Course: Introduction to Streaming Validation

Pierre Genevès
CNRS

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

University Grenoble Alpes

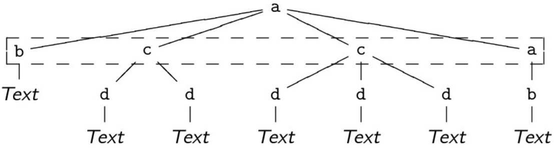
Validating XML Documents Against DTDs

- To validate against this DTD ...

DTD featuring regular expression (RE) content models

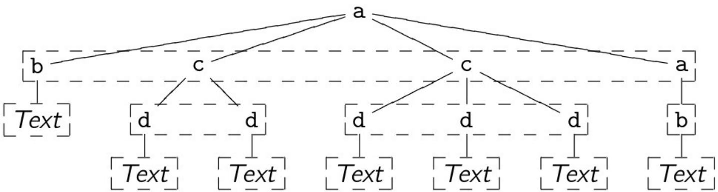
```
<!DOCTYPE a [
  <!ELEMENT a (b, c*, a?)>
  <!ELEMENT b (#PCDATA)  >
  <!ELEMENT c (d, d+)    >
  <!ELEMENT d (#PCDATA)  >
]>
```

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

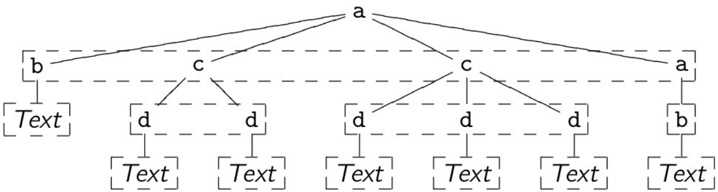


The techniques necessary for this checking are well-known from compiler-construction. We review them via an example in the sequel.

- 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:

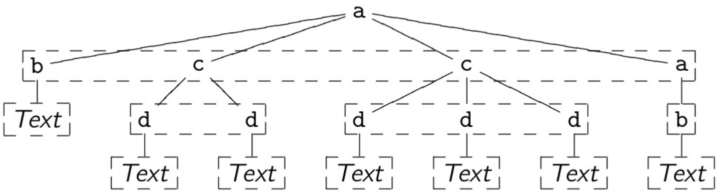


- When, during RE matching, we encounter a child element t, we need to **recursively check t's content model** $cm(t)$ in the same fashion:



$$
\begin{aligned}
cm(\mathtt{a}) &= \mathtt{b,c^{*},a?} \\
cm(\mathtt{b}) &= \mathtt{\#PCDATA} \\
cm(\mathtt{c}) &= \mathtt{d,d^{+}} \\
cm(\mathtt{d}) &= \mathtt{\#PCDATA}
\end{aligned}
$$

- 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:



$$
\begin{aligned}
cm(\mathtt{a}) &= \mathtt{b,c^*,a?} \\
cm(\mathtt{b}) &= \mathtt{\#PCDATA} \\
cm(\mathtt{c}) &= \mathtt{d,d^+} \\
cm(\mathtt{d}) &= \mathtt{\#PCDATA}
\end{aligned}
$$

✎ SAX and DTD validation?

1. Can we use SAX to drive this validation (= RE matching) process?
2. 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:

$$
\begin{array}{rcll}
RE & = & \emptyset & \text{matches } \textbf{nothing} \\
 & \mid & \varepsilon & \text{matches } \textbf{empty} \text{ sequence of SAX events} \\
 & \mid & \texttt{\#PCDATA} & \text{matches } \mathit{characters}(\cdot) \\
 & \mid & t & \text{matches } \mathit{startElement}(t, \cdot) \\
 & \mid & RE\,,RE & \textbf{concatenation} \\
 & \mid & RE^{+} & \textbf{one-or-more repetitions} \\
 & \mid & RE^{*} & \textbf{zero-or-more repetitions} \\
 & \mid & RE? & \textbf{option} \\
 & \mid & RE \mid RE & \textbf{alternative} \\
 & \mid & (RE) &
\end{array}
$$

- $\emptyset$ 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**:

$$
\begin{aligned}
L(\emptyset) &= \emptyset \\
L(\varepsilon) &= \{\varepsilon\} \\
L(\#\texttt{PCDATA}) &= \{\mathit{characters}(\cdot)\} \\
L(t) &= \{\mathit{startElement}(t,\cdot)\}^{18} \\
L(RE_1, RE_2) &= \{s_1 s_2 \mid s_1 \in L(RE_1),\, s_2 \in L(RE_2)\} \\
L(RE^+) &= \bigcup_{i=1}^{\infty} L(RE^i) \\
L(RE^*) &= \bigcup_{i=0}^{\infty} L(RE^i) \\
L(RE?) &= \{\varepsilon\} \cup L(RE) \\
L(RE_1 \mid RE_2) &= L(RE_1) \cup L(RE_2)
\end{aligned}
$$

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

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

Example

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

  $L(\#\texttt{PCDATA} \mid \texttt{b}^*)$

Example

- Which sequence of SAX events is matched by the RE `#PCDATA` | $\texttt{b}^*$?

$$L(\texttt{\#PCDATA} \mid \texttt{b}^*)$$

$$= \quad L(\texttt{\#PCDATA}) \cup L(\texttt{b}^*)$$

Example

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

$$
\begin{aligned}
& L(\#\texttt{PCDATA} \mid \texttt{b}^*) \\
& \quad = \quad L(\#\texttt{PCDATA}) \cup L(\texttt{b}^*) \\
& \quad = \quad L(\#\texttt{PCDATA}) \cup \textstyle\bigcup_{i=0}^{\infty} L(\texttt{b}^i)
\end{aligned}
$$

Example

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

$$L(\#\texttt{PCDATA} \mid \texttt{b}^*)$$

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

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

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

Example

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

$$
\begin{aligned}
& L(\texttt{\#PCDATA} \mid \texttt{b}^*) \\
& \quad = \quad L(\texttt{\#PCDATA}) \cup L(\texttt{b}^*) \\
& \quad = \quad L(\texttt{\#PCDATA}) \cup \textstyle\bigcup_{i=0}^{\infty} L(\texttt{b}^i) \\
& \quad = \quad L(\texttt{\#PCDATA}) \cup L(\texttt{b}^0) \cup \textstyle\bigcup_{i=1}^{\infty} L(\texttt{b}^i) \\
& \quad = \quad L(\texttt{\#PCDATA}) \cup L(\texttt{b}^0) \cup L(\texttt{b}^1) \cup \textstyle\bigcup_{i=2}^{\infty} L(\texttt{b}^i)
\end{aligned}
$$

Example

- Which sequence of SAX events is matched by the RE `#PCDATA` | $\mathtt{b}^*$?

$$
\begin{aligned}
& L(\#\mathtt{PCDATA} \mid \mathtt{b}^*) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\mathtt{b}^*) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup \bigcup_{i=0}^{\infty} L(\mathtt{b}^i) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\mathtt{b}^0) \cup \bigcup_{i=1}^{\infty} L(\mathtt{b}^i) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\mathtt{b}^0) \cup L(\mathtt{b}^1) \cup \bigcup_{i=2}^{\infty} L(\mathtt{b}^i) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\mathtt{b}^0) \cup L(\mathtt{b}^1) \cup L(\mathtt{b}^2) \cup \bigcup_{i=3}^{\infty} L(\mathtt{b}^i)
\end{aligned}
$$

Example

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

$$
\begin{aligned}
& L(\#\texttt{PCDATA} \mid \texttt{b}^*) \\
& \quad = \quad L(\#\texttt{PCDATA}) \cup L(\texttt{b}^*) \\
& \quad = \quad L(\#\texttt{PCDATA}) \cup \bigcup_{i=0}^{\infty} L(\texttt{b}^i) \\
& \quad = \quad L(\#\texttt{PCDATA}) \cup L(\texttt{b}^0) \cup \bigcup_{i=1}^{\infty} L(\texttt{b}^i) \\
& \quad = \quad L(\#\texttt{PCDATA}) \cup L(\texttt{b}^0) \cup L(\texttt{b}^1) \cup \bigcup_{i=2}^{\infty} L(\texttt{b}^i) \\
& \quad = \quad L(\#\texttt{PCDATA}) \cup L(\texttt{b}^0) \cup L(\texttt{b}^1) \cup L(\texttt{b}^2) \cup \bigcup_{i=3}^{\infty} L(\texttt{b}^i) \\
& \quad = \quad L(\#\texttt{PCDATA}) \cup L(\varepsilon) \cup L(\texttt{b}) \cup L(\texttt{b}, \texttt{b}^1) \cup \ldots
\end{aligned}
$$

Example

- Which sequence of SAX events is matched by the RE `#PCDATA` | $\mathtt{b}^*$?

$$
\begin{aligned}
& L(\#\mathtt{PCDATA} \mid \mathtt{b}^*) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\mathtt{b}^*) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup \textstyle\bigcup_{i=0}^{\infty} L(\mathtt{b}^i) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\mathtt{b}^0) \cup \textstyle\bigcup_{i=1}^{\infty} L(\mathtt{b}^i) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\mathtt{b}^0) \cup L(\mathtt{b}^1) \cup \textstyle\bigcup_{i=2}^{\infty} L(\mathtt{b}^i) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\mathtt{b}^0) \cup L(\mathtt{b}^1) \cup L(\mathtt{b}^2) \cup \textstyle\bigcup_{i=3}^{\infty} L(\mathtt{b}^i) \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\varepsilon) \cup L(\mathtt{b}) \cup L(\mathtt{b}, \mathtt{b}^1) \cup \ldots \\
& \quad = \quad L(\#\mathtt{PCDATA}) \cup L(\varepsilon) \cup L(\mathtt{b}) \cup \{s_1 s_2 \mid s_1 \in L(\mathtt{b}), s_2 \in L(\mathtt{b}^1)\} \cup \ldots
\end{aligned}
$$

Example

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

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

Example

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

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

✎ $L(\texttt{d},\texttt{d}^+) = \ ?$

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

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

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
$$s \stackrel{?}{\in} L(cm(t)) \quad .$$
- $L(cm(t))$, 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
$$s \stackrel{?}{\in} L(cm(t)) \quad .$$
- $L(cm(t))$, 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
$$s \stackrel{?}{\in} L(cm(t)) \enspace .$$
- $L(cm(t))$, 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** $s\backslash RE$ of RE with respect to input event s:

$$L(s\backslash RE) \quad = \quad \{s' \mid s\,s' \in L(RE)\}$$

"$s\backslash RE$ matches everything matched by RE, with head s cut off."

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

  Suppose we are to match the SAX event sequence $s_1 s_2 s_3$ against RE:

$$s_1 s_2 s_3 \in L(RE)$$

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

  Suppose we are to match the SAX event sequence $s_1 s_2 s_3$ against RE:

$$s_1 s_2 s_3 \in L(RE) \quad \Leftrightarrow \quad s_1 s_2 s_3 \varepsilon \in L(RE)$$

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

  Suppose we are to match the SAX event sequence $s_1 s_2 s_3$ against RE:

$$
\begin{aligned}
s_1 s_2 s_3 \in L(RE) \quad &\Leftrightarrow \quad s_1 s_2 s_3 \varepsilon \in L(RE) \\
&\Leftrightarrow \quad s_2 s_3 \varepsilon \in L(s_1 \backslash RE)
\end{aligned}
$$

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

  Suppose we are to match the SAX event sequence $s_1 s_2 s_3$ against RE:

$$\begin{aligned} s_1 s_2 s_3 \in L(RE) &\Leftrightarrow s_1 s_2 s_3 \varepsilon \in L(RE) \\ &\Leftrightarrow s_2 s_3 \varepsilon \in L(s_1 \backslash RE) \\ &\Leftrightarrow s_3 \varepsilon \in L\left(s_2 \backslash (s_1 \backslash RE)\right) \end{aligned}$$

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

  Suppose we are to match the SAX event sequence $s_1 s_2 s_3$ against RE:

$$
\begin{aligned}
s_1 s_2 s_3 \in L(RE) &\Leftrightarrow s_1 s_2 s_3 \varepsilon \in L(RE) \\
&\Leftrightarrow s_2 s_3 \varepsilon \in L(s_1 \backslash RE) \\
&\Leftrightarrow s_3 \varepsilon \in L\left(s_2 \backslash (s_1 \backslash RE)\right) \\
&\Leftrightarrow \varepsilon \in L\left(s_3 \backslash (s_2 \backslash (s_1 \backslash RE))\right) \quad .
\end{aligned}
$$

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

  Suppose we are to match the SAX event sequence $s_1 s_2 s_3$ against RE:

$$
\begin{aligned}
s_1 s_2 s_3 \in L(RE) &\Leftrightarrow s_1 s_2 s_3 \varepsilon \in L(RE) \\
&\Leftrightarrow s_2 s_3 \varepsilon \in L(s_1 \backslash RE) \\
&\Leftrightarrow s_3 \varepsilon \in L\left(s_2 \backslash (s_1 \backslash RE)\right) \\
&\Leftrightarrow \varepsilon \in L\left(s_3 \backslash (s_2 \backslash (s_1 \backslash RE))\right) \quad .
\end{aligned}
$$

- 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 $\backslash$ to develop a simple **RE matching procedure**.

  Suppose we are to match the SAX event sequence $s_1 s_2 s_3$ against RE:

$$
\begin{aligned}
s_1 s_2 s_3 \in L(RE) \quad &\Leftrightarrow \quad s_1 s_2 s_3 \varepsilon \in L(RE) \\
&\Leftrightarrow \quad s_2 s_3 \varepsilon \in L(s_1 \backslash RE) \\
&\Leftrightarrow \quad s_3 \varepsilon \in L\left(s_2 \backslash (s_1 \backslash RE)\right) \\
&\Leftrightarrow \quad \varepsilon \in L\left(s_3 \backslash (s_2 \backslash (s_1 \backslash RE))\right) \quad .
\end{aligned}
$$

- 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** $L(s \backslash RE)$ 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) $\mathit{nullable}(RE)$ such that

$$\mathit{nullable}(RE) \quad \Leftrightarrow \quad \varepsilon \in L(RE) \ .$$

$$
\begin{aligned}
\mathit{nullable}(\emptyset) &= \mathit{false} \\
\mathit{nullable}(\varepsilon) &= \mathit{true} \\
\mathit{nullable}(\#\texttt{PCDATA}) &= \mathit{false} \\
\mathit{nullable}(t) &= \\
\mathit{nullable}(RE_1, RE_2) &= \\
\mathit{nullable}(RE^{+}) &= \\
\mathit{nullable}(RE^{*}) &= \\
\mathit{nullable}(RE?) &= \\
\mathit{nullable}(RE_1 \mid RE_2) &=
\end{aligned}
$$

Example

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

$$\mathit{nullable}(\#\texttt{PCDATA} \mid \texttt{b}^*)$$

Example

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

$$\mathit{nullable}(\#\texttt{PCDATA} \mid \texttt{b}^*) \quad = \quad \mathit{nullable}(\#\texttt{PCDATA}) \lor \mathit{nullable}(\texttt{b}^*)$$

Example

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

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

Example

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

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

Example

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

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

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

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.

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

Correctness: Case Analysis (I)

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

$$L(s\backslash RE) \quad = \quad L(RE') \ .$$

1. $RE = \emptyset$:

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

Correctness: Case Analysis (II)

2. $RE = \varepsilon$:

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

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

$$
\begin{aligned}
L(\mathit{characters}(\cdot)\backslash\#\texttt{PCDATA}) &= \{s' \mid \mathit{characters}(\cdot)\, s' \in L(\#\texttt{PCDATA})\} \\
&= \{s' \mid \mathit{characters}(\cdot)\, s' \in \{\mathit{characters}(\cdot)\}\} \\
&= \{\varepsilon\} \\
&= L(\varepsilon).
\end{aligned}
$$

Correctness: Case Analysis (III)

$RE = \#\texttt{PCDATA}, s \neq \mathit{characters}(\cdot)$:

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

4. $RE = t$. Analogous to ③.
5. $RE = RE_1, RE_2$, $\mathit{nullable}(RE_1) = \mathit{false}$:

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

Correctness: Case Analysis (IV)

$RE = RE_1, RE_2$, $\mathit{nullable}(RE_1) = \mathit{true}$:

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

Correctness: Case Analysis (V)

6. $RE = RE_1 \mid RE_2$:

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

Correctness: Case Analysis (VI)

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

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

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

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

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 $RE' = \texttt{a}\backslash\,(\texttt{c}\,\backslash\,(\texttt{c}\,\backslash\,(\texttt{b}\,\backslash(\texttt{b,c}^*\texttt{,a?}))))$,
2. then test $\mathit{nullable}(RE')$.

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

$$
\begin{array}{rcl@{\qquad\qquad}rcl}
\varepsilon^* & = & \varepsilon & \varepsilon, RE & = & RE \\
\emptyset^* & = & \varepsilon & \emptyset, RE & = & \emptyset \\
\varepsilon^+ & = & \varepsilon & RE, \varepsilon & = & RE \\
\emptyset^+ & = & \emptyset & RE, \emptyset & = & \emptyset \\
\varepsilon? & = & \varepsilon & \emptyset \mid RE & = & RE \\
\emptyset? & = & \varepsilon & RE \mid \emptyset & = & RE
\end{array}
$$

[19] Actual event sequence:
startElement(b,·), *startElement*(c,·), *startElement*(c,·), *startElement*(a,·).

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()

$S.empty()$;
$RE \leftarrow cm(r)$;
return;

startElement(t, ·)

$RE \leftarrow t\backslash RE$;
$S.push(RE)$;
$RE \leftarrow cm(t)$;
return;

endElement(t)

if $nullable(RE)$ then
  $RE \leftarrow S.pop()$;
else
  ★ FAIL ★;
return;

characters(·)

$RE \leftarrow$
#PCDATA$\backslash RE$;
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()

$S.empty()$;
$RE \leftarrow cm(r)$;
return;

startElement$(t, \cdot)$

$RE \leftarrow t \backslash RE$;
$S.push(RE)$;
$RE \leftarrow cm(t)$;
return;

endElement(t)

if $nullable(RE)$ then
  $RE \leftarrow S.pop()$;
else
  ★ FAIL ★;
return;

characters$(\cdot)$

$RE \leftarrow$ #PCDATA$\backslash RE$;
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.

Streaming Validation Beyond DTD

Question for next time: what about streaming validation w.r.t XML Schema?