

Replace this file with `prentcsmacro.sty` for your meeting,
or with `entcsmacro.sty` for your meeting. Both can be
found at the [ENTCS Macro Home Page](#).

Testing XML constraint satisfiability

Nicole Bidoit ^{1,2} Dario Colazzo ^{1,3}

*Univ Paris Sud, UMR CNRS 8623, Orsay F-91405
CNRS, Orsay F 91405*

Abstract

In a previous paper, we have showed that *Hybrid Modal Logic* can be successfully used to model semistructured data and provides a simple and well suited formalism for capturing “well typed” references and of course a powerful language for expressing constraint. This paper builds on the previous one and provides a tableau proof technique for constraint satisfiability testing in the presence of schemas.

Keywords: Database, Semistructured data, Integrity constraints, Schema, Multimodal and Hybrid logic

1 Introduction

Schemas and integrity constraints play an important role with respect to data manipulation, reasoning and optimization. This of course applies to semistructured data and XML documents. It is also widely recognized that, being able to model schemas, constraints and queries within the same formalism (logic as a matter of fact) is a key issue to investigate problems like subtyping, constraint implication and satisfiability, query correctness etc.

The modal logic approach to model semistructured data is naturally motivated by the fact that semistructured data, hence XML documents, are commonly viewed as edge labeled graphs thus as Kripke models [24]. This approach has been investigated in different ways to tackle different problems [18]: schemas subsumption [1,14], path constraints [2], query languages [15], XPath queries [25]...

In [7] and subsequently in [8], we have investigated how to capture well-typed references as first class citizen within the definition of semistructured data schema. References are most commonly found in practice. However, surprisingly enough, the notions of schema provided in the literature [29,26] do not provide a mechanism for specifying the types of the referenced elements in a document. The notion of schema capturing well-typed references provided in [8], called ref-schema, is very general and

¹ This work has been partially supported by the French National ACI Tralala project.

² Email: nicole.bidoit@lri.fr

³ Email: dario.colazzo@lri.fr

extends the one previously introduced in [7], called normalized ref-schema. Another contribution of [7,8] has been to show that ref-schema can be expressed in *Hybrid Modal Logic* (HML)[10]. 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. HML provides a unique formalism to express sophisticated schemas, constraints and navigational queries a la XPath.

The work presented here builds naturally on [7,8] and addresses constraint satisfiability in presence of ref-schema: given a ref-schema \mathcal{G} and a constraint \mathcal{C} , does there exist a document conforming to \mathcal{G} and satisfying \mathcal{C} ? In our framework, this leads to the question: is $\tau_{\mathcal{G}} \wedge \mathcal{C}$ finitely satisfiable? The translation given in [8] of ref-schema into normalized ref-schema allows us to study this problem wlog in the case where \mathcal{G} is a normalized schema. The paper provides a proof procedure based on tableau techniques [23] for testing satisfiability by generating models. Our tableau system is naturally based on internalizing labeled deduction, as investigated in [11], which is elegant and powerful. In general, HML is not decidable [9,22] and the logic does not have the finite model property⁴. Thus, we restrict the satisfiability problem to the case of non recursive schemas. Although this assumption ensures that the depth of documents are upper bounded, we show that it is still not sufficient to entail the finite model property. The tableau system presented is showed to be sound and complete for satisfiability, although finite satisfiability is the relevant notion for reasoning about XML. This issue is discussed in the last section of the paper where some hints are given for solving finite satisfiability in presence of non recursive schema.

Related work

Quite a few studies address typing mechanisms for references in semistructred and XML data. XML Schema [29] contains some mechanisms to type references which are neither flexible nor direct. Indeed, as observed in [20], references are defined by means of XPath, which is rather complex and requires a good amount of expertise to be used correctly. Moreover, reasoning about constraints defined with XPath is highly intricate, if not impossible.

Simeon and Fan [20] propose an extension of DTD able to model classical relational and object oriented referential constraints. So the focus is on problems related to key constraints and foreign-key constraints. These constraints can be used to capture reference enforcement. It seems that this approach and that of XML are closely related. Rather negative results concerning decidability for key and foreign-key constraints have been showed in [5].

Note that we do not address the decidability and complexity issues, which have been extensively studied in [6,19] in a slightly different setting, although it appears that in our setting decidability is an open problem, as discussed in the last section of the paper.

The quite recent paper [13] provides another interesting approach to the problem

⁴ It is not the case that each satisfiable HML wff has a finite model.

of logically characterizing XML schemas and constraints. Main differences wrt to this work, and references therein, are the followings. The work [13] proposes a decidable logic able to model inclusion constraints only over one attribute, otherwise decidability is lost. So, some kind of references can be modeled in the standard way by means of inclusions constraints. Of course decidability is lost if the references require multiple attributes. Here, instead of capturing references indirectly and partially like in [13], by means of inclusion constraints, we model typed references by using primitive mechanisms, defined by means of HML primitives and which are able to describe references in an abstract and general way, as much as for object-oriented databases. Also, differently from [13], the logic we consider allows to model navigational properties requiring to visit all descendants of certain nodes (by using $G\phi$ formulae); actually, the logic [13] essentially enables only one step navigation (parent, child, ...) plus a kind of *somewhere* navigation, allowing to jump to some arbitrary node, not necessarily related with the current one. 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 they deal with ordered XML documents, we move in a more database framework, where ordering is uninfluential.

Organization

Section 2 is devoted to a short introduction to HML and its relationship with semistructured data and XML. Schemas capturing well-typed references are introduced in Section 3. The tableau system for testing satisfiability of constraints in presence of schemas is presented in Section 4. The last section is devoted to a discussion on further research directions.

2 Hybrid multimodal logic

We assume the reader familiar with modal logics and just recall here the main features of *Hybrid Modal Logic* (HML). The interested reader is invited to read [9,3,4] for a full presentation of HML. Hybrid Modal Logic is an extension of modal logics which provides a mechanism to name states (or graph nodes) and to assert that a formula is true at a named state (or graph node). 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 (under a given variable assignment); (2) the *satisfaction operator* $@_u$ applied to a formula ψ enables to check satisfaction of ψ at the state named (or denoted) by the nominal (or state variable) u ; (3) the *binder operator* $\downarrow x$ applied to a formula ψ binds the state variable x in ψ to the current state.

An HML *alphabet* is a set of propositions $P = \{p, q, \dots\}$, a set of nominals $Nom = \{a, b, \dots\}$, a set of state variables $Var = \{x, y, \dots\}$ and a finite set of labels

$\mathcal{E} = \{e_1, \dots, e_n\}$ ⁵. *Well formed formulas* (wffs) are defined by:

$$\text{WFF} ::= p \mid \top \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid [e]\psi \mid u \mid \downarrow x \psi \mid @_u\psi$$

where ψ , ψ_1 and ψ_2 are wffs, $p \in P$, $x \in Var$ and $u \in Nom \cup Var$.

Roughly, the *multimodality* of the language comes from the combination of the modal operator \square with labels leading to the finite set of modal operators $[e]$.

We also use the operators \vee , \Rightarrow and $\langle e \rangle$ classically defined by: $\psi_1 \vee \psi_2 =_{def} \neg(\neg\psi_1 \wedge \neg\psi_2)$, $\psi_1 \Rightarrow \psi_2 =_{def} \neg\psi_1 \vee \psi_2$ and $\langle e \rangle\psi =_{def} \neg[e]\neg\psi$.

A *model* \mathfrak{M} of HML is a Kripke structure (S, r, R, I_P, I_N) where: S is a set of *states* containing a distinguished element r ; $R = \{r_e \mid e \in \mathcal{E}\}$ is a set of binary *accessibility relations* on S ; the function $I_P: P \rightarrow Pow(S)$ assigns to each proposition p the set of states where p holds; the function $I_N: Nom \rightarrow S$ assigns a unique state to each nominal. A valuation is a function $g: Var \rightarrow S$ assigning a state to each state variable. By $g \overset{x}{\sim} g'$ we denote that g' is a x -variant of g .

The semantics of HML is defined as follows, where we restrict the presentation to the hybrid features: a model \mathfrak{M} satisfies the wff ψ at state s wrt a valuation g , noted $\mathfrak{M}, g, s \models \psi$, if:

$$\begin{aligned} \mathfrak{M}, g, s \models a & \quad \text{iff } I_N(a) = s, \quad \text{where } a \in Nom \\ \mathfrak{M}, g, s \models x & \quad \text{iff } g(x) = s, \quad \text{where } x \in Var \\ \mathfrak{M}, g, s \models \downarrow x \psi & \quad \text{iff } \mathfrak{M}, g', s \models \psi \text{ with } g \overset{x}{\sim} g', g'(x) = s, \text{ where } x \in Var \\ \mathfrak{M}, g, s \models @_x\psi & \quad \text{iff } \mathfrak{M}, g, g(x) \models \psi \text{ where } x \in Var \\ \mathfrak{M}, g, s \models @_a\psi & \quad \text{iff } \mathfrak{M}, g, I_N(a) \models \psi \text{ where } a \in Nom \end{aligned}$$

We write $\mathfrak{M}, s \models \psi$ when $\mathfrak{M}, g, s \models \psi$ is verified for any valuation g of the state variables.

The language is extended with two dual modalities G and F : $\mathfrak{M}, g, s \models G\psi$ iff for any state accessible via a path from the current state s , ψ holds at s , and $F\psi =_{def} \neg G\neg\psi$. We also use $G^*\psi$ for $\psi \wedge G\psi$ and $F^*\psi$ for $\psi \vee F\psi$. Recall that when such modalities are added, HML is no longer a fragment of first order logic.

The next example illustrates how a semistructured data is mapped to a model of an HML language and illustrates the semantics of the hybrid operators.

Example 2.1 The document \mathcal{X} of Figure 1 can be viewed as an HML model \mathfrak{M} where: the states of \mathfrak{M} are the nodes of \mathcal{X} ; the distinguished state r of \mathfrak{M} is the root of \mathcal{X} ; a unique nominal *root* is needed in the language to name r and thus $I_N(\text{root}) = r$; the accessibility relations of \mathfrak{M} are given by the labeled edges of \mathcal{X} ; the function I_P is given by the labeling of nodes in \mathcal{X} .

In Figure 1, the name n is given to one of the node for the sake of the presentation but it should not be considered neither as a nominal nor as a proposition. The wffs $[author]\neg Scott$, $\downarrow x \langle \overrightarrow{pubby} \rangle \langle \overrightarrow{pub_B} \rangle x$, $@_{root}[doc][article]\langle author \rangle \top$ are satisfied at n by \mathcal{X} . The wff $F Scott$ is satisfied at r but $G[doc]\langle book \rangle \top$ is not.

⁵ As usual, the sets P , Nom and Var are assumed pairwise disjoint.

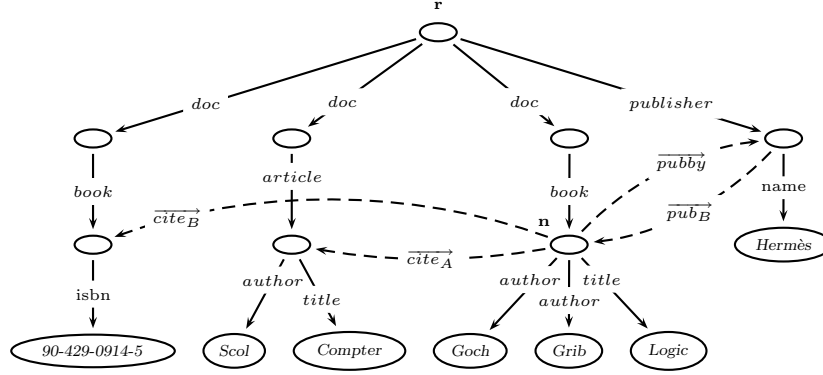


Fig. 1. Representation of a library database

In the case of modal logics, different restrictions may be imposed on the accessibility relations, thereby obtaining different logics with several styles of sound and complete proof systems [21,28]. Here, no such restriction exists and thus one can see HML as an extension of the modal logic K. Moreover, HML allows one to express such restrictions inside the logic itself [9]. For example, reflexivity of the accessibility relation r_e is expressed by $\downarrow x \langle e \rangle x$.

3 Schemas capturing well-typed references

The main contribution of [8,7] is to introduce a notion of XML schema which provides a simple and well-founded notion of reference typing. References are then first class citizen. The notion of schema introduced in [8] is fully general whether the notion of schema in [7], called here normalized ref-schema⁶ is restricted. Moreover, in [8], we show how to translate a general schema into a normalized one. This allows us here, for the purpose of testing constraint satisfiability in presence of schema, to restrict the presentation to normalized ref-schema without loss of generality.

A normalized ref-schema is specified in a style very close to that used to specify DTD. We assume that \mathcal{V} is a finite set of non-terminal symbols, containing the symbol *Start*, and Λ denoting the empty word. 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 partitioned in two disjoint sets E and \vec{E} : labels in E are called *child* labels whereas labels in \vec{E} are called *references*. We use the following convention : e (resp. \vec{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 3.1 [Normalized Ref-schema] A normalized ref-schema \mathcal{G} is given by $(\mathcal{E}, \mathcal{V}, Start, \theta)$ where the typing function θ associates to each non terminal symbol X a regular expression of the form: $R := B \mid R + R$, and $B := \Lambda \mid (\tilde{e}X)^{op} \mid B, B$ where op is either $!$, $*$ or $+$.

It is assumed that (1) in a conjunctive expression (B, B) , each occurrence of an elementary pattern $(\tilde{e}X)$ is unique and that (2) the typing function θ satisfies that: (a) for each label \tilde{e} , $type(\tilde{e})$ is a singleton, where $type(\tilde{e})$ denotes the set of non terminals occurring in elementary pattern of the form $\tilde{e}X$, and (b) for each non terminal X distinct from *Start*, $Start \Rightarrow_{\mathcal{G}}^* X$ holds where $\Rightarrow_{\mathcal{G}}^*$ is the transitive

⁶ A different terminology is used in [7] where normalized ref-schemas are called pattern schemas.

closure of the relations $\Rightarrow_{\mathcal{G}}^e$ defined by $X \Rightarrow_{\mathcal{G}}^e Y$ if Y occurs in $\theta(X)$ in some pattern $(eY)^{op}$ with e being a child label.

Example 3.2 Consider that (1) E contains the child labels $doc, editor, name, article, book, author, title, isbn$ and (2) \vec{E} contains the references $\overrightarrow{pub}_A, \overrightarrow{pub}_B, \overrightarrow{cite}_A, \overrightarrow{cite}_B, \overrightarrow{pubby}$ and (3) the non terminal symbols are $Start, Editor, Doc, Art, Book, Name, Isb$. Below, the set of rules \mathcal{R} defines a normalized ref-schema.

$$\begin{aligned} \{ \quad & Start ::= (doc\ Doc)^*, (editor\ Editor)^* \\ & Editor ::= (name\ Name)^!, (\overrightarrow{pub}_B\ Book)^*, (\overrightarrow{pub}_A\ Art)^* \\ & Doc ::= (article\ Art)^! + (book\ Book)^! \\ & Art ::= (author\ Name)^+, (title\ Name)^!, (\overrightarrow{cite}_A\ Art)^*, (\overrightarrow{cite}_B\ Book)^* \\ & Book ::= (isbn\ Isb)^!, (\overrightarrow{cite}_A\ Art)^*, (\overrightarrow{cite}_B\ Book)^* + (author\ Name)^+, \\ & \quad (title\ Name)^!, (\overrightarrow{cite}_A\ Art)^*, (\overrightarrow{cite}_B\ Book)^*, (\overrightarrow{pubby}\ Editor)^! \\ & Name ::= \Lambda \quad Isb ::= \Lambda \} \end{aligned}$$

Note that: (i) by condition (1) of the definition, it is not allowed to write a subexpression such as $(title\ Name)^!, (title\ Name)^+$, (ii) the same non terminal symbol may be associated to different labels, for instance $Name$ appears in $(author\ Name)^+$ and $(title\ Name)^!$; (iii) by condition (2a), a label (child label or reference) is always associated to the very same non terminal symbol, for instance the child label $author$ always appears in elementary pattern $(author\ Name)$ and the reference \overrightarrow{cite}_B in $(\overrightarrow{cite}_B\ Book)$. Indeed, condition (1) can be relaxed as well as condition (2a) for references (see [8] for a discussion).

Intuitively, if the elementary pattern $(doc\ Doc)^*$ matches a document at state s , then zero or more doc edges are leaving s . The expression $(article\ Art)^! + (book\ Book)^!$ says that exactly one $article$ edge or else exactly one $book$ edge has source s . The conjunctive expression $(author\ Name)^+, (title\ Name)^!, (\overrightarrow{cite}_A\ Art)^*, (\overrightarrow{cite}_B\ Book)^*$ that defines the “type” Art enforces that at least one $author$ edge, exactly one $title$ edge and possibly, some \overrightarrow{cite}_A and \overrightarrow{cite}_B references leave the state s .

Like for DTDs, normalized ref-schemas have the ability to specify *recursive* data structures, like sequences or trees (see [7]). Expressions of the form $(e_1 X_1, \dots, e_k X_k)^*$ are not allowed in normalized ref-schema but they are part of the definition of (general) ref-schema as defined in [8].

In [8,7] documents conforming to a schema \mathcal{G} , also called instances of \mathcal{G} , are defined as documents matching the expressions given by the schema. Here, for the sake of the presentation, we prefer to define an instance of \mathcal{G} directly as a model of a HML formula $\tau_{\mathcal{G}}$. The definition of an instance \mathfrak{M} of a schema \mathcal{G} requires a first structural property, namely that the “subframe” of \mathfrak{M} generated by child labels is a tree. This property which is independent of the schema \mathcal{G} is expressed by a HML formula $tree$ not presented here (see [7]). Then in a modular manner, we associate a HML wff τ_{exp} to each expression of the schema as follows.

- If exp is Λ then τ_{exp} is $\bigwedge_{\tilde{e} \in \mathcal{E}} \neg \langle \tilde{e} \rangle \top$

- If exp is of the form $exp_1 + \dots + exp_k$ then τ_{exp} is $\tau_{exp_1} \vee \dots \vee \tau_{exp_k}$
- If exp is of the form $(\tilde{e}_1 X_1)^{op_1}, \dots, (\tilde{e}_k X_k)^{op_k}$ then
 τ_{exp} is $\bigwedge_{i=1\dots k} \tau_i \quad \wedge \quad \bigwedge_{\tilde{e} \text{ not in } exp} \neg \langle \tilde{e} \rangle \top$

where if op_i is ! then $\tau_i = \downarrow x \langle \tilde{e}_i \rangle \downarrow y (\@_x [\tilde{e}_i] y)$

if op_i is * then $\tau_i = \top$, and

if op_i is + then $\tau_i = \langle \tilde{e}_i \rangle \top$

Notation: First, $type(\tilde{e})$ is a singleton $\{X\}$ where X is a non terminal symbol, thus we abusively write $type(\tilde{e})$ for X . We also use the notation τ_X instead of $\tau_{\theta(X)}$ and thus when e is a child label, $\tau_{type(e)}$ is the HML formula associated with the regular expression $\theta(type(e))$. When \vec{e} is a reference, $child(\vec{e})$ denotes the set $\{e \mid e \in E \text{ and } type(e) = type(\vec{e})\}$. For instance, with our running example, $type(author) = Name$ and $child(cite_A) = \{article\}$.

Theorem 3.3 [7] *Let \mathcal{G} be a normalized ref-schema and let \mathfrak{M} be a model (document). The model \mathfrak{M} is conforming to \mathcal{G} , denoted $\mathfrak{M} : \mathcal{G}$, iff $\mathfrak{M}, r \models \tau_{\mathcal{G}}$ where the HML wff $\tau_{\mathcal{G}} = tree \wedge \tau_{\mathcal{G}}^E \wedge \tau_{\mathcal{G}}^{\vec{E}}$ with:*

- $\tau_{\mathcal{G}}^E = \@_{root} \left(\tau_{Start} \wedge \bigwedge_{e \in E} G^*[e] \tau_{type(e)} \right)$ and
- $\tau_{\mathcal{G}}^{\vec{E}} = \@_{root} \left(\bigwedge_{\vec{e} \in \vec{E}} G^*[\vec{e}] \downarrow x \left(\bigvee_{e \in child(\vec{e})} \@_{root} F^*\langle e \rangle x \right) \right)$.

Intuitively, the wff $\tau_{\mathcal{G}}^E$ checks that, given a state x reachable by an e child edge, the outgoing edges (child edges as well as references) are the ones allowed by the schema. The wff $\tau_{\mathcal{G}}^{\vec{E}}$ is concerned with reference type checking: it checks that the targets of references have the righth types.

Example 3.4 For the normalized ref-schema \mathcal{G} of our running example, the formula $\tau_{\mathcal{G}}^E$ is

$$\begin{aligned} \@_{root} \quad & (\tau_{Start} \wedge G^*[editor] \tau_{Editor} \wedge G^*[doc] \tau_{Doc} \wedge G^*[article] \tau_{Art} \wedge \\ & G^*[book] \tau_{Book} \wedge G^*[name] \tau_{Name} \wedge G^*[author] \tau_{Name} \wedge \\ & G^*[title] \tau_{Name} \wedge G^*[isbn] \tau_{Isb}) \end{aligned}$$

and the formula $\tau_{\mathcal{G}}^{\vec{E}}$ is

$$\begin{aligned} \@_{root} \quad & ([G^*[\overrightarrow{cite_A}] \downarrow x \@_{root} F^*\langle article \rangle x] \wedge [G^*[\overrightarrow{cite_B}] \downarrow x \@_{root} F^*\langle book \rangle x] \\ & \wedge [G^*[\overrightarrow{pub_A}] \downarrow x \@_{root} F^*\langle article \rangle x] \wedge [G^*[\overrightarrow{pub_B}] \downarrow x \@_{root} F^*\langle book \rangle x] \\ & \wedge [G^*[\overrightarrow{pubby}] \downarrow x \@_{root} F^*\langle editor \rangle x]) \end{aligned}$$

For instance

$$\begin{aligned}
 \tau_{Start} &=_{def} \bigwedge_{\tilde{e} \in \mathcal{E} - \{doc, editor\}} \neg \langle \tilde{e} \rangle \top \\
 \tau_{Name} &=_{def} \bigwedge_{\tilde{e} \in \mathcal{E}} \neg \langle \tilde{e} \rangle \top \\
 \tau_{Editor} &=_{def} \downarrow x \langle name \rangle \downarrow y (@_x[name]y) \wedge \bigwedge_{\tilde{e} \in \mathcal{E} - \{name, \overline{pub}\}} \neg \langle \tilde{e} \rangle \top \\
 \tau_{Doc} &=_{def} (\downarrow x \langle article \rangle \downarrow y @_x[article]y \wedge \bigwedge_{\tilde{e} \in \mathcal{E} - \{article\}} \neg \langle \tilde{e} \rangle \top) \vee \\
 &\quad (\downarrow x \langle book \rangle \downarrow y @_x[book]y \wedge \bigwedge_{\tilde{e} \in \mathcal{E} - \{book\}} \neg \langle \tilde{e} \rangle \top)
 \end{aligned}$$

Indeed, the document given in Figure 1 is a model of $\tau_{\mathcal{G}}$, thus an instance of the schema \mathcal{G} .

4 Constraint satisfiability under non recursive schemas

In our framework, both schemas and constraints are HML wffs which advantageously entails that the problem “does a document \mathfrak{M} conforming to the schema \mathcal{G} exist such that it satisfies the constraint \mathcal{C} ?” can be restated directly as “is $\tau_{\mathcal{G}} \wedge \mathcal{C}$ finitely satisfiable?”. This section is devoted to the presentation of a tableau proof system for testing satisfiability of $\tau_{\mathcal{G}} \wedge \mathcal{C}$. Obviously, the ultimate goal is to provide a terminating proof system. However, HML is not decidable in general [9,22]. Thus we choose to consider several restrictions and relax, in a first step, finiteness over models.

Here we consider *non recursive normalized ref-schemas*, that are schemas not defining a non terminal symbol X by an expression using directly or indirectly X itself. This restriction entails that, wrt accessibility relations associated with child labels, the depth of any instance of $\tau_{\mathcal{G}}$ is bounded. Unfortunately, this is not sufficient to entail that it is finite. The next example exhibits a schema \mathcal{G} and a constraint \mathcal{C} , such that $\tau_{\mathcal{G}} \wedge \mathcal{C}$ does not have the finite model property. Although found independently, this example is strongly related to the query given in [19] to show that the non positive fragment with left-sibling axis of XPath does not have the finite model property.

Example 4.1 The schema is given by $Start := (e \text{ End})^*$ and $End := (\overrightarrow{e} \text{ End})^*$ and the constraint is $\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$ where:

$$\begin{aligned}
 \psi_1 &\text{ is } \langle e \rangle \downarrow y (\langle \overrightarrow{e} \rangle y), & \psi_2 &\text{ is } [e][\overrightarrow{e}] \downarrow y (@_{root} \langle e \rangle \downarrow z (\neg y \wedge @_y \langle \overrightarrow{e} \rangle z)), \\
 \psi_3 &\text{ is } [e] \downarrow x [\overrightarrow{e}][\overrightarrow{e}] \downarrow y @_x \langle \overrightarrow{e} \rangle y, & \psi_4 &\text{ is } [e] \downarrow x [\overrightarrow{e}] \downarrow y (@_x y \vee @_y [\overrightarrow{e}] \neg x).
 \end{aligned}$$

Intuitively, the wff ψ_1 forces the instance to contain at least one e edge followed by a (reflexive) \overrightarrow{e} reference. The wff ψ_2 forces to create a new reference edge starting from the target of an existing reference. This is the main reason of the infiniteness of all the models although it needs to be combined with the wffs ψ_3 and ψ_4 , expressing respectively the transitivity and antisymmetry over \overrightarrow{e} references. One model is given in Figure 2.

The satisfiability problem considered next is the following :

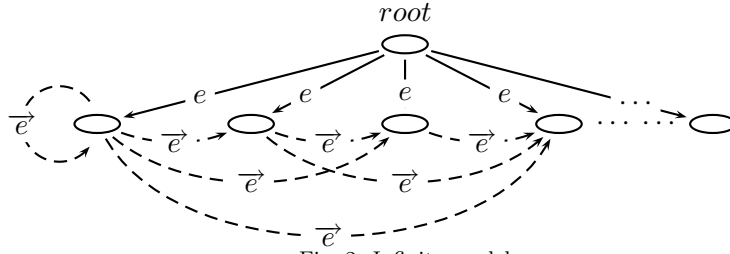


Fig. 2. Infinite model

Input :	a non recursive normalized ref-schema \mathcal{G} ,
	a HML wff \mathcal{C} without F and G modalities
Problem :	Is $\tau_{\mathcal{G}} \wedge \mathcal{C}$ satisfiable ?

In the rest of the presentation, we abusively use the term instance of a schema \mathcal{G} to designate a potentially infinite model of $\tau_{\mathcal{G}}$.

The tableau system

The tableau system defined below is geared to model building rather than refutation. This reversed use of tableau system is well-known [23]. Indeed the tableau system does not use directly the wff $\tau_{\mathcal{G}}$ but rather the wffs τ_{expr} as side effects of some of its rules. This has the advantage to make the rules for the modalities G and F unnecessary as long as they are not allowed in the constraint \mathcal{C} . Adding rules for these modalities is a subject of further study. Our tableau system is a prefixed tableau system: prefixes, here abusively called nominals⁷, are naming states and they are prefixing modal wffs. Roughly, the prefixed formula $n : \varphi$ intends to capture that during the proof, a state n has been created and that φ has to be satisfied at state n . This feature fits very well with modal logic [23,3,17] and particularly with HML [11]. Indeed, in a set of prefixed wffs Φ , the prefixed wffs of the form $n :: \langle \tilde{e} \rangle m$ can be seen as (the frame of) a Kripke model; in other words, it means that, the tableau deduction for Φ builds Kripke models of Φ internally while checking its satisfiability. The reader should pay attention to the fact that, in order to avoid ambiguity, we write $n :: \langle \tilde{e} \rangle m$ when building a new $\langle \tilde{e} \rangle$ edge of a model and we write $n : \langle \tilde{e} \rangle m$ when it has to be checked that there exists a $\langle \tilde{e} \rangle$ edge linking n and m .

We assume wlg that (1) the wff \mathcal{C} is closed rectified: at most one binding $\downarrow x$ occurs for each distinct state variable x and (2) it is in negation normal form: negation is only applied to propositions, nominals and state variables. The rules of the tableau system deal with sets of (generalized) prefixed formulas; each rule is dedicated to one formation rule and thus a (generalized) prefixed formula is distinguished in the denominator of the rule. The rules are partitioned in three groups: the propositional rules, the state variables and hybrid rules (these are the classical ones [11,12]) and finally the transition rules specified for the purpose of satisfiability checking in presence of non recursive schemas.

⁷ For obvious reasons, we prefer not to use the term label for prefix as done in the literature.

$$\begin{array}{l}
 \text{Propositional rules:} \quad (\alpha) \frac{n : \varphi \wedge \psi, \Phi}{n : \varphi, n : \psi, \Phi} \quad (\beta) \frac{n : \varphi \vee \psi, \Phi}{n : \varphi, \Phi \mid n : \psi, \Phi} \\
 \\
 \text{State variable rule:} \quad (Ref) \frac{\Phi}{n : n, \Phi} \quad \text{if } n \text{ occurs in } \Phi \\
 \\
 \text{Hybrid rules :} \quad (@) \frac{n : @_m \varphi, \Phi}{m : \varphi, \Phi} \quad (\downarrow) \frac{n : \downarrow x \varphi, \Phi}{n : \varphi[x \setminus n], \Phi}
 \end{array}$$

We now present the six transition rules. The $\langle E \rangle$ -rule below considers a distinguished prefixed wff of the form $\langle e \rangle \varphi$ with e a child label. This choice rule is defined in the case where φ is not a nominal.:

- the left part of the denominator propagates φ to an existing nominal (state) reachable by an existing e -edge,
- the right part expands the “model” with a new nominal (state) m and a new e -edge from n to m and propagates φ to m . The schema constraint $\tau_{\mathcal{G}}$ is locally enforced by introducing the prefixed wff $m : \tau_{type(e)}$.

$$\langle E \rangle \frac{n : \langle e \rangle \varphi, \Phi}{\begin{array}{c} m : \varphi, \Phi \\ \text{for } n :: \langle e \rangle m \in \Phi \end{array} \mid \begin{array}{c} n :: \langle e \rangle m, m : \tau_{type(e)}, m : \varphi, \Phi \\ \text{for a new } m \end{array}}$$

The next $\langle E_n \rangle$ -rule deals with the case where the formula φ is a nominal:

$$\langle E_n \rangle \frac{n : \langle e \rangle m, \Phi}{\Phi}$$

for $n :: \langle e \rangle m \in \Phi$

The $\langle \vec{E} \rangle$ -rule below considers a distinguished prefixed wff of the form $\langle \vec{e} \rangle \varphi$ with \vec{e} a reference. This choice rule is defined in the case where φ is not a nominal n :

- the left part of the denominator is similar to that of the $\langle E \rangle$ -rule;
- the center part creates a new \vec{e} -reference from n to an existing state m while the condition $f \in child(\vec{e})$ ensures that m is well-typed, in other word it is meant to check that m is a valid target for a \vec{e} -reference;
- the right part creates a new \vec{e} -reference edge leading to a new nominal (state) m . The wff $\pi(\vec{e}, m)$ defined below enforces that m is linked to the root of the “model” by a well-typed path of child edges.

$$\pi(\vec{e}, m) =_{def} \bigvee_{e \in child(\vec{e})} (\bigvee_{ph \in Path(e)} ph \ m)$$

where $Path(e)$ is the set of “modal” paths $\langle e_1 \rangle \cdots \langle e_n \rangle$ such that $Start \Rightarrow_{\mathcal{G}}^{e_1} X_1, \dots, \Rightarrow_{\mathcal{G}}^{e_n} X_n$ holds wrt the ref-schema \mathcal{G} and with $e_n = e$.

For instance, for our running example, $\pi(\overrightarrow{pub}_A, m)$ is $\langle doc \rangle \langle article \rangle m$.

Introducing this wff $\pi(\vec{e}, m)$ is one of the important features of our system: it entails that a dangling state is never introduced. Indeed, it is worth noticing that writing this formula is only possible because the schema \mathcal{G} is non recursive.

$$\langle \vec{E} \rangle \frac{n : \langle \vec{e} \rangle \varphi, \Phi}{\begin{array}{c} m : \varphi, \Phi \\ \text{for } n :: \langle \vec{e} \rangle m \in \Phi \end{array}} \left| \begin{array}{c} n :: \langle \vec{e} \rangle m, m : \varphi, \Phi \\ \text{for } p :: \langle f \rangle m \in \Phi \\ \text{and } f \in \text{child}(\vec{e}) \end{array} \right| \begin{array}{c} \text{root} : \pi(\vec{e}, m), m : \varphi, \\ n :: \langle \vec{e} \rangle m, \Phi \\ \text{for a new } m \text{ and} \\ \pi(\vec{e}, m) \text{ defined above} \end{array}$$

The next $\langle \vec{E}_n \rangle$ -rule takes care of the case where φ is a nominal.

$$\langle \vec{E}_n \rangle \frac{n : \langle \vec{e} \rangle m, \Phi}{\begin{array}{c} \Phi \\ \text{for } n :: \langle \vec{e} \rangle m \in \Phi \end{array}} \left| \begin{array}{c} n :: \langle \vec{e} \rangle m, \Phi \\ \text{for } p :: \langle f \rangle m \in \Phi \\ \text{and } f \in \text{child}(\vec{e}) \end{array} \right|$$

The next two rules deal with the $[\tilde{e}]$ modality.

- The $[\mathcal{E}]$ -rule considers a distinguished prefixed formula of the form $[\tilde{e}]\varphi$ with \tilde{e} being any label. It generates an initial generalized prefixed formula further used to control that φ is propagated to all \tilde{e} -successors of n . A generalized prefix formula is of the form $n : ([\tilde{e}]\varphi, \Delta)$ where Δ stores the \tilde{e} successors of n over which the formula φ has already been propagated.
- The (Y) -rule processes new \tilde{e} -successors of n by propagating φ in order to enforce $[\tilde{e}]\varphi$ at state n .

$$[\mathcal{E}] \frac{n : [\tilde{e}]\varphi, \Phi}{n : ([\tilde{e}]\varphi, \emptyset), \Phi} \quad (Y) \frac{n : ([\tilde{e}]\varphi, \Delta), \Phi}{\{m : \varphi \mid m \in \Delta'\}, n : ([\tilde{e}]\varphi, \Delta \cup \Delta'), \Phi}$$

for $\Delta' = \{m \mid n :: \langle \tilde{e} \rangle m \in \Phi\} - \Delta$

Systematic construction of a \mathcal{G} -tableau \mathbf{T} for \mathcal{C}

A \mathcal{G} -tableau \mathbf{T} for \mathcal{C} is a proof tree, each branch corresponding to an attempt to build an instance of \mathcal{G} satisfying \mathcal{C} , hence a model of $\tau_{\mathcal{G}} \wedge \mathcal{C}$. The systematic construction follows more or less a breath-first traversal strategy with some priority over rule applications.

Stage 1 : Put the prefixed formula $\text{root} : \tau_{\text{Start}} \wedge \mathcal{C}$ at the root of the tableau.

Stage $i + 1$: Choose a leaf node L of the tableau under construction as close as possible to the root of the tableau. Assume that L contains a set Φ of (generalized) prefixed formulas.

If possible, choose a prefixed formula $n : \varphi$ in Φ in order to apply one of the tableau rules other than the (Y) one with the following priority : (1) propositional rules, state variable rule, hybrid rules, (2) $\langle E \rangle$ rules, (3) $\langle \vec{E} \rangle$ rules, (4) \mathcal{E} and expand L by applying the corresponding rule with respect to $n : \varphi$ in all manners, meaning that if the rule is a choice rule, each possible application of the rule is developed, each one leading to create a new descendant for L containing the appropriate prefixed formulas.

Otherwise, apply “simultaneously” the (Y) rule to all possible generalized prefix formulas in Φ .

Proposition 4.2 (Fairness) If a (generalized) prefixed formula $n : \varphi$ belongs to

some leaf L of the tableau under construction at stage i , then it will become the distinguished formula at some later step of the systematic construction.

Note that the systematic tableau construction may go on ad infinitum : the tableau constructed for the schema \mathcal{G} and constraint \mathcal{C} of Example 4.1 is infinite.

Definition 4.3 [Open/Closed \mathcal{G} -Tableau] A \mathcal{G} -tableau T for the constraint \mathcal{C} is a (potentially infinite) tree constructed as described above. A branch⁸ \mathcal{B} is closed iff one of its nodes contains both prefixed wffs of the form $n : \varphi$ and $n : \neg\varphi$, or some statement $n : m$ for $n \neq m$. A branch is open if it is not closed and the tableau T is open iff one of its branches is open (otherwise it is closed).

Soundness and completeness of the tableau system

Due to space limitation, the proofs of soundness and completeness of the tableau system are sketched below. The proofs rest on the notion of \mathcal{G} -Hintikka set whose definition follows the line of and extends the one in [11]. Intuitively, a \mathcal{G} -Hintikka set is a potentially infinite set of prefixed formulas characterizing some model of $\tau_{\mathcal{G}}$ and thus in the proofs, \mathcal{G} -Hintikka sets bridge the gap between open branches of a \mathcal{G} -tableau T and models (potentially infinite instances of \mathcal{G}) of $\tau_{\mathcal{G}}$.

Definition 4.4 [\mathcal{G} -Hintikka sets] Let \mathcal{G} be a schema. A *potentially infinite* set of prefixed formulas H is called a \mathcal{G} -Hintikka set iff it satisfies the following conditions:

- (i) for each atom α and each nominal n , if $n : \alpha \in H$ then $n : \neg\alpha \notin H$, and conversely
- (ii) for each nominal n occurring in H , $n : n \in H$,
for each pair of nominals n and m such that $n \neq m$, $n : m \notin H$,
- (iii) (Tree condition) if $n :: \langle e \rangle m \in H$ then for each n' , e' we have $n' :: \langle e' \rangle m \in H$ implies $n' = n$ and $e' = e$;
(Unique root) there exists exactly one nominal denoted $root$, such that for each $m :: \langle e \rangle n \in H$, we have $n \neq root$, and $root : \tau_{Start} \in H$;
- (iv) (Schema enforcement for child labels) if $n :: \langle e \rangle m \in H$ then $m : \tau_{type(e)}$ belongs to H ,
- (v) (Schema enforcement for references) if $n :: \langle \vec{e} \rangle m \in H$ then there exists n' and $f \in Child(\vec{e})$ such that $n' :: \langle f \rangle m$,
- (vi) if $n : \langle \vec{e} \rangle \varphi \in H$ then there exists at least one nominal m such that $n : \langle \vec{e} \rangle m \in H$ and $m : \varphi \in H$,
- (vii) if $n : [\vec{e}] \varphi \in H$ then for each $m \in \{m' \mid n :: \langle \vec{e} \rangle m' \in H\}$, we have $m : \varphi \in H$.
- (viii) if $n : \varphi \wedge \phi \in H$ then $n : \varphi \in H$ and $n : \phi \in H$,
if $n : \varphi \vee \phi \in H$ then either $n : \varphi \in H$ or $n : \phi \in H$ or both,
- (ix) if $n : @_m \varphi \in H$ then $m : \varphi \in H$
if $n : \downarrow x \varphi \in H$ then $n : \varphi[x/n] \in H$

Next, $Nom(H)$ denotes the set of prefixes (nominals) n occurring in H . A model is associated to a \mathcal{G} -Hintikka set in the obvious manner:

⁸ A branch of T is any path from the root downwards in the proof tree.

Definition 4.5 Let H be a \mathcal{G} -Hintikka set. Then $M_H=(Nom(H), root, R, I_P, \mathcal{I}_{nom})$ is the HML model associated to H where each accessibility relation $r_{\tilde{e}}$ in R is defined by $\{(n, m) \mid n :: \langle \tilde{e} \rangle m \in H\}$, $I_P(p) = \{n \mid n : p \in H\}$, and $\mathcal{I}_{nom}(root) = root$.

The next lemma simply states that the model associated to a \mathcal{G} -Hintikka set satisfies each prefixed formula in H and is a (potentially infinite) instance of \mathcal{G} .

Lemma 4.6 Let \mathcal{G} be a schema. Let H be a \mathcal{G} -Hintikka set. Then :

- (i) if $n : \varphi \in H$ then $M_H, n \models \varphi$.
- (ii) $M_H, root \models \tau_{\mathcal{G}}$
- (iii) For each $m \in Nom(H) - \{root\}$, we have $M_H, root \models (\bigvee_{ph \in Path(e)} ph \ m)$ where $Path(e)$ is defined as before for a child label e .

The proof of Lemma 4.6 is done by induction.

We now establish the dual of Lemma 4.6 stating that, from a model \mathfrak{M} which is a (potentially infinite) instance of \mathcal{G} and satisfies a modal formula \mathcal{C} at its root, it is possible to build a \mathcal{G} -Hintikka set.

Lemma 4.7 Let \mathfrak{M} (assuming $I_N(root) = r$ as usual) be an instance of \mathcal{G} such that $\mathfrak{M}, r \models \mathcal{C}$ where \mathcal{C} is a HML constraint. Then there exists a \mathcal{G} -Hintikka set H such that $M_H = \mathfrak{M}$.

Proof (Sketch) The central idea of the proof is to define an inflationary operator T over sets of prefixed formulas. Intuitively the operator T is defined by making use of the items 4, 6, 7, 8 and 9 of Definition 4.4 as “production” rules guarded by the model \mathfrak{M} . The operator T is used to build an inflationary sequence $\mathcal{F}_{i(i \geq 0)}$ of prefixed formulas such that $\mathcal{F}_i \subseteq \mathcal{F}_{i+1}$. Because of the finite subformula property, there exists k such that for $i \geq k$, $\mathcal{F}_i = \mathcal{F}_k$. By construction, it is easy to show that \mathcal{F}_k is a \mathcal{G} -Hintikka set such that its induced model $M_{\mathcal{F}_k}$ is the model \mathfrak{M} of $\tau_{\mathcal{G}} \wedge \mathcal{C}$. \square

We are now ready to prove that our tableau system is sound which is stated by:

Theorem 4.8 (Soundness) Let T be a \mathcal{G} -tableau build for \mathcal{C} . If \mathcal{B} is an open branch of T then $H_{\mathcal{B}}$ is a \mathcal{G} -Hintikka that satisfies \mathcal{C} where $H_{\mathcal{B}}$ is the set of all prefixed formulas occurring in the branch \mathcal{B} .

Proof (Sketch) Assuming that \mathcal{B} is an open branch of a \mathcal{G} -tableau T for \mathcal{C} , it is sufficient to show that the set of prefixed wffs $H_{\mathcal{B}}$ is a \mathcal{G} -Hintikka set. Indeed, this entails that the induced model of $H_{\mathcal{B}}$ satisfies $\tau_{\mathcal{G}} \wedge \mathcal{C}$, by Lemma 4.6 since $root : \mathcal{C} \in H_{\mathcal{B}}$. To this end, assume that $H_{\mathcal{B}}$ is not a \mathcal{G} -Hintikka set. Then this means that one of the items of definition 4.4 is violated. The proof is then developed by a simple case study. \square

We now turn to proving that the tableau system is complete which is stated by:

Theorem 4.9 (Completeness) If there exists an instance \mathfrak{M} of \mathcal{G} satisfying the constraint \mathcal{C} (i.e. such that $\mathfrak{M}, r \models \mathcal{C}$) then the \mathcal{G} -tableau T for \mathcal{C} is open.

We need the following intermediate results. The first one states that adding to a \mathcal{G} -Hintikka set some prefixed “path” formula “satisfied” by M_H , leads to a \mathcal{G} -Hintikka set.

Lemma 4.10 *If H is a \mathcal{G} -Hintikka set then H^+ defined below is a \mathcal{G} -Hintikka such that $M_H = M_{H^+}$.*

$$H^+ = H \cup \{(n_0 : \langle e_1 \rangle \langle e_2 \rangle \dots \langle e_k \rangle n_{k+1}) \mid (n_i :: \langle e_{i+1} \rangle n_{i+1}) \in H \text{ for } i = 0 \dots k\}$$

Now, we show that the \mathcal{G} -Hintikka set associated with \mathfrak{M} denoted $H_{\mathfrak{M}}$ can be used as a guide to “build” (or identify) an open branch \mathcal{B} of T . Of course, the \mathcal{G} -Hintikka set $H_{\mathcal{B}}$ associated to the open branch \mathcal{B} and build using $H_{\mathfrak{M}}$ as a guide, is not necessarily equal to $H_{\mathfrak{M}}$. Given a proof segment \mathcal{P} (which is specified by a path from the root of T to some proof node), $H_{\mathcal{P}}$ denotes the set of all prefixed formulas occurring in \mathcal{P} excluding the generalized ones. The set of nominals occurring in $H_{\mathcal{P}}$ is denoted $Nom(\mathcal{P})$.

An *embedding* of the proof segment \mathcal{P} in a model \mathfrak{M} is a mapping μ from $Nom(\mathcal{P})$ to $Nom(H_{\mathfrak{M}}^+)$ such that μ is total, injective, and $\mu(H_{\mathcal{P}}) \subseteq H_{\mathfrak{M}}^+$, where $\mu(H_{\mathcal{P}})$ is defined in the natural way. Note that, for the purpose of the proof and precisely to deal with $\pi(\langle \vec{e} \rangle, m)$ prefixed formula introduced by the $\langle \vec{E} \rangle$ rule, we use here the \mathcal{G} -Hintikka set $H_{\mathfrak{M}}^+$ rather than $H_{\mathfrak{M}}$.

A proof segment \mathcal{P} is said to be a *pre-model* of \mathfrak{M} if there exists at least one embedding function of the proof segment \mathcal{P} in the model \mathfrak{M} . An *extension* of a proof segment \mathcal{P} is a segment having \mathcal{P} as a prefix.

In order to prove the completeness theorem above we need to show that if the proof segment \mathcal{P} is a pre-model of \mathfrak{M} and \mathcal{P} is not a branch (i.e. a saturated segment) then it can be extended (by the systematic construction strategy) and at least one of the extensions of the proof segment \mathcal{P} is a pre-model of \mathfrak{M} . Formally:

Lemma 4.11 Given a schema \mathcal{G} and a constraint \mathcal{C} . Let \mathfrak{M} be an instance (may be an infinite one) of \mathcal{G} satisfying the constraint \mathcal{C} (i.e. $\mathfrak{M}, r \models \tau_{\mathcal{G}} \wedge \mathcal{C}$). Let T be a \mathcal{G} -tableau for \mathcal{C} and \mathcal{P} a proof segment of T . We have:

If \mathcal{P} is a pre-model of \mathfrak{M} then either \mathcal{P} is a branch (saturated segment) or there exists at least one extension of \mathcal{P} which is a pre-model of \mathfrak{M} .

Proof (Sketch) Assume that the proof node P is the extremity of segment \mathcal{P} . Use the priority rule to expand the proof node P . From the definition of \mathcal{G} -Hintikka sets, it is easy to show that the extension of \mathcal{P} is a pre-model of \mathfrak{M} when P is expanded by applying one of the rules (α) or (β) or ($@$) or (\downarrow) or (\mathcal{E}) or (Y). The other possible rule applications need to be studied more carefully but do not present any difficulty. \square

Proof (Theorem 4.9) Lemma 4.11 entails completeness of our tableau system as follows: if $\tau_{\mathcal{G}} \wedge \mathcal{C}$ is satisfiable and \mathfrak{M} is one of its HML model then, $root : \tau_{\mathcal{G}} \wedge \mathcal{C}$ is the root of any \mathcal{G} -tableau T for \mathcal{C} and it is of course a pre-model of \mathfrak{M} ; thus Lemma 4.11 entails that there exists an open branch in T ; hence T is open, which concludes the proof. \square

5 Discussion and further research direction

This section is devoted to discuss some of the restrictions made in the presentation.

First, the tableau system is presented for (non recursive) normalized ref-schemas. In [8], we show how a general ref-schema (allowing full regular expressions) can be equivalently translated by a normalized ref-schema together with a set of structural constraints expressed by HML formulas. This implies that the tableau system can be used for checking constraints satisfiability in presence of general (non recursive) ref-schemas although a slight extension need to be done in order to deal with the modality G used in the structural constraints.

We now discuss the absence of the modalities F and G in the constraint expressions. The tableau system can be extended to integrate these modalities. The extension is quite obvious for F and G restricted to child labels because we can utilize the fact that the schemas are non recursive. However a carefull writting of the rules for F and G in the general case is required in order to control potential “loops”: roughly, attempting to check the satisfiability of $n : F\varphi$ may lead, after several steps to check again for the satisfiability of $n : F\varphi$. The idea is to design the rules in a way similar to what has been done here for the $[\tilde{e}]$ modality.

We now turn to the main limitation of the tableau system presented in the paper: it is sound and complete for checking satisfiability, but not *finite* satisfiability which is of course the relevant notion for XML databases. As usual, two directions of investigation are possible: (1) the syntactic approach leads to exhibit conditions over the constraint \mathcal{C} in order to ensure the finite satisfiability, (2) the second direction is more “proof” oriented and involves enreaching the tableau system itself in order to prune infinite open branches. We are currently investigating a combined approach. Indeed, non recursive ref-schemas are not sufficient to ensure decidability: the tiling problem can be encoded in our framework very easily. Thus syntactic restrictions over constraints are mandatory. We are currently investigating such restrictions which slightly relax those given in [16]⁹ and take care of multimodality which is also a source of undecidability [27]¹⁰. These restrictions still allow one to write constraints having infinite models and thus we are currently modifying our tableau system in order to introduce a check for potentially infinite branches based on some kind of bisimulation property over pairs of sets of prefixed formulas.

References

- [1] ALECHINA N., “Semi-structured information: a modal logic approach“, *Tech. Report CSR-97-08, School of Computer Science, The University of Birmingham*, 1997.
- [2] ALECHINA N., DEMRI S., DE RIJKE M., “A Modal Perspective on Path Constraints“, *J. Log. Comput.* 13(6), 2003, p. 939–956.
- [3] ARECES C., BLACKBURN P., MARX M., “Hybrid Logics: Characterization, Interpolation and Complexity“, *J. of Symbolic Logic*, Vol. 66, Num. 3, 2001, p. 977–1010.
- [4] ARECES C., BLACKBURN P., MARX M., “A Road-map on Complexity for Hybrid Logics“, *Proc. of the 8th Annual Conf. of the EACSL, Madrid*, LNCS, Springer, 1999, p. 307–321.
- [5] ARENAS M., FAN W. , LIBKIN L., “Consistency of XML Specifications“, *Inconsistency Tolerance*, 2005, p. 15–41.
- [6] BENEDIKT M., FAN W., GEERTS F., “XPath satisfiability in the presence of DTDs.“, *Proc. of the Twenty-fourth ACM SIGACT- SIGMOD-SIGART Symp. on Principles of Database Systems*, 2005, p. 25-36.

⁹ [16] show that the source of undecidability is the $\square\downarrow\square$ pattern.

¹⁰Note that the encoding of the tiling problem provided in [16] uses three accessibility relations.

- [7] BIDOIT N., CERRITO S., THION V., “A first step towards modeling semistructured data in hybrid multimodal logic“, *J. of Applied Non-Classical Logics*, Vol. 14, Num. 4, 2004, p. 447–476.
- [8] BIDOIT N., COLAZZO D., “XML schemas capturing references“, BDA 2006.
- [9] BLACKBURN P., SELIGMAN J., “Hybrid Languages“, *J. of Logic, Language and Information*, Vol. 4, 1995, p. 251–272.
- [10] BLACKBURN P., “Representation, Reasoning, and Relational Structures: a Hybrid Logic Manifesto“, *Logic J. of the IGPL*, Vol. 8, Num. 3, 2000, p. 339–365.
- [11] BLACKBURN P., “Internalizing labelled deduction“, *Journal of Logic and Computation*, Vol. 10, Num. 1, 2000, p. 137–168.
- [12] BLACKBURN P., MARX M., “Tableaux for Quantified Hybrid Logic“, *Proc. Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods*, 2002, p. 38–52.
- [13] BOJANCZYK M. DAVID C. MUSCHOLL A. SCHWENTICK TH. , SEGOUFIN L., “Two-variable logic on data trees and XML reasoning“, *PODS*, 2006, p. 10–19.
- [14] CALVANESE D., DE GIACOMO G. , LENZERINI M., “Representing and Reasoning on XML Documents: A Description Logic Approach.“, *J. Log. Comput.* 9(3), 1999, p. 295–318.
- [15] CARDELLI L. , GHELLI G., “A Query Language Based on the Ambient Logic“, *ESOP*, 2001, p. 1–22.
- [16] TEN CATE B. , FRANCESCHET M. “On the Complexity of Hybrid Logics with Binders“, *CSL*, 2005, p. 339–354.
- [17] DE GIACOMO G., MASSACCI F., “Combining Deduction and Model Checking into Tableaux and Algorithms for Converse-PDL“, *J. of Information and Computation*, Vol. 162, Num. 1–2, 2000, p. 117–137.
- [18] DEMRI S., “(Modal) Logics for Semistructured Data“, *Invited talk at “Third Workshop on Methods for Modalities (M4M-3)”*, 2003.
- [19] FAN W., GEERTS F., “Satisfiability of XPath Queries with Sibling Axes“, *Database Programming Languages, 10th Int. Symp.* Vol. 3774 of LNCS, 2005, p. 122–137.
- [20] FAN W. , SIMÉON J., “Integrity Constraints for XML“, *PODS*, 2000, p. 23–34.
- [21] FITTING M., *Proof Methods for Modal and Intuitionist Logic*, D.Reidel Publishing Company, 1983.
- [22] GORANKO V. “Hierarchies of Modal and Temporal Logics with Reference Pointers“, *Journal of Logic, Language and Information* Vol. 5 Num. 1, 1996, p. 1–24.
- [23] GORE R., “Tableau methods for modal and temporal logics“, ReportNum.TR-ARP-15-95 Australian National University, 1995.
- [24] KRIPKE S., “Semantic Considerations on Modal Logic“, LINSKI L., Ed., *Reference and Modality*, London, 1971, Oxford University Press, p. 63–72.
- [25] MARX M. “XPath and Modal Logics of Finite DAG’s“, *TABLEAUX*, 2003, p. 150–164.
- [26] MURATA M. , LEE D. , MANI M., “Taxonomy of XML Schema Languages Using Formal Language Theory“, *Proc. of the Extreme Markup Languages Conference*, 2001.
- [27] MUNDHENK M. , SCHNEIDER T., “Undecidability of Multi-modal Hybrid Logics“, *Hyla, LICS 2006 Workshop*
- [28] WALLEN L. A., *Automated Deduction in Nonclassical Logics: Efficient Matrix Proof Methods for Modal and Intuitionistic Logics*, MIT Press, 1990.
- [29] “XML Schema“, On <http://www.w3.org/TR/xmlschema-0/>, W3C Recommendation.