SXPath - Extending XPath towards Spatial Querying on Web Documents

Ermelinda Oro$^1$  Massimo Ruffolo$^1$  Steffen Staab$^2$

$^1$Institute of High Performance Computing and Networking of CNR (ICAR-CNR)
University of Calabria, Italy

$^2$Institute for Computer Science, University of Koblenz, Koblenz, Germany

VLDB 2011
Outline

1. Introduction
   - Motivations
   - State of the Art
   - SXPath Language

2. SXPath
   - Spatial Data Model
   - Syntax and Semantics
   - Complexity Issues
   - Implementation Issues and Experiments

3. Conclusions and Future Work
Outline

1. Introduction
   - Motivations
   - State of the Art
   - SXPath Language

2. SXPath
   - Spatial Data Model
   - Syntax and Semantics
   - Complexity Issues
   - Implementation Issues and Experiments

3. Conclusions and Future Work
Motivations

- Users need to access the Web and capture information in many application fields (e.g. business, competitive and military intelligence; content, document and knowledge management)

- Web pages are human oriented. The spatial arrangement of content items in Web pages produces visual cues that help human readers to make sense of document contents

- Well founded and known query formalisms, such as XPath and XQuery, do not consider spatial arrangements in querying Web pages
Presentation-Oriented Documents

HTML DOM allows only site-centric extraction

A Web Page

Spatial arrangements are rarely explicit and frequently hidden in complex nestings of layout elements corresponding to intricate tree structures that are conceptually difficult to query.
Outline

1. Introduction
   - Motivations
   - State of the Art
   - SXPath Language

2. SXPath
   - Spatial Data Model
   - Syntax and Semantics
   - Complexity Issues
   - Implementation Issues and Experiments

3. Conclusions and Future Work
State of the Art

- Web Query language
  - *XPath* 1.0 and *XQuery* 1.0 represent well founded and known web query languages having very intuitive navigational features, but the intricate DOM structure makes difficult to pose queries

- Visual languages
  - *Spatial Graph Grammars* [Kong et al.] are quite complex in term of both usability and efficiency
  - *Algebras* for creating and querying multimedia interactive presentations (e.g. ppt) [Subrahmanian et al.] require database for multimedia presentation should be created for the whole Web

- Web wrapper induction exploiting visual interface [Gottlob et al.] [Sahuguet et al.]
  - generate XPath location paths of DOM nodes
  - can benefit from using Spatial XPath
Outline

1. Introduction
   - Motivations
   - State of the Art
   - SXPath Language

2. SXPath
   - Spatial Data Model
   - Syntax and Semantics
   - Complexity Issues
   - Implementation Issues and Experiments

3. Conclusions and Future Work
As extension of XPath 1.0, *Spatial XPath* (SXPath):  
- adopts the intuitive path notation: `/axis::nodetest [pred1]`*  
- adds new *spatial axes* and new *spatial position functions*  
- has a natural semantics that enables spatial querying  
- maintains polynomial time combined complexity

**Advantages:**
- it is easy to learn and easier to use than pure XPath on Web pages  
- it is more tolerant to modifications of the internal structure of Web pages  
- it enables users to spatial query Web documents on the base of what they see on the document  
- it is capable to provide benefits to some current Web contents manipulation and wrapper learning approaches
Presentation-Oriented Documents

A Web Page from the lastfm Web site (http://www.lastfm.it/)
Acquiring a music band profile: *A music band photo that has at east its descriptive information*
Example 1

Exploiting XPath

for $li in document("last-fm.htm")
(1.1) //div[@id='content']//ul/li
return
  <music-band>
  (1.2) <name>
    {$ li / a / strong / text()}
  </name>
  ...
</music-band>

Exploiting SXPath

for $li in document("last-fm.htm")
(2.1) / CD::img[N|S::img]
return
  <music-band>
  (2.2) <name>
    {$img/ E::text[N,1]}
  </name>
  ...
</music-band>
Example 2

Acquiring friend lists from different social networks pages represented as couples `<photo, name>`.

Friend lists from different social networks pages (a) Bebo (b) Care (c) Netlog.

```xml
for $img in document("http://www.bebo.com/friendlist.html")
    (3.1) //img[N|S|E|W::img]
        return
        <friend>
            (3.2) <photo> {$img} </photo>
            (3.3) <name> {$img/S :: text() [N,1]} </name>
        </friend>
```
Example 2

- A single data record can be split in different sub-trees
- Wrapper induction techniques like DEPTA [Zhai et al.] recognize data records when they are encoded in the DOM as consecutive similar subtrees

```
for $img in document("http://www.bebo.com/friendlist.html")
(3.1) //img[ N|S|E|W::img ]
    return
    <friend>
(3.2) <photo> {$img} </photo>
(3.3) <name> { $img/ S :: text() [N,1] } </name>
    </friend>
```
Outline

1. Introduction
   - Motivations
   - State of the Art
   - SXPath Language

2. SXPath
   - Spatial Data Model
   - Syntax and Semantics
   - Complexity Issues
   - Implementation Issues and Experiments

3. Conclusions and Future Work
Spatial Data Model

- The Document Object Model (DOM) is the internal representation of markup languages (XML, HTML)
- The tree-based structures of XML are often not convenient and not expressive enough in order to represent spatial arrangements
- The spatial arrangements are rarely explicit and frequently hidden into intricate tree structures that are conceptually difficult to query
The **Rectangular Algebra** (RA) [Balbiani et al.] extends Allen’s temporal interval algebra (IA) to the 2-dimensional case.

RA is a very fine-grained and expressive model that allows the computations of spatial relations as well as algebraic optimizations.

RA holds many important properties (e.g. invertibility) that allows for optimized query evaluation.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Symbol</th>
<th>Meaning</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>b</td>
<td>$s_1$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>meets</td>
<td>m</td>
<td>$s_1$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>overlaps</td>
<td>o</td>
<td>$s_1$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>starts</td>
<td>s</td>
<td>$s_1$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>during</td>
<td>d</td>
<td>$s_1$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>finish</td>
<td>f</td>
<td>$s_1$</td>
<td>$s_1$</td>
</tr>
<tr>
<td>equals</td>
<td>e</td>
<td>$s_1$</td>
<td>$s_1$</td>
</tr>
</tbody>
</table>

Oro, Ruffolo, Staab
The SDOM extends the Document Object Model (DOM) by:

- RA relations existing between pairs of nodes visualized on screen
- Spatial orders among nodes

\[ n_1 \preceq n_2 = n_4 = n_6 \preceq n_3 = n_5 \]
The Spatial DOM (SDOM)

Definition

SDOM is a node labeled sibling ordered tree enriched by RA relations

\[ SDOM = \langle V, R_{\downarrow}, R_{\Rightarrow}, A, f_s \rangle \]

where:
- \( V \) is the set of labeled DOM nodes. \( V = V_v \cup V_{nv} \)
- \( R_{\downarrow} \) is the firstchild relation
- \( R_{\Rightarrow} \) is the nextsibling relation
- \( A \subseteq V_v \times V_v \)
- Let \( R_{rec} \) be the set of RA relations \( f_s : A \rightarrow R_{rec} \)
Qualitative Spatial Models

Rectangular cardinal relations

Topological relations, inspired by the Region Connection Calculus model:

- contained (CD)
- container (CR)
- equivalent (EQ)

Example

- $r \text{ NE} r_1$
- $r \text{ B} r_2$

Example

- $r \text{ CD} r_2$
- $r_2 \text{ CR} r$
Spatial Navigation Axes

- As in XPath, SXPath primitives for navigating the SDOM are called axes.
- Axes are interpreted binary relations $\chi \subseteq V \times V$. Let $self := \{\langle u, u\rangle | u \in V\}$ be the reflexive axis, remaining SXPath axes are partitioned in two sets: $\Delta_t$ and $\Delta_s$.
  - $\Delta_t = \{self, child, parent, descendant, descendant-or-self, ancestor, ancestor-or-self, following-sibling, preceding-sibling, following, preceding\}$ contains traditional XPath 1.0 axes.
  - $\Delta_s$ is the set of novel spatial axes expressed by: basic and disjunctive RCRs and topological relations that are more intuitive than RA relations.
SXPath spatial axes are interpreted binary relations $\chi_s \subseteq V_v \times V_v$ of the following form

$$
\chi_s = \{ \langle u, w \rangle | u, w \in V_v \land u \rho w \land \rho \in \mu(R) \}.
$$

Where $R$ is the RC or Topological Relation that names the spatial axis and $\mu$ is the mapping function.
Introduction

1. Introduction
   - Motivations
   - State of the Art
   - SXPath Language

2. SXPath
   - Spatial Data Model
   - Syntax and Semantics
   - Complexity Issues
   - Implementation Issues and Experiments

3. Conclusions and Future Work
SXPath expressions have the same structure as the ones in XPath:

- Location paths are sequences of location steps separated by the navigation operator "/".
- A locstep is \texttt{axis :: nodetest [pred}_1\ldots [pred}_n\]

We enrich XPath 1.0 by:

- The new set of \textit{spatial axes}
- \textit{Spatial position functions}

Specific subsets of the language with attractive properties have been characterized for XPath 1.0 [4, 6]:

- \textit{Core XPath} \Rightarrow \textit{Core SXPath}
- \textit{Wadler Fragment(WF)} \Rightarrow \textit{Spatial WF}
The main structural feature of SXPath are *expressions*, that return a value from one of the following four types: *node set*, *number*, *string*, or *Boolean*.

Every expression evaluates relative to a *context*, concept introduced by Wadler.

**Definition (Context)**

The *context* is the following 12-tuple:

\[
\tilde{c} = (n, p_{< \text{doc}}, s_{< \text{doc}}, p_{\uparrow}, s_{\uparrow}, p_{\rightarrow}, s_{\rightarrow}, p_{\downarrow}, s_{\downarrow}, p_{\leftarrow}, s_{\leftarrow}, p_{t})
\]

where:
- \( n \) is a *context node*
- \( p_{<z} \) are the *context positions* w.r.t. orders
- \( s_{<z} \) are the *context sizes*
Semantics

Definition (Location path semantics)

Let $\pi, \pi_1, \pi_2$ be location paths, let $locstep$ be a location step over an axis $\chi$, let $bexpr$ be a boolean expression and let $n$ be a context node, $P: LocationPath \rightarrow node \rightarrow nodeset$ is defined as follows:

\[
P[\pi](n) := P[root]
\]
\[
P[\pi_1/\pi_2](n) := \{ n_2 | n_1 \in P[\pi_1](n) \land n_2 \in P[\pi_2](n_1) \}
\]
\[
P[\pi_1|\pi_2](n) := P[\pi_1](n) \cup P[\pi_2](n)
\]
\[
P[axis::t](n) := \{ n' | [axis](n, n') \} \cap T(t)
\]
\[
P[locstep[bexpr]](n) := \{ n' | \bar{W} = P[locstep](n) \land n' \in \bar{W} \land \epsilon[bexpr](\bar{c}_n') = true \land
\]
\[
\bar{c}_n' := \langle n', idx\chi(n', \bar{W}), |\bar{W}|, pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}), pidx_{\leq}(n', \bar{W}), plast_{\leq}(\bar{W}) \}
\]

The semantics of spatial axis is given in terms of spatial relations among nodes

\[
[spatialAxis] := \{ (n, n') | mbr(n) \rho mbr(n') \land \rho = \mu(spatialAxis) \}
\]
Definition (Semantics of SXPath)

\[ \varepsilon : SXPathExpression \rightarrow C \rightarrow SXPathType \]

\[ \varepsilon[\pi](\vec{c}) := P[\pi](n) \]
\[ \varepsilon[position()](\vec{c}) := p_{\text{doc}} \]
\[ \varepsilon[posFromN()](\vec{c}) := p_{\downarrow} \]
\[ \varepsilon[posFromS()](\vec{c}) := p_{\uparrow} \]
\[ \varepsilon[posFromW()](\vec{c}) := p_{\rightarrow} \]
\[ \varepsilon[posFromE()](\vec{c}) := p_{\leftarrow} \]
\[ \varepsilon[posSpatialNesting()](\vec{c}) := p_t \]
\[ \varepsilon[\text{Op}(e_1, \ldots, e_m)](\vec{c}) := F[\text{Op}](\varepsilon[e_1](\vec{c}), \ldots, \varepsilon[e_m](\vec{c})) \]

\[ F[\text{RelOp}: \text{num} \times \text{num} \rightarrow \text{bool}](i_1, i_2) := i_1 \ \text{RelOp} \ i_2 \]
\[ F[\text{constant number } i: \rightarrow \text{num}](\) := i \]
\[ \ldots \]
Introduction

Motivations
State of the Art
SXPath Language

SXPath

Spatial Data Model
Syntax and Semantics
Complexity Issues
Implementation Issues and Experiments

Conclusions and Future Work
Core SXPath Complexity

Theorem (Core SXPath Combined Complexity)

Core SXPath queries can be evaluated in time $O(|D|^2 \times |Q|)$ where $|D|$ is the size of the XML document, and $|Q|$ is the size of the query $Q$

- **Proof Sketch** There are $O(|V_v|^2)$ many spatial relations to be considered in addition to the $O(|V|)$ many relations of the DOM incurring a higher polynomial worst case complexity
In order to obtain a polynomial-time combined complexity bound for SXPath query evaluation we use dynamic programming adopting the Context-Value Table (CV-Table) principle introduced by Gottlob et al.

Position and size are computed on demand, we compute all spatial position functions in a loop for all pairs previous/current nodes

Full SXPath computational costs are dominated by String Operations belonging to XPath 1.0

In SWF the computation of spatial ordering generates a higher polynomial worst-case than XPath 1.0
Comparison between complexity bound of SXPath and XPath 1.0 for a XML document $D$ and a query $Q$.

<table>
<thead>
<tr>
<th></th>
<th>XPath 1.0</th>
<th>SXPath</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Space Time Core</td>
<td>Spatial Core</td>
</tr>
<tr>
<td></td>
<td>$O(</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>$O(</td>
<td>D</td>
</tr>
<tr>
<td>Space Time EWF</td>
<td>$O(</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>$O(</td>
<td>D</td>
</tr>
<tr>
<td>Space Time Full Xpath 1.0</td>
<td>$O(</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>$O(</td>
<td>D</td>
</tr>
<tr>
<td></td>
<td>$O(</td>
<td>D</td>
</tr>
</tbody>
</table>
Outline

1. Introduction
   - Motivations
   - State of the Art
   - SXPath Language

2. SXPath
   - Spatial Data Model
   - Syntax and Semantics
   - Complexity Issues
   - Implementation Issues and Experiments

3. Conclusions and Future Work
The SXPath System

- Query
- Query Parser
- Query Parse Tree
- HTML Document
- HTML Parser
- DOM
- Layout Engine
- SDOM Builder
- SDOM
- Query Evaluator
- Answer

Introduction
SXPath
Conclusions and Future Work
Spatial Data Model
Syntax and Semantics
Complexity
Implementation Issues and Experiments

Oro, Ruffolo, Staab
SXPath
The SXPath System

GUI that supports Spatial Querying
The curves grows linear on log-log scale indicating the polynomial growth.
We have defined the user task “identify product data records and extract product names and prices” from the Web site www.bol.de

We have asked users to learn the SXPath language and complete the task by writing a sound and complete SXPath query looking only at the visualized Web page

We have asked users to answer a questionnaire based on the seven-item Likert scale: very easy/satisfactory (3) ... very difficult/unsatisfactory (-3)

<table>
<thead>
<tr>
<th>#user</th>
<th>Time (min)</th>
<th>Easiness/Difficulty</th>
<th>Satisfaction/Unsatisfaction</th>
<th>#attempts</th>
<th>name</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>75</td>
<td>2</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>65</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>125</td>
<td>-1</td>
<td>-1</td>
<td>9</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>35</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>55</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>57</td>
<td>2</td>
<td>1.2</td>
<td>4.3</td>
<td>3.6</td>
<td></td>
</tr>
<tr>
<td>σ</td>
<td>26</td>
<td>1.18</td>
<td>1.1</td>
<td>2.2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Results of Experiments
Usability Evaluation of SXPath on Deep Web Pages

We have asked users to perform the extraction task “identify product data records and extract product names and prices” for each Web site in the dataset

- only by looking at the displayed Web pages by using at the most 5 attempts
- looking at both visualized Web pages and internal page structures (i.e. DOM and SDOM), by using at the most 10 minutes
- by applying the same location path for different Web sites in the dataset having the same visual pattern. We have observed that it is possible to use the same sound and complete spatial location path for Web sites having the same visual pattern. Instead, different XPath location paths are needed

<table>
<thead>
<tr>
<th>Considering a set of Deep Web Sites</th>
<th>Querying Without DOM/SDOM</th>
<th>Querying With DOM/SDOM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SXPath</td>
<td>XPath</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>2535</td>
<td>27.3/6</td>
</tr>
<tr>
<td><strong>Recall</strong></td>
<td>100%</td>
<td>99%</td>
</tr>
<tr>
<td><strong>Precision</strong></td>
<td>99%</td>
<td>42%</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

- We have extended XPath to include spatial navigation into the query mechanism.
- The SDOM extends DOM for describing relationships between data entities.
- SXPath query language is a stepping stone for future work on extracting information from presentation-oriented documents. It could be used and extended for:
  - querying other *presentation-oriented documents* (e.g. PDF, Doc, etc.) or multimedia documents
  - recognizing and extracting ontology objects
  - automatically learning of wrappers and learning of ontology instances [Staab et al.] by exploiting spatial patterns
  - navigating and accessing Deep Web data sources and dynamic components
For Further Reading I


For Further Reading II

S. Mir, S. Staab, and I. Rojas.
Unsupervised approach for acquiring ontologies and rdf data from online life science databases.
In *ESWC*, 2010.

A. Sahuguet and F. Azavant.
Building intelligent web applications using lightweight wrappers.

P. Wadler.
Two semantics for xpath.
Example 2

Acquiring the table in the document as a set of triples of the form `<row-header, column-header, value>`.

```xml
for $rh in document("table.pdf")
  (2.1) //text [not(W::*)]
  return
  <table-triples>
    { for $ch at $j in document("table.pdf")
      (2.2) //text [not(N::*)]
      <row-header>
        (2.3) {$rh}
      </row-header>
      <column-header>
        (2.4) {$ch}
      </column-header>
      <value>
        (2.5) {$rh/E::text [W,$j]}
      </value>
    }
  </table-triples>
```
Core SXPath

Definition

The syntax of Core SXPath is defined by the following EBNF grammar

locpath ::= ‘/’ locpath | locpath ‘/’ locpath | locpath ‘|’ locpath | locstep.
locstep ::= axis ‘::’ t | locstep ‘[’ bexpr ‘]’
bexpr ::= bexpr ‘and’ bexpr | bexpr ‘or’ bexpr | ‘not (’ bexpr ‘)’ | locpath.
axis ::= xpathAxis | spatialAxis.
spatialAxis ::= topAxis | dirAxis.
topAxis ::= ‘EQ’ | ‘CD’ | ‘CR’.
dirAxis ::= ‘B’ | ‘...’ | ‘U’.
The syntax of the SWF-Queries is defined by the Core SXPath grammar with the following extensions.

\[
\begin{align*}
\text{expr} &::= \text{locpath} \mid \text{bexpr} \mid \text{nexpr} \\
\text{dirAxis} &::= \text{‘B’} \mid \ldots \mid \text{‘U’} \mid \text{disjDirAxis} \\
\text{bexpr} &::= \text{bexpr ‘and’ bexpr} \mid \text{bexpr ‘or’ bexpr} \mid \\
& \quad \text{‘not’ (‘bexpr ‘)’} \mid \text{nexpr relop nexpr} \mid \\
& \quad \text{sexpr relop sexpr} \mid \text{locpath} \mid \\
& \quad \text{locpath relop sexpr} \mid \text{locpath relop number}. \\
\text{nexpr} &::= \text{number} \mid \text{nexpr arithop nexpr} \mid \\
& \quad \text{‘position()’} \mid \text{‘last()’} \mid \text{‘posFromS()’} \mid \text{‘lastFromS()’} \mid \\
& \quad \text{‘posFromN()’} \mid \text{‘lastFromN()’} \mid \text{‘posFromW()’} \mid \text{‘lastFromW()’} \mid \\
& \quad \text{‘posFromE()’} \mid \text{‘lastFromE()’} \mid \text{‘posSpatialNesting()’} \\
\text{sexpr} &::= \text{string}. \\
\text{arithop} &::= \text{‘+’} \mid \text{‘-’} \mid \text{‘*’} \mid \text{‘div’} \mid \text{‘mod’}. \\
\text{relop} &::= \text{‘=’} \mid \text{‘!=’} \mid \text{‘<’} \mid \text{‘<=’} \mid \text{‘>’} \mid \text{‘>='}.
\end{align*}
\]
Input: A set of nodes $\Gamma$ and an axis $\chi \in \Delta$
Output: $\chi(\Gamma)$
Method: $eval_\chi(\Gamma)$

(1.1) function $eval_{self}(\Gamma) := \Gamma$.
(1.2) function $eval_{\chi_t}(\Gamma) := eval_{E(\chi_t)}(\Gamma)$.
(1.3) function $eval_{\chi_s}(\Gamma) := eval_{\{\rho_i | \rho_i \in \mu(\chi_s)\}}(\Gamma)$.
(1.4) function $eval_{\chi_s^{-1}}(\Gamma) := eval_{\{\rho_i^{-1} | \rho_i \in \mu(\chi_s)\}}(\Gamma)$.
(1.5) function $eval_{\varrho}(\Gamma)$ begin
(1.6) $\Gamma' := \emptyset$;
(1.7) foreach $u \in \Gamma \cap u \in V_v$ do
(1.8)    foreach $\rho_i \in \varrho$ do
(1.9)      $\Gamma' := \Gamma' \cup \text{set } f_{\rho_i}(u)$ od od
(1.10) return $\Gamma'$ end.
(Location step evaluation algorithm)

Input: A set of nodes $\Gamma$ and a location step $e = \chi : \tau[e_1] \ldots [e_m]$ 

Output: $P[e](\Gamma)$ 

Method: $eval(e, \Gamma)$ begin

1. $Res := \emptyset$
2. $W := \chi(\Gamma) \cap T(\tau)$;
3. for each $u \in \Gamma$ do
   4. $W' := \{ w \mid w \in W \land u \chi w \}$
5. for each $e_i$ with $1 \leq i \leq m$ (in ascending order) do
   6. $\tilde{W} := layering(W')$
   7. $W' := \{ w \mid w \in \tilde{W} \land \varepsilon[e_i](\tilde{c_w}) = true \land$
      \[ c_w := \langle w, idx_{\chi}(w, \tilde{W}), |\tilde{W}|, pidx_{\leftarrow}(w, \tilde{W}), plast_{\leftarrow}(\tilde{W}), pidx_{\rightarrow}(w, \tilde{W}), plast_{\rightarrow}(\tilde{W}), pidx_{\downarrow}(w, \tilde{W}), plast_{\downarrow}(\tilde{W}), pidx_{\uparrow}(w, \tilde{W}), plast_{\uparrow}(\tilde{W}), pidx_{\triangleright}(w, \tilde{W}), plast_{\triangleright}(\tilde{W}), pidx_{\triangleleft}(w, \tilde{W}), plast_{\triangleleft}(\tilde{W}) \rangle \}$
   8. od
   9. $Res := Res \cup W'$
8. od
9. return $Res$ end;