ELSEVIER International Journal of Approximate Reasoning 18 (1998) 107–130

INTERNATIONAL JOURNAL OF APPROXIMATE REASONING

Specialisation calculus and communication

Josep Puyol-Gruart *, Lluís Godo, Carles Sierra

Artificial Intelligence Research Institute (IIIA), Spanish Scientific Research Council (CSIC), Campus Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain

Received 1 December 1995; accepted 1 June 1997

Abstract

In this paper we propose a deductive calculus aiming at improving the query/simple-answer communication behaviour of many intelligent systems. In an uncertain reasoning context this behaviour consists of getting certainty values for propositions as answers to queries. Instead, with our calculus, answers to queries will become sets of formulas: a set of propositions and a set of specialised rules containing propositions for which the truth value is unknown in their left part. This type of behaviour is much more informative because it returns to users not only the answer to a query but all the relevant information, related to the answer, necessary to, possibly, improve the solution. To exemplify the general approach a family of propositional rule-based languages founded on multiple-valued logics is presented and formalised. The deductive system defined on top of these languages is based on a Specialisation Inference Rule (SIR): $(A_1 \wedge A_2 \wedge \cdots \wedge A_n \rightarrow P, V), (A_1, V') \vdash (A_2 \wedge \cdots \wedge A_n \rightarrow P, V'')$, where V, V' and V'' are truth intervals. This inference rule provides a way of generating rules containing less conditions in their premise by eliminating the conditions for which a definitive truth value already exists. The soundness and atom completeness of the deductive system are proved. The implementation of this deductive calculus is based on partial deduction techniques. Finally, an example of the application of the specialisation calculus to a multi-agent system is provided. © 1998 Elsevier Science Inc.

Keywords: Partial deduction; Multi-agent system; Multiple-valued logic

* Corresponding author. E-mail: puyol@iiia.csic.es; www: http://www.iiia.csic.es.

PII S0888-613X(97)10006-8

1. Introduction

1.1. Motivation

The main concern of this paper is to introduce a many-valued logical calculus based on rule specialisation to model a type of cooperative communication between autonomous agents in the presence of imperfect or imprecise knowledge. Other important communicational issues such as protocols or agent communication languages are not dealt within this paper. Rather, we focus on the informational content of the communication between agents.

For the sake of simplicity and readability, we restrict ourselves to architectures of multi-agent systems composed of a set of autonomous *rule-based agents* communicating each other by means of message passing. Moreover, we only consider two types of asynchronous communication actions: *queries* and *answers*. Finally, external users of the multi-agent system are supposed to interact with the autonomous agents through an interface that allows them to pose queries and give answers.

In a very simplified way, the standard behaviour of traditional knowledge-based agents when communicating could be described as follows: when an agent is inquired whether a given proposition holds, the agent starts its deductive machinery in order to find out a proof for that proposition. If it succeeds, it gives back either the truth value *true* in the case of classical reasoning or a partial degree of truth or certainty in the case of approximate or uncertainty reasoning. If it fails, under the open world assumption, the answer is *unknown*.

In the case of rule-based agents, we propose to improve this simple communication process by using in a more effective way the information stored in the rule base of an agent. For instance:

1. When the user of an agent makes a query, he might be interested in knowing not only about the query itself but also about other related facts that can be useful for the problem being solved. It can be also the case that the user might be interested in knowing which conclusions can be drawn from the proposition being queried.
2. When an agent is not able to answer a query because it has not been provided with enough information, he will probably answer with the value *unknown*, as already commented. However, even in this case, the answer may be much more informative if the agent let the user or another agent know which is the lacking information causing the failure of the answer.

All this 'hidden' information is somehow actually used by humans when cooperating in solving problems. Indeed, looking carefully at, for instance, how physicians cooperate and communicate in a diagnosis problem, it can be noticed that they may:

- *condition their decisions*. Suppose it is not known whether a patient is allergic to penicillin. Then a physician asked for the possibility of giving penicillin as

treatment would answer: '*Penicillin is a good treatment from a clinical point of view* provided that *the patient has no allergy to penicillin*'.

- *provide suggestions to be considered together with the answer of a query*. Instead of strictly answering whether there is an infection or not, a physician may answer: '*Pneumococcus has been isolated in the culture of sputum. In this case it is strongly* suggested *to make an antibiogram to the patient*'.
- *provide conditioned suggestions to be considered together with decisions*. This would correspond to a combination of the above two communication patterns. For instance, '*Ciprofloxacine is a good treatment, but if the patient is a woman breast-feeding she must stop breast-feeding*'.

To model such communication patterns, we need to extend the agent answering procedure, by allowing it to answer queries with sets of formulas (rules and propositions). We propose to do it by means of a calculus based on rule specialisation. Specialisation as understood in this paper is related to the notion of partial evaluation expressed in the well known Kleene's Theorem [10]. Specialisation Calculus is based on logic, then we use the term *partial deduction* instead of partial evaluation [12]. Partial deduction algorithms have been used intensively in logic programming [5,11,13,18,20], mainly for efficiency purposes.

1.2. Partial deduction of rules

In classical (boolean) rule bases, deduction is mainly based on the modus ponens inference rule

$$A, A \rightarrow B \vdash B.$$

In the case that A denotes a conjunction of conditions $A_1 \wedge A_2$, the above inference rule is only applicable when every condition of the premise, i.e. A_1 and A_2, is satisfied, otherwise nothing can be inferred. However, if we only know that condition A_1 is satisfied, due to the well-known logical equivalence $(A_1 \wedge A_2) \rightarrow B \equiv A_1 \rightarrow (A_2 \rightarrow B) \equiv A_2 \rightarrow (A_1 \rightarrow B)$, we can use partial deduction to extract the maximum information from incomplete knowledge in the sense of the following specialisation inference rule (SIR):

$$A_1, A_1 \wedge A_2 \rightarrow B \vdash A_2 \rightarrow B.$$

The rule $A_2 \rightarrow B$ is called the *specialisation* of $A_1 \wedge A_2 \rightarrow B$ with respect to the proposition A_1. Notice that in the particular case that the rule has only one condition in the premise, we may resort to the usual modus ponens rule.

The following are the corresponding functional specification of what a rule specialisation process is.

Definition 1.1 (*Rule specialisation*). Let R be a set of rules and P a set of literals. We note rules as pairs, $r = (m_r, c_r)$, where m_r is the premise (a set of literals)

and c_r is the conclusion (a literal). The rule specialisation is defined as a function:

$$\mathscr{S}_{\mathscr{R}} : R \times P \rightarrow R \times P,$$

$$\mathscr{S}_{\mathscr{R}}(r,p) = \begin{cases} (r,\emptyset) & \text{if } p \notin m_r, \\ (\emptyset, c_r) & \text{if } m_r = \{p\}, \\ ((m_r - \{p\}, c_r), \emptyset) & \text{otherwise.} \end{cases}$$

The extension to specialisation of agent's rule bases is straightforward.

Definition 1.2 (*Agent specialisation*). Let A be a set of agents. We note Agents as pairs $a = (R,P)$, where R is a set of rules and P is a set of literals. Agent specialisation is defined as a function:

$$\mathscr{S} : A \rightarrow A,$$

$$\mathscr{S}(a) = \begin{cases} \mathscr{S}((R - \{r\} + \{r'\}, P + \{p'\})) & (*), \\ a & \text{otherwise.} \end{cases}$$

$(*)$ if $P \neq \emptyset$ and $\exists p \in P$ and $\exists r \in R$ such that $\mathscr{S}_{\mathscr{R}}(r,p) = (r',p')$ and $r' \neq r$.

In other words, the specialisation of an *agent's rule base* consists on the exhaustive specialisation of its rules. Rules that only have one condition appearing in the set of literals will be eliminated and a new literal will be added. This new literal will be used again to specialise the agent. The process will finish when the agent has no rule containing on its conditions a known literal. This approach is different for instance from the logic programming one used in [13]. There, partial deduction is goal driven, whereas here partial deduction is data driven.

In this paper we propose the use of this technique to improve the communication behaviour between agents by allowing agents to answer a query with a part of the result of the specialisation of its rule base. In an approximate reasoning context we propose to extend the above boolean specialisation inference rule to encompass partial truth, for instance in the following way:

$$(A_1,\alpha), (A_1 \wedge A_2 \rightarrow B, \beta) \vdash (A_2 \rightarrow B, \beta')$$

meaning that if A_1 is known to be true at least to the degree α and the rule $A_1 \wedge A_2 \rightarrow B$ is true at least to the degree β, then the specialised rule $A_2 \rightarrow B$ is true at least to a degree $\beta' = f(\alpha,\beta)$, being f a suitable combination function.

More concretely, in Section 2 we formally describe both the semantics and syntax of a many-valued logical calculus for partial deduction of rule bases. Section 3 is devoted to the functional description of an agent specialisation mechanism. In Section 4 an example on multi-agent medical diagnosis is presented, showing the usefulness of the communication mechanism based on specialisation. Finally, a discussion on the results is presented in Section 5.

2. Formalisation of a many-valued specialisation calculus for rule bases

In this section we present a parametric family of many-valued calculi for rule specialisation. Each calculus is determined by a particular algebra of truth-values belonging to a parametric family of algebras that is described next.

Throughout this paper, an *Algebra of truth-values* $A_{n,T} = \langle A_n, \leqslant, N_n, T, I_T \rangle$ will be a finite linearly ordered residuated lattice with a negation operation, that is

1. $(A_n, \leqslant)$ is a chain of n elements: $0 = a_1 < a_2 < \cdots < a_n = 1$ where 0 and 1 are the booleans *False* and *True* respectively.
2. The negation operation N_n is a unary operation defined as $N_n(a_i) = a_{n-i+1}$, the only definable order-preserving involutive mapping in $\langle A_n, < \rangle$, i.e. it holds
   - N1: if $a < b$ then $N_n(a) > N_n(b)$ $\forall a, b \in A_n$.
   - N2: $(N_n)^2 = Id$.
3. The conjunction operator T is a binary operation such that the following properties hold $\forall a, b, c \in A_n$.
   - T1: $T(a, b) = T(b, a)$.
   - T2: $T(a, T(b, c)) = T(T(a, b), c)$.
   - T3: $T(0, a) = 0$.
   - T4: $T(1, a) = a$.
   - T5: if $a \leqslant b$ then $T(a, c) \leqslant T(b, c)$ for all c.
4. The implication operator I_T is defined by residuation with respect to T, i.e.

$$I_T(a, b) = \mathrm{Max}\{c \in A_n \mid T(a, c) \leqslant b\}.$$

Such an implication operator satisfies the following properties $\forall a, b, c \in A_n$.

- I1: $I_T(a, b) = 1$ if, and only if, $a \leqslant b$.
- I2: $I_T(1, a) = a$.
- I3: $I_T(a, I_T(b, c)) = I_T(b, I_T(a, c))$.
- I4: if $a \leqslant b$ then $I_T(a, c) \geqslant I_T(b, c)$ and $I_T(c, a) \leqslant I_T(c, b)$.
- I5: $I_T(T(a, b), c) = I_T(a, I_T(b, c))$.

As it is easy to notice from the above definition, any of such truth-values algebras is completely determined as soon as the set of truth-values A_n and the conjunction operator T are chosen. So, varying these two characteristics we generate a family of different multiple-valued logics. For instance, taking $T(a_i, a_j) = a_{\min(i,j)}$ or $T(a_i, a_j) = a_{\min(n, n-i+j)}$ we get the well-known Gödel's and Łukasiewicz's semantics (truth-tables) for finitely-valued logics [6–9].

In the following we describe the language, the semantics and the deduction system (specialisation calculus) of a particular logic corresponding to a given algebra $A_{n,T}$.

The propositional language $\mathscr{L} = (A_n, \Sigma, \mathscr{C}, \text{Mv-}\mathscr{S})$ is defined by:

- a set of truth-values A_n,
- a signature Σ consisting on a set of propositional variables plus *true*,

- a set of Connectives $\mathscr{C} = \{\neg, \wedge, \rightarrow\}$, and
- a set of Sentences Mv-$\mathscr{S}$ = Mv-Literals$(\Sigma, A_n) \cup$ Mv-Rules(Σ, A_n), where
  Mv-Atoms (Σ, A_n): $\{(p, V) \mid p \in \Sigma, V$ interval of truth-values in $A_n\}$.
  Mv-Literals (Σ, A_n): $\{(p, V), (\neg p, V) \mid (p, V) \in$ Mv-Atoms$\}$.
  Mv-Rules (Σ, A_n): $\{(p_1 \wedge p_2 \wedge \cdots \wedge p_n \rightarrow q, V) \mid p_i$ and q are literals, V is an interval of truth values in A_n, and $\forall i, j (p_i \neq p_j, p_i \neq \neg p_j, q \neq p_j, q \neq \neg p_j)\}$.

That is, sentences of $\mathscr{L}$, which will be generically called *mv-formulas*, are indeed signed formulas under the form of pairs of usual propositional formulas (restricted to be literals or rules) and intervals of truth-values.

Notation conventions. We shall commonly use:

p, q	to denote literals from Σ
φ, ψ	to denote arbitrary propositional sentences
A, B	to denote arbitrary mv-formulas, and
Γ, Δ	to denote sets of mv-formulas

Further, $a, b, \ldots$ will denote truth-values from A_n while $V, W, \ldots$ will denote intervals of truth-values. For simplicity we shall also write $(\varphi, \mathbf{a})$ to denote the mv-formula $(\varphi, [a, a])$.

The *semantics* is obviously determined by the connective operators of the truth-value algebra $A_{n,T}$. *Interpretations* are defined by valuations ρ mapping the (propositional) sentences to truth-values of A_n fulfilling the following conditions [1]:

$$\rho(true) = 1,$$

$$\rho(\neg p) = N_n(\rho(p)),$$

$$\rho(p_1 \wedge \cdots \wedge p_n \rightarrow q) = I_T(T(\rho(p_1), \ldots, \rho(p_n)), \rho(q)).$$

Having truth-values explicit in the sentences enables us to define a classical satisfaction relation in spite of the models being multiple-valued assignments. The *satisfaction relation* between interpretations and mv-formulas is defined as

$$\rho \models (\varphi, V) \quad \text{iff } \rho(\varphi) \in V$$

and it is extended to a *semantical entailment* between sets of mv-formulas and mv-formulas as usual

$$\Gamma \models (\varphi, V) \quad \text{iff } \rho \models (\varphi, V) \text{ for all } \rho \text{ such that } \rho \models A \text{ for all } A \in \Gamma.$$

Taking into account the motivations introduced in Section 1.1, the deduction system we consider for rule specialisation in our many-valued logical framework is the following one.

[1] The expression $T(r_1, r_2, r_3, \ldots)$ is the recurrent application of T as $T(r_1, T(r_2, T(r_3 \ldots)))$.

Definition 2.1. The many-valued specialisation calculus (Mv-SC *for short*) is defined by the following axioms

A1: $(\varphi, [0,1])$.
A2: $(true, 1)$.

and by the following inference rules

Weakening:	from (φ, V_1) infer (φ, V_2), where $V_1 \subseteq V_2$.
Not-introduction:	from (p, V) infer $(\neg p, N^*(V))$, where p is an atom.
Not-elimination:	from $(\neg p, V)$ infer $(p, N^*(V))$, where p is an atom.
Composition:	from (φ, V_1) and (φ, V_2) infer $(\varphi, V_1 \cap V_2)$.
Specialization:	from (p_i, V) and $(p_1 \wedge \cdots \wedge p_n \to q, W)$ infer $(p_1 \wedge \cdots \wedge p_{i-1} \wedge p_{i+1} \wedge \cdots \wedge p_n \to q, \mathrm{MP}^*_T(V, W))$,

where $N_n^*([a,b]) = [N_n(b), N_n(a)]$ and $\mathrm{MP}^*_T(V, W)$ is the minimal interval containing all solutions for z in the family of functional equations

$$I_T(a, z) = b$$

varying $a \in V$ and $b \in W$.

Remark 2.1. In the above description of the specialisation inference rule we are assuming $n \geqslant 2$. It is understood that if $n = 1$ then the specialization rule of inference turns out into the following *modus ponens* inference rule: from (p, V) and $(p \to q, W)$ infer $(q, \mathrm{MP}^*_T(V, W))$.

The notion of proof inside Mv-SC is defined as usual.

Definition 2.2. There exists a proof of A from Γ, written $\Gamma \vdash_{\mathrm{SC}} A$, if there is a finite sequence

$$B_1, \ldots, B_m = A$$

such that each B_i is either axiom A1 or A2, an mv-formula from Γ, or has been deduced from previous B_j by application of some of the above five inference rules.

It is easy to check that the above specialisation calculus is sound.

Theorem 2.1 (Soundness). *If* $\Gamma \vdash_{\mathrm{SC}} A$ then $\Gamma \models A$.

Proof. Axioms A1 and A2 are trivially satisfied by any interpretation. Weakening, not-introduction, not-elimination and composition inference rules also trivially preserve truth. Let us check the truth preservation of Specialization rule for the simplest modus ponens case, i.e. when $n = 1$. We shall prove that

$$\{(p, U), (p \to q, V)\} \models (q, W) \quad \text{if} \quad \mathrm{MP}^*_T(U, V) \subseteq W.$$

By definition, $\mathrm{MP}^*_T(U,V)$ is the minimal interval containing all the solutions of the following family of functional equations:

$$I_T(a,z) = b$$

for any $a \in U$, and $b \in V$. Suppose then that $\mathrm{MP}^*_T(U,V) \subseteq W$ and let ρ a model of (p,U) and of $(p \to q, V)$. Let $a = \rho(p)$ and $b = \rho(p \to q) = I_T(\rho(p), \rho(q))$. Then $a \in U$ and $b \in V$. By hypothesis, any solution for $\rho(p)$ satisfying the equation $I_T(a, \rho(q)) = b$ must be in W, so ρ is also a model of (q, W). □

On the other hand, it is obvious that the logic Mv-SC is not complete. For instance, if we consider the two-valued case, i.e. $A_2 = \{0,1\}$, we have $\{(p \to q, 1), (q \to r, 1)\} \models (p \to r, 1)$ but $\{(p \to q, 1), (q \to r, 1)\} \nvdash_{\mathrm{SC}} (p \to r, 1)$. It is also the case that the language is not complete for literal deduction in general. For instance, we have $\{(p \to q, 1), (\neg p \to q, 1)\} \models (q, 1)$ but $\{(p \to q, 1), (\neg p \to q, 1)\} \nvdash_{\mathrm{SC}} (q, 1)$. However, it can be shown that the system is complete for mv-atom deduction provided we further restrict the language basically by not allowing negated literals in the language. This restricted mv-atom completeness can be seen as a many-valued counterpart of the completeness of classical modus ponens for atom deduction with propositional Horn clauses. This will be shown in Section 2.1.

2.1. Mv-atom completeness

The sub-language we consider is the negation free fragment of $\mathscr{L}$. Namely we define an *Mv-Horn-Rule* as an mv-rule $(p_1 \wedge p_2 \wedge \cdots \wedge p_n \to q, V)$ such that p_i and q are atomic symbols and $V = [a,1]$ is an upper interval of truth-values of A_n with $a > 0$, and $\forall i,j (p_i \neq p_j, q \neq p_j)$. Then, we define the restricted many-valued propositional sub-language $\mathscr{RL}$ as the following 4-tuple:

$$\mathscr{RL} = (A_n, \Sigma, \mathscr{C}, \text{Mv-}\mathscr{RS})$$

being $\text{Mv-}\mathscr{RS} = \text{Mv-Atoms}(\Sigma, A_n) \cup \text{Mv-Horn-Rules}(\Sigma, A_n)$, where Mv-Horn-Rules$(\Sigma, A_n)$ denotes the set of Mv-Horn-Rules that can be built from Σ and A_n. Within this sub-language, the not-introduction and not-elimination inference rules of Mv-SC have no sense. Accordingly, we define the sub-calculus Mv-RSC by axioms $A1, A2$ and the Weakening, Composition and Specialisation inference rules. Deduction in Mv-RSC will be denoted by $\vdash_{\mathrm{RSC}}$. It is interesting to remark that, in the restricted language $\mathscr{RL}$, the specialisation inference rule takes this form.

Specialisation: from $(p_i, [a_1, a_2])$ and $(p_1 \wedge \cdots \wedge p_n \to q, [b,1])$ infer $(p_1 \wedge \cdots \wedge p_{i-1} \wedge p_{i+1} \wedge \cdots \wedge p_n \to q, [T(a_1,b),1])$ since it is easy to show from the definition of MP^*_T that $\mathrm{MP}^*_T([a_1,a_2],[b,1]) = [T(a_1,b),1]$.

The soundness of $\vdash_{RSC}$ is naturally inherited from $\vdash_{SC}$. Moreover, we shall show the following completeness result for mv-atom deduction.

Theorem 2.2 (mv-atom completeness). *Let Γ be a set of mv-formulas from Mv-RS. Then, if $\Gamma \models (p, [a, 1])$, we also have $\Gamma \vdash_{RCS} (p, [a, 1])$, for any propositional variable $p \in \Sigma$.*

As usual, we will prove that if $\Gamma \nvdash_{RCS}(p, [a, 1])$ then $\Gamma \not\models (p, [a, 1])$. To do that we shall make use of standard logical machinery adapted to our particular case.

Definition 2.3. A set Γ of mv-formulas is RSC-inconsistent if there exists a propositional variable such that $\Gamma \vdash_{RSC} (p, \emptyset)$.

Therefore, a set of mv-formulas Γ will be RSC-*consistent* if Γ is not RSC-inconsistent.

Definition 2.4. A maximally atomic-consistent set Γ is a set of mv-formulas such that:
1. for all $q \in \Sigma$, there exists $a \in A_n$ such that $(q, \mathbf{a}) \in \Gamma$,
2. Γ is RSC-consistent.

In the following we assume that Γ denotes a set of mv-formulas from $\mathscr{M}\sqsubseteq - \mathscr{RS}$. Next step is to prove that if Γ is RSC-consistent then Γ is satisfiable.

Lemma 2.1. *For any RSC-consistent Γ, if $\Gamma \cup \{(p, [a, 1])\} \vdash_{RSC} (p, \emptyset)$, then $\Gamma \vdash_{RSC} (p, [0, a))$, where $[0, a) = \{b \in A \mid b < a\}$.*

Proof. If Γ is RSC-consistent and $\Gamma \cup \{(p, [a, 1])\} \vdash_{RSC} (p, \emptyset)$, the only possibility is to have $\Gamma \vdash_{RSC} (p, W)$ such that $[a, 1] \cap W = \emptyset$, that is $W \subseteq [0, a)$, and thus we also have $\Gamma \vdash_{RSC} (p, [0, a))$. □

Lemma 2.2. *Let $a_i, a_{i+1} \in A$. If $\Gamma \cup \{(p, \mathbf{a_i})\}$ and $\Gamma \cup \{(p, \mathbf{a_{i+1}})\}$ are inconsistent then $\Gamma \cup \{(p, [a_i, a_{i+1}])\}$ is also inconsistent.*

Proof. Suppose Γ is consistent, otherwise the result is trivial, and suppose that $\Gamma \cup \{(p, \mathbf{a_i})\}$ and $\Gamma \cup \{(p, \mathbf{a_{i+1}})\}$ are inconsistent. Then, it must be the case that $\Gamma \vdash_{RSC} (p, V)$ with $a_i \notin V$ and $V \neq \emptyset$. Analogously, $\Gamma \vdash_{RSC} (p, W)$ with $a_{i+1} \notin W$ and $W \neq \emptyset$. Therefore, $\Gamma \vdash_{RSC} (p, V \cap W), V \cap W \neq \emptyset$ and $[a_i, a_{i+1}] \cap (V \cap W) = \emptyset$. Thus, we have that $\Gamma \cup \{(p, [a_i, a_{i+1}])\} \vdash_{RSC} (p, \emptyset)$, and so, $\Gamma \cup \{(p, [a_i, a_{i+1}])\}$ is inconsistent. □

Lemma 2.3. *If Γ is RSC-consistent then, for any propositional variable p, there exists $a \in A_n$ such that $\Gamma \cup \{(p, \mathbf{a})\}$ is RSC-consistent.*

Proof. Suppose $\Gamma \cup \{(p, a)\}$ is inconsistent for any $a \in A_n$. Then, by repeated application of previous lemma, $\Gamma \cup \{(p, [0, 1])\}$ is also inconsistent, but $(p, [0, 1])$ is an axiom, thus Γ itself should be inconsistent: contradiction. □

Lemma 2.4. *If Γ is RSC-consistent, it can be extended to a maximally atomic-consistent set Γ^*.*

Proof. Let $p_1, p_2, \ldots, p_n, \ldots$ be the set of propositional variables Σ. Define $\Gamma_0 = \Gamma$, and for $i > 0$ define $\Gamma_i = \Gamma_{i-1} \cup \{(p_i, \mathbf{a})\}$, a being a truth-value such that $\Gamma_{i-1} \cup \{(p_i, \mathbf{a})\}$ is consistent. By the previous lemma such an a always exists. Finally, let $\Gamma^* = \cup_{i \geqslant 0} \Gamma_i$. Then it is easy to check that Γ^* is maximally atomic-consistent.

- Γ^* is RSC-consistent. Suppose not. Then there exists a finite subset $\Gamma^0 \subseteq \Gamma$ such that $\Gamma^0 \vdash_{\mathrm{RSC}} (p, \emptyset)$, for some p. By construction, there is j such that $\Gamma^0 \subseteq \Gamma_j$. Then Γ_j would be inconsistent, contradiction.
- Γ^* is maximally atomic-consistent. By construction. □

Lemma 2.5. *If Γ is RSC-consistent then Γ is satisfiable.*

Proof. Let Γ be consistent and let Γ^* be a maximally atomic extension of Γ. Define a valuation $\rho_\Gamma : \Sigma \to A_n$ as follows: $\rho_\Gamma(p) = a$ if $(p, \mathbf{a}) \in \Gamma^*$.

Then it is easy to show that ρ_Γ is a model of Γ. Namely, we have to show that, for any $(\varphi, V) \in \Gamma$, we have that $\rho_\Gamma(\varphi) \subseteq V$. We consider only the case $(p \to q, V) \in \Gamma$. Let $(p, \mathbf{a}), (q, \mathbf{b}) \in \Gamma^*$. We shall show then that $I_T(\rho_\Gamma(p), \rho_\Gamma(q)) = I_T(a, b) \in V$. If $a \leqslant b$, then $I_T(a, b) = 1$, and obviously, by definition, $1 \in V$. Suppose otherwise that $a > b$. Since Γ^* is consistent, so $\{(p \to q, V), (p, \mathbf{a}), (q, \mathbf{b})\}$ is. But, using modus ponens, $\{(p \to q, V), (p, \mathbf{a})\} \vdash_{\mathrm{RSC}} (q, \mathrm{MP}^*_T(\{a\}, V))$ and thus $\{(p \to q, V), (p, \mathbf{a}), (q, \mathbf{b})\} \vdash_{\mathrm{RSC}} (q, \{b\} \cap \mathrm{MP}^*_T(\{a\}, V))$. Since $\{(p \to q, V), (p, \mathbf{a}), (q, \mathbf{b})\}$ is consistent, $\{b\} \cap \mathrm{MP}^*_T(\{a\}, V) \neq \emptyset$, that is, $b \in \mathrm{MP}^*_T(\{a\}, V)$. In other words, since V is an upper interval there must exist $c \in V$ such that $b \geqslant T(a, c)$. Let $d = \max\{c \mid T(a, c) \leqslant b\}$. Since V is an upper interval, $d \in V$, therefore we have $I_T(a, b) = \max\{c \mid T(a, c) \leqslant b\} = d$, and the lemma is proved. □

Finally, from this lemma, Theorem 2.2. is direct corollary.

Corollary 2.1. *If $\Gamma \models (p, [a, 1])$ then $\Gamma \vdash_{\mathrm{RSC}} (p, [a, 1])$, for any propositional variable p.*

Proof. If $\Gamma \not\vdash_{\mathrm{RSC}} (p,[a,1])$ then, by Lemma 2.1, $\Gamma \cup \{(p,[0,a))\}_{\mathrm{RSC}}(p,\emptyset)$, i.e. $\Gamma \cup \{(p,[0,a))\}$ is RSC-consistent, thus, by Lemma 2.5, $\Gamma \cup \{(p,[0,a))\}$ is satisfiable, thus there exists ρ model of Γ such that $\rho(p) < a$, i.e. $\not\models (p,[a,1])$. □

3. Inference algorithm

In this section we present an inference algorithm based on the specialisation calculus. We are interested in obtaining the intervals of truth values for the facts deducted minimising the number of deductive steps.

In order to preserve the correctness of the inference algorithm with respect to the semantics of the specialisation calculus, the algorithm does not introduce any extra-logical component. Deduction is implemented by using just the axioms and inference rules presented in the previous section.

We consider that a proposition has a *definitive* value when there are no rules that can contribute to its *provisional* value (initially [0,1]), producing a more precise one by means of applications of the composition inference rule. We will use a proposition to specialise rules only when that proposition has a definitive value. This restriction permits that a rule be substituted by its specialised versions when no more specialisation is possible for the condition being eliminated from its premise. When there are no conditions left in the premise of a rule the conclusion of the rule is generated. The weakening inference rule will not be used in the deductive process, it will only be used when necessary at query answering time.

3.1. Internal representation

We propose a slight change of representation for mv-rules that allows us to simplify the functional descriptions of the algorithm.

Definition 3.1 (*Mv-rule*). A mv-rule is a tuple $r = (m_r, c_r, \rho_r)$, where m_r is the premise (a set of literals), c_r is the conclusion (a literal) and ρ_r is the truth-value of the rule (an interval of truth-values such that $\rho_r = [\alpha, 1]$ and $\alpha \in A_n$).

For instance the rule $(c \wedge d \rightarrow e, [\rho_3, \mathbf{1}])$ – written using the notation of Section 2 – will be represented from now on as the tuple: $(\{c,d\}, e, [\rho_3, 1])$.

Next we define a representation for sets of rules and sets of propositions that we will refer to as the mental state [17] of the agent. The representation consists of mapping each atom in Σ to its current interval of truth-values and the (possibly empty) set of mv-rules that conclude it, or its negation.

Definition 3.2 (*Agent*) Let R be a set of mv-rules in language $\mathscr{L}$. We define an agent mental state AG as a mapping

$$\mathrm{AG}: \Sigma \to \mathrm{Int}(A_n) \times 2^R.$$

where, for each $f \in \Sigma$, $\mathrm{AG}(f) = (\rho_f, R_f)$, being $R_f = \{r \in R \mid r = (m_r, c_r, \rho_r)$ and $c_r = f$ or $c_r = \neg f\}$.

The representation of an agent's mental state will evolve as deduction proceeds. We represent the initial mental state of an agent as a mapping from any atom into [0,1] and the set of rules deducing it. It means that the atoms initially have their most imprecise value – that is [0,1]. Axiom A1, Definition 2.1. in Section 2. Notice that an agent with all atoms with truth-values [0,1] is always consistent in our calculus.

Example 3.1. Now we can see an example of an initial mental state. Suppose that we have the following set of mv-rules:

$$R = \{(\{a,b\}, c, [\rho_1, 1]), (\{a,f\}, \neg c, [\rho_2, 1]), (\{c,d\}, e, [\rho_3, 1])\}.$$

It is easy to see that the set Σ is

$$\Sigma = \{(a,b,c,d,e,f)\}.$$

And that the state is:

$$\begin{aligned}
&\mathrm{AG}(a) = ([0,1], \emptyset),\\
&\mathrm{AG}(b) = ([0,1], \emptyset),\\
&\mathrm{AG}(c) = ([0,1], \{(\{a,b\}, c, [\rho_1, 1]), (\{a,f\}, \neg c, [\rho_2, 1])\}),\\
&\mathrm{AG}(d) = ([0,1], \emptyset),\\
&\mathrm{AG}(e) = ([0,1], \{(\{c,d\}, e, [\rho_3, 1])\}),\\
&\mathrm{AG}(f) = ([0,1], \emptyset).
\end{aligned}$$

3.2. Specialisation

To describe the algorithm we define first of all the specialisation of a mv-rule. Giving a mv-rule and a mv-atom, the mapping $\mathscr{S}_{\mathscr{R}}$ specialises the mv-rule with respect to that mv-atom generating a specialised mv-rule, or a new mv-atom if the rule had a single condition.

Definition 3.3. We define the specialisation of a mv-rule with respect to a fact, $\mathscr{S}_{\mathscr{R}}$ as a mapping:

$$\mathscr{S}_{\mathscr{R}}: \text{Mv-rules} \times \text{Mv-Atoms} \to \text{Mv-Rules} \times \text{Mv-Atoms},$$

$$\mathscr{S}_{\mathscr{R}}(r,(p,\rho_p)) = \begin{cases} (r,(\text{true},[0,1])) & \text{if } p \notin m_r \text{ and } \neg p \notin m_r, \\ (r',(\text{true},[0,1])) & \text{if } p \in m_r \text{ or } \neg p \in m_r, \\ (\emptyset,(q,\alpha)) & \text{if } m_r = \{p\} \text{ or } m_r = \{\neg p\} \\ & \quad \text{and } c_r = q \text{ or } c_r = \neg q, \end{cases}$$

where

$$r' = \begin{cases} (m_r - \{p\}, c_r, \mathrm{MP}^*_T(\rho_p,\rho_r)) & \text{if } p \in m_r, \\ (m_r - \{\neg p\}, c_r, \mathrm{MP}^*_T(N^*_n(\rho_p),\rho_r)) & \text{if } \neg p \in m_r \end{cases}$$

and

$$\alpha = \begin{cases} \mathrm{MP}^*_T(\rho_p,\rho_r) & \text{if } m_r = \{p\} \text{ and } c_r = q\,, \\ \mathrm{MP}^*_T(N^*_n(\rho_p),\rho_r)) & \text{if } m_r = \{\neg p\} \text{ and } c_r = q, \\ N^*_n(\mathrm{MP}^*_T(\rho_p,\rho_r)) & \text{if } m_r = \{p\} \text{ and } c_r = \neg q, \\ N^*_n(\mathrm{MP}^*_T(N^*_n(\rho_p),\rho_r)) & \text{if } m_r = \{\neg p\} \text{ and } c_r = \neg q\,. \end{cases}$$

Example 3.2. We can specialise the second mv-rule of the last example with respect to $(f,[\rho_4,\rho'_4])$, obtaining

$$\begin{aligned} &\mathscr{S}_{\mathscr{R}}((\{a,f\},\neg c,[\rho_2,1]),(f,[\rho_4,\rho'_4]) \\ &\quad = ((\{a\},\neg c,[\mathrm{MP}_T(\rho_4,\rho_2),1]),(\text{true},[0,1])). \end{aligned}$$

If we specialise again the rule so obtained with respect to $(a,[\rho_5,\rho'_5])$ we get a mv-atom

$$\begin{aligned} &\mathscr{S}_{\mathscr{R}}((\{a\},\neg c,[\mathrm{MP}_T(\rho_4,\rho_2),1]),(a,[\rho_5,\rho'_5]) \\ &\quad = (\emptyset,(c,N^*_n(\mathrm{MP}^*_T([\mathrm{MP}_T(\rho_4,\rho_2),1],[\rho_5,\rho'_5])))). \end{aligned}$$

We extend now the definition of specialisation of a mv-rule to that of the specialisation of a set of rules concluding the same atom p. In doing so, we select in turn a rule r to specialise. If its specialisation, with respect to a fact f, returns a new rule, that is, $\mathscr{S}_{\mathscr{R}}(r,f) = (r',(\mathit{true},[0,1]))$, then we substitute the rule by the specialised one in the agent's mental state representation, and the truth-value of p is not changed. If the specialisation returns a new interval for p, that is $\mathscr{S}_{\mathscr{R}}(r,f) = (\emptyset,(p,v'))$, the rule is eliminated and a new truth-value for p is calculated by means of the composition inference rule.

Definition 3.4. Let R be a set of mv-rules concluding the same atom p and F a set of mv-atoms, the specialisation of R with respect to F is defined to be $\mathscr{S}_{\mathscr{C}}([0,\mathbf{1}],R,F)$, where

$$\mathscr{S}_{\mathscr{C}} : \mathrm{Int}(A_n) \times 2^{\text{Mv-Rules}} \times 2^{\text{Mv-Atoms}} \to \mathrm{Int}(A_n) \times 2^{\text{Mv-Rules}},$$

$$\mathscr{S}_{\mathscr{C}}(I,R,F) = \begin{cases} \mathscr{S}_{\mathscr{C}}(I,R-\{r\}+\{r'\},F) & (*), \\ \mathscr{S}_{\mathscr{C}}(I\cap v,R-\{r\},F) & (**), \\ (I,R) & \text{otherwise.} \end{cases}$$

(*) if $\exists r \in R, f \in F$ such that $\mathscr{S}_{\mathscr{R}}(r,f) = (r', (\text{true}, [0,1]))$ with $r' \neq r$.
(**) if $\exists r \in R, f \in F$ such that $\mathscr{S}_{\mathscr{R}}(r,f) = (\emptyset, (p,v))$.

Example 3.3. Consider the rules in our running example deducing c

$$\text{AG}(c) = ([0,1], \{(\{a,b\}, c, [\rho_1, 1]), (\{a,f\}, \neg c, [\rho_2, 1])\}).$$

Their specialisation with respect to $F = \{(a, [\rho_5, \rho_5'])\}$ is

$$\begin{aligned} &\mathscr{S}_{\mathscr{C}}([0,1], \{(\{a,b\}, c, [\rho_1, 1]), (\{a,f\}, \neg c, [\rho_2, 1])\}, \{(a, [\rho_5, \rho_5'])\}) \\ &\quad = ([0,1], \{(\{b\}, c, [\rho_6, 1]), (\{f\}, \neg c, [\rho_7, 1])\}) \end{aligned}$$

because the rules' specialisation is:

$$\begin{aligned} &\mathscr{S}_{\mathscr{R}}((\{a,b\}, c, [\rho_1, 1]), (a, [\rho_5, \rho_5'])) \\ &\quad = ((\{b\}, c, [\rho_6, 1]), (\text{true}, [0,1])), \\ &\mathscr{S}_{\mathscr{R}}((\{a,f\}, \neg c, [\rho_2, 1]), (a, [\rho_5, \rho_5'])) \\ &\quad = ((\{f\}, \neg c, [\rho_7, 1]), (\text{true}, [0,1])). \end{aligned}$$

Consider another step of specialisation, now with respect to $(f, [\rho_4, \rho_4'])$

$$\begin{aligned} &\mathscr{S}_{\mathscr{C}}([0,1], \{(\{b\}, c, [\rho_6, 1]), (\{f\}, \neg c, [\rho_7, 1])\}, \{(f, [\rho_4, \rho_4'])\}) \\ &\quad = ([0,1] \cap [0, \rho_8], \{(\{b\}, c, [\rho_6, 1])\}) \end{aligned}$$

because the specialisation of the second rule is

$$\mathscr{S}_{\mathscr{R}}((\{f\}, \neg c, [\rho_3, 1]), (f, [\rho_4, \rho_4'])) = (\emptyset, (c, [0, \rho_8])).$$

Hence, after these two specialisation steps we get a new mental state for the agent with respect to c

$$\text{AG}(c) = ([0, \rho_8], \{(\{b\}, c, [\rho_6, 1])\}).$$

$[0, \rho_8]$ is a provisional value for c because there is still a mv-rule that might conclude that atom, making its truth interval more precise.

The next definition accounts for the specialisation of an agent's mental state with respect to a set of atoms.

Definition 3.5. Let AG be an agent mental state and F a set of mv-atoms. The specialisation of AG with respect to F is defined as:

$$\mathscr{S} : \text{AG} \times 2^{\text{Mv-atoms}} \rightarrow \text{AG}$$

$$\mathscr{S}(\mathrm{AG}, F) = \begin{cases} \mathscr{S}(\mathrm{AG}[f \to (I,R)], F) & (*) \\ \mathrm{AG} & \text{otherwise} \end{cases}$$

(*) if $\exists f \in \Sigma$ such that $S_C(\text{first}(\mathrm{AG}(f)), \text{second}(\mathrm{AG}(f)), F) = (I,R)$ with $(I,R) \neq \mathrm{AG}(f)$

Note **3.1.** The notation $\mathrm{AG}[f \to (I,R)]$ represents a modification of the function AG in such a way that from now on $\mathrm{AG}(f) = (I,R)$.

Example 3.4. We can now see the specialisation of the running example with respect to the atom b.

$$\mathscr{S}(\mathrm{AG}, \{(b, [\rho_{10}, 1])\}) = \mathrm{AG}',$$

where

$$\begin{aligned}
&\mathrm{AG}'(a) = ([\rho_5, \rho_5'], \emptyset),\\
&\mathrm{AG}'(b) = ([\rho_{10}, 1], \emptyset),\\
&\mathrm{AG}'(c) = \mathscr{S}_{\mathscr{C}}([0, \rho_8], \{(\{b\}, c, [\rho_6, 1])\}, (b, [\rho_{10}, 1])) = ([\rho_{11}, \rho_{11}'], \emptyset),\\
&\mathrm{AG}'(d) = ([0, 1], \emptyset),\\
&\mathrm{AG}'(e) = ([0, 1], \{(\{c, d\}, e, [\rho_3, 1])\}),\\
&\mathrm{AG}'(f) = ([\rho_4, \rho_4'], \emptyset).
\end{aligned}$$

Now we have obtained a definitive value for fact c and we can now specialise with respect to c

$$\mathscr{S}(\mathrm{AG}', \{(c, [\rho_{11}, \rho_{11}'])\}) = \mathrm{AG}'',$$

where

$$\begin{aligned}
&\mathrm{AG}''(a) = ([\rho_5, \rho_5'], \emptyset),\\
&\mathrm{AG}''(b) = ([\rho_{10}, 1], \emptyset),\\
&\mathrm{AG}''(c) = ([\rho_{11}, \rho_{11}'], \emptyset),\\
&\mathrm{AG}''(d) = ([0, 1], \emptyset),\\
&\mathrm{AG}''(e) = \mathscr{S}_{\mathscr{C}}([0, 1], \{(\{c, d\}, e, [\rho_3, 1])\}, (c, [\rho_{11}, \rho_{11}'])),\\
&\quad = ([0, 1], \{(\{d\}, e, [\rho_{12}, 1])\}),\\
&\mathrm{AG}''(f) = ([\rho_4, \rho_4'], \emptyset).
\end{aligned}$$

To specialise a complete agent's mental state we will use each atom with definitive value in the mental state in turn to make specialisation steps that possibly will generate definitive values for other atoms to be later on used to specialise more the state. Clearly this process finishes because the number of atoms in any set of rules of the type considered is always finite. Hence the following algorithm.

Algorithm 1. *Specialisation algorithm*

$\mathrm{SC}(\mathrm{AG}) = \mathit{Definite} := \{(f, I_f) \mid \mathrm{AG}(f) = (I_f, \emptyset)\};$
while $\mathit{Definite} \neq \emptyset$ **do**
$(g, I_g) := \mathit{ChooseOne}(\mathit{Definite});$
$\mathit{New}\mathrm{AG} := S(\mathrm{AG}, \{(g, I_g)\});$
$\mathit{Definite} := \mathit{Definite} - (g, I_g) + \{(f, I_f) \mid \mathit{New}\mathrm{AG}(f) = (I_f, \emptyset) \neq \mathrm{AG}(f)\};$
$\mathrm{AG} := \mathit{New}\mathrm{AG};$
endwhile;
return AG

The complexity of this algorithm is $O(n^2)$ where $n = |\Sigma|$.

4. Example

Milord II is a modular language for knowledge engineering based on reflection mechanisms and that implements the specialisation calculus described in this paper. More general descriptions of Milord II may be found elsewhere [14,15]. The purpose of this section is only to show how the specialisation mechanism is actually used in a medical cooperative setting. In real medical environments, problems are usually solved by means of the cooperation of several human agents. The example presented in this section intends to assist physicians to diagnose pneumonia diseases, and consists of two cooperating agents.

In Milord II agents are implemented as autonomous processes in a network. Agents communicate each other by means of message passing in a mail-like system. This example is composed of two agents: the *Clinician* agent and the *Micro-biologist* agent (see Fig. 1) that assist their correspondent human physicians. The *Clinician* agent assists the physician (user of that agent), that has a close contact with the patient, to make a diagnosis of *pneumonia*. The *Clinician* agent uses its own knowledge to get an initial diagnosis of the patient from clinical signs. It also uses the services of the *Micro-biologist* agent to refine this initial diagnosis into a definitive one. The *Micro-biologist* agent provides its own opinion of the diagnosis based on the analysis of a sample (of sputum) of the patient and on the initial diagnosis made by the *Clinician* agent.

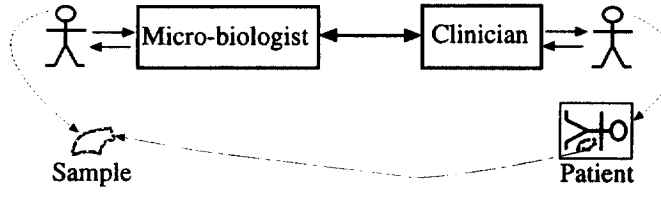

Fig. 1. General schema of the agents.

Let us explain the Milord II code of this example that can be found in Fig. 2 (*Clinician* agent) and Fig. 3 (*Micro-biologist* agent), respectively.

The *Clinician* agent declaration contains: the set of agents this agent can communicate with (acquaintances), in this case just *Micro-biologist*, the *import*

```
Agent Clinician =
    Begin
    Acquaintances Microbiologist
    Import Expectoration, Fever, Cough, Rx_Lung_Infiltrate
    Export Pneumonia
    Deductive knowledge
        Dictionary: not included here
        Rules:
        R001 If Cough and Expectoration and Fever
             then conclude Respiratory_Infection is definite
        R002 If Respiratory_Infection and Rx_Lung_Infiltrate
             then conclude Initial_Diagnosis_Pneumonia is definite
        R003 If Initial_Diagnosis_Pneumonia
             then conclude Pneumonia is possible
        R004 If Microbiologist?Pneumonia
             then conclude Pneumonia is definite
        end deductive
    end
```

Fig. 2. Clinician Agent code.

```
Agent Microbiologist =
    Begin
    Acquaintances Clinician
    Import Sputum_Gram_Positive_Cocci,
        Sputum_Culture_Streptococcus_Pneumonia
    Export Pneumonia
    Deductive knowledge
        Dictionary: not included here
        Rules:
        R001 If Sputum_Gram_Positive_Cocci and
             Sputum_Culture_Streptococcus_Pneumonia
             then conclude Pneumococcus is definite
        R002 If Pneumococcus and
             Clinician?Initial_Diagnosis_Pneumonia
             then conclude Pneumonia is definite
        end deductive
    end
```

Fig. 3. Micro-biologist agent code.

interface, that is, the set of propositions that can be asked to the user of that agent, for instance *Expectoration*; the *export interface*, that is, the set of propositions that can participate in output communication utterances, in this case *pneumonia*; and the *deductive knowledge* containing the *dictionary* with the declaration of the facts of the agent (that is, Σ) and a set of weighted propositional rules (that is, Mv-Rules).

The set of truth-values [2] used in this example is $A_5 = (impossible, slightly\text{-}possible, possible, very\text{-}possible, definite)$ where $impossible = 0$ and $definite = 1$ (see Section 2). We follow in this section a convention used in Milord II: intervals of type $[a, 1]$ are written just as a.

The *Clinician* agent exports the proposition *pneumonia*. This agent tries to deduce this proposition interacting with the known agents (*Micro-biologist*) and its user, and using its own rules. The rules may contain queries to other agents about values for particular propositions belonging to the other agent's language in the form *Agent?Proposition*. For instance, the proposition *Pneumonia* can be deduced by rule R004 from a proposition valued by agent *Micro-biologist*, that is *Micro-biologist?Pneumonia*. Propositions belonging to the import interface (for instance *Expectoration*) are asked to the user of this agent. Given an initial diagnosis of pneumonia (*Initial_Diagnosis_Pneumonia*, *definite*), the rule *R003* can be specialised to deduce the proposition (*Pneumonia*, *possible*). In the case of a definite diagnosis of pneumonia given by the *Micro-biologist* agent, the rule *R004* can be specialised deducing (*Pneumonia*, *definite*). In other words, the agent gives more importance to the micro-biological evidence of pneumonia.

Fig. 3 contains the declaration of the agent *Micro-biologist*. It knows the *Clinician* agent and needs data about the sample of sputum of the patient. To deduce the proposition *Pneumonia*, it previously needs to deduce the presence of *pneumococcus* in the sputum sample of the patient and the presence of *streptococcus pneumonia* in a culture of the sputum (rule *R001*), and it needs to know the initial diagnosis of pneumonia obtained by the *Clinician* agent (rule *R002*). Notice that the *Micro-biologist* agent cannot deduce pneumonia without an initial diagnosis (by the *Clinician* agent).

Making abstraction of the real operational semantics, let us explain the specialisation inference mechanism on this example. Consider that the physician asks for the value of the diagnosis of *Pneumonia* to the *Clinician* agent. To solve this query this agent will then specialise its own rules and will make questions to the other agents and to its user. Consider the following *Clinician* agent initial mental state:

[2] Actually the set of truth-values A_n and the connective T can be defined locally to each agent. In this case we would need to define a mapping between the different logics of the agents that can communicate (see [1] for further details on this topic).

$AG_{Clinician}$

$AG_{Clinician}$(Cough) = ([0, 1], ∅)

$AG_{Clinician}$(Expectoration) = ([0, 1], ∅)

$AG_{Clinician}$(Fever) = ([0, 1], ∅)

$AG_{Clinician}$(Rx_Lung_Infiltrate) = ([0, 1], ∅)

$AG_{Clinician}$(Microbiologist?Pneumonia) = ([0, 1], ∅)

$AG_{Clinician}$(Respiratory_Infection) =
([0, 1], {({Cough, Expectoration, Fever},
Respiratory_Infection,[definite,1])})

$AG_{Clinician}$(Initial_Diagnosis_Pneumonia) =
([0, 1], {({Respiratory_Infection, Rx_Lung_Infiltrate},
Initial_Diagnosis_Pneumonia,[definite,1])})

$AG_{Clinician}$(Pneumonia) =
([0, 1], {({Initial_Diagnosis_Pneumonia}, Pneumonia,[possible,1]),
({Microbiologist?Pneumonia}, Pneumonia,[definite,1])})

To conclude the fact *Pneumonia* the agent needs to conclude an initial diagnosis for *pneumonia* (the proposition *Initial_Diagnosis_Pneumonia*) and to ask the agent *Micro-biologist* for the value of its particular diagnosis of *Pneumonia*. Recursively, to deduce an initial diagnosis for pneumonia, the agent needs to gather all the data relative to the patient (*Cough*, *Expectoration*, *Fever* and *Rx_Lung_Infiltrate*). This gathering has to be made bay the user of the *Clinician* agent.

Consider the case of a patient who has cough, expectoration, fever and infiltration in the lung; the sample of sputum contains gram positive cocci, and the culture of sputum contains streptococcus pneumonia. It can be expressed with the following sentences.

f_1 (Cough, [definite,1])

f_2 (Expectoration, [definite,1])

f_3 (Fever, [definite,1])

f_4 (Rx_Lung_Infiltrate, [definite, 1])

f_5 (Sputum_Gram_Positive_Cocci, [definite, 1])

f_6 (Sputum_Culture_Streptococcus_Pneumonia, [definite, 1])

Consider now that the user of the agent *Clinician* gives the propositions f_1, f_2, and f_3 (cough, expectoration and fever). Then, a first specialisation step will produce the following new AG′.

$$\underline{AG'_{Clinician} = \mathscr{SC}(AG_{Clinician})}$$
$$AG'_{Clinician}(\text{Cough}) = ([\text{definition}, 1], \emptyset)$$
$$AG'_{Clinician}(\text{Expectoration}) = ([\text{definition}, 1], \emptyset)$$
$$AG'_{Clinician}(\text{Fever}) = ([\text{definition}, 1], \emptyset)$$
$$AG'_{Clinician}(\text{Rx\_Lung\_Infiltrate}) = ([0, 1], \emptyset)$$
$$AG'_{Clinician}(\text{Microbiologist?Pneumonia}) = ([0, 1], \emptyset)$$
$$AG'_{Clinician}(\text{Respiratory\_Infection}) = [\text{definite},1], \emptyset$$
$$AG'_{Clinician}(\text{Initial\_Diagnosis\_Pneumonia}) =$$
$$([0, 1], \{(\{\text{Rx\_Lung\_Infiltrate}\},$$
$$\text{Initial\_Diagnosis\_Pneumonia},[\text{definite},1])\})$$
$$AG'_{Clinician}(\text{Pneumonia}) =$$
$$([0, 1], \{(\{\text{Initial\_Diagnosis\_Pneumonia}\}, \text{Pneumonia},[\text{possible},1]),$$
$$(\{\text{Microbiologist?Pneumonia}\}, \text{Pneumonia},[\text{definite},1])\})$$

Notice that the first rule has been totally specialised to get (Respiratory_Infection, [definite, 1]). The truth-value of that proposition corresponds to the successive application of the SIR rule. For instance, we can show a specialisation step of that rule with respect to the proposition *Fever*.

$$(\text{Fever}, [\text{definite},1]), (\text{Cough} \wedge \text{Fever} \rightarrow \text{Respiratory\_Infection}, [\text{definite},1])$$
$$\vdash (\text{Cough} \rightarrow \text{Respiratory\_Infection}, MP^*_T([\text{definite},1], [\text{definite},1]))$$

The question about pneumonia made by the *Clinician* agent to the *Micro-biologist* agent will activate a deductive process in that agent. As showed in $AG_{Micro\text{-}biologist}$, to deduce the fact *pneumococcus* the agent needs to know the initial diagnosis of pneumonia made by the *Clinician* agent and the propositions related with the analysis of sputum.

$$\underline{AG_{Micro\text{-}biologist}}$$
$$AG_{Micro\text{-}biologist}(\text{Sputum\_Gram\_Positive\_Cocci}) = ([0, 1], \emptyset)$$
$$AG_{Micro\text{-}biologist}(\text{Sputum\_Culture\_Streptococcus\_Pneumonia}) = ([0, 1], \emptyset)$$
$$AG_{Micro\text{-}biologist}(\text{Clinician?Initial\_Diagnosis\_Pneumonia}) = ([0, 1], \emptyset)$$
$$AG_{Micro\text{-}biologist}(\text{Pneumococcus}) =$$

$$([0,1],\{(\{\text{Sputum\_Gram\_Positive\_Cocci},\ \text{Sputum\_Culture\_Streptococcus\_Pneumonia}\},\ \text{Pneumococcus},[\text{definite},1])\})$$

$$AG_{\textit{Micro-biologist}}(\text{Pneumonia}) = ([0,1],\{(\{\text{Pneumococcus},\text{Clinician?Initial\_Diagnosis\_Pneumonia}\},\ \text{Pneumonia},[\text{definite},1])\})$$

Notice that (see Fig. 2) the fact *Initial_Diagnosis_Pneumonia* is not exported by the *Clinician* agent. Then it can not be asked to that agent. As we will see this will force the *Clinician* agent to answer with a conditioned answer, that is, a rule.

Suppose the answers to the questions *Sputum_Gram_Positive_Cocci* and *Sputum_Culture_Streptococcus_Pneumonia* are given to the *Micro-biologist* agent by its user. Then, specialising $\mathrm{AG}_{\textit{Micro-biologist}}$ with respect to the sentences corresponding to those propositions we get:

$$\underline{AG'_{\textit{Micro-biologist}} = \mathscr{SC}(AG_{\textit{Micro-biologist}})}$$

$$AG'_{\textit{Micro-biologist}}(\text{Sputum\_Gram\_Positive\_Cocci}) = ([\text{definite},1],\emptyset)$$

$$AG'_{\textit{Micro-biologist}}(\text{Sputum\_Culture\_Streptococcus\_Pneumonia}) = ([\text{definite},1],\emptyset)$$

$$AG'_{\textit{Micro-biologist}}(\text{Clinician?Initial\_Diagnosis\_Pneumonia}) = ([0,1],\emptyset)$$

$$AG'_{\textit{Micro-biologist}}(\text{Pneumococcus}) = (\text{definite},1],\emptyset)$$

$$AG'_{\textit{Micro-biologist}}(\text{Pneumonia}) = ([0,1],\{(\{\text{Clinician?Initial\_Diagnosis\_Pneumonia}\},\ \text{Pneumonia},[\text{definite},1])\})$$

No more specialisation is possible. Then, in this case, the answer to the question *Pneumonia* given by the *Micro-biologist* agent is a specialised rule, for that agent cannot ask the *Clinician* agent a non exportable fact:

$$f_7(\text{Clinician?Initial\_Diagnosis\_Pneumonia} \rightarrow \text{Pneumonia, definite})$$

This rule is then sent back to the *Clinician* agent from the *Micro-biologist* agent, and translated [3]. In this particular case the translation is:

(Initial_Diagnosis_Pneumonia → Microbiologist?Pneumonia, definite)

Now we can see another specialisation step over $AG'_{Clinician}$. The specialisation is done to the translation of f_7 and on f_4. The result is the following:

$$\underline{AG''_{Clinician} = \mathscr{SC}(AG'_{Clinician})}$$
$$AG''_{Clinician}(\text{Cough}) = ([\text{definite}, 1], \emptyset)$$
$$AG''_{Clinician}(\text{Expectoration}) = ([\text{definite}, 1], \emptyset)$$
$$AG''_{Clinician}(\text{Fever}) = ([\text{definite}, 1], \emptyset)$$
$$AG''_{Clinician}(\text{Rx\_Lung\_Infiltrate}) = ([\text{definite}, 1], \emptyset)$$
$$AG''_{Clinician}(\text{Microbiologist?Pneumonia}) = ([\text{definite}, 1], \emptyset)$$
$$AG''_{Clinician}(\text{Respiratory\_Infection}) = ([\text{definite}, 1], \emptyset)$$
$$AG''_{Clinician}(\text{Initial\_Diagnosis\_Pneumonia}) = ([\text{possible}, 1], \emptyset)$$
$$AG''_{Clinician}(\text{Pneumonia}) = ([\text{possible}, 1] \cap [\text{definite}, 1], \emptyset) = [(\text{definite}, 1])$$

AG″ already contains a definite truth-value for the proposition *pneumonia* to give back to the user of the *Clinician* agent who started all the deductive process with the initial query. The answer is then: (*Pneumonia*, *definite*). Notice that there would be no final diagnosis for pneumonia without an initial one, and that without a micro-biological diagnosis the final diagnosis would have had as maximum truth-value *possible*.

5. Discussion

In this paper an inference calculus containing a SIR in the paradigm of multiple-valued logics is presented. The calculus is implemented using techniques of partial deduction, and is shown to be sound and complete for atom deduction.

The communication between autonomous agents based on this calculus is much more cooperative than the classical one: The answer to a query is now a set of specialised rules and propositions. Our system is thought for the coop-

[3] $Tr^{\text{agent}_2}_{\text{agent}_1}$ is a function that given a set of sentences in the language of agent_1 translates each sentence to the language of agent_2. It usually obliges to change the agent names preceeding propositions (for instance $Tr^{B}_{A}(\{(B?b, \beta)\}) = \{(b, \beta)\}$). It also changes the truth-values of the sentences to adapt to the logic of agent_2. The detailed explanation of this function is out of the scope of this paper.

eration among agents via the communication of knowledge, not just data, in a similar way to other systems [2], where the communication is about lambda-formulas; or the communication of inductive inferences as in [3], a work on multi-agent learning systems.

The specialisation calculus is also related to other work on *conditioned answers* [4,16,19] and on the treatment of *unknown information* [21]. It allows us to obtain conditioned answers after the specialisation of a rule base with the known information. Our system is able to give back useful answers even in the case of partially known information.

The main difference of specialisation calculus with respect to other uses of partial deduction, is that it is based on a multi-valued propositional language and it is oriented to the improvement of the communication among agents, not just efficiency.

This specialisation calculus can also be used to make validation of rule bases. Consider that a physician has a general rule base for pneumonia treatment, and that he wants to check it in a restricted context such as: 'women with gram-negative rods'. The specialisation mechanism allows him to obtain a new rule base specialised for pneumonia treatment in the particular case of *women* with *gramnegative rods*. The expert should agree with the behaviour of the new rule base so obtained, in that restricted context, because it is a specialisation of its original one, otherwise he must revise it. To check the behaviour of this reduced rule base he can apply any classical method (v.g. by case analysis), but to a much more reduced one, and this is the advantage of the use of the specialisation calculus. This specialisation mechanism can also be understood as a way of modularisation, by contexts, of flat and non-structured rule bases. This methodology gives then a more comprehensive and systematic way of validating rule bases than the standard methods.

Acknowledgements

Thanks to Pere Garcia for helpful discussions on specialisation rule. We also thanks Lluís Murgui for advising in the medical example. This research has been supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICYT) through the project SMASH (TIC96-1038). The final version of the paper has been produced while Carles Sierra was on sabbatical leave at the Queen Mary and Westfield College, University of London, thanks to the Spanish Ministry of Education grant PR95-313.

References

[1] J. Agustí, F. Esteva, P. Garcia, L. Godo, Ramón López de Mántaras, J. Puyol, C. Sierra, L. Murgui, Structured local fuzzy logics in Milord, in: Fuzzy Logic for the Management of Uncertainty, Wiley, New York, 1992, pp. 523–551.

[2] Claire Beyssade, Patrice Enjalbert, Claire Lefèvre, Cooperating logical agents, in: Michael Wooldridge, Jörg P. Müller, Milind Tambe (Eds.), Intelligent Agents II, vol. 1037 of Lecture Notes in Artificial Intelligence, Springer, Berlin, 1996, pp. 299–314.

[3] Winton Davies, Peter Edwards, The communication of inductive inferences, in: Gerhard Weiß (Ed.), Distributed Artificial Intelligence Meets Machine Learning, vol. 1221 of Lecture Notes in Artificial Intelligence, Springer, Berlin, 1997, pp. 223–241.

[4] Robert Demolombe, Strategies for the computation of conditional answers, in: Proceedings of the Workshop on Partial Deduction, Partial Evaluation and Intelligent Reasoning, ECAI'90, 1990, pp 5–23.

[5] J. Gallagher, Transforming logic programming by specialising interpreters, in: Proceedings ECAI'86, 1986, pp. 109–122.

[6] S. Gottwald, Mehrwertige Logik, Akademie-Verlag, Berlin, 1988.

[7] S. Gottwald, Fuzzy Sets and Fuzzy Logic, Vieweg, Braunschweig, 1993.

[8] P. Hájek. Metamathematics of fuzzy logic (book in preparation).

[9] P. Hájek, Fuzzy logic from the logical point of view, in: M. Bartošek, J. Staudek, J. Wiedermann (Eds.), SOFSEM'95: Theory and practice of informatics, vol. 1012 of Lecture Notes in Computer Science, Springer, Milovy, Czech Republic, 1995, pp. 31–49.

[10] S.C. Kleene, Introduction to Metamathematics, Van Nostrand, Princeton, NJ, 1952.

[11] H.J. Komorowski, A specification of an abstract Prolog machine and its application to partial evaluation, Ph.D. Thesis, Linkoping University, 1981.

[12] H.J. Komorowski, Towards a programming methodology founded on partial deduction, in: Proceedings of the ECAI'90, 1990, pp. 404–409.

[13] J.W. Lloyd, J.C. Shepherson, Partial evaluation in logic programming, The Journal of Logic Programming 11 (3/4) (1991) 217–242.

[14] J. Puyol, L. Godo, C. Sierra, A specialisation calculus to improve expert system communication, in: Proceedings of the ECAI'92, cls1992, pp. 144–148.

[15] Josep Puyol-Gruart, MILORD II: A Language for Knowledge-Based Systems, vol. 1 of Monografies del IIIA, IIIA-CSIC, 1996.

[16] Chiachi Sakama, Hidenori Itoh, Partial evaluation of queries in deductive databases, Technical Report TR-302, ICOT, 1986.

[17] Yoav Shoham, Agent-oriented programming, Artificial Intelligence 60 (1993) 51–92.

[18] A. Takeuchi, K. Furukawa, Partial evaluation of prolog programs and its application to meta programming, in: Information Processing 86 (1986).

[19] Phil Vasey, Qualified answers and their application to transformation, in: G. Goos, J. Hartmanis (Eds.), Third International Conference in Logic Programming, LNCS 225, Springer, Berlin, 1986, pp. 425–432.

[20] R. Venken, A prolog meta-interpreter for partial evaluation and its application to source transformation and query-optimisation, in: Proceedings of the ECAI'84, 1984, pp. 91–100.

[21] Dave Wolstenholme, Saying I don't know and conditional answers, in: D.S. Moralle (Ed.), Research and Development in Expert Systems IV, Cambridge University Press, Cambridge, 1987, pp. 115–125.