Is the Class of well-behaved Semantics so small?

JUAN CARLOS NIEVES, GABRIEL CERVANTES
Universidad de las Americas Puebla, Departamento de Ingeniería en Sistemas Computacionales,
Sta. Catarina Martir, Cholula, 72820 Puebla, Mexico
sp205000@gmail.udlap.mx - gcevantes@kim.ece.uap.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 seman-
tics has been proposed for capturing the intended meaning of a program P.
Schlipf [Schlipf, 1992] and Dix [Dix, 1993] (among others researchers) have
developed the fundamental theory of the principles of “suitable” semantics.
The main properties that different proposed semantics satisfy were investigat-
ed 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 program-
ing semantics. In 1992, Dix conjectures in [Dix, 1992] that the class of
well-behaved semantics 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 Supraclass-
sicality.
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.

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 \( a \leftarrow B^+, B^- \), where \( B^+ \) contains all the positive body atoms and \( B^- \) contains all the negative body atoms. We use \( \text{body}(C) \) to denote \( B^+ \cup \neg B^- \). A normal program is a finite set of normal program rules. If \( \text{body}(C) = \emptyset \) 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 \( \text{SEM} \) is a function on \( \text{Prog}_{\mathcal{L}} \) which associates with every program a partial interpretation.

Let \( \text{Prog}_{\mathcal{L}} \) be the set of all normal propositional programs with atoms from \( \mathcal{L} \). By \( L_P \) 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 \( \text{HEAD}(P) = \{a \mid a \leftarrow B^+, \neg B^- \in P\} \) the set of all head-atoms of \( P \). We also define \( \text{SEM}_{\min}(P) = \langle p^{\text{true}}, p^{\text{false}} \rangle \), where \( p^{\text{true}} := \{p \mid p \leftarrow \in P\} \), \( p^{\text{false}} := \{p \mid p \in L_P \setminus \text{HEAD}(P)\} \).

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

Definition 2.2 (Basic Transformation Rules)

A transformation rule is a binary relation on \( \text{Prog}_{\mathcal{L}} \). The following transformation rules are called basic. Let a program \( P \in \text{Prog}_{\mathcal{L}} \) be given.

\[ \text{http://www.udlap.mx/$\sim$josorio/jc/publications.html} \]
**RED** (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 ← body ∈ P such that ¬a ∈ body. The transformation R reduces a program P to P₂ := (P \ {b ← body}) \ {b ← (body \ {¬a})}.

**RED** (R): This transformation can be applied to P, if there is a rule a ← P and there is b ← body ∈ P such that ¬a ∈ body. R transforms P to P₂ := P \ {b ← body}.

**Sub (Sb):** This transformation can be applied to P, if P contains two clauses a ← body₁, and a ← body₂, where body₁ ⊆ body₂. Sb transforms P to the program where the clause a ← body₂ has been removed.

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

**Failure (Fa):** Suppose that P contains a clause q ← body such that a ∈ body and a ∉ HEAD(P). Then we emuse 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 P₂ results from P₁ by LP if there is a set A of atoms such that for each rule a ← body ∈ P₁, if a ∈ A, then body ∩ A ≠ ∅, P₂ := {a ← body ∈ P₁ : body ∩ A = ∅}, P₁ ≠ P₂.

**By-Cases (B-C):** P₂ result from P if the following condition holds. Suppose b an atom. Let P₃ := {a ← B⁺, ¬(B⁻ \ {b}) | a ← B⁺, ¬B⁻ ∈ P} and P₄ := {a ← B⁺ \ {b}, ¬B⁻ | a ← B⁺, ¬B⁻ ∈ P}. Let P₃’ and P₄’ programs resulting from P₃ and P₄ respectively by applying Success* and let H := {p | p ∈ P₃’ ∩ P₄’} Then the transformation By-Cases derives P ∪ {a} where a ∈ H and a ≠ b. In order to emphasis the role of a, b then we write By-Cases₆.

We use P₁ →ᵀ P₂ for denote that we get P₂ by the transformation T from P₁.

Although these rules are not really functions on ProgL, they induce a set of operators on ProgL as we will show. An operator, denoted as op, is a function over the set of programs that transforms a program P₁ to a program P₀. If the transformation can not be applied then the operator behaves as the identity function. On the other hand, if P₁ ≠ P₀, where P₁ is program and op an operator, we say that op is executed.

---

²We use P₁ →ᵀ P₂ for denote that we get P₂ by the transformation T from P₁.
³ᵀ denotes the reflexive and transitive closure of the relation T where T is any transformation defined.
Given a program \( P \) and a list of operators \( \text{ops} \), we define the application of \( \text{ops} \) to \( P \), denoted as \( P^{\text{ops}} \), as follows: \( P[] := P \), \( P[^{\text{op}[\text{ops}]}] := (P^{\text{op}})^{\text{ops}} \). We use the notation \([- | \downarrow \) as in Prolog.

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

**Definition 2.4 (Noetherian)** A rewriting system is noetherian if there is no infinite chain \( x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_i \rightarrow x_{i+1} \rightarrow \ldots \), 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 \( u \rightarrow x \) and \( u \rightarrow y \) then there is a \( z \) such that \( x \rightarrow^* z \) and \( y \rightarrow^* z \).

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

**Definition 2.7 (C3)** [Brewka, Dix and Konolige, 1997] A semantics \( SEM \) satisfies \( C3 \) iff for all the rules: \( a \leftarrow \text{body}_i \), if \( \text{body}_i \) 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 \( P^M := \{\text{rule}^M | \text{rule} \in P\} \) where \( (A \leftarrow B_1, \ldots, B_n, \neg C_1, \ldots, \neg C_m)^M \) is defined by

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

Here, rule' stands for the clause “\( A \leftarrow B'_1, \ldots, B'_n, \neg C'_1, \ldots, \neg C'_m \)”, where the set \( \{B'_i | i \in I \} \) (resp \( \{\neg C'_i | i \in I \} \)) is just an enumeration of the set \( \{B_i | i \in I \} \setminus M \) (resp \( \{\neg C_i | i \in I \} \setminus M \) ).

We define the associated language \( \mathcal{L}_{P^M} \) 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 \textit{dependencies} of and \textit{rel. rule} that are defined as follows [Brewka, Dix and Konolige, 1997]:

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

192
dependencies of \(X\) := \{a \text{ : } X \text{ depends on } a\} \footnote{The relation \textit{depends on} is defined in \cite{Brewka, Dix and Konolige, 1997}.} and \(rel_{\text{rul}}(P, X)\) is the set of relevant rules of \(P\) with respect to \(X\), i.e. the set of rules that contain an \(a \in \text{dependencies of } X\) \footnote{Let \(\text{dependencies of } \neg X := \text{dependencies of } X\), and \(rel_{\text{rul}}(P, \neg X) := rel_{\text{rul}}(P, X)\).}.

The follows definition is defined in \cite{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:

\[
SEM(P)(a) = SEM(rel_{\text{rul}}(P, a))(a).
\]

**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 \(c \leftarrow \text{body}_1, \ldots, c \leftarrow \text{body}_n\) be all the rules of \(P\) with \(c\) in their heads. Any program clause of the form \(a \leftarrow c\), \text{body} can be replaced by rules

\[
a \leftarrow \text{body}_1, \text{body} \\
\vdots \\
a \leftarrow \text{body}_n, \text{body}
\]

**Modularity**: Let \(P = P_1 \cup P_2\), and for every atom \(a\) that appear \(P_2\):

\[
rel_{\text{rul}}(P, a) \subseteq P_2.
\]

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 \(a \leftarrow \text{body}\) where \(a \in \text{body}\), 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 \(-b \notin SEM(P)\) then \(a \in SEM(P \cup \{b\})\)
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 \( WFS^*(P) = SEM_{\text{min}}(\text{res}_{\mathcal{CS}_1}(P)) \).

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 cases are relative easy to prove confluence. We present only the proof of \( B - C \) vs \( Lp \).

Suppose \( P \rightarrow Lp A P_1 \) and \( P \rightarrow B - C^o P_2 \) 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 \( b \notin \text{HEAD}(P_1) \) 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 \( P' \rightarrow B - C^o \{a\} \cup P', \ P'' \subseteq P_1, \ P'' \rightarrow B - C^o P'', \ P' \cap P'' = \phi \).

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 unwearable in the proof before, consider the program:

\[
\begin{align*}
a &\leftarrow \neg a. \\
a &\leftarrow a.
\end{align*}
\]

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

**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:

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

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.

\[\text{Where } \text{res}_{\mathcal{CS}_1}(P) \text{ is the normal form of } P \text{ under the } \mathcal{CS}_1 \text{ system.}\]
**NIEVLemma 3.1** Let $P$ a normal program and let $M_1$ and $M_2$ sets of literals. Then $P \cup (M_1 \cup M_1) = (P \cup M_1) \cup M_2$.

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

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

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

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 $(P \cup \{e\})^{\{l\}} = P^{\{l\}} \cup \{e\}$. If $e$ is a positive literal is direct. Now we consider the case when $e = \neg a$. We need to show that: $(P \cup \{-a\})^{\{l\}} = P^{\{l\}} \cup \{-a\}$: Let $b \leftarrow \text{body} \in (P \cup \{-a\})^{\{l\}}$ then $b \neq a$ and $b \neq l$. Then $b \leftarrow \text{body} \in P^{\{l\}}$. Therefore $b \leftarrow \text{body} \in P^{\{l\}} \cup \{-a\}$.

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

**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 $SEM(P^{\{-l\}}) = SEM(P^{\{-l\}}) \cup \{-l\}$.

195
Proof. We know that \(l \in \mathcal{L}_P\) and by definition of \(P^M\), \(l \notin HEAD(P^{l^-})\) Then \(l \in SEM(P^{l^-})\) ■

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 \((M = \{-a\})\). First case: \(M = \{a\}\), then to prove: \(SEM(P \uplus \{a\}) = SEM(P[\{a\}] \cup \{a\})\): We know that if \(P_1 \rightarrow^C \{a\} \rightarrow \{a\} \rightarrow P_2\) then \(SEM(P_1) = SEM(P_2)\) \(P \uplus \{a\} \rightarrow^C \{a\} \rightarrow \{a\}\). Then \(SEM(P \uplus \{a\}) = SEM(P[\{a\}] \cup \{a\})\). Therefore by lemma \(3.3\) \(SEM(P \uplus \{a\}) = SEM(P[\{a\}] \cup \{a\})\). Second case: \(M = \{-a\}\), then by proof: \(SEM(P \uplus \{-a\}) = SEM(P^{l^-}) \cup \{-a\}\): Follows the idea of the first case we show that \(P \uplus \{-a\} \rightarrow^{C\uplus} P^{l^-}\). Given that \(P \uplus \{-a\} \rightarrow^{(R^+, F^\uplus)} P^{l^-}\), then by lemma \(3.3\) \(SEM(P \uplus \{-a\}) = SEM(P^{l^-}) \cup \{-a\}\).

Induction step: To prove: \(SEM(P \uplus (M \cup \{e\})) = SEM(P(M \cup \{e\}))\) \((M \cup \{e\})\): By lemma \(3.3\) \(SEM(P \uplus (M \cup \{e\})) = SEM(P(M \uplus \{e\}))\), then by induction hypothesis \(SEM((P \uplus M) \cup \{e\}) = SEM((P \uplus M) \cup \{e\})\), the by lemma \(3.2\) \(SEM((P \uplus M) \cup \{e\}) = SEM((P \uplus M) \cup \{e\})\) \((M \cup \{e\})\) \((M \cup \{e\})\) this is last equals. Therefore by lemma \(3.3\) it is equal to \(SEM((P \uplus M) \cup \{e\})\).

Relevance: Is possible to define the concept of relevance transformation. Then we prove that the reflexive, transitive closure of the transformation system is relevant. Thus, the semantics induced by \(CS_1\) is relevant.

Cut and Rationality: By theorem \(3.1\) \(CS_1\) is confluent and terminating, moreover satisfies partial distribution, then by theorem 2 in Osorio, Dix and Zepeda, 2000 WFS\(^\#\) satisfies cut and rationality.

PPE: Let \(P \rightarrow^{PPE} P^\prime\). We need to prove \(SEM(P) = SEM(P^\prime)\). 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 \(P^\prime\) does not allow the application of \(B \rightarrow C\). 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 \(CS_1 \setminus \{B \rightarrow C\}\) 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 $P' \rightarrow^{CSi} P''$ and $P'' \rightarrow^{CSi} P'$, thus $SEM(P') = SEM(P)$). 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 $\alpha$ is not false in $SEM(P)$ Now if $a$ is undefined then there exists $a \leftarrow \alpha' \in P$ such that $a \leftarrow \alpha \in res_{CSi}(P)$ and $\alpha \subseteq \alpha'$. But we know that $\alpha$ is undefined then $\alpha'$ 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)

$P := \{ p \leftarrow q \quad q \leftarrow \neg p \quad r \leftarrow p \quad r \leftarrow q \}$. 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:

$P := \{ a \leftarrow b, \quad c, \quad a \leftarrow \neg b, \quad a \leftarrow \neg c, \quad b \leftarrow \neg c, \quad c \leftarrow \neg b \}$ where $WFS^*(P) = \phi$ and $WFS'(P) = \{ a \}$. Another semantics similar to $WFS^*$ is introduced in [Usorio, 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-complete [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].

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


---

10We mentioned this issue in the previous section.