Super Logic Programs

Stefan Brass
University of Hannover

Jürgen Dix
University of Koblenz-Landau

Teodor C. Przymusinski
University of California at Riverside

Part I: Motivation

- Why Disjunction?
- Programs and Semantics
- What Do We Want?
- What else?

Part II: The Framework

- Knowledge Bases
- Super Programs, STATIC
- Properties, Characterizations

Part III: Relation to D-WFS

- Some Transformations
- D-WFS
- STATIC versus D-WFS

1 Why Disjunction?

- *Null values:* an age “around 30”
- *Legal rules:* laws usually do not have unique models;
- *Diagnosis:* which part of some machine is faulty;
- *Biological inheritance:* if the parents have blood groups A and 0, the child must also have one of these.
- *Natural language understanding:* ambiguity, multiple intended models;
- *Conflicts in multiple inheritance:* assume disjunction of the inherited values.

2 Usual Programs

Definition I.2.1 (Logic Program P)
A logic program P is a finite set of propositional rules

$$A_1 \vee \cdots \vee A_k \leftarrow B_1 \wedge \cdots \wedge B_m \wedge \neg C_1 \wedge \cdots \wedge \neg C_n,$$

where the $A_i/B_i/C_i$ are $\mathcal{L}$-atoms different from $\bot$, $k \geq 1$, $m \geq 0$, $n \geq 0$.

- *no* **headless** *rules,*
- *no* **classical** *or* **extended** *negations,*
- $\neg$ *only applies to* **single atoms**.

3 What Do We Want?

We want to extend logic programs:

Syntax: New *syntax* should resembles the old syntax but should apply to a significantly broader class of programs;

Semantics: New *semantics* should be a natural extension:

1. **Without Negation:** **GCWA**,
2. **Without Disjunction:** **WFS**.

Computation: New procedural mechanism should allow, at least in principle, to compute the semantics.

4 What Else?

Reasonable Properties:

Consistency

Relevance

Extension-by-Definition ($\mapsto$ D-WFS)

Modularity

1 Knowledge Bases

- Extend the language $\mathcal{L}$ to a propositional modal language $\mathcal{L}_{Not}$, Not is the *default negation* operator. Atomic formulae of the form $Not\,F$, where $F \in Fml(\mathcal{L}_{Not})$, are called *default atoms*. $Fml(\mathcal{L})$ is the set of *objective* formulae.

- A Knowledge Base is any theory in $\mathcal{L}_{Not}$.

- We assume the following axioms:

  (K') $Not\,(F \vee G) \leftrightarrow Not\,F \wedge Not\,G$

  (D) $Not\,(\mathit{false})$ and $\neg Not\,(\mathit{true})$

  (N') $\frac{F \leftrightarrow G}{Not\,F \leftrightarrow Not\,G}$

Definition II.1.1 (Derivable Formulae)
For any knowledge database T we denote by $Cn_{Not}(T)$ *the smallest set of formulae of the language* $\mathcal{L}_{Not}$ *which contains* T*, all the (substitution instances of) the axioms* *(K')* *and* *(D)* *and is closed under both propositional consequence and the invariance rule* *(N')*.

By a *minimal model* of T we mean a model M of T with:

there is *no* smaller model N of T
which coincides with M on default atoms $Not\,F$.

We write: $T \models_{\min} F$

Definition II.1.2 (Static Completion)
Static completion $\overline{T}$ of a knowledge database T is defined:

$$\overline{T} =_{def} Cn_{Not}(T \cup \{Not\, F : T \models_{\min} \neg F\}),$$

where the F's range over all formulae of $\mathcal{L}_{Not}$.

Definition II.1.3 (Static Expansion, Przymusinski)
A knowledge database $T^\diamond$ is called a static expansion *of a knowledge database T if it is a fixed-point of the operator Ψ_T:*

$$\Psi_T(S) =_{def} Cn_{Not}(T \cup \{Not\, F : S \models_{\min} \neg F\}),$$

i.e. $T^\diamond = \Psi_T(T^\diamond)$.

Theorem II.1.4 (Main)
The least static expansion of a knowledge database T is obtained by a single iteration of the operator Ψ_T. In other words, the least fixed point of the operator Ψ_T coincides with the static completion $\overline{T}$ of T:

$$\mathit{lfp}(\Psi_T) = \Psi_T(T) = \overline{T} = \Psi_T(\overline{T}).$$

Definition II.1.5 (STATIC Semantics)
The static semantics of a knowledge database T is defined as the least static expansion of T. It coincides with the static completion of T.

2 Super Programs

Definition II.2.1 (Super Logic Programs)
A Super Logic Program is a non-monotonic knowledge database consisting of super-clauses of the form:

$$F \leftarrow G \wedge \mathit{Not}\, H$$

where F and G are arbitrary objective *formulae and H is an arbitrary* positive *objective formula.*

Equivalently:

$$A_1 \vee \ldots \vee A_k \leftarrow B_1 \wedge \ldots \wedge B_m \wedge Not\, C_1 \wedge \ldots \wedge Not\, C_n,$$

where A_i's and B_i's are objective atoms and C_i's are conjunctions of objective atoms.

If $k \neq 0$, for all rules, P is called *affirmative*.

Super programs allow for

- Constraints,
- $Not\, C$ where $\boxed{C \text{ is a conjunction of atoms}}$,
- various forms of extended Negations,
  (see JELIA '96, Alferes, Pereira and Przymusinski).

3 Properties

Theorem II.3.1
Let P be a super logic program, $\overline{P}$ its static completion.

- *For any F, $\overline{P} \models Not\,F \;\equiv\; \overline{P} \models_{min} \neg F$.*
- *Affirmative super programs have consistent static completions.*
- *STATIC extends both WFS for normal and GCWA for pos. dis. programs.*

Super programs do not contain nested beliefs

$$\mathcal{L}^*_{Not} = \mathcal{L} \cup \{Not\, E : E \text{ is a conj. of objective atoms}\}$$

Theorem II.3.2 (Fixed-Point Characterization of $\overline{P}$)
Static completion $\overline{P}$ is obtained by defining $P^0 = P$ and:

$$P^{n+1} =_{def} Cn\,(P \cup \{Not\, E_1 \wedge \ldots \wedge Not\, E_n \rightarrow Not\, E_0 : P^n \models_{\min} \neg E_1 \wedge \ldots \wedge \neg E_n \rightarrow \neg E_0\}),$$

where E_i's range over all conjunctions of objective atoms and Cn denotes standard *propositional consequence operator.*

The sequence $\{P^n\}$ is monotonically increasing and has a unique fixed point $P^m = P^{m+1}$, for some natural number m. Moreover, $P^m = \overline{P}|\mathcal{L}^*_{Not}$.

1 Some Transformations

We can list some natural program-transformations and require a semantics to satisfy them:

- Elimination of Tautologies :

$$\boxed{A \vee B \vee C \quad \leftarrow \quad D, A, \neg E}$$

- Elimination of Non-Minimal Rules :

$$\boxed{\begin{array}{lcl} A \vee B \vee C & \leftarrow & D, \neg E, \neg F \\ A \vee B & \leftarrow & D, \neg F \end{array}} \quad \mapsto \quad \boxed{A \vee B \quad \leftarrow \quad D, \neg F}$$

Red$^+$

$$\begin{array}{lcl} A \lor B & \leftarrow & \neg F \\ E & \leftarrow & A, \neg F \end{array} \quad \mapsto \quad \begin{array}{lcl} A \lor B & \leftarrow & \\ E & \leftarrow & A \end{array}$$

Red$^-$

$$\begin{array}{lcl} A \lor B & \leftarrow & \\ F \lor E & \leftarrow & C, \neg A, \neg B, \neg F \end{array} \quad \mapsto \quad \begin{array}{lcl} A \lor B & \leftarrow & \end{array}$$

Extension-by-Definition

$$\begin{array}{lcl} B & \leftarrow & \neg E \\ A' \lor B & \leftarrow & D, \neg C' \\ A \lor D & \leftarrow & B, \neg C \end{array} \quad \mapsto \quad \begin{array}{lcl} A \lor D \lor A' & \leftarrow & D, \neg C', \neg C \\ A \lor D & \leftarrow & \neg E, \neg C \end{array}$$

2 D-WFS

Theorem III.2.1 (D-WFS, (Brass/Dix ICLP '95)) *There exists the weakest semantics which is invariant under all the natural transformations just introduced. Moreover, this semantics is consistent and closed under logical consequences (as a set consisting of pure disjunctions). It is also modular, relevant. We call it the Disjunctive Well-Founded Semantics, or, briefly, D-WFS.*

3 D-WFS versus STATIC

Theorem III.3.1 (Properties of STATIC)
The STATIC Semantics satisfies all the program transformations that D-WFS was based on. It is therefore at least as strong as D-WFS.

Beispiel III.3.2 (STATIC is stronger than D-WFS)
Let

$$P: \begin{array}{lcl} a \vee c & \leftarrow & \neg d \\ c & \leftarrow & \neg c \\ d & \leftarrow & c \end{array}$$

Now STATIC concludes $\neg a$ but D-WFS does not. The difference is the (K')–axiom which implies $\neg c \leftarrow \neg d$ in the corresponding belief theory.

Flat STATIC coincides with D-WFS

Theorem III.3.3 (Brass, Dix, Niemelä, Przymusinski)
If we define a static completion as a fixpoint where we do not allow for nested beliefs (Flat STATIC), then this version coincides with D-WFS .