Noname manuscript No.
(will be inserted by the editor)

An Automated Framework for Detection and Resolution of Cross References in Legal Texts

Nicolas Sannier · Morayo Adedjouma · Mehrdad Sabetzadeh · Lionel Briand

Received: date / Accepted: date

Abstract When identifying and elaborating compliance requirements, analysts need to follow the cross references in legal texts and consider the additional information in the cited provisions. Enabling easier navigation and handling of cross references requires automated support for the detection of the natural language expressions used in cross references, the interpretation of cross references in their context, and the linkage of cross references to the targeted provisions. In this article, we propose an approach and tool support for automated detection and resolution of cross references. The approach leverages the structure of legal texts, formalized into a schema, and a set of natural language patterns for legal cross reference expressions. These patterns were developed based on an investigation of Luxembourg's legislation, written in French. To build confidence about their applicability beyond the context where they were observed, these patterns were validated against the Personal Health Information Protection Act (PHIPA) by the Government of Ontario, Canada, written in both French and English. We report on an empirical evaluation where we assess the accuracy and scalability of our framework over several Luxembourgish legislative texts as well as PHIPA.

Keywords Legal Compliance · Natural Language Processing (NLP) · Cross References · Conceptual Modeling

1 Introduction

In many domains such as healthcare, finance and government, software systems are subject to various laws and regulations, e.g., about security and privacy. Failure to comply with the applicable laws and regulations can have serious consequences, including fines, lawsuits, damage to public trust, and even criminal prosecution. An important complexity that arises in the analysis of legal texts is that legal provisions are typically interrelated and spread over different texts that cannot be considered in isolation of one another. The relationships between different provisions in legal texts are captured using *cross references*.

Fig. 1 provides two examples of cross references. Fig. 1(a) is an excerpt of an article from Luxembourg's Income Tax Law [17] (translated from French). In order to fully understand the scope of this provision, one needs to know what local income exactly means for a non-resident taxpayer. The necessary definition is provided in the cited provision, namely Art. 156. The second example, shown in Fig. 1(b), is a section excerpt from the Personal Health Information Protection Act (PHIPA) of the Government of Ontario, Canada [32]. The section constrains the circumstances under which personal health information can be disclosed. In particular, disclosure is possible only when the recipient entity meets the requirements that are elaborated in a different provision, namely

(a) **Art. 2.** (...) Individuals are considered non-resident taxpayers if they do not reside in Luxembourg but have a local income as per the definition of Art. 156

(b) **Disclosure for planning and management of health system**

45. (1) A health information custodian may disclose to a prescribed entity personal health information for the purpose of analysis or compiling statistical information with respect to the management of, evaluation or monitoring of (...) the health system (...) if the entity meets the requirements under subsection (3).

(3) A health information custodian may disclose personal health information to a prescribed entity under subsection (1) if,
(a) the entity has in place practices and procedures to protect the privacy of the individuals whose personal health information it receives and to maintain the confidentiality of the information;

Fig. 1 Examples of Cross References in Legal Texts

SnT Centre for Security, Reliability and Trust,
University of Luxembourg, Luxembourg.
E-mail: firstname.lastname@uni.lu

subsection (3). In the examples of Fig. 1, the citing provisions depend on the cited ones for definitions and further elaboration. Cross references can be used for a variety of other reasons, including stating exceptions and constraints, specifying priorities between provisions, and making amendments to other provisions [24].

Since laws reflect expectations in terms of rights and obligations [34], they represent an important source for software requirements. Consequently, although mainly an apparatus of legal writing, cross references have implication not only on legal texts but also on software requirements. The relevance of cross references to software requirements is highlighted by Maxwell et al. [23,24], who argue that failing to consider cross references or misunderstanding their intent can lead to costly non-compliance issues in software. Several strands of work concerned with legal analysis in Requirements Engineering take cross references into consideration. For example, Breaux & Antón [5] follow cross references during requirements elaboration and analyze the cited provisions for identifying constraints, priorities, exceptions, refinements, and conflicts between compliance requirements. Ghanavati et al. [15,14] model legal cross references as explicit goals and use these goals both for compliance analysis of business processes and for change propagation between requirements.

To perform the above activities in a more systematic and efficient manner, it is important to have legal texts structured as markup documents, e.g., in an XML format, with cross references represented as navigable links [6,19]. The resulting links on the one hand enable easier and more structured exploration of legal texts by analysts, and on the other hand, provide a basis for further analysis, particularly traceability and impact analysis [15].

Many legal texts are now available on-line via government portals and legal information databases. Some of these portal and databases provide the texts in PDF format (e.g., LegiLux[1]), and some others – in both PDF and a markup format such as XML or HTML (e.g., the French LegiFrance[2], the Canadian e-laws portal[3], and the Belgian BelgiumLex[4]). These portals sometimes further maintain navigation links for the cross references in the legal texts; however, these links are coarse-grained in the sense that they are only at the level of articles or entire legal texts.

Fine-grained links that allow navigation to smaller units of legal texts such as paragraphs and clauses are rare. Although having such fine-grained links is important, both to facilitate the navigation of legal texts and to enable establishing precise traceability links between legal requirements and legal texts, creating such links manually would be expensive. Automation is therefore essential for generating fine-grained cross reference links. To do so, we need to be able to automatically recognize the Natural Language (NL) expressions that denote cross references (*cross reference detection*), and to interpret these expressions and link them to the target provisions (*cross reference resolution*).

Several approaches already exist for cross reference detection and resolution [26,11,6,18,33,34]; but, as we argue in more detail in Section 11, certain aspects of the problem have not been adequately addressed:

– There are books and best-practice guides for drafting legal texts and cross references. For example, the Bluebook [2] and the US Association of Legal Writing Directors' (ALWD) Citation Manual [12] lay down specific conventions for cross references. These best practices, as already observed by others [18], are often inadequate for accurate detection of cross references, particularly in older legal texts. Grounded Theory studies of actual legal texts, e.g., as done by Breaux [6] and de Maat et al. [11], provide valuable insights about the flexible NL patterns used for specifying cross references. However, further investigation of actual legal texts is required to understand commonalities between legal texts across different countries and to develop reusable cross reference patterns.
– Having legal texts in a markup format, e.g., XML, is an important prerequisite for cross reference resolution. However, significant manual work is still necessary to transform a non-markup legal text (e.g., in PDF or plain text) into a markup document.
– The majority of existing work does not clearly distinguish cross reference detection and the more complex task of resolution. Important subtleties that arise during resolution have not been sufficiently covered, e.g., disambiguation when the cross reference patterns are ambiguous.

In this article, we attempt to address the above gaps by developing a flexible framework for automated detection and resolution of cross references. Fig. 2 shows our overall research method. We start with a study of several legal drafting guidelines and cross reference expressions within selected texts from the Luxembourgish and Canadian legal corpora, covering a total of 3612 expressions. Based on this study, we devise (1) a technique for modeling the structure of legal texts through schemas, and (2) a set of cross reference patterns, which, to our knowledge, is the most detailed and complete set of such patterns that exists to date. Schemas are already commonly used for capturing the structure of legal texts [13,6,19,29]; nevertheless, using these schemas in a systematic way for automated markup generation has not been studied before. Using techniques from Natural Language Processing (NLP), we provide automation for trans-

[1] http://www.legilux.public.lu
[2] http://www.legifrance.gouv.fr
[3] http://www.e-laws.gov.on.ca
[4] http://www.belgielex.be/en/index.html

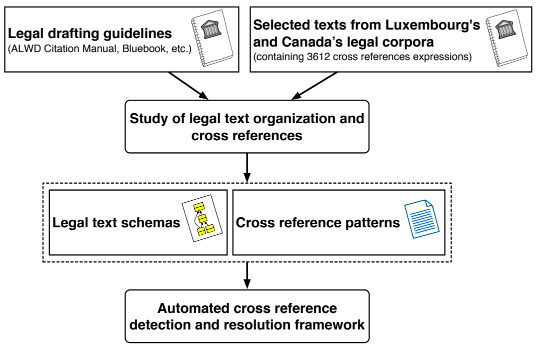

Legal drafting guidelines
(ALWD Citation Manual, Bluebook, etc.)
Selected texts from Luxembourg's and Canada's legal corpora
(containing 3612 cross references expressions)
Study of legal text organization and cross references
Legal text schemas
Cross reference patterns
Automated cross reference detection and resolution framework

Fig. 2 Overall Research Method

forming non-markup texts into texts with structural markup based on schemas. We use the resulting markup along with the identified cross reference patterns as a basis for our automated cross reference detection and resolution framework. An important feature of our framework is that it addresses, in an algorithmic way, subtleties that one needs to take into account with regards to the interpretation of complex cross reference expressions. We evaluate the accuracy of our framework through a large-scale examination of cross reference expressions in Luxembourgish and Canadian legal texts.

This article is an extension of a previous conference paper [1] published at the 22nd IEEE International Requirements Engineering Conference (RE 2014). The main extensions in this article are: (1) a succinct and yet comprehensive guide for understanding the structure of legal texts, (2) an extended description of the NLP machinery in our approach with more details and a new set of illustrating examples, (3) expansion of our work to the Canadian legal corpus through an investigation of cross references in both the French and English editions of the *Personal Health Information Protection Act* (PHIPA) [32]. This investigation is a stepping stone towards assessing the generalizability of our approach. And, (4) additional empirical evidence to demonstrate the effectiveness of our approach. In particular, our extended empirical evaluation discusses resolution accuracy for four new legal texts from Luxembourg's legislative corpus that were not considered in our previous work, as well as over PHIPA.

The remainder of this article is structured as follows: Section 2 provides background information. Section 3 gives an overview of our approach for automatic detection and resolution of cross references. Sections 4 to 7 elaborate the different steps in our approach. Section 8 outlines different uses cases that our approach enables. Section 9 presents tool support. Section 10 reports on our evaluation and discusses limitations and threats to validity. Section 11 compares with related work. Section 12 concludes the article.

The examples used throughout the article are derived from Luxembourgish legislative texts and PHIPA. The Luxembourgish texts considered in our work are all in French. PHIPA comes in both English and French. For presentation purposes, we always use English translations while preserving the structure of the original cross reference expressions.

2 Background

In this section, we present background information on legal texts and the Natural Language Processing (NLP) techniques used in the article for automated cross references detection and resolution.

2.1 Legal Texts

We begin with introductory material on how legal texts are structured and then discuss some general characteristics of cross references in these texts. The reader can find further information in legal guidelines, e.g., the ALWD Citation Manual [12], LegiFrance, LegiLux, BelgiumLex or the European Union (EU) guidelines [31].

2.1.1 Text Schemas

A natural and intuitive way to represent the structure of a legal text is through a (text) schema [13,6,19,29]. To illustrate, Fig. 3 shows a text schema derived from the EU Legislation Drafting Guidelines [31].

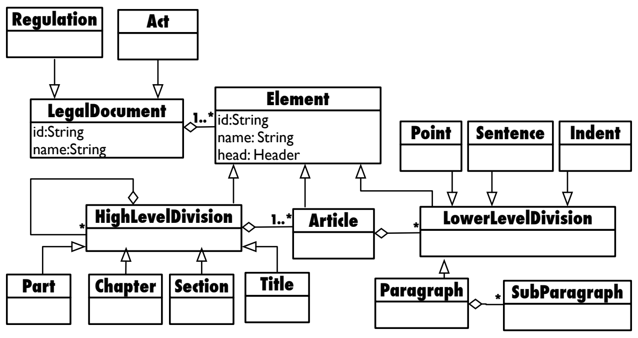


Fig. 3 Structure of European Legislative Texts as Prescribed by the EU Legislation Drafting Guidelines

Articles are the basic structural elements of EU legislative texts. At a higher level, articles may be organized into *Titles*, *Chapters*, *Parts* and *Sections*. At a lower level, subdivisions may be defined to break articles into *(Sub)paragraphs*, *Points*, *Sentences* and *Indents*. Each article is numbered according to its order of appearance in the text. If there is a new article to be inserted in between other articles, it will use an identifier composed of the preceding article's number and an extension. For example, a new article inserted between articles 2 and 3 will use an identifier such as 2-1, 2.1, 2a, 2bis, or the like. Numbering is essential for traceability and referencing. However, whereas requirements documents are often precise in terms of providing a unique identifier for each requirement, legal texts do not provide such a feature for article subdivisions.

The structure of legislative texts in several EU countries, e.g., the Netherlands [11] and Luxembourg [1], are close (but not identical) to the structure depicted in Fig. 3. Nevertheless, this schema is not universal, and important structural differences are to be expected in different countries and different legal jurisdictions. To highlight such differences, consider the following example from the Canadian legal corpus: Fig. 4 presents an excerpt of Section 43 of the English edition of the Personal Health Insurance Protection Act (PHIPA) of Ontario. Rather than being organized into *Articles*, *Paragraphs* and *Subparagraphs*, this excerpt has the following organization: *Sections*, *Subsections*, and *Clauses*. Specifically, the excerpt of Fig. 4 has subsections (1) and (2); subsection (1) contains a number of clauses, of which (a), (e), (f), and (h) are shown. The French edition of PHIPA uses a different structure and is organized into *Articles*, *Paragraphes* and *Alinéas*. The detailed schema for the English edition of PHIPA will be presented and discussed in Section 4.

Aside from differences in the lexical terms used for labeling the structural elements, legal texts may further differ with respect to the containment relationships between the elements, even when the same labels are used. For example, the French terms "paragraphe" and "alinéa" – often used as synonyms in common language – match different hierarchical levels in different legal texts. In French legislative texts, a paragraphe is a subdivision of an alinéa, whereas the opposite holds for Luxembourgish legislative texts.

A similar problem arises when legal texts need to be translated in multiple languages (e.g., as is the case with many EU and Canadian legal texts). For example, assuming the hierarchies outlined above for the English and French editions of PHIPA, the French term "paragraphe" maps onto the English term "subsection", and the English term "paragraph" maps onto the French term "disposition". Therefore "paragraph" in English and "paragraphe" in French are not the same, thus rendering the direct translation of "paragraph" to "paragraphe" incorrect and misleading.

The absence of explicit unique identifiers for numbering the provisions is yet another issue that needs attention. For example, a unique identifier such as Clause 43 (1)(a) would need to be *derived* from the structure of the underlying legal text (here, the English edition of PHIPA). In the text though, the clause is simply numbered (a) in Subsection (1) of Section 43. Consequently, knowing the structure of the text would be essential for resolving cross references to the text at a fine level of granularity.

Disclosures related to this or other Acts

43. (1) A health information custodian may disclose personal health information about an individual,

(a) for the purpose of determining, assessing or confirming capacity under the Health Care Consent Act, 1996, the Substitute Decisions Act, 1992 or this Act;

[...]

(e) to the Public Guardian and Trustee, the Children's Lawyer, a children's aid society, a Residential Placement Advisory Committee established under subsection 34 (2) of the Child and Family Services Act or a designated custodian under section 162.1 of that Act so that they can carry out their statutory functions;

(f) in the circumstances described in clause 42 (1) (c), (g) or (n) of the Freedom of Information and Protection of Privacy Act or clause 32 (c), (g) or (l) of the Municipal Freedom of Information and Protection of Privacy Act, if the custodian is an institution within the meaning of whichever of those Acts applies, or is acting as part of such an institution;

[...]

(h) subject to the requirements and restrictions, if any, that are prescribed, if permitted or required by law or by a treaty, agreement or arrangement made under an Act or an Act of Canada. 2004, c. 3, Sched. A, s. 43 (1); 2005, c. 25, s. 35; 2006, c. 34, Sched. C, s. 26; 2007, c. 10, Sched. H, s. 15.

Interpretation

(2) For the purposes of clause (1) (h) and subject to the regulations made under this Act, if an Act, an Act of Canada or a regulation made under any of those Acts specifically provides that information is exempt, under stated circumstances, from a confidentiality or secrecy requirement, that provision shall be deemed to permit the disclosure of the information in the stated circumstances. 2004, c. 3, Sched. A, s. 43 (2).

Fig. 4 Excerpt from PHIPA [32]

As we elaborate in Section 4, our solution uses schemas in order to systematically deal with structural complexities in legal texts.

2.1.2 Cross References

A *(legal) cross reference* is a citation that links one legal provision to another [24]. We distinguish cross references from cross reference expressions (CREs). A *CRE* is a natural language phrase in a legal text that represents one or more cross references. For example "clause 32 (c), (g) or (l) of the Municipal Freedom of Information and Protection of Privacy Act" is a CRE. This expression embodies three cross references: one to section 32 clause (c), one to section 32 clause (g) and another to section 32 clause (l) of the respective law.

We note that, in light of the above distinction made between cross references and cross references expressions, it would be more accurate to refer to cross reference detection and cross reference resolution as CRE detection and CRE resolution. We ignore this technicality when referring to the detection and resolution activities.

A cross reference is *internal* when it refers to a provision within the same legal text and *external* when the cross reference cites a provision in a different legal text [22]. In the example of Fig. 4, "clause (1) (h)" implies an internal cross reference while, "clause 32 (c), (g) or (l) ..." implies external cross references.

Cross references can be further classified as explicit, implicit, or delegating. If a cross reference is defined using the alphanumeric labels of the legal text, it is called *explicit*. All our examples so far where provisions were referred to by their numbers were explicit. In contrast, an *implicit* cross reference is referred to via the use of adjectives, adverbs, or anaphors [11]. For example, the cross references implied by the following CREs are implicit: "this section" and "the following paragraphs".

The third class, namely *delegating*, exclusively applies to external cross references. This class of cross references is used when a text delegates authority to another text, without explicitly naming the text, for further elaboration. For example, legislative texts seldom refer to regulations in a precise way and typically use cross references that only indicate the nature of the regulations being cited. An example of a delegating cross reference is "Grand-Ducal regulation" in the following: "A Grand-Ducal regulation shall provide the details for . . . ".

Finally and with regards to implicit cross references, there are occasions where legal texts use vague terms such as "provision" (in French: "prescription"), e.g., "the above provision". We refer to the intended cross references as *unspecific*. Unspecific cross references cannot be conclusively associated with a particular structural element, e.g., a subsection or a paragraph. They are thus difficult to resolve with reasonable accuracy through automation. Except for delegating and unspecific cross references, all cross references are in principle resolvable via automation.

In Sections 6 and 7, we will describe the technical details of our cross reference detection and our cross reference resolution solutions, respectively.

2.2 Natural Language Processing for Cross References

As we elaborate further in Section 6, we use a BNF grammar to represent the structure of CREs. This abstract representation enables the definition of pattern matching rules for detecting the CREs in legal texts. We use the GATE workbench [10] – a mature open-source Natural Language Processing (NLP) framework – for this purpose. We choose GATE primarily because of its high usability, the availability of documentation and detailed guidelines for plugin development. GATE provides various modules for processing natural language. In our work, we are interested specifically in the *Tokenizer*, *Sentence Splitter* and *Named Entity Recognizer* modules. First, the Tokenizer is executed. This module breaks up the text into units called tokens. Tokens can be words, numbers or punctuation. Next, the Sentence Splitter is executed to identify the sentences within the text. Finally, the Named Entity Recognizer is executed in order to classify text elements into certain predefined categories such as dates, locations and names. This module can be enhanced with custom categories defined through keyword lists. The keyword lists are commonly known as gazetteers [10]. In our case, we use gazetteers to cover, among other things, the terms used in implicit cross references, e.g., next and previous, as well as law titles and names that have to be handled as a whole.

The output from the sequential execution of the above modules is an annotated document with both generic annotations such as Token, Sentence, Date as well as specific ones, such as Implicit_Term. These annotations are used for finding the CREs via pattern matching. For pattern matching, GATE provides a rule-based language, called the Java Annotation Patterns Engine (JAPE). Fig. 5 shows an example of a JAPE rule, named *MarkSectionReference*. The rule matches text regions starting with either Sec. or Section and followed by an alphanumeric expression. When a match is found, the matched region is annotated as Section_Ref.

```
1. Phase: MarkReference
2. Input: Token
3. Options: control = Appelt
4. Rule: MarkSectionReference
5. (
6.   ( {Token.string==“Sec”}{Token.string==“.”} | {Token.string==“Section”})
7.    ({Token.kind=="numberPrime"} | {Token.kind=="number”})
8. ):label
9. --> label.Section_Ref = {rule=“MarkSectionRef”}
```

Fig. 5 JAPE Rule for Identifying Section References

Each JAPE rule is part of a phase, which is a collection of rules to be executed sequentially (L. 1). JAPE rules match regular expressions over annotations rather than on the strings of a document. To match a string, it is necessary to match an annotation that covers that string, i.e., a token defined by the Tokenizer. The GATE Tokenizer adds a "string" feature to each Token annotation, containing the string that the Token covers.

Each JAPE rule is made of two parts: a left-hand-side (LHS) and a right-hand-side (RHS). The LHS is the part preceding the "−−>" and the RHS is the part that follows it. The LHS specifies the pattern to be matched (L. 5-8). The RHS specifies the operation(s) to be performed when a match is found (L. 9). Annotations on the LHS may be referred to on the RHS by means of labels (L. 8).

A JAPE rule considers only the annotation types that are specified in the "Input", here only Token (L. 2). One can specify the features (and values) of an annotation to be matched in order to filter the match (L. 6-7). For instance, the annotation Token comes, by default, with the following features: kind (to differentiate, for instance, between numbers and words), length (to check the size of the token), and string (the text segment itself) (L. 6-7). JAPE provides the classic operators for regular expressions including negation (!), alternatives (|), repetitions (?,*,+), ranges ([]), equality (==, ! =), comparison (<, <=, >=, >). The language further provides more advanced operators such as contains, which checks whether an annotation completely contains another. As JAPE rules are ultimately translated into Java, the RHS of a JAPE rule can contain Java code to create or manipulate annotations with more specific or advanced operations.

JAPE rule collections can have different control options (L. 3). We will not go into the details of these options. In our case, we use the appelt style, which means that only the rule that matches the longest text segment will be fired. This option is particularly useful for ensuring that CREs are matched in their entirety. For example, given a CRE such as "clause 42 (1) (c), (g) or (n)", different matches would be possible: "clause 42 (1) (c)", "clause 42 (1) (c), (g)" and "clause 42 (1) (c), (g) or (n)". Using the appelt style, only the longest segment will be matched.

3 Approach

Our approach for automated identification and resolution of cross references is shown in Fig. 6. The approach has four main steps. The first step is manual and the remaining three steps are automated.

As explained earlier in Section 2.1, detecting and resolving cross references in a legal text requires precise knowledge of the structure of the legal text under analysis. Step 1 is concerned with the definition of a schema for expressing how a legal text is organized into subparts, e.g., sections, clauses, paragraphs. Although legal writing guidelines in a given jurisdiction typically prescribe a generic schema for structuring legal texts, such schemas may need to be tailored due to practical variations in actual legal texts. This tailoring is discussed in Section 4.

Step 2, which is detailed in Section 5, is concerned with transforming a non-markup text (e.g., in plain text or PDF format) into a markup text (e.g., in HTML or XML format). The transformation rules are automatically derived from the schema of Step 1. The main step of the approach is Step 3, which deals with the detection and resolution of cross references. This step is discussed in Sections 6 and 7. Finally, Step 4, discussed in Section 8, is concerned with using the outcomes of the resolution step for different applications such as visualization and analysis.

We note that Sections 6 and 7 are intended at providing a detailed exposition of how we handle cross references in our approach. A reader who is more interested in practical applications (Section 8) may wish to read only the beginning of Sections 6 and 7, where we provide a synopsis of these sections, and skip the more technical material in Sections 6.1, 6.2, 7.1 and 7.2.

4 Capturing the Structure of Legal Texts

Schemas, as outlined in Section 2.1, constitute the basis for capturing the structure of legal texts.

We define a schema through a UML class diagram, where classes represent the structural elements (articles, clauses, paragraphs, etc.) of a legal text. These classes are linked via aggregation associations representing the hierarchical containment relationships between the elements. We further include in the schema the multiplicity constraints that need to be satisfied for the legal text to be structurally sound.

Fig. 7 presents the detailed schema for the English edition of PHIPA. The main structural elements of this text were already highlighted in Section 2.1.

PHIPA is an individual *Act* that is hierarchically organized into *Chapters*, *Parts* and *SubParts*. At a lower level (under *Part* or *Subpart*), provisions are listed under *Sections* in a way that no part of the text falls outside some section. A *Section* is generally organized into *Subsections*. *Subsections* may be divided into *Clauses* and *Subclauses*, or *Paragraphs* and *Subparagraphs*, respectively. Optional levels in the hierarchy are captured through different aggregation paths. For example, in the schema of Fig. 7, one can go directly from *Part* to *Section* if there is no *SubPart* in the text. Instances

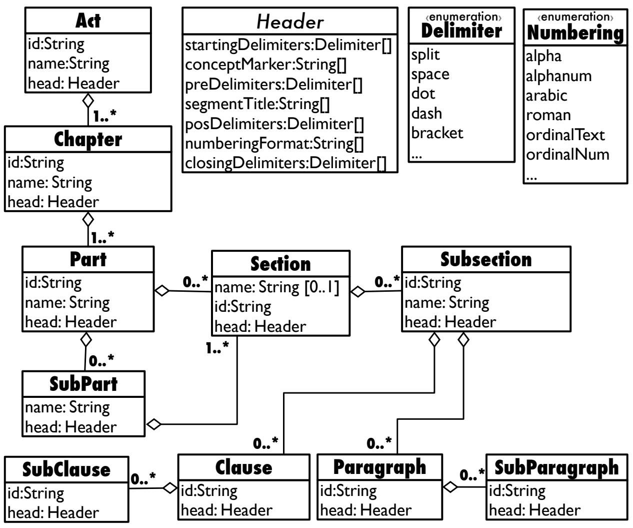


Fig. 7 (Text) Schema for the English Edition of PHIPA

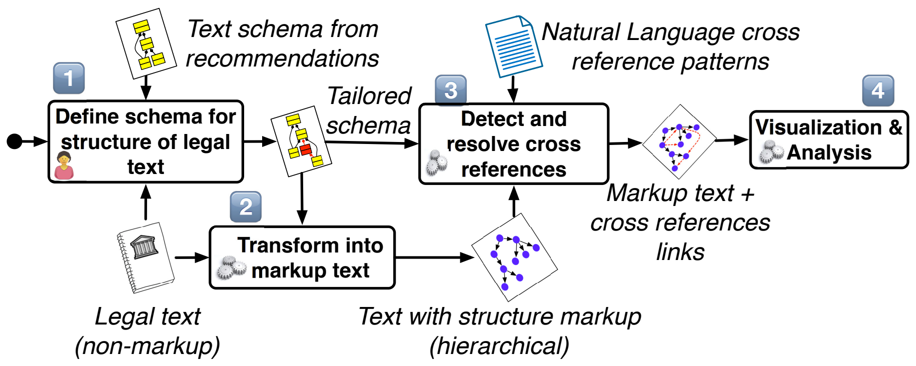

Text schema from recommendations
1
Define schema for structure of legal text
Tailored schema
3
Detect and resolve cross references
Natural Language cross reference patterns
4
Visualization & Analysis
Markup text + cross references links
2
Transform into markup text
Legal text (non-markup)
Text with structure markup (hierarchical)

Fig. 6 Approach Overview

of some of the classes of the schema of Fig. 7 can be seen in the excerpt of Fig. 4.

Each structural element has a *Header*, and optionally an *id* and a *name*. The header of a structural element *C*, called *CHeader*, provides information about how to recognize an instance of *C* in the text. In the schema of Fig. 7, only the abstract header class is shown with its attributes. Each structural element is a (static) specialization of this abstract class.

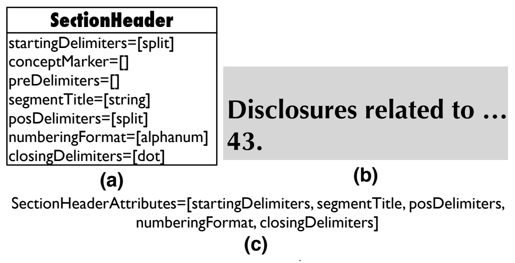

SectionHeader
startingDelimiters=[split]
conceptMarker=[]
preDelimiters=[]
segmentTitle=[string]
posDelimiters=[split]
numberingFormat=[alphanum]
closingDelimiters=[dot]
(a)
Disclosures related to ...
43.
(b)
SectionHeaderAttributes=[startingDelimiters, segmentTitle, posDelimiters, numberingFormat, closingDelimiters]
(c)

Fig. 8 (a) Header Class for Section; (b) An instance of SectionHeader; (c) Sequencing of Header Attributes of Section Expressed as a Vector

In Fig. 8(a), we present the *SectionHeader* class for a section, as well as an example in Fig. 8(b), where "Disclosures related to ..." (from Fig. 4) is the title of the section, and "43" is its label. To recognize the *CHeader*, the class attributes must be composed in a specific order. We encode this order in a vector *CHeaderAttributes*, as presented in Fig. 8(c). Here, the vector specifies that a section's header is composed in the following sequence: a starting delimiter, a segment title, a post delimiter, a numbering, and finally a closing delimiter. The starting delimiter is a *split* (carriage return or linefeed). The segment title is a string "Disclosures related to ...", followed by a post-delimiter *split*. The numbering is an alphanumeric ("43") representing the id of the section in the legal text. The section header closes with a *dot*. There is no structural element marker (i.e., an explicit "Section" label) and no pre-delimiter (e.g., a tab or parenthesis) before the start of the section content.

5 Transforming Non-Markup to Markup Text

We automatically derive from a schema, e.g., the one shown in Fig. 7, regular expressions that transform non-markup legal texts to texts with structural markup. The automation builds on a simple observation: the natural structure of a textual document is such that a particular segment of text terminates only when it reaches a new structural element that is either at the same level as the current segment or is at a level above the current segment. For example, given a document structured according to the schema of Fig. 7, and assuming that we are within a particular section, say Section 5, for this section to terminate, we either have to reach the beginning of Section 6, the beginning of a new higher-level division, e.g., a subpart or a part, or the end of the document.

The containment relationships between structural elements is never recursive. This means that we cannot have a structural element, for instance a part, which logically contains another structural element, for instance a section, and at the same time have sections that contain parts. More precisely, a schema, when viewed as a graph, is always a Directed Acyclic Graph (DAG). Consequently, there is always some ordering, known as a *topological ordering*, that respects the containment relationships between the elements [9]. Computing the topological order is inexpensive and linear in the size of the input DAG [9]. For example, one ascending topological order for the schema of Fig. 7 is: [SubParagraph, Paragraph, SubClause, Clause, Subsection, Section, SubPart, Part, Act]. Equipped with this ordering and the information from the *Header* classes in the schema, one can automatically generate the regular expressions that recognize the hierarchical structure of a document. The algorithm for generating and executing these regular expressions is shown in Algorithm 1.

We illustrate the regular expressions for header identification (HeadRegEx) and segmentation (SegmentRegEx) over the *Section* class. Generating the regular expression for marking the heads of sections (L. 3 of the algorithm) is based on in-

Algorithm 1 Build Markup for Legal Text

```
1: Let $G$ be the DAG whose nodes are the classes in a schema and whose (directed) edges are the aggregation associations in the schema.
2: Let $n$ be the number of nodes of $G$ and let $[C_1, \cdots, C_n]$ be an ascending topological ordering of the nodes in $G$
3: For $1 \leq i \leq n$: Generate a regex HeadRegEx$_i$ to recognize $C_i$ headers
4: For $1 \leq i \leq n$: Generate a regex SegmentRegEx$_i$ to recognize $C_i$ segments; i.e., a $C_i$ header followed by the header of any $C_j \in [C_i, \cdots, C_n]$
5: Run all HeadRegEx$_i$ (in any order) on the input text
6: Run all SegmentRegEx$_i$ (in any order) on the input text
```

formation that is captured in the *SectionHeader* class (Fig. 8(a)) and the *SectionHeaderAttributes* vector (Fig. 8(c)).

Fig. 9(a) shows a JAPE script, named *MarkSectionHead*, for marking section heads. The script simply matches the following sequence: one or more splits, an alphanumeric number, and a dot. Fig. 9(b) shows a script, named *MarkSectionSegment*, for marking section segments. This expression recognizes and annotates the text between the head of a given section and the head of the next structural element that is not containable in a section. From the topological ordering, we know which structural elements reside above *Section* and cannot be contained in sections. As seen from the expression in Fig. 9(b), a section's segment starts when its head is detected and stops when the immediately-following section's head or a higher level division is detected. In the script, the special token *EOD* (End Of Document), which has the largest topological order of all, terminates any segment at any other level.

(a)
```
Phase: DoMarkSectionHeader
Input: Token SpaceToken
Options: control = appelt
Rule: DoMarkSectionHeader
({Split})+
(
   ({Token.kind=="numberPrime"} | {Token.kind=="number"}) {Token.string=="."}
):reference
--> :reference.SectionHeader = {}
```

(b)
```
Phase: DoMarkSectionSegment
Input: Part_Head Subpart_Head Section_Head EOD
Options: control = appelt
Rule: DoMarkSectionSegment
(
  ({Section_Head}):left
  ({Section_Head}| {Subpart_Head}| {Part_Head} | {EOD}):right
):reference
-->:reference.Section_Segment={}
```

Fig. 9 Markup Rules for Section Headers and Segments

The annotations produced over a non-markup legal text by the regular expressions can be easily turned into a markup format, e.g., XML. The resulting markup text is the basis for cross reference detection and resolution addressed next.

Line	Simple cross reference patterns		
1	⟨*simple-ref-expr*⟩	::=	⟨*explicit-expr*⟩ | ⟨*implicit-expr*⟩
2	⟨*explicit-expr*⟩	::=	⟨*internal-expr*⟩ | ⟨*external-expr*⟩
3	⟨*internal-expr*⟩	::=	⟨*marker-term*⟩⟨*num-expr*⟩ | ⟨*ordinal-expr*⟩⟨*marker-term*⟩ | ⟨*generic-term*⟩⟨*num-expr*⟩
4	⟨*marker-term*⟩	::=	"article" | "articles" | "art." | "paragraph" | . . .
5	⟨*num-expr*⟩	::=	⟨NUMBER⟩ | ⟨LETTER⟩ | ⟨ALPHANUM⟩
6	⟨*ordinal-expr*⟩	::=	⟨TEXT-ORDINAL⟩ | ⟨NUM-ORDINAL⟩
7	⟨*generic-term*⟩	::=	"sub" | "under"
8	⟨*external-expr*⟩	::=	⟨*external-expr*$_1$⟩ | ⟨*external-expr*$_2$⟩
9	⟨*external-expr*$_1$⟩	::=	⟨*name-term*⟩ | ⟨*category-term*⟩⟨*link-term*⟩ ⟨DATE⟩| ⟨*adj-term*⟩⟨*category-term*⟩⟨*link-term*⟩ ⟨DATE⟩ | ⟨*name-term*⟩⟨*link-term*⟩ ⟨DATE⟩ | ⟨*delegating-expr*⟩
10	⟨*external-expr*$_2$⟩	::=	⟨*internal-expr*⟩⟨*auxiliary-term*⟩⟨*external-expr*$_1$⟩
11	⟨*delegating-expr*⟩	::=	⟨*delegation-term*⟩ | ⟨*adj-term*⟩⟨*delegation-term*⟩
12	⟨*category-term*⟩	::=	"law" | "decree" | "directive" |. . .
13	⟨*name-term*⟩	::=	"social insurance code"|"complementary pension law"| . . .
14	⟨*adj-term*⟩	::=	"modified" | "grand-ducal" | "ministerial"
15	⟨*auxiliary-term*⟩	::=	"as it was introduced by the" | . . .
16	⟨*delegation-term*⟩	::=	"regulation" | "memorial" | . . .
17	⟨*implicit-expr*⟩	::=	⟨*implicit-term*⟩⟨*marker-term*⟩ | ⟨*implicit-term*⟩⟨*category-term*⟩| ⟨*marker-term*⟩⟨*implicit-term*⟩ | ⟨*category-term*⟩⟨*implicit-term*⟩| ⟨*internal-expr*⟩⟨*implicit-term*⟩| ⟨*implicit-term*⟩⟨*unspecific-term*⟩| ⟨*implicit-term*⟩⟨*num-expr*⟩⟨*marker-term*⟩ | ⟨*unspecific-term*⟩⟨*implicit-term*⟩
18	⟨*implicit-term*⟩	::=	"above" | "below" | "preceding" | "following" | "that follows" | "next" | "previous" | "this" | "in question" | "same" | . . .
19	⟨*unspecific-term*⟩	::=	"provision"
20	⟨*link-term*⟩	::=	"of" | "of the" | "of a"
	Complex cross reference patterns		
21	⟨*complex-ref-expr*⟩	::=	⟨*multivalued-expr*⟩ | ⟨*multilayered-expr*⟩
22	⟨*multivalued-expr*⟩	::=	⟨*multivalued-expr*$_1$⟩ | ⟨*multivalued-expr*$_2$⟩
23	⟨*multivalued-expr*$_1$⟩	::=	⟨*internal-expr*⟩⟨*sep-term*⟩⟨*num-expr*⟩ | ⟨*external-expr*⟩⟨*sep-term*⟩⟨*num-expr*⟩⟨*sep-term*⟩ ⟨DATE⟩
24	⟨*multivalued-expr*$_2$⟩	::=	⟨*multivalued-expr*$_1$⟩⟨*sep-term*⟩⟨*num-expr*⟩ | ⟨*multivalued-expr*$_1$⟩⟨*sep-term*⟩⟨*implicit-term*⟩
25	⟨*multilayered-expr*⟩	::=	⟨*multilayered-expr*$_1$⟩ | ⟨*multilayered-expr*$_2$⟩
26	⟨*multilayered-expr*$_1$⟩	::=	⟨*internal-expr*⟩ ⟨*sep-term*⟩ ⟨*internal-expr*⟩ | ⟨*internal-expr*⟩ ⟨*sep-term*⟩ ⟨*num-expr*⟩
27	⟨*multilayered-expr*$_2$⟩	::=	⟨*multilayered-expr*$_1$⟩⟨*sep-term*⟩⟨*internal-expr*⟩ | ⟨*multilayered-expr*$_1$⟩⟨*sep-term*⟩⟨*num-expr*⟩ | ⟨*multilayered-expr*$_1$⟩⟨*link-term*⟩⟨*internal-expr*⟩ | ⟨*multilayered-expr*$_1$⟩⟨*link-term*⟩⟨*num-expr*⟩ | ⟨*multilayered-expr*$_1$⟩⟨*link-term*⟩⟨*multivalued-expr*⟩
28	⟨*sep-term*⟩	::=	"," | "–" | "and" | "or" | "to" | . . .

Fig. 10 Grammar for Natural Language Cross Reference Patterns

6 Detecting Cross Reference Expressions

Cross reference detection is based on the Natural Language (NL) patterns in the CREs. In our previous work [1], we conducted a Grounded Theory (GT) study [8] of Luxembourg's Income Tax Law (circa 2013) [17] with a total of 1223 CREs for identifying the NL patterns used in CREs. To build confidence about the generalizability of our patterns, we subsequently further studied the CREs in both the English and French editions of PHIPA. The former has 1197 CREs and the latter – 1192 CREs[5]. The analysis of PHIPA did not yield new patterns.

[5] The discrepancy between the number of CREs in the French and English editions of PHIPA is due to differences in the wording of the provisions. For example, the statement "(3) Despite clause (1) (b), the person described in that clause ..." from the English edition of PHIPA contains two CREs, namely "clause (1) (b)" and "that clause", whereas the corresponding statement in the French edition, "(3) Malgré l'alinéa (1) b), la personne qui y est visée", contains only one CRE.

Fig. 10 formalizes as a BNF grammar the pattern derived from our GT study. In the grammar, symbols in upper-case letters, e.g., ⟨NUMBER⟩, denote terminals as identified by a NL lexical analyzer. Non-terminals that end with *term*, e.g., ⟨*marker-term*⟩ and ⟨*name-term*⟩, denote elements in predefined dictionaries (gazetteers). These terms vary from one legal jurisdiction and language to another and must be specified for a specific context.

The patterns are organized into two different types: simple and complex. Complex patterns are built on top of simple patterns, providing certain advanced features that we discuss over the course of this section. We illustrate the patterns using several examples. For presentation purposes, we use English translations for the examples drawn from French texts.

We have made minor simplifications to the patterns for better readability. With regards to the French grammar, there is only one simplification to note: In French, ordinals can appear both before and after nouns (e.g., "paragraphe premier", "premier paragraphe"); whereas in English, they can appear only before (e.g., "first paragraph").

6.1 Simple Cross Reference Expressions

A simple CRE can be explicit or implicit (L. 1 of Fig. 10). Among explicit CREs, we distinguish internal and external (L. 2). Non-terminals ⟨*internal-expr*⟩ (L. 3) and ⟨*external-expr*⟩ (L. 8) respectively capture (explicit) internal and (explicit) external CREs.

An *(explicit) internal CRE* (L. 3-7) is either a structural element marker (such as "article" and "section") followed by a numerical expression, or an ordinal expression followed by a structural element marker. The numerical expression can be an arabic number ("section 1"), a roman number ("chapter IV"), an alphanumeric ("alinea 2bis"), a number written in words ("alinea four"), or a letter ("letter a"). A numerical expression may have brackets around it or at the end ("paragraph (2)", "paragraph 2)"). An ordinal expression can be numerical ("1st article") or textual ("first article"). A variant of such pattern is when a generic term (L. 7), e.g., *under*, replaces the structural element marker, e.g., *letter*. For example, "under a" may be used in an article instead of "letter a".

An *(explicit) external CRE* (L. 8-16) can be as simple as just the name of an external law, e.g., "freedom of information and protection of privacy act". Alternatively, an external CRE may be a phrase starting with an optional auxiliary term (e.g., "modified") followed by a legal text category and a date, e.g., "modified law of 23 July 1993". It is further possible for an external CRE to reference the internal provisions of an external law, e.g., "article 54bis as it was introduced by the Law of 23 July 1983". Delegating references also fall under external CREs.

A simple CRE may be *implicit* (L. 17-18), e.g., "this section". Implicit CREs may further combine implicit terms and numerical expressions, e.g., "first four alineas". Among implicit CREs, some cannot be resolved accurately because they use an unspecific term, e.g., "the following provisions".

6.2 Complex Cross Reference Expressions

Complex CREs enhance simple CREs with three additional features: enumerations, ranges, and navigation through levels. Our classification of complex CREs follows de Maat et al.'s [11]: multivalued and multilayered (L. 21). Multilayered CREs can have multivalued parts (L. 27).

A *multivalued CRE* (L. 22-24) cites many provisions within the same expression by specifying only once a structural element marker followed by a numerical expression. The numerical expression may be: (1) an AND/OR enumeration, e.g., "numbers 1, 1a, 2 and 3" and "articles 22bis or 102"; (2) a range, e.g., "subsections (3) to (11)"; (3) a combination of enumerations and ranges, e.g., "articles 119 to 121 and 124".

Similar to simple CREs, multivalued CREs can use different numbering formats, e.g., ordinals as in "second and third alineas". Our grammar allows the repetition of enumerations and ranges within a CRE to accommodate complex cases seen in our study, e.g., "articles 144, 147, 148 to 150, 158 to 160, 161, 162, and 163". We further allow multivalued CREs to include implicit terms, e.g., "articles 26-2, 27 and the following". Neither of these features are captured by de Maat et al. [11].

A *multilayered CRE* (L. 25-27) describes a navigation path through the hierarchy of a legal text. The navigation may be from an upper to a lower level, e.g., "article 91, 1st alinea, No 2". Alternatively, the navigation can be from a lower to an upper level, e.g., "second alinea of article 10 of the law of 23 may 1964". Finally, a navigation can also be mixed-mode. That is, a CRE may start at a convenient hierarchical level, navigate upward or downward in the hierarchy, and then go in the reverse direction. For example, consider the following CREs: "article 3, paragraph 2 of the Law of 8 June 1999" and "numbers 3 and 4 of article 22bis, alinea 2". The navigation in the former is *Article* → *Paragraph* → *Law* and in the latter *Number* → *Article* → *Alinea*. A multilayered CRE may navigate downward in the hierarchy by specifying only the structural element marker of the lower level, e.g., "clause 47 (15) (a)". Multilayered CREs may further use multivalued CREs in their makeup, e.g., "articles 59, alinea 3, 59bis, alinea 1, 170, alineas 2 and 3, 170bis, alineas 1 et 2, 170ter, alineas 1 and 2, and 172, alineas 4 and 5" or "clause 37 (1) (a), 38 (1) (a) or 50 (1) (e)".

7 Resolving Cross Reference Expressions

Text regions that match some NL pattern from those in the grammar of Fig. 10 will be marked as CREs. The CREs then

Example 1
CRE: current article
Context: article 4 paragraph 2
Interp.: article 4

CRE: this section
Context: section 7 subsection (1) clause (a)
Interp.: section 7

Example 2
CRE: following paragraphs
Context: article 122 paragraph 1
Interp.: paragraph 2, paragraph 2a, paragraph 3, paragraph 4

CRE: following paragraphs
Context: section 3 subsection (1)
Interp.: subsection 1, subsection 2, subsection 3, subsection 4, subsection 5, subsection 6, subsection 7, subsection 8

Example 3
CRE: same law
Prev. CRE: law of 8 june 1999
Interp.: law of 8 june 1999

CRE: that section
Prev. CRE: section 36.2
Interp.: section 36.2

Example 4
CRE: alinea 2, sub a
Interp.: alinea 2, letter a

Example 5
CRE: article 14, 61, 91 or 95
Interp.: article 14, article 61, article 91, article 95

CRE: sections 11, 12, 15, 16, 17, 33 and 34
Interp.: section 11, section 12, section 15, section 16, section 17, section 33 , section 34

Example 6
CRE: articles 99ter to 102
Context: Lux. Income Tax Law
Interp.: article 99ter, article 99quater, article 100, article 101, article 102

Example 7
CRE: paragraphs 1 to 3
Context: article 50bis, paragraph 4
Parent context: article 50bis
Interp.: paragraph 1, paragraph 2, paragraph 3
Note: First attempt in the context of article 50bis, paragraph 4 fails.
Second attempt at the level of article 50bis succeeds.

CRE: subsections (1) to (4)
Context: section 49, subsection 5
Parent context: section 49
Interp.: subsection 1, subsection 2, subsection 3
Note: First interpretation in the context of section 49, subsection 5 fails.
Second attempt at the level of section 49 succeeds.

Example 8
CRE: 1st alinea, sub d) of article 131
Interp.: article 131, 1st alinea, sub d)

CRE: subclause 21 (1) (e) (iii)
Interp.: section 21 subsection (1) clause (e) subclause (iii)

Example 9
CRE: articles 109, 1st alinea, numbers 1 to 3, 127 and 154ter
Interp.: article 109 alinea 1 number 1, article 109 alinea 1 number 1a, article 109 alinea 1 number 2, article 109 alinea 1 number 3, article 127, article 154ter

CRE: clauses 44 (6) (a) to (f)
Interp.: section 44 subsection 6 clause a, section 44 subsection 6 clause b, section 44 subsection 6 clause c, section 44 subsection 6 clause d, section 44 subsection 6 clause e, section 44 subsection 6 clause f

Fig. 11 Examples of Cross Reference Expressions and their Interpretation

undergo an interpretation step and are linked to the cited targets. In this section, we focus on interpreting and linking of internal cross references. Simple external CREs that mention only the name of the external text or the date the text was enacted can be resolved using a mapping from names and dates onto the resource locators for the texts. As for external CREs that refer to the internal provisions of an external text, interpretation is done in the same manner as that for internal CREs, once the name (or date) of the external text has been extracted from the CRE. Note that resolving such external CREs requires the cited external text to be in a markup format.

7.1 Interpreting Cross Reference Expressions

The aim of the interpretation phase is to translate each CRE into a set of individual cross references. The main complexity arising during interpretation is that some of the NL patterns discussed in Section 6 are *ambiguous*, i.e., several parse trees may exist for the same CRE. While a regular expression recognizer can delineate the start and end of each CRE even when the grammar is ambiguous, without knowing the structural markup of the underlying legal text, one cannot choose the parse tree that is suitable for the text. Parser generators such as Yacc [21] require static priorities to be defined in order to resolve ambiguities. This is inadequate for CREs, because the admissible parse tree depends on the context, i.e., the actual legal text under analysis.

Custom interpretation rules are thus necessary, as we detail in this section. To remain concise in our descriptions, we assume that the legal text under analysis has been already preprocessed. In particular, we assume that: (1) ordinals, roman numbers, and numbers spelled out in text have been replaced with arabic numerals; alphanumerics remain unchanged; (2) abbreviated structural element markers (e.g., art.) have been replaced with full labels (e.g., article).

Table 11 presents the CRE examples used in the remainder of this section for illustration. For each example, we provide (1) the CRE itself, (2) the CRE context, i.e., the precise location of the CRE, and when necessary the parent context, i.e., context at a higher hierarchical level, (3) where relevant, the immediately-preceding CRE at a certain hierarchical level, denoted Prev. CRE, and finally (4) the CRE interpretation.

7.1.1 Interpreting Simple Cross Reference Expressions

Among simple CREs, only implicit ones and those using generic terms (e.g., sub, current) need to be interpreted. We distinguish two cases for implicit CREs.

(1) Implicit CREs that are semantically equivalent to *current*, *previous*, or *next* followed by a structural element marker *C*, for instance, "current article" and "this section" in Example 1. These CREs are interpreted with respect to their precise context. In the case of *current*, the CRE is interpreted as referencing the segment of the same type as *C* containing the CRE, "article 4" and "section 7" in the case of Example 1. In the case of *previous* and *next*, e.g., "following paragraphs" in Example 2, the CRE is interpreted as referencing segment(s) of the same type as *C* that respectively precede or follow the CRE.

(2) Implicit CREs that are semantically equivalent to *same* or *that* followed by a structural element marker *C* such as "same law" or "that section" in Example 3. These CREs, commonly used to avoid repetition, need to be interpreted based on the preceding CRE. Specifically, we interpret such CREs as being equivalent to the closest CRE of type *C* which precedes the CRE in question.

We note that the interpretation in both of the above cases is a *best-guess* heuristic, as we do not interpret the semantics of the underlying text.

Interpreting generic terms such as sub in Example 4 needs to be done according to the conventions in the legal jurisdiction to which the text belongs. In Luxembourg's legislation, the specific structural element marker for a generic term can be inferred based on what is seen after the generic term. If the generic term is followed by a letter, the appropriate structural element marker is *Letter*; otherwise, the marker is *Number*. We did not observe such CREs in PHIPA.

7.1.2 Interpreting Multivalued Cross Reference Expressions

Multivalued CREs such as the ones in Example 5 are interpreted with the structural element marker added to each element of the enumeration. For example "article 14, 61, 91 or 95" will be treated as "article 14, article 61, ...".

When the CREs include ranges, e.g., as in Examples 6 and 7, we distinguish structural elements that have unique numbering across an entire legal text (e.g., *Article*) from structural elements (e.g., *Paragraph*) whose numbering is reset when a higher-level structural element is seen. For elements of the first type, we browse the entire hierarchical structure of the legal text to identify the elements in the range. For elements of the second type, the interpretation is similar but depends on the local context: We initially attempt to interpret the CRE within the innermost segment in the hierarchy where the CRE appears. If the CRE cannot be interpreted meaningfully within this context, we recursively attempt to resolve the CRE in the context of the (immediate) parent of the current segment and then in the context of the parent's parent and so on, until the right level for interpreting the CRE is reached. Recursive interpretation attempts are illustrated in Example 7.

Note that, as shown by Example 6, the actual elements of a range cannot be merely deduced by an integer enumeration because alphanumerics may be involved. For multivalued CREs including implicit terms, we apply the same process as that for simple implicit CREs, described earlier.

7.1.3 Interpreting Multilayered Cross Reference Expressions

For multilayered CREs that do not contain a multivalued segment, interpretation is performed by harmonizing the navigation order so that it is strictly downwards. To illustrate, consider the CRE "1st alinea, sub d) of article 131" in Example 8. The following four CREs are equivalent to this CRE: (1) "article 131, 1st alinea, sub d)", (2) "sub d) of article 131, 1st alinea", (3) "article 131 sub d) of 1st alinea". Only (1) is in harmonized (strictly downwards) form.

Multilayered CREs without multivalued segments may also come in another variation, e.g., "subclause 21 (1) (e) (iii)" in Example 8. This CRE follows a harmonized navigation order already, except that it starts with the structural element maker for the deepest hierarchical layer, leaving out all the other (intermediate) layers. Such CREs are harmonized by making explicit all the intermediate structural element markers. For example, "subclause 21 (1) (e) (iii)" will be transformed into "section 21 subsection (1) clause (e) subclause (iii)".

The most complex form of multilayered CREs are those in which layers are combined with multivalued parts (ranges and enumerations). The regular expressions that detect such CREs are *ambiguous*. To illustrate, consider the CRE "articles 109, 1st alinea, numbers 1 to 3, 127 and 154ter" in Example 9. Without knowing the structural organization of the underlying text, one cannot know whether "127 and 154ter" refer to articles, paragraphs, or numbers. One could take cues from punctuation and the singular versus plural structural element markers to rule out the fragment referring to paragraphs. One could further deduce that either "127" or "154ter" has to be an article because the article structural element marker is in plural form. Unfortunately, such reasoning is unreliable as punctuation and the use of singular versus plural terms are not consistently followed in legal texts. For example, the distinction between singular and plural disappears when abbreviations (e.g., art.) are used.

We interpret multilayered CREs with ranges and enumerations in a similar way to multivalued CREs. When faced with a CRE fragment whose type is unknown, an attempt is made to interpret that fragment in the deepest context previously used for interpretation. In the case of the above CRE in Example 9, this means that first, we take the numerical expression "127", whose type is unknown, to be the contin-

Example 10
(Combined) CRE: article 24, with the exception of paragraph 3
Regular CRE: article 24
Exception CRE: paragraph 3
Context (for Exception CRE): article 24
Interp. (Exception CRE): article 24 paragraph 3

(Combined) CRE: article 28, with the exception of letters h, k, p, r and s
Regular CRE: article 28
Exception CRE: letters h, k, p, r and s
Context (for Exception CRE): article 28
Interp. (Exception CRE): article 28 letter h, article 28 letter k, article 28 letter p, article 28 letter r, article 28 letter s

Example 11
CRE: article 1quinquies
Art. 2. After article 1quater, a new article 1quinquies is added with the following wording . . .
Context: Law of April 16th, 1979
Interp.: article 1quinquies of Law of April 16th, 1979

CRE: article 31.-1
Art. 36. Article 31.-1. is modified as follows:
Context: Law of April 16th, 1979
Interp.: article 31.-1 of Law of April 16th, 1979

Fig. 12 Examples of Exception and Amendment Cross Reference Expressions and their Interpretation

uation of "numbers 1 to 3". The algorithm tries to interpret "127" in the context of "article[s] 109, 1st alinea", i.e., assuming it is a number. If this attempt fails, we recursively switch to the upper-level context in the CRE, i.e., "article[s] 109" (assuming it is an alinea) and finally assuming it is an article, where the interpretation succeeds. Now that "127" has been interpreted as an article, the CRE will be seen as if the structural element marker article appeared just before "127" in the legal text. The remainder of the enumeration, i.e., "154ter", is interpreted as if the CRE is "articles 109, 1st alinea, numbers 1 to 3, **article** 127 and 154ter". We identified multilayered CREs with ranges and enumerations in PHIPA as well, e.g., "clauses 44 (6) (a) to (f)" in Example 9. The multilayered CREs in PHIPA are nevertheless structurally less complex than those found in Luxembourg's legislation.

7.1.4 Cross References Requiring Special Treatment

The interpretation rules described in Sections 7.1.1 to 7.1.3 provide the general behavior of the resolution process. There are nevertheless two special situations where this general behavior needs to be altered to increase the accuracy of the resolution process. These two situations are discussed below, with illustrative examples provided in Fig. 12.

Exception CREs. When citing one or a range of provisions, certain constituent parts or elements within the range may be excepted (excluded) from the scope of the citation. This situation is illustrated in Example 10. In such cases, the exception CRE, for instance, "paragraph 3" in Example 10, has to be interpreted in the context of the regular CRE, for instance, "article 24" in Example 10, rather than in the context of the actual provision where the exception CRE is located. The large majority (but not all) of Exception CREs are identifiable based on certain keyphrases, e.g., "with the exception of" which either precede or succeed the CREs.

Amendment CREs. Amendment CREs appear in provisions that prescribe modifications to other (external) legal texts. Two instances of Amendment CREs are shown in Example 11. Without additional processing, Amendment CREs will be treated as internal, although these CREs are always external. Amendment CREs need to be resolved in the context of the text being amended. For example, the resolution of the CREs of Example 11 must be attempted in "the Modified Law of April 16, 1979". Similar to Exception CREs, most Amendment CREs can be identified based on keyphrases preceding or succeeding them. Examples of these keyphrases include: "is added", "is modified", "a new", and "is repealed".

7.2 Linking Cross References to Targeted Provisions

Once the interpretation phase is complete, each CRE is linked to all the provisions resulting from its interpretation. The exact mechanism used for capturing the links depends on the type of markup in which the target texts are represented. For example, the links can be captured using xlink when the target texts are in XML format, and using hyperlinks (href) when the target texts are in HTML format. The technical details of establishing the links are straightforward and a matter of implementation, as the interpretation of each CRE leads to uniquely identifiable elements. An example of links for the case where the legal texts are rendered in HTML is provided in Section 8 when discussing visualization and navigation (Section 8.2).

8 Applications

In this section, we present some important use cases that build on the results of cross reference detection and resolution. An instantiation of these use cases for Luxembourg's Income Tax Law and PHIPA is available at:
`http://people.svv.lu/sannier/crossreferences/`

8.1 Identifying Unresolvable CREs

A natural byproduct of resolution are diagnostics about CREs that cannot be resolved. Failure to resolve a CRE is due to one of the following: (1) our automatic interpretation being at fault, (2) well-formedness issues in how the CREs are phrased, or (3) the citation targets of the CREs being non-existing. Independently of the cause, it is important for legal experts and for requirements analysts alike to be made aware of CREs that cannot be resolved.

Case 1 typically occurs when external CREs are erroneously deemed as being internal. For instance, there are certain circumstances where Amendment CREs (see Section 7.1.4) are difficult to identify as such because of the nuanced language of the law. An example would be the following: "The modified fiscal adjustment law of 16 October 1934 is amended with the following provision, inserted into the law as paragraph 11bis: [...]". Our approach would classify "paragraph 11bis" as being internal although the correct treatment would be "paragraph 11bis of the law of 16 October 1934". Unless a specific rule is written for this situation, the nuanced text that appears between the two CRE fragments makes it difficult to relate the two fragments. As suggested by the accuracy results in Section 10, such complex situations leading to erroneous interpretation are not common.

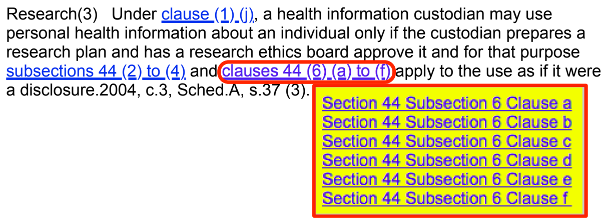
Case 2 occurs, for instance, when many hierarchy levels of a legal text are aggregated into a single numbering. An example is "Number 3e", when used as a shorthand for referring to "Number 3 Letter e". In this case, our algorithm is (legitimately) unable to resolve the CRE because the numbers that correspond to two successive levels of the hierarchy have been joined together without a blank space or separator between them.

Case 3 occurs when some target end of a CRE is dangling. An example would be "previous alinea" appearing in the first alinea of an article, i.e., where there is no previous alinea. Another example is when some hierarchical level is skipped, e.g., "Article 25 Alinea 7 Letter b" being erroneously written as "Article 25 Letter b". A CRE citing some provision whose content has been repealed (and removed) will also result in a non-existing target warning.

8.2 Visualization and Navigation

Cross reference detection and resolution is a prerequisite for generating navigable views of legal texts. As discussed previously, web portals such as LegiFrance and LegiLux already provide electronic versions for laws and navigable links. However, the markup upon which legal texts are built for these portals is not as precise. Moreover, implicit CREs and ranges are not adequately interpreted and resolved.

Our approach generates different views that can be used for different purposes. Fig. 13 shows a small excerpt of an HTML view of Subsection 37 (3) of PHIPA. In this view, the resolved CREs appear as hyperlinks. Clicking on a CRE brings up a tooltip box, allowing the user to navigate to any of the cross references entailed by the CRE. Such a view is useful during the elaboration of compliance requirements, when analysts often need to follow the cross references while looking for additional information.



Research(3) Under clause (1) (j), a health information custodian may use personal health information about an individual only if the custodian prepares a research plan and has a research ethics board approve it and for that purpose subsections 44 (2) to (4) and clauses 44 (6) (a) to (f) apply to the use as if it were a disclosure.2004, c.3, Sched.A, s.37 (3).

Section 44 Subsection 6 Clause a
Section 44 Subsection 6 Clause b
Section 44 Subsection 6 Clause c
Section 44 Subsection 6 Clause d
Section 44 Subsection 6 Clause e
Section 44 Subsection 6 Clause f

Fig. 13 HTML View of PHIPA with Cross References Rendered as Hyperlinks

In addition to hypertext content, our approach can provide alternative views on legal texts, such as an interactive tree to facilitate the navigation within a law. An example of such a view is shown in Fig. 14 where each structural element is rendered as a tree node. The tree view is useful to explore the detailed hierarchy of the legal text with subnodes representing the hierarchical structure of the legal text and color-coded nodes being internal (green) or external (red) cross references.

8.3 Advanced Text Search

Another interesting application enabled by our approach is the capability to perform advanced cross-reference search, taking into account implicit and multivalued cross references. As an illustration, consider the following example from Luxembourg's Income Tax Law: "Article 24" of this law elaborates the pension schemes recognized for taxation. A natural query for an analyst who is elaborating the compliance requirements for taxation of pensioners would be: *Where is "Article 24" cited?* A naive lookup of the string "Article 24" in the law's text yields no results, despite the article being internally cited in four places, within ranges: "Articles 4 to 155bis", "Articles 14 to 108bis" and "Articles 16 to 60" (appearing twice). A similar example from PHIPA is "Subsection 44 (3)", which is cited within two ranges "Subsections 44 (2) to (4)" and "Subsections (1) to (4) ". For both examples, without automation, identifying where the given provisions are being cited requires a manual inspection of the entire text.

8.4 Trace Link Analysis

Trace link analysis is concerned with identifying the provisions and artifacts that refer to a particular provision. Without automated trace link analysis, it would be difficult to determine how a change in a given law impacts related laws and also artifacts such requirements, websites, and forms.

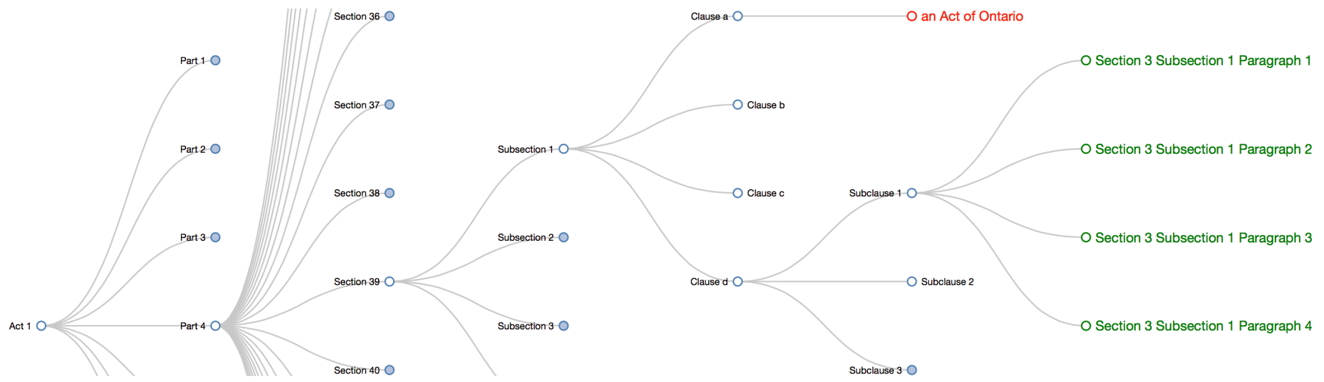


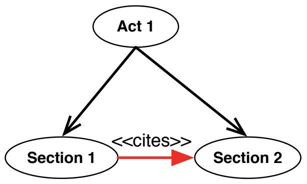
Fig. 14 Tree-like View of a Legal Text (Fragment)

Once the structure of a legal text has been extracted and its cross references have been resolved, link analysis can be done through logical means. We use the Relational Manipulation Language (RML) [4] for formulating link analysis. RML provides the expressive power of first-order logic with transitive closure and counting operators. Being able to compute transitive closure is essential both because of the transitive nature of links between legal provisions and also the hierarchical nesting of document elements. The use of RML is motivated by RML's simple syntax and its efficient interpreter, CrocoPat [4]. CrocoPat encodes relational predicates as Binary Decision Diagrams (BDDs) [7], making it scalable for handling large legal texts with potentially thousands of structural elements and cross references.

To use CrocoPat for link analysis, we need to construct a predicate database capturing the structure of the legal text in question along with its cross references. This information is conveniently expressed as a *typed graph* [27] – intuitively, a graph whose nodes and edges are typed. In our problem, graph nodes represent instances of the structural elements in a legal text, e.g., individual sections and paragraphs. Node types therefore correspond to the classes in the text schema, e.g., the schema of PHIPA in Fig. 7. Edges represent two types of relationships: (1) a containment relationship between structural elements, e.g. "Section 12" *contains* "Section 12 Subsection 3"; (2) a citation relationship between structural elements, e.g. "Section 12 Subsection 3" *cites* "Section 44 Subsection 1".

In Fig. 15(a), we show a small fragment of the typed graph for a legal text (PHIPA). Each node has a label:type annotation. The label portion of the annotation is the qualified name of the element that the node represents. Edges are marked only with types and without labels. The algorithm for transforming a typed graph into RML is straightforward – see [28]. Fig. 15(b) shows the resulting RML predicates for the typed graph of Fig. 15(a). To ensure that each graph element is uniquely represented, the translation assigns a unique *uid* to each node and edge.

(a)



(b)

```
Node("id0");                          Edge("id2");
Label("id0","Act1");                  Type("id2","Containment");
Type("id0","Act");                    Src("id2","id0");
                                      Tgt("id2","id1");
Node("id1");
Label("id1","Act 1::Section 1");      Edge("id4");
Type("id1","Section");                Type("id2","Containment");
                                      Src("id4","id0");
Node("id3");                          Tgt("id4","id3");
Label("id3","Act 1::Section 2");
Type("id3","Section");                Edge("id3192");
                                      Type("id5","Citation");
                                      Src("id5","id1");
                                      Tgt("id5","id3");
```

Fig. 15 (a) Example Typed Graph (b) RML Predicates for the Graph

Given a predicate database for a legal text in the format shown in Fig. 15(b), one can infer links between any pair of structural element instances. For example, one can identify, for each Section X, all structural element instances that directly link to X via a cross reference. The RML code snippet for this computation is shown in Fig. 16. In the snippet, we first compute a relation, Contains(x,y), that holds for all (x,y) where y is a child of x in the legal text's hierarchy tree. TrContains(x, y) computes the reachability relations via containment using the transitive closure operator (TC). TrContains(x, y) thus holds for all (x,y) where y is a descendant of x. Cites(x,y) computes (x,y) where x directly cites y via a cross reference. Finally, LinkedToArt(x, y) computes all (x,y) where y (i.e., the link target) is of type *Section*, and where x cites some element z that is transitively contained in y (e.g., a subsection of y, or a paragraph in a subsection of y). Results of link analysis can be used for creating a traceability table. An excerpt of the traceability table for PHIPA built based on the above snippet is shown in Fig. 17.

```
Contains(x, y)     :=    Node(x) & Node(y) & EX(e, Edge(e) &
                         Type(e, "Contains") & Src(e, x) & Tgt(e, y);
TrContains(x, y)  :=     TC(Contains(x,y)) | (x = y);
Cites(x, y)        :=    Node(x) & Node(y) & EX(e, Edge(e) &
                         Type(e, "Cites") & Source(e, x) & Target(e, y));
LinkedToArt(x, y) :=     Node(x) & Type(y, "Section") &
                         EX(z, Cites(x, z) & TrContains(y, z));
```

Fig. 16 RML Snippet for Trace Link Analysis

Part 4::Section 50	**0 citations:**
Part 5::Section 51	**2 citations:** • Part 5::Section 51::Subsection 2 • Part 5::Section 51::Subsection 4
Part 5::Section 52	**5 citations:** • Part 1::Section 2::Subsection 1 • Part 5::Section 52::Subsection 2 • Part 5::Section 52::Subsection 3 • Part 5::Section 52::Subsection 4 • Part 7::Section 72::Subsection 1::Clause c::Subclause 4

Fig. 17 Excerpt of PHIPA Traceability Table

8.5 Circularity Analysis

Cyclic citations are common in legal texts. A frequent type of usage is when a provision X cites a provision Y to state that X depends on Y for a definition; and Y refers back to X to state that Y provides a definition required by X. While cycles seldom indicate errors, they need to be investigated carefully to verify the absence of circular reasoning, e.g., cases where provisions X and Y both depend on each other for a definition.

Circularity analysis is performed using logical queries similar to those for trace link analysis. The RML snippet for detecting cycles of length two is given in Fig. 18. Cycles of longer lengths can be computed in an analogous manner.

```
XCitesY(x, y) :=    Node(x) & Node(y) & EX(e, Edge(e) &
                    Type(e, "Citation") & Source(e, x) & Target(e, y));
C2(a1,a2)     :=    (a1 < a2)
                    & XCitesY(a1,a2) & XCitesY(a2,a1) & (a1 != a2);
```

Fig. 18 RML Snippet for Circularity Analysis

In the snippet, we define the predicate for a citation between two nodes x and y. XCitesY(x, y) computes a relation for two nodes x and y, where x is the source of an edge e of type "citation", and y it the target of the edge. The relation C2(a1,a2) detects any cycle of length two between distinct nodes $a1$ and $a2$ of the graph, with $a1$ citing $a2$ and $a2$ citing $a1$. The $(a1 < a2)$ is for symmetry breaking so that a cycle is not detected and presented twice.

```
Part 1::Sec. 3::Subsec. 5   → Part 1::Sec. 3::Subsec. 6   → self
Part 1::Sec. 3::Subsec. 7   → Part 1::Sec. 3::Subsec. 8   → self
Part 4::Sec. 32::Subsec. 1  → Part 4::Sec. 32::Subsec. 2  → self
Part 4::Sec. 40::Subsec. 2  → Part 4::Sec. 40::Subsec. 3  → self
Part 4::Sec. 44::Subsec. 10 → Part 4::Sec. 44::Subsec. 11 → self
Part 4::Sec. 45::Subsec. 1  → Part 4::Sec. 45::Subsec. 3  → self
Part 5::Sec. 54::Subsec. 2  → Part 5::Sec. 54::Subsec. 3  → self
Part 7::Sec. 74::Subsec. 1  → Part 7::Sec. 74::Subsec. 7  → self
```

Fig. 19 Circular Citations of Length Two in PHIPA

The cyclic citations of length 2 in PHIPA are shown in Fig.19. For each line, the first provision cites the second, and the second cites back the first. We investigated these cycles and determined that all the cycles were following the same pattern where the source provision states that more details or exceptions will be provided in the (cited) second provision. The second provision cites back the first as a reminder for the initial context.

9 Tool Support

Buidling on the GATE Workbench, we implement our approach into a tool named LeCA (**Le**gal **C**ross Reference **A**nalyzer). LeCA provides automated support for (1) generation of text structure markup (Section 5), (2) cross reference detection (Section 6) and resolution (Section 7), and (3) visualization and cross reference analysis (Section 8).

Fig. 20 shows an overview of LeCA. Eclipse's model-to-text transformation facilities are used in order to derive, from a text schema, scripts for text structure markup. These scripts are then executed by GATE, followed by those for cross reference detection and resolution. The cross reference detection and resolution scripts rely on gazetteers for structural element markers and domain-specific terms. The gazetteers for structural element markers are derived from the text schema. The gazetteers for domain-specific terms, including the names of legal texts and the implicit terms, depend on the corpus to which the legal text belongs and the writing language of the text. These gazetteers thus need to be provided by the user.

As output, LeCA produces an HTML view of the input legal text with cross references represented as hyperlinks. Diagnostics are further provided for any unresolved cross references. LeCA additionally generates a logical representation of the input text's structure and cross references, which is in turn fed to CrocoPat for analysis.

LeCA has been developed primarily in JAPE. This has enabled us to seamlessly integrate cross reference detection

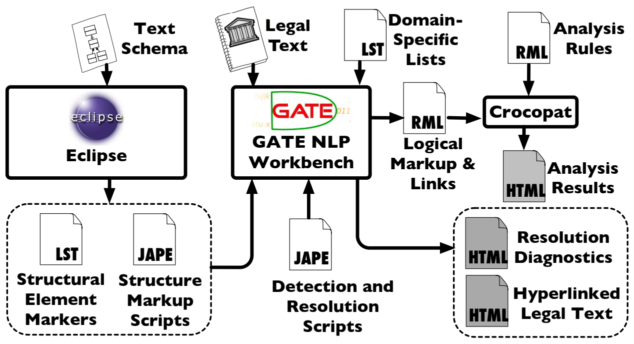

Text Schema
Legal Text
LST Domain-Specific Lists
RML Analysis Rules
Eclipse
GATE NLP Workbench
RML Logical Markup & Links
Crocopat
HTML Analysis Results
LST Structural Element Markers
JAPE Structure Markup Scripts
JAPE Detection and Resolution Scripts
HTML Resolution Diagnostics
HTML Hyperlinked Legal Text

Fig. 20 LeCA Tool Architecture

and resolution activities. The JAPE rules are augmented with Java code for the interpretation and linking activities as well as for the generation of HTML views and predicate databases for interacting with CrocoPat. Excluding comments and third-party components, LeCA consists of 114 JAPE scripts with approximately 13K lines of JAPE code. LeCA includes approximately 5K lines of additional Java code, providing various functions that are used within the JAPE scripts.

10 Evaluation

In this section, we report on an evaluation of our approach based on selected Luxembourgish legislative texts and also PHIPA from the Canadian legal corpus. The evaluation is aimed at investigating the effectiveness and scalability of the approach. We start this section with a description of our Research Questions (RQs). We then present our evaluation results, followed by a discussion of limitations and threats to validity.

10.1 Research Questions

Our evaluation is targeted at answering the following research questions (RQ).

RQ1. Is our approach effective at identifying CREs? This RQ aims at evaluating the completeness of our natural language patterns for CREs by analyzing how accurately the patterns can detect CREs in Luxembourg's legislative texts other than the Income Tax Law.

RQ2. Is our approach effective at resolving CREs? This RQ aims at measuring how accurate our approach is in resolving already-detected CREs.

RQ3. How scalable is our approach? Legal texts can be hundreds and sometimes thousands of pages long. This RQ aims at establishing whether our approach runs within reasonable time.

Table 1 Results for RQ1

CRE Type	# of CREs	Correctly Identified	Partially Identified	Missed
Internal	857	848	8	1
External	995	965	30	0
Explicit	1389	1350	38	1
Implicit	463	463	0	0
Simple	1031	1029	1	1
Complex	821	784	37	0
– *Mutlivalued*	373	372	1	0
– *Multilayered*	448	412	36	0

10.2 Evaluation Results and Discussion

Below, we present and discuss our evaluation results for each of the three RQs stated in Section 10.1.

10.2.1 RQ1: Is our approach effective at identifying CREs?

To answer RQ1, we selected 13 legislative texts with a total of 1640 pages from Luxembourg's legal corpus. We randomly chose 10% of the pages in each selected text. If a randomly-chosen page coincided with the preface, table of contents, document history, or index, the page was discarded and another random page was considered. In total, we considered 164 pages of text containing actual legal provisions. We conducted a manual inspection of these pages and highlighted the CREs found. This inspection yielded 1852 CREs.

Following the inspection, we applied our tool for detecting the CREs in these pages. The tool was applied exclusively for detection, i.e., structural markup generation and resolution were not performed. For detection, we used the structural element markers (L. 4 of the patterns in Fig. 10) prescribed for legislative texts by Luxembourg's legal writing best practices that are in effect today [3]. For generic terms, law names and auxiliary terms (respectively, L. 7, 13 and 15 of the patterns), we exploited the lists built from our investigation of the Income Tax Law. Table 1 summarizes the results for RQ1. In the table, we classify the identified CREs across three different dimensions: *Internal vs. External*, *Explicit vs. Implicit*, and *Simple vs. Complex*.

The results indicate that our patterns miss only one CRE among the ones investigated (less than one tenth of a percent). This CRE was at (1) (in French, au (1)), which referred to paragraph 1 of the article in question. The CRE can be detected by adding "at" to the generic terms (L. 7 of the patterns). However, we chose to not include this pattern because in French it is common to use this preposition followed by a number for reasons other than making a citation. Hence including the pattern could result in several false positives.

38 CREs ($\approx 2\%$) were only partially identified. The majority of these were either external or multilayered internal. The partial detection of external CREs was explained primarily by incompleteness in the lists of law names and auxiliary terms. These were names and terms that were not

present in the Income Tax Law, and thus not included in our gazetteers. As for the multilayered internal CREs, partial detection was explained mainly by incompleteness in the list of structural element markers, arising from the use of markers other than the ones stated in the best practice [3]. Detection further yielded five false positives (not shown in the table).

Without addressing the incompleteness in the lists, detection has a precision of 99.7%, recall of 97.9% and F-measure of 98.8%. If the lists are completed, these measures will respectively be: 99.7%, 100% and 99.8%. No new patterns emerged from our investigation in RQ1.

10.2.2 RQ2: Is our approach effective at resolving CREs?

We answer RQ2 based on an analysis of seven legal texts. Table 2 lists these selected texts, along with some of their characteristics, including the number of articles, the total number of CREs, and the number of internal CREs. The number of unspecific CREs (within the internal ones) is also shown.

In addition to the first five texts (T1–T5) which come from Luxembourg's legislation, we further selected PHIPA in both English and French (T6 and T7). The shortest text in our study of RQ2 is T3, which is seven pages long, with sixteen articles and 79 CREs. 45 of these CREs are internal with five being unspecific. The longest text is T5, which is 189 pages long, contains 236 articles, and has 1223 CREs of which 928 are internal, with 45 of these being unspecific.

T5, T6, and T7 were previously used in the derivation of our CRE patterns (Section 6). Despite this, we believe that using these three texts towards answering RQ2 is justified because of the following: First, from our analysis of RQ1, one can be reasonably confident that our patterns achieve high coverage in detection. Second, our resolution algorithm is instantiated based on a schema and the CREs patterns, irrespectively of the actual legal text. We thus anticipate little bias resulting from using these three texts in RQ2.

Although our patterns address both internal and external CREs, our evaluation of RQ2 is exclusively concerned with internal CREs, after excluding unspecific ones. As noted earlier, a detailed resolution of external CREs requires the text schema for the cited external texts. With regards to unspecific CREs, and again as noted before, meaningful automated resolution would be hard because of these CREs being vague. Manual interpretation would thus be necessary.

Overall, the seven analyzed texts contain 4474 CREs, of which 2595 are internal. Among the internal CREs, 71 are unspecific thus leaving 2524 CREs to resolve. We evaluate the accuracy of resolution over these 2524 CREs using precision and recall. Table 3 summarizes the results. When combined, the automatically-generated cross reference links and the warnings fully covered the internal CREs over which resolution was attempted. Column 3 of Table 3 gives the number of individual cross reference links generated for the CREs resolved (column 2) in each text (column 1). The algorithm returns several warnings about CREs that have not been resolved (column 4). Our inspection of the unresolved CREs indicated that they were either *anomalous* or *misinterpreted*.

Anomalous CREs are related to well-formedness issues and non-existing targets, i.e., cases (2) and (3) outlined in Section 8.1. Misinterpreted CREs, as the name suggests, result from our algorithm interpreting the CREs incorrectly. The consequence of a misinterpretation could be either of the following: (1) resolution fails, i.e., case (1) in Section 8.1, or (2) the resolution links the CRE to the wrong provision(s). We did not encounter the latter situation in our evaluation; nevertheless, such a situation is possible. For example, if a reference to “article 1” of some external text is incorrectly classified as being internal, the reference will most likely be resolved but incorrectly.

With regards to the accuracy of resolution, metrics are provided in columns 4 to 8 of Table 3. In our evaluation, we consider anomalous CREs as being true positives. Misinterpreted CREs are rare. We observed a maximum of three misinterpreted CREs per text, as indicated by column 4 of Table 3. Internal CREs correctly classified as being such but nevertheless misinterpreted give rise to both false positives and false negatives. Internal CREs classified as being external give rise only to false negatives. External CREs classified as being internal give rise only to false positives. External CREs classified as external are out of the scope of RQ2 as resolution was not attempted over external CREs.

Below, we elaborate the misinterpreted CREs in the studied texts. T1–T3 did not contain any misinterpreted CREs. For T4 and T5, the misinterpreted CREs are external references being interpreted as internal ones, thus affecting only precision. The example we gave earlier for case (1) in Section 8.1 (“paragraph 11bis”) is the misinterpretation seen in T5. T6 and T7 contain one misinterpreted CRE that is common between the two texts. This CRE relates to the title of subsection 26. (7): “Conflict between persons in same paragraph”. Upon a manual examination of the content of the subsection, we determined that “same paragraph” may refer to *any* paragraph in subsection 26 (1). The CRE must thus be resolved to paragraph 26. (1) 1, paragraph 26. (1) 2, . . . , paragraph 26. (1) 8. Our algorithm nevertheless fails to resolve this CRE. In our our calculation of recall, this situation counts as eight false negatives rather than just one.

As for unspecific CREs, while their numbers vary across the texts, they do not generally make up for a sizable fraction of the CREs (average of 3.4%). The highest observed percentages are for T3 (12.5%) and T5 (5.1%). T3 is a small text and hard to draw conclusions from; T5 (the Income Tax Law) is rather special in that it, on multiple occasions, cites (unspecified) regulations both to reduce complexity and fur-

Table 2 Selected Texts for RQ2

Text id	Law Name	1st Date of Publication	# of pages	# of Articles	# of CREs	# of internal CREs (unspecific)
T1	Law of August 2, 2002	August 2, 2002	23	45	312	187 (4)
T2	Law of June 30, 2003	June 30, 2003	18	102	175	118 (0)
T3	Law of May 30, 2005	July 14, 1845	7	16	79	45 (5)
T4	Law of June 25, 2009	June 25, 2009	23	92	296	208 (3)
T5	Law of January 1st, 2013	December 4th, 1967	189	236	1223	928 (45)
T6	PHIPA En	2004	49	75	1197	557 (7)
T7	PHIPA Fr	2004	49	75	1192	552 (7)
		Total:	358	641	4474	2595 (71)

Table 3 Results for RQ2

Law Name	# of CREs Resolved	# of Individual Cross Reference Links	# of Unresolved CREs (Anomalous, Misinterpreted)	Precision	Recall	F-measure	# of unspecific CREs (% of these CREs)
T1	183	226	(0, 0)	100%	100%	100%	4 (2.19%)
T2	118	145	(1, 0)	100%	100%	100%	0 (0%)
T3	40	65	(1, 0)	100%	100%	100%	5 (12.5%)
T4	205	288	(2, 3)	98.98%	100%	99.49%	3 (1.46%)
T5	883	1736	(8, 1)	99.94%	100%	99.97%	45 (5.1%)
T6	550	749	(3, 1)	99.87%	98.95%	99.41%	7 (1.27%)
T7	545	744	(3, 1)	99.87%	98.94%	99.40%	7 (1.28%)

ther to minimize changes to the text of law as the detailed taxation procedures (specified in the regulations) evolve.

Overall, when excluding unspecific CREs, the lowest level of accuracy observed is over T7 with an F-measure of 99.40%. The results thus indicate that our approach is highly accurate for cross reference resolution.

10.2.3 RQ3: How scalable is our approach?

We report on the execution time of our approach as measured on a laptop with a 2.3 GHz Intel CPU and 8GB of memory.

Cross reference detection took approximately 34 seconds over the 164 randomly-selected pages of RQ1 (1852 CREs) and approximately 15 seconds over T5 – the largest individual legal text in our study (1223 CREs).

The overall execution time of our approach is dominated by the resolution phase. For T5, it took approximately 151 seconds to interpret the CREs and a further 139 seconds to generate the cross reference links. The interpretation step has embedded into it the transformation of plain text to XML.

In Table 4, we present the execution times for the seven texts previously discussed in Tables 2 and 3. Since the resolution phase is the primary contributing factor to execution time, we further present in Fig. 21 a chart that shows the number of generated cross reference links (Y axis) in relation to execution time (X axis). As suggested by the chart, an almost-linear relationship is observed between the number of generated links and execution time. Given the short overall running time of our approach and the above linear relationship, we expect the approach to be scalable to large legal corpora.

Table 4 Results for RQ3

Text	Execution Time (in seconds)
T1	25
T2	25
T3	3
T4	79
T5	305
T6	90
T7	90

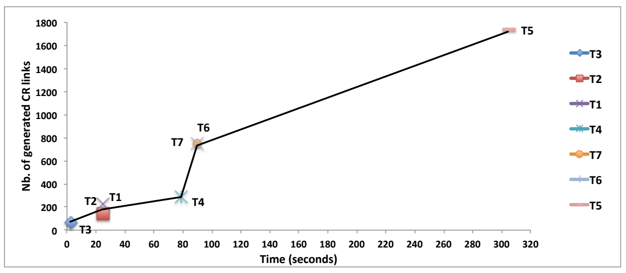


Fig. 21 Number of Generated Cross Reference Links in Relation to Execution Time

10.3 Limitations and Threats to Validity

In this section, we first discuss the limitations of our technical approach as well as those of our empirical evaluation. We then analyze the pertinent threats to validity.

Technical limitations. An issue with any rule-based approach, including ours, for cross reference detection and resolution is that one can never be entirely sure about the completeness of the rules. Furthermore, there will always be exceptional cases. For example, a text may deviate from its originally-intended schema in certain parts. Such deviations can make cross references to the structurally-inconsistent parts unresolvable. Another exceptional case are Amendment CREs. These CREs refer to provisions that have not been incorporated into the amended texts at the time the amendments are being proposed. The CREs will thus be unresolvable for as long as the amendments have not been applied into the target texts. Despite these technical limitations, we believe that our approach is worthwhile as we did not observe in our evaluation instances of incompleteness and exceptions that would significantly hamper the accuracy of our approach.

Evaluation limitations. The most conclusive evaluation of our approach would be to investigate whether requirements analysts dealing with legal requirements in realistic settings find our approach beneficial. Such an evaluation particularly has to tackle the following two questions: (1) Does our approach make it easier for analysts to extract requirements from legal texts? and (2) Does our approach improve the quality of requirements extraction from such texts? We do not address these questions in this current article, leaving them to future user studies.

External Validity. We discuss external validity separately for cross reference detection and cross references resolution.

With respect to cross reference detection, external validity has to do with how confident we are about the completeness of our patterns (extracted using grounded theory). While some degree of incompleteness in the patterns is to be expected due to the rule-based nature of our approach (discussed above), we did not observe new patterns when applying our approach to several texts from Luxembourg's legislation, nor when moving from the the Luxembourgish corpus to PHIPA. Although our results are encouraging, wider studies are necessary for building confidence about our patterns being reasonably complete.

The second dimension of external validity concerns the accuracy of cross reference resolution and whether the observed accuracy levels would carry over to other contexts. As we discussed and illustrated over the course of the article, cross reference resolution requires knowledge of the schema for the text being analyzed. Taking our approach from one legal jurisdiction or country to another will thus necessarily involve the development of new and customized schemas. In terms of customization, what we observed in the Luxembourgish context is that with more texts analyzed, a state of saturation was reached, where the analysis of a new text would simply imply the selection of an already-developed schema with very slight adjustments. While further evaluation is necessary, we believe this observation should hold in other countries and jurisdictions, thus making our approach worthwhile once the structure of a small but representative set of legal texts has been analyzed and modeled.

Internal Validity. We measured the accuracy of our approach using precision and recall, which require a gold standard. As the texts we analyzed were never provided with an adequate level of detail concerning their cross references, the first two authors built the gold standard by manually inspecting the texts to identify the CREs and later to verify which cross reference links were correct, incorrect, or missing. To ensure the quality of the gold standard and minimize errors, the first two authors did the manual inspections independently and then cross-checked the results.

Construct Validity. Our analysis is based on classification accuracy metrics (precision and recall) and scalability in terms of execution time. A more comprehensive evaluation of our approach will further have to consider cost-effectiveness, particularly in terms of the effort spent on tailoring our approach against the effort that is saved by the automatic detection and resolution of cross references. A rigorous analysis of cost-effectiveness is a topic that we plan to address in future work.

11 Related Work

Identifying and resolving cross references in legal texts is part of the more general problem of requirements traceability. Requirements traceability is commonly defined as "the ability to describe and follow the life of a requirement, in both a forwards and backwards direction" [16]. Our work is a step towards more automated management of traceability between (legal) requirements and legal texts. Below, we compare our approach with several strands of related work.

Siena et al. [30] and Ingolfo et al. [20] study variability in the law in terms of derogations and exceptions. As part of their work, they propose a formal language for the analysis of CREs and identifying the conditional structure of the law. The above work nevertheless does not address automatic detection and resolution of cross references – the topic that our article concentrates on.

Zeni et al. [34] consider the full problem of automated discovery and annotation of legal concepts in legal texts, including, among several other concepts, cross references. With regards to cross references, their approach uses structural markup based on a generic schema, along with patterns and rules for cross reference detection and interpretation. However, since the scope of the above-cited work is broader than ours, the work does not precisely detail how the detection and interpretation of cross references is performed, nor does it show empirical results dedicated to cross references.

Breaux et al. [5,6] identify natural language patterns for cross references based on a study of 118 expressions across three US regulations. They propose the use of an explicit schema for modeling the structure of legal texts. Similar to this earlier work, we study several legal texts for identifying the patterns in cross references. We nevertheless study a much larger set of cross reference expressions. We observe new patterns that were not seen in the US regulations considered. Our study also covers patterns for external cross references, which were not considered by Breaux et al. Additionally we propose automation for text structure markup.

In the context of compliance, many threads of work propose to formalize the structure of legal texts. Among others, Sannier and Baudry [29] propose a generic metamodel to structure the requirements derived from safety regulations and standards along with the relationships between these requirements. Emmerich et al. [13] propose a high-level schema to describe software quality standards in terms of requirements properties, rationale, and development policies as step towards managing compliance with standards during the software development lifecycle. These threads consider traceability from a general perspective. However, they do not specifically address cross references.

The closest study to ours in terms of cross reference patterns is by de Maat et al.'s [11] who study the patterns used in the cross references that appear in the Dutch laws. The differences in language aside, the patterns we observe in our investigation of Luxembourgish and Canadian legal texts are closely aligned with those in the Dutch laws. In this sense, our study serves as a confirmatory measure for the generalizability of previously-observed patterns. In addition, we identify important variations of these patterns. The main contributions of our work over De Maat et al.'s are: They assume that legal texts are already in a markup format with adequate structure to be transformed into the markup format required by their approach (MetaLex [19]). Our approach, in contrast, does not require pre-existing markup. Second, and more importantly, de Maat et al. do not clearly distinguish the activities of cross reference detection and resolution. They do not elaborate the resolution process, nor do they address the effectiveness of resolution in their evaluation. We instead provide a detailed treatment of resolution and measure its effectiveness in our evaluation.

Palmirani et al. [26] define cross reference patterns based on guidelines for the Italian legal corpus and apply their approach to several legal texts. However, they tackle only cross reference detection and not resolution. Their approach does not address the generation of markup documents and their patterns are insufficient for recognizing many of the rich patterns seen both in our study and that of the Dutch laws ([11]).

Hamdaqa et al. [18] propose an approach for resolving external cross references and report on a case study of three US regulations involving 122 (external) cross reference expressions. They use finite state machines for defining patterns, based on the recommendations of the US Bluebook [2] and the ALWD Citation Manual [12]. Their patterns are limited, first in that they apply only to external cross references, and second in that they are exclusively based on best practices and thus insufficient for the richer citation styles used in actual legal texts. Hamdaqa et al. consider automated markup generation through manually-written regular expressions. Our approach provides a more thorough and flexible framework than Hamdaqa et al's. Our patterns encompass both internal and external cross references, and further are based on studying actual legal text. Our approach is parameterized by a schema, which enables us to automatically derive the necessary regular expressions for text markup generation.

Tran et al. [33] apply machine learning for cross reference detection and resolution in Japanese legislative texts. Similar to them, we distinguish detection and resolution activities. However, our approach differs in that both our detection and resolution strategies are algorithmic and based on rules. Using machine learning can be advantageous in that it does not require an a-priori specification of the patterns in cross references. However, Tran et al. do not consider advanced patterns with recursive structures or multiple layers similar to those identified and addressed in our study (Section 6). It is unknown how such patterns can be effectively handled through learning. For the patterns they consider in their study, an accuracy (F-measure) of approximately 80% is reported for cross references detection and an accuracy of 67% for detection and resolution of cross references. This, compared to rule-based techniques, is low (see Section 10.2).

Several commercial services such as LexHub[6], Jureeka[7], QuickLaw[8] exist for legal citations. LexHub is a legal citation index manager that inserts cross reference links between provided citations in input documents and legislative texts or court decisions from the CanLII (Canadian Legal Information Institute) database. The input document must be provided in HTML format. Quicklaw is a legal search engine and a citation checker. It creates links from citations in an input document to the cited provisions in Canadian cases. Jureeka is a free plug-in for Mozilla and Chrome browsers, enabling the creation of cross reference links in web pages for citing US legal texts. The plugin uses regular expressions for linking explicit CREs to subsections in the US Code or to sections in the Code of Federal Regulations. The above tools are meant primarily at dealing with simple external cross references, with limited support for the detection and resolution of complex internal cross references.

Our work on cross reference analysis (Section 8) is close to that of Nentwich et al.'s [25] on consistency checking of XML documents using the xlinkit tool. xlinkit offers a rule-based language based on first-order logic and XPath (a query language for XML) for defining invariants over structured, hyperlinked documents and generation of various diagnostics. The xlinkit rule language is highly expressive and capable of capturing all the logical rules that underlie our analysis of legal CREs. xlinkit can thus be used as an alternative to the logical interpreter we use in our work.

[6] `https://lexum.com/en/resources/lexhub`

[7] `https://addons.mozilla.org/En-uS/firefox/addon/jureeka-6636/`

[8] `http://www.lexisnexis.ca/`

12 Conclusion and Future Work

In this article, we presented an approach for automatic detection and resolution of cross references in legal texts. Our approach complements existing work in a number of ways. In particular, the approach is parameterized by a text schema, making it possible to tailor the approach to different legal texts and jurisdictions. The use of schemas has further allowed us to automatically construct the structural markup that is necessary for resolving cross references. Through a study of selected Luxembourgish and Canadian legal texts, we derived natural language patterns for cross references, and provided a systematic way to interpret them. We outlined the implementation of our approach in a Natural Language Processing environment. Finally, we evaluated our approach in terms of effectiveness and scalability. Our evaluation suggests that the accuracy of our approach is high and that the running time for our automated tool chain scales to realistic settings.

For future work, we plan to evaluate our approach in terms of cost-effectiveness and also outside the legal contexts considered so far. Another interesting topic is the ability to annotate cross references with semantic information. In particular, interesting work, e.g., by Maxwell et al. [24] and Hamdaqa et al. [18], already exists on semantic taxonomies for cross references. These taxonomies provide a classification of the intent behind cross references, including providing definitions, imposing additional constraints and making exceptions. Not all cross references have the same level of impact on software requirements. Subsequently, automated classification of cross references based on their semantic intent would be a valuable next step for supporting legal compliance analysis.

Acknowledgments

Financial support for this work was provided by Luxembourg's National Centre of Information Technologies (CTIE) and Luxembourg's National Research Fund (FNR) under grant number FNR/P10/03. We are grateful to members of Luxembourg Inland Revenue Office (ACD) and CTIE, particularly, Thierry Prommenschenkel, Ludwig Balmer, Marc Blau, and Michael Masseroni for sharing their valuable knowledge and insights with us. We thank the anonymous reviewers of the RE Journal and the RE'14 conference for their insightful comments which helped us improve this article.

References

1. Morayo Adedjouma, Mehrdad Sabetzadeh, and Lionel C. Briand. Automated detection and resolution of legal cross references: Approach and a study of Luxembourg's legislation. In *Proceedings of the IEEE 22nd International on Requirements Engineering Conference, RE'14*, pages 63–72, 2014.
2. Linda J Barris. *Understanding and Mastering the Bluebook: a Guide for Students and Practitioners*. Carolina Academic Press, Durham, N.C., 2010.
3. Marc Besch. Traité de légistique formelle, 2005.
4. Dirk Beyer, Andreas Noack, and Claus Lewerentz. Efficient relational calculation for software analysis. *IEEE Transactions on Software Engineering*, 31(2):137–149, 2005.
5. Travis Breaux and Annie Antón. Analyzing regulatory rules for privacy and security requirements. *IEEE Transactions on Software Engineering*, 34(1):5–20, January 2008.
6. Travis Durand Breaux. *Legal Requirements Acquisition for the Specification of Legally Compliant Information Systems*. PhD thesis, North Carolina State University, Raleigh, North Carolina, USA, April 2009.
7. Randal Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 8:677–691, 1986.
8. Juliet Corbin and Anselm Strauss. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. SAGE Publications, 3rd edition, 2008.
9. Thomas H Cormen. *Introduction to Algorithms*. The MIT Press, Cambridge, Masachusetts; London, 2009.
10. Cunningham et al. Developing Language Processing Components with GATE Version 7 (a User Guide).
11. Emile de Maat, Radboud Winkels, and Tom van Engers. Automated detection of reference structures in law. In *Proceedings of the 2006 Conference on Legal Knowledge and Information Systems*, pages 41–50, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.
12. Darby Dickerson and Association of Legal Writing Directors. *ALWD citation manual: a professional system of citation*. Aspen Publishers, New York, NY, 2006.
13. Wolfgang Emmerich, Anthony Finkelstein, Carlo Montangero, Stefano Antonelli, Steve Armitage, and Richard Stevens. Managing standards compliance. *IEEE Transactions on Software Engineering*, 25(6):826–851, 1999.
14. Sepideh Ghanavati, Daniel Amyot, and André Rifaut. Legal goal-oriented requirement language (legal GRL) for modeling regulations. In *6th International Workshop on Modeling in Software Engineering, MiSE'14*, pages 1–6, 2014.
15. Sepideh Ghanavati, Daniel Amyot, André Rifaut, and Eric Dubois. Goal-oriented compliance with multiple regulations. In *Proceedings of the 22nd IEEE International on Requirements Engineering Conference, RE'14*, pages 73–82, 2014.
16. Orlena C. Z. Gotel and Anthony Finkelstein. An analysis of the requirements traceability problem. In *Proceedings of the First IEEE International Conference on Requirements Engineering, RE'94*, pages 94–101, 1994.
17. Government of Luxembourg. Modified Law of December 4th, 1967 (Income Tax) (In French: Loi modifiée du 4 décembre 1967 concernant l'impôt sur le revenu), 2013.
18. Mohammad Hamdaqa and Abdelwahab Hamou-Lhadj. An approach based on citation analysis to support effective handling of regulatory compliance. *Future Generation Computer Systems*, 27(4):395 – 410, 2011.
19. Rinke Hoekstra. The metalex document server legal documents as versioned linked data. In *Proceedings of the 10th International Conference on The Semantic Web - Volume Part II, ISWC'11*, pages 128–143, Berlin, Heidelberg, 2011. Springer-Verlag.
20. Silvia Ingolfo, Ivan Jureta, Alberto Siena, Anna Perini, and Angelo Susi. Nòmos 3: Legal compliance of roles and requirements. In *Proceedings of the 33rd International Conference on Conceptual Modeling,ER'14*, pages 275–288, 2014.
21. John Levine, Tony Mason, and Doug Brown. *Lex & Yacc*. O'Reilly, 1992.

22. Aaron K. Massey, Paul N. Otto, and Annie I. Antón. Prioritizing legal requirements. In *Proceedings of the 2009 Second International Workshop on Requirements Engineering and Law, RELAW'09*, pages 27–32, Washington, DC, USA, 2009. IEEE Computer Society.
23. Jeremy C. Maxwell, Annie I. Antón, and Julia B. Earp. An empirical investigation of software engineers' ability to classify legal cross-references. In *Proceedings of the 21st IEEE International on Requirements Engineering Conference, RE'13*, pages 24–31, 2013.
24. Jeremy C. Maxwell, Annie I. Antón, Peter Swire, Maria Riaz, and Christopher M. McCraw. A legal cross-references taxonomy for reasoning about compliance requirements. *Requirements Engineering*, 17(2):99–115, June 2012.
25. Christian Nentwich, Licia Capra, Wolfgang Emmerich, and Anthony Finkelstein. xlinkit: a consistency checking and smart link generation service. *ACM TOIT*, 2(2):151–185, 2002.
26. Monica Palmirani, Raffaella Brighi, and Matteo Massini. Automated extraction of normative references in legal texts. In *Proceedings of the 9th International Conference on Artificial Intelligence and Law, ICAIL'03*, pages 105–106, New York, NY, USA, 2003. ACM.
27. Grzegorz Rozenberg, editor. *Handbook of graph grammars and computing by graph transformation (Vol. 1): Foundations*. World Scientific, 1997.
28. Mehrdad Sabetzadeh, Shiva Nejati, Sotirios Liaskos, Steve M. Easterbrook, and Marsha Chechik. Consistency checking of conceptual models via model merging. In *15th IEEE International Requirements Engineering Conference, RE*, pages 221–230, 2007.
29. Nicolas Sannier and Benoit Baudry. INCREMENT: A mixed MDE-IR approach for regulatory requirements modeling and analysis. In *Proceedings of the 20th International Working Conference on Requirements Engineering: Foundation for Software Quality, REFSQ'14*, pages 135–151, 2014.
30. Alberto Siena, Ivan Jureta, Silvia Ingolfo, Angelo Susi, Anna Perini, and John Mylopoulos. Capturing variability of law with nómos 2. In *Proceedings of the 31st International Conference on Conceptual Modeling, ER'12*, pages 383–396, 2012.
31. the EU Reflection Group on Legislative Drafting. Joint Practical Guide for persons involved in the drafting of European Union legislation. Technical report, the European Parliament, the Euporean Council and the European Commission, 2013.
32. the Ontario Ministry of Consumer and Business Services and the Ontario Ministry of Health and Long Term Care. Personal Health Information Protection Act, 2004, 2004.
33. Oanh Thi Tran, Ngo Xuan Bach, Minh Le Nguyen, and Akira Shimazu. Automated reference resolution in legal texts. *Artif. Intell. Law*, 22(1):29–60, 2014.
34. Nicola Zeni, Nadzeya Kiyavitskaya, Luisa Mich, James R. Cordy, and John Mylopoulos. GaiusT: Supporting the extraction of rights and obligations for regulatory compliance. *Requirements Engineering*, 20(1):1–22, 2015.