August 2018

A Controlled Natural Language for Financial Services Compliance Checking[1]

Shaun AZZOPARDI [2], Christian COLOMBO and Gordon J. PACE
Department of Computer Science, University of Malta, Malta

Abstract. Controlled natural languages have long been used as a surface form for formal descriptions, allowing easy transitioning between natural language specifications and implementable specifications. In this paper we motivate the use of a controlled natural language in the representation and verification of financial services regulations. The verification context is that of payment applications that come with a model of their promised behaviour and which are deployed on a payments ecosystem. The semantics of this financial services regulations controlled natural language (FSRCNL) can produce compliance checks that analyse both the promised model and/or monitor the application itself after it is deployed.

Keywords. financial regulations, controlled natural language, compliance checking, regulation formalisation

1. Introduction

Financial services exist under a highly regulated legal regime, given the high risk of disruptive behaviour such as money laundering. Financial institutions usually have dedicated compliance departments that aim to reduce legal liability by ensuring legal compliance, but ensuring compliance is difficult and thus results in substantial *a priori* investment. Particularly, this need for compliance may discourage such institutions to act as service providers to financial programmes (e.g. by performing transactions as directed by payment applications), especially when they are developed and managed by newcomers to the field, given the element of risk involved.

Automated methods to compliance checking could partially circumvent this problem, allowing for confidence that payment applications are behaving correctly. In the context of compliance of applications one can make use of a multitude of already existing analysis techniques to enforce compliance. However, a particular problem with such an approach is that there is a gap between the formal specification languages of existing analysis tools and the language of the regulations to be checked or enforced. Moreover, legal experts cannot be expected to understand the former, making such specifications not amenable to validation by domain experts.

[1]This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant number 666363.

[2]Corresponding Author: Shaun Azzopardi, Department of Computer Science, University of Malta, Msida, Malta; E-mail: shaun.azzopard@um.edu.mt.

In this paper we discuss our experience in tackling this problem by developing a controlled natural language (CNL) to allow the specification of legal compliance checks in the context of financial services regulations. The *Financial Services Regulation Controlled Natural Language* (FSRCNL) is a subset of the English language, allowing for easy uptake, while it has an unambiguous semantics. We developed this CNL in the context of an industrial project, the *Open Payments Ecosystem* (OPE) [4], and used it mainly to encode the Gibraltar and UK transpositions of the Payment Services, the E-Money, and the Anti-Money Laundering EU Directives. A novel feature of this CNL is that it consists of two sublanguages which are given semantics in different logics. Although not visible from the point of view of the user, since the structure of the language is uniform, the use of different logics allows the specification of regulations that are verifiable at different phases of a payment application's lifetime.

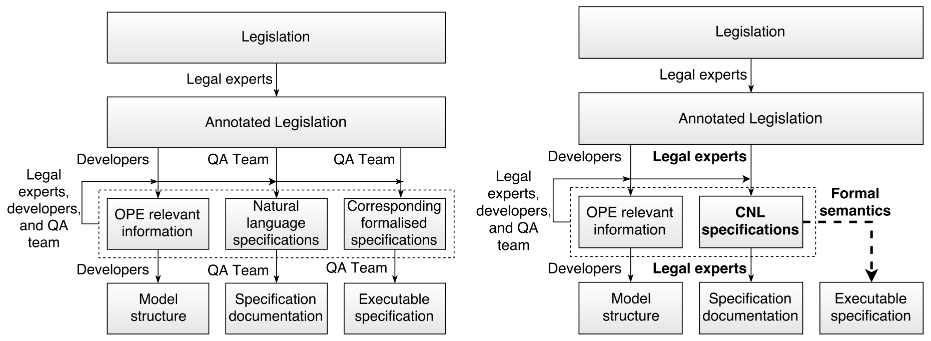
In this paper we detail our experience with this CNL, starting in Section 2 with an overview of the iterative process taken with experts to elicit the meaning and the language of the regulations. We then discuss FSRCNL by giving an overview of its grammar in Section 3 and the corresponding formalisation in Section 4. In Section 5 we discuss qualitatively the CNL and the lessons learnt from its use. In Section 6 we consider related work, before concluding in Section 7.

2. Analysing Financial Services Regulations

Identifying the relevant pieces of legislation which regulate a given system and formalising them is a challenging process. In particular, doing this manually can be prohibitively expensive, given their length and density. General approaches exist to aid such a process by automating the translation of natural language legal documents into a formal representation e.g. [17,5], with varying success. In the narrower scope of building a compliance checking engine for a particular system and domain, this problem is less acute, even though the interaction between the legal and technical experts still requires substantial effort. In our context, the compliance engine had to be incorporated within a general framework (the Open Payments Ecosystem, or OPE) to be used by developers to construct financial applications, which led to two major challenges: (i) the compliance engine had to be developed to work for any system communicating with the OPE; (ii) the legislation is different from one country to another, and changes on a regular basis, and thus a solution which supports easier update and transposition was required.

2.1. Identifying Automatically Verifiable Regulations

One of the major challenges in using automated analysis of legal texts is that the domain covered by legislation is much wider than the one for which compliance systems usually work — also true when restricting oneself to relevant legislative chapters e.g. the OPE does not implement all the payment services regulated by legislation. This means that pruning of legislation to relevant parts is a crucial first step. We thus undertook a manual process of requirements elicitation that exploited legal experts' intimate knowledge of the regulations. A law firm was consulted, were we had direct contact with three lawyers, but with other lawyers being consulted in the process from within the firm. This process started with lawyers identifying which clauses in the legislation were relevant to the business



(a) The formalisation process without FSRCNL. (b) The formalisation process with FSRCNL.

Figure 1. Alternate formalisation processes.

process of the system. Some of these clauses were however found not to be verifiable in an automatic manner. For instance, consider our challenge of incorporating a compliance engine within a library for building financial applications. A regulation may require that relevant terms and conditions should be presented to the financial application user before any transactions are carried out. However this cannot be easily verified since (i) displaying of text is performed outside the scope of the library; and (ii) ensuring that the (free) text of the terms and conditions makes sense is not possible with current natural language technology. Such clauses were thus deemed unverifiable by the compliance engine even though directly relevant to the domain. Thus, the lawyer-identified *relevant clauses* were further filtered by the system's developers to acquire a smaller set of *verifiable clauses*.

In order to support the communication between the teams involved, it was realised that we needed to be able to maintain versions of these automatically verifiable regulations in three formats (other than their representation in the regulations): informally (for the lawyers to ensure that the interpretation is in compliance with the actual law), formally (for the quality assurance team to ensure the meaning of the informal representation), and using an executable representation (to be able to verify the formal regulations). However, developing three versions of the specification in parallel (as illustrated in Figure 1a.) would require an inordinate amount of replication of work. Furthermore, if these parallel versions are developed separately, misunderstandings and ambiguities in interpretation of terms between legal experts and developers could arise. For instance, the rule stating that *"Only prepaid instruments can be used with e-money"* was misunderstood by developers, since they associated a different meaning with the word *instrument* than that meant by lawyers. This could easily have led to a mismatch between what should have been checked and what was actually checked. Iterative meetings between the different teams uncovered such discrepancies.

To avoid such problems in the future we opted to construct a CNL with a semantics that is more aligned to what a legal expert expects. It can be used to specify regulations with little prior knowledge of the system and allows executable specifications to be automatically generated. Using this CNL as the primary source of compliance checks places the duty of specification back on the actual domain expert, the legal experts. The process aided with the use of the CNL, is illustrated in Figure 1b.

Regulation	**Acronym**	**Verifiable Clauses**
The Electronic Money Regulations 2011 (SI 2011/99)	EMR	11
The Payment Services Regulations 2009 (SI 2009/209)	PSR	14
The Money Laundering Regulations 2009 (SI 2009/209)	MLR	4
Fourth Money Laundering Directive (EU) 2015/849	MLD4	0
European Commission's Proposal for a Directive Amending MLD4	MLD5	2

Table 1. List of UK regulations considered.

2.2. The Language of the Regulations

Payment services are regulated tightly by the EU, with several relevant directives being in force. The OPE is planned to be initially deployed in the UK but later to be extended to cover other countries. We thus limited ourselves to the UK-specific implementations of these directives, but covering also minor differences with the legislation of Gibraltar. Table 1 illustrates the main UK regulations considered (and directives that were not yet transposed at the time), their associated acronym as used here, and the amount of clauses identified as verifiable (and specified using our CNL) from each of them. These regulations cover programmes that perform some form of payment service (e.g. issuing payment cards and other payment instruments), with each of these services performed by a certain service provider licensed in an appropriate country. In this section we consider the language of these clauses through examples.

Regulatory documents are normative documents, specifying what regulated entities can and cannot do, starting with definitions of the domain-specific jargon, for example:[3]

EMR2(1) *"electronic money" means electronically (including magnetically) stored monetary value as represented by a claim on the electronic money issuer which (a) is issued on receipt of funds for the purpose of making payment transactions; [...]*

These terms are then used to specify obligations and prohibitions restricting behaviour:

EMR45 *An electronic money issuer must not award (a) interest in respect of the holding of electronic money; or (b) any other benefit related to the length of time during which an electronic money holder holds electronic money.*

These kind of clauses identify the responsible entity (the issuer), the kind of norm (a prohibition), and the relevant action or actions (awarding interest). To be able to enforce this we need to at least be able to represent the forbidden behaviour or state. Syntax from deontic logics can be used to model these norms e.g. using F to denote a prohibition, one can write $F_{issuer}(awardInterest)$. A regulation may also be conditioned on some limits holding, both with respect to some time-frame or a monetary value:

ML13(7)(d)(ii) *[...] if the device can be recharged, a limit of 2,500 euro is imposed on the total amount transacted in a calendar year, except when an amount of 1,000 euro or more is redeemed in the same calendar year by the bearer [...]*

In summary, a representation to encode these regulations must then allow for specifying what should take place, and negation to specify what should not. Articles in the legislation tend to be limited to particular situations and contexts, motivating the need for a way to represent the conditions necessary for a certain state to hold. Also, the representation should include finance-specific notions and constructs, while allowing for both time and monetary qualifiers.

[3]The following and all subsequent legal texts are taken from UK legislation.

3. A Financial Services Regulations Controlled Natural Language

The *Financial Services Regulations Controlled Natural Language* (FSRCNL) was designed to allow legal experts to specify compliance checks for payment services specific entities (e.g. *transaction*s, *instrument*s, *service provider*s). The vocabulary of the language also includes verbs that can be used to specify relations between these entities (e.g. programme *p is regulated in the UK*). Apart from these, the language includes (i) temporal notions allowing for the expression of when something occurred (e.g. instrument *i expired less than 12 months ago*), and (ii) monetary expressions specifying limits on a value (e.g. transaction *t deals with exactly 500 EUR*). These can be combined to specify conditions on aggregate values (e.g. *the amount redeemed from i within a calendar year is less than 1000 EUR*). Some interesting aspects of the language are the following:

Types The regulations define several payment-specific objects, as well as certain roles for entities providing payment services. To handle the several laws regulating the behaviour of entities taking such roles, we include the possibility of declaring quantified variables over such types (roles) in FSRCNL (e.g. *programme p*).

⟨Type-Name⟩	::=	programme | service provider | instrument | transaction | country
⟨Declaration⟩	::=	[(⟨Type-Name⟩,)* or]?⟨Type-Name⟩ ⟨Variable⟩

Payment relations/Sentences Basic relations between variables over the defined payment types serve as the basis for sentences. These are of the subject-verb-object form, and can be negated. They are used to specify both conditions limiting a clause and the state that must hold given the conditions, as in:

For each instrument i and programme p, where i is an instrument of p , p is regulated in the UK ,
then e-money in i is redeemed without fees.

Monetary Expressions Monetary expressions can act as objects of a relation, e.g. one can specify that a transaction *t deals with exactly 500 EUR*, allowing also for specifying *less than* or *more than* a certain value.

Temporal and Country Qualifiers Relations can be further qualified with some temporal, or location constraint, refining a relation to hold only in a certain country (e.g. given service provider *sp*, and programme *p*, *sp deploys p in the UK*). Furthermore, we can also refine sentences to a time period before or after a certain point in time (e.g. instrument *i expired less than 12 months ago*).

⟨Qualified-Sentence⟩	::=	⟨Sentence⟩
	|	⟨Sentence⟩ ⟨Temporal-Qualifier⟩
	|	⟨Sentence⟩ ⟨Country-Qualifier⟩
⟨Temporal-Qualifier⟩	::=	in less than ⟨Time⟩ | in more than ⟨Time⟩
⟨Country-Qualifier⟩	::=	in ⟨Country⟩ | not in ⟨Country⟩

Guarded Declarations and Guards A FSRCNL specification can have a list of variable declarations, which can possibly be guarded by a compound sentence (e.g. *service provider sp, and programme p where sp deploys p in the UK*).

⟨Guarded-Declaration⟩	::=	⟨Declaration-List⟩ ⟨Guard⟩
⟨Guard⟩	::=	ε | where ⟨Compound-Sentence⟩
⟨Compound-Sentence⟩	::=	[(⟨Qualified-Sentence⟩,)* and]⟨Qualified-Sentence⟩
	|	[(⟨Qualified-Sentence⟩,)* or]⟨Qualified-Sentence⟩

Quantifiers The variables declared can be universally or existentially quantified over.

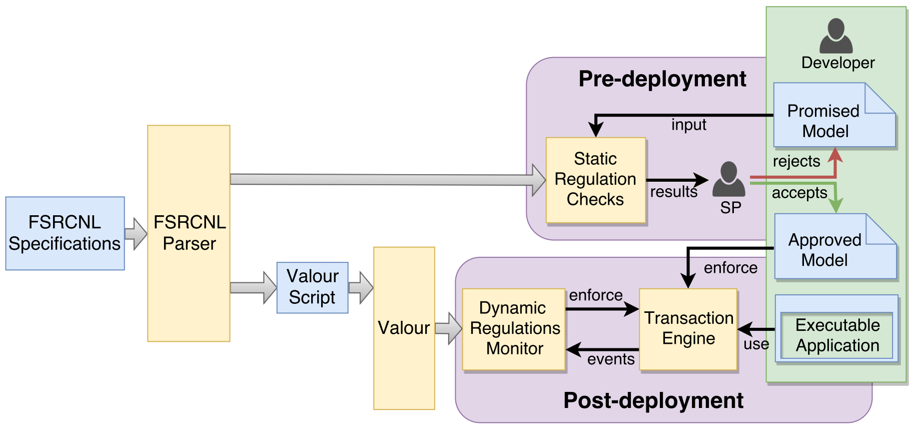


Figure 2. OPE payment application business process with verification facilitated by FSRCNL.

⟨QUANTIFIED-PROPOSITION⟩	::=	For each ⟨GUARDED-DECLARATION⟩ then ⟨COMPOUND-SENTENCE⟩
	|	For at least one ⟨GUARDED-DECLARATION⟩ then ⟨COMPOUND-SENTENCE⟩

In FSRCNL, **EMR45** would be represented as follows:

For each programme p, and instrument i, where i is an instrument of p, p is regulated in the UK, and i deals with e-money, then i does not give time-based rewards.

The language and a parser for FSRCNL was built using Haskell and the *parsec* package, using around 700 lines of code. The language was built from scratch, in a compositional manner so as to allow for easy extension when the need arises.

4. Verifying FSRCNL Rules

We used FSRCNL in the context of an ecosystem that provides common functionality needed for payment applications. The OPE was designed in such a manner that third party payment applications are seen as black boxes interacting with service providers through the OPE. While observing the runtime behaviour of applications enables the OPE to ensure compliance, ideally non-compliant applications are not allowed to use the platform upfront. Thus, developers are required to provide a *promised model* specifying assurances about the application's behaviour (e.g. the model can specify that the application will only perform transactions within the UK). The compliance verification is divided into two stages: (i) pre-deployment the OPE uses the developer-provided model of the application, by statically verifying it against the formulated legislation; while (ii) post-deployment verification involves the analysis of the dynamic features of an application in order to ensure that it is working according to the constraints of the legislation and according to the promised model that was used for the pre-deployment analysis (e.g. transaction values). In this context, FSRCNL relations can either be linked to artifacts known pre-deployment (statically) or to the actual application's behaviour post-deployment (dynamically).

Syntax and Semantics The sentence form in FSRCNL is similar to a quantified predicate logic implication, we can give static rules their semantics in terms of such a logic. For **EMR45** we can give its semantics as follows, assuming a programme *p*: $\forall i \in \texttt{instruments}(p) \cdot \mathit{regulatedIn}(p, UK) \wedge \mathit{emoney}(i) \implies \mathit{noTimeRewards}(i)$. Given such a semantics of the CNL, it is straightforward to take the payment application model (which essentially provides information about the resource flow of the financial instruments which the application will enable) and confirm that it complies to these quantified formulae.

Some relations in the FSRCNL cannot be linked to attributes known before deployment, and thus have to be pushed post-deployment. Recall regulation **ML13(7)(d)(ii)** having guard: *the amount redeemed from i within a calendar year is less than 1000 EUR*. This amount can only be known at runtime, and thus we postpone checking this rule to post-deployment using a runtime monitor. The semantics of the post-deployment logic is given in terms of an event-based language, as required for the runtime verification tool VALOUR [2].

For instance, from regulation **ML13(7)(d)(ii)**, VALOUR generates executable code for each instrument (i) keeping a running total of amounts redeemed from transactions (on that instrument), reset upon the beginning of a calendar year; and (ii) setting up a guarded event to trigger when the running total exceeds €1000, signalling a violation of the regulation.

These checks are applied directly to the OPE (via the Java code generated from the static analysis checks and the code generated by VALOUR), adding a safety layer around the applications use of the API on the OPE side — ensuring that any compliance breach is reported. Figure 2 illustrates the executable specification generation process through the use of FSRCNL, and their verification at the different stages of an application's lifetime as explained in this section.

5. Discussion and Lessons Learnt

In this section we discuss several issues related both to the semantics and syntax of the language, and lessons learnt from the process we undertook.

When analysing the regulations we found ambiguity between the terms used by the lawyers and the developers, e.g. the term *instrument* was used to mean different but similar things for both of them which created some problems in the formalisation early on. Deciding to use the system-level constructs at a high-level could have led to behaviour unintended by the legal expert authors. When designing a CNL, we can avoid such a situation by taking into account the *target authors* of the language and their preconceptions about its vocabulary, and putting that above any other considerations. In fact we geared the language of our CNL to keywords and phrases as found and used by the legal experts, hiding the verified system's constructs in the semantics of the language.

The CNL however still uses somewhat unnatural structures more reminiscent of logic rather than legal text, particularly explicit quantification and variables. These are used to remove the ambiguities of the anaphora common in natural language text. Although we aimed for the CNL to be as natural as possible, removing any avenues for ambiguity necessarily results in a somewhat formulaic structure [10]. The effect of this on the usability of the language has not yet been evaluated with legal experts. Development of FSRCNL only started after the process illustrated in Figure 1a was largely over, and given the nature of the project, the legal experts were not available for evaluation.

In the process of requirements elicitation, as common with natural language texts, we found the laws to be ambiguous or underspecified at times, e.g. one clause specifies that issuing of e-money should be done *without delay*[4]. With these kind of regulations the lawyers depended on guidance from relevant authorities and their experience, e.g. *without delay* was interpreted as meaning that a certain delay was acceptable, namely the amount of time needed to process such a request, and an approximate numeric value was agreed with the developers. Thus the semantics we gave to FSRCNL encode a specific interpretation of the regulations.

Given the narrow scope of the language, the variety of notions in the language, and the limited kind of clauses that are automatically verifiable, we do not expect the language will need substantial change in structure in the face of new regulations. This was validated by our experience when constructing the language. The language was based on two sets of regulation documents (the e-money and payment services regulations). We then further extended it with respect to the money laundering (ML) related regulations. This extension of the language only involved the introduction of new ML-specific verbs. In this case the general structure of the language was found to be adequate to represent the previously unseen regulations.

In [14] Kuhn introduces a classification scheme for CNLs, namely the PENS scheme, standing for precision, expressiveness, naturalness, and simplicity. Each dimension is restricted to five classes where, e.g., for precision (or lack of ambiguity), P^1 denotes the lowest possible precision and P^5 the highest. According to the definitions given by Kuhn, we would classify FSRCNL as $P^4E^3N^3S^4$. The language is not maximally precise since the semantics of its basic sentences depend on the underlying system, in our case the OPE, while expressiveness is limited since it does not include second-order universal quantification. Naturalness of the language is reduced by the use of variable declarations (which is not something every English user would feel to be natural) and by the repetitive nature of the specifications. Simplicity is defined in terms of the length of an “exact and comprehensive description”, S^4 denotes that FSRCNL can be described in not more than ten pages.

We consider briefly the application of the language to verification of applications. In verification although we ideally want to prove the application correct pre-deployment, this may not always be possible (e.g. its source code is not available, or Turing-completeness of the code makes this difficult), and we may have to leave some work for after deployment [3]. FSRCNL is a key part in our approach to this problem in the context of the OPE — the parser we developed for FSRCNL is able to automatically classify specifications into those that can be proven fully pre-deployment, and those that have to be left for runtime.

6. Related Work

CNLs have been applied to serve as an interface to verification tools before [15,13,12]. In particular, Grover et. al. in [12] specify a language for the verification of hardware models that translates into a temporal logic. This work proposes allowing CNLs to have some ambiguity, since some words may be inherently ambiguous, while managing it by presenting the different interpretations to the user for feedback. In our work we opted for a CNL without ambiguity for ease of processing. Future work could include considering different modalities of FSRCNL depending on the kind of user using it (legal expert or

[4]UK Electronic Money Regulations 2011 Regulation 39(a).

developer), and letting the user choose between different options of semantics for some terms (e.g. to vary the processing time allowed for “without delay”).

Similar to our paper, [15] presents a CNL for the specification of regulations to be verified against a model but in the context of railways. Like other CNLs intended for legal source representation (e.g. [1,7,16,8]), it makes explicit use of deontic notions (i.e. obligation, permission, and prohibition). We found the current form of FSRCNL to be enough to specify what should be specified, but the introduction of explicit deontic notions can be considered if it will facilitate a legal expert’s understanding of the language.

A popular English CNL that can be mapped to first-order logic is Attempto Controlled English (ACE [11]). Subsets of it has been mapped to logics for different domains, e.g. a privacy policy language [9]. Using ACE allows the use of different already existing tools to reason with the specification. We opted to develop our own CNL to be fully in control of its syntax.

Previous CNLs by authors of this paper dealing with analysis took the form of imperative commands [6,10], in that they were used to instruct the system what actions to take. On the other hand, FSRCNL sentences are declarative, with the semantics engine inferring the instructions for checking for the specified conditions, thus acting as a filter.

7. Conclusions

We have reported on our experience of developing a controlled natural language (CNL) for the compliance checking of financial services regulations in an industrial project. The language was developed to replace a manual process of developing these checks, involving multiple stakeholders and artifacts. Our intention is to use the *Financial Services Regulations CNL* to allow legal experts to specify rules that both document and automatically generate implementable compliance checks. We hypothesise that using FSRCNL will streamline this process. The next step in this work is to evaluate this language with legal experts in terms of ease of use, upon the introduction of new applicable legislation.

References

[1] Krasimir Angelov, John J. Camilleri, and Gerardo Schneider. A framework for conflict analysis of normative texts written in controlled natural language. *The Journal of Logic and Algebraic Programming*, 82(5):216 – 240, 2013. Formal Languages and Analysis of Contract-Oriented Software (FLACOS'11).

[2] Shaun Azzopardi, Christian Colombo, Jean-Paul Ebejer, Edward Mallia, and Gordon J. Pace. *Runtime Verification using* VALOUR. EPiC Series in Computing, Volume, 2016.

[3] Shaun Azzopardi, Christian Colombo, and Gordon Pace. *A Model-Based Approach to Combining Static and Dynamic Verification Techniques*, pages 416–430. Springer International Publishing, Cham, 2016.

[4] Shaun Azzopardi, Christian Colombo, Gordon J. Pace, and Brian Vella. *Compliance Checking in the Open Payments Ecosystem*, pages 337–343. Springer International Publishing, Cham, 2016.

[5] Shaun Azzopardi, Albert Gatt, and Gordon J. Pace. Integrating natural language and formal analysis for legal documents. In *10th Conference on Language Technologies and Digital Humanities 2016*, 2016.

[6] Aaron Calafato, Christian Colombo, and Gordon J. Pace. A controlled natural language for tax fraud detection. In *Proceedings of the 5th International Workshop on Controlled Natural Language - Volume 9767*, CNL 2016, pages 1–12, New York, NY, USA, 2016. Springer-Verlag New York, Inc.

[7] John J. Camilleri, Gordon J. Pace, and Michael Rosner. Controlled natural language in a game for legal assistance. In Michael Rosner and Norbert E. Fuchs, editors, *Controlled Natural Language*, pages 137–153, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[8] John J. Camilleri, Gabriele Paganelli, and Gerardo Schneider. A cnl for contract-oriented diagrams. In Brian Davis, Kaarel Kaljurand, and Tobias Kuhn, editors, *Controlled Natural Language*, pages 135–146, Cham, 2014. Springer International Publishing.

[9] Juri Luca De Coi, Philipp Kärger, Daniel Olmedilla, and Sergej Zerr. Using Natural Language Policies for Privacy Control in Social Platforms. Heraklion, Greece, Jun 2009. CEUR-WS.org.

[10] Christian Colombo, Jean-Paul Grech, and Gordon J. Pace. A controlled natural language for business intelligence monitoring. In Chris Biemann, Siegfried Handschuh, André Freitas, Farid Meziane, and Elisabeth Métais, editors, *Natural Language Processing and Information Systems*, pages 300–306, Cham, 2015. Springer International Publishing.

[11] Norbert E. Fuchs, Kaarel Kaljurand, and Tobias Kuhn. *Attempto Controlled English for Knowledge Representation*, pages 104–124. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[12] Claire Grover, Alexander Holt, Ewan Klein, and Marc Moens. Designing a controlled language for interactive model checking. In *Proceedings of the 3rd International Workshop on Controlled Language Applications (CLAW 2000)*, 2000.

[13] Kristofer Johannisson. *Natural Language Specifications*, pages 317–333. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[14] Tobias Kuhn. A survey and classification of controlled natural languages. *Comput. Linguist.*, 40(1):121–170, March 2014.

[15] Bjørnar Luteberget, John J. Camilleri, Christian Johansen, and Gerardo Schneider. Participatory verification of railway infrastructure by representing regulations in railcnl. In Alessandro Cimatti and Marjan Sirjani, editors, *Software Engineering and Formal Methods*, pages 87–103, Cham, 2017. Springer International Publishing.

[16] Gordon J. Pace and Michael Rosner. A controlled language for the specification of contracts. In Norbert E. Fuchs, editor, *Controlled Natural Language*, pages 226–245, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[17] Adam Z. Wyner and Guido Governatori. A study on translating regulatory rules from natural language to defeasible logics. In *Joint Proceedings of the 7th International Rule Challenge, the Special Track on Human Language Technology and the 3rd RuleML Doctoral Consortium, Seattle, USA, July 11 -13, 2013*, 2013.