Horn Logic Markup Languages
Herbrand Terms: Individual Constants, Variables, Flat Ground Structures, ...

Representation of Herbrand terms in XML as `<ind>` and `<struc>` elements (`<var> similar):

- **Individual constant** for channel-tunnel between Britain and France: element `<ind>channel-tunnel</ind>`

- **Ground structure**

  undersea-connection(britain,france)

  is `<struc>` element with embedded element for constructor, followed by elements for argument terms:

  ```xml
  <struc>
    <constructor>undersea-connection</constructor>
    <ind>britain</ind>
    <ind>france</ind>
  </struc>
  ```
Herbrand Terms: ..., Nested Ground Structures

Embedded `<struc>` elements:

```
service-tunnel(
  undersea-connection(
    britain,
    surrounded-country(
      belgium,
      luxembourg,
      germany,
      switzerland,
      italy,
      spain)))
```
Interim Discussion: Tag and Type

- In Prolog not clear, in isolation, that channel-tunnel is individual constant, whereas service-tunnel is constructor.

- In XML, as in strongly typed LP languages, made explicit - however at every occurrence of a symbol.

- Example gives impression of self-description advantage - but also ‘space requirement’ - of this generous application of “syntactic sugar.”
Horn Clauses: Relation Symbol Applications

Predicate or relation symbol in XML is <relator> element. For example, relation symbol travel is
<relator>travel</relator>

Relation symbol application to terms is labeled with <relationship> element.
Application travel(john,channel-tunnel) on two individual constants thus is

<relationship>
    <relator>travel</relator>
    <ind>john</ind>
    <ind>channel-tunnel</ind>
</relationship>
Horn Clauses: Facts

Hence, Horn fact can be asserted as `<hn>` element that possesses `<relationship>` elements as subelements.

In the example, the Prolog fact

travel(john,channel-tunnel).

becomes

```
<hn>
  <relationship>
    <relator>travel</relator>
    <ind>john</ind>
    <ind>channel-tunnel</ind>
  </relationship>
</hn>
```
Horn Clauses: Rules

Then, Horn rule is asserted as \textit{<hn> Element} that has a head \textit{<relationship> element} followed by at least one body \textit{<relationship> element}
So, above example generalized to Prolog rule

\texttt{travel}(\texttt{Someone,channel-tunnel}) :- \texttt{carry(eurostar,Someone)}.

and rewritten as

\texttt{travel}(\texttt{Someone,channel-tunnel}) :- \texttt{carry(eurostar,Someone)}.
Attributes for Extended Logics

Nested elements - trees - allow representation of arbitrary information, but in some situations lead to unnecessarily deeply/widely nested representations

Therefore XML attributes:
Start-Tag is ‘attributed’ with n attribute-value pairs $a_i = v_i$
Element: $\langle \text{tag} \quad a_1 = v_1 \quad ... \quad a_n = v_n \rangle \quad ... \quad \langle /\text{tag} \rangle$

Helpful Prolog uses of XML attributes are arity labelings of relation symbols such as our binary relation symbol travel:
Prolog’s travel/2 in XML with an arity attribute becomes
$\langle \text{relator} \quad \text{arity} = "2" \rangle \text{travel} \langle /\text{relator} \rangle$

Analogously, annotations become possible on arbitrary element levels: mode declarations for logic variables, determinism specifications for clauses or procedures, and context conditions for entire knowledge bases
**ID and IDREF**

Attribute types ID and IDREF for naming and referencing of elements

ID-typed value must uniquely identify an element and IDREF-typed value must occur as ID value of an element

E.g., clause can be named (in a hypothesis knowledge base):

```xml
<hn id="john-channel">
  <relationship>
    <relator>travel</relator>
    <ind>john</ind>
    <ind>channel-tunnel</ind>
  </relationship>
</hn>
```
Now a “modal Prolog fact”
belief(mary,travel(john,channel-tunnel)).
can access the "john-channel" assertion:

```xml
<hn>
  <relationship>
    <relator>belief</relator>
    <ind>mary</ind>
    <prop idref="john-channel"/>
  </relationship>
</hn>
```

Propositional argument of the belief operator written as `<prop idref="john-channel"/>` (XML abbreviation of empty elements `<tag .../>` to `<tag .../>`)

Also disbelief fact has "john-channel" access with idref: ID/IDREF “break out of the tree” and enable ‘sharing’
Up to now: Examples for Horn Logic in XML etc.
Now: General language definition

XML's Document type definitions (DTDs) initially only as ELEMENT declarations for non-attributed elements

For nonterminals: DTD \(\approx\) ordinary context-free grammar in modified (EBNF) notation

For terminals: Usually arbitrary permutations of the base alphabet ("PCDATA") instead of fixed terminal sequences

DTD grammar derives context-free word patterns: derivation trees themselves - linearized through brackets - as generated result

XML element is valid with respect to DTD: can be generated from DTD as linearized derivation tree
Syntactic ELEMENT declaration of Horn logic as a knowledge base (kb) of zero or more Horn clauses (hn*):

<!ELEMENT kb (hn*) >
<!ELEMENT hn (relationship, relationship*) >
<!ELEMENT relationship (relator, (ind | var | struc)*) >
<!ELEMENT struc (constructor, (ind | var | struc)*) >
<!ELEMENT relator (#PCDATA) >
<!ELEMENT constructor (#PCDATA) >
<!ELEMENT ind (#PCDATA) >
<!ELEMENT var (#PCDATA) >

Note struc recursion!
DTDs: Generation of the Example Rule (1)

(Start-)symbol kb brackets derived clause(s) as linearized start-tag/end-tag-tree representation `<kb> . . . </kb>`:

```
kb

`<kb> hn* </kb>
. . .
`<kb> <hn> relationship relationship </hn> </kb>
. . .
`<kb>    <hn>
    <relationship>
    `<relator>`PCDATA`<relator> `<var>`PCDATA`<var> `<ind>`PCDATA`<ind> `</relationship`
    `</relationship`
    `<relationship>
    `<relator>`PCDATA`<relator> `<ind>`PCDATA`<ind> `<var>`PCDATA`<var> `</relationship`
    `</relationship`
    `<hn>`
`</kb>
```
<kb>
  <hn>
    <relationship>
      <relator>travel</relator>
      <var>someone</var>
      <ind>channel-tunnel</ind>
    </relationship>
    <relationship>
      <relator>carry</relator>
      <ind>eurostar</ind>
      <var>someone</var>
    </relationship>
  </hn>
</kb>
DTDs for **attributed elements**: *ATTLIST declarations*, which associate element name with an attribute name plus attribute type and possible occurrence indication

1st Example: declare the relator attribute arity as CDATA-typed (cf. #PCDATA) and occurring optionally (#IMPLIED):

```xml
<!ATTLIST relator arity CDATA #IMPLIED >
```

2nd Example (Preparation): define the extended Horn logic with (named hn clauses and) embedded propositions:

```xml
<!ELEMENT relationship (relator, (ind|var|struc|prop)*) >
<!ELEMENT prop EMPTY >
```
2nd Example (Execution): Append an ATTLIST declaration that specifies, for the hn respectively prop elements, the attributes id - as optional ID type - respectively idref - as mandatory IDREF type:

`<!ATTLIST hn id ID #IMPLIED >`

`<!ATTLIST prop idref IDREF #REQUIRED >`

With entire DTD now, e.g., earlier "john-channel"-named fact and its accessing facts can be generated
Assume fact:

```xml
<hn>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>
    <ind>fred</ind>
  </relationship>
</hn>
```

carry(eurostar,fred).

A Horn-logic interpreter can use it to answer this query:

```xml
<relationship>
  <relator>carry</relator>
  <ind>eurostar</ind>
  <var>someone</var>
</relationship>
```

by binding

```xml
<var>someone</var> Someone
to
<ind>fred</ind> fred
```
With carry basis fact

\[ \text{carry(eurostar,fred).} \]

a rule is usable to dynamically derive travel assertions as needed, without having to store them all-inclusively and statically:

\[ \text{travel(\text{Someone,channel-tunnel}) :- carry(eurostar,\text{Someone}).} \]

That is, its earlier XML version is useable by a Horn-logic interpreter for inferential queries like

\[
\text{travel(\text{fred,Where}) } \Rightarrow \text{ carry(eurostar,\text{Someone}) } \Rightarrow \text{ true}
\]

\[
\begin{align*}
\text{Someone=fred} \\
\text{Where=channel-tunnel}
\end{align*}
\]
Horn Inferences in XML Notation (2)

Rule:

<hn>
  <relationship>
    <relator>travel</relator>
    <var>someone</var>
    <ind>channel-tunnel</ind>
  </relationship>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>
    <var>someone</var>
  </relationship>
</hn>

Fact:

<hn>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>
    <ind>fred</ind>
  </relationship>
</hn>

3-Step Animation:

<ind where="channel-tunnel">
  true
</ind>
Horn Inferences in XML Notation (2)

Rule:
<hn>
  <relationship>
    <relator>travel</relator>
    <var>someone</var>
    <ind>channel-tunnel</ind>
  </relationship>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>
    <var>someone</var>
  </relationship>
</hn>

Fact:
<hn>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>
    <ind>fred</ind>
  </relationship>
</hn>

3-Step Animation:

<relationship>
  <relator>travel</relator>
  <ind>fred</ind>
  <var>where</var>
</relationship>
Horn Inferences in XML Notation (2)

Rule:

<hn>
  <relationship>
    <relator>travel</relator>
    <var>someone</var>
    <ind>channel-tunnel</ind>
  </relationship>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>
    <var>someone</var>
  </relationship>
</hn>

Fact:

<hn>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>
    <ind>fred</ind>
  </relationship>
</hn>

3-Step Animation:

<relationship someone="fred" where="channel-tunnel">
  <relator>carry</relator>
  <ind>eurostar</ind>
  <var>someone</var>
</relationship>
Horn Inferences in XML Notation (2)

Rule:

<hn>
  <relationship>
    <relator>travel</relator>
    <var>someone</var>
    <ind>channel-tunnel</ind>
  </relationship>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>
    <var>someone</var>
  </relationship>
</hn>

Fact:

<hn>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>
    <ind>fred</ind>
  </relationship>
</hn>

3-Step Animation:

<ind where="channel-tunnel">
  true
</ind>
Horn Inferences: SLD-Resolution, XML-QL Implementation, Open World

- This inference is carried out as an SLD-resolution step
- The procedural semantics of SLD-resolution can be used
- An XML-QL implementation seems possible as for queries
- A functional-logic generalization of HornML is RFML

If distribution of the clauses over different documents in the Web is assumed, in this “open world” logical completeness, in particular, can hardly still be asked for