A study of explanation generation in a rule-based system

Karim El Mernissi

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

École doctorale Informatique, Télécommunications et Électronique (Paris) IBM France Lab / LIP6

Présentée par

Karim EL MERNISSI

Pour obtenir le grade de

DOCTEUR EN INFORMATIQUE

Spécialité

Intelligence Artificielle / Aide à la décision

Sujet de la thèse :

Une étude de la génération d'explication dans un système à base de règles

Présentée et soutenue publiquement le 13 décembre 2017

devant le jury composé de :

M. Nicolas MAUDET	Directeur de thèse
Mme. Wassila OUERDANE	Encadrante de thèse
M. Pierre FEILLET	Directeur industriel (invité)
M. Alexis TSOUKIAS	Rapporteur
M. Vincent MOUSSEAU	Rapporteur
M. Jean-Pierre BRIOT	Examinateur

Remerciements

Je voudrais exprimer ma gratitude à tous ceux qui m'ont soutenu durant cette aventure.

En premier lieu, je tiens à exprimer toute ma reconnaissance à Pierre Feillet, Nicolas Maudet et Wassila Ouerdane pour leur bienveillance, leur soutien et leur encadrement tout au long de la thèse. Je remercie tout particulièrement Wassila et Nicolas qui m'ont beaucoup appris sur la méthodologie et la démarche scientifique, et Pierre qui a toujours été très disponible et m'a prodigué de nombreux conseils techniques et de communication. Merci à tous les trois pour vos questions et remarques qui ont guidé ma réflexion et m'ont permis d'avancer.

Ce travail n'aurait pas été possible sans le soutien de l'IBM France Lab qui a financé mes travaux de recherche et m'a permis de m'y consacrer sereinement.

J'adresse également mes remerciements à Alexis Tsoukias et Vincent Mousseau qui ont accepté d'être mes rapporteurs et aussi à Jean-Pierre Briot qui a accepté d'être membre du jury. Je veux aussi exprimer ma gratitude à Christian de Sainte Marie, qui a mis en œuvre les moyens logistiques et administratifs pour que la thèse se déroule dans de bonnes conditions et a aussi contribué à renforcer la culture et l'ouverture scientifique au sein de l'IBM France Lab en nous proposant ses très enrichissants vendredis du CAS.

Mille mercis à mes collègues d'IBM qui ont contribué à rendre cette expérience inoubliable. Merci à Yiquan, Kevin, Moussa, Carlos, Nicolas, Patrick, Julie, Rachel, Yves, Changai et tous ceux que j'oublie pour la bonne atmosphère qu'ils ont su instaurer. Un merci tout particulier aux autres doctorants d'IBM avec qui nous avons beaucoup échangé tout au long de notre thèse. J'ai une pensée toute particulière pour Hamza, Oliver, Oumaima et Reda dont j'ai beaucoup apprécié la compagnie et sans qui tout aurait été différent.

Je tiens également à remercier mon camarade de bureau au LIP6, Cédric, pour nos nombreuses discussions et sa très appréciée présence qui a égayé mes journées passées à l'UPMC.

De la même manière, je témoigne ma reconnaissance aux doctorants du LGI qui m'ont chaleureusement intégré et avec qui j'ai eu de nombreux échanges lors de mes passages à CentraleSupélec. Un merci particulier à Selmen, Hichame, Haytem, Julien, Tasneem et Massi pour la bonne humeur ambiante et l'atmosphère de travail qu'ils ont su véhiculer.

Au terme de ce marathon, je remercie enfin celles et ceux qui me sont chers et qui se reconnaitront. Merci à Pauline, à mes amis et à ma famille qui m'ont supporté dans les bons comme dans les mauvais moments.

Résumé

Le concept de “Business Rule Management System” (BRMS) a été introduit pour faciliter la création, la vérification, le déploiement et l'exécution des politiques commerciales propres à chaque compagnie.

Basée sur une approche d'intelligence artificielle symbolique, l'idée générale est de permettre aux utilisateurs métier de gérer les changements des règles métier dans un système sans avoir besoin de recourir à des compétences techniques. Il s'agit donc de fournir à ces derniers la possibilité de formuler des politiques commerciales et d'automatiser leur traitement tout en restant proche du langage naturel.

De nos jours, avec l'expansion des systèmes de décision automatique, il faut faire face à des logiques de décision de plus en plus complexes et à de larges volumes de données. Il n'est pas toujours facile d'identifier les causes conduisant à une décision. On constate ainsi un besoin grandissant de justifier et d'optimiser les décisions dans de courts délais qui induit l'intégration à ces systèmes d'une composante d'explication évoluée.

Le principal enjeu de ces recherches est de fournir une approche industrialisable de l'explication des processus de décision d'un BRMS et plus largement d'un système à base de règles. Cette approche devra être en mesure d'apporter les informations nécessaires à la compréhension générale de la décision, de faire office de justification auprès d'entités internes et externes ainsi que de permettre l'amélioration des règles existantes.

La réflexion se porte tant sur la génération des explications en elles-mêmes que sur la manière et la forme sous lesquelles elles sont délivrées.

Il est à noter que l'approche proposée s'applique à tout moteur decisionnel raisonnant sur des relations de cause à effet notamment des moteurs de workflow ...

Abstract

The concept of "Business Rule Management System" (BRMS) has been introduced in order to facilitate the design, the management and the execution of company-specific business policies. Based on a symbolic artificial intelligence approach, the main idea behind these tools is to enable the business users to manage the business rule changes in the system without requiring programming skills. It is therefore a question of providing them with tools that enable to formulate their business policies in a near natural language form and automate their processing. Nowadays, with the expansion of intelligent systems, we have to cope with more and more complex decision logic and large volumes of data. It is not straightforward to identify the causes leading to a decision. There is a growing need to justify and optimize automated decisions in a short time frame, which motivates the integration of advanced explanatory component into its systems. Thus, the main challenge of this research is to provide an approach for explaining rule based decisions with respect of client production requirements. This approach should be able to provide the necessary information for enabling a general understanding of the decision, to serve as a justification for internal and external entities as well as to enable the improvement of existing rule engines. To this end, the focus will be on the generation of the explanations in themselves as well as on the manner and the form in which they will be delivered.

Contents

1 Introduction **15**

1.1 Context 15

1.1.1 A brief history of decision-making systems 15

1.1.2 Business rule management systems 17

1.1.3 Explanation in rule-based systems 19

1.2 Motivations and objectives 20

1.3 Thesis outlines 20

2 Rule-Based systems, BRMS and decision automation **23**

2.1 General information about rule-based systems 23

2.2 Rules in decision automation 26

2.2.1 Business rules: basic definitions 28

2.2.2 Basic architecture for rule-based systems 30

2.3 The IBM business rules management system 39

2.3.1 Hierarchy of decision service in IBM ODM 41

2.3.2 IBM Operational Decision Manager: platform and architecture 43

2.3.3 The rule engine of IBM ODM 49

2.3.4 Applications 52

2.4 Discussion and conclusions 54

3 An overview of explanation in rule-based Systems 57

3.1 The need for explanation 57

3.2 Philosophical background on the theory of explanation 60

3.3 Explanations in rule-based systems 62

3.3.1 Categorization by temporal context / explanation orientation 63

3.3.2 Type of questions 64

3.3.3 Content types of explanations 67

3.3.4 Context sensitivity 69

3.4 A quick historical overview of expert systems with explanation capabilities 72

3.5 Causality in rule-based systems 75

3.5.1 Causal explanations in rule-based systems 78

3.5.2 Choosing a causal model / formal model for causal ascription 81

3.6 Requirements for an IBM ODM explanation feature 84

4 A simplified causal model for rule-based systems 89

4.1 Introduction 89

4.2 Concepts and definitions 89

4.2.1 Business rules 89

4.2.2 Towards a normalized business rule formalism 91

4.2.3 Orchestration and execution of the business rules 93

4.2.4 Tracing the process: what should be in the decision trace? . . 96

4.3 Representing causality in business rule-based systems 97

4.3.1 Events typology in a business rule decision 99

4.3.2 Defining the signature of a rule-based system 101

4.3.3 Causality between events 103

4.3.4 Typology of the relations 104
4.3.5 Hierarchical causal model 107
4.4 Process of construction of a minimal causal model 108
4.4.1 Causal model of the system and list of relevant events 108
4.4.2 Minimal decision trace 110
4.4.3 Minimal causal model of the decision 111
4.4.4 Conclusion 115

5 Engineering the causal model 117

5.1 A Framework for Causal Ascription and Representation in Rule-Based System 119
5.1.1 Encoding business rules (BR) into BR-Objects 119
5.1.2 Extracting the business rules of a decision project 123
5.1.3 Ascribing causal relations between the business rule elements . 125
5.1.4 Recording the minimal trace of a decision 134
5.1.5 Constructing the minimal causal model of a decision 138
5.2 Experiments and assessment protocol 140
5.2.1 Reduction for the business rule-based system's causal model . 141
5.2.2 Reduction for minimal traces of the decision 142
5.3 Conclusion 144

6 Towards an architecture of an explanation service for business rule-based systems: basics and insights 147

6.1 Introduction 147
6.2 The basics towards a service architecture to support explanatory models 150
6.3 The components of an explanatory model 154

6.3.1 Conceptual model as a part of the explanatory model 154
6.3.2 User model as a part of the explanatory model 158
6.4 Exploitation of the explanatory model: discussion and insights . . . 160
6.4.1 How we deal with each criterion 160
6.4.2 A graphical representation for engineering the explanation . . 161

7 Conclusion **165**

Bibliography **171**

Annexes **185**

A Example of business rules normalization **185**

B Causal model of minimal decision trace **191**

List of Tables

2.1 Usecase: Rule Engine Execution Modes 53

3.1 Explanation Aims Table (Tintarev and Masthoff, 2007) 59

3.2 Explanation Types Table (Hovorka et al., 2008) 70

3.3 Categorization of existing solutions based on our explanation criteria *(* limited capabilities)* 71

4.1 A Business Rule Formalism for object-based applications 92

4.2 Predicates and their meaning 93

4.3 Actions and their meaning 93

List of Figures

2.1 Basic Architecture: essential components of a Rule-Based System . . 31

2.2 Compiled knowledge: from deep to surface knowledge 36

2.3 Example of ruleflow for a loan application usecase (source) 43

2.4 Operational Decision Manager Global View (source) 44

2.5 IBM ODM global view (derived from (source 1) and (source 2)) . . . 45

2.6 Business Objet Model and Vocabulary (source) 47

2.7 Execution Object Model (XOM) (source) 48

2.8 Relation between BOM and XOM (source) 49

2.9 Components of a Decision Service (source) 50

2.10 Example of Decision Service architecture (source) 50

4.1 Method overview: Constructing a minimal causal model of the decision 109

4.2 Causal network of the decision under a decision artifact perspective . 114

4.3 An example of causal network at rule level for example 4.4 115

5.1 Levels of granularity 118

5.2 UML model: Representing a BR-object (ie. business rule object) . . . 120

5.3 UML model: Extract business rules from a decision project (i.e. a rule project) 124

5.4 Example of BRL extraction 125

5.5 RelationsAnalyzer 126
5.6 Global view of the causal ascription of candidate for (in)eligibility relation 129
5.7 Action simulation 130
5.8 Condition simulation 130
5.9 Trace the execution 134
5.10 Decision Event History Classes 135
5.11 Causal Model Server Classes 139
5.12 Generated minimal causal graph of the decision at rule elements level 140
5.13 Generated minimal causal graph of the decision at rule level 140

6.1 Explanation Service Architecture 151
6.2 Explanation Service Architecture 152
6.3 Explanation Service Architecture 153
6.4 Example part of conceptual model for attribute yearlyRepayment . . 158
6.5 Example of a node information in the explanative model 164

Chapter 1

Introduction

This Ph.D. thesis deals with the generation of explanations for business rule-based systems. Our main objective is the development of a framework for the construction of generic explanations for business rule-based decisions. This introduction is structured as follows. In section 1.1 we present the context of our study. In section 1.2 we discusses the motivations and the objectives of this research work. Finally we present the structure of this thesis document in section 1.3.

1.1 Context

1.1.1 A brief history of decision-making systems

Artificial intelligence and decision theory take their roots in the research on systematic methods for problem solving and decision making that made significant breakthrough in the forties. Notably, in 1943 the logician Emil Post proved that mathematical or logical systems could be written as some sort of production system, building on the idea that such a system can be seen as a set of rules specifying how a string of symbols (antecedent) can be turned into another set of symbols (consequent) (Post, 1943). At that period, Alan Turing aimed to understand the thinking mechanisms of the human intellect. In 1948, Turing writes (Turing, 2004), an attempt to model the brain, and sets out some preliminary ideas on artificial neural networks. Two years later, he publishes (Turing, 1950) and discusses the con-

cept of a thinking machine and emphasizes that "building a mind" requires sufficient knowledge about the world to represent its states and a set of rules to model a behavior (which cannot be as complex as human behavior). This is perhaps one of the earliest leads of rule-based systems. A few years later, the term artificial intelligence was officially coined.

Thus, the works of Alan Turing and Emil Post set out the first ideas of decision-making systems as we know them today. It is also important to emphasize older contributions like the lambda calculus of Alonzo Church, the recursive function of Kurt Gödel and Jacques Herbrand and the "Entscheidungsproblem" posed by David Hilbert and Wilhelm Ackermann that help in laying the foundations for these research works. Since their contributions, mathematicians and computer scientists have anticipated (sometimes with careless promises) the day when decision-making would be delegated to machines. After the seventies, thanks to the continuous development of computers and artificial intelligence techniques, decision-making systems have become more accessible and a rise in decision automation has been observed. At that time, these systems are mainly artificial intelligence programs using knowledge base and heuristics to emulate the thought process of a human expert. For companies which perpetually look for tools to improve their organizational performance, the idea to use these "intelligent systems" to automate the application of their business policies come naturally. In practice automated decision-making systems present strong advantages:

- they combine multiple human expert intelligences and centralize the decision process,
- they increase the reliability and visibility of the decision process,
- they can deal with huge amount of information,
- they minimize the employee training cost and reduce human errors,
- they increase the efficiency by reducing the time needed to solve problems.

Because of that, most of these systems are used for automating the application of business policies but also as "intelligence augmentation" tools for *supporting* the decision makers. Thus, automated decision are widely adopted by corporations and public organizations and find applications in a variety of business areas. For example, in medicine, they are used for diagnosis of different forms of human disease. In banks, they found applications in fraud detection and loan agreement. They are often used for pricing purposes in insurance companies. These are just a small sample of the possible applications and automated decision systems are actually used to solve complex problems in many other industrial and technical areas. Because of their very broad scope, these systems encompass a wide range of software technologies which fall in two categories according to their use of "symbolic" or "non-symbolic" approaches. In the "symbolic" approaches, the knowledge about the reasoning logic is explicitly stored and can be used to make inferences about various information and data. It includes methods like case-based reasoning or heuristic and normative expert systems. Unlike these approaches, the "non-symbolic" ones doesn't explicitly represent their reasoning logic in a straightforward way. This is the case in particular of methods like neural networks. Each of these methods has its own characteristics and thus its own field of applications. Indeed, whereas "non-symbolic" approaches are very popular for image recognition and found many social applications, "symbolic" ones, as the heuristic expert systems using business rules, are preferred for critical applications requiring high levels of accuracy and transparency.

1.1.2 Business rule management systems

In order to be commercially viable, an automated decision-making system has to demonstrate a high level of performance and reliability, and also has to show good transparency and flexibility. For example, in trading or security applications, there is a possibility of great financial loss if the system malfunctions and cannot be updated quickly and regularly. For these reasons, such a system must be efficient and reliable, but it also has to be transparent and flexible enough to allow an easy

maintenance by the organization using it. Among these systems, the symbolic ones and more particularly those using business rule approaches, which are flexible and offer a great accuracy in decisioning, are fairly widespread to deal with "critical-applications" in companies. In fact, rule-based representations of knowledge have been widely used by organizations because they express situation-action heuristics in a natural way and allow to reason by making deductive inferences that make naturally sense for human users. Because of that, the correctness of the problem-solving method used by a rule-based system can be verified more easily. Moreover, monitoring rule-based decisions allows to develop learning procedures capable of inferring rules from experience. These newly learned rules can then be incorporated into the knowledge base of the system to improve the quality its reasoning.

In practice, the building and maintenance of rule-based systems are managed by Business Rule Management Systems (aka. BRMS). A Business Rule Management System is a specific implementation that encompasses the rule-based systems themselves together with the environment dedicated to their development. It provides to business analysts and experts the capability to author, test, simulate and run their decision logic in a "near natural language way". Using Business Rule Management Systems allows companies to capture, refine and reproduce human expertise by building business rule-based systems that automate problem solving and are proving to be easily maintainable and commercially viable Hayes-Roth (1985b). Because of that, Business Rule Management Systems have changed how organizations think business processes and IT but have also transformed the way they address applications development needs. The IBM corporation, which provides IBM Operational Decision Manager, is the market leader in business rule management systems and cares about increasing the value and the acceptance of its software solutions thanks to augmented explanation capabilities.

1.1.3 Explanation in rule-based systems

Indeed, a critical point for a business rule-based system to be used is the acceptance of its automated decisions by human users. A decision is recognized as correct by a human user only if the reasoning process associated to this decision is understandable and makes sense for him. In practice, the decision logic of business rule-based systems can be highly complex. As a consequence, the decisions taken by these systems can seem unclear and hard to accept and a system that cannot convince about the accuracy of its decisions has very little chance of being fully accepted and thus in end may not be used by companies which doubt about its reliability. That is why, being able to explain its reasoning is one of the most important abilities of a decision-making system. This means that rule-based systems must be able to explain their knowledge of the domain and the reasoning processes they employ to produce results and recommendations. Expert system researchers have identified several reasons why explanation capabilities are not only desirable, but crucial to success of expert systems. These reasons may vary depending on the type of user targeted:

- Understanding how the systems works by revealing the reasoning process and providing information about the contents of the system knowledge base (*Transparency*).
- Facilitating the debugging of the system during the development stages (*Scrutability*).
- Educating users both about the domain and the capabilities of the system (*Effectiveness/Efficiency*).
- Persuading users that the system's conclusions are correct so that they can ultimately accept these conclusions and trust the system's reasoning powers (*Persuasiveness/Trust*).
- Assessing the system's appropriateness for a given task. The scope of an expert

system may be quite narrow and explanation can help a user discover when a system is being pushed beyond the limits of its knowledge (*Effectiveness*).

1.2 Motivations and objectives

IBM Operational Decision Manager is widely used in the industry and notably in the insurance and banking sectors. In this thesis work, we aim at presenting a generic framework of explanation that could be used to increase the value of business-rule based applications in an industrial context. As IBM's clients need to use these systems in real time, the framework has to obey strict constraints on performance, the explanation capabilities we have to design must not be memory-consuming or greedy for computation time during the decision-process. Our proposal builds upon a simplified causal model of the system, which we show how to engineer in our industrial setting, and to how to exploit in a perspective of explanation. We believe this "case study" can be of interest beyond our specific system. Indeed, the underlying principles of the proposed framework can be extended to all devices that automate decisions based on heuristics or logical artifacts.

1.3 Thesis outlines

The rest of this thesis is structured as follows. Chapter 2 focuses on the rule-based systems. In this context, it describes what is a rule-based system, presents the IBM's BRMS. The chapter 3 states an overview of the explanation in rule-based systems. It studies in particular why causality is important for our purpose, and how it is accounted for in rule-based systems. Chapter 4 provides a method for mechanically constructing and reducing a set of simplified causal models that can be used for explaining a rule-based system and its decisions, and for reducing the size of the decision traces. Chapter 5 presents the implementation of the proposed method and an experimental protocol to measures and evaluate the obtained results. Finally, Chapter 6 proposes a generic explanation model that can be used

to construct explanations for business rule-based decisions and Chapter 7 discusses our results and opens perspectives.

Chapter 2

Rule-Based systems, BRMS and decision automation

2.1 General information about rule-based systems

According to Engelmore (1987), Artificial intelligence was primarily a field of academic research that looked for generic approaches to solve various complex problems and found its niche in "heuristic symbolic computing", where the data was largely symbolic and the problems not well-structured, requiring the use of heuristic techniques to reach good enough solutions. Practical applications of artificial intelligence gave rise to several sub-fields, such as natural language processing, computer vision, knowledge representation and reasoning, machine learning (the list is far from being exhaustive of course).

Among them, our interest lies in the area of *knowledge-based systems* that has been one of the fist areas of the artificial intelligence to be commercially fruitful and received a lot of attention as stated in (Lucas and van der Gaag, 1991). The early knowledge based-systems have been mostly used to replace or assist human experts in the solving of complex problems and are so called *expert systems*. The idea behind these systems is to provide a tool that emulates the thought process of human experts to automate business decisions. Automating decisions requires that sufficient knowledge about the business domains of the human experts has been acquired by the system. Under this perspective, several methods have been deve-

loped for building knowledge-based systems capable of making complex automated decisions. The main ones, according to Sun (1999) and Darlington (2013), fall into two categories :

(1) Symbolic approaches (aka. classical AI or GOFAI for Good Old-Fashioned Artificial Intelligence (Haugeland, 1989)) that explicitly store symbols and logically manipulate them during the reasoning process. These approaches constitute the branch of artificial intelligence that attempts to explicitly represent human knowledge in a declarative form (like facts and rules) in order to mimic the intelligence of a human expert. The idea behind this is to reproduce the mechanism of thought at a high level. The General Problem Solver presented by Newell et al. (1959), the Logic Theorist presented by Newell and Simon (1956) and later the first famous rule-based systems like Digitalis Advisor (Swartout, 1977) and MYCIN (Buchanan and Shortliffe, 1984) have been foundational to the symbolic artificial intelligence.

(2) Non-Symbolic approaches (essentially represented connectionist approaches) that do not use explicit knowledge but rather encode implicit knowledge that have a priori no meaning for a human reader. The works presented in (Rosenblatt, 1961) and (Smolensky, 1988) have been foundational for these approaches that have been less successful than the symbolic ones at the beginning of applied artificial intelligence but gain lot of importance in the last two decades. This success is explained by the increase of computing power and the memory but also because of the complementarity of the two approaches as depicted in (Harnad, 1990). Neural networks (Prieto et al., 2016) are good examples of this paradigm. The idea behind these approaches is to simulate the working of a human brain.

Business rule-based systems fall in the first category and are particularly popular with banks and insurance companies because the accuracy of their automated decisions and their flexibility make them great tool for dealing with “critical-applications”.

Indeed, Hayes-Roth (1985b) claims that "these systems automate problem-solving know-how, provide a means for capturing and refining human expertise, and are proving to be commercially viable". In other words, it means that these systems can automate the problem-solving providing that sufficient knowledge about the heuristics (what method is applied in what situation) used by human expert is available.

One of the most important aspect in these systems are the business rules themselves. Business rules constitutes a kind of declarative programming language that provides an easy way to capture and maintain the expert knowledge about heuristics and methods used to solve complex problems in a specific domain. As such, they can be seen as statements that define or constraint some aspects of a decision, which means that business rules are intended to assert a business structure that constrains and influence possible behaviors of an automated decision-making system. In Hayes-Roth (1985a), Frederick Hayes-Roth claims that intelligent systems are a critical part of an organization's information system and describes a rule as a relatively independent piece of know-how that specifies "a chunk of analytic problem-solving knowledge" and also states that "from an architectural perspective, rules are data that generally conform to highly specialized grammars capable of using symbolic expressions to define conditions and actions". Ross (2003) describes them as "A directive intended to influence or guide business behavior", von Halle (2001) as a "set of conditions that govern a business event so that it occurs in a way that is acceptable to the business" and Group (2008) as a "proposition that is a claim of obligation or of necessity". In fact, there is no standard definition for business rules but the BRG (Business Rules Group), as the most known peer group in business rules area, provides the following definition in the Guide Business Rules Project Report published in 2000:

> "A business rule is a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business. It is atomic in that it cannot be broken down or decomposed further into more detailed business rules. If

reduced any further, there would be loss of important information about the business."

In practice, business rules form a declarative language that explicitly states the business logic of a particular category of knowledge-based systems called rule-based systems. Researches focusing on building rule-based systems for industry have resulted in standard of software architecture that we describe in the next section.

2.2 Rules in decision automation

The rules are commonly specified by means of an ontology language, and often a description logic language. Depending on the contexts and uses, they have taken different form whose the most known are the following:

- **Logical Rules.** In propositional logic, a logical rule (aka. rule of inference or transformation rule) is a logical form that consists in a function which takes premisses and analyze their syntax to return conclusions. Such rules of inference include modus ponens (argument form: $((p \rightarrow q) \wedge p) \vdash q$), modus tollens (argument form: $((p \rightarrow q) \wedge \neg q) \vdash \neg p$), and contraposition (argument form: $(p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)$). In practice, a logical rule is used to infer new facts when premisses are satisfied.

  Example 2.1. *IF humans are mortal AND Socrate is a human THEN Socrate is mortal*

- **Production Rules.** The production rule formalism Lucas and van der Gaag (1991) constitutes a first attempt to adapt the logical rules to industrial applications and has been used in the first expert systems (MYCIN Buchanan and Shortliffe (1984), EMYCIN van Melle et al. (1984), DENDRAL Lindsay et al. (1993)...) A production rule represents the necessary conditions to make a state transition or an action. It means that the rule describes a set of states in its premisses, representing the initial situations that may occur and its con-

clusions modify this set of state by using an operation that can be a procedure or a method.

Example 2.2. *IF the patient has a stiff neck on flexion AND the patient has a headache THEN recommend medical check*

- **Probabilistic Rules.** The probabilistic rule formalism constitutes a first attempt to deal with uncertainty when using production rules. It augments the production rule formalism by assigning certainty factors (CF) to the conclusions of rules. This formalism is influenced by the probability theory but makes strong simplifying assumption concerning the independence of different rules. In practice, it has been shown that these simplifications could lead to erroneous conclusions.

  Example 2.3. *IF the patient has a stiff neck on flexion AND the patient has a headache THEN there is suggestive evidence that the patient's infection is meningitis (CF=0.5)*

- **Fuzzy Rules.** The fuzzy rule formalism constitutes an attempt to provide the rule-based systems with commonsense knowledge. In this formalism, a fuzzy rule defines a furry patch and connects commonsense knowledge to state-space geometry. This approach relies to the fuzzy Approximation Theorem (FAT) and is described in Kosko (1994) In practice, the fuzzy rules allow to represent fuzzy sets by using terms that can be interpreted in several way depending on the context.

  Example 2.4. *IF fever THEN recommend paracetamol*

- **Business Rules.** The business rule formalism is an attempt to adapt the production rule formalism in order to make it more compatible with the business needs of companies and organizations. The objective is to offer a more flexible and practical programming language than the production rule formalism. A business rule is a statement, taking the form of a customized "near

natural language" text, which defines a business aspect that allows to describe the decision criteria and the actions to apply for a given situation of a decision process.

Example 2.5. *IF 'the loan report' is approved AND 'the grade' is one of "A" , "B" , "C" THEN in 'the loan report', accept the loan with the message "Congratulations! Your loan has been approved" ;*

Among these rules, business rules are widely used in the industry and especially for critical business applications. Actually, as business rules have a business meaning and take the form of statements in a near natural language, manipulate them is very easy and do not require any programming skill or technical knowledge. Indeed, as statements in a near natural language, business rules can vary depending on the verbalization that has been arbitrarily adopted by the designer of the rule-based system. For these reasons, business rules can be easily used by domain experts to precisely define and maintain the problem-solving logic of an automated decision-making system and have been preferred by industrial, especially by banks and insurances.

2.2.1 Business rules: basic definitions

The most fundamental notion in Business Rule-based systems (BRBS) is the one of *business rule*. Business rules are commonly specified by means of an *ontology*, often represented by means of a *description logic* language. Each business rule can be seen as a statement, taking the form of a customized near natural language text, which defines a business aspect that allows to precisely describe the decision criteria and the actions to apply for a given situation of a decision process. Consequently, business rules provide a simple way to set the behavior of an intelligent system. For this reason, business rules are widely used in the industry and especially for critical business applications. Actually, as business rules have a business meaning and take the form of statements in a near natural language, manipulate them do not require any programming skill or technical knowledge. That is why, business rules can be

easily used by domain experts to precisely define and maintain the problem-solving logic of an automated decision-making system.

Definition 2.1 (Business Rule)**.** A *business rule* takes the form of a statement written in a business rule language and whose the logical structure is described as follow:

$$IF\ \langle premisses \rangle\ THEN\ \langle consequent \rangle$$

The premisses is a disjunction of conjunction of conditions and the consequent is a sequence of actions, thus the general structure is :

$$IF\ \langle c_1\ AND\ \cdots\ AND\ c_m \rangle\ OR \ldots OR\ \langle c_1\ AND\ \cdots\ AND\ c_n \rangle\ THEN\ \langle a_1; \cdots; a_n \rangle$$

The actions in the consequent are typically business rule language statements corresponding to variable assignments, and can involve various arithmetic operations whereas the conditions in the premisses are business rule language statements corresponding to expressions that can be evaluated.

Definition 2.2 (Eligibility and Triggering)**.** Based on the satisfaction of the conditions in its premisses, a rule is *eligible* for being triggered. When a rule is *triggered*, the sequence of actions corresponding to the consequences is executed.

We shall see later on the procedure governing the triggering of the rules.

Example 2.6. *(Example of business rule)*
A rule having the business rule language form:

> *if the score of the Borrower is higher or equal to* 10 *then set the rate of the Loan to (the score of the Borrower + the bonus of the Borrower) divided by* 100 *;*

and corresponds to the conditions-actions statement:

> *IF* $\langle c_1 \rangle$ *THEN* $\langle a_1 \rangle$*, where "*c_1*" refers to the condition statement "the score of the Borrower is higher or equal to* 10*" and "*a_1*" refers to the action statement "set*

the rate of the Loan to (*the score of the Borrower*+*the bonus of the Borrower*) *divided by* 100"

Definition 2.3 (Rule-based system)**.** A rule-based system is composed of : (1) a collection of business rules, (2) *business variables*, which may be input and output variables, and (3) an inference mechanism.

User interactions with such systems take the form of requests whose the content depends on the application.

Definition 2.4 (Request)**.** A *request* amounts to ask the value of a specific (output) variable.

In practice, the working of a such system consists in three steps: (i) the rule-based system receives a request, (ii) based on this request, it solves a specific problem and (iii) it returns the results obtained as outputs.

2.2.2 Basic architecture for rule-based systems

Business rule-based systems have been shaped by many influences that go from computing theory to psychology research. As previously described in the symbolic approaches, these systems are the result of efforts to apply general concepts from cognitive and computer sciences to the simulation of expertise. They mostly derive from the production system model used in automation theory where the main idea was to associate stimulus and responses under the form of IF-THEN propositions modeling the problem solving knowledge of a human expert Lucas and van der Gaag (1991), Ross (2003). In this perspective, computing theorists have found it convenient to describe all computational behavior in terms of state transition tables that define rules for moving between states. Each of such rules contains a small chunk of the domain knowledge that could be used to infer reasoning and solve a part of the problem in specific situations. Finally, using these rules and an inference mechanism was sufficient to mimic human expert problem-solving abilities. Thus,

the basic rule-based system was just a knowledge base and an inference mechanism but was too simplistic to be used in industrial purposes.

The usefulness of automated decision-making systems is related to their capability to guarantee strong run-time performances and their flexibility. Thus, the working of the system have to be optimized and its knowledge base shall be maintained up-to-date. To this purpose, the automated decision-making system requires a knowledge base editor. This component allows the human expert to explore and modify the knowledge base in a straightforward way. Finally, an interface is needed for providing interaction between the system and its users in a simple way. This interface is called the user interface. For all those reasons, more sophisticated architectures have been designed and reflect a global trend toward more consistent rule-based systems.

Today, most of business rule-based systems are complete computing systems and embed at least the essential components depicted in the figure Fig. 2.1 which describes a basic architecture of a rule-based system.

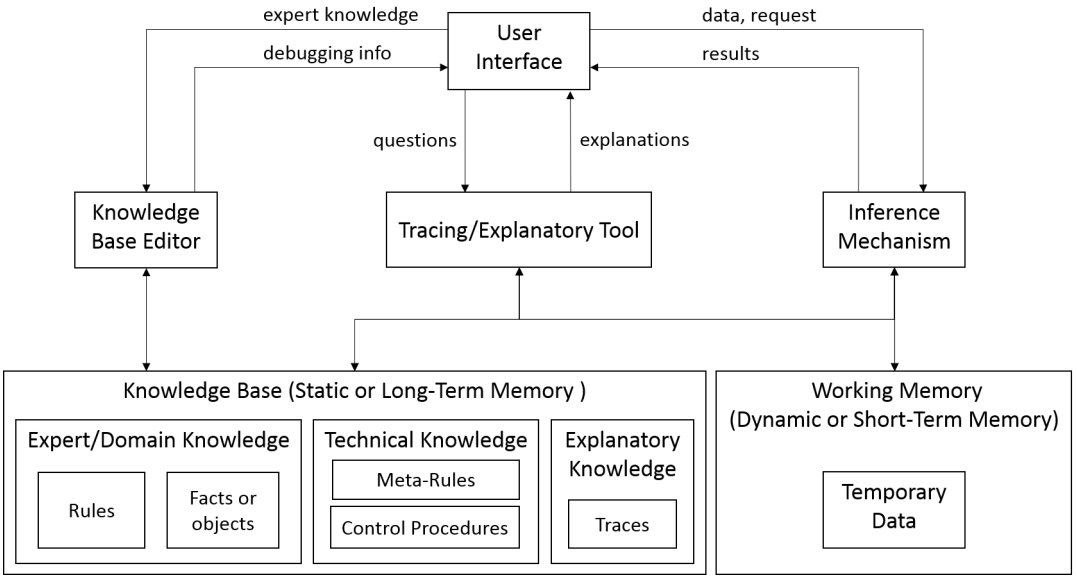


Figure 2.1: Basic Architecture: essential components of a Rule-Based System

We describe more precisely the essential components and their functions bellow.

2.2.2.1 The Knowledge Base

A rule-based system holds a collection of general principles and concepts that can potentially be applied to solve problems or answer the user requests. This static information is stored in a knowledge base which consists in a specific database representing a wide variety of knowledge and containing various kinds of data. For example, in the context of an application for banks and insurances, a user can ask the system if a particular loan request has been accepted. Then, Depending on the automated decision results, he may ask the system to explain its reasoning. In order to answer to the user request, both the inference mechanism and the explanation tool need general information that are contained in the knowledge base. Consequently, the design and the organization of a knowledge base is a very important point that requires special caution. Increasing the expressibility and the intelligibility of this knowledge base makes more efficient the knowledge exchanges between the experts and the rule-based system and facilitates the management (modification and extension) of the knowledge base.

In this perspective, a knowledge base is often decomposed into three kinds of knowledge depending on their role.

- The domain knowledge that encompasses facts and rules considered by the system for problem solving,
- The technical knowledge that describes control procedures, meta-rules and other knowledge required for the working of the inference mechanism,
- The explanatory knowledge that contains at least the traces of the automated decision taken by the rule-based system. Moreover, it may also include additional knowledge about the domain as justifications that could be needed to produce explanations.

In resume, a Knowledge Base, sometimes referred as Domain Knowledge, contains all the essential knowledge about the concepts and the heuristics used by the engine

to make inferences. It encompasses:

- the concepts manipulated in the domain: that can be represented as object classes,
- and the laws used in the domain to manipulate these objects: that take the form of business-rules.

However, characterizing more precisely the exact nature of the knowledge encoded in the knowledge base will be very important for us for, as noticed in (Chandrasekaran and Mittal, 1983), one very important aspect in the perspective of explanation is the "depth at which a rule-based system represents its knowledge and uses it to solve problems".

Deep *vs.* compiled knowledge. As knowledge-based systems, rule-based systems aim to represent enough knowledge about a domain and a problem to solve in it to automate the problem-solving. Moreover, as an expert system is commonly dedicated to a specific problem domain, it is only designed for making decisions in the specific problem context and thus presents its own features that meet the specific needs of a particular application. Consequently, the content, as well as the structure and the form of a rule-based system knowledge may change depending on its problem domain, on its users' needs and on the technical choices of its designer. For this reason, the knowledge embedded in such systems can highly vary from one to another. In the same way, this knowledge may be more or less precise or even represented at different levels depending on the human experts that provided the expertise used to solve the problem and depending how this expertise has been transcribed for being used by the system. In our explanation perspective, we would examine how the changes in the knowledge used by these systems to reason could change the quality of the information that they contain.

Intuitively, we feel that, in addition to affect the capability of a system to solve problems, the depth and the structure of a knowledge will affect the accuracy and the

quality of the information that could be extracted from it for explanation purposes. Thus, as the "depth of the knowledge represented in a rule-based system seems to have an impact on its capabilities, it could be wise to examine this measure/concept more precisely." In their contributions, Michie (1982) and Brown (1984) refer to "high road", "middle road" and "low road" approaches. In the same way, in its study, Hart (1982) makes a distinction between what he calls "deep" and "surface" systems that refer to similar concepts.

Surface systems (or *low road approaches*) have been mostly used for designing first generation expert systems. The knowledge they use has been compiled through experience in a highly refined form which is very concise and can be easily used by the system to automate its problem solving in an efficient. In practice, surface knowledge consists at best in a database of pattern-decision pairs that is eventually provided with a simple control structure allowing the navigation in it. Due to their compact form and their simplicity, these approaches are highly efficient but encounter practical issues in the variety of problems they can solve. Indeed, as their solving capabilities are limited to the pattern-decision pairs that have been explicitly described in the knowledge, they can only solve the specific cases for which their data base contain the related data associations. This limitation constrains their uses to the problems for which it is doable to describe all the possible cases and exposes them to "combinatorial explosion".

Conversely, *deep systems* correspond to high road approaches and directly use domain principles or models in their less refined forms. These principles and models come from a complete domain theory that has been picked up to be used in the expert system. In practice, such knowledge may contain first principles, basic relationships, and knowledge about functions that are applicable in a wide variety of situations. Nonetheless, there is no consensus on the content, the form and the structure that deep knowledge should have. Due to their generic knowledge about the domain, they can use it to solve more complex problems than the surface ones and do not encounter the same limitations. Their main weaknesses are their high complexity

and their inefficiency to solve simple problems.

Example 2.7. *A basic electrical problem:*

Imagine you have to analyze an electrical circuit and your first task consists in finding the current produced I through an ohmic device with a resistance $R = 1k\omega$ when you apply an electromotive force $U = 9V$. In this case, the system could address the problem with two different methods: (1) applying the Ohm's law ($I = U \div R$) to find a current $I = 9mA$ or (2) checking in a table to return the value of the current $I = 9mA$ associated to the couple of values $\langle R = 1k\Omega, U = 9V \rangle$. In this case, the method (1) which applies the Ohm's law to deduct the current through the ohmic device would rely on a "deep knowledge" and the method (2) would rely on a "surface knowledge". Nonetheless, as the notion of "deep knowledge" is subjective, someone could claim that there is deeper knowledge and that the system should rely on the generalized form of the Ohm's law using impedance to deal with non-ohmic device. The basic idea is more your knowledge about a domain will be deep, more the approach you'll apply will be general and will be able to deal with more cases. From this perspective a "deep knowledge" should rely on the more general principles you can express about a domain and which can be used to address the problem.

The fact is that these two kinds of systems have different utilities. The systems using surface knowledge provide an efficient way to solve quickly simple problems but encounter issues to deal with problems that have not been anticipated, while the systems using deep knowledge are less efficient but are supposed to handle any problem of the domain, even the more complex. In fact, deep and surface knowledge are relative terms, a deep knowledge may not be very deep (not based on quite first principles) but it can be considered as deep relative to some other pieces of knowledge which are more readily usable or refined. From this perspective, deep and surface knowledge represent the two extremes.

Between these two extremes, there are other approaches using compiled knowledge, called middle roads, which propose compromise solutions that compile less refined knowledge in order to be more general than the surface ones but less complete

than the deep ones. The underlying idea is to provide that use compiled knowledge that reconstruct and refine some chunks of the deep knowledge to apply them on more specific cases. This statements are depicted in Figure 2.2.

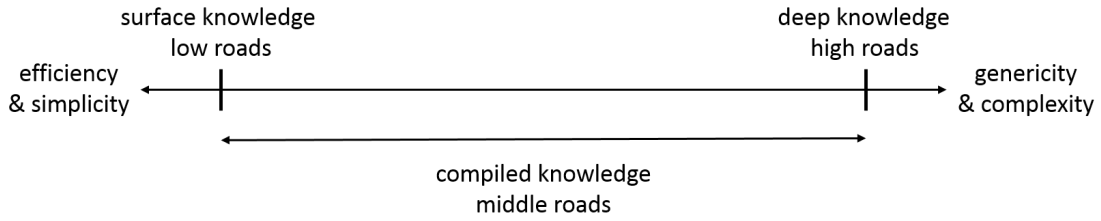


Figure 2.2: Compiled knowledge: from deep to surface knowledge

According to Anderson (1986), behind any compiled knowledge, there is a compilation process that aims to reduce the amount of knowledge used by the system during the problem-solving process. Moreover, as depicted by Pazzani (1986), such compilation of knowledge may also involve the addition of relevant pieces of knowledge which alter the problem solving process and even imply changes in the representation level of the source knowledge. For example, a chunk of knowledge can be changed from a deeper context-independent level to a more easily usable form that fit only to a specific part of the problem.

From this perspective, the knowledge compilation can be seen as the ability of a compilation process to refine the knowledge used to solve problem in order to enhance the performance of an expert system. Another important aspect to consider is the common idea that deep knowledge structures are needed for providing explanation. In fact, this idea relies on the fact that more the knowledge of a system will be refined less it will be able to provide useful information for building the explanation.

In the light of the above, we can see that the process of compiling knowledge for designing expert systems is quite near to the process of building knowledge for devising explanation features with the difference that the process of compiling knowledge is based on the deep knowledge and occurs before the process of building explanation knowledge that is based on this compiled knowledge. This means that

not enough information in the compiled knowledge systematically leads to a lack of information in the explanation knowledge. As a last resort this missing information could be obtained directly from deep knowledge when constructing the explanation.

One point worth mentioning is that a knowledge base is a static memory which provides general principles and concepts that are used by the inference mechanism but it is not sufficient to make reasoning about specific inputs. In this perspective, a dynamic memory, called the working memory, expand this knowledge base and allows the rule-based system to reason about specific data.

Moreover, the access and the maintenance of a knowledge base is often facilitated by the means of a Knowledge Base Editor that allows to manipulate the information contained in the knowledge base at high level.

2.2.2.2 The Working Memory

A rule-based system also holds a collection of specific details that apply to the current problem. This dynamic information encompasses temporary data corresponding to the current decision and includes details about the inputs, the progression of the reasoning process, the triggered rules and the outputs. A Working Memory that contains the facts base where each fact is a concept of the domain that has been instantiated with a defined value and represents a specific object. Its role is to hold these dynamic information in order to support the working of inference mechanism during the whole reasoning process.

2.2.2.3 The Inference Mechanism

An inference mechanism processes domain and technical knowledge to make deductive inferences about the temporary data contained in the working memory.

In rule-based systems, such a mechanism is typically referred as the *inference engine* or *rule engine*.

Inference engines have been used to deal with logical rules in the first expert

systems and work in a simple way. Logical rules are applied to the facts in the working memory and new facts are deduced. This process would iterate as each new fact in the working memory could trigger additional rules in the knowledge base.

A basic engine cycle has the three following phases:

(1) *Detection* (filter relevant rules): creation of the *conflict set*. Given the state of the working memory, an interpreter selects the executable [N: eligible?] rules in the knowledge base (ie. the business rules whose the conditions in the premisses are satisfied) and stores them into an agenda. This set of executable [N: eligible] business rules is called the *conflict set*.

(2) *Selection* (select the rule with the highest priority): conflict resolution and selection of a rule to be executed (control strategy). The technical knowledge is used to establish priority among the business rules.

(3) *Execution* (execute an instance of this rule): execution of the rule and modification of the working memory.

Based on the above description of an engine cycle, inference engines work primarily in the three following way. The forward chaining approach starts with the known facts and asserts new facts. This approach is based on the application of the modus ponens and makes deductive inferences (each inference involves an engine cycle). The backward chaining approach starts with goals and works backward to determine what facts must be asserted so that the goals can be achieved. This approach consists of a tree exploration and makes inductive inferences. A mixed approach that combines deductive and inductive inferences is sometimes used.

Contrary to the term inference engine which is clearly defined, the term Rule Engine is quite ambiguous in that it can designate any inference mechanism in a system that uses rules, in any form, that can be applied to data to produce outcomes and often refers to more recent engines that manipulate more complex rules (like fuzzy or business rules) and work on the same principles but provide a richer set of mechanisms to work with.

In practice, an engine (rule engine or inference engine) is used to simulate the expert reasoning by evaluating the state of the working memory, looking for some applicable rules, applying one rule instance on the working memory and then updating the working memory and repeating the process until the final decision has been determined.

2.2.2.4 The Tracing Tool

As industrial applications of automated decision making require to be monitored, it is necessary to keep track of the decisions taken in order to understand them and ensure their transparency. Indeed, monitoring the decisions taken by a rule-based system allows the experts to understand, accept and optimize its reasoning. In this perspective, tracing tools are essential for most of rule-based applications, according to Darlington (2013), such components are essential to decision understanding, monitoring and acceptation. It means that an automated decision-making system needs to embed a tracing tool, which typically simply applies a set of observers to trace basic information about specific data modifications or operations occurred during a decision. As we will later, this is not sufficient though if the objective is to explain the decisions.

2.2.2.5 The User Interface

A user interface is a means of communication between a rule-based system and its users. Thanks to this interface, the users interact with the rule-based system in a straightforward way. It is an important component to make a rule-based system because the more ergonomic a user interface is, and the more it enhances the ease of use of the rule-based system.

2.3 The IBM business rules management system

Business Rule Management Systems, often abbreviated as BRMS, have emerged from the convergence of production systems mainly used as reasoning tools for

problem-solving and business rules essentially seen by their users as a knowledge representation tool.

In fact, a BRMS consists in a software that provides a way to manage variety and complexity of the decision logic used by operational systems within organizations. The decision logic takes the form of a declarative language, expressed through business rules as described in the previous section, that enables the description of business policies, business requirements and conditional statements that are used to determine the business actions that occur during an operational decision. By using business rules, an operational decision associated to them can be defined, deployed, executed, monitored and maintained separately from its core application code.

In this perspective, BRMS provide robust platforms dedicated to the management of business rule-based systems and allow the companies to increase the readability of their knowledge base, reduce operational and maintenance costs and react more quickly to the market changes.

The field of business rule management systems is characterized by a number of normative, open source or proprietary technologies and languages that makes them far efficient than classical approaches. The key players include IBM (IBM Operational Decision Manager), Bosch Software Innovations (Visual Rules), Progress Software (Corticon), Oracle (Oracle Business Rules), Red Hat (JBoss Drools), SAP (SAP NetWeaver DSM) and FICO (FICO Blaze Advisor).

In the context of this thesis about automated decision-making and explanation, we are interested by the business rule management system provided by IBM, called IBM Operational Decision Manager. This product is the evolution of the well known ILOG JRules. As IBM Operational Decision Manager separates decision management from application code, the business experts can define and manage the business logic without the support of IT experts. By adopting this way of managing decisions, there is a huge reduction in the amount of time and effort that is required to update the business logic in production systems. Thus, by using this approach, organizations increase their ability to respond to changes in their business

environment.

For a better understanding of how IBM Operational Decision Manager works and what are its specificities, we introduce some important concepts and definitions thereafter.

2.3.1 Hierarchy of decision service in IBM ODM

The business rule applications designed by using IBM ODM rely on two key aspects:

- the establishment of a *hierarchy* among the elements of a business rule project. The hierarchy of a decision service (i.e. rule project) allows the decision service behavior to be governed and deployed consistently.
- the definition of the *vocabulary* that is used for authoring business rules. This vocabulary is used to refer to the business objects and functions that are manipulated during the decision process.

Definition 2.5 (Ruleset)**.** A ruleset is a set of coherent business rules, in the sense that each set represents a specific aspect of the decision logic and takes the form of an executable *decision unit*. A ruleset uses input and output parameters to pass data to and from the client application. Each ruleset must have a unique signature that describes the sets of input and output parameters that are used by this specific ruleset to communicate with the rest of the client application. In a decision service, the decision operations define the content and signature of each ruleset.

Each of these rulesets having its own dedicated resources and form a *decision unit* that has a all the necessary materials to be parsable and executable by the rule engine. These *decision units* can then be referred in *tasks* which are responsible for splitting the reasoning process. More concretely, each *task* embeds an independent chunk of the decision-making.

Based on that, a decision process encompasses one or more tasks whose the

executions result in the different reasoning steps of the corresponding decision. The orchestration of these reasoning steps is managed by the *ruleflows*.

Definition 2.6 (Ruleflow)**.** A ruleflow specifies a flow of execution of the tasks. In this sense, the ruleflow defines the problem-solving strategy. The operations permitted in a ruleflow are: (1) *branches* and (2) *fork and joins*. A (1) *branch* is an operation that organizes conditional transitions between the tasks and a (2) *fork and joins* splits the execution flow in several parallel paths (without using conditional test) and then combine all the transitions created when the parallel paths are completed.

Example 2.8. *In Figure 2.3, there is a good example of branch operation from the task "eligibility" to the task "pricing" and the end node. The condition associated to this task, labeled rental agreement accepted, checks is the approval status of the loan is true after the execution of the task "eligibility". If it is, then the task "pricing" will be next, if it is not the case, then the execution will be ended and the loan request will be rejected.*

Thus, *ruleflow* are a way of controlling rule executions. They are made of linked tasks that contain the instructions for which rules to execute and in what order. The links between the tasks are called transitions. A ruleflow specifies how tasks are chained together: how, when, and under what conditions they are executed. It also deals with ruleset parameters that are variables used to transfer information from one ruleflow task to another one, to determine which path to follow through the transitions and to transfer information between the ruleflow and your application. For example, you can transfer a status variable, such as isEligible, to determine which task to go to next. The diagram presented in Figure 2.3 shows the main parts of a ruleflow where (1) represents the start node, (2) a task, (3) a transition and (4) the end node:

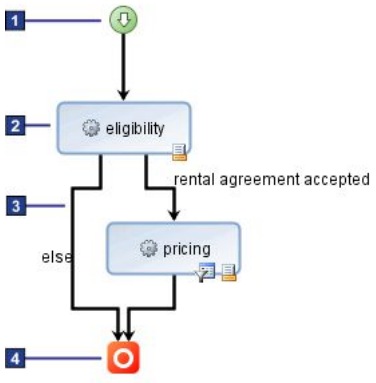


Figure 2.3: Example of ruleflow for a loan application usecase (source)

2.3.2 IBM Operational Decision Manager: platform and architecture

The BRMS called IBM *Operational Decision Manager (ODM)* is provided by IBM and can be seen as an environment for designing, developing, deploying and maintaining business rule applications. In practice, IBM Operational Decision Manager simply consists in a platform which includes a set of tools allowing to produce and maintain various business rule applications that are called decision services. This platform can be accessed by two ways:

- A *decision server* allowing IT specialists to design, author and test the business rules (functional requirements), and
- A *decision center* allowing business users to author, manage, validate and deploy business rule services without specific knowledge in informatics (business requirements).

The scheme depicted in the figure 2.4 gives a global view of this platform.

Whereas the decision server focuses on the functional requirement and provides a set of components that allows IT developers to design, author and test the business rules by the way of business rule projects, the decision center stores these business rule projects to let the business users manage decisions that are directly based on organizational knowledge and best practices with limited dependence on the IT

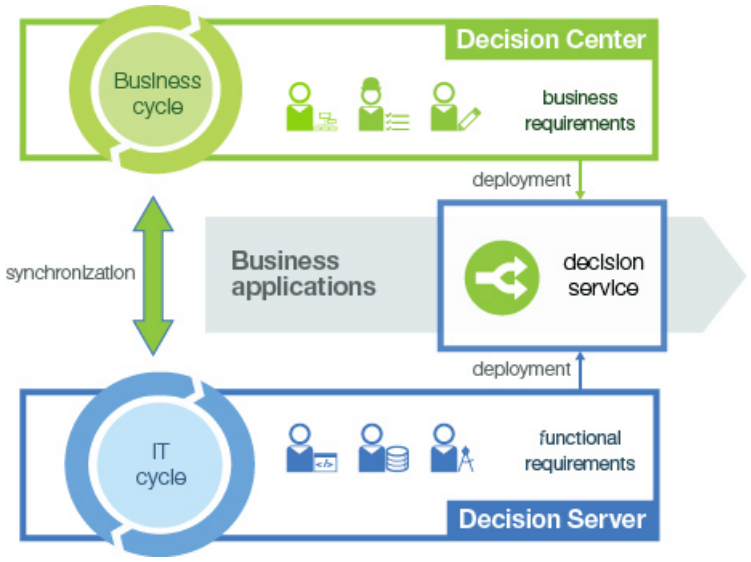


Figure 2.4: Operational Decision Manager Global View (source)

department.

This means that two cycles can evolve in parallel:

- The *IT cycle*: it encompasses the actions which consist of developing and maintaining the infrastructure through the *decision server*, and
- the *business cycle*: it refers to all the actions that consist of the definition and maintenance of the decision logic. This operations are supported by the *decision center*.

In practice, the business cycle is supported by the infrastructure provided by the IT teams and allows distributed business teams to collaborate through a web-based environment to create and maintain the decision logic.

One of the strength of this approach is that decisions can evolve as required by the business context without putting an extra load on the development of the business rule service. Each time a business rule service evolves, the decision management environment synchronizes with the development environment. With this separation, decisions and application architecture can be managed asynchronously. For example,

application developers can develop a new application version in response to changing application infrastructure and core business requirements. At the same time, policy managers can work on new decisions that are delivered in response to an evolving market, changing regulatory environment, or new patterns of events.

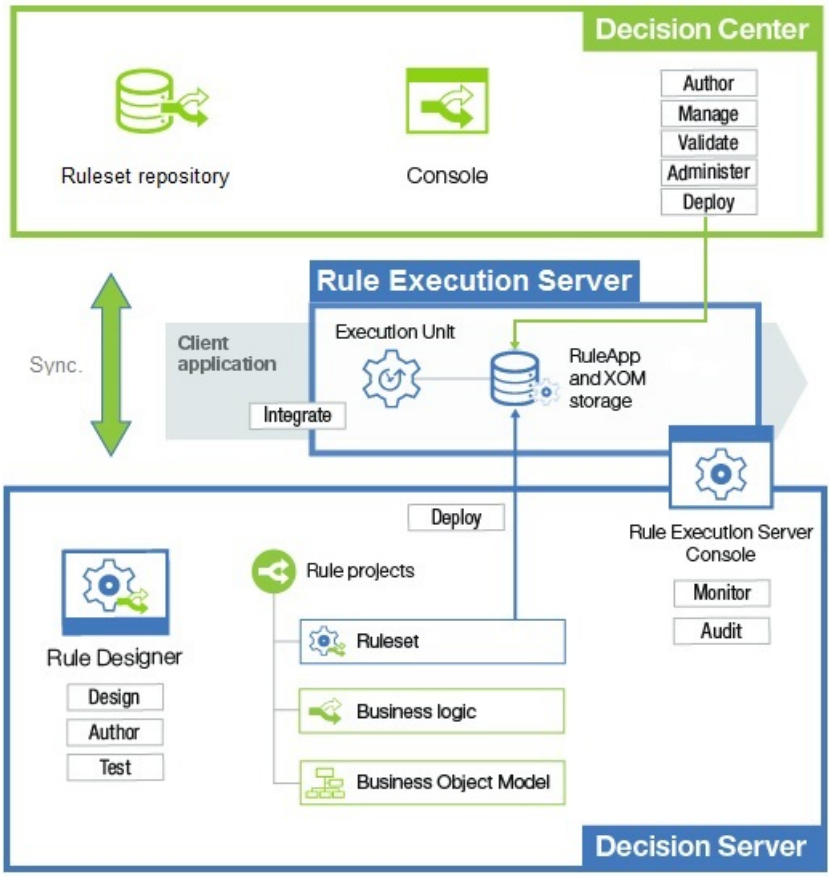


Figure 2.5: IBM ODM global view (derived from (source 1) and (source 2))

The upper part of Figure 2.5 gives a global view of the decision server which supports the development environment. As we can see on the scheme, the decision server provides the development and runtime components that are involved in the development and maintenance of rule-based solutions used to automate the response of highly variable decisions required by client applications.

To this end, the decision server uses two components:

- the *Rule Designer*: this component consists in an Eclipse-based development environment that allows to design, author, test and deploy the business rule applications. The design of a business rule application encompasses several aspects. The implementation of the necessary infrastructure for editing rules and producing the set of rules that are used as decision units. Moreover, the development of an Execution Object Model (XOM) aims to define program against which the business rules are run and the development of a Business Object Model (BOM) aims to define the elements and relationships in the vocabulary. A business rule language is then built by mapping the elements of the Business Object Model (BOM) to those of the Execution Object Model (XOM). Based on that, the configuration and the customization of tests and simulations allow to set up business user validation tools.

- The second component is the *Rule Execution Server*. This component is responsible for providing the runtime environment for running and monitoring decision services.

In short, the decision server provides the runtime (in the *Rule Execution Server*) and development components (in the *rule designer*) that are required to automate the response of highly variable decisions that are based on the specific context of a process, transaction, or interaction.

In addition to the decision server that allows the IT teams to work on the functional aspects of a decision project, the decision center provides an environment dedicated to the management of the business logic by the business experts. The lower part of the scheme presented in figure 2.5 gives a global view of the decision center by illustrating the components that support this decision management environment.

The decision center includes a rule repository and some collaborative web consoles allowing business users to author, manage, validate, and deploy rules. Thanks to the web console, business users can author business rules used in decision services. Testing and simulation features that enable business users to validate the behavior

of rules are also included in the decision center because business users must be confident that business rules are written correctly and that any update does not break the business logic encapsulated in the ruleset. When a business application is ready for production, business users can deploy decision services directly from the business console and then manage them.

As previously explained, the business rules used in IBM Operational Decision Manager are based on two essential concepts: the *Business Object Model* abbreviated as BOM and the *Execution Object Model* abbreviated as XOM.

The BOM is used to make business rule editing user-friendly by providing tools to set up a natural language vocabulary that allows business experts to describe their business logic in a *business rule language*. This *business rule language* relates on a *business ontology* which can be verbalized in more than one expression in order to provide a richer language or in order to deal with several locales including English, Chinese or Spanish. The BOM contains the classes that rule artifacts act on and consists of classes grouped into packages where each class has a set of attributes, methods and, possibly, other nested classes. Thus, the BOM constitutes the basis for the vocabulary used in business rules and takes the form of an object model which is similar to a Java object model. The figure 2.6 illustrates how the vocabulary used in a business rule is connected to the business object model.

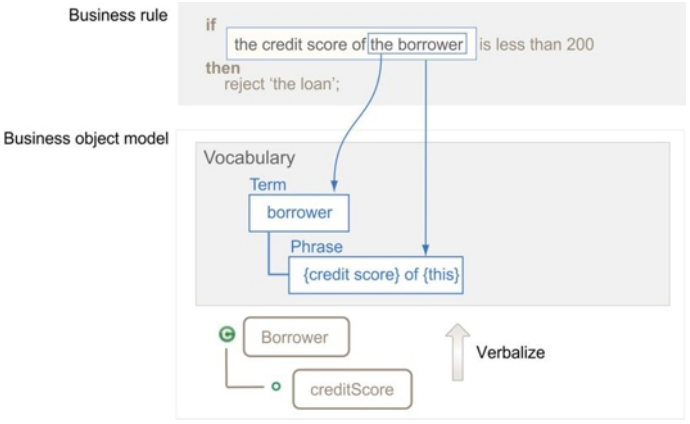


Figure 2.6: Business Objet Model and Vocabulary (source)

Whereas the BOM is an object model used to deal with the vocabulary aspect, the XOM is an object model used to deal with the functional aspect that determines rule execution. The implementation of this model relies on classes that are expressed in Java or by an XSD schema as depicted in the figure 2.7. Thus, this is the XOM that allows the rule engine to access the application objects and methods.

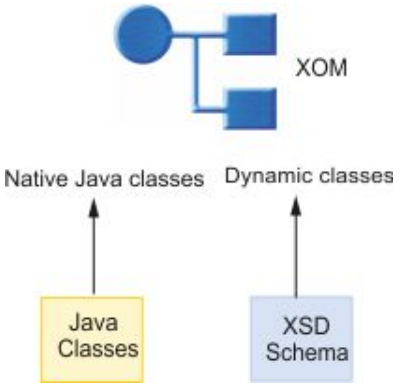


Figure 2.7: Execution Object Model (XOM) (source)

Based on that, the business rules are written against the Business Object Model (BOM) by human users, then translated into the ILOG Rule Language (IRL - when using the classical rule engine) / Advanced Rule Language (ARL - when using the decision engine) and run against the XOM (the element which is manipulated by the rule engine). Moreover, the elements in the BOM correspond to those in the XOM and are mapped together based on the "BOM to XOM mapping file" (.b2x) that defines the correspondence between the business object model (BOM) and the execution object model (XOM) used at runtime. The scheme presented in figure 2.8 described these statements.

Based on the previous definitions, it is possible to complete the picture of the content of a decision service. The figure 2.9 provides a view of the elements encompassed in a decision service.

In it, a *decision service* is composed of one or more rule projects that can be used together and establishes a hierarchy among them. Each of these rule projects can contain rulesets, variables, business object model, execution object model, bom

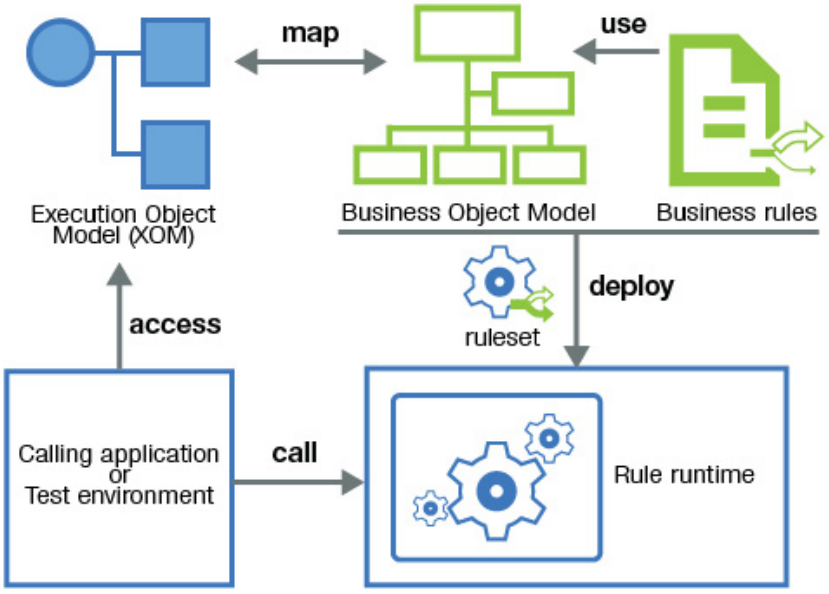


Figure 2.8: Relation between BOM and XOM (source)

to xom mapping files and decision operations that can be used when the decision service is running. The figure 2.10 shows an example of a decision service structure.

Such decision services can then be used by rule-based applications to automate decisions.

2.3.3 The rule engine of IBM ODM

Based on the elements contained in such decision services, the rule engine used by IBM ODM evaluates the rules against the application objects and executes them when appropriate. This mechanism operates on Java platforms and has three modes of execution (RetePlus, Sequential, FastPath). While the details of the implementation might not be really important (how is the "priority" property determined etc.), it might be interesting to understand exactly what they are and how they are different.

There are mainly two classes of algorithms.

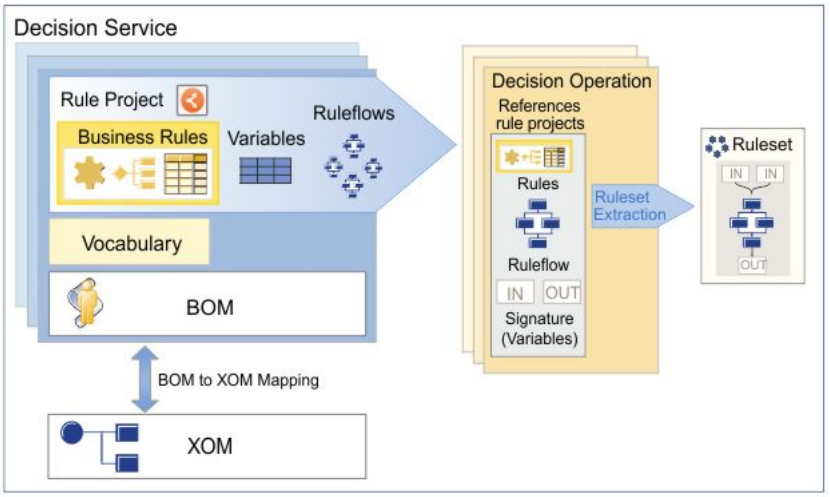


Figure 2.9: Components of a Decision Service (source)

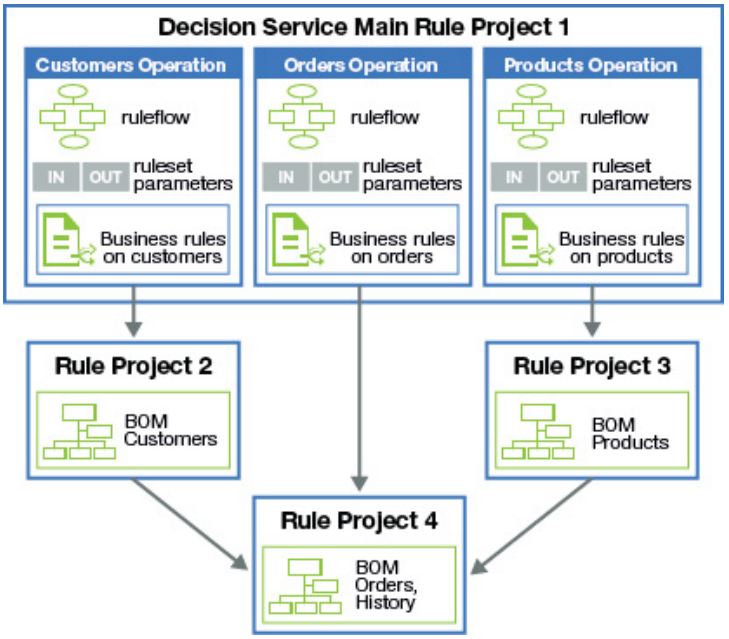


Figure 2.10: Example of Decision Service architecture (source)

- *inferential* : An inferential algorithm rely on an inference mechanism that automatically resolves rule ordering problems. The Rete-algorithm is the most known example of inferential algorithm.

- *non-inferential*: A non-inferential algorithm simply rely on explicitly specified

sequencing of rules and rule sets. Actually, non-inferential engines are quite successful because in many practical applications manual rules sequencing is a sufficient and frequently preferred option. The IBM Sequential algorithm is a good example of non-inferential algorithm.

Let us describe in detail the three execution mode.

A basic version of the first execution mode, called the RETEPLUS mode, is presented in Algorithm 1.

Algorithm 1 RetePlus

1: Find all the eligible rules in the ruleset (evaluate all the conditions of all the rules by using a pattern matching approach based on the RETE algorithm)
2: Add them to an Agenda which order them dynamically by using their "priority" property
3: Select the first Rule (the highest priority Rule) and fire it (run the actions part of the rule). It becomes uneligible until one of its conditions becomes false.
4: Repeat from 1. until there are no more eligible Rules or one of the actions ends the execution.

The second execution mode is the SEQUENTIAL mode. The important difference with RETEPLUS is that each Rule is only evaluated once. Thus, it is much simpler, has no risk of looping and is described as presented in Algorithm 2.

Algorithm 2 Sequential

1: Order the Rules in the RuleSet according to their priority property
2: **for** each rule **do**
3: evaluate its conditions
4: **if** the rule is eligible **then**
5: fire this rule
6: **end if**
7: **end for**

The third execution mode is the FastPath mode. The important difference with the Sequential mode is that actions from a Rule cannot influence the eligibility of another Rule from the same RuleSet. It can be described by the process described in Algorithm 3.

Example 2.9. *To highlight the differences between each algorithm, let's consider a*

Algorithm 3 FastPath

```
1: Order the Rules in the RuleSet according to their priority property
2: for each rule do
3:    if the rule is eligible then
4:       add the rule to the list of eligible rules
5:    end if
6: end for
7: Fire all the eligible rules in the order of the list
```

very simplified Model with a small RuleSet. The only object is F. It is an integer with an initial value of 10. It is both the input and the output. There are only four Rules, with respective priority 1, 2, 3 and 4: Rule 1: IF $lessthan(F, 4)$ *THEN* $set(F, 0)$*, Rule 2: IF* $greaterthan(F, 9)$ *THEN* $set(F, 0.5{*}F)$*, Rule 3: IF* $lessthan(F, 9)$ *THEN* $set(F, 3)$*, Rule 4: IF* $greaterthan(F, 4)$ *THEN* $set(F, 3 * F)$*.*

The Table 2.1 shows the different rule engine execution modes applied to a simple usecase using the rules below.

2.3.4 Applications

The systems we are interested in are widely used in the fields of bank, insurance and health and can perform functions like loan applications, fraud detection, pricing or even medical diagnosis. These examples are just a few of the wide range of applications that are solved using rule-based systems. The last decades have demonstrated the commercial viability as well as the strong maintainability of these systems and so they have now lots of applications. In particular, the Business Rules Management System of IBM is used by a huge variety of clients and partners. For example, most frequent business usecases of ODM are "credit and loan approvals", "claims processing", "Underwriting", "compliance and reporting", "dynamic pricing and bundling", "fraud detection", "eligbility determination", "cross-sell, up-sell product recommendations", "customer insight and loyalty programs", "customs and border control", "passenger profiling" and "medical decision support"[1] In this document we do not

[1] As reported in *(http://www-01.ibm.com/software/decision-management/operational-decision-management/scenarios/,https://developer.ibm.com/odm/2015/06/02/odm-business-rules-samples-for-healthcare-and-government/)*.

RetePlus
The RetePlus goes: Iteration 1: F = 10 Rules eligibility: 1: $(F < 4)$ = false, 2: $(F > 9)$ = true, 3: $(F < 9)$ = false, 4: $(F > 4)$ = true First eligible Rule: 2. Actions: F := 0.5*F = 5 Iteration 2: F = 5 Rules Eligibility: 1: $(F < 4)$ = false, 2: $(F > 9)$ = false (reset eligibility), 3: $(F < 9)$ = true, 4: $(F > 4)$ = true First Eligible Rule: 3. Actions: F := 3 Iteration 3: F = 3 Rules Eligibility: 1: $(F < 4)$ = true, 2: $(F > 9)$ = false, 3: $(F < 9)$ = true (ineligible due to having fired), 4: $(F > 4)$ = false First Eligible Rule: 1. Actions: F := 0 Iteration 4: F = 0 Rules Eligibility: 1: $(F < 4)$ = true (ineligible due to having fired), 2: $(F > 9)$ = false, 3: $(F < 9)$ = true (ineligible due to having fired), 4: $(F > 4)$ = false First Eligible Rule: None. Actions: Stop Execution RESULT: F = 0
Sequential
The Sequential goes: Rule 1: F = 10, Eligibility: $(F < 4)$ = false, Actions: None Rule 2: F = 10, Eligibility: $(F > 9)$ = true, Actions: F := 0.5*F = 5 Rule 3: F = 5, Eligibility: $(F < 9)$ = true, Actions: F := 3 Rule 4: F = 3, Eligibility: $(F > 4)$ = false, Actions: None RESULT: F = 3
FastPath
The FastPath goes: F = 10 Rules Eligibility: 1: $(F < 4)$ = false, 2: $(F > 9)$ = true, 3: $(F < 9)$ = false, 4: $(F > 4)$ = true Eligible Rules: Rule 2: F = 10, Actions: F := 0.5*F = 5 Rule 4: F = 5, Actions: F := 3*F = 15 RESULT: F = 15

Table 2.1: Usecase: Rule Engine Execution Modes

focus on a specific field or application but rather look for generic approaches that are independent of the application domains.

In bank and insurances, all decisions taken by the rule-based systems must be stored and available at any time for monitoring because of legal constraints. More broadly, to be acceptable, maintainable and commercially viable, they need more transparency. Indeed, the content and the form of the information provided to the user must be adapted to his individual requirements and depending on the application context a wide range of users can be involved. For all these reasons, automated decision making systems require the capability to explain their decisions.

2.4 Discussion and conclusions

The knowledge-based system we consider belong to the class of symbolic expert systems, are used to automate business decisions and can be built by using a Business Rule Management Systems (BRMS) called IBM Operational Decision Manager. IBM Operational Decision Manager allows to build rule-based sytems that automate problem solving with the knowledge of the "how" and according to Hayes-Roth (1985b) they provide a means for capturing and refining business expertise and are proving to be easily maintainable and commercially viable.

Nonetheless, as the decision logic of these systems can be strongly complex, their decisions can seem unclear and hard to accept. That is why they require a way of making their decision logic more transparent and understandable by their users. That is why, in order to help their decisions to be considered as useful and acceptable but also to monitor and continuously improve their quality, rule-based systems must be able to explain their knowledge of the domain and the reasoning processes they employ to produce results and recommendations and using a causal model can be a starting point to do that. Moreover, as IBM's clients need to use these systems in real time, our system will have strong constraints on performance. Consequently the explanation capabilities we have to design must not be memory-consuming or greedy for computation time especially at decision time. In the next chapter we will

detail further these motivations for explanation in rule-based systems and present more broadly the concept of explanation, and its application to rule-based systems.

Chapter 3

An overview of explanation in rule-based Systems

In this chapter, we start by discussing the need for explanation in general and in rule-based systems in particular (Section 3.1), then we explore the notion of explanation (Section 3.2), present an overview of the current proposals in the literature (Section 3.3), we give an historical overview of proposals for rule-based systems (Section 3.4), and detail in particular the notion of causality (Section 3.5) upon which our proposal is built. Finally, we conclude this chapter by emphasizing the requirements we have for the envisioned IBM ODM feature (Section 3.6).

3.1 The need for explanation

The following quote, taken from the website https://www.reasoncode.org/[1], illustrates a typical kind of explanation provided when someone obtains his *credit score* in the U.S.:

> "No matter where you get your score, the documents that accompany it will include up to four or five statements explaining why your score wasn't higher. These statements are called reason codes and sometimes go by several other names. Some people call them score factors; others call them adverse action codes. They're all the same thing.

[1] Retrieved on the 2017-09-26.

> The next time you see your credit score, regardless of where it comes from, look for the reason codes. They'll be worded and displayed something like this:
>
> Your Credit Score Is: 705
>
> - 32: Balances on bankcard or revolving accounts too high compared to credit limits
> - 16: The total of all balances on your open accounts is too high
> - 85: You have too many inquiries on your credit report
> - 13: Your most recently opened account is too new
>
> "

Citizens concerned by algorithmic decisions may soon be allowed to ask justifications. Recent regulations, like the *General Data Protection Regulation*, (GDPR) have put forward the topic of explanation as a hot topic in A.I. Indeed, by explicitly stating a "right to explanation" for algorithmic decisions, they constitute a challenge for many A.I. systems, which have no built-in explanatory feature. In the words of Goodman and Flaxman (2016) *"In its current form, the GDPR's requirements could require a complete overhaul of standard and widely used algorithmic techniques. The GDPR's policy on the right of citizens to receive an explanation for algorithmic decisions highlights the pressing importance of human interpretability in algorithm design"*. While there is some debate as to how bidding such regulation would be (Wachter et al., 2017), there is clear citizen concern that needs to be addressed.

However, there are several other reasons why we want to equip systems with explanatory capabilities (such objectives may change depending on the application). Tintarev and Masthoff (2007) identified (in the context of recommender systems) the most important objectives that should be considered when designing an explanation.

Aim	Description
Transparency	Explain how the system works
Scrutability	Allow users to tell the system it is wrong
Trust	Increase users condence in the system
Effectiveness	Help users make good decisions
Persuasiveness	Convince users to (try or buy)
Efficiency	Help users make decisions faster
Satisfaction	Increase the ease of usability or enjoyment

Table 3.1: Explanation Aims Table (Tintarev and Masthoff, 2007)

From this, the need for explanation in our industrial context is clear: Business Rule Management Systems have been widely adopted by financial services and public organizations to process, manage and record their decisions for later reference. More specifically, the decision systems provided by IBM are mainly used for commercial application, their acceptance and continuous improvement is a strong constraint. Moreover, as these systems are often used in the fields of banking and insurance, their decision have to be highly deterministic and clear. Large organizations have to serialize and store *billions* of decisions in "decision warehouses" for legal or analytic purposes. The ultimate objective of such tools is to present clear and precise explanations to business analysts, citizens and consumers. Delivering such explanation capabilities means that all the useful and necessary information about each decision is traced.

But is there an actual need to explain *rule-based systems*? After all, such systems symbolically encode expert knowledge which is quite interpretable. In fact, this question emerged at the very beginning of the history of expert system: while MYCIN had been equipped with basic explanatory capabilities allowing to trace back the reasoning steps of the system, using it in practice in a tutoring setting turned out to be a much more challenging task than expected:

> *" The seemingly straightforward task of converting a knowledge-based system into a computer-aided instruction program has led to a detailed reexamination of the rule base and the foundations on which rules are*

constructed, an epistemological study. "

In this chapter we will see why this is the case, and what solutions can be put forward. But before we get into these details, let us clarify what is meant by 'explanation'.

3.2 Philosophical background on the theory of explanation

The philosophy of science presents formal descriptions and requirements of the relationships entailed by explanations that differ from common understanding and that give us first basic knowledge about the very notion of explanation. In this context, it is interesting to look at the past two centuries of philosophical discussion that have produced a complex set of theories and models of explanation from which Woodward (2014) extracted five major types, also described by (Hovorka et al., 2008) in Table 3.2:

Given that, the common traits of these models and theories is that an explanation can be seen as a statement that increases somebody's understanding about something by providing him further information. Moreover, with the knowledge of these theories and models, we are able to extract some important elements we should consider when constructing an explanation.

An explanation is based on two kinds of elements:

- the *explanandum*, which is the phenomenon that needs to be explained
- and its *explanans*, which constitute the explanation content of that phenomenon.

The explanans contain:

- the cause, which is a set of elements compatible with the explanandum, required and sufficient to enable its to occur and also minimal.
- the causal information, which described relationships between the elements of the cause and the explanandum.

- some descriptive information, which provide general information about concepts/elements of the causes (description, taxonomy...)

Another specific aspect of the explanation is its rendering. A well exploitation of the explanans is needed to render an explanation which shapes to the aims, the user and the context. Moreover, we consider that the generality and the objectivity of the theory and the impact on the understanding as the essential requirements that our formal theory of explanation should address (Friedman, 1974). Some reasearchers attempted to provide a general definition of the explanation in a causal context. In particular, Halpern and Pearl (2001b) stated that a general explanation of an explanandum φ has the form (Ψ , $\vec{X} = \vec{x}$), where

- Ψ is an arbitrary formula in its causal language which consists of some causal information
- $\vec{X} = \vec{x}$ is a conjunction of primitive events representing the cause of the explanandum φ.

Moreover, faced with the variety of explanatory theories and models, Johnson and Johnson (1993) recommend the use of an unifying theory that provides the basis for judging the quality of an explanation. The proposition of Johnson and Johnson (1993) is followed by a great deal of empirical researches, for instance as in (Gregor and Benbasat, 1999), describing the factors affecting explanation usage linked to their theoretical foundations. In fact, this unified theory developed by Johnson and Johnson (1993) and then examined Gregor and Benbasat (1999) mainly use four components which are (i) the cognitive effort perspective proposed by Payne et al. (1993), (ii) the production paradox discussed by Carroll and Rosson (1987), (iii) the Toulmin's theory (Toulmin, 1958) and (iv) the ACT-R theory of skill expertise acquisition defended by Anderson (1990) and Ausubel (1985).

An important point to note here is that all these processes of providing explanation strongly depend on the aims that motivate their presence. We shall now focus on the aims and explanations that may be useful in rule-based systems.

3.3 Explanations in rule-based systems

Friedrich and Zanker (2011) suggest to categorize explanation depending on three dimensions which whose the first one refers to the nature of the exploited information (user model, recommended item, ...) , the second one refers to the paradigm used by the system (collaborative, content-based, knowledge-based) and the last one refers to the kind of reasoning model (white box explanations or black box explanations). Whilst Gregor and Benbasat (1999) categorize an explanation based on three criteria: (1) its content type, (2) its presentation format and (3) its provision mechanism. The (1) content type corresponds to what is usually referred as "explanation type" or "question type". For example, the content type can be either trace/line of reasoning, justification/support, control/strategic or terminological whereas the (2) presentation format refers to the shape of the presented information. For example, the presentation format can take the form of a simple and in this case it is classified as text-based but it can also provides graphical other other ways to deliver the information like graphics and in this case it is classified as multimedia. The last way to classify the explanation is to look at its (3) provision mechanism which can be classified as either "user-invoked" or "automatic" or "intelligent".

We chose to present explanations along four dimensions. As we shall see, some of them are not independent; but they nevertheless to intuitive ways to approach the notion of explanation.

- *timing*– explanations can be given before or after the decision or recommendation took place;

- *question types*– in Rule-Based Systems, the explanation comes down to a problem of question-answering where the most frequently asked questions have been compiled to identify the most important categories of questions for which an explanation was required and worthy.

- *content types*– the actual content of the explanation, and what it refers to.

- *context sensitivity*– whether the explanation

3.3.1 Categorization by temporal context / explanation orientation

A natural way to categorize explanation consists in looking at the explanation availability time. In fact, depending on constraints related to the resources used and on the question there are three possible cases. For an explanation using only domain knowledge, there is no temporal constraint and so the explanation to a question regarding some domain knowledge can be generated at anytime. But some explanations incorporate knowledge related to specific decisions. For these explanations there are two hypothetical situations: the explanation is required *during* the decision process or the explanation is generated *after* the decision process. In the literature, this notion has been discussed in terms of explanation orientation. From this perspective an explanation required during a decision process is referred as *feedforward* because it looks ahead to provide information about what the system is doing while an explanation generated after the decision has been taken is referred as *feedback* because it looks behind to explain what happened. Obviously, a generic explanation about some terms or concepts used by the system does not have any specific orientation and can be used either in feedforward or in feedback.

In the light of the above and based on the previous work on explanation orientation done by Arnold et al. (2006) and Dhaliwal (1993), looking at the explanation orientation criterion allows to categorizes the explanation in two distinct classes whose function can change depending on the content type of the explanation.

- Feedbacks can be seen as post-advice explanations and usually provide a record of problem solving action to enable the user to see how a conclusion was reached when the data has been completely input. These explanations are more likely to be used by experts than novices.

- Feedforwards can be seen as pre-advice explanations. They provide a means to find out why a question is being asked during a consultation and are mostly not

specific to any particular output case. These explanations are often used by novices who want learn about the domain or clarify some information they may misunderstand about the nature of a question or a term used by the system but they are seldom used by experts.

3.3.2 Type of questions

As the explanation comes down to a problem of question-answering where the most frequently asked questions have been compiled to identify the most important categories of questions.

- *How / Descriptive Explanation*: These explanations answer to a "How-Question" and are process-driven. They are used to allow the user to better understand the methods used by the system to achieve its goal. This can be done by displaying the execution steps and events of a decision process which lead to the decision output. As these explanations describe decision processes, they mainly use trace-based information to provide some schemes, graphs or text that clarifies the decision process. The main aim of these explanations is to provide more transparency and scrutability.

  Example 3.1. *Answer to a "How-Question" from MYCIN (Wallis and Shortliffe, 1977):*

  [HOW was established that the aerobicity of ORGANISM-1 is facultative?]

  The following were used:
  [3.1] RULE027 indicated there is strongly suggestive evidence(. 8) that the aerobicity of ORGANISM-1 is facultative
  [3.2] RULE027 indicated there is weakly suggestive evidence(. 2) that the that the aerobicity of ORGANISM-1 is anaerobic
  Since this gave a cumulative CF (.8) for facultative, and(.2) anaerobicity has been established that the aerobicity of ORGANISM-1 is facultative

- *Why / Reason-Happening Explanation*: These explanations answer to a "Why-Question" and give more information to the user about why a certain result has been obtained. The idea behind this explanation is to provide enough information about the behavior of an expert system to justify its decisions. These explanations are commonly constructed by using a backward approach which provides the genuine causes, some causal information and further knowledge about the domain to understand the explanandum.

  Example 3.2. *Answer to a "Why-Question" from REX (Wick and Thompson, 1992):*

  [Why erosion was the cause of the excessive load?]

  I attempted to find the cause of an excessive load on a concrete dam. Based on the broken pipes in the foundation, the sliding of the dam, the uplift pressures, and the slow drainage, I was able to make an initial hypothesis. In studying causal relations, l found that the erosion of soil from under the dam would cause broken pipes. resulting in slow drainage, thereby creating increased uplift pressures and eventually sliding of the dam downstream. This led me to conclude erosion was the cause of the excessive load.

- *Why not / Reason-Not-Hapening Explanation*: These explanations answer to a "Why not-Question" and give more information to the user about why a certain result is not the expected result and can provide some clues about the variables that should be changed to obtain the expected result. The approach is similar to that used for "Why" explanation but requires to process more causal information and to handle counterfactual information.

  Example 3.3. *Answer to a "WhyNot-Question" from (Sterling and Lalee, 1986)*

  solve(place_ in_ oven(dish3 ,X))?

  The goal place_ in_ oven(dish3 ,X) fails

  Would you like a "why not" explanation? yes.

Place_ in_ oven(dish3, top) causes the following goals to be reduced: pastry(dish3), size(dish3 ,small)

**pastry(dish3) can be successfully solved? size(dish3,small) is a missing fact.*

End of the explanation.

- *What / Interpretive Explanation*: These explanations answer to a "What-Question" and provide descriptive information about characteristics and terminologies of the elements used by the system. They allow to communicate about the meaning of the concepts used by the system by providing detailed descriptions of them and their relationships within the domain. They can be made either by a specialist of the domain or a knowledge engineer. These explanations mainly use terminological information.

  Example 3.4. *Answer to a "What-Question" from EES (Neches et al., 1985)*

  [What does adder mean?]

  An adder is a primitive system and binary operator. (identification)

  The expected value of its output terminal is equal to the sum of the expected values of its first binary input and its second binary input. (attribute)

A problem is that this criterion of question type is very ambiguous. Indeed, different questions may correspond to the same actual explanation, while on the hand the same question may be interpreted very differently, the best example being 'why-questions', which can have various readings. For example, Clancey (1983b) notes that *"MYCIN's explanations described by Shortliffe et al. (1975) are entirely in terms of its rules and goals. The question WHY means "Why do you want this information?" or "How is this information useful?" and is translated internally as "In what rule does this goal appear, and what goal does the rule conclude about?" "*

3.3.3 Content types of explanations

This criterion has been clearly identified at the end of the eighties by Swartout and Smoliar (1989) and Chandrasekaran (1990). In their scientific researches on the generation of explanations for expert systems, they contributed to identify four types of content.

- *Trace-based explanations* (Chandrasekaran, 1990) (also refered to as "Type 1 explanation" Chandrasekaran and Tanner (1986)) aim to explain why a decision was made or not made by reference to the rules and data contained in the related trace or line of reasoning.

  Example 3.5.
  User: Why is the loan allowed?
  System: Because the borrower is a gold client, and that the amount of the loan is less than 50K.

- *Justification Knowledge* (Chandrasekaran, 1990) (also refered as "Type 2 explanation" by Chandrasekaran and Tanner (1986)) aim to explain or justify a part of a reasoning process by linking it to the *deep knowledge* (see Section 2.2.2.1) from which it was derived.

  Example 3.6.
  User: Why is it that "the borrower is a gold client, and that the amount of the loan is less than 50K then the credit should be accepted?"
  System: Because gold clients provide very strong guarantees of credit solvability, and for moderate credit loan we do not require further conditions.

  Here, what is sought is a justification of the rule. In fact, depending on how the knowledge is encoded, it may be possible or not to retrieve such kind of explanations. In our example, we also see that some knowledge remain implicit:

for instance, a justification of this rule might be that it is important to maintain a high level of confidence between a gold client and the bank.

- *Strategic explanation* (Chandrasekaran, 1990) (also refered as "Type 3 explanation" by Chandrasekaran and Tanner (1986)) aims to explain the system's control behavior and problem-solving strategy. As we have seen in Section 2.2, different control strategies are available for the system, and the answer has to refer to the order in which rules are considered. Such explanations are very typical of a diagnosis context.

  Example 3.7. *In this example, we suppose that the system is helping users to diagnose why their credit loan may have been rejected by some other institutions.*
  User: Why did my loan got rejected?
  System: Do you currently have another credit?
  User: No.
  System: Did you provide all the documents in due time?
  User: No.
  System: Then it is likely that your loan was rejected because you failed
  User: Why did you ask me about the other loan first?

- Definition and terminological knowledge refered in Swartout and Smoliar (1989) and Neches et al. (1985), aim to provide terminological of definitional information to the user.

  Example 3.8.
  User: What is a N12RF document?
  System: This is document of certification provided by institution X.

3.3.4 Context sensitivity

What we call context refers to the variables that are not related to the decision or even the expert system but that have an influence on the efficiency of the explanation. In fact, when constructing an explanation, the context can be considered to make it more relevant and understandable by the user. As result, an explanation can also be categorized by whether or not it is context-tailored, that is, whether its structure and content take into account the user's knowledge of the domain, the current goals, or the current requirements. In this research, we are only interested about the user-sensitivity.

Explanation Type	Description
Descriptive/ Structural Explanation	Knowledge that a phenomenon occurs, a description of a phenomena, taxonomies and classification scheme. Work involving no theoretical grounding or interpretation of the phenomena; presentation of "objective, factual" account of events to illustrate an issue of interest Orlikowski and Baroudi (1991). – gives rise to what explanation *Example: What is the credit score? The credit score of a borrower is a measure which evaluates his reliability.*
Covering-Law Explanation	An explanation of either the D-N or I-S (Hempel, 1965b) type can be "described as an argument to the effect that the event to be explained was to be expected by virtue of certain explanatory facts" (Salmon et al., 1989). The explanatory facts must contain at least one universal or statistical law Hempel (1965a). – gives rise to how explanation based on causality *Example (D-N case): How was it found that Tom has a Down's syndrome? Tom's cells have three copies of chromosome 21 and any human whose cells have three copies of chromosome 21 has a Down's syndrome, thus Tom has a Down's syndrome.*
Statistical Relevance Explanation	"An explanation of a particular fact is an assemblage of facts statistically relevant to the fact-to-be-explained regardless of the degree of probability that results" (Salmon, 2006). *Example: Why John is in late? The probability for john to be in late was high given (P1) the factor of the bad weather, (P2) the factor of the poor road conditions, (P3) the factor of its late departure, (P4) the factor he got an accident, ...*
Pragmatic Explanation	An explanation is a context dependent answer to a "why-question" and different explanations of the same instance will result from different questions (Fraassen, 1988). – gives rise to why explanation *Example: Why does Samy have a persistent lack of energy? Samy's persistent lack of energy could be mainly explained by his hypothyroidism.*
Functional Explanation	These Explanations are framed in terms of ends or goals. A given social practice[factor] has a certain effect. When it has that effect, there is come causal mechanism that ensure a goal A continues to exist.When the practice stops having that effect, that mechanism stops working. (Lombrozo and Carey, 2006) – gives rise to another kind of how and why explanation based on causal chains *Example: Why does Cedric's cell phone plan includes data? Because cell phone plans with data can access to the internet.*

Table 3.2: Explanation Types Table (Hovorka et al., 2008)

Table 3.3: Categorization of existing solutions based on our explanation criteria *(* limited capabilities)*

System	Explanation Orientation		Content Type				Question Type				Context-Sensitivity	Genericity
	Feedforward	Feedback	Trace	Justification	Strategic	Definition	Why	How	What	Why not	User-Sensitive	Genericity
DENDRAL Lindsay et al. (1993)	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗
CENTAUR Aikins (1980), Clancey (1983a)	✓	✓	✓	✗	✓	✓	✗	✓	✓	✗	✗	✗
MYCIN Buchanan and Shortliffe (1984)	✓	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗
EMYCIN van Melle et al. (1984)	✓	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗	✓
NEOMYCIN Clancey (1983b), Dept et al. (1982)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗*	✓
Digitalis Advisor Swartout (1977)	✓	✓	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗
XPLAIN Swartout (1981, 1983)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗*	✗
EES Neches et al. (1985), Paris (1991), Swartout and Smoliar (1989)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
BLAH Weiner (1980)	✗	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓*
PROSE DOMINGUEZ (1990)	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓*
REX Wick and Thompson (1992)	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓*
ODM Explanation Service	✗	✓	✓	✓	✓*	✓	✓	✓	✓	✗	✓	✓

3.4 A quick historical overview of expert systems with explanation capabilities

As one of the greatest success of the artificial intelligence, the expert system's technology has seen an increasing use of its applications at the end of the eighties and, although these systems may seem less popular nowadays, this technology is in fact still widely used in banking and insurance and more broadly by any large company which needs to automate its business policies. As claimed by (Oz et al., 1993), the success of expert systems can be explained by their capability to improve decision reliability as well as performance in term of decision-making time. Moreover, as their decision logic could be complex and needed to be kept up to date, there was a need for monitoring, transparency and trust that prompted the first explanation capabilities. From this point onwards, numerous researches have been conducted to answer the problem of explanation generation in an automated decision context. As solving this problem goes through empirical approaches, some of these systems provided their own solution, making their contribution to the research. We detail in the chronological order the main steps and progress that have been done during the forty last years below. The first attempts to provide programs with explanation facilities take their roots in tools designed to the monitoring and debugging of softwares. From this point of view, debugging tools can be seen as the ancestor of the explanation capabilities we know because they constitute a first attempt to investigate the results provided by a computer system.

Explanations from canned-texts or templates. In these approaches the programmers have to anticipate what questions are likely to be asked by user to prepare explanatory texts, that can be either canned-text or templates, and associate them to the corresponding parts of the program that need to be explained. What we call a *canned-text* refers to an explanatory text that can be written by the programmers when programming the expert system and what we call *templates* corresponds to a structure that mixes canned-text with variables that can be filled in with values

from a specific execution of the program. Then, based on this preparation, when a user requests an explanation about some aspects of the program's behavior, the text associated with that portion of the program is simply displayed. Although explanation facilities using these approaches are easy to perform there is an issue in the fact that they produce explanations that can be inconsistent and inextensible. To answer this problem, researchers figured solutions that produce explanation directly from the system's code and give rise to the systems that have been referred as *first generation expert systems.*

Explanations by Code Translation. Often referred as first generation expert systems, these systems have been designed in order to deal with the limitations of the previous approaches. In this perspective, the idea to generate explanations by simply translating the code. The principles used by these approaches are very simple: the structure of their explanations follows exactly the structure of the program code and the names of the variables in their explanation are those used in the program. Based on this method, they can generate some texts in a near natural language that can be seen as explanations. These approaches give rise to what we called descriptive explanations. Such systems can generate trace-based explanations to answer "how" questions. To some extent, the consistency and extensibility problems have been solved. Indeed, as the explanations are produced directly from the code, any change in the code is reflected in the system's explanations automatically. Moreover, the explanations produced by this way have been described as very useful for system builders. However, as these systems are only trace-based, they do not have enough knowledge to support more complete explanations including justifications that could be more profitable to business users. The following expert systems are good representatives of this generation: MYCIN (Buchanan and Shortliffe, 1984), Digitalis Advisor (Swartout, 1977), DENDRAL (Lindsay et al., 1993) and CENTAUR (Clancey, 1983a).

Explanations using additional knowledge about the system. Commonly referred as *second generation expert systems*, these systems have been designed in order to deal with the limitations of the previous approaches. In this perspective, they have a more complex architecture designed to deal with explanation in an efficient way and integrate justification, strategic and/or terminological knowledge, in addition to the trace-based knowledge. As they use their own knowledge to produce explanation, they are able to produce abstract descriptions of their problem-solving strategies and describe their behavior or the concepts of the domain. Based on that, they can provide more complete explanations and answer more kinds of questions. The following systems are good representatives of this generation: NEOMYCIN (Dept et al., 1982) which is an evolution of MYCIN (Buchanan and Shortliffe, 1984), XPLAIN (Swartout, 1983) which is an evolution of Digitalis Advisor (Swartout, 1977) and PROSE (DOMINGUEZ, 1990). Still, these systems still encounter problems to deal with different users or context. In fact, they are inflexible and have limited capabilities to tailor explanation to different classes of users or in different contexts.

Explanation using additional knowledge about the user and the situation. These systems, that we refer as *context-responsive expert systems* contain additional knowledge about the user and the context in addition to the previous knowledge. As an evolution of the systems described in the previous approaches, they usually have strong knowledge base about domain, problem-solving, users, languages. They also could have knowledge about the situation and enhanced capability for dialog. Thanks to this, these systems can tailor explanations to different users in different contexts but they are limited their knowledge about the user and by the research in natural language. Moreover, they are hard to design and produce. Good representatives of this type of expert systems are: REX (Wick and Thompson, 1992), EES (Paris, 1991, Swartout and Smoliar, 1989) which is an evolution of XPLAIN (Swartout, 1983).

The proposal is often made that deep knowledge should represent "causal know-

ledge but Chandrasekaran and Mittal (1983) claims that using only a collection of cause-effect relations as deep knowledge is debatable and that deep reasoning should also come from other properties of the system. In fact, this intuition come from the fact that causal reasoning is important because of the frequent need to establish causal connections at different levels of detail for reasoning and explanation.

For example, if the knowledge compilation results in the approximation of a set of operations contained in the deep model by associating some data states to other data states, the is a loss of information as the system is not able to provide both the intermediate data states and the corresponding step of the reasoning. In this case, the (1) trace-based knowledge loses the information about the intermediate data states, the (2) justification knowledge loses the information about the corresponding steps of the reasoning and underlying operations and the (4) definition knowledge may eventually lose information about the existence of intermediate variables and methods. Concerning the (3) strategic knowledge, as it relies on the capability to describe what the system is doing or will do next and to navigate at different levels of the model, its content is highly dependent from the compiled knowledge structure. If a knowledge structure is easy to navigate and encompasses several levels of details, it will be easy to obtain such information. For example, if the structure of a compiled knowledge consists in a tree of tasks, where each task embeds its own set of rules. The strategic information will rely on the hierarchy established among these different elements. Thus, the decision could be described at the level of the tree, at the level of the tasks, at the levels of the rules and even at the level of their conditions and actions.

3.5 Causality in rule-based systems

As we have seen, expert systems like MYCIN (Buchanan and Shortliffe, 1984) and XPLAIN (Swartout, 1983) have demonstrated that considering causality is key for justifying automated decisions and increasing the transparency of decision-making systems and so the confidence of their users. This fact motivates a more detailed

study of the causality in rule-based systems. There are several reasons for using causal models when trying to form causal knowledge in a rule-based system. It goes from practical aspects that are related to the implementation to more theoretical aspects coming from explanation theory and psychology (focus on human understanding):

- business rule approaches offer favorable conditions to causal ascription and thus to represent the causal model of a rule-based system and its decisions. The decisions taken by using business rule approaches are known to follow causal processes. Indeed, the automated decisions taken by rule-based systems are based on inferences and consequently the related decision processes have a causal meaning and so a corresponding causal model can be found. Based on that, enabling a rule-based system to present the events occurred during a decision with their underlying causal mechanisms could greatly help to improve its transparency. Moreover, as business rules are declarative, they explicitly represent their reasoning logic. This reasoning logic can be easily understood by humans readers. Indeed, the premises and actions of a business rule take the form of near natural language statements and the IF THEN ELSE structure makes a clear split between the premises and action parts. Based on that, each rule describes causal relationships by linking its local causes and consequences.

- causal models offer a formal representation which is non-ambiguous, independent from the domain of application and can be implemented with classical methods and tools. Because of that, these models guarantee more reliability than ad-hoc methods.

- qualitative information, and especially causal one, is known to enhance the understanding of automated decisions. Indeed, revealing a decision-making process is not sufficient to make it understandable. A collaboration between a human user and a rule-based system requires a mutual understanding and consequently such a system must take into account human cognitive proces-

ses in order to explain their reasoning to human users. For instance, Tversky and Kahneman (1975) and Pennington and Hastie (1988) have shown that the human brain tends to interpret events in terms of cause-effect relations and, from this perspective, using causal models to support explanation seems natural and presents several interests. The state of the art also shown that the notion of causation is closely tied to the concept of scientific explanation which essentially focus on finding the causes of the observed facts (explanandum) and the laws that link them together. Based on that, some explanation methods used in rule-based systems and so called "causal explanations" have been specifically designed for causal models and assume that a causal interpretation of the reasoning process can be used to justify the decision results. Besnard et al. (2014b) and Clancey (1983b) even claim that enabling rule-based systems to identify and present causal dependencies between the elements of their decisions is a prerequisite to provide explanations increasing their acceptance, their maintainability and their continuous improvement. These statements are illustrated by the survey of Moore and Swartout (1988b) where most of the expert system explanation capabilities consider causation. Finally, Besnard et al. (2014a) and Halpern and Pearl (2005c) also highlight that causal models have a strong role in the construction of explanations.

As causal models can be built by using generic methods and allow to determine the implications of events occurred during a decision on the outputs, using the information offered by such models for augmenting tracing tools seems pertinent. Based on that, as we have seen, some explanation methods used in rule-based systems and so called "causal explanations" have been specifically designed for dealing with causal information and assume that a causal interpretation of the reasoning process can be used to justify the decision results, one of the first successful implementation of causal explanation features in a rule-based system–see *e.g.* (Clancey, 1983b). Thus, enabling a rule-based system to provide causal explanations implies the use of a causal model that captures the relations defined by the left hand side and right

hand side of each business rule and also the rule chaining and data modifications that drives to the final outcome.

3.5.1 Causal explanations in rule-based systems

The Stanford Heuristic Programming Project (Buchanan and Feigenbaum, 2017) has been one of the first laboratories to bring the use of A.I. programs into the research on modeling the nature of scientific reasoning processes. In this perspective, various computer programs, formalizing and embedding the human expertise in production rules, have been developed and used to simulate the behavior of human experts. The idea behind this methodology was to investigate the nature of expert reasoning processes within software in order to develop a deep understanding of its. In this perspective, the Stanford Heuristic Programming Project has conducted researches on five key scientific problems of artificial intelligence: knowledge representation, knowledge acquisition, knowledge utilization, explanation and tool construction. There is a strong connection between this five key points and some of the most influential attempts to provide rule-based systems capable of explaining their reasoning processes and decision results have emerged from the Stanford Heuristic Programming Project Buchanan and Feigenbaum (2017).

For most of the systems developed in the Stanford Heuristic Project (Buchanan and Shortliffe, 1984, Clancey, 1983b, Dept et al., 1982, Lindsay et al., 1993, Mark and Clancey, 1985, Wallis and Shortliffe, 1984), embedding causal knowledge has been a key point for explaining their reasoning processes and their decision results. It is important to note that the form and the content of this causal knowledge are not well defined and can greatly vary depending on the vision of the system designer. The content of this causal knowledge has a direct influence on the related explanation features. In a similar vein and as argued in (Swartout and Smoliar, 1988), one key point to provide explanations that increase the confidence in the behavior of a rule-based system is the notion of justification. The idea behind this notion is that the user needs to understand the connections between the reasoning steps to accept the

statements provided by the system as reasonable explanations. The production of justifications mainly relies on the abstraction of the causal reasoning followed by the system. This means that a justification has to provide, at least, the references to the causal relationships on which the rules are based. More broadly, the raw material for justifying the behavior of a rule-based system is mainly the causal knowledge. What it changes from a system to another is how this causal knowledge is acquired, represented and then used. For example:

- in the case of GUIDON-WATCH (Mark and Clancey, 1985), the causal knowledge consists in a set of causal links between findings and hypothesis where the findings correspond to the observed or measured data (used in the premises of a rule but also can be requested or inferred from other rules) and the hypothesis are only the data resulting from rule inferences (in conclusions). This causal knowledge is completed by an etiological taxonomy which represents a hierarchy among the possible diagnoses (findings) that can be displayed in a tree of possible diagnoses. Thanks to this mechanism, GUIDON is able to provide a set of graphs and trees presenting links between a sequence of hypothesis and findings in order to display qualitative information in graphical views that increase the understanding about the consultation strategy of the system.

- the work of Dept et al. (1982) on NEOMYCIN includes explicit causal knowledge consisting in a set of meta-rules that represents a direct association between the premises (observed data) and the conclusions (hypotheses in the etiological taxonomy) of rules marked as being causal and a taxonomy of diseases which aims to make the user understand both the model and the reasoning. Whereas the etiological taxonomy gives a hierarchy of links representing specialization of cause, the causal rules associate incoming data to hypotheses in the taxonomy.

- the research work done by Wallis and Shortliffe (1984) and Wallis and Shortliffe (1981) proposes a prototype system that includes a conceptual knowledge base

taking the form of a semantic network. This semantic network describes a hierarchy between rules, objects, parameters and values nodes while providing further static and dynamic information about them. The information provided inside the semantic network about the complexity and the importance of values nodes as well as the information about the type and the complexity of rule nodes allows to provide causal knowledge about the reasoning process of the system. This causal knowledge takes the form of a causal chain which alternates between rule and value nodes and describes the reasoning sequence. Moreover, when this causal chain is used to generate tailored explanations, a threshold can be set to associate text justifications to the rules and values that are complicated (the complexity is higher than the threshold).

- in XPLAIN (Swartout, 1983), the causal knowledge is contained in a domain model that embeds some descriptive facts of the domain such as causal and classification links similar to those described in Weiss et al. (1978) but non-weighted. These relations can then be used to justify the action of the rule-based system. Similarly, ABEL (Patil et al., 1981) is based on the same idea than XPLAIN but also defines aggregation links that allow to provide a view of the causal model at different levels of granularity (pathological-intermediate-clinical).

In fact, in all these attempts, the use of causal information reveal to the human users how the cause and effect are linked and provide them with a better understanding of the cognitive process than the previous approaches like (Lindsay et al., 1993) or (Wallis and Shortliffe, 1977). The causal information can thus be used as justifications to explicitly describe the links between some effects and their causes.

Nonetheless, representing a causal model of such processes is not necessary straightforward. Many parameters, such as the characteristics of the inference mechanism used, the priorities of the business rules and other system preferences, the business rules themselves and the state of the working memory must be considered and choosing the good representation and determining what should be represented in it is

not easy. Moreover, as discussed by Dubois and Prade (2003), it is difficult to define what should be considered as a causal relation or to determine what kind of causal model should be used because of the unclear nature of the notion of causality. This statement is also emphasized in (Benferhat et al., 2008) which describes the causality as as a "protean and complex notion" and propose a comparative study of the main causal models that have been developed by artificial intelligence community. In the next subsection we discuss the choice of a causal model that could be efficiently used to describe the reasoning processes of a rule-based system.

3.5.2 Choosing a causal model / formal model for causal ascription

Since business rules are declarative it enables to represent their reasoning logic without precise knowledge of the working of the inference engine and the declarative readings of the corresponding logical formulas can thus be exploited to generate causal models that provide a global and qualitative vision of the decision which offers more readability and understandability as claimed by Korver and Lucas (1993). However, there is one major remaining question. The scientific literature about causal modeling gives us several types of causal models, each one has its own specificities and choosing one over another is not obvious and should depends on the application. Indeed, the concept of causality covers some important problems of A.I. Among them are the diagnosis of the potential causes from observed effects, the inductions of causal laws from series of observations, the qualitative simulation of complex systems and others.

Accordingly, the comparative study made by Benferhat et al. (2008) presents several formal models used to represent causality and describes their strengths and weaknesses (to which we add a popular neural network): .

Structural equations (Halpern and Pearl, 2005a). This approach proposes a causal model that aims to identify the "actual causes" of an observed event. To do that, it distinguishes two kinds of variables: "endogenous" and "exogenous" variables.

The "endogenous variables" (in V) are used to model events, represented by specific values, that can be causes or caused. The "exogenous variables" (in U) are assumed to be known and out of control. Causal relations between events are represented by using a set of structural equations and contexts. Whereas the structural equations describe dependencies between "endogenous variables", the contexts represent some settings of the "exogenous variables" that make them true. A causal model is denoted $M(S, F)$ where $S = (U, V, R)$ is the signature of the system and F is a causal function which assigns to each variable in V a value depending on its parents and a context given by the variables in U. R is the non empty set of possible values for the variables in $U \cup V$.

Nonmonotonic logic. Causal explanations and nonmonotonic inferences are connected, the intuition behind this is that the causal explanations provided by humans tend to privilege abnormal facts. In nonmonotonic logic there is an assumption that some conclusions can be invalidated by adding more knowledge. The non-monotonic logic is devised to capture and represent defeasible inferences. Non-monotonic logic is useful for several artificial intelligence applications like default reasoning (where each defined rule can be used unless it is overridden by an exception), abductive reasoning (where the consequences are only deduced as most likely explanations), belief revision (where new knowledge may contradict the old one) or more broadly for reasoning about knowledge. The three methods below are based on non-monotonic logic and are used for causal ascription.

- *Nonmonotonic consequence* is based on pieces of default knowledge, and privileges the role of abnormal events in a given context.

- *Trajectory-based preference relations* starts with the idea that counter-factuality involves the computation of two kinds of evolutions of the world, namely extrapolation and update. Compute the most normal evolution of the world (called trajectory).

- *Norm-based approaches* rely on the idea that norms are crucial for people to find causes of events. If the event is considered normal then its cause is the norm itself but if it is abnormal then its cause is traced back to the violation of a norm.

Thagard's explanatory theory of coherence and its connectionist implementation (ECHO). This approach views causal ascriptions as attempts to maximize explanatory coherence between propositions. The basic idea of this approaches is that maximizing coherence would lead to accept the most plausible hypotheses that explain the accident and reject the alternative hypotheses.

Graphical models and intervention. This line of research is based on the idea that the causality becomes easier to observe when experimenting and then observing the effects of a specific manipulation on the system. In fact, graphical causal models help make explicit the assumptions needed by allowing inference from interventions as well as observations.

Neural Networks for causal learning. Also called Feed Forward Network, are able to approximate complicated functions as long as they are single-valued and follow a causal relationship that takes the form of an input-output relation.

The benefits and limitations of neural networks are described by Hall (2007), Hitchcock (2007) who argues for the superiority of structural equations. In addition, Halpern (2014) demonstrates the strong resilience of these models while Halpern and Hitchcock (2011) emphasizes the importance of the choice of the variables during their definition. Moreover, as described in Halpern and Pearl (2001a, 2005a), structural equations allows to represent causality in comprehensible networks that can easily improve the understanding of how a phenomenon occurred by displaying the relations between the elements involved in the causal process. Another account of causal explanation using the structural equations has been done in the field of data-

base. Actually, the structural equation approach tends to be highly explainable and can be applied in a mechanistic way. For these reasons, our choice is oriented towards the tool proposed by Halpern and Pearl (2005a) which is particularly well suited to describe the causality in rule-based systems because our business rule formalism is easily transferable in a structural equation form as "$Y \Leftarrow X$" as demonstrated with the logic representation of our business rule formalism.

3.6 Requirements for an IBM ODM explanation feature

In this section we discuss some specificities of the explanation feature targeted in the context of IBM ODM. The idea behind recording the traces of automated decisions is to give Business Rule Management Systems the ability to provide regulatory justifications and analytics about their decision-making processes. For most of the organizations, this is a prerequisite for monitoring and optimizing their business. In practice, most of decision traces are composed of:

1. a *request* that contains the input data and potential states to take the decisions,
2. a *sequence* of executed rules, with optionally details about the algorithm evaluation,
3. an *answer/decision* that contains the output data of the decision with the modified states.

In our industrial context, we are interested in decision services generated by the IBM Operational Decision Manager which is the BRMS provided by IBM. For such decision services, decisions are entirely automated so they do not require any human interaction and cannot be interrupted. Once a decision has been taken, a basic trace of the corresponding decision process is stored on a server database. Later, based on this trace, the information related to the corresponding decision have to be displayed in an explanative way. IBM needs the information provided in the explanation to allow to monitor the decision taken by such a decision service and to make them

transparent for the user. Because the decision processes can involve a large number of rules and data, we need to find a way to filter out non-necessary elements and to provide only the useful information to the user. Furthermore, the production of the trace has to be efficient to minimize the engine instrumentation and extra resource consumption of RAM and CPU cycles. In addition, the tool should not require any ad-hoc development specific to each rule project because it could have a negative impact on its maintainability. Moreover, as this useful information needs to be understandable by a human user, we need to enrich it with further knowledge that make it more meaningful. In fact, when an explanation is rendered to the user it should take into account the kind of user, its level of expertise, what information he can access. In the light of the above comments and in regards of our industrial context, we will discuss the requirements of our solution for each criterion.

- *Explanation orientation.* As the decisions taken by the decision services generated by the IBM BRMS cannot be interrupted and as the explanation are only provided for decision results that have been already stored in a server database. Thus our explanations have to be feedback explanations.

- *Content type.* As we want to explain decision results, our explanations obviously need trace-based knowledge as basic requirement. But as we need to provide further information about causality, we seek Type 2 explanations, as a kind of justification knowledge. We naturally provide further information about concepts and methods used, that can be considered as *terminological knowledge*. However, we will *not* get to the level of procedural reasoning, involving for instance the choice of the algorithm. Our proxy to this strategical level will be offered by the related task corresponding to a step of reasoning. To make this clear we talk about *limited strategical explanation.*

- *Question type.* In fact, in our context we do not want to answer these questions separately. We need to provide an explanative view that allows to navigate and explore the decision. This explanative view should provide information about

the step of the reasoning process (that corresponds to a "How" question) but it also should provide qualitative information allowing to understand how the the inputs and outputs of a decision are connected (that corresponds to a "Why" question) Moreover, when navigating the decision the user should be able to display information about the concepts used by the system in order to provide them more meaning (that corresponds to a "what" question). Optionally, the user would know why a result is not the result he expected, in this case we could display a counterfactual solution based on our static knowledge of the decision service.

- *Context-sensitivity.* As described above, the only thing we need to handle is the user, but there may be several types of users (business user, knowledge engineer). To account for this, a user model could be used to adapt the display and the content of our explanative view.

- *Genericity.* In fact, as lot of different decision services can be generated from the IBM BRMS, we would to design an explanation service that could handle all the decision services that already exist and that can be created in future. Indeed, it would be so costly to design a different explanation system for each of these decision services. For this reason, our system has to be generic enough to deal with a wide variety of very different decision services that can be generated by IBM ODM.

However, it is important to note that there are no standards, or at least well-established form and content criteria, for decision traces and so each rule-based system proposes ad-hoc traces whose the form and content depend upon the goodwill of the developers. Thus, some challenges remain for the production and exploitation of common rule engine traces to generate feedback explanations.

(1) *readability* – As simple logs of events occurred during decision processes, traces suffer from a lack of intelligibility and are not well suited for human readers.

(2) *sufficiency* – Common rule engine traces may not be complete enough to answer "Why" questions. In a trace, a sequence of executed rules is captured but not the information revealing the genuine causes of their eligibility.

(3) *minimality* – In reverse some rule engine traces may contain information that are unnecessary to justify a decision. For example, modification of data or rule agenda that have no impact on the outcome of a decision have no interest for the explanation. This point is important because the instrumentation of a rule engine to produce traces may significantly burden execution resources including CPU cycles and memory. Indeed, the more thorough is the tracing, the more data and rule engine internal structure will be visited, thus resulting in the serialization of larger amount of data into the traces.

(4) *maintainability* – Some common practices rely on ad-hoc messages added to the rules, or additional descriptions specific to the domain. These approaches induce a significant cost in the development of the business rules and a recurrent maintenance cost to keep in sync the explanation material with the modified business rules.

In the light of the discussion above, we guess that the content of the explanation may be our main research interest, but it may not be the only one. Following (Moore and Swartout, 1988a), the scientific research on explanation generation in expert systems is mainly focused on the three axis:

1. Basic Explanation Content Generation,
2. Responsiveness,
3. Human Computer Interface (HCI).

As IBM works with a huge variety of organizations, each one having its own practices and habits with the human-computer interfaces they used to use, we choose not to address this HCI aspect. On the other hand, we do consider responsiveness because we would like to provide an explanation capability which could take into account some user's characteristics when rendering explanations.

But in fact, our industrial context and the fact that the Business Rule Management System provided by IBM can be used to generate decisions services for a huge variety of companies and applications, it induces some new issues that leads us to consider other aspects like:

1. Genericity,
2. Overcost at runtime (in CPU and memory),
3. Overcost at exploitation and maintenance time (additional financial cost for maintenance and exploitation).

Thus, it is important to note that any explanation facility actually has to take into account the costs for companies that are looking for enhanced solutions to explain their automated decisions. For these organizations, saving resources in the development, execution and storing phases makes a significant difference due to the volume of decisions processed and recorded. These costs encompass the additional financial costs at development and maintenance required for each rule-based applications, the additional cost in memory required to store the explanation material, and the additional cost in CPU during the decision process which is related to the augmentation of the tracing tool. In this perspective, the construction of such explanations needs to take into account some theoretical and technical aspects to reach industrialization (i.e. meet the constraints for an industrial application). They have to rely on generic methods to be applied on a large variety of rule-based services with limited extra-cost in conception and development. It it possible to answer this need with a causal approach because the conception of a causal model which supports the explanation and the ascription of its causal relations can be done by using generic methods. It means that the method used to construct a causal model can be entirely automated and applied to various rule-based services with no regard for their application domains.

Chapter 4

A simplified causal model for rule-based systems

4.1 Introduction

In this chapter, we present a set of causal models and methods that can be used to support explanation capabilities in a business rule context. It is composed of: (i) a causal model of the business rule-based system, (ii) a method for minimizing the traces of its decisions and (iii) a causal model of each decision. The remainder of this chapter is as follows. In Section 4.2, we present the overall concepts and principles of business rule-based systems. In Section 4.3, we introduce the notion of causality and propose an approach to handle causality in business rule-based systems. Finally, the Section 4.4 describes the construction of our causal models and "minimized trace".

4.2 Concepts and definitions

4.2.1 Business rules

As we have seen before, the most fundamental notion in Business Rule-based systems (BRBS) is the one of *business rule*. Business rules are commonly specified by means of an ontology language, and often a description logic language. Each business rule can be seen as a statement, taking the form of a customized near natural language text, which defines a business aspect that allows to precisely describe the

decision criteria and the actions to apply for a given situation of a decision process. Recall that a business rule takes the form of a statement written in a business rule language and whose logical structure is described as follow:

$$IF\ \langle premisses \rangle\ THEN\ \langle consequent \rangle$$

where the premisses is a disjunction of conjunction of conditions and the consequent is a sequence of actions:

$$IF\ \langle c_1\ AND\ \cdots\ AND\ c_m \rangle\ OR\ ...\ OR\ \langle c_1\ AND\ \cdots\ AND\ c_n \rangle\ THEN\ \langle a_1; \cdots ; a_n \rangle$$

The actions in the consequent are typically variable assignments, and can involve various arithmetic operations whereas the conditions in the premisses are expressions that can be evaluated. Based on the satisfaction of the conditions in its premisses, a rule is *eligible* for being triggered. When a rule is *triggered*, the sequence of actions corresponding to the consequences is executed.

Example 4.1. *(Example of business rule)*
A rule having the business rule language form:

> *if the score of the Borrower is higher or equal to* 10 *then set the rate of the Loan to (the score of the Borrower + the bonus of the Borrower) divided by* 100 *;*

and corresponds to the conditions-actions statement:

> *IF* $\langle c_1 \rangle$ *THEN* $\langle a_1 \rangle$*, where "*c_1*" refers to the condition statement "the score of the Borrower is higher or equal to* 10*" and "*a_1*" refers to the action statement "set the rate of the Loan to (the score of the Borrower+the bonus of the Borrower) divided by* 100*"*

Based on this, a rule-based system is essentially a collection of rules, together with some input and output variables, called *business variables*, and an inference mechanism that aims to solve a specific problem. User interactions with such systems

take the form of requests whose content depends on the application. A request amounts to ask the value of a specific (output) variable. In practice, the working cycle of such a system consists in three steps: (i) the rule-based system receives a request, (ii) based on this request, it solves a specific problem and (iii) it returns the results obtained as outputs. More details about the basic structure of a rule-based system are given in the chapter 2.

4.2.2 Towards a normalized business rule formalism

As statements are given in a near natural language, the description of the business rules can vary depending on the verbalization that has been arbitrarily adopted by the designer of the rule-based system. Providing a generic approach for explaining business rule decisions requires that the information contained in the business rules can be extracted and represented in a normalized form which is independent from the syntax and the vocabulary used in the business rules. In this perspective, a formalism that describes each element contained in a business rule and that can be used to represent the business logic of any rule-based system has to be clearly defined. Thus, the normalized business rules in our representation uses a regular expression formalism that is based on the work done by Lucas and van der Gaag (1991) on the production rules and on the nature of the business rule used by IBM Operational Decision Manager. As the formalism given by Lucas and van der Gaag (1991) to describe production rules assumes to work only with constants and does not handle variables and business functions, we adapt it to represent business rules in our business and causal contexts. The adaption is described in what follows.

4.2.2.1 Object-Oriented vs. variable-oriented formalism

Defining a normalized business rule formalism based only on the manipulation of variables is too simple and does not capture the business meaning of the concepts manipulated by recent business rule applications that establish a taxonomy between the manipulated elements. In fact, most of modern business rule-based systems

apply these object-oriented methods. Using an object-oriented approach means that the system represents "objects" that explicitly group some properties which are mentioned in the business rules. For example, in a business rule application dedicated to answer loan requests, a "borrower" having a name, an age, some financial resources and monthly incomes can be represented as an object that models all its characteristics and which has a business meaning. In these approaches, the objects themselves are exploited for directing the inference process and so are considered when designing the business rules. We take the object-orientation into account

We obtain business rule formalism that can be used to normalize business rules in object-oriented applications. This object-oriented business rule formalism is defined in the table 4.1.

$\langle business\ rule \rangle := IF \langle premisses \rangle$ THEN $\langle consequent \rangle$
$\langle premisses \rangle := \langle conjunction \rangle \{ \ or \ \langle conjunction \rangle \ \}*$
$\langle conjunction \rangle := \langle condition \rangle \{ \ and \ \langle condition \rangle \ \}*$
$\langle condition \rangle := \langle predicate \rangle (\ \langle object \rangle, \langle attribute \rangle, \langle value \rangle \ )$
$\langle predicate \rangle := equal \vert notequal \vert greaterthan \vert lessthan \vert \vert known \vert notknown \vert ...$
$\langle consequent \rangle := \langle conclusion \rangle \ \{ ; \langle conclusion \rangle \ \}*$
$\langle conclusion \rangle := \langle action \rangle (\ \langle object \rangle, \langle attribute \rangle, \langle business\ result \rangle \ )$
$\langle business\ result \rangle := \langle business\ function \rangle \ (\ \{ \ \langle value \rangle \ \}* \ )$
$\langle action \rangle := remove \vert set \vert ...$
$\langle value \rangle := \{ \langle object \rangle, \langle attribute \rangle \} \vert constant$

Table 4.1: A Business Rule Formalism for object-based applications

In predicates and actions, the tuples we consider are "object-attribute-value", where a pair (obj, att) represents the value of the attribute att of an object obj and a value val represents either the value of a constant or another pair $(object, attribute)$. The reference to the attribute att of an object obj is abbreviated as $obj.att$. Based on these statements, the predicates and actions tables of Table 4.3 are obtained. We present a normalized business rule illustrating the object-oriented business rule formalism in the example 4.2.

Example 4.2. *Let us consider the following business rule statement:*

IF *equal(Borrower,gold,true) and lessthan(Loan,amount,500000)) or*

(greaterthan(Borrower,salary,Loan.yearlyRepayment) and lessthan(Loan,amount,50000))
THEN *set(Loan,status,accepted)*
where equal(Borrower,gold,true) and lessthan(Loan,amount,500000) is a conjunction of two conditions which is true if the attribute gold of a borrower has the value true and the amount of the loan is less than 500000 and greaterthan(Borrower,salary,⟨ Loan,yearlyRepayment⟩) and lessthan(Loan,amount,50000) is a conjunction which is true if the salary of a borrower is greater than the yearly repayment of the asked loan and its amount is lower than 50000. The action $set(Loan, status, accepted)$ *sets the status of the loan to accepted.*

Predicate noun	Logic representation for single-valued attributes
equal(obj,att,val)	$obj.att = val$
notequal(obj,att,val)	$obj.att \neq val$
greaterthan(obj,att,val)	$obj.att > val$
greaterthaneq(obj,att,val)	$obj.att \geq val$
lessthan(obj,att,val)	$obj.att < val$
lessthaneq(obj,att,val)	$obj.att \leq val$
known(obj,att)	$obj.att \neq \emptyset$
notknown(obj,att)	$obj.att = \emptyset$

Table 4.2: Predicates and their meaning

Action noun	Logic representation for single-valued attributes
set(obj,att,val)	$obj.att \leftarrow v$
remove(obj,att,val)	$obj.att \leftarrow \emptyset$

Table 4.3: Actions and their meaning

4.2.3 Orchestration and execution of the business rules

Once the rules are defined, they can be executed by the rule engine (if their conditions are satisfied), as described in Section 2.2. The main parameters influencing their orchestration are: (1) the priority among the business rules, (2) the rule engine algorithm and (3) the ruleflow.

Example 4.3 presents a business rule-based system evaluating the eligibility of somebody for a loan request based on a set of input variables. In the rest of this

chapter we will refer to this example.

Example 4.3. *The decision is taken based on the values of five business variables namely: "bankruptcy", "score", "bonus", "rate" and "eligibility". These business variables can be grouped under two business objects, respectively called "Borrower" and "Loan". A first task called "compute" is defined for computing a rate, with the ruleset RS_1 attached to it. Another task called 'evaluate" is defined for* evaluating *a loan, with the ruleset RS_2 attached to it.*

Business Variables:

variable bankruptcy : bankruptcy = {false,true} %input

variable score : score = [0,100] %input

variable bonus : bonus = [0,80] %input

variable eligibility : elibility = {false,true} %output

variable rate : rate = [0,1] %output

Task compute: compute(RS_1)

RS_1 in ODM form (before normalization):

Rule A: if it is not true that 'the borrower' has filed a bankruptcy then increase the credit score of the borrower' by 5;

Rule B: if the credit score of 'the borrower' is at least 10 then set the rate of acceptance in 'the loan report' to (the credit score of 'the borrower' + the bonus of 'the loan')/100 ;

*Rule C: if the rate of acceptance in 'the loan report' is at least 0.2 and the credit score of 'the borrower' is at least 12 then set the rate of acceptance in 'the loan report' to the rate of acceptance in 'the loan report' * 2;*

RS_1 in normalized form:

Rule A:

IF $equals(\langle Borrower, bankruptcy\rangle, false)$

THEN $set(\langle Borrower, score\rangle, \langle Borrower, score\rangle + 5)$ *;*

Rule B:

IF $greaterthaneq(\langle Borrower, score\rangle, 10)$

THEN $set(\langle Borrower, rate\rangle, (\langle Borrower, score\rangle + \langle Loan, bonus\rangle)/100$ *;*

Rule C:

IF $greaterthaneq(\langle Loan, rate\rangle, 0.2)$ *AND* $greaterthaneq(\langle Borrower, score\rangle, 12)$

THEN $set(\langle Loan, rate\rangle, \langle Loan, rate\rangle * 2)$*;*

Task evaluate: evaluate(RS_2*)*

RS_2 in ODM form (before normalization):

Rule D: if the rate of acceptance in 'the loan report' is less than 0.6 then decrease the credit score of 'the borrower' by 5;

Rule E: if the bonus of 'the loan' is less than 20 then decrease the rate of acceptance in 'the loan report' by 0.01 ; decrease the credit score of 'the borrower' by 1;

Rule F: if the rate of acceptance in 'the loan report' is at least 0.5 then make it true that 'the loan' is approved;

RS_2 in normalized form:

Rule D:

IF $lessthan(\langle Loan, rate\rangle, 0.6)$

THEN $set(\langle Borrower, score\rangle, \langle Borrower, score\rangle - 5)$ *;*

Rule E:

IF $lessthan(\langle Loan, bonus\rangle)$

THEN $set(\langle Loan, rate\rangle, \langle Loan, rate\rangle - 0.01)$*;*

$set(\langle Borrower, score\rangle, \langle Borrower, score\rangle - 1)$ *;*

Rule F:

IF $greaterthaneq(\langle Loan, rate\rangle, 0.5)$

THEN $set(\langle Loan, eligibility\rangle, true)$ *;*

4.2.4 Tracing the process: what should be in the decision trace?

Most of decision traces simply provide the complete list of rules triggered and even if they translate the execution of triggered rules, however their content may be arbitrary and not fully reliable. For instance, such traces may contain irrelevant information that can be seen as noise or even suffer from a lack of information resulting in a reduced capability to establish links between the inputs, the different rules and the outputs. Indeed, as it was emphasized for instance by Alvarez (2004), a trace of execution, unfortunately does not always hold the right or necessary information that is needed to understand and accept the outputs (decisions).

In our context, the tracing tool aims to record enough information to enable the representation of causal dependencies between the events occurred during a decision. Being able to do that means that the data recorded in the trace must inform about several important characteristics.

(i) *What has been triggered*: Having some knowledge about the nature of the event that occurred allows to infer information about its behavior. In this perspective, the trace has to contain information about *the triggered decision artifact (a business rule or a task)*;

(ii) *What has been considered to trigger the decision artifact*: knowing the parameters that have been taken into account in the premisses of a decision artifact is a part of the knowledge required to infer the potential causes of its occurrence. In practice, *the business variable(s) evaluated by the decision artifact* are sufficient;

(iii) *What has been modified by the triggering of the decision artifact*: knowing what are the consequences of the triggering of a decision artifact allows to evaluate what could be caused by its effects. Recording *the business variable(s) that have been modified by a decision artifact* is a prerequisite for establishing causation;

(iv) *When the decision artifact has been triggered*: knowing the temporal position

of a decision artifact allows to constraint what could be a cause or what could be caused by a such decision artifact. In practice, this temporal order is given by *the step (order of execution).*

Adopting the formatting syntax
"Dec. artifact instance (dec. artifact ID, evaluated variable(s), modified variable(s), step number)" for a recorded line, Example 4.4 presents the decision trace obtained by using the input parameters `(bankruptcy = false , score = 8, bonus = 15 )` for the rule-based application described in the Example 4.3.

Example 4.4. *(Ex. 4.3 Cont.)*

```
BEGIN DECISION
-> INPUTS (bankruptcy=false,score=8,bonus=15)
TaskInstance(compute ; <bankruptcy=false,score=8,bonus=15>;
 ... ; step 1)
RuleInstance(Rule A ; bankruptcy=false ; score=13 ; step 2)
RuleInstance(Rule B ; score=13 ; rate=0.28 ; step 3)
RuleInstance(Rule C ; <rate=0.28,score=13>; rate=0.56 ; step 4)
TaskInstance(evaluate; rate=0.56 ; ... ; step 5)
RuleInstance(Rule D ; rate=0.56; score=13 ; step 6)
RuleInstance(Rule E; bonus=15; <rate=0.55, score=7>; step 7)
RuleInstance(Rule F ; rate=0.55; eligibility=true ; step 8)
-> OUTPUTS (eligibility=true,rate=0.55)
END DECISION
```

Our ambition is to go beyond such traces by proposing a proper account of causality that could be used to support explanation capabilities.

4.3 Representing causality in business rule-based systems

Being able to extract and present the causal relations described by business rules could allow one to apply a causal filtering and reduce the obtained traces while

keeping all the valuable information. The idea is to get a richer structure, than a simple trace, on how the rules have been executed and influenced to achieve a result. Thanks to this method, a reduced trace that guarantees no loss of essential justificative information could be given as output during the decision process (at execution time). Moreover, this causal knowledge could also be used to present this reduced trace in a more meaningful way. To represent causality in a rule-based system we will in fact consider two causal models:

- *Causal model of the system*: it models the relations between the business rules of the BRBS (before any execution). The aim through this level is to reduce the size of the decision trace and to provide a qualitative view of the decision logic used by a BRBS;
- *Minimal causal model of the decision*: it represents the links between rules instances that have been traced for a specific decision of the RBRS. This model will be useful for the perspective of explaining a decision for such a system.

The contribution of Halpern and Pearl (2005b) on the notion of causality has been influential in many domains, including databases (Meliou et al., 2010) and specification (Chockler et al., 2008). It relies on three fundamental ideas:

1. the *counterfactual* nature of causation: A causes B if event A and event B occurred and if, had A not occurred, B would not have occurred either. Behind this notion, there is the idea that a cause should be necessary and sufficient.
2. the idea of *contingency*: an event is an *actual cause* if one can find a contingency (a context), where A could be a counter-factual cause for B.
3. *minimality*: no subset of A is an actual cause of B.

Halpern and Pearl (2005b) makes this formal by introducing a setting where systems are affected by variables, either *exogeneous* (fixed by factors external to the systems)

or *endogeneous* (which can serve as causes). Together, they define the signature S of the system. The behavior of the system is captured via *structural equations*.

In this work, we partially rely on the work of (Halpern and Pearl, 2005b) in order to ascribe the causal knowledge about links and relationships that may exist between the different parameters of a rule-based system. This causal representation mainly builds on structural equations. Intuitively and based on the concepts presented in Section 4.2.3, dealing with the execution and orchestration of the business rules, *i.e.* defining the structural equations of a rule-based system, is very challenging though: as we have seen, there are many parameters that may affect the output: the rules themselves, the ruleflow, the ordering of rules within a rule set, and the algorithm used.

4.3.1 Events typology in a business rule decision

We first approach this difficult problem by studying a notion of causality which will make a number of simplifying assumptions. One important question to ask is what events can occur in a rule-based system, *i.e.* how the working memory can be affected. We identify several possible events:

- (v) "variable x is assigned value α",
- (c) "condition c is evaluated" (satisfied or not),
- (a) "action a is applied",
- (r) "rule R is triggered",
- (t) "task T is triggered".

The need, in principle, to differentiate events (c) and (r) can be understood by recalling that some procedures do not directly trigger a rule upon evaluation of their premises (eg. FastPath). This would allow to provide causal relations mentioning this procedural level (see the notion of strategic explanation discussed in Section

3.3). For instance: “the fact that this procedure is used was a cause for triggering this rule".

In the same way, we differentiate theoretically events (a) and (v) because the application of an action does not always result in the same value assignment. (i.e. the same event (a) does not correspond to a specific (v), it can relate to various (v) depending on the context).

However, we shall abstract away from this level and assume that only events (a), (v) and (c) and are meaningful to the decision-maker. Considering this, we propose an analysis which remains independent from the ordering of the rules, and from the algorithm used.

In our context, we aim to describe some connections between the business rules that occurred during a decision process. This requires to look at their actions and conditions. It goes back to observe how the occurrence of an event (a) or (c) can affect the occurrence of another event (a) or (c). As they do not directly relate to business rules, (v) events are not considered in our representation. Moreover, to be able to represent all the aspects of the decision process in a business rule perspective, we have to define a new type of event:

- (a') "the business variable modification resulting from the application of an action a is changed." (ie. "the value of a business variable x used as parameter by an action a is modified".)

Based on some dependencies between events (a) and (a'), the fact that the application of an action can affect the modification resulting from the application of another action can be represented. The idea behind this is that an event of type (a') allows to observe how an event of type (v) corresponding to the occurrence of an event of type (a) changes another event of type (v) corresponding to another event of type (a). In fact, the introduction of an event (a') allows to represent how the modification of the working memory by an action can affect the modification this working memory resulting from another action. It is important to note that our

causal model is rule-centric and so we observe only the events corresponding to the states of rule elements. Thus, the business variables are exogenous and so the events of type (v) are not represented in our causal representation. Nonetheless they are used to provide a contingency for the relations between events of type (a),(a'),(c) and (t).

4.3.2 Defining the signature of a rule-based system

In (Halpern and Pearl, 2005b), a formal setting to capture the causality, where systems are affected by variables, is presented. More precisely, they define the signature $S = (U, V, R)$ of the system, where U is a finite set of exogenous variables (fixed by factors external to the system), V is a finite set of endogenous variables which can serve as causes) and R associates to each variable $Y \in U \cup V$ a non empty set of possible values for Y. The behavior of a system is captured via a set of causal functions F taking the form of structural equations which state causal relations between its endogenous variables. Based on this and as defined by (Halpern and Pearl, 2005b) a system can be represented by its causal model $M(S, F)$. In our context, the signature of a rule-based system is described thanks to the sets of endogenous and exogenous variables presented below.

The set of endogenous variables (V) contains variables of the forms:

- c_i is a boolean variable representing the state of a condition i. This variable is set to *true* if the condition is satisfied and $false$ otherwise.
- a_j is a boolean variable representing the state of an action j. It is set to *true* if the action is triggered and $false$ otherwise.
- a'_k is a boolean variable representing the state of an action k. This variable is set to *true* if any business variable used as parameter by the action is modified and $false$ otherwise.
- t_l is a boolean variable representing the state of a task l. This variable is *true* if the task is triggered and $false$ otherwise.

Based on that, we propose to define an event $\mathcal{X}$ as a particular setting of a set of endogenous variables of the form described above. We note that an event of the form $X_i = x_i$ is called a *primitive event* and represents the assignment of a particular value x_i to a variable X_i.

Definition 4.1. (Event)

$\mathcal{X} = \big[(X_1 = x_1) \wedge \ldots \wedge (X_i = x_i) \wedge \ldots \wedge \ (X_n = x_n)\big]$, where $X_i \in V$ and $x_i \in \{true, false\}$.

For a clarity of reading we abbreviate the events of the form $x = true$ as x and those of the form $x = false$ as $\neg x$ in the rest of the chapter.

Considering such events, some causal relations can be established to describe how the occurrence of an event can affect the occurrence of another one. The validity of these relations may depend on some other "external" factors that are modeled by exogenous variables. The setting of exogenous variables gives a contingency (or a context) under which the first event can be considered as a counterfactual cause of second one.

The set of exogenous variables (U) contains the business variables. For instance, in example 4.3, $U = \{banckruptcy, score, bonus, rate, eligibility\}$. Some values of these variables make a relation between two events counterfactual. This notion is described in the next section. In our case, we have chosen to limit the exogenous variables to the rule-based system business variables. This choice has been motivated because the business variables make sense in the decision domain and thus can have a meaning for the user. Nonetheless, technical parameters (priority among the rules, selected engine algorithm,...) could be added to the exogenous variables to take into account the influence of technical factors that are external to the decision logic described by the business rules. This technical variables should be see at a higher level of exogeneity than the business variables. Taking these variables into account could be interesting for more technical users who would observe how the rule engine settings affect the decision process or for simulating alternative paths at specific

steps of a decision in order to handle "why-not" explanation. As explained before, this kind of explanation is out of the scope of this work.

Considering the variables in U and V, R associates to each of these variables a non empty set of possible values. For all the variables in V the associated sets always contains the values $\{false, true\}$ whereas the sets associated to each exogenous variable depend on the nature of the business variable which is represented. Based on the set of exogenous variables U, the set of endogenous variables V and their possible values R that we previously defined, the signature of a business-rule based system is given by the tuple $S = (U, V, R)$.

4.3.3 Causality between events

During the reasoning process of a business rule-based system, the primitive events corresponding to specific settings of variables in V occur sequentially and under their own contexts (setting of variables in U). As each primitive event occurs alone, it is possible to isolate them and look for "local causation" between two primitive events. This allows us to simplify the problem of capturing and representing causality in a rule-based system. Hence, we propose to introduce the notion of *local cause*, based on the notion of counterfactual cause. The idea behind the local cause is to represent the fact that an event may contribute to cause another event in a specific situation and without the consideration of other events.

Definition 4.2. (Local cause) Let X and Y two primitive events, and $\mathcal{Z}$ an event that forces the value of all the variables in $V \setminus \{X, Y\}$ to $false$. X is a local cause of Y if there is a context $u_{X,Y}$ such that $X \wedge \mathcal{Z} \rightarrow Y$ and $\neg X \wedge \mathcal{Z} \rightarrow \neg Y$.

In other terms, X is a local cause of Y if X is an actual cause of Y without considering the other endogenous variables. This definition of causality is more permissive than the one given by (Halpern and Pearl, 2005b) and allows us to evaluate couples of primitive events in order to know if one may have a causal contribution on the occurrence of another. Moreover, this relation is transitive, *i.e.* if X is a local cause of Y and Y is a local cause of Z than we assume that X contributes

to cause Z. Moreover, we associate to this notion what we call the *validity domain* of a relation between two primitive events X and Y, noted $U_{X,Y}$, which represents the set of all possible contexts (contingencies) $u_{X,Y}$ for which X is a local cause of Y. Therefore, in our context a causal relation between two primitive events can be described as follows.

Definition 4.3. (Causal relation) Let X and Y two primitive events, a causal relation between X and Y, noted $X \xrightarrow{U_{X,Y}} Y$, is a tuple $(X, Y, U_{X,Y})$ such that X is a local cause of Y under the validity domain $U_{X,Y}$.

Moreover, following (Halpern and Pearl, 2005b) this causal relation can be described by a *structural equation* of the form $Y \leftarrow X$. It can also be instantiated under a specific situation by replacing its validity domain $U_{X,Y}$ with a context $u_{X,Y}$ that matches with this situation ($u_{X,Y}$ is a particular setting included in $U_{X,Y}$). This instance is called a *causal link*, represented by the tuple $(X, Y, u_{X,Y})$ and noted $X \xrightarrow{u_{X,Y}} Y$.

It is clear that depending on the nature of the events (see 4.3.1 and 4.3.2) involved in a causal relation, the relation may represent something different and may have a different meaning. In what follows we propose several types of causal relations allowing to describe the connections and links between events in a business rule-based system.

4.3.4 Typology of the relations

In this section, we describe the types of causal relations that exist in our system. Moreover, we note that all types of relations can be translated in terms of business rules. Indeed, for practical purpose, we defined an operator *rule*() which takes an event as parameter and returns the related business rule. This operator can be used to represent relations and links at a higher level of granularity by replacing the events by the corresponding rules.

Execution Relation. An execution relation models the dependence between a condition and an action of the same rule. It is a causal relation between (c) and (a) events.

Definition 4.4. (Execution Relation)
Let c_i and a_k be two primitive events. An execution relation is the causal relation of the form: $c_i \xrightarrow{U_{c_i,a_k}} a_k$ which is valid for all the settings of U_{c_i,a_k}.

Execution links of the form $(c_i \xrightarrow{u_{c_i,a_k}} a_k)$ can be derived from this causal relation.

Example 4.5 (Ex 4.3 cont.)**.** *The rule C has the following execution relations:*

- $c_1 \xrightarrow{U_{c_1,a_1}} a_1$, *where* $U_{c_1,a_1} = rate \in [0.2, 1]$; $c_1 = (rate >= 0.2)$ *and* $a_1 = (rate = rate * 2)$.
- $c_2 \xrightarrow{U_{c_2,a_1}} a_1$, *where* $U_{c_2,a_1} = score \in [12, 100]$, $c_2 = (score >= 12)$ *and* $a_1 = (rate = rate * 2)$.

(In)eligibility relation. An (in)eligibility relation describes the existing connection between the modification of a parameter resulting from the action of a business rule and a state change in the satisfaction of the condition of another business rule. It can be extracted by verifying if there is a validity domain $U_{a_k,(\neg)c_i}$ allowing the causal relation between an event a_k and an event $(\neg)c_i$.

Definition 4.5. ((In)Eligibility Relation)
Let a_k and c_i two primitive events. An (in)eligibility relation is the causal relation of the form: $a_k \xrightarrow{U_{a_k,(\neg)c_i}} (\neg)c_i$ which is valid for all the settings of validity domain $U_{a_k,(\neg)c_i}$.

(In)Eligibility links of the form $(a_k \xrightarrow{u_{a_k,c_i}} c_i)$ can be derived from this causal relation.

Example 4.6 (Ex 4.3 cont.)**.** *The rules A and B have the following eligibility relation:*

$a_1 \xrightarrow{U_{a_1,c_1}} c_1$ *where* $U_{a_1,c_1} = score \in [10, 14]$, $a_1 = score = score + 5$ *and* $c_1 = score >= 10$.

The rules E and F have the following ineligibility relation:
$a_1 \xrightarrow{U_{a_1,\neg c_1}} \neg c_1$ *where* $U_{a_1,\neg c_1} = rate \in [0.1, 0.2[$, $a_1 = rate = rate - 0.1$ *and* $c_1 = score >= 10$.

We note that we will not take into account *in*eligibility relations for building the causal model of a decision (see Section 4.4.3). More precisely, we are interested, for the moment, by capturing what really happened during the decision process. Nonetheless, ineligibility relations may be included when dealing with events that have not occurred and are interested by explaining why not considering such events. An example of why not explanation is given in (Martincic, 2003).

Computation relation. It represents how the modification of a parameter used by an action can change the modification resulting from another action. It can be extracted by analyzing if an action (a_i) modifies a business variable that is used as parameter by another action (a_j) to compute its modification.

Definition 4.6. (Computation Relation)
Let a_i and a'_j two primitive events. A computation relation is the causal relation of the form: $a_i \xrightarrow{U_{a_i,a'_j}} a'_j$ which is valid for all the settings of the validity domain U_{a_i,a'_j}.

In Fig 4.4.3, as an event a'_j relates to a corresponding event a_j, we replace the relations $a_i \xrightarrow{U_{a_i,a'_j}} a'_j$ by $a_i \xrightarrow{U_{a_i,a'_j}} a_j$

Example 4.7 (Ex 4.3 cont.)**.** *The rules B and C have the following computation relation:*
$a_1 \xrightarrow{U_{a_1,a'_1}} a'_1$ *with* $U_{a_1,a'_1} = rate \in [0, 1]$

We note that by using the rule() operator on example 4.7, where $rule(a_1) =$ *Rule B* and $rule(a_1) =$ *Rule C*)), we are able to identify the rules involved in the relation and deduce the relation: *Rule B* $\xrightarrow{U_{a_1,a'_1}}$ *Rule C*.

Based on these different relations, the next section is devoted to describe the construction of causal models for business rule-based systems.

4.3.5 Hierarchical causal model

We are now in a position to present our two-level causal model. Indeed, as previously discussed, a causal representation of a BRBS can be expressed at two levels. First, we have a *ruleflow causal model* which describes relations between tasks (as tasks are associated to rulesets they constrain the relations between rules of different rulesets). Indeed, the ruleflow is responsible for scheduling the tasks and so describes some causal relations between the tasks which are responsible for selecting the sets of business rules that can be applied at each step of the reasoning process (for information, a ruleflow can be seen as a micro-workflow which is usable in a rule context, it consists in a simplified workflow that enables only sequential working). Second, a *rule causal model* which describes relations between business rules (thanks to events related to their conditions and their actions). For each model, we propose to associate a causal model $M(S, F)$ (see section 4.3) described in what follows.

A ruleflow causal model $M(S, F)$. is represented by, on the one hand, its signature $S = (U, T, R_{UT})$, such that U represents the set of exogenous variable representing the business variables used by the Business Rule Based System and $T = \{t_i | i = 1, \ldots, l\}$ is the set of endogenous variables which relates to tasks. Thus, U is used as context to make deterministic the relations defined between the endogenous variable in T. Finally, R_{UT} associates to each variable $Y \in U \cup T$ a non empty set $R_{UT}(Y)$ of possible values for Y. On the other hand, we have the causal function F which associates to each couple of variables $(X, Y) \in T \times T$ a causal function:

$$F_{X,Y} : (\times U' \in U.R_{UT}(U')) \times (X.R_{UT}(X)) \rightarrow R_{UT}(Y) \tag{4.1}$$

This first causal model is used to take into account the fact that a rule A that belongs to a task t_i can affect another rule B that belongs to a task t_j only if t_i is

a local cause of t_j (see Def 4.2) or by considering transitivity t_i is an indirect local cause of t_j.

A rule causal model $M(S, F)$ models causal relations between rules and is represented by its signature $S = (U, V', R_{UV'})$, such that U is the same set as previously and $V' = \{c_i, a_j, a`_k | i = 1, \ldots, n; j = 1, \ldots, m; k = 1, \ldots, l\}$ is the set of endogenous variables which relates to business rule conditions and actions. The associated causal function F which associates to each couple of variables $(X, Y) \in V' \times V'$ a causal function is :

$$F_{X,Y} : (\times U' \in U.R_{UV'}(U')) \times (X.R_{UV'}(X)) \rightarrow R_{UV'}(Y) \tag{4.2}$$

We note that the two causal functions describe how some values of X contributes to cause some other values of Y given some settings of the exogenous variables.

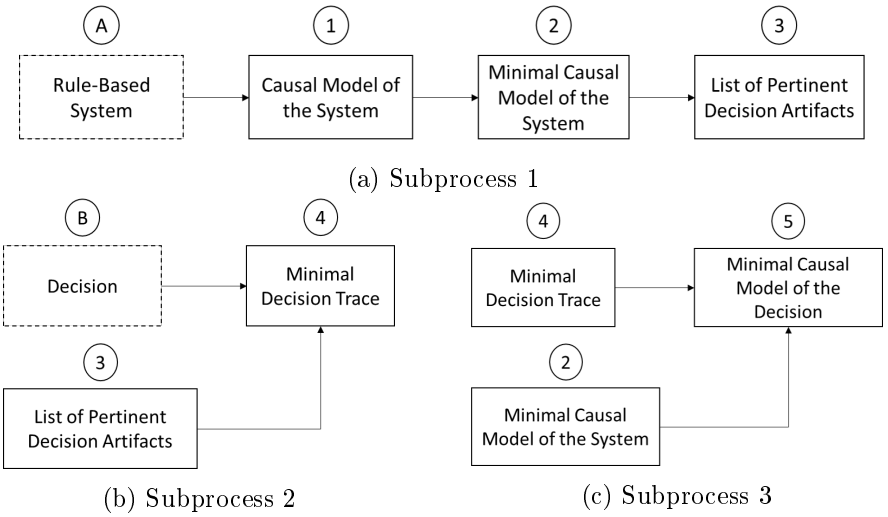
As it was previously stated, studying the causality in a Business Rule Based System has the aim to provide at the end a minimal causal model of a decision that has the advantage to include more relevant information than a decision trace. To obtain such a model, we need, as it is depicted in Figure4.1, to construct a minimal causal model of the system, and the minimal decision trace. In what follows, we describe the process to do that.

4.4 Process of construction of a minimal causal model

We now describe the whole process leading to the construction of the causal model. Figure 4.1 provides a general overview of the different stages, which we describe now.

4.4.1 Causal model of the system and list of relevant events

The first process (Fig. 4.1a) aims to produce the minimal causal model of the business rule-based system and its associated list of relevant decision artifacts. For



(a) Subprocess 1

(b) Subprocess 2

(c) Subprocess 3

Figure 4.1: Method overview: Constructing a minimal causal model of the decision

this purpose, we construct first a causal model of the system by analyzing the business rule-based system and extracting the different causal relationships. After that, this model can be reduced by removing all the relationships that have no path leading to an output parameter (a relationship has a path to an output parameter if it belongs to an oriented chain of relationships that are connected between their right and left parts and lead to a relationship whose the action in the right part modifies an output parameter). After this reduction we obtain a *minimal causal model of the system*. Based on this, a list of the decision artifacts (it can be either business rules or tasks) having an influence on the output parameters can be established. More precisely, the **causal model of the system** is built in two steps. The first step aims to represents dependencies between the tasks of a business rule-based system. Representing such dependencies is important because it allows us to constrain the couples of events that can be considered as candidate for “causal relation".

Indeed, the rules of two independent tasks have no influence on each other. More restrictively, the rules associated to a task t_i can only have a causal influence on the rules associated to any task t_j if the occurrence of t_j is influenced by the occurrence of t_i. On other terms, given two events (t_1) and (t_2) with (a_1) relates to (t_1), and (a'_2) and (c_2) relate to (t_2), then an event (a_1) may have a causal influence on

another event (a'_2) or (c_2) only if (t_1) has a causal influence on (t_2). At the end, we obtain a causal model under the form of a graph such that the nodes corresponds to the endogenous variables of the causal model, and it exists an edges between two variables X and Y if it exists a causal function $F_{X,Y}$ that changes the value of Y depending on the value of X for at least one setting of the exogenous variables (i.e. $U_{x,y}$ is not empty).

Definition 4.7. (Business Rule Based System Causal Network)
A business rule based system causal network is a graph $G = (V, E)$ where V is the set of endogenous variables of the causal model and $E = \{(X, Y) \in V \times V |\ X \xrightarrow{U_{X,Y}} Y\}$

The second step aims at reducing that graph by deleting the nodes having no path to a rule modifying an output parameter. After this, it is possible to extract a list of relevant events, which is obtained by indexing the nodes of this minimal causal network.

4.4.2 Minimal decision trace

The second process (Fig. 4.1b) consists at determining what events of a decision should be traced and even what information should be kept about these events. More precisely, at a decision time, the minimal trace of a decision can be obtained by tracing only the events contained in the list of pertinent decision artifacts (obtained at the previous step) and filtering out the others. This minimal trace is shorter than the original trace and has the advantage to keep only the the elements that could be required to understand the output parameters. For illustration, we give below the minimal trace obtained for the example 4.4.

Example 4.8 (Ex.4.4 Cont.)**.**

```
BEGIN DECISION
-> INPUTS (bankruptcy=false,score=8,bonus=15)
TaskInstance(compute ;<bankruptcy=false,score=8,bonus=15>;...; step 1)
RuleInstance(Rule A ; bankruptcy=false ; score=13 ; step 2)
```

```
RuleInstance(Rule B ; score=13 ; rate=0.28 ; step 3)
RuleInstance(Rule C ; <rate=0.28,score=13>; rate=0.56 ; step 4)
TaskInstance(evaluate ; rate=0.56 ; ... ; step 5)
RuleInstance(Rule E; bonus=15; rate=0.55; step 6)
RuleInstance(Rule F ; rate=0.55; eligibility=true ; step 7)
-> OUTPUTS (eligibility=true,rate=0.55)
END DECISION
```

The obtained trace can then be transformed in a causal graph that provides a qualitative view of the decision, revealing its reasoning. We call this causal graph, the minimal causal model of the decision and we present it in the next sub-section (Sub-Section 4.4.3).

4.4.3 Minimal causal model of the decision

The last process (Fig. 4.1c) aims to produce the minimal causal model of a decision. In order to do that, the minimal causal model of the system and the reduced trace of the decision can be exploited to derive a causal model that fits to this specific decision. More precisely, On the one hand, the trace of the decision provides the events that occurred during the decision with some useful information like their corresponding step and the context. On the other hand, the Business Rule Based System causal network gives information about the relations between these events. For instance, if an event (a) occurred under a context ($u_{(a),(c)}$) causing the occurrence of an event (c), determining a link between (a) and (c) goes back to check if there is a relation "(a) $\xrightarrow{U_{(a),(c)}}$ (c)" in the Business Rule Based System causal network where $u_{(a),(c)}$ is included in $U_{(a),(c)}$.

Example 4.9 (Exemple 4.8 cont.)**.** *minimal causal model of the decision - links list*

Eligibility links:

(1) Rules A and B: $a_1 \xrightarrow{u_{a_1,c_1}} c_1$, *with* $u_{a_1,c_1} : score = 13$

(2) Rules A and C: $a_1 \xrightarrow{u_{a_1,c_2}} c_2$, *with* $u_{a_1,c_2} : score = 13$

(3) Rules B and C: $a_1 \xrightarrow{u_{a_1,c_1}} c_1$, *with* $u_{a_1,c_1} : rate = 0.28$

(4) Rules C and F: $a_1 \xrightarrow{u_{a_1,c_1}} c_1$, *with* $u_{a_1,c_1} : rate = 0.56$

Computation links:

(5) Rules A and B: $a_1 \xrightarrow{u_{a_1,a'_1}} a'_1$, *with* $u_{a_1,a'_1} : score = 13$

(6) Rules B and C: $a_1 \xrightarrow{u_{a_1,a'_1}} a'_1$, *with* $u_{a_1,a'_1} : rate = 28$

(7) Rules B and E: $a_1 \xrightarrow{u_{a_1,a'_1}} a'_1$, *with* $u_{a_1,a'_1} : rate = 0.28$

(8) Rules C and E: $a_1 \xrightarrow{u_{a_1,a'_1}} a'_1$, *with* $u_{a_1,a'_1} : rate = 0.56$

Execution links:

(9) Rule A: $c_1 \xrightarrow{u_{c_1,a_1}} a_1$, *with* $u_{c_1,a_1} : banckruptcy = false$

(10) Rule B: $c_1 \xrightarrow{u_{c_1,a_1}} a_1$, *with* $u_{c_1,a_1} : score = 13$

(11) Rule C: $c_1 \xrightarrow{u_{c_1,a_1}} a_1$, *with* $u_{c_1,a_1} : rate = 0.28$

(12) Rule C: $c_2 \xrightarrow{u_{c_2,a_1}} a_1$, *with* $u_{c_2,a_1} : score = 13$

(13) Rule E: $c_1 \xrightarrow{u_{c_1,a_1}} a_1$, *with* $u_{c_1,a_1} : bonus = 15$

(14) Rule F: $c_1 \xrightarrow{u_{c_1,a_1}} a_1$, *with* $u_{c_1,a_1} : rate = 0.55$

At the end the decision causal model can be represented by a graph where the nodes correspond to the endogenous variables of the causal model and it exists an edges between two nodes X and Y if the causal function $F_{X,Y}$ changes the value of Y depending on the value of X for the setting $u_{X,Y}$ (see Eq 4.2).

Definition 4.8. (Decision Causal Network)
A decision causal network is a graph $G = (V, E)$ where V is the set of endogenous variables of the causal model and $E = \{(X, Y) \in V \times V \mid X \xrightarrow{u_{X,Y}} Y\}$.

We provide below an algorithm that can be used for building the decision causal network based on a trace of the decision and a Business Rule Based System causal model. The complexity of this algorithm is $o(n^2)$, where n is the number of events in the trace. In the algorithm:

- *TraceElement*[*int i*] is the function that outputs the i^{th} event (condition or action) occurred during the decision;

- $Responsible[TraceElement\ element, OutputParameter\ \varphi]$ is the function that outputs a boolean which is *true* if the observed value of the explainable output φ results from *element* (i.e. when occurred, the corresponding action set φ to its final value) and $false$ otherwise;

- $checkCausality(TraceElement\ element_1, TraceElement\ element_2)$ is the function which is in charge for building the causal links. To do this, the function checks first, in the causal model of the decision service, whether a relation (computation, execution, eligibility) exists from $element_1$ to $element_2$, and second if this relation is valid for the values of the business variables associated to $element_1$. If it is the case, the function adds $element_1$ to the set of causes and creates the corresponding causal link.

Algorithm 4 Build the causal model of the decision for the output parameters ϕ based on the trace

Require: $Trace \neq \emptyset$, $Cause = \emptyset$, $\phi \neq \emptyset$.
Ensure: $Cause$ contains the actual causal network presenting all the local causes of ϕ at the end of the algorithm.

```
for all φ ∈ φ do
  set(n, N) % number of elements in Trace
  while n ≠ 0 and Cause = ∅ do
    if Responsible[TraceElement[n], φ] then
      add(Cause, TraceElement[n])
      if step(currentElement) < n then
        set(currentElement, TraceElement[n])
      end if
    end if
    set(n, n − 1)
  end while
end for
while currentElement ≠ ∅ and step(currentElement) > 1 do
  set(currentElement, higherUnmarkedElement(Cause))
  for i = step(currentElement) − 1 to i = 0 do
    checkCausality(TraceElement[i], currentElement)
  end for
  mark(currentElement)
end while
```

Based on this algorithm and on the causal model of rule-based system, the causal

model of the decision can be built. The Figure 4.2 illustrates the decision causal network for example 4.4. It can be read as follows. At step 2, there is an execution link (9) between an event c_1 and an event a_1 related to the instance of "Rule A" belonging to the instance of the task "compute". There is an eligibility link (1) between the same event a_1(occurred at step 2) and the event c_1 (occurred at step 3) that relates to an instance of "Rule B". There is a computation link (8) between a_1(occurred at step 4) which relates to an instance of "Rule C" and a_1(corresponding to a'_1, occurred at step 6) which relates to an instance of "Rule E". Here the event a_1 of "Rule E" and the event a_1 of "Rule F" are responsible for the outputs settings (rate=0.55, eligibility=true) the other nodes are causally added to build the causal graph of the decision.

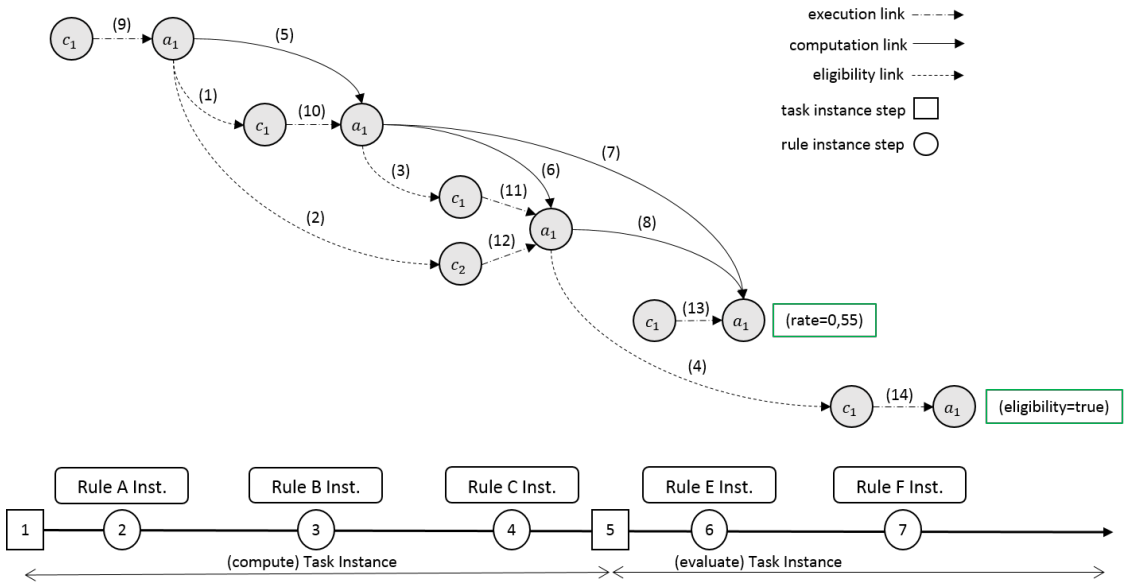


Figure 4.2: Causal network of the decision under a decision artifact perspective

Figure 4.2 presents the causal model of the decision at a rule level by grouping the rule elements in their corresponding rules. Consequently the intrinsic relations are not represented. The algorithm firstly found rule instances E and F which are connected to the outputs parameters "rate" and "eligibility", the other rule instances are added to the causal network with the corresponding links by checking the

causality with a background approach as described in the algorithm.

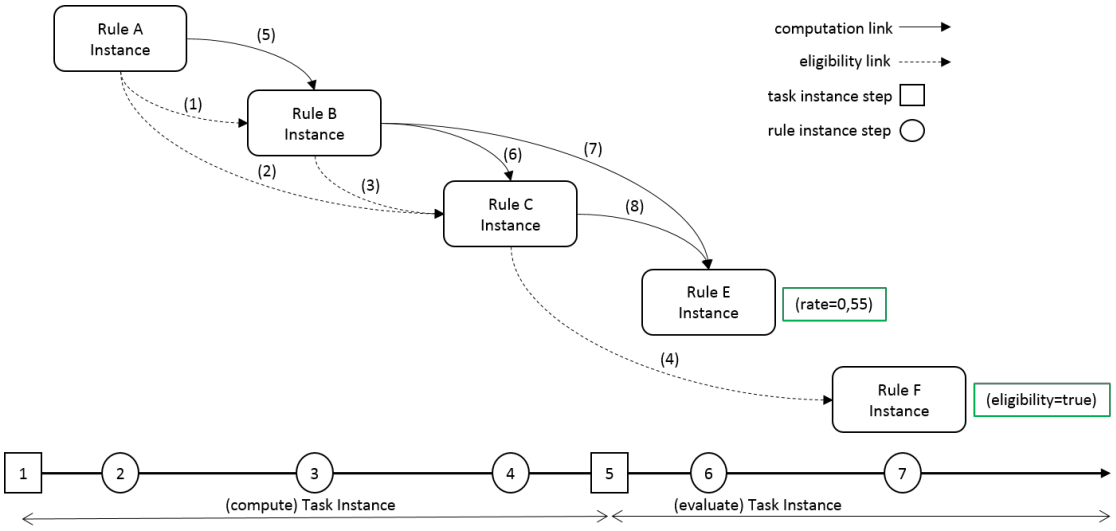


Figure 4.3: An example of causal network at rule level for example 4.4

4.4.4 Conclusion

The causal models presented in this chapter are designed as IBM ODM decision services, but are generic enough to fit with other business rule-based applications. They can be used to create a trace of your business decisions in a minimalistic and meaningful way. Despite the additional cost of such causal models, as they allows the tracing tools to trace only the pertinent information, minimal traces can save space in memory. Moreover, because of their causal nature they guarantee to keep all the information needed to provide a complete qualitative view of the decision process and constitute a good material for explanation generation. This is very useful for business rule-based systems used by insurances and banks which usually have to limit the extra-cost of their tracing tools and store billions of decision traces for later consultations. In addition to the utilities presented above, the proposed causal models may have other applications. Whereas the causal representation of a business rule-based system may be used to increase its transparency and understandability or allows one to provide some explanation capabilities at the system level,

the causal model of the decisions could be a good material to construct more specific explanations. They may contribute to increase the transparency and acceptance of such systems.

Chapter 5

Engineering the causal model

In the previous chapter we presented a methodology to generate causal networks for a Business Rule-Based System (BRBS), either for the system (in that case it captures the relations between the business rules of the BRBS, prior to any execution) in order to reduce the size of the decision trace and to provide a qualitative view of the decision logic used by the system to make its reasoning; or for a specific decision.

Each of these models is supported by a causal representation that can be expressed at different levels:

- at the ruleflow level: a decision process can be guided by at most one main ruleflow.
- at the task level: the tasks are used for selecting the sets of business rules that can be applied at each step of the reasoning process. A directed graph of tasks, called the ruleflow, is responsible for scheduling these tasks. As a result, some causal relations between the tasks can be extracted from the ruleflow and used to constraint the possible causal relations between the business rules of two different tasks.
- at the business rule level: the business rules describes the business logic of the system and, for this reason, they can be analysed to ascribe causal relations.
- at the business rule element level: Split the business rule based on their ato-

mic element allows to ascribe and represent their causal dependencies more precisely. Based on this, a business rule element can be either an action or a condition that belongs to a business rule.

The following scheme gives an overview of the different levels of granularity that may be encountered in a Business Rule-Based System. It can be read in following way. Conditions and action form the atomic level and one business rule can have several conditions and actions. A business rule belongs to a ruleset which is associated to a task and the tasks are managed by a ruleflow which represent the higher level of granularity.

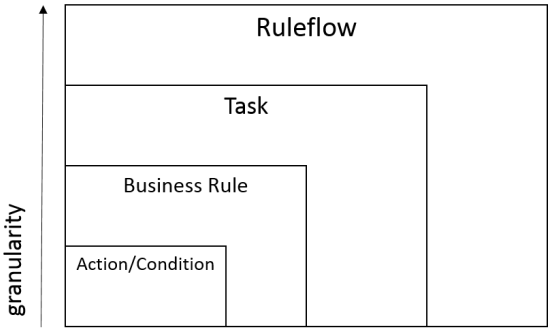


Figure 5.1: Levels of granularity

In practice, as the ruleflow is associated to the higher level of the decision logic, the causal model of the decision corresponds to the ruleflow and captures the causality at the three lower levels which are: the task level, the business rule level and the business rule element level. As it was described in Chapter 4, the construction of these causal models relies on a process that takes the rule-based system to analyse as input and returns several causal models as output (see Figure 4.1).

In this chapter, we describe how we engineer this solution. The implementation of this framework takes place in a Java environment and is tested against decision services developed by using IBM Operational Decision Manager, the Business Rules Management System provided by IBM (see Chapter 2 for more details).

5.1 A Framework for Causal Ascription and Representation in Rule-Based System

As previously described, this framework is based on a global process that can be decomposed into three sub-processes (see Fig.4.1). Each of these process is done at a different time and aims to produce specific materials that are involved in the process of representing causality for a business rule-based system and for its decisions. In practice, the first process, described by Fig. 4.1a, statically analyses the business logic and the Business Object Model of a decision project (ie. rule project) in order to represent the different classes of events that may be involved in automated decisions and ascribe the underlying causal relationships. At the end of this process the material obtained encompasses: (1) a minimal causal model of the decision service (i.e. business rule application) and (2) a list of the relevant classes of events. In what follows, we describe our implementation of the main steps for constructing a minimal causal model of the decision for a BRBS.

5.1.1 Encoding business rules (BR) into BR-Objects

Before analyzing a decision project (ie. rule project), it is important to identify, first, what information contained in a business rule should be captured and how to represent them. Such a representation should be sufficient to capture any information about a business rule that could be used for causal ascription. Moreover, as business rule-based decisions involves instances of business rules, there is a distinction between a business rule and each of its instances. Whereas a business rule represent a class of events having similar properties, each of its instances refers to a specific event of this class that occurred during a decision. Therefore, the events corresponding to different instances of a business rule share common characteristics that are defined in the business rule representation. This distinction is taken into account by using an object-oriented formalism that can be used in order to represent each business rule and each of its instances. We propose to represent a business rule by the UML model depicted in Figure 5.2. Thus, a business rule is represented by

an object "BusinessRule (BR)" that is characterized by:

- an *identifier* (*id*), which is used to identify the business rule and typically contains the business rule name;
- a *list of conditions*, containing "BusinessRuleCondition" objects;
- a *list of actions*, containing "BusinessRuleAction" objects.

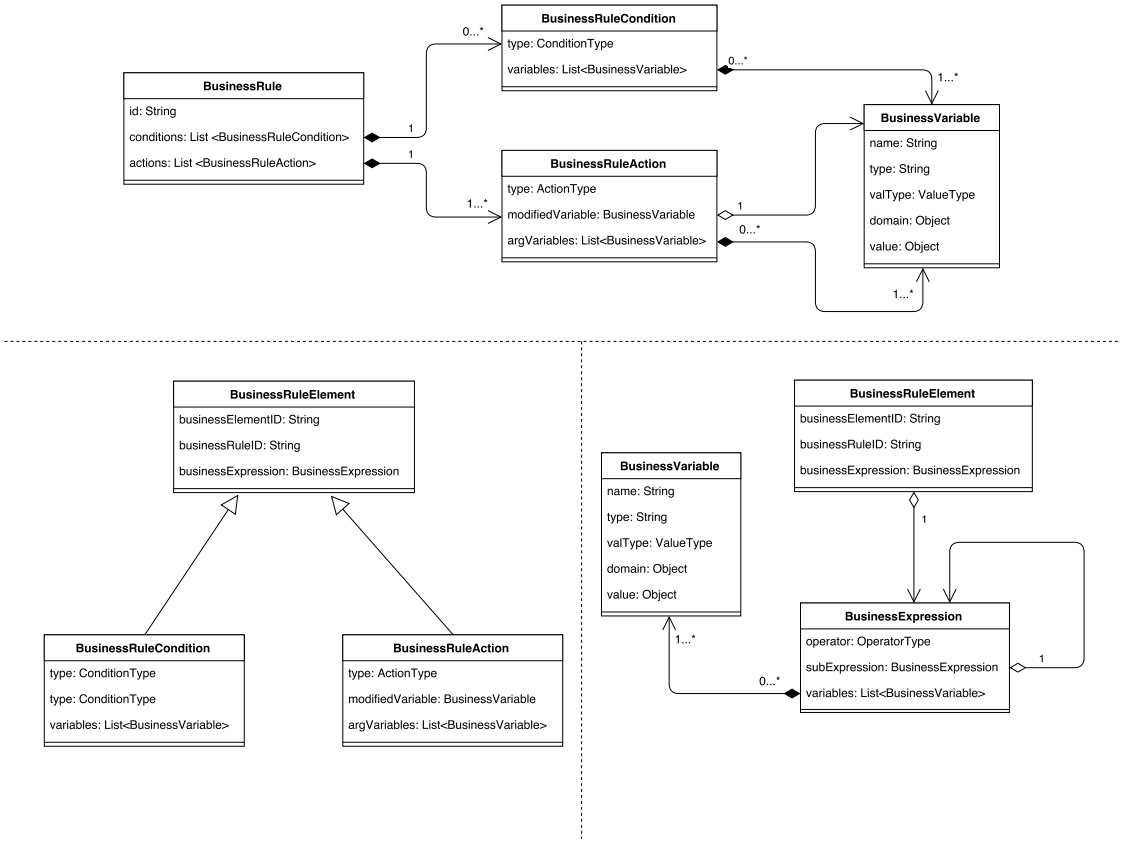


Figure 5.2: UML model: Representing a BR-object (ie. business rule object)

The "BusinessRuleCondition" and "BusinessRuleAction" objects both represent business rule elements and so inherit from a "BusinessRuleElement" class which provides them the following common properties:

- an identifier (`bussinessElementID`) which identifies the concerned business rule element (it can identify either an action or a condition),

- another identifier that allows to find the associated business rule (*businessRuleID*) and a business expression (*businessExpression*) which is a *normalized form*[1] of the part of the business rule statement corresponding to the represented business rule element that can be either a condition or an action.

In addition to these common properties, the *BusinessRuleCondition* and the *BusinessRuleAction* provide specific information:

- A *BusinessRuleCondition* contains a condition type that indicates the type of the represented condition (type) and a list of the business variables evaluated by the condition (variables list).

- A *BusinessRuleAction* informs about the type of the represented action (type), the business variable modified by the corresponding action (modifiedVariable) and the business variables used as arguments by the corresponding action (argVariables list).

Moreover, a business expression is represented by a *BusinessExpression* object that provides information about the type of the operator involved in the business expression (ex: mult, minus, add, div...), the business variables (variables) used as arguments in the current business expression. In addition, a business expression can also uses another business expression (subExpression) as an argument in the current business expression. The business variables refereed in the business expressions are represented by *BusinessVariable* objects that contain information about: the *name* of the corresponding business variable (name), the *type* of the corresponding business variable (type), the *type of the values* of the corresponding business variable (valType), the *domain* of the corresponding business variable (domain) and the *current value* of the corresponding business variable (value).

For illustration, Example 5.1 presents some rules of a simple use-case of loan agreement (the whole example is presented in Appendix A. In this example the

[1] Root form which is independent from the business rule language defined in the decision projects

business rules are shown in their business rule language form (before normalization) and in their normalized form (after normalization).

Example 5.1. *Given a decision project (RuleSet Id: Miniloan Service, Number of rules: 14), the list below represent some of the business rules as they can be read and edited in the user interface. In other terms, the Business rules are in their Business rule language.*

```
rule 0:  (..\Miniloan Service\rules\eligibility\bonusLowCreditScore.brl)
if the credit score of 'the borrower' is at most 400
and the amount of 'the loan' is less than the yearly income of 'the
    borrower'
then set the credit score of 'the borrower' to the credit score of 'the
    borrower' + 100 ;

rule 1: (..\Miniloan Service\rules\eligibility\debt2IncomeRatio.brl)
if
  the yearly repayment of 'the loan' is more than the yearly income of 'the
     borrower' * (0.3 + the credit score of 'the borrower'  / 10000)
  or 10200 is more than the yearly income of 'the borrower' * the yearly
    repayment of 'the loan'
then
  add "Too big Debt-To-Income ratio" to the messages of 'the loan' ;
  reject 'the loan' ;

rule 2: (..\Miniloan Service\rules\eligibility\duration2Score.brl)
if
  the duration of 'the loan' is at least 180
  and the duration of 'the loan' is at most 480
then
  set the credit score of 'the borrower' to the credit score of 'the
    borrower' + the duration of 'the loan' / 8 ;
```

The list below describes the previous business rules after normalization.

```
rule 0 (Id: bonusLowCreditScore)
condition 0: [belongs to rule: bonusLowCreditScore]
isLessThan( miniloan.Borrower/creditScore/GETTER#0 , 400 )
condition 1: [belongs to rule: bonusLowCreditScore]
isLessThan( miniloan.Loan/amount/GETTER#0 ,
    miniloan.Borrower/yearlyIncome/GETTER#0)
action 0: [belongs to rule: bonusLowCreditScore]
creditScore.SETTER( add( miniloan.Borrower/creditScore/GETTER#0 , 100) )

rule 1 (Id: debt2IncomeRatio)
condition 0: [belongs to rule: debt2IncomeRatio]
isGreaterThan( miniloan.Loan/yearlyRepayment/GETTER#0 , mult(
    miniloan.Borrower/yearlyIncome/GETTER#0 , add( 0.3 , div(
    miniloan.Borrower/creditScore/GETTER#0 , 10000 ) ) ) )
condition 1: [belongs to rule: debt2IncomeRatio]
```

```
isGreaterThan( 10200 , mult( miniloan.Borrower/yearlyIncome/GETTER#0 ,
    miniloan.Loan/yearlyRepayment/GETTER#0 ) )
action 0: [belongs to rule: debt2IncomeRatio]
addToMessage( Too big Debt-To-Income ratio )
action 1: [belongs to rule: debt2IncomeRatio]
approved.SETTER( false )

rule 2 (Id: duration2Score)
condition 0: [belongs to rule: duration2Score]
isGreaterThan( miniloan.Loan/duration/GETTER#0 , 180 )
condition 1: [belongs to rule: duration2Score]
isLessThan( miniloan.Loan/duration/GETTER#0 , 480 )
action 1: [belongs to rule: duration2Score]
creditScore.SETTER( add( miniloan.Borrower/creditScore/GETTER#0 , div(
    miniloan.Loan/duration/GETTER#0 , 8 ) ) )
```

5.1.2 Extracting the business rules of a decision project

The second step of the process consists in the extraction of useful information from the business rules contained in a decision project (i.e. a rule project). As previously described, the BR-object formalism provided in Fig. 5.2 states the properties and the information that must be kept when representing a business rule. Based on that, this step aims to generate and feed the BR-objects which correspond to the business rules of a decision project (i.e. a rule project). Here we describe the process of extracting business rule information from a decision project (i.e. a rule project) to generate and feed the corresponding BR-objects. Figure 5.3 provides a global view of the java classes involved in the extraction of the business rules information contained in a decision project (i.e. a rule project).

Therefore, for the extraction and the representation of business rules, we propose to follow the following steps:

- make an index of the business rules contained in a decision project (i.e. a rule project). In IBM Operational Decision Manager, each business rule is modeled by a brl-file (file with the extension .brl).

- for each indexed brl-file (as it is illustrated in Fig. 5.4):

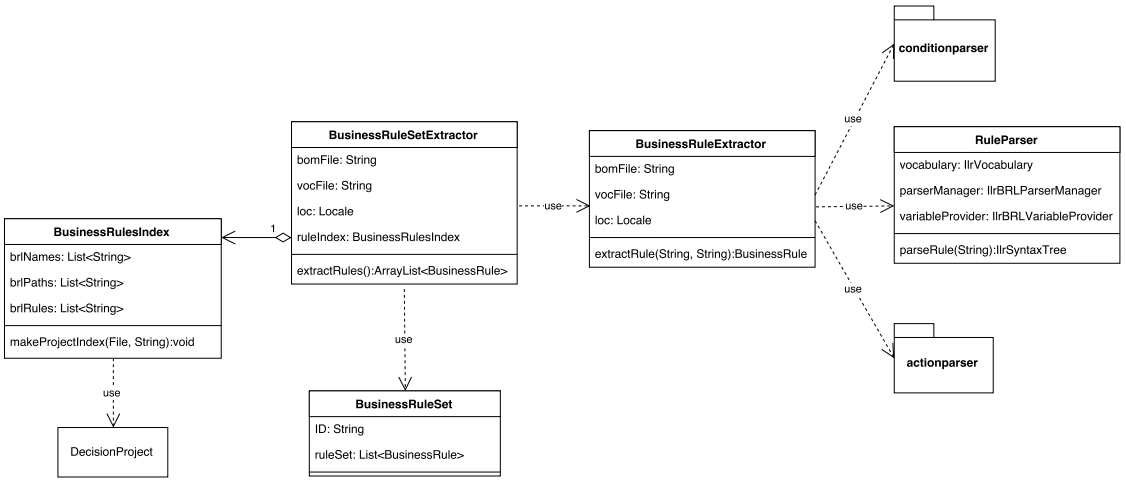


Figure 5.3: UML model: Extract business rules from a decision project (i.e. a rule project)

- extract and clean the text content in order to keep only the business rule statements written in the business rule language defined in the decision project,
- parse the business rule statements (with *parseRule(String)*) to construct the corresponding syntax tree (IlrSyntaxTree). This operation requires a set of files contained in the `bom` folder: the `bom` file (it contains references and information about the properties of the business objects, variables and functions defined in the decision project), a voc file (it contains the vocabulary related to the use of the business objects, variables and functions referred in the bom file and used in the business rule statement) and a loc file (it contains the locale that identifies the language used in the decision project to construct the business rule statements.),
- extract the relevant information from the business rule syntax tree that has been previously obtained. Then, normalize the extracted information (ie. put it in a root form which is independent from the verbalization used in the business rule statements) and use it to feed the fields of the corresponding BR-object. This task requires the use of a condition parser (that identifies, extracts and normalizes the different pieces of information needed about of each condition in the business rule syntax tree) and an

action parser (that identifies, extracts and normalizes the different pieces of information needed about each action in the business rule syntax tree),

– add the obtained BR-object to the list representing the business rules extracted from the decision project (ie rule project).

- create a ruleset with an identifier and its associated list of BR-objects. It is represented by an object "BusinessRuleSet".

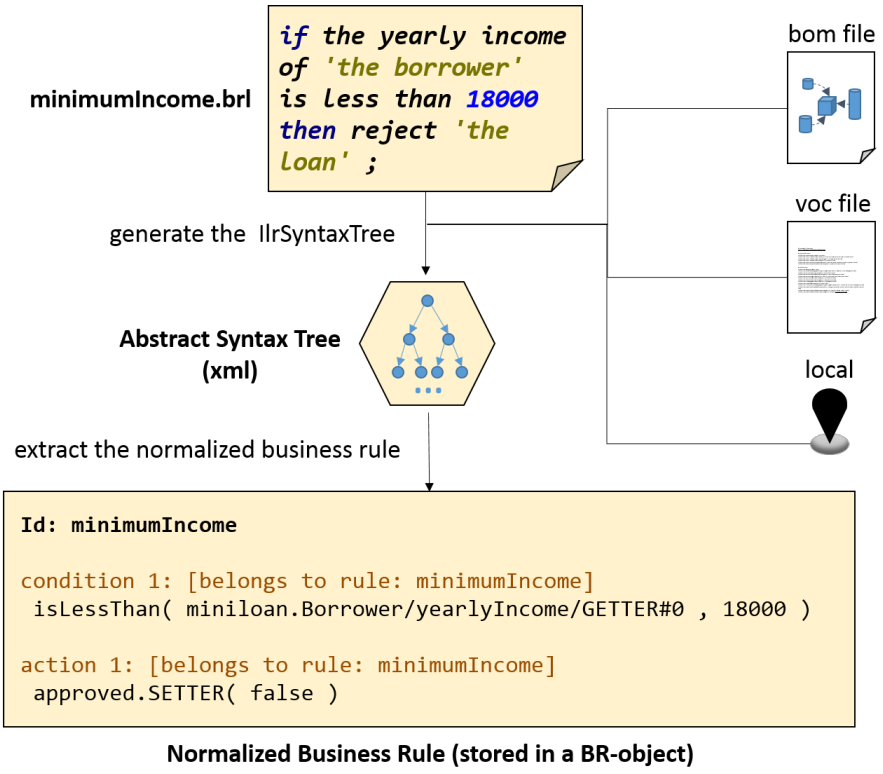


Figure 5.4: Example of BRL extraction

5.1.3 Ascribing causal relations between the business rule elements

This step aims to ascribe the causal relationships existing between the BR-objects that have been created in the previous steps. Indeed, the causal ascription relies on the evaluation of the business rule elements associated to each BR-object. The figure 5.5 describes the different classes involved in the construction of a minimal causal

model of a decision project (i.e. a rule project). The "RuleSetRelationsAnalyzer" uses a "RelationAnalyzer" to evaluate each possible relation, creates the corresponding "CausalRelations" and returns them in a "RulesetCausalModel". The user can select some outputs for which he needs explanations among the business variables used as outputs by the decision service. We call them *explainable outputs*. Based on a list of explainable outputs (business variables), the *causalFiltering* function is then used to reduce the model and generate the corresponding list of relevant elements (i.e. pertinent elements) ("pertinentElements").

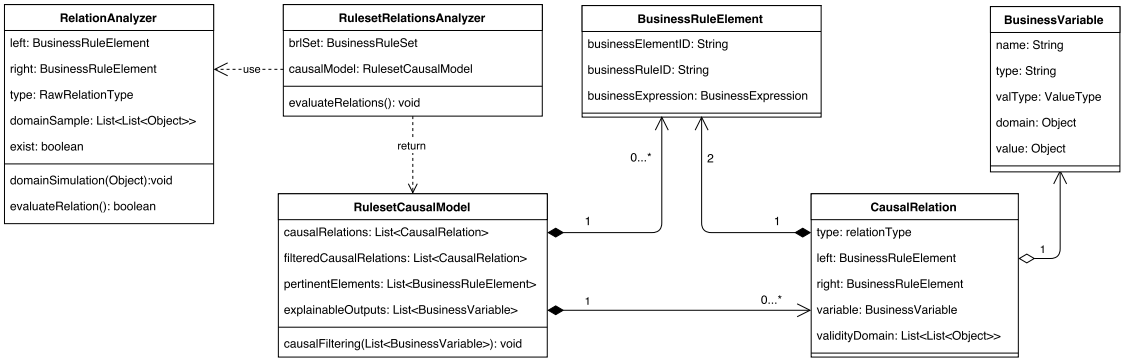


Figure 5.5: RelationsAnalyzer

The process of constructing and reducing a causal model of the decision service can be split into several sub-steps:

- Make a preliminary analysis of the business rule elements (conditions and actions) to filter out the "bad candidates", ordered couples of business rule elements that have no chance to have causal relationships. This preliminary analysis only evaluates the references contained in the business rule elements. These references correspond to business rules, business objects and business variables accessed by the business rule elements. Based on this:
  - It is possible to model how the satisfaction of a condition can impact the triggering of an action that belongs to the same business rule. This influence is modeled by an execution relation (see Section 4.3.4). Thus, determining the ascription of an execution relation boils down to verify if

the condition and the action involved in this relation belong to the same BR-object. Such a relation is valid for all possible settings of the business variables manipulated or evaluated by the concerned condition and action.

- It is possible to model the impact of the modification resulting from an action on the result of another action. This influence is represented by a computation relation (see Section 4.3.4). The determination of a computation relation relies on an analysis of the variable modified in the first action (cause) and used as argument in the second action (consequence). In practice, it is sufficient to verify that the variable, referred as "modifiedVariable" in the first action, is included in the variables referred as "argVariables" in the second action. Such a relation is valid for all possible settings of the business variables manipulated or evaluated by the concerned condition and action.
- it is possible to model how the triggering of an action can change the satisfaction of a condition. This influence can be represented either by an eligibility or an ineligibility relation. For these kinds of relations, the raw analysis allows to select good candidates for (in)eligibility relation (see Section 4.3.4). It amounts to verify if the business variable or object modified by the action is evaluated by the condition. If it is the case, the couple $\langle action, condition \rangle$ is added to the list of potential ("good") candidates. Each of these candidates need to be tested against the different settings of its exogenous variables (i.e. the different combination of values that could be taken by the business variables involved in the relation) to capture its causal function (see Eq. 4.2) and discriminate the wrong candidates.

- Make a more precise analysis of the "good candidates" that aims to find capture the causal function of each candidates and filter out the wrong candidates, those for which no causal relation has been found. In the case of execution and computation relations, the raw causal analysis is sufficient but, in the case

of eligibility and ineligibility relations, a more precise evaluation is required in order to determine if a selected candidate presents an ineligibility relation, an eligibility relation or none of those. Figure 5.6 gives a global view of the approach. In it, the system aims to evaluate if a candidate $\{a_1, c_1\}$ presents an ineligibility and/or an eligibility relation or none of those. The first step is to find the set of business variables $\langle x_1, ..., x_n \rangle$ involved in the potential relation. The δ corresponds to the sampling step that we have to set for a variable x, x_{min} and x_{max} correspond to the lower and upper bounds of the variable domain and m allows to size for the sampling. Thus, for a set of n variables, the size of the corresponding sampling will be m^n. Based on that, for each generated sampling, the effect corresponding to the function that should be applied by a_1 is computed. Then the conditional test of c_1 is test against the values of the sampling and against the values of the sampling update by a_1. From that we look for the *state* of c_1 and we have three possibilities: (1) If it goes from true to false it is an ineligibility relation, (2) if it goes from false to true it is an eligibility relation, (3) if it does not change there is no (in)eligibility relations for the values of the sampling.

As we described above, this ascription of causality is supported by testing the couple $\langle$ action, condition $\rangle$ for each settings of the business variables referred in the condition and in the action. If the modification that results from the application of the action changes the state of satisfaction of the condition then the relation can be added with its type (eligibility or ineligibility) and its corresponding settings of business variables is added to its "validityDomain. Other relations that involved the same couple $\langle$ action, condition $\rangle$ and which are of the same type but with different settings of business variables can then be grouped with the others by adding the new settings to the "validityDomain" of the causal relation. Based on this "validityDomain", the causal function of the relation is approximated.

This mechanism of causal ascription relies on a testing of the action and the

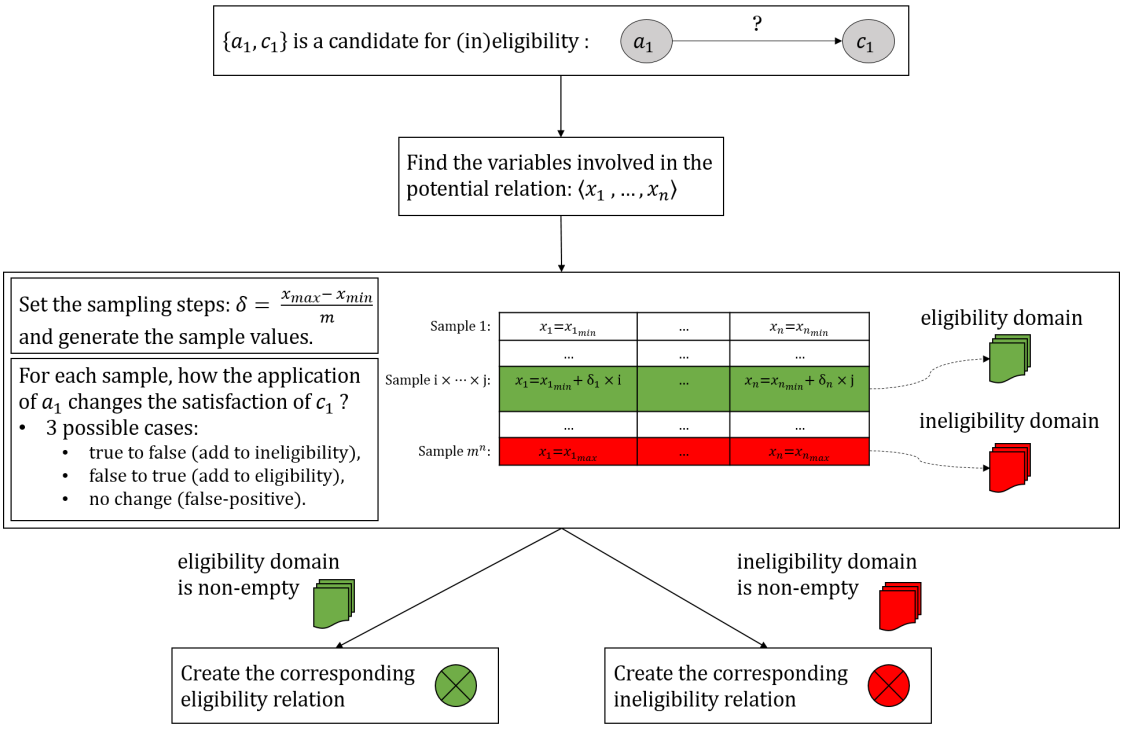


Figure 5.6: Global view of the causal ascription of candidate for (in)eligibility relation

condition for a given setting of the business variables involved.

– Figure 5.7 describes the classes associated to the process of simulating the application of an action. The "ActionInterpreter" aims to interpret the business expression given by the "BusinessRuleAction" object and to extract the business variables referenced in this business expression to enable the computation of the corresponding action. Based on this interpretation, the "ActionSimulator" can simulate the treatment of the corresponding action for all the possible settings of the business variables referenced in it (whose the values are given by "testSample").

– In the same way, Figure 5.8 describes the classes associated to the process of simulating a condition. The "ConditionInterpreter" aims to interpret the business expression given by the "BusinessRuleCondition" object and to extract the business variables referenced in this business expression to enable the computation of the corresponding condition. Based on this

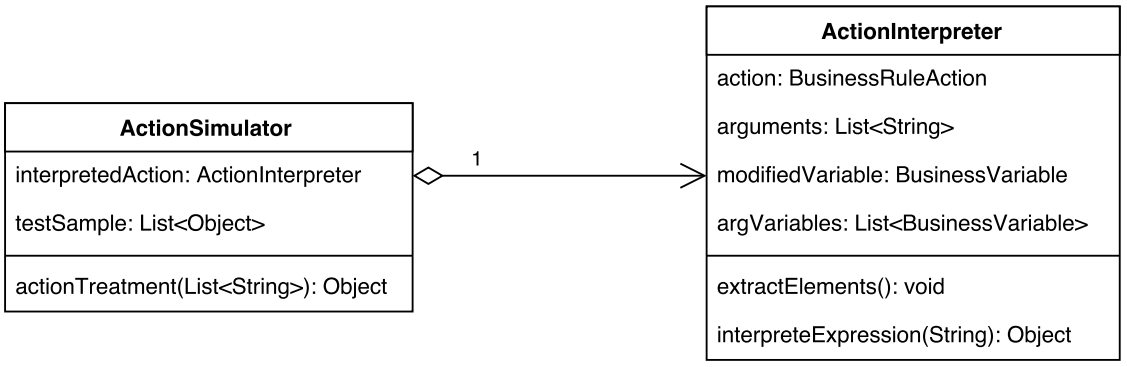


Figure 5.7: Action simulation

interpretation, the "ConditionSimulator" can simulate the treatment of the corresponding condition for all the possible settings (whose the values are given by "testSample" which corresponds to a sampling of the possible combinations of values allowed by the business variable domains).

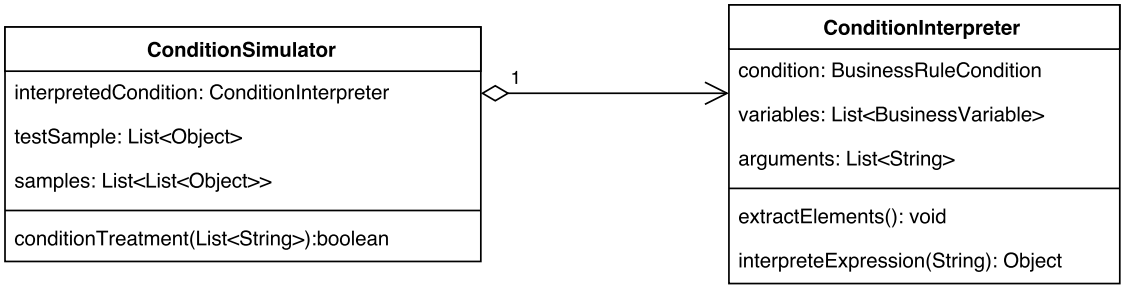


Figure 5.8: Condition simulation

- Based on all the causal relations that have been ascribed and represented, a causal model is built ("RulesetCausalModel").
- A causal filtering is applied to this causal model based on a list of explainable outputs that is given by the user ("explainableOutputs"). After reduction, a minimal causal model of the decision service and the corresponding list of relevant elements ("pertinentElements") are obtained.

Example 5.2 presents the list of causal relations from the static analysis before and after reduction.

Example 5.2. *List of the causal relations ascribed before reduction of the business rules presented in Example 5.1*

```
BusinessRule left: bonusLowCreditScore
BusinessRule right: bonusLowCreditScore
Computation relation between: a0 and a0 on: miniloan.Borrower/creditScore/
    GETTER#0

BusinessRule left: debt2IncomeRatio
BusinessRule right: debt2IncomeRatio
execution relation between: c1 and a0

BusinessRule left: bonusLowCreditScore
BusinessRule right: debt2IncomeRatio
ineligibility relation between: a0 and c0 on: miniloan.Borrower/creditScore
    /GETTER#0

BusinessRule left: bonusLowCreditScore
BusinessRule right: score2Rate
eligibility relation between: a0 and c2 on: miniloan.Borrower/creditScore/
    GETTER#0

...

Number of elements in causal model: 111
```

In the following, Examples 5.3 and 5.4 present the list of causal relations kept after a reduction based on the explainable outputs. The first one considers the variables: "year Interest Rate", "the Credit score" and the "amount". The second one considers the "approved" variable.

Example 5.3. *Causal model of the decision service after reduction on explainable outputs: yearlyInterestRate, creditScore, amount*

```
BusinessRule left: bonusLowCreditScore
BusinessRule right: bonusLowCreditScore
execution relation between: c0 and a0

BusinessRule left: bonusLowCreditScore
BusinessRule right: bonusLowCreditScore
execution relation between: c1 and a0

BusinessRule left: veryHighIncome2Score
BusinessRule right: score2Rate
eligibility relation between: a0 and c2 on: miniloan.Borrower/creditScore/
    GETTER#0

BusinessRule left: veryLowIncome2Score
```

```
BusinessRule right: score2Rate
ineligibility relation between: a0 and c2 on: miniloan.Borrower/creditScore
    /GETTER#0

...

Number of elements in filtered causal model: 84 for explainable outputs:
    miniloan.Borrower/creditScore/GETTER#0, miniloan.Loan/amount/GETTER#0,
    miniloan.Loan/yearlyInterestRate/GETTER#0.
```

Example 5.4. *Causal model of the decision service after reduction on explainable output: approved*

```


BusinessRule left: bonusLowCreditScore
BusinessRule right: bonusLowCreditScore
Computation relation between: a0 and a0 on: miniloan.Borrower/creditScore/
    GETTER#0

BusinessRule left: debt2IncomeRatio
BusinessRule right: debt2IncomeRatio
execution relation between: c0 and a1

BusinessRule left: lowIncome2Score
BusinessRule right: debt2IncomeRatio
eligibility relation between: a0 and c0 on: miniloan.Borrower/creditScore/
    GETTER#0


BusinessRule left: veryHighIncome2Score
BusinessRule right: bonusLowCreditScore
ineligibility relation between: a0 and c0 on: miniloan.Borrower/creditScore
    /GETTER#0

...

Number of elements in filtered causal model: 88 for explainable outputs:
miniloan.Loan/approved/GETTER#0.
```

In the following, we present the list of the relevant classes of events that should be considered when tracing the execution of a business rule-based decision, before and after reduction. For the later we consider the same explainable outputs as in the previous examples.

Example 5.5. *List of relevant events before any reduction*

```
bonusLowCreditScore.c0, bonusLowCreditScore.c1, bonusLowCreditScore.a0
debt2IncomeRatio.c0, debt2IncomeRatio.c1, debt2IncomeRatio.a0,
    debt2IncomeRatio.a1
duration2Score.c0, duration2Score.c1, duration2Score.a0
highIncome2Score.c0, highIncome2Score.c1, highIncome2Score.a0
lowAmountPenality.c0, lowAmountPenality.a0
lowIncome2Score.c0, lowIncome2Score.c1, lowIncome2Score.a0
messageElig.c0, messageElig.a0
score2Rate.c0, score2Rate.c1, score2Rate.c2, score2Rate.a0
veryHighIncome2Score.c0, veryHighIncome2Score.a0
veryLowIncome2Score.c0, veryLowIncome2Score.a0
maximumAmount.c0, maximumAmount.a0, maximumAmount.a1
minimumAmount.c0, minimumAmount.a0, minimumAmount.a1
minimumCreditScore.c0, minimumCreditScore.a0, minimumCreditScore.a1
minimumIncome.c0, minimumIncome.a0

Total number of elements before reduction: 39
```

Example 5.6. *Relevant events based on the selected explainable outputs: yearlyInterestRate, creditScore, amount*

```
bonusLowCreditScore.a0, bonusLowCreditScore.c0, bonusLowCreditScore.c1
duration2Score.a0, duration2Score.c0, duration2Score.c1
highIncome2Score.a0, highIncome2Score.c0, highIncome2Score.c1
lowAmountPenality.a0, lowAmountPenality.c0
lowIncome2Score.a0, lowIncome2Score.c0, lowIncome2Score.c1
veryHighIncome2Score.a0, veryHighIncome2Score.c0
veryLowIncome2Score.a0, veryLowIncome2Score.c0
score2Rate.a0, score2Rate.c0, score2Rate.c1, score2Rate.c2

Number of elements: 22 for explainable outputs:
miniloan.Borrower/creditScore/GETTER#0, miniloan.Loan/amount/GETTER#0,
    miniloan.Loan/yearlyInterestRate/GETTER#0.
```

Example 5.7. *Relevant events based on the selected explainable output: approved*

```
debt2IncomeRatio.a1, debt2IncomeRatio.c0, debt2IncomeRatio.c1
maximumAmount.a1, maximumAmount.c0
minimumAmount.a1, minimumAmount.c0
minimumCreditScore.a1, minimumCreditScore.c0
minimumIncome.a0, minimumIncome.c0
bonusLowCreditScore.a0
duration2Score.a0
highIncome2Score.a0
lowAmountPenality.a0
lowIncome2Score.a0
veryHighIncome2Score.a0
veryLowIncome2Score.a0
bonusLowCreditScore.c0, bonusLowCreditScore.c1
duration2Score.c0, duration2Score.c1
highIncome2Score.c0, highIncome2Score.c1
```

```
lowAmountPenality.c0
lowIncome2Score.c0, lowIncome2Score.c1
veryHighIncome2Score.c0
veryLowIncome2Score.c0

Number of elements: 29 for explainable outputs:
miniloan.Loan/approved/GETTER#0.
```

5.1.4 Recording the minimal trace of a decision

In this step, we present our mechanism for recording a minimal trace of a decision. It can be seen as an improvement of the classical rule engine traces. In fact, a minimal trace is more compact and informative than a classical rule engine trace because in the case of a minimal trace the system records only the minimal and sufficient information that is required to justify the explainable outputs. This task requires the use of the list of relevant elements that has been generated during the previous step. The idea here, is to keep only the events whose forms are allowed by the classes of events contained in the list of relevant business rule elements. If a class of events is not in this list, it means that all the events of this type will not have any impact on any causal process involved in an output that need to be explained. The tracing of this minimal trace ("DecisionEventHistory") is managed by the classes represented in Figure 5.9.

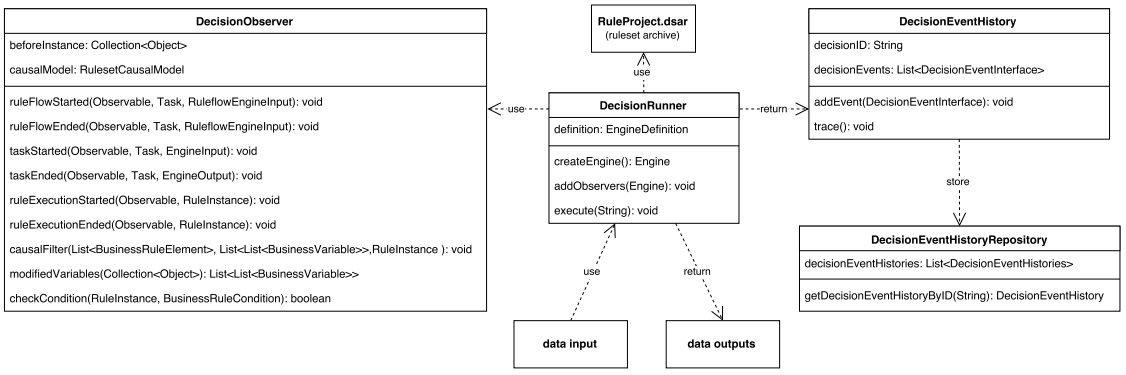


Figure 5.9: Trace the execution

In this UML model, the class "DecisionRunner" is responsible for the creation and

the configuration of the rule engine that is used to run the decision. The information about the rule engine configuration and the rule project is contained in the `.dsar` file. The class "DecisionObserver" describes the behavior of the observers that can be attached to the rule engine created in the "DecisionRunner" class by using the function "addObservers(Engine)". The running of the decision is managed by the function "execute(String)" where the string used as parameter is a list of data inputs and where the returned elements are the data outputs. A "DecisionEventHistory" object is generated during the execution based on the events recorded thanks to the decision observers.

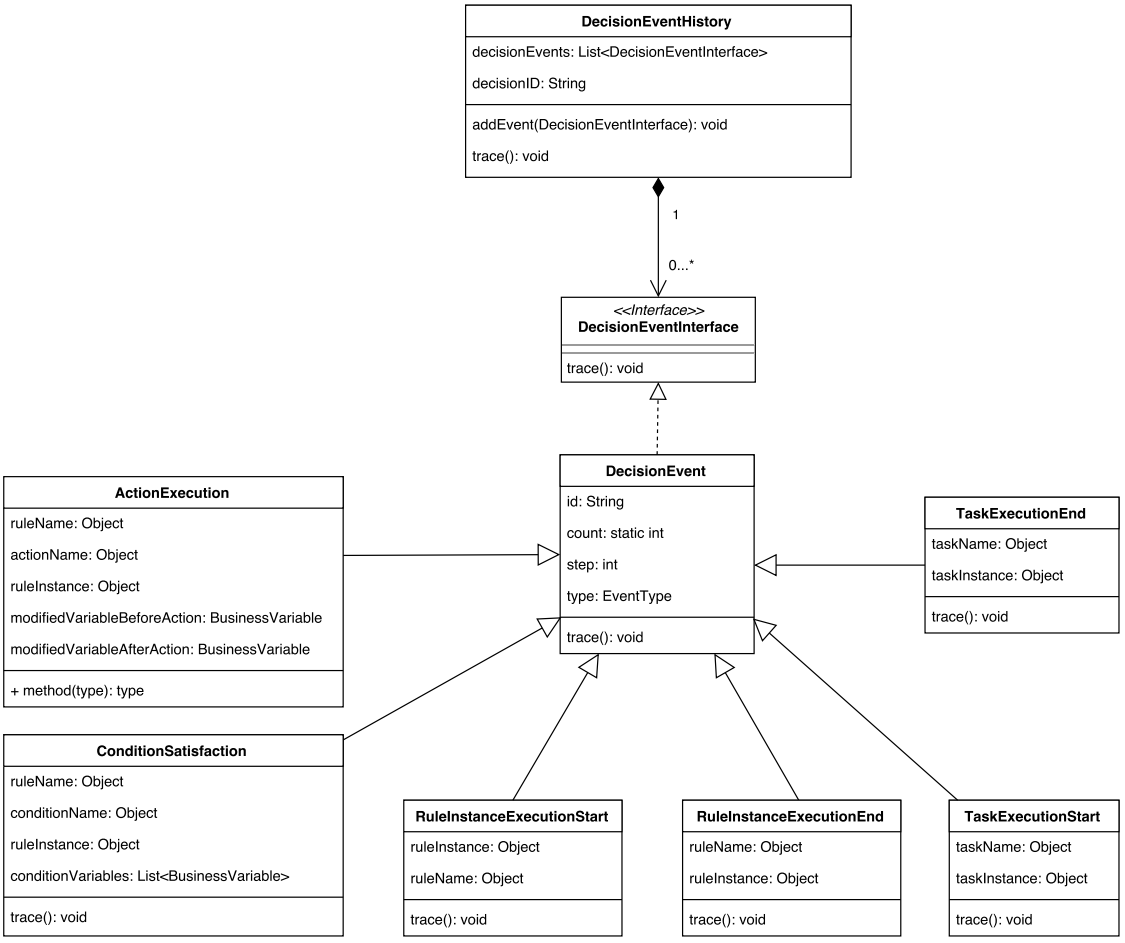


Figure 5.10: Decision Event History Classes

Figure 5.10 provides more details about the structure of these "DecisionEventHistory" objects. In this object, the events are captured thanks to the "Decisio-

nEvents" list that summarizes the different kinds of decision events that occurred during a decision. These decision events can be of various nature ("ActionExecution", "ConditionSatisfaction", "RuleInstanceExecutionStart", "RuleInstanceExecutionEnd", "TaskExecutionStart" and "TaskExecutionEnd") but all of them implement the interface "DecisionEventInterface". Each of the "DecisionEventHistory" objects that have been created transcribes a corresponding decision process and can be stored in the "DecisionEventHistoryRepository" for later use. Based on this material, the next step focuses on the construction of causal models that explain specific decisions or subsets of decisions upon request. Example 5.8 presents a minimal decision trace for the loan agreement usecase previously presented.

Example 5.8. *Minimal decision trace obtained for the business rules presented in the example 5.1 and the inputs (borrowerName: "John Doe", borrowerCreditScore: 399, borrowerYearlyIncome: 180000, loanAmount: 110000, loanDuration: 240, yearlyInterestRate: 0.05). This minimal decision trace for the explainable outputs: yearlyInterestRate, creditScore, amount*

```
CausalScope miniloan(com.ibm.rules.generated.ruleflow.miniloan.
    TaskDefinition@c7572720) {
     CausalScope eligibility.bonusLowCreditScore(com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@13785ed) {
       bonusLowCreditScore.a0(com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@13785ed)
       bonusLowCreditScore.c0(com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@13785ed)
       bonusLowCreditScore.c1(com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@13785ed)
     }
     CausalScope eligibility.highIncome2Score(com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@5dd720f5) {
       highIncome2Score.a0(com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@5dd720f5)
       highIncome2Score.c0(com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@5dd720f5)
       highIncome2Score.c1(com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@5dd720f5)
     }
     CausalScope eligibility.duration2Score(com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@8cfed081) {
       duration2Score.a0(com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@8cfed081)
       duration2Score.c0(com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@8cfed081)
       duration2Score.c1(com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@8cfed081)
     }
     CausalScope eligibility.messageElig(com.ibm.rules.engine.rete.runtime.
    util.RuleInstanceImpl@43df4a67) {
     }
}
```

List of decision events without causal filter.

```
unfiltered events: (size = 11 )

bonusLowCreditScore.a0
bonusLowCreditScore.c0
bonusLowCreditScore.c1
highIncome2Score.a0
highIncome2Score.c0
highIncome2Score.c1
duration2Score.a0
duration2Score.c0
duration2Score.c1
messageElig.a0
messageElig.c0
```

List of decision events with a filter on the explainable outputs: yearlyInterestRate, creditScore, amount.

```
filtered events: (size = 9 )

bonusLowCreditScore.a0
bonusLowCreditScore.c0
bonusLowCreditScore.c1
highIncome2Score.a0
highIncome2Score.c0
highIncome2Score.c1
duration2Score.a0
duration2Score.c0
duration2Score.c1
```

5.1.5 Constructing the minimal causal model of a decision

This step aims to build a minimal causal model of a specific decision (represented by a "CausalInstanceModel" object) upon request, in the sense that only the paths leading to explainable outputs are kept. Thus, the construction of this model mainly relies on two elements built during the previous steps:

- the minimal causal model of the decision service (represented by a "RulesetCausalModel" object) obtained in at the step 3 (see Section 5.1.3.).
- the history of the events that corresponds to this decision (represented by a "DecisionEventHistory" object).

Figure 5.11 presents the classes involved in the construction of the causal model of a decision (represented by a "CausalInstanceModel" object). In this UML model, the causal model server (represented by a "CausalModelServer" object) is responsible for providing the causal model of the decision service (represented by a "CausalModel" object, a lighter version of the "RulesetCausalModel"), the decision history (represented by a "DecisionEventHistory" object) selected in the histories repository (represented by a "DecisionEventHistoryRepository" object) and the causal model of the decision (represented by a "CausalInstanceModel" object). The causal model of the decision service (represented by a "CausalModel" object) contains a list of causal relations (represented by a "CausalRelation" objects) that contains information about the type of the relation (represented by the field "type"), the classes of the business variables involved in the relation (represented by the fields "originClasses" and

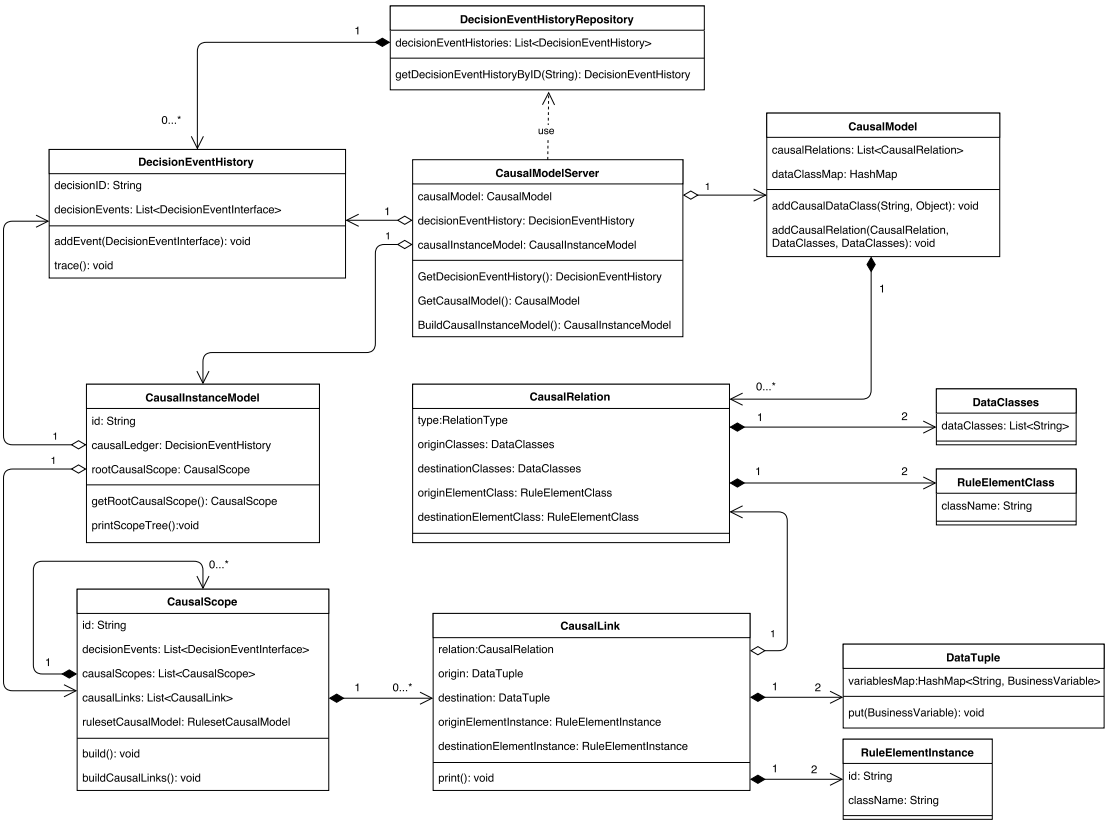


Figure 5.11: Causal Model Server Classes

"destinationClasses") and the rule elements involved in the relations (represented by the fields "originElementClass" and "destinationElementClass"). Based on that, a causal model of a decision (represented by a "CausalInstanceModel" object) has: an *id*, the related decision history (represented by a "DecisionEventHistoryRepository" object) and a causal scope (represented by a "CausalScope" object) that is in charge of the generation of a list of causal links (represented by a "CausalLink" objects). Each causal link refers to the corresponding causal relation and provides the data tuples (represented by "DataTuple" objects) corresponding to the business variable values and rule elements instances (represented by "RuleElementInstance" objects) corresponding to the two instances of business rule element involved in the causal link. Example B.1 presents the relations of the minimal causal model of the decision whose simplified graphical representations are shown in Figure 5.12 and Figure 5.13.

For example, in Figure 5.12, the dashed arrow between the action "bonusLo-

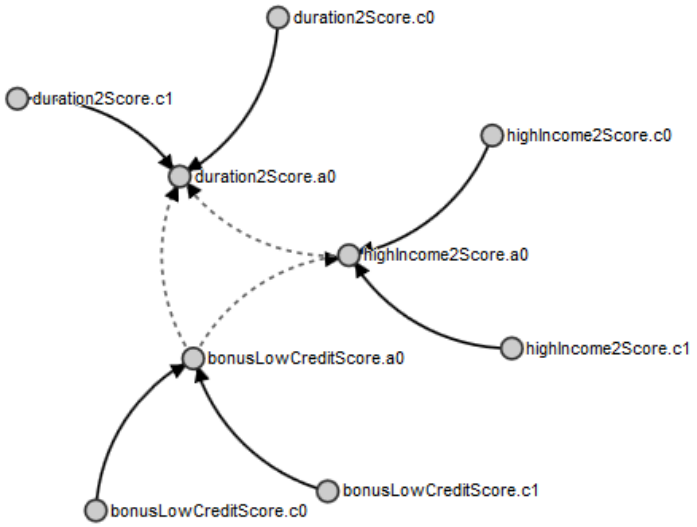


Figure 5.12: Generated minimal causal graph of the decision at rule elements level

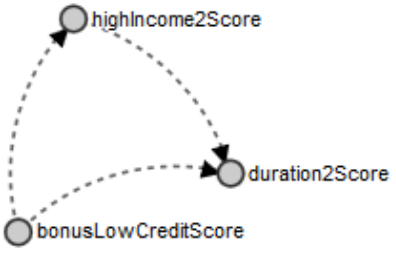


Figure 5.13: Generated minimal causal graph of the decision at rule level

wCreditScore.a0" and the action "duration2Score.a0" represents a computation relation between and the plain arrow between "highIncome2Score.c1" and "highIncome2Score.a0" models an execution relation.

5.2 Experiments and assessment protocol

In this section, we describe an experimentation protocol that can be used to evaluate the gain in reduction: (1) on the causal model of a rule-based system and (2) on the traces of its decisions. To this end, we provide a measure that we call gain in reduction which allows to estimate how many events are traced in the minimal

traces (in comparison to usual traces).

5.2.1 Reduction for the business rule-based system's causal model

Given a graph (of business rule elements or rules) $G = (N, A)$ where:

- N is a set of n nodes
- and A is a set of arrows defining ordered pairs of nodes.

and given $X = \langle x_1, ..., x_i, ..., x_m \rangle$, the set of explainable outputs.

We define $ancestor(G, X)$ as the graph of all the ancestors of any node $x_i \in X$ (that is, all nodes on a path leading to some explainable output). By $|G|$ we mean the number of nodes in a graph G. Our reduction measure simply consists in computing the ratio, in terms of number of nodes, between the two graphs.

$$Re(G) = \frac{|G| - |ancestors(G, X)|}{|G|} \tag{5.1}$$

and the participation rate:

$$\tau(G) = \frac{|ancestors(G, X)|}{|G|} \tag{5.2}$$

We note that the gain in reduction $Re(G)$ and the participation rate $\tau(G)$ are inversely proportional. Consequently, the interest in reducing the causal graph will increase for graphs with a low participation rate $\tau(G)$.

Example 5.9. *Computation of the causal model reduction for the example 5.1 with the explainable outputs:* $(X_1 = \langle creditScore, amount, yearlyInterestRate \rangle)$

The number of business rule elements is $|G| = 39$ *and based on the list of pertinent business rule elements obtained in the example 5.2, there are 22 nodes participating to the causal process involved in the computation of the explainable outputs so* $|ancestors(G, X_1)| = 22$. *We notice that the* $|ancestors(G, X_1)|$ *is simply given by the list of relevant events (i.e. pertinent events) associated to the corresponding*

reduced causal model of the decision service. Based on this, the estimated reduction is

$$Re(G) = \frac{|G| - |ancestors(G, X_1)|}{|G|} = \frac{39 - 22}{39} \approx 43,6\% \tag{5.3}$$

Example 5.10. *Computation of the causal model reduction for the example 5.1 with the explainable outputs:* $(X_2 = \langle approved \rangle)$

The number of business rule elements is $|G| = 39$ *and based on the list of pertinent business rule elements obtained in the example 5.2, there are 29 nodes participating to the causal process involved in the computation of the explainable outputs. We notice that the* $|ancestors(G, X_2)|$ *is simply given by the list of relevant events (i.e. pertinent events) associated to the corresponding reduced causal model of the decision service. Based on this, the estimated reduction is*

$$Re(G) = \frac{|G| - |ancestors(G, X_2)|}{|G|} = \frac{39 - 29}{39} \approx 25,6\% \tag{5.4}$$

5.2.2 Reduction for minimal traces of the decision

The idea behind this experimentation is to estimate the gain in reduction for minimal decision traces. In this perspective, some important aspects need to be considered. The first one is the distribution of the different requests (a request is characterized by a specific settings of the inputs). Such requests distribution provides a global view of the possible requests and raises awareness about how representative they actually are. Thus, considering the probability density associated to each request allows to reflect a more realistic view of the decisions taken by a rule-based system. By default each request has the same weight (ie. probability) but if a distribution is known (if enough data is available in a trace repository to determine accurately the distribution or an expert can provide the distribution based on its experience). The second one is the number of times that an event of a specific class occurs during a decision. Combined with the previous aspect, it can be used to estimate the probability that each class of events occurs during the decision taken by a rule-based system.

Thus, given $P = \langle \rho_1, ..., \rho_m \rangle$, the set of possible requests and $E = \langle e_1, ..., e_n \rangle$, the set of classes of events, a request ρ_j has:

- a probability $p(\rho_j)$ (that can be retrieved from the probability density obtained from the multivariate normal law with the parameters described above),
- a table of integer values $T = \langle N_{\rho_j}(e_1), ..., N_{\rho_j}(e_n) \rangle$ that counts the occurrences for each class of events in $E = \langle e_1, ..., e_n \rangle$. The value of the element $N_{\rho_j}(e_i)$ corresponds to the number of events belonging to the corresponding class e_i that occurred during a decision runs with the request ρ_j.
- the weight of an event e_i for the request ρ_j is noted:

$$\omega(\rho_j, e_i) = p(\rho_j) \times N_{\rho}(e_i) \tag{5.5}$$

- the weight after reduction of an event e_i for the request ρ_j is noted:

$$\omega'(\rho_j, e_i) = p(\rho_j) \times N_{\rho_j}(e_i) \times \lambda_i \tag{5.6}$$

where $\lambda_i = \begin{cases} 1, & \text{if } e_i \text{ is in the list of relevant classes of events} \\ 0, & \text{otherwise} \end{cases}$

Based on that, the estimated gain in reduction of the minimal trace is noted:

$$Re(G) = \frac{\sum_{\rho_j \in P} \sum_{e_i \in E} (\omega(\rho_j, e_i) - \omega'(\rho_j, e_i))}{\sum_{\rho_j \in P} \sum_{e_i \in E} \omega(\rho_j, e_i)} \tag{5.7}$$

In our examples, we try to simulate a realistic distribution of requests for a loan application. We generate the simulated data by using a multivariate Gaussian distribution based on the data provided in (IDF: typical borrower profile). The random variables of this distribution are: the credit score and the yearly income of a borrower and, the amount and duration of the requested loan.

Based on that, the main characteristics of the distribution are its means (300, 45744, 224397, 234), its standard deviations (30, 300, 10000, 60) and its covariance matrix where we consider the credit score and the yearly income of a borrower are

correlated and, the amount and duration of the requested loan are correlated:

$$\begin{pmatrix} 900 & 8000 & 0 & 0 \\ 8000 & 90000 & 0 & 0 \\ 0 & 0 & 100000000 & 50000 \\ 0 & 0 & 50000 & 3600 \end{pmatrix}$$

.

Example 5.11. *Estimation of the traces global reduction for the example 5.1 with the explainable outputs: (creditScore, amount, yearlyInterestRate)*

For the given distribution, we estimate the gain in reduction $Re(G)$ for the obtained traces based on simulations for the 8000000 inputs having the highest probability among a set of 61824000 inputs taken in the confidence interval:

$$Re(G) = \frac{\sum_{\rho_j \in P} \sum_{e_i \in E} (\omega(\rho_j, e_i) - \omega'(\rho_j, e_i))}{\sum_{\rho_j \in P} \sum_{e_i \in E} \omega(\rho_j, e_i)} \approx 1.0 - 0.31496454743397034$$
$$\approx 0,68503545256602966 \approx 68.5\%$$

Example 5.12. *Estimation of the traces global reduction for the example 5.1 with the explainable outputs: (approved)*

For the given distribution, we estimate the gain in reduction $Re(G)$ for the obtained traces based on simulations for the 4000000 inputs having the highest probability among a set of 61824000 inputs taken in the confidence interval:

$$Re(G) = \frac{\sum_{\rho_j \in P} \sum_{e_i \in E} (\omega(\rho_j, e_i) - \omega'(\rho_j, e_i))}{\sum_{\rho_j \in P} \sum_{e_i \in E} \omega(\rho_j, e_i)} \approx 1.0 - 0.4461176337589732$$
$$\approx 0,553823662410268 \approx 55.4\%$$

5.3 Conclusion

In this chapter we presented the implementation of a method for automatically: (1.1) generating the *minimal causal model of a rule-based system*, (1.2) minimizing a *decision trace at execution* and (1.3) generating the minimal causal model of a decision. We shall see in the next chapter that this causal information will serve to feed what we will call the *conceptual model*. It can be seen as an ontology describing

the objects or variables manipulated by a rule based-system and will help in the process of generating an explanation in our context.

We also proposed an experimental protocol in order to evaluate the benefits of the proposed approaches for the reduction of traces. Due to the "customized" aspect of a rule-based system, it is very difficult to give general figures about the obtained reduction. Nonetheless the presented experimentation can be applied to any rule-base system to check if there is any interest to use it. The potential reduction of the traces will be highly dependent on the causal model reduction and on the distribution of the users requests. Whereas the reduction of the causal model of the system can give a global idea of the possible reduction on the trace, the request distribution will impact the part of the causal model which is the most used. Consequently, a highly reduced causal model may lead to inefficient reduction on the traces if the requests distribution solicit the less reduced part of the causal model. Conversely, the opposite phenomenon can be observed on a marginally reduced causal model if the most reduced part of the causal model are the most solicited by the requests distribution. The main advantage of this protocol is to estimate for each rule-based system (provided with enough traces of the past decisions) if applying the proposed method could be useful. In the next chapter, we discuss how an explanatory model, that encompasses the causal material provided in this chapter and augmented with further knowledge to enable explanation capabilities, can be constructed.

Chapter 6

Towards an architecture of an explanation service for business rule-based systems: basics and insights

6.1 Introduction

This chapter is devoted to discuss and analyze what can be the concepts and elements to put forward for constructing an explanation for a decision in a business rule-based system. As it was discussed in the Chapter 3, the question of generating explanation has received a great interest in different domains. For our particular purpose, we are interested by taking benefits form causality between events to induce or construct a reasoning pattern based on rules in order to explain outputs. Under such a perspective, Halpern and Pearl (2005c) provided a formal definition of an explanation based on causality, named "generic explanation". More precisely, the explanation of an explanandum φ has the form $(\Psi\ ,\ \vec{X} = \vec{x}\ )$, where Ψ is an arbitrary formula in its causal language which consists of some causal information and $\vec{X} = \vec{x}$ is a conjunction of primitive events representing the cause of the explanandum φ. On the other hand, Besnard et al. (2010, 2014b) proposed formalisms and techniques to extend causal knowledge with domain knowledge for supporting an explicative model from which explanation links can be inferred. In these propositions, an explanation

link is based on causal and ontological links. The main idea is that an event can be simply explained by revealing its causes and associated causal links substantiated with the appropriate ontological links that can be either (is-a) links representing specialization/generalization between classes of objects or (ded-ont) links applying on literals and modeling some inherited characteristics between these objects. In other terms, the information contained in a causal model and an ontological model can be combined together with some background knowledge provided by the user to provide explanatory satisfying arguments. Thanks to such approaches a user can benefit from the advantages of a causal representation combined with those of a "taxonomy".

For our context, we believe these two approaches being promising as a first attempt to answer the question of constructing an explanation for the ODM rule-based system. Indeed, whereas a causal model of the reasoning process reveals how the events are chained together, it cannot express information about the events themselves, their underlying concepts, the potential links between theses concepts and their place in the domain model. Such aspect can be handled by augmenting the causal model with a kind of ontology fitting with our needs and goals. In the same way, it can allow to deduce some common properties between the business rules, the business object and their instances, thus applying properties that we found within the level of a rule-based system at the level of its decisions. More precisely, we propose to extend the definition of Halpern and Pearl (2005c) by adding an ontology with the aim of providing descriptive and relational information about the elements of the causal model. Thus, we propose to have the following definition.

Definition 6.1. (Generic Explanation (E))
A generic explanation of the explanandum φ is tuple $E = \{\Psi, \vec{X} = \vec{x}, \Omega\}$, where

- Ψ is an arbitrary formula in a given causal language which consists of causal information,
- $\vec{X} = \vec{x}$ is a conjunction of primitive events representing the cause of the explanandum φ.

- Ω is an ontology of the elements of Ψ and $\vec{X}$ which consist in some descriptive and relational information.

Moreover, as it was described in Chapter 3, we use four dimensions in order to describe an explanation: (1) timing / temporal context, (2) question types, (3) content type, and (4) context-sensitivity. We propose to complete the Definition 6.1 by taking into account the fourth criterion in order to adapt an explanation to a specific context. The idea is that the answer (explanation) can be customized according to a specific user and a specific question. We will call this specific explanation (see the Definition 6.2).

Definition 6.2. (Specific explanation (SE))
A specific explanation SE is an answer to a question Q asked by a user U about an explanandum φ. Thus, this explanation is tuple $SE = \{E, C\}$, where:

- $E = \{\Psi, \vec{X} = \vec{x}, \Omega\}$ is a generic explanation tuple (according to definition 6.1) and,
- $C = \{Q, U\}$ is a tuple giving information about the context.

In practice, it means that the information contained in E is exploited with regards to a given context C in order to produce the specific explanation.

In information systems, what we call generic and specific explanations can be supported by a structure that organizes and stores information about the system, the decision and the context in order to make them available for answering to different user requests provided that further treatments are appropriately applied (Gregor and Benbasat, 1999). The idea behind this is to be able to provide enough meaningful information about a decision result to enable the explanation system to answer to any question of any user with only few additional treatments to adapt automatically the shape and the content of the rendered explanations. Under such a perspective, we propose in what follows an architecture for an explanation service that will allow to extract and organize the information in such a way that replying to any question

of a user about the decision is possible. This architecture will rely on our proposition of causal model construction.

6.2 The basics towards a service architecture to support explanatory models

As it was discussed previously, in order to shape an explanation we need to obtain the causal model and conceptual knowledge about the corresponding rule based system. We also need the traces of its decisions to be available. Moreover, if further information about the user is available, the explanations can be improved to better fit with the user needs and expectations. This choice leads us to look for a systematic approach to extract knowledge related to the decision logic, the domain objects manipulated by the system and the executed decision themselves. Another point to consider is that, in ODM we have different rule based systems (see Chapter 2, section 2.3.2), thus, we would have for each rule based system computing decisions it own explanation and then a user may request explanations for decisions taken by any of these rule-based systems. Therefore, we propose, as it is illustrated in Figure 6.1, and described in what follows, to design an explanation service as the product of three sub-processes.

1. **Decision service knowledge acquisition time**. It consists in the acquisition of the decision service knowledge base. The construction of this knowledge base is a prerequisite for the second and the third processes and must be done each time the domain knowledge is updated. It contains:

    - A causal Model built by analyzing the decision service logic. It will allow to understand the causal relationships between the elements of the decision and the reasoning behind. This model is a graph that can be reduced to keep only the nodes and relations that have an impact on a set of explainable outputs,
    - A conceptual model extracted by analyzing the domain objects and functi-

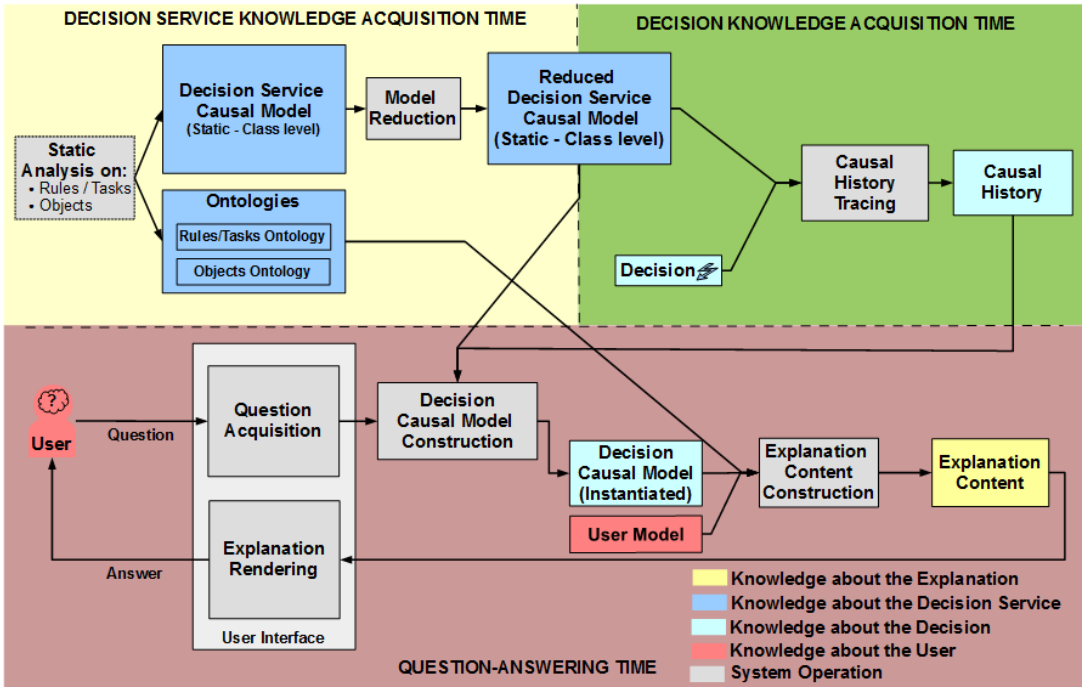


Figure 6.1: Explanation Service Architecture

ons of decision service. The conceptual knowledge (or ontology) which is useful to understand the meaning of each element presented in a causal model.

2. **Decision knowledge acquisition time**. It consists in tracing the minimal causal history based on a decision and on a list of relevant elements obtained from the causal model that has been built during the first step. What we call the minimal causal history is a reduced trace obtained by the process described in Chapter 4. The decision history acquisition is a prerequisite for the third process and must be done each time a decision is taken by the decision service.

3. **Question-answering time**. The question answering step is based on the material provided by the two previous steps and considers a query of the user about a specific element of a decision to generate explanation content that can be rendered in an explanation view. It is bases on the context knowledge which contains:

- a user model that provides information about the user helping the explanation service to provide an explanation that suits to the user needs,
- Some information specific to the query submitted by the user.

Based on the above we define the software architecture of the explanation service by the Component Diagram, depicted in Figure 6.2. Moreover, the Sequence Diagram depicted in Figure 6.3 gives an overview of the explanation service behavior. This diagram provides a better understanding of how the system uses each component exploit the explanatory models and render explanations.

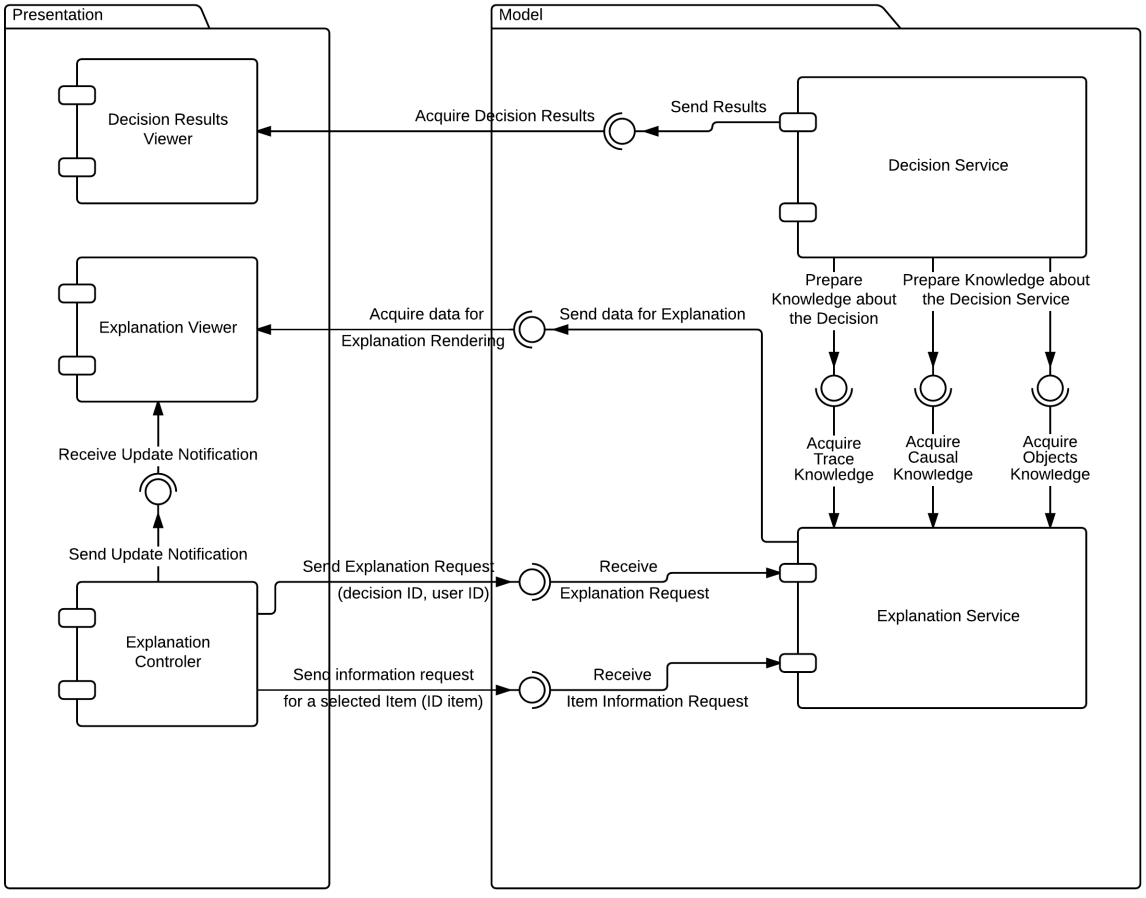


Figure 6.2: Explanation Service Architecture

In summary, the explanation generation for rule-based systems in our industrial context can be translated by using an explanation service embedding explanatory

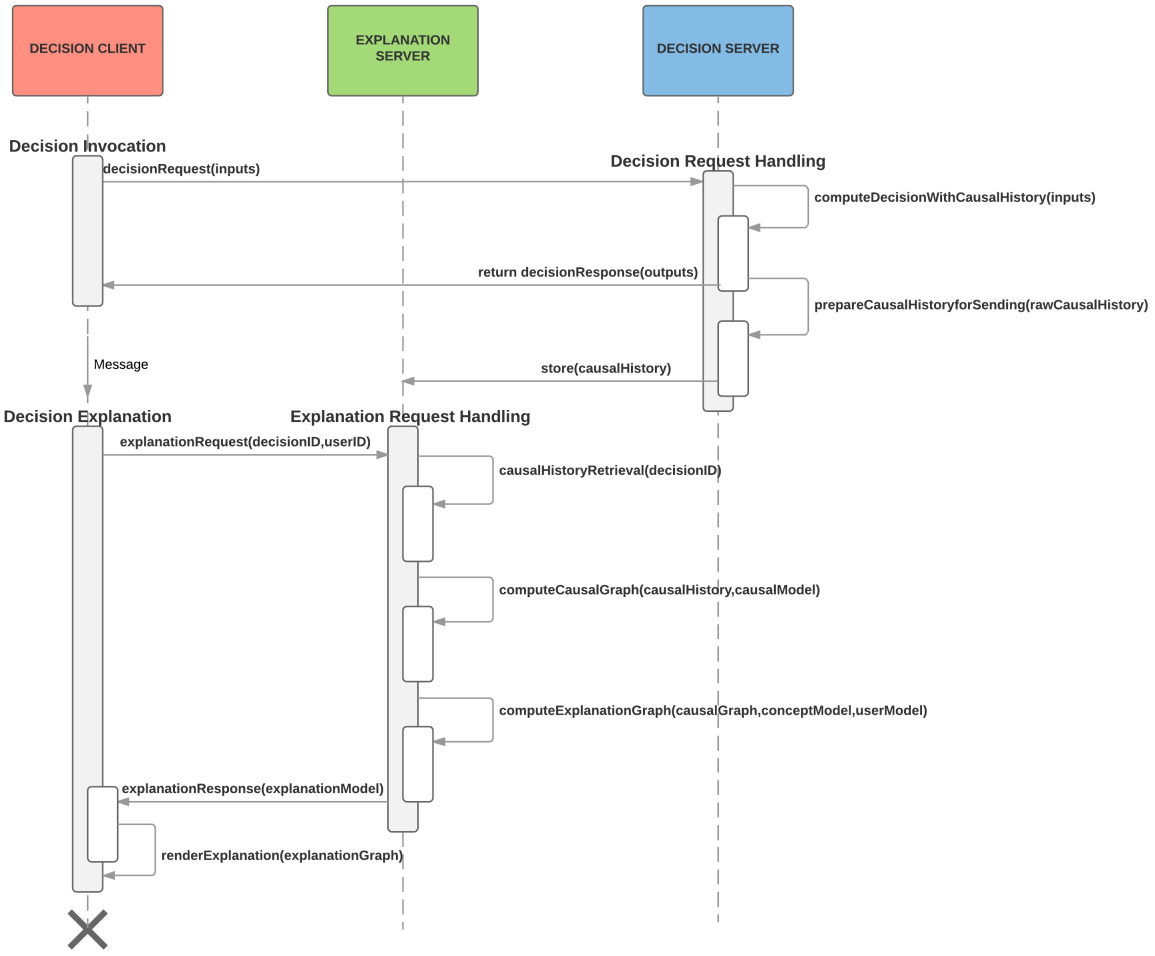


Figure 6.3: Explanation Service Architecture

models, such that each *Explanatory Model* is an articulation of:

- *A causal model*: it encompasses a minimal causal model of the decision service and the minimal causal models of the decisions that need to be explained. This model is responsible for organizing and making available any causal knowledge that could be required in order to describe how some events of a decision are linked.

- *A conceptual model*: it provides descriptive information about the elements of the domain/decision and also does a taxonomy of relationships among them. This model is required to describe the articulation (hierarchy/connections)

between elements of the domain, their meaning and their function.

- *A contextual model*: it is responsible for handling the context of an explanation. In our case, it is limited to a user model and the explanation query. This model allows to adapt the information contained in the causal and conceptual models to the request of a specific user. To this end, it takes into account the type, the expertise, the privacy and the spoken language of the user, and the explanation(/question) query.

6.3 The components of an explanatory model

As it as described just before, an explanatory model in our context is the combination of three distinct models: a causal one, a conceptual one and finally a contextual one. These models will be articulated generating generic (see definition 6.1) and specific (see definition 6.2) explanations. In what follows, we will describe in detail the conceptual and the contextual model. Indeed, we refer the reader to the Chapter 4 for the causal model.

6.3.1 Conceptual model as a part of the explanatory model

The purpose of this model is to provide further information about the objects or variables manipulated by a rule-based system. In this perspective, the conceptual model can be seen as an ontology describing the concepts manipulated in the domain. In other terms, it contains information about types, properties and relationships related to these concepts. It is an important part of the explanatory model as it allows to give a "conceptual-meaning" to the elements manipulated by the rule-based system with the aim to complete the information provided by the causal model. Consequently, the comprehension of the domain concepts with the comprehension of the causal relationships which link them, allows to obtain enough information about the rule-based system and to provide an efficient material for explanation.

Moreover, the conceptual model construction contains descriptive and relational

information about the elements of the related rule-based system (or decision service) and it is premised on a static program analysis of it. Here, the term “static” means the analysis of a software is performed without actually executing its programs. What is considered in this analysis depends on the kind of the decisions taken by the rule-based system and what is meant by the "kind of decisions" refers to the nature of the parameters used by the rule-based system during its decision processes. If the rule-based system use variables then its decisions will be seen as "variable-oriented", while if the same system use objects and attributes then its decision will be seen as "object-oriented". These two kinds of decision will not have the same informational value because an object-oriented approach allows to extract more information about the structure of the knowledge in the domain. For example, whereas a variable can be described by its type and eventually by the formula and some related variables from which it was derived, in the same way, an attribute or an object can be described by its type and eventually the other objects or attributes from which it was derived but it also can be described as a part of another object.

Therefore, we believe that at least three kinds of knowledge must exist in order to enable a comprehensive description of the objects existing in the domain: (1) knowledge about the nature of an element, (2) knowledge about a hierarchy between the elements of the domain, (3) knowledge about the origin of the value of an element of the domain. A fourth one can be considered to add a descriptive knowledge informing about the meaning or the function of an element. We describe below each of these types of knowledge.

1. **Type knowledge**. It provides information about the *type* of the element, in an object perspective it means that this kind of knowledge gives information about the type of either an object or an attribute or a variable. It is used to make a link between an instance and the corresponding object class / business rule. This information is available under a label "$IS-A$" attached to the concerned element.

2. **Hierarchy knowledge**. The second kind of knowledge provides information

about the structure of the information and specify a hierarchy between the elements of the domain. It is used to describe a hierarchy between business objects and business variables but also between tasks, business rules and business rule elements (business rule actions and conditions). This information is available under a label "$IS-PART-OF$" attached to the concerned element. In the same way it contains the information to which task a business rule belongs and to which business rule an action or a condition belongs.

3. **Value knowledge**. This knowledge describes the dependencies between the values of the elements in the domain. For an attribute or a variable, this knowledge informs about the attributes or variables from which it is derived and describes how each of them influences its value based on the formula responsible for computing its value. This information is available under a label "$IS-DERIVED-FROM$" attached to the concerned element.

4. **Functional knowledge**. This knowledge describes the function, or at least the meaning, of an element in the domain. For an attribute or an object, it informs about its meaning and for a task, a business rule, an action or a condition, it informs about it underlying function. This information is available under a label "$IS-DESIGNED-FOR$" attached to the concerned element.

Moreover, to these labels the two following complementary sub-labels can be added.

- **Language knowledge.** This sub-label is annotated with the two first letters of the referred language - for example, "$-FR$" for french and "$-EN$" for english - and allows to duplicate the knowledge of the corresponding label to handle several languages.

- **Expertise knowledge.** This sub-label is annotated with a letter which refers to the user type and a number which refers to its level of expertise. For example, $-K1$ refers to a knowledge engineer with no expertise about the domain and

$-B2$ refers to a business user with an intermediate level about the domain. If an information should not be available for a specific type of user then the sub-label will not be available for this information. For example, if "$IS-A-FR-B2$" is not available for an element, it means that an intermediate level business user does not need this information. Another example, for the same element, "$IS-A-FR-B1$" and "$IS-A-FR-B3$" can both exist but have different contents.

After defining the different labels, we propose a representation of the conceptual model. To illustrate our proposal we use the following example, which is represented in Figure 6.4.

Example 6.1. *[conceptual model for an attribute yearlyRepayment of the decision presented in the object-oriented usecase]*
In this example, for the attribute yearlyRepayment, we have the following labels: The label "$IS-A-EN$: the yearly repayment" and the label "$IS-A-FR$: remboursement annuel" contain a generic term used to describe the class in english and french, the label "$IS-DERIVED-FROM$: amount, yearlyInterestRate, duration " contains references to the amount, the yearly interest rate and the duration of the loan because they are in the formula used to compute the yearly interest rate value, the label "$IS-PART-OF-EN$: Joe's loan" and "$IS-PART-OF-FR$: prêt de Joe" give the reference to the instance of the object to which the yearlyRepayment attribute belongs, the term describing the class of this instance can then be found with the "$IS-A$" label of this instance.

Even if at first sight, this model seems to provide sufficient information to construct explanations, there is an issue in using an explanatory model embedding only causal and conceptual models. When the explanation is constructed, if it does not care about the user and, consequently, it is not capable to adapt the shape of the explanation or the vocabulary used in it to the recipient of the explanation. This observation leads to think that the effectiveness of the explanations generated by

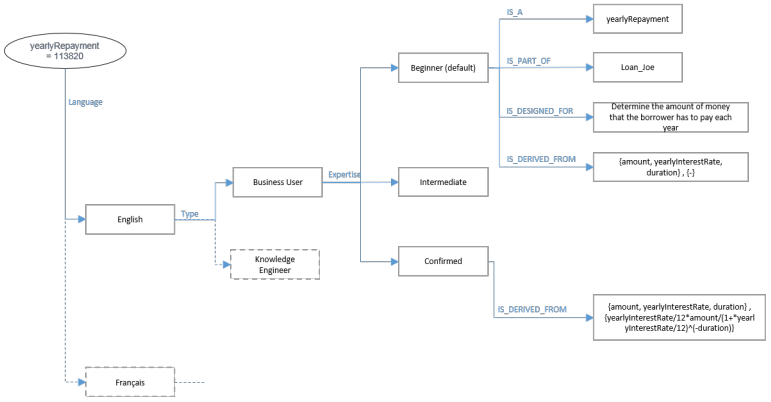


Figure 6.4: Example part of conceptual model for attribute yearlyRepayment

using this model could be improved by adapting them to the user profile. That is, what we try to do with our sub-labels but we need to obtain information about the user to be able to use them in a good way. To do that, we propose to design a user model allowing to capture the most important characteristics of the user in our perspective of explanation. This model can then be added to the causal and conceptual models to augment our explanatory model.

6.3.2 User model as a part of the explanatory model

Based on our observation, what should be considered in the user model are the knowledge about user type (business user, knowledge engineer...), the knowledge about his level of expertise (novice, intermediate, expert...), the privacy level of the user which contains the knowledge about what information can be accessed by the user and the user language which contains information about the language used by the user, described in what follows and illustrated in the Example 6.2

- **User Type**. The user type is a parameter indicating the type of the user. By default, we consider two types of user (business user, knowledge engineer) but any type of user can be defined depending on the needs of the explanation system.

- **Expertise Level**. The expertise level is a parameter indicating how the user

masters the domain. By default, we consider tree values of user (novice, intermediate, expert) but other values can be defined depending on the needs of the explanation system.

- **Privacy Level.** The privacy level is a parameter indicating what information the level is allowed to know. By default, we consider tree values of privacy (1, 2, 3). The value "1" means that the user can access the information that everyone can access, the level of privacy 2 means that the user can access to corporate information and the level of privacy 3 is restricted to a smaller group. On the same principle other values can be defined depending on the needs of the explanation system.

- **User language**. The user language is a parameter indicating what language should be used to deliver the explanation to the user. This parameter takes the form of a list of language where the order indicate the level of preference for the considered language. For example if the user language contains "French,English,Chinese" it means that the user speak these three language but prefer access the information in french and then English and then Chinese depending on their availability.

Example 6.2. *[User model]*
$UserID : Carlos$
$UserLanguage : \{French(FR), English(EN), Portuguese(PO)\}$,
$ExpertiseLevel : \{Novice\}$,
$PrivacyLevel : \{1\}$,
$UserType : \{BusinessUser\}$

6.4 Exploitation of the explanatory model: discussion and insights

As it was discussed in Chapter 3, different characteristics can be defined to design an explanation. In our context, we believe that, first, we must provide (1.1) *feedback explanations* (explanation orientation criteria) as we seek to explain what happened after a decision. After that the content of the explanation should correspond to (2.1) *trace-based*, (2.2) *justification*, (2.3) *strategic* and (2.4) *definition / terminological*. Furhtermore, the types of questions that our explanatory model can handle are: (3.1) How-Q, (3.2) Why-Q and (3.3) What-Q. In addtion, the proposed explanatory model is (4) *context-sensitive* (limited to the user), as the explanation service is not intended to provide dialog capabilities in its initial phase. And last but not least, since IBM is a software provider, its clients can have highly various needs and thus use its Business Rule Management System to develop and run wide range of applications. Consequently, the explanation features should be generic enough to limit the costs of the modification required for being usable on to each client application. Thus, the (5) *genericity* of the provided solution is a must have.

6.4.1 How we deal with each criterion

Concerning the explanation orientation, some (1.1) *feedback explanations* about a specific decision or about the rule-based system itself can be asked at anytime. Each explanation is constructed using the knowledge contained in the explanatory model and the minimal trace of the concerned decision that has been stored in a repository.

Concerning the (2.1) *trace-based content*, the explanation simply relates on the stored decision traces. The (2.2) *justification content* is obtained exploiting the knowledge contained in the causal model. The (2.3) *strategic content* can be obtained using the hierarchical knowledge of the conceptual model to explain the decision at three different levels (tasks / business rules / business rule elements). Using this knowledge allows to set up the granularity of the explanation. Finally, the (2.4)

definition content is obtained from the conceptual model and allows to provide meaningful descriptions about the elements (business objects, business variables, tasks, business rules, actions and conditions) manipulated during the decision process.

The different types of questions [(3.1) How-Q, (3.2) Why-Q and (3.3) What-Q] handled relies on the explanation strategies predefined. Each explanation strategy corresponds to a specific treatment that exploit the explanatory model to answer the related question. A user can customize its own questions and associated explanation strategies to enrich the list of possible questions.

In our case, the (4) *context-sensitivity* is limited to the user but can be extended by adding additional knowledge to the contextual model. We deal with the user-sensitivity by using the information contained in the user model (or user profile) to determine what information in the conceptual and causal models should be presented to the user.

We ensure a good genericity by using an explanatory model whose the feeding of the content essentially relies on a generic method. Moreover the explanatory models generated with this method are not stored in their corresponding rule-based systems but rather centralized in an explanation service that is dedicated to their exploitation.

6.4.2 A graphical representation for engineering the explanation

Along this chapter we proposed different models composing our explanatory model for our business rule based system. Now, to derive an explanation, we propose a graphical approach that exploits such different models. The idea is that the explanation corresponds to displaying a graph corresponding to the minimal causal model of the decision that needs to be explained. In addition to this "causal graph", the information contained in the conceptual model are also retrieved with the aim to augment the causal graph with it. The obtained graph will describe the causal relations involved in the decision process while informing about the concepts under use and can be navigated in a simple way.

Based on that, each element of the decision is presented with additional descriptive and relational information. Whereas the descriptive information obtained from type and functional knowledge (see Section 6.3.1) gives a better understanding about the the concept behind the presented element and its function, the relational information obtained from value and hierarchy knowledge allows to navigate through the graph depending on the level of detail that the user wants to observe. Finally the information that allows to understand how the decision is taken and what are the relevant reasoning steps that should be observed to get it is given by the causal relations.

Moreover, the information displayed by the graph is pre-selected by using the user model. Based on that, the information displayed about the elements of the graph will correspond to the language, level of expertise, the type and the privacy level of the user who requests the explanation. The following example illustrates our purpose.

Example 6.3. *Node presentation*

Let's consider a user Carlos having the user model presented below.

```
User:
{
 UserID : Carlos;
 UserLanguage : French(FR), English(EN), Portuguese(PO);
 ExpertiseLevel : Novice;
 PrivacyLevel : 1;
 UserType : BusinessUser
}
```

Example of user model

The privacy level is based on a table which associates a boolean to each element of the decision that could be referred in the explanation (task, rule, condition, action, object or attribute). If the boolean is true then the corresponding element can be referred but if it is false it cannot. In our example, the privacy level 1 refers to a table which does not allow the access to the conditions (c_0 and c_1) and the action (a_0) of the rule duration2Score but allows the access to all the other element of the decision.

```
Table 1:
{
 ...
 duration2Score.c0=false;
 duration2Score.c1=false;
 duration2Score.a0=false;
 duration2Score=true;
 duration=true;
 Loan=true;
 eligibility=true;
 creditScore=true;
 Borrower=true;
 ...
}
```

Privacy table for privacy level 1

The user Carlos displays the graph corresponding to Example 5.13 and looks at the node representing the instance of "duration2Score". The descriptive information of this rule instance corresponds to its functional knowledge and informs about the attributes evaluated and modified by the rule. In our example, the descriptive information about the rule duration2Score contains:

(1-en) *"evaluates ⟨ the duration of the loan ⟩ to modify (the credit score of the borrower)";*

(1-fr) *"évalue ⟨ la durée du prêt ⟩ pour modifier ⟨ le score de Joe ⟩ ";*

(2-en) *"evaluates if (c_0 desc. info and c_1 desc. info) to set (a_0 desc. info)";*

(2-en) *"évalue si (c_0 desc. info et c_1 desc. info) pour mettre (a_0 desc. info)"; ...*

As the privacy level of Carlos is 1, the information provided to him will be restricted to (1). Moreover, as the first language of Carlos is the French, the description (1-fr) "évalue la durée du prêt pour modifier le score de Joe" will be presented to him. In addition, as the rule instance "duration2Score" is a part of the task "eligibility", the system would be able to provide further information about the aim of this rule at a higher level by using the descriptive information of the task "eligibility": (en) "aiming to determine the eligibility of the loan", (fr) "dans le but de determiner l'éligibilité du prêt". Thus, the node will be presented with the information "la

règle duration2Score évalue la durée du prêt pour modifier le score de Joe dans le but de déterminer l'éligibilité du prêt" to Carlos. We assume that in our example we presented only the content corresponding to the novice (default) level of expertise.

Nonetheless, in addition to the proposed descriptive information, the user can add more specific descriptions. The idea here is to provide the explanation material with the most basic descriptions that the user can complete or modify depending on its needs. In the same way, new expertise and privacy levels can be designed to fit with the user needs.

This mechanism is supported by the structure presented in the figure 6.5. In it, the I-D-Fo links are used to associate the descriptions that we presented above, whereas the IS-A links allows to refer corresponding items or classes to find the adapted information and the I-P-O links establish a hierarchy between the elements presented in the descriptions.

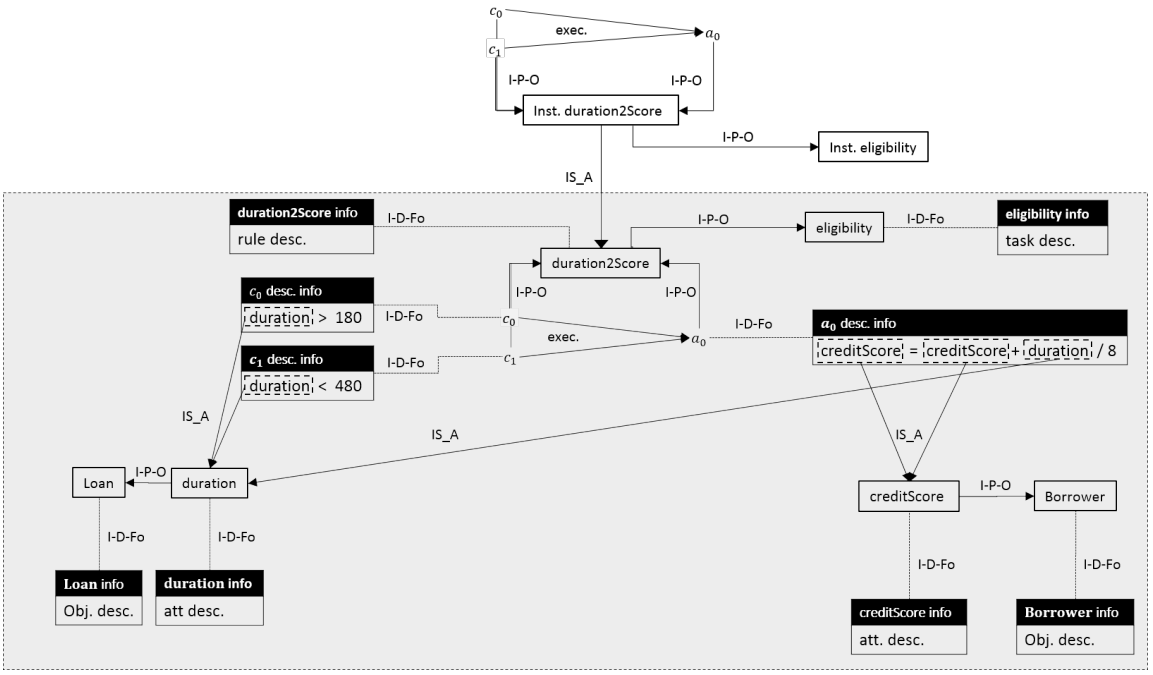


Figure 6.5: Example of a node information in the explanative model

Chapter 7

Conclusion

During the last decades, Business Rule Management Systems have been widely used by various organizations and companies to enhance the management of their automated decision-making systems. In this perspective, to gain their acceptance by the industry, they received strong incentives to provide, at least, basic explanation capabilities. The ultimate goal of such explanations was mainly to increase the transparency and the scrutability of automated decisions by revealing meaningful information about reasoning processes and knowledge base to the users, by educating them about the decision domain and the system capabilities, by facilitating the debugging and the monitoring of the system during the development stages.

Nonetheless, despite the benefits of explanation capabilities, since it is not straightforward to develop and maintain useful explanation features, many of the current industrial Business Rule Management Systems are still provided with poor explanation capabilities, often limited to basic debugging and tracing tools (plus eventually some monitoring tools).

More recently, governments show a growing interest for making it mandatory to provide automated decision making systems with the ability to justify their decisions. In this perspective, the European General Data Protection Regulation even mention a "right for explanation" (and non-discrimination) as discussed by Goodman and Flaxman (2016). Because of that, these systems receive even more incentives to provide better explanation capabilities. Indeed, there is a growing interest from the

scientific community on the topic of explanation generation in rule-based systems and it is mainly focused on three axis: (1) basic explanation content generation, (2) responsiveness and (3) human computer interface.

The work presented here is in line with the first axis, namely generating explanation for decisions of business rules-based systems. More precisely, we were interested by the IBM ODM system and discussed the opportunities to construct explanations for decisions issued from such system. In order to reach the explanation feature, we proposed in this work to take advantage from the fact that the reasoning process behind a rule-based system can be described by the causal links that may exists between the rules that effectively played a role in generating the decision. A first step toward this construction was to propose a framework (or a process) for mechanically building a set of causal models that can be exploited to represent a rule-based system and its decisions. These causal models encompass different notion of causality (between the rules and inside the same rule) with the aim to translate the logic followed by the system to construct a decision. We provide after that a method for minimizing them depending on the decision outputs that need to be explained. This minimization has the aim to keep only the necessary and minimal information needed to produce the decision. Moreover, a method using these models for minimizing the size of the histories obtained at execution by tracing only the pertinent events is also provided. An implementation of these methods was provided and described. This implementation was used to test the proposed framework and illustrate this method with meaningful examples. We also provided an assessment protocol and some measurement tools to evaluate the gain in reduction for a rule-based system and estimate the average gain in reduction for its decision traces.

Now, to answer the explanation question for the ODM system, we discussed at the end of this work the features that can be used to characterized an explanation in our context. More precisely, we discussed an architecture for an explanatory service consisting in a set of causal, conceptual and contextual models that could be used together to provide complementary information supporting the construction of

various kinds of explanations. This architecture relies on the analysis of a rule-based system and its decision traces and aims to extract sufficient information from the business rules, the business objects and the decision traces to construct the causal and conceptual models, the user profile can then be obtained from the user or from the rule-based system knowledge manager.

The research work presented through this manuscript opens up new directions of research. In particular, the conception of efficient algorithms dedicated to the exploitation of the explanatory model for generating high level explanations that answer to various user's questions is one of them. In fact the idea behind the proposed explanation service is to provide an explanation model and a method to generate generic answers to the most commons user questions. More specific and complex strategies to answer other questions could then be added by the industrial users depending on their domains, their habits and their needs.

Moreover, for our need, we limited the contextual model to the user model but this contextual model can be extended by adding some knowledge concerning the dialog between the user and the system (previous questions asked by the users) could be added to extend the model in order to handle an interaction context when providing answer to user questions. In that vein, as it was not in our scope we did not focus on language generation aspects and natural language processing. This issue could be considered to improve the communication between the system and the user. In the same way, as each organization as its own interface formalism, the research on human computer interface aspect was not in our focuses. Finally, some machine-learning aspects could be added in the causal ascription phase but also for completing the conceptual model with domain information that are not present in the Business Object Model of a decision service and for improving the use of the contextual model. Eventually, using collaborative filtering approaches could help to complete missing information in users models or to predict what information could suit the most to a user needs based on the data about others similar users.

Bibliography

Bibliography

Janice S Aikins. Representation of control knowledge in expert systems. In *AAAI*, volume 1, pages 121–123, 1980. 71

Isabelle Alvarez. Explaining the result of a decision tree to the end-user. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 411–415. IOS Press, 2004. 96

John R Anderson. Knowledge compilation: The general learning mechanism. *Machine learning: An artificial intelligence approach*, 2:289–310, 1986. 36

John R Anderson. *Cognitive psychology and its implications* . WH Freeman/Times Books/Henry Holt & Co, 1990. 61

Vicky Arnold, Nicole Clark, Philip A Collier, Stewart A Leech, and Steve G Sutton. The differential use and effect of knowledge-based system explanations in novice and expert judgment decisions. *Mis Quarterly*, pages 79–97, 2006. 63

D. Ausubel. Learning as constructing meaning. *New Directions in Educational Psychology*, pages 71–82, 1985. 61

Salem Benferhat et al. A comparative study of six formal models of causal ascription. In *Proceedings of the 2nd International Conference on Scalable Uncertainty Management*, pages 47–62. Springer, 2008. 81

Philippe Besnard, Marie-Odile Cordier, and Yves Moinard. Ontology-based inference for causal explanation. *arXiv preprint arXiv:1004.4801*, 2010. 147

Philippe Besnard, Marie-Odile Cordier, and Yves Moinard. Arguments using ontological and causal knowledge. In *Proceedings of the 8th International Symposium on Foundations of Information and Knowledge Systems*, pages 79–96. Springer, 2014a. 77

Philippe Besnard, Marie-Odile Cordier, and Yves Moinard. Arguments using ontological and causal knowledge. In *Foundations of Information and Knowledge Systems - 8th International Symposium, FoIKS 2014, Bordeaux, France, March 3-7, 2014. Proceedings*, pages 79–96, 2014b. 77, 147

J. S. Brown. The low road the middle road and the high road. In P. H. Winston and K. H. Prendergast, editors, *The AI Business*, pages 81–90. MIT Press, Cambridge, MA, 1984. 34

Bruce G Buchanan and Edward A Feigenbaum. The stanford heuristic programming project: Goals and activities. *AI Magazine*, 1(1):25, 2017. 78

Bruce G. Buchanan and Edward H. Shortliffe. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley Series in Artificial Fgence)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984. ISBN 0201101726. 24, 26, 71, 73, 74, 75, 78

John M. Carroll and Mary Beth Rosson. Paradox of the active user, 1987. 61

B. Chandrasekaran. Explanation and the theory of expert problem solving. Technical report, DTIC Document, 1990. 67, 68

B Chandrasekaran and Sanjay Mittal. Deep versus compiled knowledge approaches to diagnostic problem-solving. *International Journal of Man-Machine Studies*, 19 (5):425–436, 1983. 33, 75

B Chandrasekaran and Michael C Tanner. Uncertainty handling in expert systems: uniform vs. task-specific formalisms. In *Uncertainty in Artificial Intelligence*, pages 102–113. North Holland Amsterdam, 1986. 67, 68

Hana Chockler, Joseph Y Halpern, and Orna Kupferman. What causes a system to satisfy a specification? *ACM Transactions on Computational Logic (TOCL)*, 9 (3):20, 2008. 98

William J Clancey. The advantages of abstract control knowledge in expert system design. Technical report, DTIC Document, 1983a. 71, 73

William J Clancey. The epistemology of a rule-based expert system—a framework for explanation. *Artificial intelligence*, 20(3):215–251, 1983b. 66, 71, 77, 78

Keith Darlington. Aspects of intelligent systems explanation. *Universal Journal of Control and Automation*, 1(2):40–51, 2013. 24, 39

Stanford University. Computer Science Dept, W.J. Clancey, R. Letsinger, and Stanford University. Computer Science Dept. Heuristic Programming Project. *Neomycin: reconfiguring a rule-based expert system for application to teaching*. Report (Stanford University. Computer Science Department). Department of Computer Science, Stanford University, 1982. 71, 74, 78, 79

Jasbir Singh Dhaliwal. *An Experimental Investigation of the Use of Explanations Provided by Knowledge-based Systems*. PhD thesis, University of British Columbia, 1993. 63

C. JIMENEZ DOMINGUEZ. *Sur l'explication dans les systemes a base de regles : le systeme PROSE*. PhD thesis, UPMC, 1990. Thèse de doctorat dirigee par J. Pitrat Sciences appliquees Paris 6 1990. 71, 74

Didier Dubois and Henri Prade. Liens causaux et explications. Problemes de modelisation : une discussion preliminaire . In D. Kayser, P. Marquis, and A. Napoli, editors, *Actes des 3Eme Journees Nationales sur les Modèles de Raisonnement GDR I3 Information-Interaction-Intelligence, Paris, France*, pages 81–90. INRIA, novembre 2003. 81

Robert S Engelmore. Artificial intelligence and knowledge based systems: Orig-

ins, methods and opportunities for nde. In *Review of Progress in Quantitative Nondestructive Evaluation*, pages 1–20. Springer, 1987. 23

Bas C. Van Fraassen. The pragmatic theory of explanation. In Joseph C. Pitt, editor, *Theories of Explanation*. Oxford University Press, 1988. 70

M. Friedman. Explanation and scientific understanding. *The Journal of Philosophy*, pages 5–19, 1974. 61

Gerhard Friedrich and Markus Zanker. A taxonomy for generating explanations in recommender systems. *AI Magazine*, pages 90–98, 2011. 62

Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*, 2016. 58, 165

Shirley Gregor and Izak Benbasat. Explanations from intelligent systems: Theoretical foundations and implications for practice. *MIS quarterly*, pages 497–530, 1999. 61, 62, 149

The Object Management Group. Semantics of business vocabulary and business rules. Technical report, The Object Management Group, 2008. 25

N. Hall. Structural equations and causation. *Philosophical Studies*, 132(1):109–136, 2007. 83

J. Halpern and J. Pearl. Causes and Explanations: A Structural-Model Approach. Part I: Causes. In J. S. Breese and D. Koller, editors, *UAI-01*, pages 194–202, Seattle, 2001a. Morgan Kaufmann. 83

J. Halpern and J. Pearl. Causes and Explanations: A Structural-Model Approach. Part II: Explanations. In J. S. Breese and D. Koller, editors, *IJCAI-01*, pages 27–34, Seattle, 2001b. Morgan Kaufmann. 61

J. Halpern and J. Pearl. Causes and Explanations: A Structural-Model Approach, Part I: Causes. *British Journal of Philosophy of Science*, 56(4):843–887, 2005a. 81, 83, 84

Joseph Y. Halpern. Appropriate causal models and the stability of causation. *CoRR*, abs/1412.3518, 2014. 83

Joseph Y Halpern and Christopher Hitchcock. Actual causation and the art of modeling. *arXiv preprint arXiv:1106.2652*, 2011. 83

Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part i: Causes. *The British journal for the philosophy of science*, 56(4): 843–887, 2005b. 98, 99, 101, 103, 104

Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part ii: Explanations. *The British journal for the philosophy of science*, 56(4):889–911, 2005c. 77, 147, 148

Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335 – 346, 1990. ISSN 0167-2789. 24

Peter E Hart. Directions for ai in the eighties. *ACM SIGART Bulletin*, (79):11–16, 1982. 34

John Haugeland. *Artificial intelligence: The very idea*. MIT press, 1989. 24

Frederick Hayes-Roth. Rule-based systems. *Commun. ACM*, 28(9):921–932, September 1985a. ISSN 0001-0782. doi: 10.1145/4284.4286. 25

Frederick Hayes-Roth. Rule-based systems. *Commun. ACM*, 28:921–932, 1985b. 18, 25, 54

Carl Hempel. Two models of scientific explanation. 1965a. 70

Carl Gustav Hempel. Inductive-statistical explanation. In *Aspects of Scientific Explanation, and Other Essays in the Philosophy of Science*. Free Press, 1965b. 70

C. Hitchcock. What's Wrong with Neuron Diagrams? In J. K. Campbell, M. O'Rourke, and H. Silverstein, editors, *Causation and Explanation*, pages 69–92. MIT Press, Cambridge, MA, 2007. 83

Dirk S. Hovorka, Matt Germonprez, and Kai R. Larsen. Explanation in information systems. *Inf. Syst. J.*, 18(1):23–43, 2008. 11, 60, 70

Hilary Johnson and Peter Johnson. Explanation facilities and interactive systems. pages 159–166, 1993. 61

M. Korver and P. J. F. Lucas. Converting a rule-based expert system into a belief network. *Medical Informatics*, 18:219–241, 1993. 81

Bart Kosko. Fuzzy systems as universal approximators. *IEEE transactions on computers*, 43(11):1329–1333, 1994. 27

Robert K. Lindsay, Bruce G. Buchanan, Edward A. Feigenbaum, and Joshua Lederberg. Dendral: A case study of the first expert system for scientific hypothesis formation. *Artificial Intelligence*, 61(2):209 – 261, 1993. 26, 71, 73, 78, 80

Tania Lombrozo and Susan Carey. Functional explanation and the function of explanation. *Cognition*, 99(2):167 – 204, 2006. 70

Peter Lucas and Linda van der Gaag. *Principles of Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1991. ISBN 0-201-41640-9. 23, 26, 30, 91

H Mark and Richer William J Clancey. Guidon-watch: A graphic interface for viewing a knowledge-based system. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1985. 78, 79

Cynthia J Martincic. Que: an expert system explanation facility that answers" why not" types of questions. *Journal of Computing Sciences in Colleges*, 19(1): 336–348, 2003. 106

Alexandra Meliou, Wolfgang Gatterbauer, Joseph Y Halpern, Christoph Koch, Katherine F Moore, and Dan Suciu. Causality in databases. *IEEE Data Eng. Bull.*, 33(EPFL-ARTICLE-165841):59–67, 2010. 98

Donald Michie. High-road and low-road programs. *AI magazine*, 3(1):21, 1982. 34

Johanna D. Moore and William R. Swartout. Explanation in expert systems: A survey. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA, 1988a. 87

Johanna D. Moore and William Roy Swartout. Explanation in expert systems : a survey. Technical Report ISI-RR-88-228, University of Southern California (Marina del Rey, CA US), 1988b. 77

Robert Neches, William R Swartout, and Johanna D Moore. Explainable (and maintainable) expert systems. In *IJCAI*, volume 85, page 382. Citeseer, 1985. 66, 68, 71

Allen Newell and Herbert Simon. The logic theory machine–a complex information processing system. *IRE Transactions on information theory*, 2(3):61–79, 1956. 24

Allen Newell, John C Shaw, and Herbert A Simon. Report on a general problem solving program. In *IFIP congress*, volume 256, page 64. Pittsburgh, PA, 1959. 24

Wanda J Orlikowski and Jack J Baroudi. Studying information technology in organizations: Research approaches and assumptions. *Information systems research*, 2(1):1–28, 1991. 70

Effy Oz, Jane Fedorowicz, and Tim Stapleton. Improving quality, speed and confidence in decision-making. *Information and Management*, 24(2):71 – 82, 1993. 72

Cecile L Paris. Generation and explanation: Building an explanation facility for the explainable expert systems framework. In *Natural language generation in artificial intelligence and computational linguistics*, pages 49–82. Springer, 1991. 71, 74

Ramesh S Patil, Peter Szolovits, and William B Schwartz. Causal understanding of patient illness in medical diagnosis. In *IJCAI*, volume 81, pages 893–899, 1981. 80

John W. Payne, James R. Bettman, and Eric J. Johnson. *The adaptive decision maker*. Cambridge University Press, 1993. 61

Michael J Pazzani. Refining the knowledge base of a diagnostic expert system: An application of failure-driven learning. In *AAAI*, pages 1029–1035, 1986. 36

Nancy Pennington and Reid Hastie. Explanation-based decision making: Effects of memory structure on judgment. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14(3):521, 1988. 77

Emil Post. Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics*, 65(2):197–215, 1943. 15

Alberto Prieto, Beatriz Prieto, Eva Martinez Ortigosa, Eduardo Ros, Francisco Pelayo, Julio Ortega, and Ignacio Rojas. Neural networks: An overview of early research, current frameworks and new challenges. *Neurocomputing*, 214:242 – 268, 2016. ISSN 0925-2312. 24

Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, CORNELL AERONAUTICAL LAB INC BUFFALO NY, 1961. 24

Ronald G. Ross. *Principles of the Business Rule Approach*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003. ISBN 0201788934. 25, 30

Wesley C Salmon. *Four decades of scientific explanation*. University of Pittsburgh press, 2006. 70

Wesley C Salmon et al. Four decades of scientific explanation. *Scientific explanation*, 13:3–219, 1989. 70

Edward H Shortliffe, Randall Davis, Stanton G Axline, Bruce G Buchanan, C Cordell Green, and Stanley N Cohen. Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the mycin system. *Computers and biomedical research*, 8(4):303–320, 1975. 66

Paul Smolensky. On the proper treatment of connectionism. *Behavioral and brain sciences*, 11(1):1–23, 1988. 24

Leon Sterling and Marucha Lalee. An explanation shell for expert systems. *Computational Intelligence*, 2(1):136–141, 1986. 65

Ron Sun. Artificial intelligence: Connectionist and symbolic approaches, 1999. 24

W. R. Swartout. A digitalis therapy advisor with explanations. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1977. 24, 71, 73, 74

William R. Swartout. Explaining and justifying expert consulting programs. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 815–823, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. 71

William R. Swartout. Xplain: A system for creating and explaining expert consulting programs. *Artificial Intelligence*, 21:285–325, 1983. 71, 74, 75, 80

William R Swartout and Stephen W Smoliar. Explaining the link between causal reasoning and expert behavior. In *Selected Topics in Medical Artificial Intelligence*, pages 71–84. Springer, 1988. 78

WR Swartout and SW Smoliar. On making expert systems more like experts. In *AI tools and techniques*, pages 197–216. Ablex Publishing Corp., 1989. 67, 68, 71, 74

Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *Proceedings of the 2007 IEEE 23rd International Conference on Data*

Engineering Workshop, ICDEW '07, pages 801–810, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 978-1-4244-0831-3. 11, 58, 59

S. Toulmin. *The Uses of Argument*. Cambridge University Press, Cambridge, UK, 1958. 61

Alan Turing. Intelligent machinery (1948). *B. Jack Copeland*, page 395, 2004. 15

Alan M Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950. 15

Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. In *Utility, probability, and human decision making*, pages 141–162. Springer, 1975. 77

William van Melle, Edward H Shortliffe, and Bruce G Buchanan. Emycin: A knowledge engineer's tool for constructing rule-based expert systems. *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project*, pages 302–313, 1984. 26, 71

Barbara von Halle. *Business Rules Applied: Building Better Systems Using the Business Rules Approach*. Wiley Publishing, 1st edition, 2001. ISBN 0471412937, 9780471412939. 25

Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, 7(2):76–99, 2017. 58

Jerold W Wallis and Edward H Shortliffe. Method for generating explanation. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, pages 338–362, 1977. 64, 80

Jerold W Wallis and Edward H Shortliffe. Explanatory power for medical expert systems: studies in the representation of causal relationships for clinical consultations. Technical report, STANFORD UNIV CA, 1981. 79

Jerold W Wallis and Edward H Shortliffe. Customized explanations using causal knowledge. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, pages 371–388, 1984. 78, 79

J.L. Weiner. Blah, a system which explains its reasoning. *Artificial Intelligence*, 15: 19 – 48, 1980. 71

Sholom M. Weiss, Casimir A. Kulikowski, Saul Amarel, and Aran Safir. A model-based method for computer-aided medical decision-making. *Artificial Intelligence*, 11(1):145 – 172, 1978. ISSN 0004-3702. Applications to the Sciences and Medicine. 80

Michael R. Wick and William B. Thompson. Reconstructive expert system explanation. *Artif. Intell.*, 54(1-2):33–70, March 1992. ISSN 0004-3702. doi: 10.1016/0004-3702(92)90087-E. 65, 71, 74

James Woodward. Scientific explanation. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 2014 edition, 2014. 60

Annexes

Appendix A

Example of business rules normalization

Example A.1. *Normalize the business rule language contained in business rule files*

Given a decision project (RuleSet Id: Miniloan Service, Number of rules: 14), the list below represent the business rules as they can be read and edited in the user interface.

```
rule 0:  (..\Miniloan Service\rules\eligibility\bonusLowCreditScore.brl)
if the credit score of 'the borrower' is at most 400
and the amount of 'the loan' is less than the yearly income of 'the
    borrower'
then set the credit score of 'the borrower' to the credit score of 'the
    borrower' + 100 ;

rule 1: (..\Miniloan Service\rules\eligibility\debt2IncomeRatio.brl)
if
  the yearly repayment of 'the loan' is more than the yearly income of 'the
     borrower' * (0.3 + the credit score of 'the borrower'  / 10000)
  or 10200 is more than the yearly income of 'the borrower' * the yearly
    repayment of 'the loan'
then
  add "Too big Debt-To-Income ratio" to the messages of 'the loan' ;
  reject 'the loan' ;

rule 2: (..\Miniloan Service\rules\eligibility\duration2Score.brl)
if
  the duration of 'the loan' is at least 180
  and the duration of 'the loan' is at most 480
then
  set the credit score of 'the borrower' to the credit score of 'the
    borrower' + the duration of 'the loan' / 8 ;

rule 3: (..\Miniloan Service\rules\eligibility\highIncome2Score.brl)
if
```

```
  the yearly income of 'the borrower' is at least 80000
  and the yearly income of 'the borrower' is less than 200000
then
  set the credit score of 'the borrower' to the credit score of 'the
    borrower' + 50 ;

rule 4: (..\Miniloan Service\rules\eligibility\lowAmountPenality.brl)
if the amount of 'the loan' is less than 10000
then set the credit score of 'the borrower' to the credit score of 'the
    borrower' - 100 ;

rule 5: (..\Miniloan Service\rules\eligibility\lowIncome2Score.brl)
if
  the yearly income of 'the borrower' is more than 10000
  and the yearly income of 'the borrower' is less than 30000
then
  set the credit score of 'the borrower' to the credit score of 'the
    borrower' - 50 ;

rule 6: (..\Miniloan Service\rules\eligibility\messageElig.brl)
if
  the yearly interest rate of 'the loan' is more than 0.015
then
  add "ok rate" to the messages of 'the loan' ;

rule 7: (..\Miniloan Service\rules\eligibility\score2Rate.brl)
if
  the amount of 'the loan' is more than 100000
  and the duration of 'the loan' is more than 120
  and the credit score of 'the borrower' is more than 500
then
  change the yearly interest rate of 'the loan' to 10 / the credit score of
     'the borrower' ;

rule 8: (..\Miniloan Service\rules\eligibility\veryHighIncome2Score.brl)
if
  the yearly income of 'the borrower' is more than 200000
then
  set the credit score of 'the borrower' to the credit score of 'the
    borrower' + 400 ;

rule 9: (..\Miniloan Service\rules\eligibility\veryLowIncome2Score.brl)
if
  the yearly income of 'the borrower' is less than 10000
then
  set the credit score of 'the borrower' to the credit score of 'the
    borrower' - 100 ;

rule 10: (..\Miniloan Service\rules\validation\maximumAmount.brl)
if
 the amount of 'the loan' is more than 1.000.000
then
  add "The loan cannot exceed 1,000,000" to the messages of 'the loan' ;
  reject 'the loan' ;

rule 11: (..\Miniloan Service\rules\validation\minimumAmount.brl)
```

```
if
  the amount of 'the loan' is less than 1000
then
  add "The loan cannot be lower than 1,000" to the messages of 'the loan' ;
  reject 'the loan' ;

rule 12: (..\Miniloan Service\rules\validation\minimumCreditScore.brl)
if
    the credit score of 'the borrower' is less than 200
then
    add "Credit score below 200" to the messages of 'the loan' ;
    reject 'the loan' ;

rule 13: (..\Miniloan Service\rules\validation\minimumIncome.brl)
if
  the yearly income of 'the borrower' is less than 18000
then
  reject 'the loan' ;
```

Business Rules in their Business Rule Language

The list below describes the business rules of the example after normalization.

```
rule 0 (Id: bonusLowCreditScore)
condition 0: [belongs to rule: bonusLowCreditScore]
isLessThan( miniloan.Borrower/creditScore/GETTER#0 , 400 )
condition 1: [belongs to rule: bonusLowCreditScore]
isLessThan( miniloan.Loan/amount/GETTER#0 ,
    miniloan.Borrower/yearlyIncome/GETTER#0)
action 0: [belongs to rule: bonusLowCreditScore]
creditScore.SETTER( add( miniloan.Borrower/creditScore/GETTER#0 , 100) )

rule 1 (Id: debt2IncomeRatio)
condition 0: [belongs to rule: debt2IncomeRatio]
isGreaterThan( miniloan.Loan/yearlyRepayment/GETTER#0 , mult(
    miniloan.Borrower/yearlyIncome/GETTER#0 , add( 0.3 , div(
    miniloan.Borrower/creditScore/GETTER#0 , 10000 ) ) ) )
condition 1: [belongs to rule: debt2IncomeRatio]
isGreaterThan( 10200 , mult( miniloan.Borrower/yearlyIncome/GETTER#0 ,
    miniloan.Loan/yearlyRepayment/GETTER#0 ) )
action 0: [belongs to rule: debt2IncomeRatio]
addToMessage( Too big Debt-To-Income ratio )
action 1: [belongs to rule: debt2IncomeRatio]
approved.SETTER( false )

rule 2 (Id: duration2Score)
condition 0: [belongs to rule: duration2Score]
isGreaterThan( miniloan.Loan/duration/GETTER#0 , 180 )
condition 1: [belongs to rule: duration2Score]
isLessThan( miniloan.Loan/duration/GETTER#0 , 480 )
action 0: [belongs to rule: duration2Score]
creditScore.SETTER( add( miniloan.Borrower/creditScore/GETTER#0 , div(
    miniloan.Loan/duration/GETTER#0 , 8 ) ) )

rule 3 (Id: highIncome2Score)
condition 0: [belongs to rule: highIncome2Score]
```

```
isGreaterThan( miniloan.Borrower/yearlyIncome/GETTER#0 , 80000 )
condition 1: [belongs to rule: highIncome2Score]
isLessThan( miniloan.Borrower/yearlyIncome/GETTER#0 , 200000 )
action 0: [belongs to rule: highIncome2Score]
creditScore.SETTER( add( miniloan.Borrower/creditScore/GETTER#0 , 50 ) )

rule 4 (Id: lowAmountPenality)
condition 0: [belongs to rule: lowAmountPenality]
isLessThan( miniloan.Loan/amount/GETTER#0 , 10000 )
action 0: [belongs to rule: lowAmountPenality]
creditScore.SETTER( minus( miniloan.Borrower/creditScore/GETTER#0 , 100 ) )

rule 5 (Id: lowIncome2Score)
condition 0: [belongs to rule: lowIncome2Score]
isGreaterThan( miniloan.Borrower/yearlyIncome/GETTER#0 , 10000 )
condition 1: [belongs to rule: lowIncome2Score]
isLessThan( miniloan.Borrower/yearlyIncome/GETTER#0 , 30000 )
action 0: [belongs to rule: lowIncome2Score]
creditScore.SETTER( minus( miniloan.Borrower/creditScore/GETTER#0 , 50 ) )

rule 6 (Id: messageElig)
condition 0: [belongs to rule: messageElig]
isGreaterThan( miniloan.Loan/yearlyInterestRate/GETTER#0 , 0.015 )
action 0: [belongs to rule: messageElig]
addToMessage(ok rate )

rule 7 (Id: score2Rate)
condition 0: [belongs to rule: score2Rate]
isGreaterThan( miniloan.Loan/amount/GETTER#0 , 100000 )
condition 1: [belongs to rule: score2Rate]
isGreaterThan( miniloan.Loan/duration/GETTER#0 , 120 )
condition 2: [belongs to rule: score2Rate]
isGreaterThan( miniloan.Borrower/creditScore/GETTER#0 , 500 )
action 0: [belongs to rule: score2Rate]
yearlyInterestRate.SETTER( div( 10 , miniloan.Borrower/creditScore/GETTER#0
    ) )

rule 8 (Id: veryHighIncome2Score)
condition 0: [belongs to rule: veryHighIncome2Score]
isGreaterThan( miniloan.Borrower/yearlyIncome/GETTER#0 , 200000 )
action 0: [belongs to rule: veryHighIncome2Score]
creditScore.SETTER( add( miniloan.Borrower/creditScore/GETTER#0 , 400 ) )

rule 9 (Id: veryLowIncome2Score)
condition 0: [belongs to rule: veryLowIncome2Score]
isLessThan( miniloan.Borrower/yearlyIncome/GETTER#0 , 10000 )
action 0: [belongs to rule: veryLowIncome2Score]
creditScore.SETTER( minus( miniloan.Borrower/creditScore/GETTER#0 , 100 ) )

rule 10 (Id: maximumAmount)
condition 0: [belongs to rule: maximumAmount]
isGreaterThan( miniloan.Loan/amount/GETTER#0 , 1000000 )
action 0: [belongs to rule: maximumAmount]
addToMessage( The loan cannot exceed 1,000,000 )
action 1: [belongs to rule: maximumAmount]
approved.SETTER( false )
```

```
rule 11 (Id: minimumAmount)
condition 0: [belongs to rule: minimumAmount]
isLessThan( miniloan.Loan/amount/GETTER#0 , 1000 )
action 0: [belongs to rule: minimumAmount]
addToMessage( The loan cannot be lower than 1,000 )
action 1: [belongs to rule: minimumAmount]
approved.SETTER( false )

rule 12 (Id: minimumCreditScore)
condition 0: [belongs to rule: minimumCreditScore]
isLessThan( miniloan.Borrower/creditScore/GETTER#0 , 200 )
action 0: [belongs to rule: minimumCreditScore]
addToMessage( Credit score below 200 )
action 1: [belongs to rule: minimumCreditScore]
approved.SETTER( false )

rule 13 (Id: minimumIncome)
condition 0: [belongs to rule: minimumIncome]
isLessThan( miniloan.Borrower/yearlyIncome/GETTER#0 , 18000 )
action 0: [belongs to rule: minimumIncome]
approved.SETTER( false )
```

Business Rules in their Normalized Form

Appendix B

Causal model of minimal decision trace

Example B.1. *Causal model of the minimal decision trace given in the example 5.3 for the explainable outputs: yearlyInterestRate, creditScore, amount*


```
causal link:{"relation": {"type": "Computation","originClass": {"dataCauses
    ": ["miniloan.Borrower/creditScore/GETTER#0"]},"destinationClass": {"
    dataCauses": ["miniloan.Borrower/creditScore/GETTER#0"]},"
    originElementClass": {"name": "bonusLowCreditScore.a0"},"
    destinationElementClass": {"name": "highIncome2Score.a0"}},"origin": {"
    variablesMap": {"miniloan.Borrower/creditScore/GETTER#0": {"name": "
    miniloan.Borrower/creditScore/GETTER#0","valType": "Float","type": "
    Default","domain": "[100,1000]","min": 100.0,"max": 1000.0,"
    defaultValue": 499}}},"destination": {"variablesMap": {"miniloan.
    Borrower/creditScore/GETTER#0": {"name": "miniloan.Borrower/creditScore
    /GETTER#0","valType": "Float","type": "Default","domain": "[100,1000]",
    "min": 100.0,"max": 1000.0,"defaultValue": 549}}},"originElement": {"
    className": "bonusLowCreditScore.a0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@13785ed"},"destinationElement": {"
    className": "highIncome2Score.a0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@5dd720f5"}}
causal link: {"relation": {"type": "Computation","originClass": {"
    dataCauses": ["miniloan.Borrower/creditScore/GETTER#0"]},"
    destinationClass": {"dataCauses": ["miniloan.Borrower/creditScore/
    GETTER#0"]},"originElementClass": {"name": "bonusLowCreditScore.a0"},"
    destinationElementClass": {"name": "duration2Score.a0"}},"origin": {"
    variablesMap": {"miniloan.Borrower/creditScore/GETTER#0": {"name": "
    miniloan.Borrower/creditScore/GETTER#0","valType": "Float","type": "
    Default","domain": "[100,1000]","min": 100.0,"max": 1000.0,"
    defaultValue": 499}}},"destination": {"variablesMap": {"miniloan.
    Borrower/creditScore/GETTER#0": {"name": "miniloan.Borrower/creditScore
    /GETTER#0","valType": "Float","type": "Default","domain": "[100,1000]",
    "min": 100.0,"max": 1000.0,"defaultValue": 579}}},"originElement": {"
    className": "bonusLowCreditScore.a0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@13785ed"},"destinationElement": {"
    className": "duration2Score.a0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@8cfed081"}}
```

```
causal link: {"relation": {"type": "Execution","originClass": {"dataCauses"
    : ["miniloan.Borrower/creditScore/GETTER#0"]},"destinationClass": {"
    dataCauses": ["miniloan.Borrower/creditScore/GETTER#0"]},"
    originElementClass": {"name": "bonusLowCreditScore.c0"},"
    destinationElementClass": {"name": "bonusLowCreditScore.a0"}},"origin":
     {"variablesMap": {"miniloan.Borrower/creditScore/GETTER#0": {"name": "
    miniloan.Borrower/creditScore/GETTER#0","valType": "Float","type": "
    Default","domain": "[100,1000]","min": 100.0,"max": 1000.0,"
    defaultValue": 399.0}}},"destination": {"variablesMap": {"miniloan.
    Borrower/creditScore/GETTER#0": {"name": "miniloan.Borrower/creditScore
    /GETTER#0","valType": "Float","type": "Default","domain": "[100,1000]",
    "min": 100.0,"max": 1000.0,"defaultValue": 499}}},"originElement": {"
    className": "bonusLowCreditScore.c0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@13785ed"},"destinationElement": {"
    className": "bonusLowCreditScore.a0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@13785ed"}}
causal link: {"relation": {"type": "Execution","originClass": {"dataCauses"
    : ["miniloan.Loan/amount/GETTER#0","miniloan.Borrower/yearlyIncome/
    GETTER#0"]},"destinationClass": {"dataCauses": ["miniloan.Borrower/
    creditScore/GETTER#0"]},"originElementClass": {"name": "
    bonusLowCreditScore.c1"},"destinationElementClass": {"name": "
    bonusLowCreditScore.a0"}},"origin": {"variablesMap": {"miniloan.Loan/
    amount/GETTER#0": {"name": "miniloan.Loan/amount/GETTER#0","valType": "
    Float","type": "Default","domain": "[1000,100000]","min": 1000.0,"max":
     500000.0,"defaultValue": 110000.0},"miniloan.Borrower/yearlyIncome/
    GETTER#0": {"name": "miniloan.Borrower/yearlyIncome/GETTER#0","valType"
    : "Float","type": "Default","domain": "[10000,50000]","min": 10000.0,"
    max": 200000.0,"defaultValue": 180000.0}}},"destination": {"
    variablesMap": {"miniloan.Borrower/creditScore/GETTER#0": {"name": "
    miniloan.Borrower/creditScore/GETTER#0","valType": "Float","type": "
    Default","domain": "[100,1000]","min": 100.0,"max": 1000.0,"
    defaultValue": 499}}},"originElement": {"className": "
    bonusLowCreditScore.c1","id": "com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@13785ed"},"destinationElement": {"className": "
    bonusLowCreditScore.a0","id": "com.ibm.rules.engine.rete.runtime.util.
    RuleInstanceImpl@13785ed"}}
causal link: {"relation": {"type": "Computation","originClass": {"
    dataCauses": ["miniloan.Borrower/creditScore/GETTER#0"]},"
    destinationClass": {"dataCauses": ["miniloan.Borrower/creditScore/
    GETTER#0"]},"originElementClass": {"name": "highIncome2Score.a0"},"
    destinationElementClass": {"name": "duration2Score.a0"}},"origin": {"
    variablesMap": {"miniloan.Borrower/creditScore/GETTER#0": {"name": "
    miniloan.Borrower/creditScore/GETTER#0","valType": "Float","type": "
    Default","domain": "[100,1000]","min": 100.0,"max": 1000.0,"
    defaultValue": 549}}},"destination": {"variablesMap": {"miniloan.
    Borrower/creditScore/GETTER#0": {"name": "miniloan.Borrower/creditScore
    /GETTER#0","valType": "Float","type": "Default","domain": "[100,1000]",
    "min": 100.0,"max": 1000.0,"defaultValue": 579}}},"originElement": {"
    className": "highIncome2Score.a0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@5dd720f5"},"destinationElement": {"
    className": "duration2Score.a0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@8cfed081"}}
causal link: {"relation": {"type": "Execution","originClass": {"dataCauses"
    : ["miniloan.Borrower/yearlyIncome/GETTER#0"]},"destinationClass": {"
    dataCauses": ["miniloan.Borrower/creditScore/GETTER#0"]},"
    originElementClass": {"name": "highIncome2Score.c0"},"
```

```
    destinationElementClass": {"name": "highIncome2Score.a0"}},"origin": {"
    variablesMap": {"miniloan.Borrower/yearlyIncome/GETTER#0": {"name": "
    miniloan.Borrower/yearlyIncome/GETTER#0","valType": "Float","type": "
    Default","domain": "[10000,50000]","min": 10000.0,"max": 200000.0,"
    defaultValue": 180000.0}}},"destination": {"variablesMap": {"miniloan.
    Borrower/creditScore/GETTER#0": {"name": "miniloan.Borrower/creditScore
    /GETTER#0","valType": "Float","type": "Default","domain": "[100,1000]",
    "min": 100.0,"max": 1000.0,"defaultValue": 549}}},"originElement": {"
    className": "highIncome2Score.c0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@5dd720f5"},"destinationElement": {"
    className": "highIncome2Score.a0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@5dd720f5"}}
causal link: {"relation": {"type": "Execution","originClass": {"dataCauses"
    : ["miniloan.Borrower/yearlyIncome/GETTER#0"]},"destinationClass": {"
    dataCauses": ["miniloan.Borrower/creditScore/GETTER#0"]},"
    originElementClass": {"name": "highIncome2Score.c1"},"
    destinationElementClass": {"name": "highIncome2Score.a0"}},"origin": {"
    variablesMap": {"miniloan.Borrower/yearlyIncome/GETTER#0": {"name": "
    miniloan.Borrower/yearlyIncome/GETTER#0","valType": "Float","type": "
    Default","domain": "[10000,50000]","min": 10000.0,"max": 200000.0,"
    defaultValue": 180000.0}}},"destination": {"variablesMap": {"miniloan.
    Borrower/creditScore/GETTER#0": {"name": "miniloan.Borrower/creditScore
    /GETTER#0","valType": "Float","type": "Default","domain": "[100,1000]",
    "min": 100.0,"max": 1000.0,"defaultValue": 549}}},"originElement": {"
    className": "highIncome2Score.c1","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@5dd720f5"},"destinationElement": {"
    className": "highIncome2Score.a0","id": "com.ibm.rules.engine.rete.
    runtime.util.RuleInstanceImpl@5dd720f5"}}
causal link: {"relation": {"type": "Execution","originClass": {"dataCauses"
    : ["miniloan.Loan/duration/GETTER#0"]},"destinationClass": {"dataCauses
    ": ["miniloan.Borrower/creditScore/GETTER#0"]},"originElementClass": {"
    name": "duration2Score.c0"},"destinationElementClass": {"name": "
    duration2Score.a0"}},"origin": {"variablesMap": {"miniloan.Loan/
    duration/GETTER#0": {"name": "miniloan.Loan/duration/GETTER#0","valType
    ": "Integer","type": "Default","domain": "[12,240]","min": 12,"max":
    500,"defaultValue": 240.0}}},"destination": {"variablesMap": {"miniloan
    .Borrower/creditScore/GETTER#0": {"name": "miniloan.Borrower/
    creditScore/GETTER#0","valType": "Float","type": "Default","domain": "
    [100,1000]","min": 100.0,"max": 1000.0,"defaultValue": 579}}},"
    originElement": {"className": "duration2Score.c0","id": "com.ibm.rules.
    engine.rete.runtime.util.RuleInstanceImpl@8cfed081"},"
    destinationElement": {"className": "duration2Score.a0","id": "com.ibm.
    rules.engine.rete.runtime.util.RuleInstanceImpl@8cfed081"}}
causal link: {"relation": {"type": "Execution","originClass": {"dataCauses"
    : ["miniloan.Loan/duration/GETTER#0"]},"destinationClass": {"dataCauses
    ": ["miniloan.Borrower/creditScore/GETTER#0"]},"originElementClass": {"
    name": "duration2Score.c1"},"destinationElementClass": {"name": "
    duration2Score.a0"}},"origin": {"variablesMap": {"miniloan.Loan/
    duration/GETTER#0": {"name": "miniloan.Loan/duration/GETTER#0","valType
    ": "Integer","type": "Default","domain": "[12,240]","min": 12,"max":
    500,"defaultValue": 240.0}}},"destination": {"variablesMap": {"miniloan
    .Borrower/creditScore/GETTER#0": {"name": "miniloan.Borrower/
    creditScore/GETTER#0","valType": "Float","type": "Default","domain": "
    [100,1000]","min": 100.0,"max": 1000.0,"defaultValue": 579}}},"
    originElement": {"className": "duration2Score.c1","id": "com.ibm.rules.
    engine.rete.runtime.util.RuleInstanceImpl@8cfed081"},"
```

```
destinationElement": {"className": "duration2Score.a0","id": "com.ibm.
rules.engine.rete.runtime.util.RuleInstanceImpl08cfed081"}}
```


Minimal decision causal links