doi:10.3233/SHTI210540

Automated Creation of Expert Systems with the InteKRator Toolbox

Daan APELDOORN[a,b,1] and Torsten PANHOLZER[a]
[a] *Institute of Medical Biostatistics, Epidemiology and Informatics (IMBEI), University Medical Center of the Johannes Gutenberg University Mainz, Germany*
[b] *Z Quadrat GmbH Mainz, Germany*

Abstract. Expert systems have a long tradition in both medical informatics and artificial intelligence research. Traditionally, such systems are created by implementing knowledge provided by experts in a system that can be queried for answers. To automatically generate such knowledge directly from data, the lightweight InteKRator toolbox will be introduced here, which combines knowledge representation and machine learning approaches. The learned knowledge is represented in the form of rules with exceptions that can be inspected and that are easily comprehensible. An inference module allows for the efficient answering of queries, while at the same time offering the possibility of providing explanations for the inference results. The learned knowledge can be revised manually or automatically with new evidence after learning.

Keywords. Expert system, knowledge representation, machine learning

1. Introduction

1.1. Background

In recent years, artificial intelligence (AI) experienced a boost in popularity. Especially machine learning, as a subfield of AI, raised a lot of attraction. While many machine learning approaches deal with numerical methods to learn from data (e.g., for classification or image recognition), the traditional discipline of expert systems is related to *knowledge representation*, a subfield of AI, where knowledge is modeled by rules and logic-based approaches.

Expert systems have a long tradition, especially in the medical sector (e.g., MYCIN [17]), and are used to represent and manage the knowledge of experts in a way that allows to query it for answers. The knowledge usually exists in an explicit form and thereby can provide transparency (to some degree), which can help to understand the inference results.

Machine learning approaches (like neural networks) have shown impressing results, e.g., in image recognition tasks. However, such approaches are often known for lacking transparency and explainability of the results, since the learned knowledge is implicitly contained in a huge numerical representation (e.g., millions of weights in a neural network).

[1] Corresponding author, daan.apeldoorn@uni-mainz.de

The main idea of the InteKRator approach is to bring machine learning and knowledge representation closer together to create an expert system automatically from data, instead of manually modeling the system's knowledge. Figure 1 visualizes this idea in comparison to the manual creation of an expert system.

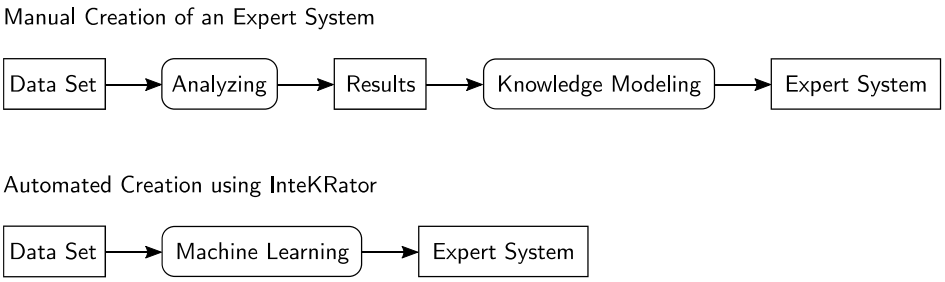

Figure 1. Comparison of Manual vs Automated Creation of an Expert System: The figure compares the manual creation against the InteKRator approach. The InteKRator toolbox can overtake several steps in the creation process using its machine learning capabilities.

1.2. Motivation

The InteKRator toolbox aims at implementing several results in the context of a special knowledge bases and corresponding algorithms, that have been developed in the recent years and presented, e.g., in [2,3,5] (among other works, see Section 2 or [12] for a more elaborate list of related works). The knowledge bases were especially geared toward the representation of knowledge learned by agents in a *human-readable* way.

InteKRator further aims at providing the implementations in a *lightweight* toolbox (where the term "lightweight" refers to (1) having *no external dependencies*, (2) being *extremely small in size*, and (3) having an *easy-to-use interface*—both as a library as well as a stand-alone command line application).

Since the properties of the resulting toolbox seem to cover well the major requirements of expert systems in the medical domain, such as the measurement of the inference quality (see the upcoming section 1.3), another motivation of this work is to further outline its usefulness in the medical context.

InteKRator also supports knowledge representation for continuous data, a property that is rather rarely provided by other symbolic or logic based approaches.

The InteKRator toolbox can be considered the *reference implementation* of the underlying knowledge representation paradigm, as well as for the learning, inference and revision algorithms. To the extent of the authors' knowledge, there is no other library available that consolidates and implements these approaches.

1.3. Requirements

To be able to learn an expert system from data in the context of medical applications, the following requirements are of importance:

- *Machine Learning*: Rule-based knowledge must be learned from a data set.
- *Comprehensibility*: The learned rule-based knowledge must be comprehensible for humans (also for people not familiar with logic-based approaches).
- *Certainty*: The quality of the learned knowledge must be measurable.

- *Inference*: It must be possible to draw meaningful conclusion from the knowledge that has been learned.
- *Revision*: It should be possible to integrate new evidence in the already learned knowledge (e.g., in case a new study provides new evidence after the knowledge was learned).
- *Usability and Tooling*: The aforementioned requirements should be easy to use and integrate in other applications (like web applications, smartphone apps, etc.).

The InteKRator toolbox [12], meets the above requirements in a lightweight manner: Besides a machine learning module that creates comprehensible knowledge based on rules with exceptions from a data set, it allows to evaluate the resulting knowledge against the original (or other) data sets. InteKRator allows for retrieving inferences for new cases and knowledge that was once learned can be later revised manually or automatically. Furthermore, being implemented in plain Java [18], no additional libraries are required and the toolbox can be used as a Java library or as a stand alone application from the command line. (Details will be provided in Section 3.)

2. State of the art

Expert systems have a long tradition in medical informatics and AI research. On the one hand, software exists that can be used for creating expert systems, which is mostly geared toward knowledge engineers, for modeling rule-based knowledge manually. Examples for these kinds of approaches are logic programming systems like Prolog [16] or *answer set programming* (ASP) [7] solvers like Clingo [8]. Unlike these approaches, the InteKRator toolbox does not only support the manual creation of knowledge in the form of rules with exceptions, but also allows for *learning* such knowledge directly from a data set. Due to relying on the intuitive principle of rules with exception, InteKRator does not require a user to have expertise in logic, neither for modeling nor for reading automatically learned knowledge.

On the other hand, there are approaches from statistics and machine learning, like decision trees (e.g., [6], Section 5.3) or Bayesian networks (e.g., [6], Section 12.2), which are able to learn and/or represent relations in the data. Decision trees are able to learn a classification scheme of the data, which is represented in the form of a tree structure. Additional explanatory information may increase their comprehensibility. However, the knowledge is not represented in the form of formal rules and, in contrast to a knowledge base produced by InteKRator, a decision tree focuses more on the structural dependencies among variables rather than on rule-based dependencies of their values. Additionally, as many other graph-based approaches, a visualization of the tree structure may have limitations in the representation of the knowledge, especially when dealing with real-world problems having a large number of nodes.

Bayesian networks are oftentimes used to predetermine structural dependencies and to learn the corresponding conditional probability distributions from data. Although there exists also methods to learn the network structure as well [10], it can be hard to interpret the results especially in case of a larger number of nodes.

A further, well-established approach that combines learning with the idea of creating rules, is the Apriori algorithm by Agrawal et al. [1]. This approach is usually used for

learning association rules (as known, e.g., from recommender systems) rather than learning a complete knowledge base. However, it uses techniques that are related to preliminary works of InteKRator's basic learning algorithm, which has been considered in detail in [5].

Nowadays, there is still a need for expert systems in the medical sector and expert systems research remains an active field. A recent example is [19], where an approach uses ASP for representing the knowledge of cancer therapies. There, the knowledge was gathered from clinical experts over about 16 years (according to [19]) and later formalized manually in the form of rules, e.g., with default negations, without involving machine learning for automating this process.

In contrast to the aforementioned approaches, the InteKRator toolbox offers a machine learning module that allows for learning a complete knowledge base directly from a provided data set. One of the major strengths of InteKRator is that the resulting knowledge bases are built on the principle of rules with exceptions and are therefore intuitively comprehensible, even for people who do not have a strong background in logic: In a study by Krüger et al. [14], the comprehensibility exceeded that of ASP.

Preliminary works of InteKRator (e.g., [5]) have been tested and used in different applications: in the context of AI in games (e.g., [9]), for improving learning capabilities of agents (e.g., [4]), and for solving job-shop problems [13]. Recently, an earlier version of the InteKRator toolbox has also been used to learn behavioral rules for optimizing hospital processes [3]. However, to the best of our knowledge, this is the first work proposing the InteKRator toolbox as a whole and applying its capabilities for learning of medical expert systems.

3. Concept

InteKRator is a lightweight toolbox that integrates knowledge representation techniques and machine learning. The main idea is to learn a knowledge base from a data set, so that the knowledge base describes the inherent structure of the data set compactly in a human readable way. Moreover, InteKRator allows for inference queries on such knowledge bases and it is also possible to modify a knowledge base with new evidence. With InteKRator, a knowledge base can also be evaluated against a data set, to measure the certainty of the knowledge. Furthermore, InteKRator's functionalities can be easily integrated in web and other applications (both as a Java library and as an external process), and they can also be accessed by a command line interface. With these features, InteKRator shows great potential to satisfy the requirements from Section 1.3.

For the data processing, InteKRator reads and writes plain (space-separated) text files and can also provide its output on the standard out of the process. By this, it can be easily embedded in diverse applications, both as a library as well as an external process, or it can be used as a stand-alone command line application. A detailed explanation on the format of input and output files can be found in [12].

The following subsections will now demonstrate and explain the central concepts of the InteKRator toolbox in the context of a small synthetic example data set.

3.1. Learning Module

The input of the learning module is a text file of $(n + 1)$ space-separated columns: The first n columns represent the values of the features of the data set; the remaining last

column contains the values of an outcome variable. The output is a text file containing a knowledge base comprising the rule-based knowledge about the data set. The upper part of Figure 2 shows a small synthetic input data set as an example; the lower part shows the resulting knowledge base (which will be considered later).

Applying the learning module to the data set results in a knowledge base which compactly represents the knowledge contained in the data on several levels of abstraction: The topmost level represents the most general rule, whereas the lower levels comprise the more specific knowledge in the form of exceptions of rules on the higher levels. The learning algorithm tries to represent the knowledge with as few as possible rules and as few as possible exceptions to create a compact and comprehensible representation. Figure 2 shows an example of the knowledge base extraction process together with an interpretation in natural language (on the bottom right), where the shades outline how the knowledge base can be read.

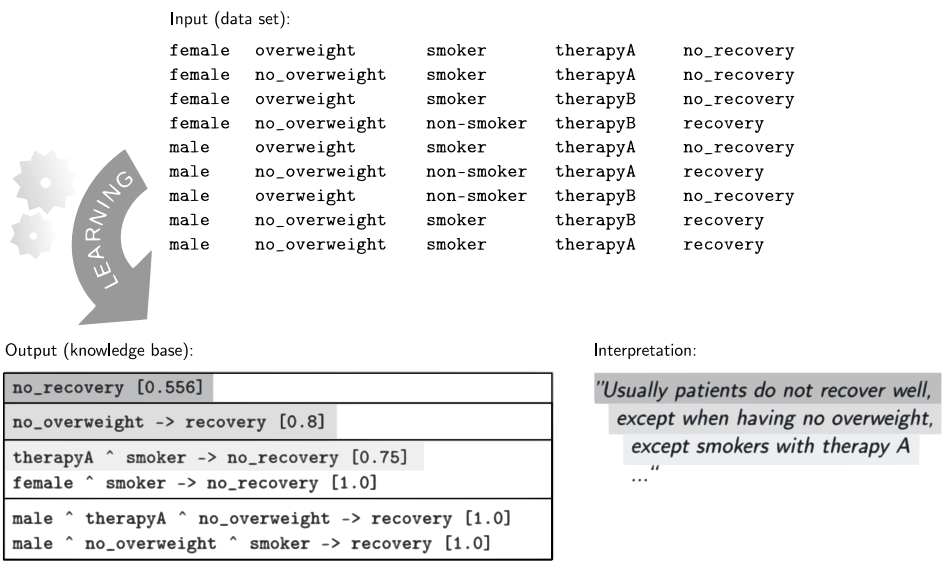

Figure 2. From Data Set to Expert System: The upper part shows a small synthetic data set as an example. The left side of the lower part shows the knowledge base learned from data using InteKRator. The right side of the lower part shows that the knowledge base can be read top-down (as indicated by the shades).

The numbers that are attached to the rules in the learned knowledge base on the bottom left of Figure 2 represent the conditional probabilities $P(conclusion \mid premise)$.

The learning module of InteKRator is not only able to handle columns of discrete data. For continuous data columns, it offers advanced discretization options based on an iterated clustering approach: By automatically detecting data columns of numeric data, the algorithm performs a standard *k-means clustering* and iterates it with increasing k (if desired), until no further non-empty clusters can be found. All values belonging to the same cluster are then treated as one discrete value by the learning module. A naming option allows for mapping eligible names to the learned clusters.

3.2. Inference Module

The inference module allows for querying a learned knowledge base for answers. Such a mechanism is an essential part of an expert system. InteKRator's inference module allows for providing queries to the knowledge base, by providing a set of feature values. The inference algorithm, which has its origins in [4], searches the knowledge base upwards for the most specific rule whose premise is satisfied by the provided input values (i.e., whose premise is a subset of the query's set of feature values). Note that the rules are ordered from general to specific in the knowledge base (as described in Section 3.1). The output is the conclusion of the found rule.

In the recent years, similar to field of machine learning, it became also more and more relevant for inference systems do not only provide conclusions, but to also explain the inference results. InteKRator's inference module provides an option that allows for explaining based on which rule the result was inferred (together with the conditional probability of the result). Figure 3 visualizes the concept of the inference module.

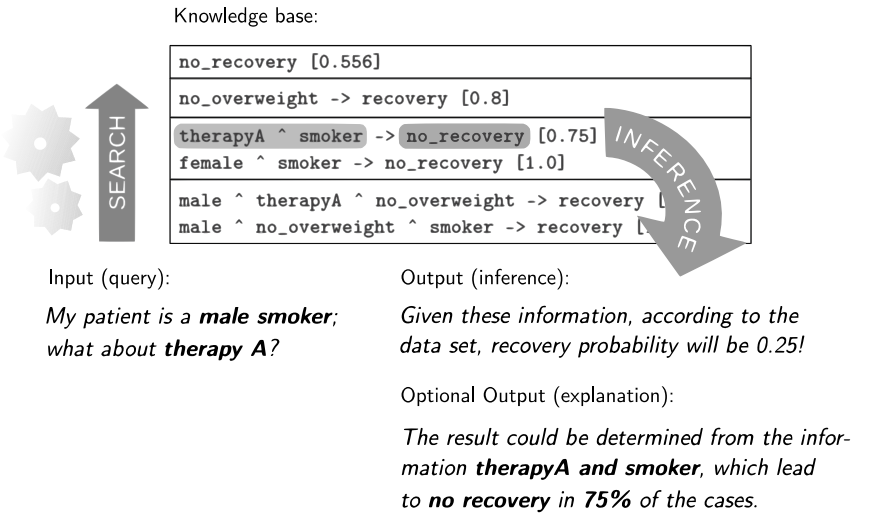

Figure 3. Inference Query and Outputs: The inference module of the InteKRator toolbox searches the learned knowledge base upwards to find the most specific rule whose premise is satisfied by the input query. The output is the conclusion of the found rule together with the conditional probability $P(conclusion \mid premise)$. Optionally, an explanation from which rule the conclusion was inferred can be provided.

3.3. Check Module

In the previous two subsections it has been described, how an expert system can be learned from data and how inference queries can be performed on such a system. An interesting question is now, how trustworthy such a learned knowledge base is. For this purpose, InteKRator offers a check module, that takes a data set and a knowledge base as input and performs an inference query on the knowledge base for each row of the data set. It is then evaluated, for how many data rows the correct conclusion could be drawn through the knowledge base (i.e., for how many rows the outcome value can be inferred from the n feature values of the respective row, cf. Section 3.1). Note that the data set must not necessarily be the one from which the knowledge base was learned (e.g., it is

also possible to split a data set in training and test data and to use the check module with the test data set, as usually in machine learning).

In the case of the synthetic data set considered in Figure 2 and Figure 3, in none of the data rows the exact same combination of values refers to a different outcome. Thus, the certainty provided by the check module is 1.0 here.

3.4. Revision Module

Another challenge of expert systems is the integration of new evident knowledge (e.g., in case a new study reveals new insights).

A solution to the problem of incorporating new knowledge is provided by the AI subfield of *belief revision*. The revision module of the InteKRator toolbox offers a revision mechanism that allows for incorporating new evidences in a learned knowledge base. The revision algorithm, which has been introduced in [9], tries to incorporate a new rule by affecting as few as possible of the knowledge that is already contained in the knowledge base. This is done, according to [9], by adding the new rule as exception on the most specific (i.e., the bottommost) level of the knowledge base. If a conflicting rule does already exist there, the conflicting rule will be removed and if the correct conclusion still cannot be drawn, the new rule with the new evidence will be added.

The presented approach conforms to common quality criteria for believe revision [11], as has been outlined in [2]. Note that, in principle, InteKRator can perform revision on any level of the knowledge base. However the quality criteria are only valid in case revision is done on the most specific level with a complete premise (i.e., on the $(n + 1)$-th level, where n is the number of features; see also Section 3.1).

4. Implementation

The InteKRator toolbox is implemented in plain Java [18]. No additional libraries are needed. The toolbox consists of a single .jar file that can be used as a command line tool, an external process or a Java library in other projects. The InteKRator project is open source, licensed under the GNU Public License Version 3.0 (GPL v3.0). The source code is well documented using Java's documentation tool JavaDoc. InteKRator operates on plain text files for in and output. This renders InteKRator a lightweight and efficient tool that can be easily integrated in web and other applications.

The details of the implementation of the algorithms in InteKRator will go beyond the scope here, but can be found in further literature, e.g., [4] and [9].

InteKRator can be downloaded on its GitLab repository site [12]. There, also further help on how to use the different modules of the toolbox is (which can also be accessed through the command line interface).

5. Lessons learned

InteKRator is an eligible toolbox for quickly setting up an expert system directly from a data set. It thereby contributes to simplify the creation process of an expert system. The resulting knowledge bases can be queried efficiently for ansers and a revision module

allows for incorporating new evidence (which should be done on the bottommost level to satisfy common quality criteria for revision; see Section 3.4).

5.1 Benefits and Potential of the InteKRator toolbox

The benefits and the potential of InteKRator can be summarized as follows:

- Learning a knowledge base that can be queried as an expert system directly from data instead of manual analysis and modeling steps (see Figure 1).
- Comprehensible resulting rule-based knowledge, due to its representation on several levels of abstraction—by this means, the learned knowledge can be read top-down up to an eligible level of abstraction (see Figure 2).
- Easy evaluation of the learned knowledge through InteKRator's built-in check module (see Section 3.3).
- Efficient and transparent inferences that can optionally provide explanations from which rule a conclusion is drawn (see Figure 3).
- Automated incorporation of new evident knowledge (see Section 3.4).
- Simple to use and lightweight integration in applications, like web and other applications (see Section 4).

5.2 Comparison to Existing Solutions and Innovation

An established logic-based solution for representing knowledge in practical applications is *answer set programming* (ASP) [7], which is implemented , e.g., by the well-known solver Clingo [8]. ASP has been used for representing knowledge bases in the medical context (see [19] for a recent work). However, relying on the concept of *default negation* (which is related to *default logic*, e.g., Reiter [15]), it may be hard to understand for non-logicians and may lack a transparent explanation of inference results. It has been studied in [14], that the special knowledge bases used by InteKRator (originating from [4] and [5]) offer an increased comprehensibility to humans compared to ASP. Furthermore, it has also been studied in [14], that inferences can be provided more efficiently in comparison to Clingo. Moreover, to the extent of the authors' knowledge, Clingo does not provide an opportunity to learn answer set programs as knowledge bases automatically from a data set.

A large collection of Java libraries related to AI and logic is TweetyProject [20]. Besides ASP, TweetyProject covers a large amount of further logic-based approaches. In contrast to that, the InteKRator toolbox focuses on simplicity and a lightweight interface, using approaches presented, e.g., in [5], [2], [3] (among others), which TweetyProject currently does not seem to support. These approaches were designed for the comprehensive representation of knowledge learned from data (originally in the context of agents) and can be intuitively read—even by users not having expertise in logic (cf. [14]). Thus, InteKRator could be an eligible choice for applications in multi-disciplinary environments, as it is the case for the medical domain.

As has been shown already in Figure 1, lacking learning capabilities in knowledge representation tools can result in additional work of analyzing the data and modeling the results as rule-based knowledge to set up an expert system. Furthermore, the knowledge represented by logic-based approaches can be difficult to understand, especially for people not having a strong background in logic (see [14] for a comparison of

comprehensibility of the knowledge bases used by InteKRator against answer set programs).

Machine learning solutions perform well for learning relations from a data set (e.g., for classification tasks), but the learned knowledge often has a black box character, since it is represented by a large number of weights (e.g., in case of neural networks). In medical applications, such approaches might be less trustworthy than the explicit rule-based approach used by InteKRator (whose weights relies on conditional probabilities $P(conclusion \mid premise)$, which are in general well understood).

Methods that are able to learn rule-based knowledge from data (as known, e.g., from recommender systems), can be limited to single rules (or a set of rules) only. In contrast to that, InteKRator learns a complete knowledge base and provides an extended toolbox for further tasks that are important in the context of an expert system (like inference, evaluation and revision modules).

6 Conclusion

This paper introduced the InteKRator toolbox as an eligible lightweight instrument to automatically create an expert system from data. InteKRator thereby contributes to simplify the creation process of an expert system by avoiding manual analysis and knowledge modeling steps. The resulting knowledge is represented in a comprehensible way, allows for efficient inferences and can be revised with new evidence.

While such a system has the potential to offer great opportunities for quickly applying results retrieved from data in practice, it should, however, always be used with care: For example, knowledge contained in a data set can be misleading (depending on how the data was collected) and, in principle, every information technology system may be subject to errors or security issues. Thus, expert systems created with the InteKRator toolbox might serve best for supporting human decisions only (especially in critical areas).

Besides the further development of the toolbox, future work may comprise the development of a graphical user interface (GUI) and the implementation in the context of medical applications. First attempts have already shown promising results.

Furthermore, a comparative study regarding other knowledge representation and/or explainable machine learning approaches that allow for learning comprehensive human-readable knowledge from data could be performed in the future. This could help to gain more insights into InteKRator's strengths, especially its performance and comprehensibility properties. First attempts on that have been made in [14], comparing its underlying knowledge representation paradigm and the inference algorithm to ASP and the ASP-solver Clingo [8] (ASP has been considered in the medical domain, e.g., in [19]).

Even if the check module allows already for measuring the quality of a learned expert system, an evaluation of the inferences by a learned expert system in the context of a real-world data set against those of a human expert could also be of interest.

Conflict of Interest

The authors state that they have no conflict of interests.

References

[1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo, Fast Discovery of Association Rules. In: Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.), *Advances in Knowledge Discovery and Data Mining*, 307–328. The MIT Press, Cambridge, Massachusetts (1996).

[2] D. Apeldoorn, A. Dockhorn, Exception-Tolerant Hierarchical Knowledge Bases for Forward Model Learning. IEEE Transaction on Games (early access) (2020). DOI: 10.1109/TG.2020.3008002

[3] D.Apeldoorn, L. Hadidi, T. Panholzer, Learning Behavioral Rules from Multi-Agent Simulations for Optimizing Hospital Processes. In: Chomphuwiset, P., Kim, J., Pawara, P. (eds.), *Multi-disciplinary Trends in Artificial Intelligence – 14th International Conference, MIWAI 2021, Virtual Event, July 2–3, 2021, Proceedings*, 14–26. Springer, Cham (2021). DOI: 10.1007/978-3-030-80253-0_2

[4] D. Apeldoorn, G. Kern-Isberner, When Should Learning Agents Switch to Explicit Knowledge? In: C. Benzmüller, G. Sutcliffe, R. Rojas (eds.), *GCAI 2016. 2nd Global Conference on Artificial Intelligence*. EPiC Series in Computing, vol. 41, 174–186. EasyChair Publications (2016).

[5] D. Apeldoorn, G. Kern-Isberner, Towards an Understanding of What is Learned: Extracting Multi-Abstraction-Level Knowledge from Learning Agents. In: V. Rus, Z. Markov (eds.), *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference*, 764–767. AAAI Press, Palo Alto, California (2017).

[6] C. Beierle, G. Kern-Isberner, Methoden wissensbasierter Systeme – Grundlagen, Algorithmen, Anwendungen (4. Auflage). Vieweg+Teubner, Wiesbaden (2008).

[7] G. Brewka, T. Eiter, M. Truszczyński, Anser Set Programming at a Glance. Commun. ACM, 54(12):92–103, 2011.

[8] Clingo – A Grounder and Solver for Logic Programs: https://github.com/potassco/clingo, accessed on Mar 31st, 2021.

[9] A. Dockhorn, D. Apeldoorn: Forward Model Approximation for General Video Game Learning. In: C. Browne, M. H. M. Winands, J. Liu, M. Preuss (eds.), *Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG'18)*, 425–432, IEEE, Piscataway (2018).

[10] D. Fierens, H. Blockeel (Supervisor), M. Bruynooghe (Supervisor), Learning Directed Probabilistic Logical Models from Relational Data (Het leren van gerichte probabilistisch-logische modellen uit relationele gegevens). PhD Thesis, KU Leuven, Leuven (2008).

[11] S. O. Hansson, Logic of Belief Revision. In: E. N. Zalta (ed.), The Stanford Encyclopedia of Philosophy (2017).

[12] InteKRator Toolbox: https://www.gitlab.com/dapel1/intekrator_toolbox, accessed on Mar 31st, 2021.

[13] I. Kuhn, Heuristische Optimierung durch menschliche Intuition – Das beste aus zwei Welten. In: M. Becker (ed.), *SKILL 2019 – Studierendenkonferenz Informatik*, 97–108, Gesellschaft für Informatik e. V. (2019).

[14] C. Krüger, D. Apeldoorn, G. Kern-Isberner, Comparing Answer Set Programming and Hierarchical Knowledge Bases Regarding Comprehensibility and Reasoning Efficiency in the Context of Agents. In: *Proceedings of the 30th International Workshop on Qualitative Reasoning (QR 2017) at International Joint Conference on Artificial Intelligence (IJCAI 2017) in Melbourne Australia*, Northwestern University, Evanston, Illinois (2017).

[15] R. Reiter, A Logic for Default Reasoning. Artificial Intelligence, 13(1–2):81–132, 1980.

[16] SWI-Prolog: www.swi-prolog.org, accessed on Mar 31st, 2021.

[17] J G Sotos, MYCIN and NEOMYCIN: Two Approaches to Generating Explanations in Rule-Based Expert Systems. Aviat Space Environ Med. 1990 Oct;61(10):950-4.

[18] Sun Microsystems, Java Language Overview. Sun Whitepaper, 1995.

[19] A. Thevapalan, G. Kern-Isberner, D. Howey, C. Beierle, R. Meyer, M. Nietzke, Decision Support Core System for Cancer Therapies Using ASP-HEX. In: K. Brawner, V. Rus (eds.), *Processdings of the Theirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida, USA. May 21-23 2018*, 531–536. AAAI Press (2018).

[20] TweetyProject: https://tweetyproject.org, accessed on Jul 24th, 2021.