Anwendung AST/TA Ansatz für Aufbereitung von C-Quellen
(zur Dokumentation Gewinnung)

Guido Draheim
<draheim@informatik.hu-berlin.de>

Januar 2003

SGML Markup
- Unterscheidung in UrDaten und MetaDaten
- Annotierung der UrDaten mit MetaDaten
.... sie Pattern Beispiel mit Max&Moritz – als explizitMachung letztlich schon vorhandener Strukturen
.... Gegenfall Datenmodellierung heute, bei der Attributwerte Daten ersten Ranges sein können
- C Quellen als UrDaten
... nebenlauf: MBS/MSC fur frame-urdaten, metadaten-generierung, daten-transformation
Systeme der Darstellung
- xm-tool .... linearer Text mit eingebetten XML tags
- xmlg ... ast/ta mit linearem text und XML baum
- libxml .. dom mit textstücken je knoten im XML baum
- xml/db ... xml-extended db, mischform, felder gewertet wie xml-elemente,
..... die selbst xml textstücke enthalten.
Operationen
- Pattern Recognition ... und Markup Anbringung
- Markup Combination and Recording ... Markup und Attribut Anbringung
- Information Lookup and Recording ... zumeist Attribut Anbringung / linkrefs
- Selection and Reordering .. zur Report Generierung
- Formatting ... umformung von xml in zielform
..... grob
- Einlesen = UrDaten + Markup Gewinnung
- Ausgeben = Report + Daten Reduktion
Vergleiche

Wertung

Ausblick
- XLST mit PCRE ?
- lokale Prozeduren – Interpretersprache, distrib. Processing – sekundärspeicher-geschwindigkeit
- trennung der phasen günstig! zwecks positionserhaltung – muss das so sein?

Wilhelm Busch schrieb Max und Moritz

```
<?xml version=“1.0“ encoding=“UTF-8“ ?>
<!DOCTYPE book SYSTEM “book.dtd“>
<book>
  Wilhelm Busch schrieb Max und Moritz
</book>
```

.author. >> *[name] schrieb [name]* << .title.

```
<?xml version=“1.0“ encoding=“UTF-8“ ?>
<!DOCTYPE book SYSTEM “book.dtd“>
<book>
  <author>Wilhelm Busch</author> schrieb
  <title>Max und Moritz</title>
</book>
```

```
<?xml version=“1.0“ encoding=“UTF-8“ ?>
<!DOCTYPE book SYSTEM “book.dtd“>
<book>
  <author>Wilhelm Busch</author> schrieb
  <title>Max und Moritz</title>
  im Jahre <year>1895</year>
</book>
```

<author><title><year> .... <title published=“[year]“>

```
<?xml version=“1.0“ encoding=“UTF-8“ ?>
<!DOCTYPE book SYSTEM “book.dtd“>
<book>
  <author>Wilhelm Busch</author> schrieb
  <title published=“1895“
        >Max und Moritz</title>
  im Jahre <year>1895</year>
</book>
```

```
<?xml version=“1.0“ encoding=“UTF-8“ ?>
<!DOCTYPE book SYSTEM “book.dtd“>
<book>
  <author>Wilhelm Busch</author> schrieb
  <title published=“1895“
        >Max und Moritz</title>
<!-- im Jahre <year>1895</year> -->
</book>
```

```
<?xml version=“1.0“ encoding=“UTF-8“ ?>
<!DOCTYPE book SYSTEM “book.dtd“>
<book>
  <author>Wilhelm Busch</author> schrieb
  <title published=“1895“
        >Max und Moritz</title>
<!-- im Jahre <year>1895</year> -->
</book>
```

```
<?xml version=“1.0“ encoding=“UTF-8“ ?>
<!DOCTYPE book SYSTEM “book.dtd“>
<book>
  <author>Wilhelm Busch</author> schrieb
  <title published=“1895“
        >Max und Moritz</title>
<!-- im Jahre <year>1895</year> -->
</book>
```

When=“//title[@published='1895']“

Wert:

Max und Moritz

perl-xpath

```
$t =~ s{
      (<author(?:\s[^<>]*)?>)
          ((?:.(?!</?author[\s>]))*.)
              (</author(?:\s[^<>]*)?>)
      ((?:.(?!</?book[\s>]))*.?)
      (<title(?:\s[^<>]*)?>)
          ((?:.(?!</?title[\s>]))*.)
              (</title(?:\s[^<>]*)?>)
              }
```

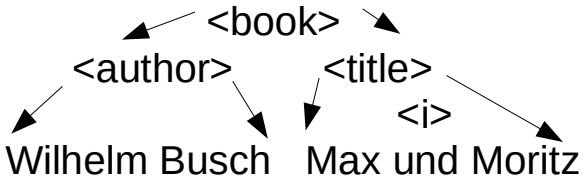
//author+title
//title

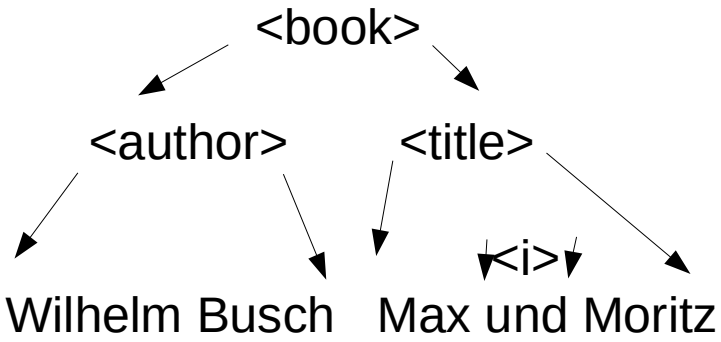
wie drückt man ne schachtelung aus?

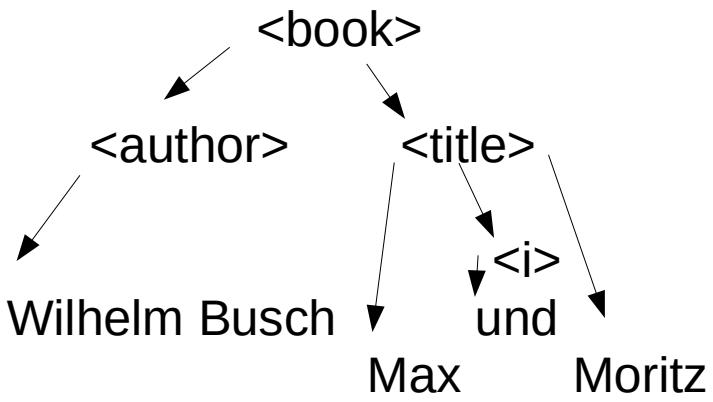
//book/title

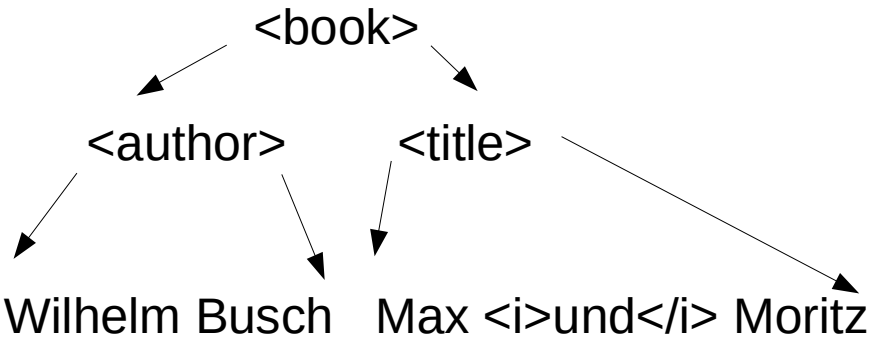
Hinweis: xmleDB – hineinparsen in xml-stücke? fast genauso!
Achtung: xm-tool format kein echtes xml, PI, comments, etc.

```
<book>
  <author>Wilhelm Busch</author>
  <title>Max <i>und</i> Moritz</title>
</book>
```









```
<publishdata>
   <author>Wilhelm Busch</author>
   <book>Max und Moritz</book>
   <published>1895</published>
</publishdata>

<publishdata>
   <author>Wilhelm Busch</author>
   <book published="1895">Max und Moritz</book>
</publishdata>

<publishdata>
  <book author="Wilhelm Busch" published="1895">Max und Moritz</book>
</publishdata>

<publishdata book="Max und Moritz"
          author="Wilhelm Busch" published="1895" />
```

tatsächlich: drei Objekte / Informationsteile
syntaktische Beziehung / Anordnung jeweils gegeben
bedeutungszuordnung zwischen den teilen nicht hier

Unterscheidung UrDaten .vs. Metadaten?

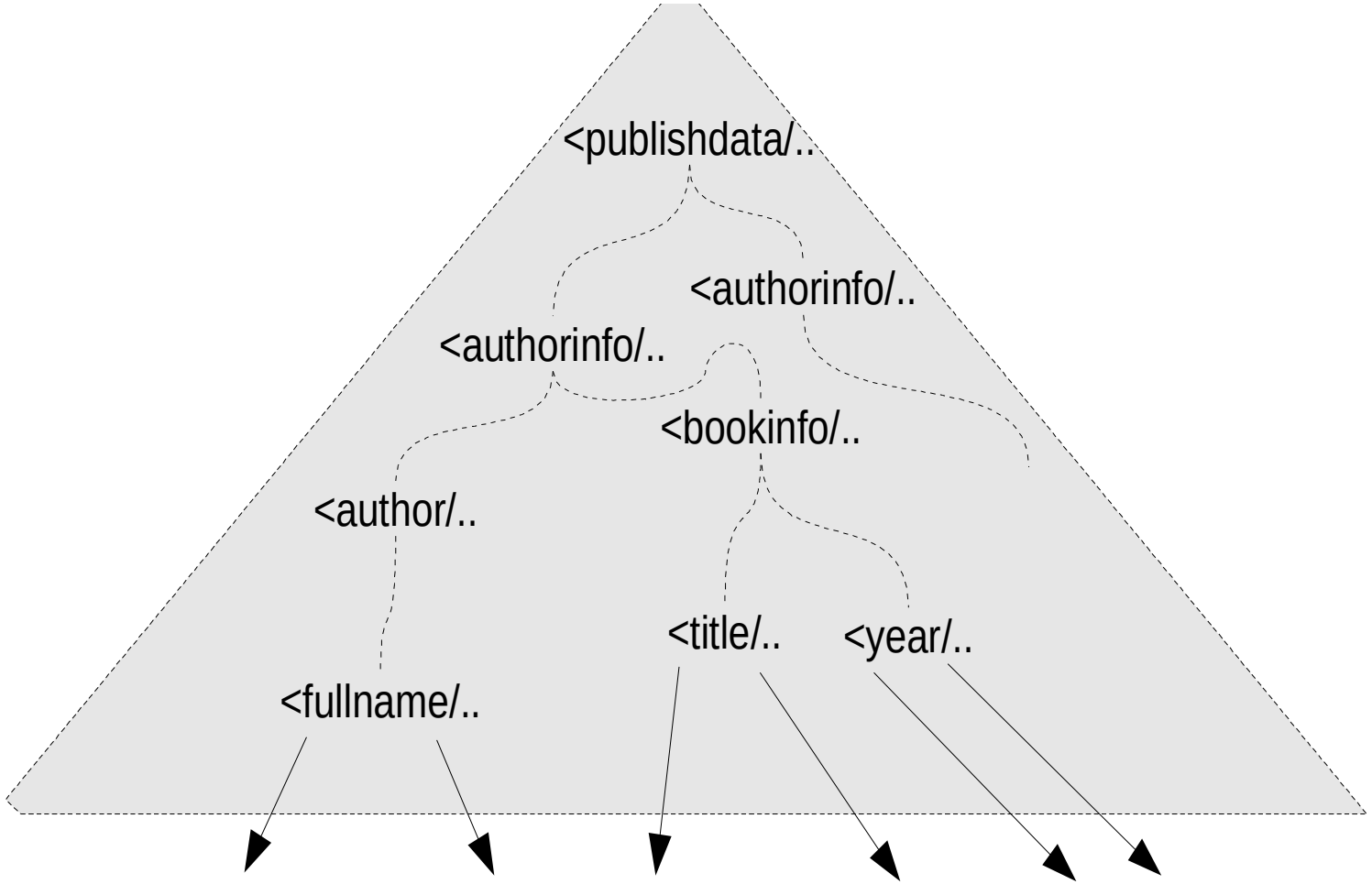
```
<?xml ... ?>
<!DOCTYPE ...>
<book>
  <record>
    <author/>

    <title/>
  </record>
</book>
```

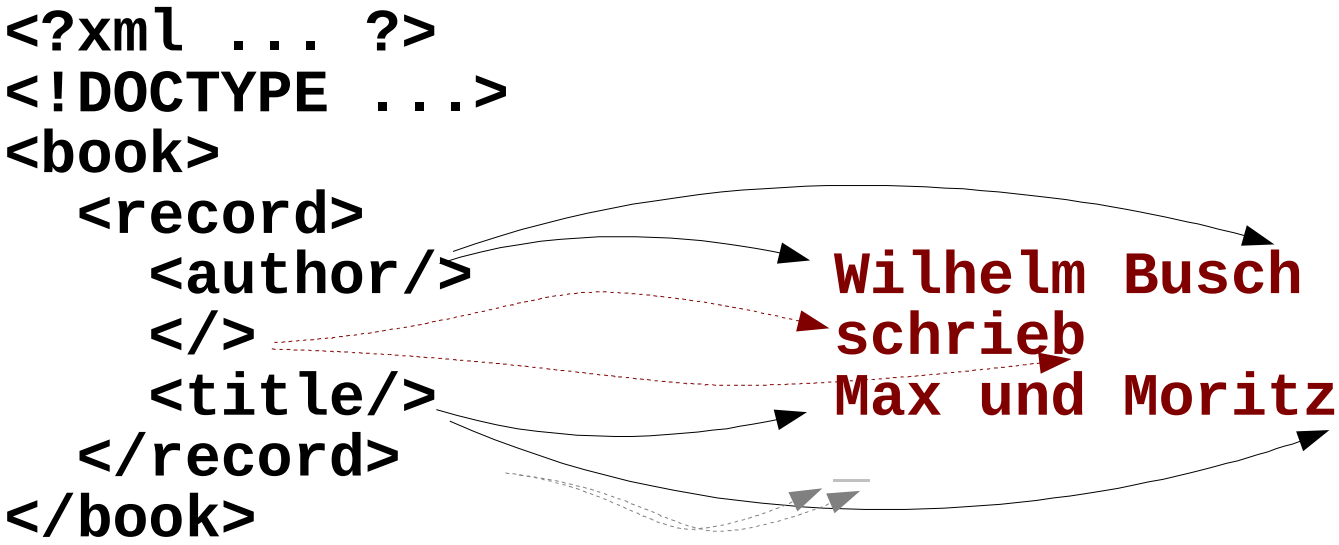
Wilhelm Busch
schrieb
Max und Moritz

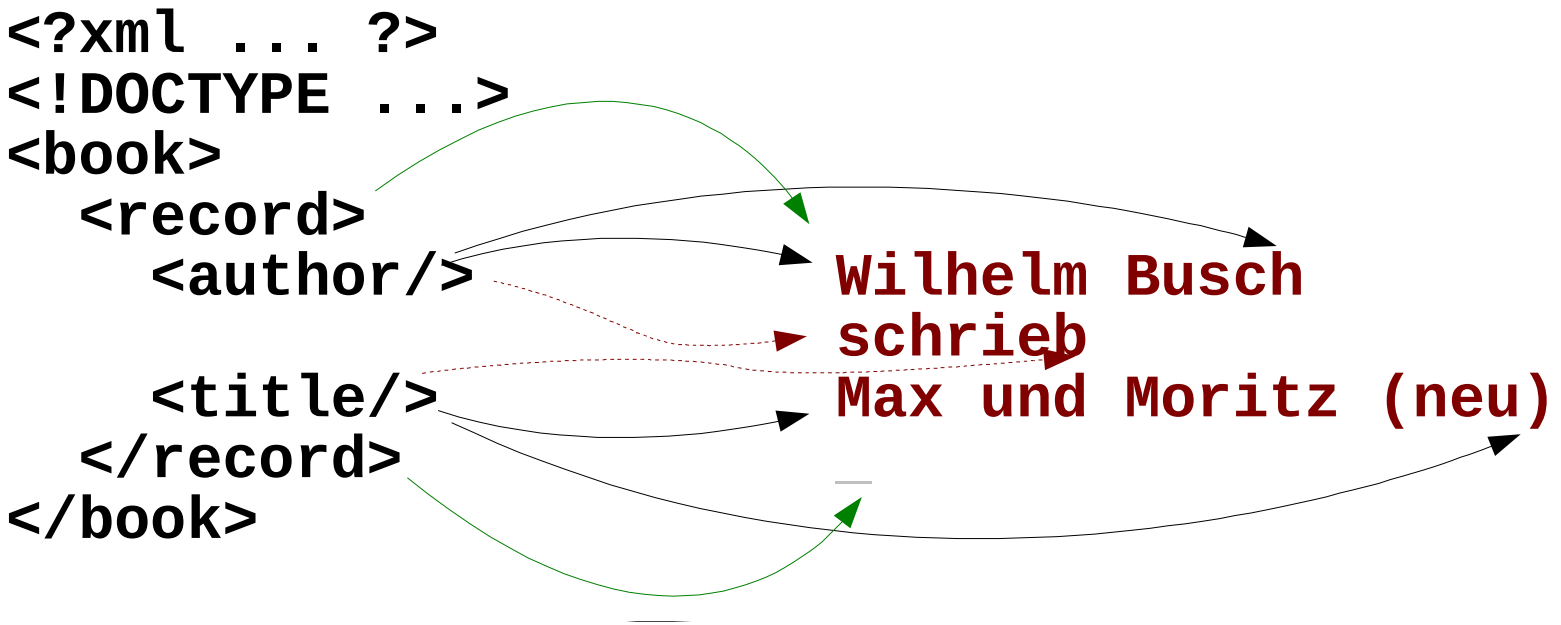
```
<?xml ... ?>
<!DOCTYPE ...>
<book>
  <record>
    <author/>

    <title/>
  </record>
</book>
```

Wilhelm Busch
schrieb
Max und Moritz (neu)

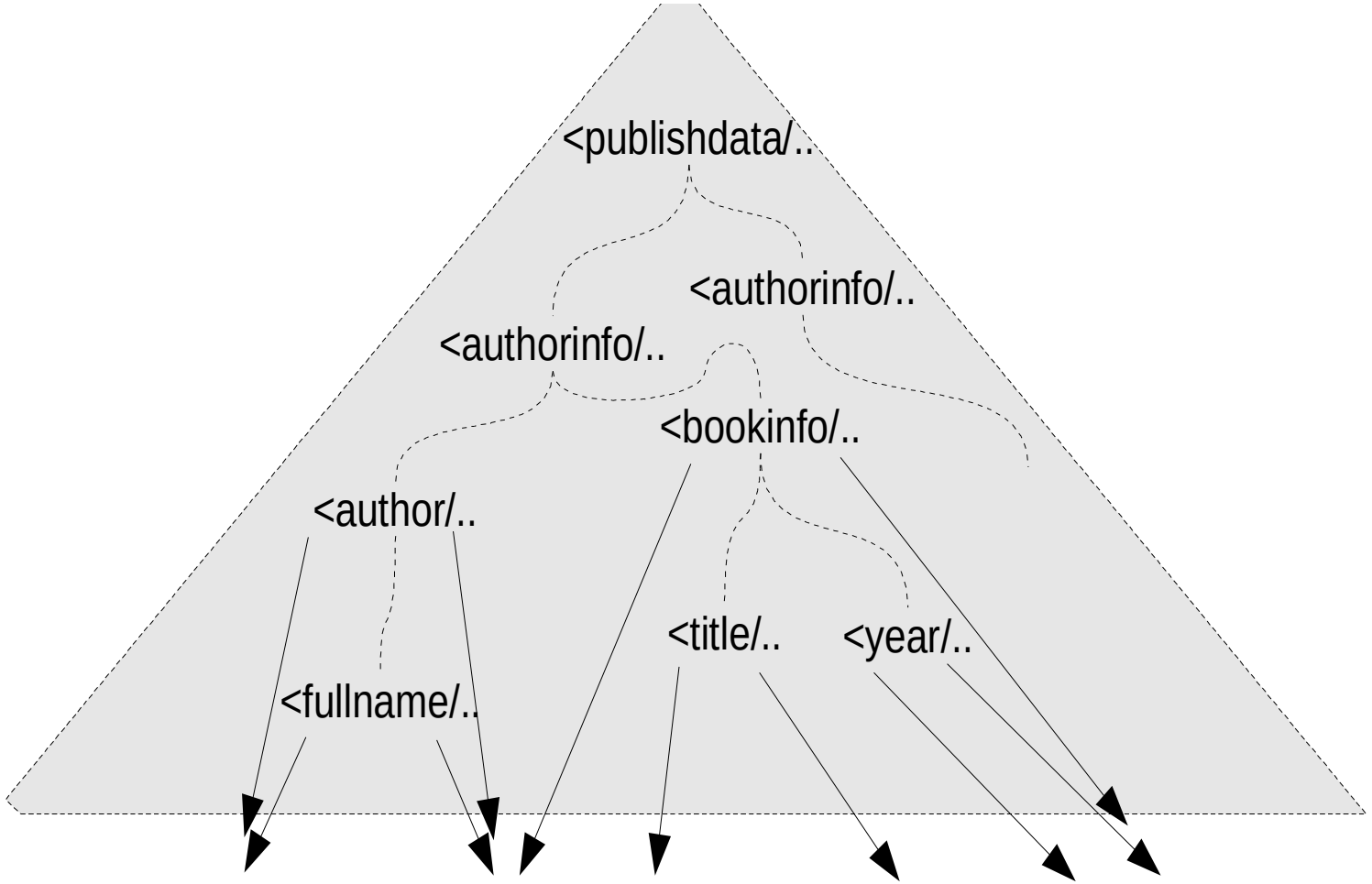
AST/TA – Baum


<publishdata/..
<authorinfo/..
<authorinfo/..
<bookinfo/..
<author/..
<title/..
<year/..
<fullname/..
Wilhelm Busch schrieb Max und Moritz im Jahre 1865

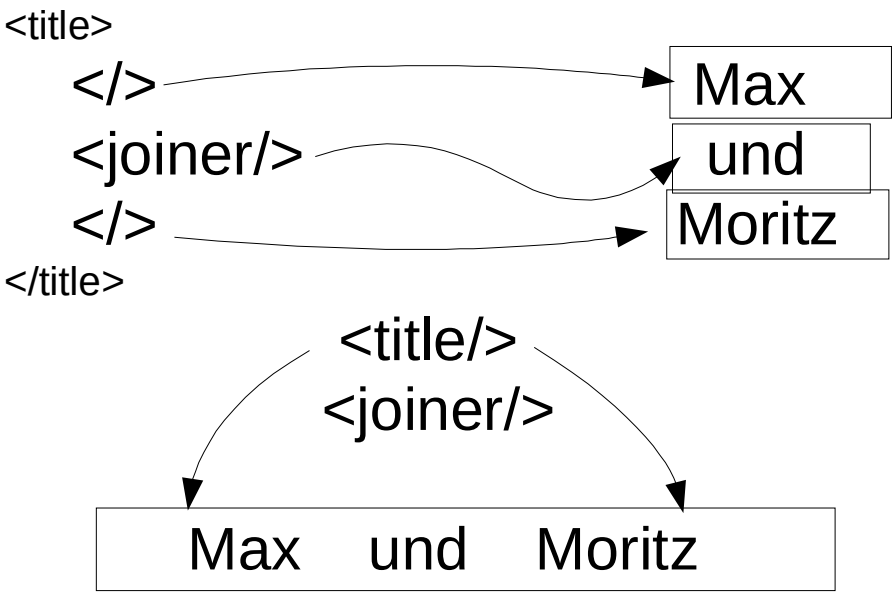

<?xml ... ?>
<!DOCTYPE ...>
<book>
<record>
<author/>
</>
<title/>
</record>
</book>
Wilhelm Busch
schrieb
Max und Moritz


<?xml ... ?>
<!DOCTYPE ...>
<book>
<record>
<author/>
<title/>
</record>
</book>
Wilhelm Busch
schrieb
Max und Moritz (neu)

AST/TA – abschnitte



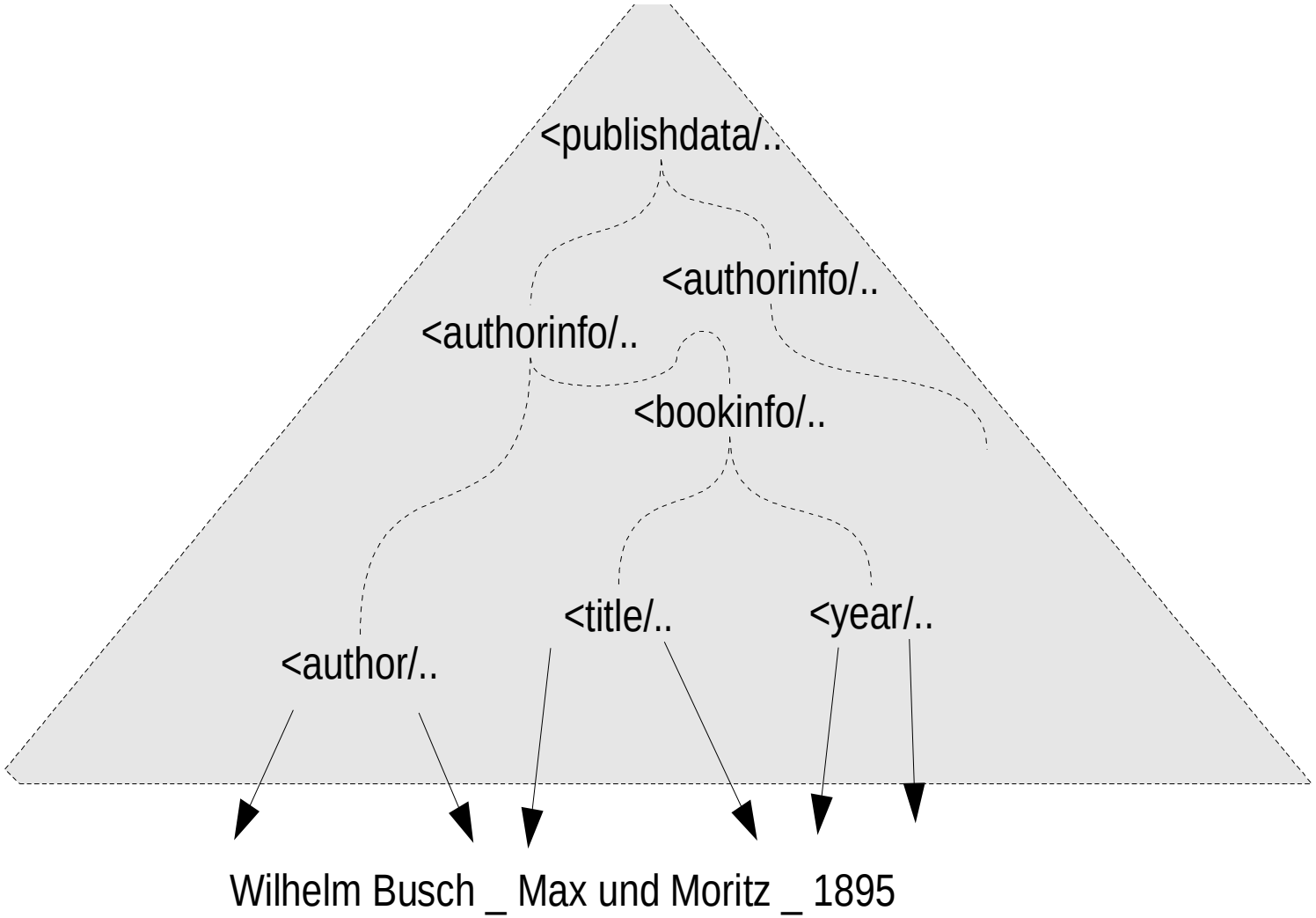
<title>Max <joiner>und</joiner> Moritz</title>



suche nach
"Max und Moritz"

<word><big>X</big>ML</word>

und wo ist
XML ?

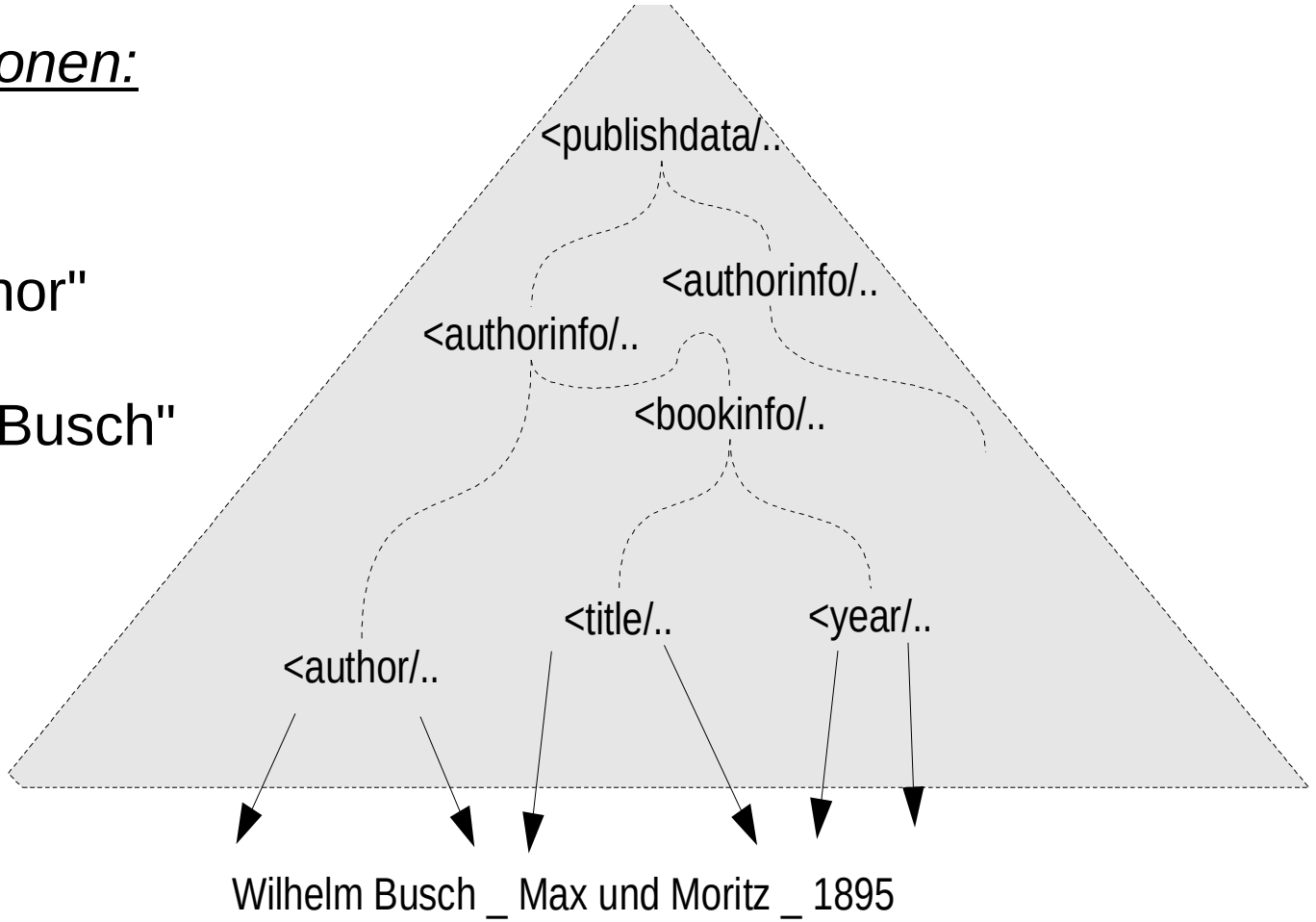


... <author>Wilhelm Busch</author><bookinfo ...
><title>Max und Moritz</title><year>1895</year></bookinfo> ...

spez. Operationen:

xpath: "//author"

PCRE: "/w+\sBusch"



xmlg/libpcre zeigt Verwendung
Anpassung an TextArray auf Sekundärspeicher
Zeichen-Hol-Funktion - vorwärts/backtracking
Optimierung auf möglichst lokalen Zugriff

aus Text Mining / Retrieval
Index-listen der Worte
Thesauri Match und Suche
Abgleichen bei Insert/Delete

SGML Markup
Systeme der Darstellung

Operationen

Vorteil
.... geschwindigkeit – in C oder perl
.... markupgewinnung bringt keine string-operation, sondern verlinkung im hauptspeicher
.... reportgewinnung auf xpath selections – keine textdurchsung noetig
Nachteil
.... pattern auf elementnamen schwieriger – in xmlg eingebettet als pcre auf element/attribnamen
.... überspringen nichtzubeachtender anteile wie kommentaren; wenn sie im xml-baum liegen einfacher, sonst ausblenden moeglich – ausblenden heisst, sie aus dem text in den xml-baum zu uberfuehren, und hinterher wieder zurueck
Vergleich
.... Messung
Wertung
.... wiederverwendung von algorithmen zur baumsuche und geschwisterbildung
.... wiederverwendung von pcre auf dem textstring – beide uebrigens die reVM nicht modifiziert
Verbesserung
.... mehrpass-system derzeit – sowohl zur gewinnung wie extraktion – besser die lokalitaeten ausnutzen ... dazu ein beispiel!
Ausblick
- XLST mit PCRE ?
- lokale Prozeduren – Interpretersprache, distrib. Processing – sekundärspeicher-geschwindigkeit
- trennung der phasen günstig! zwecks positionserhaltung – muss das so sein?

xm-tool
- in perl
- xml-linear
- 40min etwa
xmlg
- in C/glib
- ast/ta ansatz
- 10sek = disk/io

erkennen:
 fcode delcare
 compiles declare
 listwords declare
 fcode comment
 export-entries
combining:
 export-types
 export-list
 name and comment
 external examples
 und weiteres
generation:
 per word
 per wordset
 hyperlinked

```
/** HERE-WORD ( char "name<char>" -- )
 * a FIG-compatible WORD. Where ANSI says
 * "skip leading delimiters" this one acts as
 * "skip leading whitespace". And it will not
 * return anything and have the string parsed
 * to => HERE
 */
FCode (p4_here_word)
{
    p4_here_word ((char) FX_POP);
}
P4COMPILES(p4_here_word, p4_here_execution);

P4_LISTWORDS (your) =
{
    P4_INTO ("EXTENSIONS", 0),
    P4_SXco ("INTO",            p4_into),
    P4_SNYM ("&OF",            "INTO"),

    P4_FXco (".H2",             p4_dot_h2),
    P4_FXco ("HERE-WORD",       p4_here_word),
    P4_xOLD ("FIG-WORD",        "HERE-WORD"),
};
P4_COUNTWORDS (your, "YOUR kernel extensions");
```

```
</br>}</cblock></item_cblock><br line="332 ../../doc/pfe/_zz-ext.c">
</br><item_cblocc line="333 ../../doc/pfe/_zz-
ext.c">P4COMPILES(p4_z_backslash_quote, p4_z_quote_XT,<br line="333
../../doc/pfe/_zz-ext.c">
</br>           p4_z_quote_SEE,
P4_DEFAULT_STYLE)<cblocc>;</cblocc></item_cblocc><br line="334
../../doc/pfe/_zz-ext.c">
</br><br line="335 ../../doc/pfe/_zz-ext.c">
</br><br line="336 ../../doc/pfe/_zz-ext.c">
</br><item_cblock line="337 ../../doc/pfe/_zz-ext.c"
p4_listwords="zchar">P4_LISTWORDS(zchar) =<br line="337
../../doc/pfe/_zz-ext.c">
</br><cblock wordset="zchar">{<br line="338 ../../doc/pfe/_zz-ext.c">
</br><export_line export_type="- loading into" forth_name="FORTH"
line="338 ../../doc/pfe/_zz-ext.c" wordset="zchar">
<export_type>P4_INTO</export_type>
(<sliteral>&quot;<export_string>FORTH</export_string>&quot;</sliteral>,
<export_value>0 </export_value>),</export_line><br line="339
../../doc/pfe/_zz-ext.c">
</br><export_line export_type="compiling primitive"
forth_name="Z&quot;" line="339 ../../doc/pfe/_zz-ext.c"
wordset="zchar">    <export_type>P4_SXco</export_type>
(<sliteral>&quot;<export_string>Z\&quot;</export_string>&quot;</slitera
l>,              <export_value>p4_z_quote</export_value>),
</export_line><br line="3
```

for each wordset //wordset
for each line //br+br

```
___ static const gchar* names1[] =
    { "", "export_type", "export_string", 0};
xml_pcre_match_add9 (
    node->text->str, line->end, ends->off,
    "?" "\\s*(\\w+)" "\\s*\\("
    "\\s*\"((?:[^\"\\\\]|\\\\.)*)\""
    "\\s*\\)\\s*,\\s*",
    node, names1);
____;
___ static const gchar* names2[] =
    { "", "export_type", "export_string", "export_value", 0};
xml_pcre_match_add9 (
    node->text->str, line->end, ends->off,
    "?" "\\s*(\\w+)" "\\s*\\("
    "\\s*\"((?:[^\"\\\\]|\\\\.)*)\""
    "\\s*,\\s*([^()]+)\\)\\s*,\\s*",
    node, names2);

xml_pcre_match_add9 (
    node->text->str, line->end, ends->off,
    "?" "\\s*(\\w+)" "\\s*\\("
    "\\s*\"((?:[^\"\\\\]|\\\\.)*)\""
    "\\s*,\\s*([^()]*\\([^()]*\\)[^()]*)\\)\\s*,\\s*",
    node, names2);
____;
```

```
for (; node ; node = next)
{
    next = node->next;

    if (! xml_node_hasname_as_(node, "*comment"))
        continue;

    while (next && xml_node_hasname_eq_(next, "br")) next = next->next;
    if (! next || !xml_node_hasname_as_(next, "*bloc")) continue;

    /* check if the comment-node is followed by an FCode declaration */
    if (! xml_text_match1 (
            node->text->str, node->end, next->off, "*FCode"))
        continue;

    if (xml_pcre_match_add9 (
            node->text->str, node->end, next->off,
            "+" "((?:[a-z]+\\s+)*" "F?X?Code)"
            "\\s*\\(\\s*" "(\\w+)" "\\s*\\)\\s*",
            tree, names))
    {
```

```
test "`./xml-get-list e.xml //F -space`" = "//A/E/F"
test "`./xml-get-list e.xml //C@width -space`" = "//KK/C"
test "`./xml-get-list e.xml /*/*@width -space`" = "//KK/C"
test "`./xml-get-list e.xml //*@width -space`" = "//KK/C"
test "`./xml-get-list e.xml '//(D|E)' -space`" = "//A/B/D //A/E"
test "`./xml-get-list e.xml '//*K' -space`" = "//KK"
test "`./xml-get-list e.xml '//?K+' -space`" = "//KK"
test "`./xml-get-list e.xml '//?K+/I' -space`" = "//KK/I"
test "`./xml-get-list e.xml '//*@*wid' -space`" = "//KK/C"
test "`./xml-get-list e.xml '//?C+@*wid' -space`" = "//KK/C"
test "`./xml-get-list e.xml '//?C+@?wid.*' -space`" = "//KK/C"
test "`./xml-get-list e.xml '//*@?[wx].*' -space`" = "//KK/I
//KK/C"
test "`./xml-get-list e.xml '//*@?\w.*' -space`" = "//KK/I //KK/C"
test "`./xml-get-list e.xml '//*@?\w+' -space`" = "//KK/C"
test "`./xml-get-list e.xml '//*@+\w+' -space`" = "//KK/I //KK/C"
```

selection of textarray-area – rückgabe als liste von basic xpath – intern sind diese eindeutig zu paaren von textarray offsets assoziert (anfang...ende).

typische operationen:

_ xml_path_nodes_list (tree, xpath) -> node-list
  finde alle knoten die auf xpath passen (xpath maschine), gib knoten-liste zurueck,
  anschliessend freies bearbeiten – haeufig aufruf subtree mit speziellen funktionen

_ xml_path_nodes_foreach (tree, xpath, func, data)
  lokaler – rufe fuer jeden knoten (mit dessen unterbaum) funk/data auf, es darf
  im unterbaum modifizieren. - schachtelung!

_ xml_path_nodes_add (tree, xpath, regex, markup-list)
  vereinfach, je xpath, setze xpath auf textarray anteil um, gefunde patterns
  werden als markups abgespeichert

_ xml_pcre_followedby (node, key-regex, text-regex)
  und aehnliche test-funktionen auf knoten, zur kombination von logischen
  ausdruecken direkt in C, oft in schleife ueber nodes_list mit or-continue.

_ save_text_to_attribute(node, attrib-name, replacing)
  ersetze inhalt des textarray an dieser schnell mit leerzeichen (als weisser separator)
  zur beschleunigung nachfolgender pcre-matches auf dem textinhalt.

_ add_new(tree, name, off1, off2)
  das ergebnis eines pcre-matches (anfang,ende) als markup hinzufuegen

_ group_add(node1, node2, name)
  fasse knoten zwischen 1 und 2 unter neuem knoten "name" zusammen

_ group_cut (node)
  vernichte knoten, fasse unterliegende in vater's kindliste zusammen

_ lookup routinen
  fur attribute in knoten, und pcre matches auf attribut-name und element-namen

_ xml_node_append_copy (tree, node)
  kopiere unterbaum unter node in einen ziel-report-baum

_ xml_find_node2 (tree, off1, off2) / xml_path_strdup (node)
  finde gemeinsamen oberknoten fuer off1 und off2, erstelle basic-xpath fuer diesen

PFE selbst:
- 1MB C Quellen
- 10MB XML Markup
- 1100 manpages

Tek/MForth
- 2,2 MB C Quellen
- 30MB XML Markup
- 2100 manpages

Ohne swig Modules.
- "> 6000" manpages

Zeit:
- xm-tool/perl, 35min für PFE auf homePC
- xm-tool/perl, 45min für Tek/MForth auf 4way sparc
- xmlg/pcre 20sec auf homePC, 10sec auf e10000
- zwischenformat etwa identisch.

Programmgrössen:
- pfe/doc spezifisch in xm-tool: 1900 zeilen
- pfe/doc spezifisch in xmlg: 1000 zeilen

(schachtelungsproblematik, comment-spanning)

zusätzlich:
- externe xml-tools auf zwischenformat von xm-tool
  nicht direkt anwendbar, da xml syntax fehler enthalten,
  vornehmlich oops
- firma stellt andere docs auf docbook um.

next: warum....

<author>Wilhelm Busch</author> schrieb <book>Max und Moritz</book>.

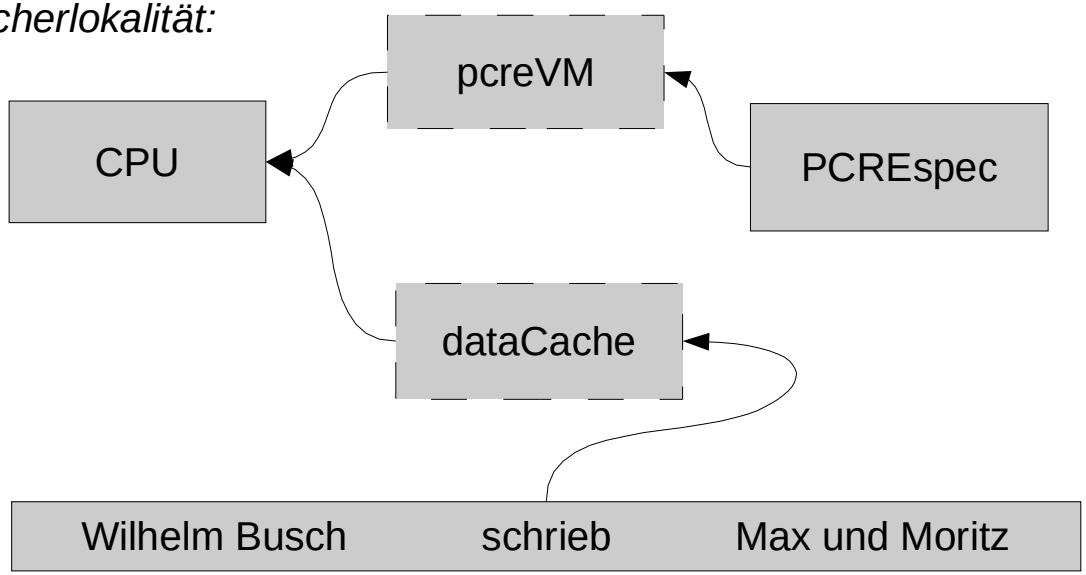
Wilhelm Busch schrieb Max und Moritz.

als PRCE

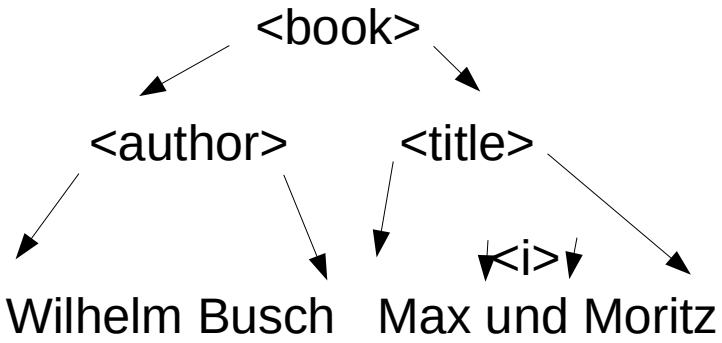
<[^<>]*>Wilhelm Busch (?:<[^<>]*>|\w+|\s*)? Max und Moritz (?!\s*\w)

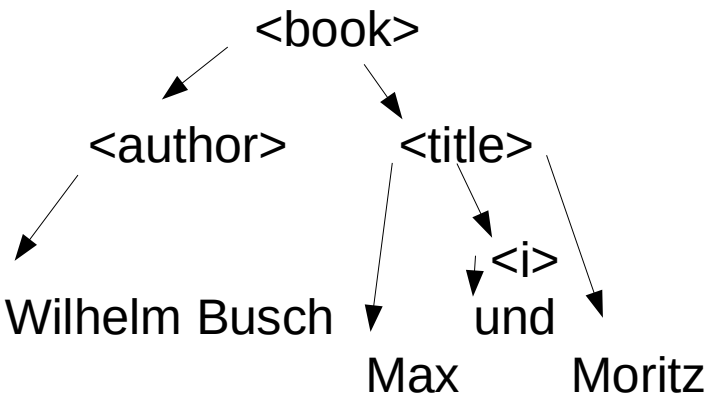
Wilhelm Busch [\w\s]* Max und Moritz(?!\s*\w)

speicherlokalität:

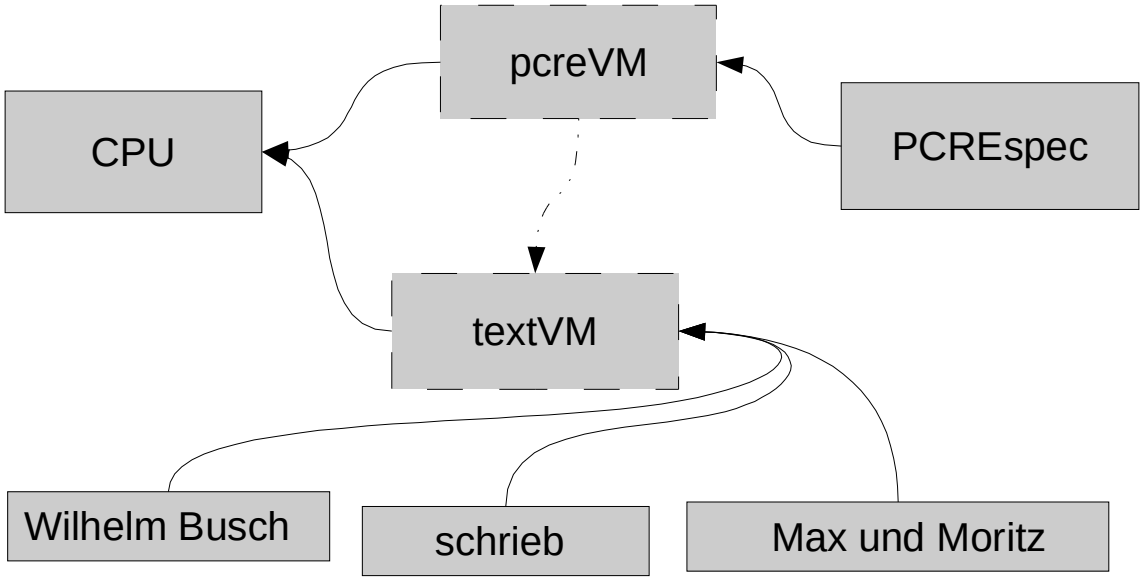


note: beides wiederverwendung pcreVM auf lin.text

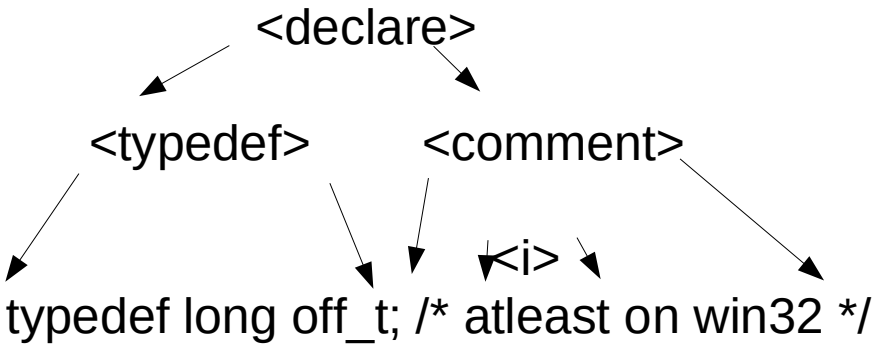




speicherlokalität:



note: modifizierte pcreVM, notwendig textVM



Verbreiterung Lokalität:

- Finde eine deklaration (typedef... ;)
- Bestimme dess Typ (von basistype)
- Bestimme Querreferenzen (verwendet in)
- suche vorstehenden/nachfolgenden kommentar
- erweitere declare-span um dieses feld
- parse den kommentar nach auszeichung
- parse nach querreferenzen
- fuege querreferenzen hier ein (als attrib)

und

- reportgenerierung, nimm alle felder, und uberführe sie in ausgabe-xml – etwa xml refentry (synopsis, include, see-also)

bisher:

- gehe uber ganzen text mit pcre
- finde damit deklarationen
- für-alle "//declare" bestimmte typ
- für-alle "//name@typedef" finde querrefs
- für-alle "//declare" finde nachfolgend comment-block
- für-alle "//declare/comment" parse speziell nach see-also einträgen

und

- xslt – when //decare bestimme felder do-elements für report

XSLT mit PCRE
- kritikpunkt an XSLT/XPath/XQuery etc, matching auf teile von inhalten statt direkt der struktur
- schachtelungsformen – kann pcre eingrenzung verwendet werden für virtuelle struktur
- wie können virt.pattern im xslt weitergereicht werden, für verschachtelung
- wie kann bearbeitungslokalität erreicht werden, gerade in hinblick auf sekundärspeicher

Xupdate mit PCRE
- statt report-generierung, arbeit auf dem originalen Datenbestand und modifikation
- welche transformationen koennen so ausgefuehrt werden
- kann xslt in-place ausgeführt werden

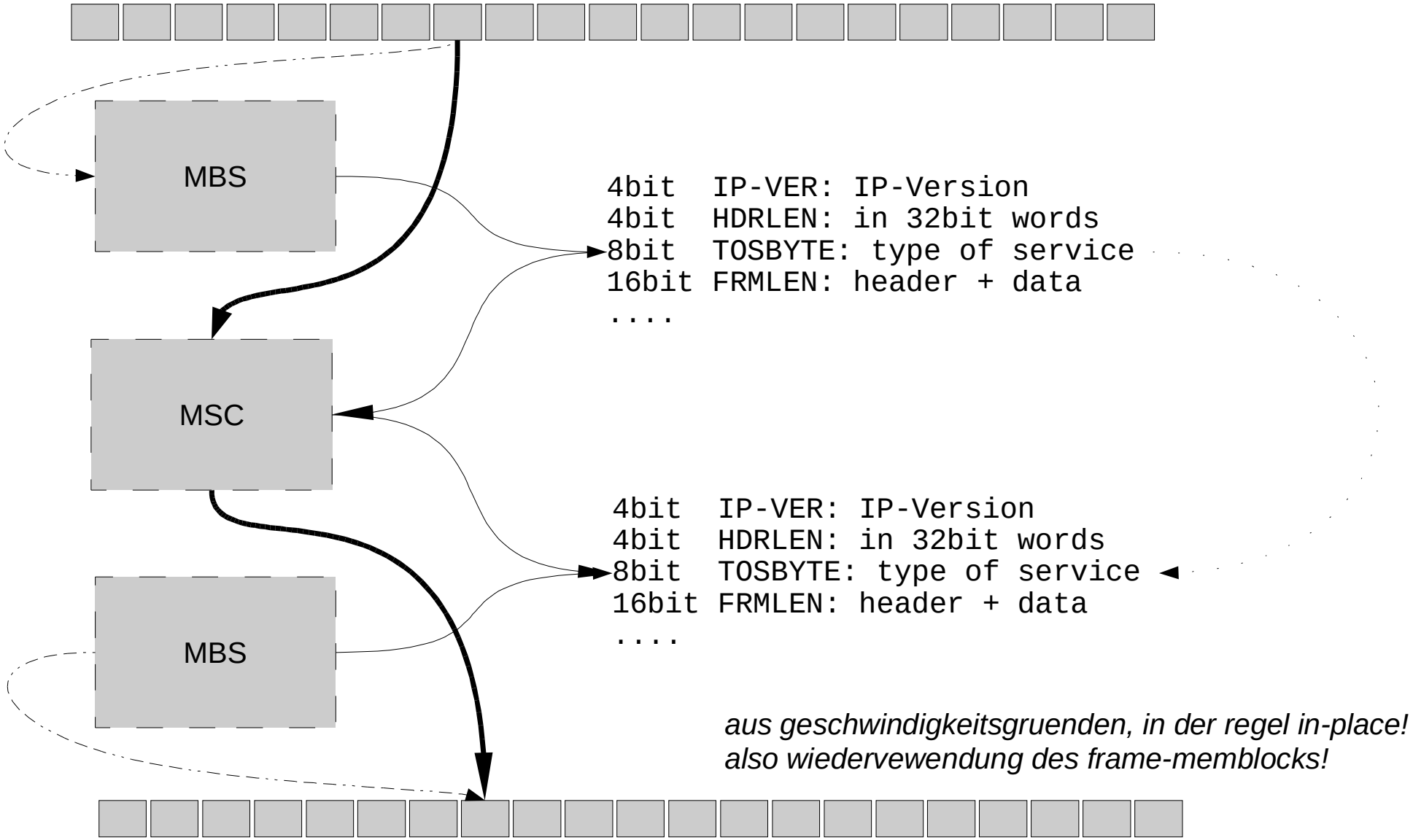
Frage ast/TA
- hilft ast/ta für pattern-gewinnung in-place (JA!)
- ist lokalität gegeben (besser, schachtelung!)
- transformations-script in xml, mit pcre (nicht definiert)
  ... xupdate/dbupdate beziehen sich auf xmleDB
- firstclass trans-scripte und parametriesierung
  ... ebenfalls ausdrückbar, da TA da kaum gebraucht wird,
  und sonst wohl literal zu verwenden ist (cdata output).

xmlg erweiterung:
- C darstellung von schachtelung ist mühsam!
- derzeit xpath und pcre enthalten,
- gebraucht, relations-angaben zwischen strukturen/textabschnitten
- eigene ausdruckssprache – an xslt anlehnen – als in-place update
- kompilierung/abspeicherung/optimierung zur weiteren beschleunigung

AST/TA Vorteile:
- Anpassung vorhander PatternAlgo auf Daten ist no-brainer
- Verwendung der Speicherstellen im TA als Indexe in den AST ! (Knotenfindung, struktur-invert-lookup)
- lokalitaet der datenzugriffen hoeher (xmldoc .vs. ast/ta .vs. dom/str)
- in-place update für xml-struktur und in-place update für urdaten-werte

AST/TA Nachteile
- erstmal keine die nicht auch bei umstellung von "echten" DB auf XML auftreten.
- das separator-problem, physisch zusammenliegende DB-Elemente mussen aufge"sperr"t werden.
- global positions vs database die keine ordnung unter felder benötigt.
- positionsupdate bei textupdate – da text nicht allein relativ zu struktur, sondern position ist global.
- daraus folgend – konkurriende updates und suchfolgen (sie next=node->next problem im beispiel!)
- in-place update sprache? mehr als db-orientiert!

MBS – Message Building System – auch erkennung von strukturen!
MSC – Message Sequence Charts – simulation, umsetzung von events



aus geschwindigkeitsgruenden, in der regel in-place!
also wiedervewendung des frame-memblocks!

arbeiten:
- andere pattern-algorithmen untersuchen
- pcre auf andere xml/db konzepte ansetzen
- xslt erweitern, virtuelle elemente
- in-place update prozesse für ast/ta untersuchen
- transformationen speichern(xml), modifizieren, ausführen (perVM)
- je geschwindigkeitsvergleiche – ist es immer schneller, corner cases

erfahrungen:
- xmlg zeigt struktur-update leicht
- xmlg zeigt pcre einsatz auf textarray
- xee zeigt sekundärspeicher einsatz
- xee update zu positionsbildung (oops)
- vm building fuer query processor

projekte:
- xm-tool.sf.net ... xmltext mit perl / markup generierung
- xmlg.sf.net .... ast/ta ansatz mit glib und pcre
- xee ... ansatz auf sekundärspeicher
- pfe.sf.net ... forth vm und compiler, eingesetzt auch im rahmen
  es mbs/msc ansatzes bei Tektronix berlin
- (update langs)