Journal of Applied Non-Classical Logics
Vol. 00, No. 00, Month 201X, 1–19

RESEARCH ARTICLE

Stable Reasoning

Pedro Cabalar[a], David Pearce[b]* and Agustín Valverde[c]

[a] University of Corunna (Spain)
cabalar@udc.es

[b] Universidad Politécnica de Madrid, Spain,
david.pearce@upm.es

[c] Universidad de Málaga, Spain,
a_valverde@ctima.uma.es

(*Received 00 Month 201X; final version received 00 Month 201X*)

We give an account of stable reasoning, a recent and novel approach to problem solving from a formal, logical point of view. We describe the underlying logic of stable reasoning and illustrate how it is used to model different domains and solve practical reasoning problems. We discuss some of the main differences with respect to reasoning in classical logic and we examine an ongoing research programme for the rational reconstruction of human knowledge that may be considered a successor to the logical empiricists' programme of the mid-20th Century.

1. Introduction

1.1 *Logic and the Classical Paradigm*

For most of the 20th Century logic as a scientific discipline was dominated by the paradigm of Frege and Hilbert and the aim of providing mathematics with a secure logical foundation. This challenging goal continued to dominate research programmes in logic long after Gödel's incompleteness theorems showed that the initial expectations were unreachable. Above all the Frege-Hilbert paradigm was and still is based on a standardised, classical conception of logic, on the axiomatic method as the basis for reconstructing mathematical knowledge and on the method of deduction as the central element in logical theorem proving.

The paradigm of Frege and Hilbert was challenged by various critics throughout its lifetime but only in recent years has it been eclipsed by more progressive research programmes within computational logic. A vibrant and searing critique of the Frege-Hilbert paradigm was published in 1998 by the Italian logician, Carlo Cellucci in his book *Le Ragioni della Logica* (Cellucci, 1998). Cellucci calls the standard paradigm *mathematical logic* and discusses at length many of its significant features. Besides its emphasis on providing a secure foundation for mathematics, key features include the prominence of the axiomatic method, the idea of theories as embodying certain truths and theorems as logical deductions from these. Cellucci questions many of the assumptions of the paradigm, especially the manner in which it focuses on closed conceptual systems and problems of justification, while paying almost no attention to problems of (mathematical) discovery that may involve hypothetical reasoning, induction,

This work has been partially supported by the Spanish project TIN2013-42149-P and GPC 2016/035 (Xunta de Galicia, Spain).
*Corresponding author: david.pearce@upm.es

	Classical Approach	**Stable Reasoning**
$\neg P(x) =$	$P(x)$ is **false**	$P(x)$ cannot be proved
No info on $P(x)$	**Free** (all possible models)	$\neg P(x)$ (closed-world assumption)
Predicates are	**Extensional**	**Intensional** (allows inductive definitions)
Inference	**Monotonic**	**Non-monotonic** (allows default reasoning)
Problem solving	**Predicate** based 1 solution = 1 tuple for $P(x)$	**Model** based 1 solution = 1 model for $P(x)$

Figure 1. Comparison between classical and stable reasoning.

abduction, analogy, heuristics and other methods. As a contrast to the axiomatic method Cellucci devotes much attention to describing and motivating the *analytic* method in mathematical discovery and problem solving[1] and justifying the importance of treating open systems and fallible reasoning.

While Cellucci doesn't offer a detailed description of modern *computational logic*, it is evident that some of the positive features of the analytic method can be found in computational approaches to logic and logic-based programming languages. He himself cites with approval six different features of Prolog that mirror aspects of the analytic method and reasoning with open systems (while at the same time noting that there are other important features of the analytic method that are not captured in Prolog). Since the publication of (Cellucci, 1998) computational logic has made many advances. One of them has been the elaboration of a new approach to logic-based programming known as *answer set programming* or ASP and the development of its underlying logical paradigm which we will call *stable reasoning*. Unlike Prolog, this is not a fully-fledged programming language, but rather a general approach to logic-based problem solving that can also be efficiently implemented in (answer set) solvers.

We cannot claim that stable reasoning currently satisfies all the requirements proposed by Cellucci for constituting a new and wholly adequate paradigm for logic. For one thing, the focus in (Cellucci, 1998) is on mathematics and mathematical problem-solving and discovery, while stable reasoning has a broad range of applications to many areas of inquiry. However, we do suggest that stable reasoning is:

(1) distinctly different from the Frege-Hilbert paradigm of mathematic logic;
(2) close to the analytic method, sharing significant points in problem-solving;
(3) able to embrace various aspects and methods of discovery;
(4) able to deal with dynamical and open systems.

In this paper we will discuss stable reasoning, focussing mainly on items 1 and 2 above, while mentioning 3 and 4 briefly towards the end.

[1] Closely associated with Plato and other classical scholars.

1.2 *Stable Reasoning*

Stable reasoning is a recent and novel approach to problem solving from a formal, logical point of view. An important difference compared to reasoning based on classical logic is that stable reasoning can take account of default assumptions and conclusions that follow from them (see Figure 1). It is not based on two-valued classical logic, since this does not allow for the distinction between certain truth and truth-by-default or the kind of truth that can be assumed when there is no evidence to the contrary. To account for defaults in the setting of classical logic usually special syntactic devices are employed or one has to distinguish between different kinds of inference, some defeasible others not. Stable reasoning does not require any special inference rules or syntactic devices because it is based on a many-valued logic where precisely the distinction between certain truth and truth-by-default can be made. There is just one basic kind of negation and one kind of inference.

There is another important difference compared to the axiomatic tradition of Frege and Hilbert that dominated the methodology of formal logic for much of the 20th Century. Stable reasoning is closer to what might be termed a *problem-solving* approach to formal reasoning. In the axiomatic tradition, a mathematical or empirical domain is formalised by introducing a language or vocabulary and a set of sentences (axioms) of the language intending to capture once and for all the entirety of knowledge governing that domain. Mathematical theorems are inferred through logical deduction from the axioms. Predictions or explanations from empirical theories are also deduced from their axioms once the initial conditions of a system are specified. R6

By contrast, stable reasoning is problem-driven. One wishes to find a solution or several possible solutions to a certain problem. One describes the problem domain by specifying the entities and relations that govern the domain. This description need not be complete nor need it capture the entirety of knowledge once and for all, but merely offer hypotheses sufficient to produce adequate answers or solutions. Some descriptions may indeed be robust and reusable, others may be fragile and of temporary validity/use. As in the classical, axiomatic case, answers will be produced once certain facts or initial conditions are specified. These answers or solutions are in general model-based. They are produced not by means of logical deduction or inference from axioms plus initial conditions, but rather by computing a model or state of affairs that embodies the solution in some obvious way. The table in Figure 1 summarises the most prominent differences between stable and classical reasoning.

To illustrate the difference between the two orientations, consider the following well-known graph-theoretical problem.

Example 1 (Hamiltonian cycles)**.** *A* Hamiltonian cycle *of a graph G is a cyclic path that traverses each node in G exactly once. The same graph may have different Hamiltonian cycles or none at all.* □

Suppose we want to obtain the Hamiltonian cycles of graph G in Figure 2. To this aim we must decide:

(a) how to represent the graph;
(b) how to represent the cycles;
(c) the type of logical reasoning task to obtain the desired solution.

Regarding (a), the obvious solution in Predicate Calculus is just using a pair of predicates, say $Node(x)$ and $Edge(x, y)$, to respectively describe the nodes and edges of G. Still, even at this elementary level, a first difference between the classical approach and stable reasoning already

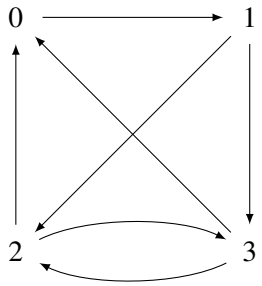

Figure 2. A simple graph with a pair of Hamiltonian cycles.

arises. In classical logic, if we just list the set of ground atoms:

$$\begin{aligned} &Node(0), \quad Node(1), \quad Node(2), \quad Node(3), \\ &Edge(0,1), \quad Edge(1,2), \quad Edge(1,3), \quad Edge(2,0), \\ &Edge(2,3), \quad Edge(3,2), \quad Edge(3,0). \end{aligned} \tag{1}$$

this will leave free the possibility for other objects of becoming nodes or edges – we would have models where, for instance, $Edge(0,2)$ could become true. In order to precisely capture the nodes and edges we should actually use instead the pair of formulas[1]:

$$Node(x) \leftrightarrow x = 1 \vee x = 2 \vee x = 3 \vee x = 4 \tag{2}$$

$$\begin{aligned} Edge(x,y) \leftrightarrow\ & (x = 0 \wedge y = 1) \vee (x = 1 \wedge y = 2) \\ & \vee (x = 1 \wedge y = 3) \vee (x = 2 \wedge y = 0) \\ & \vee (x = 2 \wedge y = 3) \vee (x = 3 \wedge y = 2) \vee (x = 3 \wedge y = 0) \end{aligned} \tag{3}$$

In the stable reasoning approach, however, the set of facts (1) would suffice because predicates are subject to the so-called *Closed World Assumption* (CWA). Informally speaking, this means that anything not explicitly stated will be false by default. We could, of course, still use the stronger version (2) $\wedge$ (3) for stable reasoning, but the set of facts (1) is more flexible in the sense that new nodes or edges can be included by the simple *addition* of new formulas, rather than manipulating the existing ones (this feature is often referred as *elaboration tolerance* (McCarthy, 1998)). Predicates like $Node(x)$ or $Edge(x,y)$ whose extensions are defined by enumerating their lists of (true) atoms conform what is usually called the *extensional database* (a term inherited from Database theory) as opposed to those described by additional conditional formulas (or *rules* in Logic Programming) which receive the name of *intensional predicates*.

R7

Another important feature about CWA is that, if necessary, we can remove this assumption in a selective way for any formula ϕ or, in particular, any atom $P(\overline{x})$. To this aim, it just suffices with adding an axiom like:

$$P(\overline{x}) \vee \neg P(\overline{x}) \tag{4}$$

Note that although (4) has the form of a classical tautology (the so-called *excluded middle axiom*), it is not a tautology for stable reasoning, since negation here has a different meaning. As said before, the effect of (4) is that CWA for predicate P is removed, so it behaves in a "classical" way. For instance, adding the axiom $Node(x) \vee \neg Node(x)$ would remove $Node$ to be false by default.

[1] We assume that free variables, like x, y here, are implicitly universally quantified.

Let us move now to consider problem (b), that is, how to represent Hamiltonian cycles. As any of them visits all nodes in the graph, we obviously must refer to the set of edges involved in the cycle to differentiate one from each other. In classical logic, we would typically represent sets of edges using some additional notation: assume, for instance, that standard set terms are allowed. Then, we would use some predicate, say $HamCycle(s)$, to represent that the set s of edges constitutes a Hamiltonian cycle for the graph, including a hypothetical set of axioms $\Gamma(HAM_G)$ that includes the graph description (2) $\wedge$ (3) and the meaning of predicate $HamCycle$. Under the axiomatic method, our problem (c) would then reduce to decide for which sets s of edges we can derive $\Gamma(HAM_G) \vdash HamCycle(s)$. For instance, in our example, we should conclude that this holds for $s = \{(0,1),(1,3),(3,2),(2,0)\}$ and for $s = \{(0,1),(1,2),(2,3),(3,0)\}$. It is worth to notice that, as these two facts for predicate $HamCycle(s)$ are derived as theorems, they will be simultaneously true in *all models* of $\Gamma(HAM_G)$. In the general case, the extension of $HamCycle$ would contain the whole set of Hamiltonian cycles in *any* model of $\Gamma(HAM_G)$.

Under the stable reasoning approach, tasks (b) and (c) become model-oriented, rather than theorem-based. One of the main features of this methodology is that we identify each solution of the original problem with (a distinguished part of) each model of our logical representation. For instance, in our example, rather than collecting all cycles in a single predicate $HamCycle(s)$, our purpose is obtaining a *different model* per each possible Hamiltonian cycle of the original graph. Using this approach, we need not dealing with sets any more: we can just collect the edges *in* the current cycle (that is, the cycle represented inside the current model) as the extent of some predicate, call it $In(x,y)$. Thus, if we call again $\Gamma(HAM_G)$ to our logical representation of the problem, we are interested in obtaining the different models M such that $M \models \Gamma(HAM_G)$ so that each Hamiltonian cycle can be directly obtained by selecting some relevant information in each obtained model M. In our example, we should obtain a pair of models M_1 and M_2 such that:

$$M_1 \models In(0,1) \wedge In(1,3) \wedge In(3,2) \wedge In(2,0)$$
$$M_2 \models In(0,1) \wedge In(1,2) \wedge In(2,3) \wedge In(3,0)$$

and no other atoms for $In(x,y)$ hold in each model.

Let us consider now how to specify $\Gamma(HAM_G)$ under the stable reasoning approach. Typically, we would first consider models for all possible subsets of edges, and then include additional formulas that rule out the ones that do not correspond to Hamiltonian cycles. In a first attempt, we could include:

$$Edge(x,y) \rightarrow \big(In(x,y) \vee \neg In(x,y)\big) \tag{5}$$
$$In(x,y) \wedge In(x,z) \rightarrow y = z \tag{6}$$
$$In(y,x) \wedge In(z,x) \rightarrow y = z \tag{7}$$

together with a graph description, that is, an extensional database like (1).

As a first important remark, note that the consequent $In(x,y) \vee \neg In(x,y)$ in (5) has the form of an excluded middle formula like (4). As we explained before, this means that predicate $In(x,y)$ will not be subject to CWA, provided that x,y is a pair of nodes forming an edge. As a result, the effect of (5) is that we would have a model per each possible subset of edges in the graph. Formula (6) (resp. (7)) specifies that we never pick two different outgoing (resp. incoming) edges for a given node x. These two formulas can also be represented by using $\bot$ in

the consequent (formulas like these receive the name of *constraints*):

$$In(x,y) \wedge In(x,z) \wedge y \neq z \rightarrow \bot$$
$$In(y,x) \wedge In(z,x) \wedge y \neq z \rightarrow \bot$$

This first attempt (5)-(7), however, is not enough to capture Hamiltonian cycles yet. We can still get disconnected groups of edges like, for instance, $\{(0,1),(2,3),(3,2)\}$ or even nodes that are not connected at all. To rule out these cases, we should further specify that we have a path connecting any pair of nodes: since (6)-(7) already guarantee linearity, the only option then is forming a cycle. Curiously, this property (reachability in an arbitrary graph) is a well-known example of a problem that *cannot be represented* in classical First Order Logic. So, in fact, if we tried to represent Hamiltonian cycles using the classical axiomatic method, $\Gamma(HAM_G)$ should be represented in Second Order Logic. Under stable reasoning, however, we can easily capture reachability with an auxiliary predicate $Reach(x,y)$ defined with the pair of formulas:

$$In(x,y) \rightarrow Reach(x,y) \tag{8}$$
$$In(x,z) \wedge Reach(z,y) \rightarrow Reach(x,y) \tag{9}$$

Since $Reach(x,y)$ is subject to CWA, these two formulas behave as an inductive definition, as those frequently used in Mathematics. In other words, $Reach(x,y)$ is true if: (1) we have taken the edge $In(x,y)$; or (2) we took an edge to some z and we can reach y from that z, $Reach(z,y)$. Otherwise, $Reach(x,y)$ will be false, due to CWA. A predicate like this, whose extension is determined by some rules like those above in combination with CWA, receives the name of *intensional* predicate.

Now, to guarantee that "there is a path connecting any pair of nodes" we must be slightly careful. If we just write a direct translation of the previous sentence:

$$Node(x) \wedge Node(y) \rightarrow Reach(x,y)$$

this formula would not rule out undesired solutions but would rather become a third option for defining $Reach(x,y)$ facts. In particular, we would actually get that any pair of nodes would always be connected, regardless the edges we had in the graph. Instead, we must use the negated formula:

$$\neg\exists x,y(\ Node(x) \wedge Node(y) \wedge \neg Reach(x,y)\ )$$

which, though classically equivalent, has a quite different meaning under stable reasoning, acting as the *constraint* "forbid pairs of disconnected nodes," also representable as the implication:

$$Node(x) \wedge Node(y) \wedge \neg Reach(x,y) \rightarrow \bot \tag{10}$$

Proposition 1. *Let* $\Gamma(G)$ *be the set of formulas* (5)-(10) *plus an extensional database for graph* G *like* (1)*. Then,* M *is a stable model of theory* $\Gamma(G)$ *iff* $\{(x,y) \mid M \models In(x,y)\}$ *is a Hamiltonian cycle for* G*.* □

2. A logic from first principles

Stable reasoning has very simple logical underpinnings. Its base logic can be constructed from first principles in a few easy steps and with only a few fundamental assumptions. The first idea

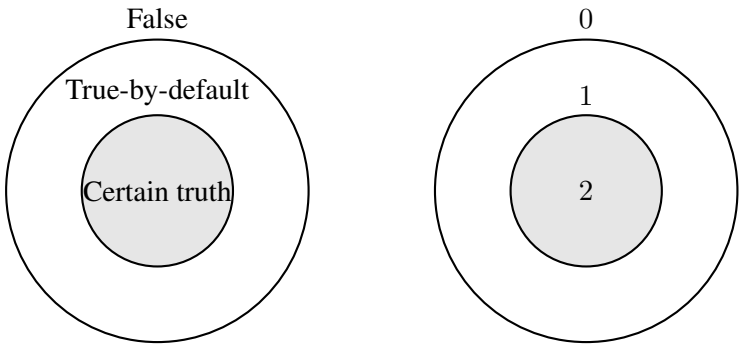

Figure 3. Graphical representation of truth types in stable reasoning.

$\wedge$	0	1	2
0	0	0	0
1	0	1	1
2	0	1	2

$\vee$	0	1	2
0	0	1	2
1	1	1	2
2	2	2	2

$\rightarrow$	0	1	2
0	2	2	2
1	0	2	2
2	0	1	2

	$\neg$
0	2
1	0
2	0

Figure 4. Truth tables for here-and-there connectives.

is that formulas may be true or false, but that there are two kinds of truth: certain truth and truth-by-default. We can picture this as shown on the left in Figure 3

Suppose that the disk represents the collection of true formulas; while outside the outer circle formulas are false. True formulas come in two kinds: those that are only weakly true or more particularly *true by default* lie within the unshaded, outer part of the disk, those lying within the shaded, inner circle are *certain*, or *true in the strong sense*. Truth in the weak or general sense covers both certain truth and truth by default, so the outer circle contains all the formulas of the disk, both inner and outer parts.

We can assign numbers to these different semantic properties, say 0 for falsity and 2 for certain truth. We assign the value 1 to formulas that lie within the outer but not the inner circle: they are only true in the weaker but not the stronger sense. The picture is shown on the right in Figure 3

Let us say that our base logic comprises, in the propositional case, the usual set of logical connectives $\{\wedge, \vee, \rightarrow, \neg\}$, standing for conjunction, disjunction, implication and negation. We may also make use of the falsum constant, $\perp$. There is an infinite set $Prop$ of propositional atoms from which formulas are constructed in the usual way, as in classical or intuitionistic logic. Let $p, q, r, \ldots$ stand for atomic propositions. As prescribed, the semantics is given by the three truth-values, 2, 1 and 0.

In the picture on the left hand side of Figure 5, p is certain, q and s are true by default and r and t are false. The values for complex formulas are assigned according to the usual meaning of connectives. In the picture $q \wedge p$ takes the value 1, since only p is certain, while q is not. On the other hand the fact that p is certain is sufficient for assigning $p \vee q$ and even $p \vee r$ the value 2. Since r is false, also $p \wedge r$ and $q \wedge r$ must be false, while $q \vee r$ takes the value 1.

Following this reasoning we can build the complete truth tables for conjunction and disjunction; they are similar to those of other well-known logics.

Some care has to be taken with the values of formulas of form $A \rightarrow B$. In the picture we would expect both $p \rightarrow r$ and $q \rightarrow r$ to be false since in both cases we have a true antecedent and false consequent. Notice that although q is not certain, we cannot put $q \rightarrow r$ in the inner circle since that would make it strongly true and yet false at the same time. On the other hand, since r is false, evidently both $r \rightarrow q$, $r \rightarrow p$ and $r \rightarrow t$ are true, and even in the strong sense. Since p is certain, it is also true in the general sense and therefore $p \rightarrow q$ must be at least weakly

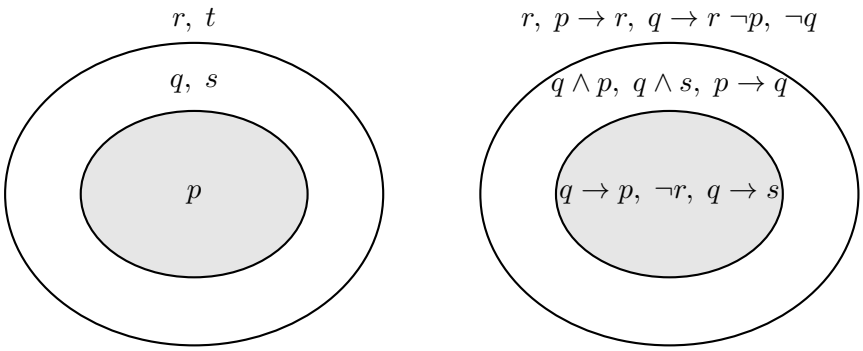

Figure 5. Atomic and compound formulas

true. But it is not certain, since q is not. By contrast, since p is certain, $q \to p$ is also. Lastly, consider the case of two formulas q, s that lie only in the outer circle. Since neither atom is certain, but neither is false, the implications $q \to s$ and $s \to q$ can safely be placed in the inner circle of certainty.

The complete truth matrix for implication is given in the figure. The case of negation is straightforward. Negating a false proposition such as r produces a certain proposition $\neg r$; while negating either kind of true formula produces a false one: in the picture both $\neg p$ and $\neg q$ will be false. Equivalently we can regard negation to be definable, by $\neg p := p \to \bot$, where $\bot$ takes the constant value 0. Then the table for negation is derivable from that for implication. As we can see, negation is not involutive, that is, $\neg\neg p$ is not equivalent to p. In fact, $\neg\neg p$ is 2 when p is different from 0, and so, it captures the meaning "p is true by default," being weaker than the formula p, which means "certain truth."

R2

Our base logic is by no means new. It was first introduced by Heyting (1930) in his study of intuitionistic logic. Shortly after it reappeared in Gödel (1932) showing that intuitionistic logic is not tabular (finite-valued). Heyting provided the truth matrices while subsequently Łukasiewicz (1941) studied the logic in greater depth and gave the first axiomatisation based on the axioms and rules of the intuitionistic calculus extended by the addition of a weaker form of Peirce's Law, viz:

$$(\neg\alpha \to \beta) \to (((\beta \to \alpha) \to \beta) \to \beta). \tag{11}$$

Subsequently, the logic was studied by Smetanich (1960) and Umezawa (1959) who gave an alternative axiom to that of Łukasiewicz:

$$\alpha \vee \neg\beta \vee (\alpha \to \beta)$$

The completeness of this system was then proved by Hosoi (1966).

Although Heyting made use of the logic as a technical device, it was clear from the start that it was of interest not only from a purely formal point of view. Łukasiewicz made a detailed comparison with his own 3-valued logic and Heyting already provided a natural interpretation of the third truth-value, claiming that it applies to a correct proposition that cannot be false but whose correctness cannot be proved. If we re-phrase this in terms of truth, then the interpretation 'true-by-default' is a natural one since it conveys the idea of a judgement that is accepted although not formally derivable in the system at hand.[1]

Our logic is known under a variety of names, we prefer to call it the logic of *here-and-there*, in symbols **HT**, whose etymology is easily explained by looking at an alternative way to describe

[1] Strictly speaking we should say that the base logic gives a monotonic approximation to reasoning by default. There is an additional aspect to defaults that emerges in the nonmonotonic extension of the logic.

the semantics. It is well-known that intuitionistic logic is complete for possible world models, that is triples of the form $\langle W, \leq, I\rangle$, where W is a non-empty set (of points, states or worlds), $\leq$ is a partial ordering on W, and I is an interpretation function assigning a set of (verified) atoms to each $w \in W$, such that $I(x) \subseteq I(y)$ whenever $x \leq y$. The logic **HT** is also complete for a class of such models, but of an especially simple form: W comprises just two points, say h ('here') and t ('there'), with $h \leq t$. It follows that we can represent a model simply as an ordered pair $\langle H, T\rangle$ of sets of atoms, where $I(h) = H$ and $I(t) = T$. Evidently, we always have $H \subseteq T$. These are then *here-and-there* models. Applying the usual semantics to evaluate formulas at worlds one can easily verify the correspondence to our earlier truth-tables. Note that in a here-and-there model $\langle H, T\rangle$, H represents the certain atoms having value 2, while T represents the non-false atoms (of value 1 or 2). Those in $T \setminus H$ are true-by-default, ie take the value 1. The complement of T in $Prop$ is the set of atoms that are false in the model, ie those corresponding to the value 0 in our truth tables. We adopt the usual convention and say that a formula φ is *satisfied* or *holds* in a model $\langle H, T\rangle$, in symbols $\langle H, T\rangle \models \varphi$, if it is strongly true or certain in the model, i.e. is satisfied at the world h.

As a simple example, consider how to evaluate an implication $p \rightarrow q$ in a model $\langle H, T\rangle$, for atomic p, q. By the possible world semantics, this formula is true at t if $q \in T$ or $p \notin T$. In terms of truth-values it means that $p \rightarrow q$ takes a value of at least 1 if $q \in T$ no matter what value is given to p. However in the case that $p \notin T$, then also $p \notin H$. This means that $p \rightarrow q$ is true also at h and so the value of $p \rightarrow q$ is then 2.

Much is known about **HT** and its properties. It is one of 7 superintuitionistic (SI) logics having the interpolation property, (Maksimova, 1977), and it is also the strongest SI-logic properly contained in classical logic, all other SI-logics being properly contained in it.

3. Basic stable reasoning

The core logic of stable reasoning is based on **HT** but is actually a non-monotonic extension of it. It can be characterised in terms of a preferential entailment relation in the sense of Shoham (1988) defined on here-and-there models. The preference relation can be explained by considering some features of defaults.

When we build up a partial description of the world or of our problem domain we specify a set of formulas that we suppose, at least hypothetically, to be true in the certain sense. Call this our 'theory'. When we then consider the three-valued **HT**-models of this theory (assuming it is consistent), they in turn generally admit formulas that are true only in the weak or default sense. We typically have models $\langle H, T\rangle$ where $H \subset T$ and then some formulas will be true only in the default sense, for instance all the propositions in $T \setminus H$. Equally, by consistency, we will also have models of the form $\langle T, T\rangle$ where all truths are certain. Our preference condition is to select just those models $\langle T, T\rangle$ where all truths are certain *and our theory does not admit any model with uncertainty whose true atoms are exactly* T; in other words there is no model of our theory of the form $\langle H, T\rangle$, where $H \subset T$. In this sense our theory justifies the choice of T by not accepting any model where T forms the set of truths in the general sense but some of them, $T \setminus H$, are only true in the weak sense.

So here is our definition in full. A model $\langle H, T\rangle$ of a theory $\mathcal{T}$ is said to be an *equilibrium* model of $\mathcal{T}$, if (i) $H = T$ and (ii) for any H such that $H \subset T$, $\langle H, T\rangle \not\models \mathcal{T}$. The term equilibrium model derives from (Pearce, 1997), and the ensuing logic associated with this is equilibrium logic. However there is complete agreement between equilibrium models and the stable models of logic programs as defined by Gelfond and Lifschitz (1988). That is to say, if the formulas of a theory have precisely the shape of rules of logic programs, then a set of

atoms T is a stable model of the theory if and only if $\langle T, T\rangle$ is an equilibrium model of it.[1] For this reason, equilibrium logic can serve as a foundation for stable reasoning.[2] Though they are defined differently, we often use the terms stable model, answer set and equilibrium model interchangeably.

4. Implementation of stable reasoning

Just as classical deduction is implemented in automated provers such as Prover9 or SAT-solvers, so stable reasoning has been implemented in various systems, collectively known as *answer set solvers*. These have obtained a fairly high degree of efficiency and sophistication and can be used to model and solve real-world problems in domains such as software verification, security and configuration management, model checking, agent technologies, constraint satisfaction, reasoning for the semantic web, software synthesis from specifications, knowledge representation, data and information integration, planning and diagnosis. The inputs to answer set solvers are called *answer set programs*, whereas the branch of computer science and programming dealing with these is called *answer set programming*, or ASP for short. A closely related domain is that of DATALOG and deductive database systems. Indeed data and information management is one of the principal application areas of ASP, and one that currently enjoys some commercial success.

While they are successful in supporting real-world problem solving, ASP systems do not implement deduction in a formal logic. They are not designed to deduce theorems or prove the correctness of logical inferences, even though the solutions they compute can be precisely understood in terms of formal models of a logical system of deduction. This correspondence to logic however was a more recent discovery and did not directly guide the initial ASP implementations. While in traditional logic programming, systems of logical inference came first and computer programming applications came after, in ASP this order was reversed. The theory preceded the computer implementations that in turn preceded the logic. However, nowadays the logic exerts a growing influence on the development and comprehension of new systems, especially extensions of the initial ASP family of languages. It is also fundamental to understanding stable reasoning from a foundational point of view.

5. Practical examples

Let us now consider some additional examples of typical commonsense reasoning tasks that can be easily represented under the stable reasoning approach. We also show in some cases how these examples can be implemented using existing solvers. For instance, Figure 6 shows a possible implementation of the Hamiltonian cycles problem for graph in Figure 2, using the input language of the Answer Set Programming (ASP) solver `clingo`[1]. The column in the right shows the correspondence to each formula in $\Gamma(HAM_G)$. The translation of our example theory into ASP language follows some standard syntactic conventions from Logic Programming and, in this case, is quite straightforward. Variables begin with uppercase letters and predicates

R3

[1] In logical notation the rules of a (disjunctive) logic program have the form

$$b_1 \wedge \ldots \wedge b_m \wedge \neg b_{m+1} \wedge \ldots \wedge \neg b_n \rightarrow a_1 \vee a_2 \vee \ldots \vee a_k$$

where the a_i, b_j are atoms. If $k = 1$ everywhere the program is said to be *normal*.

[2] Since the late 1990s the stable model semantics has been systematically extended to embrace wider classes of formulas, more recently including arbitrary propositional and even first-order theories. All the widely accepted extensions have coincided with equilibrium logic.

[1] Available at `http://potassco.org`. In fact, the example is still syntactically correct in the current standard input language (ASP core 2.0) except for the use of negation in the head.

```
node(0). node(1). node(2). node(3).
edge(0,1). edge(1,2). edge(1,3).                    (1)
edge(2,0). edge(2,3). edge(3,2). edge(3,0).

in(X,Y) | not in(X,Y) :- edge(X,Y).                 (5)

:- in(X,Y), in(X,Z), Y!=Z.                          (6)
:- in(Y,X), in(Z,X), Y!=Z.                          (7)

reach(X,Y) :- in(X,Y).                              (8)
reach(X,Y) :- in(X,Z), reach(Z,Y).                  (9)

:- not reach(X,Y), node(X), node(Y).                (10)
```

Figure 6. Hamiltonian cycle representation $\Gamma(HAM_G)$ for the graph in Figure 2 written for `clingo`.

with lowercase letters. Implications are called *rules* and their direction is reversed, so that α :- β stands for $\beta \rightarrow \alpha$, being α and β respectively called the *head* and the *body*. When the head is $\alpha = \bot$ it is just omitted. Negation $\neg$ is written `not` and all rules are ended by a full stop. Finally, '|' represents disjunction and commas represent conjunctions. R3

A fundamental property of stable reasoning we have not exploited yet in the previous example is the use of default negation for non-monotonic reasoning (NMR). To illustrate this concept, consider the following classical example in the NMR literature. We want to capture the default "birds typically fly" and the exception "penguins are birds but do not fly." These two assertions can be respectively encoded as:

$$Bird(x) \wedge \neg CannotFly(x) \rightarrow Flies(x) \tag{12}$$

$$Penguin(x) \rightarrow Bird(x) \wedge CannotFly(x) \tag{13}$$

Suppose we have theory Γ' containing the two formulas above plus the ground atom $Bird(Tweety)$. As there is no evidence that $Tweety$ is a penguin, $\neg Penguin(Tweety)$ is derived by CWA. This falsifies the antecedent of (13), and so no evidence can be obtained from (13) on $CannotFly(Tweety)$. In fact, no evidence on $CannotFly(x)$ can be obtained from (12) either. This is because implication in stable reasoning acquires a kind of directionality. In particular, (12) should be read as a *definition rule* for $Flies(x)$, saying that it will hold when x is a bird for which we have no evidence on $CannotFly(x)$. As a result, we conclude $\neg CannotFly(Tweety)$ by CWA and then, $Flies(Tweety)$ from (12).

This simple example became a challenge for NMR approaches because a wrong predicate minimisation policy could easily lead to models where $\neg Flies(Tweety)$ was decided first and then $CannotFly(Tweety)$ was derived by applying (12) as the classically equivalent formula:

$$Bird(x) \wedge \neg Flies(x) \rightarrow CannotFly(x)$$

In stable reasoning, such an equivalence does not hold[1]. In fact, the formula above has a quite different reading from (12): it states that a bird x cannot fly if we cannot find any evidence for $Flies(x)$.

To complete the example, consider now the extended theory

$$\Gamma'' = \Gamma' \cup \{Penguin(Tweety)\}.$$

[1] **HT** satisfies contraposition: $A \rightarrow B$ entails $\neg B \rightarrow \neg A$. However, by contraposition, $\neg CannotFly(x) \rightarrow Flies(x)$ entails $\neg Flies(x) \rightarrow \neg\neg CannotFly(x)$ and, as we said, the consequent of the latter is not equivalent to $CannotFly(x)$.

Since we do obtain now evidence on $CannotFly(Tweety)$ from (13), we lose the justification for $Flies(Tweety)$ we obtained before from (12). As a result, $Flies(Tweety)$ is not derived any more. This shows the non-monotonic nature of stable inference, since the addition of a new formula, $Penguin(Tweety)$, has made a previous conclusion $Flies(Tweety)$ to be retracted.

The representation of this example in ASP notation is shown below:

```
flies(X) :- bird(X), not cannotfly(X).
bird(X) :- penguin(X).
cannotfly(X) :- penguin(X).
bird(tweety).
penguin(tweety).
```

6. Stable reasoning, open systems and the analytic method

In this final section we look briefly at one of the directions in which stable reasoning has developed into a programme for the logical reconstruction of human knowledge in a practical setting. Then we return to some of the conditions suggested by Cellucci (1998) that should be fulfilled by a logical paradigm to replace the axiomatic tradition of mathematical logic. We consider briefly some of the ways in which stable reasoning conforms to these requirements.

6.1 *Gelfond's programme*

The rise of the axiomatic method and formal reasoning based on classical logic in the 20th Century was closely linked to two philosophical schools based in central Europe: the Lwow-Warsaw School of Logic and Philosophy and the logical empiricist movement of the Vienna Circle. Particularly the latter school endorsed a programme of rational reconstruction of scientific and other forms of knowledge as well as adopting the idea of explicating philosophically important concepts typically by formalising them within a logico-mathematical system. The limits of the logical empiricist programme and the growing criticism it faced from the 1960s onwards are well-known and well-documented. Especially vulnerable was the ideal to reconstruct (ultimately all of) scientific knowledge in formal languages governed by the classical laws of deduction. A pivotal point of this criticism was the claim that logic and experience are not sufficient to explain the rationality of science, catalogue its methods and reconstruct the knowledge it generates.

Ironically, the downfall of the empiricist programme came about just at a moment in time when logic was about to undergo a radical change and a major shift in its boundaries. This change was led by Hintikka in the study of propositional attitudes of knowledge and belief, by Montague in natural language processing, by McCarthy in artificial intelligence and commonsense reasoning, by Simon in learning and scientific discovery, and by a range of applications in computer science and programming.

Stable reasoning has spawned a successor to the logical empiricist programme of rational reconstruction. It is a research programme that aims to reconstruct some of the most basic forms of human knowledge and to exploit this knowledge for practical problem solving. While logical empiricist efforts were largely theoretically oriented, this programme deals with a mix of theory and practice. It combines scientific and engineering knowledge of real systems with practical human skills and abilities and commonsense reasoning. It deals with both static and dynamic domains. The scientist most closely identified with this programme is Michael Gelfond, also a co-founder of stable reasoning itself. His programme combines the physicalist language of engineering and physical systems with epistemic notions such as belief, agency and action. The new programme of rational reconstruction is much less self-conscious than its predecessor. The latter formed part of a manifesto with a clear philosophical, and sometimes political, mes-

sage. It was stated and re-stated many times. The new programme is scarcely articulated and hardly known. Nevertheless it's goals and methodology are largely clear, if sometimes buried in technical articles and lectures.[1]

Gelfond's programme for representing and reasoning about knowledge has two main objectives. First it aims to achieve an understanding of "basic commonsense notions we use to think about the world: beliefs, knowledge, defaults, causality, intentions, probability, etc., and to learn how one ought to reason about them." Secondly it aims "to understand how to build software components of agents – entities which observe and act upon an environment and direct its activity towards achieving goals." These goals shape the criteria used to evaluate and select languages for Knowledge Representation (KR). In particular he endorses four main adequacy criteria:

(1) Clarity: the logical vocabulary should have a clear and intuitive meaning.
(2) Elegance: the corresponding mathematics should be simple and elegant.
(3) Expressiveness: the KR language should suggest systematic and elaboration tolerant representations of a broad class of phenomena of natural language, including belief, knowledge, defaults, causality and others.
(4) Relevance: a large number of interesting computational problems should be reducible to reasoning about theories formulated in this language.

It is interesting to compare these criteria with the requirements that Carnap proposes for the adequate explication of concepts (Carnap, 1950). Criterion 2 is close to Carnap's requirement for the task of concept explication that "the explicatum should be as *simple* as possible." Criterion 4, on the other hand, and to a somewhat lesser extent 3, are close to Carnap's idea that an explicatum is to be a *fruitful* concept, "that is, useful for the formulation of many universal statements." Although Elaboration Tolerance is a more modern idea, one may suppose that it would also feature among the properties of fruitfulness. Carnap's suggestion that the "explicatum is to be *similar to the explicandum*" in many cases of usage does not appear explicitly in Gelfond's list. However since Gelfond's aim is to reconstruct commonsense knowledge and practical reasoning one can assume that Carnap's requirement is one he would also endorse and is somehow implicit in his programme.[1]

It is also revealing to consider some adequacy criteria for KR languages that Gelfond *rejects*. Among these are two that in the past were often considered sacrosanct in the AI community of Knowledge Representation. One is the idea that any KR language should be *supra-classical* ie. extend first-order classical logic. The other is the requirement of *efficiency* understood in a computational sense. We have seen already that the underlying logic of stable reasoning is not classical, and nor does it become so when additional features and functionalities are provided. On the other hand, the expressiveness of the basic language – a positive feature – also results in it being less efficient computationally than some other languages.[2]

6.2 *Open conceptual systems*

Let us return to the idea of open conceptual systems that we mentioned in the introduction. According to Cellucci there are many similarities between open conceptual systems and the notion of open physical systems. Here is a summary of some of the basic features of the former taken from (Cellucci, 1998) (pp. 313–315).

[1] But see specially (Gelfond, 2011) for an overview.
[1] All quotations above are from (Carnap, 1950), Introduction.
[2] Generally speaking the complexity of stable reasoning lies at the second level of the polynomial hierarchy. Nevertheless answer set solvers are relatively efficient in being able to deal with quite large amounts of data. Notice that, from a representational point of view, answer set programs are very efficient in being able to encode complex problems in a concise manner.

(1) Open conceptual systems take account of the manner in which the solution of a problem is to be arrived at.
(2) Unlike in the axiomatic approach there is no unifying idea that serves as a foundation once and for all.
(3) The unifying impulse for problem solving is data driven rather than reductionist.
(4) Open conceptual systems tackle the problem to be solved directly, from first principles. There is no *a priori* set of concepts and principles that precede the problem formulation.
(5) The rules of the game are not given at the beginning and fixed once and for all, but may be introduced and changed during the course of the game.
(6) Open systems are dialogical, since partially given information may be extended through interaction with other systems.
(7) The rules of the system give only a partial and dynamically changing representation of knowledge.

Besides these characteristics, Cellucci emphasises the ampliative nature of logical inference, as well as the need to deal with with global inconsistencies and incoherences. There are many other features of open systems and the analytic method discussed in (Cellucci, 1998) and a detailed examination would take us beyond the scope of this paper. Likewise, here we have described only some core features of stable reasoning and ASP. Many other features emerge in the practical development of systems and their application to problem solving. We conclude with a shortlist of some of the characteristics that may bear on Cellucci's challenge to develop an alternative logical paradigm.

- Stable reasoning in its basic form already deals with weak and strong exceptions to defaults. However to deal with contradictions that arise indirectly as consequences of default conclusions, an extension of ASP with consistency-restoring rules (CR-Prolog) was developed and applied by Balduccini and Gelfond (2003). This is essentially an abductive mechanism.
- Another feature of Gelfond's programme has been the aim to reason about the degrees of belief of a rational agent. This has led to a system (P–log) that combines logical and probabilistic reasoning based on ASP (Baral, Gelfond, & Rushton, 2009).
- Although in its basic form stable reasoning is highly declarative, when ASP is used in practice the problem representation takes account of the way in which the solver will successfully and economically reach a solution. Features such as cardinality and integrity constraints and, more generally, *aggregates* are employed to direct the computational mechanism and possibly enhance efficiency (Gebser, Kaminski, Kaufmann, & Schaub, 2012).
- Basic ASP already deals with some problems of temporal projection. However, to deal with a wider range of problems for dynamically changing domains, a temporal version of equilibrium logic and ASP has been developed and studied, following (Cabalar & Pérez-Vega, 2007).
- A central property of open systems is the necessity to interact with other systems. Scholars have developed different logical semantics based on ASP that facilitate this interaction. In particular, logic program rules may contain concepts whose meanings are partially determined by external data sources such as knowledge bases or ontologies (Eiter, Ianni, Schindlauer, & Tompits, 2005; Rosati, 2006). This gives rise to hybrid theories that mix different reasoning systems (e.g. monotonic and non-monotonic). Equilibrium logic can be applied to give a simple and uniform treatment of such theories (de Bruijn, Pearce, Polleres, & Valverde, 2010).
- To deal with problem solving in a dynamically evolving setting, it is important to consider the problem of updating knowledge in light of new data and knowledge discovery. This has led to the study of theory and program updates in the framework of ASP (Slota &

Leite, 2010).

6.2.0.1 Acknowledgements. Work on this paper has been partially supported by the projects FOREST (TIN2015-70266-C2), MERLOT (TIN 2013-42149-P) and Xunta de Galicia GPC 2016/035. It is a great pleasure to dedicate this paper to our friend Luis Fariñas del Cerro who has given us generous encouragement and support over a long period. After devoting many years to the advancement of logic and its applications, Luis has recently become also a major contributor to the programme of stable reasoning. We look forward to many more years of intense and fruitful cooperation.

References

Balduccini, M., & Gelfond, M. (2003). Logic programs with consistency-restoring rules. In *International symposium on logical formalization of commonsense reasoning, aaai 2003 spring symposium series* (pp. 9–18).

Baral, C., Gelfond, M., & Rushton, J. N. (2009). Probabilistic reasoning with answer sets. *TPLP*, *9*(1), 57–144.

Cabalar, P., & Pérez-Vega, G. (2007). Temporal equilibrium logic: a first approach. In *Computer aided systems theory* (pp. 241–248).

Carnap, R. (1950). *The logical foundations of probability*. University of Chicago Press.

Cellucci, C. (1998). *Le ragioni della logica*. Roma-Bari: Laterza.

de Bruijn, J., Pearce, D., Polleres, A., & Valverde, A. (2010). A semantical framework for hybrid knowledge bases. *Knowledge Information Systems*, *25*(1), 81–104.

Eiter, T., Ianni, G., Schindlauer, R., & Tompits, H. (2005). A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In *Proceedings of the nineteenth international joint conference on artificial intelligence (IJCAI-05)* (pp. 90–96).

Gebser, M., Kaminski, R., Kaufmann, B., & Schaub, T. (2012). *Answer set solving in practice*. Morgan & Claypool Publishers.

Gelfond, M. (2011). *Personal perspective on the development of logic programming based KR languages.* (Unpublished draft, available online at `http://www.depts.ttu.edu/cs/research/krlab/papers.php`)

Gödel, K. (1932). Zum intuitionistischen aussagenkalkül. *Anzeiger der Akademie der Wissenschaften Wien, mathematisch, naturwissenschaftliche Klasse*, *69*, 65–66.

Heyting, A. (1930). Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse*, 42–56.

Hosoi, T. (1966). The Axiomatization of the Intermediate Propositional Systems S_2 of Gödel. *Journal of the Faculty of Science of the University of Tokyo*, *13*(2), 183–187.

Łukasiewicz, J. (1941). Die logik und das grundlagenproblem. *Les Entreties de Zürich sur les Fondaments et la Méthode des Sciences Mathématiques*, *12*(6-9), 82–100.

Maksimova, L. (1977). Craig's interpolation theorem and amalgamable varieties. *Doklady Akademii Nauk SSSR*, *237*(6), 1281–1284.

McCarthy, J. (1998). Elaboration tolerance. In *Proc. of the 4th symposium on logical formalizations of commonsense reasoning (common sense 98)* (pp. 198–217). London, UK.

Pearce, D. (1997). A new logical characterisation of stable models and answer sets. In *Non monotonic extensions of logic programming. proc. NMELP'96. (LNAI 1216).* Springer-Verlag.

Rosati, R. (2006). DL+log: Tight integration of description logics and disjunctive datalog. In *Proceedings of the 10th international conference on principles of knowledge representation and reasoning (KR'06)* (pp. 68–78).

Slota, M., & Leite, J. (2010). On semantic update operators for answer-set programs. In *Proceeddings of the 19th european conference on artificial intelligence (ECAI 2010)* (pp. 957–962).

Smetanich, Y. S. (1960). On completeness of a propositional calculus with an additional operation of one variable (in Russian). *Trudy Moscovskogo Matematiceskogo Obscestova*, *9*, 357–372.

Umezawa, T. (1959). On intermediate many-valued logics. *Journal of the Mathematical Society of Japan*, *11*(2).

Summary of changes

First of all, we wish to thank the reviewer for the useful suggestions that have helped to improve the paper. To simplify the task of a second review, in the following sections we enumerated and answered all the review items using a code RXX (XX=item number). In most cases, the corresponding text in the paper has been colored and signaled with that same code as a side margin note.

Reviewer 1

R1 *In the Hamiltonian example, the "final" formula (between (9) and (10)) may appear unmotivated and/or artificial to the casual reader.*

We have rephrased the text, trying to be more precise. Constraint (10) expresses the constraint "forbid pairs of disconnected nodes."

For instance, one might expect some formula requiring the existence of a root for a path traversing all nodes instead of the current formula requiring a path from each node to all nodes.

The declaration of a root node is, indeed, the usual way to represent this problem in Answer Set Programming. This has the advantage of a considerable size reduction of the ground program, as we only need to fix a root, say node 1, and a monadic predicate $Reach(X)$ to assert which nodes are accessible from 1. However, this is less elaboration tolerant, as we force to describe graphs with some node named 1 or, instead, to provide some input predicate $Root(X)$ asserting which is the root node. We could even decide the root node non-deterministically, but again, all these options just complicate the representation, whose simplicity is our goal here.

R2 *In Section 2, why not provide insighful properties that hold for classical logic but fails here: e.g., the deduction theorem, and, on the contrary discuss why some non-classical properties such as the disjunction theorem (in connection with the intuitionist nature of stable reasoning) hold or fail with stable reasoning...*

David ???

Moreover, the paragraph prior to (11) discusses negation hence I would have expected a few words on the fact that negation is non-involutive in the truth-tables of Figure 4.

Done. We also added a comment explaining that $\neg\neg p$ means "true-by-default" while p represents "certain truth."

R3 *In the second paragraph of Section 5, the sentence "replacement is correct provided that $Out(x, y)$ is not used elsewhere" sounds like black magic, please explain.*

Predicate $Out(x, y)$ was needed before because we could not use negation in the rule heads in the input language of solver `DLV`. This was an auxiliary predicate and, what we meant by correctness of the replacement is that we can use both versions of the formulation indistinctly, even when included in a larger theory or context, provided that the auxiliary predicate $Out(x, y)$ is hidden, that is, is not used (actually is not a rule head) in the external, larger context.

In the current version of the paper, however, we decided to switch to the input language of solver `clingo` which already accepts negation in the rule head. In this way, the translation from formulas to programs is more natural and predicate $Out(x, y)$ is not needed.

R4 *Later in Section 5, there is an interesting point about $\neg Flies$ and $CannotFly$. Does this relate to the failure of the inference $\neg A \to B / \neg B \to A$ in the intuitionist setting?*

Thanks for this nice remark. We have added a footnote clarifying that **HT** satisfies contraposition but $\neg A \to B$ entails $\neg B \to \neg\neg A$ rather than $\neg B \to A$.

R5 *In Section 6.1. and culminating in Footnote 12, the authors seem to make what seems*

to me an overclaim: "encode complex statements in a concise way" (not exactly the authors' wording but I feel I do not betray. In view of the failure of common classical equivalences, it may be really hard upon the user to figure out how to represent a given piece of knowledge. Illustration about this are the Flies/CannotFly example, the "final" statement for the Hamiltonian cycle example, the impact of non-involutive negation on representing knowledge,... A corollary is that the process of representing knowledge may become error-prone to a more acute way (again, see the difference between writing $\neg CannotFly \rightarrow Flies$ *(correct wrt intent) and* $\neg Flies \rightarrow CannotFly$ *(incorrect)).*

David???Pedro: I'm not sure about what he is asking here. The point is not about "complex statements" but lack of elaboration tolerance of Classical Logic. There is no CWA, no way to represent induction, no way to represent defaults (frame problem, qualification problem, ramification problem, etc). Should perhaps we mention this in Section 6.1?

R6 *Typo: "for all the entirely of"* $\rightarrow$ *"for all the entirety of"*

Done

R7 *Typo: "Close World"* $\rightarrow$ *"Closed World" (I guess?)*

Yes. Corrected.

R8 *"Neverthless"*

Corrected.