# $W_{s,c}$-Stable Semantics for propositional theories

**Mauricio Osorio, Juan Carlos Nieves**
Universidad de las Americas
CENTIA
Sta. Catarina Martir, Cholula, Puebla.
72820, México
josorio@mail.udlap.mx

**Abstract**

We define new semantics that extend the STABLE semantics in different directions. In particular, we introduce two semantics for general propositional theories that we call $w_s - stable$ and $w_c - stable$. Our semantics have the main properties: $w_s - stable$ and $w_c - stable$ are defined over arbitrary propositional theories.

They agree with the stable model semantics when the given theory has at least one stable model. Our proposed semantics are invariant over intuitionistic equivalent theories. Finally, intuitionistic consistent theories have at least one $w_s - stable$ model and one $w_c - stable$ model. Our results naturally generalize to universal theories.

**Keywords:** Logic programming, intuitionistic logic, stable models.

## 1 Introduction

Applications using logic programming are wide, just to mention a recent one, Kowalski and Sadri in [1] propose an approach to agents (KS-agents) within an extended logic programming framework. Grosof in [2] observe that, logic programs, as a core declarative knowledge representation, has wide current presence and major representational advantages compared to other rules approaches. The stable semantics is among the most important semantics for logic programming, and has proved to has appealing and enduring features [3].

Answer Set Programming (ASP) (Stable Logic Programming or A-Prolog) is the realization of much theoretical work on Nonmonotonic Reasoning and AI applications of Logic Programming in the last 12 years. It is based on view of program statements as constraints on the solution of a given problem. Subsequently, each model of the program encodes a solution to the program itself. For instance, an ASP program encoding a planing scenario has as many models as valid plans. This schema is similar to that underlying the applications

of SAT algorithms to AI, and in fact the range of applicability of these two techniques is similar. However, thanks to the inherent causal aspect of Answer Set semantics, we can represent default assumptions constraints, uncertainly and non determinism in a direct way. Several ASP systems are now available, among them are DeReS, dlv, smodels and XSB. Several others can be found through the library of Logic Programming Systems and Test Cases. These systems support provably correct inferences and are at least as fast and scalable as SAT checkers. These are exciting results for NMR community and they are attracting the attention of researchers from fields such as planning, product configuration, verification of distributed systems, combinatorial problems and logical cryptanalysis.

The term Answer Set Programming, that we use now, was coined by Vladimir Lifschitz. It nicely captures two key ideas behind the approach: solution or answers are sets and logic programming with the answer-set semantics ( a generalization of stable models semantics) is an instantiation of the general approach.

However, it is well known that the stable semantics usually is too strong and sometimes a program do not possess a stable model [4, 5, 6]. As an example, in [7], the authors show that STABLE has some problems in modeling partial-order programs [8, 5].

Some authors have even argued that STABLE does not even define the intended models [9]. From the point of view of structural properties, Dix has suggested the notion of well-behaved semantics [6]. It turns out that STABLE is not well-behaved [6]. Some proposals to improve the stable semantics can be found in [5, 10]. It is also possible that there is not a best semantics as [11] comments.

In this paper we propose a solution to one problem of stable: That some programs do not have stable models. We introduce two semantics for general propositional theories that we call $w_s - stable$ and $w_c - stable$. Our semantics have the main properties: First, $w_s - stable$ and $w_c - stable$ are defined over arbitrary propositional theories. Second, they agree with the stable model semantics when the given theory has at least one stable model. Third, our proposed semantics are invariant over intuitionistic equivalent theories. Finally, intuitionistic consistent theories have at least one $w_s - stable$ model and one $w_c - stable$.

Our results naturally generalize to universal theories. The reason of this claim is because in logic programming is standard to work on the grounded form of a program.

Our paper is structured as follows: In section 2 we introduce our background. In section 3 we introduce our proposed semantics. Finally, in section 4 we present our conclusions.

## 2 Background

A signature $\mathcal{L}$ in a finite set of elements that we call atoms. By $\mathcal{L}_P$ we understand it to mean the signature of P, i.e. the set of atoms that occurs in P. A

literal is an atom or the negation of an atom $a$ that we denote as $\neg a$. 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 theory is built up using the constants $\wedge, \vee, \rightarrow, \neg$ and atoms (from $\mathcal{L}$). A disjunctive clause is a clause of the form: $a_1 \vee a_2 \vee ... \vee a_n \leftarrow l_1, l_2, ..., l_m$[1] where $n \geq 1, m \geq 0$, every $a_j$ is an atom and every $l_i$ is a literal. Sometimes, we denote a clause $C$ by $\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$, where $\mathcal{A}$ contains all the head atoms, $\mathcal{B}^+$ contains all the positive body atoms and $\mathcal{B}^-$ contains all the negative body atoms. We also use $body(C)$ to denote $\mathcal{B}^+ \cup \neg\mathcal{B}^-$. When $\mathcal{A}$ is a singleton set, the clause can be regarded as a normal clause. A definite clause [12] is a normal clause with $\mathcal{B}^- = \emptyset$. A disjunctive program is a finite set of disjunctive clauses. A normal program is built up with normal clauses. A definite program is a set of definite clauses [12]. Given a signature $\mathcal{L}$, we write $Prog_{\mathcal{L}}$ to denote the set of all programs defined over $\mathcal{L}$.

Given a theory $T$ defined over the signature $\mathcal{L}$ we define an interpretation over $T$ as a set $P \cup \neg N$ such that $P \cap N = \phi$ and $P \cup N = \mathcal{L}$.

A semantics is a mapping that associated to each theory $T$ a set of interpretations. Given two theories $T_1$ and $T_2$ we define $T_1 \equiv_L T_2$ iff for all formula $w$, $T_1 \vdash_L w$ iff $T_2 \vdash_L w$ where $\vdash_L$ is the provability relation in the same logic $L$.

If the set of interpretations that the semantics associates to $P_1$ is the same as the set of interpretations that the semantics associates to $P_2$, then we denote this by $P_1 \equiv_{sem} P_2$.

## 2.1 Stable Models

The Gelfond-Lifschitz transformation (GL-transformation) of a logic program $P$ w.r.t. an interpretation $M$ is obtained from $P$ by deleting

(i) each rule that has a negative literal $\neg B$ in its body with $B \in M$, and

(ii) all negative literals in the bodies of the remaining rules.

Clearly the program $GL(P, M)$ resulting from applying the GL-transformation is negation-free, so $GL(P, M)$ has at least a model. $M$ is a stable model iff $M$ is a minimal model of $GL(P, M)$.

**Definition 1 (Equivalence Transformation,[13])** *Given a semantics SEM, a program transformation $\mapsto$ is a SEM-equivalence transformation iff for all P, P' with $P \mapsto P'$: SEM(P) = SEM(P').*

The following transformations are defined in [14, 15] and generalize the corresponding definitions for normal programs.

**Definition 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] This clause represents the formula $l_1 \wedge l_2 ... \wedge l_m \rightarrow a_1 \vee a_2 \vee ... \vee a_n$.

**RED$^+$ ($\mathbf{R}^+$):** *Replace a rule* $\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$ *by* $\mathcal{A} \leftarrow \mathcal{B}^+, \neg(\mathcal{B}^- \cap HEAD(P))$.

**RED$^-$ ($\mathbf{R}^-$):** *Delete a clause* $\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$ *if there is a clause* $\mathcal{A}' \leftarrow$ *true such that* $\mathcal{A}' \subseteq \mathcal{B}^-$.

**Subsumption(Sb):** *Delete a clause* $\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$ *if there is another clause* $\mathcal{A}_1 \leftarrow \mathcal{B}_1^+, \neg\mathcal{B}_1^-$ *such that* $\mathcal{A}_1 \subseteq \mathcal{A}$, $\mathcal{B}_1^+ \subseteq \mathcal{B}^+$, $\mathcal{B}_1^- \subseteq \mathcal{B}^-$.

**Taut (Ta):** *Delete a rule* $\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$ *with* $\mathcal{A} \cap \mathcal{B}^+ = \phi$.

**Failure (F):** *Suppose that* $P$ *includes an atom* $a$ *and a clause* $q \leftarrow Body$ *such that* $a \notin HEAD(P)$ *and* $a$ *is a positive literal in* $Body$. *Then we erase the given clause.*

**Contradictions (Cn):** *Delete a rule* $\mathcal{A} \leftarrow \mathcal{B}^+, \neg\mathcal{B}^-$ *with* $\mathcal{B}^+ \cap \mathcal{B}^- \neq \phi$.

**Example 3 (Transformation)**
*Let* $\mathcal{L} = \{a, b, c, d, e\}$ *and let* $P$ *be the program:*

$\mathtt{a} \vee \mathtt{b} \leftarrow \mathtt{c}, \neg\mathtt{c}, \neg\mathtt{d}.$
$\mathtt{a} \vee \mathtt{c} \leftarrow \mathtt{b}.$
$\mathtt{c} \vee \mathtt{d} \leftarrow \neg\mathtt{e}.$
$\mathtt{b} \leftarrow \neg\mathtt{c}, \neg\mathtt{d}, \neg\mathtt{e}.$

*then* $HEAD(P) = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}\}$.
*We can apply* **RED$^+$** *to get the program* $P_1$*:*

$\mathtt{a} \vee \mathtt{b} \leftarrow \mathtt{c}, \neg\mathtt{c}, \neg\mathtt{d}.$
$\mathtt{a} \vee \mathtt{c} \leftarrow \mathtt{b}.$
$\mathtt{c} \vee \mathtt{d} \leftarrow true.$
$\mathtt{b} \leftarrow \neg\mathtt{c}, \neg\mathtt{d}, \neg\mathtt{e}.$

*If we apply* **RED$^+$** *again, we get program* $P_2$*:*

$\mathtt{a} \vee \mathtt{b} \leftarrow \mathtt{c}, \neg\mathtt{c}, \neg\mathtt{d}.$
$\mathtt{a} \vee \mathtt{c} \leftarrow \mathtt{b}.$
$\mathtt{c} \vee \mathtt{d} \leftarrow true.$
$\mathtt{b} \leftarrow \neg\mathtt{c}, \neg\mathtt{d}.$

*Now, we can apply* **SUB** *to get program* $P_3$*:*

$\mathtt{a} \vee \mathtt{c} \leftarrow \mathtt{b}.$
$\mathtt{c} \vee \mathtt{d} \leftarrow true.$
$\mathtt{b} \leftarrow \neg\mathtt{c}, \neg\mathtt{d}.$

Let **Dsuc** be the natural generalization of **suc** to disjunctive programs, formally:

**Definition 4 (Dsuc,[16])**
*Suppose that* $P$ *is a program that includes a fact* $a \leftarrow true$ *and a clause* $\mathcal{A} \leftarrow Body$ *such that* $a \in Body$. *Then we replace this clause by the clause* $\mathcal{A} \leftarrow Body \setminus \{a\}$.

**Definition 5 ($\mathcal{CS}_2$)**
*Let* $\mathcal{CS}_2$ *be the rewriting system based on the transformations* {*Sb*, $R^+$, $R^-$, *Dsuc, F, Ta, Cn* }.

## 2.2 Intuitionistic logic

Intuitionistic logic is based on the concept of proof. Given a property $P$ and an element $E$, intuitionistic logic does not accept the decidability whether $E$ has the property $P$ or not. Due to Brouwer, intuitionistic logic must exhibit the object of our pertained proof, or at least, indicates a method by which one could in principle find our object. Valid formulas in intuitionistic logic, will be also valid in classical logic. For intuitionistic logic, the concept of truth in not used for explain the meaning and the use of the logical constants or quantifiers. We use the concept of *proof*. We denote $\vdash_I$ to denote provability in intuitionistic logic. Ten axioms are defined in intuitionistic logic:

- $\mathcal{A} \rightarrow (\mathcal{B} \rightarrow \mathcal{A})$
- $(\mathcal{A} \rightarrow (\mathcal{B} \rightarrow \mathcal{C})) \rightarrow ((\mathcal{A} \rightarrow \mathcal{B}) \rightarrow (\mathcal{A} \rightarrow \mathcal{C}))$
- $\mathcal{A} \wedge \mathcal{B} \rightarrow \mathcal{A}$
- $\mathcal{A} \wedge \mathcal{B} \rightarrow \mathcal{B}$
- $\mathcal{A} \rightarrow (\mathcal{B} \rightarrow (\mathcal{A} \wedge \mathcal{B}))$
- $\mathcal{A} \rightarrow (\mathcal{A} \vee \mathcal{B})$
- $\mathcal{B} \rightarrow (\mathcal{A} \vee \mathcal{B})$
- $(\mathcal{A} \rightarrow \mathcal{C}) \rightarrow ((\mathcal{B} \rightarrow \mathcal{C}) \rightarrow (\mathcal{A} \vee \mathcal{B} \rightarrow \mathcal{C}))$
- $(\mathcal{A} \rightarrow \mathcal{B}) \rightarrow ((\mathcal{A} \rightarrow \neg\mathcal{B}) \rightarrow \neg\mathcal{A})$
- $\neg\mathcal{A} \rightarrow (\mathcal{A} \rightarrow \mathcal{B})$[2]

**Lemma 6** *In intuitionistic logic every consistent theory has a complete and consistent extension*

**Proof.** Let $T$ be a consistent theory, and $T_1$ a maximal consistent theory of $T$, then $T_1$ is complete. The proof of this fact is by contradiction. Suppose $T_1$ is maximal consistent but it is not complete. Then, there is formula $\alpha$ such that $T_1 \nvdash_I \alpha$ and $T_1 \nvdash_I \neg\alpha$. Since $T_1$ is maximal consistent, $T_1 \cup \{\alpha\} \vdash_I \neg\alpha$. But then $T_1 \vdash_I \alpha \rightarrow \neg\alpha$ and so $T_1 \vdash_I \neg\alpha$. ■

## 2.3 Characterization of stable semantics based on intuitionist logic

Pearce showed the following: a formula is entailed by a program in the stable model semantics if and only if it belongs to every intuitionistically complete and consistent extension of the program formed by adding only negated atoms [17].

[2] If we replace this axiom schema by $\neg\neg A \rightarrow A$ then we obtain classical logic.

### 2.4 Transformation of Programs

The following lemma is well known for normal programs [18], and recently it has been generalized to disjunctive programs [19].

**Lemma 7 (STABLE is closed under $\mathcal{CS}_2$ transformations,[19])** *Let $P_1$ and $P_2$ two programs related by any transformation in $\mathcal{CS}_2$. Then $P_1$ and $P_2$ have the same STABLE models.*

The following result (taken from [19])suggests that it makes sense to reduce a program by $\mathcal{CS}_2$, because this reduction can be computed efficiently.

**Lemma 8 ($\mathcal{CS}_2$ is polynomial time computable, [19])** *Let $P$ be a program and $P_1$ a reduced form of $P$ under $\mathcal{CS}_2$ (namely, that $P_1$ is obtained from $P$ by a sequence of reductions from $P$ and $P_1$ is already irreducible by $\mathcal{CS}_2$). Then $P_1$ is obtained in polynomial time computable with respect to the size of $P$.*

## 3 A proposal for a new semantics

First of all, let us consider some definitions:

**Definition 9** *Let $P$ be a theory. We denote $A$ an appendix of $P$ any set of literals over the signature $\mathcal{L}$.*

**Definition 10** *Given an appendix $I$ of $P$,we use $Positive(I)$ to denote the set of positive literals in $I$.*

**Definition 11** *Given a theory $P$, we define two different partial order among appendices of $P$ as follows: $I \leq_s J$ iff $Positive(I) \subseteq Positive(J)$, another partial order $I \leq_c J$ is defined iff $\mid Positive(I) \mid \ \leq \ \mid Positive(J) \mid$*

**Definition 12** *We say that a theory $P$ is literal-complete w.r.t. intuitionistic logic iff $\forall a \in \mathcal{L}_p$, $P \vdash_I a$ or $P \vdash_I \neg a$.*

**Definition 13** *We say that a theory $P$ is consistent w.r.t. intuitionistic logic $\mathcal{I}$ iff there is no formula $\alpha$ such that $P \vdash_I \alpha$ and $P \vdash_I \neg\alpha$*

**Definition 14** *Given a program $P$ we define a $\mathcal{L}-$completion of $P$ as $(P, I')$ such that $P \cup I^{'} = P^{'}$ where $I^{'}$ is an appendix of $P$, and $P^{'}$ is literal complete and consistent with respect to intuitionistic logic.*

**Definition 15** *Given $\mathtt{x} \in \{s, c\}$ and a program $P$ we define a suitable $\mathcal{L}_{\mathtt{x}}-$completion of $P' := (P, I')$ of $P$ iff $(P, I')$ is a $\mathcal{L}_{\mathtt{x}}-$completion of $P$ and for any other $\mathcal{L}_{\mathtt{x}}-$completion $P'' := (P, I'')$ of $P$ it is false that $I^{''} \leq_x I^{'}$.*

**Definition 16** *Given $\mathtt{x} \in \{s, c\}$, A $w_{\mathtt{x}}-$stable model for $P$ is the set of provable literals in a suitable $\mathcal{L}_{\mathtt{x}}-$completion $P$.*

The following results show that our proposal semantics satisfies suitable properties with respect to the original stable semantics. Hence, it is an adequate generalization of the above semantics.

**Theorem 17** *If $P$ admits stable models, then $M$ is a $w_s - stable - model$ iff $M$ is a $stable - model$.*

**Proof.** Is $P$ admits a stable model said $M$ then, by [17], $P$ has a *suitable* $\mathcal{L}_s - completion\ P$ using an appendix $I$ with no positive literals. Therefore $Positive(I) = \emptyset$, and so there is no *suitable* $\mathcal{L}_s - completion\ P$ that includes positive literals. ■

**Corollary 18** *For stratified normal programs,*

$$w_s - stable - models(P) = stratified - models(P).$$

**Proof.** If $P$ is a stratified program, then it has exactly one *stable* model $M$ which corresponds with the stratified model. The conclusion follows immediately from the last theorem. ■

**Theorem 19** *Every consistent theory has at least one $w_s - stable - model$.*

**Proof.** If $T$ is a consistent theory, then by lemma 6 it has a complete and consistent extension $T'$. Let $I := \{l | l\ is\ a\ literal\ and\ T' \vdash_I l\}$. So $(T, I')$ is a $\mathcal{L} - completion$ of $T$. Therefore $T$ has a suitable $\mathcal{L}_s - completion$. Hence $T$ has $w_s - stable$ model. ■

**Corollary 20** *Every disjunctive program has at least one $w_s - stable - model$.*

**Proof.** Just note that a disjunctive program is consistent. ■

**Theorem 21** *Given two programs such that $P_1 \equiv_I P_2$ then $P_1 \equiv_{w_s - stable} P_2$*

**Proof.** It is enough of show that if $P_1 \cup I_1$ is a *suitable* $\mathcal{L}_s - completion\ of\ P_1$ then $P_2 \cup I_1$ also is a *suitable* $\mathcal{L}_s -\ completion$ for $P_2$. First, note that $P_2 \cup I_1$ is literal complete, for hypothesis, $P_1 \equiv_I P_2$ then $P_1 \cup I_1 \equiv_I P_2 \cup I_1$. Given that $P_1 \cup I_1$ is a $\mathcal{L}_s - completion\ of\ P_1$, then $P_2 \cup I_1$ is a $\mathcal{L}_s - completion\ of\ P_2$.

Moreover, $P_2 \cup I_1$ is a suitable completion, for if not, $\exists\ I^{'}$ such that $I^{'} \leq_s I_1$ and also $P_1 \cup I^{'}$ would be a *suitable* $\mathcal{L}_s - completion\ of\ P_1$. Contradiction. ■

**Theorem 22** *Let $P_1$ be a logical program. Given any transformation $T \in \{R^-, Sb, Dsuc, Ta, Cn\}$ then $T$ is a $w_s - stable - equivalence\ transformation$.*

**Proof.** By cases: For $Sc$ let $\{\ a, \alpha \wedge a \rightarrow \beta\} \subseteq P_1$ [3]. It is enough prove that $\{a, \alpha \wedge a \rightarrow beta\} \equiv_I \{a, \alpha \rightarrow \beta\}$[4]. So $a, \alpha \rightarrow (a \rightarrow \beta) \vdash_I \alpha \rightarrow \beta$ and

[3] We use $\alpha, \beta$ ... to denote conjunction of literals in the body of any clause and the disjunction of atoms in the head of any clause

[4] $\alpha \wedge a \rightarrow b$ and $\alpha \rightarrow (a \rightarrow b)$

$\alpha \rightarrow \beta, a \vdash_I a \rightarrow (\alpha \rightarrow \beta)$. The result follows immediately. For $Ta$ : Then $P_2 = P_1 \setminus \{a \rightarrow (l_1 \rightarrow (l_2 \rightarrow (....(l_n \rightarrow a)....)$. So $a \wedge \alpha \vdash_I a$. For $R^-$, let $P_1 = P \cup \{b, \quad \alpha \rightarrow (\neg b \rightarrow a)\}$, then $P_2 = P \cup \{b \quad \}$. Given that $a \rightarrow (\alpha \rightarrow a)$ in an axiom, $\alpha \rightarrow a, \alpha \vdash_I a$ follows. For $Sb$, the proof straightforward follows from $\vdash_I \neg b \rightarrow (b \rightarrow a)$ and two successive applications of *Modus ponens.* ■

**Remark 23** *The reader could prove directly that the last result can be also be proved if we change* $I \leq_s J$ *by* $I \leq_c J$

**Lemma 24** *Let* $P$ *a program. Then* $M$ *is a* $w_c - stable$ *of* $P$ *then* $M$ *is a* $w_s - stable$ *model.*

**Proof.** Straightfoward. ■

**Remark 25** *Note that the converse is not true as it is shown in the following example:*
*Let* $P$ *the following program:*

$f \leftarrow e, \neg f$
$d \leftarrow c$
$d \leftarrow \neg c$
$c \leftarrow d, \neg e$

*Note that* $M_1 := \{\neg e, d, \neg f, c\}$ *and* $M_2 := \{e, f, \neg c, d\}$ *are the* $w_s - stable$ *models of* $P$. *However, only* $M_1$ *is a* $w_c - stable$ *model of* $P$.

Due to lack of space, we present only two examples where the stable models semantics has no models but there are $w_{\mathbf{s}} - stable - models$.

**Example 26** *Let* $P$ *be the program:*

$a \leftarrow b$
$b \leftarrow \neg b$
$b \leftarrow a$

*then the unique* $w_{\mathbf{s}} - stable$ *is* $\{a, b\}$.

**Example 27** *Let* $P$ *be the program:*

$p \leftarrow \neg p$
$a \leftarrow b$
$b \leftarrow a$
$h \leftarrow \neg d$

then the unique $w_{\mathbf{s}} - stable$ is $\{p, h\}$.

# 4 Conclusion

We defined two new semantics that generalized the stable semantics such that for every (stable) consistent disjunctive program our new semantics agree with the original stable semantics. However our new semantics also give a meaning to programs where the original stable semantics is inconsistent. Furthermore, our

new semantics can give a meaning to any propositional theory. Our approach is based on intuitionistic logic and open new lines of research. Our theorems 17, 19, 21 and 22 show that our semantics is an adequate generalization of the stable semantics.

## Acknowledgments

This research is sponsored by the Mexican National Council of Science and Technology, CONACyT (project 35804-A).

## References

[1] Sadri Kowalsky. Towards a unified agent architecture that combines rationality with reactivity. In *Logic in databases, Intl. Workshop LID 96, LNCS 1154*, pages 137–149. Berlin, 1996.

[2] B. Grosof. Applications of declarative logic programs to multi-agent e-commerce: Opportunities ans challenges. In Toni Rochefoert, Sadri, editor, *Multi-Agent Systems in Logic Programming*. 1999.

[3] D. Pearce. Back and forth semantics for normal, disjuntive and extended logic program. In *APPIA-GULP-PRODE*. 1998.

[4] Jia-Huai You and Li-Yan Yuan. A three-valued semantics for deductive databases and logic programs. *Journal of Computer and System Sciences*, 49(2):334–361, 1994.

[5] J. Dix, Mauricio Osorio, and Claudia Zepeda. A general theory of confluent rewriting systems for logic programming and its applications. *Annals of Pure and Applied Logic*, 108(1-3):153–188, 2001.

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

[7] Mauricio Osorio and Bharat Jayaraman. Aggregation and negation-as-failure. *New generation computing*, 17(3):255–284, 1999.

[8] Bharat Jayaraman Mauricio Osorio and David Plaisted. Theory of partial-order programming. *Science of Computer Programming*, 34(3):207–238, 1999.

[9] Teodor Przymusinski. Well-founded completions of logic programs. In *Proceedings of the Eigth International Logic Programming Conference, Paris, France*, pages 726–744, Cambridge, Mass., July 1991. MIT Press.

[10] Jürgen Dix and Martin Müller. Partial Evaluation and Relevance for Approximations of the Stable Semantics. In Z.W. Ras and M. Zemankova, editors, *Proceedings of the 8th Int. Symp. on Methodologies for Intelligent Systems, Charlotte, NC, 1994*, LNAI 869, pages 511–520, Berlin, 1994. Springer.

[11] P. M. Dung. On the relations between stable and wellfounded semantics of logic programs. *Theoretical Computer Science*, 105:7–25, 1992.

[12] John W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, 1987. 2nd edition.

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

[14] Stefan Brass and Jürgen Dix. Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. *Journal of Logic Programming*, 32(3):207–228, 1997. (Extended abstract appeared in: Characterizations of the Stable Semantics by Partial Evaluation *LPNMR, Proceedings of the Third International Conference, Kentucky*, pages 85–98, 1995. LNCS 928, Springer.).

[15] Gerhard Brewka and Jürgen Dix. Knowledge representation with logic programs. Technical report, Tutorial Notes of the 12th European Conference on Artificial Intelligence (ECAI '96), 1996. Also appeared as Technical Report 15/96, Dept. of CS of the University of Koblenz-Landau. Will appear as Chapter 6 in *Handbook of Philosophical Logic*, 2nd edition (1998), Volume 6, Methodologies.

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

[17] David Pearce. Stable inference as intuitionistic validity. *Logic Programming*, 38:79–91, 1999.

[18] Stefan Brass, Ulrich Zukowski, and Burkhard Freitag. Transformation-based bottom-up computation of the well-founded model. In Jürgen Dix, Luis Moniz Pereira, and Teodor C. Przymusinski, editors, *Non-Monotonic Extensions of Logic Programming (NMELP'96)*, number 1216 in LNAI, pages 171–201. Springer, 1997.

[19] Mauricio Osorio, J. C. Nieves, and Chis Giannella. Useful transformations in answer set programming. In Alessandro Provetti and Tran Cao Son, editors, *Proceedings of the AAAI 2001 Spring Symposium Series*, pages 146–152. AAAI press, Stanford, E.U., 2001.