Merging requirements from a set of ranked agents

Laurence Cholvy
ONERA-Toulouse
2 avenue Ed Belin
31055 Toulouse, France
and
Anthony Hunter
Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK

September 22, 2000

Abstract

Handling inconsistency is an increasingly important issue in data and knowledge engineering. A number of logic-based proposals have been made for handling aspects of inconsistency in beliefs (where we use beliefs as a general term to encompass representations of the real world) including belief revision theory, truth-maintenance, argumentation systems, and knowledgebase merging. In contrast, there are relatively few logic-based proposals for handling inconsistency in requirements (where we use requirements as a general term to encompass representations of a required world). Whilst there is a significant overlap between the issues of concern for handling inconsistency in beliefs and in requirements, there are also some significant differences. In this paper, we present a logic-based framework for merging requirements from a set of ranked agents. In the following, we will present and motivate the framework, and then compare the framework with related logic-based proposals.

1 Introduction

Requirements can be expressed by agents (human or artificial) in the course of addressing goals of those agents. Often an agent does not operate in isolation. This may involve a set of agents each with some objectives and some autonomy who need to collaborate or co-operate in order to meet their own objectives. In this paper, we restrict consideration to some examples of human agents buying, or building, something.

The first phase of building something (eg engineering a software system), or buying something, is the requirement elicitation phase. Requirements are sentences which express some properties that the artefact, when built, or purchased, should satisfy. The requirement elicitation phase then consists in collecting the requirements from one or more agents.

Even in the simple case when the requirements are provided by only one agent, the problem of their "correctness" is raised. In [Cho98], one can find the formal definition of requirement correctness. Roughly speaking, the requirements are correct if and only if they do not specify an artefact which *cannot* be built, because of some domain constraints, and they do not specify an artefact which, even if it can be built, *will violate* some regulations.

In the case where several agents are involved in the process of expressing requirements about an artefact, the problem of requirements merging is raised. And, of course, even in this case, the problem of the correctness of the whole requirements set is also raised. In other words, even if the requirements set provided by each agent is independently correct, it may happen that collecting them leads to an incorrect set of requirements. We will also say "incompatible" requirements. Let us give an example in order to illustrate intuitively this problem.

Example 1.1 *Consider that two agents, named 1 and 2, express requirements about a house to be built. Assume that agent 1 requires that: (i) The house to be built must have white external walls; and (ii) The house must be big. Assume that agent 2 requires that: (i) The house must have external walls in brick; (ii) There must be a garden around the house; (iii) The house must be downtown; and (iv) The house must be less than 100 000 dollars.*

These requirements may be globally "incompatible", in a given context. Assume for instance, that, in the town where this house will be built, building a big house costs more than 100 000 dollars. Furthermore, assume that there is a regulation which forbids downtown houses having external walls painted white. Then according to this context, the whole set of requirements provided by the two agents are "incompatible": the house cannot be big and be less than 100 000 dollars. I.e, it is impossible to build a big house less than 100 000 dollars. Furthermore, even if it could be built, this house would violate the regulation, since downtown, external walls must not be painted in white.

This paper addresses the problem of merging requirement sets provided by several agents. In section 2, we present the way we have chosen for representing the information which are pertinent to this problem: the requirements, the domain constraints and the regulations. And we introduce the notion of compatible requirements. In section 3, we assume that the agents have the opportunity to express preferences on their own sets of requirements. This notion of preference between requirements is semantically characterized in terms of preferred sets of worlds, and also syntactically characterized in terms of a set of formulae which represents a set of compatible requirements. Section 4 addresses the problem of merging requirements from several agents. We first define the notion of compatible merged requirements. In most cases in practice, merged requirements, provided by several agents, will not be compatible. This raises the need of strategies for getting compatible requirements from some incompatible merged requirements. We present two based on the assumption that the agents can be ranked according to a total order. In section 5, we provide a comparison with related logical approaches — in particular belief revision theory, knowledgebase merging, and logical inferencing with inconsistent information. Finally in section 6, we discuss issues of applying the framework — in particular handling agents that are not totally ranked, expectations on merges in practice, and on handling contingent inconsistencies that may arise in practice.

2 Representing information for merging

When addressing the problem of merging requirements, there is the the problem of formally representing the requirements. In addition, as mentioned in the introduction, we also need to represent two other kinds of information pertinent to the problem.

The second kind of information which needs to be represented is called "domain constraints". These are sentences which express what is true in any instance of the real world. We call them "domain constraints" because of course, they depend on the domain of application. For instance, if the artefact to be built is a house, then the domain constraints are relative to the domain of architecture. For instance, they express that a level is more or less 3 meters high; the depth of the foundation depends, by a specific function, of the building height and the material it is made of. And so on. If the artefact to be designed is a plane, then the domain constraints are those of fluid mechanics, geometry, structure and stability theories.

Finally, the third kind of information which needs to be represented is called "regulations". These are sentences which express what is obligatory, permitted or prohibitted in the considered domain i.e, the domain relative to the artefact to be designed. For instance, if the artefact to be designed is a house, then several regulations must be taken into account. There is the regulation specific to the town in which the house will be built. Such a regulation for instance, defines the minimal distance between a house and the boundaries of the land, or between the house and the street. It also defines the maximal height of the houses and so on.

Below, we present the way we formalize requirements, domain constraints and regulations. In this paper, we have chosen to use classical propositional logic to represent this information. We are aware of the fact that modal logics are more appropriate if one wants to define a logic for reasoning with all these different types of information. For instance, in [Cho98], several modal operators are introduced in order to distinguish between requirements, constraints and regulations. The different axioms associated with these modal operators are then collected together in order to define a logic.

However, in the present paper, our aim is not to define a logic but only to give a semantical characterization of what are compatible merged requirements and how to get them. Any strategy which will be presented in section 4 could, of course, leads to the definition of a logic. And for doing this, using modal logics will be necessary. But we must insist on the fact that these logics are not yet defined.

2.1 Representing requirements

In this work, we consider that a requirement set, provided by an agent, is a consistent set of propositional formulae which express the properties that, according to this agent, the object should satisfy. We restrict to consistent sets since it is reasonable to assume that the agent does not require, even implicitly, a property and its negation.

Let us denote L the set of formulae of a propositional language.

Definition 2.1 *The requirement set provided by the agent x is denoted Δ_x and defined as a finite consistent subset of L.*

Example 2.1 *In the example given in introduction, there are two requirement sets:*

$$\Delta_1 = \{big\_house, white\_walls\}$$
$$\Delta_2 = \{brick\_walls, garden, downtown, less\_100000\}.$$

2.2 Representing domain constraints

Domain constraints are sentences which express what is necessarly true in the real world i.e, what is true in any instance of the real world. As said previously, in this paper, we consider that domain constraints are expressed by propositional formulae. This leads to the following definition.

Definition 2.2 *The set of domain constraints is denoted Dom and defined as a consistent subset of L.*

Example 2.2 *The domain constraint set of example 1.1 can be represented by:* $Dom = \{big\_house \rightarrow \neg less\_100000\}$

2.3 Representing regulations

Regulations are sets of sentences which express properties which are obligatory (or by duality whose negation are forbidden) or permitted in the real world. In fact, for the problem of checking the compatibility of requirements, and the compatibility of merged requirements, we only need to focus on obligations and prohibitions. Indeed, the problem is to check if the requirements do not specify an object whose some properties are forbidden (or by duality whose negation are obligatory). This is why we can consider only the parts of the regulations which define obligations (thus, prohibitions) and represent them by propositional formulae which express what is obligatory (or by duality, what is forbidden).

Definition 2.3 *A regulation is denoted Reg and defined as a consistent subset of L.*

We restrict to a consistent subset since we assume that a regulation is not self-contradictory i.e, it does not oblige both something and its contrary.

Example 2.3 *The regulation of the example 1.1, can be represented by:*

$$Reg = \{downtown \rightarrow \neg white\_walls, \}$$

Definition 2.4 (Coherence assumption) *This is the assumption that* $Dom \cup Reg$ *is consistent. That is, the regulations are consistent with the domain constraints.*

This assumption is legitimate since regulations are supposed to be compatible with the domain constraints. I.e, anything which is obligated (i.e, $\models Reg \rightarrow a$) must be possible (i.e, $\not\models Dom \rightarrow \neg a$).

The main reason we separate Dom and Reg is that the information these sets provide are of different types. Of course, here, we restrict the generality by assuming that they can both be represented in classical propositional logic. But, if we wanted to be more general, these two types of information should be represented by more complicated and different logics. For instance, modelling norms in regulations implies using a deontic logic and the kind of deontic logic depends on the type of the norms we want to model. Furthermore, for conflicts on requirements resolution, we need to be clear about the different types of information held by an agent, and so structuring the different types of information should facilitate analysis or negotiation processes that are required in the case where the agents involved do not agree on a suggested merge of the requirements. Notice that, if we do use different logics for the domain constraints and regulations then we need a framework for integrating the reasoning. There is a range of possibilities. A simple option is just assuming the inferential closure of the domain constraints and of the regulations, and then just querying the closure as a database. A more sophisticated option is a hybrid reasoning system such as based on fibred semantics [Gab99].

2.4 Compatibility of requirements

Finally, we focus on the notion of compatibility of requirements.

Definition 2.5 *Let Δ_x be a requirement set provided by an agent x. Let Dom be a domain constraint set and Reg be a regulation. Δ_x is correct with respect to Dom and Reg iff $\Delta_x \cup Dom \cup Reg$ is consistent. We will also say that the requirements in Δ_x are compatible given Dom and Reg.*

There is a close relationship between finding a compatible set of requirements and finding a set of beliefs consistent with integrity constraints. We discuss this relationship with respect to specific proposals in section 5.

3 Preferences over requirements

In this section, we extend the formalization given in 2.1, so that the user can express preferences on his requirements. This extension is motivated by the fact that, in the general case, the requirements expressed by an agent may be incompatible given the constraints and the regulations. This means that some of his requirements must be rejected. The intuitive idea is to use the preference of the user, so that the most preferred requirements are kept if possible (given the constraints and regulations).

However, we insist on the fact that, even when ordered, all the requirements are required by the user. I.e, if it was possible, given the constraints and the regulations, the user would prefer that the object being built or bought satisfies all his requirements. Preferences are a form of meta-level information that are used to help combine or merge object-level information [CH97].

3.1 Representation of preferences

The set of requirements of an agent x is then represented as a set of formulae Δ_x together with a preference over these formulae. The preference is linear and total so the requirements for an agent x together its ordering can be represented as a tuple denoted Γ_x. At each point in the ordering, only a single formula is assigned. (If more then one formula is required at the point, the conjunction of them is used).

Definition 3.1 *Let $\Delta_x = \{\alpha_1, ..., \alpha_n\}$ be the requirements for agent x. The tuple $\Gamma_x = [\alpha_1, ..., \alpha_n]$ is called the* **position** *for agent x. Furthermore, Γ_x contains exactly the same formulae as Δ_x and the ordering is such that α_i occurs to the left of α_j in the tuple iff α_i is preferred to α_j. Therefore a position for agent x implicitly defines a linear (irreflexive, antisymmetric, and transitive) ordering relation over Δ_x called the preference relation for agent x.*

Definition 3.2 *Let $\mathcal{M}$ be* **the set of classical propositional models**. *For a formula α, a delineation, denoted $[[\alpha]]$, is the set of models that satisfy α:*

$$[[\alpha]] = \{M \in \mathcal{M} \mid M \models \alpha\}$$

More generally,

$$[[\alpha_1, .., \alpha_n]] = \{M \in \mathcal{M} \mid M \models \alpha_1 \textit{ and } \cdots \textit{ and } M \models \alpha_n\}$$

Let $\mathcal{Q}$ *be* **the set of delineations** *where,*

$$\mathcal{Q} = \{[[\alpha_1, .., \alpha_n]] \mid \alpha_1, .., \alpha_n \in \mathcal{L}\}$$

Clearly, $[[\top]] = \mathcal{M}$ and $[[\bot]] = \emptyset$.

3.2 Semantical characterization of preferences

This section gives a semantical characaterization in terms of sets of models, of the position of an agent, that is, the semantical characterization of his preferences over a set of requirements. Let us first illustrate intuitively the semantics by giving examples.

Example 3.1 *Let $\Gamma_x = [\alpha, \beta]$. First, this means that the agent has expressed two requirements α and β, i.e, he expects that the object to be built will satisfy the two properties α and β. Secondly, this means that the agent prefers α to β. I.e, the agent would like the object to satisfy both α and β. But, if it is not possible (because of the constraints and the regulations) then the agent accepts that the object satifies α only, (and thus $\neg\beta$). But if it is not possible (because of the constraints and the regulations) then he accepts that the object satifies β only (and thus $\neg\alpha$). Finally, at the end, if this is not possible (because of the constraints and the regulations), then he accepts that the object satifies neither α nor β.*

This means that the position $\Gamma_x = [\alpha, \beta]$ leads to the following ordering over delineations :

$[[\alpha, \beta]] >_x [[\alpha, \neg\beta]] >_x [[\neg\alpha, \beta]] >_x [[\neg\alpha, \neg\beta]]$

(where, here, $>_x$ denotes the order relation over delineations)

Example 3.2 *Let $\Gamma_x = [\alpha, \beta, \gamma]$. This position induces the following ordering on delineations:*

$$\begin{aligned}&[[\alpha, \beta, \gamma]] >_x [[\alpha, \beta, \neg\gamma]] >_x [[\alpha, \neg\beta, \gamma]] >_x [[\alpha, \neg\beta, \neg\gamma]] \\ &\qquad >_x [[\neg\alpha, \beta, \gamma]] >_x [[\neg\alpha, \beta, \neg\gamma]] >_x [[\neg\alpha, \neg\beta, \gamma]] >_x [[\neg\alpha, \neg\beta, \neg\gamma]]\end{aligned}$$

The lexicographic ordering in both examples seems to be a reasonable interpretation of preferences over a set of requirements.

We now generalize these examples and formalize the generation of the ordering over delineations that can be obtained from a position. First, we define a function, which associates any position with a set of delineations. Then, we define a linear ordering on non-empty delineations.

Definition 3.3 *Let Compromises be a function which associates any position with a set of delineations, such that:*

$$Compromises([\alpha_1, ..\alpha_n]) = \{[[\beta_1, .., \beta_n]] \mid \beta_1 \in \{\alpha_1, \neg\alpha_1\} \text{ and } \cdots \text{ and } \beta_n \in \{\alpha_n, \neg\alpha_n\}\}$$

Example 3.3 *Continuing 3.1, for the position $[\alpha, \beta]$, we obtain:*

$$Compromises([\alpha, \beta]) = \{[[\alpha, \beta]], [[\alpha, \neg\beta]], [[\neg\alpha, \beta]], [[\neg\alpha, \neg\beta]]\}$$

Definition 3.4 *An n-place binary ordering is $(N, >)$, where N is the set of n-digit binary numbers (from 0 to 2^n), and $>$ is the usual ordering over the binary numbers. Let*

$$\mathcal{N} = \{N \mid N \text{ is the set of n-digit binary numbers and } n \text{ is a natural number}\}$$

Definition 3.5 *Let* $\Gamma_x = [\alpha_1, .., \alpha_n]$ *be a position and let* $(N, >)$ *be a binary ordering. Let* C *be a function,* $C : \mathcal{N} \longrightarrow \mathcal{Q}$*, such that for* $p \in N$ *there is* $C(p) \in \mathcal{Q}$ *where (1) if the ith digit of the binary number* p *is 1, then the ith term of the compromise is* α_i*; and (2) if the ith digit of the binary number* p *is 0, then the ith term of the compromise is* $\neg\alpha_i$*.*

Definition 3.6 *Let* $\Gamma_x = [\alpha_1, .., \alpha_n]$ *be a position with a n-place binary ordering* $(N, >)$*. Let* C *be a function defined by the previous definition. A binary relation* R_x *is defined on* $Compromises(\Gamma_x)$ *by:* $C(p) \quad R_x \quad C(q)$ *iff* $p > q$

Definition 3.7 *Let* $\Gamma_x = [\alpha_1, .., \alpha_n]$ *be a position. Let* R_x *be the binary relation previously defined. A binary relation* $>_x$ *is defined on* $Compromises(\Gamma_x) \setminus \{\emptyset\}$ *by: let* E *and* F *be two delineations in* $Compromises(\Gamma_x) \setminus \{\emptyset\}$*.* $E \;>_x\; F$ *iff* $E \quad R_x \quad F$*.*

Proposition 3.1 *Let* $\Gamma = [\alpha_1 ... \alpha_n]$ *be a position. The relation* $>_x$ *previously defined is a linear order over* $Compromises(\Gamma_x) \setminus \{\emptyset\})$*.*

Proof. We prove that all the delineations in $Compromises([\alpha_1, ..., \alpha_n] \setminus \{\emptyset\})$ are different.

Recall that a delineation is the set of models of a set of formulas of the form $\{P_1, ..., P_n\}$ where $\forall i = 1...n, P_i = \alpha_i$ or $P_i = \neg\alpha_i$. Assume that there are two delineations in $Compromises([\alpha_1, ..., \alpha_n] \setminus \{\emptyset\})$ which are identical. Let us write them:

$[[\{P_i \mid i \in I_1\}, \{P_i \mid i \in I_2\}]] = [[\{P_i \mid i \in I_1\}, \{\neg P_i \mid i \in I_2\}]]$

This implies: $\models \{P_i \mid i \in I_1\} \cup \{P_i \mid i \in I_2\} \rightarrow \{\neg P_i \mid i \in I_2\}$

Thus, $\{P_i \mid i \in I_1\} \cup \{P_i \mid i \in I_2\} \cup \vee_{i \in I_2}\{P_i\}$ is inconsistent

I.e, $\{P_i \mid i \in I_1\} \cup \{P_i \mid i \in I_2\}$ is inconsistent,

So $||\{P_i \mid i \in I_1\} \cup \{P_i \mid i \in I_2\}|| = \emptyset$.

This is impossible since this delineation is different from $\emptyset$ by assumption.

Thus, all the delineations of $Compromises([\alpha_1, ..., \alpha_n] \setminus \{\emptyset\})$ are different

(End of proof)

Example 3.4 *Let us return to Example 3.1, where* $\Gamma_x = [\alpha, \beta]$*. Where* α *and* β *are two propositional variables.*

So, $n = 2$*, and therefore* $N = \{11, 10, 01, 00\}$*. The compromises are given by* C *below:*

$$\begin{array}{l} C(11) = [[\alpha, \beta]] \\ C(10) = [[\alpha, \neg\beta]] \\ C(01) = [[\neg\alpha, \beta]] \\ C(00) = [[\neg\alpha, \neg\beta]] \end{array}$$

So, $[[\alpha, \beta]] \quad R_x \quad [[\alpha, \neg\beta]]$*,* $[[\alpha, \neg\beta]] \quad R_x \quad [[\neg\alpha, \beta]]$*,* $[[\neg\alpha, \beta]] \quad R_x \quad [[\neg\alpha, \neg\beta]]$

Finally, this gives the order: $[[\alpha, \beta]] >_x [[\alpha, \neg\beta]] >_x [[\neg\alpha, \beta]] >_x [[\neg\alpha, \neg\beta]]$

Example 3.5 *Now assume that* $\Gamma_x = [\alpha, \alpha \vee \beta]$.

We notice that $[[\alpha, \neg(\alpha \vee \beta)]] = \emptyset$. *So we get the order:* $[[\alpha]] >_x [[\neg\alpha, \beta]] >_x [[\neg\alpha, \neg\beta]]$

In the two next sections, we show that, given a set of constraints Dom and a regulation Reg, any set of totally ordered requirements Γ_x, may be associated with a set of requirements which are compatible with Dom and Reg, even if the initial requirements were not compatible.

In section 3.3, this set of compatible requirements is semantically characterized, by the delineation which is the maximal delineation, in $Compromises(\Gamma_x) \setminus \{\emptyset\}$, a model of which is also a model of $Dom \cup Reg$.

In section 3.4, this set of compatible requirements is syntactically characterized as the result of a recursive algorithm on the initial set of requirements. The equivalence of the semantical characterization and the syntactical characterization is proved.

3.3 Semantical characterization of a set of compatible requirements

In this section, we show how to use the ranking $>_x$ over delineations obtained from a position Γ_x, in order to semantically characterize a set of requirements which are compatible given the constraints and the regulations.

First, we show, by the following proposition, that, if Γ_x is a position, at most one delineation in $Compromises(\Gamma_x) \setminus \{\emptyset\}$ intersects $[[Dom \wedge Reg]]$.

Proposition 3.2 *Let Dom be a set of constraints, Reg be a regulation and* Γ_x *be a position. Then, there is a non-empty delineation D, in* $Compromises(\Gamma_x)$ *such that:* $\exists M \in D$ *and* $M \models Dom \cup Reg$.

Proof: If D_i is a delineation in $Compromises(\Gamma_x)$ we denote d_i a formula such that $[[d_i]] = D_i$.

We first prove that $d_1 \vee d_2 \vee ... \vee d_{2^n}$ is equivalent to true. This is obvious by definition of $Compromises(\Gamma_x)$.

Now, assume that $\forall D_i \in Compromises(\Gamma_x), \forall M \in D_i, M \not\models Dom \cup Reg$. This means that $\forall D_i \in Compromises(\Gamma_x), d_i \cup Dom \cup Reg$ is inconsistent. This implies that $\forall D_i \in Compromises(\Gamma_x), \models Dom \cup Reg \rightarrow \neg d_i$. Thus, $\models Dom \cup Reg \rightarrow \neg(d_1 \vee ... \vee d_{2^n})$. This implies that $Dom \cup Reg$ is inconsistent, which is impossible by the Coherence assumption.

(End of proof)

So, the set $\{D : D \in Compromises(\Gamma_x) \;\; and \;\; \exists M \in D \;\; such\ that\ M \models Dom \cup Reg\}$ is not empty. So, the maximal[1] element, of this set and for the relation $>_x$, exists. It is unique since $>_x$ is linear on $Compromises(\Gamma_x) \setminus \{\emptyset\}$. Thus, the following definition has a meaning.

Notation. Let us denote γ_x a set of formulae such that

$$[[\gamma_x]] = Max_{>_x}\{D : D \in Compromises(\Gamma_x) \;\; and \;\; \exists M \in D \; such\ that\ M \models Dom \cup Reg\}$$

[1] If E is a set and $>$ an order relation on E, then the maximal elements for $>$ in E are $\{e : e \in E \;\; and \;\; (\forall f \in E,$ if $f > e$ then $f = e)\}$

Proposition 3.3 *Let Dom be a set of constraints and Reg be a regulation. Let Γ_x be the position of agent x and γ_x defined as previously. Then γ_x is a set of requirements, compatible with Dom and Reg.*

Proof: γ_x is a compatible with Dom and Reg iff $\gamma_x \cup Dom \cup Reg$ is consistent. That is, iff there is a model M in $[[\gamma_x]]$ which is also a model of $Dom \cup Reg$. But, this is ensured by definition of $[[\gamma_x]]$. So, γ_x is a set requirements compatible with Dom and Reg.

(End of proof)

This proposition shows that, given Dom and Reg, a set of ordered requirements Γ_x, possibly incompatible, can be associated with a set of requirements γ_x which are compatible with Dom and Reg and such that $[[\gamma_x]]$ is the most preferred, in terms of the relation $>_x$, of the delineations of $Compromises(\Gamma_x)$ a model of which is also a model of $Dom \cup Reg$.

Proposition 3.4 *If Γ_x are compatible requirements then γ_x is equivalent to Γ_x.*

Proof: If Γ_x is a set of compatible requirements, this implies that $\Gamma_x \cup Dom \cup Reg$ is consistent. Thus, there exists a model in $[[\Gamma_x]]$ which is also a model of $Dom \cup Reg$. So, because $[[\Gamma_x]]$ is the maximal element in $Compromises(\Gamma_x) \setminus \{\emptyset\}$ for $>_x$, , it is then the maximal element of $\{D : D \in Compromises(\Gamma_x) \;\; and \;\; \exists M \in D \;\; such\ that \;\; M \models Dom \cup Reg\}$. So $[[\Gamma_x]] = [[\gamma_x]]$. This proves that Γ_x and γ_x are equivalent.

(End of proof)

Example 3.6 *Let us again consider $\Gamma_x = [\alpha, \beta]$. Assume that $Dom = \{\neg(\alpha \wedge \beta)\}$ and $Reg = \emptyset$. One can notice that the requirements in Γ_x are not compatible given Dom and Reg. Then, according to the previous definition, one can define a set of compatible requirements, γ_x such that $[[\gamma_x]] = [[\alpha, \neg\beta]]$.*

Assume now that $Dom = \{\neg\alpha\}$ and $Reg = \{\neg\beta\}$. Then according to the definition, one can define a set of compatible requirements, γ_x such that $[[\gamma_x]] = [[\neg\alpha, \neg\beta]]$.

3.4 Syntactical characterization of a set of compatible requirements

Definition 3.8 *Let $\Gamma_x = [\alpha_1, ..., \alpha_n]$ be the position of agent x. We define n functions, $f_1, ..., f_n$, which associate a set of formulae E such that $E \cup Dom \cup Reg$ is consistent, with the sets $f_1(E), ..., f_n(E)$ as follows:*

- If $E \cup \{\alpha_i\} \cup Dom \cup Reg$ is consistent then, $f_i(E) = E \cup \{\alpha_i\}$
- Else, (in such a case $E \cup \{\neg\alpha_i\} \cup Dom \cup Reg$ is consistent), $f_i(E) = E \cup \{\neg\alpha_i\}$

Notice that, for any α_i and any set E such that $E \cup Dom \cup Reg$ which is consistent, $E \cup \{\alpha_i\} \cup Dom \cup Reg$ and $E \cup \{\neg\alpha_i\} \cup Dom \cup Reg$ cannot be both inconsistent. Indeed, if it was the case, we could infer that $E \cup Dom \cup Reg$ is inconsistent. This justifies why there are only two cases in the previous definition.

Proposition 3.5 *Let* $\Gamma_x = [\alpha_1, ..., \alpha_n]$ *be the position of an agent* x*. Then* $f_n(....f_2(f_1(\emptyset)))$ *is a set of requirements which are compatible given Dom and Reg.*

Proof: This proposition is proved by induction on n. The compatibility is obvious, by definition of the functions f_i's.

This proposition means that from a set of requirements expressed by an agent, and from his preference over this set, taking the constraints and the regulations into account, one can characterize syntactically, by the previous functions, a set of requirements which are compatible with the constraints and the regulation and which optimizes the user's preference.

The following proposition shows the equivalence between the semantical characterization of a set of compatible requirements, obtained from a position, a set of constraints and a regulation, given in section 3.3 and the syntactical characterization given here.

Proposition 3.6 *Let Dom be a set of constraints, Reg be a regulation and* $\Gamma_x = [\alpha_1, ..., \alpha_n]$ *be the position of agent* x*. Then,* $[[f_n(...f_2(f_1(\emptyset))..)]] = [[\gamma_x]]$.

Proof:

- We first prove that $[[f_n(...f_1(\emptyset)..)]]$ belongs to $Compromises(\Gamma_x)$ by proving that $\forall i, \alpha_i \in f_n(...f_1(\emptyset)..)$ or $\neg\alpha_i \in f_n(...f_1(\emptyset)..)$.
- We then prove that $\exists M \in [[f_n(...f_1(\emptyset)..)]]$ such that $M \models Dom \cup Reg$. This is obvious, because, by definition $f_n(...f_1(\emptyset)) \cup Dom \cup Reg$ is consistent.
- Finally, we prove that $[[f_n(...f_1(\emptyset))]]$ is maximal for $>_x$.
  Assume the existence of a delineation, denoted $[[D_0]]$ (where D_0 is a set of formulae) , such that:

  (a) $[[D_0]] \in \{D : D \in Compromises(\Gamma_x) \text{and} \exists M \in D, M \models Dom \cup Reg\}$, and

  (b) $[[D_0]] >_x [[f_n(...f_1(\emptyset)...)]]$.

  If $[[D_0]] >_x [[f_n(...f_1(\emptyset)...)]]$, then $\exists i, \neg\alpha_i \in f_n(...f_1(\emptyset)...)$ and $\vdash D_0 \rightarrow \alpha_i$ and $\forall j < i, \alpha_j \in f_n(...f_1(\emptyset)...)$ iff $\vdash D_0 \rightarrow \alpha_j$.

  Let us denote $fi(...f1(\emptyset)...) = E_{i-1} \cup \{\neg\alpha_i\}$. So by definition of the $fi's$, $E_{i-1} \cup \{\neg\alpha_i\} \cup Dom \cup Reg$ is consistent.

  Thus, $E_{i-1} \cup \{\alpha_i\} \cup Dom \cup Reg$ is inconsistent.

  So, $\forall M, \text{if } M \models E_{i-1} \cup \{\alpha_i\} \text{then} M \not\models Dom \cup Reg$.

  Thus, $\forall M \in [[D_0]], M \not\models Dom \cup Reg$. This contradicts the assumption (a).

  So, such a delineation $[[D_0]]$ does not exist.

(End of proof)

Example 3.7 *Let us consider* $Dom = \{downtown \rightarrow \neg less\_100000\}$ *and* $Reg = \{downtown \rightarrow \neg white\_walls\}$*. Assume that an agent has the following position:*

$$\Gamma = [downtown, less\_100000, white\_walls, garden].$$

This means that, by order of preference, the agent requires his house to be downtown, to be less than 100000 dollars, to have white walls and a garden. According to the previous definitions, we get the following set of requirements:

$$\gamma = \{downtown, \neg less\_100000, \neg white\_walls, garden\}$$

which are compatible given Dom and Reg.

Indeed, since it is impossible to build a house downtown for less than 100000 dollars, and because the agent wants his house to be downtown more than to be less than 100000 dollars, we get a requirement set which specifies a house downtown but which will be more than 100000 dollars. Furthermore since downtown it is forbidden to paint walls in white and since the agent wants the house to be downtown more than to have white walls, then we get a requirement set which specifies a house downtown the walls of which will not be painted in white. Finally, nothing prevents the user nor forbids him to have a garden, so we get a requirement set which requires a garden as initially required.

Example 3.8 *Let us consider another example with disjunctive requirements. Assume that:*

$\Gamma = \{downtown \vee nearby\_station, white\_walls, quiet]$

$Dom = \{nearby\_station \rightarrow \neg quiet\}$ *and*

$Reg = \{downtown \rightarrow \neg white\_walls\}$

The γ *is equivalent to:*

$$\{downtown \vee nearby\_station, white\_walls, \neg quiet\}$$

Thus, by taking constraints and regulations into account, we can say that any house which satisfies $\{nearby\_station, white\_walls, \neg quiet\}$ *will fulfill the user's requirements at the best.*

This also means that any house situated near a station with white_walls will unfortunately not be quiet (due to the constraint) but will satisfies the most preferred requirement (it is downtown or near a station) and also fulfill the second most preferred requirement: it has white walls.

One can wonder why a quiet house, situated downtown (thus with non white walls) does not suit more the user's requirements ? In fact such a house also satisfies the most preferred requirement (it is downtown or near a station) but it does not fulfill the second most preferred requirement but only the third one. This is why such a solution is not provided.

4 Merging requirements

In this section we generalize the notion of compatibility of requirements to merged requirements, and then consider strategies for merging. We consider two strategies M1 and M2 in detail.

4.1 Compatibility of merged requirements

The definition of compatible requirements introduced in section 2.5 is here extended to the case of merged requirements.

Definition 4.1 *Let Δ_1 ... Δ_n be requirement sets provided by some agents named 1 ... n. Let Dom a domain constraint set and Reg a regulation. The merged requirement set is correct in regard with Dom and Reg iff $\Delta_1 \cup ... \cup \Delta_n$ is correct in with respect to Dom and Reg i.e, iff $\Delta_1 \cup ... \cup \Delta_n \cup Dom \cup Reg$ is consistent. We will also say that the merged requirements are compatible given Dom and Reg.*

Example 4.1 *Let us come back to example 1.1 again. Obviously, the requirements of $\Delta_1 \cup \Delta_2$ are not compatible given Dom and Reg. Indeed, the set as represented below is not consistent.*

$$\begin{array}{cc} big\_house, & white\_walls, \\ brick\_walls, & garden, \\ downtown, & less\_100000, \\ big\_house \rightarrow \neg less\_100000, & downtown \rightarrow \neg white\_walls \end{array}$$

The requirement elicitation phase aims to collect requirements which are compatible with the constraints and the regulations. Indeed, as seen previously, if the requirements are not compatible with the constraints and the regulations, this means that it will be impossible to build the object that is being specified or if it can be built, it will violate some regulation. So, if the requirement elicitation phase leads to requirements which are not compatible (with the constraints and the regulations), one has to identify some subset of them which are compatible. In the following subsections, we address this problem.

4.2 Generalities about strategies for merging requirements

Several strategies can be defined for using the preferences the agents have expressed on their own requirements, in order to characterize requirements which are compatible with the constraints and the regulations. In this paper we focus on strategies based on a ranking over agents. This is motivated by the idea that, even if, from the preferences an agent expresses on his own requirements can be used to build a compatible set of requirements (by the process described previously) the resulting sets of requirements can be incompatible. So, the individual preferences are not sufficient to define a global set of compatible requirements.

We claim that other "meta" information can be used to solve this problem: this is an order between the agents. This ranking among the agents can allow someone to express a kind of importance attributed to the different agents. Let us consider an example in order to illustrate this idea.

Example 4.2 *Consider a commercial research center in the process of choosing a new computer. Since the centre depends on intellectual property with copyrights and patents, some regulations oblige the centre to chose a computer system that ensures the confidentiality and integrity of classified data. Besides that, the budget for buying the computer is limited. Before buying the computer system, the manager asks some people for their own requirements regarding the new computer. These people are: the computer service manager, one secretary and one researcher. These last two people will be users of the new computer, while the first one will have to install it and maintain it. Since the way these agents will use the machine are different and since their particular interests are different too, the whole set of requirements may be incompatible: the secretaries, who do not know how machines and software are built, may require things that cannot be developed. The researcher may want an efficient and up-to-date machine that the computer service manager does not want to install and maintain or which is too expensive. Conversely, the computer service manager may require a machine of the kind he knows the best but which does not suit the researcher or/and to the secretary or which does not ensure the confidentiality of data.*

We claim that, in such a situation, if a negotiation is not envisaged, the centre manager will make choices by expressing priorities among the three agents. For instance, he can consider that, since the centre is a research centre, the most important requirements about the new machine are the ones expressed by the researcher. The other most important requirements are the ones expressed by the secretary since one of his job is to help the researcher by writing reports, sending papers... Finally, the least important requirements are the ones of the computer service manager.

Of course, the centre manager may express a different priority among the requirements. He can consider that the one who knows the things the best is the computer service manager, so he can give priority to his requirements. Then he can decide that the secretary's requirements must be fulfilled if possible, because the union of the secretaries in the centre is very strong and he does not want a strike. Finally, requirements of the researcher are considered as the least important because, anyway, researchers can use any kind of machine.

In the next two subsections, we consider two strategies. Unfortunately, expressing a ranking over agents (or a total ranking) is not always possible. In such a case, we have to find other kind strategies. We discuss this problem in [CH99].

4.3 Merging strategy M1

Definition 4.2 (Merging strategy M1) *Let us assume n agents, denoted 1...n, who express requirements and preferences on them. Let us denote $\Gamma_1 ... \Gamma_n$ their associated positions. Let us denote $\gamma_1 ... \gamma_n$ the sets of requirements, associated with $\Gamma_1 ... \Gamma_n$, compatible with the constraints and the regulations, which have been syntactically characterized in section 3.4. Let us assume a total order, denoted $\succ$, on the agents. For simplicity, but without loosing generality, we consider that the ranking is: $1 \succ ... \succ n$. A new position*[2]*, denoted Γ is defined by: $\Gamma = [\gamma_1, ..., \gamma_n]$.*

Definition 4.3 *The set of requirements obtained from merging $\Gamma_1 ... \Gamma_n$ and assuming the ranking $1 \succ ... \succ n$ is defined by γ associated with Γ and syntactically characterized in section 3.4.*

This defines the merged requirements by a two-steps method:

- With each position, Γ_i, we associate (by the syntactical charcaterization described in section 3.4) a set of requirements, compatible with the constraints and the regulations: γ_i.
- Then we define, from these sets and the ranking over the agents $1 \succ ... \succ n$ a new position: $\Gamma = [\gamma_1, ..., \gamma_n]$. Finally, with position Γ, we associate, through the same process, the set γ of requirements which are proved to be, by definition, compatible with the constraints and the regulations.

Example 4.3 *Let us take again the example given in the introduction and assume that the preferences of each agent are the following:*

$$\Gamma_1 = [big\_house, white\_walls]$$
$$\Gamma_2 = [less\_100000, downtown, garden, brick\_walls].$$

Remember that $Dom = \{big\_house \rightarrow \neg less\_100000\}$ and that $Reg = \{downtown \rightarrow \neg white\_walls\}$.

[2] We should index Γ by the order $1 \succ ... \succ n$ and by the sets $\gamma_1 ... \gamma_n$ but we omit this index for clarity

Assume that the agents are ranked as $1 \succ 2$. Then strategy M1 leads to the following set of requirements: $\gamma = \{big\_house, white\_walls, \neg(less\_100000 \wedge downtown \wedge garden \wedge brick\_walls)\}$.

Assume now that the agents are ranked as $2 \succ 1$. Then strategy M1 leads to the following set of requirements: $\gamma = \{less\_100000, downtown, garden, brick\_walls, \neg(big\_house \wedge white\_walls)\}$.

4.4 Merging strategy M2

Definition 4.4 (Merging strategy M2) *Let us consider again n positions $\Gamma_1 ... \Gamma_n$, such that Γ_i is $[\alpha_i^1 ... \alpha_i^{k_i}]$, for $i \in \{1, ..., n\}$. Let us assume a ranking on the agents. Again, for simplicity, but without loosing generality, we consider the ranking: : $1 \succ ... \succ n$. A new position*[3]*, denoted Γ is defined by : $\Gamma = [\alpha_1^1, ..., \alpha_1^{k_1}, ..., \alpha_n^1, ..., \alpha_n^{k_n}]$.*

The results of section 3 ensure that this position Γ is associated with a set of requirements, compatible with the constraints and the regulations. This set, denoted γ, is syntactically characterized by the process of section 3.4.

Definition 4.5 *As previously, the set of requirements obtained from merging $\Gamma_1 ... \Gamma_n$ and assuming the ranking $i_1 \succ ... \succ i_n$ is defined by γ associated with Γ.*

Example 4.4 *Let us consider again the previous example. If the ranking over the agents is: $1 \succ 2$, then strategy M2 leads to the following set of requirements: $\{big\_house, white\_walls, \neg less\_10000, \neg downtown, garden, brick\_walls\}$. If the ranking is $2 \succ 1$, then it leads to $\{less\_100000, downtown, garden, brick\_walls, \neg big\_house, \neg white\_walls\}$*

Definition 4.6 *Let $\Gamma = [\alpha_1, ..., \alpha_k]$ and $\Gamma' = [\alpha\prime_1, ..., \alpha\prime_k]$ be two positions. We say that they are equivalent (denoted $\Gamma \sim \Gamma'$) iff $\forall i \in \{1...k\} \models \alpha_i \leftrightarrow \alpha\prime_i$.*

Proposition 4.1 *Let $\Gamma_1 ... \Gamma_n$ be n positions and $\Gamma'_1 ... \Gamma'_n$ be n other positions. Let $\gamma_{1 \succ ... \succ n}$ (resp, $\gamma\prime_{1 \succ ... \succ n}$) denote the result by strategy M2 of merging the requirements $\Gamma_1 ... \Gamma_n$ (resp, $\Gamma'_1 ... \Gamma'_n$) when the ranking on the agents is $1 \succ ... \succ n$. If $\forall i \in \{1...n\}, \Gamma_i \sim \Gamma'_i$, then $\models \gamma_{1 \succ ... \succ n} \leftrightarrow \gamma\prime_{1 \succ ... \succ n}$*

Proof: Let us write: $\Gamma_i = [\alpha_i^1, ..., \alpha_i^{k_i}]$ and $\Gamma'_i = [\alpha_i'^1, ..., \alpha_i'^{k_i}]$ for $i \in \{1, ..., n\}$.

By proposition 3.6, $\gamma_{1 \succ ... \succ n}$ is equivalent to $f_n^{k_n}(...f_1^1(\emptyset)..)$ where

$f_i^j(E) = E \cup \{\alpha_i^j\}$ if $E \cup \{\alpha_i^j\} \cup Dom \cup Reg$ is consistent and $f_i^j(E) = E \cup \{\neg \alpha_i^j\}$ else.

and $\gamma'_{1 \succ ... \succ n}$ is equivalent to $f_n'^{k_n}(...f_1'^1(\emptyset)..)$ where

$f_i'^j(E) = E \cup \{\alpha_i'^j\}$ if $E \cup \{\alpha_i'^j\} \cup Dom \cup Reg$ is consistent and $f_i'^j(E) = E \cup \{\neg \alpha_i'^j\}$ else.

Since $\forall i \in \{1...n\} \forall j \in \{1...k_i\} \models \alpha_i^j \leftrightarrow \alpha_i'^j$, we have $\models f_n^{k_n}(...f_1^1(\emptyset)..) \leftrightarrow f_n'^{k_n}(...f_1'^1(\emptyset)..)$

So $\models \gamma_{1 \succ ... \succ n} \leftrightarrow \gamma'_{1 \succ ... \succ n}$

(End of proof)

[3] Again, we should index Γ by the order $1 \succ ... \succ n$ and by the sets $\Gamma_1 ... \Gamma_n$ but we omit this index for clarity

This proposition ensures that the strategy M2 is syntax-independent. I.e, if the agents express different but equivalent requirements ordered by the same preference, then we obtain an equivalent set of merged requirements.

Proposition 4.2 *If* $\models \gamma_1 \rightarrow \beta$ *then* $\models \gamma_{1 \succ \ldots \succ n} \rightarrow \beta$.

Proof: We have : $\gamma_{1\succ\ldots\succ n} \leftrightarrow f_n^{k_n}(\ldots f_n^1(\ldots.(f_1^{k_1}\ldots f_1^1(\emptyset)\ldots)..)..)$.

By definition $(f_1^{k_1}\ldots f_1^1(\emptyset)\ldots)$ is included in $f_n^{k_n}(\ldots f_n^1(\ldots.(f_1^{k_1}\ldots f_1^1(\emptyset)\ldots)..)..)$.

Thus any formula implied by $(f_1^{k_1}\ldots f_1^1(\emptyset)\ldots)$ is also implied by $f_n^{k_n}(\ldots f_n^1(\ldots.(f_1^{k_1}\ldots f_1^1(\emptyset)\ldots)..)..)$.

So, any formula implied by γ_1 is also implied by $\gamma_{1\succ\ldots\succ n}$.

(End of proof)

This proposition ensures that the set of requirements compatible with the constraints and the regulations which are characterized by the position of the agent who is considered as the most important, belongs to the merged requirements.

Note, that it can happen that some formula β, implied by γ_2 and consistent with $\gamma_1 \cup Dom \cup Reg$ is not implied by $\gamma_{1\succ 2}$, although it seems to be the case. Here is a counter-example:

Example 4.5 $\Gamma_1 = [a]$, $\Gamma_2 = [b, c]$. $Dom = \{b \rightarrow \neg c\}$ *and* $Reg = \{\neg(a \wedge b)\}$.

$\gamma_1 = \{a\}, \gamma_2 = \{b, \neg c\}$ *and* $\gamma_{1\succ 2} = \{a, \neg b, c\}$.

γ_2 *implies* $\neg c$, $\gamma_1 \cup Dom \cup Reg \cup \{\neg c\}$ *is consistent but* $\gamma_{1\succ 2}$ *does not imply* $\neg c$.

Indeed, $\neg c$ *was in* γ_2 *because* c *was required by agent* 2*, but* b *was also required by agent* 2 *and most preferred. Because of the constraint,* c *was not kept as a requirement, but* $\neg c$ *was. When merging the requirements from the two agents, assuming that* $1 \succ 2$*, because the most preferred requirement is now* a*, requirement* b *is rejected and requirement* c *now can be kept. So,* $\neg c$ *is not kept.*

5 Comparison with related logical approaches

In this section, we compare our approach with belief revision theory, knowledgebase merging techniques, and logical inferencing with inconsistent information.

5.1 General observations

Let us make some general observations to compare and contrast the essential characteristics of beliefs and of requirements.

Beliefs When finding a set of beliefs consistent with a set of integrity constraints, for example in database updating, or in a diagnostic decision-support system, there is uncertainty about whether the beliefs hold. Deciding on which beliefs to adopt is about constructing a model of the real-world. Some or all of the beliefs might actually be unjustified, and so these beliefs might actually turn out to be false when checked against the real world. In finding a set of beliefs consistent with a set of integrity constraints, the aim is to maintain as many of the beliefs as possible, and only allow a belief to be dropped when it violates the integrity constraints. When a belief is dropped, the negation of the belief does not necessarily have to be adopted.

Requirements When finding a compatible set of requirements, there is no uncertainty about whether the requirements hold since the choice of requirements is still in the hands of the agents. Nothing has been committed yet. Deciding on which requirements to adopt is about constructing a model of an idealized world, and so it is not a matter of a requirements being true or false in the real-world. In finding a compatible set of requirements, the aim is to maintain as many of the requirements as possible, and only allow a requirement to be dropped when it is inconsistent with the domain and regulation set. However, when a requirement is dropped, the negation of the requirement is necessarily adopted.

Adopting the negation of a desired requirement in case of conflict differentiates our approach from the proposals on database updating, knowledgebase merging, and reasoning with maximally consistent subsets of inconsistent data. Nonetheless, there are clearly significant similarities between handling beliefs and handling requirements. Therefore, in the following subsections we compare our framework with some key proposals for handling beliefs.

5.2 Belief revision theory

The previous general observations led us to conclude that reasoning with incompatible requirements and reasoning with inconsistent beliefs are not exactly the same problem. In the same way, finding a set of compatible requirements and finding a set of consistent beliefs are not exactly the same problem. However, they do have some similarities and it is pertinent to compare our approach to finding correct requirements with that of belief revision theory [AGM85, Gar88].

The way we proceed for this comparision is the following: we first recall the set of postulates which have been accepted by the community to characterize belief revision operators. We then show that it is not possible to fit our approach into this framework. So, we turn to characterize the requirement merging process, not as a single function, but as the iterative application of one operator. We show that this operator satisfies the postulates for requirement revision which are an adaptation of the belief revision postulates for our problem.

Definition 5.1 *If K is a belief state and ϕ is a formula, then $K * \phi$ is a belief state, the result of revising K with ϕ. We assume here classical logic with the usual definition for models (the entailment relation). Note, $*$ is a belief revision function, L is the set of all sentences in the language and $K + \phi = Cn(K \cup \{\phi\})$.*

$K1$ $K * \phi$ *is a deductively closed theory*
$K2$ $\phi \in K * \phi$
$K3$ $K * \phi \subseteq K + \phi$
$K4$ *If* $\neg\phi \notin K$ *then* $K = \phi \subseteq K * \phi$
$K5$ $K * \phi = L$ *imlies* $\phi = \bot$
$K6$ *If* $\models \phi \leftrightarrow \psi$ *then* $K * \phi = K * \psi$
$K7$ $K * (\phi \wedge \psi) \subseteq K * \phi + \psi$
$K8$ *If* $\neg\psi \notin K * \phi$ *then* $K * \phi + \psi \subseteq K * (\phi \wedge \psi)$

We would like to view our framework as fitting into the framework of belief revision theory. However, we have the notion of a position and so require some form of iterated belief revision where the consistency $Dom \cup Reg$ takes priority over all the formulae in the position, and furthermore where $Dom \cup Reg$ is not an inference unless it is also expressed as a requirement. To address this, we require an alternative set of postulates, and so we propose the following requirements revision postulates. For requirements revision, we suggest the need for an iterative revision that is qualified by consistency with a formula X. The motivation for each of $R1, ...R8$ corresponds to the motivation for each of $K1, .., K8$ respectively. In addition, we add $R9$.

Definition 5.2 *If K, X, and ϕ are formulas, then $K *_X \phi$ is a revised formula, the result of revising K with ϕ, with respect to X. R1 to R9 are requirements revision postulates:*

$R1$ $\quad K *_X \phi$ *is a formula*
$R2$ $\quad \phi \in K *_X \phi$
$R3$ $\quad$ *If* $K \wedge \phi \wedge X$ *is consistent then* $K *_X \phi \subseteq K + \phi$
$R4$ $\quad$ *If* $K \wedge \phi \wedge X$ *is consistent, then* $K + \phi \subseteq K *_X \phi$
$R5$ $\quad$ *If* $K *_X \phi \wedge X$ *is inconsistent then* $\phi \wedge X$ *is inconsistent*
$R6$ $\quad$ *If* $\models \phi \leftrightarrow \psi$ *then* $K *_X \phi = K *_X \psi$
$R7$ $\quad$ *If* $K \wedge \phi \wedge \psi \wedge X$ *is consistent, then* $K *_X (\phi \wedge \psi) \subseteq (K *_X \phi) + \psi$
$R8$ $\quad$ *If* $(K *_X \phi) \wedge \psi \wedge X$ *is consistent, then* $(K *_X \phi) + \psi \subseteq K *_X (\phi \wedge \psi)$
$R9$ $\quad (K \in K *_X \phi)$ *or* $(\neg K \in K *_X \phi)$

We now consider the following particular function.

Definition 5.3 *The operator $*_{DR} : L \times L \rightarrow L$ is defined as follows, where DR is the conjunction of formulas of $Dom \cup Reg$:*

$\psi *_{DR} \phi = \phi \wedge \psi \qquad$ *iff* $\phi \wedge \psi \wedge DR$ *is consistent*

$\psi *_{DR} \phi = \phi \wedge \neg\psi \qquad$ *else.*

Proposition 5.1 *$*_{DR}$ satisfies R1...R9*

So, $*_{DR}$, is a requirement revision function.

Proposition 5.2 *If $\Gamma = [\alpha_1 ... \alpha_n]$ is a position, Dom is a set of constraints, Reg be a regulation, and γ is the set of requirements, obtained from Γ and compatible with Dom and Reg, then the following holds:*

$$\gamma = (\alpha_n *_{DR} (....(\alpha_2 *_{DR} (\alpha_1 *_{DR} True))...))$$

Proof: Let $f_1, ..., f_n$ be function as defined as in Defintion 3.7. For f_1, $f_1(\emptyset) = \alpha_1 * \top$, and for each $i > 1$, we have

$$f_i(...(f_1(\emptyset))) = \alpha_i * (....(\alpha_1 * \top))$$

So by Proposition 3.5, we obtain from f_n the result that γ is the set of requirements obtained from Γ and compatible with $Dom \cup Reg$.

(End of proof)

So, the process of defining a set of requirements compatible with some constraints and regulations, can be seen as an iterated process of requirements revision.

There have been many other developments of belief revision theory including iterated belief revision [DP97, Leh95], and relating belief revision to database updating [KM92]. These also offer intuitive abstract constraints for revision/updating. For a review of belief revision theory see [DP98].

There are some more concrete proposals for knowledgebase merging that adhere to belief revision postulates. In Konieczny and Pino Perez [KP98], there is a proposal for merging beliefs based on semantically characterizing interpretations which are "closest" to some sets of interpretations. But the approach does not exploit any meta-level information such as preferences. The approach has been generalized by considering merging with respect to integrity constraints [KP99].

Another approach that extends belief revision theory, called arbitration operators, is by Liberatore and Schaerf [LS98]. This form of merging is restricted to merging only two knowledgebases and it forces the result to be the disjunction of the two original knowledgebases. Also it does not use any meta-level information such as preferences.

More concrete proposals for belief revision that do incorporate priorities include ordered theory presentations [Rya92] and prioritized revision [GR96, GR97]. These approaches aim to keep as much of the original beliefs by adopting a more sophisticated view of refining beliefs. In ordered theory presentations, if a formula is less preferred than another which contradicts it, those aspects of it which are not contradicted are preserved. This is done by adopting an inferentially weaker formula to avoid the contradiction with the more preferred formula. This merging can be undertaken in an arbitrarily large partially ordering of formulae. In prioritized revision, a belief revision operator is defined in terms of selecting the model that satisfies the new belief and is nearest to the existing beliefs. The measure of nearness can be used in iterated belief revision where the more preferred items are used in later revisions.

We can see the difference between these approaches and our approach by considering the very simple example of a knowledgebase containing two items $\{\neg\psi, \phi \wedge \psi\}$, where the first item is preferred to the second. In our approach, we obtain $\{\neg\psi\}$ as the merged set of requirements, whereas in the other approaches we obtain $\{\neg\psi, \phi\}$. In this way, we see that our approach is more concerned with identifying requirements being accepted or negated in the merged form. In contrast, these alternatives are concerned with using the priorities to preserve as much of the beliefs as is permitted without violating consistency.

5.3 Knowledgebase merging techniques

In addition to the techniques of Konieczny and Pino Perez [KP98, KP99] that we discussed in the previous subsection, we need to consider knowledgebase merging, by Baral et al [BKMS92] which offers two approaches to combining prioritized theories. These two approaches assume a linear ordering over a set of sets of formulae. The first technique is top-down merging and the second is bottom-up merging.

In the top-down version of merging, the most preferred set Δ_1 is taken and combined with the next most preferred set Δ_2 by taking the maximal subset of Δ_2 that is consistent with Δ_1. This technique is then iterated by taking the result from the previous cycle the most preferred set, and Δ_3 is the next most preferred set. This process is repeated until all sets have been considered.

In the bottom-up version of combination, the least preferred set Δ_n is taken and combined with the next least preferred set Δ_{n-1} by taking the maximal subset of Δ_n that is consistent with Δ_{n-1}. This technique is then iterated by taking the result from the previous cycle as regarded as the least preferred set, and Δ_{n-2} is the next least preferred set. This process is repeated until all sets have been considered.

Example 5.1 *Consider the following sets where Δ_0 is a set of integrity constraints, Δ_1 is preferred to Δ_2 and Δ_0 is preferred to both:*

$$\Delta_0 = \{c \to \neg a, \neg c \to \neg b\}$$
$$\Delta_1 = \{a, b\}$$
$$\Delta_2 = \{c\}$$

The top-down merging, and the bottom-up merging, gives either $\{a\}$ or $\{b, c\}$, whereas $M2$ gives $\{(\neg a \vee \neg b), c\}$, assuming that Δ_0 is treated as Dom $\cup$ Reg.

So whilst this approach gives the same result as our approach in many cases if we equate integrity constraints with the union of the domain and regulation sets, there is a significant difference between their approach and ours. As discussed in Section 2.4 in our approach, if a requirement is withdrawn, then the negation of the requirement is included in the merged knowledgebase. This is not the case with withdrawing beliefs in [BKMS92]. In addition, in our approach we generate a merged position in the merging process, and so information about priorities is maintained, whereas in top-down and bottom-up merging, the priorities are lost.

5.4 Logical inferencing with inconsistent information

So far we have focussed on merging or revising a knowledgebase by rejecting items that cause inconsistency. A closely related approach is to keep all formulae, but to reason with maximally consistent subsets of the knowledgebase (see for example [MR70, BDP93, EGH95]). One of the problems with reasoning with maximally consistent subsets is that there may be many of them. This can be ameliorated by assuming meta-level information such as priorities over formulae. Here we consider some proposals based on this idea.

In [Bre89], a knowledgebase Δ is a tuple $(\Delta_1, ..., \Delta_n)$ where each Δ_i is a set of classical formulae. Information in Δ_i is preferred to (or more certain than) information in Δ_j if $i < j$. Given a knowledgebase Δ, a preferred subtheory $\Phi \subseteq \Delta$ is obtained using the following definition:

$$\begin{aligned}&\Phi = \Phi_1 \cup ... \cup \Phi_n \text{ is a preferred subtheory of } \Delta \\ &\qquad \text{iff } \forall k(1 \leq k \leq n)\ \Phi_1 \cup ... \cup \Phi_k \text{ is a maximal consistent subset of } \Delta_1 \cup ... \cup \Delta_k\end{aligned}$$

So to obtain a preferred subtheory of Δ, we have to start with any maximal consistent subset of Δ_1, add as many formulae from Δ_2 as consistently can be added and continue the process with Δ_3,...,Δ_n. Reasoning is then done using classical logic with the preferred subtheory. We can regard a preferred subtheory as a merged knowlegebase. Indeed without integrity constraints it is the same method as the top-down merging process of [BKMS92].

In [BDP95], another approach to reasoning from an inconsistent knowledgebase is proposed, where each knowledgebase Δ is partitioned into a sequence of disjoint subsets Δ_1,...,Δ_n, and Δ_i is more certain than Δ_j if $i < j$ holds. From these subsets, the following sets can be formed for i $(1 \leq i \leq n)$ $\Delta^i = \Delta_1 \cup ... \cup \Delta_i$. Assuming Δ be a knowledgebase, we can also form the following sets,

$$MaxCon(\Delta) = \{\Gamma \mid \Gamma \text{ is a maximally consistent subset of } \Delta\}$$

$$Free(\Delta) = \bigcap MaxCon(\Delta)$$

$$Inc(\Delta) = \Delta - Free(\Delta)$$

An inference α follows from Δ iff there is a positive integer such that α is a classical inference from $Free(\Delta^i)$. We can regard $Free(\Delta^i)$ as a merged knowledgebase though it is more constrained than other proposals. Instead of considering a maximally consistent subset as acceptable. This approach requires the intersection of all maximally consistent subsets with respect to requirements this seems overconstrained, since valuable information maybe dropped as a result.

In [CRS93], preferences over individual formulae in a knowledgebase are used to generate preferences over subsets of the knowledgebase. Given a knowledgebase Δ, and a preference relation $<^*$ over Δ, where for $x, y \in \Delta$, $x <^* y$ iff x is less preferable than y. A preference relation over $\wp(\Delta)$ is obtained from $(\Delta, <^*)$ using either the democratism aggregation principle, or the stratification aggregation principle, defined as follows:

Democratism Let Φ and Ψ be two non-empty subsets of Δ. Φ is democratically preferred to Ψ, denoted $\Psi <^*_{demo} \Phi$ iff for any $\psi \in \Phi \setminus \Psi$, there is $\phi \in \Psi \setminus \Phi$ such that $\psi <^* \phi$ and $\phi \not<^* \psi$.

Stratification Let Φ and Ψ be two non-empty subsets of Δ. Φ is preferred by stratification to Ψ, denoted $\Psi <^*_{strat} \Phi$ iff for all $\phi \in \Phi$, and for all $\psi \in \Psi$, $\psi <^* \phi$ and $\phi \not<^* \psi$.

In general, the democratically preferred subsets of a knowledgebase Δ are preferred subtheories of Δ (using the definition of [Bre89]), and the preferred subtheories of Δ are stratification-based preferred subsets of Δ. However, if $<^*$ is a total pre-ordering, then $<^*_{strat}$ and $<^*_{demo}$ coincide.

We can compare this approach with ours, if we can generate an ordering over individual formulae based on the ranking of agents. Then we can regard the maximal items in $<^*_{strat}$ and $<^*_{demo}$ that are consistent as merged knowledgebases. Since we assume the agents are totally ordered, then $<^*$ will be a total pre-ordering, and so this approach will coincide with that of preferred sub-theories.

Our review here of logical inferencing with inconsistent information has focussed on approaches closely related to knowledgebase merging. However, there are a number of other significant proposals for logical inferencing with inconsistent information (for a review see [Hun98]).

6 Applying the framework

In this section we consider applying the framework. In particular, techniques for handling agents who are not totally ordered, expectations users might have on merging in practice, and introduce the notion of scenarios that are used to identify contingent inconsistencies that could arise in specifications.

6.1 Handling agents who are not totally ordered

So far in our framework, we have focussed on totally ordered sets of agents. However, we can apply these techniques even if the agents are not totally ordered. For any set poset $(A, \geq_A)$, where A is a set of agents and $\geq_A$ is an ordering over them, if $\geq_A$ is transitive and reflexive, then we can form a set of linear ordering from $\geq_A$ by adopting the following linearization function.

Definition 6.1 *Let $\geq_A$ be a transitive reflexive relation. A linearization function, h, is mapping from the set pre-ordering relations to the power set of pre-ordering relations.*

$$h(\geq_A) = \{R \mid R \textit{ is a total ordering over } A \textit{ satisfying the following conditions } \}$$

$$\forall x, y\; ((x \geq_A y) \wedge (y \not\geq_A x)) \rightarrow (xRy) \qquad \text{[Condition 1]}$$

$$\forall x, y\; ((x \geq_A y) \wedge (y \geq_A x)) \rightarrow ((xRy) \vee (yRx)) \qquad \text{[Condition 2]}$$

$$\forall x, y\; ((x \not\geq_A y) \wedge (y \not\geq_A x)) \rightarrow ((xRy) \vee (yRx)) \qquad \text{[Condition 3]}$$

The three conditions on h, mean it is not an ordering preserving mapping. However, if $\geq_A$ is also anti-symmetric, then h is order-preserving, ie.

$$\forall x, y\; x \not\geq_A y \rightarrow xRy$$

Furthermore, if $(A, \geq_A)$ is also acyclic, and so it constitutes a directed acyclic graph, then h gives the topological sorts of the graph.

Example 6.1 *Let A = a,b,c,d, and $\geq_A = \{(a, b), (b, d), (a, c), (c, d)\}$. Here, if we adopt the linearization function, we obtain two preordering relations:*

$$R_1 = \{(a, b), (b, c), (c, d)\}$$
$$R_2 = \{(a, c), (c, b), (b, d)\}$$

Of course there is a cost involved in adopting this approach since the number of merges considered increases enourmously. However, it does not always seem necessary to generate all these ordering. Rather, just adopting one linearization often seems appropriate until that linearization is untenable and should be dropped in favour of alternative linearizations. An analogous approach is taken in [CRS93] that generalizes democratism and stratification discussed in section 5.4.

We don't necessarily expect a linear order at the start. Rather we expect a dialogue with the users to develop a linear ordering. So adopting a linearization from a partial ordering over agents offers a perturbation that may lead to decision being made. It offers a way of exploring possibilities.

In the extreme case, we may have no priorities over agents. We consider this case in [CH99] by adopting a measure of the distance of each possible model of the merged requirements from each agent's position. Using this measure, we can identify an optimal choise in terms of minimizing the sum of the distances for each agent to that choice. This can be seem as adaptation of approaches such as by Katsumo and Mendelzon [KM91].

6.2 Expectations on merges in practice

We now consider how the framework fits into wider technology solutions, and consider the expectations this raises on knowledgebase merging. We start with considering software engineering, and then distributed multi-agent systems.

The development of most large and complex systems necessarily involves many people, each with their own perspectives on the system defined by their knowledge, responsibilities, and commitments. Inevitably, the different perspectives of those involved in the process intersect, giving rise to conflicts. In a formal methods approach, a conflict can be manifested as a logical contradiction, giving rise to the need for consistency checking and inconsistency handling techniques [FGH+94]. However, contradictions are often difficult to handle in requirements specifications. There is often a vast amount of information in any real specification. To address this problem, we can consider decomposing a specification, so as to select the aspects of the specification that are more likely to be in conflict, or where it is more important to know that they are consistent. In addition, we require paraconsistent reasoning and analystical techniques for localizing inconsistency and qualifying inferences in the presence of inconsistency, such as those proposed in [HN97, HN98]. In this context, the proposal in this paper offer further useful tools for software engineering.

If we are using these merging techniques in decision-support, say in software engineering, we are not necessarily looking for the best merge. Rather we are looking at possible or reasonable merges, providing the original agents with a reduced search space of options from which the agents can then select, or if none of the options is chosen, then it may provide further discussions so relevant and provocative choices are useful. For this reason, the underlying logic is not necessarily the same as that for reasoning about beliefs or for reasoning about predictions.

If we are using these merging techniques as part of a resolution strategy to be used by autonomous agents, then the possible merges can be used as part of negotiation strategy — a number of proposal are now being made for automated negotiation for example for buying and selling items over the web — and we could see the proposals in this paper as being incorporated into such systems. Here the emphasis would be on co-operation — aiming to find possible or reasonable merges, providing the original agents with a reduced search space of options from which the agents can then select, or if none of the options is chosen, then options for compromise or automated negotiation.

In all the applications we have considered in this paper, we have adopted the coherence assumption (ie that the union of the domain and regulations is consistent). We assume that a group of agents would normally find it easier to decide on a mutually acceptable coherent set of domain and regulation formulae prior to considering their requirements. Another assumption we have made throughout this paper is that each position is totally ordered. Yet, it in practice it would be desirable to relax this constraint. There are a number of strategies for addressing this — analogous to the techniques used for agents who are not totally ordered (as discussed in section 6.1).

6.3 Incorporating scenarios

In order to explore a specification, with respect to domain knowledge and regulations, we need to also consider some kinds of information that *may* hold, but don't necessarily hold. For example, if we are designing an umbrella, then we may want to know how the specification would be behave if it were raining, or if it were blustery and raining, or if it were sunny and cold. Each of these situations can be regarded as a scenario, and can be represented by a set of formulae. So when we are considering the consistency of our merged requirements, we may also which to consider these scenarios. If a merged requirement was consistent with respect to the union of the Domain and Regulation sets, but inconsistent with respect to the union of the Domain set, the Regulation set, and a scenario set, then the inconsistency is a contingent inconsistency.

The simplest solution to incorporating scenarios is to add the elements of a scenario to the *Dom* set, and obtain a merged set of requirements. Then repeat the process with a different scenario. If the second set of requirements obtained from merging is different from the first, then the requirements require amendment otherwise continue with another scenario.

7 Discussion

In this paper, we have presented a framework for merging requirements from a set of ranked agents. Whilst there are similarities with approaches to merging beliefs from a set of ranked agents, there are fundamental differences. Essentially the differences emanate from viewing beliefs as being for constructing models of the real-world. This is in contrast to viewing requirements as being for constructing models of an ideal world.

We believe that the framework presented here is applicable to a range of data and knowledge engineering problems — in particular for developing decision-support technology for software engineering, and as part of conflict resolution and negotiation strategies for distributed multi-agent systems.

Acknowledgements

This research was partly supported by the ESPRIT FUSION Working Group.

References

[AGM85] C Alchourrón, P Gärdenfors, and D Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[BDP93] S Benferhat, D Dubois, and H Prade. Argumentative inference in uncertain and inconsistent knowledge bases. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 411–419. Morgan Kaufmann, 1993.

[BDP95] S Benferhat, D Dubois, and H Prade. A logical approach to reasoning under inconsistency in stratified knowledge bases. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, volume 956 of *Lecture Notes in Computer Science*, pages 36–43. Springer, 1995.

[BKMS92] C Baral, S Kraus, J Minker, and V Subrahmanian. Combining knowledgebases of consisting of first-order theories. *Computational Intelligence*, 8:45–71, 1992.

[Bre89] G Brewka. Preferred subtheories: An extended logical framework for default reasoning. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 1043–1048, 1989.

[CH97] L Cholvy and A Hunter. Information fusion in logic: A brief overview. In *Symbolic and Quantitative Approaches to Uncertainty*, Lecture Notes in Computer Science. Springer, 1997.

[CH99] L Cholvy and A Hunter. Merging requirements from a set of unranked agents. *Draft paper*, 1999.

[Cho98] L Cholvy. Correctness of requirements in regard with constraints and regulations. Technical report, ONERA-CERT, Toulouse, 1998.

[CRS93] C Cayrol, V Royer, and C Saurel. Management of preferences in assumption based reasoning. In *Information Processing and the Management of Uncertainty in Knowledge based Systems (IPMU'92)*, volume 682 of *Lecture Notes in Computer Science*. Springer, 1993.

[DP97] A Darwiche and J Pearl. On the logic of iterated belief revision. *Artificial Intelligence*, 89:1–29, 1997.

[DP98] D Dubois and H Prade, editors. *Handbook of Defeasible Resoning and Uncertainty Management Systems*, volume 3. Kluwer, 1998.

[EGH95] M Elvang-Goransson and A Hunter. Argumentative logics: Reasoning from classically inconsistent information. *Data and Knowledge Engineering*, 16:125–145, 1995.

[FGH+94] A Finkelstein, D Gabbay, A Hunter, J Kramer, and B Nuseibeh. Inconsistency handling in multi-perspective specifications. *Transactions on Software Engineering*, 20(8):569–578, 1994.

[Gab99] D Gabbay. *Fibring Logics*, volume 38 of *Oxford Logic Guides*. Oxford University Press, 1999.

[Gar88] P Gardenfors. *Knowledge in Flux*. MIT Press, 1988.

[GR96] D Gabbay and O Rodrigues. A methodology for iterated theory change. In *Practical Reasoning*, volume 1085 of *Lecture Notes in Computer Science*. Springer, 1996.

[GR97] D Gabbay and O Rodrigues. Structured belief bases: A practical approach to prioritized base revision. In *Qualitative and Quantitative Practical Reasoning*, volume 1244 of *Lecture Notes in Computer Science*. Springer, 1997.

[HN97] A Hunter and B Nuseibeh. Analysing inconsistent specifications. In *Proceedings of 3rd International Symposium on Requirements Engineering*, pages 78–86. IEEE Computer Society Press, 1997.

[HN98] A Hunter and B Nuseibeh. Managing inconsistent specifications: Reasoning, analysis and action. *ACM Transactions on Software Engineering Methodology*, 7:335–367, 1998.

[Hun98] A Hunter. Paraconsistent logics. In D Gabbay and Ph Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, pages 11–36. Kluwer, 1998.

[KM91] H Katsuno and A Mendelzon. Propositional knowledge base revision and minimal change. *Artificial Intelligence*, 52:263–294, 1991.

[KM92] H Katsuno and A Mendelzon. On the difference between updating a knowledgebase and revising it. *Belief Revision*, pages 183–203, 1992.

[KP98] S Konieczny and R Pino Perez. On the logic of merging. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR98)*, pages 488–498. Morgan Kaufmann, 1998.

[KP99] S Konieczny and R Pino Perez. Merging with integrity constraints. In *Qualitative and Quantitative Approaches to Reasoning with Uncertainty*, volume 1638 of *Lecture Notes in Computer Science*. Springer, 1999.

[Leh95] D Lehmann. Belief revision, revised. In *Proceedings IJCAI-95*, pages 1534–1540, 1995.

[LS98] P Liberatore and M Schaerf. Arbitration (or how to merge knowledgebases). *IEEE Transactions on Knowledge and Data Engineering*, 10:76–90, 1998.

[MR70] R Manor and N Rescher. On inferences from inconsistent information. *Theory and Decision*, 1:179–219, 1970.

[Rya92] M Ryan. Representing defaults as sentences with reduced priority. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*. Morgan Kaufmann, 1992.