A Specialisation Calculus to improve Expert Systems Communication *

Josep Puyol-Gruart, Lluís Godo, Carles Sierra
Institut d'Investigació en Intel·ligència Artificial(IIIA)
Centre d'Estudis Avançats de Blanes (CSIC)
Camí de Santa Bàrbara, 17300 Blanes (Girona), Spain

Abstract. The motivation of this work is the improvement of the classical input/output expert systems behaviour. In an uncertain reasoning context this behaviour consists of just getting certainty values for propositions. Instead, the answer of an expert system will be a set of formulas: a set of propositions and a set of specialised rules containing unknown propositions in their left part. This type of behaviour is much more informative than the classical one because gives to users not only the answer to a query but all the relevant information to improve the solution. A family of propositional rule-based languages founded on multiple-valued logics is presented and formalised. The deductive system defined on top of it is based on a *Specialisation Inference Rule* (SIR): $(A_1 \wedge A_2 ... \wedge A_n \rightarrow P, V), (A_1, V') \vdash (A_2 \wedge \ldots \wedge A_n \rightarrow P, V'')$, where V, V' and V'' are uncertainty intervals. This inference rule provides a way of obtaining rules containing unknown conditions in their premise as the result of the deductive process. The soundness and literal completeness of the deductive system are proved. The implementation of this deductive calculus is based on techniques of partial evaluation. Moreover, the specialisation mechanism provides an interesting way of validating knowledge bases. Keywords: Partial Evaluation, Expert Systems, Multiple-valued Logic.

1 Introduction and Motivation

Looking at an Expert System (ES) as a *blackbox*, the standard behaviour we can observe is as follows. The user queries to the system whether a given proposition can be deduced. If the system is able to deduce the proposition, its certainty value is given back. Otherwise the answer is *unknown* (open world assumption).

This behaviour is rather poor because the system usually has much more information that could be useful to the user, for instance:

1. When the system is able to answer the user's query, the user might also be interested in knowing other deductive paths that would be useful to improve the solution, or to know other conclusions that are deducible from the proposition answered.

2. When the system is not able to answer a query, it gives back the value *unknown* maybe because the user did not provided enough information to the system. Thus, the communication will be much more informative if the system is able to answer, not *unknown*, but with the information the user should know to come up with a value for the query.

All this hidden information can be used to better modelise communication among human experts. Looking carefully at how experts communicate their knowledge and at their problem solving procedures, we can find complex communication patterns. Sometimes experts cannot reduce their interaction only to the communication of certainty values for propositions. For instance, in medical diagnosis, when experts communicate, they also need:

1. **To condition their decisions**. Suppose that it is not known whether a patient is allergic to penicillin. An expert considering the possibility of giving penicillin as treatment would say: *Penicillin is a good treatment from a clinical point of view provided that the patient has no allergy to penicillin*.

2. **To give suggestions that must be considered with solutions**. Experts usually give other suggestions (*antibiogram*) that are related to the solution (*pneumococcus*). For instance the expert might say: *Pneumococcus has been isolated in the culture of sputum. In this case it is strongly suggested to make an antibiogram to the patient*.

*This research has been supported by the CICYT TIC91- 0430 project TESEU and the Esprit Basic Research Action number 3085 (DRUMS).

3. **To give conditioned suggestions to be considered together with decisions**. Another example of complex communication is the combination of the above two communication patterns: *Ciprofloxacine is a good treatment, but if the patient is a woman on breast-feeding period she must stop breast-feeding*.

To model such communication protocols, we need to extend the ES answering procedure, by allowing to answer queries with sets of formulas (rules and propositions). We propose to do it by means of an Specialization Calculus of KBs.

Specialisation is based on the notion of partial evaluation expressed in the well known Kleene's Theorem. Partial evaluation algorithms have been intensively in logic programming [9] [2] [4] [8] [5] mainly for efficiency purposes. In this paper we propose the use of this technique to improve the communication behavior of ESs. With this purpose in section 2 we propose a partial evaluation mechanism for rule bases. In section 3 we formalise an Specialisation Calculus. Finally a little example and conclusions are presented in sections 4 and 5 respectively.

2 Proposal: Partial Evaluation in Rule Bases with Uncertainty

In rule bases, deduction is mainly based on the modus ponens inference rule:

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

This inference rule is only applicable when every condition of the premise is satisfied, otherwise nothing can be inferred. We will use partial evaluation to extract the maximum information from incomplete knowledge.

We base the partial evaluation in a rule base context on the well known logical equivalence $(A \wedge B) \rightarrow C \equiv A \rightarrow (B \rightarrow C)$ which leads to the following boolean specialisation inference rule:

$$A, A \wedge B \rightarrow C \vdash B \rightarrow C$$

The rule $B \rightarrow C$ is called the *specialisation* of $A \wedge B \rightarrow C$ with respect to the proposition A. Notice that in the particular case of $B = \emptyset$, we recover the usual modus ponens rule.

In a more formal way we give the following definitions.

Definition 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:*

$$\mathcal{S}_{\mathcal{R}} : R \times P \rightarrow R \times P$$

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

Definition 2 (KB Specialisation) *Let KB be a set of knowledge bases. We note KBs as pairs $kb = (R_{kb}, P_{kb})$ where R_{kb} is a set of rules and P_{kb} is a set of propositions. KB specialisation is defined as a function:*

$$\mathcal{S}_{\mathcal{KB}} : KB \rightarrow KB$$

$$\mathcal{S}_{\mathcal{KB}}(kb) = \begin{cases} \mathcal{S}_{\mathcal{KB}}((R_{kb} - \{r\} + \{r'\}, P_{kb} + \{p'\})), (*) \\ kb \text{ , otherwise} \end{cases}$$

(*) if $P_{kb} \neq \emptyset$ and $\exists p \in P_{kb}$ and $\exists r \in R_{kb}$ such that $\mathcal{S}_{\mathcal{R}}(r,p) = (r', p')$ and $r' \neq r$

In other words, the specialisation of a *kb* consists on the exhaustive specialisation of its rules. Rules whose conditions contain propositions with known values are replaced by their specialisations. Rules that only have one condition will be eliminated and a new proposition will be added. This new proposition will be used again to specialise the *kb*. The process will finish when the *kb* has no rule containing on its conditions a known proposition. This approach is different for instance from the logic programming one used in [5]. There, partial evaluation is goal driven, whereas here partial evaluation is data driven.

In an uncertain reasoning context we propose to extend the above boolean specialisation inference rule as follows:

Definition 3 (SIR) *Given a proposition A with certainty value α, and a rule with certainty value ρ, then*

$$(A, \alpha), (A \wedge B \rightarrow C, \rho) \vdash (B \rightarrow C, \rho')$$

where $\rho' = MP^1(\alpha, \rho)$ is the new value of the rule.

Therefore we need to extend the previous definition of the function $\mathcal{S}_{\mathcal{R}}$ to allow the handling of certainty values.

Definition 4 (Specialisation of Uncertain Rules) *Let R^* be a set of weighted rules and P^* a set of weighted literals. We note weighted rules as pairs,*

[1] SIR is parametric on the uncertainty propagation function MP (modus ponens), particular for each uncertainty calculus.

$r^* = (r, u_r)$ where r is a classical rule and u_r is the certainty value of r. And we note weighted literals as pairs, $p^* = (p, u_p)$ where p is a classical literal and u_p is the certainty value of p.

$$\mathcal{S}_{\mathcal{R}} : R^* \times P^* \rightarrow R^* \times P^*$$

$$\mathcal{S}_{\mathcal{R}}(r^*, p^*) = \begin{cases} (r^*, \emptyset) & \text{if } p \notin m_r \\ (\emptyset, p^{*'}) & \text{if } m_r = \{p\} \\ (r^{*'}, \emptyset) & \text{otherwise} \end{cases}$$

where $r^{*'} = (m_r - \{p\}, c_r, MP(u_p, u_r))$ and $, p^{*'} = (c_r, MP(u_p, u_r))$.

It is easy to extend (not included here) the classical KB specialisation to an uncertain KB specialisation.

Now, the answer to a query can be considered as a specialised *kb*: The specialised *kb* obtained from $kb = (R^*, P^*)$ where R^* is the set of rules in deductive paths to and from the query. And P^* is the set of propositions defining a case.

3 Formalisation of a Specialisation Calculus for Rule Bases

In this section we present the definition of a family of multiple-valued logics with a deductive system based on an specialisation inference rule. Some aspects of these logics have been already described in [1]. Each logic is determined by a particular algebra of truth-values from a parametric family that is described next.

An **algebra of truth-values** is a finite algebra $\mathcal{A}_T^n =< A_n, N_n, T, I_T >$ such that:

- The set of truth-values A_n is a chain:

  $$0 = a_1 < a_2 < \cdots < a_n = 1$$

  where 0 and 1 are the booleans False and True respectively.

- The negation operator N_n is an unary operation defined as $N_n(a_i) = a_{n-i+1}$, the only one that fulfills the following properties:

  N1: if $a < b$ then $N_n(a) > N_n(b)$, $\forall a, b \in A_n$

  N2: $N_n^2 = Id$.

- The conjunction operation T is a binary operation satisfying $\forall a, b, c \in A_n$:

  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 $a \leq b$ then $T(a, c) \leq T(b, c)$ for all c

- The implication operator I_T is defined by residuation with respect to T, i.e. $I_T(a, b) = \text{Max}\ \{c \in A_n | T(a, c) \leq b\}$, and satisfies the following properties:

  I1: $I_T(a, b) = 1$ if, and only if, $a \leq b$.

  I2: $I_T(1, a) = a$

  I3: $I_T(a, I_T(b, c)) = I_T(b, I_T(a, c))$

  I4: If $a \leq b$, then $I_T(a, c) \geq I_T(b, c)$ and $I_T(c, a) \leq I_T(c, b)$

  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 determined. So, varying this two parameters we obtain a family of multiple-valued logics, including, among others, Kleene's and Lukasiewicz's logics.

In the following description of the language, the semantics and the deduction system (specialisation calculus) of a particular logic, we suposse fixed an algebra A_T^n. This calculus is proved to be sound and also complete if constrained to the case of literals [3].

3.1 Syntax

A propositional language $\mathcal{L}_n = (A_n, \Sigma, \mathcal{C}, \mathcal{S}_n)$ is defined by:

- A signature Σ consisting on a set of atomic symbols plus *true* and *false*.

- A set of Connectives: $\mathcal{C} = \{\neg, \wedge, \rightarrow\}$

- A set of Sentences: $\mathcal{S}_n$ = W-Literals $\cup$ W-Rules

  Sentences are pairs of classical-like propositional sentences and intervals of truth-values. The classical-like propositional sentences are restricted to be literals or rules. Thus, the sentences of the language are of the following types:

  W-Literals: $\{(p, V) \mid p$ is a literal and V is an interval of truth-values of $A_n\}$

  W-Rules: $\{(p_1 \wedge p_2 \wedge \cdots \wedge p_n \rightarrow q, V) \mid p_i$ and q are literals, V is an interval of truth-values of A_n, and $\forall i, j (p_i \neq p_j, p_i \neq \neg p_j, q \neq p_j)$ and $V = [a, 1]$ where $a > 0\}$

3.2 Semantics

- Models M_ρ are defined by valuations ρ, i.e. mappings from the firsts components of sentences to A_n such that:

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

  $\rho(p_1 \wedge p_2) = T(\rho(p_1), \rho(p_2))$

  $\rho(p \rightarrow q) = I_T(\rho(p), \rho(q))$

  $\rho(true) = 1$

  $\rho(false) = 0$

- The Satisfaction Relation between models and sentences is defined by:

  $$M_\rho \models (p, V) \text{ iff } \rho(p) \in V$$

  It is easy to check that the following properties hold for the corresponding semantical entailment.

  $SR1 : (p, V) \models (\neg p, W) \Leftrightarrow N_n^*(V) \subseteq W$

  $SR2 : (p, V_1), (p, V_2) \models (p, W) \Leftrightarrow V_1 \cap V_2 \subseteq W$

  $SR3 : (p_i, V_i), (p_1 \wedge \cdots \wedge p_n \rightarrow q, V) \models (p_1 \wedge \cdots \wedge p_{i-1} \wedge p_{i+1} \wedge \cdots \wedge p_n \rightarrow q, W) \Leftrightarrow MP_T^*(V_i, V) \subseteq W$

  where N_n^* and MP_T^* are the point-wise extensions [2] of N_n and MP_T respectively. MP_T is a function from A_n to the set of intervals of A_n defined as:

  $$MP_T(a, b) = \begin{cases} \emptyset & \text{if } a \text{ and } b \text{ are inconsistent } [3] \\ [a, 1] & \text{if } b = 1 \\ T(a, b) & \text{otherwise} \end{cases}$$

  This is a functional expression of the multiple-valued version of the classical modus ponens rule, i.e. $MP_T(a, b)$ is the set of solutions for $\rho(q)$ in the equation system: $\{\rho(p) = a; \rho(p \rightarrow q) = b\}$.

3.3 Specialisation Calculus

The specialisation calculus is based on:

1. The following axioms:

   AS1: $(\neg\neg p \rightarrow p, [1, 1])$

   AS2: $(p, [0, 1])$

   A1: $(true, [1, 1])$

   A2: $(false, [0, 0])$

2. The following inference rules:

   Weakening: $(p, V_1) \vdash (p, V_2)$ where $V_1 \subseteq V_2$

   Not-introduction: $(p, V) \vdash (\neg p, N_n^*(V))$

   Composition: $(p, V_1), (p, V_2) \vdash (p, V_1 \cap V_2)$

   SIR: $(p_i, V_i), (p_1 \wedge \cdots \wedge p_i \wedge \cdots \wedge pn \rightarrow q, V_r) \vdash (p_1 \wedge \cdots \wedge p_{i-1} \wedge p_{i+1} \wedge \cdots \wedge p_n \rightarrow q, MP_T^*(V_i, V_r))$

From properties SR1, SR2 and SR3 of the semantical entailment, it is easy to check that this deductive system is sound.

Theorem 1 (Soundness) *Let A be a sentence and Γ a set of sentences. Then $\Gamma \vdash A$ implies $\Gamma \models A$*

On the other hand, it is straightforward to see that our deductive system is not complete. For instance, we have $\{(p \rightarrow q, 1), (q \rightarrow r, 1)\} \models (p \rightarrow r, 1)$ but $\{(p \rightarrow q, 1), (q \rightarrow r, 1)\} \not\vdash (p \rightarrow r, 1)$. However, it can be proved that the system is complete for literal deduction, i.e., any literal that is satisfactible from a set of formulas is deducible in our deductive system.

Theorem 2 (Literal Completeness) *Let Γ be a set of sentences and (p, V) a literal. Then $\Gamma \models (p, V)$ implies $\Gamma \vdash (p, V)$.*

4 Example

Milord II is a modular language for knowledge engineering that manages uncertainty and reflection. It includes an inference engine that implements the specialisation calculus described in this paper [7] [6]. In this section an example will be presented. This example is part of a real application for pneumonia treatment written in Milord II, named Terap-IA. When writting the example we will use some extensions of the language described in section 3.

The set of truth-values used is A_n =*(impossible, slightly-possible, possible, very-possible, definite)* where *impossible* $= 0$ and $definite = 1$.

Consider the following rules for pneumonia treatment [4]:

R0 (H-Influenzae $\rightarrow$ Quinolones, *possible*)
R1 (female $\wedge$ young $\wedge$ pregnant $\wedge$ Legionella-sp $\rightarrow$ Co-trimoxazole, *slightly-possible*)
R2 (female $\wedge$ young $\wedge$ breast-feeding $\wedge$ $\geq$(Quinolones, *possible*)[5] $\rightarrow$ stop-breast-feeding, *definite*)
R3 (breast-feeding $\wedge$ Co-trimoxazole $\rightarrow$ stop-breast-feeding, *definite*)

[2] Actually, MP_T^* has to be defined to give the minimal interval containing the point-wise extension.

[3] a and b are inconsistent if there exists no c such that $I(a, c) = b$.

[4] In this rules *H-Influenzae* and *Legionella-sp* are possible diagnosis, and *Quinolones* and *Co-trimoxazole* are antibiotics. Also, we have simplified the intervals syntax, as it is done in Milord II. Intervals of the type $[a, 1]$ appearing as values of rules and propositions are written just as a.

Consider the case of a young female patient with a diagnosis of *H-Influenzae*. The propositions representing this case are:

(H-Influenzae, *very-possible*)
(female, *definite*)
(young, *definite*)

If we specialise the *kb* composed by the rules and the propositions above presented, it is easy to see that the final set of rules obtained is:

R1' (pregnant ∧ Legionella-sp → Co-trimoxazole, *slightly-possible*)
R2' (breast-feeding → stop-breast-feeding, *definite*)
R3 (breast-feeding ∧ Co-trimoxazole → stop-breast-feeding, *definite*)

and the the final set of propositions is.

(H-Influenzae, *definite*)
(female, *definite*)
(young, *definite*)
(Quinolones, *possible*)

Then, we can interpret this result as a new *kb* specialised for a particular patient. On the other hand, for the same example of specialisation we can see an example of communication. Suppose that the user queries the system for a certainty value for *Quinolones*.

Then, the system shows the propositions and rules related to the query *Quinolones*. That is,

(Quinolones, *possible*)
R1" (breast-feeding → stop-breast-feeding, *definite*)

In natural language the answer would be: *For the case of a H. Influenzae diagnosis for a young female, quinolones is possible, and if she is on breast-feeding period, she has to stop breast-feeding.*

5 Discussion

In this paper a new communication protocol for ES's is presented. It is based on an inference calculus containing an Specialisation Inference Rule in the paradigm of multiple-valued logics. This specialisation calculus is implemented using techniques of partial evaluation, and it is shown to be sound and complete for literals.

[5]"$\geq$" is a boolean metapredicate of Milord II such that "$\geq (p, a)$" is true if and only if $(p, [b, c])$ has been deduced with $b \geq a$.

The communication so obtained is much more cooperative with users than the classical one: The answer to a query is a set of specialised rules and propositions.

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

References

[1] J. Agusti, J. Esteva, P. Garcia, L. Godo, and C. Sierra. Combining multiple-valued logics in modular expert systems. In *Proceedings 7th Conference on Uncertainty in AI*, 1991.

[2] J. Gallagher. Transforming logic programming by specialising interpreters. In *Proceedings ECAI'86*, pages 109–122, 1986.

[3] L. Godo, J. Puyol, and C. Sierra. A specialisation calculus to improve expert system communication(long paper). Technical Report 92/8, Institut d'Investigació en Intel.ligència Artificial(IIIA), 1992.

[4] H. J. Komorowski. *A specification of an abstract Prolog machine and its application to partial evaluation.* PhD thesis, Linkoping University, 1981.

[5] J. W. Lloyd and J. C. Shepherson. Partial evaluation in logic programming. *Logic Programming*, 11(3/4):217–242, October/November 1991.

[6] J. Puyol, C. Sierra, and J. Agusti. Partial evaluation in MILORD II: A language for knowledge engineering. In *Proceedings Europ-IA'91*, pages 193–207, 1991.

[7] C. Sierra and J. Agusti. Colapses: Towards a methodology and a language for knowledge engineering. In *Proceedings AVIGNON'91*, pages 407–423, 1991.

[8] A. Takeuchi and K. Furukawa. Partial evaluation of prolog programs and its application to meta programming. In *Information Processing 86*, 1986.

[9] R. Venken. A prolog meta-interpreter for partial evaluation and its application to source transformation and query-optimisation. In *Proceedings ECAI'84*, pages 91–100, 1984.