Efficient XML Processing with Tree Automata

Alexandru Berlea

Technische Universität München

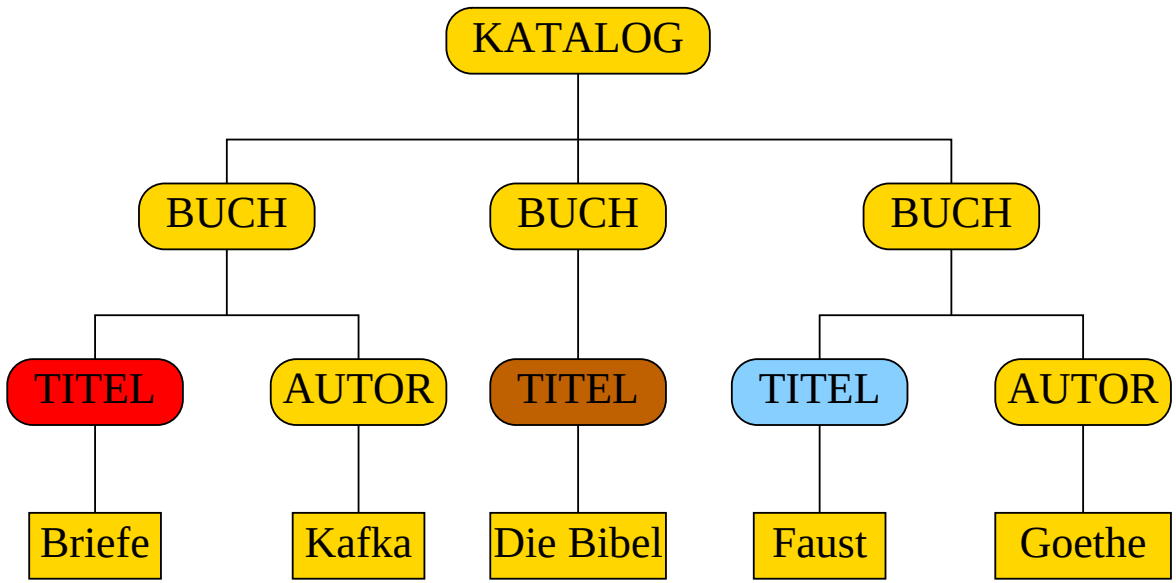
Überblick

Teil I : Suche in XML-Dokumenten

Teil II : Transformation von XML-Dokumenten

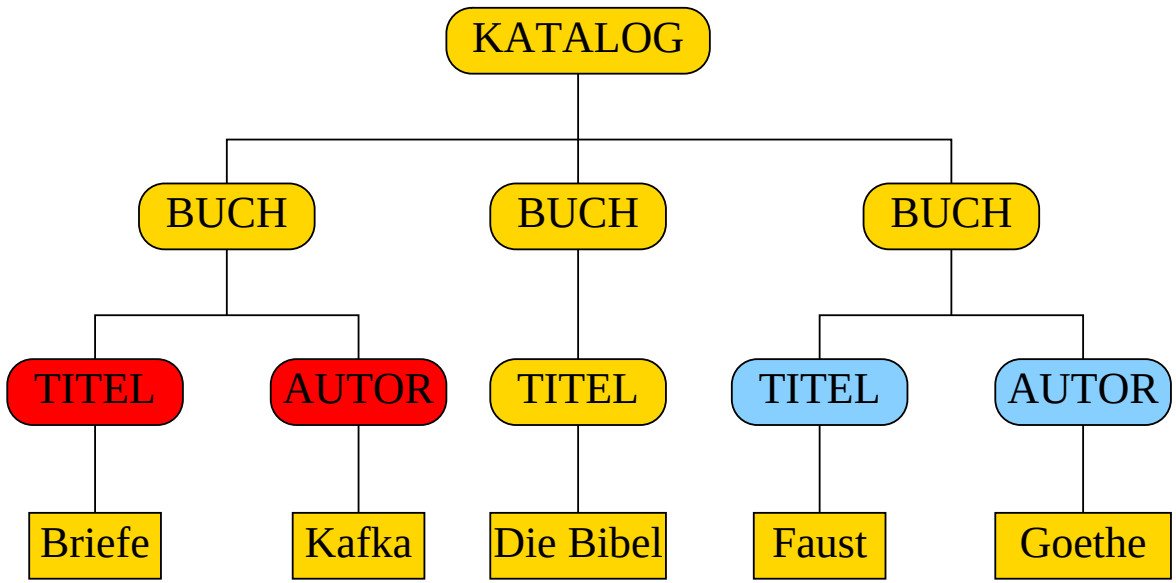
XML-Anfrage: *Titel*

```
<KATALOG>
  <BUCH>
    <TITEL>Briefe</TITEL>
    <AUTOR>Kafka</AUTOR>
  </BUCH>
  <BUCH>
    <TITEL>Die Bibel</TITEL>
  </BUCH>
  <BUCH>
    <TITEL>Faust</TITEL>
    <AUTOR>Goethe</AUTOR>
  </BUCH>
</KATALOG>
```



Zweistellige XML-Anfrage: *Titel zusammen mit Autoren*

```
<KATALOG>
  <BUCH>
    <TITEL>Briefe</TITEL>
    <AUTOR>Kafka</AUTOR>
  </BUCH>
  <BUCH>
    <TITEL>Die Bibel</TITEL>
  </BUCH>
  <BUCH>
    <TITEL>Faust</TITEL>
    <AUTOR>Goethe</AUTOR>
  </BUCH>
</KATALOG>
```



Eine XML-Anfragesprache: Fxgrep

- basiert auf Dokumentgrammatiken (DTD, XML Schema)
  - ▷ einstellige Anfragen [Neumann und Seidl 1998]
  - ▷ zweistellige Anfragen [Dissertation]

Eine XML-Anfragesprache: Fxgrep

- basiert auf Dokumentgrammatiken (DTD, XML Schema)
  - ▷ einstellige Anfragen [Neumann und Seidl 1998]
  - ▷ zweistellige Anfragen [Dissertation]
- Syntax ähnlich wie XPath: `//BUCH[AUTOR]/TITEL`
  - Fxgrep $\bigcap$ XPath = CoreXPath

Eine XML-Anfragesprache: Fxgrep

- basiert auf Dokumentgrammatiken (DTD, XML Schema)
  - ▷ einstellige Anfragen [Neumann und Seidl 1998]
  - ▷ zweistellige Anfragen [Dissertation]
- Syntax ähnlich wie XPath: `//BUCH[AUTOR]/TITEL`
  - Fxgrep $\bigcap$ XPath = CoreXPath
  - Fxgrep $\setminus$ XPath :

    zweistellige Anfragen `//BUCH[%AUTOR]/TITEL`

Eine XML-Anfragesprache: Fxgrep

- basiert auf Dokumentgrammatiken (DTD, XML Schema)
  - ▷ einstellige Anfragen [Neumann und Seidl 1998]
  - ▷ zweistellige Anfragen [Dissertation]
- Syntax ähnlich wie XPath: `//BUCH[AUTOR]/TITEL`
  - Fxgrep $\bigcap$ XPath = CoreXPath
  - Fxgrep $\setminus$ XPath :

    zweistellige Anfragen `//BUCH[%AUTOR]/TITEL`

    horizontale Eigenschaften

    `//BUCH[TITEL (AUTOR|REDAKTEUR)+]/PREIS`

Eine XML-Anfragesprache: Fxgrep

- basiert auf Dokumentgrammatiken (DTD, XML Schema)
  - ▷ einstellige Anfragen [Neumann und Seidl 1998]
  - ▷ zweistellige Anfragen [Dissertation]
- Syntax ähnlich wie XPath: `//BUCH[AUTOR]/TITEL`
  - Fxgrep $\bigcap$ XPath = CoreXPath
  - Fxgrep $\setminus$ XPath :

    zweistellige Anfragen `//BUCH[%AUTOR]/TITEL`

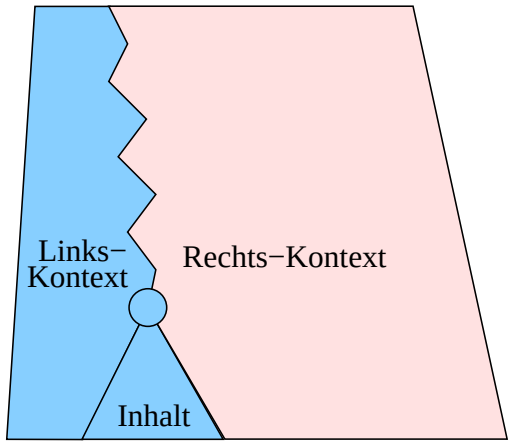
    horizontale Eigenschaften

    `//BUCH[TITEL (AUTOR|REDAKTEUR)+]/PREIS`

    vertikale Eigenschaften `(SEKTION/)+ TITEL`

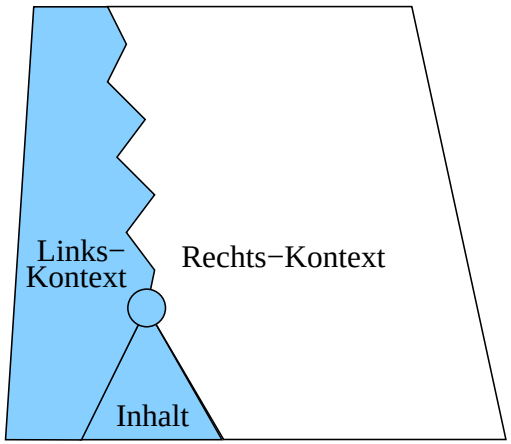
Auswertung einstelliger Anfragen: Baum-Automaten

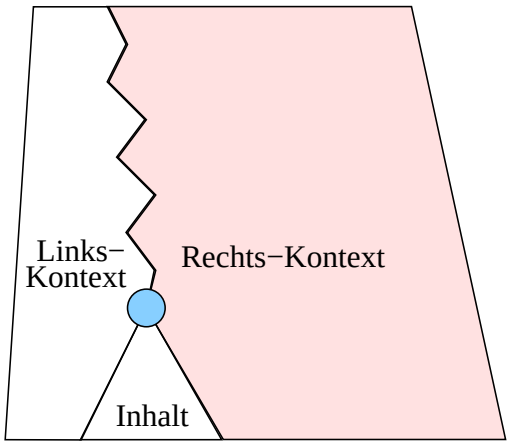
1. Lösungsansatz [Neumann und Seidl 1998]: Zwei Tiefendurchläufe



Auswertung einstelliger Anfragen

1. Lösungsansatz [Neumann und Seidl 1998]: Zwei Tiefendurchläufe





Links-Rechts

- speichere an jedem Knoten Informationen über linken Kontext und Inhalt

Rechts-Links

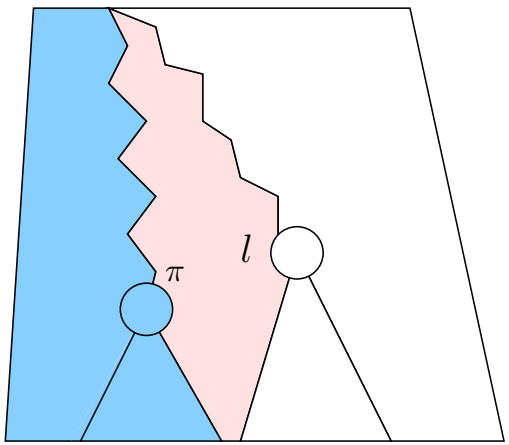
- propagiere Information über rechten Kontext
- signalisiere Treffer

$\Longrightarrow$ Zwischendatenstruktur im Speicher für das gesamte Dokument

$\Longrightarrow$ nicht einsetzbar für sehr große Dokumente

Auswertung einstelliger Anfragen

2. Lösungsansatz [Dissertation]: Ein einziger Tiefendurchlauf



Im Allgemeinen ist nur ein Teil des Rechts-Kontexts relevant $\Longrightarrow$

- merke potentielle Treffer
- erkenne relevanten Rechtskontext eines Treffers

Auswertung einstelliger Anfragen

Vorteile der Auswertung in einem Durchlauf [Dissertation]:

- Baum muss nicht im Speicher aufgebaut werden
  $\Longrightarrow$ Sehr große Dokumente können durchsucht werden
- "On-the-fly" Vearbeitung während des Einlesens
- Potentielle Treffer werden zum frühestmöglichen Zeitpunkt bestätigt bzw. verworfen.

Auswertung zweistelliger Anfragen

- möglich in zwei Durchläufen [$\longrightarrow$ Dissertation]
- Zeit-Komplexität:
  - im schlimmsten Fall $\mathcal{O}(n^2)$
  - meist ähnlich wie für einstellige Anfragen $\Rightarrow \mathcal{O}(n)$

Teil II

XML-Transformationssprachen

... verwenden XML-Anfragensprachen

- XSLT, XQuery verwenden XPath
- Fxt verwendet Fxgrep

Teil II

XML-Transformationssprachen

... verwenden XML-Anfragensprachen

- XSLT, XQuery [W3C-Empfehlungen] verwenden XPath
- Fxt [Dissertation-Empfehlung] verwendet Fxgrep
  - intuitiv $\Leftarrow$ regel-basiert
  - mächtig $\Leftarrow$ SML-Code Einbettung
  - effizient $\Leftarrow$ Fxgrep, zweistellige Anfragen

Regel-basierte XML-Transformationen

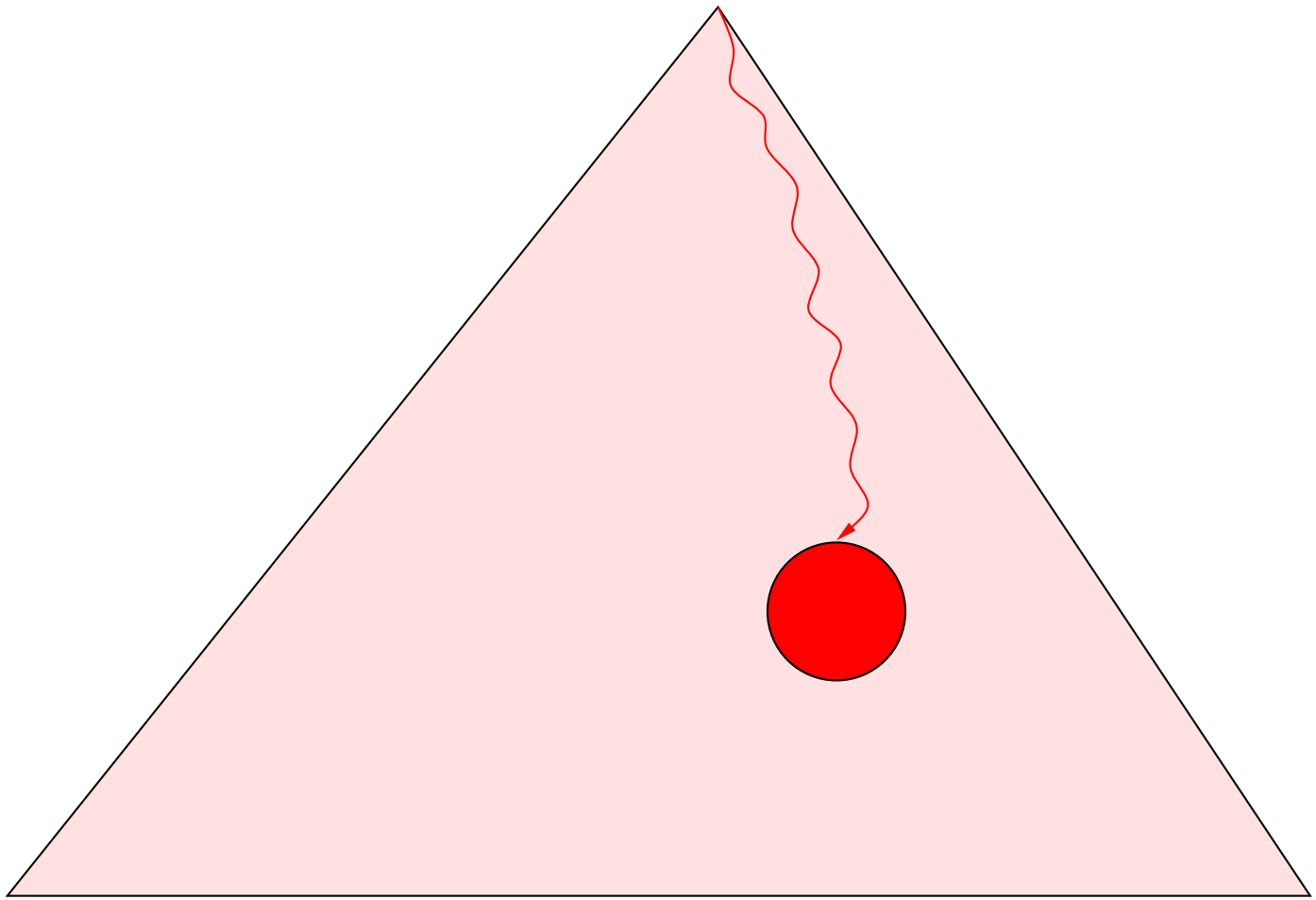
Spezifikation = Menge von Regeln

Regel $\approx$ Fall der Transformationsfunktion

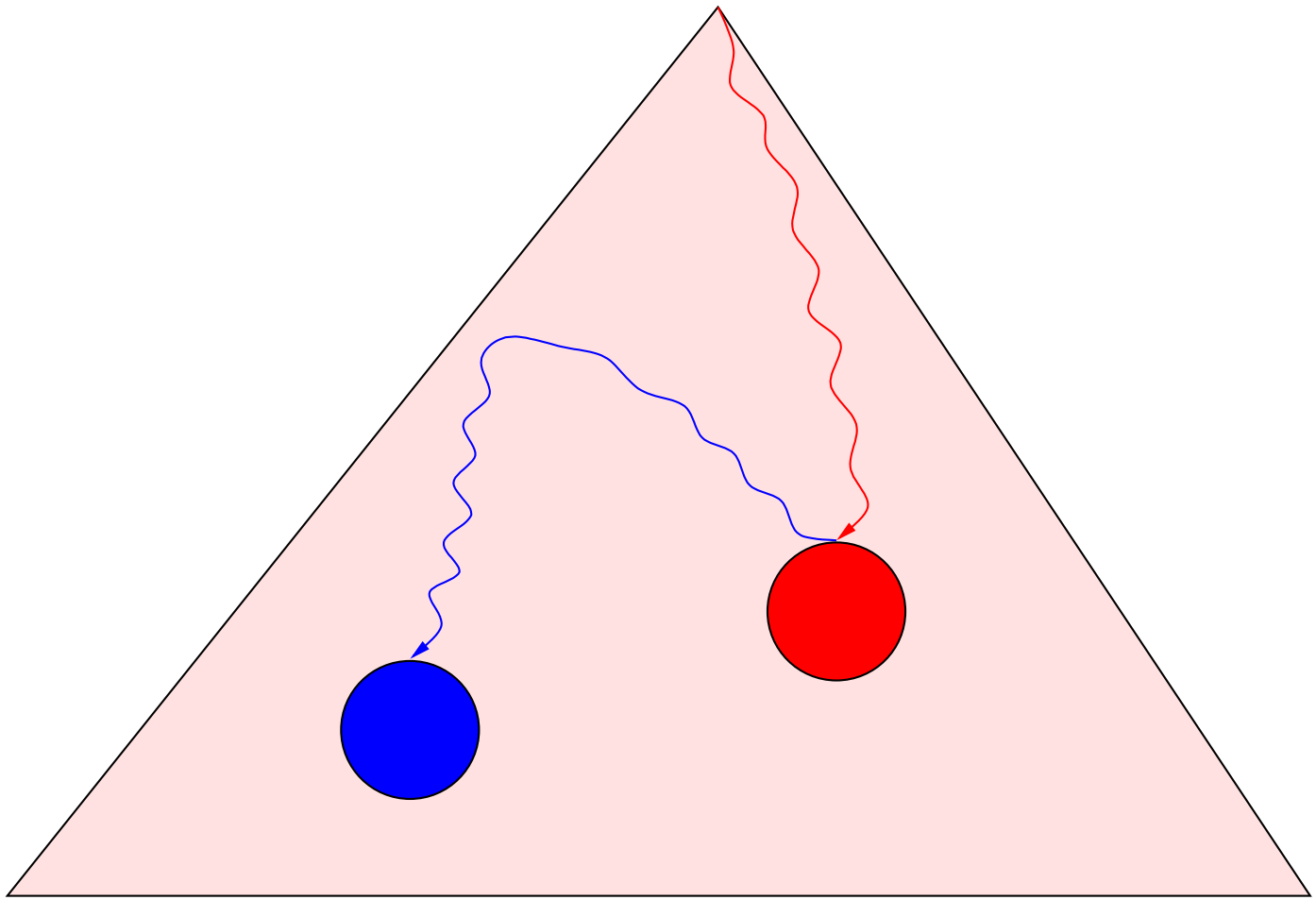
- Match-Muster zur Fallunterscheidung
- Inhalt, auf den ein Knoten abgebildet werden soll
  - textueller XML-Inhalt
  - Einbezug von Knoten aus Eingabe $\Leftarrow$ Select-Muster

```
<xsl:template match="//BUCH[AUTOR]/TITEL]">
  <TR>
    <TD><xsl:copy-of select="."/></TD>
    <TD><xsl:copy-of select="../AUTOR"/></TD>
  </TR>
</xsl:template>
```

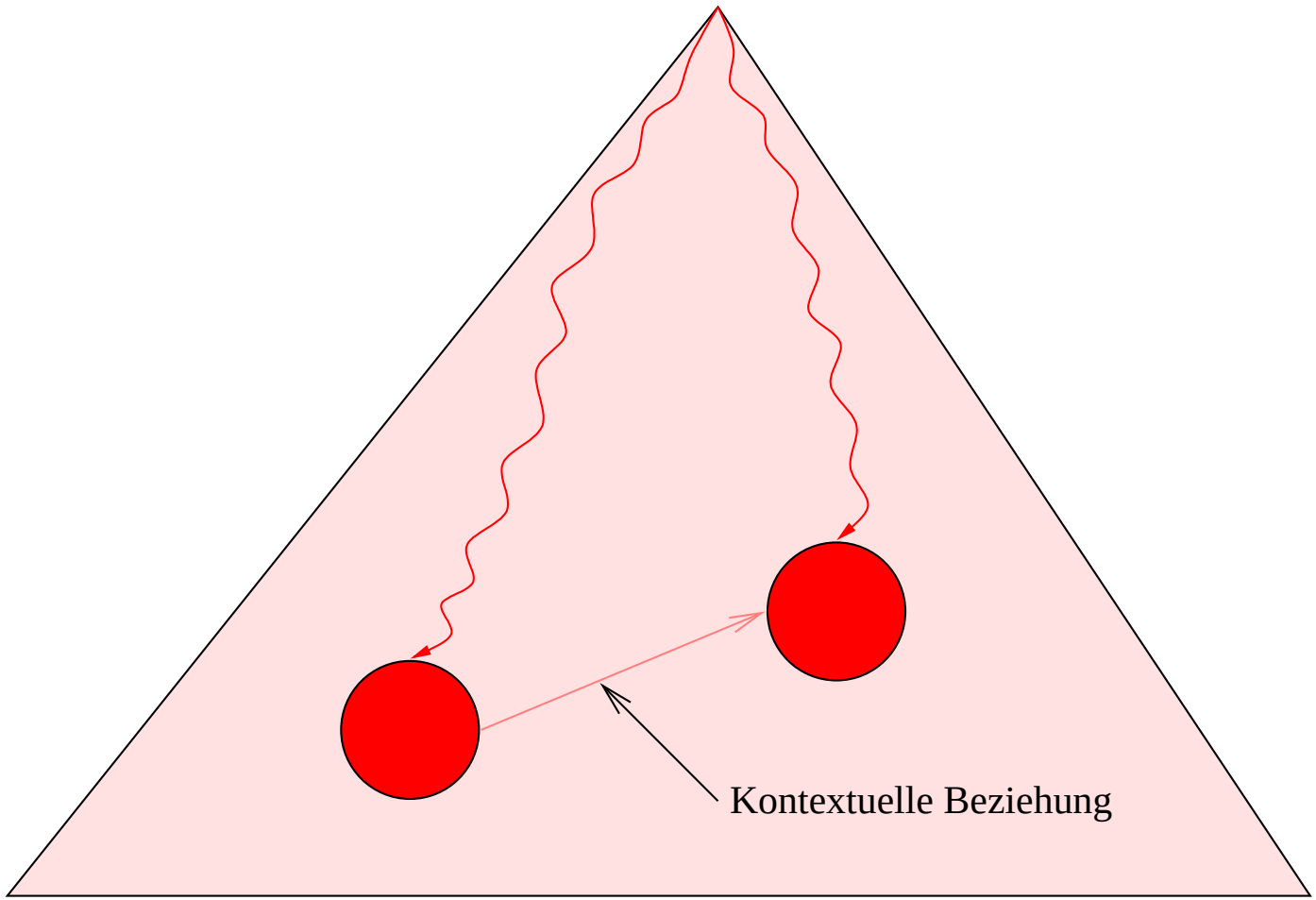
Match-Muster



Match-Muster + Select-Muster



Idee: Nutze zweistellige Match-Muster



Vorteile zweistelliger Match-Muster

XSLT:

```
<xsl:template match="//BUCH[AUTOR]/TITEL]">
  <TR>
    <TD><xsl:copy-of select="."/></TD>
    <TD><xsl:copy-of select="../AUTOR"/></TD>
  </TR>
</xsl:template>
```

Fxt:

```
<fxt:pat>//BUCH[%AUTOR]/TITEL</fxt:pat>
  <TR>
    <TD><fxt:copyContent/></TD>
    <TD><fxt:copyContent select="1"/></TD>
  </TR>
```

→ Expressivität Selektion von Knoten aus dem Kontext ohne explizite Navigation

→ Effizienz Kein redundanter Durchlauf zur Auswertung von Select-Mustern

Ausblick

- Zweistellige Anfragen zur Auswertung von XQuery
- Auswertung k-stelliger Anfragen
- “On-the-fly” Auswertung 2-stelliger Anfragen
- “On-the-fly” XML-Transformationen

XML: Dokumentenbeschreibungssprache

- Für hierarchisch strukturierte Dokumente
- Hauptsächlich benutzt als
  1. Speicherformat in Dokumentkollektionen
     $\Longrightarrow$ Verschiedene Darstellungen durch Transformationen
     $\Longrightarrow$ vereinfachte Konsistenzerhaltung
  2. Austauschformat
     $\Longrightarrow$ Transformation von und zur internen Repräsentation
     $\Longrightarrow$ erhöhte Interoperabilität
- Transformation ist inhärent zu XML-Anwendungen
- Suche ist inhährent zu XML-Anwendungen

Eine Dokumentgrammatik

- DTD-Schreibweise

  `sample.dtd`:

```
<!ELEMENT BOOK (TITLE, SUBTITLE?, CHAPTER+, APPENDIX?)>
<!ELEMENT CHAPTER (TITLE, (CHAPTER|PAR)+)>
<!ELEMENT APPENDIX (CHAPTER+)>
```

  Start-Element:

```
<!DOCTYPE BOOK SYSTEM "sample.dtd">
```

- Grammatik-Schreibweise:

x_{book}	$\rightarrow$	<BOOK>	$x_{title},\ x^{?}_{subtitle},\ x^{+}_{chapter},\ x^{?}_{appendix}$	</BOOK>
$x_{chapter}$	$\rightarrow$	<CHAPTER>	$x_{title},\ (x_{chapter}\|x_{par})^{+}$	</CHAPTER>
$x_{appendix}$	$\rightarrow$	<APPENDIX>	$x^{+}_{chapter}$	</APPENDIX>

  Start-Element: x_{book}

Grammatiken spezifizieren mögliche Ableitungen

Rules:

$x_\top \rightarrow$ `<a>`$x_\top^*$`</a>`

$x_\top \rightarrow$ `<b>`$x_\top^*$`</b>`

$x_\top \rightarrow$ `<c>`$x_\top^*$`</c>`

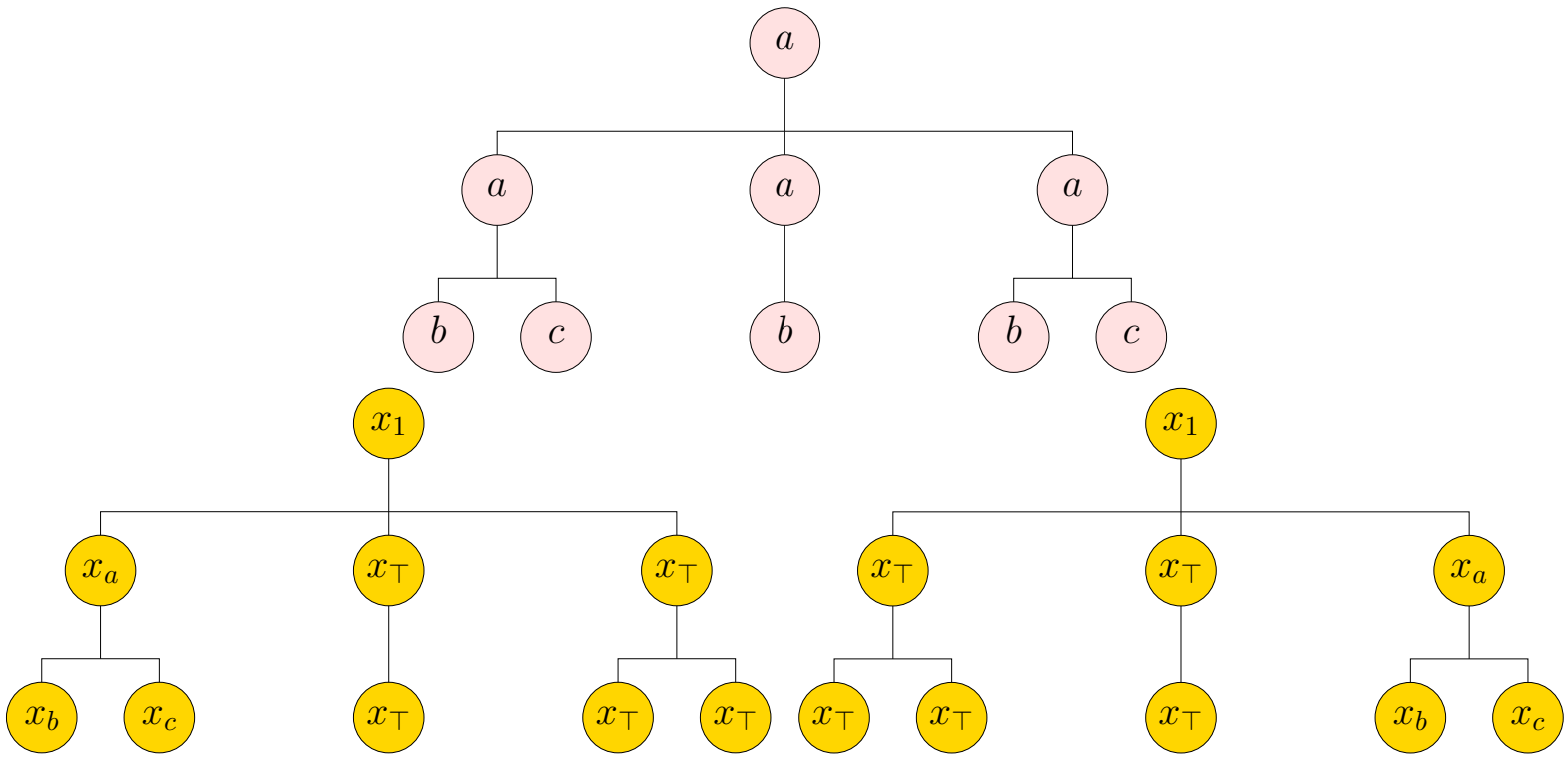
$x_1 \rightarrow$ `<a>`$x_\top^*, (x_1 | x_a), x_\top^*$`</a>`

$x_a \rightarrow$ `<a>`x_b, x_c`</a>`

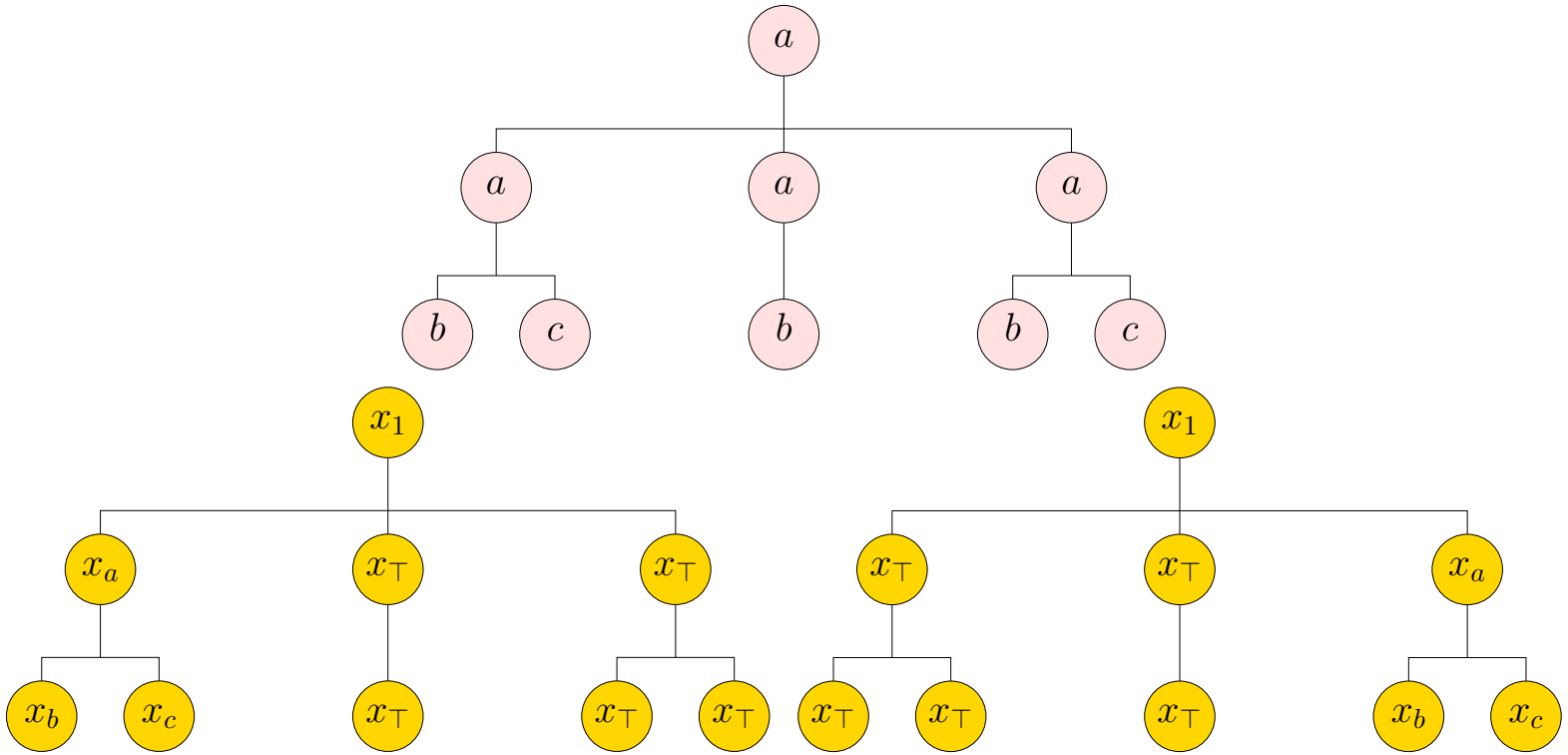
$x_b \rightarrow$ `<b>`$x_\top^*$`</b>`

$x_c \rightarrow$ `<c>`$x_\top^*$`</c>`

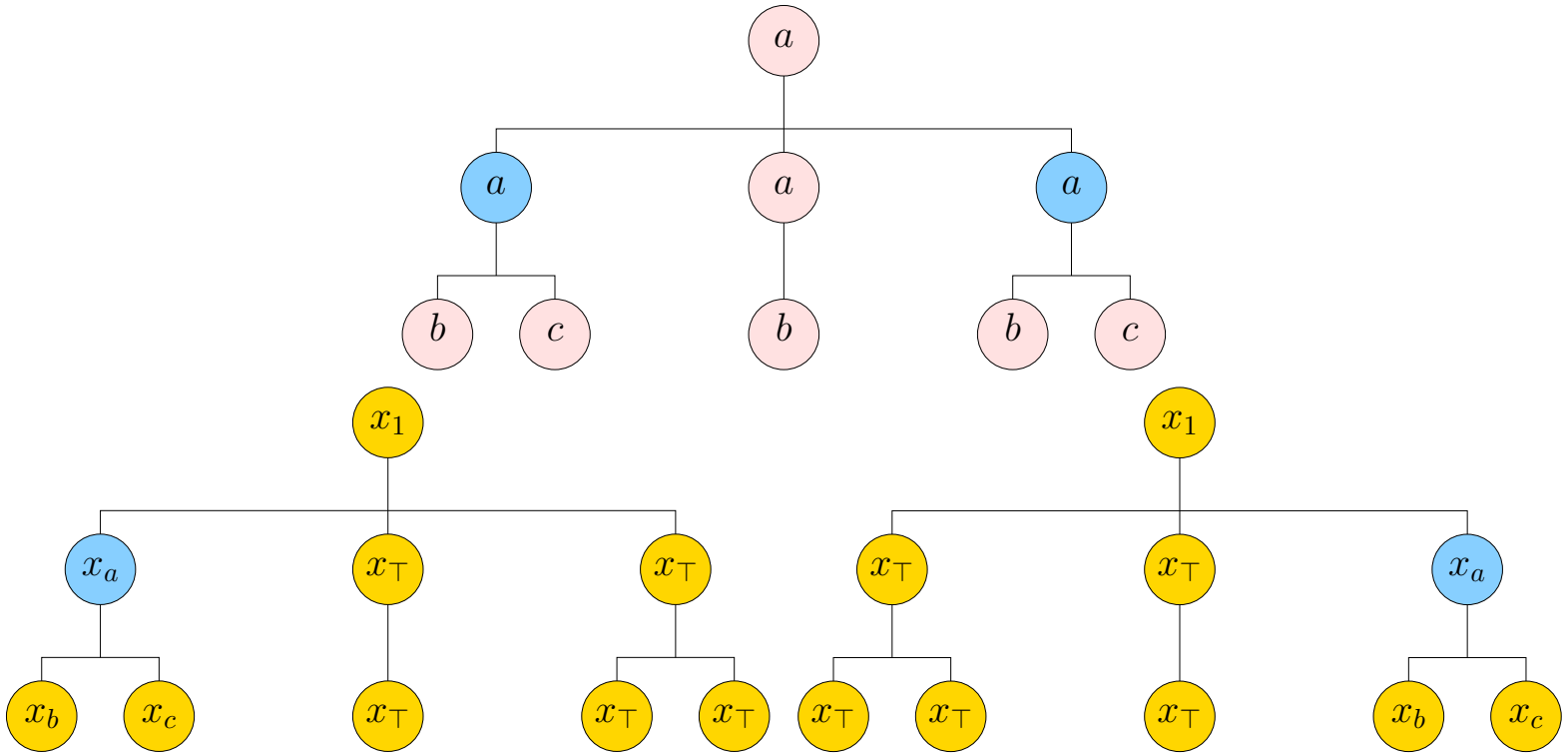
Start: x_1



Nicht-Terminale spezifizieren Anfragen

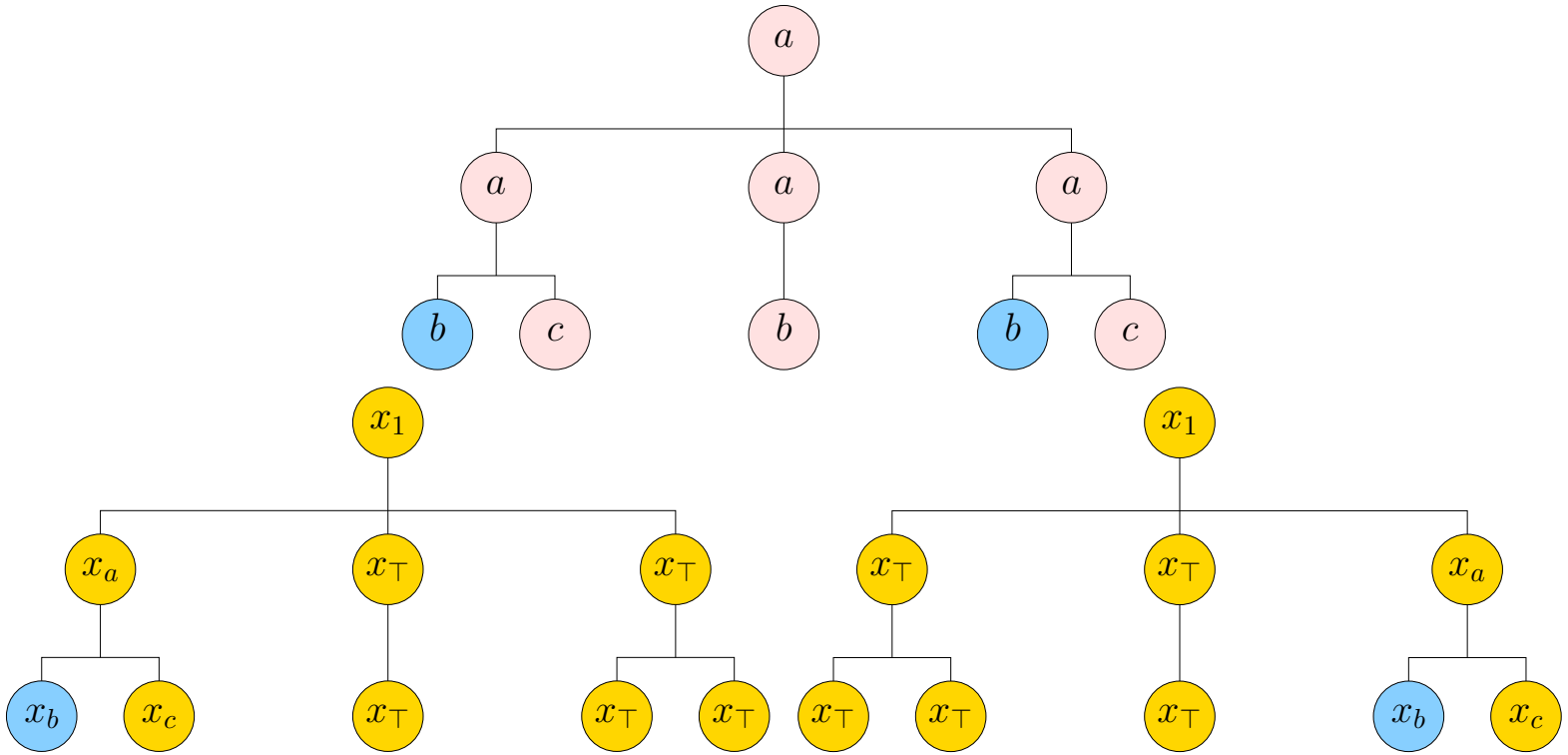


Nicht-Terminale spezifizieren Anfragen



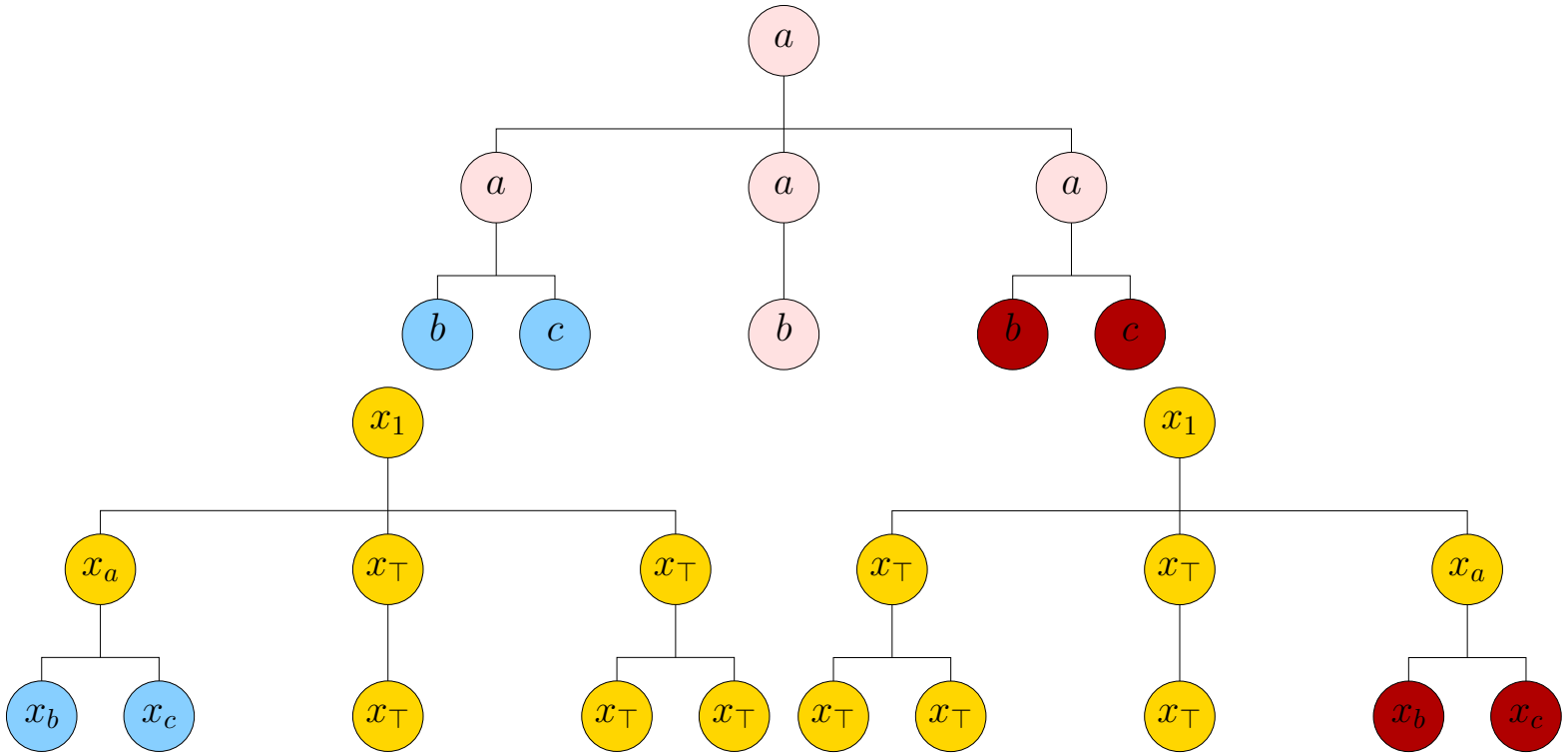
- (G, x_a)

Nicht-Terminale spezifizieren Anfragen



- (G, x_a)
- (G, x_b)

Nicht-Terminale spezifizieren Anfragen

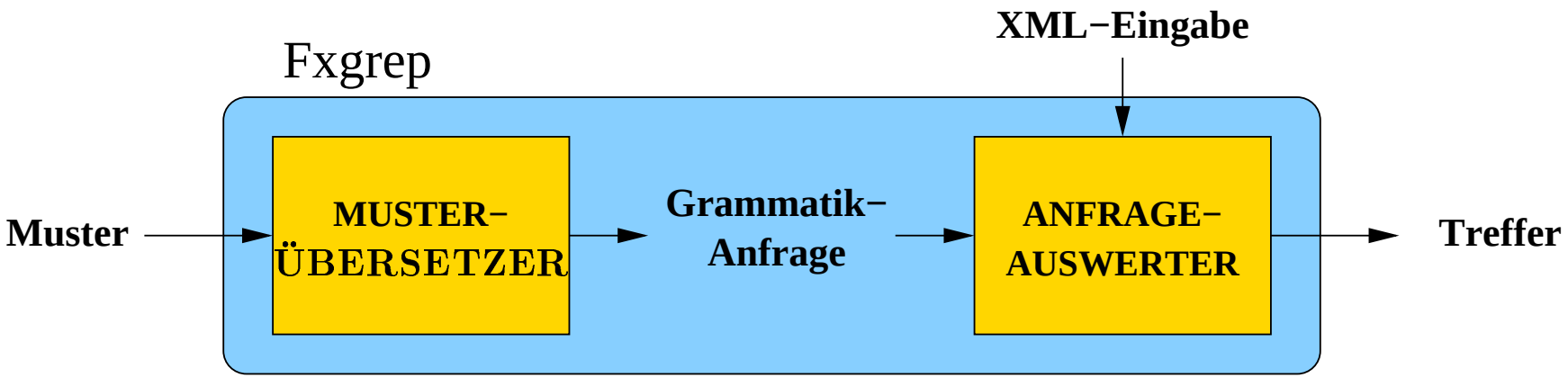


- (G, x_a)
- (G, x_b)
- $(G, (x_b, x_c))$

Spezifikation von XML-Anfragen

Dokumentgrammatiken (DTD, XMLSchema)

- können XML-Anfragen spezifizieren:
  - ▷ einstellige Anfragen [Neumann und Seidl 1998]
  - ▷ mehrstellige Anfragen [Dissertation]
- sind ausdrucksstark... aber langatmig und nicht intuitiv
  - ⟶ spezifiziere Anfragen mit einer intuitiveren Muster-Sprache;
  - ⟶ übersetze Muster in Grammatik-Anfragen.



Fxgrep-Muster

Syntax ähnlich wie XPath

- `/book//section[subsection]`

Fxgrep-Muster

Syntax ähnlich wie XPath

- `/book//section[subsection]`

Verallgemeinerung regulärer Ausdrücke auf Bäumen

- Reguläre Pfade
  - `(section/)+title`

Fxgrep-Muster

Syntax ähnlich wie XPath

- ▷ `/book//section[subsection]`

Verallgemeinerung regulärer Ausdrücke auf Bäumen

- Reguläre Pfade
  - ▷ `(section/)+title`
- Reguläre Strukturbedingungen
  - ▷ `//section[title image+]/title`

Fxgrep-Muster

- Reguläre Kontextbedingungen
  - ▷ `book[section+ # reference+]/section`

Fxgrep-Muster

- Reguläre Kontextbedingungen
  - ▷ `book[section+ # reference+]/section`
  - ▷ `//section[^#]/subsection`
  - ▷ `//section[#$]/subsection`
- Zweistellige Anfragen
  - ▷ `//buch[%autor]/titel`

Fxgrep-Muster

- Reguläre Kontextbedingungen
  - ▷ `book[section+ # reference+]/section`
  - ▷ `//section[^#]/subsection`
  - ▷ `//section[#$]/subsection`
- Zweistellige Anfragen
  - ▷ `//buch[%autor]/titel`
  - ▷ `//buch[(autor/%"escu$")]/titel`

Ereignisgesteuerter Tiefendurchlauf

- XML-Dokument:

```
<a>
  <b>
    <c></c>
  </b>
  <b>
    <c></c>
  </b>
</a>
```

- XML-Ereignisstrom:

```
<a><b><c></c></b><b><c></c></b></a>
```

Ereignisgesteuerter Tiefendurchlauf

```
<a><b><c></c></b><b><c></c></b></a>
```

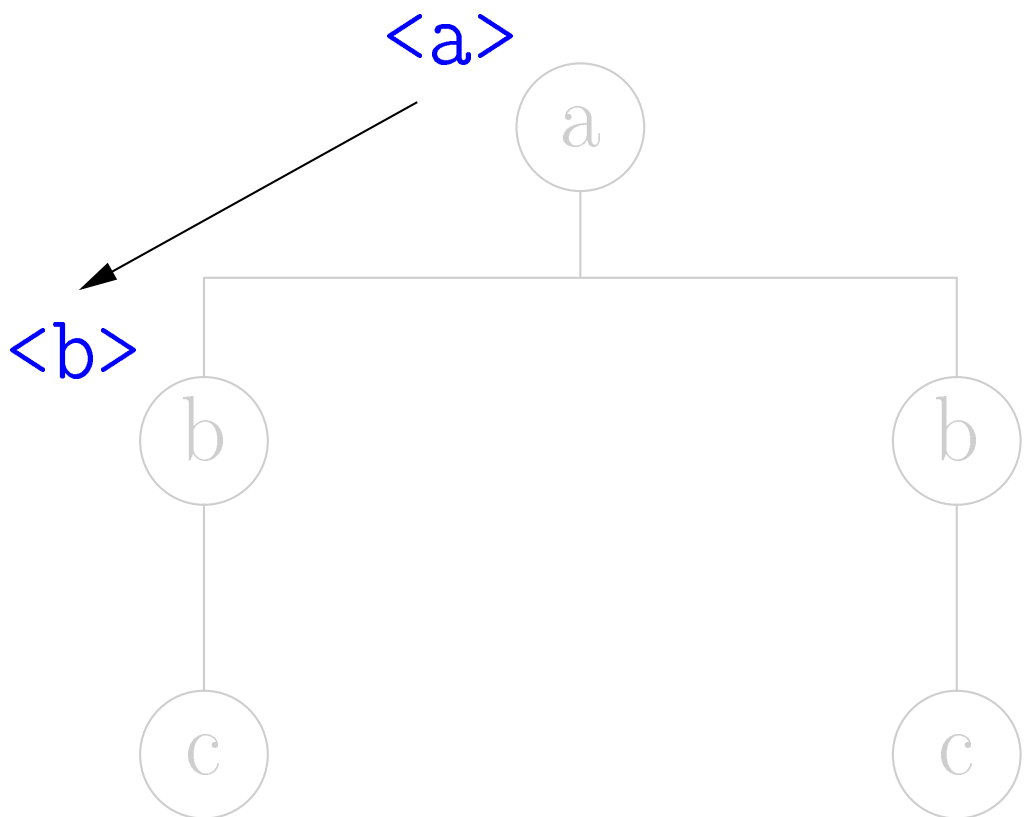
Ereignisgesteuerter Tiefendurchlauf

<a><c></c><c></c>

<a>

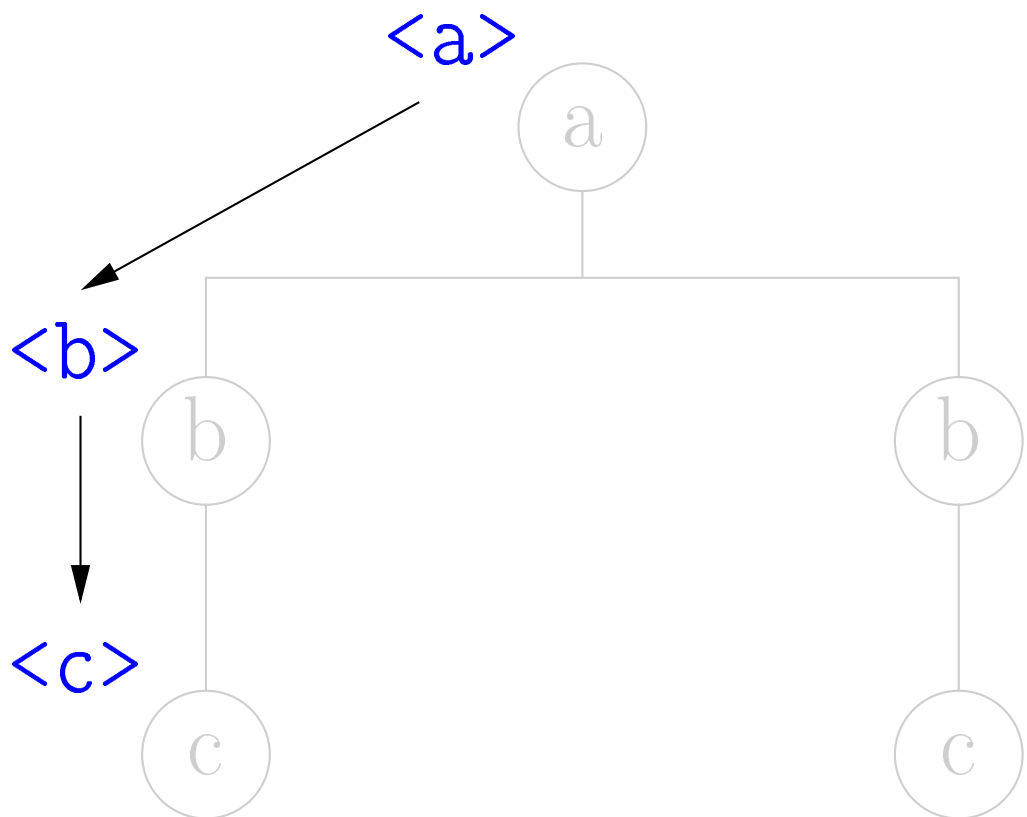
Ereignisgesteuerter Tiefendurchlauf

<a><c></c><c></c>



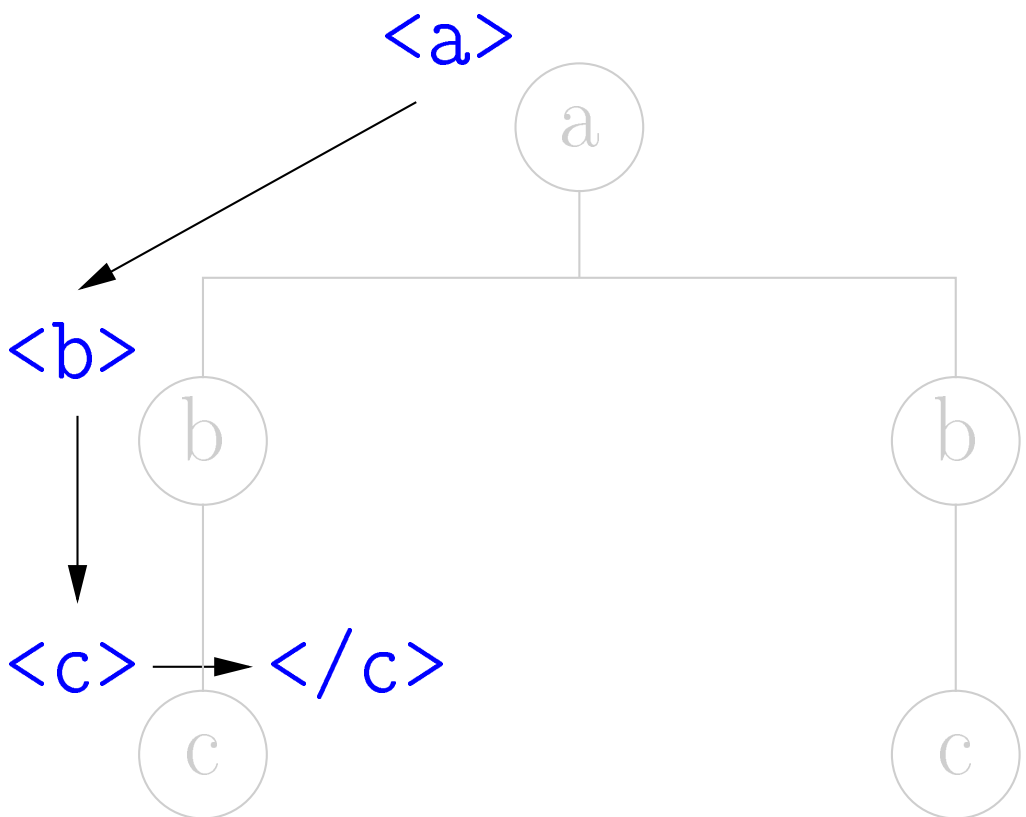
Ereignisgesteuerter Tiefendurchlauf

<a><c></c><c></c>



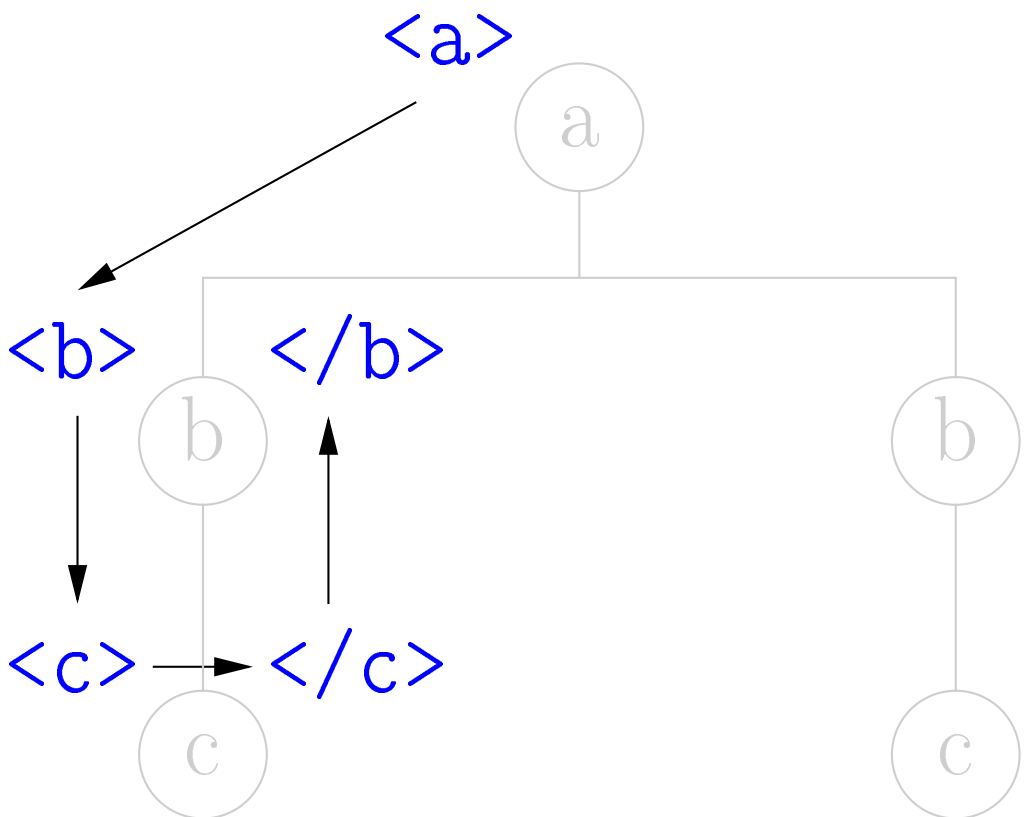
Ereignisgesteuerter Tiefendurchlauf

<a><c></c><c></c>



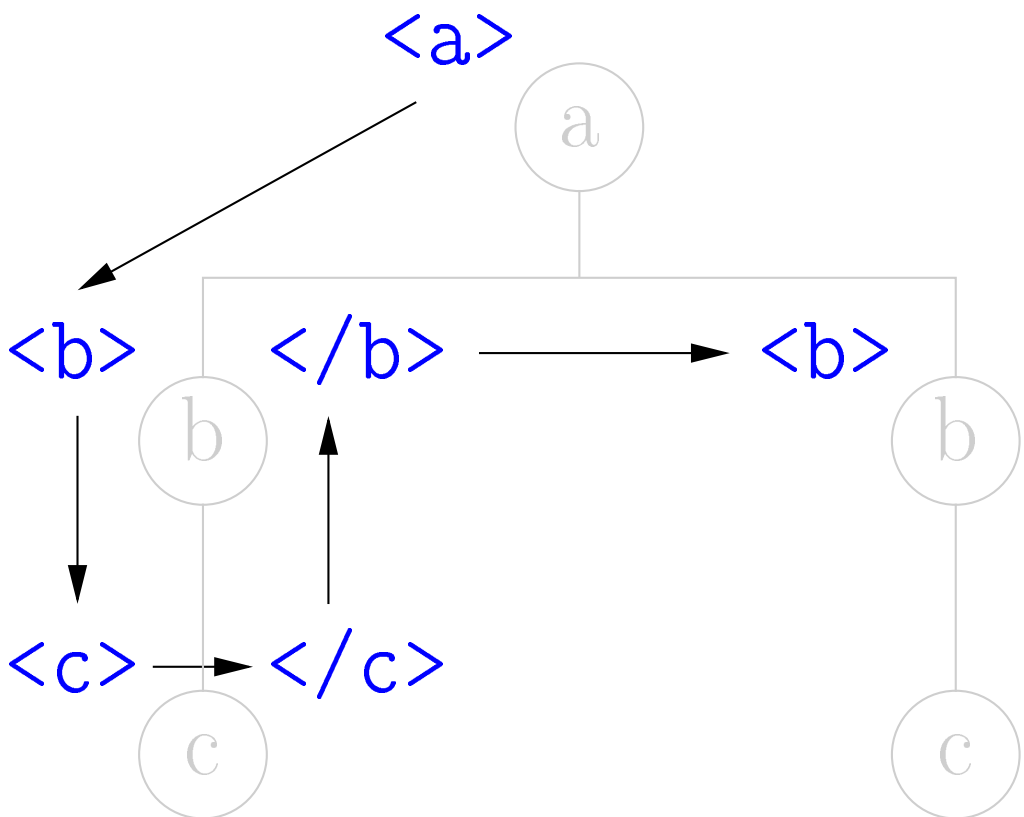
Ereignisgesteuerter Tiefendurchlauf

<a><c></c><c></c>



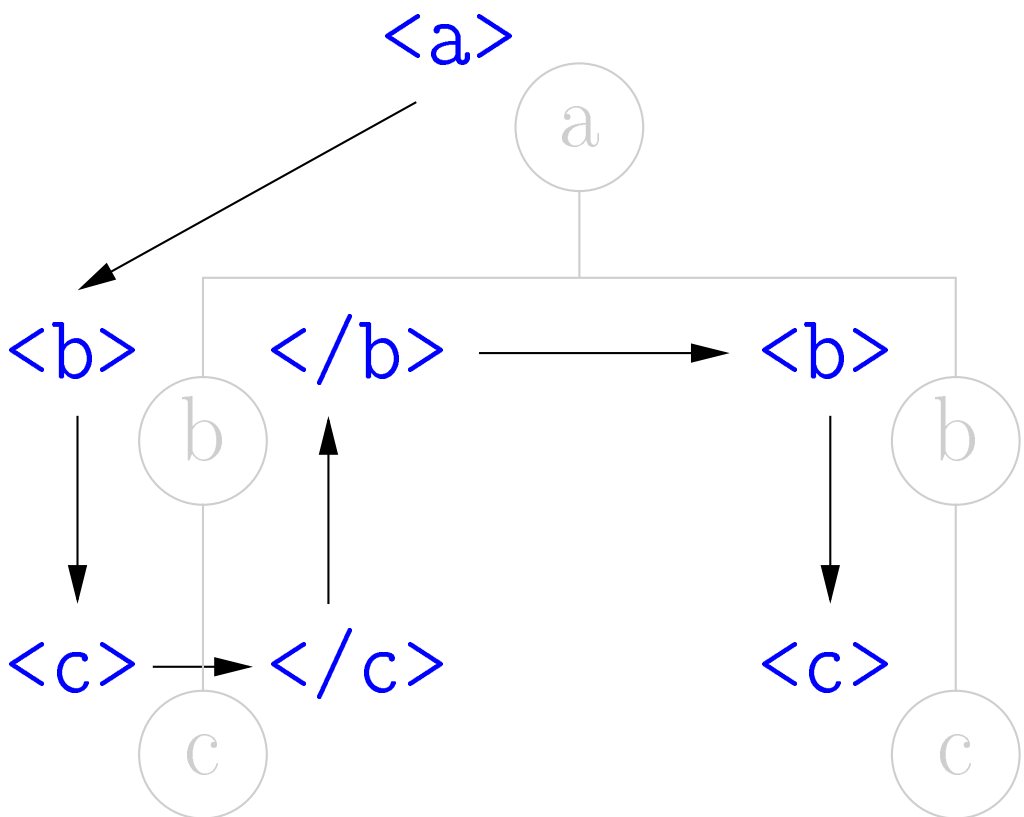
Ereignisgesteuerter Tiefendurchlauf

<a><c></c><c></c>



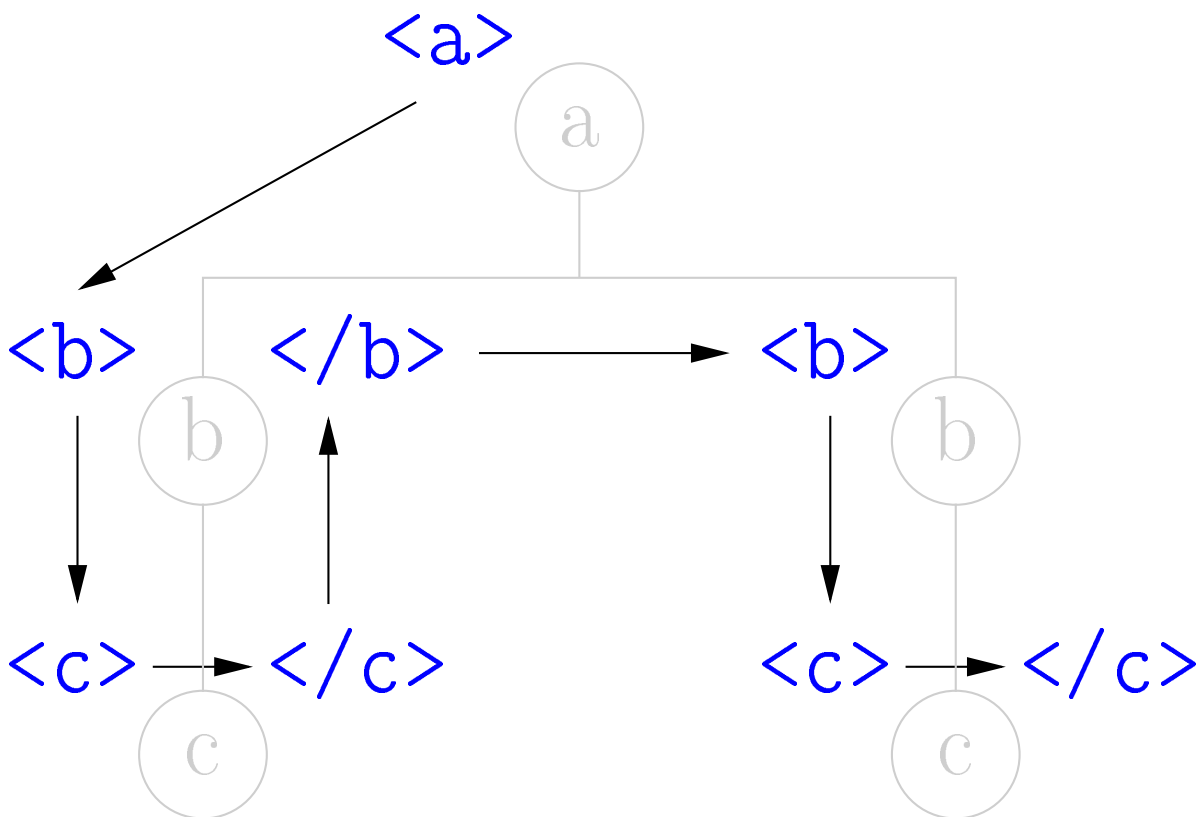
Ereignisgesteuerter Tiefendurchlauf

<a><c></c><c></c>



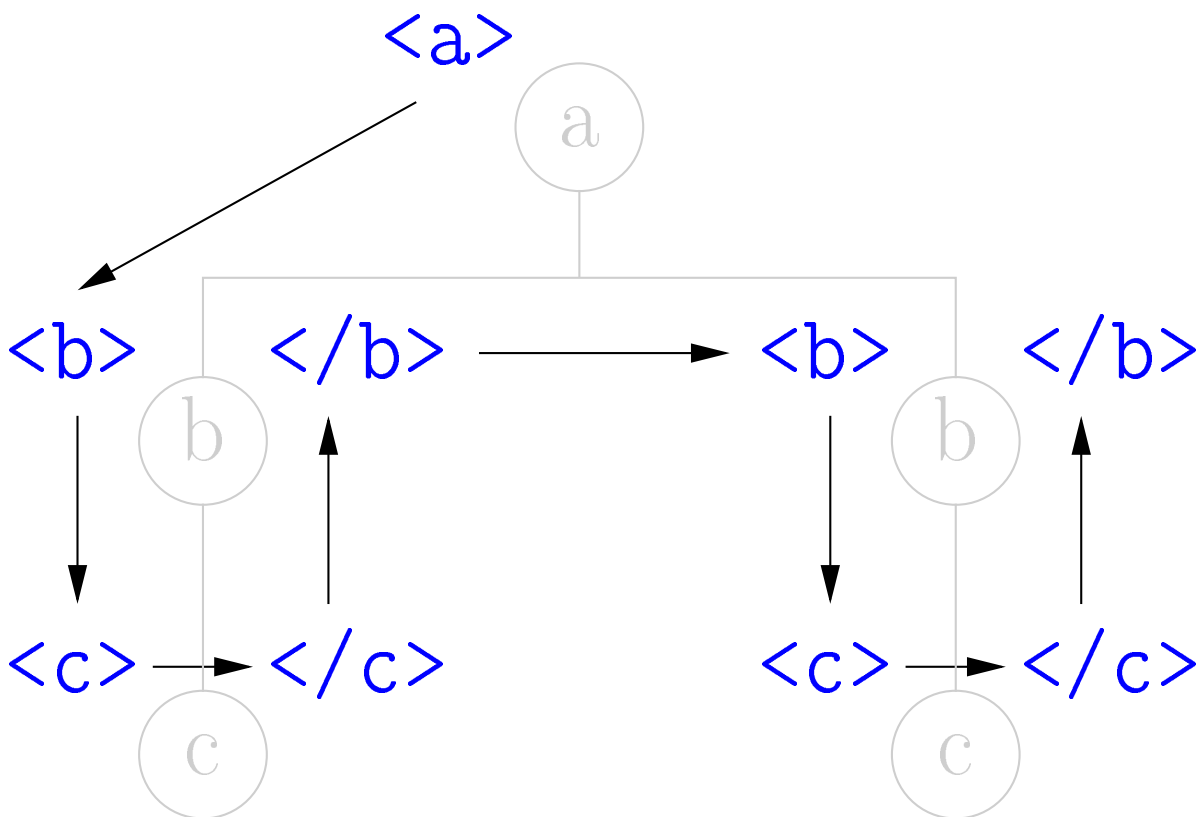
Ereignisgesteuerter Tiefendurchlauf

<a><c></c><c></c>



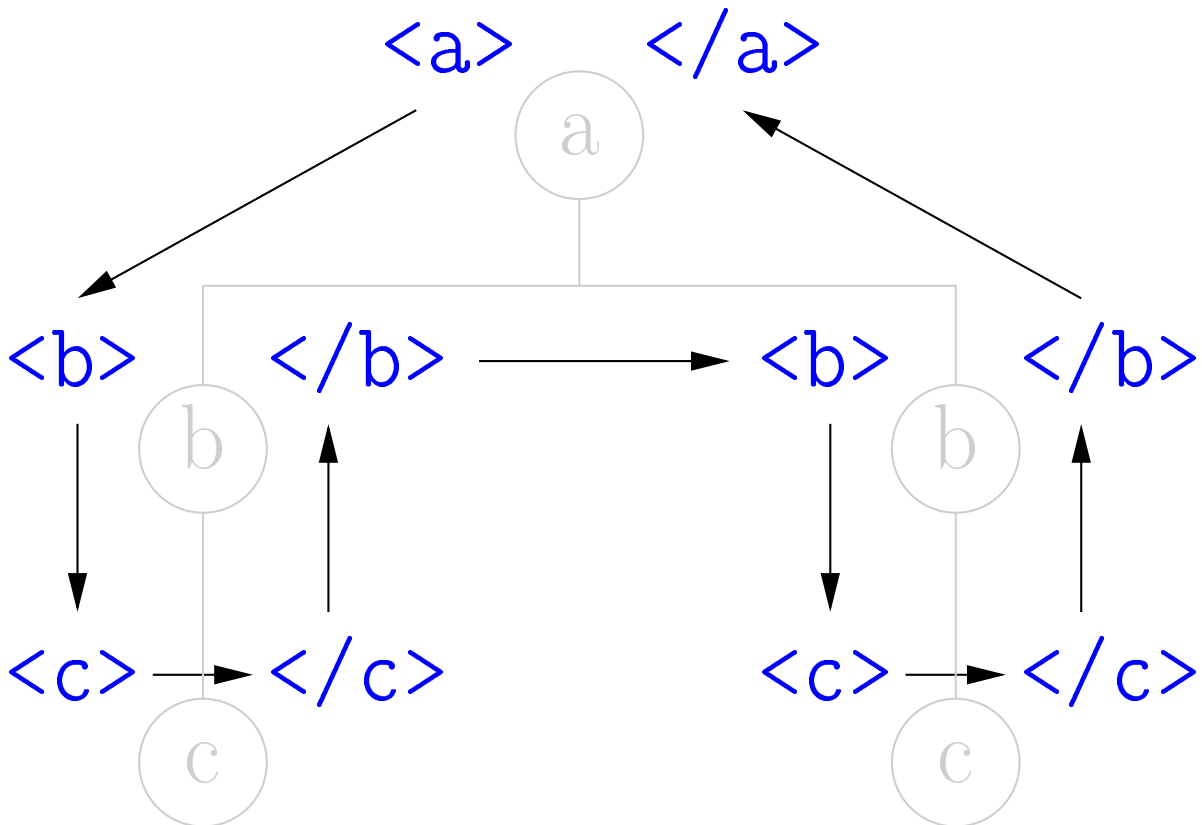
Ereignisgesteuerter Tiefendurchlauf

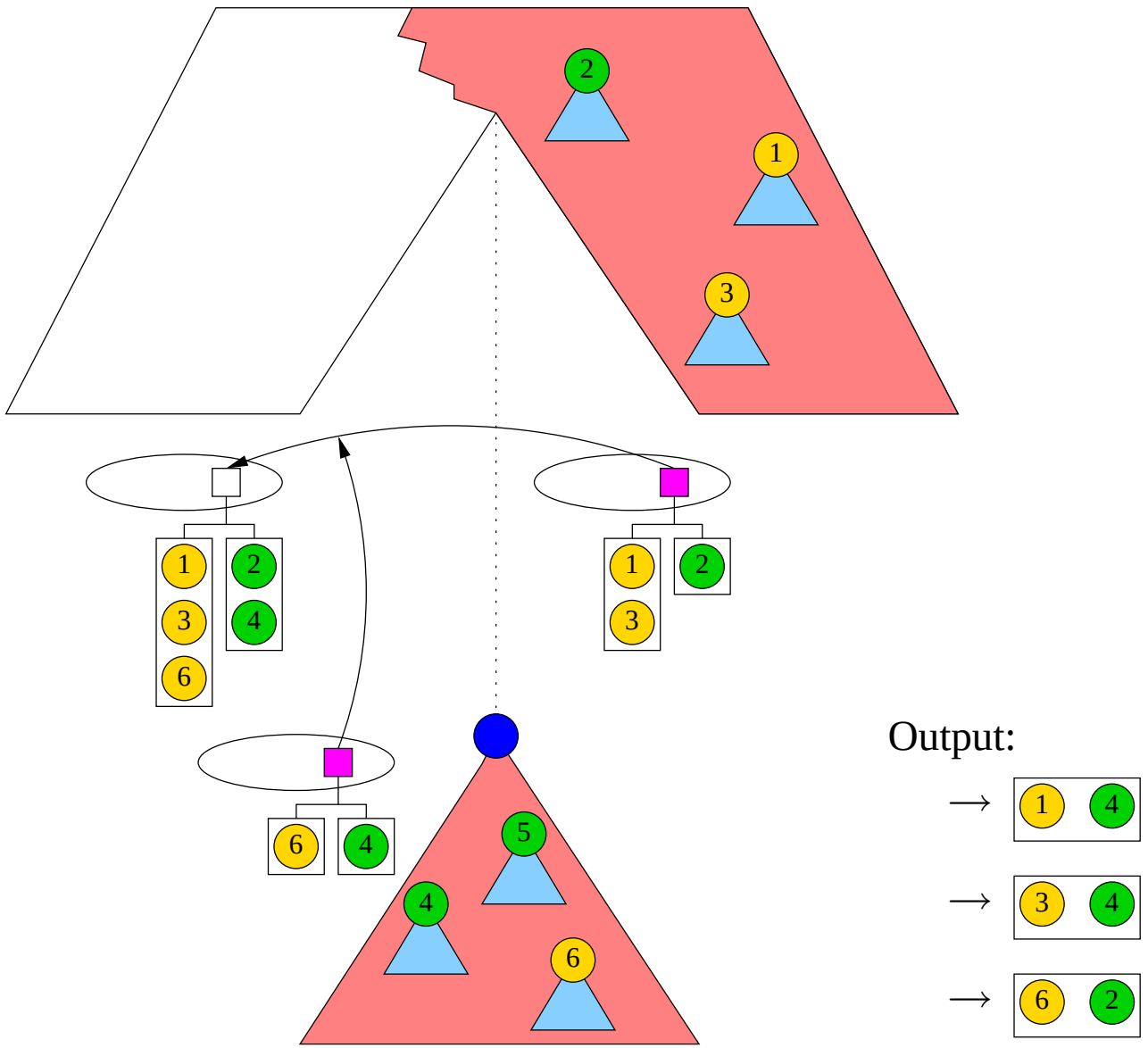
<a><c></c><c></c>



Ereignisgesteuerter Tiefendurchlauf

<a><c></c><c></c>




2
1
3
1
2
3
4
6
1
2
3
6
4
5
4
6
Output:
1
4
3
4
6
2

Umwandlung von Select-Mustern

Eingabebeispiel:

```
<program>
  <declarations>
    <warning>Re-declared variable voila</warning>
  </declarations>
  <error>Class HokusPokus not found</error>
  <main>
    <error>Undeclared variable abracadabra</error>
    <warning>Deprecated method simsalabim</warning>
  </main>
</program>
```

Umwandlung von Select-Mustern

Knotenselektion mittels Select-Muster

```
<fxt:spec>
  <fxt:pat>//program</fxt:pat>
    <list>
      Errors   : <fxt:apply select="//error"/>
      Warnings: <fxt:apply select="//warning"/>
    </list>

  <fxt:pat>//error</fxt:pat>
    <b><fxt:apply/></b>

  <fxt:pat>//warning</fxt:pat>
    <i><fxt:apply/></i>
</fxt:spec>
```

Umwandlung von Select-Mustern

Knotenselektion mittels zweistelliger Match-Muster

```
<fxt:spec>
  <fxt:pat>//program[(//%error)?][(//%warning)?]</fxt:pat>
    <list>
      Errors   : <fxt:apply select="1"/>
      Warnings: <fxt:apply select="2"/>
    </list>

  <fxt:pat>//error</fxt:pat>
    <b><fxt:apply/></b>

  <fxt:pat>//warning</fxt:pat>
    <i><fxt:apply/></i>
</fxt:spec>
```

Fxgrep vs. XPath

XPath:

- Numerical predicates: `//a[42]`
- Data values comparisons: `//a[b=c]`

fxgrep:

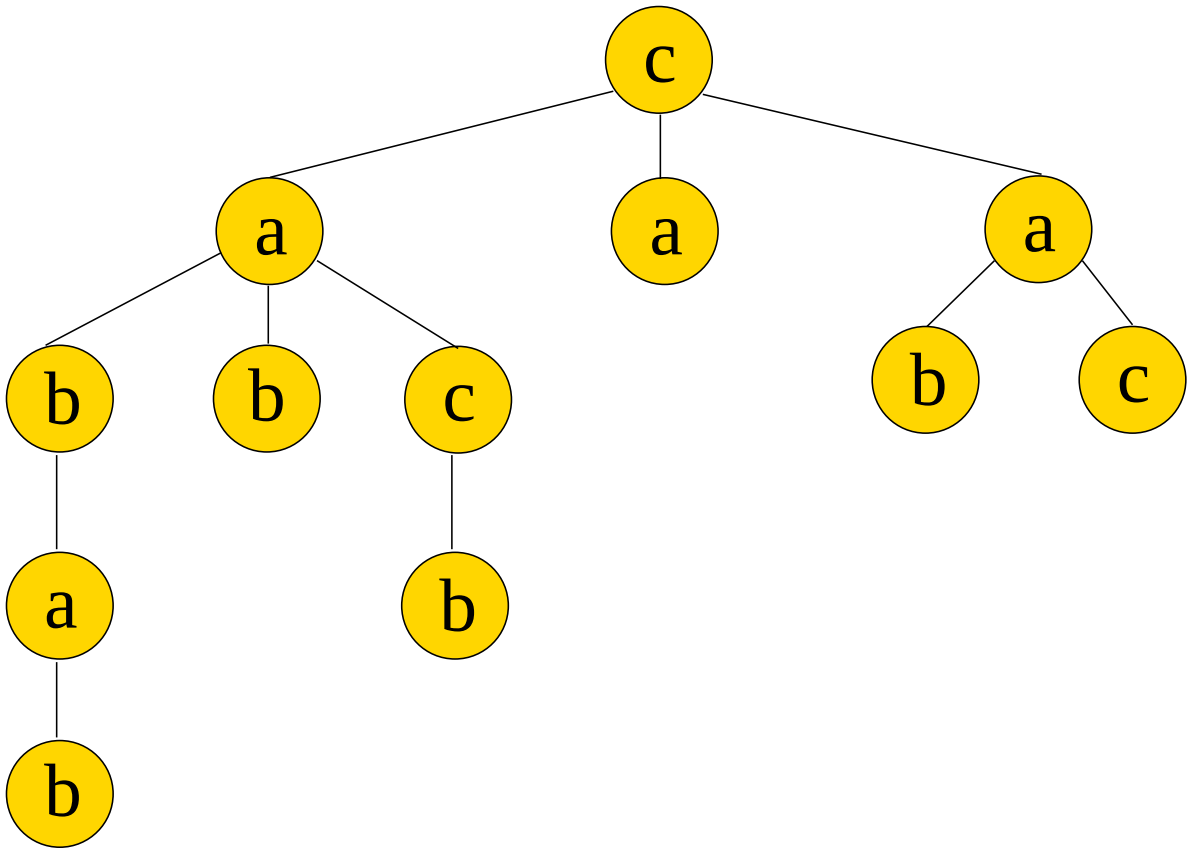
- Regular constraints on paths: `(a/b/)+ c`
- Regular constraints on siblings patterns: `//a[b+ c d+]`

XPath's operational model is completely different from fxgrep model

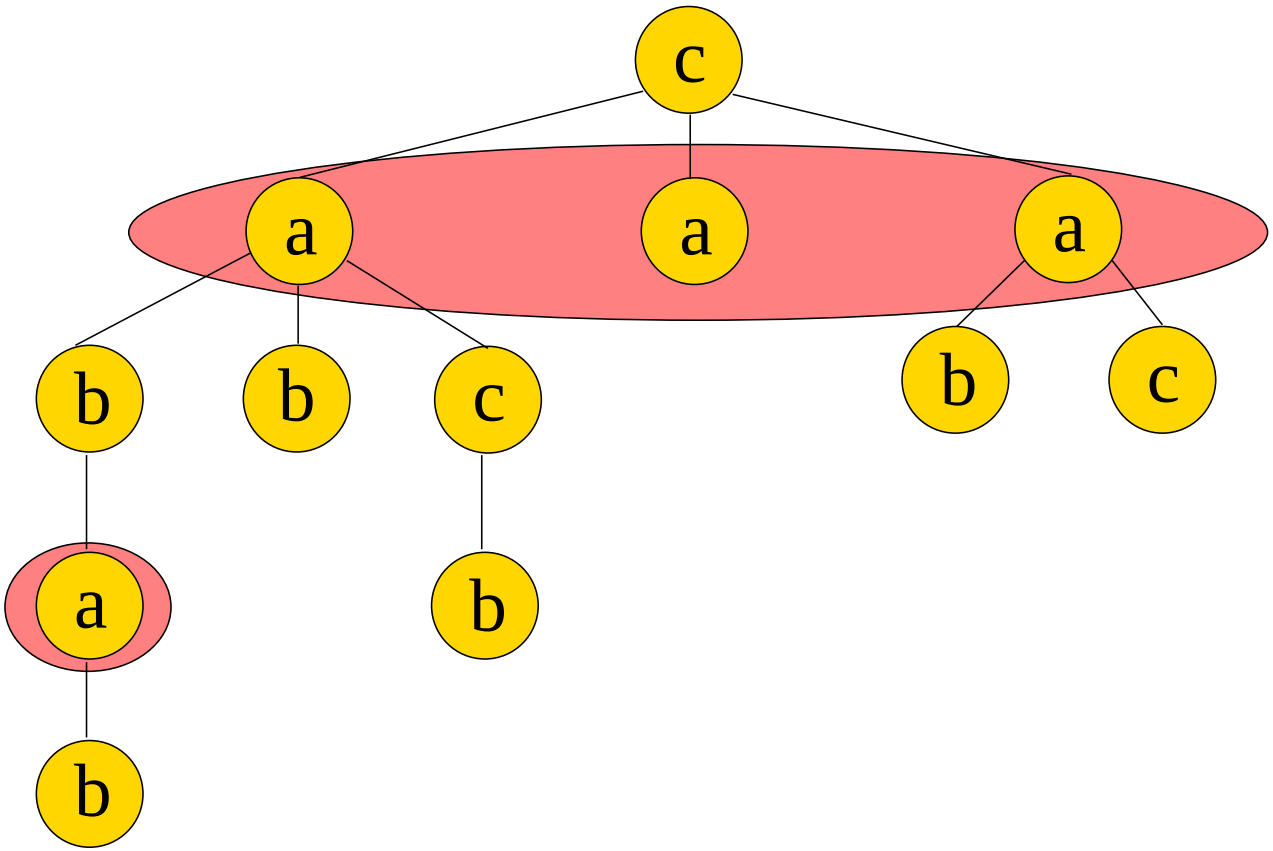
- a number of successive filtering steps
- arbitrary navigation through the document tree

//a//b

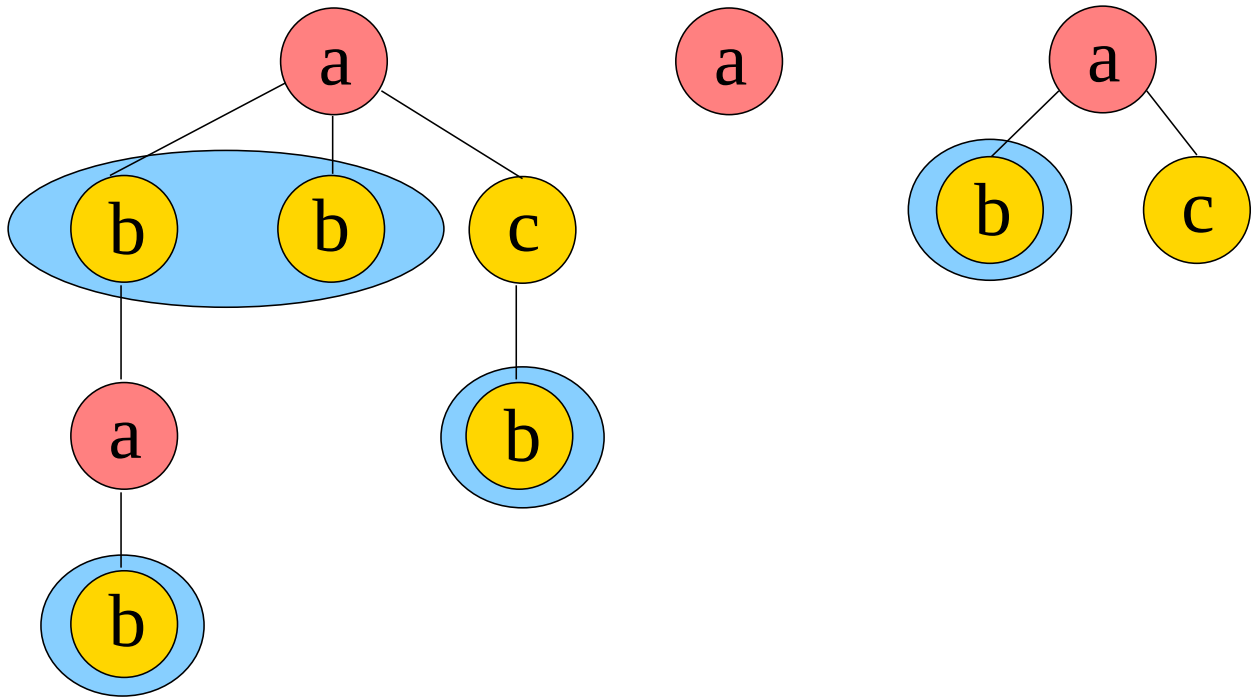
```
//a//b
```



```
//a//b
```

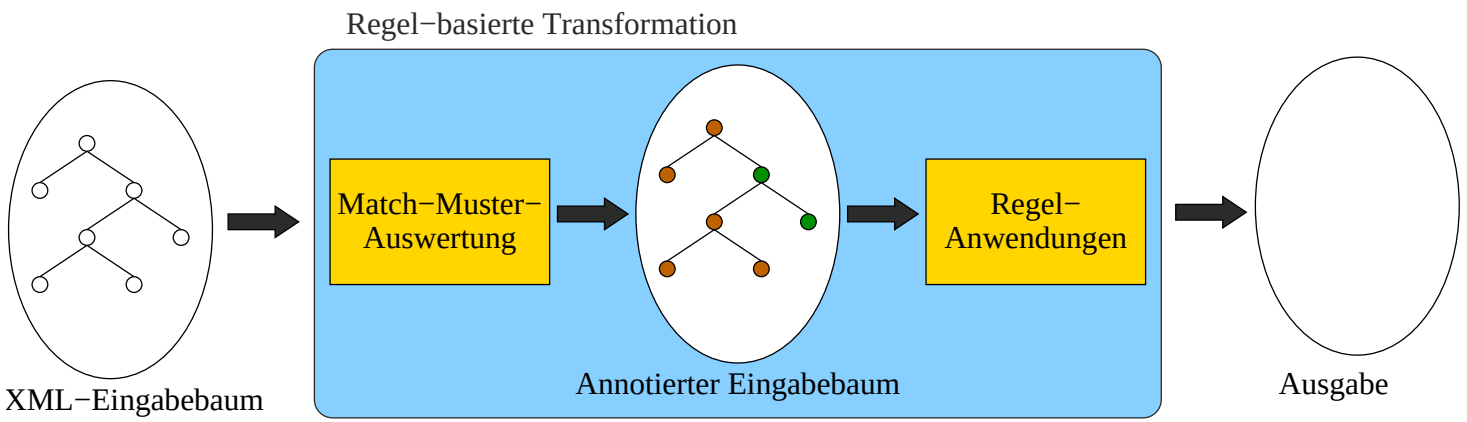


//a//b



Match-Muster versus Select-Muster

- Match-Muster
  - beziehen sich auf den Wurzel-Knoten
  - können vor der Anwendung der Regeln ausgewertet werden
- Select-Muster
  - beziehen sich auf den Argument-Knoten einer Regel-Anwendung
  - werden bei Regel-Anwendung ausgewertet



Generalization

All (unary) patterns in XML applications are either absolute or relative:

```
for $x1 in document("doc.xml")//a
return
  <A>
  {for $x2 in $x1//b
    return
      <B>
      {for $x3 in $x2//c
        return
          <C/>}
      </B>}
  </A>
```

Generalization

All (unary) patterns in XML applications are either absolute or relative:

```
for $x1 in document("doc.xml")//a
return
  <A>
  {for $x2 in $x1//b
    return
      <B>
      {for $x3 in $x2//c
        return
          <C/>}
      </B>}
  </A>
```

Let p_2 be a pattern relative to p_1

$\Longrightarrow$ there is an (absolute) binary pattern p s.t. (n_1, n_2) is a matc

n_1 is a match of p_1 and n_2 is a match of p_2.

Generalization

All (unary) patterns in XML applications are either absolute or relative:

```
for $x1 in document("doc.xml")//a
return
  <A>
  {for $x2 in $x1//b                    ⟹  //a[(//%b)?]
    return
      <B>
      {for $x3 in $x2//c                ⟹  //a//b[(//%c)?]
        return
          <C/>}
      </B>}
  </A>
```

Let p_2 be a pattern relative to p_1

$\Longrightarrow$ there is an (absolute) binary pattern p s.t. (n_1, n_2) is a matc

n_1 is a match of p_1 and n_2 is a match of p_2.

XQuery Patterns

```
for $x in //a return
  for $y in $x//b return
    $y//c
```

$\$n_1$	return
1	<table><tr><td>$\$n_2$</td><td>return</td></tr><tr><td>2</td><td><table><tr><td>$\$n_3$</td><td>return</td></tr><tr><td>3</td><td>3</td></tr><tr><td>6</td><td>6</td></tr></table></td></tr><tr><td>5</td><td>6</td></tr></table>
4	<table><tr><td>$\$n_2$</td><td>return</td></tr><tr><td>5</td><td><table><tr><td>$\$n_3$</td><td>return</td></tr><tr><td>6</td><td>6</td></tr></table></td></tr></table>

$Tab_{\mathcal{M}_{Q_1}}$:

n_1
1
4

$Tab_{\mathcal{M}_{Q_{1,2}}}$:

n_1	n_2
1	2 5
4	5

$Tab_{\mathcal{M}_{Q_{2,3}}}$:

n_2	n_3
2	3 6
5	6