БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ . BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, 55
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, 55

София . 2005 . Sofia

Legal Document – a Formal Model

Vanyo Peychev

Faculty of Mathematics and Informatics, Sofia University

E-mail: vanyo@fmi.uni-sofia.bg

Introduction

The electronic data processing involves processes such as: creation, publication, access rights, searching, sending, archiving and etc. Every process is realized with operations above document. The operations are presented in easier way if the model for them has been created. One solution is to present documents by appropriate form for running operations. This means that we include additional meta information in documents and the structure for the documents is created. *Structured documents* we will call documents for which tags are included. Basic operations with structured document elements are: adding a new element, deleting, updating and searching an element.

One of the most important tasks for document processing is search operation. Finding appropriate document content that satisfies search conditions in maximal degree is a base problem. For solving this problem it is most important to create a document structure.

A formal model for structuring documents is presented in this paper. The introduced formal model is used to describe a legal document as a structured document. *Z*-notation is the formalism, used to describe legal documents. *Z*-notation is also used to explain the operations with legal documents.

Structured documents – formal model

Let's look at the set $\boldsymbol{D}$ of documents having the same kind of structure. The set $\boldsymbol{X}$ defines all tags for given set of documents $\boldsymbol{D}$.

$$\boldsymbol{X} = \{ \text{ set of tags for given document class } \}.$$

The tags are metadata for a document. They describe the document structure and the hyperlinks in the document. This means that every document can involve not

only its own information tags but also information parts provided by other documents of different document class. In general it is not necessary that the other documents are structured ones. Hyperlinks can also point to internal parts of the same document. In Fig. 1 is present a document with hyperlinks and relations with external documents.

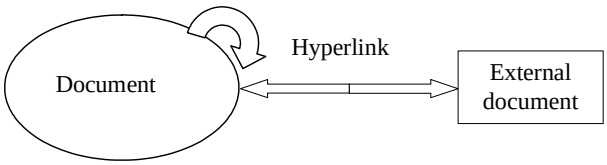

Fig. 1. Document with hyperlinks

Let T be a set containing the paragraphs or texts in a given document,

$$T = \textbf{\{ paragraphs \}}.$$

Every subset of the set $T \times X$ is called a ***document***. Therefore for every document $\boldsymbol{D}$, $\boldsymbol{D} \subseteq T \times X$ is true. $\boldsymbol{P_D}$ is the set containing all elements of a given document**.**

$$\boldsymbol{P_D} \triangleq \text{seq } X.$$

The set $\boldsymbol{P_D}$ specifies the type and order of the elements of every different document $\boldsymbol{D}$. Two different documents could have equal types of elements in the same order. Every subset of the set $T \times P$ we call a ***structured document***. If $\boldsymbol{D}$ is document ($\boldsymbol{D} \subseteq T \times X$), $\boldsymbol{D}$ is structured document if $\boldsymbol{D} \subseteq T \times P$. We define the representation of set $\boldsymbol{T}$ in set $\boldsymbol{P_D}$ as sequence of particular functions:

$$T \rightarrow \text{seq}\boldsymbol{P_D}. \tag{1}$$

The set of particular functions (1) describes the structure and the text for a given structured document $\boldsymbol{D}$.

Advantages and disadvantages

Documents with predefined structure have the following advantages, considering documents automatic processing:

- document information is saved in the unified way for all documents;
- document creation and visualization are the same for all documents;
- control brought in the process of documents creation and saving;
- reduction of the number of human errors when the documents are creating;
- improvement of the searching process;
- others.

The possibility of saving document information in the same way allows extension of the classes of documents. We can save a variety of templates in the same way. Therefore various data can be processed in a unified way.

The human errors, when documents are created and saved, are less, if control rules are introduced. Data will be checked before input.

Creating a structure for a document would improve the searching efficiency. Each structured document can be presented with a tree structure. This representation allows the application of investigated and established search methods over structured documents.

The disadvantages mainly concern:

- definition of the document structure;
- document visualization.

The basic disadvantage in managing structured documents is that for each different type of documents we should create appropriate information structure in which the document will be saved. This means, that we should define a template for the document class and choose a way of describing the document structure.

Defining a template for the document class is a non-trivial task. Usually, we create document class templates only for particular documents. The developers are the people involved in solving of this task. As a result, the creating of a template for a document class depends on the developer's experience and professional training. This is absolutely unreasonable, because there is much subjectivity. Moreover, the way of describing the document structure is important for the template definition and user interface. This dependency has negative influence on document structuring.

In spite of these disadvantages, the creating and saving of templates for document classes is an important problem, because it extends the possibilities for document management.

Operations on legal documents

Z-notation is a relatively simplified algebraic notation which describes the processes and definitions through schema presented block structures [1-3]. The operations over legal documents are described by Z-notation. To simplify operations over legal documents we will avoid examining hyperlinks as part of the elements of a legal document. The basic operations that we shall describe are:

- add a new element;
- delete existing element;
- update existing element;
- search for element.

Every legal document consists of different elements like parts, clauses, terms, etc. [4]. An element of a legal document is presented as shown

LegalElement

Type : Nomenclature
Content : Text

where

$$\text{Nomenclature} \triangleq \boldsymbol{P}_{\boldsymbol{D}},$$
$$\text{Content} \triangleq T.$$

Considering the structured document definition and the defined above concepts, we specify a legal document as:

$$\text{LegalDocument} \triangleq \text{seq}\ \boldsymbol{P}_{\boldsymbol{D}}.$$

Limitations:

- element order – defined as:

$$\textbf{LegalElement}_{i} < \textbf{LegalElement}_{i+1}$$

- at the beginning the legal document is empty

$$\text{LegalDocument}_{\text{INIT}} \triangleq \{\ \}.$$

Each operation on a legal document transforms the document from a state before operation execution (LegalDocumentBefore) to a state after operation execution (LegalDocumentAfter). The definition of every operation includes the state of the document before and after the operation execution. With ΔLegalDocument we note the document change as a result of operation execution:

ΔLegalDocument ≙ [LegalDocumentBefore, LegalDocumentAfter: LegalDocument].

Some operations don't change the document. We note this as follows:

≡LegalDocument ≙ [ΔLegalDocument: LegalDocumentBefore =
=LegalDocumentAfter].

The operations which do not change the document save its contents and elements' order.

To specify the operations over the elements of a legal document we introduce additional functions as follows:

• The function ***TakeElement*** has two arguments. For a given ***LegalDocument*** and ***Index***, the function returns ***LegalElement***, which is saved on a pointed position in the document. We suppose that we have index access to the elements of legal document:

TakeElement

LegalDocument
Index : Integer
LegalElement?
LegalElement ⫨ $LegalElement_{Index}$

• The function ***ChangeOrder*** has two arguments. For a given ***LegalDocuemt*** and ***Index***, it shows the position from which the index of elements will increase with 1. The function returns a legal document in which on the position ***i*** there is an empty element.

ChangeOrder

```
LegalDocument
Index          : Integer
I              : Integer
LegalElement
I = 0
LegalDocumentAfter = {}
FOR EACH LegalElement IN LegalDocumentBefore
   IF I = Index BEGIN
      LegalDocumentAfter ⫨ LegalDocumentAfter ∪ {}
   END
   ELSE BEGIN
      IF I > Index BEGIN
         LegalElement ⫨ TakeElement(LegalDocumentBefore,I)
         LegalDocumentAfter ⫨ LegalDocumentAfter∪LegalElement(I+1)
      END
      ELSE BEGIN
         LegalElement ⫨ TakeElement(LegalDocumentBefore,I)
   LegalDocumentAfter ⫨ LegalDocumentAfter∪{LegalElement(I)}
      END
NEXT I
```

Two base operations are adding and deleting of the last element in a document.

1. Add a new element
   - At the end of the document

NewLegalElement

ΔLegalDocument
LegalElement?
LegalDocumentAfter ≒ LegalDocumentBefore∪{LegalElement?$_{Position}$}
NEXT I

   - Add a new element on a chosen position

NewLegelElement

ΔLegalDocument
LegalElement?
Position

ChangeOrder(LegalDocumentBefore, Position)
LegalDocumentAfter ≒ LegalDocumentBefore∪{LegalElement?}

2. Delete element – this operation is not practically used with legal documents. We present it here just for completeness of the set of operations over legal documents.

DeleteElement

ΔLegalDocument
LegalElement?

LegalDocumentAfter ≒ LegalDocumentBefore –{LegalElement?}

3. Update element – the updating of a document contents is done either for context associate elements (clauses, terms, etc.) or for context unassociated elements (paragraph parts). Here we will only define update operation only for context associate elements.

DeleteElement

ΔLegalDocument
ElementType?: Nomenclature
ElementText! : Text

LegalDocumentAfter≒LegalDocumentBefore⊕{ElementType?→ElementText}

4. Search

4.1. By element type

SearchLegalElementByType

≡LegalDocument
ElementType?: Nomenclature
LegalElement!: LegalDocument

LegalElement!≒{ElementText:LegalDocument|ElementType?∩ElementTextType≠{}}

4.2. By element contents (text)

SearchLegalElementByType

≡LegalDocument
ElementType?: Nomenclature
LegalElement!: LegalDocument

LegalElement! ≒ {ElementText:LegalDocument|ElementType?∩ElementTextType≠{}}

The function ***Coincidence(Text)*** in a real application must be defined in dependence on the application language. It can allow and execute operations as a template matching.

Model advantages and disadvantages

Advantages:

- Management of different types of structured documents by unified way;
- Defining a set of the base operations. Every operation over structured document can be present as superposition of these basic operations.

Disadvantages:

- The model is limited only for legal document presentation;
- The operations presented are not designed to manage hyperlinks.

Conclusion

The presented formal model of structured documents can be a good base for complete and complex investigation over structured documents. The article has shown an example of operations description over legal document according to the model. For the operation description Z notation is used.

References

1. Davies, J, J. Woodcock. Using Z: Specification, Refinement and Proof. Prentice Hall International, 1996
2. Information Technology – Z Formal Specification Notation - Syntax, Type System and Semantics. ISO/IEC 13568, 2002.
3. Spivey, M. The Z Notation: A Reference Manual. 2nd edition. Prentice Hall International, 1992.
4. Peychev, V, V. Dimitrov. Dynamic WAP application for information search in legal documents. – In: Proceedings of 31st spring conference of the Union of Bulgarian Mathematicians, Borovets, 2002

Формальная модель оффициального документа

Ваньо Пейчев

Факультет математики и информатики, Софийский университет
E-mail: vanyo@fmi.uni-sofia.bg

(Р е з ю м е)

Формализованное представление документов ускоряет их доступность и обработку. Нормативные акты – типичный пример структурирования документов. Необходимость ускорения процессов обработки и максимально точное получение информации из нормативных актов большая. Проблем с нахождением документального содержания, соответствующего критериям поиска – основной в системах управления содержания (контент мениджмент системы (CMS)). Представленная модель структурированного документа и примерное описание закона с введенной модели – шаг в этом направлении.