Refining the Taxonomy of XML Schema Languages. A new Approach for Categorizing XML Schema Languages in Terms of Processing Complexity

Maik Stührenberg
Christian Wurm
Bielefeld University

The genesis of this talk

In the beginning...

The genesis of this talk

In the beginning...

- Some years ago the department for computational linguistics and text technology changed areas of specialization because of staffing
- Where once were people like Andreas Witt were involved in doing text technology we now have two professorships:
  - one for mathematical and computational linguistics and
  - one for applied computational linguistics and text technology

The genesis of this talk

The problem is as follows:

- The professorship for applied computational linguistics and text technology has not been assigned for three years (but will be assigned in a short time)
- So far I tried waving the XML flag as the last member of the former text technology working group
- I was searching for a topic that could bring the members of our department together – that is: me (the XML guy) and my colleagues (formal logics guys)
- The results of these efforts are presented right here

Overview of the rest of this talk

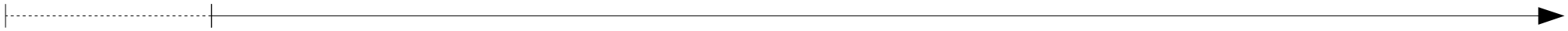
- XML schema languages – too many, too few?
- The need for clarification
- The Murata hierarchy of XML schema languages
- Confronting the real world with formal grammars
- Enhancing the Murata hierarchy by rendering it more precisely
- Conclusion

XML schema languages – too many, too few?

- XML-encoded data is a vital part of everyday's business
- Often data instances appear first and afterwards a document grammar for describing these instances has to be developed
- The question that arises at this point is: what kind of document grammar to choose – what schema language?

XML schema languages – too many, too few?

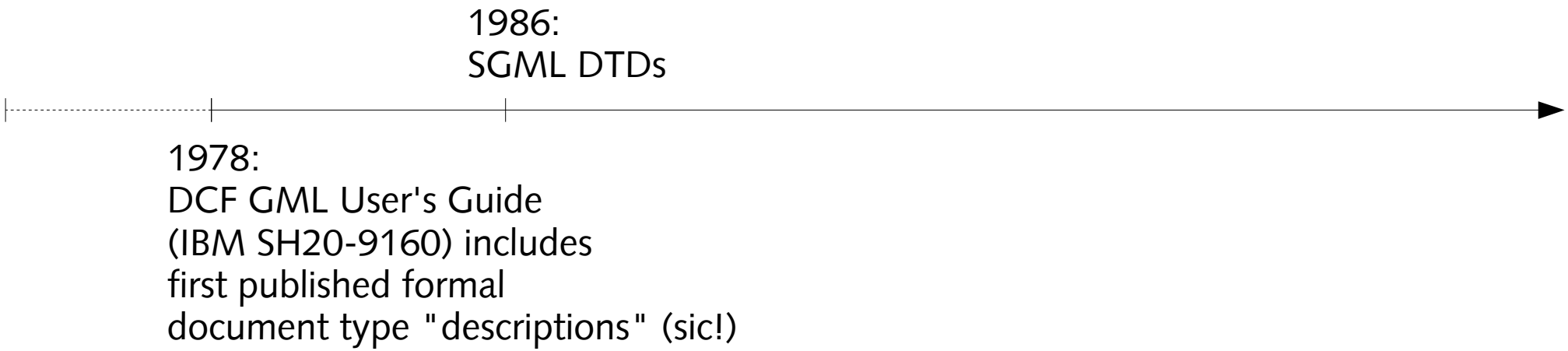
Evolution of document grammars



1978:
DCF GML User's Guide
(IBM SH20-9160) includes
first published formal
document type "descriptions" (sic!)

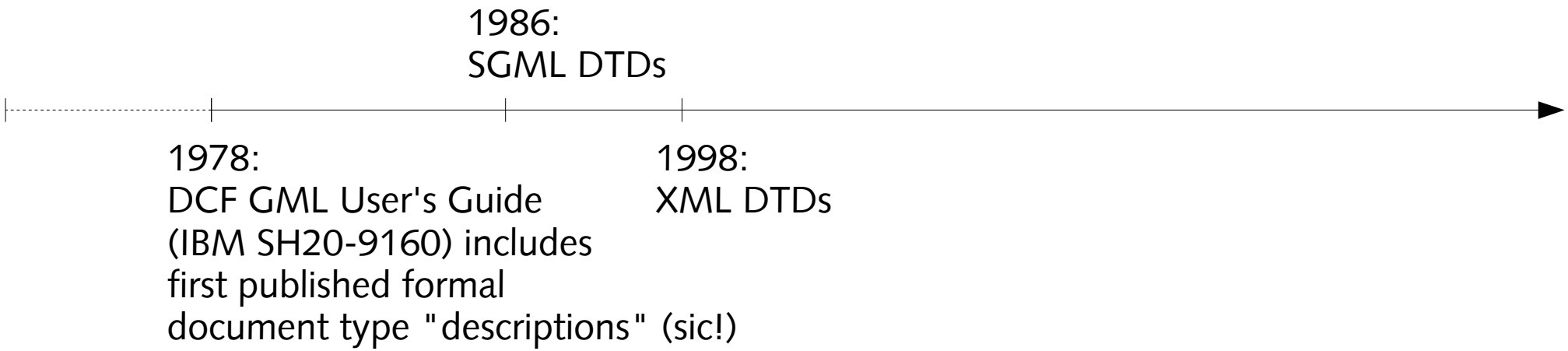
XML schema languages – too many, too few?

Evolution of document grammars



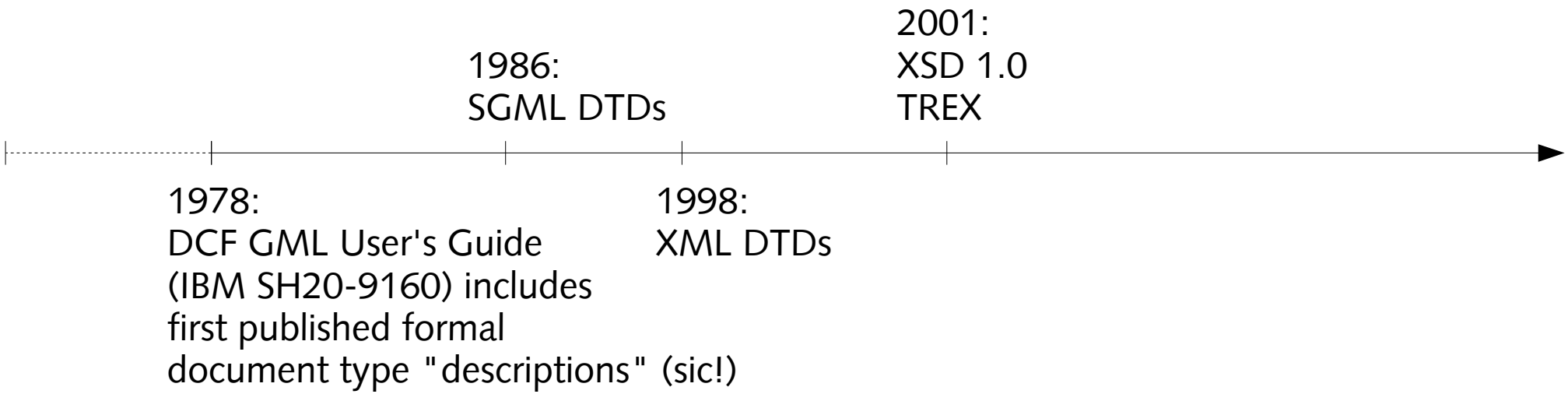
XML schema languages – too many, too few?

Evolution of document grammars



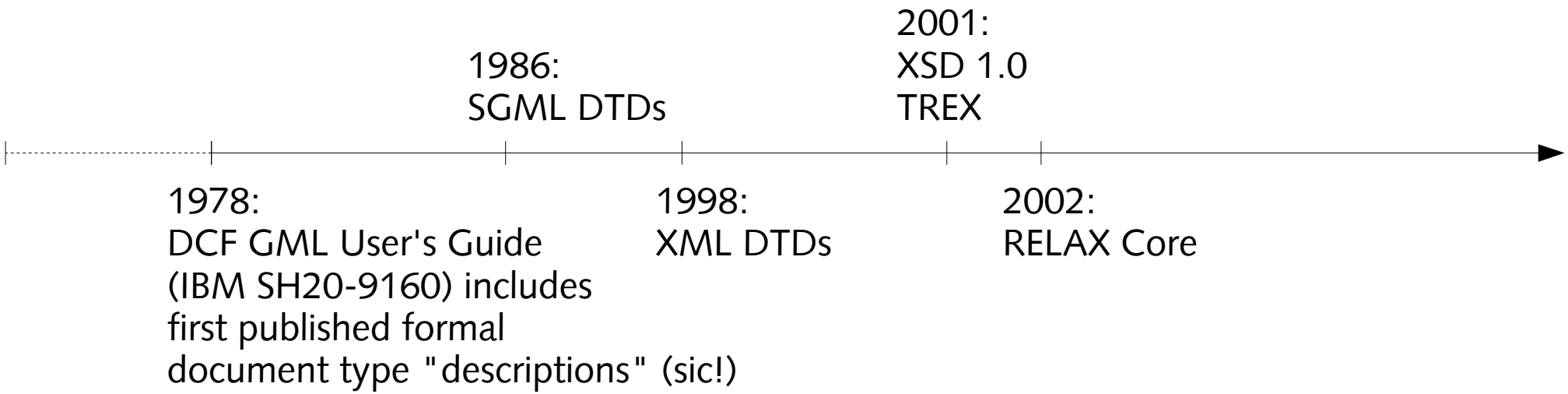
XML schema languages – too many, too few?

Evolution of document grammars



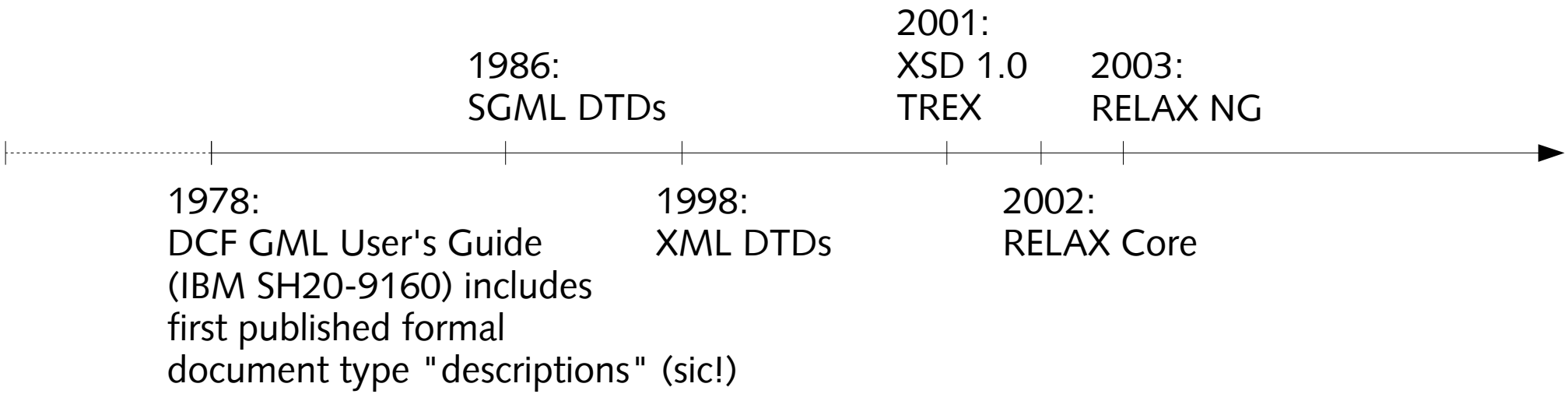
XML schema languages – too many, too few?

Evolution of document grammars



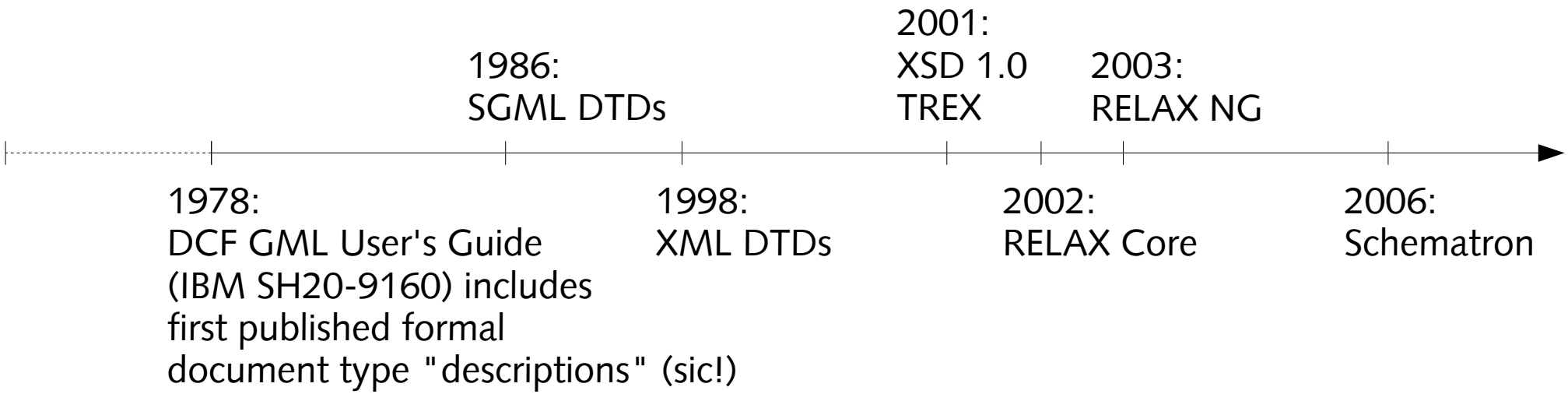
XML schema languages – too many, too few?

Evolution of document grammars



XML schema languages – too many, too few?

Evolution of document grammars



XML schema languages – too many, too few?

Evolution of document grammars

1955:
Noam Chomsky's Theory of Formal Grammars

1978:
DCF GML User's Guide (IBM SH20-9160) includes first published formal document type "descriptions" (sic!)

1986:
SGML DTDs

1998:
XML DTDs

2001:
XSD 1.0
TREX

2002:
RELAX Core

2003:
RELAX NG

2006:
Schematron

XML schema languages – too many, too few?

Some more words about what this is all about:

- In this talk we try to bring together both sides of the coin, the formal aspects and the technical, XML aspects
- We will focus on grammar based XML schema languages following the definition of Costello and Simons 2008:

> *"A grammar-based schema language specifies the structure and contents of elements and attributes in an XML instance document. For example, a grammar-based schema language can specify the presence and order of elements in an XML instance document, the number of occurrences of each element, and the contents and datatype of each element and attribute."*

XML schema languages – too many, too few?

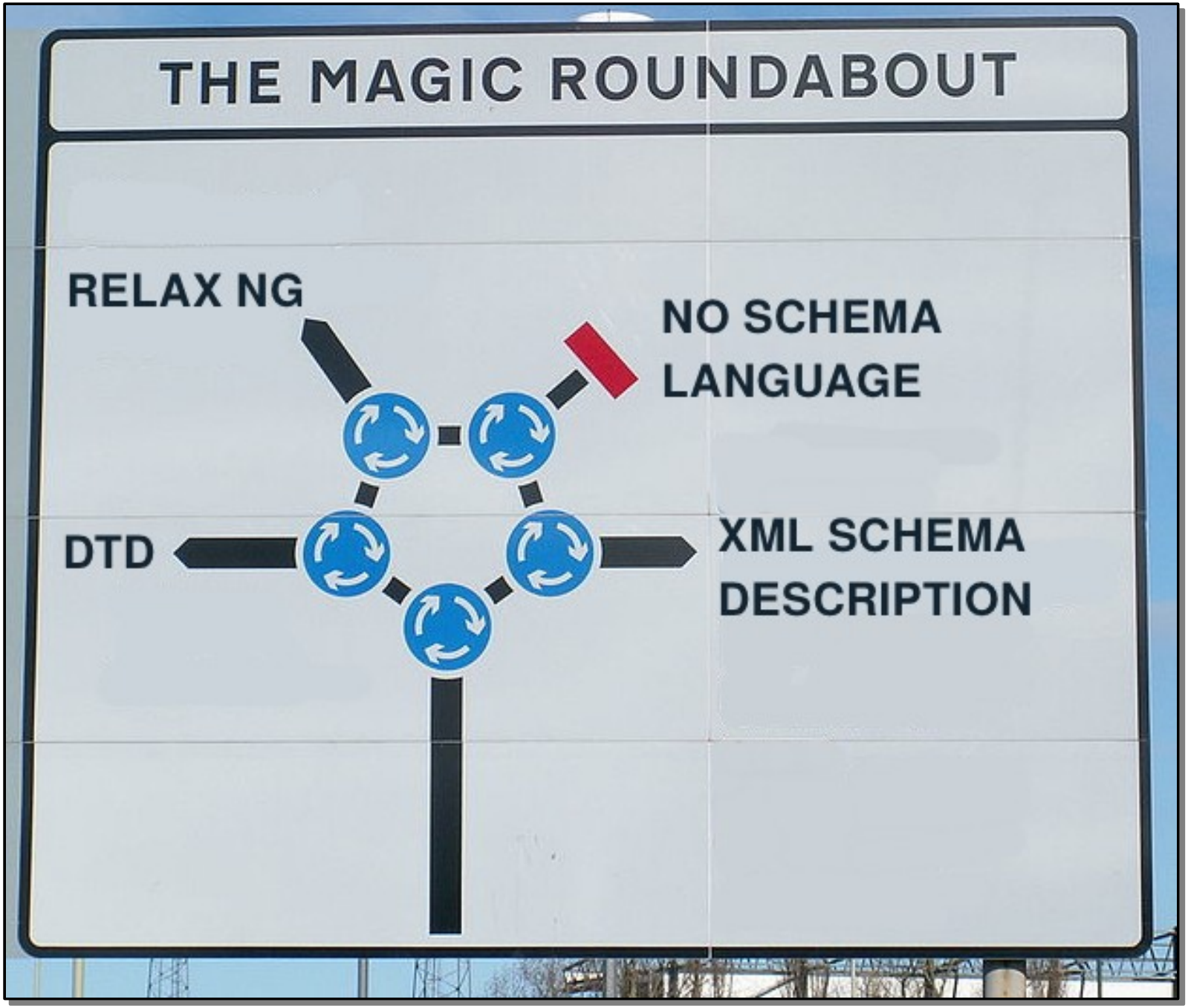
Some more words about what this is all about:

- In contrast, rule-based or constraint-based languages such as Schematron or the Constraint Language in XML (CLiXML) are not observed

  > *"A rule-based schema language specifies the relationships that must hold between the elements and attributes in an XML instance document. For example, a rule-based schema language can specify that the value of certain elements must conform to a rule or algorithm."*

- To cut a long story short: we will focus on the three most used grammar based XML schema languages, that is XML DTD, XML Schema Description and RELAX NG – no DSD2

XML schema languages – too many, too few?



XML schema languages – too many, too few?

Top 5 reasons to choose a specific schema language:

XML schema languages – too many, too few?

Top 5 reasons to choose a specific schema language:

5. "I'm used to this language since decades"

XML schema languages – too many, too few?

Top 5 reasons to choose a specific schema language:

5. "I'm used to this language since decades"
4. "This is the schema language that is broadly supported"

XML schema languages – too many, too few?

Top 5 reasons to choose a specific schema language:

5. "I'm used to this language since decades"
4. "This is the schema language that is broadly supported"
3. "It uses the spiffy XML syntax"

XML schema languages – too many, too few?

Top 5 reasons to choose a specific schema language:

5. "I'm used to this language since decades"
4. "This is the schema language that is broadly supported"
3. "It uses the spiffy XML syntax"
2. "It uses a very easy syntax of its own – not the verbose XML syntax"

XML schema languages – too many, too few?

Top 5 reasons to choose a specific schema language:

5. "I'm used to this language since decades"
4. "This is the schema language that is broadly supported"
3. "It uses the spiffy XML syntax"
2. "It uses a very easy syntax of its own – not the verbose XML syntax,"
1. "I really like the funny name"

XML schema languages – too many, too few?

Additional reasons might be more important:

- "It allows me to express non-deterministic content models"
- "It allows me to re-use existing model groups"
- "It has a strong datatype library"
- "It allows for XML's referencing mechanism"

XML schema languages – too many, too few?

But what if we want to classify these XML schema languages according to their expressivity?

The need for clarification

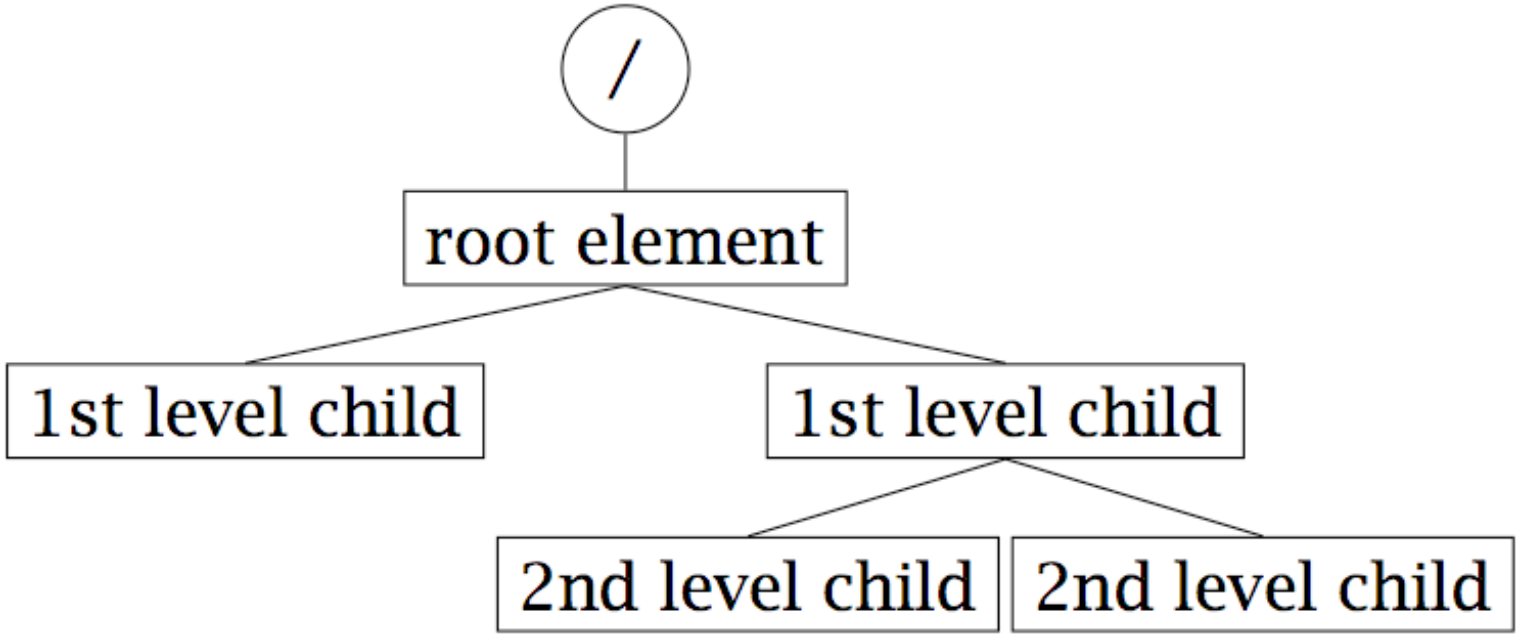
Before we start, we have to go through some bloody formal basics:

- During the writing process of this paper discussions between me and my colleague Christian often lead to misunderstandings because of slightly different concepts used in the XML compared to the world or formal logics

The need for clarification

Some words about trees and trees

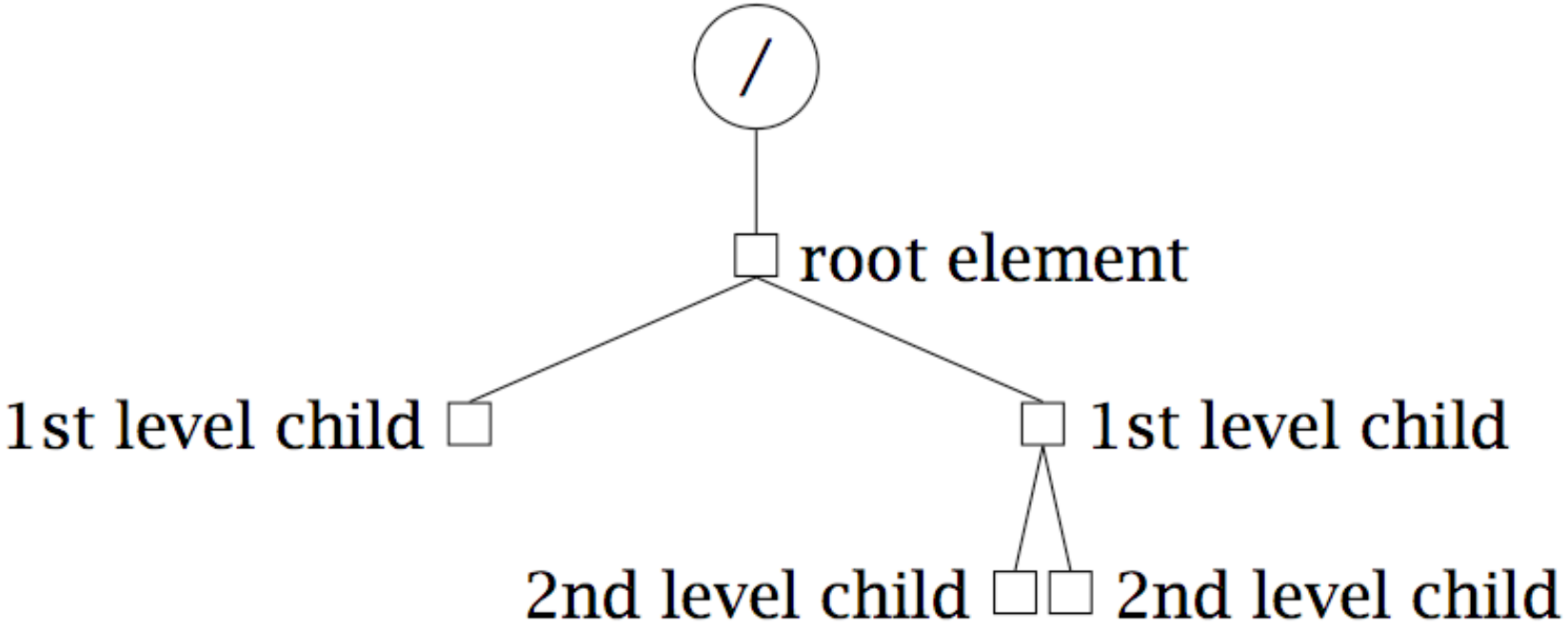
- When we are dealing with XML we are usually familiar with figures like this:



The need for clarification

Some words about trees and trees

- When we are dealing with formal grammars node labels and the differentiation between nodes and labels come into play:



Please keep this in mind when we're talking about nodes and node labels

The need for clarification

Some simple basics:

- (Markup) languages can be represented by grammars. A grammar is a 4-tuple (N,T,S,P), where:
  - N is a finite set of nonterminals,
  - T is a finite set of terminals,
  - $S \in N$ is a distinguished nonterminal called the start symbol,
  - P is a finite set of production rules of the form
    $\alpha \rightarrow \beta$
    where α and β are variables of sequences of terminals or nonterminals

The need for clarification

Some simple basics:

- Furthermore, grammars can be differentiated by gradually increasing the restriction on the form of the production rules following Chomsky's hierarchy:
  - Type-0: unrestricted grammar
  - Type-1: context-sensitive grammar
  - Type-2: context-free grammar
  - Type-3: regular grammar

The need for clarification

But!

- These grammars were developed for characterizing strings
- Although some authors used these grammar classes to characterize XML DTDs as extended context-free grammars, i.e. on the right hand side of a production rule are regular expression allowed (Hopcroft 2000, Rizzi 2001):
  $A \rightarrow r$
  where r is a regular expression over N
- If we talk about XML we don't talk about strings – we talk about trees
- So we have to deal with tree grammars

The Murata hierarchy of XML schema languages

Luckily there are already extensive works regarding XML schema languages and formal grammars:

- Brüggemann-Klein and Wood 1992, 1997, 2002, 2004
- Brüggemann-Klein 1993
- Hopcroft et al. 2000
- Rizzi 2001
- Murata et al., 2001, 2005
- Sperberg-McQueen 2003
- Klarlund et al. 2003
- Comon et al. 2008

The Murata hierarchy of XML schema languages

Luckily there are already extensive works regarding XML schema languages and formal grammars:

- Brüggemann-Klein and Wood 1992, 1997, 2002, 2004
- Brüggemann-Klein 1993
- Hopcroft et al. 2000
- Rizzi 2001
- Murata et al., 2001, 2005
- Sperberg-McQueen 2003
- Klarlund et al. 2003
- Comon et al. 2008

The Murata hierarchy of XML schema languages

Murata et al. established a taxonomy of XML schema languages using formal language theory

- They formulated different classes of grammars, mainly
  - regular tree grammar (RTG)
  - single type tree grammar (STG)
  - local tree grammar (LTG)
- Any STG is an RTG, and any LTG is an STG

The Murata hierarchy of XML schema languages

The definition of an RTG:

- A regular tree grammar (RTG) is a 4-tuple (N,T,S,P), where
  - N is a finite set of nonterminals,
  - T is a finite set of terminals,
  - S ∈ N is a distinguished nonterminal called the start symbol,
  - P is a finite set of production rules of the form
    $A \rightarrow a(r)$
    where A ∈ N, a ∈ T, and r is a regular expression over elements of N
  - We call A the left hand side of a rule, a the terminal or label which is introduced by the rule, and r its content model
- We generally use uppercase letters for nonterminals, and lower-case letters for terminals
- Note that the nonterminals do not remain the labels of the (non-leaf) nodes they introduce, but are substituted by the terminal labels

The Murata hierarchy of XML schema languages

Competing rules:

- Two rules of an RTG are competing, if they introduce the same terminal nodes, but have different left hand sides
- Thus, A → a(r) and B → a(r') are competing
- In general, in an RTG we can merge any two rules which have the same left-hand side and introduce the same terminal, by merging their content models, because for any two regular expressions we can easily form a single expression which denotes is the union of both
- As a consequence, the concept of competing rules is the crucial point if we deal with determinism and ambiguity

The Murata hierarchy of XML schema languages

The definition of an LTG:

- A local tree grammar (LTG) is an RTG with no competing rules
- In an LTG, we have thus a one-to-one correspondence of nonterminals and terminals, which makes them very similar to context-free grammars – which is why CFG where originally used for characterizing DTDs (e.g. by Hopcroft et al. 2000 and Rizzi 2001)

The Murata hierarchy of XML schema languages

The definition of a STG:

- A single type tree grammar (STG) is an RTG, where competing nonterminals must not occur in the same content model

The Murata hierarchy of XML schema languages

The definition of an RCG:

- A restrained competition grammar (RCG) is an RTG, where competing nonterminals must not occur in the same content model and with the same prefix of nonterminals
- We thus disallow rules with identical left-hand side, terminals, and content models of the form (Γ A Δ) and (Γ B Δ'), where A and B are competing nonterminals, and where uppercase Greek letters refer to possibly empty sequences of nonterminals

The Murata hierarchy of XML schema languages

Ambiguity and deterministic content models

- To make assumptions about non deterministic content models interpretations of trees can be used
- An interpretation I of a tree t against a grammar G is a mapping from each node label of t, denoted by e, to a nonterminal N of the grammar, such that
  - I(e) is a start symbol when e is the root of t,
  - for each e and its daughter nodes e0, e1, ... en, there is a production rule A → a(r) in G, such that
    - I(e) is A,
    - the label of e is a,
    - $I(e_0)$, $I(e_1)$, ..., $I(e_n)$ matches r

The Murata hierarchy of XML schema languages

Ambiguity and deterministic content models:

- Any tree has at most one interpretation against a local tree grammar
- Any tree has at most one interpretation against a single type tree grammar
- A tree may have more than one interpretation against a regular tree grammar
- Thus, RTGs allow for non deterministic content models since non unique interpretations are possible
- Note, that it is crucial to distinguish between the label of a node and its interpretation:
  - the label of a node corresponds to its terminal in the production rule and it is immediately visible in the tree
  - by the interpretation of a node we denote the nonterminal by which the node label has been produced and that has to be inferred

Confronting the real world with the formal grammars

Meanwhile on the other side of the mirror
or
bringing formal logic and XML schema languages together

Confronting the real world with the formal grammars

For clarification reasons we use a simple example grammar for structuring arbitrary texts:

S → text(author,Title? (Section|Para))
Section → section(Title? (Section|Para))
Title → title(#pcdata)
Para → para(id, #pcdata|Xref)
Xref → xref(href,ε)

Confronting the real world with the formal grammars

An example XML instance could like like this:

```
<text author="maik">
  <title>A simple title</title>
  <section>
    <title>A section title</title>
    <para id="p1">Introductory para</para>
    <section>
      <title>A subsection title</title>
      <para>Some text with a reference: <xref href="p1"/>.</para>
    </section>
  </section>
</text>
```

Confronting the real world with the formal grammars

Following Murata, LTG correspond to DTDs

- DTDs supports only globally declared elements
  → no competing nonterminals

```
<!ELEMENT text (title?, (section | para)+)>
<!ATTLIST text author CDATA #IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT section (title?, (section | para)+)>
<!ELEMENT para (#PCDATA | xref)*>
<!ATTLIST para id ID #IMPLIED>
<!ELEMENT xref EMPTY>
<!ATTLIST xref href IDREF #REQUIRED>
```

Confronting the real world with the formal grammars

Following Murata, LTG correspond to DTDs

- DTDs supports only globally declared elements
  → no competing nonterminals

```
<!ELEMENT text (title?, (section | para)+)>
<!ATTLIST text author CDATA #IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT section (title?, (section | para)+)>
<!ELEMENT para (#PCDATA | xref)*>
<!ATTLIST para id ID #IMPLIED>
<!ELEMENT xref EMPTY>
<!ATTLIST xref href IDREF #REQUIRED>
```

Confronting the real world with the formal grammars

Following Murata, LTG correspond to DTDs

- DTDs supports only globally declared elements
  → no competing nonterminals
- DTDs support only deterministic content models
  → uniqueness of interpretation

```
<!ELEMENT text (title?, (section | para)+)>
<!ATTLIST text author CDATA #IMPLIED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT section (title?, (section | para)+)>
<!ELEMENT para (#PCDATA | xref)*>
<!ATTLIST para id ID #IMPLIED>
<!ELEMENT xref EMPTY>
<!ATTLIST xref href IDREF #REQUIRED>
```

Confronting the real world with the formal grammars

Following Murata, STG roughly correspond to XSDs

- XSDs supports globally and locally declared elements
  → for each production rule, nonterminals in its content model do not compete with each other, i.e. no two production rules share the same nonterminal in the left-hand side and share the same terminal in the right-hand side at the same time
  → within a content model there must not occur any competing nonterminals
- XSDs support only deterministic content models
  → uniqueness of interpretation
- Wildcards in XSDs could lead to non single type tree grammars but RCGs

Confronting the real world with the formal grammars

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="text">
    <xs:complexType>
      <!-- [...] -->
            <xs:element name="title" type="xs:string" minOccurs="0"/>
      <!-- [...] -->
    </xs:complexType>
  </xs:element>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="section" type="textType"/>
  <xs:element name="para">
    <!-- [...] -->
  </xs:element>
  <xs:attribute name="id"
  <xs:complexType name="textType">
    <xs:sequence>
      <xs:element ref="title" minOccurs="0"/>
      <xs:group ref="sectOrPara" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:group name="sectOrPara">
    <!-- [...] -->
  </xs:group>
</xs:schema>
```

globally declared title element vs. locally declared title element – the respecitve nonterminals do not compete with each other

Confronting the real world with the formal grammars

Following Murata, RTG roughly correspond to RELAX NG

- RELAX NG supports non deterministic content models and attribute-element-constraints (or other co-occurrence-constraints)
  → no uniqueness of interpretation

Confronting the real world with the formal grammars

To demonstrate non deterministic content models supported by RTGs (and RELAX NG) we have to extend our example grammar by adding a type information to the section element

- If the type is set to the value "global" other section child elements are allowed as part of the content model
- If its value is set to "sub" only para child elements are allowed

```
<define name="element.section">
  <choice>
    <oneOrMore>
      <element name="section">
        <optional>
          <ref name="element.title"/>
        </optional>
        <optional>
          <attribute name="type">
            <value>global</value>
          </attribute>
        </optional>
        <oneOrMore>
          <choice>
            <ref name="element.section"/>
            <ref name="element.para"/>
          </choice>
        </oneOrMore>
      </element>
    </oneOrMore>
    <oneOrMore>
      <element name="section">
        <optional>
          <ref name="element.title"/>
        </optional>
        <optional>
          <attribute name="type">
            <value>sub</value>
          </attribute>
        </optional>
        <ref name="element.para"/>
      </element>
    </oneOrMore>
  </choice>
</define>
```

```
<define name="element.section">
  <choice>
    <oneOrMore>
      <element name="section">
        <optional>
          <ref name="element.title"/>
        </optional>
        <optional>
          <attribute name="type">
            <value>global</value>
          </attribute>
        </optional>
        <oneOrMore>
          <choice>
            <ref name="element.section"/>
            <ref name="element.para"/>
          </choice>
        </oneOrMore>
      </element>
    </oneOrMore>
    <oneOrMore>
      <element name="section">
        <optional>
          <ref name="element.title"/>
        </optional>
        <optional>
          <attribute name="type">
            <value>sub</value>
          </attribute>
        </optional>
        <ref name="element.para"/>
      </element>
    </oneOrMore>
  </choice>
</define>
```

attribute-element-constraint:
content model is declared according to the value of the type attribute

Confronting the real world with the formal grammars

Interlude: a similar restriction could be realized by XSD 1.1 assertions:

```
<xs:element name="section">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="title" minOccurs="0"/>
      <xs:group ref="sectOrPara" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="global"/>
          <xs:enumeration value="sub"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:assert
        test="@type!='sub' and (child::para | child::section) or @type='sub' and
not(child::section)"
      />
  </xs:complexType>
</xs:element>
```

Enhancing the Murata hierarchy by rendering it more precisely

Although the Murata hierachy seems to be exhaustive, from the formal grammar point of view some enhancements could still be made:

- For any given grammar type the information needed to assure a unique interpretation of a local node or subtree is not stated clearly enough
- RCGs have some restrictions:
  - the unique interpretation of a label depends on a left sibling
  - although a asymmetric counterpart of RCG could be equally defined it would not really generalize the concept
- RTGs are very powerful but
  - the validating parser shown in Murata 2005 does not construct a unique interpretation of the document and does not enumerate all interpretations
  - is this kind of expressivity really needed in the wild?

Enhancing the Murata hierarchy by rendering it more precisely

We try to address these issues:

- Regarding the first issue please have a look at our paper in the proceedings
- Regarding the restrictions that apply to RCGs and the expressivity of RTGs our solution is as follows:
- Under the assumption that the full power of RTGs (and especially RELAX NG grammars) is often not needed we construct a new grammar type – the generalized restrained competition grammar (GRCG)

Enhancing the Murata hierarchy by rendering it more precisely

Let's recall the definition of an RCG:

- A restrained competition grammar (RCG) is an RTG, where competing nonterminals must not occur in the same content model and with the same prefix of nonterminals
- We thus disallow rules with identical left-hand side, terminals, and content models of the form (Γ A Δ) and (Γ B Δ'), where A and B are competing nonterminals, and where uppercase Greek letters refer to possibly empty sequences of nonterminals
  - this restriction concerns only the left context of the competing nonterminals
  - of course, there exists a parallel definition for the right context
- The problem is that these definitions lack some generalization, as they both generate different classes of languages, and there is no inclusion in either direction

Enhancing the Murata hierarchy by rendering it more precisely

Trying to generalize the RCG model – introducing the generalized restrained competition grammar (GRCG):

- In a GRCG, for any two competing nonterminals A and B within a single content model r, one of (Γ A Δ) and (Γ B Δ) fails to match r
  → (ABC|ADE) is allowed – (ABC|ADC) is not, if B and D are competing
- We now have generalized the restriction from the left (right, respectively) to the entire context
- Note that we have relaxed the overall restriction on the grammar, by making the restriction on content models more specific (indeed, this type properly includes the RCGs).

Enhancing the Murata hierarchy by rendering it more precisely

This little relaxation however causes a vast increase in processing complexity:

- In order to give its unique interpretation to any node a in any context, in the worst case one needs to know the interpretation of its mother, the interpretation of its siblings, and the interpretation of its subtrees
- Even then, GRCGs might still be ambiguous, allowing more than one interpretation for a entire single tree
- Neither a bottom-up nor a top-down parser is capable of assigning a unique interpretation locally, and maybe not even globally

Enhancing the Murata hierarchy by rendering it more precisely

We therefore introduce a subtype of GRCG, the unambiguous restrained competition grammar (URCG)

- Our goal is to eliminate is ambiguity, it should be possible to yield the unique interpretation of a node from the interpretation of its mother and the labels (not interpretations) of its sisters
- This type should be properly included in the class of GRCG grammars, and includes properly the class of STGs as well as RCGs

Enhancing the Murata hierarchy by rendering it more precisely

We characterize the URCG in the following terms:

- We introduce an alphabet of meta-variables O, which we use in the following way:
  - We form a set of sets of all competing nonterminals from N – the competition sets
  - To every competition set, we assign a single symbol from O
  - If every nonterminal occurs in exactly one competition set, we call this an (O)-assignment
  - Then, for all content models, we check for all nonterminals, whether the content models still satisfy the GRCG condition, if we replace all other nonterminals by the symbols from O they are assigned to
    - In case the assignment is not unique, i.e., a single nonterminal belongs to more than one competition set, we have to iterate this for every possible assignment
    - If for every assignment, nonterminal and content model, the resulting grammar is a GRCG, then the original grammar is a URCG

Enhancing the Murata hierarchy by rendering it more precisely

We characterize the URCG in the following terms:

- The assignments are only introduced for this evaluation procedure
- We will call contexts which become identical through the O-assignment *similar*
- We define a URCG as a grammar where competing nonterminals must not occur in the same content model and in *similar* contexts (this obviously subsumes identical contexts)
- The result is that competing nonterminals must not occur within the same contexts of labels (as opposed to nonterminals)

Enhancing the Murata hierarchy by rendering it more precisely

Why an URCG?

- There is no global ambiguity
- It is the strongest of the non-ambiguous grammar types we have considered
  → Note, that in order to provide the unique interpretation of a node, we still might have to check all *labels* of its sister nodes
- URCGs properly include RCGs, as both left and right context can count as distinctive
- Actually every regular tree language (RTL) can be generated by a GRCG, but there are languages for which there are no unambiguous grammars, and, obviously, URCGs are always unambiguous
- The search problem for URCGs is still linear, since we only need to go down the path from the root to a given node, and in addition check finitely many sister labels

Conclusion

Some results:

- Although Relax NG as realization of Regular Tree Grammars may be the most expressive XML schema language grammars in the wild tend not to use the whole RTG expressivity but usually URCGs are used – especially if you present your schema in different schema languages (e.g. TEI, DocBook, HTML)
- URCGs have only a slight loss regarding the expressivity compared to RTG but does not allow global ambiguity and have linear search times – and are more expressive than RCG
- These fundings could be used in creating efficient but fast parsers that could cope with 99.9% (roughly estimated) of schemas in the wild

Conclusion

Or to use a special money quote:

UCRGs

"~~640k~~ should be enough for everyone."

Outlook

Possible future research in this field

- Write a parser for URCGs
- Further extend the hierarchy of XML schema languages by adding XSD 1.1 and DSD2

Last but not least...

Thank your for your attention!

{maik.stuehrenberg|cwurm}@uni-bielefeld.de