XPath and Modal Logics of Finite DAG's

Maarten Marx*

ILLC, Universiteit van Amsterdam, The Netherlands
marx@science.uva.nl

Abstract. XPath, CTL and the modal logics proposed by Blackburn et al, Palm and Kracht are variable free formalisms to describe and reason about (finite) trees. XPath expressions evaluated at the root of a tree correspond to existential positive modal formulas. The models of XPath expressions are finite ordered trees, or in the presence of XML's ID/IDREF mechanism graphs. The ID/IDREF mechanism can be seen as a device for naming nodes. Naming devices have been studied in hybrid logic by nominals. We add nominals to the modal logic of Palm and interpret the language on directed acyclic graphs. We give an algorithm which decides the consequence problem of this logic in exponential time. This yields a complexity result for query containment of the corresponding extension of XPath.

1 Introduction

This paper is about reasoning in languages interpreted on finite trees and directed acyclic graphs (DAGs). These finite structures are the core interest in both theoretical linguistics (parsing a sentence leads to a finite tree or DAG) and in the world of XML databases (an XML document is modeled as a finite tree, or in the presence of ID/IDREF attributes as a finite graph). In the field of XML databases, a key problem is the equivalence or containment of XPath expressions possibly in the presence of a Document Type Definition (DTD). This problem can be seen as an instance of the consequence problem in logic. We study the complexity of this problem in the setting in which the relevant structures are finite trees or DAGs. The language used to describe these structures is the modal tree language proposed by Palm [20].This is a fragment of Propositional Dynamic Logic (PDL) with four basic programs corresponding to the four basic movements in finite ordered trees: mother, daughter, left sister and right sister.

The novelty of this paper is the addition of nominals to this language in order to simulate XML's ID/IDREF mechanism, and the generalization of the class of models from trees to rooted DAGs. The main result is that the satisfiability problem interpreted on rooted DAGs is in EXPTIME.

We started our work by building a tableau system for a fragment of the language. But this system seemed to be horribly inefficient, as it had to build a tree from the root. It is straightforward to devise for every natural number n,

* Research funded by NWO grant 612.000.106.

M. Cialdea Mayer and F. Pirri (Eds.): TABLEAUX 2003, LNAI 2796, pp. 150–164, 2003.

a satisfiable formula of size $O(n^2)$ whose minimal model is a binary branching tree of depth 2^n, a structure with 2^{2^n} many nodes [5]. The known lower bound of the satisfiability problem was EXPTIME, so somewhere there was something wrong.

The decision algorithm presented here searches for a pseudomodel of the formula to be satisfied. The pseudomodel is such that it can be transformed into a finite structure in which the original formula is still satisfied. The pseudomodel on the other hand has size bound by just a single exponential in the input. Algorithms using pseudomodels are the natural alternative when either tableaux do not (or not easy) terminate or when they take more time or space than needed to solve the problem. Pseudomodels are often used in decision procedures in modal and temporal logic, but also in e.g., the family of guarded fragments, see [5] for a number of examples.

Our work is related to many areas in logic. We mention the most relevant. The EXPTIME lower bound of PDL [12] transfers to finite models and our similarity type. Upper bounds for (converse) PDL do not transfer to our case, since we are working on *finite* trees and DAGs, whence have a different logic. For instance, versions of Löb's axiom hold on finite trees and DAGs, but fail for PDL. Our EXPTIME algorithm uses several features of the one for PDL by Pratt [21]. The novelty of this paper is the adaptation of Pratt's method to the finite case: the new part in the algorithm prevents building models with cycles or infinite paths. Alternative EXPTIME lower bounds can be extracted from results about XPath query containment under DTD's by Neven and Schwentick [18]. For unordered finite trees, complexity results for part of the language can be obtained by an interpretation into CTL* [11]. The connection between CTL and XPath is first made in [17]. The addition of nominals to the language places this work in the tradition of hybrid logic [4]. The algorithm presented here uses as a subroutine (cf. Figure 4) the decision procedure for Palms language on finite trees from [6]. Alechina, de Rijke and Demri [2] analyze path constraints for semistructured data and obtain complexity results by an embedding into converse PDL with nominals. The difference with the present work is twofold. Firstly, they consider arbitrary graphs as models. Secondly, they consider edge labeled structures, while we are interested in node labeled structures (like XML documents). This shows in the difference in signatures: we consider just the four basic moves in a tree and allow whatever node label (i.e., propositional variable); [2] has no propositional variables, but arbitrary edge labels (i.e., atomic programs).

Organization. The next section presents two modal logics of finite trees and establishes the relation to first and second order logic of trees. Then follow two sections about XML motivating our work. These two sections are not needed to understand the technical part of the paper. After that we concentrate on the decision algorithm and its correctness proof. We conclude with a number of open problems.

2 Modal Logic of Finite Trees

We first recall the modal logic of finite trees proposed by Marcus Kracht in [14, 15]. The language will be called $\mathcal{L}_K$. $\mathcal{L}_K$ is a propositional modal language identical to Propositional Dynamic Logic (PDL) [13] over four basic programs: $\leftarrow$, $\rightarrow$, $\uparrow$ and $\downarrow$ which explore the left-sister, right-sister, mother-of and daughter-of relations. Recall that PDL has two sorts of expressions: programs and propositions. We suppose we have fixed a non-empty, finite or countably infinite, set of atomic symbols A whose elements are typically denoted by p. $\mathcal{L}_K$'s syntax is as follows, writing π for programs and ϕ for propositions:

$$\pi ::= \leftarrow | \rightarrow | \uparrow | \downarrow | \pi;\pi | \pi \cup \pi | \pi^* | ?\phi$$
$$\phi ::= p | \top | \neg\phi | \phi \wedge \phi | \langle\pi\rangle\phi.$$

We employ the usual boolean abbreviations and write $[\pi]\phi$ instead of $\neg\langle\pi\rangle\neg\phi$.

$\mathcal{L}_K$ is interpreted on *finite ordered trees* whose nodes are *labeled* with symbols drawn from A. We assume that the reader is familiar with finite trees and such concepts as 'daughter-of', 'mother-of', 'sister-of', 'root-node', 'terminal-node', and so on. If a node has no sister to the immediate right we call it a last node, and if it has no sister to the immediate left we call it a first node. Note that the root node is both first and last. The root node will always be called *root*. A labeling of a finite tree associates a subset of A with each tree node.

Formally, we present finite ordered trees as tuples $\mathbf{T} = (T, R_{\rightarrow}, R_{\downarrow})$. Here T is the set of tree nodes and $R_{\rightarrow}$ and $R_{\downarrow}$ are the immediate right-sister and daughter-of relations respectively. A pair $\mathfrak{M} = (\mathbf{T}, V)$, where $\mathbf{T}$ is a finite tree and $V : A \longrightarrow Pow(T)$, is called a *model*, and we say that V is a *labeling function* or a *valuation*. Given a model $\mathfrak{M}$, we simultaneously define a set of relations on $T \times T$ and the interpretation of the language $\mathcal{L}_K$ on $\mathfrak{M}$:

$$\begin{array}{ll} R_{\uparrow} = R_{\downarrow}^{-1} & R_{\pi\cup\pi'} = R_{\pi} \cup R_{\pi'} \\ R_{\leftarrow} = R_{\rightarrow}^{-1} & R_{\pi;\pi'} = R_{\pi} \circ R_{\pi'} \\ R_{\pi^*} = R_{\pi}^* & R_{?\phi} = \{(t,t) \mid \mathfrak{M}, t \models \phi\} \end{array}$$

$$\begin{array}{rll} \mathfrak{M}, t \models p & \text{iff} & t \in V(p), \text{ for all } p \in \mathrm{A} \\ \mathfrak{M}, t \models \top & \text{iff} & t \in T \\ \mathfrak{M}, t \models \neg\phi & \text{iff} & \mathfrak{M}, t \not\models \phi \\ \mathfrak{M}, t \models \phi \wedge \psi & \text{iff} & \mathfrak{M}, t \models \phi \text{ and } \mathfrak{M}, t \models \psi \\ \mathfrak{M}, t \models \langle\pi\rangle\phi & \text{iff} & \exists t' \, (tR_{\pi}t' \text{ and } \mathfrak{M}, t' \models \phi). \end{array}$$

For any formula ϕ, if there is a model $\mathfrak{M}$ such that $\mathfrak{M}, \mathit{root} \models \phi$, then we say that ϕ is *satisfiable*.

We note that we could have generated the same language by taking $\downarrow$ and $\rightarrow$ as primitive programs and closing the set of programs under converses. We use

a number of formulas as abbreviations:

$$t \models \mathit{root} \iff t \models \neg\langle\uparrow\rangle\top \iff t \text{ is the root}$$
$$t \models \mathit{leaf} \iff t \models \neg\langle\downarrow\rangle\top \iff t \text{ is a terminal node}$$
$$t \models \mathit{first} \iff t \models \neg\langle\leftarrow\rangle\top \iff t \text{ is a first node}$$
$$t \models \mathit{last} \iff t \models \neg\langle\rightarrow\rangle\top \iff t \text{ is a last node}$$

We now discuss the expressivity of this and related languages. First two examples: (1) says that every a node has a b and a c daughter, in that order, and no other daughters; and (2) says that every a node has a b first daughter followed by some number of c daughters, and no other daughters.

(1) $a \rightarrow \langle\downarrow\rangle(\mathit{first} \wedge b \wedge \langle\rightarrow\rangle(c \wedge \mathit{last}))$

(2) $a \rightarrow \langle\downarrow\rangle(\neg\langle\leftarrow\rangle\top \wedge b \wedge \langle(\rightarrow; ?c)^*\rangle \mathit{last})$.

$\mathcal{L}_K$ can express properties beyond the power of the first order logic of ordered labeled trees[1] For example, it can express the property of having an odd number of daughters: $\langle\downarrow\rangle(\mathit{first} \wedge \langle(\rightarrow;\rightarrow)^*\rangle \mathit{last})$.

Palm [20, 19] proposed a fragment of $\mathcal{L}_K$ which is functionally complete with respect to first order logic of ordered labeled trees (an extension of results by Schlingloff [24]). There are two equivalent formulations (cf., [6]) of this language, which we both denote by $\mathcal{L}_P$. The first is by restricting the set of programs to

$$\pi ::= \leftarrow | \rightarrow | \uparrow | \downarrow | ?\phi; \pi | \pi^*.$$

The second is more economic in its modal operators and resembles temporal logic: let $\mathcal{L}_P$ be the modal language with the following four binary modal operators: for $\pi \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$, $\mathfrak{M}, t \models \mathit{Until}_\pi(\phi, \psi)$ iff there exists a t' such that $tR_{\pi^+}t'$ and $\mathfrak{M}, t' \models \phi$ and for all t'' such that $tR_{\pi^+}t''R_{\pi^+}t'$ it holds that $\mathfrak{M}, t'' \models \psi$.

Present proposals for XPath [8] don't go beyond first order expressivity. For that reason we focus on $\mathcal{L}_P$ from now on. We study the complexity of the consequence problem: $\Gamma \models \phi$ (ϕ true at every state on each finite ordered tree on which all of Γ is true at every state). For finite Γ, this reduces to the satisfiability problem because $\Gamma \models \phi$ if and only if it is not the case that $[\downarrow^*]\Gamma \wedge \langle\downarrow^*\rangle\neg\phi$ is satisfiable. We will improve on the following theorem:

Theorem 1 ([6]). *The satisfiability problem for $\mathcal{L}_P$ is in EXPTIME.*[2]

We note the remarkable fact that the satisfiability for the equally expressive first order logic on finite trees is decidable but with a non–elementary lower bound [23].

[1] That is first order logic in the signature with binary $R^*_\downarrow, R^*_\rightarrow$ and countably many unary predicates, interpreted on labeled ordered trees.

[2] EXPTIME is the class of all problems solvable in exponential time. A problem is solvable in exponential time if there is a deterministic exponentially time bounded Turing machine that solves it. A deterministic Turing machine is exponentially time bounded if there is a polynomial $p(n)$ such that the machine always halts after at most $2^{p(n)}$ steps, where n is the length of the input.

3 XML and XPath

XML is a new standard adopted by the World Wide Web Consortium (W3C) to complement HTML for data exchange on the web. In its simplest form XML looks just like HTML. The main difference is that the user can define its own tags and specifies a Document Type Definition (DTD) which serves as a grammar for the underlying XML document. Figure 1 contains a DTD and an XML document that conforms to it. An XML document is most naturally viewed as a finite ordered node–labeled tree. The tag–names form the labels of the non–terminal nodes and the terminals are labeled with the data (in our example of type `CDATA`). We assume familiarity with these concepts, for an introduction cf. e.g., [1]. XPath is a simple language for navigating an XML tree and selecting a set of element nodes [7]. It's grammar resembles the file selection mechanism in UNIX. As an example, the XPath expression `/a//b[*/c]/g` selects nodes labeled with `g` (`g`–nodes for short) that are children of `b`–nodes, which have an `c`–node as a grandchild and which are themselves descendants of the root `a`–node. A clear explanation of the semantics of XPath is given in [3]. XPath queries starting with the root symbol `/` can easily be translated into expressions in the positive existential fragment of $\mathcal{L}_P$. `/a//b[*/c]/g` selects the same nodes as

$$g \wedge \langle\uparrow\rangle(b \wedge \langle\downarrow\rangle\langle\downarrow\rangle c \wedge \langle\uparrow^*\rangle(a \wedge \mathit{root})).$$

```
1. <!ELEMENT Collection (Painter+)>
2. <!ELEMENT Painter  (Name, Painting*)>
3. <!ELEMENT Name  CDATA  >
4. <!ELEMENT Painting CDATA>

<Collection>
  <Painter>
    <Name> Rembrandt </Name>
    <Painting> de Nachtwacht </Painting>
    <Painting> de Staalmeesters </Painting>
  </Painter>
  <Painter>
    <Name> Vermeer </Name>
    <Painting> het Melkmeisje </Painting>
  </Painter>
</Collection
```

Fig. 1. An XML DTD and document

DTD's can also be translated into $\mathcal{L}_K$. The DTD from Figure 1 translates into[3]

$$\begin{aligned} \mathit{Collection} &\rightarrow \langle\downarrow;?\mathit{first};?\mathit{Painter};(\rightarrow;?\mathit{Painter})^*\rangle\mathit{last}.\\ \mathit{Painter} &\rightarrow \langle\downarrow;?\mathit{first};?\mathit{Name};(\rightarrow;\mathit{Painting})^*\rangle\mathit{last}.\\ \mathit{Name} &\rightarrow \langle\downarrow;?\mathit{first};?\mathit{CDATA}\rangle\mathit{last}.\\ \mathit{Painting} &\rightarrow \langle\downarrow;?\mathit{first};?\mathit{CDATA}\rangle\mathit{last}. \end{aligned}$$

These translations (which can be performed in polynomial time) make that the containment problem of XPath expressions under a DTD can be reduced to the consequence problem in the modal logic of finite ordered trees. This problem has received quite some attention lately [25, 9, 17, 18]. The situation here is quite comparable to that in description logic: an effort is made to map out the complexity landscape for a great number of XPath fragments. The result which is of interest here is that query containment under a DTD is EXPTIME hard for XPath expressions in which one can use `/,//,|` or `/,//,[],*`. When the non-deterministic operators `//,|,*` are left out (leaving only `/,[]`) the problem is complete for CO–NP. Both results are in [18].

Theorem 1 now yields a matching upper bound for a large extension of these XPath fragments. Note that XPath statements correspond to existential positive modal formulas. But Theorem 1 works for the whole modal language which is of course closed under full negation, but also can express until–like constructions. We can now consider queries like

- select all A that only have B children;
- select all couples with a completely Greek descendant line (in a genealogy tree in which nationality is coded).

The first uses negation, the second the until construction.

The part of the landscape that has been investigated until now views XML documents as trees. But in the presence of XML's ID/IDREF mechanism they are really graphs. We turn to these models in the next section.

4 From Trees to DAGs

So far we have discussed XML documents as if they were trees. But XML contains a mechanism for defining and using references and, hence for describing graphs rather than trees. XML allows the association of unique identifiers to elements as the value of a certain attribute. These are attributes of type ID, and the referencing is done with an attribute of type IDREF. How this is done exactly is not important for our discussion. Figure 2 contains a DTD[4] using this

[3] This DTD translates to $\mathcal{L}_P$. But we need $\mathcal{L}_K$ to translate a rule like `<!ELEMENT Collection (Painter,Painting)+>`. For lack of space we cannot give the translation algorithm. For that see [16].

[4] Instead of the official but rather cumbersome, `<!ELEMENT Countries (State,City*)>` we simply write the equivalent context free grammar rule `Countries` $\rightarrow$ `(State,City*)`.

```
countries  → (state(id)*,city*)
state      → name
city       → (name,capitol_of)
capitol_of → state(idref)
name       → CDATA.
```

```
<countries>
   <state ID=A1>
      <name Holland />
   </state>
   <city>
       <name Amsterdam />
       <capitol_of IDREF=A1 />
   </city>
</countries>
```

Fig. 2. A DTD and an XML document with ID/IDREF

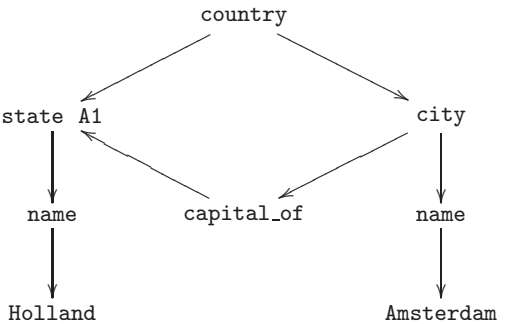

Fig. 3. Acyclic graph for the document in Figure 2

mechanism, and a document which conforms to it. The corresponding acyclic graph is drawn in Figure 3. More abstractly, the models we consider are node labeled graphs in which the nodes may have a unique name besides their label. In modal logic names for states are known as *nominals* and modal logics containing names are referred to as *hybrid logics* [4]. In a modal language a nominal is nothing but a special propositional variable which can only be true at exactly one state. Modal languages with the Difference operator D can express that p behaves like a nominal by stating (here $\mathsf{E}\phi$ abbreviates $\phi \vee \mathsf{D}\phi$):

$$\mathsf{E}(p \wedge \neg \mathsf{D}p).$$

The difference operator D is defined by $\mathfrak{M}, t \models \mathsf{D}\phi$ iff there exists a $t' \neq t$ such that $\mathfrak{M}, t' \models \phi$. On finite ordered trees $\mathsf{D}\phi$ is term definable as

$$\mathsf{D}\phi \equiv \langle \downarrow^+ \rangle \phi \vee \langle \uparrow^+ \rangle \phi \vee \langle \uparrow^* ; (\leftarrow^+ \cup \rightarrow^+) ; \downarrow^* \rangle \phi.$$

So in a sense, we have nominals in our modal language. But a referencing mechanism on trees is not very interesting nor can we make the connection with the XML graph models. Instead of interpreting the modal language of trees on arbitrary graphs we decided to make a smaller move, remaining as close to trees as possible. Here's the definition. We call a directed acyclic graph (DAG) $(N, R_\downarrow)$ *rooted* if there exists an $r \in N$ without ancestors and r is the ancestor of each $n \in N$. Note that a rooted DAG is a tree if every node except the root has exactly

one parent. It is useful to distinguish nodes with one parent from nodes with multiple parents. The latter correspond to nodes with a name. For $NOM \subseteq N$, we call a structure $(N, R_\downarrow, NOM)$ a nominalized rooted DAG (NDAG for short) if $(N, R_\downarrow)$ is a rooted DAG and all elements in $N \setminus (NOM \cup \{root\})$ have exactly one parent.

The restriction to NDAG's is rather natural from an XML point of view, and similar in spirit to the restriction to trees encountered in the literature. Depending on certain syntactic properties of the DTD, we can restrict the class of models accordingly. If the DTD contains no ID/IDREF only trees have to be considered. If the DTD does not specify a *cycle* of naming and referencing only NDAG's need to be considered. This we will do in the next section.

We note that on DAGs there are strictly less validities than on trees. For instance, $\langle\downarrow;\uparrow\rangle\phi \rightarrow \phi$ is valid on trees but not on DAGs. Moreover $\mathcal{L}_K$ is not strong enough to capture all first order properties of NDAG's. For instance, $\exists yzw(y \neq z \wedge xR_\downarrow y \wedge yR_\downarrow w \wedge xR_\downarrow z \wedge zR_\downarrow w)$ is not expressible by an $\mathcal{L}_K$ formula as an easy bisimulation argument shows. Of course on trees, this formula is not satisfiable, whence simply expressible by $\bot$.

5 Deciding the Modal Logic of Finite Rooted Nominalized DAGs

At present we do not know the complexity of the full PDL language nor of Palms fragment with nominals on ordered DAGs. We make a restriction common in the literature on XPath query containment and remove the two sister axis[5] from the language $\mathcal{L}_P$. To this language we add a modal constant id and nominals and interpret it on NDAG's. Formally, there is a special set of propositional variables called nominals. On an NDAG $(N, R_\downarrow, NOM)$ each nominal is interpreted as a singleton subset of NOM. The interpretation of the modal constant id is exactly the set NOM. We call the resulting logic $\mathcal{L}_{DAG}$[6]. The $\mathcal{L}_{DAG}$ consequence problem consists of all pairs (Γ, χ) with $\Gamma \cup \{\chi\}$ a finite set of $\mathcal{L}_{DAG}$ formulas such that $\Gamma \models \chi$ on finite NDAG's.

The following three validities are noteworthy. (3) states that all nominals are interpreted in the set NOM; (4) that there are no cycles and (5) that nodes which are not in the set NOM have at most one parent.

(3) $\quad i \rightarrow id$ for i a nominal

(4) $\quad i \rightarrow \neg\langle\downarrow^+\rangle i$ for i a nominal

(5) $\neg id \rightarrow (\langle\uparrow\rangle\phi \rightarrow [\uparrow]\phi)$.

[5] On ordered DAGs the interpretation of the sister relation is problematic: should they share one or all parents? In the former case the tree validity $\langle\uparrow\rangle\phi \rightarrow [\leftarrow^* \cup \rightarrow^*]\langle\uparrow\rangle\phi$ does not hold. In the latter, we cannot mark first and last nodes anymore. We note that without the sister axis DTD's cannot be expressed anymore. Thus the present result only yields a decision procedure for XPath root queries without a DTD.

[6] Hybrid logics usually have besides nominals also the satisfaction operator @. Here we do not add it because $@_i\phi$ is term definable as $\langle\uparrow^*\rangle(root \wedge \langle\downarrow^*\rangle(i \wedge \phi))$.

Theorem 2. *The $\mathcal{L}_{DAG}$ consequence problem is in EXPTIME.*

The proof consists of a linear reduction and a decision algorithm. The reduction removes the transitive closure operation by adding new propositional symbols. Similar techniques are employed in [22, 10] for obtaining normalized monadic second order formulas. The reduction is most simply presented in the formulation of Palms language using the until operators.

Let $\chi \in \mathcal{L}_{DAG}$. Let $Cl(\chi)$ be the smallest set of formulas containing all subformulas of χ, the constants *root* and *leaf*, and which is closed under taking single negations and under the rule: $Until_\pi(\phi,\psi) \in Cl(\chi) \Rightarrow \psi \wedge Until_\pi(\phi,\psi) \in Cl(\chi)$.

We associate a formula $\nabla(\chi)$ with χ as follows. We create for each $\phi \in Cl(\chi)$, a new propositional variable q_ϕ. Now $\nabla(\chi)$ "axiomatizes" these new variables as follows:

$$
\begin{array}{ll}
q_p & \leftrightarrow p \\
q_{\neg\phi} & \leftrightarrow \neg q_\phi \\
q_{\phi\wedge\psi} & \leftrightarrow q_\phi \wedge q_\psi \\
q_{Until_\pi(\phi,\psi)} & \leftrightarrow \langle\pi\rangle q_\phi \vee \langle\pi\rangle q_{(\psi\wedge Until_\pi(\phi,\psi))} \qquad \text{for } \pi \in \{\downarrow,\uparrow\}.
\end{array}
$$

Lemma 1. *(i) For every model $\mathfrak{M}$ which validates $\nabla(\chi)$, for every node n and for every subformula $\phi \in Cl(\chi)$, $\mathfrak{M}, n \models q_\phi$ iff $\mathfrak{M}, n \models \phi$.*
(ii) Thus for all $\gamma, \chi \in \mathcal{L}_{DAG}$, it holds that $\gamma \models \chi \iff \nabla(\gamma \wedge \chi), q_\gamma \models q_\chi$.

The proof is by induction on the structure of the formula, and for the left to right direction of the until case by induction on the depth of direction of π. Note that it is crucial that the models are finite and acyclic. Also note that this reduction does not work (at least not directly) for formulas of the form $\langle(\uparrow;\downarrow)^*\rangle\phi$ or even $\langle(\downarrow^*)^*\rangle\phi$.

Finally note that the right hand side of the statement in Lemma 1.(ii) contains only diamonds of the form $\langle\uparrow\rangle$ and $\langle\downarrow\rangle$. As the reduction is linear we can thus decide the consequence problem for this restricted language.

We will now give an EXPTIME algorithm that on input formulas γ, χ decides whether there exists a model $\mathfrak{M}$ in which γ is true everywhere and χ is true at the root. To this the consequence problem reduces because $\gamma \not\models \chi$ iff there exists a model in which $\gamma \wedge (p \leftrightarrow \neg\chi \vee \langle\downarrow\rangle p)$ is true everywhere and p is true at the root. Here p is a new propositional variable whose intended meaning is $\langle\downarrow^*\rangle\neg\chi$.

Preliminaries. The next notion is well known. Hintikka sets are used to label nodes of models with a set of formulas which are supposed to be true at that node. The first condition ensures that γ and $\top$ are true in every node. The other two ensure the correct behaviour of the Booleans.

Definition 1 (Hintikka Set). *Let $A \subseteq Cl(\{\gamma,\chi\})$. We call A a Hintikka Set if A satisfies the following conditions:*

1. $\gamma \in A$ *and* $\top \in A$.
2. *If* $\phi \in Cl(\{\gamma, \chi\})$ *then* $\phi \in A$ *iff* $\neg\phi \notin A$.
3. *If* $\phi \wedge \psi \in Cl(\{\gamma, \chi\})$ *then* $\phi \wedge \psi \in A$ *iff* $\phi \in A$ *and* $\psi \in A$.

Let $HS(\gamma, \chi)$ denote the set of all Hintikka Sets which are a subset of $Cl(\gamma, \chi)$. Note that $|HS(\gamma, \chi)| \leq 2^{|Cl(\gamma,\chi)|}$.

For H a set of Hintikka sets, let $l : H \longrightarrow \{0, 1, \ldots, |H|\}$ be a function assigning to each $A \in H$ a level. We call a structure (H, l) an ordered set of Hintikka sets. For notational convenience we introduce a binary relation on Hintikka sets specifying that it is not directly inconsistent that the two sets stand in the parent relation in the tree: For A, B Hintikka sets, A child B holds if

1. $l(A) > l(B)$
2. for all $\langle\downarrow\rangle\psi \in Cl(\phi)$, if $\psi \in B$, then $\langle\downarrow\rangle\psi \in A$;
3. for all $\langle\uparrow\rangle\psi \in Cl(\phi)$, if $\psi \in A$, then $\langle\uparrow\rangle\psi \in B$;
4. if $id \notin B$ then also for all $\langle\uparrow\rangle\psi \in Cl(\phi)$, $\langle\uparrow\rangle\psi \in B$ implies $\psi \in A$.

The definition of saturation is the crucial one in any mosaic style proof. Informally it states that a set of Hintikka Sets is large enough to build a model from. In the temporal logic literature, the diamond formulas in Hintikka set are called *unfulfilled eventualities*.

Definition 2 (Saturation). *Let* (H, l) *be an ordered set of Hintikka sets. We call* (H, l) down- saturated *if for all* $A \in H$, $\langle\downarrow\rangle\phi \in A$ *only if there exists a* $B \in H$ *such that* $\phi \in B$ *and* A child B.

We call (H, l) up- saturated *if for all* $A \in H$ *containing id,* $\langle\uparrow\rangle\phi \in A$ *only if there exists a* $B \in H$ *such that* $\phi \in B$ *and* B child A.

We call (H, l) saturated *if it is both up and down saturated.*

The next definition specifies when an ordered saturated set of Hintikka sets can be turned into an NDAG.

Definition 3. *We call an ordered saturated set of Hintikka sets* (H, l) *rooted and nominalized if*

1. *There is exactly one* $A \in H$ *with root* $\in H$, *and for every nominal* $i \in CL(\{\gamma, \chi\})$ *there exists exactly one* $A \in H$ *such that* $i \in A$.
2. *(everyone has a predecessor) For every* B *in* H *there is a path* $C_0, \ldots, C_k$ *of Hintikka sets in* H *with* $B = C_k$ *such that*
   (a) *root* $\in C_0$
   (b) C_j child C_{j+1}.

We can now make the connection between satisfiability and the existence of certain sets of Hintikka sets.

Lemma 2. *The following are equivalent:*

1. *There exists a model over a finite NDAG in which* γ *is true everywhere and* χ *is true at the root;*

2. *There exists a rooted nominalized saturated ordered set of Hintikka Sets* (H, l)*, with* $H \subseteq HS(\gamma, \chi)$ *and there is an* $A \in H$ *with* $\{root, \chi\} \subseteq A$.

Proof. First assume $\mathfrak{M}$ is a model over a finite NDAG in which γ is true everywhere and χ is true at the root. For each node t define $A_t = \{\psi \in Cl(\gamma, \chi) \mid \mathfrak{M}, t \models \psi\}$. Obviously each A_t is a Hintikka set and there is an A with $\{root, \chi\} \subseteq A$. Let H be the set of all such A_t. Inductively define the level function on H. First define which Hintikka Sets are of level 0: $l(A) = 0$ if $leaf \in A$. Next, suppose the j-th level is defined. First define: $S_j = \{A \in H \mid l(A) \leq j\}$. Next, for $A \in H \backslash S_j$, $l(A) = i + 1$ if $\mathfrak{M}, root \models \langle\downarrow^*\rangle(\hat{A} \wedge [\downarrow][\downarrow^*] \bigvee_{B \in S_j} \hat{B})$. It is not hard to show that (H, l) is ordered, saturated, rooted, nominalized and A_{root} contains *root* and χ.

Now assume (H, l) is a rooted nominalized saturated ordered set of Hintikka Sets and there is an $A_{root} \in H$ with $\{root, \chi\} \subseteq A_{root}$. For each nominal i, let A_i be the Hintikka Set containing i. Let $\mathcal{T} = (T, R_\downarrow)$ be a tree with root t_0 of depth $l(A_{root})$ and branching width $|H|$. The function $\mathsf{depth}(\cdot)$ measures the depth of nodes in the tree (with $\mathsf{depth}(t_0) = 0$). Let $h : T \longrightarrow H$ be a partial function satisfying

root $h(t_0) = A_{root}$.
max if $tR_\downarrow t'$ and h is defined on t and t', then $h(t)$ child $h(t')$.
min if $h(t)$ child B then either there exists a $t' \in T$ such that $tR_\downarrow t'$ and $h(t') = B$, or B contains a nominal.
nom for each nominal $i \in Cl(\{\gamma, \chi\})$ there exists exactly one t such that $h(t) = A_i$ and for any t', $h(t')$ child A_i implies that $\mathsf{depth}(t) > \mathsf{depth}(t')$.

It is straightforward to show that such h can be defined (by a step-by-step construction for instance). Now we turn $\mathcal{T}$ into an NDAG. First let $\mathcal{T}'$ be the largest subtree of $\mathcal{T}$ on which h is total. Second, let $\mathcal{T}''$ be $\mathcal{T}'$ with the following arrows added:

if $h(t)$ child $h(t')$ and $\mathsf{depth}(t) < \mathsf{depth}(t')$ and $id \in h(t')$, then add $tR_\downarrow t'$.

We claim that $(\mathcal{T}'', NOM)$ with $NOM = \{t \mid id \in h(t)\}$ is a rooted NDAG satisfying

up-min if B child $h(t')$ and $id \in h(t')$, then there exists a t such that $tR_\downarrow t'$ and $h(t) = B$.
nom-min if $h(t)$ child B and B contains a nominal, then there exists a $t' \in T$ such that $tR_\downarrow t'$ and $h(t') = B$.

By construction $(\mathcal{T}'', NOM)$ is a rooted NDAG. To show **up-min**, assume that B child $h(t')$ holds. Then $l(B) > l(h(t'))$. By Definition 3.2 there exists a path A_{root} child $\ldots$ child B, say of length k. Whence, by **min**, there exists a node t with $h(t) = B$ and $\mathsf{depth}(t) = k$. By **max**, the depth of t' must be strictly larger than k because $l(B) > l(h(t'))$. Thus an arrow from t to t' has been added. The proof for **nom-min** is similar.

```
begin
   L    :=  {A ∈ Choice(γ, χ) | leaf ∈ A};
   Pool :=  Choice(γ, χ)\ L;
   S    :=  L;
   k    :=  0;
   l    :=  {(A, k) | A ∈ L};
   do  L ≠ ∅  →
       L    := {A ∈ Pool | (S ∪ {A}, l ∪ (A, k + 1))
               is a down--saturated ordered set of
               Hintikka Sets };
       Pool := Pool \ L;
       S    := S ∪ L;
       k    := k+1;
       l    := l ∪ {(A, k) | A ∈ L}
   od
end
```

Fig. 4. The algorithm *elimination of* $Choice(\gamma, \chi)$

Let $V(p) = \{t \mid p \in h(t)\}$, for p a nominal or a propositional variable. Then $\mathfrak{M} = (\mathcal{T}'', NOM, V)$ is a model, because every nominal is true at exactly one node in NOM.

We claim that $\mathfrak{M} \models \gamma$ and $\mathfrak{M}, t_0 \models \chi$. By assumption $\chi \in h(t_0)$ and γ is in every Hintikka set, thus it is sufficient to prove the truth lemma

for all $\psi \in CL(\gamma, \chi)$, for all nodes t, $\mathfrak{M}, t \models \psi$ if and only if $\psi \in h(t)$.

The base case is by definition of V. The case for *id* is by definition of NOM. The boolean cases are by the conditions on Hintikka sets. The left to right direction for both modalities follows from max. The other direction for $\langle\downarrow\rangle\psi$ and $\langle\uparrow\rangle\psi$ follows from min, nom-min and up-min and saturation.

The algorithm. We now describe the algorithm for finding a saturated, ordered, rooted and nominalized set of Hintikka Sets. It consists of five different stages. Let γ, χ be the formulas for which we decide the existence of a model in which γ holds everywhere and χ at the root.

(1) Create Hintikka Sets The algorithm creates $HS(\gamma, \chi)$. $HS(\gamma, \chi)$ contains $2^{O(|\gamma \wedge \chi|)}$ sets of size $O(|\gamma \wedge \chi|^2)$.

(2) Choose Named Elements Choose a set $NAMED \subseteq HS(\gamma, \chi)$ having a Hintikka set containing χ and the root symbol *root* and exactly one Hintikka set containing the nominal i for each $i \in CL(\gamma, \chi)$.
There are at most $|HS(\gamma, \chi)| \cdot \ldots \cdot |HS(\gamma, \chi)|$ (as many as there are nominals in $\gamma \wedge \chi$ plus one) many choices, that is at most $2^{O(|\gamma \wedge \chi|^2)}$. Let $Choice(\gamma, \chi)$ be $NAMED \cup HS(\gamma, \chi) \setminus \{A \in HS(\gamma, \chi) \mid A$ contains a nominal or *root*$\}$.

(3) Create Down–Saturated Ordered Set Run the algorithm from Figure 4.

Lemma 3. *1. Elimination of Choice(γ, χ) terminates after at most $|HS|$ rounds of the do loop.*
2. The statement "$\langle \mathtt{S}, \mathtt{l} \rangle$ is a down–saturated ordered set of Hintikka sets" holds after the do loop.

Proof. (1) The bound function of the do loop is the size of `Pool` which is being reduced in every round, or the loop terminates because `L`$= \emptyset$. The initial size of `Pool` is bounded by $|HS|$.
(2) Because the statement "*Choice*$(\gamma, \chi) =$ `Pool` $\uplus$ `S` and $\langle \mathtt{S}, \mathtt{l} \rangle$ is a down–saturated ordered set of Hintikka sets" holds before the do loop and is an invariant of the do loop.

(4) Make (H, l) **Rooted** Let (H, l) be the output $\langle \mathtt{S}, \mathtt{l} \rangle$ of the previous stage. Delete all elements from H for which condition 2 in Definition 3 does not hold.

(5) Test Let (H, l) be the output of the previous stage. Check whether (H, l) contains a Hintikka set A with *root* $\in A$ and $\chi \in A$. Check whether (H, l) is up–saturated. And check whether (H, l) contains for each nominal $i \in Cl(\gamma, \chi)$ a Hintikka set containing i.

Lemma 4. *If all these checks succeed, (H, l) is an up and down–saturated rooted and nominalized ordered set of Hintikka sets.*

The algorithm succeeds iff there is a choice in stage 2 for which the checks in stage 5 succeed.

The algorithm is correct by Lemma 2. Let us check that the algorithm runs in time exponential in the length of the input. The first stage is clear. For the second stage it has to perform the rest of the algorithm for at most $2^{O(|\gamma,\chi|^2)}$ many choices. So it is sufficient to show that stages 3–5 can be performed in exponential time. The algorithm of stage 3 terminates after at most $|HS(\gamma, \chi)| \leq 2^{O(|\gamma,\chi|)}$ rounds of the do loop. As in [21], the tests inside the do loop take time polynomially bounded by $|HS(\gamma, \chi)|$. Thus stage 3 takes time exponentially bounded by $|\gamma, \chi|$. It is clear that stages 4 and 5 can all be performed in time polynomially bounded by $|HS(\gamma, \chi)|$. Thus the algorithm is in EXPTIME.

6 Conclusions

We have given an exponential time decision algorithm for a modal language with nominals interpreted on finite rooted nominalized DAGs. This is –as far as we know– the first result which yields a decision algorithm for XPath query containment in the presence of XML's ID/IDREF referencing mechanism.

Obviously the algorithm is not that easy to implement, so that's a next research question. Another question is whether we can get the same exponential upper bound if we interpret the language with both sibling axis on ordered NDAG's.

The function h defined in the proof of Lemma 2 is almost a surjective bounded morphism (the zag direction might break for $R_{\uparrow}$ from Hintikka sets not containing id). This leads us to conjecture that a slight improvement of that Lemma can be used to prove a completeness theorem for this logic. At present no axiomatization is known.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the web*. Morgan Kaufman, 2000. 154

[2] N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation*, 13:1–18, 2003. 151

[3] M. Benedikt, W. Fan, and G. Kuper. Structural properties of XPath fragments, 2003. In Proc. ICDT'03, 2003. To appear. 154

[4] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000. 151, 156

[5] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001. 151

[6] P. Blackburn, B. Gaiffe, and M. Marx. Variable free reasoning on finite trees. In *Proceedings of Mathematics of Language (MOL–8), Bloomington*, 2003. 151, 153

[7] J. Clark. XML Path Language (XPath). `http://www.w3.org/TR/xpath`. 154

[8] World-Wide Web Consortium. Xml path language (xpath): Version 2.0. `http://www.w3.org/TR/xpath20/`. 153

[9] Alin Deutsch and Val Tannen. Containment of regular path expressions under integrity constraints. In *Knowledge Representation Meets Databases*, 2001. 155

[10] J. Doner. Tree acceptors and some of their applications. *J. Comput. Syst. Sci.*, 4:405–451, 1970. 158

[11] E. Emerson and J. Halpern. "Sometimes and Not Never"; revisited: on branching versus linear time temporal logic. *Journal of the ACM (JACM)*, 33(1):151–178, 1986. 151

[12] M. Fisher and R. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979. 151

[13] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000. 152

[14] M. Kracht. Syntactic codes and grammar refinement. *Journal of Logic, Language and Information*, 4:41–60, 1995. 152

[15] M. Kracht. Inessential features. In Christian Retore, editor, *Logical Aspects of Computational Linguistics*, number 1328 in LNAI, pages 43–62. Springer, 1997. 152

[16] M. Marx. XPath and modal logic of finite trees. In *Proceedings of M4M 2003, Nancy*, 2003. 155

[17] G. Miklau and D. Suciu. Containment and equivalence for an XPath fragment. In *Proc. PODS'02*, pages 65–76, 2002. 151, 155

[18] F. Neven and T. Schwentick. XPath containment in the presence of disjunction, DTDs, and variables. In *ICDT 2003*, 2003. 151, 155

[19] A. Palm. *Transforming tree constraints into formal grammars*. PhD thesis, Universität Passau, 1997. 153

[20] A. Palm. Propositional tense logic for trees. In *Sixth Meeting on Mathematics of Language*. University of Central Florida, Orlando, Florida, 1999. 150, 153

[21] V. Pratt. Models of program logics. In *Proceedings of the 20th IEEE symposium on Foundations of Computer Science*, pages 115–122, 1979. 151, 162

[22] M. Rabin. Decidability of second order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969. 158

[23] K. Reinhardt. The complexity of translating logic to finite automata. In E. Grädel et al., editor, *Automata, Logics, and Infinite Games*, volume 2500 of *LNCS*, pages 231–238. Springer, 2002. 153

[24] B-H. Schlingloff. Expressive completeness of temporal logic of trees. *Journal of Applied Non–Classical Logics*, 2(2):157–180, 1992. 153

[25] P. Wood. On the equivalence of XML patterns. In *Proc. 1st Int. Conf. on Computational Logic*, volume 1861 of *LNCS*, pages 1152–1166, 2000. 155