A formal model of pattern matching in XSL

Philip Wadler
Bell Labs
Lucent Technologies
wadler@research.bell-labs.com

January 8, 1999

1 Introduction

This note presents a formal model of the semantics of pattern matching in XSL [XSL].

The formal semantics brings to light issues that can be hard to spot in an english language description. For instance, here is how [XSL] defines match:

> The result of the *MatchExpr* is true if, for any node in the document that contains the context of the *MatchExpr*, the result of evaluating the *SelectExpr* with that node as context contains the context of the *MatchExpr*. Otherwise the result is false.

As James Clark noted [Clark], this sentence is ambiguous: does 'contains' qualify 'dcoument' or 'node in the document'? (The intention is the former.) The formal specification given here presents the same information in just one line, and avoids the ambiguity.

The semantics is formalized using sets. When a selection pattern is processed using `for-each`, the selected nodes are presented in document order.

We use the usual notation for sets, including set comprehensions. See any standard textbook for an introduction, such as [Halmos]. We write $Set(A)$ for the type of all finite sets with elements of type A, and $Set_1(A)$ for the subtype of $Set(A)$ where the sets have at most one element. We write

$$\mathit{first}, \mathit{last} : Set(Node) \to Set_1(Node)$$

for the functions that take a set and return the singleton set containing just its first or last node in document order, or return the empty set if the given set is empty.

We represent a relation on nodes by a function $r : Node \to Set(Node)$. We define the transitive closure r^+ and the reflexive and transitive closure r^* in the usual way.

$$\begin{array}{lcl} r^+(x) & = & \{\, x_2 \mid x_1 \in r(x),\ x_2 \in r^*(x_1) \,\} \\ r^*(x) & = & \{\, x \,\} \cup r^+(x) \end{array}$$

If $\mathit{subnodes} : Node \to Set(Node)$ returns the immdediate descendants of a node, then $\mathit{subnodes}^+(x)$ returns all proper descendants of node x, and $\mathit{subnodes}^*(x)$ returns all descendants of node x including x itself.

Throughout we use the following conventions for variables.

$$\begin{array}{lcl} n & : & \mathit{Name} \\ p & : & \mathit{Pattern} \\ q & : & \mathit{Qualifier} \\ s & : & \mathit{String} \\ x & : & \mathit{Node} \end{array}$$

2 Access to nodes

We take as primitive functions to determine the type of a node and to access its components, as specified in [XSL, Section 2.4].

$$
\begin{array}{lcl}
\mathit{isRoot} & : & \mathit{Node} \to \mathit{Boolean} \\
\mathit{isElement} & : & \mathit{Node} \to \mathit{Boolean} \\
\mathit{isAttribute} & : & \mathit{Node} \to \mathit{Boolean} \\
\mathit{isText} & : & \mathit{Node} \to \mathit{Boolean} \\
\mathit{isComment} & : & \mathit{Node} \to \mathit{Boolean} \\
\mathit{isPI} & : & \mathit{Node} \to \mathit{Boolean} \\
 & & \\
\mathit{parent} & : & \mathit{Node} \to \mathit{Set}_1(\mathit{Node}) \\
\mathit{children} & : & \mathit{Node} \to \mathit{Set}(\mathit{Node}) \\
\mathit{attributes} & : & \mathit{Node} \to \mathit{Set}(\mathit{Node}) \\
\mathit{name} & : & \mathit{Node} \to \mathit{Name} \\
\mathit{value} & : & \mathit{Node} \to \mathit{String}
\end{array}
$$

The function *parent* returns a singleton set containing the parent of the node, or returns the empty set if the node is a root. The function *children* returns the empty set if the node is a leaf.

Each node belongs to a document, and each document has a root node.

$$
\begin{array}{lcl}
\mathit{document} & : & \mathit{Node} \to \mathit{Document} \\
\mathit{root} & : & \mathit{Document} \to \mathit{Node}
\end{array}
$$

We represent identifiers by strings. The function *id* takes a string and a document and returns the singleton set containing the unique element within the document which has the string as its 'id' attribute, or returns the empty set if no such element exists. The function *split* takes a string into the set of space-separated identifiers within it.

$$
\begin{array}{lcl}
\mathit{id} & : & \mathit{Document} \times \mathit{String} \to \mathit{Set}_1(\mathit{Node}) \\
\mathit{split} & : & \mathit{String} \to \mathit{Set}(\mathit{String})
\end{array}
$$

The subnodes of a node are all the immediate descendants, whether children or attributes.

$$
\begin{array}{lcl}
\mathit{subnodes} & : & \mathit{Node} \to \mathit{Set}(\mathit{Node}) \\
\mathit{subnodes}(x) & = & \text{if } \mathit{isElement}(x) \text{ then } \mathit{children}(x) \cup \mathit{attributes}(x) \text{ else } \mathit{children}(x)
\end{array}
$$

What follows refers only to subnodes, not attributes or children. For all nodes x we have $x \in \mathit{subnodes}^*(\mathit{root}(\mathit{document}(x)))$. Further, for all nodes x and x_1 we have $\neg \mathit{isRoot}(x) \wedge \mathit{parent}(x) = x_1$ if and only if $x \in \mathit{subnodes}(x_1)$.

The function *siblingElements* returns all siblings of a node that are elements, and the function *siblingsOfType* returns all siblings of an element that have the same tag as the element.

$$
\begin{array}{lcl}
\mathit{siblingElements} & : & \mathit{Node} \to \mathit{Set}(\mathit{Node}) \\
\mathit{siblingElements}(x) & = & \{\, x_2 \mid x_1 \in \mathit{parent}(x),\ x_2 \in \mathit{subnodes}(x_1),\ \mathit{isElement}(x_2) \,\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{siblingsOfType} & : & \mathit{Node} \to \mathit{Set}(\mathit{Node}) \\
\mathit{siblingsOfType}(x) & = & \{\, x_1 \mid x_1 \in \mathit{siblingElements}(x),\ \mathit{isElement}(x),\ \mathit{name}(x) = \mathit{name}(x_1) \,\}
\end{array}
$$

3 Semantics of patterns

Figure 1 shows the semantics of patterns, corresponding to [XSL, Section 2.6]. The term $\mathcal{M}[\![p]\!]x$ is true if pattern p matches node x, the term $\mathcal{S}[\![p]\!]x$ denotes the set of nodes selected by pattern p from current node x, and the term $\mathcal{Q}[\![q]\!]x$ is true if qualifier q holds for current node x.

References

[Clark] James Clark, Message to XSL mailing list, 1 January 1999.

[Halmos] Paul R. Halmos, *Naive Set Theory*, Springer-Verlag, 1970.

[XSL] Extensible Stylesheet Language (XSL), Version 1.0, World Wide Web Consortium Working Draft 16 December 1998. `http://www.w3.org/TR/WD-xsl`

$$
\begin{array}{lcl}
\mathcal{M} & : & \mathit{Pattern} \times \mathit{Node} \to \mathit{Boolean} \\
\mathcal{M}[\![p]\!]x & = & x \in \{\, x_2 \mid x_1 \in \mathit{subnodes}^*(\mathit{root}(\mathit{document}(x))),\ x_2 \in \mathcal{S}[\![p]\!](x_1) \,\}
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{S} & : & \mathit{Pattern} \times \mathit{Node} \to \mathit{Set}(\mathit{Node}) \\
\mathcal{S}[\![p_1\,\texttt{|}\,p_2]\!]x & = & \mathcal{S}[\![p_1]\!]x \cup \mathcal{S}[\![p_2]\!]x \\
\mathcal{S}[\![\texttt{/}p]\!]x & = & \mathcal{S}[\![p]\!](\mathit{root}(\mathit{document}(x))) \\
\mathcal{S}[\![\texttt{//}p]\!]x & = & \{\, x_2 \mid x_1 \in \mathit{subnodes}^*(\mathit{root}(\mathit{document}(x))),\ x_2 \in \mathcal{S}[\![p]\!]x_1 \,\} \\
\mathcal{S}[\![p_1\texttt{/}p_2]\!]x & = & \{\, x_2 \mid x_1 \in \mathcal{S}[\![p_1]\!]x,\ x_2 \in \mathcal{S}[\![p_2]\!]x_1 \,\} \\
\mathcal{S}[\![p_1\texttt{//}p_2]\!]x & = & \{\, x_3 \mid x_1 \in \mathcal{S}[\![p_1]\!]x,\ x_2 \in \mathit{subnodes}^*(x_1),\ x_3 \in \mathcal{S}[\![p_2]\!]x_2 \,\} \\
\mathcal{S}[\![p\texttt{[}q\texttt{]}]\!]x & = & \{\, x_1 \mid x_1 \in \mathcal{S}[\![p]\!]x,\ \mathcal{Q}[\![q]\!]x_1 \,\} \\
\mathcal{S}[\![n]\!]x & = & \{\, x_1 \mid x_1 \in \mathit{subnodes}(x),\ \mathit{isElement}(x_1),\ \mathit{name}(x_1) = n \,\} \\
\mathcal{S}[\![\texttt{*}]\!]x & = & \{\, x_1 \mid x_1 \in \mathit{subnodes}(x),\ \mathit{isElement}(x_1) \,\} \\
\mathcal{S}[\![\texttt{@}n]\!]x & = & \{\, x_1 \mid x_1 \in \mathit{subnodes}(x),\ \mathit{isAttribute}(x_1),\ \mathit{name}(x_1) = n \,\} \\
\mathcal{S}[\![\texttt{@*}]\!]x & = & \{\, x_1 \mid x_1 \in \mathit{subnodes}(x),\ \mathit{isAttribute}(x_1) \,\} \\
\mathcal{S}[\![\texttt{text()}]\!]x & = & \{\, x_1 \mid x_1 \in \mathit{subnodes}(x),\ \mathit{isText}(x_1) \,\} \\
\mathcal{S}[\![\texttt{comment()}]\!]x & = & \{\, x_1 \mid x_1 \in \mathit{subnodes}(x),\ \mathit{isComment}(x_1) \,\} \\
\mathcal{S}[\![\texttt{pi(}n\texttt{)}]\!]x & = & \{\, x_1 \mid x_1 \in \mathit{subnodes}(x),\ \mathit{isPI}(x_1),\ \mathit{name}(x_1) = n \,\} \\
\mathcal{S}[\![\texttt{pi()}]\!]x & = & \{\, x_1 \mid x_1 \in \mathit{subnodes}(x),\ \mathit{isPI}(x_1) \,\} \\
\mathcal{S}[\![\texttt{id(}p\texttt{)}]\!]x & = & \{\, x_2 \mid x_1 \in \mathcal{S}[\![p]\!]x,\ s \in \mathit{split}(\mathit{value}(x_1)),\ x_2 \in \mathit{id}(\mathit{document}(x), s) \,\} \\
\mathcal{S}[\![\texttt{id(}s\texttt{)}]\!]x & = & \{\, x_1 \mid s_1 \in \mathit{split}(s),\ x_1 \in \mathit{id}(\mathit{document}(x), s_1) \,\} \\
\mathcal{S}[\![\texttt{ancestor(}p\texttt{)}]\!]x & = & \mathit{last}(\{\, x_1 \mid x_1 \in \mathit{parent}^+(x),\ \mathcal{M}[\![p]\!]x_1 \,\}) \\
\mathcal{S}[\![\texttt{ancestor-or-self(}p\texttt{)}]\!]x & = & \mathit{last}(\{\, x_1 \mid x_1 \in \mathit{parent}^*(x),\ \mathcal{M}[\![p]\!]x_1 \,\}) \\
\mathcal{S}[\![\texttt{.}]\!]x & = & \{\, x \,\} \\
\mathcal{S}[\![\texttt{..}]\!]x & = & \mathit{parent}(x)
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{Q} & : & \mathit{Qualifier} \times \mathit{Node} \to \mathit{Boolean} \\
\mathcal{Q}[\![q_1\ \texttt{and}\ q_2]\!]x & = & \mathcal{Q}[\![q_1]\!]x \wedge \mathcal{Q}[\![q_2]\!]x \\
\mathcal{Q}[\![q_1\ \texttt{or}\ q_2]\!]x & = & \mathcal{Q}[\![q_1]\!]x \vee \mathcal{Q}[\![q_2]\!]x \\
\mathcal{Q}[\![\texttt{not(}q\texttt{)}]\!]x & = & \neg\mathcal{Q}[\![q]\!]x \\
\mathcal{Q}[\![\texttt{first-of-type()}]\!]x & = & x \in \mathit{first}(\mathit{siblingsOfType}(x)) \\
\mathcal{Q}[\![\texttt{last-of-type()}]\!]x & = & x \in \mathit{last}(\mathit{siblingsOfType}(x)) \\
\mathcal{Q}[\![\texttt{first-of-any()}]\!]x & = & x \in \mathit{first}(\mathit{siblingElements}(x)) \\
\mathcal{Q}[\![\texttt{last-of-any()}]\!]x & = & x \in \mathit{last}(\mathit{siblingElements}(x)) \\
\mathcal{Q}[\![p\texttt{=}s]\!]x & = & \{\, x_1 \mid x_1 \in \mathcal{S}[\![p]\!]x,\ \mathit{value}(x_1) = s \,\} \neq \emptyset \\
\mathcal{Q}[\![p]\!]x & = & \mathcal{S}[\![p]\!]x \neq \emptyset
\end{array}
$$

Figure 1: Semantics of patterns