Automated Detection and Resolution of Legal Cross References: Approach and a Study of Luxembourg's Legislation

Morayo Adedjouma, Mehrdad Sabetzadeh, Lionel C. Briand
SnT Centre for Security, Reliability and Trust
University of Luxembourg, Luxembourg
{morayo.adedjouma, mehrdad.sabetzadeh, lionel.briand}@uni.lu

***Abstract*—When elaborating compliance requirements, analysts need to follow the cross references in the underlying legal texts and consider the additional information in the cited provisions. To enable easier navigation and handling of cross references, automation is necessary for recognizing the natural language patterns used in cross reference expressions (*cross reference detection*), and for interpreting these expressions and linking them to the target provisions (*cross reference resolution*). In this paper, we propose a solution for automated detection and resolution of legal cross references. We ground our work on Luxembourg's legislative texts, both for studying the natural language patterns in cross reference expressions and for evaluating the accuracy and scalability of our solution.**

***Index Terms*—Legal Compliance, Legal Texts, Cross References.**

I. Introduction

Legal compliance is a key concern for many software applications. Information systems in domains such as government, healthcare and finance are, for example, subject to a host of laws and regulations aimed at protecting security and privacy. Systems used by governments for public administration further need to comply with public regulatory frameworks such as tax and social welfare laws. In all cases, non-compliance can have serious consequences, including fines, lawsuits, damage to public trust, loss of business, and even criminal prosecution.

To identify and elaborate compliance requirements, analysts need to review and analyze the relevant legal texts. An important complexity that arises in so doing is that legal provisions are interrelated and cannot be considered in isolation. The relationships between provisions in legal texts are captured using *cross references*. To illustrate, consider the excerpt in Fig. 1 from Article 2 of Luxembourg's Income Tax Law [1]:

> **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.

Fig. 1. Using cross references for relating legal provisions

The excerpt (translated from French) has a cross reference to "Art. 156". To understand what a non-resident taxpayer is, one needs knowledge of what local income is. This example shows only one usage of cross references, i.e., specifying a dependency for a definition – in this case, the definition of local income. Cross references may be used, among other reasons, for adding exceptions and constraints, specifying priorities between provisions, and making amendments [2].

While mainly an apparatus of legal writing, cross references have implications on software requirements [2]. Maxwell et al. [3] argue that failing to consider cross references or misunderstanding their intent can lead to costly non-compliance issues in software systems. Several strands of work concerned with legal applications in the Requirements Engineering literature take cross references into consideration. For example, Breaux & Antón [4] 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. [5], [6] model legal cross references as explicit goals and use these goals both for compliance analysis of business processes and for change propagation between legal 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 [7], [8]. Doing so requires the ability to first recognize the natural language expressions that denote cross references (*cross reference detection*), and then to interpret and link these expressions to the target provisions (*cross reference resolution*). 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 [5]. A typical legal text can contain hundreds and sometimes thousands of cross references. Automated support is thus necessary for cross reference detection and resolution.

In this paper, we develop a framework for automated detection and resolution of cross references in legal texts. Several approaches already exist for this purpose [9], [10], [11], [7], [12], [13]; but, as we argue in more detail in Section X, certain facets have not been adequately explored. Most notably:

- Cross reference detection and resolution require knowledge of the structure of legal texts, i.e., how these texts are organized into sections, articles, and so on. Typically, a schema is defined to express this structure [14], [7], [8]. Manual work is however necessary to transform a document without markup (e.g., in PDF format) into a markup document (e.g., in XML format) that conforms to the schema.
- In some countries, best practices exist on how to phrase and use cross references in legal texts. For example, in the US,

978-1-4799-3032-6/14/$31.00 © 2014 IEEE

RE 2014, Karlskrona, Sweden

the Bluebook [15] and the US Association of Legal Writing Directors' (ALWD) Citation Manual [16] lay down specific conventions for cross references. These best practices, as already observed by others [12], are often inadequate for accurate detection of cross references, particularly in older legal corpora, e.g., public law. Grounded Theory studies of actual legal texts, e.g., as done by Breaux [7] and de Maat et al. [10], provide valuable insights about the flexible natural language patterns used for specifying cross references; however, further investigations of actual legal texts is required to understand commonalities between legal texts across different countries and to develop reusable natural language patterns for cross reference detection.

- The majority of existing work does not clearly distinguish cross reference detection and the more complex task of resolution. Consequently, important subtleties that arise during resolution have not been adequately addressed, e.g., disambiguation when the cross reference patterns are ambiguous.

To address the above, we make the following contributions:

1. We describe how a schema for the structure of legal texts can be used to *automatically* transform non-markup texts into texts with structural markup. This task is a prerequisite for cross reference resolution and, if done manually, is laborious.

2. We report on a thorough examination of the cross reference expressions in Luxembourg's Income Tax Law [1], deriving natural language patterns for these expressions that we believe are likely to generalize to other texts and other countries.

3. We provide a systematic treatment of cross reference resolution, describing the subtleties in the interpretation of complex cross reference expressions. While our treatment is applicable to both *intra-* and *inter*-document cross references, only the former category (commonly referred to as *internal* cross references) is considered in this paper.

4. Building on a Natural Language Processing (NLP) platform, GATE [17], and a first-order logic interpreter, Crocopat [18], we develop tool support for automated detection and resolution of cross references and conducting cross reference analysis.

The remainder of the paper is organized as follows: Section II provides background and terminology. Section III presents an overview of our approach, followed by Sections IV through VII, where we elaborate the technical components of the approach. Section VIII outlines tool support. Section IX describes an evaluation of our approach. Section X discusses related work and Section XI concludes the paper.

The examples used throughout the paper are from Luxembourg's Income Tax Law. This law is in French. For presentation purposes, we use English translations while preserving the structure of the original cross reference expressions.

II. Background and Terminology

A *(legal) cross reference* is a citation that links one legal provision to another [2]. We distinguish cross references from cross reference expressions. A *cross reference expression* is a Natural Language (NL) phrase that represents one or more cross references. For example, "Articles 30 and 102 of the law of April 29, 1964 concerning family benefits" is a cross reference expression. This expression embodies two cross references: one to "Article 30" and another to "Article 102" of the respective law. To avoid repetition, we abbreviate "cross reference" to ***CR*** and "cross reference expression" to ***CRE*** in the rest of the paper. We note that, with the distinction made between CR and CRE, it would be more accurate to refer to CR detection and CR resolution as CRE detection and CRE resolution, respectively. We ignore this technicality when referring to detection and resolution activities.

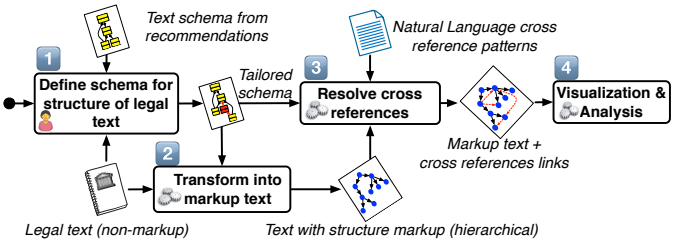

Fig. 2. Approach Overview

When a CRE points to provisions within the same legal text as where the CRE appears, the CRE is called *internal*; otherwise, when a CRE points to provisions in a different legal text, it is called *external* [19]. In the example of Fig. 1, "Art. 156" is an internal CRE. The example in the previous paragraph (i.e., "Articles 30 and 102 of ...") is an external CRE.

CREs can be further classified as explicit, implicit, or delegating. If a CRE is defined in terms of the alphanumeric labels of legal text elements, it is called *explicit*. All our examples so far where articles were referred to by their numbers were explicit. In contrast, an *implicit* CRE uses some adjective, adverb, or anaphor to refer to the target provisions [10] , e.g., "this article" and "the following paragraphs". The third class, namely *delegating*, exclusively applies to external CREs. This class of CREs is used when a legislative text delegates authority to regulations for further elaboration. Regulations reside at a lower level than legislation in the hierarchy of legal texts and are usually subject to more frequent changes. Legislative texts seldom refer to regulations in a precise way and typically only indicate the nature of the regulations being cited. An example of delegating CRE is "Grand-Ducal regulation" in the following: "A Grand-Ducal regulation shall provide the details for [...]".

Finally and with regards to implicit CREs, there are occasions where legal texts use vague terms such as "provision" (in French: "disposition" or "prescription"), e.g., "the above provision". We refer to these CREs as *unspecific*. Unspecific CREs cannot be conclusively associated with specific structural elements, e.g., articles or paragraphs. They cannot thus be resolved with reasonable accuracy through automation. All CREs except for delegating and unspecific ones are in principle resolvable through automation.

III. Approach Overview

In this section, we provide an overview of our approach. The approach, shown in Fig. 2, has four main steps. The first step is manual and the remaining three steps are automatic.

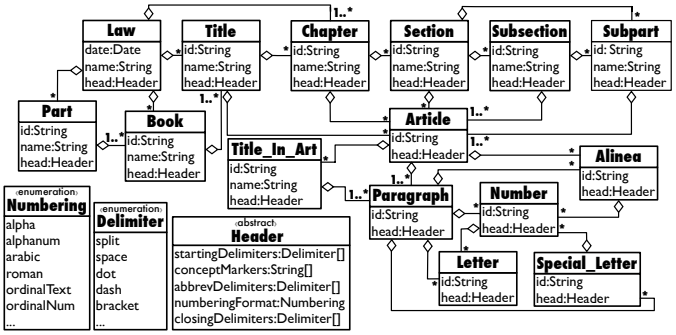

Fig. 3. Text schema for Luxembourg's Income Tax Law

Step 1 is concerned with defining a schema (metamodel) for expressing how a legal text is organized into subparts, e.g., chapters, articles, paragraphs and so on. In many jurisdictions, legal writing guidelines prescribe a generic schema for structuring legal texts. Actual legal texts may nonetheless be only partially aligned with the generic schema. Some tailoring of the generic schema may thus be necessary in order to adapt it to a specific set of legal texts. The aim of Step 1, discussed in Section IV, is to do this tailoring.

Step 2, discussed in Section V, is concerned with transforming a non-markup text (e.g., in plain text or PDF format) into a markup text (e.g., in XML format). The transformation rules are automatically derived from the schema of Step 1.

The main step of the approach is Step 3. This step, discussed in Section VI, addresses the detection and resolution of CREs. Prerequisite to this step is knowledge of the NL patterns used in the CREs. These patterns, combined with the schema from Step 1, provide the syntactic and semantic basis for the CREs. As we argued earlier, the NL patterns used in actual legal texts can be richer and more versatile than what is prescribed in best practices. This necessitates examining actual legal texts for more accurate identification of the patterns.

Step 4, discussed in Section VII, is concerned with using the outcomes of resolution for visualization and analysis.

IV. Defining a Legal Text Schema

A text schema defines the grouping concepts, e.g., Chapter, Section and Article, used for organizing the content of a legal text. Having a text schema is essential for CR detection and resolution as both the syntax and interpretation of CREs depend on the underlying schema. We define a text schema via a UML class diagram. The classes in the diagram represent the grouping concepts used in a legal text. These classes are linked via aggregation associations representing the containment relationships between the concepts.

Fig. 3 shows the text schema for Luxembourg's Income Tax Law. To build this schema, we first developed a generic schema based on the legal writing guidelines published by the State Council of Luxembourg [20]. The resulting schema was then discussed with legal experts and enhanced with additional grouping concepts specific to the tax legislation.

An individual legal text constitutes a *Law*. A *Law* may be hierarchically organized into *Part*s, *Book*s, *Title*s, *Chapter*s, *Section*s, and so on. Many levels in the hierarchy are optional, e.g., if *Part* is not used, one can go directly from *Law* to *Book*, *Title*, or *Chapter*. These alternatives are captured in the model of Fig. 3 through different aggregation paths. The core grouping concept in a Luxembourgish legal text is that of an *Article*. Articles are typically organized into *Paragraph*s. Paragraphs are often made up of *Number*s and *Letter*s. Occasionally, a special type of numbered statements, known as *Alinea*s, are used for decomposing articles and paragraphs. *Title_In_Art* and *Special_Letter* are two other nuanced groups used in certain cases within articles and paragraphs, respectively. Each grouping concept has a *header* (explained below) and an *id*. There is only one instance of *Law* per text, so the *id* is implicit in this case. Nevertheless, each law is associated with a publication date that needs to be captured. Some grouping concepts, e.g., chapters, have both an *id* and a *name* (chapter title). The header of a grouping concept *C*, called *CHeader*, provides information as to how the beginning of an instance of *C* is recognized in the text. In the model of Fig. 3, only the abstract header class is shown. Each grouping concept defines a (static) specialization of this abstract class. As an example, we show *ArticleHeader* in Fig. 4. Here, the starting delimiter is a split (carriage return or linefeed), and the concept marker is "Art" or "Article". When the abbreviated concept marker is used, it may be (optionally) followed by a dot. The numbering is alphanumeric representing the identifier of the article in the legal text. The article header closes with a dot.

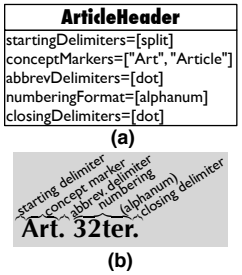

Fig. 4. (a) Article header class; (b) Example of an article head

V. Transforming Non-Markup to Markup Text

We automatically derive from a text schema, e.g., the one in Fig. 3, 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 we see a new grouping concept that is either at the same level as the current segment or at a level above the current segment. For example, assume that we have a document structured according to the schema of Fig. 3, and suppose that we are within a particular section, say Section 3. For this section to terminate, we either have to reach the beginning of Section 4, or, if there are no further sections, the beginning of a new chapter, a new title, etc.

The containment relationships between grouping concepts is never recursive, i.e., we cannot have a concept, say Chapter, which logically contains another concept, say Section, and at the same time have Sections that contain Chapters. More precisely, a text schema, when viewed as a graph, is always an *Acyclic Direct Graph (DAG)*. The transitive closure (reachability) relation in a text schema is therefore always a partial order. Let $<$ denote this partial order and let C_i and C_j be a pair of grouping concepts from the given schema. A relation $C_i < C_j$ implies that C_j directly or indirectly contains C_i. For example,

Section < *Chapter*, *Paragraph* < *Section* and *Letter* < *Article* are some of the relations in the partial order for the text schema of Fig. 3. Equipped with this partial order and information from the *CHeader* classes (Fig. 4), one can automatically generate the regular expressions that recognize the hierarchical structure of a legal text. Algorithm 1 shows the procedure for generating and executing these regular expressions.

Algorithm 1 Build Markup for Legal Text

1: Let G be a DAG whose nodes are the grouping concepts in the text schema and whose (directed) edges are the aggregations in the schema;
2: Let n be the number of nodes in G and let $<$ be the partial order induced by the reachability relation 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 any C_j header such that either $C_i \leq C_j$ or C_i and C_j are not comparable;
5: Run all HeadRegEx$_i$ (in any order) on the input text;
6: Run all SegmentRegEx$_i$ (in any order) on the input text;

We illustrate the regular expressions for header identification (HeadRegEx) and segmentation (SegmentRegEx) over the *Article* grouping concept. Generating the regular expression for marking the head of articles (line 3 of the algorithm) is straightforward based on the information in the *ArticleHeader* class (Fig. 4). In Fig. 5(a), we show a regular expression, named *MarkArticleHead* and written in GATE's regular expression language [17], for marking article heads. Intuitively and in line with our previous discussion about *ArticleHeader*, *MarkArticleHead* looks for the following sequence: a split, an admissible concept marker, optionally a dot, an alphanumeric number, and a final dot. Note that at the time this regular expression is run, the text has already undergone lexical analysis with information about its tokens available.

Fig. 5(b) shows a regular expression, *MarkArticleSegment*, for marking article segments. This expression, also written in GATE's regular expression language, recognizes and annotates the text between the head of a given article and the head of the next grouping concept instance that is not logically containable in that article. More specifically, the segment for an article starts when its head is detected. The segment ends when one of the following is detected: the head of another article, the head of a higher-level concept (as per the partial order), e.g., a chapter or a book, or the head of an unrelated concept. Since *Article* is comparable with all other grouping concepts in the schema of Fig. 3, there are no unrelated concepts to consider here. This is not the case for all concepts. For example, Alinea and Special_Letter are unrelated and should thus terminate one another's segments if used together in the same legal text. Finally, we note that in the regular expression of Fig. 5(b), there is a special *EOD* (End Of Document) level which logically contains all grouping concepts, meaning that the occurrence of *EOD* terminates any segment at any level.

(a)
```
Rule: MarkArticleHead
(({Split})+ ({Token.string=="Art"} | {Token.string=="Article"}) ({Token.string=="."})?
{Token.kind=="alphanum"} {Token.string=="."}):ref -->:ref.ArticleHead= {}
```

(b)
```
Rule: MarkArticleSegment
(({ArticleHead}):start ({LawHead} | {PartHead} | {BookHead} | {TitleHead} |
{ChapterHead} | {SectionHead} | {SubsectionHead} | {SubpartHead} |
{ArticleHead} | {EOD}):end):ref -->:ref.ArticleSegment= {}
```

Fig. 5. Example of markup rules for Article

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*⟩
27	⟨*multilayered-expr*$_2$⟩	::=	⟨*multilayered-expr*$_1$⟩⟨*sep-term*⟩⟨*internal-expr*⟩ | ⟨*multilayered-expr*$_1$⟩⟨*link-term*⟩⟨*internal-expr*⟩ | ⟨*multilayered-expr*$_1$⟩⟨*link-term*⟩⟨*multivalued-expr*⟩
28	⟨*sep-term*⟩	::=	"," | "–" | "and" | "or" | "to" | ...

Fig. 6. Grammar for natural language cross reference patterns

Due to space reasons, we cannot present all the technical details for generating the markup regular expressions. For example, groups such as chapters that have both ids and names have slightly more complex regular expressions that those for articles which only have ids. 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 the resolution process (Section VI).

VI. Detecting and Resolving Cross References

We automatically detect the CREs in a given legal text based on predefined patterns (Section VI-A). We then interpret the detected CREs into a set of individual CRs and link the resulting CRs to the target provisions (Section VI-B).

A. NL Patterns for Cross References Expressions

Our NL patterns were derived from a study of (Luxembourg's) Income Tax Law [1]. The 2013 edition of this law is 189 pages long and organized into 3 titles that collectively contain 15 chapters. The law has 236 articles, composed of 767 paragraphs. The large size and complexity of this law provides a rich basis for our investigation. Basing our work

on the Income Tax Law was further motivated by our access to legal experts who could help us with understanding the structure and the content of this law. We analyzed all the 1223 CREs in the Income Tax Law and developed regular expressions to formalize the patterns observed. These patterns, shown in Fig. 6, are organized into simple and complex. Complex patterns build on top of simple patterns, providing certain advanced features discussed later in this section.

We explain and illustrate our patterns, drawing on the terminology defined in Section II. We have made minor simplifications to the patterns for better readability. While the patterns were developed over French text, they carry over to English almost verbatim. 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"). In Fig. 6, symbols in upper-case letters, e.g., ⟨NUMBER⟩ denote terminals as identified by an NL lexical analyzer. Non-terminals that end with *term*, e.g., ⟨*marker-term*⟩ and ⟨*name-term*⟩ denote elements in predefined dictionaries (commonly known as gazetteers [17] in NLP). These terms vary from one legal jurisdiction and language to another and must be specified for a specific context.

Simple CREs. A simple CRE can be explicit or implicit (L. 1 of Fig. 6). 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 concept marker, e.g., article, followed by a numerical expression, or an ordinal expression followed by a concept marker. The numerical expression can be an arabic number ("article 1"), a roman number ("chapter IV"), an alphanumeric ("alinea 2bis"), a number written out in text ("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 new pattern observed in our study is when a generic term, e.g., under, replaces a concept 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., "social insurance code". 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 point to the internal provisions of an external law, e.g., "article 54bis as it was introduced by the Law of 23 July 1983". Delegating references (see Section II) also fall under external CREs.

A simple CRE may be implicit (L. 17-19), e.g., "this article". Among implicit CREs, there are some that cannot be resolved in an accurate way because they use an unspecific term, e.g., "the following provisions". An interesting variation of implicit CREs not previously reported are those that combine implicit terms and numerical expressions, e.g., "first four alineas".

Complex CREs. 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 [10]: *multivalued* and *multilayered* (L. 21). Multilayered CREs can have multivalued parts (L. 27).

A multivalued CRE (L. 22-24) cites many provisions in the same expression by specifying only once a concept 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., "numbers 1 to 3"; or (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 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. [10].

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 sentence of article 10 of the law of 23 may 1964". Finally, a navigation can 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*. Multilayered CRE 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".

TABLE I
CREs IN INCOME TAX LAW

CRE Type	# of CREs
Internal	928
– *Unspecific*	45
External	295
– *Delegating*	169
Explicit	839
Implicit	384
Simple	706
Complex	517
– *Mutlivalued*	192
– *Multilayered*	325

Table I provides a breakdown of the 1223 CREs in the Income Tax Law across three different dimensions: *(1) Internal vs. External*: 928 CREs are internal, of which 45 are unspecific, and 295 are external, of which 169 are delegating; *(2) Explicit vs. Implicit:* 839 CREs are explicit and 384 are implicit; and *(3) Simple vs. Complex:* 709 CREs are simple and 519 are complex. Among complex CREs, 192 are multivalued and 325 are multilayered.

B. Resolving Cross Reference Expressions

The CREs that match the NL patterns go through an interpretation phase and are then linked to the cited targets.

The aim of the interpretation phase is to translate each CRE into a set of individual CRs. The main complexity arising during interpretation is that some of the NL patterns in Section VI-A 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; and, (2) abbreviated concept markers (e.g., art.) have been replaced with full labels (e.g., article).

Interpreting Simple CREs. Among simple CREs, only implicit ones and those using generic terms (e.g., sub) need treatment. We distinguish two cases for implicit CREs: (1) The ones that are semantically equivalent to *current*, *previous*, or *next* followed by a concept marker *C*: In the case of *current*, the CRE is interpreted as pointing to the segment of the same type as *C* containing the CRE (Example 1). In the case of *previous* and *next*, the CRE is interpreted as pointing to segment(s) of the same type as *C* that precede or succeed the CRE, respectively (Example 2). (2) Implicit CREs that are semantically equivalent to *same* or *this* followed by a concept marker *C*. We interpret such a CRE as being equivalent to the closest CRE of type *C* that comes before the CRE in question (Example 3). Note that in both of the cases above, our interpretation is a *best-guess* heuristic, as we do not interpret the semantics of the underlying text.

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

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

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

Interpreting generic terms such as sub needs to be done according to the conventions in the legal jurisdiction to which the text belongs. In Luxembourg's legislation, the specific concept 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 concept marker is *Letter*; otherwise, the concept marker is *Number* (Example 4).

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

Interpreting multivalued CREs. A multivalued CRE that uses an enumeration is interpreted with the concept marker added to each element of the enumeration (Example 5). For interpreting CREs involving ranges, we distinguish grouping concepts that have unique numbering across an entire legal text (e.g., *Article*) and grouping concepts (e.g., *Chapter*, *Paragraph*) whose numbering is reset when a higher-level grouping concept is seen. For the former, we browse the entire hierarchical structure of the legal text to identify the elements in the range (Example 6). For the latter (grouping concepts whose numbering is reset), the interpretation depends on the context. We first 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 immediate parent of the current segment and then the parent's parent and so on, until we arrive at the right level for interpreting the CRE (Example 7). It is important to emphasize that the actual elements of a range cannot be deduced independently of the legal text because alphanumerics may be used in the numbering, as illustrated in Example 6. When multivalued CREs include implicit terms, we apply the same process as for simple CREs (see Examples 1 – 4).

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

Example 6
CRE: articles 99ter to 102
Legal Text: 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 interpretation attempt in the context of article 50bis, paragraph 4 fails. Second attempt at the level of article 50bis succeeds.

Interpreting multilayered CREs. For multilayered CREs that do not contain multivalued CREs, interpretation is performed by harmonizing the navigation order so that it is strictly upward or downward. For example, all the following CREs point to the same provision: (1) "article 131, 1st alinea, sub d)", (2) "sub d) of article 131, 1st alinea", (3) "1st alinea, sub d) of article 131", (4) "article 131 sub d) of 1st alinea". Only (1) is in a harmonized form. The harmonized form is easy to compute based on the topological order discussed in Section V. The most complex form of CREs arises when layers are combined with ranges and enumerations. The regular expressions that detect such CREs are *ambiguous*.

To illustrate, consider the CRE in Example 8. Without knowing the structural organization of the underlying text, one cannot know if the "127 and 154ter" fragment in this CRE refers to articles, paragraphs, or numbers. One could take cues from punctuation and the singular vs. plural concept 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 concept marker is in plural form. Unfortunately, such reasoning is unreliable as punctuation and the use of singular vs. plural are not consistently followed in legal texts. For example, the distinction between singular and plural disappears when abbreviations (e.g., art.) are used.

Example 8
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

We interpret multilayered CREs with ranges and enumerations in a similar manner to Example 7. 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 CRE in Example 8, this means that first, we take the numerical expression "127" (whose type is unknown) to be the continuation of "numbers 1 to 3". An attempt is made to interpret "127" in the context of "article[s] 109,1st alinea", i.e., the same context where "numbers 1 to 3"

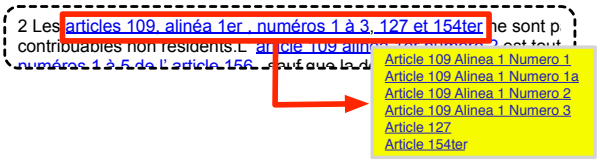

Fig. 7. A hyperlinked view of Luxembourg's Income Tax Law

was interpreted. After this attempt fails, we recursively switch to the upper-level context in the CRE, i.e., "article[s] 109" (the same context where "1st alinea" was interpreted). This second attempt also fails. The third attempt tries to interpret "127" in the context of all articles. This attempt succeeds. Now that "127" has been interpreted as an article, the CRE will be seen as if the concept 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".

Once the interpretation phase is completed, we link each CRE to all the provisions resulting from its interpretation. The details of establishing these links are straightforward. An example of how the resulting links are rendered is provided in Section VII when discussing visualization and navigation.

VII. Applications

In this section, we outline some important use cases that build on the results of CR detection and resolution.

Visualization and Navigation. CR detection and resolution is a prerequisite for generating navigable views of legal texts. In Fig. 7, we show a small excerpt of an HTML view of Luxembourg's Income Tax Law. In this view, resolved CREs appear as hyperlinks. Clicking on a CRE brings up a tooltip box, allowing the user to navigate to any of the CRs entailed by the CRE. A view like this is useful during the elaboration of compliance requirements, where analysts often need to follow the CRs in search of additional relevant information.

Identifying Anomalous CREs. A natural byproduct of the resolution process are diagnostics about CREs that cannot be resolved. Failure to resolve a CRE may be due to it being non-well-formed, or due to the CRE targets being non-existing. In either case, it is important for both legal experts and requirements analysts to be made aware of anomalous CREs.

Trace Link Analysis. Trace link analysis is concerned with identifying the artifacts or provisions that refer to a particular provision. While this type of analysis has a broad spectrum of applications, our work on the subject is motivated by two specific software engineering scenarios: (1) Many governmental websites and web forms, e.g., tax declaration forms, cite legal provisions to clarify how their content relates to the law. Without automated trace link analysis, it is very difficult to determine how a change in a given law affects the government's websites and forms. (2) Advanced text search: To illustrate this usage scenario, consider the following example from Luxembourg's Income Tax Law: "Art. 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 is: *Where is "Art. 24" referred to?* A naive lookup of the string "Art. 24" in the law's

```
Contains(x, y)      := Node(x) & Node(y) &
                       EX(e, Edge(e) & Type(e, "Contains") & Source(e, x) & Target(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, "Article") & EX(z, Cites(x, z) & TrContains(y, z));
```

Fig. 8. Logical queries for trace link analysis

text yields no results, despite the article being internally cited in four places. On all occasions, the article is cited within ranges: "Art. 4 to 155bis", "Art. 14 to 108bis" and "Art. 16 to 60" (appearing twice). Without automation, identifying where a given article is being cited requires a manual scan of the entire law, i.e., 189 pages of text in the case of the Income Tax Law.

To automate trace link analysis, we need a logical representation of the structure of the legal text(s) in question along with their CR links. This information is conveniently expressed as a *typed graph* [22] – intuitively, a graph whose nodes and edges are typed. In our problem, graph nodes represent instances of the grouping concepts in a legal text, e.g., individual chapters, sections, articles, and paragraphs. Each node type is thus one of the classes of the text schema. For Luxembourg's Income Tax Law, this is the schema of Fig. 3. Edges may represent two types of relationships: first, that a grouping concept instance *contains* another, e.g. Article 4 *contains* Article 4 Paragraph 2; and second, that a grouping concept instance (directly) *cites* another, e.g. Article 4 Paragraph 2 *cites* Article 46 Number 3.

Given such a logical representation, one can infer links between any pair of grouping concept instances. For example, one can identify all citations to a given article. We show the logical queries for this computation in the snippet of Fig. 8. The snippet is written in the Relational Manipulation Language (RML) – the query language for a first-order logic interpreter, named Crocopat [18]. In the snippet, we 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 CR. Finally, LinkedToArt(x, y) computes all (x,y) where y (i.e., the link target) is of type *Article*, and where x cites some element z that is transitively contained in y (e.g., a paragraph of y, or a letter in a paragraph of y).

Circularity Analysis. Cyclic citations are common in legal documents. A frequent usage is when a provision X refers to 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 absence of circular reasoning, e.g., cases where provisions X and Y both depend on one another for a definition. Circularity analysis is performed using similar logical queries to what we illustrated in Fig. 8. We do not elaborate the queries due to space constraints.

VIII. Tool Support

We implement our approach in a tool named LeCA (**Le**gal **C**ross Reference **A**nalyzer). LeCA provides automated support for (1) generation of text structure markup (Section V), (2) CR

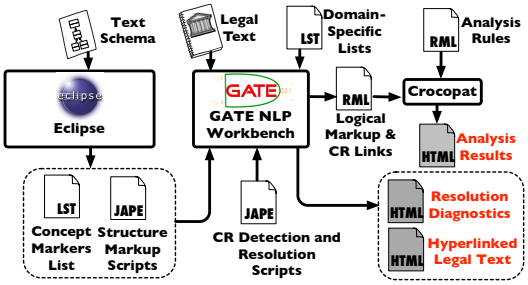

Fig. 9. LeCA tool architecture

detection and resolution (Section VI), and (3) visualization and CR analysis (Section VII). LeCA builds on the GATE Workbench [17] – a mature open-source NLP framework.

Fig. 9 shows an overview of LeCA. Eclipse's model-to-text transformation facilities are used in order to derive, from a text schema, JAPE scripts for text structure markup. These scripts are then executed by GATE, followed by those for CR detection and resolution. The CR detection and resolution scripts rely on several keyword lists (gazetteers), e.g., concept markers, names of legal texts, and implicit terms. The list for concept markers is derived from the text schema. The remaining lists are domain-specific and depend on the language of the legal text and the specific jurisdiction to which the legal text belongs. These lists thus need to be provided by the user.

As output, LeCA produces an HTML view of the input legal text with CRs represented as hyperlinks. Diagnostics are provided for any unresolved CRs. LeCA additionally generates a logical representation of the input text's structure and CRs, which is in turn fed to a first-order logic interpreter, Crocopat [18], for analysis. Our analysis rules currently cover traceability link generation and identification of circular citation paths.

LeCA is written in GATE's regular expression language, JAPE (Java Annotation Patterns Engine). In total, LeCA implements 113 JAPE scripts with approximately 10,000 lines of JAPE code, excluding comments and third-party libraries.

IX. Evaluation

In this section, we describe an evaluation of our approach, aimed at answering the following Research Questions (RQs):

RQ1. How accurate is our approach at detecting CREs? Our CRE patterns were gleaned from Luxembourg's Income Tax Law. RQ1 aims at evaluating the completeness of the patterns by analyzing their accuracy for detecting CREs in legislative texts other than the Income Tax Law.

RQ2. How accurate is our approach at resolving CREs? RQ2 aims at measuring how accurate our approach is in resolving CREs that have been already detected.

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

RQ1. To answer RQ1, we selected 13 legislative texts with a total of 1640 pages from Luxembourg's legal corpora. Our main selection criteria were to cover (1) a diverse set of texts, and (2) a large timespan in terms of when the texts were first drafted. The selected texts were drawn from civil laws, criminal and penal laws, social welfare, pensions, housing, commerce, and healthcare. The oldest text in our selection dates back to 1808 and the newest one to 2011.

TABLE II
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

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 concept markers (line 4 of the patterns in Fig. 6) prescribed for legislative texts by Luxembourg's legal writing best practices that are in effect today [20]. For generic terms, law names and auxiliary terms (respectively, lines 7, 13 and 15 of the patterns), we exploited the lists built from our investigation of the Income Tax Law. Table II summarizes the results for RQ1. In the table, we classify the identified CREs across the same dimensions as those used in Table I for the Income Tax Law.

The results indicate that our patterns miss only one CRE amongst 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 (line 7 of the patterns). 38 CREs ($\approx$ 2%) were only partially identified. All cases were attributed to incompleteness in the lists of concept markers, law names, and auxiliary terms. Detection further yielded 5 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, providing confidence about the completeness of our patterns for Luxembourg's legislation.

RQ2. We answer RQ2 based on the resolution of internal CREs in Luxembourg's Income Tax Law. Although this law is the same legal text from which we drew our CRE patterns, we believe that using this text towards answering RQ2 is justified in the light 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 only the text schema and the CREs patterns, irrespectively of the actual legal text. We thus anticipate little bias resulting from using the Income Tax Law in RQ2. We further need to note that although our patterns address both internal and external CREs, our evaluation of RQ2 is exclusively concerned with internal ones. A detailed resolution of external CREs requires the text schema for the external legal texts.

We attempted automated resolution for all internal CREs except unspecific ones, i.e., a total of 883 CREs (see Table I in Section VI-A). This resulted in 1736 CR links plus 9 warnings for the CRs whose target provision could not be found. Two of the authors independently inspected the resulting CR links and the warnings. All the 1736 automatically-generated links were deemed correct. Of the 9 warnings, 8 arose due to CREs that were indeed anomalous in the text of law. When combined, the CR links and the 8 warnings fully covered the internal CREs over which resolution was attempted.

The last warning was due to our algorithm incorrectly interpreting an external CRE as an internal one. The CRE in question is paragraph 11bis in the following excerpt: "The modified fiscal adjustment law of 16 October 1934 is amended with the following provision, inserted into the law as paragraph 11bis: [...]". Since our approach does not analyze the semantics of sentences, it interprets paragraph 11bis as being internal although the correct treatment is paragraph 11bis of the law of 16 October 1934.

With regards to the accuracy of resolution over the Income Tax Law, we get a precision of 99.9%, recall of 100%, and F-measure of 99.9% when we exclude unspecific CREs. From a practical standpoint, since we rely on a human expert to resolve the unspecific CREs, it seems more reasonable to treat these CREs as false negatives. This treatment provide a better reflection of the amount of manual work saved by automated resolution. Under this treatment, we obtain a precision of 99.9%, a recall of 97.5%, and F-measure of 98.7%. We further note that amendments, an example of which was given above, may result in situations where our approach cannot distinguish external CREs from internal ones. We thus expect lower accuracy over laws that make a large number of amendments.

RQ3. The execution time of our approach is dominated by the resolution phase. For the Income Tax Law, interpreting CREs into individual CRs took $\approx$151 seconds. Linking the CRs to the target provisions took a further $\approx$139 seconds. The interpretation step has embedded into it the time spent for transforming the Income Tax Law from plain text to XML. The execution time associated with the identification of CREs using regular expressions was $\approx$15 seconds for the Income Tax Law and $\approx$34 seconds for the selected pages in RQ1. Execution times were measured on a laptop with a 2.3 GHz Intel CPU and 8GB of memory. Given the short overall running time of our approach, we expect it to be scalable to larger legal corpora with thousands of pages.

X. Related Work

Breaux et al. [4], [7] identify natural language patterns for CREs 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 actual legal texts for identifying CRE patterns and use an explicit model to characterize the structure of legal texts. We consider a single but complete legal text (Luxembourg's Income Tax Law) with a total of 1223 CREs. We observe new patterns, not seen in the US regulations considered. Our study covers patterns for external CREs, not considered in [4], [7]. We propose automation for text structure markup, which is a manual process in [7].

Kiyavitskaya et al. [11] automatically generate annotations for legal texts, extracting a wide range of concepts including actors, rights, obligations, and cross references. With respect to cross references, our investigation identifies new CRE patterns not reported in this earlier work. We further propose a method that automatically transforms legal text schemas into pattern matchers for generating structural markup.

The closest study to ours in terms of CRE patterns is de Maat et al.'s study of Dutch laws [10]. The differences in language aside, the patterns we observe in our investigation of Luxembourgish laws (written in French) are closely aligned with those in Dutch laws. In this sense, our study serves as a confirmatory measure for the generalizability of previously-observed patterns. In addition, we identify natural and important variations of these patterns. The main contributions of our work over de Maat et al.'s are: First, 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 [8]). Our approach, in contrast, does not require pre-existing markup. Second, and more importantly, de Maat et al. do not clearly distinguish detection and resolution activities. 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. [9] define CRE patterns based on legal writing guidelines for the Italian legal corpora. They tackle only detection and not resolution. Their approach does not address the generation of markup documents and their patterns are insufficient for recognizing many of the patterns seen in our study and that of the Dutch laws discussed above [10].

Hamdaqa et al. [12] propose an approach for resolving external CREs, whereby finite state machines are used for capturing the CRE patterns recommended by the Bluebook [15] and ALWD's Citation Manual [16]. The approach further considers markup generation through manually-written pattern matchers. The CRE patterns in this earlier work cover only external CREs and are exclusively based on best practice recommendations. Our approach provides a more thorough treatment: Our patterns encompass both internal and external CREs, and further are based on studying actual legal texts. Our approach automatically derives, from a given legal text schema, pattern matchers for text markup generation.

Tran et al. [13] apply machine learning for CR detection and resolution in Japanese legislative texts. Similar to them, we distinguish CR detection and resolution. However, our detec-

tion and resolution strategies are algorithmic and rule-based. Using machine learning can be advantageous in that it does not require an a priori specification of CRE patterns. 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 VI-A). It is unknown how such patterns can be handled through learning. Further, for the patterns they do consider, they report an accuracy (F-measure) of $\approx 80\%$ and $\approx 86\%$ for detection and resolution respectively, with a combined accuracy of 67%. This, compared to rule-based techniques, is low (see Section IX).

Our work on cross reference analysis (Section VII) is close to that of Nentwich et al. [23] 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 cross references. xlinkit can thus be used as an alternative to the logical interpreter we use in our work (Crocopat [18]).

XI. Conclusion

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. This text schema further allowed us to automatically enhance non-markup legal texts with the structural markup that is necessary for resolving cross references. Through a study of legislative texts in Luxembourg, we extended existing natural language patterns for cross reference expressions and provided a systematic way to interpret these expressions. We outlined the implementation of our approach in a Natural Language Processing environment. Our evaluation of the approach indicates that it is scalable and accurate in detecting and resolving cross references in Luxembourg's legislation.

Our work in this paper is part of a collaborative effort with the Government of Luxembourg on legal compliance in eGovernment services. An important challenge in this context is managing the continuous evolution of government IT systems as well as the laws and regulations that apply to these systems. Cross references provide important cues not only about the relationships between legal provisions but also about the relationships between the compliance requirements derived from these provisions. Consequently proper handling of cross references is essential for automated traceability and change analysis over compliance requirements.

In future work, we plan to conduct larger-scale evaluations of our approach and analyze the effectiveness of resolution for external cross references. We further plan to investigate how changes in legal provisions induce changes in compliance requirements. Another interesting topic for future work is to generalize our approach beyond legislative texts, and to regulations, circular letters, and parliamentary proceedings. Being able to automatically link and navigate these different types of legal texts makes it easier to analyze the rationale behind compliance requirements and the way they need to be operationalized in software applications.

Acknowledgments

Funding was provided by Luxembourg's National Centre for 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.

References

[1] Govt. of Luxembourg, "Modified Law of Dec. 4, 1967 (Income Tax) (Loi modifiée du 4 déc 1967 concernant l'impôt sur le revenu)," 2013.
[2] J. Maxwell, A. Antón, P. Swire, M. Riaz, and C. McCraw, "A legal cross-references taxonomy for reasoning about compliance requirements," *REJ*, vol. 17, no. 2, pp. 99–115, 2012.
[3] J. Maxwell, A. Antón, and J. Earp, "An empirical investigation of software engineers' ability to classify legal cross-references," in *RE*, 2013, pp. 24–31.
[4] T. Breaux and A. Antón, "Analyzing regulatory rules for privacy and security requirements," *IEEE TSE*, vol. 34, no. 1, pp. 5–20, 2008.
[5] S. Ghanavati, D. Amyot, A. Rifaut, and E. Dubois, "Goal-oriented compliance with multiple regulations," in *RE*, 2014, to appear.
[6] S. Ghanavati, D. Amyot, and A. Rifaut, "Legal goal-oriented requirement language (legal GRL) for modeling regulations," in *MiSE*, 2014, pp. 1–6.
[7] T. Breaux, "Legal requirements acquisition for the specification of legally compliant information systems," Ph.D. dissertation, North Carolina State University, 2009.
[8] R. Hoekstra, "The metalex document server," in *10th Intl. Conf. on The Semantic Web*, 2011, pp. 128–143.
[9] M. Palmirani, R. Brighi, and M. Massini, "Automated extraction of normative references in legal texts," in *9th Intl. Conf. on AI and Law*, 2003, pp. 105–106.
[10] E. de Maat, R. Winkels, and T. van Engers, "Automated detection of reference structures in law," in *19th Annual Conf. on Legal Knowledge and Information Systems*, 2006, pp. 41–50.
[11] N. Kiyavitskaya, N. Zeni, T. Breaux, A. Antón, J. Cordy, L. Mich, and J. Mylopoulos, "Automating the extraction of rights and obligations for regulatory compliance," in *ER*, 2008, pp. 154–168.
[12] M. Hamdaqa and A. Hamou-Lhadj, "An approach based on citation analysis to support effective handling of regulatory compliance," *Future Generation Computer Systems*, vol. 27, no. 4, pp. 395–410, 2011.
[13] O. Tran, M. Le Nguyen, and A. Shimazu, "Reference resolution in legal texts," in *14th Intl. Conf. on AI and Law*, 2013, pp. 101–110.
[14] W. Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, S. Armitage, and R. Stevens, "Managing standards compliance," *IEEE TSE*, vol. 25, no. 6, pp. 826–851, 1999.
[15] *The Bluebook: A Uniform System of Citation*, 19th ed. Harvard Law Review, 2010.
[16] Association of Legal Writing Directors and D. Dickerson, *The ALWD Citation Manual*, 4th ed. Aspen Publishers, 2010.
[17] Cunningham et al, "Developing Language Processing Components with GATE Version 7 (a User Guide)." [Online]. Available: http://gate.ac.uk
[18] D. Beyer, A. Noack, and C. Lewerentz, "Efficient relational calculation for software analysis," *IEEE TSE*, vol. 31, no. 2, pp. 137–149, 2005.
[19] A. Massey, P. Otto, and A. Antón, "Prioritizing legal requirements," in *RELAW*, 2009, pp. 27–32.
[20] M. Besch, "Traité de légistique formelle," 2005.
[21] J. Levine, T. Mason, and D. Brown, *Lex & Yacc*. O'Reilly, 1992.
[22] G. Rozenberg, Ed., *Handbook of graph grammars and computing by graph transformation (Vol. 1): Foundations*. World Scientific, 1997.
[23] C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein, "xlinkit: a consistency checking and smart link generation service," *ACM TOIT*, vol. 2, no. 2, pp. 151–185, 2002.