# Hybrid Logic for expressing XML schemas with typed references.

Nicole Bidoit
Université Paris Sud, UMR 8623, Orsay F-91405
CNRS, UMR 8623, Orsay F-91405
e-mail: nicole.bidoit[at]lri.fr

Dario Colazzo
Université Paris Sud, UMR 8623, Orsay F-91405
CNRS, UMR 8623, Orsay F-91405
e-mail: dario.colazzo[at]lri.fr

## ABSTRACT

The aim of the paper is to provide a fully general notion of schema capturing well-typed references, called ref-schemas, allowing for general regular expressions The main contribution of the paper is to show that ref-schemas are expressible in Hybrid Modal Logic which entails that tools like for instance the tableau system developed in [9] can be used in order to check for constraint satisfiability in presence of ref-schemas.

## 1. INTRODUCTION

*Hybrid Modal Logic* (HML)[10, 12, 24, 20, 25] is a modal logic whose main distinctive features lie in its ability of naming states of a model, and on the possibility of using state names in order to easily and directly express properties relating multiples model states. HML was previously investigated by [1, 18] for expressing constraints on XML data. The work in [7] shows that HML is powerfull enough to express general constraints subsuming path contraints [14]. In a way, these schemas further called normalized ref-schemas extend *normalized* DTDs [5].Along the lines of [7], in this paper we focus on the problem of expressing XML schemas by means of HML. We show that HML is able to fully capture the expressive power XML schemas where element content is described by means of *unordered* (or commutative) Regular Expressions (REs in the following). This is not obvious since HML does not directly provide mechanism to describe properties like mutual exclusion, expressed by REs of the form $r+s$ (union), or co-occurrence properties, generally expressed by REs of the form $r\&s$, where & is the unordered (or *shuffle*) concatenation operator. Previous works [19] show how these properties can be captured by some ad-hoc variable-free logic when severe restrictions are posed on the structure of REs.

Here, in order to fully characterize unordered REs by means of HML, we will proceed in three subsequent steps. Essentially, given a RE $r$ we will first show how to obtain an expression $r'$ which is an over-approximation of $r$, obtained by relaxing co-occurrence constraints; we then show how to derive from $r'$, an HML formula $\phi_{r'}$ which preserves $r'$ meaning, by using some techniques provided in [7]; finally, we show how the use of HML mechanism like $\downarrow$ and @ can lead us to recover co-occurrence constraints in a formula $\sigma_r$, and missing in $r'$, so that the meaning of $r$ will be given by the conjunction $\phi_{r'} \wedge \sigma_r$. The paper is organized as follows. The next section is devoted to a short introduction to Hybrid Modal Logic. Section 3 introduces general schemas capturing references as "first class citizen" and then shows how general ref-schema can be expressed in HML. Section 4 discusses related work and further research directions.

## 2. PRELIMINARIES

We assume the reader familiar with modal logics and just recall here the main features of *Hybrid modal logic* (HML) [11, 2, 3]. Previous work [1, 18, 16, 7] have investigated a modal logic approach to model semistructured data which is naturally motivated by the fact that such data are commonly viewed as edge labelled graphs thus as Kripke models [22]. Although modal logic is a simple formalism for working with graphs, it has no mechanism for referring to and reasoning about the individual nodes in such structures. HML increases the effectiveness of modal logic by allowing one to grasp the nodes via formulas.This is made possible by four fundamental features: (1) a *nominal* or a *state variable* is a special atomic formula that names or denotes the unique state where the formula holds; (2) in $@_u\psi$, the *satisfaction operator* $@_u$ enables to check the formula $\psi$ at the state named (or denoted) by the nominal (or state variable) $u$; (3) in $\downarrow x\ \psi$, the *binder operator* $\downarrow x$ binds, in $\psi$, the state variable $x$ to the current state. The *alphabet* of a HML language is given by four disjoint sets: propositions $PROP=\{p, q, \cdots\}$, nominals $NOM=\{a, b, \cdots\}$, state variables $SVAR=\{x, y, \cdots\}$ and labels $\mathcal{E}=\{e_1, ..., e_n\}$. *Well formed formulas* (wffs) are recursively defined by: (i) each proposition $p$ is a wff, (ii) each nominal $a$ and each state variable $x$ is a wff, (iii) assuming that $\psi_1$ and $\psi_2$ are wffs, then the following expressions are wffs: $\neg\psi_1$, $\psi_1 \wedge \psi_2$, $[e]\psi_1$ where $e$ is a label, $G\psi_1$, $\downarrow x\ \psi_1$ where $x$ is a state variable, $@_u\psi_1$ where $u$ is either a nominal or a state variable.

A *model* (a document) $\mathfrak{M}$ of HML is a Kripke structure $(S, r, R, V, \mathcal{I}_{nom})$ where: $S$ is a finite set of *states* (the nodes of the document) containing a distinguished element $r$ (the root of the document); $R = \{r_e | e \in \mathcal{E}\}$ is a set of binary *accessibility relations* on $S$ (the labelled edges of the document linking nodes); the function $V{:}PROP \rightarrow Pow(S)$ assignes to each proposition $p$ the set of states where $p$ holds (the data component of the document); the function $\mathcal{I}_{nom}{:}NOM \rightarrow S$ assignes a unique state to each nominal.

A valuation $g : SVAR \rightarrow S$ assignes a state to each state variable. By $g \overset{x}{\sim} g'$ we denote that $g'$ is a $x$-variant of $g$.

A model $\mathfrak{M}$ satisfies the wff $\psi$ at state $s$ wrt a valuation $g$, denoted by $\mathfrak{M}, g, s \models \psi$, iff:

$-\psi$ is $p$ and $s \in V(p)$, where $p \in PROP$
$-\psi$ is $a$ and $I_{nom}(a) = s$
$-\psi$ is $x$ and $g(x) = s$
$-\psi$ is $\psi_1 \wedge \psi_2$ and $\mathfrak{M}, g, s \models \psi_1$ and $\mathfrak{M}, g, s \models \psi_2$
$-\psi$ is $\neg\psi$ and $\mathfrak{M}, g, s \not\models \psi$
$-\psi$ is $[e]\psi$ and for any $s' \in S$, $(s, s') \in r_e$ implies $\mathfrak{M}, g, s' \models \psi$
$-\psi$ is $G\psi$ and for any $s' \in S$ accessible from $s$ (by a path), we have: $\mathfrak{M}, g, s' \models \psi$
$-\psi$ is $\downarrow x\ \psi$ and $\mathfrak{M}, g', s \models \psi$ with $g \overset{x}{\sim} g'$, $g'(x) = s$
$-\psi$ is $@_x\psi$ and $\mathfrak{M}, g, g(x) \models \psi$
$-\psi$ is $@_a\psi$ and $\mathfrak{M}, g, I_{nom}(a) \models \psi$

The modalities $\langle e\rangle$, $F$, $G^*$ and $F^*$ are defined[1] by: $\langle e\rangle\psi =_{def} \neg[e]\neg\psi$ and $F\psi =_{def} \neg G\neg\psi$, $G^*\psi =_{def} \psi \wedge G\psi$ and $F^*\psi =_{def} \psi \vee F\psi$.

### XML document and references

In general, XML documents are modeled by ordered trees. Here, in order have a simpler representation of references, we prefer to use a graph representation. An XML document with references is represented in Figure 1 as a simple rooted graph with two kinds of labelled edges: plain edges and dashed ones. First of all, let us look at the graph restricted to plain edges, further called child edges : notice that it is a tree like XML documents are (except that XML documents are ordered). The root node (named $r$) has 3 (unnamed) children, all of them being connected to $r$ through a child edge labbed by $e$. The dashed edges represent references, that is "pointers" from elements to elements. Next, dashed edges will be simply called references. The left most child of the root $r$ is the source of a reference labelled by $\overrightarrow{r}$. For the sake of the presentation, a node of a document which is the target of a $\tilde{e}$ labelled child edge is called a $\tilde{e}$-node. All three child nodes of the root $r$ are $e$-nodes.

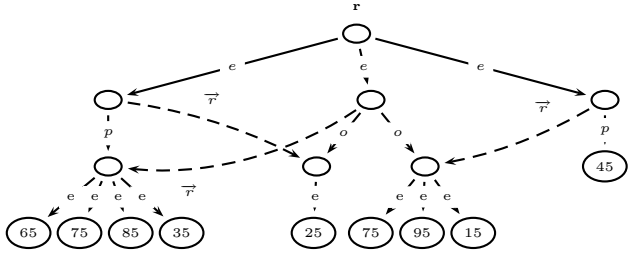


Figure 1. A document

From now on, we make the assumption that a document with references is always build from a simple one (a rooted unordered labelled tree). It explains the distinction made between child edges (those of the tree) and references (those connecting elements). Indeed, our data model is comparable to the OEM data model.

The document of Figure 1 can also be seen as the representation of some Kripke model $\mathfrak{M}$. The root of the document will be captured by the use, in our language, of a unique nominal *root* and hence, here *root* will be interpreted by $r$ that is $\mathcal{I}_{nom}(root)=r$. This assumption is made in the rest of the paper. Note that, the Kripke model $\mathfrak{M}$ of Figure 1 does not contain any proposition except for the data appearing in the leaves of the underlying tree. Propositions will be used later on for marking nodes of the document.

## 3. WELL-TYPED REFERENCES

This section is devoted to the presentation of a general notion of schema *a la* DTD capturing well typed references. Next, we assume that $\mathcal{V}$ is a finite set of non-terminal symbols, containing the symbol *Start*, and the empty word $\Lambda$. By convention, a non terminal symbol starts with a capital letter while a label starts with a non-capital letter. The set of labels $\mathcal{E}$ is partionned in two disjoint sets $E$ and $\overrightarrow{E}$: labels in $E$ are called *child* labels whereas labels in $\overrightarrow{E}$ are called *references*. We use the following convention: $e$ (resp. $\overrightarrow{e}$, $\tilde{e}$) denotes a child label (resp. a reference, any label). For the sake of the presentation, we avoid to consider base types.

DEFINITION 1 (REF-SCHEMA). *A ref-schema $\mathcal{G}$ is given by $(\mathcal{E}, \mathcal{V}, Start, \theta)$ where $\mathcal{E}=E\cup\overrightarrow{E}$, $\mathcal{V}$ and Start are defined as above and where the typing function $\theta$ associates to each non terminal symbol $X$ a regular expression of the form:*

$$R ::= \tilde{e}\ X \mid R + R \mid R\ \&\ R \mid R^* \mid \Lambda.$$

*The typing function $\theta$ needs to satisfy that: (1) for each child-label $e$, Type($e$) is a singleton, where Type($\tilde{e}$) denotes the set of non terminals $X$ in elementary patterns $\tilde{e}X$ occurring in some type definition given by $\theta$, and (2) for each non terminal $X$ distinct from Start, $Start \Rightarrow^*_{\mathcal{G}} X$ holds where $\Rightarrow^*_{\mathcal{G}}$ is the transitive closure of the relation $\Rightarrow_{\mathcal{G}}$ naturally defined by $X \Rightarrow_{\mathcal{G}} Y$ if $Y$ occurs in $\theta(X)$.*

Note that condition (1) is the usual one considered for DTD. Intuitively, condition (2) ensures that all (definitions of) types are linked to the initial type *Start*. To simplify the presentation, the definition of $\theta$ is given by "assignements" of the form $X := R$ in place of $\theta(X) = R$.

In order to define what are documents conforming to a ref-schema $\mathcal{G}$, we need to associate to a regular expression $R$ its extension $[\![R]\!]$.

Given an alphabet $\mathcal{A}$, an unordered word over $\mathcal{A}$ is denoted $a_1\&\ldots\&a_n$ where $a_i\in\mathcal{A}$ for $i = 1..n$. It should be understood that for any permutation $\rho$ over $[1..n]$, the two words $a_1\&\ldots\&a_n$ and $a_{\rho(1)}\&\ldots\&a_{\rho(n)}$ are equal. The empty word is denoted by $\_$. Given two sets of unordered words $W_1$ and $W_2$, the unordered concatenation $W_1\&W_2$ is simply defined by $\{w_1\&w_2 \mid w_1\in W_1 \text{ and } w_2\in W_2\}$. Obviously, from the previous property, we have that $W_1\&W_2=W_2\&W_1$.

The set $[\![R]\!]$ of unordered words build from elementary patterns is defined according to the following equations:
$[\![\tilde{e}X]\!]=\{\tilde{e}X\}$, $[\![R_1+R_2]\!]=[\![R_1]\!]\cup[\![R_2]\!]$, $[\![R_1\&R_2]\!]=[\![R_1]\!]\&[\![R_2]\!]$, $[\![R^*]\!]=\cup_{n>0}\ (\&_{i=1..n}[\![R]\!])\cup\{\_\}$, $[\![\Lambda]\!]=\{\_\}$

We are now ready to formally define what are the instances of a ref-schema. Recall that a document is a (Kripke) model.

*Definition* 2. *Let $\mathfrak{M}=(S, r, R, V, \mathcal{I}_{nom})$ be a finite model and $\mathcal{G}=(\mathcal{E}, \mathcal{V}, Start, \theta)$ be a ref-schema. We say that $\mathfrak{M}$ satisfies the ref-schema $\mathcal{G}$, denoted $\mathfrak{M}:G$, if:*
*(1) the model $\mathfrak{M}$ restricted to the relations $r_e$ where $e$ is a child label, is a tree, and*
*(2) there exists a <u>total</u> mapping $\vartheta : S \to \mathcal{V}$ such that:*
*(a) $\vartheta(r) = Start$,*
*(b) forall $n\in S$, if $\vartheta(n)=X$ and $\theta(X)=R$, then $\&_{\tilde{e}Y\in\Gamma_n}\{\tilde{e}Y\}\in[\![R]\!]$ where $\Gamma_n=\{\{\tilde{e}Y \mid (n,n')\in R_{\tilde{e}} \text{ and } Y=\vartheta(n')\}\}$.*

The document $\mathfrak{M}$ of Figure 1 conforms to the ref-schema $\mathcal{G}$ specified by:

$$\begin{aligned} Start &:= (\ e\ X\ )^* \\ X &:= (\ p\ Y + o\ Z\ )^*\ \&\ (\ \overrightarrow{r}\ Y + \overrightarrow{r}\ Z\ )^* \\ Y &:= (\ e\ X\ \&\ e\ X\ )^* \\ Z &:= (\ e\ X\ \&\ e\ X\ )^*\ \&\ (\ e\ X\ ) \end{aligned}$$

**The ref-schema $\mathcal{G}$.**

### Flat ref-schemas.

For the sake of translating ref-schemas into HML formulas, and indeed in order to associated a normalized ref-schema to a ref-schema, we proceed to a simplication of the regular expressions, called flattening, which is only possible because of the unordered assumption.

*Lemma* 3. *Let $R$ be a regular expression. There exists a flattened expression Flat($R$) such that $[\![R]\!] = [\![Flat(R)]\!]$ where a flattened expression is defined according to the grammar:*

$$\begin{aligned} R &::= B \mid R + R & A &::= \tilde{e}\ X \mid A\ \&\ A \\ B &::= \tilde{e}\ X \mid B\ \&\ B \mid A^* \mid \Lambda \end{aligned}$$

### Marked flat ref-schemas.

Next, wlog, all expressions considered are flattened ones. A last slight modification of expressions is needed in order to cope with the following situation: in a ref-schema $\mathcal{G}$, it may

[1] We would like to recall here that, because of the use of the modality $G$, the HML language considered is no longer a fragment of first order logic.

happen that an elementary pattern $(\tilde{e}X)$ has multiple occurrences. Rather than disallowing such cases and in order to keep our framework fully general, we introduce markers in order to distinguish these occurrences. Markers are propositions. They are introduced in the definition of flattened regular expression by replacing $\tilde{e}X$ by $(\tilde{e},q)X$ where $q$ is a proposition symbol of the language.

Definitions 1 and 2 of a ref-schema and of ref-schema instances need to be sligthly modified. The changes are technical but not very deep. Thus:
• A marked ref-schema $\mathcal{G}$ is given by $(Prop, \mathcal{E}, \mathcal{V}, Start, \theta)$ where $Prop$ is a finite set of proposition symbols and $\theta$ associates to each non terminal symbol $X$ a marked flattened regular expression with conditions (1) and (2) of Definition 1 plus the following one[2]:
(3) if $(\tilde{e},q_1)X$ and $(\tilde{f},q_2)Y$ occurs in some type definitions in $\mathcal{G}$ then $q_1 \neq q_2$.
• In the definition of the extension $[\![R]\!]$ of $R$, the case of $[\![\tilde{e}X]\!]$ is now replaced by $[\![(\tilde{e},q)X]\!]=\{(\tilde{e},q)X\}$.
• Finally, in definition 2, the statement of condition (b) is changed to: forall $n \in S$, if $\vartheta(n)=X$ and $\theta(X)=R$, then there exists a word $w \in [\![R]\!]$ such that there exists a bijection $\beta$ from $w$ (considered as a multiset) to $\{(\tilde{e},n') \mid (n,n') \in R_{\tilde{e}}\}$ such that for each $(\tilde{e},q)Y \in \omega$, we have $\beta((\tilde{e},q)Y)=(\tilde{e},n')$ implies $\vartheta(n')=Y$ and $q \in V(n')$.

Intuitively, if we consider a finite state automata associated to the schema, markers $q_i$ can be seen as unique identifiers of automata states (assume that each $\tilde{e}X$ corresponds to an automata state). Then, over schema instances, the set of $q_i$'s associated to each node represent the set of automata states that successfully recognised that node (a node can be analized in multiple states due to references). So proposition symbols associated to the instance nodes can actually be seen as a successful run of the schema automata.
For the sake of simplicity, in the following, we will omit the marker $q$ for elementary pattern $\tilde{e}X$ that have a single occurence.

*Lemma* 4. *Let $\mathcal{G}$ be a flat ref-schema defined over $\mathcal{L}$ and let $\mathcal{G}_{mark}$ be a marked ref-schema associated with $\mathcal{G}$. Then:*
*(1) for each model $\mathcal{M} : \mathcal{G}$, there exists a model $\mathcal{N} : \mathcal{G}_{mark}$ such that $\mathcal{M} = \mathcal{N}_{|\mathcal{L}}$, and*
*(2) for each model $\mathcal{N} : \mathcal{G}_{mark}$, we have $\mathcal{N}_{|\mathcal{L}} : \mathcal{G}$ where $\mathcal{N}_{|\mathcal{L}}$ denotes the restriction of the Kripke structure $\mathcal{N}$ to edges whose labels are in $\mathcal{L}$ and to proposition assignments for propositions in $\mathcal{L}$.*

### *Normalized ref-schema.*

[7] introduces schemas capturing well-typed references in a limited way. Intuitively, these schemas, further called normalized ref-schemas[3] do not allow one for fully general expression and can be viewed as an extension of the normalized DTD of [5] with well typed references. The main contribution of [7] is to show that documents conforming to a given normalized ref-schemas $\mathcal{G}$ are exactly the finite Kripke models satisfying a HML wff $\tau_{\mathcal{G}}$.

*Definition* 5. *A normalized ref-schema $(\mathcal{E}, \mathcal{V}, Start, \theta)$ is a (marked) ref-schema based on the normalized regular expressions defined by:*

$R \ := B \mid R \ + \ R$ *and*
$B \ := \Lambda \mid ((\tilde{e},q)X)^{op} \mid B \ \& \ B$ *where op is either* ! *or* $^*$.

For instance, normalized ref-schema forbid expression of the following form: $((\tilde{e_1},q_1)X_1 \& \cdots \& (\tilde{e_k},q_k)X_k))^*$.
The next result was proved in [7] although without markers. Because of space limitation, it is stated without proof.

[2]Condition (3) should hold for any $X$ and $Y$, even if $X=Y$
[3]In [7], they are called pattern schemas.

*Theorem* 6. *[7] Let $\mathcal{G}$ be a normalized ref-schema. There exist an HML wff $\tau_{\mathcal{G}}$ such that for any model (document) $\mathfrak{M}$, we have: $\mathfrak{M} : \mathcal{G}$ iff $\mathfrak{M}, g, r \models \tau_{\mathcal{G}}$ where $g$ is any state variable valuation.*

### *Capturing Ref-schemas by HML*

We now show that HML is powerfull enough to express ref-schemas. Intuitively, we show that a ref-schema is equivalent to a normalized ref-schema together with a constraint. From the previous section, we know that a normalized ref-schema can be equivalently specified by an HML formula. This yields to expressing a general ref-schema in HML.
The next result formalizes the fact that general ref-schemas are expressible in HML although stating this result requires to put some restriction over references in the type definition. A ref-schema $\mathcal{G}$ is said *ref-normalized* if, in the flattened type definitions $\theta(X)$ of each type $X$, elementary patterns $\overrightarrow{e}Y$ only occur in simple expressions either of the form $(\overrightarrow{e}Y)^*$ or of the form $(\overrightarrow{e}Y)^!$.

*Theorem* 7. *Let $\mathcal{G}$ be a ref-schema over $\mathcal{L}$. If $\mathcal{G}$ is ref-normalized then there exists a normalized ref-schema $\mathcal{G}_{norm}$ and a constraint given by an HML formula $\mathcal{C}_{\mathcal{G}}$ such that:*
*(1) for each model $\mathcal{M} : \mathcal{G}$, there exists a model $\mathcal{N} : \mathcal{G}_{norm}$ such that: $\mathcal{N}, g, r \models \mathcal{C}_{\mathcal{G}}$ and $\mathcal{N}_{|\mathcal{L}}=\mathcal{M}$.*
*(2) for each model $\mathcal{N} : \mathcal{G}_{norm}$ such that $\mathcal{N}, g, r \models \mathcal{C}_{\mathcal{G}}$, we have that $\mathcal{N}_{|\mathcal{L}} : \mathcal{G}$.*
*Above, $\mathcal{N}_{|\mathcal{L}}$ denotes the restriction of the Kripke structure $\mathcal{N}$ to edges whose labels are in $\mathcal{L}$ and to proposition assignments for propositions in $\mathcal{L}$ as well.*

Intuitively, the theorem above states a strong correspondence between $\mathcal{G}$ and $\mathcal{G}_{norm}$ "plus" $\mathcal{C}_{\mathcal{G}}$ implying, to some extends, that it does not make any difference to work with a document conforming to $\mathcal{G}$ or to work with a document conforming to $\mathcal{G}_{norm}$ as soon as it satisfies the structural constraint $\mathcal{C}_{\mathcal{G}}$.
Theorem 7 also entails that the ref-schema $\mathcal{G}$ (when it is a ref-normalized schema) can be captured by the HML formula $\tau_{\mathcal{G}_{norm}} \wedge \mathcal{C}_{\mathcal{G}}$.

*Corollary* 8. *Let $\mathcal{G}$ be a ref-normalized schema. There exists an HML formula $\tau_{\mathcal{G}}$ such that*[4]*, for any HLM formula $\varphi$ over the language induced by $\mathcal{G}$, the two following statement are equivalent:*
*(1) there exists $\mathcal{M}$ such that $\mathcal{M} : \mathcal{G}$ and $\mathcal{M} \models \varphi$*
*(2) there exists $\mathcal{N}$ such that $\mathcal{N} \models \tau_{\mathcal{G}} \wedge \varphi$.*

It may be useful for the reader to note that, Theorem 7 allows for the following situation: it may be the case that a model $\mathcal{N}$ conforms to the normalized ref-schema $\mathcal{G}_{norm}$ and moreover that $\mathcal{N}_{|\mathcal{L}} : G$ although $\mathcal{N} \not\models \mathcal{C}_{\mathcal{G}}$.

## 4. RELATED WORKS

Most of existing type languages for XML disregard the problem of typing references. Concerning type languages assuming sequence ordering, languages like DTD and all languages *a la* XDuce/CDuce [21, 6] do not permit to describe data with typed references. Concerning type languages not assuming sequence ordering, probably the most relevant one is the TQL language [15], which does not contain mechanisms to define references.
Besides the previous work [8], quite a few studies address typing mechanisms for references for semistructred / XML data. XML Schema [23] contains some mechanisms to type references. They do not allow one to specify references and their target type in a flexible and direct manner. Indeed, XML Schema uses XPath to specify typed references. XPath [17] is a rather complex, requires a good amount of expertize to be used correctly and moreover, reasoning about constraints defined with XPath is highly intricate, if not impossible. XML Schema does not allow one to specify references

[4]Indeed, $\tau_{\mathcal{G}} \equiv_{def} \tau_{\mathcal{G}_{norm}} \wedge \mathcal{C}_{\mathcal{G}}$

whose target elements are possibly of different type. No mechanism is provided to define and check generic integrity constraints.
Simeon and Fan [17] propose an extension of DTD able to model classical relational and object oriented referential constraints. (key constraints and foreign-key constraints). Their approach and that of XML are closely related. Rather negative results concerning decidability for key and foreign-key constraints have been showed in [4].
The work [13] provides another interesting approach to the logical characterization of XML schemas and constraints. Their work proposes a decidable logic able to model inclusion constraints only over one attribute, otherwise decidability is lost. As stated in [13], if the proposed decidable logic is extended with navigational mechanisms like $G$, decidability is not proved to hold (it is an open problem). However, in our context, we are quite confident that this property holds for a wide class of constraints (involving F and G) under the assumption that data are constrained by a schema, and currently we are actively investigating this possibility.
Finally, a further difference wrt [13] , and works referenced therein, is that while [13] deals with ordered XML documents, we are concerned with database issues, where ordering is uninfluential.

To summarize, the aim of our study is to pave the way to a unique logical framework, allowing to deal with type and integrity constraints (as well as with queries) at the same time and in a clear, simple and flexible way. We would like to outline that the present work should not to be intended as yet another formalism to describe XML data with well-typed references, although ref-schemas are elegant extensions of DTDs. Indeed, this work presents basic techniques that can be easily extended in order to obtain HML logical characterisation of schemata defined according to existing techniques, like the two above cited ones, and of generic XML structural constraints . As stated in Section 3, this allows the use of existing techniques [9] to formally check important properties like constraint implication and constraint consistency in a *unique* framework, without the need of developing potential complex and error prone tools to cope with formalisms of different nature, like, say, DTD for types and XPath for constraints [23].

One of the limitation of our framework is that, currently, we are not able to manage sequence ordering. This is not a strong limitation because the class of database applications where XML sequence ordering is non-influential is quite wide. However, HML seems able to describe sequence ordering as well, by using techniques similar to those we used to encode co-occurrences properties.

## 5. ACKNOWLEDGEMENT

This work has been partially funded by the ANR Project Codex (DEFIS 2008).

## REFERENCES

[1] N. Alechina, S. Demri, and M. De Rijke. Path constraints from a modal logic point of view. In *Proc. of the 8th Int. Workshop on Knowledge Representation meets Databases*, 2001.

[2] C. Areces, P. Blackburn, and M. Marx. Road-map on complexity for hybrid logics. In *Proc. of the 8th Annual Conf. of the EACSL*, pages 307–321. LNCS, Springer, 1999.

[3] C. Areces, P. Blackburn, and M. Marx. Hybrid logics: Characterization, interpolation and complexity. *J. of Symbolic Logic*, 66(3):977–1010, 2001.

[4] M. Arenas, W. Fan, and L. Libkin. Consistency of xml specifications. In *Inconsistency Tolerance*, pages 15–41, 2005.

[5] M. Benedikt, W. Fan, and F. Geerts. Xpath satisfiability in the presence of dtds. In *Proc. of the 24th Symp. on Principles of Database Systems*, pages 25–36. ACM SIGACT-SIGMOD-SIGART, 2005.

[6] V. Benzaken, G. Castagna, and A. Frisch. Cduce: an xml-centric general-purpose language. In C. Runciman and O. Shivers, editors, *ICFP*, pages 51–63. ACM, 2003.

[7] N. Bidoit, S. Cerrito, and V. Thion. A first step towards modeling semistructured data in hybrid multimodal logic. *J. of Applied Non-Classical Logics*, 14(4):447–476, 2004.

[8] N. Bidoit, S. Cerrito, and V. Thion. A first step towards modeling semistructured data in hybrid multimodal logic. *Journal of Applied Non-Classical Logics*, 14(4):447–475, 2004.

[9] N. Bidoit and D. Colazzo. Testing xml constraint satisfiability. Hylo, 2006.

[10] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic J. of the IGPL*, 8(3):339–365, 2000.

[11] P. Blackburn and J. Seligman. Hybrid languages. *J. of Logic, Language and Information*, 4:251–272, 1995.

[12] P. Blackburn, J. van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*, volume 3. Elsevier, 2007.

[13] M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and xml reasoning. In *PODS*, pages 10–19, 2006.

[14] P. Buneman, W. Fan, and S. Weinstein. Path constraints on semistructured and structured data. In *Proc. of the 17th Symp. on Principles of Database Systems*, pages 129–138. ACM SIGACT-SIGMOD-SIGART, 1998.

[15] G. Conforti, G. Ghelli, A. Albano, D. Colazzo, P. Manghi, and C. Sartiani. The query language tql. In *WebDB*, pages 13–18, 2002.

[16] S. Demri. (modal) logics for semistructured data. In *"Third Workshop on Methods for Modalities.* (Invited talk), 2003.

[17] W. Fan and J. Siméon. Integrity constraints for xml. In *PODS*, pages 23–34. ACM, 2000.

[18] M. Franceschet and M. De Rijke. Model checking for hybrid logics. In *Workshop Methods for Modalities*, 2003.

[19] G. Ghelli, D. Colazzo, and C. Sartiani. Efficient inclusion for a class of XML types with interleaving and counting. In *DBPL*, 2007.

[20] G. Hoffmann and C. Areces. Htab: A terminating tableaux system for hybrid logic. In *Methods for Modalities Workshop*, 2007.

[21] H. Hosoya and B. C. Pierce. Xduce: A typed xml processing language (preliminary report). In D. Suciu and G. Vossen, editors, *WebDB (Selected Papers)*, volume 1997 of *Lecture Notes in Computer Science*, pages 226–244. Springer, 2000.

[22] S. Kripke. *Semantic Considerations on Modal Logic in Reference and Modality.* Oxford University Press, London, "1971".

[23] W. Recommendation. Xml schema. In *http://www.w3.org/TR/xmlschema-0/.*

[24] B. ten Cate. Xml from the viewpoint of modal logic. In *Invited talk at the Methods for Modalities Workshop*, 2007.

[25] B. ten Cate and M. Franceschet. On the complexity of hybrid logics with binders. In *CSL*, pages 339–354, 2005.