A Logical Framework for XML Reference Specification

C. Combi, A. Masini, B. Oliboni, and M. Zorzi(✉)

Department of Computer Science – University of Verona,
Cà Vignal 2, Strada le Grazie 15, 37134 Verona, Italy
{carlo.combi,andrea.masini,barbara.oliboni,margherita.zorzi}@univr.it

Abstract. In this paper we focus on a (as much as possible) simple logic, called XHyb, expressive enough to allow the specification of the most common integrity constraints in XML documents. In particular we will deal with constraints on ID and IDREF(S) attributes, which are the common way of logically connecting parts of XML documents, besides the usual containment relation of XML elements.

1 Introduction

XML (eXtensible Markup Language) is the main mark up language used for representing data to exchange on the Web and for data integration. XML allows one to represent structured and semistructured data through a hierarchical organization of mark up elements. An XML document is typically endowed with a DTD (Data TypeDefinition). DTDs allow the specification in a simple and compact way of the main structural features of XML documents. DTDs easily express hierarchies, order between elements, and several types of element attributes. In particular, the ID/IDREF mechanism of DTDs describes identifiers and references in a similar (but not equivalent) way to keys and foreign keys in a relational setting. The value of an attribute of type ID uniquely identifies an element among all the elements of the entire document; the value of an attribute of type IDREF(S) allows the reference to element(s) on the base of their ID values. DTD simplicity is paid in terms of expressiveness: a DTD efficiently models the structure of XML documents (it is able to provide a "syntactical" control such as context-free grammar), but it is not powerful enough for capturing subtle, semantic features. As an example, (unique) values of ID attributes have the overall document as a scope. Consequently, attributes of type IDREF(S) cannot be constrained to refer to only a subset of elements. Complex specification languages such as XML Schema [9] represent a powerful alternative to DTD: XML Schema supports the specification of a very rich set of constraints (in terms of XPath expressions) and seems to overcome DTD issues and limitations. Unfortunately, as observed in [3,4], XML Schema is too complicate and not compact at all in the specification of even simple integrity constraints.

In this paper we focus on the issue of retaining in a logical framework the simplicity of DTDs with the capability of expressing meaningful integrity

Q. Chen et al. (Eds.): DEXA 2015, Part II, LNCS 9262, pp. 258–267, 2015.
DOI: 10.1007/978-3-319-22852-5_22

constraints. In this context, some interesting theoretical solutions have been proposed [3,4]. With respect to previous proposals, the novelty of our work is that we look specifically for a very simple formal language which is able to model constraints with respect to XML reference specification. In this direction, we propose a logical language, called XHyb, able to express *in a direct and explicit way* constraints on XML documents.

2 Motivating Example

In this paper we will use the DTD shown in Fig. 1 as a running example. The considered DTD describes a subset of information related to the university domain. It represents the fact that a university is composed of many students, professors, courses, and examinations; a student may have a supervisor, when she starts her thesis work; a professor may act as both thesis supervisor and thesis reviewer.

```
<!ELEMENT university (student*,professor*,course*,examination*)>
<!ELEMENT student (name,surname,supervisor?)>
<!ELEMENT professor (name,surname,thesis_stud?,thesis_reviewer?)>
<!ELEMENT course (title)>
<!ELEMENT examination (mark)>

<!ELEMENT name      (#PCDATA)>
<!ELEMENT surname   (#PCDATA)>
<!ELEMENT title     (#PCDATA)>
<!ELEMENT mark      (#PCDATA)>
<!ELEMENT supervisor        EMPTY>
<!ELEMENT thesis_stud       EMPTY>
<!ELEMENT thesis_reviewer   EMPTY>

<!ATTLIST student         stud_id   ID      #REQUIRED>
<!ATTLIST professor       prof_id   ID      #REQUIRED>
<!ATTLIST course          cour_id   ID      #REQUIRED
                          prof_ref  IDREF   #REQUIRED>
<!ATTLIST examination     stud_ref  IDREF   #REQUIRED
                          cour_ref  IDREF   #REQUIRED>
<!ATTLIST supervisor      prof_ref  IDREF   #REQUIRED>
<!ATTLIST thesis_stud     stud_refs IDREFS  #REQUIRED>
<!ATTLIST thesis_reviewer prof_ref IDREFS  #REQUIRED>
```

Fig. 1. An example of DTD for XML documents.

The link between a student and her supervisor is modeled by using attribute `prof_ref` of type `IDREF` within element `supervisor` (which is contained in element `student`). On the other side, the corresponding link between a professor and her thesis students is modeled by means of attribute `stud_refs` of type `IDREFS` within element `thesis_stud`. Both attributes `supervisor` and `stud_refs` refer to elements identified by a suitable attribute of type `ID`. It is worth noting that the DTD grammar does not allow us, for example, to constrain the value of a `prof_ref` to correspond to the value of attribute `prof_id` of some element `professor`.

In general, DTD grammar allows us only to validate containment relations (restriction on the element structure of the document [5]) and links between

IDREF/IDREFS values and ID values within the whole document. Thus, many domain-related constraints cannot be explicitly modeled and some XML documents could be valid according to the given DTD but provide meaningless information (such as, for example, that a thesis reviewer is a course).

In Fig. 2 we report an example of XML document valid against DTD in Fig. 1. Let us consider in the following some examples of requirements we would like to represent and verify in XML documents related to the university domain.

– The supervisor of a student must be a professor.
– A professor may be the supervisor of one or more students.
– A student can be evaluated only once for a given course.

These constraints are clearly not expressible by DTD, as well as more complex constraints such us the following ones, which require to linguistically express the interplay between containment relation and reference constraints specification:

– A professor cannot be both supervisor and reviewer of the same student.
– A professor can be supervisor only for students that attended and passed a course she taught.

In the following section we will introduce the *referential logic* XHyb. In XHyb it is possible to encode constraints in terms of (as much as possible) simple modal formulas. Moreover, the Kripke-style XHyb models naturally fit the shape of XML documents, representing explicitly and distinctly both containment relation and reference specification. The formal description of the relationship XML-documents/XHyb and the encoding in XHyb of the constraints above are in Sects. 4 and 5 respectively.

3 XHyb: Hybrid Logic for XML Reference Constraints

Logic XHyb is an extension of a fragment of *hybrid logic* [1,2], obtained by adding to the syntax the operator $@_a$ (where a ranges over a particular set of variables, called nominal variables): $@_a$ is the hybrid *at* operator and it provides a direct access to the state (uniquely) named by a. The peculiar feature of XHyb is the extension of quantified hybrid logic by means of a new modal operator $*_c$, which explicitly captures the presence of ID/IDREF(S) relation between elements of XML documents.

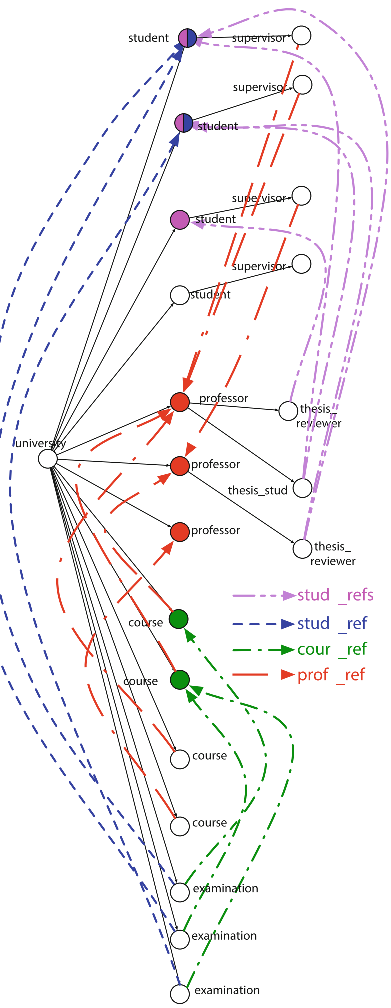
3.1 Syntax

The alphabet of XHyb is built out of some sets of symbols for constants and variables. We define three distinct sets of constants:
$E = e_0, \ldots, e_k$ is a finite set of *element names*; $R = r_0, \ldots, r_p$: a finite set of *reference names*; $C = c_0, \ldots, c_m$ is a finite set of *colors*.

In the following we will use symbols e, f for element names, r, s for reference names, and c, d for color names, possibly in their indexed version. We assume C, E, R are pairwise disjoint.

```
<university>
  <student stud_id="stud1">
    <name> Al </name> <surname> Jones </surname>
    <supervisor prof_ref="prof1"/>
  </student>
  <student stud_id="stud2">
    <name> Sue </name> <surname> Storme </surname>
    <supervisor prof_ref="prof1"/>
  </student>
  <student stud_id="stud3">
    <name> Bill </name> <surname> Taylor </surname>
    <supervisor prof_ref="prof2"/>
  </student>
  <student stud_id="stud4">
    <name> Sam </name> <surname> Evans </surname>
  </student>
  <professor prof_id="prof1">
    <name> Ted </name> <surname> Wilson </surname>
    <thesis_stud stud_refs="stud1 stud2"/>
    <thesis_reviewer stud_refs="stud3"/>
  </professor>
  <professor prof_id="prof2">
    <name> May </name> <surname> West </surname>
    <thesis_reviewer stud_refs="stud1 stud2"/>
  </professor>
  <professor prof_id="prof3">
    <name> Ed </name> <surname> Smith </surname>
  </professor>
  <course cour_id="cour1" prof_ref="prof2">
    <title> Information Systems </title>
  </course>
  <course cour_id="cour2" prof_ref="prof1">
    <title> Computability </title>
  </course>
  <course cour_id="cour3" prof_ref="prof1">
    <title> Logics </title>
  </course>
  <course cour_id="cour4" prof_ref="prof3">
    <title> Databases </title>
  </course>
  <examination stud_ref="stud1" cour_ref="cour1">
    <mark>A</mark>
  </examination>
  <examination stud_ref="stud1" cour_ref="cour2">
    <mark>B</mark>
  </examination>
  <examination stud_ref="stud2" cour_ref="cour2">
    <mark>A</mark>
  </examination>
</university>
```

(a)



(b)

Fig. 2. An XML document valid against the DTD in Fig. 1 and its graphical representation (reference colors are represented through lines with both colors and different dashes) (Color figure online).

Set PROP of propositional symbols is the union of the above sets, i.e. PROP $= C \cup E \cup R$.

Moreover, we define the following sets of variables for *nominals* and *sequences of nominals*:

$N = i_0, i_1, \ldots$ is a denumerable set of nominals; $\Gamma = \gamma_0, \gamma_1, \ldots$ is a denumerable set of variables for finite sequences of nominals. We assume that $\Gamma \cap N = \emptyset$ and $\Delta = N \cup \Gamma$.

In the following we will use symbols i, j, l for nominals, γ, δ for sequences of nominals, and x, y for nominals/nominal sequences, possibly in their indexed version.

Set Θ of *terms* τ is the smallest set Y defined by stipulating that: $N \subseteq Y$; $\Gamma \subseteq Y$; if $\tau', \tau'' \in Y$ then $\tau'\tau'' \in Y$.

We equip the language of XHyb with logical connectives $\rightarrow, \perp, \forall, \in, *_c, @_a, \Box$ and $\bigcirc^{\forall}$. Formulas are built out of the set of terms by means of logical connectives. Formally, the set Z of *well-formed formulas* (only *formulas* in the following, ranged by A, B, C possibly indexed), is the smallest set Y such that: $N \subseteq Y$; $\mathsf{PROP} \subseteq Y$; **if** $i \in N$ **and** $A \in Y$ **then** $(@_i.A) \in Y$; **if** $i \in N$ **and** $\tau \in \Theta$ **then** $(i \in \tau) \in Y$; **if** $\tau \in \Theta$ and $c \in C$ **then** $*_c(\tau) \in Y$; **if** $i \in N$ **and** $A \in Y$ **then** $(\forall i.A) \in Y$; **if** $\gamma \in \Gamma$ **and** $A \in Y$ **then**$(\forall\gamma.A) \in Y$; **if** $A, B \in Y$ **then** $(A \rightarrow B) \in Y$; $\perp \in Y$; **if** $A \in Y$ **then** $\Box A, \bigcirc^{\forall} A \in Y$.

Connectives $\rightarrow, \perp, \forall$ are defined in the usual way. The intuition about the other connectives is as follows:

– $@_i A$ means that formula A holds at state i. Following hybrid logic tradition, equality between two worlds i and j is represented as $@_i j$;
– $\Box A$ means that A holds at the current state and at all the descendant states;
– $\bigcirc^{\forall} A$ means that A holds in each children of the current state;
– $*_c$ is the *reference operator*: if $*_c(i)$ holds in a given state, then there exists a reference, labelled by c, to state i.

Notation 1. *In the rest of the paper, we will use the following (quite standard) abbreviations:* $\neg A$ *stands for* $A \rightarrow \perp$*;* $A \vee B$ *stands for* $(\neg A) \rightarrow B)$*;* $A \wedge B$ *stands for* $\neg(A \vee B)$*;* $\bigwedge_k A(k)$ *stands for* $A(c_0) \wedge (A(c_1) \wedge (\cdots \wedge A(c_k))))$*;* $\bigvee_k A(k)$ *stands for* $A(c_0) \vee (A(c_1) \vee (\cdots \vee A(c_k))))$*;* $\exists i.A$ *stands for* $\neg(\forall i.(\neg A))$*;* $\exists\gamma.A$ *stands for* $\neg(\forall\gamma.(\neg A))$*;* $\bigcirc^{\exists} A$ *stands for* $\neg \bigcirc^{\forall} \neg A$*;* $\Diamond A$ *stands for* $\neg\Box\neg A$*.*

In the following, given γ *and* γ' *sequences of nominals, we will write* $\gamma \subseteq \gamma'$ *for* $\forall i.(i \in \gamma \rightarrow i \in \gamma')$*. We will always omit the most external parentheses in formulas. Moreover we will adopt useful precedence between operators in order to simplify the readings of formulas, in particular we stipulate that* $\neg, \forall, @, \Box, \bigcirc^{\forall}$ *have the higher priority.*

The only binder for variables is $\forall$*. Therefore, the definition of the set of free variables in terms and formulas is standard.*

Definition 1 (Free and Bound Variables). *The set FV of names of free variables for terms and formulas is inductively defined as follows:* $FV[i] = \{i\}$*;* $FV[\tau'\tau''] = FV[\tau'] \cup FV[\tau'']$*;* $FV[@_i.A] = \{i\} \cup FV[A]$*;* $FV[i \in \tau] = \{i\} \cup FV[\tau]$*;* $FV[*_c(\tau)] = FV[\tau]$*;* $FV[\gamma] = \{\gamma\}$*;* $FV[\forall i.A] = FV[A] - \{i\}$*;* $FV[\forall\gamma.A] = FV[A] - \{\gamma\}$*;* $FV[A \rightarrow B] = FV[A] \cup FV[B]$*;* $FV[\perp] = \emptyset$*;* $FV[p] = \emptyset$ *for* $p \in \mathsf{PROP}$*;* $FV[\Box A] = FV[A]$*;* $FV[\bigcirc^{\forall} A] = FV[A]$*.*

An occurrence of i *(of* γ*) in a formula* A *is bound iff there is a sub-formula of* A *of the kind* $C = \forall i.B$ *(*$C = \forall\gamma.B$*). In this case we say also that* B *is the* *scope of i (of γ). We say that an occurrence of i (of γ) in a formula A is free iff it is not bound.*

3.2 Semantics

Definition 2 (Frames). *A* ***structure*** *is a tuple $S = \langle W, V_E, V_C, V_R, \curlyvee, \mathcal{N}\rangle$ where:$|W| < \aleph_0$ is a set of* worlds*; $V_E : E \to 2^W$, $V_C : C \to 2^W$, $V_R : R \to 2^W$ and $V : \mathsf{PROP} \to 2^W$ is defined as $V = V_E \cup V_C \cup V_R$; $\curlyvee$: $W \to 2^W$ is the* reference relation*. $\mathcal{N} \subseteq W \times W$ is the relation father-son.*

An ***interpretation*** *is a tuple $\mathcal{I} = \langle S, g, h, w\rangle$ where S is a structure, $g : N \to W$, $h : \Gamma \to 2^W$, and $w \in W$.*

Informally, the reference relation maps a world w into the sets of worlds the state w "points to". As usual, we will denote by $\mathcal{N}^*$ the transitive and reflexive closure of $\mathcal{N}$.

Definition 3 (Satisfaction). *The satisfiability relation $\mathcal{I} \models A$ is defined in the following way:*

1. $S, g, h, w \not\models \bot$
2. $S, g, h, w \models p \Leftrightarrow w \in V(p)$ *with* $p \in \mathsf{PROP}$
3. $S, g, h, w \models i \Leftrightarrow w = g(i)$
4. $S, g, h, w \models *_c(x_1 \dots x_n) \Leftrightarrow$
   $V = \{v | v \in (g \cup h)(x_1) \cup \dots \cup (g \cup h)(x_n)\} \subseteq \curlyvee (w)$,
   $\forall v \in V, S, g, h, v \models c$;
5. $S, g, h, w \models @_i.A \Leftrightarrow S, g, h, g(i) \models A$
6. $S, g, h, w \models \forall i.A \Leftrightarrow \forall v \in W, S, g[i \mapsto v], h, w \models A$
7. $S, g, h, w \models \forall \gamma.A \Leftrightarrow \forall M \in 2^W, S, g, h[\gamma \mapsto M], w \models A$
8. $S, g, h, w \models A \to B \Leftrightarrow S, g, h, w \not\models A$ ***or*** $S, g, h, w \models B$
9. $S, g, h, w \models \Box A \Leftrightarrow \forall v \in W (w\mathcal{N}^* v \Rightarrow S, g, h, v \models A)$;
10. $S, g, h, w \models \bigcirc^{\forall} A \Leftrightarrow \forall v \in W (w\mathcal{N} v \Rightarrow S, g, h, v \models A)$;

If $S, g, w \models A$ we say that $\langle S, g, h, w\rangle$ satisfies A.

We say that: A **is satisfiable** if there exists $\mathcal{I}$ s.t. $\mathcal{I} \models A$; S **is a model** of $A](S \models A)$ if for each g, h, w, $S, g, h, w \models A$; ***A is valid*** ($\models A$) if for each S, $S \models A$; ***A is semantical consequence of a finite set Σ of formulas*** ($\Sigma \models A$) if $\forall \mathcal{I}((\forall B \in \Sigma.\mathcal{I} \models B) \Rightarrow \mathcal{I} \models A)$.

Let us now briefly focus on the semantics of XHyb particular connectives. The meaning of a formula $@_i A$ is defined by stipulating that A holds in a world w if and only if $w = g(i)$, i.e. the interpretation by g of the nominal i is exactly w. The meaning of a formula $*_c(x_1 \dots x_n)$ is defined upon the relation $\curlyvee$. $*_c(x_1 \dots x_n)$ holds in a world w if and only if the interpretation by g or h of variable x_i ($i = 1, \dots, n$) (for nominals or sequences of nominals) belongs to the set of worlds w points to according to $\curlyvee$. Moreover, the proposition $c \in C$ holds in each $v = (g \cup h)(x_i)$ for some $i = 1, \dots, n$.

Table 1. From XHyb to XML

XHyb constructs	XML interpretation
C (Colors)	IDREF(S) attribute declared in the DTD
E (Element names)	Tag names declared in the DTD
R (Identifier Names)	ID attributes declared in the DTD
W (Worlds)	Values of ID attributes in the XML document
$V_E : E \rightarrow 2^W$	Each element name e is mapped to the set of ID values identifying occurrences of e
$V_C : C \rightarrow 2^W$	Each attribute name of type `IDREF(S)` is mapped to the set of ID values referenced by values of the given attribute
$V_R : R \rightarrow 2^W$	Each attribute name of type `ID` is mapped to the set of corresponding ID values in the document
$\mathcal{N} : W \rightarrow 2^W$	Containment relation (parent-child relation)
$\curlyvee : W \rightarrow 2^W$	Each attribute name of type `ID` is mapped to the set of corresponding ID values in the document

4 From XML to XHyb

In this section we describe the relationship between the XHyb logic and XML documents. In Table 1 we summarize the XML interpretation of XHyb, by providing a simple mapping between XHyb syntactical and semantic objects and the corresponding meaning in the XML document.

It is mandatory to say that the tree-like structure of XML documents naturally fits the shape of (most) modal/temporal logic Kripke models. This has been observed and exploited in [6,7]. In this paper we start from the same observation, maintaining a slightly different viewpoint. Given an XML document, we will adopt the (quite) standard graph-representation (see e.g. [3]), but we choose a bit more informative graphical depiction:

– we represent XML elements as nodes, labeled with the element name and, when explicitly required, the ID attribute;
– black edges represent the containment relation;
– colored edges represent the presence of an ID/IDREF(S) link;
– nodes pointed by colored edges are colored accordingly.

More formally, the overall structure of an XML document may be represented as in the following.

Definition 4 (Colored XGraph, Xtree and colored Xstructure). *A colored XGraph is a tuple $C\mathcal{G} = \{P, E, r, Col, E_{Col}, l_v\}$ such that: P is a set of nodes and r is a particular node called* root*; E is a set of ordered pairs of nodes where, for all $v \in P - \{r\}$, there exists a node $u \in P$ such that $(u, v) \in E$ and if $(u_1, v) \in E$ and $(u_2, v) \in E$ then $u_1 = u_2$; Col is a set of color labels; l_e is a labeling function $l_e : P \rightarrow Col$. E_{Col} is a set of pairs $((u, v), c)$ where (u, w) is an ordered pair of nodes, $c \in Col$ and if $((u, w), c) \in E_{Col}$ then $l_v(w) = c$.*

Table 2. XHyb overall picture

Connectives	$*_c$	$\Box, \bigcirc^{\forall}$	$*_c + \Box, \bigcirc^{\forall}$
Relations/constraints	References	Containment	References + Containment
Shape of the models	colored Xstructure	Xtree	colored Xgraph

– Xtree *is the substructure* $\{P, E\}$*;*
– colored Xstructure *is the substructure* $\{P, E_{Col}, Col, l\}$.

The introduction of colored Xgraphs allows us to represent at the same time both the containment relation and the accessibility relation (through references) between nodes. This is possible since in XHyb IDREF(s) attributes are explicitly denotable (thanks to the reference operator $*_c$) and their linguistic treatment is completely independent from the denotation of the containment relation (Table 2). Our graphical representation reflects the way the syntax and the semantics of XHyb are defined. In particular, we can stipulate a bijection between the set of color labels Col and the IDREF(s) declaration in the DTD and so with the set of constant C.

Let us now sketch the translation of the DTD University Record and the XML documents proposed in Fig. 2 into the Referential Logic XHyb. We will actually build a concrete alphabet for the XHyb language and a related semantical model by processing the content of the DTD and the XML instance. Intuitively, this can be achieved by reading right-left Table 1 and building step-by-step propositional symbols (the constants of the logic) and a semantical structure (actually a colored Xgraph: a set of nodes equipped with two distinct accessibility relations). Notice that we need both the DTD and the XML instances, since names of elements and attributes (in particular ID and IDREF(S) attributes) can be "statically" determined from the DTD, whereas element occurrences, ID values, and IDREF(s) values can be only "dynamically" extracted from to the XML instance.

In Fig. 2.(b) we propose the graphical representation of the XML document reported in Fig. 2.(a), which is valid against the DTD in Fig. 1. We assume that red, blue, green and pink represent the attributes `prof_ref`, `stud_ref`, `cour_ref` and `stud_refs` respectively. As an example, consider a node `professor`. It is red (i.e., it has the same color of link `prof_ref`), since it is pointed by a node `supervisor` through a (red) IDREF `prof_ref`. Any attribute IDREF corresponds, in XHyb, to a propositional symbol: in the example, `prof_ref` belongs to set C of colors and thus to set PROP. By Definition 3, it is possible to see where propositional symbol/color `prof_ref` holds. The presence of the IDREF relation between supervisor and professor can be easily encoded as $*_{\texttt{prof\_ref}}(\texttt{professor})$. This formula clearly holds in a node (a world) `supervisor`, i.e., we can state (forgetting about interpretation) $\texttt{supervisor} \models *_{\texttt{prof\_ref}}(\texttt{professor})$. Following Definition 3, Case 4, clearly $\texttt{professor} \models \texttt{prof\_ref}$.

Summing up, the way the logic has been defined allows: (i) to express reference constraints in terms of (simple) XHyb formulae, overcoming DTDs expressive limitations. Some interesting examples related to the university domain are provided in Sect. 5, and (ii) to map an XML document into an XHyb (Kripke-like) model. This does not only confirm that XHyb is a suitable formalism to reason about XML, but it also represents the first step toward the static automated verification of XML constraints.

5 Expressing XML Constraints by XHyb

We provide now an XHyb encoding of some interesting constraints (non-expressible by DTD) that must hold for the XML document reported in Fig. 2. In the following, i, j, k, m, n are variables for nominals and γ is a variable for a finite sequence of nominals.

1. The supervisor of a student must be a professor.
   Attribute `prof_ref` *of element* `supervisor` *must refer to an element* `professor`.

$$\forall i((\texttt{supervisor} \wedge *_{\texttt{prof\_ref}}(i)) \rightarrow @_i\texttt{professor})$$

2. A professor may be the supervisor of one or more students.
   When `thesis_stud` *appears, its attribute* `stud_refs` *must refer to at least one* `student` *element.*

$$\texttt{thesis\_stud} \rightarrow \exists\gamma.(*_{\texttt{stud\_refs}}(\gamma) \wedge \forall i.(i \in \gamma \rightarrow @_i\texttt{student}))$$

3. A course must be taught by a professor.
   Attribute `prof_ref` *of element* `course` *must refer to an element* `professor`.

$$\texttt{course} \rightarrow \exists k(*_{\texttt{proof\_ref}}(k) \wedge @_k\texttt{professor})$$

4. An examination must be related to a student.
   Attribute `stud_ref` *of element* `examination` *must refer to an element* `student`.

$$\texttt{examination} \rightarrow \exists k(*_{\texttt{stud\_ref}}(k) \wedge @_k\texttt{student})$$

5. A student can be evaluated only once for a given course.
   Attributes `stud_ref` *and* `cour_ref` *of an element* `examination` *cannot have the same values (couple of values) in different* `examination` *elements.*

$$\begin{aligned}&\forall i.\forall j.((@_i\texttt{student} \wedge @_j\texttt{course}) \rightarrow \\ &\quad \forall m.\forall n(@_m(\texttt{examination} \wedge *_{\texttt{stud\_ref}}(i) \wedge *_{\texttt{cour\_ref}}(j)) \\ &\quad \wedge @_n(\texttt{examination} \wedge *_{\texttt{stud\_ref}}(i) \wedge *_{\texttt{cour\_ref}}(j)) \rightarrow @_m n)\end{aligned}$$

6. A professor cannot be both supervisor and reviewer of the same student.
   Attribute `stud_refs` *of element* `thesis_stud` *and attribute* `stud_refs` *of element* `thesis_reviewer`, *when* `thesis_stud` `thesis_reviewer` *are in the same element* `professor`, *refer to two different and disjoint sets of elements* `student`.

$$\begin{aligned}&\neg\exists k.j.\gamma(@_k(\texttt{professor} \wedge \bigcirc^{\exists}(\texttt{thesis\_reviewer} \wedge *_{\texttt{stud\_refs}}(\gamma) \wedge j \in \gamma))\wedge \\ &\quad @_j(\texttt{student} \wedge \bigcirc^{\exists}(\texttt{supervisor} \wedge *_{\texttt{proof\_ref}}(k))))\end{aligned}$$

7. A professor can be supervisor only for students that attended and passed a course she taught. *Attribute* `stud_refs` *of a given element* `thesis_stud` *must have values among those of attribute* `stud_ref` *of an element* `examination`*, where its attribute* `cour_ref` *refers to an element* `course` *having attribute* `prof_ref` *referring to the element* `professor` *containing the given element* `thesis_stud`*.*

$$\forall i(@_i\texttt{professor} \rightarrow \forall k(@_k(\texttt{student} \wedge \bigcirc^{\exists}(\texttt{supervisor} \wedge *_{\texttt{proof\_ref}}(i))) \rightarrow \exists m(@_m(\texttt{course} \wedge *_{\texttt{proof\_ref}}(i) \wedge \exists n(@_n(\texttt{examination} \wedge *_{\texttt{cour\_ref}}(m) \wedge *_{\texttt{stud\_ref}}(k)))))))$$

6 Conclusions

In this paper we proposed a simple extension of hybrid logic with a reference operator $*_c$. We show how this logic, called XHyb, is suitable to express references specification, overcoming, in an feasible way, some limitations of DTD expressiveness.

References

1. Blackburn, P.: Representation, reasoning, and relational structures: a hybrid logic manifesto. Logic J. IGPL **8**(3), 339–365 (2000)
2. Blackburn, P., Tzakova, M.: Hybridizing concept languages. Ann. Math. Artif. Intell. **24**(1–4), 23–49 (1998)
3. Fan, W., Libkin, L.: On XML integrity constraints in the presence of DTDs. J. ACM **49**(3), 368–406 (2002)
4. Fan, W., Siméon, J.: Integrity constraints for XML. J. Comput. Syst. Sci. **66**(1), 254–291 (2003). Special Issue on PODS 2000
5. Fan, W.: Xml constraints: specification, analysis, and applications. In: Proceedings, DEXA, pp. 805–809 (2005)
6. Franceschet, M., de Rijke, M.: Model checking hybrid logics (with an application to semistructured data). J. Appl. Logic **4**(3), 279–304 (2006)
7. Marx, M.: Xpath and modal logics of finite dag's. In: Proceedings of Automated Reasoning with Analytic Tableaux and Related Methods, International Conference, TABLEAUX 2003, Rome, Italy, 9–12 September 2003, pp. 150–164 (2003)
8. Rodrigues, K.R., dos Santos Mello, R.: A faceted taxonomy of semantic integrity constraints for the XML data model. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 65–74. Springer, Heidelberg (2007)
9. van der Vlist, E.: XML Schema - The W3C's Object-oriented Descriptions for XML. O'Reilly, Sebastopol (2002)