A Constructivist Approach to Rule Bases

Giovanni Sileno, Alexander Boer and Tom Van Engers
Leibniz Center for Law, University of Amsterdam, Amsterdam, The Netherlands

Keywords: Rule Bases, Constructivism, Rules Revision, Rules Optimization, Adaptation, Regulated Systems.

Abstract: The paper presents a set of algorithms for the conversion of rule bases between priority-based and constraint-based representations. Inspired by research in precedential reasoning in law, such algorithms can be used for the analysis of a rule base, and for the study of the impact of the introduction of new rules. In addition, the paper explores an optimization mechanism, built upon assumptions about the world in which the rule-based system operates, providing a model of environmental adaptation. The investigation is relevant to practical reasoning, agent modeling and agent programming.

1 INTRODUCTION

Practical reasoning in everyday life is often not based on repeated deliberation, but on behavioural scripts constructed and refined by some adaptation process. Let us consider this simple story: *Raphael, who lives in a rainy country, always checks before he goes to work whether it will rain, in which case he will bring his umbrella. Samuel, who lives in a sunny country, usually goes out without checking the weather. Both of them believe in taking an umbrella if it rains, however.* Intuitively, if Raphael or Samuel moved from one country to the other, we would expect that they would change their habits eventually.

According to traditional optimization theory, adaptation comes from the agent's efforts to obtain a better overall pay-off. In contrast, Heiner's theory of predictable behaviour (Heiner, 1983) explains how behavioural regularities arise in the presence of unresolvable *uncertainty* about the "right" course of action to follow: being predictable seems to pay off by itself.

Regardless of its explanation, we acknowledge the existence of a sort of structuring process in the agent, which results in behavioural rules or scripts that are followed without consciously *reflecting* on them. The foundations of agent modeling, agent programming, and before those, of expert systems, are for the most based on such behavioural rules or scripts.

A similar analysis can be performed on collective agencies, whose behaviour is described/prescribed via formalized artifacts and procedures. Focusing on administrative organizations, Boer and Van Engers recognize in (Boer and Engers, 2013) three spheres of activity: *operations, planning/design, policy making*. The three spheres are concerned by different activities and often utilize different resources. For instance, people at the operations provide services depending on the procedures and the resources assigned by the design department.

For simplicity reasons, this paper divides the agency/system in only two levels: the *regulated* (non-reflective) component and the *regulatory* (reflective) component. In contrast to approaches like machine learning, theory induction, etc. our agent is not constructing the rules that govern his behaviour from the facts he observes, but he follows the prescription a given set of rules (cf. agent programming), possibly making its operationalization more efficient.

Two research questions are mainly addressed. First, *how to revise (the operational knowledge of) a rule base if a new rule is introduced?* This problem is common in practical reasoning, statutory law and case law. Horty recently explored in (Horty, 2011) the dynamics of normative systems based on common law, examining phenomena like *distinguishing* on binding precedent rules. Inspired by his work, we generalize the analysis of revision in law to a generic intelligent system whose operational knowledge is described by a rule base. Focusing on two ways to represent a system of rules, *priority-based* and *constraint-based* (§ 2), we identify and organize the algorithms to pass from one representation to the other (§ 3).

Then, analyzing the interactions of the evaluation of rules with default assumptions, we extend the analysis to consider the rewriting of the rule base due to new informational commitments, in order to tackle down the second question: *how to refine rules when certain expectations are ascribed to the environment?* (§ 4) As the change of informational commitment and the beginning of the rule revision process are only asynchronously linked, the coupling is worth further analysis (§ 5).

Both the rules that govern the practical behaviour of the agent and the default assumptions about the world may change. For these reasons, our work is affine to the spirit of defeasible logic. Rather than focusing on a formal framework following this tradition, however, we prefer at this stage to propose a direct computational implementation.

2 REPRESENTATION TYPES

We summarize some practical definitions that will be used throughout the paper.

- A *fact* f is a true proposition.
- A *rule* r is a conditional statement relating a *premise* with a *conclusion*. Premise and conclusion are propositions, or expressions of propositions. They are called also *body* and *head*.
- Given a rule r, $Premise(r)$ is a function returning the set of atomic propositions that should be true when its *premise* is true. These components are also called *factors*. $Conclusion(r)$ is a function returning the atomic propositions asserted by the rule when the premise is true.
- A *situation* Σ is a set of facts: $\Sigma = \{f_1, f_2, \ldots, f_n\}$.
- The *applicability* condition is expressed by $\Sigma \models Premise(r)$.
- A *rule base* is a set of rules: $\Delta = \{r_1, r_2, \ldots, r_n\}$, possibly partially ordered.

Evidently, different rules may share part of the premises, or stated equivalently, share part of their domain of applicability. In general, a set of facts can apply on several rules. Furthermore, when attacking each other conclusions, rules are potentially *in conflict*. When conflicting rules apply, we have the problem of determining which is the right conclusion. Two solutions are possible:

- at *horizontal level*, modifying adequately the content of rules, i.e. redefining the constraints of applicability of mutually conflicting rules;
- at *vertical level*, defining relations of hierarchy or *priority* between rules.

Two representational attitudes are directly related to those approaches: *constraint-based* and *priority-based*.

2.1 Constraint-based

A rule base written following a strict constraint-based approach is a non-ordered set. The relative position of rules (e.g. the position in the artifact describing the regulatory system, the chronological time of creation, etc.) is not relevant. A rule is required to transport all what is needed to check its applicability, i.e. all the relevant *factors* have to be made explicit.

On the down side, however, the complete set of rules has to be (maintained) consistent as a whole. The problem can be explained referring to the programming construct *switch...case*, which separates execution paths depending on a certain condition:

```
switch (condition) {
  case value : [...]; break;
  [...]
}
```

In order to be functional, the *condition* variable has to index precisely one value position in the list.

As the application of a rule is triggered by the relevant factors in the premise being true, the most basic encoding is built upon the permutations of all factors described in the rule base. If the rule base is sensitive to N factors, there are 2^N situations to be encoded. Such *full-tabular* encoding implies that the requests for a rule are inflected also in terms of factors which are irrelevant for the rule in itself, but are relevant to other rules. Other more efficient encoding are possible.

2.2 Priority-based

In all acts of communication there is an intrinsic strictly ordered dimension, usually referred to as the *discourse*. It can be temporal, as in the case of interactions; or positional, as with texts.

Priority-based representations take advantage of the meta-information intrinsic to the composition of content to reduce its redundancy. In programming, something similar occurs with the construct *if...else*:

```
if (condition1) [...];
else if (condition2) [...];
else [...];
```

In order to avoid to repeat the same calculations, a good programmer implements the conditions to be checked in a convenient way, conscious of the sequential processing of the code. For instance, he knows that if *condition2* is being evaluated, *condition1* is necessarily false.

Considering such structured enchaining in the evaluation of rules, each new evaluation could remove the conjunction of factors which have been already computed. The resulting rewriting of the rules is more efficient (in size terms) than the original version. Conversely, the application of a rule may require more computational effort than a constraint-based representation, as the applicability must be strictly checked following the given ordering.

So far, we have considered priority as directly related to ordering (positional or chronological). In such conditions, if we introduce a new rule with a certain domain of application, we overwrite all existing rules whose domains subsume that of the new rule. There are situations in which it is useful to avoid this mechanism. This occurs for instance in law with the principle called *lex specialis derogat legi generali*: a more specific law overrules the more general. Principles like this create *meta-rules*: their conclusion is not a fact, but a *priority constraint*. In the following we suppose that we are always able to apply meta-rules to obtain the relative priority between rules.[1]

3 INCREMENTAL DYNAMICS

Depending on the underlying representation, interventions on a rule base require a different attention:

- in a *constraint-based* rule-base, we need some kind of rule revision process (Horty calls it *refinement*) on the rules whose applicability is affected by the addition, so that after the update each rule is again consistent with the whole, and can be also read separately;
- in a *priority-based* rule-base, the addition of rules or modification of priorities is straightforward; however, we cannot infer the resulting practical consequences without taking into account the whole rule-base.

Our objective is therefore to provide a conversion method from one representation to the other, in order to cope with the respective limitations. While the natural way of handling the incremental construction of a rule-base is via priority-based representations, the more direct way to access the actual applicability of a rule comes from a constraint-based representation.

From Compound to Simple Rules. In the following, we will often refer to *simple rules*, i.e. rules with no disjunction in the body, nor conjunction in the head. They can be seen as the atomic components of the conditional. Compound rules can be reduced to *simple rules* applying the following properties:

$$(a \to c \land d) \Leftrightarrow (a \to c) \land (a \to d) \tag{1}$$

$$(a \lor b \to c) \Leftrightarrow (a \to c) \land (b \to c) \tag{2}$$

Relevance. Given two simple rules r_1 and r_2, we say that r_1 is *relevant* in respect to r_2, if the propositional content of their conclusions are positively or negatively aligned: $Conclusion(r_1) = Conclusion(r_2)$ or $Conclusion(r_1) = \neg Conclusion(r_2)$.[2]

Illustrative Example. Let us assume the existence of a principle like *lex posterior derogat legi prior*, i.e. the last rule has higher priority than the previous ones. Consider a rule base consisting of just one rule $r_1 : A \to P$. With one factor (A), we can have two situations, but our rule refers only to one of them. The associated representations are:[3]

Constraint-Based	Priority-Based
$A \to P$	$A \to P$
$[\neg A \to ?]$	

At this point, we add a second rule: $r_2 : B \to \neg P$. Considering two factors (A, B), we have four possible situations. In a constraint-based representation, we have to refine the previous rule in order not to lose consistency, as r_1 and r_2 have opposite conclusions, and therefore may be in conflict. Removing the domain of applicability of r_2 from the domain of r_1 we obtain the following representations:

CB (full-tabular)	CB (intermediate)	PB
$A \land \neg B \to P$	$A \land \neg B \to P$	$A \to P$
$A \land B \to \neg P$	$B \to \neg P$	$B \to \neg P$
$\neg A \land B \to \neg P$		
$[\neg A \land \neg B \to ?]$		

The table distinguishes two types of constraint-based representations: *full-tabular*, where the configuration of all factors is written explicitly for each rule, and *intermediate*, where redundancy is reduced via boolean simplification.

We follow this illustrative example to propose some algorithms to transform one representation into another. This is sufficient to handle the problem of incremental construction. Given a constraint-based rule base Δ, the refinement required for the introduction

[1]For instance, the implicit ordering of rules with factors expressed at different levels of generalization can be obtained via Pearl's *system Z* (Pearl, 1990).

[2]This is a "narrow" definition of relevance. In general, given a rule, we care also of the rules whose conclusions are premise for the rule, etc.

[3]For completeness, we use the expression $\to ?$ to identify the lack of consequence, for situations which do not imply any conclusion.

of a new rule r_n is equivalent to the conversion of the priority-based rule-base $\Delta + \{r_n\}$, where r_n has higher priority than all other rules.

3.1 From Priority-based to Intermediate Constraint-based Representation

Suppose we have a strictly ordered set of simple rules $\Delta = \{r_1, r_2, .., r_n\}$, where r_1 is the rule with lowest priority, r_n with the highest. The following algorithm overwrites the elements of Δ with their translation in a constraint-based intermediate representation:[4]

```
for each $r_i \in \Delta$, starting from $r_n$ down to $r_2$
  for each $r_j \in \Delta$, starting from $r_{i-1}$ down to $r_1$
    if $Conclusion(r_j) = \neg Conclusion(r_i)$ then
      $r_j := \bigwedge Premise(r_j) \wedge \neg \bigwedge Premise(r_i)$
          $\to Conclusion(r_j)$
```

Stated in words, each cycle adds the complement of the domain of applicability of the rule with higher priority to the rule with lower priority.

The negation in the core formula usually applies to sequence of multiple *and* expressions, so it is equivalent a sequence of *or* expressions. Using the *body disjunction* formula (2) we can divide such compound rule in its simple components, and some of those may be redundant with other rules in the rule base. This is not the only potential cause of redundancy: for historical or clarity reasons, redundant rules may have been introduced already in the priority-based account (ex. $r_3 : A \wedge \neg B \to P$). As a consequence, the previous algorithm may produce subsumed or cloned rules[5], which can be removed without harm.

3.2 From Intermediate to Full-tabular Constraint-based Representation

The same regulation can be described with several intermediate representations. A way to compare them is to fully compute their extensional meaning, i.e. the underlying full-tabular representation. In order to do that, we can expand each rule with the unspecified factors which are relevant (i.e. are part of the premises) for relevant rules (i.e. having the same conclusion or its complement) in the rule base.

For instance, in the illustrative example, r_1 and r_2 are relevant to each other, and refer to the two factors A and B. However, $r_2 : B \to \neg P$ does not explicitly refer to the factor A, but it can be rewritten as two full-specified rules using some boolean properties:

$$B \to \neg P \Rightarrow (B \wedge A \to \neg P) \wedge (B \wedge \neg A \to \neg P)$$

If a rule has n unspecified factors in the constraint-based intermediate representation, then it is written as 2^n rules in the full-tabular representation.

3.3 From Full-tabular to Minimal Constraint-based Representation

Our objective at this point is to find a way to produce a *minimal* intermediate form, i.e. removing all redundancy. The solution we propose is to pass by the full-tabular representation and then simplify using known algorithms for boolean simplification.

We have a set of rules $\Delta = \{r_1, r_2, .., r_n\}$, represented in a full-tabular constraint-based way. Its characterization can be visualized using a table similar to a truth table. The only difference is that we are not interested in showing the specific truth value of the output variable for all combinations of inputs, but in mapping *where* specific outputs are defined. Considering the illustrative example, we have:

A	B	P	$\neg P$	?
T	T		X	
T	F	X		
F	T		X	
F	F			X

To reduce the implicit redundancy of the full-tabular representation, we can apply then the same principle of the *minterm* expansion used for logic circuits, in order to compute the characteristic function.

First, we aggregate all rules which share the same conclusion ϕ into *seed* rules r_ϕ:

```
for each $r_i \in \Delta$, with $r_i \neq r_\phi$
  $\phi := Conclusion(r_i)$
  if $r_\phi$ is not set then $r_\phi := r_i$
  else $r_\phi := \bigwedge Premise(r_\phi) \vee \bigwedge Premise(r_i) \to \phi$
```

Second, we apply an algorithm for boolean reduction — e.g. Quine-McCluskey (Mccluskey, 1956)[6] — to reduce each r_ϕ to its minimal form.

These algorithms take as input the logical addition (disjuction, *or*) of the products (conjunction, *and*) of factors producing a target ϕ. The output is a rule r'_ϕ which may be a compound rule. In this case, it can be

[4] $\bigwedge\{f_1, f_2, ..., f_n\} \Leftrightarrow f_1 \wedge f_2 \wedge ... \wedge f_n$

[5] Given two rules r_i and r_j such that $Conclusion(r_i) = Conclusion(r_j)$, if $Premise(r_i) \supseteq Premise(r_j)$ then r_i subsumes r_j; if $Premise(r_i) = Premise(r_j)$ then r_i clones r_j.

[6] The Quine-McCluskey algorithm provides an optimal solution, but is costly in computational terms. There are heuristics which provide suboptimal solutions consuming less computational power exist, e.g. Espresso.

divided in its components $r'_{\phi 1}$, $r'_{\phi 2}$, … using the *body disjunction* formula.

Finally, considering both positive and negative characterizations of a certain conclusion ϕ, we obtain $\Delta_\phi = \{r'_{\phi 1}, r'_{\phi 2}, \dots, r'_{\neg\phi 1}, r'_{\neg\phi 2}, \dots\}$, which is by construction the minimal constraint-based representation of the ϕ-relevant component of the initial rule-base.

3.4 From Constraint-based to Priority-based Representation

Suppose we have a set of rules Δ, given in a minimal constraint-based representation. In order to obtain an associated priority-based representation we need some partial ordering between the rules. Such ordering can be: (a) *hard-coded*, i.e. provided as input, (b) obtained by some evaluative function which takes the rules as inputs (§ 4). In the following we suppose the ordering is already available; for simplicity, we assume that the input priorities constructs a strict order on the rule set (we label r_n the rule intended to be with the highest priority, r_1 with the lowest one). The following algorithm performs the transformation of Δ according to such priority labeling:

```
for each $r_i$ in $\Delta$, from $r_n$ to $r_1$
  $\phi = Conclusion(r_i)$
  $factors := RelevantFactors(\Delta, \phi)$
  if $notYetEvalSituations_\phi$ is not set
    $notYetEvalSituations_\phi := Allocate(factors)$
  if $establishedFactors_\phi$ is not set
    $establishedFactors_\phi = \varnothing$
  create two empty rule bases $\Delta_i$ and $\Delta'_i$
  add $r_i$ to $\Delta_i$
  convert $\Delta_i$ to full tabular considering factors
  for each $r_j \in \Delta_i$
    if $Premise(r_j) \in notYetEvalSituations_\phi$
      $newPremise := Premise(r_j)$; $apply :=$ true
      for each $f \in establishedFactors_\phi$
        if $\neg f \in Premise(r_j)$
          $apply :=$ false; break
        else if $f \in Premise(r_j)$
          $newPremise := newPremise \setminus \{f\}$
      if $apply \wedge newPremise \neq \varnothing$
        $r'_j := \bigwedge newPremise \rightarrow \phi$; add $r'_j$ to $\Delta'_i$
      $notYetEvalSituations_\phi :=$
        $notYetEvalSituations_\phi \setminus Premise(r_j)$
      $establishedFactors_\phi :=$
        $ExtractFacts(notYetEvalSituations_\phi)$
  apply Quine-McCluskey on $\Delta'_i$ obtaining $r'_i$
```

The general principle of the algorithm is to create *clusters* of rules depending on the conclusion, and to synthesize the reciprocal dependency only within these clusters. We leverage the following information: which factors are relevant, considering the whole rule base, in respect to the conclusion [$Relevant(\Delta, \phi)$]; the set of all possible situations, via the allocation of a truth value to such relevant factors [$Allocate(\cdot)$]. Expanding the rule to its full-tabular components, we prune the situations which are evaluated, and we extract the set of factors which have been established on the not yet evaluated situations [$ExtractFacts(\cdot)$].[7]

4 INTEGRATING ASSUMPTIONS

In a constraint-based representations all relevant and discriminatory factors are made explicit. In an operational setting, however, it is difficult and not efficient to evaluate all of them: the check of conditions may require external investigations.

Following optimization principles, it is important to put in the beginning checks on conditions which cost less (more in general, have better pay-off) and provide more discrimination in the set of possible execution paths (i.e. are more *informative*, in Shannon's terms). This means that knowledge about the world and of the operational mechanisms of the system *do* provide elements to create a priority in the evaluation. Consider for instance this example:

```
if (f()) [...];
else if (g()) [...];
else [...];
```

This implementation would be really poor if `f()` is very costly and often false, while `g()` is not computationally expensive and often true.

4.1 Payoff Analysis

In general, the expected payoff in performing an action can be decomposed as:

$$E[payoff] = p(success) \cdot E[payoff\ of\ success] + p(failure) \cdot E[payoff\ of\ failure]$$

Actions are expected to have some side effects, which may occur at symbolic level (e.g. acquiring new information) and at practical level (e.g. modifying the physical environment). The second aspect may be critical in certain settings. For instance, trying to get a certain evidence often brings the risk to destroy other evidence. Such side effects modify what the agent could do in the following step. For simplicity, we neglect this dynamic component in this work. Considering a (non invasive) action, the most important voice

[7]An implementation in Java/Groovy of the algorithms is available on our site. http://justinian.leibnizcenter.org/rulebaseconverter

related to side effects remains a quantification of the expected cost.

Suppose that the action is an *investigation*. The success corresponds to reach a certain conclusion about ϕ. Assuming the cost is known and the same for both success and failure, the previous equation becomes:

$$\begin{aligned} E[payoff] = {} & p(success) \cdot E[payoff\ of\ concluding\ \phi] \\ & + p(failure) \cdot E[payoff\ of\ non\ concluding\ \phi] \\ & - cost \end{aligned}$$

The investigation is worth if $E[payoff] > 0$, i.e. if

$$E[payoff\ of\ concluding\ \phi] > \frac{cost - (1 - p(success)) \cdot E[payoff\ of\ non\ concl.\ \phi]}{p(success)}$$

In the case in which no positive or negative outcome is entailed from not concluding ϕ, the previous constraint is simplified to:

$$E[payoff\ of\ concluding\ \phi] > \frac{cost}{p(success)} \qquad (3)$$

4.2 Rule Guided Investigation

Suppose our knowledge is described in the form of a rule base. Let us consider a target matter ϕ. We know that the factors concerning ϕ, i.e. the factors relevant for rules implying ϕ or $\neg\phi$, are the *reasons* which possibly allow us to conclude ϕ or $\neg\phi$. In Shannon's terms, they provide the dictionary of symbols which should be possibly extracted from the informational source for our investigative purposes.

4.2.1 Individual Evaluation

Suppose we have a rule whose conclusion is ϕ, belonging to a rule base Δ described in a constraint-based representation. The probability that the state of the world satisfies the domain of applicability of r is given by:

$$p(applicable(r)) = p(\bigwedge Premise(r))$$

An action on investigation performed following a certain rule is successful if it is able to bring to the intended conclusion ϕ. Therefore, accounting the set of known facts K:

$$p(success) = p(\bigwedge Premise(r)|K)$$

Going further, the function quantifying the informational cost can be written in the form $c(D,K)$, where D is a set of factors we need to investigate, and K is the set of known facts.

$$cost = c(D,K)$$

Thus, the rule r should be evaluated only if:

$$E[payoff\ of\ concluding\ \phi] > \frac{c(D,K)}{p(\bigwedge Premise(r)|K)}$$

4.2.2 Optimization

In general, the rule base may contain several rules which entail ϕ. The choice about which rule should be evaluated first is optimal if the rule maximizes the investigative payoff.

The payoff of reaching a conclusion about ϕ is independent of the rule used, therefore can be labeled as a constant:

$$G = E[payoff\ of\ concluding\ \phi]$$

Furthermore, in respect to a rule r, the set of desired factors D consists of the factors in the premise which are not known:

$$D = Premise(r) \setminus K$$

Neglecting the payoff of not concluding ϕ, we have:

$$E[payoff] = G \cdot p(\bigwedge Premise(r)|K) - c(D,K) \quad (4)$$

The next paragraphs consider a few simple configurations.

Equal Probabilities. Suppose that there are n factors relevant for ϕ, statistically independent and with equal probability. Supposing $c(D,K)$ negligible, we have:

$$E[payoff] \propto \frac{1}{n}^{\#D}$$

Suppose also we are ignorant about the world, therefore $\#D = \#Premise(r)$. In a full-tabular constraint-based representation, all rules have the same number of premises, therefore the expected payoff is the same for all rules. Conversely, in a minimal constraint-based representation, rules have been constructed by functional aggregation reducing the number of premises. Therefore the optimal choice is to choose the rule with less conditions in the premise.

Unequal Probabilities. In condition of ignorance, and with statistically independent factors, we have

$$p(Premise(r)) = p(f_1) \cdot p(f_2) \cdot \ldots \cdot p(f_n)$$

At this point, it is interesting to refer to the theory of communication introduced by Shannon (Shannon, 1948). Defining the information related to a certain symbol s as:

$$I(s) = -\log_2(p(s))$$

the information of a message m composed of different and statistically independent symbols (s_1, s_2, ..., s_n) is given by the sum of information of each symbol:

$$I(m) = I(s_1) + I(s_2) + \ldots + I(s_n)$$

Thus, assigning each factor to a symbol, the premise of a rule r can be interpreted as a message. As information maintains monotonicity, we can consider the best rule to be chosen not in terms of maximizing the product of probabilities, but as minimizing the sum of information required to reach the target conclusion, possibly integrating the cost component.

4.3 Default Assumptions

Let us consider again the story reported in the introduction. Both Samuel and Raphael think it is normal to take the umbrella if it is raining. Simplifying[8], we can write this practical rule as:

$$rain \rightarrow umbrella$$

With no prior assumptions, the rule base of both agents consists of a simple rule. Now, suppose the agents have a probabilistic model of the world. According to (4), the payoff of the evaluation of that rule is equal to:

$$E[payoff] = G \cdot p(rain) - c(\{rain\}, K)$$

If the agent already knows the fact that it will rain, supposing retrieval costs negligible, the overall cost c is null. In such case, the payoff of the investigation is always positive. It tends to 0 when the agent thinks that the probability of rain is very low, as in the case of Samuel, living in a sunny country: $p(rain) \sim 0$.

However, if Samuel's knowledge base does not contain the "rain" fact, the expected payoff of the investigation may be negative: if $G \cdot p(rain)$ is small enough, the payoff constraint is violated, so the rule shouldn't be evaluated.[9] Raphael is in the opposite condition: as $p(rain)$ is not null, the positive addend in the expected payoff of investigation makes the cost of checking the weather negligible.

4.3.1 Prioritization with Default Rules

The analysis of evaluation payoffs provides an optimal order of investigation and such ordering can feed the conversion from constraint-based to priority-based representations (§ 3.4). However, if the expected payoff is null or negative, the related investigation is not worth and the agent should rather consider to introduce a default assumption rule. A simple way to represent that is to introduce the *negation as failure* operator, as defined for instance in *Answer Set Programming* (Baral and Gelfond, 1994; Lifschitz, 2008). In practice, when we commit to a default assumption concerning the fact f, we add to the rule base the following rules:

$$(\text{not}\neg f \rightarrow f) \wedge (\text{not}\neg f \wedge \text{not}\, f \rightarrow \neg f)$$

Note that there is an implicit priority between the two rules, cf. *system Z* (Pearl, 1990). Such intervention does not modify the previous rules, but overrides the investigations concerning f. In other words, they force the agent not to perform the investigation but just to refer to his current knowledge, so that the payoff constraint is not violated.[10]

5 CONSTRUCTION AND RECONSTRUCTION

The paper investigated two types of events changing a rule base. The first type of events consisted of incremental modifications: new rules are added by external intervention, determining a partial reconfiguration of the operational knowledge used by the agent. Because of distinguishing actions, the new rules brings to the foreground factors left implicit in the previous rules.

The second family of events concerned *ad-hoc* reorganizations, aiming for better adaptation. However, when a rule base is "compiled" to a more efficient priority-based form, the agent loses the reasons motivating that structure (e.g. probabilistic assumptions), and therefore he cannot check if those reasons are still valid.

To rewrite the rule base again, the agent has to *reflect* over the rule base. He has to unveil the underlying constraint-based representation, removing all default assumptions and recompute the optimal priority-based indexing.

What may motivate the agent to do that? An intuitive answer is because of (repeated) failures of his current practical reasoning. For instance, if Samuel moves in a rainy place, his default assumption about the rain will lead him to get wet often. If the number of practical failures exceeds a certain (subjective) threshold, we expect the agent will look for a better *adaptation* to the context, asynchronously starting the reflection cycle.

[8] We put aside all issues related to causality, intentionality, action, etc.

[9] The payoff G of concluding the fact that it rains can be interpreted in terms of the practical consequences that such conclusion entails. For instance, the agent will avoid to become wet, by taking the umbrella. How much this is valuable to the agent depends on subjective components, and plausibly changes in time.

[10] There may be more radical transformations. For instance, destructive simplifications can be imagined removing f from the premises of the rules, removing from the rule base rules which have $\neg f$ in the premises, etc.

6 CONCLUSION

Part of a wider research concerning *agility* in public administration and policy making (Boer and Engers, 2013), the present paper starts the development of a computational framework operationalizing a *constructivist* approach to rule bases. By dividing agency in a *regulated* and a *regulatory* sub-systems, we explicitly disjoin the processing of facts, depending on the rule base, from the modification of the rule base.

The analysis we presented is not targeting beliefs—in the traditional sense of the *belief revision* literature, e.g. (Fermé, 2011)—nor built upon a model of dynamic *theory revision* of knowledge accounting for both facts and rules, as in *machine learning*, see e.g. (Omlin and Giles, 1996), (Goldsmith and Sloan, 2005). Similarly to Horty (Horty, 2011), our specific scope is on *rules*, as components of a rule base, *already defined at symbolic level*.

In psychology, the theory of constructivism is traditionally related to Jean Piaget (Piaget et al., 2001), who investigated the mechanisms under which knowledge is internalized by learners. He argued that individuals *construct* their knowledge through the two processes of:

- *assimilation*: the process of framing new experiences through the existent knowledge framework, without changing it; the structure exists, it is filled by data;
- *accomodation*: the process of re-framing the agent's knowledge framework, usually responding to contradictions or operational failures of their knowledge framework; the structure is reorganized.

This theory is aligned with our contribution, as:

- rule bases are interpreted as compiled programs for reactive symbolic processing modules, which *respond* to facts (e.g. data fed by sensors), possibly performing actions (e.g. sending stimuli to actuators, communicating, etc.);
- construction and reconstruction of rule bases provide the reflective, adaptive processing dimension, which occurs concurrently to the first one.

The paper is a starting operationalization, therefore several research directions remain to be investigated. First, an evaluation of the application of our proposal on more complex rule bases, possibly in comparison with other approaches from the *expert systems* literature. Related to that, an in-depth analysis of the proposed algorithms is required, measuring their complexity, and suggesting possible optimizations. Second, an investigation of possible theoretical interactions with the insights coming from belief revision, e.g. (Dubois, 2008), but applied on our definition of rule revision.

Finally, the paper explores adaptation only as a problem of *maximization of payoffs*, studied in decision theory, game theory and similar *top-down* perspectives. In the future, we plan to analyze it through the lens of Heiner's theory of predictable behaviour (Heiner, 1983); in this way, we expect to be able to model adaptation as a *bottom-up* mechanism as well.

REFERENCES

Baral, C. and Gelfond, M. (1994). Logic programming and knowledge representation. *The Journal of Logic Programming*, 19:73–148.

Boer, A. and Engers, T. (2013). Agile: a problem-based model of regulatory policy making. *Artificial Intelligence and Law*, 21(4):399–423.

Dubois, D. (2008). Three scenarios for the revision of epistemic states. *Journal of Logic and Computation*.

Fermé, E. (2011). *On the Logic of Theory Change: Extending the AGM Model*. PhD thesis, Royal Institute of Technology.

Goldsmith, J. and Sloan, R. H. (2005). New Horn Revision Algorithms. *Journal of Machine Learning Research*, 6:1919–1938.

Heiner, R. (1983). The origin of predictable behavior. *The American economic review*, 73(4):560–595.

Horty, J. F. (2011). Rules and Reasons in the Theory of Precedent. *Legal Theory*, 17(01):1–33.

Lifschitz, V. (2008). What Is Answer Set Programming? *Proceedings of the AAAI Conference on Artificial Intelligence*.

Mccluskey, E. J. (1956). Minimization of Boolean functions. *The Bell System Technical Journal*, 35(5):1417–1444.

Omlin, C. and Giles, C. (1996). Rule revision with recurrent neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):183–188.

Pearl, J. (1990). System Z: A natural ordering of defaults with tractable applications to nonmonotonic reasoning. *Proceedings of the 3rd conference on Theoretical Aspects of Reasoning about Knowledge (TARK'90)*, 3:121–135.

Piaget, J., Piercy, M., and Berlyne, D. E. (2001). *The Psychology of Intelligence*. Routledge classics. Routledge.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423/623–656.