# (Semi-)Automatic Translation of Legal Regulations to Formal Representations: Expanding the Horizon of EDA Applications

Oliver Keszocze[1,2] Betina Keiner[3] Matthias Richter[3] Gottfried Antpöhler[4] Robert Wille[1,2]
[1] Cyber-Physical Systems, DFKI GmbH, Bremen, Germany
[2] Department of Mathematics and Computer Science, University of Bremen, Germany
[3] gradient.Systemintegration GmbH, Singen, Germany
[4] Kassenärztliche Vereinigung Bremen, Bremen, Germany
{keszocze,rwille}@informatik.uni-bremen.de [firstname.lastname]@gradient.de g.antpoehler@kvhb.de

***Abstract*—Caused by the challenges in the design of today's hardware and software systems, tools for *Electronic Design Automation* (EDA) became impressively powerful. However, these accomplishments can also be exploited in other domains. In fact, the steps of formalizing and checking legal regulations shares many similarities with established EDA design steps. In this work, this is demonstrated by proposing the application of EDA tools in the domain of law processing. We propose a (semi-)automatic translation of real rules and regulations into a formal representation. Afterwards, we discuss how – similar to the hardware/software design – these formalization can be utilized in the respective domain.**

## I. Introduction

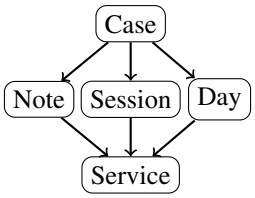
The ever increasing complexity of hardware and software systems lead to the development of elaborated design flows for *Electronic Design Automation* (EDA). Within these flows, how to check whether the system works as intended gains more and more relevance. *Requirement engineering* [1] and design at the *Formal Specification Level* [2] exploiting languages such as *UML* [3] or *OCL* [4] provide proper solutions for this purpose. Based on the initially given (textual) specification, engineers use these solutions to (formally) design and, eventually, verify the desired system to be implemented. This led to the availability of very efficient EDA tools (see e.g. [5], [6], [7]). However, the application of these tools is not necessarily bounded to the design of hardware/software systems.

In fact, *legal regulations* can also be seen as a special kind of specification. Although they do not directly specify the functionality of a system to be developed, many computer applications heavily rely on them. In fact, (software) systems exist which are "in charge" of checking whether e.g. tax returns, accounting and billing, stock market transactions, etc. are processed in line with the rules and regulations of the respective field and country. Hence, as in hardware/software design, a first (manual) design step in the development of such systems is to formalize these regulations. Considering that legal rules and laws are usually provided in a structured and precise (albeit not always comprehensive) fashion motivates the use of natural language processing for their formalization.

In this work, we present a methodology which (semi-)automatically formalizes given legal regulations (provided as real-world wording of the law) into a formal representation. As case study, we consider rules from the *German Regulations on Scales of Fees for Medical Doctors*. We show, how the proposed approach and the resulting formal representation can be used as blueprint to incorporate respective checks into an existing software system. Moreover, the obtained formal descriptions of the considered legal regulations even allow to check whether the law is consistent by itself. This does not only advance the development of respective systems, but also shows directions how EDA tools can be utilized to make better and non-contradictory laws in general.

## II. Considered Domain and Problem Formulation

In this work, the translation of legal regulations to a formal representation is considered and discussed by considering rules from the *German Regulations on Scales of Fees for Medical Doctors* [8] provided in terms of a so-called *Uniform Assessment Standard* (German: *Einheitlicher Bewertungsmaßstab*; in the following: EBM). These regulations specify how medical doctors in Germany are supposed to generate his/her invoices. For this purpose, all possible services are listed and structured by means of *cases*. Each case is composed of *notes* and spans over one or more *sessions* which are conducted in one or more *days*. Eventually, one or more *services* are conducted in each case at the respective sessions. This structure is required since some services can only be accounted once (independently of how many sessions were required) or solely for each session. The EBM eventually defines which services can be accounted and to what amount.




(a) Model of the domain

"The service 45678 is not to be billed together with EBM position 54687 within the third and fourth quarter of 2014."

(b) A EBM rule

```
if (
 Case.quarter >= new Quarter(32014) &&
 Case.quarter <= new Quarter(42014)
) {
 if (together(45678,54687)) {
  error("45678,54687 not in combination")
 }
}
```

(c) A DSL condition checking for the EBM rule

Fig. 1: Considered domain and problem

**Example 1.** *Fig. 1a sketches the structure of a case in terms of an abstract UML model. At the same time, Fig. 1b provides an example of an EBM regulation*[1]*. More precisely, this regulation states that the service assigned the ID 45678 cannot be accounted together with the service assigned the ID 54687.*

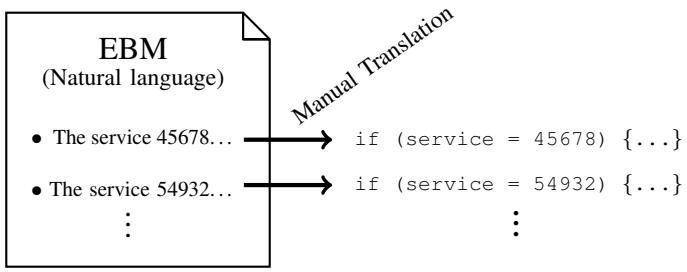
Medical doctors, hospitals, etc. prepare their invoice according to these regulations. However, before the submitted invoices are indeed payed, they are usually checked for correctness and consistency with respect to the EBM. Due to the large amount of regulations, this usually is performed by means of automatic checkers, i.e. the submitted bills are checked by tools. These tools adopt the structure sketched above and in Fig. 1a in terms of data-structures and databases, storing cases as well as their respective notes, sessions, dates, and services. Having initialized all cases submitted by a medical doctor, the initialized data is checked against all EBM regulations. This, however, requires to have all the EBM regulations available in a formal description which can be used in the tool's programming language. For this purpose, usually a *Domain Specific Language* (DSL) is provided.

**Example 2.** *Consider again the EBM regulation shown in Fig. 1b. In order to automatically process this, a DSL-statement as shown in Fig. 1c (and using the data-structure from Fig. 1a) is required.*

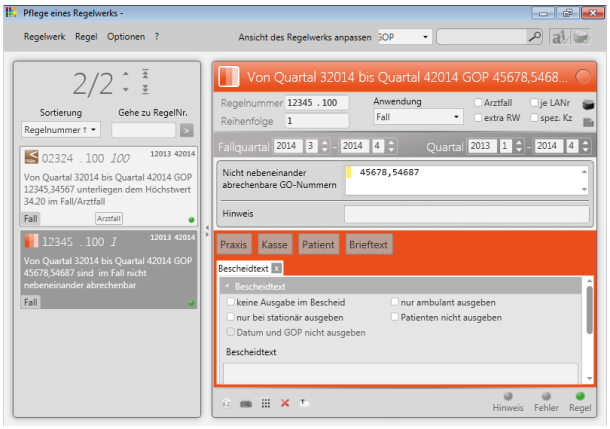
## III. Existing Solution

Obviously, translating these regulations into a formal representation, such as a DSL, is a crucial step within the development of the respective tools. Thus far, the existing design flow heavily relies on manual interaction.

[1]Note that all regulations in the EBM are originally provided in German language. However, in order to describe the approach, all examples discussed in this paper have been translated.




(a) Design flow




(b) Screenshot of a template

Fig. 2: Existing solution

More precisely, a design flow as sketched in Fig. 2a is applied. The original EBM is assumed to be available in terms of a structured document separately listing all rules and regulations (denoted as *EBM rules*). This structure and, by this, the respective EBM rules can easily be generated from the articles and sections of the originally given EBM wording of the law. Using this separation, each single rule can be processed separately.

The actual translation of each EBM rule into a formal DSL representation is then conducted manually. In order to aid this, so called *templates* are utilized. They represent frequently occurring cases and provide some guidance for the user. These guidances are e.g. tooltips pinpointing the user to incorrectly spelled EBM rules. For the considered EBM regulations, a total of 18 templates are provided. If the respectively considered regulation does not fit to any of these templates, additionally an DSL editor with a context assistant is provided. Fig. 2b exemplarily shows a template capturing the case of the regulation shown in Fig. 1b.

Note that this process has to be conducted on a regular basis. Since legal regulations such as the EBM are subject to constant changes, newly added or revised rules have to be formalized frequently. While the backend of the respective tools such as data-structures remains rather stable, (re-)formalizing the regulations has become the major bottleneck in the development of such tools.

## IV. Proposed Solution

In order to ease the process sketched in the previous section, we propose solutions aiming for an automation of the translation from natural language legal regulations (here: EBM rules) into a formal description (here: DSL descriptions). For this purpose, two complementary directions are investigated:

- *Exploitation of Regular Expressions*
  Here, the EBM rules are distinguished by their respective structure. Then, rules with similar and/or very regular structures are automatically processed by means of regular expressions from which the desired formal description is derived.
- *Exploitation of Natural Language Processing*
  Here, existing methods from the domain of natural language processing such as *typed dependencies* [9] are applied to extract information from given EBM rules. In contrast to regular expressions, this technique is not just syntactical but employs meta-information of the respective sentences.

In the following, both schemes are described in more detail in Section IV-B and Section IV-C, respectively. Before, a preprocessing step is introduced which normalizes the given EBM rules and, hence, improves the impact of both schemes.

### A. Normalizing EBM Rules

The considered regulations use a wide range of different terms which basically describe the same issues. As an example, recall the sentence from Fig. 1b. The words "service" and "EBM position" are synonymous, i.e. they refer to the same entity in the domain: a service. When (automatically) translating such rules, it is essential that, whenever possible, a consistent terminology is applied. In order to achieve this, a normalization is conducted prior to the actual translation.

For this purpose, a knowledge database is applied in order to replace all synonyms by a uniform representative. This knowledge database has to be initially filled by an expert that knows and understands both, the domain of interest as well as the wording of the law. While building up the database still requires manual interaction, the result can heavily be re-used in all upcoming (and possibly future) formalization processes. Moreover, the database also allows for further simplifications of the considered sentences. For example, parts of sentences often include recurring natural language descriptions of particular services. Using the database, they can also be normalized to their corresponding EBM position.

### B. Exploitation of Regular Expressions

The sentences describing the EBM rules are very structured and in a certain way repetitive. This is caused by the fact that many regulations are structurally identical and just need to be applied in different contexts. For example, consider the rule from Fig. 1b inheriting the structure "The service <ID> is not to be billed together with services <ID, ID, ...> within <timeframe>" with "ID" referring to a particular EBM position. Such rules frequently occur in the EBM just with different EBM positions and/or timeframes.

Identifying such structures allow for and easy an powerful formalization procedure. In fact, once a formalization for such a sentence is available, it can be re-used in order to determine formal representations of structurally identical sentences. Regular expressions are an obvious tool for this purpose[2]. For example, the regular expression determining enumerations of EBM positions is given by:

```
ID =[0-9]{5}[A-Z]?
enum=<ID>(,\s+<ID>|(
    (,| as well as)\s+<ID>)?
    \s*(and|to)\s*<ID>)*
```

This, as well as similar regular expressions for further issues such as timeframes, can be utilized to (automatically) process all further sentences composed of such a structure. More precisely, if a rule such as shown in Fig. 1b is translated to a formal representation for the first time (e.g. by manual interaction), structurally similar rules can then automatically be translated and adjusted e.g. by EBM positions and/or timeframes.

While this scheme works very well for a large fraction of EBM rules (as discussed later in Section V), of course it is limited by the structural properties of the natural language sentences. As soon as variances occur which cannot efficiently be expressed in terms of regular expressions, this scheme does not lead to the desired results anymore. To cope with this, an alternative approach is introduced in the next section.

### C. Exploitation of Natural Language Processing

Besides pure structural information, also meta-information e.g. obtained by grammatical analysis can be utilized when translating natural language rules into a formal representation. For this purpose, established techniques from the domain of natural language processing such as *typed dependencies* [9] can be employed. Typed dependencies create a relationship between words of a sentence and, furthermore, type them with respect to their grammatical relation. For example the words "the" and "service" have a *determiner* (short: *det*) dependency to each other. All typed dependencies of a sentence eventually form a graph based on which a translation into a formal representation can automatically be conducted in many cases.

In the following, the general idea of this approach is illustrated again by means of the example from Fig. 1b. The corresponding typed dependency graph of the original German sentence, obtained using the tool ParZu [10], is shown in

[2]Alternatively, also *boilerplates* as e.g. considered in [1] can be applied.

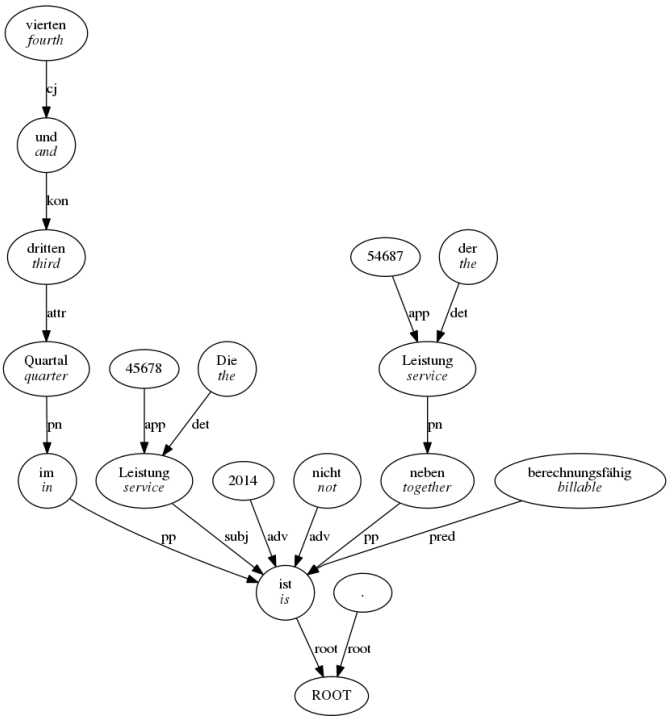



Fig. 3: Typed dependencies of the sentence of Fig. 1b

Fig. 3[3]. To ease the understanding of the graph, an English translation of each word is additionally provided in *italic*. Note that this rule has already been normalized as described in Section IV-A.

Using this information, first a root node is defined that will serve as the starting point for all further investigations. This node is defined by the node which (1) has a *root* dependency to the node labeled "ROOT" and (2) is not labeled with a puncutation mark. Hence, in the considered example, we consider the root node labeled "ist/*is*" as the root node (see Fig. 3). Then, the following information is gathered:

1) The subject (*subj*) of the sentence is "Leistung"/*service*. This becomes the reference object in the model of the domain. Dependencies of type *app* refer to the precise EBM position of the service, i.e. 45678 in the considered example.
2) In order to determine further EBM positions that are referred to in the sentence, *pn-app* paths are checked. In the running example, this gives the excluded EBM position 54687. Enumerations of EBM positions result in a long path of *kon*- and *cj*-type dependencies after the initial *app*-typed dependency.
3) The context of the rule, i.e. the timeframe in which the regulation is to be applied, is "the third and fourth quarter of 2014". This is examined from the *pp*- and *adv*-typed dependencies (in many cases, the *pp*-typed dependencies are sufficient). If no context could be determined, it is assumed that the respective session is applied. This assumption originated from background information provided by an expert in the field who stated that, if no context is explicitly mentioned, this is indeed the default case.
4) Negations in a sentence, e.g. two services are *not* to be billed together, are usually determined by examining the *adv*-typed dependencies of the root node.

All this information can then be compiled together into a corresponding formal representation. More precisely, the EBM position *45678* (Information 1) stands in a relation to the EBM position 54687 (Information 2), i.e. `if (together(45678,54687))`, iff the third and fourth quarter of 2014 is considered (Information 3), i.e.

```
if (   Case.quarter >= new Quarter(32014)
    && Case.quarter <= new Quarter(42014))
```

The negation (Information 4) indicates that this is a forbidden scenario, i.e. should lead to an error. This eventually can be represented to a formal DSL description similar to the one shown in Fig. 1c.

In a similar fashion, other EBM rules can be processed. While this obviously cannot cover all possible means, it constitutes an improvement to the rather simple scheme introduced in Section IV-B. In addition to this generic NLP approach, specialized schemes for classes of sentences have been developed. We use regular expressions to decide which class a sentence belongs to and, then, use the respective scheme to extract the formal description. Note that even though the regular expressions are able to determine the class of the sentence, they were not fully capable of correctly extracting the information.

## V. Evaluation

The approaches introduced above have been implemented in Java and evaluated by considering a recent version of the EBM. In total, 2932 rules have been processed from which

- a total of 958 rules (i.e. approx. 33%) could have been translated into a formal representation using the scheme described in Section IV-B based on regular expressions,
- a total of 1929 rules (i.e. approx. 66%) could have been translated into a formal representation using the scheme described in Section IV-C based on techniques for natural language processing, and
- a total of 1003 rules (i.e. approx. 34%) could not be translated by either of these approaches.

[3]The dependency graph of the English translation differs from its "German counterpart" and, hence, would require an adjustment of the following steps. Although such adjustments would be moderate, an exhaustive evaluation was not possible due to the missing translation of the entire EBM. Hence, we continue the description of the scheme using the original German EBM rule.

This evaluation shows that already the rather simple approach based on regular expressions leads to some very satisfying results. Those can be further be improved by more sophisticated analyses and eventually lead to a subset of two Third of all rules for which a formal representation can automatically be translated. Considering that, thus far, *all* rules have to be manually translated this is a considerable improvement.

Note that the quality of the results significantly depends on the initial information e.g. provided by experts in terms of knowledge databases or by determined the applied regular expressions. However, this has only to be provided once and, afterwards, can be re-used for the remaining process as well as frequently occurring future updates. Furthermore, note that the correctness of the generated formal representations is not automatically checked yet. The proposed flow still requires a user which manually approves the determined results. But still, this check can be done significantly faster than determining the entire formal description from scratch.

## VI. Conclusion and Future Work

In this work, we introduced approaches aiming for the automatic translation of legal regulations into a formal description. For this purpose, we considered rules from the *German Regulations on Scales of Fees for Medical Doctors* provided in terms of the EBM. Our solutions exploited structural information (employed by regular expressions) as well as grammatical information (employed by type dependencies). Our evaluations showed that, using the proposed schemes, up to 66% of the considered rules can automatically be translated.

By this, we demonstrated how EDA tools can be applied in the domain of law processing. While, in this paper, we mainly focused on the generation of formal checking conditions for the EBM, manifold further applications exists in which the formal representation of legal regulations can be utilized. For example:

- The formal representation can be applied to optimize the billing data of medical doctors in order to achieve the maximum outcome – usually with respect to the charged money.
- A formal representation of rules can be used to check whether the model (and, by this, the legal regulations themselves) is consistent, i.e. free of any contradicting rules. This may help lawmakers to "verify" the soundness of legal regulations at least up to a certain amount.
- In a similar fashion, one could look for corner cases that are allowed by the given regulations but, from a political, social, or other perspective, should be prohibited. Also this information can be used e.g. by lawmakers in the process of creating new regulations as it easily allows you to check the impact of new rules.
- The formal representation allows to check out for loopholes in the regulations – either to exploit or to fix them.
- Finally, these applications can also be used in other domains that regulate how things are to be paid. A good example for this would be tax regulations. In fact, software that aids tax payers hand in their annual tax declaration and optimizing it already is a big market.

Future work will be devoted to address these further directions in detail.

## Acknowledgements

This work was supported by the German Federal Ministry for Economic Affairs and Energy (BMWi) under grant no. KF2054902MS2 and KF2013014MS2 as well as the German Research Foundation (DFG) under grant no. WI 3401/5-1. We would like to thank Lucjan Suchy for many fruitful discussions.

## References

[1] M. E. C. Hull, K. Jackson, and J. Dick, *Requirements Engineering, Second Edition*. Springer, 2005.

[2] R. Drechsler, M. Soeken, and R. Wille, "Formal specification level: Towards verification-driven design based on natural language processing," in *Forum on Specification & Design Languages*, 2012.

[3] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language reference manual*. Essex, UK: Addison-Wesley Longman, Jan. 1999.

[4] J. Warmer and A. Kleppe, *The Object Constraint Language: Precise modeling with UML*. Boston, MA, USA: Addison-Wesley Longman, Mar. 1999.

[5] B. Beckert, R. Hähnle, and P. Schmitt, *Verification of Object-Oriented Software: The KeY Approach*. Secaucus, NJ, USA: Springer, Oct. 2007.

[6] K. Anastasakis, B. Bordbar, G. Georg, and I. Ray, "UML2Alloy: A Challenging Model Transformation," in *Int'l Conf. on Model Driven Engineering Languages and Systems*. Springer, Oct. 2007, pp. 436–450.

[7] M. Soeken, R. Wille, and R. Drechsler, "Encoding OCL Data Types for SAT-Based Verification of UML/OCL Models," in *Tests and Proofs*, ser. Lecture Notes in Computer Science, vol. 6706. Springer, Jun. 2011, pp. 152–170.

[8] "Gebührenordnung für Ärzte (GOÄ)," Online available at http://www.e-bis.de/goae/defaultFrame.htm.

[9] M.-C. de Marneffe, B. MacCartney, and C. D. Manning, "Generating Typed Dependency Parses from Phrase Structure Parses," in *Conf. on Language Resources and Evaluation*, 2006, pp. 449–454.

[10] R. Sennrich, G. Schneider, M. Volk, and M. Warin, "A new hybrid dependency parser for German," in *Proceedings of the German Society for Computational Linguistics and Language Technology*, 2009, pp. 115–124.