Is the Class of well-behaved Semantics so small?

Juan Carlos Nieves, Gabriel Cervantes
Universidad de las Americas Puebla, Departamento de Ingenieria en Sistemas Computacionales,
Sta. Catarina Martir, Cholula, 72820 Puebla, Mexico
sp205000@mail.udlap.mx - gcervantes@kim.ece.buap.mx

Abstract.

We introduce a new semantics (WFS*) that is well behaved, rational, and satisfies C3, different than WFS and WFS'. This broke a conjecture that stood for the last eight years. Since WFS* extends WFS, has suitable properties and furthermore it is polynomial time computable, we believe that it can be taken as a possible substitute of WFS.

1 Introduction

There has been a continued research towards the "best" properties that a semantics should verify. Too many efforts has been done and many semantics has been proposed for capturing the intended meaning of a program P. Schlipf [Schipf, 1992] and Dix [Dix, 1995] (among others researchers) have developed the fundamental theory of the principles of "suitable" semantics. The main properties that different proposed semantics satisfy were investigated by Dix [Apt and Bol, 1994]. Dix defines the notion of a well-behaved semantics and shows that the WFS semantics is the weakest well-behaved semantics. Schlipf [Levi, 1994] suggests to use this notion of well-behaved semantics as a standard for motivation and comparison of logic programming semantics. In 1992, Dix conjectures in [Dix, 1992] that the class of well-behaved sematics is too small. Later, in [Dix, 1995] he restates these conjectures, namely:

Conjecture 1 (Characterization of WFS, WFS$^+$ and WFS')
There are no well-behaved and rational semantics other that WFS, WFS$^+$ and WFS'.

Conjecture 2 (WFS$^+$)
WFS$^+$ is the only well-behaved and rational semantics satisfying Supraclassicality.

Proceedings of the Fifth ESSLLI Student Session
Catherine Pilière (editor)

Conjecture 3 (Characterization of WFS and WFS')
There are no well-behaved and rational semantics satisfying C3 other than WFS and WFS'.

These conjectures have been restated in a recent book (end of chapter 7) [Brewka, Dix and Konolige, 1997], where the authors call them "important representation theorems". In [Osorio, Dix and Zepeda, 2000] it is shown that the the first conjecture is false. The goal of our paper is to show that the third conjecture is also false.

The rest of this paper is organized as follows: In section 2 we provide some basic background on well-behaved semantics and others concepts. In section 3 we present our main results. Finally in last section we present our conclusions.[1]

2 Background

A signature $\mathcal{L}$ in a finite set of elements that we call atoms. A literal is an atom or its negation. Given a set of atoms $\{a_1, ..., a_n\}$, we write $\neg\{a_1, ..., a_n\}$ to denote the set $\{\neg a_1, ..., \neg a_n\}$.

A normal program clause C is a rule $a \leftarrow l_1, \ldots, l_n$, where a is an atom and each l_i is a literal. We also use the notation $a \leftarrow \mathcal{B}^+,\ \neg\mathcal{B}^-$, where $\mathcal{B}^+$ contains all the positive body atoms and $\mathcal{B}^-$ contains all the negative body atoms. We use $body(C)$ to denote $\mathcal{B}^+ \cup \neg\mathcal{B}^-$. A normal program is a finite set of normal program rules. If $body(C) = \emptyset$ then we say that the rule is a fact that we denote by $a \leftarrow$ or just as a. A (partial) interpretation based on a signature $\mathcal{L}$ is a disjoint pair of sets $\langle I_1, I_2\rangle$ such that $I_1 \cup I_2 \subseteq \mathcal{L}$. A partial interpretation is total if $I_1 \cup I_2 = \mathcal{L}$. A general semantics SEM is a function on $Prog_{\mathcal{L}}$ which associates with every program a partial interpretation.

Let $Prog_{\mathcal{L}}$ be the set of all normal propositional programs with atoms from $\mathcal{L}$. By $\mathcal{L}_P$ we understand it to mean the signature of P, i.e. the set of atoms that occur in P.

Definition 2.1 (SEM_{min}) *[Osorio, Dix and Zepeda, 2000]*
For any program P we define $HEAD(P) = \{a|\ a \leftarrow \mathcal{B}^+, \neg\mathcal{B}^- \in P\}$ the set of all head-atoms of P. We also define $SEM_{min}(P) = \langle P^{true}, P^{false}\rangle$, where $P^{true} := \{p|\ p \leftarrow\ \in P\}$, $P^{false} := \{p|\ p \in \mathcal{L}_P \backslash HEAD(P)\}$.

The main concept on which our semantis is based, is the concept of a *transformation* rule.

Definition 2.2 (Basic Transformation Rules)
A transformation rule is a binary relation on $Prog_{\mathcal{L}}$. The following transformation rules are called basic. *Let a program $P \in Prog_{\mathcal{L}}$ be given.*

[1] The interested reader could find a more complete version of this paper in : `http://www.udlap.mx/$\sim$josorio/jc/publications.html`

RED$^+$ ($\mathbf{R}^+$): *This transformation can be applied to P, if there is an atom a which does not occur in HEAD(P) and there is $b \leftarrow body \in P$ such that $\neg a \in body$. The transformation R^+, reduces a program P to $P_2 := (P \setminus \{b \leftarrow body\}) \cup \{b \leftarrow (body \setminus \{\neg a\})\}$.*

RED$^-$ ($\mathbf{R}^-$): *This transformation can be applied to P, if there is a rule $a \leftarrow \in P$ and there is $b \leftarrow body \in P$ such that $\neg a \in body$. R^- transforms P to $P_2 := P \setminus \{b \leftarrow body\}$.*

Sub (Sb): *This transformation can be applied to P, if P contains two clauses $a \leftarrow body_1$, and $a \leftarrow body_2$, where $body_1 \subseteq body_2$. Sb transforms P to the program where the clause $a \leftarrow body_2$ has been removed.*

Success (Sc): *Suppose that P includes a fact a and a clause $q \leftarrow body$ such that $a \in body$. Then we replace the clause $q \leftarrow body$ by $q \leftarrow body \setminus \{a\}$.*

Failure (Fa): *Suppose that P contains a clause $q \leftarrow body$ such that $a \in body$ and $a \notin HEAD(P)$. Then we erase the given clause.*

Equivalence (Eq): *Suppose P contains a rule C which has the same atom in its head and in its positive body. Then we remove this rule.*

Loop (Lp): *We say that P_2 results from P_1 by Lp_A if there is a set A of atoms such that for each rule $a \leftarrow body \in P_1$, if $a \in A$, then $body \cap A \neq \emptyset$, $P_2 := \{a \leftarrow body \in P_1 : body \cap A = \emptyset\}$, $P_1 \neq P_2$.*

By-Cases (B-C): *P_2 result from P if the following condition holds. Suppose b an atom. Let $P_3 := \{a \leftarrow \mathcal{B}^+, \neg(\mathcal{B}^- \setminus \{b\}) \mid a \leftarrow \mathcal{B}^+, \neg\mathcal{B}^- \in P\}$ and $P_4 := \{a \leftarrow \mathcal{B}^+ \setminus \{b\}, \neg\mathcal{B}^- \mid a \leftarrow \mathcal{B}^+, \neg\mathcal{B}^- \in P\}$. Let P_3' and P_4' programs resulting from P_3 and P_4 repectively by appling Success* and let $H := \{p \mid p \in P_3' \cap P_4'\}$ Then the transformation $By-Cases$ derives $P \cup \{a\}$ where $a \in H$ and $a \neq b$. In order to emphasis the role of a, b then we write $By-Cases_b^a$.*[2] [3]

We use $P_1 \rightarrow^T P_2$ for denote that we get P_2 by the transformation T from P_1.

Although these rules are not really *functions* on $Prog_{\mathcal{L}}$, they induce a set of operators on $\mathbf{Prog}_{\mathcal{L}}$ as we will show. An operator, denoted as *op*, is a function over the set of programs that transforms a program P to a program P^{op}. If the transformation can not be applied then the operator behaves as the identity function. On the other hand, if $P \neq P^{op}$, where P is program and *op* an operator, we say that *op* is executed.

[2] We use $P_1 \rightarrow^T P_2$ for denote that we get P_2 by the transformation T from P_1.

[3] T^* denote the reflexive and transitive closure of the relation T where T is any tranformation defined.

Given a program P and a list of operators ops, we define the application of ops to P, denoted as P^{ops}, as follows: $P^{[]} := P$, $P^{[op|ops]} := (P^{op})^{ops}$ (We use the notation $[\_ \mid \_]$ as in Prolog).

Definition 2.3 (Confluence) *A rewriting system is confluent if whenever $u \to^* x$ and $u \to^* y$ then there is a z such that $x \to^* z$ and $y \to^* z$.*[4]

Definition 2.4 (Noetherian) *A rewriting system is noetherian if there is no infinite chain $x_1 \to x_2 \to ... \to x_i \to x_{i+1} \to ...$, where for all i the elements x_i and x_{i+1} are different.*

Definition 2.5 (Locally confluent) *A rewriting system is locally confluent if whenever $u \to x$ and $u \to y$ then there is a z such that $x \to^* z$ and $y \to^* z$.*

Definition 2.6 (Partial Distribution) *[Osorio, Dix and Zepeda, 2000]*
A confluent LP-system $\mathcal{CS}$ *satisfies* partial distribution *if for every op, P and a such that $a \in HEAD(P^{op})$ and P^{op} is executed, then $(P \cup \{a\})^{op} = P^{op} \cup \{a\}$.*

Definition 2.7 (C3) *[Brewka, Dix and Konolige, 1997]*
A semantics SEM satisfies C3 iff for all the rules: $a \leftarrow body_i$, if $body_i$ is false in SEM then a is false in SEM.

Definition 2.8 (P reduced by M) *[Brewka, Dix and Konolige, 1997]*
Let P be a program and M be a set of literals. "P reduced by M" is the program $P^M := \{rule^M \mid rule \in P\}$ where $(A \leftarrow B_1, ..., B_n, \neg C_1, ..., \neg C_m)^M$ is defined by

$$(A \leftarrow B_1, ..., B_n, \neg C_1, ..., \neg C_m)^M = \begin{cases} delete & if \ \ \exists j \mid C_j \in M \ or \ \neg B_j \in M, \\ delete & if \ \ A \in M \ or \ \neg A \in M, \\ rule' & otherwise. \end{cases}$$

Here, $rule'$ stands for the clause "$A \leftarrow B'_1, ..., B'_{n'}, \neg C'_1, ..., \neg C'_{m'}$", where the set $\{B'_i \mid i \in I'\}$ (resp $\{\neg C'_i \mid i \in I'\}$) is just an enumeration of the set $\{B_i \mid i \in I\} \setminus M$ (resp. $\{\neg C_i \mid \in I\} \setminus M$).

We define the associated lenguage $\mathcal{L}_{P^M}$ for P^M to be $\mathcal{L}_P \setminus M$, i.e. $\mathcal{L}_{P^M}$ consists of all symbols occurring in P but different from those in M.

The condition of Relevance uses the notions of *dependencies_of* and *rel_rul* that are defined as follows[Brewka, Dix and Konolige, 1997]:

[4]Where $\to$ is a binary relation on a given set S. Let $\to^*$ be the reflexive, and transitive closure of $\to$. When $x \to^* y$ we say that x reduces to y

$dependencies\_of(X) := \{a : X \text{ depends on } a\}$,[5] and rel_rul(P,X) is the set of relevant rules of P with respect to X, i.e. the set of rules that contain an $a \in dependencies\_of(X)$.[6]

The follows definition is defined in [Brewka, Dix and Konolige, 1997].

Definition 2.9 (Well-behaved Semantics) *A well-behaved semantics SEM is semantics such that the following conditions are satisfied: Relevance, Reduction, PPE, Modularity, Isomorphy, Equivalence and Cut.*

Relevance *: The principle of Relevance states:*

$$SEM(P)(a) = SEM(rel\_rul(P, a))(a).$$

Reduction *: Let a set of literals $M \subseteq B_P \cup \neg B_P$ be given. The principle of Reduction states that $SEM(P \cup M) = SEM(P^M) \cup M$.*

PPE *: Let P be a propositional program and let an atom c occur only positively in P. Let $c \leftarrow body_1, ..., c \leftarrow body_n$ be all the rules of P with c in their heads. Any program clause of the form $a \leftarrow c, body$ can be replaced by rules*

$$\begin{array}{lcl} a & \leftarrow & body_1,\ body \\ & . & \\ & . & \\ & . & \\ a & \leftarrow & body_n,\ body \end{array}$$

Modularity *: Let $P = P_1 \cup P_2$, and for every atom a that appear P_2: $rel\_rul(P, a) \subseteq P_2$. The principle of Modularity is: $SEM(P) = SEM(P_1^{SEM(P_2)} \cup P_2)$.*

Isomorphy *: A semantics SEM satisfies Isomorphy, iff $SEM(\mathcal{I}(P)) = \mathcal{I}(SEM)(P))$ for all programs P and isomorphisms $\mathcal{I}$.*

Equivalence *: A semantics SEM allows Equivalence, iff rules of the form $a \leftarrow body$ where $a \in body$, can be eliminated without changing the semantics.*

Cut *: A semantics SEM satisfies Cut iff $a \in SEM(P)$ and $b \in SEM(P \cup \{a\})$ then $b \in SEM(P)$*

Definition 2.10 (Rationality) *A semantics SEM satisfies Rationality iff $a \in SEM(P)$ and $\neg b \notin SEM(P)$ then $a \in SEM(P \cup \{b\})$*

[5] The relation *depends on* is denined in [Brewka, Dix and Konolige, 1997].

[6] Let $dependencies\_of(\neg X) := dependencies\_of(X)$, and $rel\_rul(P, \neg X) := rel\_rul(P, X)$.

3 Declarative Semantics

Let $\mathcal{CS}_1$ be the rewriting system which contains exactly the transformations defined in the definition 2.2. Our main results are given in this section.

Theorem 3.1 (Confluent and Terminating) *The system $\mathcal{CS}_1$ is confluent and terminating. It induces a semantics that we call WFS^* and we define it as $WFS^*(P) = SEM_{min}(res_{\mathcal{CS}_1}(P))$.* [7]

Proof. Clearly, the system is noetherian. To prove that the system is confluent, note first that $\mathcal{CS}_1 \setminus \{B-C\}$ is known to be confluent. Thus we need to verify that B-C is locally confluent with respect to the other operators. The most interesting case is B-C *vs* Lp. The others caes are relative easy to prove confluence. We present only the proof of B-C *vs* Lp.

Suppose $P \rightarrow^{Lp_A} P_1$ and $P \rightarrow^{B-C_b^a} P_2$ where $P \neq P_1$ and $P \neq P_2$. Then we have two cases: 1.- $b \in A$ or 2.- $b \notin A$.

Case 1: First, is easy to verify that $P_2 \rightarrow^{Lp_A} P_3$ (where $P_2 \neq P_3$). Clearly $b \notin HEAD(P_1)$ and so we can take P_1 and apply $(R^+)^*$ and Sc^* to get P_4 and $a \in P_4$ (that is $P_1 \rightarrow^{(R^+)^*, Sc^*} P_4$). In the same way we can take P_3 and apply as well $(R^+)^*$ and Sc^* to get P_5. But due to the fact that $P_3 = P_1 \cup \{a\}$ and $a \in P_4$ then $P_4 = P_5$. Case 2: Here is not hard to check that Lp and $B-C$ commute because of the following reason: P is the form $P' \cup P''$ such that $P' \rightarrow^{B-C_b^a} \{a\} \cup P'$, $P' \subseteq P_1$, $P'' \rightarrow^{B-C_b^a} P''$, $P' \cap P'' = \phi$.

Thus in both cases $B-C$ and Lp are locally confluent with the help of R^+ and Sc). ■

Remark: The condition that $a \neq b$ in the definition of B-C is unweakable in the proof before, consider the program:

$$\begin{array}{lll} \text{a} & \leftarrow & \neg\, \text{a.} \\ \text{a} & \leftarrow & \text{a.} \end{array}$$

Then we would get the irreducible program $\{a \leftarrow \neg a.\}$. On the other hand we would derive also $\{a \leftarrow \quad\}$.

Definition 3.1 *Let P a normal program and M a set of consistent literals such that $M = M^+ \cup M^-$ where M^+ contains all the positive literals from M and M^- contains all the negative literals from M. Then $P \uplus M$ is defined as follows:*

$$P \uplus M = P \setminus \{l \leftarrow body \in P \mid l^c \in M^-\} \cup M^+$$

where l^c is define as follows: $l^c = \neg l$ if l is a positive atom, or $l^c = l$ if l is a negative atom.

[7] Where $res_{\mathcal{CS}_1}(P)$ is the normal form of P under the $\mathcal{CS}_1$ system.

NIEVlemma 3.1 *Let P a normal program and let M_1 and M_2 sets of literals. Then $P \uplus (M_1 \cup M_1) = (P \uplus M_1) \uplus M_2$.*

Proof. We first prove ($\subseteq$): Let $a \leftarrow body \in P \uplus (M_1 \cup M_1)$, then $\neg a \notin M_1^- \cup M_2^-$ and $a \leftarrow body \in P$. Then $\neg a \notin M_1^-$, therefore $a \leftarrow body \in P \uplus M_1$ by definition. Moreover we know that $\neg a \notin M_2^-$, then $a \leftarrow body \in (P \uplus M_1) \uplus M_2$.

Now we will prove ($\supseteq$): Let $a \leftarrow body \in (P \uplus M_1) \uplus M_2$, then $a \leftarrow body \in P \uplus M_1$ and $\neg a \notin M_2^-$ since $a \leftarrow body \in P \uplus M_1$ then $\neg a \notin M_1^-$. Therefore $\neg a \notin M_1^- \cup M_2^-$, then $a \leftarrow body \in P \uplus (M_1 \cup M_1)$. ■

NIEVlemma 3.2 *Let P a normal program and let M be a sets of literals and let l a literal such that $l^c, l \notin M$, then $(P \uplus M)^{\{l\}} = P^{\{l\}} \uplus M$.*

Proof. The proof is by induction w.r.t. the size of M. **Base case**: If $|M| = 0$ the proof is trivial now if $|M| = 1$. To prove $(P \uplus \{e\})^{\{l\}} = P^{\{l\}} \uplus \{e\}$. If e is a positive literal is direct. Now we consider the case when $e = \neg a$. We need to show that: $(P \uplus \{\neg a\})^{\{l\}} = P^{\{l\}} \uplus \{\neg a\}$: Let $b \leftarrow body \in (P \uplus \{\neg a\})^{\{l\}}$ then $b \neq a$ and $b \neq l$. Then $b \leftarrow body \in P^{\{l\}}$. Therefore $b \leftarrow body \in P^{\{l\}} \uplus \{\neg a\}$.

Inductive step: We know that $(P \uplus M)^{\{l\}} = P^{\{l\}} \uplus M$ is true when $|M| = K$ by induction hypothesis. Now we will prove when $|M| = K + 1$. Proof: $(P \uplus M)^{\{l\}} = (P \uplus (\{m\} \cup M \setminus \{m\}))^{\{l\}}$ by lemma 3.1 $(P \uplus (\{m\} \cup M \setminus \{m\}))^{\{l\}} = ((P \uplus \{m\}) \uplus M \setminus \{m\})^{\{l\}}$ by induction hypothesis and taking $(P \uplus \{m\})$ as a program $((P \uplus \{m\}) \uplus M \setminus \{m\})^{\{l\}} = (P \uplus \{m\})^{\{l\}} \uplus M \setminus \{m\}$ by induction hypothesis $(P \uplus \{m\})^{\{l\}} \uplus M \setminus \{m\} = (P^{\{l\}} \uplus \{m\}) \uplus M \setminus \{m\}$ by lemma 3.1 $(P^{\{l\}} \uplus \{m\}) \uplus M \setminus \{m\} = P^{\{l\}} \uplus (M \setminus \{m\} \cup \{m\})$ and $P^{\{l\}} \uplus (M \setminus \{m\} \cup \{m\}) = P^{\{l\}} \uplus M$ ■

NIEVlemma 3.3 *Let P a normal program and let A and M sets of literals such that $M \cap A = \emptyset$, then $(P^M)^A = P^{M \cup A}$.*

Proof. Is straightforward by definition of the reduction P^M. ■

Corollary 3.1 *Let P a normal program and M a set of literals, then $(P^M)^M = P^{M \cup M} = P^M$.*

Proof. The proof is direct. ■

NIEVlemma 3.4 *Let P a normal program and let l a positive literal, then $SEM(P^{\{l\}} \cup \{l\}) = SEM(P^{\{l\}}) \cup \{l\}$.*

Proof. Straightforward. ■

NIEVlemma 3.5 *Let P a normal program and let l a negative literal and $\neg l \in \mathcal{L}_P$, then $SEM(P^{\{\neg l\}}) = SEM(P^{\{\neg l\}}) \cup \{\neg l\}$.*

Proof. We know that $\neg l \in \mathcal{L}_P$ and by definition of P^M, $l \notin HEAD(P^{\{\neg l\}})$ Then $\neg l \in SEM(P^{\{\neg l\}})$ ■

Theorem 3.2 *The WFS^* semantics is well-behaved, rational and satisfies $C3$.*

Proof. (Sketch)

Equivalence: Follows straightforward by construction of our semantics.

Reduction: The proof is by induction w.r.t. the size of M. **Base case**: If $|M| = 0$ the proof is trivial. Let $|M| = 1$, then there are two cases, when M is a positive literal ($M = \{a\}$) and when M is a negative literal ($M = \{\neg a\}$). First case: $M = \{a\}$, then to prove: $SEM(P \uplus \{a\}) = SEM(P^{\{a\}}) \cup \{a\}$: We know that if $P_1 \rightarrow^{\mathcal{CS}_1} P_2$ then $SEM(P_1) = SEM(P_2)$ that $P \uplus \{a\} \rightarrow^{\{Sb,Sc,R^-\}^*} P^{\{a\}} \cup \{a\}$. Then $SEM(P \uplus \{a\}) = SEM(P^{\{a\}} \cup \{a\})$. Therefore by lemma 3.4, $SEM(P \uplus \{a\}) = SEM(P^{\{a\}}) \cup \{a\}$. Second case : $M = \{\neg a\}$, then by proof: $SEM(P \uplus \{\neg a\}) = SEM(P^{\{\neg a\}}) \cup \{\neg a\}$: Follows the idea of the first case we show that $P \uplus \{\neg a\} \rightarrow^{\mathcal{CS}_1} P^{\{\neg a\}}$. Given that $P \uplus \{\neg a\} \rightarrow^{\{R^+,Fa\}^*} P^{\{\neg a\}}$, then by lemma 3.5 $SEM(P \uplus \{\neg a\}) = SEM(P^{\{\neg a\}}) \cup \{\neg a\}$. **Induction step**: To prove: $SEM(P \uplus (M \cup \{e\})) = SEM(P^{(M \cup \{e\})}) \cup (M \cup \{e\})$: By lemma 3.1 $SEM(P \uplus (M \cup \{e\})) = SEM((P \uplus M) \uplus \{e\})$, then by induction hypothesis[8] $SEM((P \uplus M) \uplus \{e\}) = SEM((P \uplus M)^{\{e\}}) \cup \{e\}$, the by lemma 3.2 $SEM((P \uplus M)^{\{e\}}) \cup \{e\} = SEM(P^{\{e\}} \uplus M) \cup \{e\}$, by induction hypothesis $SEM((P^{\{e\}})^M) \cup M \cup \{e\}$ this is last equals. Therefore by lemma 3.3, it is equal to $SEM((P^{\{e\} \cup M}) \cup (M \cup \{e\})$

Relevance: Is possible to define the concept of relevance transformation. Then we prove that the reflexive, transitive clousure of the transformation system is relevant. Thus, the semantics induced by $\mathcal{CS}_1$ is relevant.

Cut and Rationality : By theorem 3.1 $\mathcal{CS}_1$ is confluent and terminating, moreover satisfies partial distribution[9] then by theorem 2 in [Osorio, Dix and Zepeda, 2000] WFS^* satisfies cut and rationality.

PPE: Let $P \rightarrow^{PPE} P'$. We need to prove $SEM(P) = SEM(P')$. The proof is by induction over the number of transformation steps applied to P to obtain its normal form. **Base case**: If P is in normal form then since P' does not allow the application of $B - C^*$. The result follows immediately. **Induction step**: Suppose that P reduces to its

[8] Taking $(P \uplus M)$ as a program.

[9] In [Osorio, Dix and Zepeda, 2000] shown that $\mathcal{CS}_1 \setminus \{B - C\}$ satisfies also partial distribution as the reader can verify.

normal form in n steps ($n > 0$). Let P_1 be obtained on the first step. Let $P_1 \rightarrow^{PPE} P_1'$, by induction hypothesis $SEM(P_1) = SEM(P_1')$ but also $SEM(P) = SEM(P_1)$. Is easy to check that exists P'' such that $P' \rightarrow^{\mathcal{CS}_1} P''$ and $P_1' \rightarrow^{\mathcal{CS}_1} P''$, thus $SEM(P') = SEM(P_1')$. So $SEM(P) = SEM(P_1) = SEM(P_1') = SEM(P')$, as we wanted to show.

Modularity: We already showed that our semantics satisfies Relevance, Reduction. Extended Cumulativity[Dix, 1995], our semantics also satisfies, as the reader could verify, then by Lemma 5.18 in [Dix, 1995] satisfies Modularity.

C3: Suppose that $\neg a \notin SEM(P)$ then we have two cases: 1) $a \in SEM(P)$ or 2) a is undefined. If $a \in SEM(P)$ then there is $a \leftarrow \alpha \in P$ such that α is not false in $SEM(P)$ Now if a is undefined then there exists $a \leftarrow \alpha' \in P$ such that $a \leftarrow \alpha \in res_{\mathcal{CS}_1}(P)$ and $\alpha \subseteq \alpha'$. But we know that α is undefined then α' is also undefined.

Isomorphy: We already showed that our semantics satisfies Relevance. Then, following the idea of Dix in [Brewka, Dix and Konolige, 1997] we can prove that, a non-trivial semantics that satisfies Relevance also satisfies Isomorphy.

■

We now show that WFS^* is different from WFS and WFS'. Consider the next program given in [Apt and Bol, 1994] (section 7.4)

$P := \{p \leftarrow \neg q \quad q \leftarrow \neg p \quad r \leftarrow p \quad r \leftarrow q\}$. Then $WFS(P) = \phi$ and $WFS^*(P) = \{r\}$. Note in addition that [Apt and Bol, 1994] presented the given example as a drawback of the WFS semantics because it can not derive r. Therefore WFS^* can also be considered as a proposal to improve WFS.

Moreover WFS^* is different to WFS', as the follows program shows: $P := \{a \leftarrow b,\ c. \quad a \leftarrow \neg b. \quad a \leftarrow \neg c. \quad b \leftarrow \neg c. \quad c \leftarrow \neg b.\}$ where $WFS^*(P) = \phi$ and $WFS'(P) = \{a\}$. Another semantics similar to WFS^* is introduced in [Osorio, Dix and Zepeda, 2000]. But this semantics is not well-behaved. A nice property of WFS^* is that it is polynomial time computable, however WFS' is **co-NP-compl**[Dix, 1995 - b].

4 Conclusion

We exhibit a semantics that is well behaved, rational, and satisfies C3, different than WFS and WFS'. This broke the conjecture that stood for the last eight years. Our result has three main implications: First, it shows that the class of well-behaved semantics is bigger than expected and so it opens future research to obtain its real characterization. Second, WFS* provides a

partial solution to the drawback of WFS as noted in [Apt and Bol, 1994][10]. Third, since WFS* extends WFS, has suitable properties and furthermore it is polynomial time computable. We believe that WFS^* can be taken as a possible substitute of WFS.

Bibliography

[Apt and Bol, 1994] Krzysztof R. Apt and Roland N. Bol. Logic Programming and Negation: A Survey. *Journal of Logic Programming*, 19-20:9–71, 1994.

[Arrazola, Dix and Osorio,1999] J. Dix J. Arrazola and Mauricio Osorio. Confluent term rewriting systems for non-monotonic reasoning. *Computación y Sistemas*, II(2-3):299–324, 1999.

[Brewka, Dix and Konolige, 1997] Gerd Brewka, Jürgen Dix, and Kurt Konolige. *Nonmonotonic Reasoning: An Overview.* CSLI Lecture Notes 73. CSLI Publications, Stanford, CA, 1997.

[Dix, 1992] Jürgen Dix. A Framework for Representing and Characterizing Semantics of Logic Programs. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR '92)*, pages 591–602. San Mateo, CA, Morgan Kaufmann, 1992.

[Dix, 1995] Jürgen Dix. A Classification-Theory of Semantics of Normal Logic Programs: II. Weak Properties. *Fundamenta Informaticae*, XXII(3):257–288, 1995.

[Dix, 1995 - b] Jürgen Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. An Overview. In A. Fuhrmann and Hans Rott (eds.), *Logic, Action and Information—Essays on Logic in Philosophy and Artificial Intelligence*, pages 241–327, De Gruyter, September 1995.

[Osorio, Dix and Zepeda, 2000] Mauricio Osorio J. Dix and Claudia Zepeda. A general theory of confluent rewriting systems for logic programming and its applications. *Annals of Pure and Applied Logic*, ?:?, 2000.

[Levi, 1994] Giorgio Levi. *Advances in logic programming theory.* Oxford:Clarendon Press, 1994.

[Schipf, 1992] John S. Schlipf. Formalizing a Logic for Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 5:279–302, 1992.

[10] We mentioned this issue in the previous section