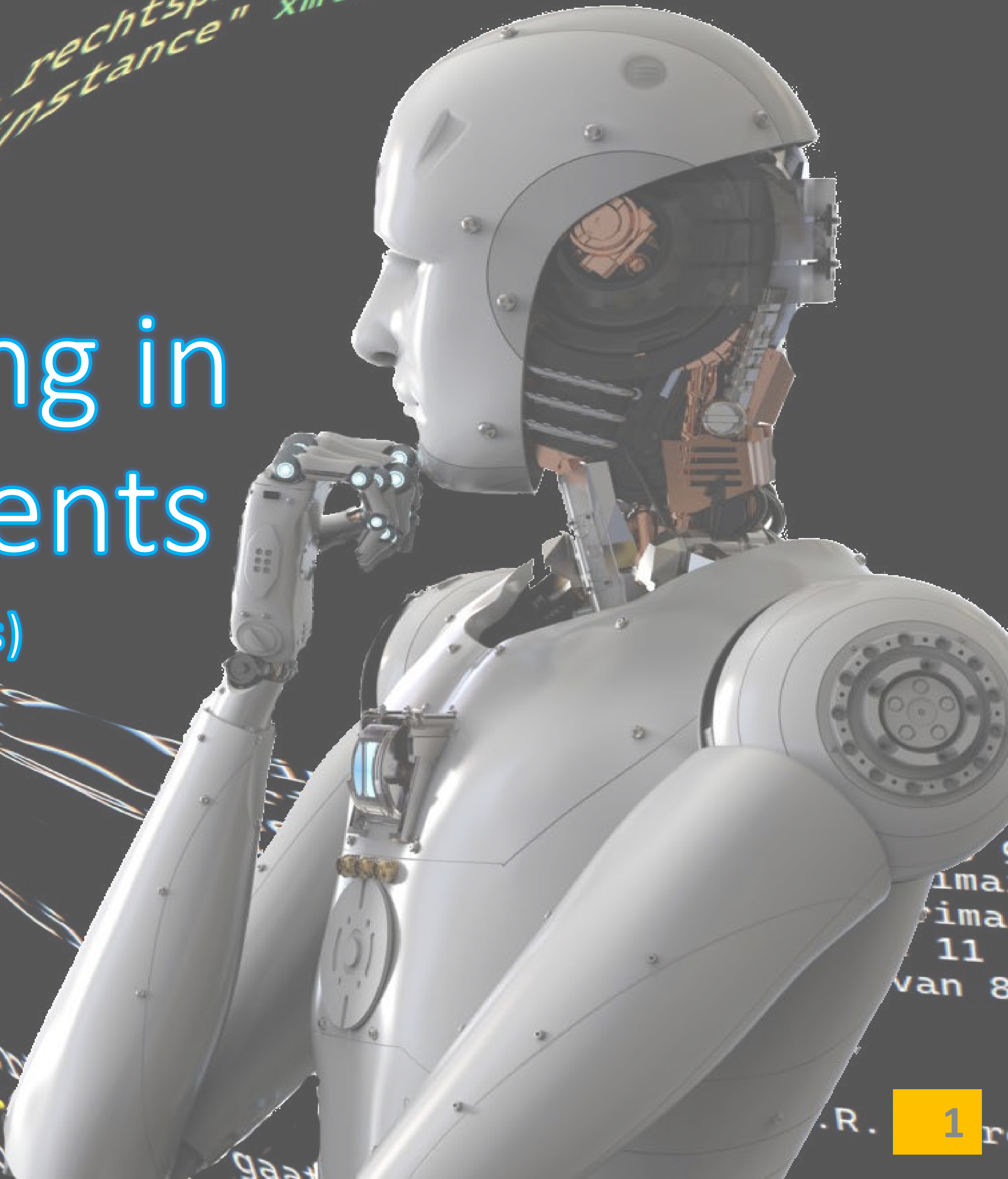
Plain text processing in structured documents

Nico Verwer (Rakensi, Netherlands)



Introduction

Link eXtractor (Dutch center for governmental publications, KOOP)

- Find case law citations and other references, and add markup with links.

- [...] the opinion of the Advocate General for the judgment of the European Court of Justice of 22 April 1997 (case C-180/95).

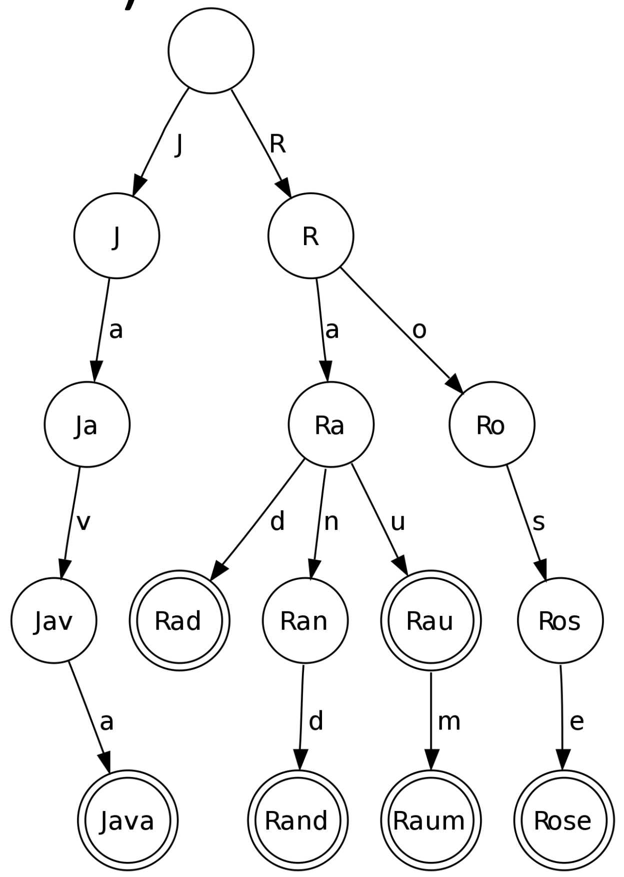
- [...] the <link ref="ECLI:EU:C:1997:11">opinion of the Advocate General</link> for the <link ref="ECLI:EU:C:1997:208">judgment of the European Court of Justice of 22 April 1997 (case C-180/95)</link>.

Named Entity Recognition

- Efficient recognition of text fragments (“named entities”)
  - Titles and abbreviations of laws (~ 250k)
  - ECHR applicants and cases (~ 100k)
  - Case law aliases (~ 3k)

- Uses a *trie* to match text efficiently

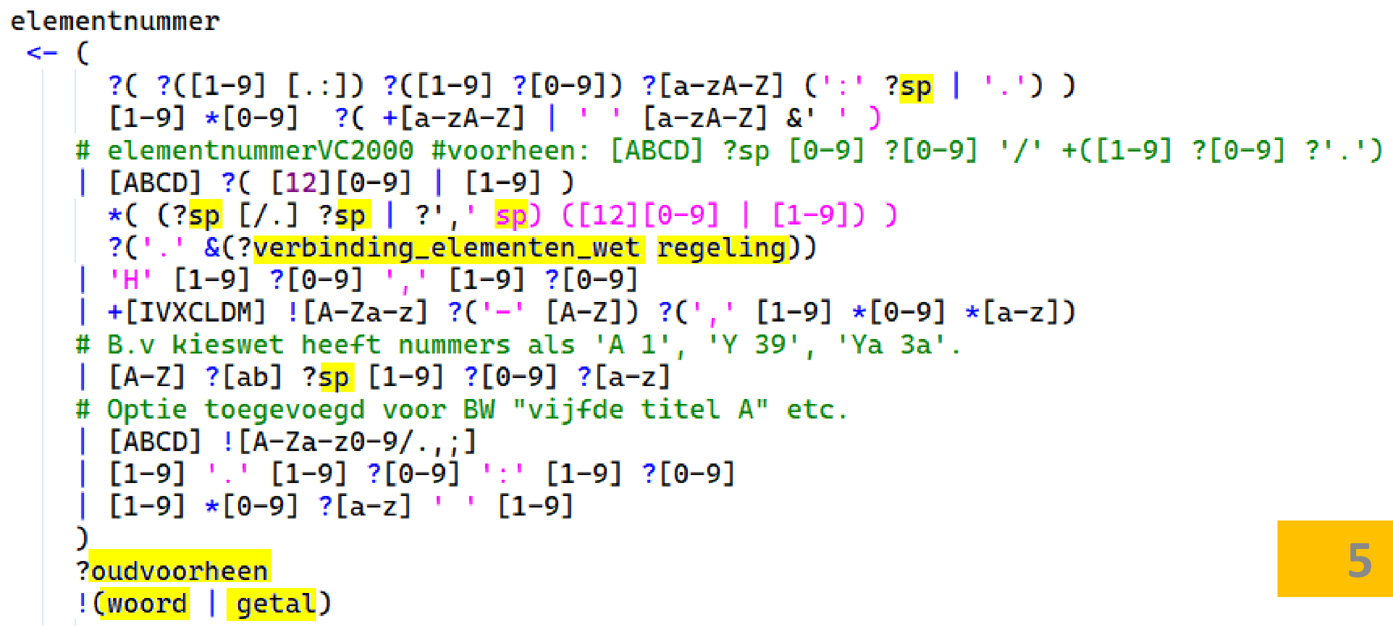
- Scans plain text (no XML)
  and marks named entities (XML)



Parsing Expression Grammars

- PEG is like regular expressions, with named sub-expressions
- PEG is like context-free grammars, efficiently solving ambiguities without back-tracking

- Parsing plain text (no XML) results in a parse tree (XML)



```
elementnummer
 <- (
      ?( ?([1-9] [.:]) ?([1-9] ?[0-9]) ?[a-zA-Z] (':' ?sp | '.') )
      [1-9] *[0-9]  ?( +[a-zA-Z] | ' ' [a-zA-Z] &' ' )
    # elementnummerVC2000 #voorheen: [ABCD] ?sp [0-9] ?[0-9] '/' +([1-9] ?[0-9] ?'.')
    | [ABCD] ?( [12][0-9] | [1-9] )
      *( (?sp [/.] ?sp | ?',' sp) ([12][0-9] | [1-9]) )
      ?('.' &(?verbinding_elementen_wet regeling))
    | 'H' [1-9] ?[0-9] ',' [1-9] ?[0-9]
    | +[IVXCLDM] ![A-Za-z] ?('-' [A-Z]) ?(',' [1-9] *[0-9] *[a-z])
    # B.v kieswet heeft nummers als 'A 1', 'Y 39', 'Ya 3a'.
    | [A-Z] ?[ab] ?sp [1-9] ?[0-9] ?[a-z]
    # Optie toegevoegd voor BW "vijfde titel A" etc.
    | [ABCD] ![A-Za-z0-9/.,;]
    | [1-9] '.' [1-9] ?[0-9] ':' [1-9] ?[0-9]
    | [1-9] *[0-9] ?[a-z] ' ' [1-9]
    )
    ?oudvoorheen
    !(woord | getal)
```

Adding structure to a structured document

- **expressed by the Supreme Court in its judgment of 16 February 2010, published in NS 2010, 98, also in NJ 2010, 232 with annotation of M.J. Borgers, and RvdW 2007, 420 (case R06/090)**

- Preserve structure of the input (XML) document, or a partially processed document

<link ecli="ECLI:NL:HR:2010:BK6357">

- expressed by <em>the <lx:INSTANTIE norm="HR">Supreme Court</lx:INSTANTIE></em> in its judgment of <date iso-8601-date="2010-02-16">16 February 2010</date>, published in NS 2010, 98, also in NJ 2010, 232 with annotation of M.J. Borgers, and RvdW 2007, 420 (case R06/090)

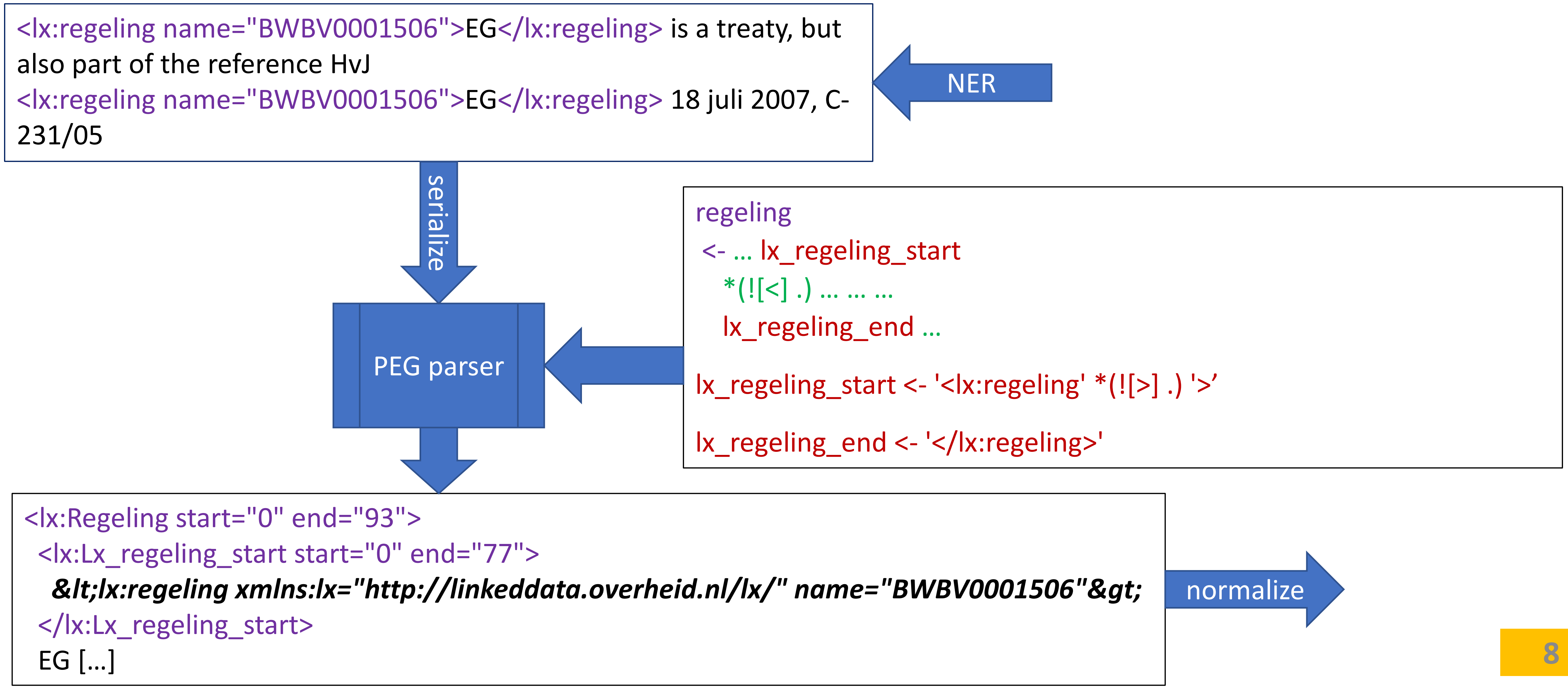
Implementing the LX

- A Java / C# / … program?

- One or more XSLTs?
  - NER & PEG parsing with extension functions
  - How to get just the text for parsing, and keep the structure

- A mix of 70 XSLT and Java components in a pipeline!
  - NER and PEG parser must recognize (or ignore) embedded XML structure
  - Still a lot of accidental complexity

Example of accidental complexity



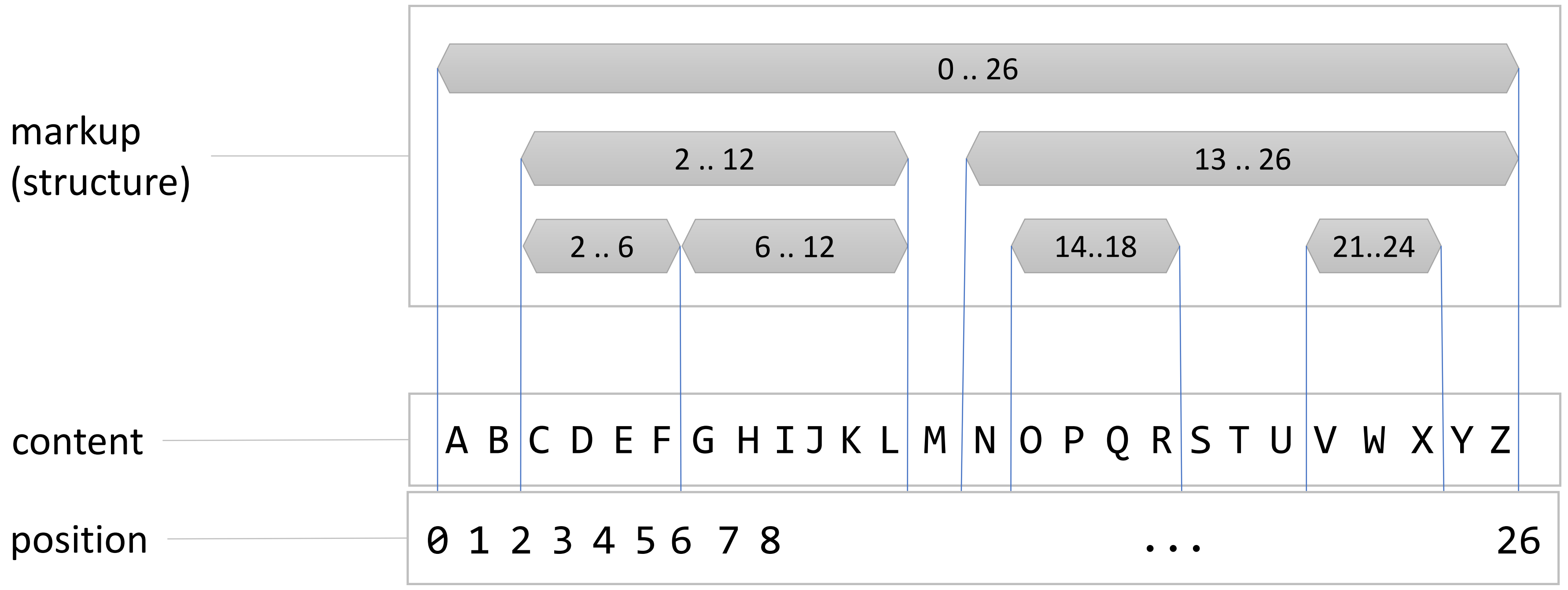
```
<lx:regeling name="BWBV0001506">EG</lx:regeling> is a treaty, but
also part of the reference HvJ
<lx:regeling name="BWBV0001506">EG</lx:regeling> 18 juli 2007, C-
231/05
```

```
regeling
 <- … lx_regeling_start
   *(![<] .) … … …
   lx_regeling_end …

lx_regeling_start <- '<lx:regeling' *(![>] .) '>’

lx_regeling_end <- '</lx:regeling>'
```

```
<lx:Regeling start="0" end="93">
  <lx:Lx_regeling_start start="0" end="77">
   &lt;lx:regeling xmlns:lx="http://linkeddata.overheid.nl/lx/" name="BWBV0001506"&gt;
  </lx:Lx_regeling_start>
  EG […]
```

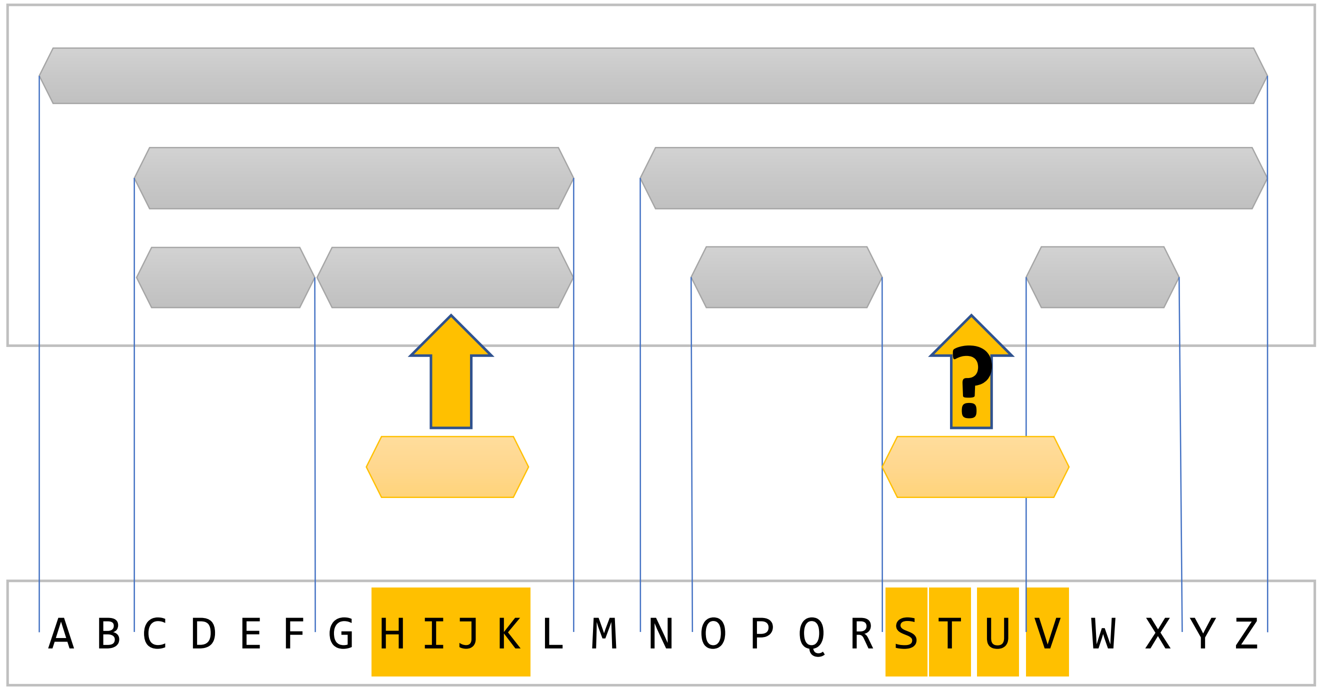
SMAX: Separated Markup API for XML

SMAX representation of XML

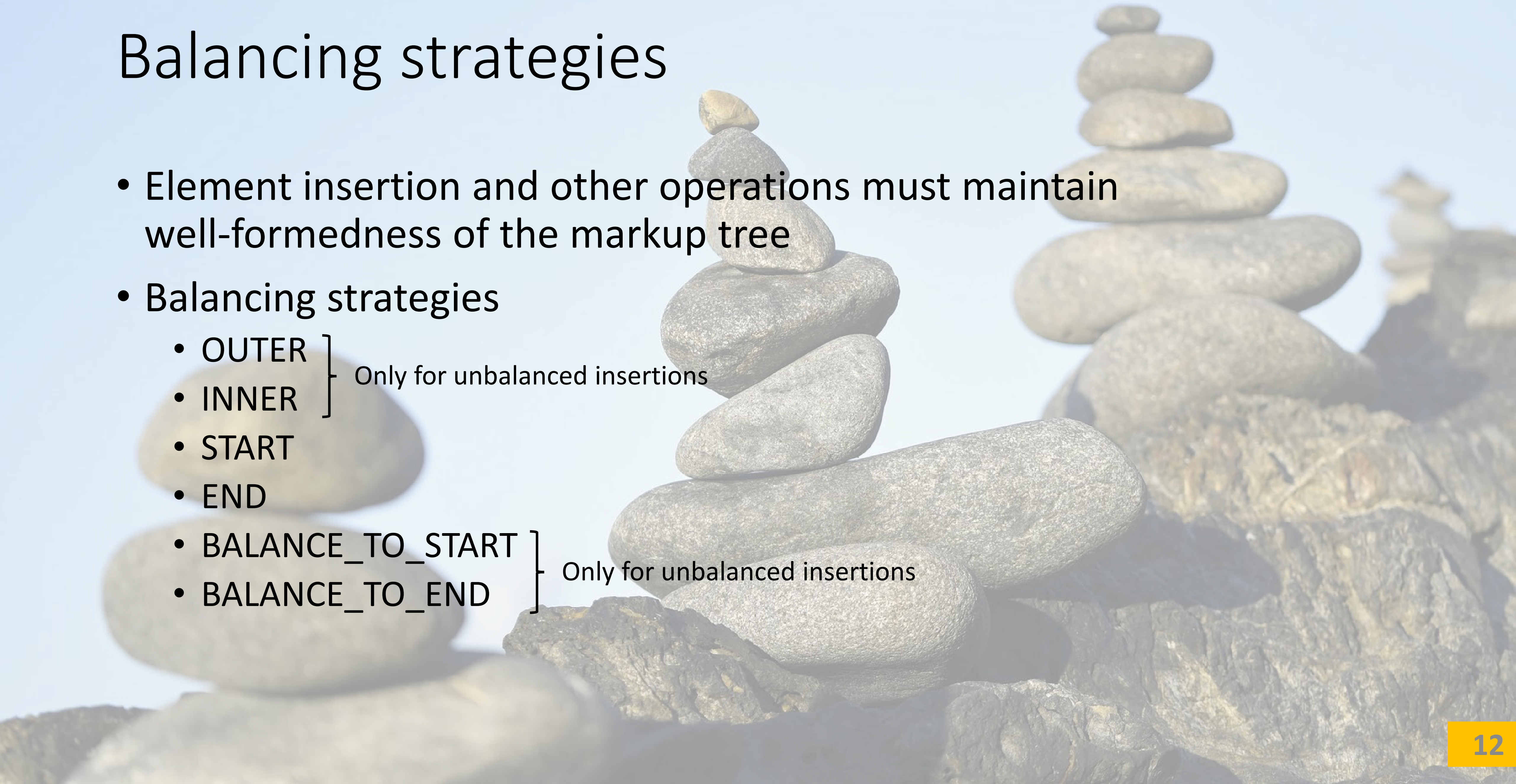


SMAX element insertion

insertMarkup(SmaxNode)



Balancing strategies

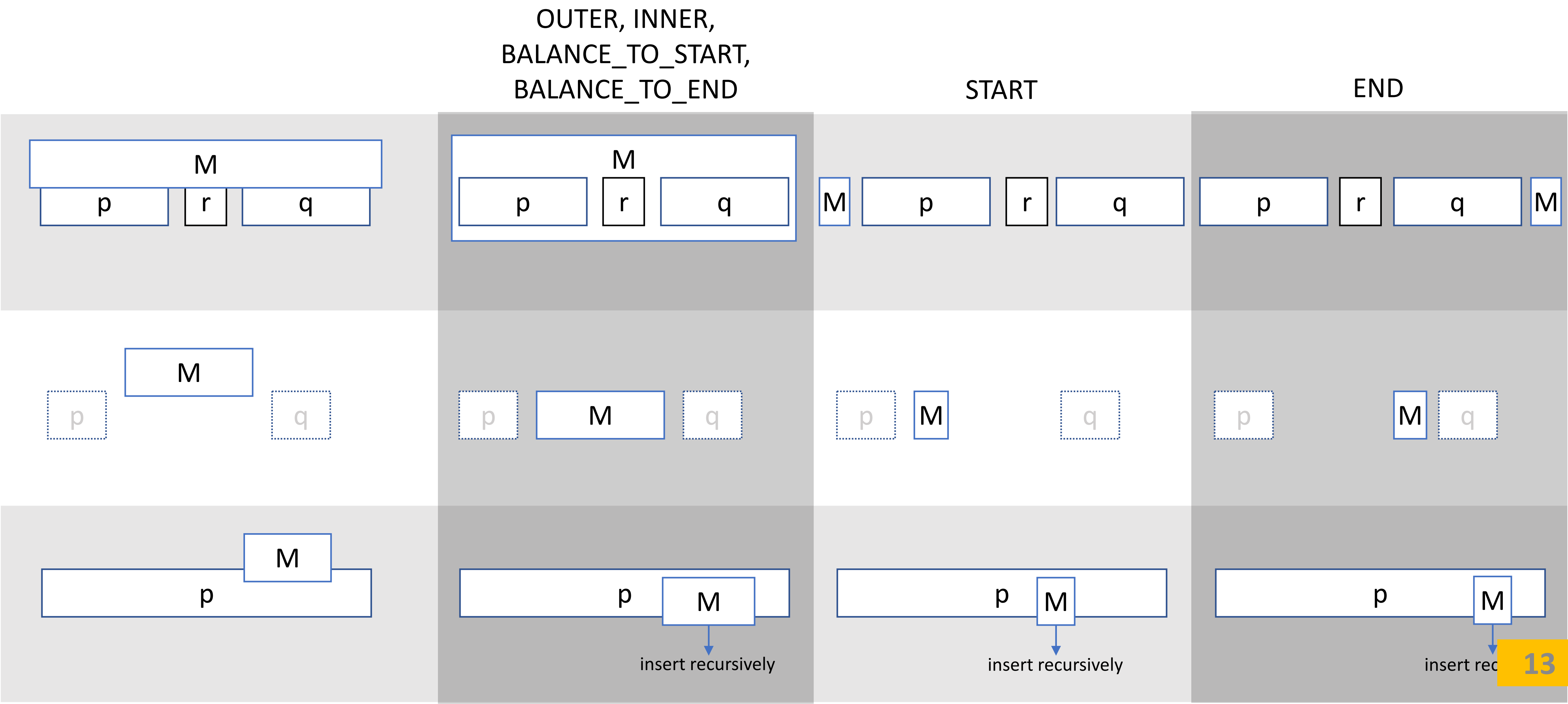


- Element insertion and other operations must maintain well-formedness of the markup tree
- Balancing strategies
  - OUTER
  - INNER

    Only for unbalanced insertions
  - START
  - END
  - BALANCE_TO_START
  - BALANCE_TO_END

    Only for unbalanced insertions

Well-formed (balanced) insertion



Well-formed insertions

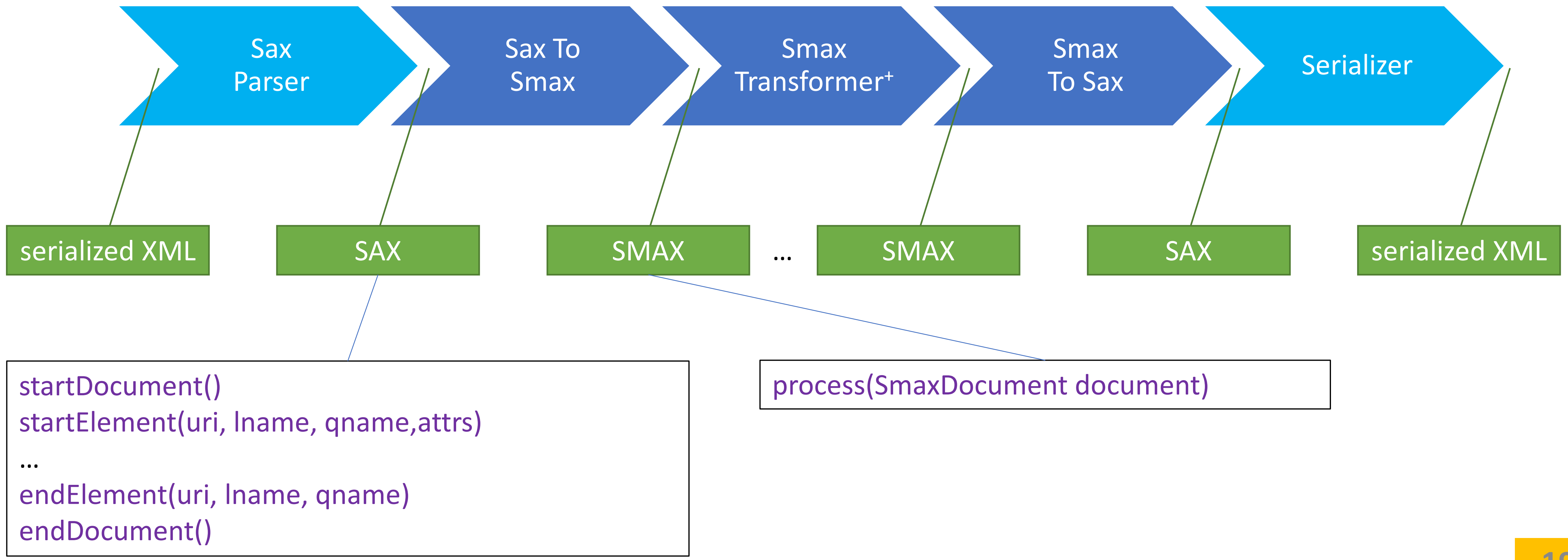
	OUTER, INNER	START	END
<p>..**!!!**..</p>	<p>..<M>!!!</M>..</p>	<p>..<M/>!!!..</p>	<p>..!!!<M/>..</p>
..**!<p>!</p><q>!</q>!**..	..<M>!<p>!</p><q>!</q>!</M>..	..<M/>!<p>!</p><q>!</q>!..	..!<p>!</p><q>!</q>!<M/>..
<p>..</p>..**!!!**..<q>..</q>	<p>..</p>..<M>!!!</M>..<q>..</q>	<p>..</p>..<M/>!!!..<q>..</q>	<p>..</p>..!!!<M/>..<q>..</q>

Non-well-formed insertions

	OUTER	INNER	START, BALANCE_TO_START
<p>.**!</p>!<q>!**.</q>	<M><p>.!</p>!<q>!.</q></M>	<p>.!</p><M>!</M><q>!.</q>	<p>.<M/>!</p>!<q>!.</q>
<p>.**!</p>!!**<q>.</q>	<M><p>.!</p>!!</M><q>.</q>	<p>.!</p><M>!!</M><q>.</q>	<p>.<M/>!</p>!!<q>.</q>
<p>.</p**>!!<q>!**.</q>	<p>.</p><M>!!<q>!.</q></M>	<p>.</p><M>!!</M><q>!.</q>	<p>.</p><M/>!!<q>!.</q>

SPEAT: Simple Pipelines of Event API Transformers

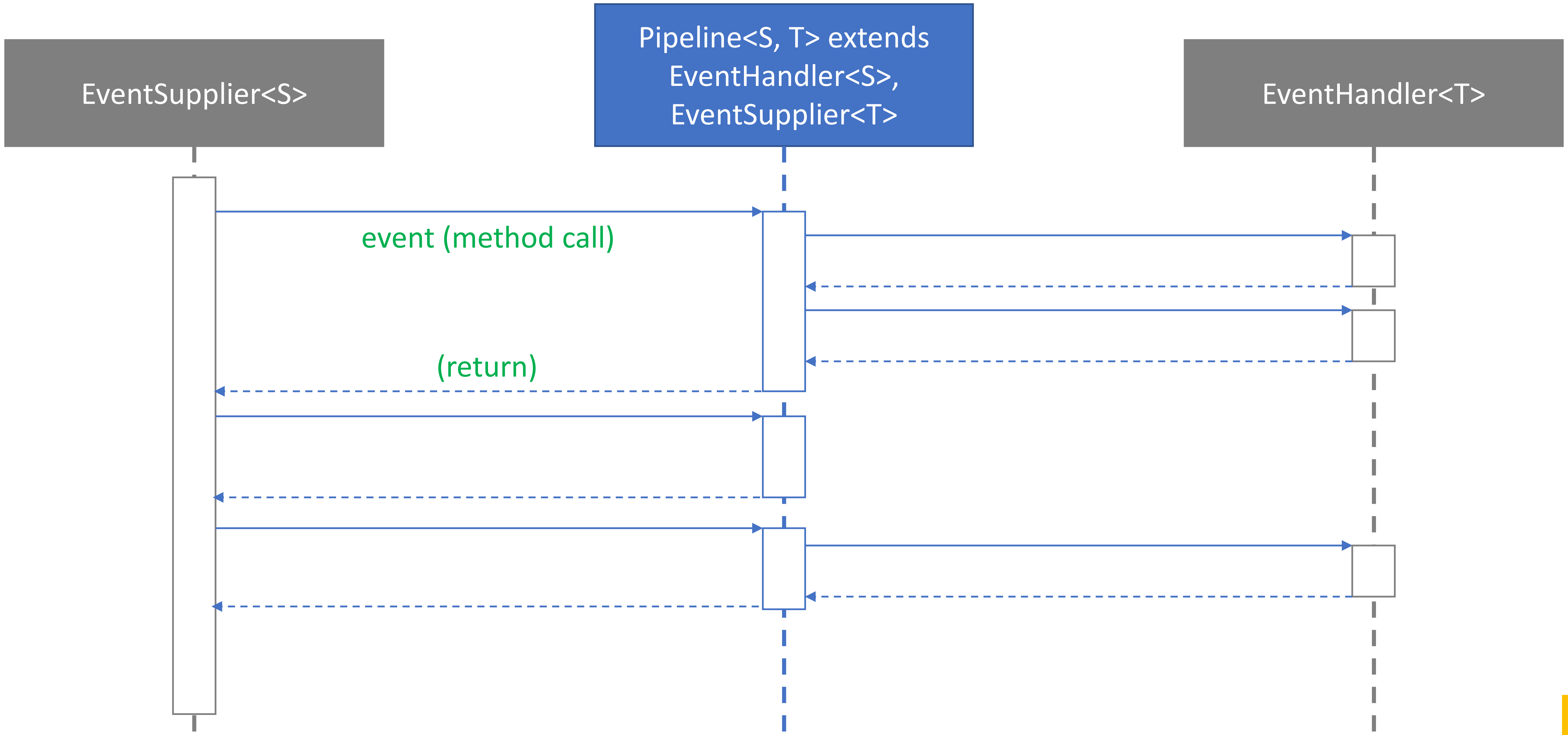
Pipelines of event API transformers



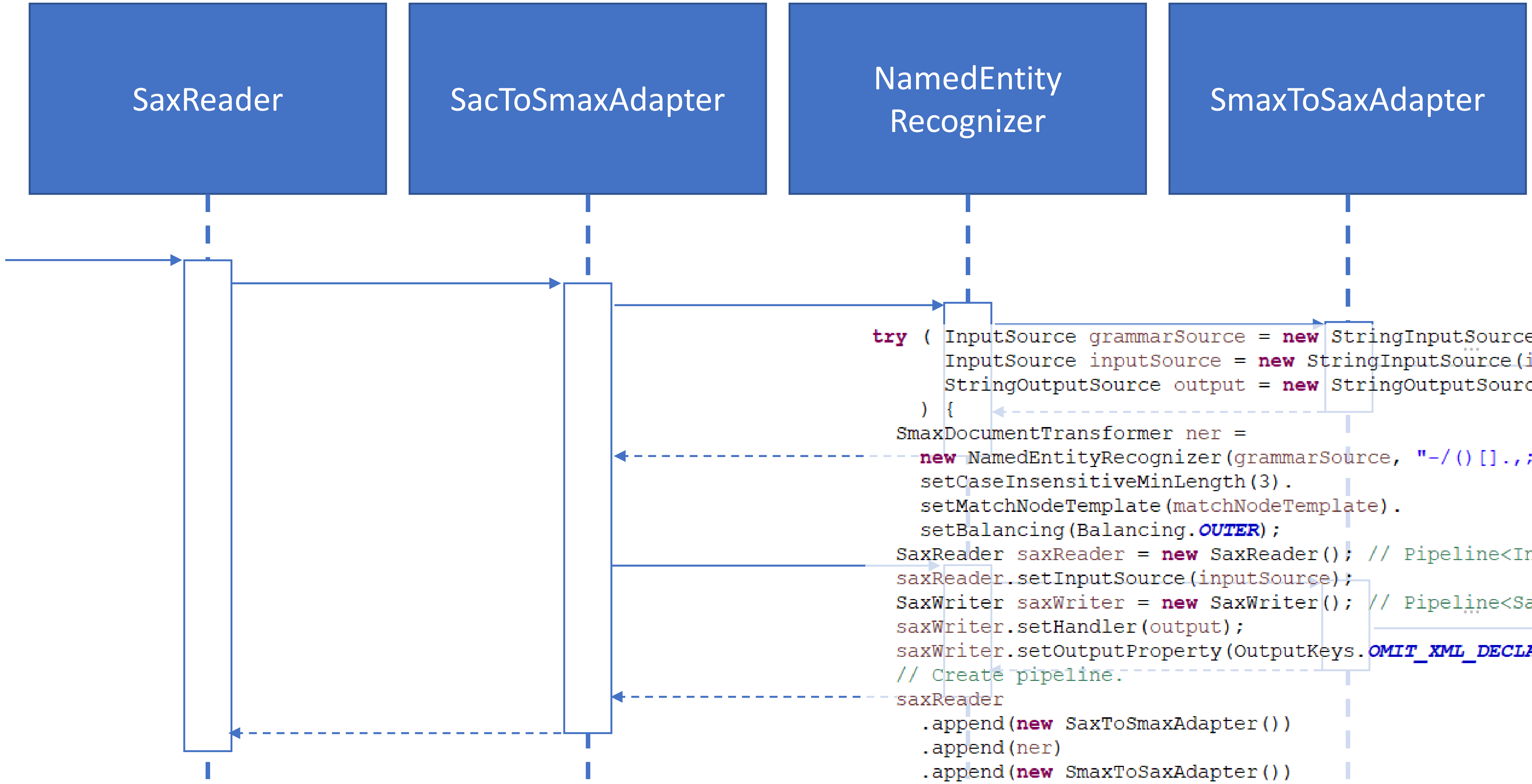
```
startDocument()
startElement(uri, lname, qname,attrs)
…
endElement(uri, lname, qname)
endDocument()
```

```
process(SmaxDocument document)
```

Event API transformers



Pipeline



```
try ( InputSource grammarSource = new StringInputSource(grammar);
      InputSource inputSource = new StringInputSource(input);
      StringOutputSource output = new StringOutputSource();
    ) {
  SmaxDocumentTransformer ner =
    new NamedEntityRecognizer(grammarSource, "-/()[].,;:'\"", null, null).
    setCaseInsensitiveMinLength(3).
    setMatchNodeTemplate(matchNodeTemplate).
    setBalancing(Balancing.OUTER);
  SaxReader saxReader = new SaxReader(); // Pipeline<InputSource, Sax>
  saxReader.setInputSource(inputSource);
  SaxWriter saxWriter = new SaxWriter(); // Pipeline<Sax, OutputSource>
  saxWriter.setHandler(output);
  saxWriter.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");
  // Create pipeline.
  saxReader
    .append(new SaxToSmaxAdapter())
    .append(ner)
    .append(new SmaxToSaxAdapter())
    .append(saxWriter);
```

https://github.com/nverwer/SPEAT

- Code is available as open source
- Some pipeline components are available
- Adapter for Apache Cocoon has been made
- Adapter for an Xproc 3 implementation would be great
- Not a framework, but a library