uDecide: A Protégé Plugin for Multiattribute Decision Making

Erman Acar, Manuel Fink, Christian Meilicke and Heiner Stuckenschmidt
Data & Web Science Group
University of Mannheim
D-68159, Mannheim, Germany
{firstname}@informatik.uni-mannheim.de

ABSTRACT

This paper introduces the Protégé plugin uDecide. With the help of uDecide it is possible to solve multi-attribute decision making problems encoded in a straight forward extension of standard Description Logics. The formalism allows to specify background knowledge in terms of an ontology, while each attribute is represented as a weighted class expression. On top of such an approach one can compute the best choice (or the best k-choices) taking background knowledge into account in the appropriate way. We show how to implement the approach on top of existing semantic web technologies and demonstrate its benefits with the help of an interesting use case that illustrates how to convert an existing web resource into an expert system with the help of uDecide.

Categories and Subject Descriptors

[**Artificial Intelligence**]: Knowledge Representation and Reasoning – *Description Logics*; [**Information Systems Application**]: Decision Support Systems – *Expert systems*; [**Web data description languages**]: Semantic web description languages – Web Ontology Language (OWL)

General Terms

Theory, Economics

Keywords

Description Logics, Utility Theory, Decision Making

1. INTRODUCTION

The study of preferences and decision support systems is an ongoing research subject in artificial intelligence, gaining more popularity every day. Since the first attention of multi-attribute utility theory in [7], numerous approaches have been proposed, including probabilistic, possibilistic, fuzzy and graphical models [6] amongst others. One recent approach stepping forward is the use of logical languages, e.g., [4, 9] to encode decision-theoretic problems.

We follow this line of research and introduce a Protégé plugin, uDecide, to encode decision making problems in the semantic web language OWL. uDecide uses standard reasoning techniques to perform the non-standard reasoning task of ranking choices with respect to a set of weighted attributes specified by a user. In this regard, it can be thought of as a decision support system. The plugin uses an ontology as background knowledge. A subset of the individuals that appear in this ontology can be defined as possible choices. uDecide is based on a multi-attribute utility theoretic assessment to yield the ranking of choices. Each attribute is represented by a class description weighted by a utility value which is asserted by the user. This yields a compact representation for a user's preference over attributes. Then, the preference relation is lifted to the set of choices via the aggregation of attributes that the choices satisfy due to their class membership.

The theoretical underpinnings of uDecide are weighted logics (e.g., see [5]). In particular, uDecide is based on a weighted Description Logics (DL) framework [1], referred to as DL decision bases, that does not require any specific DL language. This provides flexibility in the sense that one can use tractable fragments e.g., the DL_{Lite} family [3] or $\mathcal{EL}$ [2] if scalability is important, or expressive fragments when this is required by the domain that needs to be modelled. Also, our approach supports data types which is a desired property, since the use of numeric domains is common in the literature of decision theory (see [8]). In this work, we briefly present parts of the theoretical framework proposed in [1], and introduce our Protégé plugin uDecide which is based on this formalism.

uDecide can be understood as a generic out-of-the-box expert system that turns an ontology for a specific domain into a powerful decision support system for the domain described by that ontology. Such an expert system can be applied to support decision making in various domains. Within this paper we present a use case to illustrate how to use uDecide to generate reading proposals. This use case will also demonstrate that it is very easy to apply uDecide to any kind of domain for which an ontological representation is available as a knowledge base.

We introduce the theoretical foundations of our framework in Section 2. In Section 3, we first give a general description of our plugin. Then we present a use case which is based on an excerpt from DBpedia [1] that deals with books and authors. Finally, we conclude in Section 4.

2. THEORETICAL FOUNDATIONS

In this section, we introduce the theoretical underpinning of our plugin, which is a framework on weighted description logics. Our aim is to use an *a priori* preference relation over attributes (ontological classes) to derive an *a posteriori* preference relation over choices (ontological individuals).

[1] `http://wiki.dbpedia.org/`

We introduce a utility function U defined over the set of attributes $\mathcal{X}$, while we define a depending utility function u over choices which can be derived from U.

We represent each attribute in the original decision making problem as a class. This can be a named class or a complex class description defined with the vocabulary of the ontoloy. We assume that a total and transitive preference relation (i.e., $\succeq_{\mathcal{X}}$) over $\mathcal{X}$ is given as well as a function $U: \mathcal{X} \to \mathbb{R}$ that represents $\succeq$ (i.e., $U(X_1) \geq U(X_2)$ iff $X_1 \succeq_{\mathcal{X}} X_2$ for $X_1, X_2 \in \mathcal{X}$). The function U is a weight function, which assigns a weight to each class $X \in \mathcal{X}$. We denote the utility of a class $X \in \mathcal{X}$ by $U(X)$. U reflects an agent's preference relation over the set of attributes $\mathcal{X}$. The greater the utility of an attribute the more preferable the attribute is. Furthermore, we partition the attribute set $\mathcal{X}$ into two subsets; *desirable* that is the set of attributes with non-negative weights, denoted $\mathcal{X}^+$, and *undesirable* $\mathcal{X}^-$, that is the set of attributes with negative weights.

We call N_I as the set of named individuals. Let now $\mathcal{C} \subseteq N_I$ denote the finite set of choices. In order to derive a preference relation (*a posteriori*) over $\mathcal{C}$ (i.e., $\succeq_{\mathcal{C}}$) which respects $\succeq_{\mathcal{X}}$, we will introduce a utility function $u(c)$, which measures the utility of a choice c relative to the attribute set $\mathcal{X}$ and the utility function U over attributes as an aggregator. For simplicity, we will abuse the notation and use the symbol $\succeq$ for both choices and sets of attributes whenever it is obvious from the context. In the following, we define a particular u, which we call σ-utility.

Definition 2.1 (σ-utility of a choice). *Given a consistent knowledge base $\mathcal{K}$, and a set of choices $\mathcal{C}$,* the utility of a choice $c \in \mathcal{C}$ *is* $u_\sigma(c) = \sum_{X \in \mathcal{X} \wedge \mathcal{K} \models X(c)} U(X)$.

It is easy to see that u_σ induces a preference relation over $\mathcal{C}$ i.e., $u_\sigma(c_1) \geq u_\sigma(c_2)$ iff $c_1 \succeq c_2$. Also, notice that each choice corresponds to a set of attributes whose membership is logically entailed i.e., $\mathcal{K} \models X(c)$. We can now introduce the notion of a generic UBox, denoted $\mathcal{U}$, as follows.

Definition 2.2 (UBox). *A UBox is a pair $\mathcal{U} = (u, U)$, where U is a utility function defined over $\mathcal{X}$ and u is a utility function defined over $\mathcal{C}$.*

Informally, a UBox $\mathcal{U}$ encodes user preferences U and defines their aggregation via u which defines the utility of choices. Next, putting things together we introduce the notion of a *decision base*, which can be interpreted as a model for an artificial agent in a decision situation. A decision base is a triple which consists of a DL knowledge base $\mathcal{K}$, a finite set of available *choices* $\mathcal{C}$, and encoded user preferences along with the utility function of choices, UBox $\mathcal{U}$.

Definition 2.3 (Decision Base). *A* decision base *is a quadruple $\mathcal{D} = (\mathcal{K}, \mathcal{C}, \mathcal{U})$ where $\mathcal{K} = (T, A)$ is a consistent knowledge base (with T is an acyclic TBox and A is an ABox), $\mathcal{C} \subseteq N_I$ is the set of choices, $\mathcal{U} = (u, U)$ is a UBox.*

Informally, the role of $\mathcal{K}$ is to provide assertional information about the choices at hand, along with the general terminological knowledge information that the agent may require to reason further over choices. In this work, we will assume the commonly used *rationality criterion* that is the rational agent will always pick up the choice(s) with the maximum utility [7, 8].

3. SYSTEM DESCRIPTION

We implemented our approach as Protégé plugin available at `https://code.google.com/p/udecide/`. We first briefly describe the functionality and architecture of our plugin in Section 3.1. Then we present a use case that shows how a user interacts with the plugin in Section 3.2. This use case also illustrates how the plugin can be used as an out-of-the-box expert system for any knowledge domain available as ontology.

3.1 Implementation

Our Protégé plugin is compatible with both Protégé Desktop version 4.3 and 5.0. As reasoning component we used the Konclude reasoner [10] which turned out to be the best OWL reasoner for our purpose with regards to performance issues. Our choice was motivated by the evaluation results reported at `http://dl.kr.org/ore2014/results.html`.[2] uDecide requires Konclude to be running in the background to connect to it via OWLlink.

Our implementation is straight forward. First, an ontology needs to be loaded via the standard Protégé file menu. This ontology acts as a knowledge base $\mathcal{K}$. After switching to the uDecide tab, the user can specify the set of possible choices by specifying a class C defined in $\mathcal{K}$. All instances of C are treated as choices, which corresponds to the set of choices $\mathcal{C}$ in the theoretical framework. The attributes and their utility can then be specified on top of vocabulary defined in $\mathcal{K}$. Once the type of choices and the attributes with their corresponding utility have been specified, a connection to Konclude is established via HTTP. We will illustrate these steps in the subsequent section in more details. For each attribute, we request from the reasoner all named individuals satisfying the intersection of the attribute's class expression and the class that defines the type of choices. The result shown consists of a ranked list of all individuals returned by at least one query and their utility which is derived from the satisfied attributes. Since Konclude does not support instance satisfaction queries for anonymous class expressions, we create a temporary ontology that is transferred to Konclude on calculation time and merged with the knowledge base ontology. We add to this ontology an equivalent classes axiom between each utility assertion's class expression and a named dummy class. We then separately query the individuals for each named dummy class.

As described on our homepage, we recommend to configure Konclude to load the knowledge base already on start-up to speed up the calculation. Because of increasing computation time and memory limitations it is required to do this when working with large knowledge bases. If the knowledge base was already loaded into Konclude on start-up, only the (very small) temporary ontology needs to be transferred to Konclude. Otherwise, the union of both the temporary ontology

[2]We would like to thank Andreas Steigmiller for his support related to using Konclude.

and the (potentially very big) knowledge base is transferred.

3.2 Use Case

As an illustrating use case, we applied our approach to the domain of books and authors. In particular, we used our framework to support a user in finding interesting authors by specifying her interests as attributes. Instead of working with an artificial example, we used an existing subset of DBpedia that deals with the chosen topic. The core domain contains relevant information about books and their authors. With respect to our use case, DBpedia suffers from its restricted set of terminological axioms and its incompleteness regarding the sparse usage of some properties. To overcome these problems, we decided to extend the core domain with information about cities. In particular, we added for each city that was listed as birth or death place of an author the country in which it is located. Furthermore, we added some axioms specifying nationality classes e.g., *Spanish* is defined as the class of those persons that were born in or died in a city located in Spain or whose nationality is Spain. Thus, by using the nationality classes, the nationality of authors for whom no nationality object property assertion exist, can still be inferred by their birth and death places. Note that the nationality has been specified directly only for 21.3% authors, while 46,9% have a "derived nationality" via our axiomatization. Obviously there exist some people that were born in Spain whose nationality is not Spanish. However, since the final choice is made by the user, the gain incoverage might be more useful than the loss in precision.

This extension illustrates that, in the context of a reasoning based approach, it is possible to leverage background knowledge that seemed not to be relevant at first sight. The information that Barcelona is located in Spain and that some author was born in Barcelona can thus affect the ranking of choices if we specified that we prefer Spanish authors as an attribute. It also shows that a reasoning based approach can help to overcome some problems related to incomplete data in the knowledge base. The dataset and some instructions on how to use it can be found at `https://code.google.com/p/udecide/wiki/BookUseCaseExample`.

Suppose that a user wants to find a new author who writes books that are similar to the ones that she likes. First of all, feasible choices have to be defined as the instances of the class *dbp:Author*. Figure 1 depicts a screenshot of the uDecide tab. The class to which the choices belong has been specified in the respective text field in the upper right corner. An arbitrary concept description can be specified as long as it is in the signature of the previously loaded ontology.

Now suppose that our user likes the author Stephen King. Thus, she likes to read authors that are influenced by Stephen King which is expressed by the the positive weight attached to the attribute $\exists influencedBy.\{Stephen\_King\}$. The attributes specified by a user can be seen in the uDecide tab on the left side of Figure 1. Note that the concept descriptions are specified in the Manchester syntax[3] supported by the Protégé Editor. All attributes are specified within a dialog box that uses the auto-complete functionality of Protégé as well as its syntax checking capability. Only if a class expression is syntactically correct, a button will be enabled to add it to the UBox.

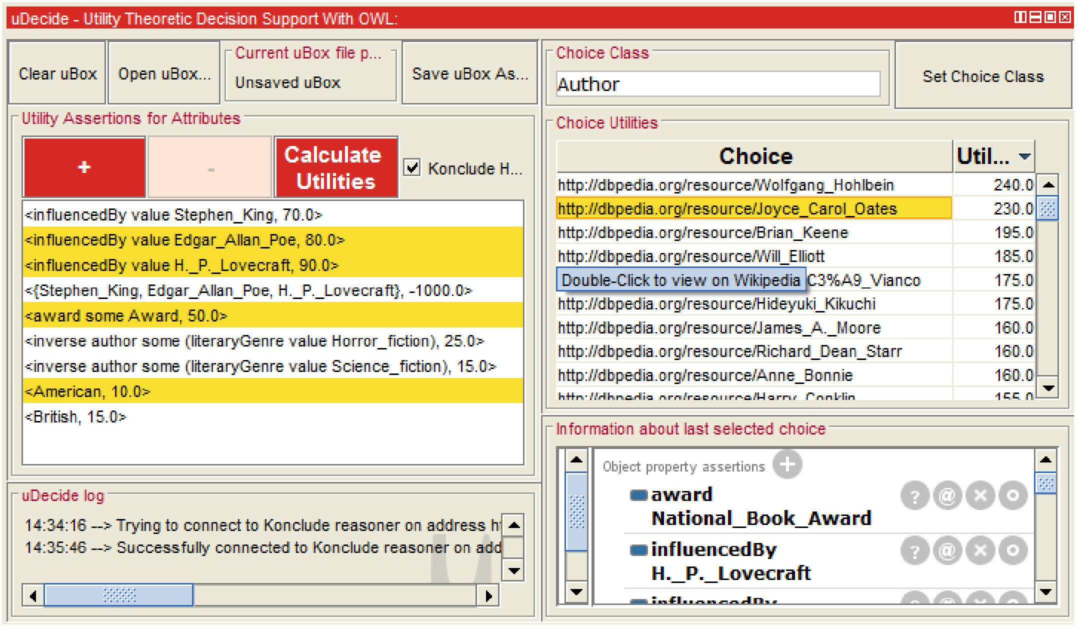
Overall, nine attributes have been specified. The first three attributes express that the user prefers authors that are influenced by her favorite authors. By adding a negative value to the fourth attribute, the user ensures that the three authors that she already knows will be ranked low in the ranking of choices. The fifth attribute is added to increase the utility of those authors that received some award by 50. Moreover, the user specifies that she likes authors writing books that belong to the genre of horror fiction or science fiction. These attributes have a relatively low utility value. Finally, it is specified that the user likes American and British authors, slightly preferring British.

The results that are finally calculated will only include individuals that satisfy the choice class expression and at least one of the attributes. This calculation is started by clicking on the "Calculate Utilities" button. The ranked choices are presented on the right side of Figure 1 in descending order based on their utility. The best choice is the author Wolfgang Hohlbein (240), followed by Joyce Carol Oates (230) and many more lower ranked choices. Thus, the most reasonable choice for the user is to look at the author Wolfgang Hohlbein in more details, given that her attribute specification and the underlying knowledge base are complete and correct. However, it might often be the case that a user wants to explore the results in more detail, for example to get an explanation about their ranking position. This can be done by clicking on one of the proposed choices. Figure 1 illustrates this for Joyce Carol Oates. The utility score of 230 is based on the fact that Joyce Carol Oates was influenced by Edgar Allen Poe and by H.P. Lovecraft, that she won at least one award and that she was born in New York, therefore being classified as American. Each of the satisfied attributes is highlighted in the left panel. Furthermore, all assertions about the selected choice are shown in a panel in the lower right corner. Again, we have used the Protégé default of presenting this information. Vice versa, it is also possible to select one (or multiple) of the attributes. This results in those choices being highlighted that satisfy the selected (all selected) attribute(s) (not shown in Figure 1).

Our use case and the presented example illustrates both the benefits as well as some drawbacks of our approach. First of all, we could apply our Protégé plugin directly to the domain of books and authors without the need for any further modifications or extensions. This resulted in an expert system which makes proposals about interesting authors or books. The only required ingredient was an ontology that covers the domain in an appropriate way. We decided to use DBpedia, which features a comprehensive ABox but a flat and inexpressive TBox. Thus, the potential reasoning capabilities of our approach have only a limited impact with respect to our use case. For the majority of attributes, whether or not it holds, can be decided by a direct look-up. We gave one example (nationality classes) to illustrate how to overcome these limitations by adding additional axioms and relevant data from other domains covered in DBpedia.

The chosen use case is an extreme case where we have a very

[3]`http://www.w3.org/TR/owl2-manchester-syntax/#The_Grammar`



<influencedBy value Stephen_King, 70.0>
<influencedBy value Edgar_Allan_Poe, 80.0>
<influencedBy value H._P._Lovecraft, 90.0>
<{Stephen_King, Edgar_Allan_Poe, H._P._Lovecraft}, -1000.0>
<award some Award, 50.0>
<inverse author some (literaryGenre value Horror_fiction), 25.0>
<inverse author some (literaryGenre value Science_fiction), 15.0>
<American, 10.0>
<British, 15.0>

```
14:34:16 --> Trying to connect to Konclude reasoner on address h
14:35:46 --> Successfully connected to Konclude reasoner on add
```

Choice	Util...
http://dbpedia.org/resource/Wolfgang_Hohlbein	240.0
http://dbpedia.org/resource/Joyce_Carol_Oates	230.0
http://dbpedia.org/resource/Brian_Keene	195.0
http://dbpedia.org/resource/Will_Elliott	185.0
Double-Click to view on Wikipedia C3%A9_Vianco	175.0
http://dbpedia.org/resource/Hideyuki_Kikuchi	175.0
http://dbpedia.org/resource/James_A._Moore	160.0
http://dbpedia.org/resource/Richard_Dean_Starr	160.0
http://dbpedia.org/resource/Anne_Bonnie	160.0

award National_Book_Award
influencedBy H._P._Lovecraft

Figure 1: Screenshot of uDecide showing a ranked list of authors according to the attribute specification.

large ABox and a comprehensive set of choices that are not all known in advance to the agent, while we have a relatively inexpressive TBox. On the other hand, there might be cases where we have only a limited number of choices together with a rich logical axiomatization of relevant background knowledge. Our tool is designed to support both cases as well as any kind of hybrid scenario.

4. CONCLUSION AND FURTHER WORK

In this paper, we presented the Protégé plugin uDecide. uDecide computes the utility for a set of choices by aggregating the utility value for each satisfied attribute. Since each attribute corresponds to a class description, standard reasoning techniques can be used to check whether an attribute is satisfied. We used the Konclude [10] reasoning system to conduct the required reasoning tasks. The results of this computation are presented to the user as a ranked list of choices. To our knowledge, we have introduced the first system that allows to encode and solve multiattribute decision problems in terms of weighted description logics. Since we implemented our approach as a Protégé plugin, our approach can easily be used by the DL community.

We have demonstrated within our use case how to use uDecide as an expert system that recommends new authors to a user. Moreover, we have also shown that our current implementation, by using the reasoning system Konclude, is capable to deal with large real-world datasets. We are currently investigating datasets from the biomedical domain and from the domain of life sciences. As a direction for future research, we are aiming to extend our framework and the plugin to deal with uncertainty (e.g., it is uncertain to a particular degree that an author born in Barcelona is a Spanish person), and lifting utilities of choices to expected utilities.

5. REFERENCES

[1] E. Acar and C. Meilicke. Multi-attribute decision making using weighted description logics. In T. Lukasiewicz, R. Peñaloza, and A.-Y. Turhan, editors, *PRUV*, volume 1205 of *CEUR Workshop Proceedings*, pages 1–14. CEUR-WS.org, 2014.

[2] F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In *IJCAI*, pages 364–369. Professional Book Center, 2005.

[3] D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.

[4] Y. Chevaleyre, U. Endriss, and J. Lang. Expressive power of weighted propositional formulas for cardinal preference modelling, Dec. 08 2006.

[5] S. Kaci. *Working with Preferences: Less is More.* Springer Verlag, Berlin, 2011.

[6] C. Kahraman. *Multi-Criteria Decision Making: Theory and Applications with Recent Developments.* Springer, 2008.

[7] R. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs.* J. Wiley, New York, 1976.

[8] G. Parmigiani and L. Y. T. Inoue. *Decision Theory Principles and Approaches.* John Wiley & Sons, Ltd, 2009.

[9] A. Ragone, T. D. Noia, F. M. Donini, E. D. Sciascio, and M. P. Wellman. Weighted description logics preference formulas for multiattribute negotiation. In *Proceedings of Scalable Uncertainty Management, Third International Conference, SUM 2009.*

[10] A. Steigmiller, T. Liebig, and B. Glimm. Konclude: system description. *Web Semantics: Science, Services and Agents on the World Wide Web*, 27:78–85, 2014.