Integration of Information in Four-valued Logics under Non-Uniform Assumptions

Yann Loyer, Nicolas Spyratos and Daniel Stamate

Laboratoire de Recherche en Informatique, UMR 8623,
Université de Paris Sud, Bat. 490, 91405 Orsay cedex, France
{loyer,spyratos,daniel}@lri.fr

Abstract

We study the problem of integrating information coming from different sources in a distributed environment. We assume a central server that collects facts from sources and tries to combine them using a set of logical rules, i.e. a logic program. We provide a formal framework for the integration of information in such a setting, under non-uniform assumptions on the missing information.

Keywords : *four-valued logics, integration of heterogeneous information, logics of knowledge, inconsistency, logic programming.*

1 Introduction

We study the problem of integrating information coming from different sources in a distributed environment. We assume a central server that collects facts from sources and tries to combine them using a set of logical rules, i.e. a logic program.

In such a setting, incomplete information from a source, or contradictory information coming from different sources, necessitate the use of many-valued logics, in which programs have to be evaluated. However, unlike in conventional logic programming, we $cannot$ assume here the same default value for all items whose truth-values cannot be inferred from the program. Indeed, as the sources are usually autonomous, it is not realistic to assume that all such items are false (as in the closed world assumption [5, 10, 11]), or that all such items are unknown (as in the Kripke-Kleene semantics [2]). Instead, we want to be able to assign *different* default values, depending on the source where the information comes from. Let us see an example.

Example 1 *Consider a legal case where a judge (the central server) receives information from two sources that influence the decision: the prosecutor who deems a person is suspect if he has a motive or if there is a witness, and the defense lawyer who claims that his customer cannot be a suspect if he has an alibi (assuming this alibi is not provided by friends). The information provided by these sources is then "integrated" by the judge to decide whether there must be a trial. This decision problem is described by the following program.*

$$\mathcal{P}\begin{cases} \textit{trial}(X) & \leftarrow & \textit{suspect}(X) \oplus \neg \textit{innocent}(X) \\ \textit{suspect(X)} & \leftarrow & \textit{motive}(X) \vee \textit{witness}(X) \\ \textit{innocent}(X) & \leftarrow & \exists Y (\textit{alibi}(X,Y) \wedge \neg \textit{friends}(X,Y)) \\ \textit{friends}(X,Y) & \leftarrow & \textit{friends}(Y,X) \\ & & \vee (\textit{friends}(X,Z) \wedge \textit{friends}(Z,Y)) \end{cases}$$

Here the connective $\oplus$ in the first rule puts together the information obtained from the prosecutor and the defense lawyer as follows:
- if the information is contradictory, then the program will assign to the predicate $trial(X)$ the logical value $inconsistent$,
- and if one of the sources does not provide any information, then the value of the other source determines the logical value of the predicate $trial(X)$.

The second rule describes how the prosecutor works. The predicates used in this rule must be assigned by default the logical value $false$: if in doubt, we assume that there is no motive, no witness and that the person is not a suspect.

The third rule describes how the lawyer works and depends on the fourth rule which defines the relation $friends$. The predicates used in the last two rules must be assigned by default the logical value $unknown$, because if the default value was $true$ or $false$, then, in doubt, we would deduce that the person is not innocent. So, if in doubt, we assume nothing.

As the previous example shows, there are two basic issues to be dealt with in our setting:

- *Multi-valued logics*, i.e. the need for more than two logical values (and operators on these values). Indeed, apart from the classical values, $true$ and $false$,

we need at least two more values, one to represent $undefined$ (i.e. absence of information) and the other to represent $overdefined$ (i.e. inconsistent information). Moreover, we need new operators on these values such as $\oplus$, as explained earlier.

- *Non-uniform assumptions*, i.e. the need for (potentially) assigning *different* default values to predicates whose logical values cannot be inferred from the rules.

The main contribution of this paper is to provide a formal framework in which the above issues can be handled. This is done as follows:
(1) We work with Belnap's four-valued logic [1], and we consider the class of logic programs defined by Fitting [4].
(2) We define an operator for computing the semantics of Fitting programs under non-uniform assumptions, and we give an algorithm for their effective evaluation [9].

Previous work by the authors [8] introduced a framework for the unified treatment of conventional program semantics under uniform assumptions for the missing information. The present paper extends that work to the case of non-uniform assumptions thus allowing to reason about information integration in a distributed environment.

In the remaining of the paper, we first recall very briefly some definitions and notations from three- and four-valued logics and then proceed to define parameterized semantics for Fitting programs inspired by [4, 8]. These semantics capture the usual semantics of logic programs, thus allowing their comparison. Proofs of theorems are omitted due to lack of space.

2 Preliminaries

2.1 Three-valued logics

Stable models and well founded semantics. Gelfond and Lifschitz introduced the notion of stable model [5], in the framework of classical logic under the closed world assumption. This notion was then extended to three-valued logics and partial interpretations: Van Gelder, Ross and Schlipf introduced the well-founded semantics [11], and Przymusinski defined the three-valued stable models and showed in [10] that his extension captures both the bi-valued stable models and the well-founded semantics.

In Przymusinski's approach, a conjunctive logic program is a set of clauses of the form $A \longleftarrow B_1 \wedge ... \wedge B_n \wedge \neg C_1 \wedge ... \wedge \neg C_m$, where $B_1, ..., B_n, C_1, ..., C_m$ are atoms. In this context, a valuation is a mapping that assigns to each ground atom a truth value from the set $\{false, unknown, true\}$. A valuation can be extended to ground literals and conjunctions of ground literals in the usual way. To define the stable models and well-founded semantics of a program $\mathcal{P}$, one uses the extended Gelfond-Lifschitz transformation $GL_{\mathcal{P}}$ [10] which assigns to each valuation v another valuation $GL_{\mathcal{P}}(v)$ defined as follows :
(1) Transform $\mathcal{P}$ into a positive program $\mathcal{P}_{/v}$ by replacing all negative literals by their values from v.
(2) Compute the least fixpoint of an immediate consequence operator Φ defined as follows :

- if the ground atom A is not in the head of any rule of Inst-$\mathcal{P}_{/v}$[1], then $\Phi_{\mathcal{P}_{/v}}(v)(A) = false$;
- if the rule "$A \longleftarrow$" is in Inst-$\mathcal{P}_{/v}$, then $\Phi_{\mathcal{P}_{/v}}(v)(A) = true$;
- else $\Phi_{\mathcal{P}_{/v}}(v)(A) = \bigvee\{v(B)/A \leftarrow B \in \text{Inst-}\mathcal{P}_{/v}\}$, where $\vee$ is the extension of classical disjunction defined by: $f \vee u = u$; $t \vee u = t$; $u \vee u = u$.

The valuation v is a three-valued stable model of $\mathcal{P}$ if $GL_{\mathcal{P}}(v) = v$, and the least three-valued stable model coincides with the well-founded semantics of $\mathcal{P}$.

It follows from the definition of Φ above that this approach gives greater importance to negative information, so it can be called a $pessimistic$ approach.

Kripke-Kleene semantics. Working with a three-valued logic, or Kleene's logic as well, Fitting introduced the Kripke-Kleene semantics [2]. The program $\mathcal{P}$ has the same definition as for stable models, but the operator Φ is now defined as follows : given a valuation v and a ground atom A in Inst-$\mathcal{P}$,
(1) if there is a rule in Inst-$\mathcal{P}$ with head A, and the truth value of the body under v is $true$, then $\Phi_{\mathcal{P}}(v)(A) = true$;
(2) if there is a rule in Inst-$\mathcal{P}$ with head A, and for every rule in Inst-$\mathcal{P}$ with head A the truth value of the body under v is false, then $\Phi_{\mathcal{P}}(v)(A) = false$;
(3) else $\Phi_{\mathcal{P}}(v)(A) = unknown$.

It follows that this approah gives greater importance to the lack of information since $unknown$ is assigned to the atoms whose logical values cannot be inferred from the rules, so it can be called a $skeptical$ approach.

2.2 Four-valued logics

Belnap's logic. In [1], Belnap defines a logic called $\mathcal{FOUR}$ intended to deal with incomplete and inconsistent information. Belnap's logic uses four logical values, that we shall denote by $\mathcal{F}$, $\mathcal{T}$, $\mathcal{U}$ and $\mathcal{I}$, i.e. $\mathcal{FOUR}$ = {$\mathcal{F}$, $\mathcal{T}$, $\mathcal{U}$, $\mathcal{I}$}. These values can be compared using two orderings, the knowledge ordering and the truth ordering.

In the knowledge ordering, denoted by $\leq_k$, the four values are ordered as follows: $\mathcal{U} \leq_k \mathcal{F}$, $\mathcal{U} \leq_k \mathcal{T}$, $\mathcal{F} \leq_k \mathcal{I}$, $\mathcal{T} \leq_k \mathcal{I}$. Intuitively, according to this ordering, each value of $\mathcal{FOUR}$ is seen as a possible knowledge that one can have about the truth of a given statement. More precisely, this knowledge is expressed as a set of classical truth values that hold for that statement. Thus, $\mathcal{F}$ is seen as $\{false\}$, $\mathcal{T}$ is seen as $\{true\}$, $\mathcal{U}$ is seen as $\emptyset$ and $\mathcal{I}$ is seen as $\{false,true\}$. Following this viewpoint, the knowledge ordering is just the set inclusion ordering.

[1] Inst-$\mathcal{P}_{/v}$ denotes the set of all instanciations of rules of $\mathcal{P}_{/v}$

In the truth ordering, denoted by $\leq_t$, the four logical values are ordered as follows: $\mathcal{F} \leq_t \mathcal{U}$, $\mathcal{F} \leq_t \mathcal{I}$, $\mathcal{U} \leq_t \mathcal{T}$, $\mathcal{I} \leq_t \mathcal{T}$. Intuitively, according to this ordering, each value of $\mathcal{FOUR}$ is seen as the degree of truth of a given statement. $\mathcal{U}$ and $\mathcal{I}$ are both less false than $\mathcal{F}$, and less true than $\mathcal{T}$, but $\mathcal{U}$ and $\mathcal{I}$ are not comparable.

The two orderings are represented in the double Hasse diagram of Figure 1.

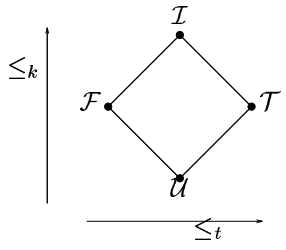

Figure 1. The logic $\mathcal{FOUR}$

As shown in [6], $\mathcal{FOUR}$ is a bilattice under the two orderings. Meet and join under the truth ordering are denoted by $\wedge$ and $\vee$, and they are natural generalizations of the usual notions of conjunction and disjunction. In particular, $\mathcal{U} \wedge \mathcal{I} = \mathcal{F}$ and $\mathcal{U} \vee \mathcal{I} = \mathcal{T}$. Under the knowledge ordering, meet and join are denoted by $\otimes$ and $\oplus$, and are called the *consensus* and *gullibility*, respectively: $x \otimes y$ represents the maximal information on which x and y agree, whereas $x \oplus y$ adds the knowledge represented by x to that represented by y. In particular, $\mathcal{F} \otimes \mathcal{T} = \mathcal{U}$ and $\mathcal{F} \oplus \mathcal{T} = \mathcal{I}$. $\mathcal{FOUR}$ is an infinitary distributive bilattice which satisfies the infinitary interlacing laws (i.e. each of the operations $\wedge$, $\vee$, $\otimes$, $\oplus$ is monotone with respect to both orderings, for example, if $\forall n \in S, a_n \leq_k b_n$ then $\bigvee\{a_n | n \in S\} \leq_k \bigvee\{b_n | n \in S\}$).

There is a natural notion of negation in the truth ordering denoted by $\neg$, and we have: $\neg\, \mathcal{T} = \mathcal{F}$, $\neg\, \mathcal{F} = \mathcal{T}$, $\neg\, \mathcal{U} = \mathcal{U}$, $\neg\, \mathcal{I} = \mathcal{I}$. There is a similar notion for the knowledge ordering, called *conflation*, denoted by -, and we have: $-\,\mathcal{U} = \mathcal{I}$, $-\,\mathcal{I} = \mathcal{U}$, $-\,\mathcal{F} = \mathcal{F}$, $-\,\mathcal{T} = \mathcal{T}$.

The operations $\vee, \wedge, \neg$ restricted to the values $\mathcal{T}$ and $\mathcal{F}$ are those of classical logic, and if we add to these operations and values the value $\mathcal{U}$, then they are those of Kleene's strong three-valued logic.

Fitting programs. Conventional logic programming has the set $\{\mathcal{F}, \mathcal{T}\}$ as its intended space of truth values, but since not every query may produce an answer, partial models are often allowed (i.e. $\mathcal{U}$ is added). If we want to deal with inconsistency as well, then $\mathcal{I}$ must be added. Thus it seems natural to work with Belnap's logic. Fitting extended the notion of logic program to Belnap's logic [4] as follows:
- A formula is an expression built up from literals and elements of $\mathcal{FOUR}$, using $\wedge, \vee, \otimes, \oplus, \exists, \forall$.
- A clause is of the form $P(x_1, ..., x_n) \longleftarrow \phi(x_1, ..., x_n)$, where the atomic formula $P(x_1, ..., x_n)$ is the head, and the formula $\phi(x_1, ..., x_n)$ is the body. It is assumed that the free variables of the body are among $x_1, ..., x_n$.
- A program is a finite set of clauses with no predicate letter appearing in the head of more than one clause (this apparent restriction causes no loss of generality[3]).

We shall refer to such an extended logic program as a *Fitting program*.[2] Fitting also defined the family of conventional logic programs. A *conventional logic program* is one whose underlying truth-value space is the bilattice $\mathcal{FOUR}$ and which does not involve $\otimes, \oplus, \forall, \mathcal{U}, \mathcal{I}$. Such programs can be written in the customary way, using commas to denote conjunction.

3 Parameterized semantics for Fitting programs

In the following, $\mathcal{P}$ is a Fitting program, $\mathcal{V}(\mathcal{FOUR})$ is the set of all valuations in $\mathcal{FOUR}$ and Inst-$\mathcal{P}$ is the set of all ground instances of rules of $\mathcal{P}$. Some of the results in this section are inspired by [4] which deals only with the case where $\alpha_{\mathcal{P}}$ is the valuation that assigns the value false to every literal.

3.1 Immediate Consequence Operators

First, we extend the two orderings on $\mathcal{FOUR}$ to the space of valuations $\mathcal{V}(\mathcal{FOUR})$.

Definition 1 *Let v_1 and v_2 be in $\mathcal{V}(\mathcal{FOUR})$, then*
$v_1 \leq_t v_2$ iff $v_1(A) \leq_t v_2(A)$ for all ground atoms A;
$v_1 \leq_k v_2$ iff $v_1(A) \leq_k v_2(A)$ for all ground atoms A.

Under these two orderings $\mathcal{V}(\mathcal{FOUR})$ becomes a bilattice, and we have $(v \wedge w)(A) = v(A) \wedge w(A)$, and similarly for the other operators. $\mathcal{V}(\mathcal{FOUR})$ is infinitely distributive, satisfies the infinitely interlacing conditions and has a negation and a conflation.
The actions of valuations can be extended from atoms to formulas as follows:

$v(X \wedge Y) = v(X) \wedge v(Y)$, and similarly for the other operators,

$v((\exists x)\phi(x)) = \bigvee_{t=closedterm} v(\phi(t))$, and

$v((\forall x)\phi(x)) = \bigwedge_{t=closedterm} v(\phi(t))$.

The following contrajoin operation assigns to a ground atom A a truth value independently of the truth value assigned to the negation of A.[3]

Definition 2 (contrajoin) *Let v and w be in $\mathcal{V}(\mathcal{FOUR})$. The contrajoin $v \triangle w$ is defined as follows: for each ground atom A, $v \triangle w(A) = v(A)$ and $v \triangle w(\neg A) = \neg w(A)$.*

[2] Actually, Fitting defined logic programs in the context of general bilattices, but in this article we restrict our attention to the minimal bilattice, $\mathcal{FOUR}$.

[3] Our contrajoin operation is exactly the same as pseudovaluation in [4]. However, we prefer the term contrajoin of v and w as it is more indicative of the fact that an operation is performed on valuations v and w.

Contrajoin operations are extended to formulas by induction. The idea is that v represents the information about A, and w the information about $\neg A$. For example, if $v(innocent(John)) = \mathcal{T}$ and $w(innocent(John)) = \mathcal{U}$ then $v \triangle w(innocent(John)) = \mathcal{T}$, whereas $\neg(v \triangle w(\neg innocent(John)) = \mathcal{U}$.

We can now define a new operator $\Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}$ which is inspired by [4]. It infers new information from a contrajoin operation in a way that depends on the valuation $\alpha_\mathcal{P}$.

Definition 3 *Let $\alpha_\mathcal{P}$, v and w be in $\mathcal{V}(\mathcal{FOUR})$. The valuation $\Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}(v,w)$ is defined as follows:*
(1) if the ground atom A is not the head of any rule of Inst-$\mathcal{P}$, then $\Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}(v,w)(A)$=$\alpha_\mathcal{P}(A)$
(2) if $A \leftarrow B$ occurs in Inst-$\mathcal{P}$, then

$$\Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}(v,w)(A) = v \triangle w(B)$$

Clearly, the valuation $\Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}(v,w)$ is in $\mathcal{V}(\mathcal{FOUR})$, and as the interlacing conditions are satisfied by $\mathcal{V}(\mathcal{FOUR})$, we can prove the following proposition.

Proposition 1 *Let $\mathcal{P}$ be a Fitting program.*
(1) Under the knowledge ordering, $\Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}$ is monotonic in both arguments;
(2) Under the truth ordering, $\Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}$ is monotonic (and moreover continuous) in its first argument, and anti-monotonic in its second argument.

In order to define the operator ${\Psi'}_{\mathcal{P}}^{\alpha_\mathcal{P}}$, we need to define the notion of stratification with respect to positive cycles.

Definition 4 (positive cycle) *A positive cycle of a Fitting program $\mathcal{P}$ is a set $\{R_1, ..., R_n\}$ of rules of $\mathcal{P}$ such that for all i in [1...n], the head of R_i appears in the body of R_{i+1} and the head of R_n appears in the body of R_1.*

Definition 5 (Stratification w.r.t. positive cycles) *A stratification w.r.t. positive cycles of a Fitting program $\mathcal{P}$ is a sequence of Fitting programs $P_1, ..., P_n$ such that for the mapping σ from rules of $\mathcal{P}$ to [1...n],*
(1) $P_1, ..., P_n$ is a partition of $\mathcal{P}$.
(2) Every rule R is in $\mathcal{P}_{\sigma(R)}$.
(3) If R_1 and R_2 are two rules of $\mathcal{P}$ such that the head of R_2 appears in the body of R_1 and there is no positive cycle of rules of $\mathcal{P}$ containing R_1, then $\sigma(R_1) = \sigma(R_2)$.
(4) If R_1 and R_2 are two rules of $\mathcal{P}$ appearing in the same positive cycle of rules of $\mathcal{P}$, then $\sigma(R_1) = \sigma(R_2)$.
(5) If R_1 and R_2 are two rules of $\mathcal{P}$ such that the head of R_2 appears in the body of R_1 and there is a positive cycle of rules of $\mathcal{P}$ containing R_1 but no positive cycle of rules of $\mathcal{P}$ containing both R_1 and R_2, then $\sigma(R_2) < \sigma(R_1)$

Proposition 2 *Every Fitting program has a stratification w.r.t. positive cycles.*

Before studying general Fitting programs, we just consider programs which have a stratification w.r.t. positive cycles containing only one stratum.

Proposition 3 *Let $\mathcal{P}$ be a Fitting program that has a stratification w.r.t. positive cycles containing only one stratum, and let v be a valuation over $\mathcal{FOUR}$. Let $\alpha_\mathcal{P}$ be a valuation over $\mathcal{FOUR}$ such that for all positive cycles $\{R_1, ..., R_n\}$ of $\mathcal{P}$, $\alpha_\mathcal{P}(hd(R_1)) = ... = \alpha_\mathcal{P}(hd(R_n))$. Then the sequence defined by $a_0 = \alpha_\mathcal{P}$ and $a_{n+1} = \Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}(a_n, v)$ converges.*

We can now define the operator ${\Psi'}_{\mathcal{P}}^{\alpha_\mathcal{P}}$ derived from $\Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}$.

Definition 6 (The operator ${\Psi'}_{\mathcal{P}}^{\alpha_\mathcal{P}}$) *Let $\mathcal{P}$ be a Fitting program that has a stratification w.r.t. positive cycles $P_1, ..., P_n$ and let v be a valuation over $\mathcal{FOUR}$. Let $\alpha_\mathcal{P}$ be a valuation over $\mathcal{FOUR}$ such that for all positive cycle $\{R_1, ..., R_n\}$ of $\mathcal{P}$, $\alpha_\mathcal{P}(hd(R_1)) = ... = \alpha_\mathcal{P}(hd(R_n))$.*

Then ${\Psi'}_{\mathcal{P}}^{\alpha_\mathcal{P}}(v)$ is the iterated fixpoint of the following sequence :
- a_1 is the iterated fixpoint of the function $\lambda x.\Psi_{\mathcal{P}_1}^{\alpha_\mathcal{P}}(x,v)$ obtained when beginning the computation with $\alpha_\mathcal{P}$.
- a_i is the iterated fixpoint of the function $\lambda x.\Psi_{\mathcal{P}_i}^{\alpha_\mathcal{P}}(x,v)$ obtained when beginning the computation with a_{i-1}.

Generally, a program may have more than one stratification w.r.t. positive cycles. However, the result of the computation does not depend on the stratification used for the computation.

We would like to point out that the notion of stratification and the condition on $\alpha_\mathcal{P}$ that we have seen are indispensable for the convergence of the computation. To illustrate this point, consider the following examples.

Example 2 *Let $\mathcal{P}$ be a Fitting program and $\alpha_\mathcal{P}$ a valuation defined by:*

$$\mathcal{P}\begin{cases} A & \leftarrow & \mathcal{F} \\ B & \leftarrow & \mathcal{F} \\ C & \leftarrow & D \vee A \\ D & \leftarrow & C \vee B \end{cases} \text{ and } \alpha_\mathcal{P} = \begin{pmatrix} A & B & C & D \\ \mathcal{T} & \mathcal{F} & \mathcal{F} & \mathcal{F} \end{pmatrix}$$

If we compute the sequence $a_0 = \alpha_\mathcal{P}$, $a_i = \Psi_{\mathcal{P}}^{\alpha_\mathcal{P}}(a_{i-1}, v)$ then we have

$a_1 = \begin{pmatrix} A & B & C & D \\ \mathcal{F} & \mathcal{F} & \mathcal{T} & \mathcal{F} \end{pmatrix}$, $a_2 = \begin{pmatrix} A & B & C & D \\ \mathcal{F} & \mathcal{F} & \mathcal{F} & \mathcal{T} \end{pmatrix}$, $a_3 = \begin{pmatrix} A & B & C & D \\ \mathcal{F} & \mathcal{F} & \mathcal{T} & \mathcal{F} \end{pmatrix}$ ...

This computation does not terminate. Now, if we consider our definition of ${\Psi'}_{\mathcal{P}}^{\alpha_\mathcal{P}}$, then the computation terminates, and we have:

$$a_1 = \begin{pmatrix} A & B & C & D \\ \mathcal{F} & \mathcal{F} & \mathcal{F} & \mathcal{F} \end{pmatrix} = a_2$$

Example 3 *Let $\mathcal{P}$ be a Fitting program and $\alpha_\mathcal{P}$ a valuation defined by:*

$$\mathcal{P}\left\{\begin{array}{lcl} A & \leftarrow & B \\ B & \leftarrow & A \end{array}\right. \text{ and } \alpha_{\mathcal{P}} = \left(\begin{array}{cc} A & B \\ \mathcal{T} & \mathcal{F} \end{array}\right).$$

The condition that $\alpha_{\mathcal{P}}$ is a valuation over $\mathcal{FOUR}$ such that for all positive cycle $R_1, ..., R_n$ of $\mathcal{P}$, $\alpha_{\mathcal{P}}(hd(R_1)) = ... = \alpha_{\mathcal{P}}(hd(R_n))$ is not satisfied and the computation does not terminate.

3.2 The family of $\alpha_{\mathcal{P}}$-fixed models

In the following, α_β is the valuation that assigns to every literal of $\mathcal{P}$ the value β in $\mathcal{FOUR}$ and $\alpha_{\mathcal{P}}$ verifies the condition: for all positive cycle $R_1, ..., R_n$ of $\mathcal{P}$, $\alpha_{\mathcal{P}}(hd(R_1)) = ... = \alpha_{\mathcal{P}}(hd(R_n))$.

Lemma 1 *If $v = \Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}(v)$ then $v = \Psi^{\alpha_{\mathcal{P}}}_{\mathcal{P}}(v, v)$.*

We recall that a valuation v is a model of a program $\mathcal{P}$ if and only if for all rules $A \longleftarrow B$ in Inst-$\mathcal{P}$, $v(A) \leq_t v(B)$. By definition of $\Psi^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$, a valuation v that verifies $\Psi^{\alpha_{\mathcal{P}}}_{\mathcal{P}}(v, v) = v$ is a model of $\mathcal{P}$. Therefore, if m is a fixpoint of $\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}(v)$, then m is a model of $\mathcal{P}$. So we can define new families of models that we call $\alpha_{\mathcal{P}}$-fixed models.

Definition 7 ($\alpha_{\mathcal{P}}$-fixed models) *Let v be a valuation over $\mathcal{FOUR}$. v is a $\alpha_{\mathcal{P}}$-fixed model of a program $\mathcal{P}$ if and only if v is a fixpoint of $\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$.*

We can consider four particular families corresponding to the four extremal values of $\alpha_{\mathcal{P}}$: $\alpha_{\mathcal{F}}$, $\alpha_{\mathcal{T}}$, $\alpha_{\mathcal{U}}$ and $\alpha_{\mathcal{I}}$. From now on, $\alpha_{\mathcal{F}}$-fixed models will be called *pessimistic*, $\alpha_{\mathcal{T}}$-fixed models *optimistic*, $\alpha_{\mathcal{U}}$-fixed models *skeptical*, and $\alpha_{\mathcal{I}}$-fixed models *inconsistent*. We can now study the family of $\alpha_{\mathcal{P}}$-fixed models.

Theorem 1 *$\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$ is monotonic under $\leq_k$, and anti-monotonic under $\leq_t$.*

Given the monotonicity of $\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$ under the knowledge ordering and the complete lattice structure of $\mathcal{V}(\mathcal{FOUR})$ under this ordering, we can apply the Knaster-Tarski theorem.

Theorem 2 *$\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$ has a least fixpoint , denoted $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{U}}$, and a greatest fixpoint, denoted $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{I}}$, with respect to the knowledge ordering.*[4]

From now on, $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{U}}$ will be called the $\alpha_{\mathcal{P}}$-fixed semantics of $\mathcal{P}$. The $\alpha_{\mathcal{P}}$-fixed semantics corresponding to the four extremal values of $\alpha_{\mathcal{P}}$: $\alpha_{\mathcal{F}}$, $\alpha_{\mathcal{T}}$, $\alpha_{\mathcal{U}}$ and $\alpha_{\mathcal{I}}$, will be called *pessimistic, optimistic, skeptical*, and *inconsistent* semantics, respectively.

The behavior of $\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$ with respect to the truth ordering is less obvious because $\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$ is anti-monotonic under this ordering. However, there is a modification of the Knaster-Tarski theorem dealing with precisely this case:

[4]Actually, $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{U}}$ and $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{I}}$ refer both to program $\mathcal{P}$, and should be denoted as $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{P},\mathcal{U}}$ and $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{P},\mathcal{I}}$, respectively. However, in order to simplify the presentation, we shall omit $\mathcal{P}$ in our notations.

Lemma 2 ([12]) *Suppose that a function f is anti-monotonic on a complete lattice $\mathcal{L}$. Then there are two elements μ and ν of $\mathcal{L}$, called extreme oscillation points of f, such that the following hold:*
- μ and ν are the least and greatest fixpoint of f^2 (i.e. of f composed with itself);
- f oscillates between μ and ν in the sense that $f(\mu) = \nu$ and $f(\nu) = \mu$;
- if x and y are also elements of $\mathcal{L}$ between which f oscillates then x and y lie between μ and ν.

Under the truth ordering, $\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$ is anti-monotonic and $\mathcal{V}(\mathcal{FOUR})$ is a complete lattice, so $\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$ has two extreme oscillation points under this ordering.

Proposition 4 *$\Psi'^{\alpha_{\mathcal{P}}}_{\mathcal{P}}$ has two extreme oscillation points denoted $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{F}}$ and $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{T}}$, with $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{F}} \leq_t Fix^{\alpha_{\mathcal{P}}}_{\mathcal{T}}$, under the truth ordering.*

We can now extend a result presented in [8] from values of $\mathcal{FOUR}$ to a valuation $\alpha_{\mathcal{P}}$.

Theorem 3 *Let $\mathcal{P}$ be a Fitting program. Then we have:*

$$\begin{array}{lcll} Fix^{\alpha_{\mathcal{P}}}_{\mathcal{U}} & = & Fix^{\alpha_{\mathcal{P}}}_{\mathcal{F}} \otimes Fix^{\alpha_{\mathcal{P}}}_{\mathcal{T}} & , \\ Fix^{\alpha_{\mathcal{P}}}_{\mathcal{I}} & = & Fix^{\alpha_{\mathcal{P}}}_{\mathcal{F}} \oplus Fix^{\alpha_{\mathcal{P}}}_{\mathcal{T}} & , \\ Fix^{\alpha_{\mathcal{P}}}_{\mathcal{F}} & = & Fix^{\alpha_{\mathcal{P}}}_{\mathcal{U}} \wedge Fix^{\alpha_{\mathcal{P}}}_{\mathcal{I}} & , \\ Fix^{\alpha_{\mathcal{P}}}_{\mathcal{T}} & = & Fix^{\alpha_{\mathcal{P}}}_{\mathcal{U}} \vee Fix^{\alpha_{\mathcal{P}}}_{\mathcal{I}} & . \end{array}$$

The family of $\alpha_{\mathcal{P}}$-fixed models of a program is bounded for each $\alpha_{\mathcal{P}} \in \mathcal{V}(\mathcal{FOUR})$ as follows: in the knowledge ordering, all $\alpha_{\mathcal{P}}$-fixed models are between $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{U}}$ and $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{I}}$ which are the least and greatest $\alpha_{\mathcal{P}}$-fixed models, respectively; in the truth ordering, all $\alpha_{\mathcal{P}}$-fixed models are between $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{F}}$ and $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{T}}$ which are not necessarily $\alpha_{\mathcal{P}}$-fixed models of $\mathcal{P}$.

It is interesting to note that for $\alpha_{\mathcal{P}} = \alpha_{\mathcal{F}}$ the first equality of theorem 3 relates two different definitions of the well-founded semantics: the left-hand side, $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{U}}$, represents the definition of Przymusinski [10] via three-valued stable models, whereas the right-hand side, $Fix^{\alpha_{\mathcal{P}}}_{\mathcal{F}} \otimes Fix^{\alpha_{\mathcal{P}}}_{\mathcal{T}}$, represents the definition of Van Gelder via alternating fixpoints [11]. We note here that Fitting, working with bilattices, generalized the approaches of Van Gelder and Przymusinski [4].

4 Comparing the usual semantics of logic programs

The following theorem states that the family of pessimistic (resp. optimistic, skeptical, inconsistent) fixed models under uniform assumption presented in [8] is included in the family of pessimistic (resp. optimistic, skeptical, inconsistent) fixed models presented in this paper.

Theorem 4 *Let $\mathcal{P}$ be a Fitting program and $\beta \in \{\mathcal{F}, \mathcal{T}, \mathcal{U}, \mathcal{I}\}$. Then we have $Fix^{\alpha_\beta}_{\mathcal{U}} = Fix^{\beta}_{\mathcal{U}}$.*

We would like to point out that the computation of $Fix_{\mathcal{U}}^{\alpha_\beta}$ uses a notion of program stratification. This is not the case of the computation of $Fix_{\mathcal{U}}^{\beta}$. However, the two methods compute the same semantics.

The family of pessimistic models presented in [8] extends stable models from conventionnal logic programs to Fitting programs; the well-founded semantics and the Kripke-Kleene semantics are also captured (and similarly extended) by the appproach of [8]. Consequently, they are also captured by our approach.

Corollary 1 *Let $\mathcal{P}$ be a conventional logic program.*
(1) If v is a three-valued stable model of $\mathcal{P}$, then v is a pessimistic fixed model.
(2) If v is the well-founded semantics of $\mathcal{P}$, then v = $Fix_{\mathcal{U}}^{\alpha_\mathcal{F}}$;
(3) If v is the Kripke-Kleene semantics of $\mathcal{P}$, then v= $Fix_{\mathcal{U}}^{\alpha_\mathcal{U}}$.

It is important to recall here that, in our approach, positive and negative information are treated separately during the computation of $Fix_{\mathcal{U}}^{\alpha_\mathcal{U}}$. This is not the case with the computation of Kripke-Kleene semantics. Nevertheless, when we restrict our attention to conventional programs, the two methods compute the same semantics. Our approach unifies the computation of the semantics used in conventional logic programming, and thus it allows us to compare those semantics:

Theorem 5 *Let $\mathcal{P}$ be a Fitting program.*

$$\text{If } \alpha_\mathcal{P} \leq_k \alpha'_\mathcal{P} \text{ then } Fix_{\mathcal{U}}^{\alpha_\mathcal{P}} \leq_k Fix_{\mathcal{U}}^{\alpha'_\mathcal{P}}$$

From this result, we can infer that $Fix_{\mathcal{U}}^{\alpha_\mathcal{U}} \leq_k Fix_{\mathcal{U}}^{\alpha_\mathcal{P}}$, for any $\alpha_\mathcal{P}$.

That is, the skeptical semantics gives less information than any other $\alpha_\mathcal{P}$-fixed semantics and in particular than the pessimistic and optimistic semantics. From this theorem, we can infer the following result:

Corollary 2 *Let $\mathcal{P}$ be a Fitting program. Then we have:*

$$Fix_{\mathcal{U}}^{\alpha_\mathcal{U}} \leq_k Fix_{\mathcal{U}}^{\alpha_\mathcal{P}} \otimes Fix_{\mathcal{U}}^{\neg\alpha_\mathcal{P}}.$$

The equality is satisfied for positive programs, but if we accept negation then it is false in general. This corollary suggests the possibility of defining a new semantics, namely $Fix_{\mathcal{U}}^{\alpha_\mathcal{P}} \otimes Fix_{\mathcal{U}}^{\neg\alpha_\mathcal{P}}$, which is smaller than the pessimistic and optimistic semantics but greater than the skeptical semantics. The following example shows that this semantics can be useful in certain contexts.

Example 4 *Let $\mathcal{P}$ be the program: $\mathcal{P} : \mathcal{A} \longleftarrow \mathcal{B} \vee \neg\mathcal{B}$*

Sem of $\mathcal{P}$	$Fix_{\mathcal{U}}^{\alpha_\mathcal{F}}$	$Fix_{\mathcal{U}}^{\alpha_\mathcal{T}}$	$Fix_{\mathcal{U}}^{\alpha_\mathcal{U}}$	$Fix_{\mathcal{U}}^{\alpha_\mathcal{F}} \otimes Fix_{\mathcal{U}}^{\alpha_\mathcal{T}}$
A	$\mathcal{T}$	$\mathcal{T}$	$\mathcal{U}$	$\mathcal{T}$
B	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{U}$	$\mathcal{U}$

The program $\mathcal{P}$ asserts that A is always true (because it is inferred from either B or $\neg B$), and this conclusion is reached by both the optimistic and the pessimistic semantics. Intuitively however, there is no reason why we should choose between B $true$ and B $false$ when we cannot assert anything about the value of B. It seems therefore more natural in this case to take the consensus between the pessimistic and optimistic semantics, which gives the value $unknown$ to B.

Although this seems to give the right semantics in our example, one has to check under what conditions $Fix_{\mathcal{U}}^{\alpha_\mathcal{P}} \otimes Fix_{\mathcal{U}}^{\neg\alpha_\mathcal{P}}$ is actually a model. Assuming that it is a model, we can call it the *consensus semantics*.

5 Conclusion

We have seen a general framework for reasoning about information integration in a distributed environment. Our framework uses parametrized semantics of Fitting programs under non-uniform assumptions for the missing information. We have also seen that when we restrict our framework to uniform assumptions only, then our approach captures the semantics of conventional logic programs. We have exploited this fact in order to compare conventional semantics among them and also to show how to combine them in order to define new semantics such as the consensus semantics that we have seen in this paper.

Extending this work to general bilattices and logics with signs and annotations is a topic for future work.

References

[1] BELNAP, N. D., Jr, *A Useful Four-Valued Logic*, in: J. M. Dunn and G. Epstein (eds.), Modern Uses of Multiple-valued Logic, D. Reichel, Dordrecht, 1977.

[2] FITTING, M. C., *A Kripke/Kleene Semantics for Logic Programs*, J. Logic Programming, 2:295-312 (1985).

[3] FITTING, M. C., *Bilattices and the Semantics of Logic Programming*, J. Logic Programming, 11:91-116 (1991).

[4] FITTING, M. C., *The Family of Stable Models*, J. Logic Programming, 17:197-225 (1993).

[5] GELFOND, M. and LIFSCHITZ, V., *The Stable Model Semantics for Logic Programming*, in: R. Kowalski and K. Bowen (eds.), Proceedings of the Fifth Logic Programming Symposium MIT Press, Cambridge, MA, 978-992, 1988.

[6] GINSBERG, M. L., *Multivalued Logics: a Uniform Approach to Reasonning in Artificial Intelligence*, Computationnal Intelligence, 4:265-316, 1988.

[7] GINSBERG, M. L., *Bilattices and modal operators*, J. of Logic Computation, 1:41-69, 1990.

[8] LOYER, Y., SPYRATOS, N., STAMATE, D., *Computing and Comparing Semantics of Programs in Four-valued Logics*, in: M. Kutylowski, L. Pacholski and T. Wierzbicki (eds.), Mathematical Foundations of Computer Science ($MFCS'99$), LNCS 1672, Springer Verlag, Szklarska Poreba, Poland, 1999.

[9] LOYER, Y., SPYRATOS, N., STAMATE, D., *Integration of Information in Four-valued Logics under Non-Uniform Assumptions*, Technical Report, to appear.

[10] PRZYMUSINSKI, T. C., *Well-Founded Semantics Coincides with Three-Valued Stable Semantics*, Fund. Inform., 13:445-463, 1990.

[11] VAN GELDER, A., ROSS, K. A., SCHLIPF, J. S., *The Well-Founded Semantics for General Logic Programs*, J. ACM, 38:620-650, 1991.

[12] YABLO, S., *Truth and Reflection*, J. Philos. Logic, 14:297-349, 1985.