# Revisiting $C_1$

Mauricio Osorio[2], José Luis Carballido[1], and Claudia Zepeda[1]

[1] Benemérita Universidad Atónoma de Puebla
[2] Universidad de las Américas - Puebla
{osoriomauri,jlcarballido7,czepedac}@gmail.com

**Abstract** We show that logic $C_1$ cannot be extended to a paraconsistent logic in which the substitution theorem is valid. We show with the help of an answer set programming (ASP) tool called `clasp`, that $C_1$ is robust with respect to certain three valued paraconsistent logics with some desirable properties. In particular $C_1$ is robust with respect to three-valued logic $P_2$, a logic for which some of the De Morgan laws are valid.
**Keywords**: `clasp`, multi-valued logics, substitution theorem.

## 1 Introduction

According to our experience, we know that always it is very useful to have software tools that help us to analyze logics. One of these tools is the Answer Set Programming (ASP) tool called `clasp`[1], which computes the answer sets of logic programs. ASP is a declarative knowledge representation and logic programming language [7]. We use `clasp`-encodings of multi-valued logics for analyzing logics.

Two main approaches are common to define a logic, the Hilbert axiomatic system and the use of multi-valued tables that define the connectives of the logic. In the first approach the validity of a formula is determined by a set of axioms and a family of inference rules, namely, if the formula can be derived from those axioms and the use of the inference rules, then the formula is valid, otherwise it is not valid. In general, there are many ways of choosing the family of axioms to define a logic and Modus Ponens is one of the most common inference rules appearing in the definition of logics. In the second approach, the tables used to define the logic are called truth tables, each connective is regarded as a function taking values in a set of numbers (usually integers) that are specified from the beginning and are called truth values. Some of the values are chosen as designated values. Any formula that evaluates to one of the designated values regardless of the truth values taken by the atoms that appear in the formula, is considered valid. In this paper, we combine both approaches.

In logic, as in any other area of mathematics, when choosing a family of axioms to define a logic, it is desirable to have independence of the axioms, that is, any formula chosen as an axiom should be independent from the other axioms. Multi-valued logics can be used for this purpose (see an example of this in [10]), and in [16] is explained how to do it using an answer set-encoding. This

[1] http://potassco.sourceforge.net

methodology sometimes could have limitations (see [8]), however it is useful to researchers interested in the study of logics, such as in our case.

One of the properties a logic can have and in which we are particularly interested is paraconsistency. Following Béziau [1], a logic is paraconsistent if it has a negation $\neg$, which is paraconsistent in the sense that $a, \neg a \nvdash b$, and at the same time has enough strong properties to be called a negation. Paraconsistent logics have important applications, specifically [4] mention three applications in different fields: Mathematics, Artificial Intelligence and Philosophy. In relation to the second one, the authors mention that in certain domains, such as the construction of expert systems, the presence of inconsistencies is almost unavoidable (see for example [11]). An application that has not been fully recognized is the use of paraconsistent logics in non-monotonic reasoning. In this sense [6,14] illustrate such novel applications. Thus, the research on paraconsistent logics is far from being over. A three-valued paraconsistent logic of particular interest to us is $C_1$, which has been studied in [9].

In this paper, we present two results, first we show that there is not paraconsistent logic that extends $C_1$ and for which the substitution theorem holds, second we explain how to construct a program that obtains a paraconsistent three-valued logic, called $P_2$ [3], such that the logic $C_1$ is sound w.r.t. $P_2$.

We take advantage of `clasp` since it allows us to define *redundant constraints* to define easily the primitive connectives as mathematical functions such as the $\vee$, $\neg$, $\wedge$, and $\rightarrow$. For instance, in `clasp` we write

$1\{and(X,Y,Z) : v(Z)\}1 \leftarrow v(X), v(Y)$ instead of writing

$and(X,Y,Z) \leftarrow v(X), v(Z), v(Y), not\ nothera(X,Y,Z).$

$nothera(X,Y,Z) \leftarrow and(X,Y,Z1), Z! = Z1, v(X), v(Y), v(Z), v(Z1).$

as we wrote in a preliminary work using DLV [16]. We also use the called *conditions* in `clasp` to define easily and briefly some constraints in our encoding. For instance, in `clasp` we write $\leftarrow not\ f(X) : not\ sel(X) : v(X)$ instead of writing $\leftarrow notf(1), f(2)$ as we wrote in [16]. Moreover in our `clasp` *conditions* we use `not` which is the negation as failure used in DLV.

Our paper is structured as follows. In section 2, we summarize some basic concepts and definitions. In section 3, we show our results, a theorem and a `clasp`-encoding. Finally, in section 4, we present some conclusions.

## 2 Background

There are two ways to define a logic: by giving a set of axioms and specifying a set of inference rules; and by the use of truth values and interpretations. In this section we summarize each of them and we present some basic concepts and definitions useful to understand this paper.

### 2.1 Hilbert style

In Hilbert style proof systems, also known as axiomatic systems, a logic is specified by giving a set of axioms and a set of inference rules. In these systems, it

is common to use the notation $\vdash_X F$ for provability of a logic formula $F$ in the logic $X$. In that case we say that $F$ is a theorem of $X$.

We say that a logic $X$ is paraconsistent if the formula $(A \wedge \neg A) \rightarrow B$ is not a theorem [2].

The relevance of logics for which the formula $(A \wedge \neg A) \rightarrow B$ is not a theorem, is that they are useful to define alternative semantics that can be applied in the study of non monotonic reasoning as we mentioned in the introduction section.

A very important property satisfied by many logics is the substitution theorem which we present now.

**Definition 1.** *A logic $X$ satisfies the substitution theorem if: $\Gamma \vdash_X \alpha \leftrightarrow \beta$*[3] *then $\Gamma \vdash_X \Psi[\alpha/p] \leftrightarrow \Psi[\beta/p]$ for any formulas $\alpha$, $\beta$, and $\Psi$ and any atom $p$ that appear in $\Psi$ where $\Psi[\alpha/p]$ denotes the resulting formula that is left after every occurrence of $p$ is substituted by the formula $\alpha$.*

As examples of axiomatic systems, we present three logics: the positive logic [12], the $C_\omega$ logic which is a paraconsistent logic defined by daCosta [5], and $C_1$ logic [9]. In Table 1 we present a list of axioms, the first eight of them define positive logic. $C_\omega$ logic is defined by the axioms of positive logic plus axioms $C_\omega 1$ and $C_\omega 2$.

| | |
|---|---|
| **Pos1:** $A \rightarrow (B \rightarrow A)$ | $C_\omega$**1:** $A \vee \neg A$ |
| **Pos2:** $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$ | $C_\omega$**2:** $\neg\neg A \rightarrow A$ |
| **Pos3:** $A \wedge B \rightarrow A$ | |
| **Pos4:** $A \wedge B \rightarrow B$ | |
| **Pos5:** $A \rightarrow (B \rightarrow (A \wedge B))$ | |
| **Pos6:** $A \rightarrow (A \vee B)$ | |
| **Pos7:** $B \rightarrow (A \vee B)$ | |
| **Pos8:** $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (A \vee B \rightarrow C))$ | |

**Table 1.** Axiomatization of $C_\omega$.

$C_1$ logic is defined by the axioms of $C_\omega$ plus the following two axioms:

$$\neg_1\textbf{:}\ B^\circ \rightarrow ((A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A))$$
$$\neg_2\textbf{:}\ A^\circ \wedge B^\circ \rightarrow (A \wedge B)^\circ \wedge (A \vee B)^\circ \wedge (A \rightarrow B)^\circ$$

where $B^\circ = \neg(B \wedge \neg B)$.

[2] For any logic X that contains Pos1 and Pos2 (axioms of positive logic defined in Table 1) among its axioms and Modus Ponens as its unique inference rule, the formula $(A \wedge \neg A) \rightarrow B$ is a theorem if and only if $A, \neg A \vdash_X B$.

[3] Here we use the notation $\Gamma \vdash_X$ to indicate that the formula that follows is a theorem or a tautology depending on how the logic is defined.

### 2.2 Multi-valued logics

An alternative way to define a logic is by the use of truth values and interpretations. Multi-valued logics generalize the idea of using truth tables to determine the validity of formulas in classical logic. The core of a multi-valued logic is its *domain* of values $\mathcal{D}$, where some of such values are special and identified as *designated* or `select` values. Logic connectives (e.g. $\wedge$, $\vee$, $\rightarrow$, $\neg$) are then introduced as operators over $\mathcal{D}$ according to the particular definition of the logic, see [10].

An *interpretation* is a function $I\colon \mathcal{L} \rightarrow \mathcal{D}$ that maps atoms to elements in the domain. The application of $I$ is then extended to arbitrary formulas by mapping first the atoms to values in $\mathcal{D}$, and then evaluating the resulting expression in terms of the connectives of the logic (which are defined over $\mathcal{D}$). It is understood in general that, if $I$ is an interpretation defined on the arbitrary formulas of a given program $P$, then $I(P)$ is defined as the function $I$ applied to the conjunction of all the formulas in $P$. A formula $F$ is said to be a *tautology*, denoted usually by $\models F$ if, for every possible interpretation, the formula $F$ evaluates to a designated value. The simplest example of a multi-valued logic is classical logic where: $\mathcal{D} = \{0, 1\}$, 1 is the unique designated value, and the connectives are defined through the usual basic truth tables.

Note that in a multi-valued logic, so that it can truly be a *logic*, the implication connective has to satisfy the following property: for every value $x \in \mathcal{D}$, if there is a designated value $y \in \mathcal{D}$ such that $y \rightarrow x$ is designated, then $x$ must also be a designated value. This restriction enforces the validity of Modus Ponens in the logic.

As an example of a multi-valued logic, we present the well known paraconsistent logic $\mathbf{P}^2$ [3] (also called **Cive**), a three-valued logic that is relevant in this work. The truth values of logic $\mathbf{P}^2$ are in the domain $D = \{0, 1, 2\}$ where 1 and 2 are the designated values. The $\wedge$, $\vee$, $\rightarrow$, and $\neg$ connectives are defined according to the truth tables given in Table 2.

| $\wedge$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 2 | 2 |
| 2 | 0 | 2 | 2 |

| $\vee$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 0 | 2 | 2 |
| 1 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 |

| $\rightarrow$ | 0 | 1 | 2 |
|---|---|---|---|
| 0 | 2 | 2 | 2 |
| 1 | 0 | 2 | 2 |
| 2 | 0 | 2 | 2 |

| $x$ | $\neg x$ |
|---|---|
| 0 | 2 |
| 1 | 1 |
| 2 | 0 |

**Table 2.** Truth tables of connectives $\wedge$, $\vee$, $\rightarrow$, and $\neg$ in $\mathbf{P}^2$.

## 3 Main contribution

An interesting theoretical question that arises in the study of logics is whether a given logic satisfies the substitution theorem [15]. It is well known that there are several paraconsistent logics for which that theorem is not valid [2]. We show that logic $C_1$ can not be extended to a paraconsistent logic in which the substitution theorem is valid. In order to do this we provide a definition and some results.

**Definition 2.** *A logic X satisfies the weak substitution property if:* $\Gamma \vdash_X \alpha \leftrightarrow \beta$ *then* $\Gamma \vdash_X \neg\alpha \leftrightarrow \neg\beta$.

**Theorem 1.** *[13] Any logic stronger than* $\mathrm{C}_\omega$ *satisfies the weak substitution property iff satisfies the substitution property.*

The first result of our paper is that $C_1$ can not be extended to a paraconsistent logic in which the substitution theorem holds.

**Theorem 2.** *Any extension of logic* $C_1$ *to a logic where the substitution theorem holds, is not paraconsistent.*

*Proof.* It is easy to see that $a, \neg a \vdash (a \wedge \neg a) \leftrightarrow a$.
By substitution theorem we have $a, \neg a \vdash \neg(a \wedge \neg a) \leftrightarrow \neg a$,
but $a, \neg a \vdash \neg a$,
therefore $a, \neg a \vdash \neg(a \wedge \neg a)$.
We also know by an instance of axiom $\neg_1$ that
$\vdash \neg(a \wedge \neg a) \rightarrow ((\neg b \rightarrow a) \rightarrow ((\neg b \rightarrow \neg a) \rightarrow \neg\neg b)))$,
then by using modus ponens in the previous line
$a, \neg a \vdash ((\neg b \rightarrow a) \rightarrow ((\neg b \rightarrow \neg a) \rightarrow \neg\neg b)))$,
now we use **Pos1** and modus ponens and $a, \neg a \vdash (\neg b \rightarrow a)$,
therefore by usung modus ponens again $\vdash (\neg b \rightarrow \neg a) \rightarrow \neg\neg b$,
we also have $a, \neg a \vdash (\neg b \rightarrow a)$.
then $a, \neg a \vdash \neg\neg b$,
using $\mathrm{C}_\omega\mathbf{2}$ $\neg\neg b \vdash b$.
Finally, we have $a, \neg a \vdash b$.
This last line shows that paraconsistency does not hold. □

Our second result is based on an ASP encoding. ASP has been used to develop different approaches in the areas of planning, logical agents and artificial intelligence. However, as far as the authors know, it has not been used as a tool to study logics. Here we use ASP to represent axioms and the inference rule Moduls Ponens in `clasp` in order to find three-valued logics that are paraconsistent and for which the axioms of $C_1$ are tautologies.

In what follows, we explain how to construct a program that obtains three-valued logics such that the logic $C_1$ is sound w.r.t. them. We use this program repeatedly, first for the axioms of $C_1$ and then for the axioms of a logic called $C_1^+$, and then for larger family that define the logic $P_2$.

Based on a set of values, and a subset of values corresponding to the property of being select, the `clasp`-encoding constructs the adequate truth tables for the connectives of a multi-valued logic that makes all instances of the axioms of the given logic select. These truth tables are built in such a way that Modus Ponens preserves the property of being select. The encoding also includes the adequate conditions that each connective of the logic should satisfy, such as the arity, and the uniqueness; the definition of Modus Ponens; and all of the axioms of the logic. It is worth mentioning that all of the axioms are encoded as constraints. When each axiom is encoded as a constraint, the elimination of those assignment values of the logic connectives for which the axioms are not select, is guaranteed.

Now we present the `clasp`-encoding, $\Pi$ to search for three-valued logics for which logic $C_1$ is sound by extending $C_1$ with the addition of desirable properties represented in terms of axioms. We present the encoding of one of the axioms of $C_1$, the encoding of the other axioms is similar. The encoding uses the values 0, 1, and 2 to create the truth tables for the connectives of the logic. The select value are 1 and 2 [4]. Thus, for any assignment of the values 0, 1 and 2 to the statements letters of a formula $F$, the tables determine a corresponding value for $F$. If $F$ always takes the values 1 and 2, $F$ will be called select. Furthermore, $\Pi$ is a propositional program. As usual in ASP, we take for granted that programs with predicate symbols are only an abbreviation of the ground program. The encoding $\Pi$ corresponds to the program $P_{val} \cup P_{prim} \cup P_{def} \cup P_{Ax} \cup P_{par}$ that we present below. We want to remark that $P_{def}$ includes all the defined connectives of the $C_1$ logic, $P_{Ax}$ includes the axioms of $C_1$ logic.

$$P_{val}: \begin{cases} \%\textbf{Truth values:} 0,1,2 \\ v(0;1;2). \\ \%\textbf{Select value:} 0 \\ sel(1).sel(2). \end{cases} \qquad P_{prim}: \begin{cases} \%\textbf{Primitive connectives} \\ \%\textbf{implies, not, or, and, ...} \\ 1\{impl(X,Y,Z) : v(Z)\}1 \leftarrow v(X), v(Y). \\ 1\{neg(X,Z) : v(Z)\}1 \leftarrow v(X). \\ \ldots \end{cases}$$

$$P_{def}: \begin{cases} \%\ B^{\circ} \\ bl(X,Z) \leftarrow neg(X,X1), and(X,X1,Y), neg(Y,Z). \\ \%\textbf{Modus Ponens} \\ \leftarrow impl(X,Y,Z), sel(X), sel(Z), not\ sel(Y), v(X), v(Y), v(Z). \end{cases}$$

$$P_{Ax}: \begin{cases} \%\textbf{Axioms of } C_1 \\ \%\textbf{A1} : A \rightarrow (B \rightarrow A) \\ \leftarrow impl(B,A,Z), impl(A,Z,R), v(R), not\ sel(R). \\ \ldots \end{cases}$$

$$P_{par}: \begin{cases} \%\textbf{Paraconsistency: } (A \wedge \neg A) \rightarrow B \text{ is not a theorem.} \\ eval(R) \leftarrow neg(A,A1), and(A,A1,L), impl(L,B,R), v(R). \\ \leftarrow not\ eval(X) : not\ sel(X) : v(X). \end{cases}$$

We can see that the paraconsistency property is encoded as a constraint in $P_{par}$. This means that in case of obtaining answer sets they must satisfy the paraconsistency property.

When we execute the `clasp`-encoding $\Pi$, we obtain 8192 answer sets. Each of them corresponds to an adequate set of truth tables for the connectives of a paraconsistent three-valued logic that make logic $C_1$ sound. These 8192 three-valued logics really represent 4096 different three-valued logics because of isomorphisms created by the two designated values.

It is always convenient to look for logics as close to classical logic as possible, then we go one step further to obtain an extension proposed by Beziau for which certain De Morgan laws are valid. In this sense, if we replace the axiom $\neg_2$ in $C_1$ by the following stronger axiom, called $\neg_3$, then we obtain logic $C_1^+$ [9]:

$$A^{\circ} \vee B^{\circ} \rightarrow (A \wedge B) \circ \wedge (A \vee B) \circ \wedge (A \rightarrow B)^{\circ}$$

[4] We choose two designated values because we already know that when running the same program with the option of one designated value it does not return any logic.

When we implement this replacement in the `clasp`-encoding $\Pi$ and we execute it, we obtain 8 answer sets. Each of them corresponds to an adequate set of truth tables for the connectives of a three-valued logic that is sound with respect to the $C_1^+$ logic and satisfies the paraconsistency property. These 8 three-valued logics really represent 4 different three-valued logics because of isomorphisms created by the two designated values.

Now, if we add the following two axioms to $C_1^+$ logic

$$(A \vee B)^\circ$$

$$(A \to B)^\circ$$

and we add their implementation in the `clasp`-encoding $\Pi$, we obtain 2 answer sets, each of them corresponds to an adequate set of truth tables for the connectives of a three-valued logic thats make logic $P_2$ [3] sound and satisfies the paraconsistency property. These two three-valued logics are isomorphic due to the symmetry generated by the two designated values.

## 4 Conclusions

ASP have been used to develop different approaches in the areas of planning, logical agents and artificial intelligence. However, as far as the authors know, it has not been used as a tool to study logics. We provide a `clasp`-encoding that can be used to obtain paraconsistent multi-valued logics. We also proved that $C_1$ logic can not be extended to a paraconsistent logic where the substitution theorem holds.

## References


1. J. Y. Béziau. The paraconsistent logic Z. A possible solution to jaskowski's problem. *Logic and logical philosophy*, 15:99–111, 2006.
2. W. A. Carnielli and J. Marcos. Limits for paraconsistent calculi. *Notre Dame Journal of Formal Logic*, 40(3):375–390, 1999.
3. W. A. Carnielli and J. Marcos. A taxonomy of **C**-Systems. In *Paraconsistency: The Logical Way to the Inconsistent, Proceedings of the Second World Congress on Paraconsistency (WCP 2000)*, number 228 in Lecture Notes in Pure and Applied Mathematics, pages 1–94. Marcel Dekker, Inc., 2002.
4. N. C. A. da Costa, J.-Y. Béziau, and O. A. S. Bueno. Aspects of paraconsistent logic. *Logic Journal of the IGPL*, 3(4):597–614, 1995.
5. N. C. A. daCosta. *On the theory of inconsistent formal systems (in Portuguese)*. PhD thesis, Curitiva: Editora UFPR, Brazil, 1963.
6. M. J. O. Galindo, J. R. A. Ramírez, and J. L. Carballido. Logical weak completions of paraconsistent logics. *J. Log. Comput.*, 18(6):913–940, 2008.
7. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

8. K. Gödel. Über unabhängigkeitsbeweise im aussagenkalkül(sobre pruebas de independencia en el cálculo conectivo). In ergebnisse eines mathematischen kolloquiums, ed. by k. menger, num. 4 (1931-32), pages 9-10.
9. J.-Y.Béziau. Logiques construites suivant les methodes de dacosta. *Logique et Analyse*, 33:259–272, 1990.
10. E. Mendelson. *Introduction to Mathematical Logic*. Wadsworth, Belmont, CA, third edition, 1987.
11. V. S. S. N. da Costa. Paraconsistent logics as a formalism for reasoning about inconsistent knowledge bases. *Artificial Intelligence in Medicine*, 1:167–174, 1989.
12. M. Osorio and J. L. Carballido. Brief study of G'$_3$ logic. *Journal of Applied Non-Classical Logic*, 18(4):475–499, 2008.
13. M. Osorio, J. L. Carballido, and C. Zepeda. An application of clasp in the study of logics. In J. P. Delgrande and W. Faber, editors, *LPNMR*, volume 6645 of *Lecture Notes in Computer Science*, pages 278–283. Springer, 2011.
14. M. Osorio, J. A. Navarro, J. Arrazola, and V. Borja. Logics with common weak completions. *Journal of Logic and Computation*, 16(6):867–890, 2006.
15. D. van Dalen. *Logic and Structure*. Springer, Berlin, second edition, 1980.
16. C. Zepeda, J. L. Carballido, A. Marín, and M. Osorio. Answer set programming for studying logics. In *Special Session MICAI 2009. Ed. IEEE Computer Society*, pages 153–158, Puebla, Mxico. ISBN: 13 978-0-7695-3933-1.