# Three-Valued Semantics for Extended Logic Programs

**Piero A. Bonatti and Laura Giordano**
*Dipartimento di Informatica, Università di Torino*
*Corso Svizzera 185, I-10149 Torino, Italy*
email: {bonatti,laura}@di.unito.it tel: +39-11-7429 111

**Abstract**

In [4], the semantics of monotonic (i.e. `not`-free) extended logic programs (ELPs) has been rephrased in three-valued logic for two purposes: achieving tractable reasoning with incomplete information and understanding the relationships between the existing semantics and many-valued logics. In this paper, we generalize this approach to unrestricted ELPs. We obtain a unifying view of many formalisms, including the answer set semantics, the well-founded semantics, generalized stable models (as in [11]), default logic, autoepistemic logic (AEL) and some of its variants (three-valued AEL and Schwartz's reflexive AEL). Our framework highlights surprising similarities between previously unrelated formalisms, such as TMS's with dependency directed backtracking, the WFSX semantics by Alferes and Pereira, and reflexive AEL. Moreover, we obtain very interesting new semantics, which make it possible to solve many hard benchmark problems with a substantial gain in elegance and efficiency.

KEYWORDS: Three-valued Logic, Negation as Failure, Incomplete Information

## 1 Introduction

Research on extended logic programs (ELP's) has been focussed on the semantics of default negation, `not`, and on its interplay with explicit negation, $\neg$ (cf. [16]). While several semantics have been proposed for `not`, there is little variety of meanings for the `not`-free fragment of the language, hereafter called *monotonic*.

In [4], the semantics of monotonic ELP's has been rephrased in three-valued logic, for different purposes. The first goal was improving our understanding of the relationships between ELP's and multiple valued logics, which are not completely clear, despite numerous superficial similarities. The second goal was a purely semantic account of ELP's, whose meaning is often defined through pervasive syntactic manipulations that transform negative literals into new atoms, both within

programs and within interpretations. The final goal was finding a more powerful tractable semantics for ELP, capable of expressing and using incomplete knowledge without resorting to more complex forms of reasoning involving combinatorial search or noncomputable inferences. The approach of [4] tackles all the above problems and leads to a unifying view of different semantics for monotonic ELP's. The basic idea is considering a `not`-free program rule as a formula

$$(\ldots(L_0 \leftarrow L_1) \leftarrow \ldots) \leftarrow L_n\,.$$

By interpreting $\leftarrow$ as one of the standard three-valued implications illustrated in Fig. 1, different semantics can be captured.

It is interesting to see how this approach behaves when we introduce default negation through standard nonmonotonic constructions. This is the purpose of the present paper. For the sake of generality we shall model different possible meanings of `not` through a general construction (stable classes) that generalizes stable and well-founded semantics. By tuning two parameters, that is, the truth table of implication and the program transformation involved in the nonmonotonic construction (one of which is the familiar Gelfond-Lifschitz transformation), we obtain a unifying view of many formalisms and highlight surprising similarities between previously unrelated formalisms. We shall prove that Lukasiewicz's implication induces some very interesting new semantics, which make it possible to solve many hard benchmark problems with a substantial gain in elegance and efficiency. Moreover, a well-founded version of Lukasiewicz's semantics constitutes a natural semantics for Reason Maintenance, more powerful and more efficient than the skeptical belief revision model of [18].

# 2 Preliminaries

## 2.1 Three-Valued Logic

Three-valued interpretations are mappings from the set of ground atoms into the set $\{F, U, T\}$. As usual, three-valued interpretations will be represented as consistent sets of ground literals, so that $A$ is $T$ (resp. $F$) in $I$ iff $A \in I$ (resp. $\neg A \in I$). In the literature there is general agreement about the meaning of $\neg$, while the meaning of implication is controvertial. Three of the major proposals are recalled in Fig. 1 where $\leftarrow_K$ is Kleene's implication, $\leftarrow_L$ is the one proposed by Lukasiewicz, and $\Leftarrow$ is a less famous but important implication which has been considered by several authors (cf. [1]) and has been applied to non-monotonic reasoning [5]. The classical equivalence $A \vee B \equiv A \leftarrow_K \neg B$ holds for Kleene's implication but not for $\leftarrow_L$. The latter corresponds to a disjunction, usually denoted by $\oplus$, defined by $A \oplus B \equiv A \leftarrow_L \neg B$.

| $A$ | $\neg A$ |
|---|---|
| $F$ | $T$ |
| $U$ | $U$ |
| $T$ | $F$ |

| $\leftarrow_K$ | $F$ | $U$ | $T$ |
|---|---|---|---|
| $F$ | $T$ | $U$ | $F$ |
| $U$ | $T$ | $U$ | $U$ |
| $T$ | $T$ | $T$ | $T$ |

| $\leftarrow_L$ | $F$ | $U$ | $T$ |
|---|---|---|---|
| $F$ | $T$ | $U$ | $F$ |
| $U$ | $T$ | $T$ | $U$ |
| $T$ | $T$ | $T$ | $T$ |

| $\Leftarrow$ | $F$ | $U$ | $T$ |
|---|---|---|---|
| $F$ | $T$ | $T$ | $F$ |
| $U$ | $T$ | $T$ | $U$ |
| $T$ | $T$ | $T$ | $T$ |

Figure 1: Truth tables for negation and various forms of implication

## 2.2 Default and Autoepistemic Logic

We assume the reader to be familiar with default logic (DL) and autoepistemic logic (AEL). For an extensive treatment, see [12, 13]. A few variants of DL and AEL are briefly recalled in this section.

Baral and Subrahmanian [3] generalized default extensions by introducing the notion of *extension class*, that is a family of sets $\mathcal{F}$ such that $\mathcal{F} = \{ \Gamma_\Delta(E) \mid E \in \mathcal{F} \}$, where $\Gamma_\Delta$ is Reiter's operator. This approach is more robust than DL; in fact, every finite closed default theory has an extension class.

In [17], Schwartz introduced a variant of AEL based on the notion of *reflexive expansions*, which are the solutions of

$$E = \{ \psi \mid T \cup (LE \equiv E) \cup \neg L\overline{E} \vdash \psi \}, \tag{1}$$

where $LE \equiv E$ is an abbreviation for $\{ L\psi \leftrightarrow \psi \mid \psi \in E \}$ and $\neg L\overline{E} = \{ \neg L\psi \mid \psi \notin E \}$. Many theories without stable expansions have a reflexive expansion. Moreover, reflexive expansions contain no weakly-grounded (i.e. self-supporting) beliefs.

Three-valued autoepistemic logic (3AEL) [5] tackles similar problems with a different technique. The basic idea is that agents may have doubts. For a large and expressive class of theories, which generalize Konolige's autoepistemic normal form, we have that every consistent theory has one minimal *generalized stable expansion* (GSE), which enjoys a fixpoint construction and contains no weakly grounded beliefs.

## 2.3 Extended Logic Programs

The set of objective literals, denoted by LIT, consists of all atoms and negated atoms of the form $\neg A$. For all sets of sentences $S$, LIT($S$) denotes $S \cap$ LIT. Similarly, AT($S$) denotes the set of atoms in $S$. As usual, we say that $A$ and $\neg A$ are complementary, and let $\overline{L}$ denote the literal complementary to $L$. A default literal is a formula $\texttt{not}\, L$ where $L$ is an objective literal. We assume the reader to be familiar with extended logic programs (ELP's); see [10] for the definition of their syntax and semantics. *Notation*: the unique answer set of monotonic programs will be denoted by ANS($P$); the Gelfond-Lifschitz transformation of $P$ w.r.t. $S \subseteq$ LIT will be denoted by $P^S_{\rm GL}$.

Giordano and Martelli [11] introduced a different transformation, and a notion of generalized stable model (GSM) for normal logic programs with constraints, which

captures the dependency directed backtracking (DDB) mechanism of TMS's. Given a classical interpretation $M$, their transformation, hereafter denoted by $P^M_{\rm GM}$, is obtained from $P$ in three steps. First, all the literals $\texttt{not}\, B$ such that $B \notin M$ are removed. Then the remaining default literals $\texttt{not}\, B$ are replaced by $\neg B$. Finally, among the resulting rules, select those which are *strictly satisfied* by $M$, i.e. the rules $L_1 \leftarrow L_2, \ldots, L_n$ such that $M$ evaluates $n-1$ of the literals $L_1, \overline{L_2}, \ldots, \overline{L_n}$ to $F$ and one of them to $T$. The GSM's of $P$ are the classical models of $P$ (when default literals $\texttt{not}\, B$ in $P$ are replaced by $\neg B$) that are solutions of the equation

$$\mathrm{AT}(M) = \mathrm{AT}(\mathrm{Cn}(P^M_{\rm GM}))\,.$$

In [4], the semantics of monotonic ELP's has been rephrased in three-valued logic. The basic idea is regarding ELP rules as sentences of the form

$$(\ldots(L_0 \leftarrow L_1) \leftarrow \ldots) \leftarrow L_n\,,$$

where $\leftarrow$ is one of the standard implications illustrated in Fig. 1. The semantics of a monotonic ELP $P$ is captured by $\mathrm{ANS}_X(P)$, which denotes the set of literals that are logical consequences of $P$ in $X$-valued logic $(X = 2, 3)$. It has been shown that $\Leftarrow$ yields the answer set semantics, while $\leftarrow_K$ captures classical logic.

**Theorem 2.1 ([4])** *For all monotonic ELP's $P$,*

i) *If $\leftarrow$ is $\Leftarrow$, then $\mathrm{ANS}_3(P) = \mathrm{ANS}(P)$.*

ii) *If $\leftarrow$ is $\leftarrow_K$, then $\mathrm{ANS}_3(P) = \mathrm{ANS}_2(P)$.*

Lukasiewicz's implication yields a new interesting semantics, whose operational semantics is a restricted form of unit resolution $(\vdash_{\rm UR})$, where clauses should be treated as multisets, rather than sets. A similar restriction of input linear resolution $(\vdash_{\rm IR})$, constitutes an equivalent operational semantics, which models top-down, SLD-like computations.

**Theorem 2.2 ([4])** *For all monotonic ELP's $P$ where $\leftarrow$ is $\leftarrow_L$,*

i) *If $P$ is consistent, then $\mathrm{ANS}_3(P)$ is the least model of $P$.*

ii) *If $P$ is consistent, then its declarative, operational and fixpoint semantics coincide. If $P$ is inconsistent, then the three semantics are all inconsistent.*

iii) *$\mathrm{ANS}_3(P) = \mathrm{ANS}(\mathrm{CNT}_P)$, where $\mathrm{CNT}_P$ is the contrapositive completion of $P$, that is, $P \cup \{\, \overline{L_k} \leftarrow L_1, \ldots, L_{k-1}, \overline{L_0}, L_{k+1}, \ldots L_n \mid L_0 \leftarrow L_1, \ldots, L_n \in P \,\}$.*

# 3 Three-Valued Semantics for ELP's

In this section, the three-valued semantics for monotonic ELP's introduced in [4] is extended to unrestricted ELP's. Default negation is interpreted through the construction underlying stable classes, which captures well-founded and stable semantics in a uniform way. In our framework, the meaning of an ELP is determined by two parameters, namely, the program transformation (e.g. GL, GM) and the truth table of implication (cf. Fig. 1). Accordingly, we replace the operator $F_P$ investigated by Baral and Subrahmanian with $F_P^{\mathrm{TR},\leftarrow}(X) = \mathrm{ANS}_3(P_{\mathrm{TR}}^X)$, where TR is a program transformation and implication is interpreted as $\leftarrow$, which should be one of the connectives illustrated in Fig. 1.

**Definition 3.1** A nonempty family of sets $\mathcal{F}$, contained in the powerset of LIT, is a *TR($\leftarrow$)-answer class* of an ELP $P$ iff $\mathcal{F} = \{ F_P^{\mathrm{TR},\leftarrow}(X) \mid X \in \mathcal{F} \}$. If $\mathcal{F} = \{ S \}$, i.e. if $S$ is a fixpoint of $F_P^{\mathrm{TR},\leftarrow}$, then $S$ is called a *TR($\leftarrow$)-answer set* of $P$. If $\mathcal{F} = \{ S_1, S_2 \}$, where $S_1 \subseteq S_2$, $F_P^{\mathrm{TR},\leftarrow}(S_1) = S_2$ and $F_P^{\mathrm{TR},\leftarrow}(S_2) = S_1$, then we say that $\mathcal{F}$ is an *alternating TR($\leftarrow$)-answer set* of $P$, and denote $\mathcal{F}$ by $(S_1, S_2)$.

The major results of [2] can be immediately extended to our framework. First of all, $F_P^{\mathrm{TR},\leftarrow}$ is anti-monotonic when TR is GL or the transformation RE which will be introduced in Sec. 3.2. Secondly, when $F_P^{\mathrm{TR},\leftarrow}$ is anti-monotonic, its square power is monotonic, and every program $P$ has an alternating TR($\leftarrow$)-answer set

$$\mathcal{F} = \left( \mathrm{lfp}\left( \left[ F_P^{\mathrm{TR},\leftarrow} \right]^2 \right), \mathrm{gfp}\left( \left[ F_P^{\mathrm{TR},\leftarrow} \right]^2 \right) \right), \tag{2}$$

which is also the least TR($\leftarrow$)-answer class under Hoare's ordering. The proofs of these claims are simple adaptations of the proofs in [2] and are left to the reader.

We say that an objective literal $L$ is derivable from an answer class $\mathcal{F}$ iff, for all $S \in \mathcal{F}$, $L \in S$. We say that a default literal $\mathtt{not}\, L$ is derivable from an answer class $\mathcal{F}$ iff, for all $S \in \mathcal{F}$, $L \notin S$. In particular, if $\mathcal{F}$ consists of an answer set $S$, then $L$ is derivable iff $L \in S$, and $\mathtt{not}\, L$ is derivable iff $L \notin S$. When $\mathcal{F}$ is an alternating answer set $(S_1, S_2)$, $L$ is derivable iff $L \in S_1$, and $\mathtt{not}\, L$ is derivable iff $L \notin S_2$. Under this interpretation, the least alternating answer set (2) induces well-founded semantics for ELP's. In the rest of this section we investigate the relations between the above framework and the semantics proposed so far.

## 3.1 GL-Answer Classes

First we study the relationships between GL-answer classes and the existing semantics of ELP's. In [4], it was proved that $\Leftarrow$, which behaves much like an inference rule, preserves the standard meaning of monotonic ELP's. This result can easily be extended to unrestricted ELP's. Actually, we shall prove a more general result, relating ELP's with default logic. By following the terminology in [14], given a program $P$, we will denote by $\mathrm{tr}_1(P)$ the default theory $\langle \emptyset, D \rangle$

where $D$ is the set of defaults $L_1 \wedge \ldots \wedge L_m : \neg L_{m+1}, \ldots, \neg L_n \,/\, L_0$, such that $L_0 \leftarrow L_1, \ldots, L_m, \texttt{not}\, L_{m+1}, \ldots, \texttt{not}\, L_n$ is in $P$.

**Theorem 3.2** *For all ELP's $P$, there is a one to one correspondence between extension classes of* $\mathrm{tr}_1(P)$ *and GL($\Leftarrow$)-answer classes of $P$.*

As a special case of extension classes, we capture various semantics of normal logic programs (cf. [2, 10]). GL($\Leftarrow$)-answer classes capture stable classes; the well-founded model is captured by the least alternating GL($\Leftarrow$)-answer set of $P$. Moreover:

**Corollary 3.3** *For all ELP's $P$, every answer set of $P$ is a GL($\Leftarrow$)-answer set of $P$ and vice-versa.*

The least alternating GL($\Leftarrow$)-answer sets of ELP's are a natural generalization of the well-founded semantics. By Theorem 3.2, they correspond to Baral and Subrahmanian's *well-founded semantics of default logic* [2].

Next we focus on Kleene's valuation, which yields a semantics whose monotonic inferences are exactly the ones supported by classical logic (cf. Theorem 2.1). The corresponding embedding in default logic is $\mathrm{tr}_2$, introduced in [14], which translates $L_0 \leftarrow L_1, \ldots, L_m, \texttt{not}\, L_{m+1}, \ldots, \texttt{not}\, L_n$ into : $\neg L_{m+1}, \ldots, \neg L_n \,/\, L_0 \leftarrow L_1 \wedge \ldots \wedge L_m$.

**Theorem 3.4** *For all ELP's $P$, there is a one to one correspondence between extension classes of* $\mathrm{tr}_2(P)$ *and GL($\leftarrow_K$)-answer classes of $P$.*

Next we clarify the correspondence with 3AEL, which encompasses AEL as a special case.

**Theorem 3.5** *Let $P$ be an ELP and let* AE($P$) *be the autoepistemic translation of $P$ obtained by replacing* `not` *with $\neg L$ and $\leftarrow$ with $\Leftarrow$. There is a one to one correspondence between:*

i) *GSE's of* AE($P$) *and consistent alternating GL($\leftarrow_K$)-answer sets of $P$.*

ii) *standard stable expansions of* AE($P$) *and GL($\leftarrow_K$)-answer sets of $P$.*

Finally, we consider Łukasiewicz's implication. It induces new semantics where contraposition is allowed. In particular, for a given program $P$, let $\mathrm{CNT}_P$ be the *contrapositive completion of $P$*, consisting of $P$ and all the contrapositives

$$\overline{L_i} \leftarrow L_1, \ldots, L_{i-1}, \overline{L_0}, L_{i+1}, \ldots, L_m, \texttt{not}\, L_{m+1}, \ldots, \texttt{not}\, L_n\,, \tag{3}$$

of the rules $L_0 \leftarrow L_1, \ldots, L_m, \texttt{not}\, L_{m+1}, \ldots, \texttt{not}\, L_n$ in $P$.The following proposition is an easy consequence of Theorem 2.2.(iii).

**Proposition 3.6** *For all ELP's $P$, the GL($\leftarrow_L$)-answer classes of $P$ and the GL($\Leftarrow$)-answer classes of* $\mathrm{CNT}_P$ *coincide. Moreover, the GL($\leftarrow_L$)-answer sets of $P$ are the (standard) answer sets of* $\mathrm{CNT}_P$.

1. $a(x) \leftarrow w(x).\quad a(x) \leftarrow f(x).\quad \ldots\quad a(x) \leftarrow s(x)$
2. $w(w_1).\quad f(f_1).\quad b(b_1).\quad c(c_1).\quad s(s_1)$
3. $g(g_1).\quad p(x) \leftarrow g(x)$
4. $sm(x,y) \leftarrow b(y), c(x)$
5. $sm(x,y) \leftarrow b(y), s(x)$
6. $sm(x,y) \leftarrow b(x), f(y)$
7. $sm(x,y) \leftarrow f(x), w(y)$
8. $\neg l(x,y) \leftarrow w(x), f(y)$
9. $\neg l(x,y) \leftarrow w(x), g(y)$
10. $\neg l(x,y) \leftarrow b(x), s(y)$
11. $l(x,y) \leftarrow b(x), c(y)$
12. $l(x, p_2(x)) \leftarrow c(x).\quad p(p_2(x)) \leftarrow c(x)$
13. $l(x, p_3(x)) \leftarrow s(x).\quad p(p_3(x)) \leftarrow s(x)$
14. $[\, l(x,y) \leftarrow p(x) \,] \oplus [\, l(x,z) \leftarrow a(z), sm(z,x), p(u), l(z,u) \,] \leftarrow a(x)$

Figure 2: An ELP for Schubert's steamroller

Contrapositives may not seem a significative extension, at first glance. On the contrary, contraposition makes it possible to solve in a natural and efficient way many difficult benchmark problems for automatic theorem provers. About 60% of the benchmarks without equality listed in [15] — including some of the most difficult ones, according to Pelletier's rating—can be solved through Łukasiewicz's semantics, and contraposition proves to be essential (cf. [4]). This is an astonishing result for a monotonic logic programming language. The problems that can be successfully solved include Schubert's Steamroller—one of the two most difficult problems of [15]—and the Dreadsbury Mansion Mistery. Both have been recently considered in [19], where a nonmonotonic version of the Steamroller is introduced and proposed as a benchmark. In the following example, we show how the non-monotonic Steamroller can be solved through Łukasiewicz's semantics. To enhance readability, we shall use $\oplus$ as syntactic sugar. Note that every formula in the following example can be turned into an equivalent ELP rule of the same size.

**Example 3.7** The following is a description of the non-monotonic version of Schubert's Steamroller due to Wagner; numbers refer to the formalization illustrated in Fig. 2, where $\leftarrow$ should be interpreted as $\leftarrow_L$. Default rules like "Normally F if G" are expressed through semi-normal rules $F(x) \leftarrow G(x)$, `not` $ab_R(x)$, `not` $\neg F(x)$.

> Wolves, foxes, birds caterpillars and snails are animals, and there are some of each of them (1–2). Also, there are some grains and grains are plants (3). Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves (4–7). Normally, wolves do not like to eat foxes or grains, while birds like to eat caterpillars

but not snails (8–11). Normally, caterpillars and snails like to eat some plants (12–13). Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants (14). *Conclusion*: there is an animal that likes to eat a grain-eating animal.

The ELP in Fig. 2 has one GL($\leftarrow_L$)-answer set $S$. This answer set contains $l(b_1, g_1)$, $l(w_1, b_1)$ and $l(f_1, b_1)$, which constitute two constructive solutions to the puzzle (in the sense that $\langle w_1, b_1, g_1 \rangle$ and $\langle f_1, b_1, g_1 \rangle$ are witnesses to the answer). Contrapositive reasoning is essential to the solution, because (14) has to be used in three different directions, for each solution. To illustrate the non-monotonic character of the above formalization, Wagner introduces an abnormal bird $O$ that dislikes caterpillars, and a fox $F$ that likes to eat all smaller animals which are apparently vegetarian.

15. $b(O)$
16. $\neg l(O, x) \leftarrow c(x)$
17. $f(F)$
18. $l(F, x) \leftarrow a(x), sm(x, F), \texttt{not} \neg veg(x)$
19. $\neg veg(x) \leftarrow a(y), l(x, y)$.

Since $O$ dislikes caterpillars, two facts can be deduced. First, there is no evidence that $O$ likes to eat any animal. By (18), it follows that $F$ would like to eat $O$. Secondly, through (14), it can be concluded that $O$ likes to eat some grain. Thus we get one more solution to the puzzle. Wagner proposes this inference as a benchmark test for automated reasoning systems. The extended ELP consisting of (1)–(19) solves the puzzle. Its unique GL($\leftarrow_L$)-answer set $S'$ contains $l(F, O)$ and $l(O, g_1)$. Note that well-founded Łukasiewicz's semantics yields the same results. More precisely, the least alternating GL($\leftarrow_L$)-answer sets of the two programs are $(S, S)$ and $(S', S')$, respectively. This is interesting, because this well-founded semantics can be computed efficiently (it can be shown that, for all ground programs $P$, the least alternating answer set can be computed in time $O(|P|^2)$). Unlike most (all$\Gamma$) of the non-monotonic formalisms in PTIME, the well-founded semantics based on Łukasiewicz's logic is able to solve complex reasoning problems involving incomplete information.

A common objection is that through strong or pseudo negation (and hence answer sets, WFSX, etc.) one can easily solve the same benchmark problems, by explicitly adding the contrapositives of program rules; moreover, this solution is claimed to be more flexible, since programmers can decide whether they want contrapositives or not. This objection disregards the computational cost of the proposed solution, which may cause a quadratic blow-up of space and time complexity. Therefore, this solution is not acceptable in practice for large programs. Secondly, adopting Łukasiewicz's semantics causes no loss of flexibility. When a one-way rule "if $B$ then $A$" is really needed (although I know of no really convincing motivation for such rules) then it suffices to rewrite the rule as $A \leftarrow B, B$. We conclude that, in general, removing contrapositives when they are not needed is better than adding contrapositives when they are needed.

## 3.2 Reflexive Transformation

The Gelfond-Lifschitz transformation removes all the program rules which cannot produce any interesting consequence when implication is interpreted as $\Leftarrow$. However, when the truth tables by Kleene and Łukasiewicz are adopted, these rules may produce interesting conclusions. It is interesting to study a different transformation which turns each ELP into a monotonic ELP without removing any rule.

**Definition 3.8** The *reflexive transformation* of $P$ with respect to $S$, denoted by $P^S_{\mathrm{RE}}$, is obtained from $P$ by deleting all literals $\mathtt{not}\, L$ such that $L \notin S$, and by replacing each literal $\mathtt{not}\, L$ such that $L \in S$ with $\overline{L}$.

The name "reflexive" is due to a deep analogy with reflexive expansions. If not is interpreted as $\neg L$, then (1) can be rephrased as

$$E = \{\, \varphi \mid \mathrm{AE}(P) \cup \{\, \mathtt{not}\, \psi \mid \psi \notin E \,\} \cup \{\, \mathtt{not}\, \psi \leftrightarrow \neg\psi \mid \psi \in E \,\} \vdash \varphi \,\}\,.$$

Clearly, $P^E_{\mathrm{RE}}$ is nothing but a simplification of $\mathrm{AE}(P)$ using $\{\, \mathtt{not}\, \psi \mid \psi \notin E \,\}$ and $\{\, \mathtt{not}\, \psi \leftrightarrow \neg\psi \mid \psi \in E \,\}$. Not surprisingly, we obtain the following result.

**Theorem 3.9** *For all ELP's $P$, $S$ is an RE($\leftarrow_K$)-answer set of $P$ iff* $\mathrm{AE}(P)$ *has a reflexive expansion $T$ such that* $\mathrm{LIT}(T) = S$.

Alternating RE($\leftarrow_K$)-answer sets do not correspond to any semantics for extended logic programs proposed in the literature.

**Example 3.10** Consider the normal program $P = \{\, A \leftarrow_K \mathtt{not}\, B,\ B \leftarrow_K \mathtt{not}\, A,\ C \leftarrow_K A,\ C \leftarrow_K \mathtt{not}\, A \,\}$. While the well-founded model of $P$ corresponds to the alternating sets $(\emptyset, \{\, A, B, C \,\})$, the least alternating answer set of $P$ is $(\{C\}, \{\, A, B, C \,\})$. Note that $C$ is derived although neither $A$ nor $\mathtt{not}\, A$ are derivable; as expected, Kleene's semantics supports nonconstructive inferences.

Next we focus on $\Leftarrow$. Of course, since $\Leftarrow$ is similar to an inference rule, we obtain a semantics which is very similar to standard answer sets.

**Theorem 3.11** *For all ELP's $P$, there is a one to one correspondence between consistent RE($\Leftarrow$)-answer sets of $P$ and answer sets of $P$.*

The difference between RE and GL becomes evident when alternating answer sets are considered.

**Example 3.12** Let $P = \{\, A \Leftarrow \mathtt{not}\, B,\ \neg A \Leftarrow \mathtt{not}\, C,\ B \Leftarrow \mathtt{not}\, C,\ \neg C \,\}$. The least alternating GL($\Leftarrow$)-answer set is $(\{\, \neg C \,\}, \mathrm{LIT})$, while the least alternating RE($\Leftarrow$)-answer set is $(S, S)$, where $S = \{\, \neg C,\ B,\ \neg A \,\}$. Here, RE translates $\mathtt{not}\, C$ into $\neg C$; so, from $\neg C$ we can derive $\neg A$ and $B$. The effect is similar to the coherence principle of WFSX (from $\neg C$ infer $\mathtt{not}\, C$). In fact, the least alternating RE($\Leftarrow$)-answer set

is always equivalent to WFSX, when the second element of the former is not LIT. In general, however, WFSX is stronger. Consider $P' = \{ A \Leftarrow \mathtt{not}\, \neg A,\ \neg A \Leftarrow \mathtt{not}\, A,\ B \Leftarrow \mathtt{not}\, C,\ \neg C,\ D \Leftarrow \mathtt{not}\, E \}$. The least alternating RE($\Leftarrow$)-answer set of $P'$ is $M = (\{ \neg C,\ B \}, \mathrm{LIT})$. WFSX allows to conclude also $\mathtt{not}\, E$ and $D$.

For normal programs, one can prove that the least alternating RE($\Leftarrow$)-answer sets capture exactly the well-founded semantics.

Finally, we consider Łukasiewicz's implication. RE($\leftarrow_L$)-answer sets are in one-to-one correspondence with the generalized stable models of [11], and hence, they provide the dependency-directed backtracking mechanism of justification-based TMSs [6] with a logical characterization.

**Theorem 3.13** *Let $P$ be a normal logic program with constraints. There is a one to one correspondence between consistent RE($\leftarrow_L$)-answer sets of $P$ and generalized stable models of $P$.*

Thus, we can give a logical explanation of an apparently non-logical step in the transformation by Giordano and Martelli, namely, the elimination of the clauses which are not strictly satisfied.

**Example 3.14** Let $P = \{ p \leftarrow \mathtt{not}\, p \}$. The GM transformation must remove the unique rule of $P$, otherwise $p$ would be a classical consequence of $P_{\mathrm{GM}}^{\{p\}}$, and hence $\{p\}$ would be a GSM of $P$, while there is no TMS labelling for $P$. The rule is actually removed, since $p \leftarrow \neg p$ is not strictly satisfied by $\{p\}$. However, there is no independent semantic motivation for this operation. In Łukasiewicz's logic, $p$ is not a logical consequence of $p \leftarrow \neg p$ (the least model of this rule is the empty interpretation). Consequently, we need not eliminate this rule.

Concerning alternating answer sets, with Łukasiewicz's implication we get a well-founded version of the generalized stable model semantics, very similar to the *restricted belief revision model* of [18]. The two semantics agree on all the examples mentioned in [18]. However, in general, they do not coincide.

**Example 3.15** Let $P$ be the program with constraints

$$A \quad B_1 \quad C_1 \leftarrow_L A, \mathtt{not}\, B_2 \quad C_2 \leftarrow_L \mathtt{not}\, D \quad -\leftarrow_L B_1, B_2 \quad -\leftarrow_L C_1, C_2 .$$

The least alternating RE($\leftarrow_L$)-answer set of $P$ corresponds to the RE($\leftarrow_L$)-answer set $S = \{ A, B_1, \neg B_2, C_1, \neg C_2, D \}$, while the restricted belief revision model of $P$ corresponds to $M = (\{ A, B_1, \neg B_2 \}, \mathrm{LIT})$.

As noted in [18], the meaning of strong negation is not always preserved in restricted belief revision models; this motivated a weaker notion called *skeptical belief revision model*. RE($\leftarrow_L$)-answer classes constitute an interesting alternative; the meaning of strong negation is preserved by construction; more inferences can be drawn. Nonetheless, the least alternating RE($\leftarrow_L$)-answer set can be computed in quadratic time, while the skeptical belief revision model requires cubic time.

| | $\Leftarrow$ | $\leftarrow_L$ | $\leftarrow_K$ |
|---|---|---|---|
| GL | Default Logic* under $tr_1$; Answer sets | Idem + Contrapositives | Default Logic* under $tr_2$; AEL and 3AEL |
| RE | Consistent answer sets | Gen. stable models as in [11]; DDB in TMS | Reflexive AEL |

* Correspondence is with all extension classes.

Table 1: Relations with existing formalisms.

# 4 Discussion and Conclusions

We have investigated nonmonotonic extension of three-valued logic as a semantic of ELP's. By tuning two parameters, that is, the truth table of implication and the program transformation involved in the nonmonotonic construction, we obtained a unifying view of many formalisms, some of which had never been related before. The main results are summarized in Fig. 1.

This framework highlights surprising similarities between previously unrelated formalisms. The dependency directed backtracking of TMS's can be regarded as a variant of reflexive AEL based on Lukasiewicz's logic. The WFSX semantics by Alferes and Pereira can be regarded as a variant of reflexive AEL based on $\Leftarrow$, modulo the extra non-monotonic inferences supported by the paraconsistent behaviour of WFSX, which make WFSX more powerful.

Łukasiewicz's implication induces some very interesting new semantics, which support contraposition. This apparently minor improvement makes it possible to solve many hard benchmark problems involving incomplete information, with a substantial gain in elegance and efficiency.

Contrapositives model in a natural way the dependency-directed backtracking mechanism of TMS's (Theorem 3.13). The alternating RE($\leftarrow_L$)-amswer sets constitute an appealing alternative to the skeptical belief revision models of [18]. The former are at the same time more powerful and more efficient.

# References


[1] A. Avron. Natural 3-valued logics – characterization and proof theory. *The Journal of Symbolic Logic*, 56(1):276-294, (1991).

[2] C.R. Baral, V.S. Subrahmanian. Dualities between Alternative Semantics for Logic Programming and Nonmonotonic Reasoning *The Journal of Automated Reasoning*, 10:399–420, 1993.

[3] C.R. Baral, V.S. Subrahmanian. Stable and Extension Class Theory for Logic Programming and Nonmonotonic Reasoning *The Journal of Automated Reasoning*, 8:345–366, 1992.

[4] P.A. Bonatti. On the monotonic fragment of extended logic programs. Submitted.

[5] P.A. Bonatti. Autoepistemic logics as a unifying framework for the semantics of logic programs. *The Journal of Logic Programming*, 22(2):91-149 (1995).

[6] J. Doyle. A Truth Maintenance System. *Artificial Intelligence*, 12:231–272, (1979).

[7] D.M. Gabbay, F. Guenthner. *Handbook of philosophical logic, Vol. III: Alternatives to classical logic*. Reidel Publishing Co., Dordrecht, Holland, 1986.

[8] D.W. Loveland. *Automated Theorem Proving: A Logical Basis*. North-Holland, Amsterdam, 1978.

[9] M. Gelfond. On stratified autoepistemic theories. In *Proc. AAAI-87*, pages 207–211, 1987.

[10] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proc. ICLP-90*, pages 579–597, Jerusalem, 1990.

[11] L. Giordano and A. Martelli. Generalized stable models, truth maintenance and conflict resolution. In *Proc. ICLP-90*, pages 427–441, Jerusalem, 1990.

[12] W. Lukasiewicz. *Non-Monotonic Reasoning*. Ellis Horwood Limited, Chichester, England, 1990.

[13] W. Marek and M. Truszczyński. *Nonmonotonic Logics – Context-Dependent Reasoning*. Springer Verlag, 1993.

[14] W. Marek and M. Truszczyński. Stable semantics for logic programs and default theories. In *Proc. of NACLP 89*, pages 243–256, 1989. Springer Verlag, 1993.

[15] F.J. Pelletier. Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning*, 2:191-216, (1986).

[16] L.M. Pereira, J.J. Alferes. Well Founded Semantics for Logic Programs with Explicit Negation. In B. Neumann (ed.), *Proc. of ECAI-92*, pages 102–106, John Wiley & Sons, Chichester, 1992.

[17] G. Schwarz Autoepistemic logic of knowledge. In *Proc. Logic Programming and Non-monotonic Reasoning*, pages 260–274, MIT Press, 1991.

[18] C. Witteveen and G. Brewka. Skeptical reason maintenance and belief revision. *Artificial Intelligence*, 61:1–36, 1993.

[19] G. Wagner. Solving the Steamroller and other puzzles with extended disjunctive logic programs. Proc. of the ICLP-94 Workshop on Non-Monotonic Extensions of Logic Programming, S. Margherita Ligure, Italy, 1994.