the body of a rule of an OES. We call this modified OES, a *Paraconsistent Objective Epistemic Specification.* This we denote by POES.

By applying *Approach* $C - C_d$ to OES, we have the same advantages as when we apply *Approach* $C - C_d$ to ELP (refer to Chapter 7). The following examples illustrates the point clearly.

Example 8.1.1 *We take one of the same motivational examples we took in Chapter 7 when we applied* Approach C *to ELP, because, the motivations to apply the inconsistency handling approach is the same as it was for ELP. Let us consider the ELP in Example 7.2.1 (which is also an OES):*

$$\{p \leftarrow not\ q;\ \sim p \leftarrow not\ r;\ e \leftarrow a;\ a \leftarrow\}$$

This is a significant example where the semantics for objective epistemic specifications (which allows formulas in the body and the head of a rule) (refer to Chapter 3), a sublanguage of epistemic specifications [29] fails to give an intuitive semantics.
By the semantics of OES we get a world view, i.e. a set of all simple literals, by the trivialization approach in the face of inconsistency. But intuitively we can expect to get at least a and e as a consequence of the above set of clauses as it is contradiction free. By the semantics that we propose, we get the following set of conclusions: $\{a, e, Cp\}$ *which follows the intuitive meaning as given by the specification. We not only get the information a and e, but also the information that we have contradictory information about p, which we represent by* Cp. □

Example 8.1.2 *Let us consider Example 3.5.7 consisting of the rules*

1. $\sim p \leftarrow$
2. $p \leftarrow$
3. $q \leftarrow \sim (p \wedge \sim p)$

This specification does not have a world view by the answer set semantics of OES. Because of the inconsistency owing to the the first and second rules, the theory is

trivialized.

In the language of POES, we would like to get the following intuitive set of conclusions: $\{\mathcal{C}p, \mathcal{C}_d q\}$. *By* Approach $\mathcal{C}$ *we capture the inconsistency implicitly by* explicit paraconsistency *and by* Approach $\mathcal{C}_d$ *we get* $\mathcal{C}_d q$ *by tagging the information* q *derived from rule (3) whose premise is affected by the paraconsistent information* $\mathcal{C}(p \wedge \sim p)$ *which is implied by the theory. In this example we see how complex sentences play an interesting role in the language of POES.* □

The overall motivation of extending the language of OES is to make it *explicitly paraconsistent* so that it can handle inconsistency present in the form of both simple literals (e.g. p and $\sim p$) and simple formulas (e.g. $p \wedge q$ and $\sim (p \wedge q)$). It is also to give OES the ability to *reason beyond paraconsistency*, such that the system is able to infer both contradiction-affected literals (e.g. $\mathcal{C}_d p$) and contradiction-affected simple formulas (e.g. $\mathcal{C}_d(p \wedge q)$).

8.2 Formalizing Paraconsistent Objective Epistemic Specifications (POES)

In this section we describe the language, define the semantics of the framework of POES and shall investigate the properties of the formalism.

8.2.1 Syntax

Here we use the same notations we introduced in Chapter 5 and Chapter 7. We extend our language $\mathcal{L}$ with the connective *or* (epistemic disjunction) and parentheses (,). Formulas of $\mathcal{L}$ are defined similarly to what we defined for PAS except that we introduce epistemic disjunction. By *extended literals* we mean an element in the set $\mathcal{X} = \Phi \cup \Phi^x \cup \Psi$, where Φ is the set of all simple literals, Φ^x is the set of all default literals and Ψ is the set of all paraconsistent literals.

Simple formulas in our language are formulas formed from simple literals ($\in \Phi$) connected by the logical connectives $\wedge$, *or*, $\sim$, (and) in the usual way. We denote the set of all simple formulas in our language by $\mathcal{F}$. By $\tilde{F}$ ($F \in \mathcal{F}$) we denote $\sim F'$, if $F = F'$; if $F = \sim F'$, $\tilde{F}$ denotes F'. By $\ddot{F}$ ($F \in \mathcal{F}$) we denote the positive part of F or $\tilde{F}$. More complex formulas in our language are formed by formulas like F, *not* F, $\mathcal{C}\ddot{F}$ and $\mathcal{C}_d F$ connected by logical connectives $\wedge$, *or*, $\sim$, (and) where $F \in \mathcal{F}$. We denote the set of all such complex formulae by $\mathcal{G}$.

We restrict ourself to a subset of the language of $\mathcal{L}$, i.e. *Paraconsistent Objective Epistemic Specifications* (POES). We allow contradictions to prevail in a theory of a POES by cautiously capturing it by $\mathcal{C}$ and inferences from contradiction affected premises by $\mathcal{C}_d$. The formal notion of such a specification which is also an extension of an *extended disjunctive database* [31] is captured by the following definition:

Definition 8.2.1 (Paraconsistent Objective Epistemic Specification (POES)) *A paraconsistent objective epistemic specification P is a collection of rules of the form*

$$F \leftarrow G_1 \wedge \ldots \wedge G_m$$

where, $F \in \mathcal{F}$, G_is $\in \mathcal{G}$ and $m \geq 0$. □

We assume that the premise of a rule in a POES does not have a self-inconsistent formula. The reason for this consideration has been discussed for PS and it is the same here.

So the subset of our language $\mathcal{L}$ we are considering here is that of paraconsistent objective epistemic specifications (POES), syntactically and semantically an extension of OES.

Remark 8.2.1 *'$\leftarrow$' and 'or' have the same meaning here as they have in OES (refer to Chapter 3 Section 3.5).*

8.2.2 Semantics

The basics of the semantics that we are proposing here is the same as the semantics of PAS, as proposed in Chapter 7. A simple formula F and its negation $\sim F$ are not related to each other in the usual classical logic sense, i.e. if F is *true* we should not conclude that $\sim F$ is *false* or vice-versa. By our semantics the truth-theoretic valuation (truth or falsity) of F and that of $\sim F$ are independent of each other.

We give the truth-theoretic evaluation of sentences in our language in two steps adhering to the notions we developed in Chapter 4, Section 4.4. First we define the truth-valuation in an inconsistent context, where we allow both simple formulas F and $\sim F$ to be simultaneously **true**. In this step we only deal with simple formulas. In the second step we define the truth-valuation of sentences in our language in a paraconsistent context when the inconsistent sentences are no longer **true** but a new formula $\mathcal{C}\ddot{F}$ capturing the inconsistency is **true**. In this second step of truth-valuation of sentences in a paraconsistent context we deal with both simple formulas ($\in \mathcal{F}$) and complex formulas ($\in \mathcal{G}$).

We now define the truth (or falsity) of simple formulas in an inconsistent context. Let us consider a set W of simple literals ($\in \Phi$). W represents the current (working) possibly inconsistent set of beliefs of a reasoner. We define the notion of truth ($\models_{in}$) and falsity ($=|_{in}$) in an inconsistent context of simple formulas of $\mathcal{L}$ w.r.t. W.

Definition 8.2.2 (Truth-valuation in an inconsistent context)
Let $\phi, \tilde{\phi} \in \Phi$ and $F, F_i s \in \mathcal{F}$.

(1) $W \models_{in} \phi$ *iff* $\phi \in W$

(2) $W \models_{in} F_1 \wedge F_2$ *iff* $W \models_{in} F_1$ *and* $W \models_{in} F_2$

(3) $W \models_{in} F_1$ *or* F_2 *iff* $W \models_{in} F_1$ *or* $W \models_{in} F_2$

(4) $W \models_{in} \sim F$ *iff* $W =|_{in} F$

(5) $W =|_{in} \phi$ *iff* $\tilde{\phi} \in W$

(6) $W =|_{in} F_1 \wedge F_2$ *iff* $W =|_{in} F_1$ *or* $W =|_{in} F_2$

(7) $W =|_{in} F_1$ *or* F_2 *iff* $W =|_{in} F_1$ *and* $W =|_{in} F_2$

(8) $W \mid\sim_{in} F$ *iff* $W \models_{in} F$

□

We now define the truth-valuation of sentences in a paraconsistent context. Let us consider a set W^p of simple literals and paraconsistent literals. Let us also consider a set W of simple literals. W adheres to the truth-valuation $\models_{in}$. W^p can be thought of as a possible paraconsistent belief set of a reasoner. We define the notion of truth ($\models_{para}$) and falsity ($=|_{para}$) in a paraconsistent context of formulae of $\mathcal{L}$ w.r.t. a pair $M^p =< W^p, W >$.

Definition 8.2.3 (Truth-valuation in a paraconsistent context)
Let $\phi, \tilde{\phi} \in \Phi$, F, $\tilde{F}$, F_is $\in \mathcal{F}$ *and* G, G_is $\in \mathcal{G}$.

(1) $M^p \models_{para} C\ddot{\phi}$ *iff* $M \models_{in} \phi$, $M \models_{in} \tilde{\phi}$ *and* $C\ddot{\phi} \in W^p$.

(2) $M^p \models_{para} \phi$ *iff* $M^p \not\models_{para} C\ddot{\phi}$, $M \models_{in} \phi$ *and* $\phi \in W^p$.

(3) $M^p \models_{para} C_d\phi$ *iff* $C_d\phi \in W^p$.

(4) $M^p \models_{para} G_1 \wedge \ldots \wedge G_n$ *iff* $M^p \models_{para} G_i$ *for all* $i = 1, \ldots, n$.

(5) $M^p \models_{para} G_1$ *or* $\ldots$ *or* G_n *iff* $M^p \models_{para} G_i$ *for some* $i = 1, \ldots, n$.

(6) $M^p \models_{para} C\ddot{F}$ *iff* $M \models_{in} F$, $M \models_{in} \tilde{F}$.

(7) $M^p \models_{para} C_d F$ *iff* $M^p \models_{para} C_d F_i$ *for all* $i = 1, \ldots, n$, *such that* $F = F_1 \wedge \ldots \wedge F_n$.

(8) $M^p \models_{para} C_d F$ *iff* $M^p \models_{para} C_d F_i$ *for some* $i = 1, \ldots, n$, *such that* $F = F_1$ *or* $\ldots$ *or* F_n.

(9) $M^p \models_{para} \sim G$ *iff* $M^p =|_{para} G$.

(10) $M^p \models_{para} notF$ *iff* $M^p \not\models_{para} F$ *(resp.* $C\ddot{F}$*)*.

(11) $M^p =|_{para} \phi$ *iff* $M^p \not\models_{para} C\ddot{\phi}$, $M \models_{in} \tilde{\phi}$ *and* $\tilde{\phi} \in W^p$.

(12) $M^p =|_{para} G_1 \wedge \ldots \wedge G_n$ *iff* $M^p =|_{para} G_i$ *for some* $i = 1, \ldots, n$.

(13) $M^p =|_{para} G_1$ *or* $\ldots$ *or* G_n *iff* $M^p =|_{para} G_i$ *for all* $i = 1, \ldots, n$.

(14) $M^p =|_{para} C\ddot{F}$ *iff* $M^p \models_{para} F$ *or* $M^p \models_{para} \hat{F}$.

(15) $M^p =|_{para} \sim G$ *iff* $M^p \models_{para} G$.

(16) $M^p =|_{para} notF$ *iff* $M^p \models_{para} F$ *(resp.* $C\ddot{F}$*).*

□

Remark 8.2.2 *The sentences formed by the connective or in our language can also be semantically defined as follows: Let* $G_1, G_2 \in \mathcal{G}$,

$$(G_1 \; or \; G_2) \; iff \; \sim(\sim G_1 \wedge \sim G_2) \tag{8.1}$$

Equation 8.1 is easily provable from Definition 8.2.3.

Now we define a set W^p of simple literals and paraconsistent literals satisfying a POES P. We call such a set an *answer set* of P. The precise definition of these notions are given in the following steps.

First we define the notion of a consistent set.

Definition 8.2.4 (Consistent set) *A set* W^p *which does not have paraconsistent literals is called a* consistent *set.* □

Definition 8.2.5 *Let* P_0 *be a paraconsistent objective epistemic specification consisting of rules of the form*

$$F \leftarrow$$

(a) If there exists a minimal consistent set ($\subseteq \Phi$) *with the property* $M^p \models_{para} F$ *for every rule from* P_0 *it is called a* belief set *of* P_0.

(b) If there does not exist a belief set by (a), a minimal set ($\subseteq \Phi \cup \Psi$) *with the property* $M^p \models_{para} F$ *(or* $C\ddot{F}$ *or* $C_d F$*) for every rule from* P_0 *is called a* belief set *of* P_0. □

Example 8.2.1 Let P consists of rules:

1. $p \; or \; q \leftarrow$
2. $\sim q \leftarrow$

3. $r \; or \; s \leftarrow$

We can see that P has two belief sets by condition (a) of Definition 8.2.5:

$$\{p, \sim q, r\} \; \{p, \sim q, s\}.$$

□

Example 8.2.2 Let P consists of clauses:

1. $p \; or \; \sim p \leftarrow$
2. $\sim p \leftarrow$
3. $p \leftarrow$

Let the set of possible inconsistent sets be $W = \{p, \sim p\}$ and the set of possible paraconsistent sets be $W^p = \{\mathcal{C}p\}$. Now by Definitions 8.2.2 and 8.2.3 we can get the following:

$p, \sim p \in W \Rightarrow M =|_{in} p$ and $M =|_{in} \sim p \Rightarrow M =|_{in} p \; or \; \sim p \Rightarrow M \models_{in} \sim (p \; or \; \sim p)$ (a).

$p, \sim p \in W \Rightarrow M \models_{in} p$ or $M \models_{in} \sim p \Rightarrow M \models_{in} p \; or \; \sim p$ — (b).

(a) and (b) $\Rightarrow M^p \models_{para} \mathcal{C}(p \; or \; \sim p)$.

We can see that P has the belief set $W^p = \{\mathcal{C}p\}$ by condition (b) of Definition 8.2.5 as $M^p \models_{para} \mathcal{C}(p \; or \; \sim p)$ for rule (1) and $M^p \models_{para} \mathcal{C}p$ for rules (2) and (3). □

Here we first try to get belief sets that are consistent and minimal following condition (a) of the definition. If we fail to get any such set, we try to get belief sets which are just minimal (but not consistent) by condition (b) of the definition.

In Example 8.2.1 we already get two consistent minimal belief sets (as given in the example) by condition (a) of the definition. Therefore we do not consider the sets $\{\mathcal{C}q, r\}$ *or* $\{\mathcal{C}q, s\}$ which we get by condition (b) of the definition, though they are (set theoretically) smaller then the belief sets we already got by condition (a). The reasoning behind this is that inconsistency rising from information coming from an epistemic disjunctive information is epistemically weaker than information coming from an indefeasible information. In the example $\sim q$ has a higher epistemic status than q in $p \; or \; q$.

In Example 8.2.2 as we do not get any minimal consistent set by condition (a) of the definition we accept the minimal sets (as given in the example) by condition (b) of the definition.

Example 8.2.3 Let the POES P be

$$\{p \text{ or } q \leftarrow;\ \sim q \leftarrow;\ p \leftarrow\}$$

P has the belief set $\{\sim q, p\}$ by condition (a) of Definition 8.2.5. □

Example 8.2.4 Let P consists of clauses:

1. $p \wedge \sim p \leftarrow$
2. $\sim p \leftarrow$
3. $p \leftarrow$
3. $q \leftarrow$

Let $W = \{p, \sim p, q\}$ and $W^p = \{\mathcal{C}p, q\}$.
$M^p \models_{para} \mathcal{C}(p \wedge \sim p)$, thus satisfying rule (1) by Definition 8.2.5(b). Similarly rules (2) and (3) are satisfied by Definition 8.2.5(b). Hence W^p is a belief set of P. □

Now we will define some rules for the propagation of the direct or indirect affect of contradiction on subformulas[2] w.r.t. a formula. We apply these rules to check when a formula in the body of a rule will affect the head of a rule and propagate reasoning beyond paraconsistency. The rules are given in the following definition:

Definition 8.2.6 (Rules for propagation of contradiction) *The rules by which we can say that a formula* can propagate contradiction *are as follows:*

- *For a formula $G = G_1 \wedge \ldots \wedge G_m$ if $G_i = \mathcal{C}\ddot{F}$ (or $\mathcal{C}_d F$) for an i and $M^p \models_{para} G$, then G can propagate contradiction.*
- *For a formula $G = G_1 \wedge \ldots \wedge G_m$ if $M^p \models_{para} \mathcal{C}\ddot{G}_i$ (or $\mathcal{C}_d G_i$) for an i and $M^p \models_{para} G_j$ for $j \neq i$, then G can propagate contradiction.*

[2]If F is a subformula w.r.t. a formula G then either $F = G$ or F is a constituting formula of G.

- *For a formula $G = G_1$ or ... or G_m if $M^p \models_{para} C\ddot{G}_i$ (or $C_d G_i$, or G_i such that $G_i = C\ddot{F}$ (or $C_d F$)) for an i and $M^p \not\models_{para} G_j$ for any $i \neq j$, then G can propagate contradiction.*
- *For a formula $G = G_1 \wedge \ldots \wedge G_m$ if a G_is can propagate contradiction and $M^p \models_{para} G_j$ for any $i \neq j$, then G can propagate contradiction.*
- *For a formula $G = G_1$ or ... or G_m if a G_is can propagate contradiction and $M^p \not\models_{para} G_j$ for any $i \neq j$, then G can propagate contradiction.*

v we give the definition which states under what conditions an arbitrary POES be transformed into a POES with rules like $F \leftarrow$ (i.e. body less rules). Based on t we define the answer set of a POES.

finition 8.2.7 *Let P be an arbitrary POES, W be a set of simple literals in P W^p be a set of elements ($\in \Phi \cup \Psi$) w.r.t. P. Let F, F', F_is $\in \mathcal{F}$ and G, G_i, G_i's . For every $M^p =< W^p, W >$, P_{M^p} denotes the POES obtained from P by:*

1. *Replacing all rules R of the form:*

$$F \leftarrow G_1 \wedge \ldots \wedge G_m$$

by rules of the form:

$$F \leftarrow$$

iff $M^p \models_{para} C_d F$ and $G_1 \wedge \ldots \wedge G_m$ can propagate contradiction.

2. *Removing from the premises of the remaining rules of P all formulae G such that $M^p \models_{para} G$.*

3. *Removing all remaining rules with non-empty premises.*

en W^p is called an answer set of P iff W^p is a belief set *of P_{M^p}.* □

Given the notations have their usual meanings as given in the definition, the first ıdition can be explained as follows: A rule $R : F \leftarrow G_1 \wedge \ldots \wedge G_m$ is replaced by

a rule $R' : F \leftarrow$, iff

the conclusion of R is affected by contradiction (i.e. $\mathcal{C}_d F$ is **true** in a paraconsistent context), and the premise of the rule R can propagate contradiction.

Basically the first transformation rule indirectly assures that by a belief set a contradiction-affected form of the conclusion of a rule is **true** in a paraconsistent context iff the premise of the rule is affected by contradiction in some way. The other two transformation rules are easy to follow. The second one assures that the conclusion of a rule is **true** in a paraconsistent context w.r.t. a belief set iff the premises are **true**. The third transformation rule in the above definition assures that the conclusion of a rule is not **true** in a paraconsistent context w.r.t. a belief set if its premise is not **true**, unless the conclusion is **true** as the head of some other rule.

By the following examples we demonstrate the working of the last two definitions in computing an answer set of a POES.

Example 8.2.5 Let a POES P be

$$\{p \; or \; q \leftarrow not \; t \wedge (r \; or \; s); \sim q \leftarrow; \; r \; or \; s \leftarrow \sim q\}$$

Let $W_1^p = \{p, \sim q, r\}$ and $W_2^p = \{p, \sim p, s\}$.

By POES transformation of Definition 8.2.7 we get the POES P_{M^p}

$$\{p \; or \; q \leftarrow; \sim q \leftarrow; \; r \; or \; s \leftarrow\}$$

By Example 8.2.1 the W_1^p and W_2^p are the belief sets of P_{M^p}. Hence W_1^p and W_2^p are the answer sets of P. □

Example 8.2.6 Let a POES P be

$$\{p \; or \; \sim p \leftarrow not(p \wedge \sim p); \; p \leftarrow; \sim p \leftarrow\}$$

Let $W = \{p, \sim p\}$ and $W^p = \{\mathcal{C}p\}$.

By POES transformation of Definition 8.2.7 we get the POES P_{M^p}

$$\{p \; or \; \sim p \leftarrow; \; p \leftarrow; \; p \leftarrow\}$$

By Example 8.2.2 W^p is the belief set of P_{M^p}. Hence W^p is the answer set of P. □

Example 8.2.7 Let us consider the POES in Example 8.1.1:

$$\{p \leftarrow not\ q;\ \sim p \leftarrow not\ r;\ c \leftarrow a;\ a \leftarrow\}.$$

We get the following unique answer set: $\{a, c, \mathcal{C}p\}$. □

Example 8.2.8 Let us consider the POES P of Example 8.1.2 consisting of the rules

1. $\sim p \leftarrow$
2. $p \leftarrow$
3. $q \leftarrow \sim (p \wedge \sim p)$

Let $W = \{p, \sim p\}$ and $W^p = \{\mathcal{C}p, \mathcal{C}_d q\}$

$M^p \models_{para} \mathcal{C}(p \wedge \sim p)$. Thus condition [1(b)] of Definition 8.2.7 is met for rule (3). Thus we get the transformed rule (3) as $q \leftarrow$ in the transformed POES P_{M^p}. Rules (1) and (2) remains the same in P_{M^p}.

Now W^p is the belief set of the POES P_{M^p}. Hence the answer set of P is W^p. □

Example 8.2.9 Let us consider the POES P consisting of clauses:

1. $q \leftarrow p\ or\ \sim p$
2. $\sim p \leftarrow$
3. $\ldots$

We can see that P has an unique answer set:

$\{\mathcal{C}p, \mathcal{C}_d q\}$. This answer set is intuitively appropriate. Rule (1) is not directly applicable here as the premise of the rule is not **true**, i.e. $M^p \not\models_{para} p\ or\ \sim p$. $\mathcal{C}p$ holds the information that both p and $\sim p$ is **true** by the theory P (in an inconsistent context). We cannot hold $p\ or\ \sim p$ **true** as it means either $M^p \models_{para} p$ or $M^p \models_{para} \sim p$, which goes against the meaning of $\mathcal{C}p$. But rule (1) is applicable in an indirect way. $M^p \models_{para} \mathcal{C}(p\ or\ \sim p)$. Hence by Definition 8.2.7 condition (1b) and by rule (1) we get $\mathcal{C}_d q$ in the answer set. □

Example 8.2.10 Let POES P consist of the rules

1. $p\ or\ q \leftarrow not\ s$
2. $r \leftarrow not\ p$

We can see that P has two answer sets:

$$\{p\},\ \{q, r\}$$

□

8.2.3 Some properties

Proposition 8.2.1

[Relating Truth-valuation in inconsistent to paraconsistent context]

Any simple formula F of our language $\mathcal{L}$ **true** *by the truth-valuation in an inconsistent context is not necessarily* **true** *by the truth-valuation in a paraconsistent context.* □

Proof: Let $M \models_{in} F$ and $M \models_{in} \sim F$, where $F \in \mathcal{F}$.

Hence by Definition 8.2.3(6) $M^p \models_{para} \mathcal{C}\ddot{F}$ - (a).

Now let us assume that $M^p \models_{para} F$ (or $\sim F$) - (b). Hence by Definition 8.2.3(14) $=|_{para} \mathcal{C}\ddot{F}$.

This is in contrary to (a). Therefore we cannot have (b). Hence the proposition is proved. □

With the following example we illustrate the above proposition.

Example 8.2.11 Let the POES P be

$$\{p \text{ or } q \leftarrow;\ \sim p \wedge \sim q \leftarrow\}$$

P has two answer sets: $\{\mathcal{C}p, \sim q\}$, $\{\mathcal{C}q, \sim p\}$.

By both the answer sets (and their corresponding inconsistent sets) $M \models_{in} p \text{ or } q$ (and $M \models_{in} \sim p \wedge \sim q$, but neither $M^p \models_{para} p \text{ or } q$ nor $M^p \mid_{para} \sim p \wedge \sim q$.) □

Proposition 8.2.2 *A self-inconsistent formula (i.e. a formula of the form $F \wedge \tilde{F}$, where $F \in \mathcal{F}$) is not true in a paraconsistent context.* □

Proof: Let us assume that $M^p (=< W^p, W >) \models_{para} F \wedge \tilde{F}$.

Therefore $M^p \models_{para} F$ and $M^p \models_{para} \tilde{F}$ by Definition 8.2.3(5).

Therefore (a) F, $\dot{F} \in W$ and (b) $M^p \not\models_{para} C\ddot{\phi}_i$ (where $F = F_1 \wedge \ldots \wedge F_m$, $m \geq 1$ and $F_i = \phi_1$ *or* $\ldots$ *or* ϕ_n, $i = 1, \ldots, m$, $n \geq 0$) by Definition 8.2.3(2).
By (a) and Definition 8.2.3(1) $\models_{para} C\ddot{\phi}_i$. This is in contrary to (b).
Therefore our assumption $M^p \models_{para} F \wedge \dot{F}$ proved to be false. Hence the proposition is proved. □

8.3 Relating POES to OES and PAS

In this section we relate the answer set semantics of POES to the answer set semantics of PAS and OES.

The following proposition establishes the relationship of OES with POES.

Proposition 8.3.1 *W^p is an answer set of a non-contradictory OES P by the semantics of OES iff it is an answer set of P by the semantics of POES.* □

Proof: A non-contradictory OES P is contradiction free. Therefore the logical mechanisms to handle inconsistency embedded in the semantical definitions for an answer set of an OES are redundant. The OES P is also a POES. Therefore the logical mechanisms to handle inconsistency embedded in the semantical definitions for an answer set of a POES are also redundant in computing an answer set for the non-contradictory OES P. Thus effectively the semantical definitions for an answer set of OES and the semantical definitions for an answer set of POES has a one to one correspondence in computing an answer set of P. Therefore an answer set of the OES P by the semantics of OES and the answer set of P by the semantics of POES coincides. Hence the proposition is proved. □

The database in the Example 8.2.7 and others involving simple elements belonging to Φ, Φ^x, Ψ, *or* Ψ^x in forming the premises of rules and simple literals forming the heads of rules can be considered as a PAS. The *answer set* given by the POES is respectively the answer set as given by the answer set semantics of PAS. The semantics

for POES gives the same intended semantics for a PAS as given by the answer set semantics for PAS.

The following proposition establishes the relationship of PAS with POES.

Proposition 8.3.2 *W^P is an answer set of a PAS P by the answer set semantics of PAS (as given in Chapter 7) iff it is an answer set of P by the semantics of POES.*
□

Proof: A PAS P is a POES. The language of PAS is a subset of the language of POES. Simple and complex formulae are allowed in the language of POES, but are not allowed in the language of PAS.

The logical mechanisms embedded in the semantical definitions of a POES that handles simple and complex formulae are redundant in computing an answer set of P. Therefore effectively the semantical definitions for an answer set of a PAS has a one to one correspondence to the semantical definitions for an answer set of a POES for P. Hence an answer set of P by the answer set semantics of PAS coincides with the answer set of P by the semantics of POES. □

8.4 Summary

We have expanded the language of objective epistemic specifications to handle inconsistency. Instead of discarding the inconsistent answer sets by trivializing them we make them viable by explicitly capturing the inconsistent information in the theory and thus concluding more information from a theory. This was achieved by the application of Approach $\mathcal{C}$ to OES. By applying $\mathcal{C}_d$ to OES we are able to reason beyond paraconsistency, i.e. we are able to reason from inconsistent information and distinguish between conclusions inferred from consistent and inconsistent information respectively. We now have an expressive language where contradiction in simple formulas are handled and where the information of being being affected by a contradiction are propagated through simple formulas.

Future work needs to be done in applying *Approach* $\mathcal{C} - \mathcal{C}_d$ to the full language of ES.

Chapter 9

Approach $C - C_d$ and Existing Logics

In this chapter we compare *Approach* $C - C_d$ based systems with some of the existing logics handling inconsistency that we reviewed in Chapter 2. This comparison gives us an insight into how *Approach* $C - C_d$ based systems contribute to the enhancement of rational logicality by dealing with inconsistency in an intuitively satisfactory and pragmatic way.

For the discussions in this chapter we do not differentiate between classical negation ($\neg$) and explicit negation ($\sim$) as they are very close in spirit. Wherever reference to classical negation is made consider it as explicit negation.

9.1 Relating to classical logic

In Chapter 1 we discussed the restriction of classical logic and showed how paraconsistent logics are a natural enhancement towards logical rationality. The systems that we have developed based on the inconsistency handling strategies *Approach* $C - C_d$ are all paraconsistent in nature. Moreover they are *explicitly paraconsistent* and have

the added feature of *reasoning beyond paraconsistency*[1]. In Chapter 4 we discussed the system PS that we build over a subset of classical logic (i.e. positive logic programs with classical negation). There we showed how *Approach* $\mathcal{C} - \mathcal{C}_d$-integrated system PS deals with inconsistency in a more intuitive and computationally satisfactory way compared to classical logic's approach of ECSQ (for Ex Contradictione Sequitur Quodlibet).

In Chapter 7 we presented a nonmonotonic system PAS (built on extended logic programs, a nonmonotonic formalism) integrated with *Approach* $\mathcal{C} - \mathcal{C}_d$. The nonmonotonic building block of extended logic programs inherited the classical approach of inconsistency handling (ECSQ). PAS handles inconsistency equipped with the intuitive and logically rational characteristics of *Approach* $\mathcal{C} - \mathcal{C}_d$. This we have again discussed in Chapter 7.

9.2 Relating to traditional paraconsistent approaches

In this section we briefly discuss the limitations of the existing approaches of traditional paraconsistent logics (as also discussed by Priest and Routley [67]). Then we show how *Approach* $\mathcal{C} - \mathcal{C}_d$ is an improvement over them in handling inconsistency.

We do not go into the details of the demerits of the non-adjunctive approaches to paraconsistency as pioneered by Jaskowski [42] or further developed in a thinly disguised form, in the work of Rescher and Brandom [75]. For extensive discussion about the unsuitability of the non-adjunctive approach as a paraconsistent logic, we suggest reference to the paper by Priest and Routley [67]. We sum up the discussion against the approach as given by Priest and Routley below.

Discursive logic (the logic developed on the non-adjunctive approach to avoid ECSQ) may be either single premised or multiple premised. In the first case it is

[1] We discussed this characteristic in details in Chapter 4. *Approach* $\mathcal{C}_d$ delivers it.

classical. In the second it is not really a logic any more. In neither case it is suitable for the investigation of inconsistent theories. The main problem with the discursive approach is just that it does not take the dialethic motivation (that there are true contradictions) seriously. Contradictions may be "true" but this amounts to no more than "true in different worlds". Moreover each possible world is as consistent as any classicist would wish: the approach is much too modally based to accommodate inconsistency satisfactorily.

For a more detailed discussion about objections against the positive-plus approach of da Costa [13] as a paraconsistent logic we again suggest reference to [67]. Here we just summarily mention some of them. The motivation for an evaluation condition in da Costa's logic: $v(A) = 1$ *if* $v(\neg\neg A) = 1$ is ill-motivated. The second objection to da Costa's semantics is that they are non-recursive. Negation in the positive plus approach is not classical negation, but a sub-contrary forming functor and virtually has none of the inferential properties traditionally associated with classical negation. Moreover a change from da Costa's C_w to C_i systems with the 'classicality operator' ($^\circ$) does not change any of the objections already made about the C_w system, but leads to new trouble, as formulas of the form $B \wedge \neg B \wedge B^\circ$ are provable in the system that leads to ECSQ.

Priest and Routley [67] have argued in favor of *relevant* approach to paraconsistency against the last two and have showed its adequacy as a better paraconsistent system. Unlike the non-adjunctive systems, it has an adequate conjunction and a decidedly non-trivial multi-premise deducibility relation. The properties of negation are neat and simple and no extra semantic postulates are added, as in da Costa's approach, to ensure bits of double negation. Moreover, there can be no doubt that the negation of this approach is negation. The semantics are recursive and extensional. Thus $\neg$ is not an intensional functor. Both the laws of excluded middle and non-contradiction hold and negation has all the deducibility relations one would expect. We now [illegible] the merits and demerits of *Approach* $C - C_d$-integrated paraconsistent

systems with respect to relevant systems.

The arguments that Priest and Routley made against the non-adjunctive approach and positive-plus approach does not hold for *Approach* $\mathcal{C} - C_d$ and the arguments in favor of the relevant approach also holds for a paraconsistent system based on *Approach* $\mathcal{C} - C_d$. The system POES based on *Approach* $\mathcal{C} - C_d$ (refer to Chapter 8) has conjunction ($\wedge$) and negation ($\sim$) operators that are close to their classical counterparts. So most of the normal relations between conjunction, disjunction and negation holds. For example:

1. $\models_{para} \sim a \Leftrightarrow \models_{para} \sim (a \wedge b)$
2. $\models_{para} \sim a \wedge \sim b \Leftrightarrow \models_{para} \sim (a \; or \; b)$
3. $\models_{para} \sim a \; or \; \sim b \Leftrightarrow \models_{para} \sim (a \wedge b)$

By *Approach* $\mathcal{C} - C_d$ we capture contradiction by the operator $\mathcal{C}$ and consider that it is true in a single world instead of in different worlds as in non-adjunctive approach. By allowing two levels in our interpretation: an inconsistent level, where we allow formulas of the form F and $\sim F$ to be simultaneously **true**, and an explicit paraconsistent level, where formulas are consistent, i.e. formulas of the form F and $\sim F$ are both not **true** in a theory, but inconsistent complementary pairs of formulas at the inconsistent level are captured explicitly by the operator $\mathcal{C}$, thus making contradictions **true** in an indirect syntactical way at the paraconsistent level. We still allow the law of non-contradiction, as it does not clash with the paraconsistency conditions and as we also want the theory to be *self-consistent.* So this is a unique approach in which the law of non-contradiction (i.e. formulas of the form $F \wedge \sim F$ are **false**) is upheld along with the paraconsistent conditions (i.e. nontrivial reasoning can be carried out in presence of contradiction).

It is confusing to perceive a world where a sentence is assigned the truth-value both **true** and **false**. Priest's relevant approach [63] takes this approach. The problem lies in trying to interpret the world in the same lines as classical logic. We break away

from it by providing a two tier approach in interpreting the world. At the lower level of interpretation (where we allow inconsistency), both a sentence and its negation are assigned unique truth-values, i.e. either **true** or **false**, independently. This clears up the perception of the world. We discuss this point in more details when we discuss the drawbacks of multiple-valued logics in the next section.

In Priest's relevant logic the principle of disjunctive syllogism is invalid, i.e. $\{a, \neg a \vee b\} \not\models_R b$. In POES from a similar specification, i.e. $\{a \leftarrow, \sim a \text{ or } b \leftarrow\}$, we get the only answer set $\{a, b\}$. Thus $\models_{para} b$. So the principle of disjunctive syllogism is valid by our formalism.

Furthermore, we introduce a new horizon in paraconsistent reasoning, *reasoning beyond paraconsistency* which is an important enhancement over the three traditional paraconsistent approaches. We are able to reason not just in presence of inconsistent information but from inconsistent information. In relevant approach what one should get from a theory as given below in the example is not dealt with appropriately.

Example 9.2.1 *We consider a set of formulae*

$$T = \{\neg a;\ a;\ a \rightarrow b\}.$$

where $\rightarrow$ *is material (classical) implication.*

There is a notion that if the *sense* (or objective content) (refer to [65]) of a is less than or equal to the sense of b then $a \rightarrow b$ is **true**. For the above case the objective content of a is more than that of b. Thus $a \rightarrow b$ is **true**, but b is not inferable from the theory, as the principle of disjunctive syllogism is invalid in this logic (this we discussed in Chapter 2 Section 2.1.1 when we reviewed a relevant approach), i.e.

$$\{a;\ \neg a;\ a \rightarrow b\} \not\models_R b$$

So we see that implicative deducibility is very weak. We are not able to gain much information from T by relevant approach. By *Approach* $\mathcal{C}_d$ we derive some very intuitive information: $\{\mathcal{C}a, \mathcal{C}_d b\}$. We not only implicitly capture contradiction and derive it in an explicit syntactic form, but we also infer from it. The inferred belief is

such, that it is affected by contradiction is explicit. Thus, differentiating it from the inferences that are not affected by contradiction.

9.3 Relating to new paraconsistent approaches

In this section we compare *Approach* $\mathcal{C} - \mathcal{C}_d$ to recent approaches of paraconsistency.

9.3.1 Comparing with Multiple-valued Logics

Use of multiple-valued logic provides an inconsistent system S with an epistemological meaning in the sense that S can reflect beliefs of a reasoner (human or machine) who may happen to hold conflicting beliefs [55, 8, 25, 23]. By say Belnap's logic $\mathcal{FOUR}$ [55], from Example 9.2.1 we get the following model: $\{a = \top, b = \top\}$. The model we get by our approach is to some extent comparable with the above model but is different in its epistemic perspective. Saying that *a is believed both* **true** *and* **false** as understood in $\mathcal{FOUR}$ by $a = \top$ is different from the meaning of $\mathcal{C}a$, which means that *a and* $\neg a$ *are both* **true** *in an inconsistent context*, i.e. there is contradictory information available about a and $\neg a$. The multi-valued logic systems remain ontologically inconsistent as far as it agreed that the reality complies with the laws of *Excluded Middle* and *Non-Contradiction*, which means that in every possible state of the real world every statement is either **true** or **false** exclusively. Only a statement can be believed to be **true** and **false** simultaneously, but cannot be **true** and **false** in reality, i.e. ontologically. So there is no ontological paraconsistency, but a kind of epistemic paraconsistency.

Furthermore, the truth-value of an inferred element in an implication (e.g. $\top$ for b in the above example) doesn't reflect the truth-status of b appropriately. b is inferred from an inconsistent information, but by itself is not inconsistent. So giving the truth-value $\top$ to b is committing it beyond what is given by the theory. So, in actuality it would be more intuitively appropriate, if b is evaluated to **false** (if we

pertain to a two-valued logic and take a closed world perspective) or to **unknown** (in a three-valued scenario). The theory is more informative and intuitive then the last two approaches if b is inferred with explicit reference that it is derived from inconsistency. This is achieved by *Approach* $\mathcal{C} - \mathcal{C}_a$.

Belnap's logic has one more drawback. The notion of logical entailment differs from a standard one and there is an unpleasant asymmetry in the semantics of implication: normally logic formulas may assume any truth value from the lattice, except for implications, which are strictly 2-valued.

In none of the systems we have developed we have provided a semantics for classical implication. But it can be easily included. As our logic is two-valued and most of the interaction between *or*, $\wedge$ and $\sim$ are similar to classical logic, an implication ($\sim>$) close in spirit to classical implication can be semantically defined in terms of *or* and $\sim$:

$$\models_{para} F \sim> G \text{ iff } \sim F \text{ or } G.$$

It can be simple enough to introduce a classical disjunction operator ($\vee$) in our language. Then we can have an implication operator closer in spirit to classical implication ($\rightarrow$) as we can express it as

$$\models_{para} a \rightarrow b \text{ iff } \sim a \vee b.$$

Rest of the logical development based on implication follow suits. But it has a little twist because of the paraconsistent nature of our formalism.

Subramanian's [8] paraconsistent logic programming framework is based on multi-valued approach. But a basic element in that language is an annotated literal[2]. Their interpretation of an annotated inconsistent literal $a : \top$ is: it is *known* that a is **inconsistent**, i.e. both **true** and **false**. It is different from Belnap's approach of *believing* a to be **inconsistent**. Subramanian's intuition is closer to how we model the inconsistency in the world, i.e. objectively. But accepting something to be be both **true** and **false** from any perspective objective (i.e. as known) or subjective

[2]Annotated literals have been discussed in Section 2.2.2, Chapter 2 in reviewing *RI* logic.

, as believed) is in some ways unintuitive. Implicitly assigning two truth-values a thesis a by assigning it the truth-value **inconsistent** is viewing the world in a torted way. It is interpreting the world with double meaning, thus quite confusing. The problem lies in trying to interpret a world having inconsistency the same y as done for a consistent world, where every positive thesis is assigned a unique ıth-value, which makes the world very easy and clear to understand. In this way of erpretation for the presence of a negative thesis in a theory the positive thesis is signed the truth-value **false**. But existence of both, a positive and a negative thesis a theory jeopardizes this way of interpreting the world. The approach which we opose departs from the existing model theoretic approach of interpreting the world, the way that both negative and a positive thesis are assigned truth-values inde-ndently. Furthermore, any thesis is either assigned the truth-value **true** or **false**, ıt not simultaneously. We refer to this level of interpretation as *interpretation in an consistent context*. In the meta-level of interpreting the world by introducing the nnective $\mathcal{C}$ the inconsistency in the inconsistent context is captured, thus providing paraconsistent environment in the meta-level. By this approach of interpreting the orld the confusion arising from the double meaning of the truth-value **inconsistent** cleared away.

Now let us consider the set of formula

$$S = \{\neg a : t;\ a : t;\ b : t \leftarrow a : t\}.$$

his has two models by Subramanian's formalism:

$$\{a : \top, b : t\} \text{ and } \{a : \top, b : \top\}.$$

he first is the least model, hence giving the intended semantics of S. Our argument gainst this is that the meaning of the program is lost as the consequence of the ause does not have any relevance to the premise w.r.t. the model. If a premise is **ıconsistent**, it is unintuitive to say that the consequence is **true**. We get a more ıtuitive meaning of S by our approach.

One more thing we criticize about both Belnap's and Subramanian's semantics.

They assign the truth-value **inconsistent** to the formula $F \wedge \neg F$. This is a self-inconsistent statement. We are strong about refuting any such statement in our formalism. It is irrational for an intelligent system to accept self-inconsistent formulas with any level of truth. Under Belnap and Subramanian's semantics it is truth valuated to **inconsistent**. Thus there is an ambivalence to the truth status of the formula. There is a chance that it can also be **true**. There should never be any doubt about a self-inconsistent formula's truth status in the mind of a rational reasoner. That is why by our semantics it always gets the truth value **false**.

9.3.2 Comparing with *RI* logic

The main restriction of *RI* logic with respect to any system based on *Approach* $\mathcal{C} - \mathcal{C}_d$ is that it is motivated by a view that the real world is inherently consistent, while inconsistency of its logical description occurs only in the mind of the beholder. This is different from our view in which we not only say that inconsistency can arise because of the reasoner's belief but also because the world is inherently inconsistent, i.e. ontologically inconsistent. It can be a reality that a computer's knowledge base contains contradictory information. Considering this inconsistency at the epistemic level portrays the characteristic of the intelligent computer system to be self-inconsistent. This we do not agree to be a desirable feature for an intelligent system. Why should we rely on a confused reasoner. Characteristically the inconsistency would be appropriately portrayed if the contradictory information is considered at an ontological level. For example, a situation in which the computer stores contradictory information coming from different sources that are true in their own right in the circumstance and thus contradictory in reality when they come together.

In *Approach* $\mathcal{C} - \mathcal{C}_d$ an epistemic notion of inconsistency is allowed. But it is a very cautious approach towards inconsistency compared to the epistemic notions in *RI* logic or other lattice-valued logics (which we referred to in the last section). Intelligent systems modeled on our approach do not consider its information as beliefs but as

knowledge. Information which are explicitly considered assumptions (i.e. negation by default) are considered as beliefs. And contradictory information derived from assumptions are beliefs too and they provide the epistemic level of inconsistency. But for our systems as the manifestation of both epistemic and ontological inconsistency is similar, they are dealt in the same way. (This we discussed earlier in Chapter 4.)

RI is tolerant to only epistemic inconsistency (different from our notion of epistemic inconsistency) but intolerant to ontological inconsistency, whereas, our approach based systems are tolerant to both epistemic and ontological inconsistency, but without letting the theory becoming inconsistent. The specification in Example 9.2.1 can be represented in *RI* logic as follows: (refer to Section 2.2.2 in Chapter 2)

$$\{\neg a : t;\ a : t;\ a : t \rightarrow b : t\}.$$

It has no models by *RI* logic as it is ontologically inconsistent. But a theory as given below:

$$S_1 = \{a : \top;\ a : t \rightarrow b : t\} \models_{RI} b : t.$$

The problem that we face here is that when we have the information

$$T_2 = \{c : t;\ c : t \rightarrow \neg b : t\}$$

along with S_1. Then we are unable to differentiate between the epistemic status of $b : t$ inferred from part S_1 and $\neg b : t$ inferred from part S_2. Hence, $S_1 \cup S_2$ does not have any model since it produces ontological inconsistency. But in actuality $b : t$ inferred from S_1 is inferred from inconsistency. Thus it should have a lower epistemicity than $\neg b : t$ inferred from the part S_2 of the theory. So $\neg b : t$ should be obtainable in a model of $S_1 \cup S_2$ instead of following the RAA (for *reductio ad absurdum*) approach. *Approach* C_d takes care of this problem by distinguishing between $b : t$ inferred from S_1 and $\neg b : t$ inferred from S_2 by prefixing the former by C_d.

Epistemic entailment ($|\!\approx_{RI}$) does neither provide much information from S_1 if we replace ontological implication by epistemic implication. Thus,

$$T = \{a : \top;\ a : t \sim\!> b : t\} \ |\!\approx_{RI} b : t.$$

But $T \ |\!\approx_{RI} b : \bot$, which is a little more informative than the former. This still lacks

the depth of information which we get by *Approach* $\mathcal{C}_d$.

9.3.3 Comparing with *LEI*

The Logic of Epistemic Inconsistency [58] (presented in Chapter 2), as its name suggests only takes care of epistemic inconsistency. It is not capable of handling ontological inconsistency. Whereas *Approach* $\mathcal{C} - \mathcal{C}_d$ based systems handles both epistemic and and ontological inconsistency. Moreover as we mentioned earlier *Approach* $\mathcal{C}_d$ opens up a new realm of reasoning, that of *reasoning beyond paraconsistency*, which is beyond the scope of *LEI*.

9.3.4 Comparing with Vivid logic

Wagner's Vivid logic [93] (as reviewed in Section 2.2.4), provides four different reasoning schemes where contradiction is considered undesirable in different degrees and are neutralized accordingly. Let us reconsider Example 2.2.2 given in presenting Vivid logic in Chapter 2:

Example 9.3.1 *Let* $X = \{p;\ \sim p;\ \sim q;\ q \leftarrow p;\ r \leftarrow p;\ \sim r \leftarrow q\}$ *then:*

$LC(X) = \{p, \sim p, \sim q, q, r, \sim r\}$ *(by liberal reasoning)*

$CrC(X) = \{\sim q, r\}$ *(by credulous reasoning)*

$CC(X) = \{\sim q\}$ *(by conservative reasoning)*

$SC(X) = \emptyset$ *(by skeptical reasoning)*

Although p is contradictory, it constitutes evidence for a counter-argument against $\sim q$ *in skeptical VL. This is not the case in conservative VL, where no counter-argument against* $\sim q$ *is possible. In credulous VL it suffices for r to hold that it is supported by (the contradictory) p and not doubted by any counter-argument.* □

In our intuitive understanding liberal reasoning has a glut of information with no restraint to what can be inferred in the presence of contradiction. Thus though it is paraconsistent it stops being a rational reasoning process anymore. Credulous,

conservative and skeptical reasoning have a strong notion of rationality based on reductio ad absurdum. They neutralize the effect of contradiction in different degrees, but as we see not much information is inferred. At every stricter level of reasoning we tend to lose more information. Intuition says that we should be able to infer as much information as possible without being irrational about what we are inferring. The neutralization strategy follows the classical or the traditional information processing trend of reducing absurdity (here absurdity is contradictions and inferences from them). So though it provides schemes to reason in presence of contradiction with different degrees of neutralization depending upon the support and suspicion, it is some problems.

If a system based on Wagner's approach is provided with a mechanism to find inconsistent pairs in a model by liberal reasoning, then one can follow one of the other reasoning methods provided by the system, i.e. credulous, conservative or skeptical reasoning. But one would not know which one to choose. Moreover the last three reasoning methods have a strong notion of rationality based on reductio ad absurdum. They neutralize the effect of contradiction in different degrees. But as we see the information about either p or $\sim p$ is lost. So the inconsistency in the system is not recorded. Hence the reasoner based on the last three reasoning methods would be unaware of the inconsistency at the liberal reasoning level if there is no way to capture inconsistency after liberal reasoning. A similar mechanism of recognizing/capturing inconsistency have to be there after credulous and conservative reasoning levels to be able to go on to the corresponding next levels of reasoning, i.e. conservative and skeptical reasoning respectively. So we see that the whole system turns out to be not very efficient with all four reasoning methods working together if there is an inconsistency in the system.

By *Approach* $\mathcal{C} - \mathcal{C}_d$ based systems (e.g. system PS as given in Chapter 5) we get a unique answer set from the theory in the above example:

$$\{\mathcal{C}p, \sim q, \mathcal{C}_d q, \mathcal{C}_d r, \mathcal{C}_d \sim r\}$$

Here we do not make the contradiction disappear as against neutralizing in vivid logic. But we capture the contradiction explicitly. Moreover in any inference made from contradiction or contradiction-affected information, the information that it is inferred from contradiction related information is propagated. So the neutralizing effects at different levels of reasoning as in Wagner's approach becomes unnecessary.

9.3.5 Comparing with Extended WFSX for Paraconsistent Logic Programs

Sakama [81] (refer to Chapter 2) in trying to give a proper semantics to extended logic programs giving inconsistent models resorted to a seven-valued logic following in the footsteps of Ginsberg [37]. We are critical about his approach in the same tune as we are about the other multi-valued logics we have discussed. Their notion of inconsistency is of epistemic inconsistency (which is different from our notion of it), i.e. if a thesis p and its negation $\neg p$ are in a theory, then p is assigned the truth-value **inconsistent** ($\top$), which means p is 'believed' to be both **true** and **false**. This is actually reporting self-inconsistency. Reporting self-inconsistency is an appropriate action, but to start with allowing it is inappropriate. Moreover this proliferation of truth-values can be a little bewildering.

In our approach p and $\neg p$ in a theory is considered as ontological inconsistency. Our approach of modeling the information in the world is objective, unlike Sakama's and some of the existing multi-valued approaches, which are subjective. Instead of internalizing the information (that is considering the information as ones own belief), an intelligent system based on our approach considers the information present in its knowledge base as information which it has gathered from the environment. Presuming that an intelligent system would gather information which it considers authentic, our approach based intelligent systems face inconsistent information that is ontological. This is a more appropriate way to model an inconsistent world in the context of an intelligent system whose information is from the environment around

it, presumably from experts feeding information to it. Moreover our formalism is two-valued and thus straight forward to understand. It behaves like a traditional classical logic in absence of inconsistency.

A formula of the form $p \wedge \neg p$ is truth-valuated to **inconsistent** in Sakama's framework. We argue against it in the same line as we did for Subramanian's formalism (refer to Section 9.3.1).

Sakama independently developed an idea similar to *reasoning beyond paraconsistency*. In his framework he provides a way to detect a fact affected by an inconsistent information and distinguish it from other meaningful information. For this purpose he introduced the notation of *suffixed literal* L^{Γ}, where L is a literal from the Herbrand base w.r.t. a program P and Γ is a collection of sets of ground literals. Each of these sets are comprised of facts which are used in deriving L in P. A literal in a fixpoint (as defined by Sakama), thus a proven fact is *suspicious* if every proof of the fact contains an inconsistent information. Hence from the program below:

$$P = \{a \leftarrow \neg b;\ \neg b \leftarrow c \wedge not\ b;\ c \leftarrow;\ \neg c \leftarrow;\ d \leftarrow\}$$

we get the *suspicious well-founded model*:

$$< \{c^{\{\emptyset\}}, \neg c^{\{\emptyset\}}, d^{\{\emptyset\}}, a^{\{\{\neg b, c, not\ b\}\}}, \neg b^{\{\{c, not\ b\}\}}\}; \{b, \neg a, \neg d\} >$$

Thus, d is **true**, c is **contradictory**, while a and b are **true with suspect** and **false with suspect**, respectively. $b, \neg a$ and $\neg d$ are **true by default**.

Problem with Sakama's approach of *reasoning beyond paraconsistency* is that one have to keep track of all the facts which have been used in proving an element in the model. This is computationally staggering. Moreover another mechanism have to be added over the existing framework to detect whether there is an inconsistency among the facts which have been used in proving a literal. Along with that for the model theoretic approach for this formalism two more truth-values have to be added to the existing seven, which is in a way a bit too ad-hoc in terms of juggling with truth-values.

Our formalism provides a more natural and computationally efficient way to detect

a suspicious fact. If contradiction affects a premise of a rule that information is propagated through the consequence of the rule and hence to any other premise which is affected by this consequence. So the information of being affected by a contradiction is transitive by the inherent mechanism of our formalism and the bookkeeping of proving facts need not be carried around to detect whether a proven fact is proved using a contradictory information. By our formalism we get the following answer set from P by the answer set semantics of PAS:

$$\{\mathcal{C}\ddot{c}, d, \mathcal{C}_d\neg b, \mathcal{C}_d a\}$$

from which it is evident what is contradictory and what is contradiction-affected or suspicious.

9.4 Relating to new classical/information processing approaches

In this section we compare *Approach* $\mathcal{C} - \mathcal{C}_d$ based systems with new approaches (though they follow the same classical strategy of *ECSQ* and *RAA*) to handle inconsistency.

9.4.1 Comparing With $\mathcal{CRSX}$

Two critical remarks can be made about the $\mathcal{CRSX}$ semantics [60] (which we reviewed in Chapter 2). Firstly, it is not clear whether the revision policy of $\mathcal{CRSX}$ satisfies the principle of *minimal change*, i.e. whether it is generally guaranteed that never 'too much' information gets lost by the suggested revisions (e.g. for $X_1 = \{p \leftarrow not\neg p;\ q \leftarrow p;\ \neg q \leftarrow p\}$ the $\mathcal{CRSX}$ model is $< \emptyset, \{q\}, \emptyset, \{p, q\} >$ while the intended answer set following *Approach* $\mathcal{C}-\mathcal{C}_d$ (refer to Chapter 7) is $\{p, \mathcal{C}q\}$). Our approach provides more intuitive information from X_1. Secondly, no solution is given for 'hard' (ontological) contradictions not depending on weakly negated premises, i.e. contradictions which are not arising on account of assumptions. Such programs do

not have any meaning in the $\mathcal{CRSX}$ semantics. This is an important contribution of *Approach* $\mathcal{C} - \mathcal{C}_4$.

9.4.2 Comparing with the 'Semantics of weighted mc-subsets'

The *semantics of weighted mc-subsets* of Lozinski [51] (which we reviewed in Chapter 2) is a maximal consistency based approach, where inconsistent sets are divided into maximally consistent subsets. Splitting out of inconsistent sets into maximally consistent subsets has the effect of precluding the purely logical analysis of the whole situation. This we see in the probabilistic heuristics taken by Lozinski's approach.

We are also critical about the unintuitive conclusions we get in certain scenarios by his approach. Let us check it out with the following example:

Example 9.4.1 *Given a set of formulae:*

$$S = \{a;\ \neg a;\ a \rightarrow b\},$$

it has two maximally consistent subsets:

$$s_1 = \{a;\ a \rightarrow b\} \text{ and } s_2 = \{\neg a;\ a \rightarrow b\},$$

their sizes are:

$$| s_1 | = 2 \text{ and } | s | = 2,$$
$$\text{and for } S \text{ the size } | S | = 3.$$

and weights are:

$$w(s_1) = \tfrac{1}{3} \text{ and } w(s_2) = \tfrac{1}{3}.$$

s_1 *has the model* $\{a, b\}$ *and* s_2 *has two models* $\{\neg a, b\}$ *and* $\{\neg a, \neg b\}$. *Evidence of* a *and* $\neg a$ *in* S *is as follows:*

$$E(S, a) = \frac{\frac{2}{3}}{\frac{2}{3}+\frac{2}{3}+\frac{2}{3}} = \frac{1}{3} \text{ and } E(S, \neg a) = \frac{\frac{2}{3}+\frac{2}{3}}{\frac{2}{3}+\frac{2}{3}+\frac{2}{3}} = \frac{2}{3}$$

Thus it is most likely from the above example that $\neg a$ is **true** as $E(S, a) \prec E(S, \neg a)$. But this is unintuitive. Lozinskii's semantics does not give the intuitive meaning of the program. What is intended by the formulas is (if $\rightarrow$ denotes classical implication) as follows:
By classical logic S does not have any model as a and $\neg a$ both can never be **true** in a model. So let we assume that truth-values of a and $\neg a$ are decoupled, i.e. they are independent of each other. Then all the three formulas of S are **true** only when a, $\neg a$ and b are all assigned **true** by an interpretation. Hence the model of S is $M = \{a, \neg a, b\}$. By the semantics of PS (refer to Section 5), M is considered as the interpretation of S, and the corresponding p-interpretation, which is also a model of S is $\{\mathcal{C}a, \mathcal{C}_d b\}$.

Here the possibility of a and $\neg a$ being **true** is equal by the model as that is what $\mathcal{C}a$ captures. In addition, as the inference of b is affected by the paraconsistent information $\mathcal{C}a$, we get $\mathcal{C}_d b$ by *Approach* $\mathcal{C}_d$.

We have already pointed out while comparing with the other logics dealing with inconsistency that, our approach based systems open up a new horizon of reasoning i.e. reasoning beyond paraconsistency, that is beyond Lozinskii's approach.

9.5 Summary

In this chapter we compared *Approach* $\mathcal{C} - \mathcal{C}_d$ to the existing traditional and new approaches in the classical realm and also in the paraconsistent realm. By a direct comparison with individual systems we come to an understanding of the merits of our system with respect to the existing systems handling inconsistency.

Chapter 10

Conclusions

In this final chapter, we summarize the results of this research, highlight the salient features of our contribution and describe the future directions.

10.1 Summary of Research

In this section we summarize the results of our research.

In Chapter 4 we proposed some fundamentally new ideas of handling inconsistency. Based on those ideas we developed two strategies to handle inconsistency. They are *Approach* $\mathcal{C}$ and *Approach* $\mathcal{C}_d$. We developed three logical systems based on these approaches in the broader framework of logic programming.

The first system we developed is that of Paraconsistent Specifications (PS) an extension of positive logic programs. In Chapter 5 we presented the model theoretic semantics of PS and discussed its properties. In Chapter 6 we presented a constructive semantics for PS and proved its correspondence to the model theoretic semantics.

The second system we developed is a nonmonotonic logical system that we call Paraconsistent Assumptive Specifications (PAS). It is based on extended logic programs, with the inconsistency handling strategies *Approach* $\mathcal{C} - \mathcal{C}_d$ embedded in it. This we presented in Chapter 7.

Next we developed a very generalized logic programming framework based on the inconsistency handling strategies. This we called Paraconsistent Objective Epistemic Specifications (POES). We presented it in Chapter 8.

Finally in Chapter 9 we compared our approach of handling inconsistency to several other existing approaches of inconsistency handling.

10.2 Contributions and discussion

In this thesis we have proposed some fundamental ideas of how to handle inconsistency. Based on which we developed two strategies to handle inconsistency in the realm of logical theories in a rational way. Classical logic's inaptitude to handle inconsistency in a satisfactory way is a strong motivation for our work. Furthermore the traditional and the new approaches of handling inconsistency, whether they are *paraconsistent* or *information processing* approaches, have their limitations. *Approach* $\mathcal{C} - \mathcal{C}_d$ is able to resolve many of them. Furthermore, it opens up a new horizon in reasoning, *reasoning beyond paraconsistency* i.e. to be able to reason from inconsistent information and at the same time distinguish between information inferred from inconsistent information and that from consistent information. This is beyond the scope of the existing logics handling inconsistency.

Approach $\mathcal{C}$ presents a new knowledge representational scheme to capture inconsistent information in a syntactically explicit way, without making the theory inconsistent. *Approach* $\mathcal{C}_d$ presents a new knowledge representational scheme to represent information inferred from inconsistent information. Hence, enabling us to reason from inconsistent information without loosing the rationale that, inconsistency based inferences have a different epistemic status from inferences made from consistent information.

We are strong proponents of paraconsistent approaches to common-sense or artificial intelligence applications, though there may be possible objections to be raised

against this idea.

The first is a conceptual objection based on the argument that contradiction is indicative of error occurrence and therefore the efforts should be directed towards the correction of these errors and not in propagating them. It could be added that an inconsistent description is a description of nothing. The second is a technical objection based on the belief that, in any reasonable logical system, inconsistent theories are trivial (everything is a theorem).

We think that *Approach* $C - C_d$ based systems presented here is evident enough to remove the second objection. A logical system able to perform reasoning in the presence of contradiction, without trivializing, being at the same time strong enough to make this reasoning useful, is perfectly plausible. We would further discuss the first point.

Contradiction is effectively a test for error. If a contradiction is deduced from a set of premises, this imply the inconsistency of this set. Semantically this means that no models can exist in which all of them are true. Thus it is a positive indication that these premises make a bad description of the world. To stay free of contradiction is one of the main methodological prescription of standard scientific practice.

But the situation is different when common-sense reasoning and artificial intelligence applications are considered. Then the inaccuracy of the knowledge is recognized in advance, and so the occurrence of contradictions does not provide such strong indication. It may demand an effort to get more precise informations, but this refinement cannot be done beyond the limits of the knowledge available at a given time. In spite of it, reasoning must be done and decisions taken.

This kind of situation reveals that the role of reasoning is not exactly to come up with conclusions to be assumed as true in situations satisfying the premises. This picture fits well with deductive reasoning and it is so prevalent as a paradigm that it is often taken as a general expression for reasoning.

Actually, the role of reasoning is to perform an analysis of the epistemic relations

within the information available. It is to compose and judge evidences, to resolve conflicts (when possible) and to come up with relations between evidences and possible conclusions. These conclusions, in opposition to deductive ones, can never be detached from the premises in support of them. Thus the system evolved is *relevant*[1] in nature in these situations. A contradiction then means simply that there are evidences in support of F as well there are evidences supporting $\neg F$.

It is worthwhile at this point to make a clear statement about the position which we defend. We are not affirming here (at least not yet) to the stronger motivation of paraconsistency i.e. *true contradictions are a reality* but to the weaker pragmatic motivation that, in common-sense and artificial intelligence applications involved in computer information processing, inconsistency is an inevitability (for various reasons and circumstances that we have elaborated in the introductory chapter). So the development of the paraconsistent approach $\mathcal{C}-\mathcal{C}_d$ and paraconsistent systems based on it is a pragmatic approach towards handling inconsistency in realistic logical systems.

We started with the motivation of developing a paraconsistent system which does not have the limitation of trivialization of the classical system. In classical logic if ϕ is **true** in τ it is a logical consequence of τ, i.e. $\tau \models \phi$. If ϕ and $\dot{\phi}$ is **true** in τ then $\tau \models \varphi$, but φ may not be **true** in τ. By $\tau \models_{\mathcal{C}-\mathcal{C}_d} \phi$, it means that ϕ is a paraconsistent consequence of τ, i.e. if ϕ is **true** in τ and $\tau \not\models_{\mathcal{C}-\mathcal{C}_d} \mathcal{C}\phi$ then ϕ is entailed from τ. An arbitrary literal φ not **true** in τ cannot be paraconsistently entailed by τ. The reason for being able to deduce arbitrary literal in classical logic is the Principle of Or Introduction along with the presence of inconsistency. If $\tau \models \alpha$ then $\tau \models \alpha$ *or* β (by Or Introduction). Now if $\tau \models \sim \alpha$, then $\tau \cup \{\alpha \text{ or } \beta\} \models \beta$. Thus one can entail arbitrary elements based on inconsistent pairs. But by our approah as soon as we have α and $\sim \alpha$ **true** in τ, we get $\tau \models_{\mathcal{C}-\mathcal{C}_d} \mathcal{C}\alpha$. Hence the Or introduction is restricted. Thus we are unable to get arbitrary elements based on an inconsistent pair and Or Introduction, by our paraconsistent system.

[1]Relevaat approaches have been discussed in Chapter 2, Section 2.1.1.

10.3 Future Work

In this section we discuss the work that can be done in the future based on the research that we have presented here. We itemize them below:

- In this work we have developed an intuitive paraconsistent approach to handle inconsistency and have enhanced the horizon of paraconsistent reasoning by developing a strategy that allows *reasoning beyond paraconsistency*. We have shown its applicability to different systems classical and nonclassical (here, nonmonotonic) and have developed those systems integrated by the paraconsistent approaches, *Approach* $\mathcal{C}$ and *Approach* $\mathcal{C}_d$. We have presented a semantical foundation for the systems. But what can be pursued beyond our work is that of developing a computational procedure for the systems. What can be done as future work is to develop proof procedures for systems PS, PAS and POES.

- All the systems that we have developed are an extension of the restricted framework of logic programming. It would be a interesting endeavour to develop an extension of first order logical framework based on our new ideas of inconsistency handling.

- As we have mentioned in summarizing Chapter 8, as a future scope of the work, *Approach* $\mathcal{C} - \mathcal{C}_d$ can be applied to the complete language of ES (for epistemic specifications) [29, 33]. Investigations into this will give insights into the dynamics of interactions between intensional functors (i.e. the modal operators K and M) and the underlying paraconsistent logic. Furthermore, paraconsistent bases would allow for the elaboration of novel modal logics. Little has yet been done in this area.

- The standard approaches to probability theory are squarely based on classical logic. However, they can be alternatively and easily based on a paraconsistent approach (such as *Approach* $\mathcal{C} - \mathcal{C}_d$). We perceive, doing so produces several

advantages [66]. An especial advantage is that we can have sensible evaluations of probabilities of statements relative to inconsistent information. This can be another direction in which research can be pursued based on our work.

Bibliography

[1] W. Ackermann. Begruendung einer strengen implikation. *Journal of Symbolic Logic*, 21:113–128, 1956.

[2] Carlos E. Alchourron, Peter Gardenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Symbolic Logic*, 50(2):510–530, June, 1985.

[3] J. Alferes, P. Dung, and L. Pereira. Scenario semantics of extended logic programs. In Proc. Second Workshop on LPNMR. MIT Press, 1993.

[4] A. R. Anderson and N. D. Belnap. *Entailment.* Princeton University Press, 1975.

[5] A.I. Arruda. A survey of paraconsistent logic. In *Mathematical Logic in Latin America*, pages 1–41. Reidel, Dordrecht, 1979.

[6] F. G. Asenjo and J. Tamburino. Logic of antinomies. *Notre Dame Journal of Formal Logic*, 16:272–278, 1975.

[7] G. Berkeley. The analyst. In A. C. Fraser, editor, *Collected Works*, volume 3. Oxford, 1901. First published in 1734.

[8] H. Blair and V. S. Subrahmanian. Paraconsistent foundations for logic programming. *Journal of Non Classical Logic*, 1989.

[9] H. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computing Science*, 68:135–154, 1989.

[10] C. Boyer. *The History of the Calculus and its Conceptual Development*. Dover, New York, 1949.

[11] W. A. Carnielli, L. Farinas del Cerro, and M. Lima Marques. Contextual negations and reasoning with contradictions. In *Proc. of Intl. Joint Conf. on AI*, pages 532–537, 1991.

[12] L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logics and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[13] N.C.A. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15:497–510, 1974.

[14] N.C.A. da Costa and E.H. Alves. A semantical analysis of the calculi C_n. *Notre Dame Journal of Formal Logic*, 18:621–630, 1977.

[15] N.C.A. da Costa and E.H. Alves. Relations between paraconsistent logic and many-valued logic. *Bull. Section on Logic*, 10:185–191, 1981.

[16] N.C.A. da Costa, J.J. Lu L.J. Henschen, and V.S. Subrahmanian. Automatic theorem proving in paraconsistent logics. In *Proc. of 10th. Intl. Conf. on Automated Deduction*, 1990.

[17] N.C.A. da Costa and D. Marconi. An overview of paraconsistent logic in the 80's. In *Logica Nova*. Akademie Verlag, Berlin, 1990.

[18] P. Dung. Negations as hypotheses: An abductive foundation for logic programming. In *Logic Programming: Proceedings of the 8th International Conference*, pages 3–17, 1991.

[19] R. Dworkin. *Taking Rights Seriously*. Duckworth, London, 1977.

[20] M. Van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of ACM*, 23(4):37–54, 1976.

[21] J.A. Fernandez, J. Lobo, J. Minker, and V.S. Subrahmanain. Disjunctive lp + integrity constraints = stable model semantics. *Annals of Mathematics and Artificial Intelligence*, 8(3-4), 1993.

[22] P. Feyerabend. In defence of aristotle. In G. Radnitzky and G. Anderson, editors, *Progress and Rationality in Science*. Reidel, Dordrecht, 1978.

[23] M. Fitting. Billatices and the semantics of logic programming. *The Journal of Logic Programming*, 11:91–116, 1991.

[24] Melvin Fitting. A kripke-kleene semantics for logic programming. *Journal of Logic Programming*, 4:295–312, 1985.

[25] Melvin Fitting. Negation as refutation. In *Proceedings of the IEEE Fourth Annual Symposium On Logic In Computer Science*, pages 63–70. IEEE Computer Society Press, 1989.

[26] J. Fox, P. Krause, and S. Ambler. Arguments, contradictions and practical reasoning. In *Proceedings, ECAI-92*, pages 623–627. Wiley, 1992.

[27] 1848-1925 Frege, Gottlob. *Logical investigations*. Basil Blackwell, 1977.

[28] M. Gelfond. On stratified autoepistemic theories. In *Proceedings of AAAI-87*, pages 207–211, 1987.

[29] M. Gelfond. Strong introspection. In *Proceedings of AAAI-91*, pages 386–391, 1991.

[30] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proceedings of the 7th. International Conference on Logic Programming*, pages 579–597. MIT Press, 1990.

[31] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9 (Nos. 3 and 4), 1991. Selected papers from the 7th. ICLP, 1990.

[32] M. Gelfond and H. Przymusinska. Definitions in epistemic specifications. In *Proceedings of the 1st. Intl. Workshop of Logic Pogramming and Non-monotonic Reasoning*, pages 245–259. MIT Press, 1991.

[33] M. Gelfond and H. Przymusinska. Reasoning in open domains. In *Proceedings of the 2nd. Intl. Workshop of Logic Pogramming and Non-monotonic Reasoning*, pages 397–413, 1993.

[34] M. Gelfond and V.Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proceedings of the 5th. Logic programming symposium*, pages 1070–1080. MIT Press, Cambridge, Massachussets, 1988.

[35] Suryanil Ghosh. Approach $\mathcal{C} - \mathcal{C}_d$: A new contradiction handling strategy & Extended Logic Programs. In *Proceedings of the Third Golden West International Conference on Intelligent Systems*. Kluwer Academic Press, June 6-8 1994.

[36] Suryanil Ghosh. Paraconsistency and beyond: A new approach to inconsistency handling. In *Proceedings of the Eighth International Symposium on Methodologies for Intelligent Systems (published as Lecture Notes in Computer Science, LNAI, vol 869)*, pages 531–540. Springer-Verlag, October 1994.

[37] M. L. Ginsberg. Multi-valued logics. In *Proc. AAAI-86*, pages 243–247. Morgan Kaufmann, 1986.

[38] Matthew Ginsberg, editor. *Readings in Nonmonotonic Reasoning*. Morgan Kaufman, Los Altos, 1987.

[39] S. Hanks and D. McDermott. Nonmonotonic logic and temporal projection. *AI*, 33, 1987.

[40] S. Hook. *The Paradoxes of Freedom.* University of California Press, 1962.

[41] K. Inoue, M. Koshimura, and R. Hasegawa. Embedding negation as failure into a model generation theorem prover. In D. Kapur, editor, *Proceedings of the 11th. Intl. Conference on Automated Deduction*, pages 400–415, Saratoga Springs, NY, USA, June 1992. Springer-Verlag.

[42] S. Jaskowski. A calculus of propositions for contradictory deductive systems. *Studio Soc. Scu. Torunensis*, section A.1:55–77, 1948.

[43] S. Jaskowski. The propositional calculus for contradictory deductive systems. *Studia Logica*, 24:143–157, 1969.

[44] M. Kifer and A. Li. On the semantics of rule-based expert systems with uncertainty. In M. Gyssens, J. Paredaens, and D. V. Gucht, editors, *Proc. of the 2nd. Intl. Conf. on Database Theory: Lecture Notes in Computer Science*, volume 286, pages 102–117. Springer-Verlag, Bruges, Belgium, 1988.

[45] Michael Kifer and Eliezer L. Lozinskii. RI: A logic for reasoning with inconsistency. In *Proceedings of the IEEE Fourth Annual Symposium On Logic In Computer Science*, pages 253–262. IEEE Computer Society Press, 1989.

[46] S.C. Kleene. *Introduction to Metamathematics.* Van Nostrand Reinhold, New York, 1957.

[47] T. Krishnaprasad, M. Kifer, and D. S. Warren. On the declarative semantics of inheritance networks. In *Proc. of Intl. Joint Conference in AI*, pages 1099–1103. Morgan Kaufmann, 1989.

[48] I. Lakatos. Falsification and the methodology of scientific research programs. In *Collected Works*, volume 1. Cambridge University Press, 1978. First published in 1970.

[49] H. J. Levesque. Making believers out of computers. *AI*, 30:81–107, 1986.

[50] J. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1987.

[51] E. L. Lozinskii. Resolving contradictions: A plausible semantics for inconsistent systems. To appear in Journal of Automated Reasoning, 1993.

[52] J. Lukasiewicz. *Selected Works.* North-Holland, Amsterdam, 1970. the work referred to was written in 1910.

[53] Jack Minker and Carolina Ruiz. On extended disjunctive logic programs. In J. Komorowski and Z. W. Ras, editors, *Proceedings of the Seventh International Symposium on Methodologies for Intelligent Systems*, pages 1–18. Springer-Verlag, 1993. Lecture notes in AI, June 1993.

[54] P. H. Morris. Autoepistemic stable closures and contradiction resolution. In *Proceedings of the Second Workshop on Nonmonotonic Reasoning*, pages 60–73, 1988.

[55] Jr. N. D. Belnap. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*, pages 8–37. D. Reidel Publishing Company, 1977.

[56] D. Nelson. Constructive falsity. *Journal of Symbolic Logic*, 14(1):16–26, March 1949.

[57] Tarcisio Pequeno. A logic for inconsistent nonmonotonic reasoning. Technical report, Department of Computing Science, Imperial College, London, 1990. Technical Report 90/6.

[58] Tarcisio Pequeno and Arthur Buchsbaum. The logic of epistemic inconsistency. In *Proc. of Second Intl. Conference of Principles of Knowledge Representation and Reasoning*, pages 453–460. Morgan Kaufmann, 1991.

[59] L. M. Pereira and J. J. Alferes. Wellfounded semantics for logic programs with explicit negation. In *Proceedings, ECAI-92.* Wiley, 1992.

[60] L. M. Pereira, J. J. Alferes, and J. N. Aparicio. Contradiction removal semantics with explicit negation. In *Proceedings, Applied Logics Conference*, 1992.

[61] L. M. Pereira, J. J. Alferes, and J. N. Aparicio. Contradiction removal within well founded semantics. In *Proceedings, 1st. Intl. Workshop on Logic Programming and Nonmonotonic reasoning*. MIT Press, July, 1991.

[62] S. G. Pimentel and W. L. Rodi. Belief revision and paraconsistency in a logic programming framework. In *Proceedings, 1st. Intl. Workshop on Logic Programming and Nonmonotonic reasoning*. MIT Press, July, 1991.

[63] G. Priest. Logic of paradox. *Journal of Philosophical Logic*, 8:219–241, 1979.

[64] G. Priest. A dialectical account of the growth of science. unpublished typescript, 1980.

[65] G. Priest. Sense, entailment and modus ponens. *Journal of Philosophical Logic*, 9:415–435, 1980.

[66] G. Priest and R. Routley. Applications of paraconsistent logic. In G. Priest, R. Routley, and J. Norman, editors, *Paraconsistent Logic: Essays on the Inconsistent*, chapter 13, pages 367–392. Philosophia Verlag, Munchen, 1989.

[67] G. Priest and R. Routley. Systems of paraconsistent logic. In G. Priest, R. Routley, and J. Norman, editors, *Paraconsistent Logic: Essays on the Inconsistent*, chapter 5, pages 151–185. Philosophia Verlag, Munchen, 1989.

[68] G. Priest, R. Routley, and Jean Norman. *Paraconsistent Logic: Essays on the Inconsistent*. Philosophia Verlag, Munchen, 1989.

[69] H. Przymusinska and T. C. Przymusinski. Semantic issues in deductive databases and logic programs. In A. Banerji, editor, *Sourcebook on the Formal Approaches in Artificial Intelligence*, pages 321–367. North-Holland, Amsterdam, 1990.

[70] T. Przymusinski. Every logic program has a natural stratification and an iterated fixed point model. In *Proceedings of the 8th ACM Symposium on principles of database systems*, pages 22–33, 1989.

[71] T. Przymusinski. Well-founded semantics coincides with three-valued stable semantics. *Fundamenta Informaticae*, 1989.

[72] Teodor C. Przymusinski. Extended stable semantics for normal and disjunctive programs. In *Proc. of the 7th. Intl. Conference on Logic Programming*, pages 459–477, 1990.

[73] R. Reiter. On closed-world databases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, New York, 1978.

[74] R. Reiter. A logic of default reasoning. *Artificial Inteligence*, 13:81–132, 1980.

[75] N. Rescher and R. Brandom. *The Logic of Inconsistency*. Billing & Sons, Ltd., Oxford, UK, 1980.

[76] R. Routley, ℜ. K. Meyer, V. Plumwood, and R. T. Brady. *Relevant Logics and Their Rivals*. Ridgeview, Atascadero, CA, 1982.

[77] R. Routley and V. Routley. Semantics of first degree entailment. *Nous*, 6:335–359, 1972.

[78] R. Routley and V. Routley. The role of inconsistent and incomplete theories in the logic of belief. *Communication and Cognition*, 8:185–235, 1975.

[79] 1872-1970 Russell, Bertrand. *Logic and knowledge; essays, 1901-1950*. G. Allen/Unwin, 1956.

[80] B. Russell. A critical exposition of the philosophy of leibniz, 1900.

[81] Ch. Sakama. Extended wellfounded semantics for paraconsistent logic programs. In *Proceedings, International Conference on Fifth Generation Computer Systems, 1992, ICOT*, pages 592–599, 1992.

[82] Dana Scott. Outline of a mathematical theory of computation. In *Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems*, pages 169–176, 1970.

[83] Dana Scott. Continuous lattices: Toposes, algebraic geometry and logic. *Springer Lecture Notes in Mathematics*, 274:97–136, 1972.

[84] Dana Scott. Models for various type-free calculi. In Suppes, Henkin, Juja, and Moisil, editors, *Proceedings of the Fourth International Congress for Logic, Methodology, and the Philosophy of Science, Bucharest, 1971*. North-Holland, 1973.

[85] L. A. Stein. Skeptical inheritance: Computing the intersection of credulous extensions. In *Proc. of Intl. Joint Conference in AI*, pages 1153–1158. Morgan Kaufmann, 1989.

[86] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math*, 5:285–309, 1955.

[87] D. S. Touretzky, J. F. Horty, and R. H. Thomason. A clash of intuitions: The current state of nonmonotonic multiple inheritance. In *Proc. of IJCAI-87*, pages 476–482, 1987.

[88] N. A. Vasil'ev. On particular propositions, the triangle of oppositions, and the law of excluded fourth. Uceniezapiski Kanzanskogo Universitieta, 1911.

[89] A. Visser. Four valued semantics and the liar. *Journal of Philosophical Logic*, 13:181–212, 1984.

[90] Gerd Wagner. The two sources of nonmonotonicity in vivid logic - inconsistency handling and weak falsity. In *Proc. of GMD Workshop on Nonmonotonic Reasoning, Bonn*, 1989.

[91] Gerd Wagner. A database needs two kinds of negation. In *Proc. of the 3rd. Intl. Symposium on Mathematical Fundamentals of database and Knowledge Base Systems*. Springer-Verlag, 1991.

[92] Gerd Wagner. Ex contradictione nihil sequitir. In *Proc. of Intl. Joint Conference in AI*, pages 538–543. MIT Press, 1991.

[93] Gerd Wagner. Reasoning with inconsistency in extended deductive databases. In *Proc. of the 2nd. Intl. Workshop of Logic Programming and Nonmonotonic Reasoning, Lisbon, Portugal*, pages 300–315. MIT Press, 1993.

[94] J. You and R. Cartwright. Tractable argumentation semantics via iterative belief revision. In Proc. International Logic Programming Symposium, pages 239–253. MIT Press, 1994.

[95] J. You and S. Ghosh. Introspective paraconsistent logic programs.