On 3-valued paraconsistent Logic Programming

Marcelo E. Coniglio
Kleidson E. Oliveira

Institute of Philosophy and Human Sciences and
Centre For Logic, Epistemology and the History of Science, UNICAMP, Brazil

Support: FAPESP

Syntax Meets Semantics 2016

Summary

Introduction

- Application of Non-classical Logics in logic programs is more delicated that appears.
- Several proposals can be seem in Kifer and Subrahmanian [9], Ginsber [8] and Fitting [7]. 3-valued logic programming was considered in Przymusinski [10] and Delahaye and Thibau [6]. Paraconsistent logic programming was also investigated by Blair and Subrahmanian [1] and by Damásio and Pereira [5].

- Based on previous studies of Rodrigues [11] on the foundations of Paraconsistent Logic Programming based on several paraconsistent logics in the so called hierarchy of *Logics of Formal Inconsistency* (LFIs, see Carnielli, Coniglio and Marcos [3]), we will describe in this talk some results on the theory of clausal resolution and on Logic Programming system based on paraconsistent logic QMPT0.

Presenting QMPT0

At first, we will present the two negations to be considered, $\neg$ and $\sim$. Being that $\neg$ is the weak negation and $\sim$ is the strong negation. The tables are as follows:

Table of Negations

P	¬P	∼P	∼¬P	∼∼P	¬¬P	¬∼P
1	0	0	1	1	1	1
B	B	0	0	1	B	1
0	1	1	0	0	0	0

The third value "B" (both) is distinguished, thus the weak negation of B is B and the strong negation of B is 0. We assume that B is an intermediate value between 0 and 1.

Presenting MPT0 and QMPT0

The 3-valued matrix logic MPT0 (see [4]) can be defined over the signature $\{\wedge, \vee, \rightarrow, \neg, \sim\}$ in the domain $\{1, B, 0\}$ in which $D = \{1, B\}$ is the set of distinguished values. The tables that interpret the connectives are as follows:

Connectives of MPT0

$\rightarrow$	1	B	0
1	1	B	0
B	1	B	0
0	1	1	1

$\wedge$	1	B	0
1	1	B	0
B	B	B	0
0	0	0	0

$\vee$	1	B	0
1	1	1	1
B	1	B	B
0	1	B	0

p	$\neg p$
1	0
B	B
0	1

p	$\sim p$
1	0
B	0
0	1

The equivalence connective $\equiv$ is defined as
$(\alpha \equiv \beta) =_{def} (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$, whose table is as follows:

$\equiv$	1	B	0
1	1	B	0
B	B	B	0
0	0	0	1

Now, before we give the axiomatization to MPT0, we have to remember the axiomatization of **mbC**.

The basic (**LFIs**) is the (propositional) logic **mbC**, defined over the signature $\{\wedge, \vee, \rightarrow, \neg, \circ\}$ as follows:

Axioms:

(A1) $\alpha \rightarrow (\beta \rightarrow \alpha)$

(A2) $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow (\alpha \rightarrow \gamma))$

(A3) $\alpha \rightarrow (\beta \rightarrow (\alpha \wedge \beta))$

(A4) $(\alpha \wedge \beta) \rightarrow \alpha$

(A5) $(\alpha \wedge \beta) \rightarrow \beta$

(A6) $\alpha \rightarrow (\alpha \vee \beta)$

(A7) $\beta \rightarrow (\alpha \vee \beta)$

(A8) $(\alpha \rightarrow \gamma) \rightarrow ((\beta \rightarrow \gamma) \rightarrow ((\alpha \vee \beta) \rightarrow \gamma))$

(A9) $\alpha \vee (\alpha \rightarrow \beta)$

Axioms: (Cont.)

(A10) $\alpha \vee \neg\alpha$

(bc1) $\circ\alpha \to (\alpha \to (\neg\alpha \to \beta))$

Inference Rule:

$$\text{(MP)} \qquad \frac{\alpha, \alpha \to \beta}{\beta}$$

Observe that (A1)-(A9) plus (MP) is positive classical logic

Now, a sound and complete Hilbert calculus for **MPT0**, called **LPT0**, will be defined.

Definition - The calculus LPT0 for MPT0

Let Σ_1 be the signature $\{\wedge, \vee, \rightarrow, \neg, \sim\}$. The Hilbert calculus **LPT0** over Σ_1 is defined by taking axiom schemas (A1)-(A10) from **mbC**, **MP**, plus the following:

(These axiom schemas are present in the book Paraconsistent Logic: Consistency, contradiction and negation, by Carnielli and Coniglio [2])

Axiom schemas:

$\alpha \vee \sim\alpha$	(**TND**)
$\alpha \to (\sim\alpha \to \beta)$	(**exp**)
$\neg\sim\alpha \to \alpha$	(**dneg**)
$\neg\neg\alpha \to \alpha$	(**cf**)
$\alpha \to \neg\neg\alpha$	(**ce**)
$\neg(\alpha \vee \beta) \to (\neg\alpha \wedge \neg\beta)$	(**neg**$\vee_1$)
$(\neg\alpha \wedge \neg\beta) \to \neg(\alpha \vee \beta)$	(**neg**$\vee_2$)
$\neg(\alpha \wedge \beta) \to (\neg\alpha \vee \neg\beta)$	(**neg**$\wedge_1$)
$(\neg\alpha \vee \neg\beta) \to \neg(\alpha \wedge \beta)$	(**neg**$\wedge_2$)
$\neg(\alpha \to \beta) \to (\alpha \wedge \neg\beta)$	(**lr**$_\to$)
$(\alpha \wedge \neg\beta) \to \neg(\alpha \to \beta)$	(**lp**$_\to$)

From this we have the following:

Lemma

1. $\neg\neg A \equiv A$
2. $\sim\sim A \equiv A$
3. $\neg\sim A \equiv A$

From this we have the following:

Lemma

1. $\neg\neg A \equiv A$
2. $\sim\sim A \equiv A$
3. $\neg\sim A \equiv A$
4. $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$
5. $(A \vee B) \wedge C \equiv (A \wedge C) \vee (B \wedge C)$
6. $\neg(A \vee B) \equiv \neg(A) \wedge \neg(B)$
7. $\neg(A \wedge B) \equiv \neg(A) \vee \neg(B)$
8. $\sim(A \vee B) \equiv \sim(A) \wedge \sim(B)$
9. $\sim(A \wedge B) \equiv \sim(A) \vee \sim(B)$

In MPT0 we can introduce the following notions:

- A *literal* is a formula of the form A, $\neg A$, $\sim A$ or $\sim\neg A$, in which A is a atomic formula. In each case it is said that the literal contains the atomic formula A.
- Literals of the form A or $\neg A$ are called *positive*, the others are called *negative*
- A formula A is called of *atom* when there are only positive literals in A

This is motivated by the fact that, in MPT0, there are only four formulas (up to equivalence) based on p constructed with $\neg$ and $\sim$: p, $\neg p$, $\sim p$ and $\sim\neg p$. As you can see again in the table of negations.

Table of Negations

P	¬P	∼P	∼¬P	∼∼P	¬¬P	¬∼P
1	0	0	1	1	1	1
B	B	0	0	1	B	1
0	1	1	0	0	0	0

- A *clause* of MPT0 is a formula of the form:

$$L_1 \vee \cdots \vee L_k \vee \sim L_{k+1} \vee \cdots \vee \sim L_{k+m}$$

  such that each L_i is a positive literal in MPT0.
- A clause is called *positive* (*negative*) if it contains only positive (negative) literals.
- A set S of clauses is called *satisfiable* if there is a valuation on MPT0 such that $v(K) \in \{1, \mathrm{B}\}$ for all clauses K in S. In that case v is called a *model* of S.
- A clause K in MPT0 is consequence of a set of clauses S (denoted by $S \models_{MPT0} K$), if for all valuations v, if $v(S) \subseteq D$ then also holds $v(K) \in D$.

The MPT0 extension to first-order logic is known as QMPT0. The semantics of QMPT0 (which will be described very briefly here) is given by the so-called *pragmatic structures*, introduced by da Costa, Mikenberg and Chuaqui in the context of the theory of quasi-truth. Afterwards it was generalized by Coniglio and Silvestrini.

A pragmatic structure is an structure $\mathfrak{A} = \langle D, (\cdot)^{\mathfrak{A}} \rangle$ appropriate to interpret the first order languages in Tarskian style, in which D is a non-empty set (the domain of $\mathfrak{A}$) and $(\cdot)^{\mathfrak{A}}$ interprets the symbols of the language in the usual way, with this difference: each n-ary predicate p in the language is interpreted as a triple $p^{\mathfrak{A}} = \langle p_{+}^{\mathfrak{A}}, p_{-}^{\mathfrak{A}}, p_{B}^{\mathfrak{A}} \rangle$ where p_{+}, p_{-} and p_B are mutually disjoint sets such that $p_{+} \cup p_{-} \cup p_B = D^n$. The elements of p_{+}, p_{-} and p_B are the n-uples that satisfy p, do not satisfy p, and that simultaneously satisfy and do not satisfy p, respectively.

Appropriate operations are defined between such triples for interpreting the connectives and the quantifiers. Thus, a formula φ with free variables $x_1, \ldots, x_n$ generates a triple $\varphi^{\mathfrak{A}} = \langle \varphi^{\mathfrak{A}}_{+}, \varphi^{\mathfrak{A}}_{-}, \varphi^{\mathfrak{A}}_{B} \rangle$ such that

- $\varphi_{+}^{\mathfrak{A}} = \{\vec{a} \in D^n \ : \ \mathfrak{A} \models \varphi[\vec{a}] \text{ and } \mathfrak{A} \not\models \neg\varphi[\vec{a}]\}$;
- $\varphi_{-}^{\mathfrak{A}} = \{\vec{a} \in D^n \ : \ \mathfrak{A} \not\models \varphi[\vec{a}] \text{ and } \mathfrak{A} \models \neg\varphi[\vec{a}]\}$;
- $\varphi_{B}^{\mathfrak{A}} = \{\vec{a} \in D^n \ : \ \mathfrak{A} \models \varphi[\vec{a}] \text{ and } \mathfrak{A} \models \neg\varphi[\vec{a}]\}$;
- $\varphi_{+}^{\mathfrak{A}} \cup \varphi_{B}^{\mathfrak{A}} = \{\vec{a} \in D^n \ : \ \mathfrak{A} \models \varphi[\vec{a}]\}$;
- $\varphi_{-}^{\mathfrak{A}} \cup \varphi_{B}^{\mathfrak{A}} = \{\vec{a} \in D^n \ : \ \mathfrak{A} \models \neg\varphi[\vec{a}]\}$.

A *closed* (or *ground*) term is a term with no occurrences of variables. A *closed formula* is a formula that not contains free occurrences of variables.

The universal and existential closure of a formula are defined as usual.

In QMPT0 we introduce the following notions:

- A first-order *literal* is a formula of the form A, $\neg A$, $\sim A$ or $\sim\neg A$, in which A is an atomic formula of the first-order language. In each case it is said that the literal contains the atomic formula A.
- Literals of the form A or $\neg A$ are called *positive*, the others are called *negative*
- A positive literal is also called, a *atom*.

- A *clause* of QMPT0 is a closed formula of the form:

$$\forall x_1 \cdots \forall x_n (L_1 \vee \cdots \vee L_k \vee \sim L_{k+1} \vee \cdots \vee \sim L_{k+m})$$

such that each L_i is a positive literal in QMPT0 and $x_1, \ldots, x_n$ are all the variables occuring in $(L_1 \vee \ldots \vee L_k \vee \sim L_{k+1} \vee \ldots \vee \sim L_{k+m})$. The usual notation we will use for clauses, equivalent to the above presented is:

$$\forall x_1 \cdots \forall x_n (L_1 \vee \cdots \vee L_k \leftarrow L_{k+1} \wedge \cdots \wedge L_{k+m})$$

or just

$$L_1, \ldots, L_k \leftarrow L_{k+1}, \ldots, L_{k+m}$$

- A clause is called *positive* (*negative*) if it contains only positive (negative) literals.
- A set S of clauses is called *satisfiable* if there is a pragmatic structure $\mathfrak{A}$ such that $\mathfrak{A} \models K$ for all clauses K in S. In that case $\mathfrak{A}$ is called a *model* of S.
- A clause K is a consequence from a set of clauses S (denoted by $S \models_{QMPT0} K$), if for all models $\mathfrak{A}$ of S also holds that $\mathfrak{A} \models K$.

A *definite program clause* is a clause of the form

$$L \leftarrow K_1, \ldots, K_n$$

containing exactly one atom in its consequent. The positive literal L is called *head* and $K_1, \ldots, K_n$ is called *body* of the programe clause.

Alternatively, a definite program clause may be represented as

$$(\neg)A \leftarrow (\neg)A_1, \ldots, (\neg)A_n$$

where $(\neg)A$ denotes one of the literals A (atomic formula) or $\neg A$ (paraconsistent negation of an atomic formula).

A *unit* is a clause of the form

$$L \leftarrow$$

that is, a definite program clause with empty body.

A *definite program* $\mathcal{P}$ is a finite set of definite programs clauses.

The *empty clause*, denoted by $\Box$, is a clause whose antecedent and consequent are empty and it can be interpreted as a classic contradiction in the sense that it is unsatisfiable.

Let α be a first-order formula without quantifiers in a signature with at least one individual constant. We define $S_\alpha = \{\bar{\alpha} \ : \ \bar{\alpha}$ is a ground instance $\alpha\}$. If Γ is a set of formulas without quantifiers, then $\bar{\Gamma} = \bigcup\{S_\alpha \ : \ \alpha \in \Gamma\}$.

Proposition

Given a set of clauses S, $\bar{S}$ is satisfiable in MPT0 (as a set of propositional clauses) if and only if S is satisfiable in QMPT0.

A Resolution Calculus for QMPT0

Inspired by the approach to paracomplete 3-valued clausal resolution introduced in 1986 by P. H. Schmitt in [12], we set a resolution calculus for QMPT0. For this, just a *basic resolution rule* will be considered as an inference rule, as can be seen in the coming slides. Since the clauses are implicitly universally quantified, an auxiliary concept will be necessary.

Definition

Two literals of QMPT0, L_1 and L_2, are said to be *complementary* if one of the following conditions holds:

1. L_1 is positive and L_2 is $\sim L_1$.
2. L_2 is positive and L_1 is $\sim L_2$.

Below are two examples of resolution rules:

$\dfrac{L \vee \bigvee_{i=1}^{n} A \qquad K \vee \bigvee_{j=1}^{m} {\sim} A}{L \vee K}$	$\dfrac{L \vee \bigvee_{i=1}^{n} \neg A \qquad K \vee \bigvee_{j=1}^{m} {\sim}\neg A}{L \vee K}$

Formally, we have:

Definition

Let $K_1 = L_{1,1} \vee \ldots \vee L_{1,n}$ and $K_2 = L_{2,1} \vee \ldots \vee L_{2,r}$ be two clauses. A clause K is obtained from K_1 and K_2 through a *basic resolution step* if there are literals L_1 and L_2, with L_i occurring in K_i $(i = 1, 2)$, and a substitution σ such that $\sigma(L_1)$ and $\sigma(L_2)$ are complementary literals, being σ the most general unifier with this property. In this case, the resolvent is $K = \sigma(K_0)$, where K_0 is the disjunction of literals that appear in K_1 (unless all instances of L_1 as literal in K_1) or in K_2 (unless all instances of L_2 as literal in K_2). We say that K is a *basic resolvent* of K_1 and K_2.

Continue...

From K_1 and K_2 we obtain K by a *general resolution step* if there are renaming substitutions μ_1 and μ_2 such that K can be obtained from $\mu_1(K_1)$ e $\mu_2(K_2)$ by a basic resolution step.

Definition

For a set S of clauses:

- $Res(S)$ denotes the closure of S by resolution
- $Subst(S)$ is the set of all the substituition instances of clauses in S.

Lemma

In QMPT0, a set of clauses S is satisfiable iff $Res(S)$ is satisfiable.

The importance of the Paraconsistent third-excluded law

Let $\mathcal{P}_1$ and $\mathcal{P}_2$ be the programs:

$$\mathcal{P}_1 = \begin{cases} A \longleftarrow C \\ A \longleftarrow \neg C \end{cases} \quad \text{and} \quad \mathcal{P}_2 = \begin{cases} A \longleftarrow \neg C \\ C \longleftarrow A \end{cases}$$

We need $C \vee \neg C$ to derive $A \vee A$ in the first program and $C \vee C$ in the second, as it can be seen in the following derivations by resolution:

$$\dfrac{\dfrac{A \vee {\sim}\neg C \qquad C \vee \neg C}{A \vee C \qquad A \vee {\sim} C}}{A \vee A}$$

and

$$\dfrac{\dfrac{A \vee {\sim}\neg C \qquad C \vee \neg C}{A \vee C \qquad C \vee {\sim} A}}{C \vee C}$$

This lead us to define the following:

Definitions

- Given a set of clauses S, We define the *support for S* as the set $\text{Sup}(S) = \{p(x_1, \ldots, x_n) : p$ is a n-ary predicate symbol that occurs in $S\}$.
- Let S be a set of clauses, we define S_+ as the set $S \cup \{p(x_1, \ldots, x_n) \vee \neg p(x_1, \ldots, x_n) : p(x_1, \ldots, x_n) \in \text{Sup}(S)\}$. In other words S_+ denotes the union of S with relevant instances of the third-excluded law.

Thanks to Sup(S), we guarantee the completeness of resolution.

Theorem - Completeness of Clausal Resolution in QMPT0, version 1

Let S be a set of clauses. Then, S_+ is satisfiable in QMPT0 iff the empty clause does not belong to $Res(S_+)$.

Theorem - Completeness of Clausal Resolution in QMPT0, version 2

Let S be a set of clauses which is satisfiable in QMPT0, and let L be a ground literal (that is, without variables). Then, $S \models_{QMPT0} L$ iff $\bigvee_{i=1}^{k} L \in Res(S_+)$ for some $k \geq 1$.

Corollary - Completeness of Clausal Resolution in QMPT0, version 3

Let S be a set of clauses in QMPT0, and let L be a ground literal. Then, $S \models_{QMPT0} L$ iff the empty clause belongs to $Res(S_+ \cup \{\sim L\})$.

Now we define *Herbrand Universe* and *Herbrand Base* for a first order language $\mathcal{L}$.

Definition - Herbrand Universe

The *Herbrand Universe* $U_{\mathcal{L}}$ for $\mathcal{L}$ is the set of all ground terms that can be formed from constant and function symbols that appear in $\mathcal{L}$.

Definition - Herbrand Base

The Herbrand base $B_{\mathcal{L}}$ for QMPT0 is the set of all ground positive literals that can be formed using the predicate symbols of QMPT0 with the ground terms of Herbrand Universe as parameters.

Consider now the ground atoms $\{P_1, P_2, P_3, P_4\}$ and the following situation:
$\mathcal{I} = \{P_1, P_2, \neg P_3\}$ is a partial interpretation. (No information about P_4
$\mathcal{I} = \{P_1, P_2, \neg P_3, P_4\}$ is a total interpretation.
$\mathcal{I} = \{P_1, P_2, \neg P_3\ P_4, \neg P_4\}$ is a total interpretation.
This induces the following definitions.

Definition

Given a program $\mathcal{P}$, a subset $\mathcal{I}$ of $B_{\mathcal{P}}$ is called a *Herbrand partial interpretation*. If $\mathcal{I}$ contains all the ground literals that are logical consequences of the program $\mathcal{P}$ in QMPT0, $\mathcal{I}$ is called a *Herbrand partial model*.

Given the Herbrand base $B_{\mathcal{L}}$, we have that $B_{\mathcal{L}}^{+}$ is a subset of $B_{\mathcal{L}}$ formed by the ground atomic formulas. Then we can define a *Herbrand pragmatic interpretation*.

Definition

A *Herbrand (pragmatic) interpretation* for $\mathcal{L}$ is a subset $\mathcal{I}$ of $B_{\mathcal{L}}$ with the following property: for each $A \in B_{\mathcal{L}}^{+}$, either $A \in \mathcal{I}$ or $\neg A \in \mathcal{I}$.

Any Herbrand interpretation $\mathcal{I}$ generates in fact a pragmatic structure $\mathfrak{A}$ as follows: for each n-ary predicate symbol p,

- $p_{+}^{\mathfrak{A}} = \{(t_1, \ldots, t_n) \in U_{\mathcal{L}}^{n} : p(t_1, \ldots, t_n) \in \mathcal{I} \text{ and } \neg p(t_1, \ldots, t_n) \notin \mathcal{I}\}$;
- $p_{-}^{\mathfrak{A}} = \{(t_1, \ldots, t_n) \in U_{\mathcal{L}}^{n} : p(t_1, \ldots, t_n) \notin \mathcal{I} \text{ and } \neg p(t_1, \ldots, t_n) \in \mathcal{I}\}$;
- $p_{B}^{\mathfrak{A}} = \{(t_1, \ldots, t_n) \in U_{\mathcal{L}}^{n} : p(t_1, \ldots, t_n) \in \mathcal{I} \text{ and } \neg p(t_1, \ldots, t_n) \in \mathcal{I}\}$.

Observe that either $p(t_1, \ldots, t_n) \in \mathcal{I}$ or $\neg p(t_1, \ldots, t_n) \in \mathcal{I}$.

With some abuse of language, when we have a definite program $\mathcal{P}$, we will refer to the Herbrand universe $U_{\mathcal{P}}$ and the Herbrand base $B_{\mathcal{P}}$ of $\mathcal{P}$.

Definition

Let $\mathcal{L}$ be a first-order language and S the set of closed formulas in $\mathcal{L}$. A *Herbrand model* for S is a Herbrand interpretation for $\mathcal{L}$ that is a model of S.

Lemma

Let S be a set of clauses and suppose that S has a model.Then S has a Herbrand model.

Proof:

Let $\mathfrak{A}$ be a pragmatic interpretation which is a model of S. We define a Herbrand interpretation $\mathcal{I}$ as follows:

$$\mathcal{I} = \{L \in B_{\mathcal{L}} \ : \ \mathfrak{A} \models L\}$$

Using that $\mathfrak{A}$ is a model of S, it follows that the pragmatic structure $\mathfrak{A}_{\mathcal{I}}$ generated by $\mathcal{I}$ is also a model of S.

Declarative Semantics

Each classic logic program is associated with a monotonic function that plays a very important role in the theory. This technique is adapted to the case of QMPT0.

Definition

Let $\mathcal{P}$ be a definite program. The mapping $T_{\mathcal{P}}: 2^{B_{\mathcal{P}}} \to 2^{B_{\mathcal{P}}}$ is defined as follows.

$T_{\mathcal{P}}(\mathcal{I}) = \{L \in B_{\mathcal{P}} \ : \ \exists K \in \bar{\mathcal{P}}\,(\text{head}(K) = L \text{ and } \text{body}(K) \subseteq \mathcal{I}\}.$

Proposition

Let $\mathcal{P}$ be a definite program. Then the mapping $T_{\mathcal{P}}$ is monotonic and continuous.

Definition

Given a program $\mathcal{P}$, the *Least partial Model for* $\mathcal{P}$ is the set $M_{\mathcal{P}} = \{L \in B_{\mathcal{P}} \ : \ \mathcal{P} \models_{QMPT0} L\}$.

Definition

Given a program $\mathcal{P}$, and $\bar{\mathcal{P}}$ the set of ground clauses obtained from it, we define the *ground support of* $\mathcal{P}$ as the set
$\bar{\text{Sup}}(\mathcal{P}) = \{A : A \in B^{+}_{\mathcal{P}}$, and there exist $\text{K} \in \bar{\mathcal{P}}$ such that
A occurs in K or $\neg A$ occurs in K $\}$.

As before, $\bar{\text{Sup}}(\mathcal{P})$ serves to collect the potential applications of the excluded middle of the form $A \vee \neg A$.

Suppose that in a certain program $\mathcal{P}$ we have two potential applications of the third excluded law, say
$G \ and \ \neg G \ and \ H \ and \ \neg H$. This will generate the 4 extended programs:

$\mathcal{P} \cup \{G, H\}$
$\mathcal{P} \cup \{G, \neg H\}$
$\mathcal{P} \cup \{\neg G, H\}$
$\mathcal{P} \cup \{\neg G, \neg H\}$

Formally, we have:

Definition

Let $\lambda(\mathcal{P})$ be the cardinal of $\bar{\text{Sup}}(\mathcal{P})$, and let $\bar{\text{Sup}}(\mathcal{P}) = \{A_i \ : \ i < \lambda(\mathcal{P})\}$ be an enumeration of $\bar{\text{Sup}}(\mathcal{P})$. We define

$$\mathcal{P}^+ = \mathcal{P} \cup \{A_i \vee \neg A_i \ : \ i < \lambda(\mathcal{P})\}.$$

Finally, for $\gamma \in 2^{\lambda(\mathcal{P})}$ let

$$L_i^\gamma = \begin{cases} A_i \text{ if } \gamma(i) = 0 \\ \neg A_i \text{ if } \gamma(i) = 1 \end{cases}$$

Let $\mathcal{I}_\gamma^\mathcal{P} = \{L_i^\gamma \ : \ i < \lambda(\mathcal{P})\}$.
This produces the *extended program* $\mathcal{P}_\gamma = \mathcal{P} \cup \mathcal{I}_\gamma^\mathcal{P}$, for each $\gamma \in 2^{\lambda(\mathcal{P})}$.

Observe that $\mathcal{P}_\gamma$ is possible infinite.

From the definitions before, immediately we prove the following:

Proposition

Let $\mathcal{P}$ be a definite program and K a clause without variables. So: $K \in Res(\mathcal{P}_+)$ implies that $K \in Res(\mathcal{P}^+)$.

We now need to establish some auxiliary technical results on classical first order logic CL and its relationship with QMPT0. To differentiate resolution systems, we denote by Res_{QMPT0} and Res_{CL} the clausal resolution operators of the first order logics QMPT0 and CL, respectively.

Recall first a classical result.

Theorem

Let $\mathcal{P}$ be a definite program in CL, and A a ground atom. Then,

$$\mathcal{P} \models_{CL} A \quad \text{iff} \quad A \in T_{\mathcal{P}}\uparrow\omega.$$

From that result from classical logic programming, we obtain its analogue for QMPT0. The formulation is more complicated, because of the potential applications of the excluded middle, which forces to consider all extensions $\mathcal{P}_\gamma$ from the original program $\mathcal{P}$. We must first state an additional Lemma:

lemma

Let $\mathcal{P}$ be a definite program in QMPT0, and let $\mathcal{P}'$ be the definite program in CL obtained by applying the following translation $(\cdot)'$:
$\neg p(t_1, \ldots, t_n)$ by $p'(t_1, \ldots, t_n)$ and
$\sim\neg p(t_1, \ldots, t_n)$ by $\sim p'(t_1, \ldots, t_n)$.)
So for all ground positive literal L, $L \in T_{\mathcal{P}}\uparrow\omega$ in QMPT0 iff $L' \in T_{\mathcal{P}'}\uparrow\omega$ in CL.

Finally, we arrive to the following result

Theorem - Fixpoint Characterization of Herbrand Least Partial Model for QMPT0

Let $\mathcal{P}$ be a definite program in QMPT0, and L a ground positive literal. Then,

$$\mathcal{P} \models_{QMPT0} L \quad \text{iff} \quad L \in \bigcap_{\gamma \in 2^{\lambda(\mathcal{P})}} T_{\mathcal{P}_\gamma} \uparrow \omega.$$

Observe that all the extensions of $\mathcal{P}$ must be considered, because of the third-excluded law.

Example:
Let $\mathcal{P}$ the logic program below.

$$\mathcal{P} = \begin{cases} A \longleftarrow H \\ C \longleftarrow \neg H \\ D \longleftarrow C \\ A \longleftarrow D, \neg G \\ A \longleftarrow G, \neg E \\ \neg E \longleftarrow \end{cases}$$

We find the set of ground literal that are a consequence from it, using the characterization given by the previous theorem. To simplify the exposition, we assume that the program is propositional and finite. We will also consider in $\bar{\text{Sup}}(\mathcal{P})$ only the atoms that are needed to generated $M_{\mathcal{P}}$, to shorten the presentation.

Note that the set $M_{\mathcal{P}}$ of ground literal deductible from $\mathcal{P}$ is $\{\neg E, A\}$. However, to deduce A, two applications of the excluded middle are required: $H \vee \neg H$ and $G \vee \neg G$. Thus, we define the following:

$\bar{\text{Sup}}(\mathcal{P}) = \{H, G\} = \{A_0, A_1\}$
$\lambda(\mathcal{P}) = 2 = \{0, 1\}$
$A_0 = H$ and $A_1 = G$
$2^{\lambda(\mathcal{P})} = \{(0,0), (0,1), (1,0), (1,1)\}$
$\mathcal{I}_{(0,0)} = \{H, G\}$, $\mathcal{I}_{(0,1)} = \{H, \neg G\}$, $\mathcal{I}_{(1,0)} = \{\neg H, G\}$,
$\mathcal{I}_{(1,1)} = \{\neg H, \neg G\}$.

- $T_{\mathcal{P}_{(0,0)}}\uparrow 0 = \emptyset$
  $T_{\mathcal{P}_{(0,0)}}\uparrow 1 = \{\neg E, H, G\}$
  $T_{\mathcal{P}_{(0,0)}}\uparrow 2 = T_{\mathcal{P}_{(0,0)}}(\{\neg E, H, G\}) = \{\neg E, A, H, G\} = T_{\mathcal{P}_{(0,0)}}\uparrow \omega$.
- $T_{\mathcal{P}(0,1)}\uparrow 0 = \emptyset$
  $T_{\mathcal{P}(0,1)}\uparrow 1 = \{\neg E, H, \neg G\}$
  $T_{\mathcal{P}(0,1)}\uparrow 2 = T_{\mathcal{P}(0,1)}(\{\neg E, H, \neg G\}) = \{\neg E, A, H, \neg G\} = T_{\mathcal{P}(0,1)}\uparrow \omega$.

- $T_{\mathcal{P}_{(1,0)}}\uparrow 0 = \emptyset$
  $T_{\mathcal{P}_{(1,0)}}\uparrow 1 = \{\neg E, \neg H, G\}$
  $T_{\mathcal{P}_{(1,0)}}\uparrow 2 = T_{\mathcal{P}_{(1,0)}}(\neg E, \neg H, G) = \{\neg E, \neg H, G, A, C\}$
  $T_{\mathcal{P}_{(1,0)}}\uparrow 3 = T_{\mathcal{P}_{(1,0)}}(\{\neg E, \neg H, G, A, C\}) =$
  $\{\neg E, \neg H, G, A, C, D\} = T_{\mathcal{P}_{(1,0)}}\uparrow \omega$.
- $T_{\mathcal{P}_{(1,1)}}\uparrow 0 = \emptyset$
  $T_{\mathcal{P}_{(1,1)}}\uparrow 1 = \{\neg E, \neg H, \neg G\}$
  $T_{\mathcal{P}_{(1,1)}}\uparrow 2 = T_{\mathcal{P}_{(1,1)}}(\{\neg E, \neg H, \neg G\}) = \{\neg E, C, \neg H, \neg G\}$
  $T_{\mathcal{P}_{(1,1)}}\uparrow 3 = T_{\mathcal{P}_{(1,1)}}(\{\neg E, C, \neg H, \neg G\}) = \{\neg E, C, D, \neg H, \neg G\}$
  $T_{\mathcal{P}_{(1,1)}}\uparrow 4 = T_{\mathcal{P}_{(1,1)}}(\{\neg E, C, D, \neg H, \neg G\}) =$
  $\{\neg E, C, D, \neg H, \neg G, A\} = T_{\mathcal{P}_{(1,1)}}\uparrow \omega$.
- Finally, $\bigcap T_{\mathcal{P}_\gamma}\uparrow \omega = \{\neg E, A\} = M_{\mathcal{P}}$

The Work Goes On

The next steps include defining the procedural semantics of programs for QMPT0 by a combination of SLD and SLI-resolution techniques. From the results obtained and the given examples, it is clear that one of the key issues is to reduce the size of the support $\mathrm{Sup}(\mathcal{P})$ (or $\bar{\mathrm{Sup}}(\mathcal{P})$) without prejudicing the completeness, in order to obtain a more feasible system in terms of implementation.

Thank You!

H. A. Blair and V. S. Subrahmanian.
Paraconsistent logic programming.
Theoretical Computer Science, 68:135–154, 1989.

W. A. Carnielli and M. E. Coniglio.
Paraconsistent Logic: Consistency, Contradiction and Negation, volume 40 of *Logic, Epistemology, and the Unity of Science*.
Springer, 2016.

W. A. Carnielli, M. E. Coniglio, and J. Marcos.
Logics of formal inconsistency.
In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 14, pages 1–93. Springer, 2nd edition, 2007.

M. E. Coniglio and L. H. C. Silvestrini.

An alternative approach for quasi-truth.
Logic Journal of the IGPL, 22(2):387–410, 2014.

C. V. Damásio and L. M. Pereira.
A survey of paraconsistent semantics for logic programs.
In *Reasoning with Actual and Potential Contradictions*, pages 241–320. Springer, 1998.

J. P. Delahaye and V. Thibau.
Programming in three-valued logic.
Theoretical Computer Science, 78(1):189–216, 1991.

M. C. Fitting.
Bilattices and the semantics of logic programming.
Journal of Logic Programming, 11:91–116, 1991.

M. Ginsber.

Multivalued logics: A uniform approach to reasoning in artificial intelligence.
Computational Intelligence, 4:265–316, 1988.

M. Kifer and V. S. Subrahmanian.
Theory of generalized annotated logic programming and its applications.
Journal of Logic Programming, 12(4):335–368, 1992.

T. C. Przymusinski.
Well-founded semantics coincides with three-valued stable semantics.
Fundamenta Informaticae, 13(4):445–463, 1990.

T. G. Rodrigues.
Sobre os fundamentos da programação lógica paraconsistente.

Master's thesis, IFCH - Universidade Estadual de Campinas, Brasil, 2010.

P. H. Schmitt.
Computational aspects of three-valued logic.
In *8th International Conference on Automated Deduction*, pages 190–198. Springer, 1986.