The Essence of XML: Matching in detail

Erik Knoop
Center for Software Technology
Department of Information and Computing Sciences
Utrecht University
eknoop@cs.uu.nl

January 13th 2004

1 Introduction

In this article we will match a XML document against two XML Schemas. That is, we give a validated document in the syntax of [1] and a schema, also in the same syntax and match the former against the latter. After that we make a small adjustment to the schema and see what happens.

2 Running example

The example XML Schema is defined below.

```
define element configuration of type configurationType
define type configurationType {
 element shuttle of type miles,
 element laser of type miles
}
define type miles restricts xs:integer
define type feet restricts xs:integer
```

The example document is defined below.

```
element configuration of type configurationType{
 element shuttle of type miles { 120 },
 element laser of type miles { 10023 }
}
```

3 Matching the first example

In this section we show a derivation in the style of [1]. We start with the main judgement in figure 1.

```
 configuration
yields
 configuration of type configurationType

 configurationType
resolves to
 element shuttle of type miles,
 element laser of type miles

 configuration
substitutes for
 configuration

 configurationType
derives from
 configurationType

 element shuttle of type miles { 120 },
 element laser of type miles { 10023 }
matches
 element shuttle of type miles,
 element laser of type miles
```

```
 element configuration of type configurationType {
  element shuttle of type miles { 120 },
  element laser of type miles { 10023 }
 }
matches
  element configuration
```

Figure 1: Matching example 1 (part 1)

define element configuration of type configurationType

---

configuration
yields
configuration of type configurationType

define type configurationType {
element shuttle of type miles,
element laser of type miles
}

---

configurationType
resolves to
element shuttle of type miles,
element laser of type miles

---

configuration
substitutes for
configuration

---

configurationType
derives from
configurationType

element shuttle of type miles { 120 }
matches
element shuttle of type miles

element laser of type miles { 10023 }
matches
element laser of type miles

---

element shuttle of type miles { 120 },
element laser of type miles { 10023 }
matches
element shuttle of type miles,
element laser of type miles

Figure 2: Matching example 1 (part 2)

element shuttle of type miles
yields
element shuttle of type miles

miles
resolves to
xs:integer

shuttle
substitutes for
shuttle

miles
derives from
miles

120
matches
xs:integer

---

element shuttle of type miles { 120 }
matches
element shuttle of type miles

xs:integer
derives from
xs:integer

120
matches
xs:integer

---

xs:integer
derives from
xs:integer

Figure 3: Matching example 1 (part 3)

element laser of type miles
yields
element laser of type miles

miles
resolves to
xs:integer

laser
substitutes for
laser

miles
derives from
miles

10023
matches
xs:integer

---

element laser of type miles { 10023 }
matches
element laser of type miles

xs:integer
derives from
xs:integer

10023
matches
xs:integer

---

xs:integer
derives from
xs:integer

Figure 4: Matching example 1 (part 4)

configuration
yields
configuration of type configurationType

configurationType
resolves to
element shuttle of type feet,
element laser of type miles

configuration
substitutes for
configuration

configurationType
derives from
configurationType

element shuttle of type miles { 120 },
element laser of type miles { 10023 }
matches
element shuttle of type feet,
element laser of type miles

---

element configuration of type configurationType {
element shuttle of type miles { 120 },
element laser of type miles { 10023 }
}
matches
element configuration

Figure 5: Matching example 2 (part 1)

4 A slight change

What happens if we change our example a bit?

```
define element configuration of type configurationType
define type configurationType {
 element shuttle of type feet,
 element laser of type miles
}
define type miles restricts xs:integer
define type feet restricts xs:integer
```

The example document remains the same.

```
element configuration of type configurationType{
 element shuttle of type miles { 120 },
 element laser of type miles { 10023 }
}
```

5 Matching the second example

In this section we show a derivation in the style of [1]. We start with the main judgement in figure 5.

define element configuration of type configurationType

---

configuration
yields
configuration of type configurationType

define type configurationType {
element shuttle of type feet,
element laser of type miles
}

---

configurationType
resolves to
element shuttle of type feet,
element laser of type miles

---

configuration
substitutes for
configuration

---

configurationType
derives from
configurationType

element shuttle of type miles { 120 }
matches
element shuttle of type feet

element laser of type miles { 10023 }
matches
element laser of type miles

---

element shuttle of type miles { 120 },
element laser of type miles { 10023 }
matches
element shuttle of type feet,
element laser of type miles

Figure 6: Matching example 2 (part 2)

element shuttle of type feet
yields
element shuttle of type feet

feet
resolves to
xs:integer

shuttle
substitutes for
shuttle

miles
derives from FALSE!
feet

120
matches
xs:integer

---

element shuttle of type miles { 120 }
matches
element shuttle of type feet

xs:integer
derives from
xs:integer

120
matches
xs:integer

---

xs:integer
derives from
xs:integer

Figure 7: Matching example 2 (part 3)

element laser of type miles
yields
element laser of type miles

miles
resolves to
xs:integer

laser
substitutes for
laser

miles
derives from
miles

10023
matches
xs:integer

---

element laser of type miles { 10023 }
matches
element laser of type miles

xs:integer
derives from
xs:integer

10023
matches
xs:integer

---

xs:integer
derives from
xs:integer

Figure 8: Matching example 2 (part 4)

6 Conclusion

In the the second example matching there was a judgement which didn't hold. That is, the whole matching process fails.

Whether this is the named typing of [1] is to be found out. It might be hint, but the only way to be certain where the is named typing in [1] is, is to define a formal mapping between MSL and [1].

References

[1] Jérôme Siméon and Philip Wadler. The Essence of XML. In *Proceedings of POPL 2003*, New Orleans, Louisiana, USA, 2003