Detecting the Erosion of Hierarchic Information Structures

P.D. Bruza, T.W.C. Huibers, J. van der Linden, and T. van Opstal

RUU-CS-93-10
March 1993

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Detecting the Erosion of Hierarchic Information Structures

P.D. Bruza, T.W.C. Huibers, J. van der Linden, and T. van Opstal

Technical Report RUU-CS-93-10
March 1993

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924-3275

Detecting the Erosion of Hierarchic Information Structures

P.D. Bruza[*†] *T.W.C. Huibers*[*] *J. van der Linden* [‡]
T. van Opstal[‡]

Abstract

In this paper the problem of detecting erosion in hierarchical information structures is studied. This problem manifests itself particularly in environments where there is little or no possibility to guarantee structures in accordance with a specification and furthermore where there is a high frequency of mutation of structures. The notion of information structure is formalized in terms of so called structural expressions. The erosion of information structures is paralleled by increased optionality in the associated structural expressions. In order to detect erosion the *Choice Normal Form* for structural expressions is introduced. In this normal form all implicit optionality is made explicit. It is shown how arbitrary structural expressions can be brought into the normal form via transformations which are semantic preserving. A graphical formalism is presented whereby analysis of erosion is made more convenient.

1 Introduction

In the world of publishing there is increasing competition to produce information products quicker and cheaper for narrowing market segments. Crucial in this regard is quick retrieval of the relevant information to fill in the content of a new product. Unfortunately, in many situations, quick product development is hampered because the raw material of the information products, the texts, are stored in a layout-oriented fashion suitable for a particular type-setting system [RP92]. In this way, pages can be quickly printed, but in order to create a new product the developer must sometimes know precisely which pages contain the relevant fragments of information. When these pages have been identified, they are fed to the type setting system, printed out and bundled to form the

[*]Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. e-mail: peterb@cs.ruu.nl

[†]This work was partly performed while employed at the *Katholieke Universiteit Nijmegen*

[‡]Samsom H.D. Tjeenk Willink, P.O. Box 4, 2400 MA Alphen aan den Rijn, The Netherlands

product. Sometimes there is no possibility to retrieve pages via content-based queries. In such cases, it takes years of experience for the product developer to become familiar enough with the material in order to develop products quickly and efficiently.

In short, the product developer does not design a product at the conceptual level, but typically works at the physical page level. This is in some ways comparable to the situation in the days of pre-relational databases. Developers in those days struggled with aspects related to the physical storage of the information. The advent of the conceptual approach to database design allows designers to first focus on the "what" of the problem, that is, the conceptual aspects, and produce a so called *conceptual description*. This can thereafter be (automatically) translated to an internal representation where aspects such as physical storage are taken into account (the "how" aspects) .

A similar line of development is now emerging in the world of textual information. The lack of conceptual description of the information has been acknowledged and is being filled by standards such as SGML (Standard General Markup Language) [Smi89] and ODA (Office Document Architecture) [HH86]. These standards are doing for the world of textual information what the conceptual schema approach did for database development. Information products are no longer a series of amorphous pages; they have a structure. This structure is not only useful for disclosing the information in the product [Wri92], but can be maintained in accordance with a structural specification. The structural specification opens the door to designing information products at a conceptual level and thereby promotes development in a media independent way.

The ability to maintain the structure of a product is important, particularly for those products with a high frequency of mutation. As mentioned above, in the current situation products are typically not developed from a conceptual description, so structural integrity cannot be guaranteed. As a consequence, the structure of the product can become more and more chaotic. We will refer to this process as structural erosion. It is the central theme of this paper.

A concrete example of structural erosion can be found in [HvL92]. This example deals with a rapidly mutating information product based on loose-leaf legal texts. An important element of these texts is a **commentary** on a legal article. Analysis of the structure of **commentaries** over a period of time revealed the following: Sometimes a **commentary** was a short introductory paragraph, sometimes it was a table of contents, followed by the introduction. In other cases the structure of the **commentary** was an introduction followed by items of description broken down by legal article. Still in other cases the articles were preceded by a table of contents.

The research cited above revealed that some structural variations of an element are not due to changing product requirements, but appear more or less by chance, for example, an error, or a change of product developer. As the number of possible variations of an information product rises, so to does its inconsistency; sometimes the commentary is presented in one form, and in the following

version of the product it has another form. Such inconsistency may not only be annoying for the user but can hamper them in finding relevant information.

This paper presents the theoretical basis of a method designed for identifying structural erosion in hierarchical structures, the skeletons of information products. The insights for this theory emerged from a case study in a publisher's house [HvL92]. Our point of departure is the *Information Systems Paradigm* which features the well know distinction between the conceptual description of information and the information itself. An important component of the conceptual description are so called *structural expressions*, with which hierarchical structures are specified. Structural expressions are brought into the *Choice Normal Form* so that structural erosion can be detected. The erosion can thereafter be filtered out in a restructuring process.

2 Formalization of the Conceptual Description of Hierarchic Information Structures

Usually information is structured according to some rules. For example, if the information has the form of a book, it usually consists of chapters, a chapter consists of sections, and so forth. Another example is legal texts. These generally must adhere to fairly strict guidelines. Adoption of the Information Systems Paradigm (see figure 1) implies that an explicit distinction be made between the rules and the information it governs [Bub86]. More specifically, a *conceptual description* describes the rules pertaining to the structure of the stored information. The *information base* contains the actual information. The information base is sometimes referred to as an instantiation (population) of the conceptual description. For the research presented in this paper the *information processor* and *user interface* are not relevant.

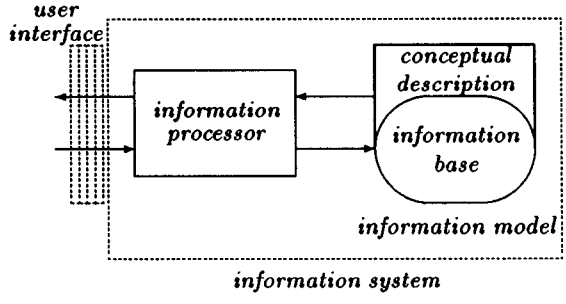

Figure 1: Information Systems Paradigm

The Information Systems Paradigm is a general framework applicable in various contexts. As such, a number of formalisms are available to specify a conceptual description. In traditional database design support for the speci-

fication of entity and relationship types are featured. In the world of textual information, context free grammars are a well known formalism. (See [GT87, BvdW92, Mac91, ISO86]). The ISO document standard SGML uses context free grammars as the core of its so called *document type definitions*. Basically, a document type definition is a specification of a set of permitted hierarchically structured documents, or said equivalently in terms of formal language theory: the DTD is a grammar where each parse tree generatable from the grammar corresponds to a valid structured document. In terms of the Information Systems Paradigm, a set of context free production rules can be viewed as constituting a conceptual description, and the associated parse trees as being valid elements of the population of the information base. As SGML is widely used in the publisher's world and together with the fact that context free rules form a useful mechanism for structural specification, we pursue the grammar-based approach further.

The starting point is the notion of a conceptual description. We follow a similar line to [BvdW92] in which a basic formal language approach is taken, but where a strict separation is made between structure and content.

Definition 2.1

A conceptual description *is a tuple* $\Sigma = (E, \mathcal{R})$ *where*

- E *is a set of names denoting* structural elements
- $\mathcal{R}$ *is a set of context free* production rules

□

The structural element names E can be compared to the entity type names of traditional database conceptual schemata. Basically, a structural element name denotes a class of structural elements. For example, the name `commentary` introduced in the previous section is such a class name. The information base contains a population of structural elements which are members of the class `commentary`. Users often use the strings in E as footholds for localizing a search for information. The symbol ϵ (ϵ in E) will be used as the name of the empty structural element.

A production rule specifies how a structural element is broken down. For example, the rule

`Book` $\rightarrow$ `Title,Chapter+`

specifies that a book constitutes a title followed by one or more chapters. As the above example illustrates, a production rule has a left hand side (LHS) and a right hand side (RHS). The LHS is the name $N, (N \in E)$ of a structural element. The RHS describes a so-called *structural expression*. Structural expressions will play a central role in this paper. It is these expressions which are normalized for the purpose of detecting so called structural erosion. The normal form thereafter serves as the basis of a restructuring process for the rectification of the erosion.

Structural expressions are basically an extension of the regular expressions of formal language theory. They are defined as follows:

- Each member of E is a structural expression
- If `A` and `B` are structural expressions, so are `(A,B)`, `(A | B)` and `(A & B)`
- If `A` is a structural expression, so are `A*`, `A+`, `A?` and `(A)`

In the above,

- `A,B` denotes that `A` precedes `B`
- `A & B` denotes the order of `A` and `B` is arbitrary
- `A | B` denotes a choice between `A` and `B` and
- `A?` denotes that `A` is optional, that is ϵ `| A`
- In `A+`, the + operator is used for denoting a sequence of one or more `A`'s.
- Finally, `A*` denotes optional repetition, that is `(A+)?`

$\mathcal{L}_E$ will be used to denote the structural expressions over E. Formally then, the production rules P are a subset of $E \times \mathcal{L}_E$. The following is an example of a production rule

`Book` $\rightarrow$ `ContentsTable?,(Chapter+ | Paragraph*),(Appendix & Index)`

which specifies that a book may have a table of contents, consists of either a series of one or more chapters or a series of zero or more paragraphs, followed by an appendix and index which may be in any order.

The semantics of structural expressions can be established in the usual way by mapping structural expressions to formal languages [LP81]. Structural expressions with the same semantics are said to be equivalent. The normalizations of structural expressions presented later in this paper are semantic preserving. This is important because in our normalization of expressions to a form suitable for erosion detection, we do not wish to disturb the intention of the structures suggested by the expressions.

Note the above definition of a conceptual description consists solely of rules and structural element names. This can be viewed as a pool of rules from which hierarchical information structures can be defined using a *information structure description*. This notion is defined as follows:

Definition 2.2

Let $\Sigma = (E, \mathcal{R})$ *be a conceptual description. An* information structure description *is a tuple* (P, S) *where*

- $P \subseteq \mathcal{R}$

- $S \in E$, *the* start symbol

□

An information structure description is basically a context free grammar. The parse trees derivable from this grammar correspond to *information structures*. These reside in the information base and are composed of the structural elements mentioned earlier.

3 The Erosion of Information Structures

The second law of thermodynamics pronounces that everything proceeds towards chaos. So it is with production rules, in the sense that via a series of mutations a rule ends up implying more and more possibilities, or choices. Figure 2 depicts the atomic mutations of a structural expression `A` via the addition of an operator. Note that if `A` constitutes the RHS of a rule it implicitly specifies the constraint that there is a *single* occurrence of `A`. Now, if `A` mutates to `A?` the constraint is weakened as this signifies zero *or* one occurrences of `A`. Said differently, there is now a choice between `A` and ϵ, the empty information structure. This is an example of erosion. An other example is the mutation from `A` to `A+`. This means that the constraint that there be a *single* occurrence of `A` is weakened to there being *at least one* `A`. In other words, after the mutation there are the possibilities `A` or `A,A` or `A,A,A` etc. A similar situation arises when `A?` mutates to `A*`. When `A?` or `A+` mutate to `A*` erosion also occurs as they both imply an increase in the number of inherent possibilities.

The mutation from `A` to `A | B` is an increase in optionality and therefore erosion. Note that the mutation from `A` to `A,B` is not erosion as it does not imply an increase in the number of inherent possibilities. However, the mutation from `A,B` to `A & B` constitutes erosion because this represents a shift from a mandatory ordering to a choice between the ordering `A,B` or `B,A`.

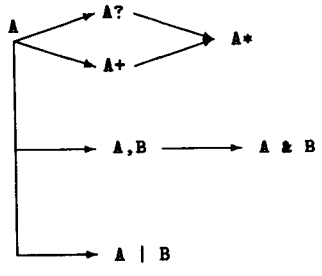

Figure 2: "From order to chaos"

Mutations often arise out of the need to extend an information structure. For example, a product developer may consciously mutate some rules in order to increase the possibilities of a product. In practice, not all mutations are atomic. For example, a frequently occurring mutation is from `A` to `A,B*`. Note that this comprises two atomic mutations: From `A` to `A,B` to `A,B*`. The first atomic mutation is not considered erosion, whereas the second is. Therefore, the combination of the two mutations is considered to be erosion.

A concrete example of erosion

In this example we will trace the mutations of a table of contents from an information product on environmental pollution laws [HvL92]. Environmental pollution is currently a very relevant topic so this product has a high frequency of mutation. The initial table of contents is depicted in figure 3. Formally, the table of contents is described by the following Information Structure Description:

- Production Rules: $P = \{$`ContentsTable` $\rightarrow$ `(Sectno,Title,Pageno)+`$\}$
- Start Symbol: $S =$ `ContentsTable`

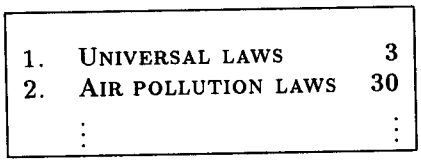

Figure 3: The original Table of Contents

Soon the sections become divided into subsections (see figure 4). Some sections have subsections and some do not. This constraint is expressed by using optional repetition:

$$\texttt{ContentsTable} \rightarrow \left(\begin{array}{l} \texttt{(Sectno,Title,Pageno),} \\ \texttt{(Subsectno,Title,Pageno)*} \end{array} \right)+$$

After some time the developer wishes to extend the product to include aspects of jurisprudence which leads to a completely new option in the current structure description.

$$\begin{array}{lll} \texttt{ContentsTable} & \rightarrow & \left(\begin{array}{l} \texttt{(Sectno,Title,Pageno),} \\ \texttt{(Subsectno,Title,Pageno)*} \end{array} \right)+ \\ & & | \\ & & \texttt{(Decision,Date,Code,Definition)+} \end{array}$$

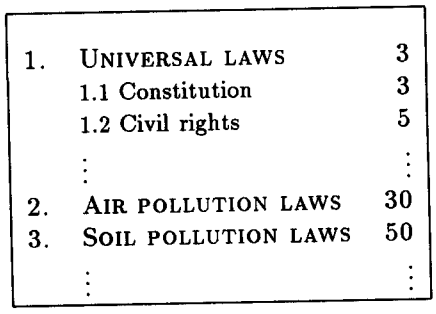

Figure 4: The changed Table of Contents

As an aside, this last mutation has undesirable consequences for the consistency of the table of contents. Previously searching involved page numbers, now searching for a decision involves looking up dates and codes.

The erosion is not yet complete. Passing the time, some elements become optional.

```
ContentsTable → ( (Sectno,Title,Pageno),
                  (Subsectno,Title,Pageno)* )+
                |
                (Decision?,Date,Code?,Definition)+
```

This example of erosion will be used throughout this paper. For notational convenience structural element names are abbreviated as follows:

```
CT → ((S,T,P),(Ss?,T,P)*)+ | (De?,Da,C?,Df)+
```

4 The Choice Normal Form, an enumeration of possibilities

In order to detect erosion it is necessary to analyze the structural expressions in a given Information Structure Description. We have seen in the previous section that erosion has to do with an increase in the number of inherent possibilities. Note, then, that `&`, `?` and `*` entail implicit choices. For example, `A?` denotes two possibilities: `A` or ϵ. It is necessary in erosion detection to have all such possibilities explicitly represented so that the developer can easily see which possibilities are desirable or not.

The Choice Normal Form is an enumeration of explicit possibilities. For this reason `*`, `&` and `?` are not featured.

Definition 4.1
Let (P, S) be an information structure in the context of the conceptual

description $(E, \mathcal{R})$. *Furthermore, let* $X \in \mathcal{L}_E$. *Then,* X *is in* Choice Normal Form (CNF) *if and only if*

$$X = \mathtt{A}_1 \mid \ldots \mid \mathtt{A}_i$$

where $\mathtt{A}_j = \mathtt{B}_1, \ldots, \mathtt{B}_k, 1 \leq j \leq i$ *such that*

- $\mathtt{B}_l \in E$, *or*
- $\mathtt{B}_l = \mathtt{C}^{m\rightarrow}$

where $1 \leq l \leq k, m > 0$ *and* C *in* CNF. □

Remark 4.1
The notation $\mathtt{C}^{m\rightarrow}$ *is a convenient way to signify that there are at least* m *elements in a sequence of* C*'s. Formally,* $\mathtt{C}^{m\rightarrow} = \mathtt{C}^m \mid \mathtt{C}^{m+1} \mid \ldots$ *where* $\mathtt{C}^i$ *denotes a sequence comprising* i C*'s:* $\underbrace{\mathtt{C}, \ldots, \mathtt{C}}_{i}$. *Note that there are infinite number of choices implied by* $\mathtt{C}^{m\rightarrow}$. *The purpose of this notation will become clear when we discuss the translation of mandatory repetition to* CNF. □

In order to bring an arbitrary structural expression into *CNF* it is necessary to filter out the *, & and ? operators without disturbing the semantics of the expression at hand. This is the purpose of the *Explicit Choice Function*:

Definition 4.2
Let A, B $\in \mathcal{L}_E$ *and* $N \in E$. *Then,*

$$\begin{aligned}
ECF(N) &= N \\
ECF(\mathtt{A+}) &= ECF(\mathtt{A})^{1\rightarrow} \\
ECF(\mathtt{A*}) &= ECF(\epsilon \mid \mathtt{A+}) \\
ECF(\mathtt{A?}) &= ECF(\epsilon \mid \mathtt{A}) \\
ECF(\mathtt{A|B}) &= ECF(\mathtt{A}) \mid ECF(\mathtt{B}) \\
ECF(\mathtt{A,B}) &= ECF(\mathtt{A}), ECF(\mathtt{B}) \\
ECF(\mathtt{A\&B}) &= ECF(\mathtt{A,B}) \mid ECF(\mathtt{B,A}) \\
ECF(\mathtt{(A)}) &= (ECF(\mathtt{A}))
\end{aligned}$$

□

Remark 4.2
With regard to translation of mandatory repetition: A+ *means that there is a sequence of* A*'s, but we do not know how long the sequence is. We only know that the sequence contains* at least one A. *In terms of the notation introduced earlier we express this as* $\mathtt{A}^{1\rightarrow}$. *Sometimes, however, it is known*

that there must be at least i `A`*'s in the sequence. Such cases will be signified by* $\mathtt{A}^{i\rightarrow}$.

We do not deem it necessary to specify a maximum for sequences because in this case the number of choices is finite. These choices can be specified by explicitly stating them. □

The Explicit Choice function removes all the hidden choices within an expression. It will be evident, however, that after applying the function the resultant expression need not be in Choice Normal Form. For example,

$$ECF(\mathtt{A,(B\ |\ C)}) = \mathtt{A,(B\ |\ C)}$$

The Choice Normal Form equivalent of this expression is `(A,B) | (A,C)`. Therefore, in order to bring *ECF* expressions into *CNF* the following two distributivity rules should be brought to bear. The symbol ≡ denotes semantic equivalence.

$$\begin{array}{lcl} \mathtt{A,(B\ |\ C)} & \equiv & \mathtt{(A,B)\ |\ (A,C)} \\ \mathtt{(B\ |\ C),A} & \equiv & \mathtt{(B,A)\ |\ (C,A)} \end{array}$$

In summary, employment of the Explicit Choice Function and thereafter distributing the sequence operator , over the choice operator | is sufficient to bring an arbitrary structural expression in Choice Normal Form.

Example 4.1

The Choice Normal Form of our running example is as follows:

$$\left(\begin{array}{l} \mathtt{(S,T,P)} \\ \mid\ \mathtt{((S,T,P),(Ss,T,P)}^{1\rightarrow}\mathtt{)} \\ \mid\ \mathtt{((S,T,P),(T,P)}^{1\rightarrow}\mathtt{)} \end{array}\right)^{1\rightarrow}$$

$$\mid$$

$$\left(\begin{array}{l} \mathtt{(De,Da,C,Df)} \\ \mid\ \mathtt{(Da,C,Df)} \\ \mid\ \mathtt{(Da,Df)} \\ \mid\ \mathtt{(De,Da,Df)} \end{array}\right)^{1\rightarrow}$$

□

Expressions which are in Choice Normal Form can be simplified via the usual transformations from formal language theory. As a special notation $(i \rightarrow)$ has been introduced additional transformations are necessary. We have found the following transformations to be useful. Here it is assumed that `A` and `B` are

structural expressions and $i, j > 0$.

$$
\begin{array}{lcll}
\mathbf{A}^{i}, \mathbf{A}^{j\rightarrow} & \equiv & \mathbf{A}^{j\rightarrow}, \mathbf{A}^{i} & \equiv \mathbf{A}^{i+j\rightarrow} \\
\mathbf{A}^{j\rightarrow}, \mathbf{A}^{i\rightarrow} & \equiv & \mathbf{A}^{i\rightarrow}, \mathbf{A}^{j\rightarrow} & \equiv \mathbf{A}^{i+j\rightarrow} \\
\epsilon, \mathbf{A}^{i\rightarrow} & \equiv & \mathbf{A}^{i\rightarrow}, \epsilon & \equiv \mathbf{A}^{i\rightarrow} \\
(\mathbf{A}^{i\rightarrow})^{j\rightarrow} & \equiv & \mathbf{A}^{ij\rightarrow} & \\
\epsilon^{i\rightarrow} & \equiv & \epsilon & \\
(\mathbf{A}^{i\rightarrow} \mid \mathbf{A}^{j\rightarrow}) & \equiv & \mathbf{A}^{min(i,j)\rightarrow} & \\
(\mathbf{A}^{1\rightarrow} \mid \mathbf{B})^{i\rightarrow} & \equiv & (\mathbf{A} \mid \mathbf{B})^{i\rightarrow} & \\
(\mathbf{A} \mid \epsilon)^{i\rightarrow} & \equiv & \mathbf{A}^{i\rightarrow} \mid \epsilon &
\end{array}
$$

(The proofs of these transformations can be acquired by filtering out the $\rightarrow$'s with the formal language equivalent).

Example 4.2

$$
\begin{array}{ll}
(\mathbf{A}^{2\rightarrow} \mid \epsilon)^{3\rightarrow}, \mathbf{A}^{5\rightarrow} & \equiv \\
((\mathbf{A}^{2\rightarrow})^{3\rightarrow} \mid \epsilon), \mathbf{A}^{5\rightarrow} & \equiv \\
(\mathbf{A}^{6\rightarrow} \mid \epsilon), \mathbf{A}^{5\rightarrow} & \equiv \\
(\mathbf{A}^{6\rightarrow}, \mathbf{A}^{5\rightarrow}) \mid (\epsilon, \mathbf{A}^{5\rightarrow}) & \equiv \\
\mathbf{A}^{11\rightarrow} \mid \mathbf{A}^{5\rightarrow} & \equiv \\
\mathbf{A}^{5\rightarrow} &
\end{array}
$$

□

5 Analyzing *CNF* Expressions for Erosion

Once a structural expression has been transformed into Choice Normal Form all possibilities are explicitly represented. Erosion detection, then, amounts to identifying unwanted possibilities. The algebraic form of the CNF expression can be translated to a graphical formalism for a more convenient perusal of possibilities. Figure 5 shows the general scheme for translating an arbitrary *CNF* expression into a graphical representation by means of the function Δ. The $i-1$ next to the loop in the translation of $\Delta(\mathbf{A}^{i\rightarrow})$ means that the loop has to be followed *at least* $i-1$ times. In this way is a sequence of at least i $\Delta(\mathbf{A})$'s is forced. Note that the translation of $\Delta(\mathbf{A}^{1\rightarrow})$ involves no constraints on the loop. Figure 6 depicts the graphical representation of our running example.

The graphical representation (such as depicted in figure 6) can be compared to a road map. A journey through the graph corresponds to a possible information structure. The build up of such a structure occurs as follows: Starting from the root node one can traverse the flows. Passing through a labeled node means that the corresponding name denotes a participating structural element. Nodes without names can be compared to a road junction. A loop back to a junction

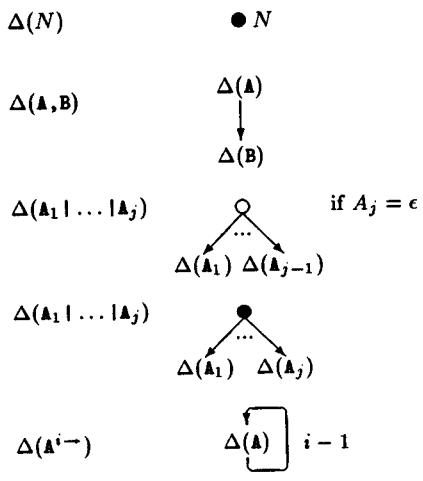

Figure 5: *CNF* expressions and their graphical representation

node allows a choice to be made between the same or alternative substructures. A journey ends at any of the leaf nodes.

Erosion can be analyzed by examining junctions, loops and so called *open nodes* which are denoted by ○. Each of these express optionality. All branches dangling from an open node depict substructures that are not mandatory components of the information structure. The question should be asked whether it is indeed the intention that all of these substructures may be optional. A junction with many branchings may depict an advanced stage of erosion. Analysis should be performed to determine whether all of the possibilities suggested by the branchings are indeed desirable. Furthermore, combinations of junctions, loops and open nodes should gain special attention because it almost certainly corresponds to a weak structural specification. For example, consider figure 7. This corresponds to the formal language $\{A_1, \ldots, A_j\}^*$. This is the weakest form of specification because it entails all possibilities over the alphabet $A_1, \ldots, A_j$.

Looking at our example (see figure 6) we see that there are no open nodes. There is, however, a considerable amount of branching combined with loops. We could therefore conclude that the table of contents is fairly eroded. It was indeed concluded in [HvL92] that only a fairly small subset of the possibilities suggested by figure 6 did constitute desired tables of contents.

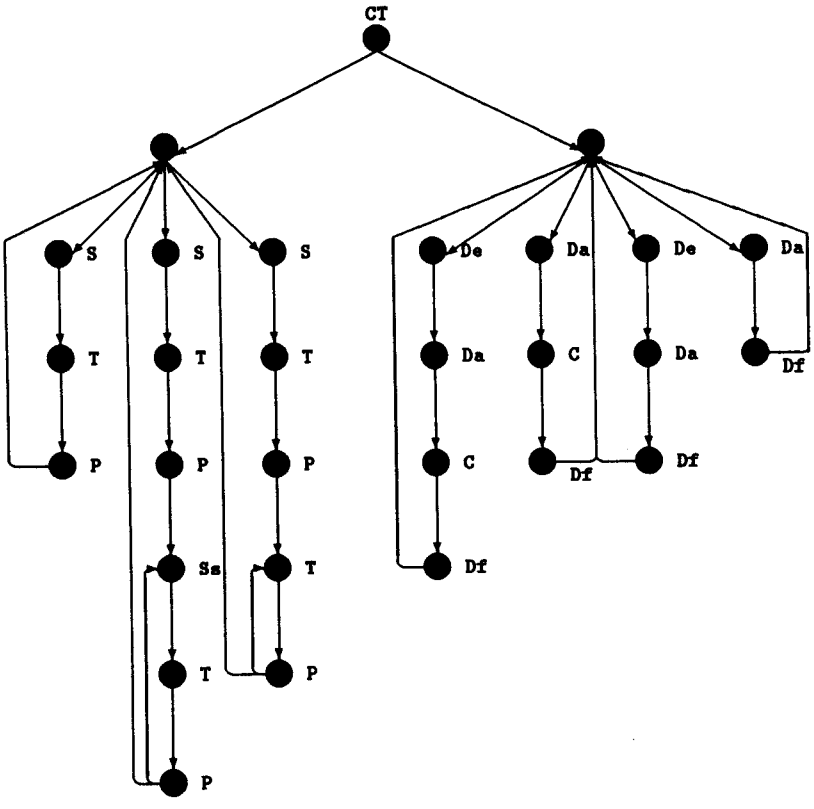

Figure 6: Graphical representation of the running example

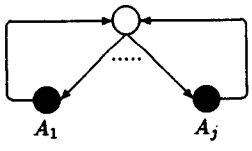

Figure 7: The weakest structural specification

6 Conclusions

The theory presented has been tested with real life examples like the one running through this paper. It was surprising in some cases how much hidden optionality there was. For example, one seemingly harmless structural expression revealed one hundred and twenty-eight explicit possibilities after being rendered in *Choice Normal Form*. In such cases the graphical form of the expression proved unwieldy. Nevertheless, the formalism proved useful in communicating structural specifications to product managers who have no background in formal language theory.

The transformation of an arbitrary structural expression to the normal form is a straightforward process as it involves purely syntactic transformations. The choice for purely syntactic transformations was deliberate as semantic-based transformations are sometimes complex, and furthermore, difficult to automate.

Even though the theory is hinges on the notion of an Information Structure Specification, it can applied in environments where these specifications are unavailable. In such cases one need only reverse-engineer the structural specification at different points in the structure's lifetime. The resulting specifications can then be analyzed for erosion. This technique was applied with success in the tests mentioned above.

Even though the normal form makes all possibilities explicit, more practical experience is necessary in formalizing the analysis of erosion. First steps in this direction are described in [HvL92]. Structural expressions are classified according to the types of operator used. On the basis of this a formalized notion of the consistency of a structural specification can be defined.

Acknowledgements

The authors would like acknowledge the foresight of Jan Maasdam from *Samsom-Sijthoff/Intermedia* for promoting this research.

Contents

References

[Bub86] J.A. Bubenko. Information system methodologies - a research view. In T.W. Olle, H.G. Sol, and A.A. Verrijn Stuart, editors, *Information System Design Methodologies: Improving the Practice*, pages 289–318. North-Holland, 1986.

[BvdW92] P.D. Bruza and T.P. van der Weide. Stratified Hypermedia Structures for Information Disclosure. *The Computer Journal*, 35(3):208–220, 1992.

[GT87] G. Gonnet and F. Tompa. Mind Your Grammar: a New Approach to Modelling Text. In *Proceedings of the Thirteenth Conference on Very Large Data Bases*, pages 339–346, 1987.

[HH86] W. Horak and K. Hoffman. Office Document Architecture and Office Document Interchange Formats–Current status of international standardization. In *ESPRIT'85: Status Report of Continuing Work*, pages 903–917. The Commision of the European Communities, Elsevier Science Publishers, 1986.

[HvL92] T.W.C. Huibers and G-J. van Leeuwen. Structuuronderzoek bij een Uitgeverij (Structure Research at a Publishing House). Master's thesis, University of Nijmegen, 1992. (In Dutch).

[ISO86] ISO. *Information Processing - Text and Office Systems - Standard General MarkUp Language (SGML).* ISO8879, 1986.

[LP81] H.R. Lewis and C.H. Papadimitriou. *Elements of the Theory of Computation.* Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[Mac91] I.A. MacLeod. A Query Language for Retrieving Information from Hierarchic Text Structures. *The Computer Journal*, 34(3):254–264, 1991.

[RP92] L.R. Reynolds and S.J. Perose. Electronic Books. *BYTE*, pages 263–268, June 1992.

[Smi89] J.M. Smith. Standard Generalized Markup Language and related standards. *Computer Communications*, 12(2):80–84, April 1989.

[Wri92] H. Wright. SGML Frees Information. *BYTE*, pages 279–286, June 1992.