Combining DMN and the Knowledge Base Paradigm for Flexible Decision Enactment

Ingmar Dasseville[1], Laurent Janssens[1,2], Gerda Janssens[1], Jan Vanthienen[2], and Marc Denecker[1]

[1] KU Leuven, DLS,
Celestijnenlaan 200A, 3000 Leuven, Belgium
{first.last}@cs.kuleuven.be
[2] KU Leuven, LIRIS,
Naamsestraat 69, 3000 Leuven, Belgium
{first.last}@kuleuven.be

Abstract. Representing business rules and the rules governing businesses is in itself a challenging task. Supporting the enactment of the represented rules poses even greater challenges. We present a novel approach to enact decisions represented using the Decision Model and Notation standard. The IDP knowledge base system is used as an inference engine for DMN decision models. The different forms of inference provided by the knowledge base system allow for flexible decision enactment. Our approach additionally allows for decision enactment by automatically generating an interactive graphical interface from the specification of the decisions.

Keywords: Knowledge Representation, Decision Modeling, Decision Enactment, DMN, Knowledge Base Paradigm

1 Introduction

Representing business rules and the rules governing businesses is in itself a challenging task [22]. Supporting the enactment of the represented rules poses even greater challenges. Many representation forms for both business rules and decision management have been proposed [11–13, 18], as well as several rule engines aimed at the execution of business rules [9, 16].

We present a novel approach for representing and enacting business rules and decisions. Our approach leverages the simplicity and expressive power of the Decision Model and Notation (DMN) standard to represent decisions and the capabilities of a knowledge base system, equipped with multiple inferences [7], to enact them. A user-friendly graphical user interface allows for interactive decision enactment on the modelled decisions.

We present our approach using the well-known uServ case study [8]. The rules entailed in the case study were represented in the DMN standard. To enact the represented decisions our approach utilizes a knowledge base system, IDP. In

IDP knowledge is represented using an extension of first-order logic, FO(·) language. By utilizing the available inferences of the IDP system we show how the capabilities of current decision management solutions can be expanded. In particular we show how the IDP knowledge base system can be used as an inference engine for DMN. Additionally we discuss several extensions to the uServ use case which, to the authors' knowledge, are difficult to achieve in existing solutions. The system can be accessed on `http://krr.bitbucket.org/autoconfig/`.

Our proposed system guides the user through the decision enactment process by automatically propagating the user's input, and thus immediately showing the consequences of their choices. In many rule languages, rules are computational devices computing heads from bodies; information flow is from bodies to heads. In the FO(·) language underlying the IDP system, sets of rules are definitions. Rules *define* heads in terms of bodies, but do not impose a computational direction. The IDP system can compute the defined concepts from known input, but it can also compute possible values for unknown input, given a known value for the defined symbols. As such, the information flow can go from body to head or from head to body, or a combination of both.

The remainder of the paper is structured as follows. Section 2 provides a background on the recent DMN standard, and on knowledge representation and in particular the IDP knowledge base system. The used example, the uServ case, is detailed and representations in both DMN and FO(·) are provided in Section 3. Section 4 gives an explanation of our approach. Several scenarios for enactment are provided in detail in Section 5. An overview of possible extensions to our approach is given in Section 6. The paper is concluded in Section 7.

2 Background

The approach proposed in this paper uses a knowledge base to represent the decision rules usually maintained in decision tables or other decision or business rules management solutions. While knowledge representation and decision management emerged in different fields, they share similar goals. Both domains are concerned with finding an optimal representation for knowledge. In the case of decision management this information entails the business decisions of an organization. In the domain of knowledge representation the information is represented in a general manner and can be used in several ways, e.g. to enact decisions. In both domains the emphasis is on ensuring the represented knowledge is actionable.

This section provides some background of both decision management and knowledge representation. The next subsection describes the newly proposed decision management standard, DMN. Subsection 2.2 describes the field of knowledge representation, and in particular the knowledge base paradigm used in our approach.

2.1 Decision Model and Notation

DMN, Decision Model and Notation, is a new standard managed by OMG, the Object Management Group, to describe decision logic in a vendor-independent language and executable notation [18, 19]. Without DMN, the logic of decisions (pricing, eligibility, loans, compliance, ...) is all too often described in imperfect requirements documents and then translated into enterprise systems, with numerous disadvantages, such as: traceability problems, long revision cycles, interpretation errors, poor consistency checking, lack of flexibility, etc.

DMN provides a tool-independent decision modelling notation, and is giving rise to a class of executable modelling tools with the ability to specify and execute business-friendly and verifiable decision models. Key features of DMN are: (1) Decision Requirements Diagrams (DRD) describing the dependencies of a decision on other supporting decisions and information sources and (2) the decision logic, typically either a decision table or an expression in a new expressions language (FEEL), usable by domain experts but rich enough to handle real-world decision logic. Decision tables have been around in various forms for decades, and DMN imposes a number of constraints on their format.

2.2 Knowledge Representation

The field of knowledge representation and reasoning is aimed at developing fundamental scientific understanding of knowledge and its uses to solve problems. Thanks to spectacular progress in solver technology in the fields of SAT [3], constraint programming [1], Answer Set Programming [2] this field is reaching maturity. This can be seen in the development of increasingly expressive formal specification languages (overlapping or extending classical logic, inductive definability, higher order aggregates and quantification, bounded arithmetic, etc.), an improved understanding of expressing expert knowledge as formal specifications in these rich languages, an increased understanding of how computational tasks can be decomposed in different inference tasks and the development of a range of effective inference engines for solving various types of tasks from formal declarative specifications.

The area is broadening its view to new areas of application. Business Systems present exciting challenges to the field of KRR: realistic applications of knowledge based nature, of great practical and economic interest and posing many challenging research questions because of the great variety of human knowledge and the need for modularizing it [10], the need for a range of different functionalities [17] (to compute decisions and run business processes), for verification tools (for compliance checking and others) [4, 10, 23] and the need of efficient scalable reasoning.

Most languages in these fields are designed with focus on a single or a few forms of inference. The aim of the FO($\cdot$)-KB project [7] is to build a logic (FO($\cdot$)) extending classical logic (FO) with language constructs of proven value from various KR languages and to develop the knowledge base system IDP to manage an FO($\cdot$) knowledge base and use it to solve a range of problems using various forms of inference. The value of being able to reuse the same declarative knowledge base for a range of functionalities was demonstrated recently in [14], where 8 different forms of inference reuse the same knowledge base to provide a flexible user-friendly interactive configuration system. More information on the IDP system can be found at `https://dtai.cs.kuleuven.be/software/idp`.

3 The Running Example

3.1 DMN

The dependencies, i.e. information requirements of the decisions in the uServ case are represented using a decision requirement diagram (DRD) in Figure 1. Square boxes, such as the "Eligible for Insurance" box, denote decisions, while rounded boxes, such as "Price", denote input elements. The arrows show the direction of the information requirements, e.g. the Eligible for Insurance decision requires the result of the Client Score decision and information on whether or not the client is a long term client.

For reasons of clarity the DRD shown in Figure 1 is limited to the case of insurance policies covering a single car and driver. Adding support for multiple cars and drivers requires the use of business knowledge models, which represent reusable pieces of knowledge. For this particular example this would significantly increase the amount of elements in the DRD.

The description of the logic of each decision is then detailed at the decision logic level, using decision tables, boxed expressions, and the FEEL language.

In Figure 2 the decision table of the Potential Theft Rating decision is shown. As is evident from the DRD this decision requires three inputs: whether the car model is on the High Theft Probability Auto list, whether it is a convertible, and the car's price. These elements are represented by the columns of the table. Each row then corresponds to one of the rules detailed in the uServ case.

3.2 FO model

The knowledge represented by the DMN model described in the previous subsection can be translated to FO(·), in the form of an inductive definition as follows.

$$\left\{\begin{array}{ll}
\forall c\ [Car] : PTR(c) = High & \leftarrow HighTheftProbabibilityList(CarModel(c)).\\
\forall c\ [Car] : PTR(c) = High & \leftarrow Properties(c, Convertible).\\
\forall c\ [Car] : PTR(c) = High & \leftarrow Price(c) > 45000.\\
\forall c\ [Car] : PTR(c) = Moderate & \leftarrow 20000 \leq Price(c) \leq 45000\\
 & \quad \wedge\ PTR(c) \neq High.\\
\forall c\ [Car] : PTR(c) = Low & \leftarrow Price(c) < 20000\\
 & \quad \wedge\ PTR(c) \neq High\\
 & \quad \wedge\ PTR(c) \neq Moderate.
\end{array}\right\}$$

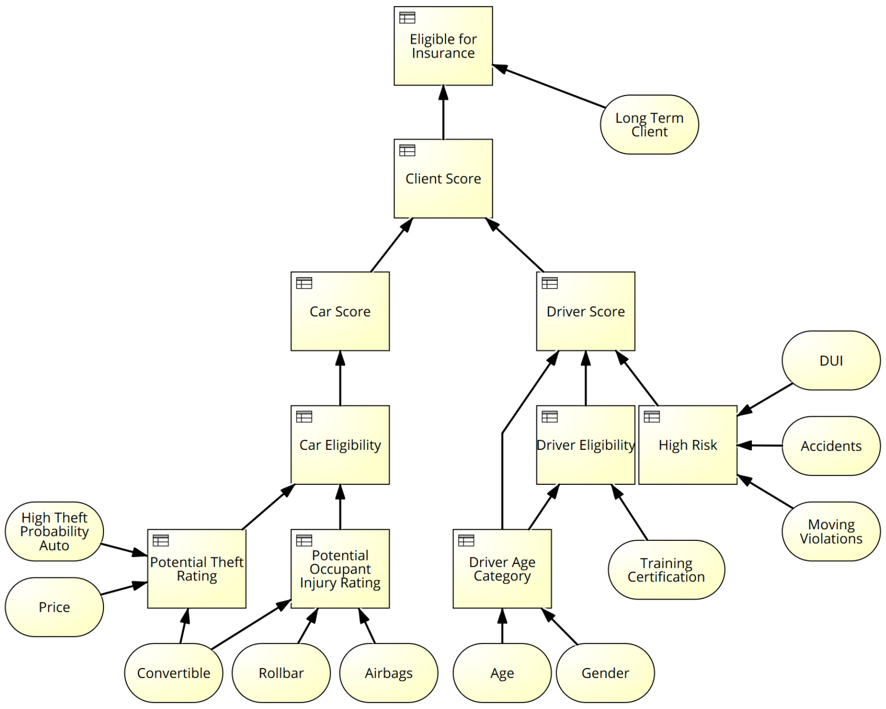
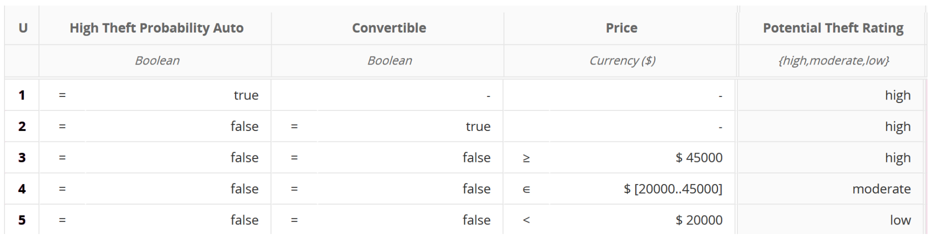


Fig. 1. Decision Requirement Diagram for the uServ case, for the case with a single car and single driver.



U	High Theft Probability Auto		Convertible		Price		Potential Theft Rating
	Boolean		Boolean		Currency ($)		{high,moderate,low}
1	=	true		-		-	high
2	=	false	=	true		-	high
3	=	false	=	false	≥	$ 45000	high
4	=	false	=	false	∈	$ [20000..45000]	moderate
5	=	false	=	false	<	$ 20000	low

Fig. 2. Decision table containing the decision rules for the Potential Theft Rating decision.

The rules have a one to one correspondence with the rules in the decision table in Figure 2. In other words the given table and this definition represent the same knowledge. All decision tables used to describe the uServ case can in a similar fashion be translated to definitions. An automatic translation from DMN to FO$(\cdot)$ is still ongoing work.

Since, in general, knowledge represented in the form of decision tables or DMN's boxed expressions can be translated to FO$(\cdot)$ the inferences present in

IDP's inference engine become available for use with DMN decision models. This allows business users to represent the needed knowledge in a business oriented way and leverage the power of generally applicable inferences to enact the modelled decisions.

In Section 5 three scenarios are introduced which show the power and usage of several of these inferences using an automatically generated graphical user interface.

4 From Decision Modelling to Enactment

As mentioned in Subsection 3.2 the rules specified in FO(·) for the uServ case have a one-to-one correspondence to decision tables and other DMN constructs. As such a translator from DMN to FO(·) would allow DMN to be used to model business decisions, or other applications where the knowledge is representable in DMN, while IDP can be used as the back-end to perform the required inferences.

4.1 Methodology

Building a new application using our approach is a three step process. In a first step the decisions are modelled using DMN. Thus allowing to leverage the understandability of DMN to represent the required knowledge. Translating this DMN decision model to FO(·) forms the second step. Currently this translation has to be done manually, however there is already a simple automated solution for automatically translating the decision tables to RULEML (as indicated in earlier work [21]), this approach can be extended to allow for a translation to FO(·). A direct translation from DMN to FO(·) is currently in progress. After the translation, as a last step, the interface is automatically generated as for the uServ demo. Using IDP's inference engine then allows to enact the decisions in a flexible way.

4.2 Architectural Information

The theory editor and interaction with the IDP software is the same as for the IDP Web-IDE[5]. The theory editor helps the user with syntax highlighting, some syntactic sugar and useful error messages and warnings.

Apart from the pre-existing IDP and IDE-tools. The application was fully developed using web technologies. Selections of the user are communicated using JSON objects and AngularJS[20] is used to tie the IDP interaction to the web interface.

Apart from the default theory, the tool itself is completely independent of the uServ use case. This means that this tool provides the same functionalities for all FO(·) theories and responds immediately to any change. For instance, adding new cars to the high theft probability list, will immediately be reflected in the user interface.

4.3 Execution

The IDP system supports the various functionalities through a combination of techniques from SAT, constraint programming and Answer Set Programming. A full description of the system can be found in [6].

The specification is not executable by itself and the functionality which is needed determines how the execution is done. For example, the optimal propagation does a full reduction of the problem to SAT. The approximate version propagates information using symbolic methods and does not do this transformation.

5 Enactment Scenarios

This section discusses three enactment scenarios provided by our approach. Firstly direct and indirect consequences of a user's choices are determined, this allows for interactive enactment which guides the user through the decisions. Secondly a partially provided solution can be extended to a full solution. This can be used to generate a possible example, or to determine the outcome of a decision when input is provided. Lastly our approach allows the optimisation of a decision outcome with respect to a given criterion. These scenarios are illustrated with several examples, using a graphical user interface.

The demo's user interface consists of two tabs. The theory tab consists of an editor for the IDP language, which consists of components described in [5]. In this tab an arbitrary FO(.) specification can be placed. In the configuration tab, this specification is converted into an interactive form. It consists of a set of properties, which can have one or multiple values. Every possible value for a property is listed under the property name, together with a "+" and a "-" button. With these buttons, the user can indicate whether this value is true for the property. A "+" indicates the value is true. If neither symbol is selected, it is unknown whether the value is true or not.

This interface allows various functionalities based on the specification in theory tab. These will be explored in the rest of this section.

5.1 Determining Consequences

In decisions input and output are strongly related, i.e. even when only partial input is available, information can already be derived. Often it is even possible to exclude certain outcomes, in some cases a unique outcome can already be determined [15]. In other words it is possible that choices made by the user, force some other values to hold, or prevent some other options. In our demo the system communicates this to the user through a green or red background for the respectively forced or impossible values. It also disables the buttons corresponding to that value, thus preventing the user from making inconsistent choices.

In IDP this functionality is provided by the *propagation* inference. This inference tries to find the intersection of all solutions which are still possible, given the set of current choices. Depending on the size of the problem, this can be done in an exact way, but there are also approximate (more efficient) versions of this inference available for problems where exhaustive search is not feasible.

Figure 3 shows the initial configuration of the system. For clarity the examples in the figures have been limited to the decisions involving only cars, and the policy is limited to a single car.

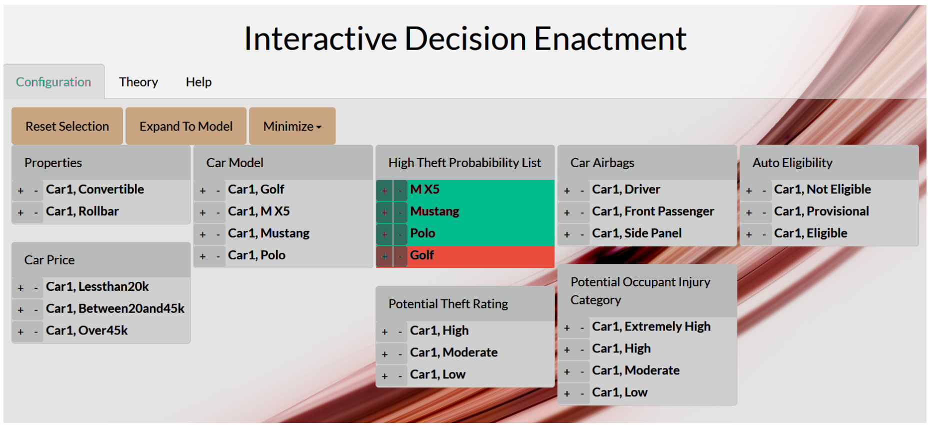


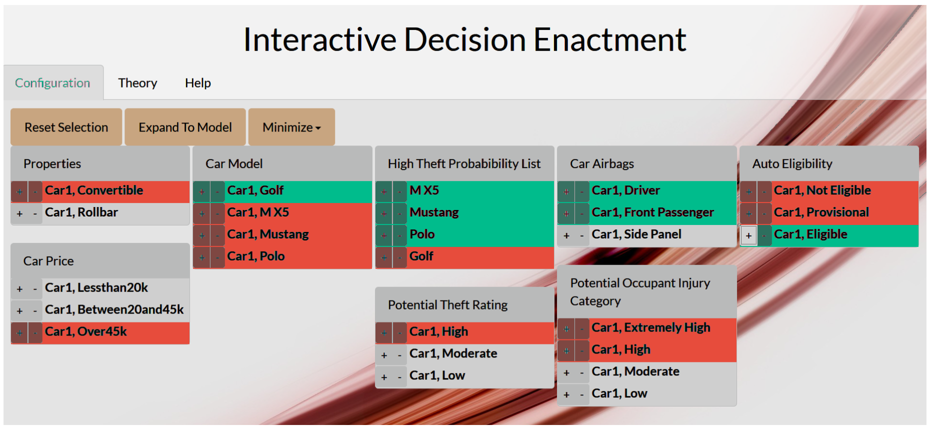
Fig. 3. Initial situation of the interface for the subset of decision related to cars, in the case of a single car.

Example 1. When the outcome for the Auto Eligibility decision for `Car1` is selected to be `Eligible` this choice is automatically propagated using the represented knowledge. This will effectively tell the user the minimum requirements for eligibility.

As shown in Figure 4 this implies the car must be a `Golf`, since the other models are all on the `High Theft Probability List` and would ensure for the car to get at most a `Provisional` eligibility label. For the same reason the car can not be a convertible of priced over \$45 000. Since the `Potential Occupant Injury Risk` of a car can be at most `Moderate` for the car to be `Eligible`, the system can derive the car must have at least `Driver` and `Front Passenger` airbags.

This example shows how a single choice of a desired outcome can have a large effect on the remaining possibilities. Figure 4 demonstrates how the extra information about the choices is communicated to the user. Partially provided input can have a similar effect.

Note the difference in background of the '+' sign for `Auto Eligibility(Car1, Eligible)` and that of the propagated information. A coloured background, as for `Properties(Car1, Convertible)` denote this information is a consequences of the choices that were made. In contrast the clear background of `Car1,` `Eligible` denotes that this tuple was a choice, i.e. it can be undone by the user. Undoing one of the choices can have an effect on the propagated information. The same equally applies for '-' signs.


Interactive Decision Enactment

Properties
- Car1, Convertible
- Car1, Rollbar

Car Price
- Car1, Lessthan20k
- Car1, Between20and45k
- Car1, Over45k

Car Model
- Car1, Golf
- Car1, M X5
- Car1, Mustang
- Car1, Polo

High Theft Probability List
- M X5
- Mustang
- Polo
- Golf

Potential Theft Rating
- Car1, High
- Car1, Moderate
- Car1, Low

Car Airbags
- Car1, Driver
- Car1, Front Passenger
- Car1, Side Panel

Potential Occupant Injury Category
- Car1, Extremely High
- Car1, High
- Car1, Moderate
- Car1, Low

Auto Eligibility
- Car1, Not Eligible
- Car1, Provisional
- Car1, Eligible

Fig. 4. Example of Figure 3 with the car chosen to be eligible for insurance.

It is important to note that if the inputs for a decision are completely provided, propagation will automatically determine the decision's outcome.

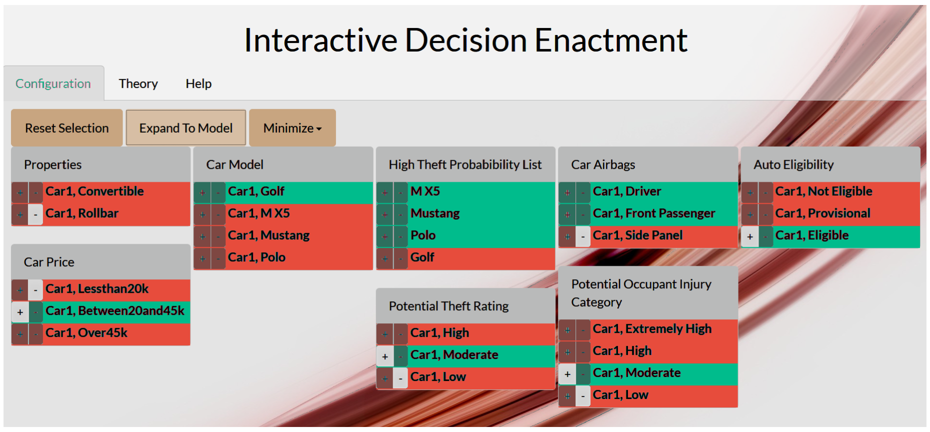
5.2 Extending a Partial to a Complete Solution

In some situations one can be interested in a possible solution given the current set of choices, e.g. retaking the previous example, Figure 4 can be expanded to a full solution, thus showing a possible situation for an eligible car. This functionality is available in IDP as the *model expansion* inference. Figure 5 shows a possible result of the system after a model expansion. Unlike for propagation the information added by expanding the current information to a full solution is not a direct consequence of user choices. Thus all information added by performing *model expansion* has the signs marked by a clear background, i.e. they are choices which can be undone.

5.3 Optimisation

Model expansion gives you an arbitrary solution of the problem. Sometimes one can be interested in a particular optimum according to some criterion such as cost or time. For example in the situation where you want to buy a car, and want to know what car to buy so your insurance policy is as cheap as possible.

The inference needed for solving optimization problems like this is also provided by IDP. The possible optimisation criteria defined in the specification are listed under the drop down menu "Optimize". The use of this inference is shown



Properties
- 🟥 Car1, Convertible
- 🟥 - Car1, Rollbar

Car Price
- 🟥 - Car1, Lessthan20k
- 🟩 + Car1, Between20and45k
- 🟥 Car1, Over45k

Car Model
- 🟩 Car1, Golf
- 🟥 Car1, M X5
- 🟥 Car1, Mustang
- 🟥 Car1, Polo

High Theft Probability List
- 🟩 M X5
- 🟩 Mustang
- 🟩 Polo
- 🟥 Golf

Potential Theft Rating
- 🟥 Car1, High
- 🟩 + Car1, Moderate
- 🟥 - Car1, Low

Car Airbags
- 🟩 Car1, Driver
- 🟩 Car1, Front Passenger
- 🟥 - Car1, Side Panel

Potential Occupant Injury Category
- 🟥 Car1, Extremely High
- 🟥 Car1, High
- 🟩 + Car1, Moderate
- 🟥 - Car1, Low

Auto Eligibility
- 🟥 Car1, Not Eligible
- 🟥 Car1, Provisional
- 🟩 + Car1, Eligible

Fig. 5. Example of Figure 4 after expanding the selection to a model.

in Example 2. All rules up to determining the policy's eligibility are used. Additionally, to allow for a meaningful use case, the example entails two drivers and three cars.

Example 2. The optimisation inference can for example be used to enact the eligibility decision for a customer asking for a policy covering as many of her cars as possible as well as preferably covering her grand son, however he has been convicted of a DUI in the past. She owns three cars, an `MX5`, a `Golf`, and a `Polo`. The Polo does not have driver side airbags. The policy should not be immediately rejected, however requiring an underwrite would not be a problem. The customer is not an `elite`, `preferred`, or `long term client`. After filling in all these details the optimize inference with the `Max Drivers Cars` term can be used to infer the most optimal policy.

The resulting policy covers both the customer and grandson, and two of her three cars. Using the interface it is also possible to see what would be the result if the grandson was not covered by the insurance policy, by deselecting him from the `Covers Driver` predicate. In this case, the third car, can also be covered by the policy. This leaves the customer with two choices, which are equivalent according to the criterion.

This inference can be used with any user defined term, similar to the `Max Drivers Cars` term. This term represents the sum of the amount of drivers and cars covered by the insurance policy. Since internally the minimize inference is used, the term is negated to allow for maximization. The `Max Drivers Cars` term is specified in IDP as follows.

```
term MaxDriversCars : V {
− (#{d : CoversDriver(d)} + #{c : CoversCar(c)})
}
```

6 Major Contributions and Future Work

The power of our proposed approach is in the availability of multiple forms of inference. The transformation from DMN to FO(·) allows the flexible reasoning, available in IDP, to be used for DMN models. In particular this means the same DMN decision model can be used for different scenarios. The demo explained in Section 5 shows only the propagation, model expansion, and minimization inference. However the other inferences available in the IDP system can equally be useful for business applications.

If the logic of a business process is represented in FO(·) the *progression* inference can be used to enact the process. Allowing the repercussions of performed actions to influence the state of the process. This would allow for the integration of decision and process management if both the decision logic and process logic is represented.

Additionally derived inferences can be implemented specifically geared towards decision management. Inferences can be defined which check the completeness and exclusivity of a decision, by using *model expansion* as a base for these inferences counter examples can be generated in case the decision in question is not complete of exclusive. These counter examples can be help in detecting where the violations occur, and aid in resolving them.

The graphical user interface provided for the demo, shows exactly one outcome of a decision enactment. However in many cases alternative outcome may be possible. Especially in the case of only partially provided input information, there may not be a unique outcome. The *model expansion* inference can be used to determine all possible outcome alternatives, allowing the user to choose its most preferable model. This extends to the use of the *optimize* inference. In Example 2 two outcomes are possible, one covering the grandson, and another covering the third car. Using a different user interface, the system would be able to provide both these alternatives.

The interface offers a way to show options, user choices and their consequences in a general manner. Currently it displays all possibilities in the form of a check box for true and false. For larger domains and applications this may become cumbersome. In the future the interface will be extended to allow the efficient visualisation of larger domains, as well as allow for more clear modularisation of the decisions. This would allow to include the price of the policy in the modelling, e.g. allowing it to be used with the optimisation inferences to minimize the price of the policy.

7 Conclusion

Enacting business decisions is challenging, especially for an expressive decision modelling language, such as DMN. Our approach uses the knowledge base system IDP as an inference engine for DMN. Through translation from DMN to an extension of first-order logic, we can leverage the flexibility available in IDP. Additionally a graphical user interface is provided which is automatically generated from the translated modelling. In this manner we make sure our approach is generally applicable. We presented our work using the well-known uServ product derby case study as a test case. The examples show our approach is able to enact decisions in several ways. It guides the user by immediately showing the consequences of the choices that are made, it allows a partial solution to be expanded to a total solution, and it allows to ask for an optimal solution with respect to a chosen criterion.

Our approach consists of three steps. The decision knowledge is modelled using DMN. This modelling is translated to an extension of first-order logic, FO($\cdot$). The graphical user interface is automatically generated from this representation, and uses IDP's inference engine to enact the decisions. Several extensions to this approach are mentioned as future work. The user interface can be extended to allow for more complex data types. The automation of the translation from DMN to FO($\cdot$), which is currently done manually, is also left as future work. New, derived inferences specific to decision enactment will be devised to provide more possibilities to use the represented decisions.

References

1. Krzysztof R. Apt. *Principles of Constraint Programming.* Cambridge University Press, 2003.
2. Chitta Baral. *Knowledge Representation, Reasoning, and Declarative Problem Solving.* Cambridge University Press, New York, NY, USA, 2003.
3. Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications.* IOS Press, 2009.
4. Elio Damaggio, Alin Deutsch, Richard Hull, and Victor Vianu. Automatic verification of data-centric business processes. In *Business Process Management - 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30 - September 2, 2011. Proceedings*, pages 3–16, 2011.
5. Ingmar Dasseville and Gerda Janssens. A web-based IDE for IDP. *CoRR*, abs/1511.00920, 2015.
6. Broes De Cat. *Separating Knowledge from Computation: An FO(·) Knowledge Base System and its Model Expansion Inference.* PhD thesis, KU Leuven, Leuven, Belgium, May 2014.
7. Broes De Cat, Bart Bogaerts, Maurice Bruynooghe, Gerda Janssens, and Marc Denecker. Predicate logic as a modelling language: The IDP system. *CoRR*, abs/1401.6312v2, 2016.
8. Jacob Feldman and James Taylor. Good old userv product derby in the brave new world of decision management. In *Building Business Capability 2015.* 2015.
9. E. Friedman-Hill. *Jess in action: Rule-based systems in Java.* In Action series. Manning, 2003.
10. Laura Giordano, Alberto Martelli, Matteo Spiotta, and Daniele Theseider Dupré. Business process verification with constraint temporal answer set programming. *Theory and Practice of Logic Programming*, 13(4-5):641–655, 2013.
11. Stijn Goedertier, Christophe Mues, and Jan Vanthienen. Specifying process-aware access control rules in sbvr. In *Advances in Rule Interchange and Applications*, pages 39–52. Springer, 2007.

12. Guido Governatori. Representing business contracts in *RuleML. Int. J. Cooperative Inf. Syst.*, 14(2-3):181–216, 2005.
13. Object Management Group. Semantics of business vocabulary and business rules (sbvr). OMG document number formal/08-01-02, January 2008. Version 1.0.
14. Pieter Van Hertum, Ingmar Dasseville, Gerda Janssens, and Marc Denecker. The KB paradigm and its application to interactive configuration. In *Practical Aspects of Declarative Languages - 18th International Symposium, PADL 2016, St. Petersburg, FL, USA, January 18-19, 2016. Proceedings*, pages 13–29, 2016.
15. Laurent Janssens, Johannes De Smedt, and Jan Vanthienen. Modeling and enacting enterprise decisions. In *Advanced Information Systems Engineering Workshops.* Springer, 2016. Accepted.
16. Christoph Nagl, Florian Rosenberg, and Schahram Dustdar. Vidre–a distributed service-oriented business rule engine based on ruleml. In *Enterprise Distributed Object Computing Conference, 2006. EDOC'06. 10th IEEE International*, pages 35–44. IEEE, 2006.
17. Anil Nigam and Nathan S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.
18. OMG. Decision Model and Notation, 2015. Accessed on 17.05.2016.
19. James Taylor, Alan Fish, Jan Vanthienen, and Paul Vincent. Emerging standards in decision modeling. In *Intelligent BPM Systems: Impact and Opportunity*, pages 133–146. iBPMS Expo, 2013.
20. AngularJS. `https://angularjs.org/`.
21. Jan Vanthienen. PROLOGA: from business knowledge modeling to ruleml. In *Rule Representation, Interchange and Reasoning on the Web. Series: Lecture Notes in Computer Science, Vol. 5321 The International RuleML Symposium on Rule Interchange and Applications (RuleML-2008)*, page 3p. Springer, 2008.
22. Jan Vanthienen and Filip Caron. Modeling business decisions and processes–which comes first? In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval and the International Conference on Knowledge Management and Information Sharing*, pages 451–456. KMIS, 2014.
23. Ingo Weber, Jörg Hoffmann, and Jan Mendling. Beyond soundness: on the verification of semantic business process models. *Distributed and Parallel Databases*, 27(3):271–343, 2010.