

Introduction to Streaming Validation

Pierre Genevès
CNRS

(slides mostly based on Marc H. Scholl's ones)

Grenoble Universités, 2011–2012

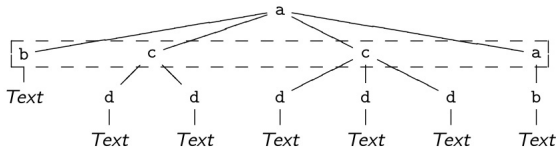
Validating XML Documents Against DTDs

- To validate against this DTD ...

DTD featuring regular expression (RE) content models

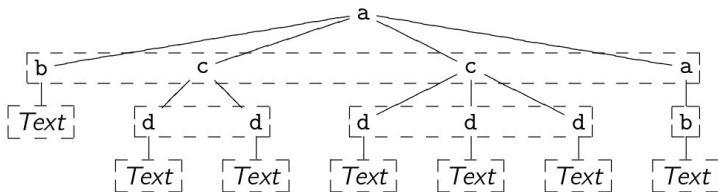
```
1 <!DOCTYPE a [  
2   <!ELEMENT a (b, c*, a?)>  
3   <!ELEMENT b (#PCDATA) >  
4   <!ELEMENT c (d, d+) >  
5   <!ELEMENT d (#PCDATA) >  
6 ]>
```

... means to check that the **sequence of child nodes** for each element **matches** its RE content model:

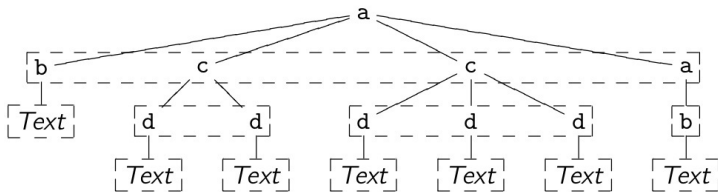


The techniques necessary for this checking are well-known from compiler-construction. We review them via an example in the sequel.

- When, during RE matching, we encounter a child element t , we need to **recursively check t 's content model $cm(t)$** in the same fashion:

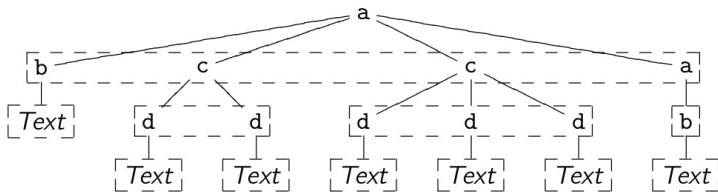


- When, during RE matching, we encounter a child element t , we need to **recursively check t 's content model $cm(t)$** in the same fashion:



$cm(a) = b, c^*, a?$
 $cm(b) = \#PCDATA$
 $cm(c) = d, d^+$
 $cm(d) = \#PCDATA$

- When, during RE matching, we encounter a child element t , we need to **recursively check t 's content model $cm(t)$** in the same fashion:



$cm(a) = b, c^*, a?$
 $cm(b) = \#PCDATA$
 $cm(c) = d, d^+$
 $cm(d) = \#PCDATA$

SAX and DTD validation?

- Can we use SAX to drive this validation (= RE matching) process?
- If so, which SAX events do we need to catch to implement this?

Regular Expressions

- To provide adequate support for SAX-based XML validation, we assume REs of the following structure:

$RE = \emptyset$	matches nothing
ϵ	matches empty sequence of SAX events
$\#PCDATA$	matches <i>characters</i> (.)
t	matches <i>startElement</i> (t, \cdot)
RE, RE	concatenation
RE^+	one-or-more repetitions
RE^*	zero-or-more repetitions
$RE?$	option
$RE RE$	alternative
(RE)	



- \emptyset and ϵ are *not* the same thing.
- In the *startElement*(t, \cdot) callback we can process `<!ATTLIST t ...>` declarations (not discussed here)

- Associated with each RE is the **regular language** $L(RE)$ (here: sets of sequences of SAX events) this RE **accepts**:

$$L(\emptyset) = \emptyset$$

$$L(\epsilon) = \{\epsilon\}$$

$$L(\#PCDATA) = \{characters(\cdot)\}$$

$$L(t) = \{startElement(t, \cdot)\}^{18}$$

$$L(RE_1, RE_2) = \{s_1 s_2 \mid s_1 \in L(RE_1), s_2 \in L(RE_2)\}$$

$$L(RE^+) = \bigcup_{i=1}^{\infty} L(RE^i)$$

$$L(RE^*) = \bigcup_{i=0}^{\infty} L(RE^i)$$

$$L(RE?) = \{\epsilon\} \cup L(RE)$$

$$L(RE_1 \mid RE_2) = L(RE_1) \cup L(RE_2)$$

- N.B.:** $RE^0 = \epsilon$ and $RE^i = RE, RE^{i-1}$.

¹⁸To save trees, we will abbreviate this as $\{t\}$ from now on.

Example

- Which sequence of SAX events is matched by the RE `#PCDATA | b*`?

$L(\#PCDATA | b^*)$

Example

- Which sequence of SAX events is matched by the RE `#PCDATA | b*`?

$$L(\#PCDATA | b^*)$$

$$= L(\#PCDATA) \cup L(b^*)$$

Example

- Which sequence of SAX events is matched by the RE $\#PCDATA \mid b^*$?

$$L(\#PCDATA \mid b^*)$$

$$= L(\#PCDATA) \cup L(b^*)$$

$$= L(\#PCDATA) \cup \bigcup_{i=0}^{\infty} L(b^i)$$

Example

- Which sequence of SAX events is matched by the RE $\#PCDATA \mid b^*$?

$$L(\#PCDATA \mid b^*)$$

$$= L(\#PCDATA) \cup L(b^*)$$

$$= L(\#PCDATA) \cup \bigcup_{i=0}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup \bigcup_{i=1}^{\infty} L(b^i)$$

Example

- Which sequence of SAX events is matched by the RE $\#PCDATA \mid b^*$?

$$L(\#PCDATA \mid b^*)$$

$$= L(\#PCDATA) \cup L(b^*)$$

$$= L(\#PCDATA) \cup \bigcup_{i=0}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup \bigcup_{i=1}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup \bigcup_{i=2}^{\infty} L(b^i)$$

Example

- Which sequence of SAX events is matched by the RE #PCDATA | b*?

$$L(\#PCDATA | b^*)$$

$$= L(\#PCDATA) \cup L(b^*)$$

$$= L(\#PCDATA) \cup \bigcup_{i=0}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup \bigcup_{i=1}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup \bigcup_{i=2}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup L(b^2) \cup \bigcup_{i=3}^{\infty} L(b^i)$$

Example

- Which sequence of SAX events is matched by the RE #PCDATA | b*?

$$L(\#PCDATA | b^*)$$

$$= L(\#PCDATA) \cup L(b^*)$$

$$= L(\#PCDATA) \cup \bigcup_{i=0}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup \bigcup_{i=1}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup \bigcup_{i=2}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup L(b^2) \cup \bigcup_{i=3}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(\epsilon) \cup L(b) \cup L(b, b^1) \cup \dots$$

Example

- Which sequence of SAX events is matched by the RE #PCDATA | b*?

$$L(\#PCDATA | b^*)$$

$$= L(\#PCDATA) \cup L(b^*)$$

$$= L(\#PCDATA) \cup \bigcup_{i=0}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup \bigcup_{i=1}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup \bigcup_{i=2}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup L(b^2) \cup \bigcup_{i=3}^{\infty} L(b^i)$$

$$= L(\#PCDATA) \cup L(\epsilon) \cup L(b) \cup L(b, b^1) \cup \dots$$

$$= L(\#PCDATA) \cup L(\epsilon) \cup L(b) \cup \{s_1 s_2 \mid s_1 \in L(b), s_2 \in L(b^1)\} \cup \dots$$

Example

- Which sequence of SAX events is matched by the RE $\#PCDATA \mid b^*$?

$$\begin{aligned} &L(\#PCDATA \mid b^*) \\ &= L(\#PCDATA) \cup L(b^*) \\ &= L(\#PCDATA) \cup \bigcup_{i=0}^{\infty} L(b^i) \\ &= L(\#PCDATA) \cup L(b^0) \cup \bigcup_{i=1}^{\infty} L(b^i) \\ &= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup \bigcup_{i=2}^{\infty} L(b^i) \\ &= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup L(b^2) \cup \bigcup_{i=3}^{\infty} L(b^i) \\ &= L(\#PCDATA) \cup L(\varepsilon) \cup L(b) \cup L(b, b^1) \cup \dots \\ &= L(\#PCDATA) \cup L(\varepsilon) \cup L(b) \cup \{s_1 s_2 \mid s_1 \in L(b), s_2 \in L(b^1)\} \cup \dots \\ &= \{characters(\cdot), \varepsilon, b, bb, \dots\} \end{aligned}$$

Example

- Which sequence of SAX events is matched by the RE $\#PCDATA \mid b^*$?

$$\begin{aligned} &L(\#PCDATA \mid b^*) \\ &= L(\#PCDATA) \cup L(b^*) \\ &= L(\#PCDATA) \cup \bigcup_{i=0}^{\infty} L(b^i) \\ &= L(\#PCDATA) \cup L(b^0) \cup \bigcup_{i=1}^{\infty} L(b^i) \\ &= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup \bigcup_{i=2}^{\infty} L(b^i) \\ &= L(\#PCDATA) \cup L(b^0) \cup L(b^1) \cup L(b^2) \cup \bigcup_{i=3}^{\infty} L(b^i) \\ &= L(\#PCDATA) \cup L(\varepsilon) \cup L(b) \cup L(b, b^1) \cup \dots \\ &= L(\#PCDATA) \cup L(\varepsilon) \cup L(b) \cup \{s_1 s_2 \mid s_1 \in L(b), s_2 \in L(b^1)\} \cup \dots \\ &= \{\text{characters}(\cdot), \varepsilon, b, bb, \dots\} \end{aligned}$$

 $L(d, d^+) = ?$

Evaluating Regular Expressions (Matching)

- Now that we are this far, we know that matching a sequence of SAX events s against the content model of element t means to carry out the test

$$s \stackrel{?}{\in} L(cm(t)) .$$

Evaluating Regular Expressions (Matching)

- Now that we are this far, we know that matching a sequence of SAX events s against the content model of element t means to carry out the test

$$s \stackrel{?}{\in} L(cm(t)) .$$

- $L(cm(t))$, however, might be infinite or otherwise too costly to construct inside our DTD validator.

Evaluating Regular Expressions (Matching)

- Now that we are this far, we know that matching a sequence of SAX events s against the content model of element t means to carry out the test

$$s \stackrel{?}{\in} L(cm(t)) .$$

- $L(cm(t))$, however, might be infinite or otherwise too costly to construct inside our DTD validator.
- We thus follow a different path that avoids to enumerate $L(cm(t))$ at all.

Evaluating Regular Expressions (Matching)

- Now that we are this far, we know that matching a sequence of SAX events s against the content model of element t means to carry out the test

$$s \overset{?}{\in} L(cm(t)) .$$

- $L(cm(t))$, however, might be infinite or otherwise too costly to construct inside our DTD validator.
- We thus follow a different path that avoids to enumerate $L(cm(t))$ at all.
- Instead, we will use the **derivative** $s \setminus RE$ of RE with respect to input event s :

$$L(s \setminus RE) = \{s' \mid s s' \in L(RE)\}$$

“ $s \setminus RE$ matches everything matched by RE , with head s cut off.”

- We can use the derivative operator \backslash to develop a simple **RE matching procedure**.

Suppose we are to match the SAX event sequence $s_1s_2s_3$ against RE :

$$s_1s_2s_3 \in L(RE)$$

- We can use the derivate operator \backslash to develop a simple **RE matching procedure**.

Suppose we are to match the SAX event sequence $s_1s_2s_3$ against RE :

$$s_1s_2s_3 \in L(RE) \iff s_1s_2s_3\epsilon \in L(RE)$$

- We can use the derivate operator \setminus to develop a simple **RE matching procedure**.

Suppose we are to match the SAX event sequence $s_1s_2s_3$ against RE :

$$\begin{aligned} s_1s_2s_3 \in L(RE) &\Leftrightarrow s_1s_2s_3\epsilon \in L(RE) \\ &\Leftrightarrow s_2s_3\epsilon \in L(s_1 \setminus RE) \end{aligned}$$

- We can use the derivate operator \setminus to develop a simple **RE matching procedure**.

Suppose we are to match the SAX event sequence $s_1s_2s_3$ against RE :

$$\begin{aligned} s_1s_2s_3 \in L(RE) &\Leftrightarrow s_1s_2s_3\epsilon \in L(RE) \\ &\Leftrightarrow s_2s_3\epsilon \in L(s_1 \setminus RE) \\ &\Leftrightarrow s_3\epsilon \in L(s_2 \setminus (s_1 \setminus RE)) \end{aligned}$$

- We can use the derivate operator \setminus to develop a simple **RE matching procedure**.

Suppose we are to match the SAX event sequence $s_1s_2s_3$ against RE :

$$\begin{aligned}
 s_1s_2s_3 \in L(RE) &\Leftrightarrow s_1s_2s_3\varepsilon \in L(RE) \\
 &\Leftrightarrow s_2s_3\varepsilon \in L(s_1 \setminus RE) \\
 &\Leftrightarrow s_3\varepsilon \in L(s_2 \setminus (s_1 \setminus RE)) \\
 &\Leftrightarrow \varepsilon \in L(s_3 \setminus (s_2 \setminus (s_1 \setminus RE))) .
 \end{aligned}$$

- We can use the derivate operator \setminus to develop a simple **RE matching procedure**.

Suppose we are to match the SAX event sequence $s_1s_2s_3$ against RE :

$$\begin{aligned}
 s_1s_2s_3 \in L(RE) &\Leftrightarrow s_1s_2s_3\varepsilon \in L(RE) \\
 &\Leftrightarrow s_2s_3\varepsilon \in L(s_1 \setminus RE) \\
 &\Leftrightarrow s_3\varepsilon \in L(s_2 \setminus (s_1 \setminus RE)) \\
 &\Leftrightarrow \varepsilon \in L(s_3 \setminus (s_2 \setminus (s_1 \setminus RE))) .
 \end{aligned}$$

- We thus have solved our matching problem if
 - ① we can efficiently **test for ε -containment** for a given RE, and

- We can use the derivate operator \setminus to develop a simple **RE matching procedure**.

Suppose we are to match the SAX event sequence $s_1s_2s_3$ against RE :

$$\begin{aligned}
 s_1s_2s_3 \in L(RE) &\Leftrightarrow s_1s_2s_3\varepsilon \in L(RE) \\
 &\Leftrightarrow s_2s_3\varepsilon \in L(s_1 \setminus RE) \\
 &\Leftrightarrow s_3\varepsilon \in L(s_2 \setminus (s_1 \setminus RE)) \\
 &\Leftrightarrow \varepsilon \in L(s_3 \setminus (s_2 \setminus (s_1 \setminus RE))) .
 \end{aligned}$$

- We thus have solved our matching problem if
 - ① we can efficiently **test for ε -containment** for a given RE , and
 - ② we are able to **compute** $L(s \setminus RE)$ for any given input event s and any RE .

 Ad ①: Testing for ϵ 's presence in a regular language.

Define a predicate (boolean function) $nullable(RE)$ such that

$$nullable(RE) \Leftrightarrow \epsilon \in L(RE) .$$

$$nullable(\emptyset) = false$$

$$nullable(\epsilon) = true$$

$$nullable(\#PCDATA) = false$$

$$nullable(t) =$$

$$nullable(RE_1, RE_2) =$$

$$nullable(RE^+) =$$

$$nullable(RE^*) =$$

$$nullable(RE?) =$$

$$nullable(RE_1 | RE_2) =$$

Example

Does $L(\#PCDATA \mid b^*)$ contain the empty SAX event sequence ε ?

nullable($\#PCDATA \mid b^*$)

Example

Does $L(\#PCDATA \mid \mathbf{b}^*)$ contain the empty SAX event sequence ε ?

$$\text{nullable}(\#PCDATA \mid \mathbf{b}^*) = \text{nullable}(\#PCDATA) \vee \text{nullable}(\mathbf{b}^*)$$

Example

Does $L(\#PCDATA \mid \mathbf{b}^*)$ contain the empty SAX event sequence ε ?

$$\begin{aligned} \text{nullable}(\#PCDATA \mid \mathbf{b}^*) &= \text{nullable}(\#PCDATA) \vee \text{nullable}(\mathbf{b}^*) \\ &= \text{false} \vee \text{true} \end{aligned}$$

Example

Does $L(\#PCDATA \mid b^*)$ contain the empty SAX event sequence ε ?

$$\begin{aligned} nullable(\#PCDATA \mid b^*) &= nullable(\#PCDATA) \vee nullable(b^*) \\ &= false \vee true \\ &= true . \end{aligned}$$

Example

Does $L(\#PCDATA \mid b^*)$ contain the empty SAX event sequence ε ?

$$\begin{aligned} nullable(\#PCDATA \mid b^*) &= nullable(\#PCDATA) \vee nullable(b^*) \\ &= false \vee true \\ &= true . \end{aligned}$$

 $nullable(\text{Prof?}, \text{Dr}, (\text{rernat} \mid \text{emer} \mid \text{phil})^+) = ?$

Ad ②: Note that the **derivative** $s\backslash$ is an operator on REs (to REs). We define it like follows and justify this definition on the next slides.

$$\begin{aligned}
 s\backslash\emptyset &= \emptyset \\
 s\backslash\varepsilon &= \emptyset \\
 s\backslash\#\text{PCDATA} &= \begin{cases} \varepsilon & \text{if } s = \text{characters}(\cdot) \\ \emptyset & \text{otherwise} \end{cases} \\
 s\backslash t &= \begin{cases} \varepsilon & \text{if } s = \text{startElement}(t, \cdot) \quad // \star \text{ recursively match } cm(t) \\ \emptyset & \text{otherwise} \end{cases} \\
 s\backslash(RE_1, RE_2) &= \begin{cases} ((s\backslash RE_1), RE_2) \mid (s\backslash RE_2) & \text{if } nullable(RE_1) \\ (s\backslash RE_1), RE_2 & \text{otherwise} \end{cases} \\
 s\backslash RE^+ &= (s\backslash RE), RE^* \\
 s\backslash RE^* &= (s\backslash RE), RE^* \\
 s\backslash RE? &= s\backslash RE \\
 s\backslash(RE_1 \mid RE_2) &= (s\backslash RE_1) \mid (s\backslash RE_2)
 \end{aligned}$$

Correctness: Case Analysis (I)

To assess the correctness of this derivative construction $s \setminus RE = RE'$ we can systematically check all 9 cases for **language equivalence**, *i.e.*

$$L(s \setminus RE) = L(RE') .$$

1 $RE = \emptyset$:

$$\begin{aligned} L(s \setminus \emptyset) &= \{s' \mid s s' \in L(\emptyset)\} \\ &= \{s' \mid s s' \in \emptyset\} \\ &= \emptyset \\ &= L(\emptyset). \end{aligned}$$

Correctness: Case Analysis (II)

2 $RE = \varepsilon$:

$$\begin{aligned}L(s \setminus \varepsilon) &= \{s' \mid s s' \in L(\varepsilon)\} \\ &= \{s' \mid s s' \in \{\varepsilon\}\} \\ &= \emptyset \\ &= L(\emptyset).\end{aligned}$$

3 $RE = \#PCDATA, s = \text{characters}(\cdot)$:

$$\begin{aligned}L(\text{characters}(\cdot) \setminus \#PCDATA) &= \{s' \mid \text{characters}(\cdot) s' \in L(\#PCDATA)\} \\ &= \{s' \mid \text{characters}(\cdot) s' \in \{\text{characters}(\cdot)\}\} \\ &= \{\varepsilon\} \\ &= L(\varepsilon).\end{aligned}$$

Correctness: Case Analysis (III)

$RE = \#PCDATA, s \neq characters(\cdot)$:

$$\begin{aligned}L(s \setminus \#PCDATA) &= \{s' \mid s s' \in L(\#PCDATA)\} \\ &= \{s' \mid s s' \in \{characters(\cdot)\}\} \\ &= \emptyset \\ &= L(\emptyset).\end{aligned}$$

④ $RE = t$. Analogous to ③.

⑤ $RE = RE_1, RE_2, nullable(RE_1) = false$:

$$\begin{aligned}L(s \setminus (RE_1, RE_2)) &= \{s' \mid s s' \in L(RE_1, RE_2)\} \\ &= \{s' \mid s' \in L((s \setminus RE_1), RE_2)\} \\ &= L((s \setminus RE_1), RE_2).\end{aligned}$$

Correctness: Case Analysis (IV)

$RE = RE_1, RE_2$, $nullable(RE_1) = true$:

$$\begin{aligned}L(s \setminus (RE_1, RE_2)) &= \{s' \mid s s' \in L(RE_1, RE_2)\} \\&= \{s' \mid s s' \in L(RE_2) \vee s s' \in L(RE_1, RE_2)\} \\&= \{s' \mid s' \in L(s \setminus RE_2) \vee s' \in L((s \setminus RE_1), RE_2)\} \\&= \{s' \mid s' \in L(s \setminus RE_2)\} \cup \{s' \mid s' \in L((s \setminus RE_1), RE_2)\} \\&= L(s \setminus RE_2) \cup L((s \setminus RE_1), RE_2) \\&= L((s \setminus RE_2) \mid ((s \setminus RE_1), RE_2)).\end{aligned}$$

Correctness: Case Analysis (V)

6 $RE = RE_1 \mid RE_2$:

$$\begin{aligned}L(s \setminus (RE_1 \mid RE_2)) &= \{s' \mid s s' \in L(RE_1 \mid RE_2)\} \\&= \{s' \mid s s' \in L(RE_1) \cup L(RE_2)\} \\&= \{s' \mid s s' \in L(RE_1)\} \cup \{s' \mid s s' \in L(RE_2)\} \\&= \{s' \mid s' \in L(s \setminus RE_1)\} \cup \{s' \mid s' \in L(s \setminus RE_2)\} \\&= L(s \setminus RE_1) \cup L(s \setminus RE_2) \\&= L((s \setminus RE_1) \mid (s \setminus RE_2)).\end{aligned}$$

Correctness: Case Analysis (VI)

7 $RE = RE^*$, $nullable(RE) = false$:

$$\begin{aligned}L(s \setminus RE^*) &= L(s \setminus (\epsilon \mid (RE, RE^*))) \\ &= L(s \setminus \epsilon) \cup L(s \setminus (RE, RE^*)) \\ &= L(s \setminus (RE, RE^*)) \\ &= L((s \setminus RE), RE^*).\end{aligned}$$

$RE = RE^*$, $nullable(RE) = true$:

$$\begin{aligned}L(s \setminus RE^*) &= L(s \setminus (\epsilon \mid (RE, RE^*))) \\ &= L((s \setminus \epsilon) \mid (s \setminus (RE, RE^*))) \\ &= L(\emptyset \mid (s \setminus (RE, RE^*))) \\ &= L(s \setminus (RE, RE^*)) \\ &= L((s \setminus RE^*) \mid ((s \setminus RE), RE^*)) \\ &= L(s \setminus RE^*) \cup L((s \setminus RE), RE^*) \\ &= L((s \setminus RE), RE^*).\end{aligned}$$

Correctness: Case Analysis (VII)

- 8 $RE = RE^+$. Follows from $RE^+ = RE \mid RE, RE^*$.
- 9 $RE = RE?$. Follows from $RE? = \varepsilon \mid RE$.

Matching SAX events against an RE

Assume the RE content model $b, c^*, a?$ is to be matched against the SAX events $bcca$.¹⁹

To validate,

- 1 construct the corresponding derivative $RE' = a \setminus (c \setminus (c \setminus (b \setminus (b, c^*, a?))))$,
- 2 then test $nullable(RE')$.

Hint: To simplify phase ①, use the following **laws**, valid for REs in general:

ϵ^*	$=$	ϵ	ϵ, RE	$=$	RE
\emptyset^*	$=$	ϵ	\emptyset, RE	$=$	\emptyset
ϵ^+	$=$	ϵ	RE, ϵ	$=$	RE
\emptyset^+	$=$	\emptyset	RE, \emptyset	$=$	\emptyset
$\epsilon?$	$=$	ϵ	$\emptyset \mid RE$	$=$	RE
$\emptyset?$	$=$	ϵ	$RE \mid \emptyset$	$=$	RE

¹⁹Actual event sequence:

$startElement(b, \cdot), startElement(c, \cdot), startElement(c, \cdot), startElement(a, \cdot)$.

Plugging It All Together

The following SAX callbacks use the aforementioned RE matching techniques to (partially) implement DTD validation **while parsing** the input XML document:

The input DTD (declaring the content models $cm(\cdot)$) is

```
<!DOCTYPE r [ ... ]>
```

Plugging It All Together

The following SAX callbacks use the aforementioned RE matching techniques to (partially) implement DTD validation **while parsing** the input XML document:

The input DTD (declaring the content models $cm(\cdot)$) is

```
<!DOCTYPE r [ ... ]>
```

startDocument()

```
S.empty();  
RE ← cm(r);  
return;
```

characters(·)

```
RE ←  
#PCDATA\RE;  
return;
```

startElement(t, ·)

```
RE ← t\RE;  
S.push(RE);  
RE ← cm(t);  
return;
```

endElement(t)

```
if nullable(RE) then  
| RE ← S.pop();  
else  
| ★ FAIL ★;  
return;
```

endDocument()

```
★ OK ★;
```

Plugging It All Together

The following SAX callbacks use the aforementioned RE matching techniques to (partially) implement DTD validation **while parsing** the input XML document:

The input DTD (declaring the content models $cm(\cdot)$) is

```
<!DOCTYPE r [ ... ]>
```

startDocument()

```
S.empty();  
RE ← cm(r);  
return;
```

characters(·)

```
RE ←  
#PCDATA\RE;  
return;
```

startElement(t, ·)

```
RE ← t\RE;  
S.push(RE);  
RE ← cm(t);  
return;
```

endElement(t)

```
if nullable(RE) then  
| RE ← S.pop();  
else  
| ★ FAIL ★;  
return;
```

endDocument()

```
★ OK ★;
```

N.B. Stack S is used to suspend [resume] the RE matching for a specific element node whenever SAX descends [ascends] the XML document tree.