Einführung in XML

Seminar: XML in der Bioinformatik

Frank Schönmann

WS 2002/03

Überblick

- Entwicklung von XML
- Regeln und Eigenschaften von XML
- Document Type Definition (DTD)
- Vor- und Nachteile von XML
- XML in der Bioinformatik
- Zusammenfassung

Motivation

- Praxis: WWW dient als Oberfläche für viele Informationssysteme (Datenbanken, Warenkataloge, Mail-Archive)
- reiche innere Struktur von Dokumenten geht jedoch durch Darstellung mit HTML verloren
- Nutzung der Daten ist nicht mehr möglich, höchstens noch $Cut\ \&\ Paste$ des Textes
- Dies verhindert die Verwendung des Web als Plattform für den Informationsaustausch im großen Rahmen.

Allgemeines

- Name: *Extensible Markup Language* (Erweiterbare Auszeichnungssprache)
- Universalkonzept zur Datenspeicherung
- Aufteilung von Daten in Inhalt, Struktur und Layout
- Entwicklung des W3C (*World Wide Web Consortium*)

XML und SGML

- Teilmenge von SGML (*Standard Generalized Markup Language*), eine Sprache zur Dokumentenrepräsentation, die Struktur von Dokumenten beschreibt (ISO-8879)
- XML ist weniger umfangreich, aber ähnlich leistungsfähig
- abschreckende SGML-Syntax wurde vereinfacht, komplexe und selten verwendete Eigenschaften von SGML entfernt
- zum Vergleich: XML auf 33 Seiten formal definiert, SGML auf mehr als 500

Entwicklung von XML

1994 Tim Berners Lee gründet am MIT ein Entwickler-Forum unter Leitung von Jon Bosak, um Definitionssprache für Auszeichnungssprachen zu entwickeln. Zunächst gab es Konzepte, die SGML außer Acht ließen, später wurde XML selbst in SGML definiert

1996 auf der *SGML 96 Conference* (Boston) wird ein Erstentwurf für XML vorgestellt

1998 erste sog. Empfehlung (*Recommendation*) des W3C für XML; danach Bemühungen, XML zu etablieren und wichtige, bestehende Auszeichnungssprachen (z. B. HTML) in XML zu definieren

Bekannte XML-Anwendungen

- XHTML: Neudefinition von HTML (ursprünglich in SGML definiert)

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html><head>
<title>Einfache (X)HTML-Seite</title>
</head><body>
<h1>Überschrift</h1>
<img alt="Bild" src="bild.png" />
<hr />
</body></html>
```

Bekannte XML-Anwendungen

- MathML: Mathematische Formelsprache
  (1998: MathML 1.0, 2001: MathML 2.0)

```
<msup>
    <mfenced>
        <mi>a</mi> <mo>+</mo> <mi>b</mi>
    </mfenced>
    <mn>2</mn>
</msup>
```

entspricht der Formel: $(a + b)^2$

Bekannte XML-Anwendungen

- CML: *Chemical Markup Language*
- BSML: *Bioinformatic Sequence Markup Language*
  Speicherung von Sequenzdaten (DNA, RNA, Proteine), Struktur und Informationsgehalt ähnlich zu EMBL, GenBank oder DDBJ
- BioML: *Biopolymer Markup Language*

Regeln und Eigenschaften: Allgemeines

- XML-Dokument besteht immer aus einer Instanz und u. U. einer DTD (*Document Type Definition*)
- DTD: eine Definition, in der Tags und ihre erlaubte Schachtelung vereinbart werden
- Instanz: eine XML-Datei, die sich an die vereinbarten Regeln hält

Regeln und Eigenschaften: Allgemeines

- XML-Datei beginnt mit einer *processing instruction*, der XML-Deklaration (zur Unterscheidung von XML- und SGML-Instanzen)
- zur Verwendung einer DTD benötigt man eine *document type declaration*

```
<?xml version="1.0" ?>
<!DOCTYPE dtdname SYSTEM "dtdname.dtd">
```

Regeln und Eigenschaften: Tags

- *Tags* werden als benannte Klammern verwendet, um den Inhalt in Elemente zu gliedern

```
<name>Hans Müller</name>
```

- Starttag (<name>) kennzeichnet den Beginn, ein Endtag (</name>) das Ende eines Elements.

Regeln und Eigenschaften: Tags

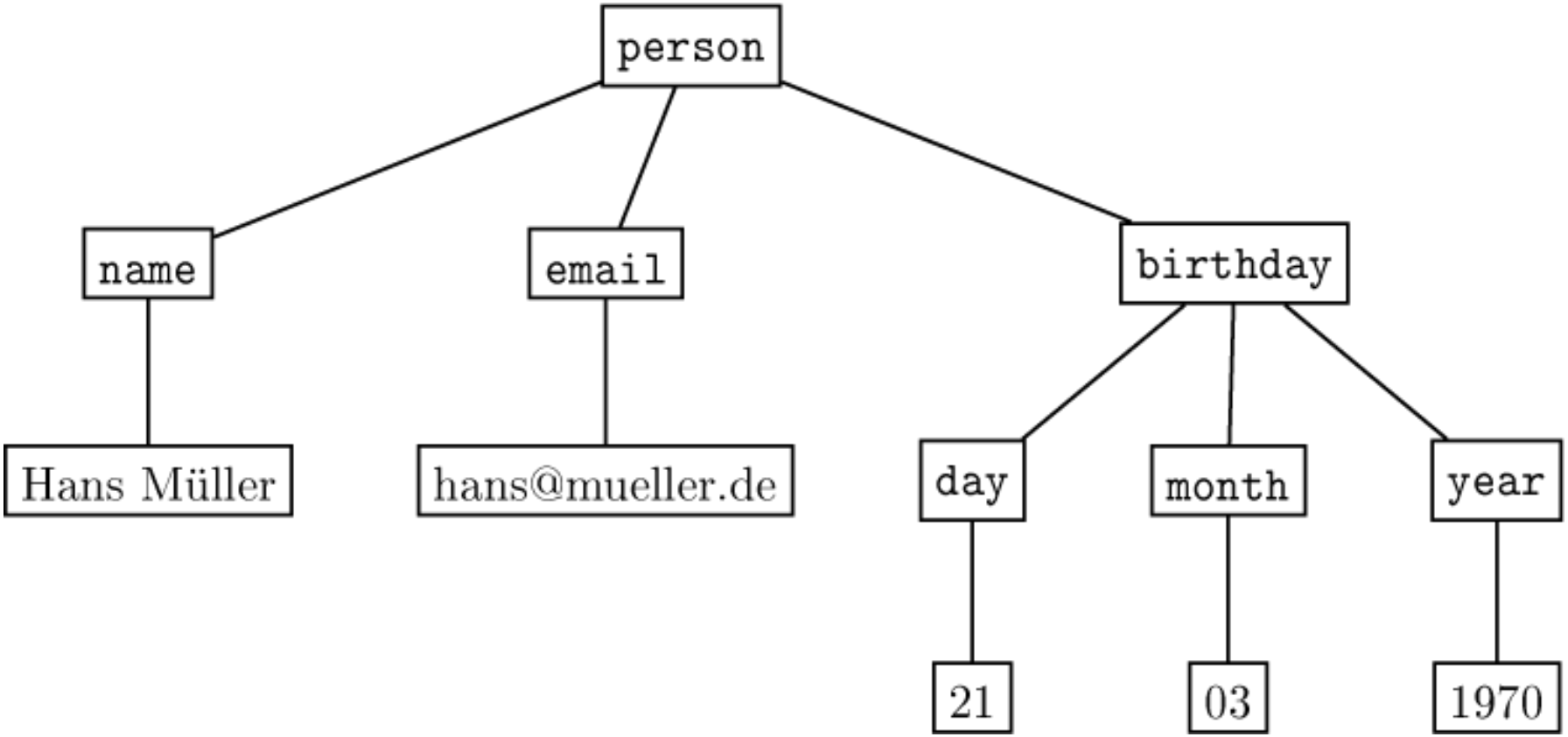
- jedes öffnende Tag **muss** auch wieder mit einem Endtag geschlossen werden: `<tag>Inhalt</tag>`
- Alternative Syntax für Tags ohne Inhalt: `<tag/>` oder `<tag />`
- Elementnamen dürfen nicht mit der Zeichenfolge `xml` und nicht mit Ziffern beginnen; sie dürfen keine =-Zeichen und keine Leerzeichen enthalten

Regeln und Eigenschaften: Tags

- durch Schachtelung von Elementen ergibt sich eine Baumstruktur
- innere Knoten drücken die Struktur aus, Blätter den Inhalt und Elemente mit leerem Inhaltsmodell

```
<person>
    <name>Hans Müller</name>
    <email>hans@mueller.de</email>
    <birthday> <day>21</day><month>03</month><year>1970</year> </birthday>
</person>
```

Regeln und Eigenschaften: Tags



Regeln und Eigenschaften: Kommentare

- Kommentare können beliebigen Text enthalten
- beginnen mit der Zeichenfolge <!-- und enden mit -->
- Kommentar dürfen **nicht** geschachtelt werden

```
<!-- Kommentar -->

<!-- mehrzeiliger
     Kommentar... -->
```

Regeln und Eigenschaften: Attribute

- Tags können mit Attributen versehen werden, durch die sich die Funktion des Tags in der Dokumentinstanz weiter spezialisieren lässt
- Attribute sind Name-/Wert-Paare
- Werte müssen in Anführungszeichen stehen (’ oder ")

Regeln und Eigenschaften: Beispiel

```
<?xml version="1.0"?>
<!DOCTYPE contacts SYSTEM "contacts.dtd">

<contacts type="private" desc="Persönliches Addressbuch">
    <person id="42">
        <name>Hans Müller</name>
        <email>hans@mueller.de</email>
        <birthday>
            <day>21</day><month>03</month><year>1970</year>
        </birthday>
    </person>
</contacts>
```

Regeln und Eigenschaften: Zeichensätze

- Groß- und Kleinschreibung werden unterschieden
- Internationalisierung: alle Zeichen des Unicode-Standards (offizielle Implementierung von ISO-10646) sind erlaubt
- dadurch ist XML z. B. auch für asiatische Sprachen geeignet

Regeln und Eigenschaften: Sonderzeichen

- XML-Sonderzeichen müssen zur Verwendung im normalen Text maskiert werden:

<	<
>	>
&	&
"	"
’	'

Regeln und Eigenschaften: Bezeichnungen

- *well formed*: Regeln für XML werden eingehalten:
  - XML-Deklaration (*processing instruction*)
  - mindestens ein Datenelement
  - genau ein äußerstes Element

- *valid*: Dokument ist wohlgeformt und genügt zusätzlich den Regeln einer DTD

$\Rightarrow$ Dokumente ohne DTD sind möglich

DTD: Allgemeines

- *Document Type Definition*
- formale Grammatik, die eine bestimmte XML-Sprache definiert
- definiert werden Namen der in Dokumentinstanzen erlaubten Attribute, Entitäten sowie Tags und ihre mögliche Schachtelung (Inhaltsmodell)
- nur Syntax, keine semantische Aussage
- besonderer Tag: Dokumenttyp; trägt den selben Namen wie die Sprache und dient als Startsymbol

DTD: Inhaltsmodell

- Bestandteile der Grammatik:
  - andere Tags ($\Rightarrow$ Baumstruktur)
  - `#PCDATA` (*parseable character data*, d. h. die gültigen Zeichen des Alphabets)
  - `EMPTY`: leeres Inhaltsmodell
  - `ANY`: beliebiges Inhaltsmodell
- Verknüpfung dieser Elemente durch Operatoren

DTD: Inhaltsmodell-Operatoren

,	notwendig Elemente in fester Reihenfolge
|	alternative Elemente
?	optionale Elemente
+	1 . . . n Wiederholungen
*	0 . . . n Wiederholungen
()	Klammerung von Ausdrücken

DTD: Einfaches Beispiel

```
<!ELEMENT contacts (person*)>

<!ELEMENT person (name, email*, birthday?)>

<!ELEMENT name  (#PCDATA)>
<!ELEMENT email (#PCDATA)>

<!ELEMENT birthday (year, month, day)>

<!ELEMENT year  (#PCDATA)>
<!ELEMENT month (#PCDATA)>
<!ELEMENT day   (#PCDATA)>
```

DTD: Attribute

- für jedes Element lassen sich erlaubte Attribute angeben
- jedem Attribut ist ein Typ zugeordnet: `CDATA`, `ID`, `IDREF`/`IDREFS`, Liste von Werten
- Attribute sind optional, obligatorisch oder konstant: `#IMPLIED`, `#REQUIRED`, `#FIXED` Wert, Standardwert

DTD: Beispiel mit Attributen

```
<!ELEMENT contacts  (person*)>
<!ATTLIST contacts
    type    (private|work) #IMPLIED
    desc    CDATA>
<!ELEMENT person    (name, email*, birthday?)>
<!ATTLIST person id ID #REQUIRED>

<!ELEMENT name      (#PCDATA)>
<!ELEMENT email     (#PCDATA)>
<!ELEMENT birthday  (year, month, day)>
<!ELEMENT year      (#PCDATA)>
<!ELEMENT month     (#PCDATA)>
<!ELEMENT day       (#PCDATA)>
```

DTD: Entitäten

- mit Entitäten (*entities*) lassen sich Kürzel definieren, die später in der DTD oder der XML-Instanz verwendet werden können
- Entitäten als Textbausteine: Makros zur Verwendung im XML-Dokument mit `&entityname;` (z. B. Umlaute in XHTML)
- *parameter entities*: Makros zur Verwendung in der DTD

DTD: Beispiel für Entitäten

```
<!ENTITY   xml        "Extensible Markup Language">
<!ENTITY   auml       "&#228;">
<!ENTITY % bool       "(true|false)">

<!ELEMENT  person     (name, email*, birthday?)>
<!ATTLIST  person
    id                ID       #REQUIRED
    premiumcustomer   %bool;   "false"
>
<!ELEMENT  birthday   (year, month, day)>
<!ATTLIST  birthday
    sendgreeting      %bool;   #IMPLIED
>
```

Vorteile

- Standardisierung:
  - wichtiger Schritt durch Standardisierung von XML selbst
  - bisherige Datenformate in der Bioinformatik sind proprietär
- Offenheit:
  - es entstehen leicht durchschaubare Auszeichnungssprachen, die von vielen genutzt werden können
  - DTD definiert Syntax

Vorteile

- Einfachheit:
  - schnelle Validierung durch einfache aber strenge Syntax
  - einfache Transformation in anderes Zielformat (XSL-FO, XSLT)
  - einfache Änderungen durch lesbares Textformat (*content scalability*)
- Standardisierte Verarbeitungsmodelle:
  - SAX-Parser: ereignis-basierter Parser
  - DOM-Parser: Operationen auf dem XML-Baum

Vorteile

- Internet-Orientierung:
  - reichhalte Fähigkeiten zur Verlinkung von Daten sind möglich (Entities, XLL)
  - direkte Einbettung von Daten an anderen Adressen

Nachteile

- Daten-Overhead:
  - durch textbasiertes Format größerer Speicherbedarf (25% bis 70%)
  - höhere Übertragungsmenge
- Fehlende Skalierbarkeit:
  - schlechte Performance bei Datenhaltung
  - mögliche Abhilfe durch Kombination mit Datenbanken (XML als Input/Output)

Nachteile

- Limitierte Modellierungsfähigkeiten:
  - beschränkte Datentypisierung: fast nur Strings, keine Zahlen, Tabellen, Matrizen, . . .
  - kein Mechanismus zur Beschreibung von Beziehungen zwischen Elementen (Nachbildung durch `ID` und `IDREF` möglich)
  - mögliche Abhilfe mit XML Schema

XML in der Bioinformatik

- Daten benötigen eine komplexe Modellierung und es gibt viele verschiedene Datentypen
- es entstehen häufig neue Arten von Daten
- Daten werden häufig aktualisiert und von Wissenschaftlern stark übers Internet verbreitet

XML in der Bioinformatik

- verschiedene Typen von Benutzern (Biologen, Programmierer, . . .) brauchen Zugriff auf die Daten

- momentane Situation: bestehende Datenbanken mit uneinheitlichem Format; Daten sind mehrfach vorhanden und nur schwach oder gar nicht miteinander verknüpft

$\Rightarrow$ Vorteile von XML decken diese Aspekte ab, die meisten genannten Nachteile lassen sich umgehen

Vergleich von Lösungen

	XML-Format	Feld/Wert	(OO)DBMS
Ausdruckskraft	**	*	****
Modellierungsfähigkeit	**	*	****
Selbstbeschreibbarkeit	ja	nein	ja
Abfragesprache	ja	nein	ja
Flexibilität	****	*	****
Einfachheit	****	****	**
Skalierbarkeit	**	*	****
Kompatibilität	****	*	***

- (OO)DBMS zur Datenverwaltung, XML als Schnittstelle

Zusammenfassung

- benötigt werden einheitliche XML-Formate für biologische Zwecke
- XML als Schnittstelle geeignet (z. B. standardisierter Austausch zwischen Programmen)
- zur Datenverwaltung zu langsam und wenig skalierbar
- Problem der Modellierungsfähigkeit bleibt bestehen, lässt sich aber möglicherweise durch XML Schema lösen