Minimal and Absent Information in Contexts

Floris Roelofsen[1] and **Luciano Serafini**[2]
[1] Institute for Logic, Language, and Computation, Amsterdam
[2] Instituto per la Ricerca Scientifica e Tecnologica, Trento

Abstract

Multi-context systems (MCS) represent contextual information flow. We show that the semantics of an MCS is completely determined by the information that is obtained when simulating the MCS, in such a way that a *minimal* amount of information is deduced at each step of the simulation.

In MCS, the acquisition of new information is based on the *presence* of other information only. We give a generalized account to model situations in which information can be obtained as a result of the *absence* of other information as well.

1 Introduction

Based on motivational papers by McCarthy [1987] and Giunchiglia [1993] several formalizations of contextual information and inter-contextual information flow have been proposed. Most notable are the propositional logic of context developed by McCarthy, Buvač and Mason [1993; 1998], and the multi-context systems devised by Giunchiglia and Serafini [1994], which later have been associated with the local model semantics introduced by Giunchiglia and Ghidini [2001]. Serafini and Bouquet [2004] have argued from a technical point of view that multi-context systems constitute the most general formal framework. This conclusion is supported by a more conceptual argument of Benerecetti et.al. [2000].

A multi-context system describes the information available in a number of contexts (i.e., to a number of people / agents / databases, etc.) and specifies the information flow between those contexts. The local model semantics defines a system to entail a piece of information, if and only if that piece of information is acquired, independently of how the information flow described by the system is accomplished.

Our first contribution is based on the observation that the local model semantics of a multi-context system is completely determined by the information that is obtained when simulating the information flow specified by the system, in such a way that a *minimal* amount of information is deduced at each step of the simulation. We define an operator which suitably implements such a simulation, and thus determines the information entailed by the system. This operator constitutes a first constructive account of the local model semantics.

Our second contribution is based on the observation that in multi-context systems, new information is derived based on the *presence* of other information only. However, in many natural situations (concrete examples will be given below), new information is obtained due to a *lack* of other information. We propose a generalized framework so as to account for such situations. Non-monotonic reasoning techniques are applied to formulate a suitable semantics for this framework.

We proceed, in section 2, with a brief review of multi-context system syntax and local model semantics. Minimality is discussed in section 3, and the generalized framework is presented in section 4. We conclude, in section 5, with a concise recapitulation of our main observations and results.[1]

2 Preliminaries

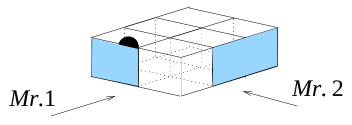

Figure 1: a magic box.

A simple illustration of the main intuitions underlying the multi-context system framework is provided by the situation depicted in figure 1. Two agents, Mr.1 and Mr.2, are looking at a box from different angles. The box is called magic, because neither Mr.1 nor Mr.2 can make out its depth. As some sections of the box are out of sight, both agents have partial information about the box. To express this information, Mr.1 only uses proposition letters l (there is a ball on the left) and r (there is a ball on the right), while Mr.2 also uses a third proposition letter c (there is a ball in the center).

In general, we consider a set of contexts I, and a language L_i for each context $i \in I$. We assume I and $\{L_i\}_{i \in I}$ to be fixed, unless specified otherwise. Moreover, for the purpose of this paper we assume each L_i to be built over a finite set of proposition letters, using standard propositional connectives.

To state that the information expressed by a formula $\varphi \in L_i$ is established in context i we use so-called *labeled formulas* of the form $i : \varphi$ (if no ambiguity arises, we simply refer

[1] See `http://home.student.uva.nl/f.roelofsen/` for all proofs that are omitted or merely sketched here.

to labeled formulas as formulas, and we even use capital letters F, G, and H to denote labeled formulas, if the context label is irrelevant). A *rule* r is an expression of the form:

$$F \leftarrow G_1 \wedge \ldots \wedge G_n \tag{1}$$

where F and all G's are labeled formulas; F is called the consequence of r and is denoted by $cons(r)$; all G's are called premises of r and together make up the set $prem(r)$. Rules without premises are called *facts*. Rules with at least one premiss are called *bridge rules*. A *multi-context system* (system hereafter) is a finite set of rules. A fact describes information that is established in a certain context, independently of which information is obtained in other contexts. A bridge rule specifies which information is obtained in one context, if other pieces of information are acquired in different contexts. Thus a system can be thought of as a specification of contextual information and an inter-contextual information flow.

Example 1 *The situation in figure 1 can be modeled by the following system:*

$$\begin{array}{lll}
1 : \neg r & \leftarrow & \\
2 : l & \leftarrow & \\
1 : l \vee r & \leftarrow & 2 : l \vee c \vee r \\
2 : l \vee c \vee r & \leftarrow & 1 : l \vee r
\end{array}$$

Mr.1 knows that there is no ball on the right, Mr.2 knows that there is a ball on the left, and if any agent gets to know that there is a ball in the box, then he will inform the other agent.

A classical interpretation m of language L_i is called a *local model* of context i. A set of local models is called a *local information state*. Intuitively, every local model in a local information state represents a "possible state of affairs". If a local information state contains exactly one local model, then it represents complete information. If it contains more than one local model, then it represents partial information: more than one state of affairs is considered possible. A *distributed information state* is a set of local information states, one for each context. In conformity with the literature, we will refer to distributed information states as *chains*.

Example 2 *The situation in figure 1, in which Mr.1 knows that there is no ball on the right but does not know whether there is a ball on the left, is represented by a chain whose first component* $[\ \{l, \neg r\}, \{\neg l, \neg r\}\ ]$ *contains two local models. As such, the chain reflects Mr.1's uncertainty about the left section of the box.*

A chain c *satisfies* a labeled formula $i : \varphi$ (denoted $c \models i : \varphi$) iff all local models in its i^{th} component classically satisfy φ. A rule r is *applicable* with respect to a chain c iff c satisfies every premiss of r. Notice that facts are applicable with respect to any chain. A chain c *complies with* a rule r, iff, whenever r is applicable with respect to c, then c satisfies r's consequence. We call c a *solution chain* of a system S iff it complies with every rule in S. A formula F is *true* in S (denoted $S \models F$) iff every solution chain of S satisfies F.

Let **C** denote the set of all chains. Notice that, as each L_i is assumed to be built over a finite set of proposition letters, **C** is assumed to be finite. Let $c^\perp$ denote the chain containing every local model of every context (notice that $c^\perp$ does not satisfy any non-tautological expression); let $c^\top$ denote the chain containing no local models at all (notice that $c^\top$ satisfies any expression). If C is a set of chains, then the component-wise union of C is the chain, whose i^{th} component consists of all local models that are in the i^{th} component of some chain in C. If c and c' are chains, then $c \setminus c'$ denotes the chain, whose i^{th} component consists of all local models that are in c_i but not in c'_i. Finally, we sometimes say that a local model m is (not) in c, when we actually mean that m is (not) in some (any) component c_i of c.

3 Minimality

We order chains according to the amount of information they convey. Intuitively, the more local models a chain component contains, the more possibilities it permits, so the less informative it is. Formally, we say that c is *less informative* than c' ($c \preceq c'$), if for every i we have $c_i \supseteq c'_i$. If, moreover, for at least one i we have $c_i \supset c'_i$, then we say that c is *strictly less informative* than c' ($c \prec c'$).

Lemma 1 *Let c and c' be two chains, such that $c \preceq c'$. Then any formula that is satisfied by c is also satisfied by c'.*

We call c *minimal* among a set of chains C, iff c is in C and no other chain c' in C is strictly less informative than c. In particular, we call c a *minimal solution chain* of a system S, if it is minimal among the set of all solution chains of S.

Theorem 1 *Every system S has a unique minimal solution chain c_S.*

Proof. Let C_S be the set of all solution chains of S. $C_S \neq \emptyset$ as $c^\top \in C_S$ for any S. Let c_S be the component-wise union of all chains in C_S. Then $c_S \in C_S$ and $c_S \preceq c$ for any $c \in C_S$. So c_S is the unique minimal solution chain of S. □

Theorem 2 *The semantics of a system S is completely determined by its unique minimal solution chain c_S:*

$$S \models F \quad \Leftrightarrow \quad c_S \models F$$

Proof. F is true in S iff F is satisfied by all solution chains of S iff F is satisfied by the component-wise union c_S of all solution chains of S. □

Theorem (1) and (2) are extremely useful, because they establish that, to answer queries about a system S, it is no longer necessary to compute all solution chains of S; we only need to consider the system's minimal solution chain c_S.

3.1 Computing the Minimal Solution Chain

The minimal solution chain of S can be characterized as the $\preceq$-least fixpoint of an operator $\mathbf{T}_S$, which, intuitively, simulates the information flow specified by S. Let $S^*(c)$ denote the set of rules in S, which are applicable w.r.t. c. Then we define:

$$\mathbf{T}_S(c) = c \setminus \{m \mid \exists r \in S^*(c) : m \not\models cons(r)\} \tag{2}$$

For every rule r in S that is applicable w.r.t. c, $\mathbf{T}_S$ removes from c all local models that do not satisfy $cons(r)$.

Intuitively, this corresponds to augmenting c with the information expressed by $cons(r)$. In this sense, $\mathbf{T}_S$ simulates the information flow described by S. As $\mathbf{T}_S(c)$ is obtained from c only by *removing* local models from it, $\mathbf{T}_S(c)$ is always more informative than c.

As $(\mathbf{C}, \preceq)$ forms a complete lattice, and $\mathbf{T}_S$ is monotone and continuous w.r.t. $\preceq$, [Tarski, 1955] yields:

Theorem 3 *$\mathbf{T}_S$ has a $\preceq$-least fixpoint, which is obtained after a finite number of consecutive applications of $\mathbf{T}_S$ to $c^{\perp}$.*

Lemma 2 *Let c be a chain and let S be a system. Then c is a fixpoint of $\mathbf{T}_S$ if and only if c is a solution chain of S.*

Theorem 4 *Let S be a system. Then the minimal solution chain c_S of S coincides with the $\preceq$-least fixpoint of $\mathbf{T}_S$.*

From theorems 3 and 4 we conclude that the minimal solution chain c_S of a system S is obtained after a finite number of applications of $\mathbf{T}_S$ to the least informative chain $c^{\perp}$. But we can even prove a slightly stronger result:

Theorem 5 *Let S be a system and let $|S|$ denote the number of* bridge rules *in S. Then the minimal solution chain c_S of S is obtained after at most $|S|+1$ applications of $\mathbf{T}_S$ to $c^{\perp}$.*

In fact, a slightly more involved, but essentially equivalent procedure was introduced for rather different reasons by Roelofsen et.al. [2004]. This procedure was shown to have worst-case time complexity $O(|S|^2 \times 2^M)$, where M is the maximum number of propositional variables in either one of the contexts involved in S.

Example 3 *Consider the system from example 1. Applying $\mathbf{T}_S$ to $c^{\perp}$ establishes the facts given by the first two rules of the system. But then Mr.2 knows that there is a ball in the box, so the next application of $\mathbf{T}_S$ simulates the information flow specified by the third rule of the system: Mr.2 informs Mr.1 of the presence of the ball. The resulting chain is left unaltered by any further application of $\mathbf{T}_S$, and therefore constitutes the minimal solution chain of S. The fact that this chain satisfies the formula $1 : l$ reflects, as desired, that Mr.1 has come to know that there is a ball in the left section of the box.*

4 Absent Information

Rules of the form (1) only allow us to model a rather restricted kind of information flow, namely one in which new information is established based on the *presence* of other information only. There are many natural situations in which information is obtained as a result of the *absence* of other information. Such situations cannot be modeled by the present formalism.

Example 4 (Coordination) *Let d_1, d_2 be two meteorological databases that collect their respective data from sensors located in different parts of the country. At the end of the day each database produces a weather forecast based on its own data but also on the information obtained by the other database. For example, d_1 predicts rain, if that follows from its own data and if, moreover, d_2 does not maintain that it won't rain:*

$$1 : r \leftarrow 1 : r \wedge \textbf{not}\ 2 : \neg r$$

Example 5 (Integration) *Let d_1 and d_2 be as in example 4 and let d_3 be a third database, which integrates the information obtained in d_1 and d_2, respectively. Any piece of information that is established by d_1 and not refuted by d_2 (or vice versa) is included in d_3:*

$$\begin{aligned} 3 : \varphi &\leftarrow 1 : \varphi \wedge \textbf{not}\ 2 : \neg\varphi \\ 3 : \varphi &\leftarrow 2 : \varphi \wedge \textbf{not}\ 1 : \neg\varphi \end{aligned}$$

Example 6 (Trust) *Let d_1, d_2, and d_3 be as in example 5. It would be natural for d_3 to regard d_1 as more trustworthy than d_2 (or vice versa). In this case any piece of information that is established in d_1 is automatically included in d_3, but information obtained in d_2 is only included in d_3 if it is not refuted by d_1:*

$$\begin{aligned} 3 : \varphi &\leftarrow 1 : \varphi \\ 3 : \varphi &\leftarrow 2 : \varphi \wedge \textbf{not}\ 1 : \neg\varphi \end{aligned}$$

To model these situations we need rules r of the form:

$$F \leftarrow G_1 \wedge \ldots \wedge G_m \wedge \textbf{not}\ H_1 \wedge \ldots \wedge \textbf{not}\ H_n \qquad (3)$$

where F, all G's, and all H's are labeled formulas. As before, F is called the consequence of r ($cons(r)$). $G_1, \ldots, G_m$ are called *positive premises* of r and together constitute the set $prem^{+}(r)$. $H_1, \ldots, H_n$ are called *negative premises* of r and make up the set $prem^{-}(r)$. A rule does not necessarily have any premises ($m, n \geq 0$). In analogy with commonplace terminology in deductive database and logic programming theory, we call such rules *normal* rules, and finite sets of them *normal multi-context systems* (normal systems for short). If a rule only has positive premises, we call it a *positive* rule. Note that a system, which consists of positive rules only conforms with the original definition of multi-context systems. From now on we call such systems *positive* systems.

Our aim is to generalize the result obtained section 3, i.e. to define the semantics of a normal system S in terms of a single *canonical* chain c_S of S, such that, whenever S is a positive system, c_S coincides with the minimal solution chain of S.

A first naive attempt would be to say that a chain c complies with a normal rule r iff it satisfies r's consequence, whenever it satisfies every positive premise of r and does not satisfy any negative premise of r. The (minimal) solution chains of a normal system can then be defined as for positive systems. However, as the following example shows, a normal system does not generally have a unique minimal solution chain, and worse, its minimal solution chains do not generally correspond with its intended meaning.

Example 7 *Let a system S be given by the following rule:*

$$1 : p \leftarrow \textbf{not}\ 2 : q$$

Then S has two minimal solution chains:

$$\begin{aligned} c^p &= \left\{ \begin{bmatrix} \{p\} \\ \phantom{\{\neg p\}} \end{bmatrix}_1 \begin{bmatrix} \{q\} \\ \{\neg q\} \end{bmatrix}_2 \right\} \\ c^q &= \left\{ \begin{bmatrix} \{p\} \\ \{\neg p\} \end{bmatrix}_1 \begin{bmatrix} \{q\} \\ \phantom{\{\neg q\}} \end{bmatrix}_2 \right\} \end{aligned}$$

Intuitively, S provides no ground for deriving q in context 2. Thus, p should be derived in context 1, and every "proper" canonical chain of S should satisfy $1 : p$. As c^q fails to do so, it should be rejected as such. But how, then, should the canonical chain of a normal system be characterized?

Extensive research efforts have been involved with an analogous question in the setting of logic programming, when, in the late 80's / early 90's, a proper semantics for normal logic programs was sought. In motivating our characterization of canonical chains for normal multi-context systems, we will recall some important intuitions and adapt some crucial definitions that have resulted from these efforts.

A first desired property of canonical chains, introduced in the setting of logic programming by Apt et.al. [1988] and Bidoit and Froidevaux [1991], is termed *supportedness*. Intuitively, a chain c is a supported solution chain of a normal system S iff, whenever c satisfies a formula F, then S *provides an explanation for why this is so*.

Definition 1 *We call a chain c a supported solution chain of a normal system S iff, whenever c satisfies a formula F, then S contains a set R of rules, such that:*

- $\forall r \in R : \begin{cases} \forall G \in prem^{+}(r) : c \models G \\ \forall H \in prem^{-}(r) : c \not\models H \end{cases}$
- $\bigcup_{r \in R} cons(r) \models F$

Example 8 *In example 7, as desired, c^p is a supported solution chain of S, while c^q is not. But both c^p and c^q are supported solution chains of the following extension S' of S:*

$$\begin{array}{rcl} 1:p & \leftarrow & \textbf{not}\ 2:q \\ 2:q & \leftarrow & 2:q \end{array}$$

Intuitively, c^p should be accepted as a canonical chain of S', but c^q should be rejected as such, because the explanation provided by S' for the fact that c^q satisfies $2 : q$ is *circular*, i.e., it relies on the very fact that c^q satisfies $2 : q$. So, in general, the concept of supportedness does not satisfactorily characterize the canonical chain of a normal system.

The notion of *well-supportedness*, first introduced for logic programs by Fages [1991], refines the notion of supportedness to avoid the counter-intuitive result obtained in example 8. Intuitively, a chain c is a well-supported solution chain of a normal system S iff, whenever c satisfies a formula F, then S provides a *non-circular* explanation for why this is so.

Fages also proved this notion to be equivalent to the notion of *stability*, which had been defined somewhat earlier by Gelfond and Lifschitz [1988]. The results obtained in section 3 pave the way for a straightforward adaptation of the notion of stability to our present setting.

Definition 2 *Let c be a chain and S a normal system. Define:*

$$\begin{array}{rcl} S'(c) & = & \{r \in S \mid \forall H \in prem^{-}(r) : c \not\models H\} \\ S''(c) & = & pos(S'(c)) \end{array}$$

where $pos(S'(c))$ is obtained from $S'(c)$ by removing all negative premises from its rules. Then, c is a stable solution chain of S, iff it is the unique minimal solution chain of $S''(c)$.

Intuitively, a solution chain c of a system S is stable iff, whenever the information represented by c is assumed, then the information flow specified by S reproduces exactly c. Namely, if c is assumed to contain valid information, then any rule in S, one of whose negative premises is satisfied by c, is certainly not applicable w.r.t. c. Negative premises which are *not* satisfied by c can be removed from the remaining rules, because they do certainly not inhibit those rules from being applicable w.r.t. c. Thus, c is stable iff it corresponds exactly to the meaning of $S''(c)$, i.e., by Theorem 2, to its minimal solution chain.

Example 9 *In example 8, as desired, c^p is a stable solution chain of S', while c^q is not.*

For many systems, stability suitably characterizes a unique canonical chain. There are still some special cases, however, in which it fails to do so. We give some typical examples.

Example 10 *Both c^p and c^q from example 7 are stable solution chains of the system given by the following rules:*

$$\begin{array}{rcl} 1:p & \leftarrow & \textbf{not}\ 2:q \\ 2:q & \leftarrow & \textbf{not}\ 1:p \end{array}$$

Example 11 *The following system does not have any stable solution chains.*

$$1:p \quad \leftarrow \quad \textbf{not}\ 1:p$$

In both cases we think it is most reasonable to conclude that no information is derived at all, i.e. to regard $c^{\perp}$ as the proper canonical chain.

Example 12 *The following system does not have any stable solution chains either.*

$$\begin{array}{rcl} 1:p & \leftarrow & \textbf{not}\ 1:p \\ 1:t & \leftarrow & \textbf{not}\ 2:q \\ 2:r & \leftarrow & 1:t \end{array}$$

Example 12 illustrates that, even if the rest of the system is unproblematic, one single rule (in this case the first one) can cause the system not to have any stable solution chain at all. In this case, t and r should be derived in context 1 and 2, resp.

The *well-founded semantics*, first proposed for logic programs by van Gelder et.al. [1991] avoids the problems encountered in the above examples. The well-founded model of a program is defined as the least fixpoint of an operator, which, given an interpretation, determines the atoms that are necessarily true and those that are necessarily not true with respect to the program and the interpretation. It assigns $true$ to the former set of atoms, and $false$ to the latter. As a result, more atoms may become necessarily true or necessarily not true. Corresponding truth values are assigned until a fixpoint is reached. All atoms that have not been assigned a definite truth value, are interpreted as *unknown*.

Our approach shares an important intuition with the well-founded semantics for logic programs, namely, that while constructing the canonical chain of a system, it is not only important to accumulate the information that *can certainly be derived* from the system, but also to keep track of information that *can certainly not be derived* from the system.

But the two approaches are also fundamentally different. The well-founded semantics constructs a 3-valued interpretation I, which is minimal with respect to a *truth order* $\sqsubseteq$ (i.e. $I \sqsubseteq I'$ iff I makes less atoms true and more atoms false than I'), whereas we seek a chain which is minimal with respect to an *information order* $\preceq$ (i.e. $c \preceq c'$ iff c makes less expressions either true or false than c'). This particularly results in a different treatment of expressions that are found *not* to be true. To regard these expressions as false, as the well-founded semantics does, would be to introduce redundant information. Instead, in our setting, such expressions should simply be recorded as not being derivable.

4.1 Constructing the Canonical Chain

The canonical chain of a normal system S, henceforward denoted by c_S, is constructed by an iterative transformation of a datastructure $\langle c, a\rangle$, where:

- c is the "canonical chain under construction". Initially, $c = c^{\perp}$. Every transformation of c removes from it those local models that are found not to be in c_S. So at any phase of the construction of c_S, c contains those local models that are *possibly* in c_S, and as such represents the information that is *necessarily* conveyed by c_S.
- a is the "anti-chain". Initially, $c = c^{\top}$. Every transformation of a adds to it those local models that are found to be in c_S. So at any phase of the construction of c_S, a contains those local models that are *necessarily* in c_S, and as such represents the information that is *possibly* conveyed by c_S.

Observation 1 *By construction, we have* $c \preceq c_S \preceq a$. *Therefore, by lemma 1, for any formula* F*:*

$$\begin{array}{rcl} c \models F & \Rightarrow & c_S \models F \\ a \not\models F & \Rightarrow & c_S \not\models F \end{array}$$

□

We call a chain-anti-chain pair $\langle c, a\rangle$ *less evolved* than another such pair $\langle c', a'\rangle$ (denoted as $\langle c, a\rangle \leq \langle c', a'\rangle$) iff c is less informative than c' and a is more informative than a'. If, moreover, c is strictly less informative than c' or a is strictly more informative than a', then we say that $\langle c, a\rangle$ is strictly less evolved than $\langle c', a'\rangle$. We call $\langle c, a\rangle$ *minimal* among a set $\mathcal{CA}$ of chain-anti-chain pairs, iff $\langle c, a\rangle \in \mathcal{CA}$ and no other chain-anti-chain pair $\langle c', a'\rangle$ in $\mathcal{CA}$ is strictly less evolved than $\langle c, a\rangle$. Notice that, if $\langle c, a\rangle$ is minimal among $\mathcal{CA}$, then c is minimal among $\{c \mid \langle c, a\rangle \in \mathcal{CA}\}$.

Given a certain chain-anti-chain pair $\langle c, a\rangle$, the intended transformation $\mathbf{\Psi}_S$ first determines which rules in S will (not) be applicable w.r.t. c_S, and then refines $\langle c, a\rangle$ accordingly. The canonical chain c_S of S will be characterized as the first component of the $\leq$-least fixpoint of $\mathbf{\Psi}_S$.

We first specify how $\mathbf{\Psi}_S$ determines which rules will (not) be applicable w.r.t. c_S. Let $\langle c, a\rangle$ and a rule r in S be given. If r has a positive premise G, which is satisfied by c, then G will also be satisfied by c_S. On the other hand, if r has a negative premiss H, which is *not* satisfied by a, then H will not be satisfied by c_S either. So if all positive premises of r are satisfied by c and all negative premises of r are not satisfied by a, then r will be applicable with respect to c_S:

$$S^{+}(c,a) = \left\{ r \in S \;\middle|\; \begin{array}{c} \forall G \in prem^{+}(r) : c \models G \\ \text{and} \\ \forall H \in prem^{-}(r) : a \not\models H \end{array} \right\}$$

If r has a positive premise G, which is not satisfied by a, then G will not be satisfied by c_S either. If r has a negative premise H, which is satisfied by c, then H will be satisfied by c_S as well. In both cases r will certainly not be applicable with respect to c_S:

$$S^{-}(c,a) = \left\{ r \in S \;\middle|\; \begin{array}{c} \exists G \in prem^{+}(r) : a \not\models G \\ \text{or} \\ \exists H \in prem^{-}(r) : c \models H \end{array} \right\}$$

For convenience, we write:

$$S^{\sim}(c,a) = S \setminus S^{-}(c,a)$$

Think of $S^{\sim}(c,a)$ as the set of rules that are *possibly* applicable with respect to c_S, and notice that $S^{+}(c,a) \subseteq S^{\sim}(c,a)$, whenever $c \preceq a$, and that $S^{+}(c,a) = S^{\sim}(c,a)$, if $c = a$.

Lemma 3 *If* S *is a normal system and* $\langle c, a\rangle$ *and* $\langle c', a'\rangle$ *are two chain-anti-chain pairs s.t.* $\langle c, a\rangle \leq \langle c', a'\rangle$*, then we have:*

1. $S^{+}(c,a) \subseteq S^{+}(c',a')$
2. $S^{-}(c,a) \subseteq S^{-}(c',a')$
3. $S^{\sim}(c,a) \supseteq S^{\sim}(c',a')$

Proof. Suppose that $\langle c, a\rangle \leq \langle c', a'\rangle$. Then, by definition, $c \preceq c'$ and $a' \preceq a$. Let r be a rule in S. For the first statement, suppose that $r \in S^{+}(c,a)$. Then c satisfies all of r's positive premises, and a does not satisfy any of r's negative premises. By lemma 1, the same goes for c' and a', respectively, which implies that $r \in S^{+}(c',a')$. The second statement is proven analogously; the third follows directly from the second. □

Next, we specify how $\mathbf{\Psi}_S$ refines $\langle c, a\rangle$, based on $S^{+}(c,a)$ and $S^{\sim}(c,a)$. Every local model $m \in c_i$ that does not satisfy the consequence of a rule in $S^{+}(c,a)$ should certainly not be in c_S and is therefore removed from c. On the other hand, every local model $m \in c_i$ that satisfies the consequences of every rule in $S^{\sim}(c,a)$ should certainly be in c_S (S provides no ground for removing it) and is therefore added to a.

$$\mathbf{\Psi}_S(\langle c, a\rangle) = \langle \mathbf{\Psi}^{c}_S(\langle c, a\rangle), \mathbf{\Psi}^{a}_S(\langle c, a\rangle)\rangle$$

where:

$$\begin{array}{rcl} \mathbf{\Psi}^{c}_S(\langle c, a\rangle) & = & c \setminus \{m \mid \exists r \in S^{+}(c,a) : m \not\models cons(r)\} \\ \mathbf{\Psi}^{a}_S(\langle c, a\rangle) & = & a \cup \{m \mid \forall r \in S^{\sim}(c,a) : m \models cons(r)\} \end{array}$$

As $(\mathbf{C} \times \mathbf{C}, \leq)$ forms a complete lattice, and $\mathbf{\Psi}_S$ is monotone and continuous w.r.t. $\leq$, [Tarski, 1955] yields:

Theorem 6 $\mathbf{\Psi}_S$ *has a* $\leq$*-least fixpoint, which is obtained after finitely many iterations of* $\mathbf{\Psi}_S$*, starting with* $\langle c^{\perp}, c^{\top}\rangle$.

Definition 3 *Let S be a normal system, and let $\langle c_S, a_S \rangle$ be the $\leq$-least fixpoint of $\mathbf{\Psi}_S$. We define c_S to be the canonical chain of S, and we define the semantics of S to be completely determined by c_S. That is, for every formula F:*

$$S \models F \quad \equiv \quad c_S \models F$$

$\square$

A bound on the number of iterations needed by $\mathbf{\Psi}_S$ to reach its $\leq$-least fixpoint can be formulated in terms of the number of bridge rules in S.

Theorem 7 *Let $|S|$ denote the number of* bridge rules *of a normal system S. Then, starting with $\langle c^\perp, c^\top \rangle$, $\mathbf{\Psi}_S$ will reach its $\leq$-least fixpoint after at most $|S| + 1$ iterations.*

The semantics for normal systems defined above properly generalizes the local model semantics for positive systems.

Theorem 8 *The canonical chain of a positive system S coincides with its minimal solution chain.*

Proof. If S is positive, then for every pair $\langle c, a \rangle$, $S^+(\langle c, a \rangle)$ and $S^*(c)$ coincide, so $\mathbf{\Psi}^c_S(\langle c, a \rangle)$ does not depend on a. As a consequence, c_S is the $\preceq$-least fixpoint of $\mathbf{T}_S$ iff, for some anti-chain a_S, $\langle c_S, a_S \rangle$ is the $\leq$-least fixpoint of $\mathbf{\Psi}_S$. $\square$

The canonical chain of a system S, and other fixpoints of $\mathbf{\Psi}_S$, are intimately related to the stable solution chains of S.

Theorem 9 *Let S be a normal system, let $\langle c_S, a_S \rangle$ be the $\leq$-least fixpoint of $\mathbf{\Psi}_S$, and let c_{stable} be any stable solution chain of S. Then $c_S \preceq c_{stable} \preceq a_S$.*

Theorem 10 *Let S be a normal system and let $\langle c_S, a_S \rangle$ be the $\leq$-least fixpoint of $\mathbf{\Psi}_S$. If c_S and a_S coincide, then c_S is the unique stable solution chain of S.*

Finally, we remark that, in our view, all the examples presented above are suitably dealt with by the present analysis. We treat one of them explicitly.

Example 13 *Let S be the system from example 12. Then:*

$$c_S \quad = \quad \left\{ \begin{bmatrix} \{p, t\} \\ \{\neg p, t\} \end{bmatrix}_1 \quad \begin{bmatrix} \{q, r\} \\ \{\neg q, r\} \end{bmatrix}_2 \right\}$$

As desired, no information is derived about p and q, while t and r are indeed established in context 1 and 2, respectively.

5 Conclusions

We showed that the semantics of a multi-context system is completely determined by its least informative solution chain. We provided a way to compute this chain, and thus gave a first constructive account of the local model semantics.

We presented a generalized framework in which new information can be derived based on the absence of other information. We applied non-monotonic reasoning techniques to establish a suitable semantics for this framework.

References

[Apt *et al.*, 1988] K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. 1988.

[Benerecetti *et al.*, 2000] M. Benerecetti, P. Bouquet, and C. Ghidini. Contextual reasoning distilled. *Journal of Experimental and Theoretical Artificial Intelligence*, 12(3):279–305, 2000.

[Bidoit and Froidevaux, 1991] N. Bidoit and C. Froidevaux. General logical databases and programs: Default logic semantics and stratification. *Information and Computation*, 91:15–54, 1991.

[Fages, 1991] F. Fages. A new fixpoint semantics for general logic programs compared with the wellfounded and the stable model semantics. *New Generation Computing*, 9(4), 1991.

[Gelfond and Lifschitz, 1988] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *International Conference on Logic Programming (ICLP 88)*, pages 1070–1080, 1988.

[Ghidini and Giunchiglia, 2001] C. Ghidini and F. Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.

[Giunchiglia and Serafini, 1994] F. Giunchiglia and L. Serafini. Multilanguage hierarchical logics, or: how we can do without modal logics. *Artificial Intelligence*, 65(1):29–70, 1994.

[Giunchiglia, 1993] F. Giunchiglia. Contextual reasoning. *Epistemologia*, XVI:345–364, 1993.

[McCarthy and Buvač, 1998] J. McCarthy and S. Buvač. Formalizing context (expanded notes). In *Computing Natural Language*, volume 81 of *CSLI Lecture Notes*, pages 13–50. 1998.

[McCarthy, 1987] J. McCarthy. Generality in artificial intelligence. *Communications of ACM*, 30(12):1030–1035, 1987.

[McCarthy, 1993] J. McCarthy. Notes on formalizing context. In *International Joint Conference on Artificial Intelligence (IJCAI 93)*, pages 555–560, 1993.

[Roelofsen *et al.*, 2004] F. Roelofsen, L. Serafini, and A. Cimatti. Many hands make light work: Localized satisfiability for multi-context systems. In *European Conference on Artificial Intelligence (ECAI 04)*, pages 58–62, 2004.

[Serafini and Bouquet, 2004] L. Serafini and P. Bouquet. Comparing formal theories of context in AI. *Artificial Intelligence*, 155:41–67, 2004.

[Tarski, 1955] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[van Gelder *et al.*, 1991] A. van Gelder, K. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.