Horn Logic Markup Languages

HornML

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:

```
<struc>
  <constructor>undersea-connection</constructor>
  <ind>britain</ind>
  <ind>france</ind>
</struc>
```

HorML

Herbrand Terms: ..., Nested Ground Structures

Embedded <struc> elements:

```
service-tunnel(
  undersea-connection(
    britain,
    surrounded-country(
      belgium,
      luxembourg,
      germany,
      switzerland,
      italy,
      spain)))
```

```
<struc>
  <constructor>service-tunnel</constructor>
  <struc>
    <constructor>undersea-connection</constructor>
    <ind>britain</ind>
    <struc>
      <constructor>surrounded-country</constructor>
      <ind>belgium</ind>
      <ind>luxembourg</ind>
      <ind>germany</ind>
      <ind>switzerland</ind>
      <ind>italy</ind>
      <ind>spain</ind>
    </struc>
  </struc>
</struc>
```

HornML

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”

HornML

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>
```

HorML

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>
```

HornML

Horn Clauses: Rules

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

travel(Someone,channel-tunnel) :- carry(eurostar,Someone).

and rewritten as

```
<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>
```

HornML

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: <*tag* a_1=v_1 ... a_n=v_n> . . . </*tag*>

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
<relator arity="2">travel</relator>

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

HorML

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):

```
<hn id="john-channel">
  <relationship>
    <relator>travel</relator>
    <ind>john</ind>
    <ind>channel-tunnel</ind>
  </relationship>
</hn>
```

HornML

ID and *IDREF*

Now a “modal Prolog fact”

belief(mary,travel(john,channel-tunnel)).

can access the "john-channel" assertion:

```
<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* ...> </*tag*> to <*tag* .../>)

Also disbelief fact has "john-channel" access with idref: ID/IDREF “break out of the tree” and enable ‘sharing’

HornML

DTDs: Elements as Derivation Trees

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 ≈ 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

DTDs: Defining Horn Logic in XML

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!

HornML

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>
   <relator>#PCDATA</relator> <ind>#PCDATA</ind> <var>#PCDATA</var>
  </relationship>
 </hn>
</kb>

HornML

DTDs: Generation of the Example Rule (2)

```
<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>
```

HornML

Attribute DTDs (1)

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):

<!ATTLIST relator arity CDATA #IMPLIED >

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

<!ELEMENT relationship (relator, (ind|var|struc|prop)*) >

<!ELEMENT prop EMPTY >

HornML

Attribute DTDs (2)

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

HorNM L

Horn Queries in XML Notation

Assume fact:

```
<hn>
  <relationship>
    <relator>carry</relator>
    <ind>eurostar</ind>          carry(eurostar,fred).
    <ind>fred</ind>
  </relationship>
</hn>
```

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

```
<relationship>
  <relator>carry</relator>
  <ind>eurostar</ind>            carry(eurostar,Someone)
  <var>someone</var>
</relationship>
```

by binding

```
<var>someone</var>             Someone
```

to

```
<ind>fred</ind>                fred
```

HornML

Horn Inferences in XML Notation (1)

With carry basis fact

carry(eurostar,fred).

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

travel(Someone,channel-tunnel) :- carry(eurostar,Someone).

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

travel(fred,Where) ⇒ carry(eurostar,Someone) ⇒ true

Someone=fred

Where=channel-tunnel

Where=channel-tunnel

HornML

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>
```

HorNML

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>
```

HorNML

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>
```

HorML

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>
```

HorML

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

HornML