Program Structures for Learning in Constraint Handling Rules

Bachelor Thesis Proposal

1 Introduction and Motivation

Generally, we can distinguish between several types of learning. In the field of Computer Science, implementing a program can be seen as 'learning by being told' where a 'teacher', i.e. the programmer, tells the 'learner', here the program, exactly what to do. The opposite of this form is 'learning by discovery' where the learner is on its own and has to gain knowledge by observing the environment and conducting experiments and tests. However, the problem with these two kinds of learning is that the main burden falls completely on either the teacher, as in the first one, or on the learner, as in the latter one. This is probably why the most common form is 'learning by examples', also called 'inductive learning', where the teacher presents appropriate examples to the learner whose task it is to derive generalizations that can explain the given examples. [1]

Referring to the creation of programs, one of the advantages of this learning type is the work distribution between teacher and learner. Because of that, the programmer only has to implement a basic program structure and provide suitable examples allowing the learning program to adapt its behavior accordingly without the need for aimlessly searching for appropriate data on its own.

Besides the above forms of *how* learning can take place, we can further specify *what* we want to learn. According to [2], learning can be divided into four kinds. These are fact learning, skill acquisition, fact strengthening and skill conditioning, where the first two refer to the gaining of knowledge (divided into facts and rules) and the other ones to its adaption or 'tuning'. This separation can be derived from empirical studies and is furthermore embedded in the state-of-the-art cognitive architecture ACT-R[1] which, for example, could assert itself in modeling U-shaped learning in the context of language acquisition (see [3]).

The probabilistic-logic formalism CHRiSM is an examplary case where learning by examples is possible. As described in [4], the formalism combines the declarative, rule-based constraint programming language CHR, i.e. short for Constraint Handling Rules, with the probabilistic logic programming language PRISM. Here, a program is defined by a set of so-called chance rules which, basically, are CHR rules assigned with a certain parameter determining the probability of an applicable rule to fire. Referring to that, learning is given by the adaption of these probability parameters based on observed examples.

Nevertheless, the possibilities of learning in CHR are quite limited which is why research has to be conducted.

[1]Adaptive Control of Thought-Rational

2 Problem Statement

So far, structures for CHR that are capable of learning are given in the extended version CHRiSM, introduced in [4], which combines Constraint Handling Rules with the probabilistic logic programming language PRISM. The interesting point in this context is the language's ability of changing the rules' probability values based on observed examples, which is where the learning takes place. However, responsible for this parameter adapting is the expectation-maximization (EM) learning algorithm being embedded in PRISM. Hence, this kind of learning can not be simply transferred to other CHR systems that use alternative host languages like Prolog for instance. Furthermore, learning only refers to the probability parameter of rules here, although there are more aspects that need to be considered, like fact learning for instance, as already mentioned in the Introduction section 1.

Because of that, there is need for the creation of program structures in pure Constraint Handling Rules that enable learning as defined by the four types mentioned in [2]. Therefore, we take a closer look at these points and their feasibility in CHR:

1. *Fact learning*: Due to the inherent distinction of constraints and rules in CHR, it makes sense to represent facts as such constraints allowing fact learning simply by introducing constraints to the constraint store.

2. *Skill acquisition*: In order to gain new skills modeled as rules, a CHR program should be able to add new rules during runtime. As this only works for constraints, rules have to be represented as such. Hence, a structure is necessary testing and conducting the application of these abstract rules.

3. *Fact strengthening*: This aspect is based on associating facts with activation values influencing their chance of being chosen during computation. In [5], CHR^{rp} is introduced extending CHR by rule priorities, for which the author describes a translation in pure CHR, too. The related report [6] also discusses constraint priorities which can be seen as some kind of activation. Nevertheless, the paper only addresses the selection based on priorities, but not how such an value can be determined.

4. *Skill conditioning*: Similar to the activation of facts, rules possess an utility value. Choosing rules based on such priorities is already implemented in form of CHR^{rp} as mentioned before. An alternative way can be the implementation of conflict resolution being part of the production system translation described in [7]. However, as for fact strengthening, the adaption of the utility value has still to be taken care of.

Summarized, the two main issues that we have to deal with are the abstract rule representation and the adaption of constraint activation and rule utility, as they are not covered in the field of Constraint Handling Rules yet.

Nevertheless, the above aspects only serve as a basic structure for learning, which means that, for example, in order to learn how to solve a specific problem, appropriate strategies have to be represented in form of CHR rules (based on the abstract rule representation mentioned above). Two typically used strategies are *retrieval* and *analogy* which is why they will be considered as well (see e.g. [3] and [8]). Here, the first one is used when the solution to a problem is already known and thus, can be retrieved from the memory, while the second strategy tries to solve a given task by transferring it to a similar, already solved, problem.

3 Purpose of the study

The purpose of this study is the creation of program structures in Constraint Handling Rules that realize a basis for learning in the sense of gaining and tuning knowledge in form of facts and rules. These structures can then be further used to implement models that are capable of adapting their behavior according to given example data.

4 Review of the literature

CHRiSM: Chance Rules induce Statistical Models [4]

Authors: Jon Sneyers, Wannes Meert, and Joost Vennekens
Summary: The probabilistic-logic formalism CHRiSM is introduced, combining CHR with the probabilistic logic programming language PRISM.
Significance: The article shows that learning by examples is possible in an CHR system. However, the learning takes place in form of adapting rule priorities based on the underlying PRISM system. Hence, the language gives rise to find a realization of learning directly in Constraint Handling Rules.

A Rule-Based Implementation of ACT-R Using Constraint Handling Rules [9]

Authors: Daniel Gall
Summary: A way to translate ACT-R models to Constraint Handling Rules is presented.
Significance: As mentioned in section 1, the two basic learning aspects, i.e. gaining and adapting knowledge, that we build our research upon, also play a part in the cognitive architecture ACT-R. In this context, the paper suggests, due to the translation it contains, that it is generally possible to realize the earlier introduced four types of learning in CHR.

How Can the Human Mind Occur in the Physical Universe? (p. 92 - 95) [2]

Authors: John R. Anderson
Summary: On the pages 92 to 95, the 'Varieties of Learning' are presented, describing fact learning, (fact) strengthening, skill acquisition and (skill) conditioning.
Significance: This book excerpt defines the four kinds of learning our study is based on, thus, leading to the necessary program structure for the implementation.

User-definable Rule Priorities for CHR [5]

Authors: Leslie De Koninck, Tom Schrijvers, Bart Demoen
Summary: The CHR extension CHR^{rp} is introduced where the choosing of applicable rules depends on corresponding priority values.
Significance: The presented translation of CHR^{rp} into (original) CHR can be used to realize constraint and rule selection based on activation and utility values.

Constraint Handling Rules [7]

Authors: Thom Frühwirth
Summary: The author presents Constraint Handling Rules as a rule-based constraint programming language with declarative and multiset transformation properties.
Significance: This book serves as basis for the implementation in CHR. Furthermore, it delivers a realization of conflict resolution which is generally used to choose from various possibilities, thus, relating to constraint activation and rule utility.

An Integrated Theory of the Mind [10]

Authors: John R. Anderson, Daniel Bothell, Michael D. Byrne, Scott Douglass, Christian Lebiere, Yulin Qin
Summary: The article describes ACT-R with focus on the perceptual-motor, the goal and the declarative memory module. In this context, computation steps are demonstrated including the selection of production rules based on subsymbolic processes.
Significance: Definitions of how activation and utility values can be determined are given. Even though the paper refers to the complex cognitive architecture ACT-R, the composition of activation and utility values could be transferred to the intended implementation in CHR.

Why do children learn to say “Broke”? A model of learning the past tense without feedback [3]

Authors: Niels A. Taatgen, John R. Anderson
Summary: An ACT-R model is presented, demonstrating the phenomenon of U-shaped learning in the acquisition of the English past tense.
Significance: The model is an example of how learning based on the separation of gaining and adapting knowledge can take place. Related to this, the strategies ’retrieval’ and ’analogy’ are applied, which can serve as an orientation for their implementation in CHR.

5 Research questions and/or Hypotheses

The general question of this research is how we can enable learning in Constraint Handling Rules. With reference to the definition of learning in [2], the following similar questions emerge:

1. How can *fact learning* be realized in CHR?
   a) How should facts be represented?
   b) Can facts be forgotten/removed?
2. How can *skill acquisition* be realized in CHR?
   a) How should skills be represented?
   b) Can skills consist of more than one rule?
3. How can *fact strengthening* be realized in CHR?
   a) What defines the activation/strength of a fact?
   b) How can the activation be altered appropriately?
   c) How can we choose between different facts based on their activation?
4. How can *skill conditioning* be realized in CHR?
   a) What defines the utility of a skill?
   b) How can the utility value be altered appropriately?
   c) How can we choose between different skills based on their utility?
5. How can these aspects be implemented united in one program?

6 The Design - Methods and Procedures

Some ideas for the research's design are already given in section 2 where the feasibility of the four learning types is discussed. In the following, we extend these presented thoughts.

Fact learning: As mentioned before, facts can be modeled as constraints, thus, realizing the learning of new facts simply by adding the related constraints to the constraint store. As this process is already part of CHR, further work could include to take a look at alternative representation forms based on connections like chunks in ACT-R for example [10].

Skill acquisition: Skills are represented in form of rules, which means that acquiring new skills is equivalent to the introduction of new rules. Therefore, they have to be wrapped in constraints enabling their adding during runtime. In order to realize such an abstract representation, a rule constraint could be of the form *r(Hk,Hr,G,B)* where *Hk* stands for the kept head constraints, *Hr* for the removed head constraints, *G* for the guard and *B* for the body of the rule. Furthermore, constraints are also wrapped in a constraint *c(X)*, where X represents the original constraint, simplifying the matching between the constraints in a rule r(Hk,Hr,G,B) and those in the constraint store.

Fact strengthening: For this part, the constraints are extended by an argument used for its activation. Whenever a constraint appears in a successful rule application, this value is adjusted. In order to determine how such an activation value can be constructed, we take a look at [10] for orientation. Besides, the selection based on activation has to be considered as well, which is where constraint priorities described in [6] and conflict resolution as explained in [7] are regarded in more detail.

Skill conditioning: This topic is quite similar to fact strengthening, hence, the approach will be similar. Especially the article [11] serves as a good foundation as it describes among others how the utility of productions in ACT-R is built-up.

Even though there are detailed definitions of activation and utility values presented in [10], we intend to find a simpler composition to reduce complexity. Further in the context of these subsymbolic parameters, methods for their alteration must be found. A possible way to do this could be based on 'rewards' that occur in reinforcement learning (see e.g. [12]). There, they serve as feedback allowing the learner to adapt its behavior which basically is what we need for the constraint activation and rule utility values.

Besides, although the knowledge in our CHR system is based on a dual representation defined by the distinction of facts and rules, it may be worth to take a look at artificial neural networks as well. In these networks, learning is achieved by the adaption of weights (referring to connections between nodes), usually based on the comparison of intended and actually computed result [13].

In the actual creation of the CHR program, skill acquisition is implemented at first. Afterwards, the aspects strengthening and conditioning are dealt with, building upon the abstract rule representation to unite all four learning types in one program structure.

The strategies *retrieval* and *analogy* can be found in [3], therefore, the paper can give a basic description of how they can look like. In [8], a general definition of analogy is given which can be used as a basis for the implementation in CHR. Because of the usage of an abstract rule representation, both strategies have to be realized as such rules as well, enabling their assessment based on the utility value.

7 Limitations and Delimitations

The research does not intend to realize learning in an highly accurate cognitive way taking into account how the brain concretely works. Instead, the focus lies on enabling learning in CHR generally.

In the context of skill acquisition and by talking about analogy as learning strategy, one would probably expect that production compilation, as described in [14], will be part of this study, too. However, we will not go into detail, focusing mainly on the basic structure given by the four learning aspects illustrated in the problem statement in section 2.

8 Significance of the study

Implementing structures being capable of learning extends the application area of Constraint Handling Rules by enabling self-adapting programs. Here, the learning not only refers to the modification of rule priorities, but to constraint priorities, too. Furthermore, the abstract representation for both constraints and rules allows the introduction of the latter during runtime, thus, giving rise to new possibilities for CHR programming in general.

On the other side, properties of CHR, like the abstract execution by constraints, contribute to a simpler analysis of learning models. For a confluent program structure, concurrent execution would be possible as well (see [7]).

9 Planning

9.1 Own Background

Due to the lectures 'Constraint Programming' and 'Rule-based Programming', the language CHR and topics like conflict resolution are well known.

9.2 Work packages

This small section gives a general overview of the work that will be done each month. However, it should be more regarded as an orientation.

M1 Design development for abstract rule representation, constraint activation and rule utility.

M2 Implementation of abstract rule representation combined with activation/utility and development of strategies 'retrieval' and 'analogy'. Furthermore, defining structure for paper.

M3 Testing/analysis of complete implementation with both strategies and writing of paper.

M4/M5 Continuing of writing and refinement of implementation.

M6 Finishing writing process.

References

[1] I. Bratko, *Prolog Programming for Artificial Intelligence.* Pearson Education, 2001.

[2] J. R. Anderson, *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press, 2007, pp. 92–95.

[3] N. A. Taatgen and J. R. Anderson, "Why do children learn to say "Broke"? A model of learning the past tense without feedback," *Cognition 86*, pp. 123–155, 2002.

[4] J. Sneyers, W. Meert, and J. Vennekens, "CHRiSM: Chance rules induce statistical models," in *Proceedings of the Sixth International Workshop on Constraint Handling Rules (CHR'09)*, 2009, pp. 62–76.

[5] L. De Koninck, T. Schrijvers, and B. Demoen, “User-definable rule priorities for CHR,” in *Proceedings of the 9th ACM SIGPLAN international conference on Principles and practice of declarative programming*, ACM, 2007, pp. 25–36.

[6] ——, “CHR-rp: Constraint Handling Rules with rule priorities,” Department of Computer Science, K.U.Leuven, Report CW479, 2007.

[7] T. Frühwirth, *Constraint Handling Rules.* Cambridge University Press, 2009.

[8] R. J. Anderson and R. Thompson, “Use of analogy in a production system architecture,” in *Similarity and Analogical Reasoning.* Cambridge University Press, 1989, pp. 267–297. [Online]. Available: `http://act-r.psy.cmu.edu/wordpress/wp-content/uploads/2012/12/115UseAnalogy.pdf` (visited on 05/12/2018).

[9] D. Gall, “A Rule-Based Implementation of ACT-R Using Constraint Handling Rules,” Master’s thesis, University Ulm, Faculty of Engineering and Computer Science, Institute of Software Engineering and Compiler Construction, 2013.

[10] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin, “An Integrated Theory of the Mind,” *Psychological review*, vol. 111, no. 4, p. 1036, 2004.

[11] J. R. Anderson and C. Schunn, “Implications of the act-r learning theory: No magic bullets,” *Advances in instructional psychology, Educational design and cognitive science*, pp. 1–33, 2000.

[12] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995, 580ff.

[13] I. A. Basheer and M. Hajmeer, “Artificial neural networks: Fundamentals, computing, design, and application,” *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[14] N. A. Taatgen and F. J. Lee, “Production Compilation: A Simple Mechanism to Model Complex Skill Acquisition,” *Human Factors*, vol. 45, no. 1, pp. 61–76, 2003.