Tableaux for Hybrid XPath with Data

Carlos Areces[*,†], Raul Fervari[*,†], and Nahuel Seiler[*]

[*]FaMAF, Universidad Nacional de Córdoba, Argentina
[†]CONICET, Argentina

Abstract. We provide a sound, complete and terminating tableau procedure to check satisfiability of downward XPath$_=$ formulas enriched with nominals and satisfaction operators. The calculus is inspired by ideas introduced to ensure termination of tableau calculi for certain Hybrid Logics. We prove that even though we increased the expressive power of XPath by introducing hybrid operators, the satisfiability problem for the obtained logic is still PSPACE-complete.

Keywords: XPath, hybrid logic, tableaux, termination, complexity

1 Introduction

In many applications, dealing with actual data is an important challenge. For instance, applications which manage large volumes of web or medical data require, in many cases, more complex models than those that can be encoded in classical relational databases. These models are often defined and studied in the context of *semi-structured data* [9]. A semi-structured data model is based on an organization of data in labeled trees or graphs, and on query languages for accessing and updating these structures. These representations can contain labels coming from a finite alphabet (capturing the *structural* information), or from an infinite alphabet (capturing also the actual *data* in the database). Most query languages focus only on how to access the structural information, in this article we focus on languages that also handle data.

XML (eXtensible Markup Language) is the most successful data model that captures both structural information and data. An XML document is a hierarchical structure represented by an unranked finite ordered tree, where nodes have labels (either letters from a finite alphabet, or data values from an infinite alphabet). XPath is, arguably, the most widely used XML query language, with application in specification and update languages. XPath is, fundamentally, a general purpose language for addressing, searching, and matching pieces of an XML document. It is an open standard and constitutes a World Wide Web Consortium (W3C) Recommendation [11]. Core-XPath [16] is the fragment of XPath 1.0 containing the navigational behavior of XPath. It can express properties of the underlying tree structure of the XML document, such as the label (tag name) of a node, but it cannot express conditions on the actual data contained in the attributes. In other words, it is essentially a *classical modal logic* [5,6].

Email: {areces,fervari,ngs0108}@famaf.unc.edu.ar

However, without the ability to relate nodes based on the actual data values of the attributes, the expressive power of Core-XPath is inappropriate for many applications. In fact, it is not possible to define the most important construct in a database query language: the *join*. The extension of Core-XPath with (in)equality tests between attributes of elements in an XML document is named Core-Data-XPath in [7]. Here, we will call this logic XPath$_=$. Models of XPath$_=$ are data trees which can be seen as abstractions of XML documents. A data tree is a tree whose nodes contain a label from a finite alphabet and a data value from an infinite domain (see Figure 1 for an example). In this article we will consider the case where models can be arbitrary *graphs* and not just finite trees. XPath$_=$ allows formulas of the form $\langle \alpha = \beta \rangle$ and $\langle \alpha \neq \beta \rangle$, where α, β are path expressions that navigate the model using axes: descendant, child, ancestor, next-sibling, etc. and can make tests in intermediate nodes. The formula $\langle \alpha = \beta \rangle$ (respectively $\langle \alpha \neq \beta \rangle$) is true at a node x of a data tree if there are nodes y, z that can be reached by paths denoted by α, β, and such that the data value of y is equal (respectively different) to the data value of z. For instance, in Figure 1 the expression *"there is a one-step descendant and a two-steps descendant with the same data value"* is satisfied at x, given the presence of u and z. The expression *"there are two children with distinct data value"* is also true at x, because y and z have different data.

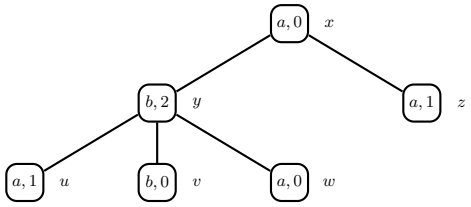

Fig. 1. An example of a data tree. Letters are labels, numbers are data.

Notice that XPath$_=$ allows to compare data values at the end of a path, by equality or inequality. However, it does not grant access to the concrete data value of nodes (in the example, 0, 1 or 2). As a result, it is possible to work with an abstraction of data trees: instead of having concrete data values in each node, we have an equivalence relation between nodes. In the data tree from Figure 1, the relation consists of three equivalence classes: $\{x, v, w\}$, $\{u, z\}$ and $\{y\}$.

Recent articles investigate XPath$_=$ from a modal perspective. For example, satisfiability and evaluation are discussed in [13,14], while model theory and expressivity are studied in [15,2]. A Gentzen-style sequent calculus is given in [4] for a very restricted fragment of XPath$_=$. An extension of the equational axiomatic system from [10] is introduced in [1], allowing downward navigation and equality/inequality tests. [3] provides an axiomatization for the previous logic, extended with upward navigation, nominals and satisfaction operators.

Contributions. In this article we introduce a sound, complete and terminating tableau calculus for $\text{XPath}_=$ with downward navigation, where node expressions are extended with nominals (special labels that are true in only one node), and path expressions are extended with the satisfaction operator (allowing the navigation to some particular named node). We call this logic $\text{HXPath}_=(\downarrow)$.

We will follow ideas introduced in [8] to design terminating tableau calculi for hybrid logics. The main intuition is that nominals and satisfaction operators can be used in tableaux to keep track of the evaluation of a formula during an attempt to build a model. This way, tableaux rules and the completeness proof are more intuitive, obtaining a simple proof theory for XPath with data.

Organization. In Section 2 we define the syntax and semantics of $\text{HXPath}_=(\downarrow)$, and give examples to show its expressive power. Section 3 introduces a tableau calculus for $\text{HXPath}_=(\downarrow)$, its completeness is proved in Section 4, and termination in Section 5. We also show that the satisfiability problem for $\text{HXPath}_=(\downarrow)$ is PSPACE-complete. We include some final remarks and future work in Section 6.

2 Basic Definitions

We start by defining the structures that will be used to evaluate formulas in the language. We assume basic knowledge of classical modal logic [5].

Definition 1. *Let* PROP *(the set of propositional symbols) be an infinite countable set, let* NOM *(the set of nominals) be an infinite countable well-ordered*[1] *set such that* $\mathsf{NOM} \cap \mathsf{PROP} = \emptyset$*, and let* $\mathsf{ATOM} = \mathsf{PROP} \cup \mathsf{NOM}$ *be the set of atomic formulas (or atoms for short).*

An abstract hybrid data model *is a tuple* $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom \rangle$*, where* M *is a non-empty set of elements,* $\sim \subseteq M^2$ *is an equivalence relation between elements of* M*,* $\rightarrow \subseteq M^2$ *is an accessibility relation,* $label : M \rightarrow 2^{\mathsf{PROP}}$ *is a labeling function and* $nom : \mathsf{NOM} \rightarrow M$ *is a function that assigns nominals to certain elements.*

Concrete data models[2] are most commonly used in application, where we encounter data from an infinite alphabet (e.g., alphabetic strings) associated to the nodes in a semi-structured database. It is easy to see that each concrete data model has an associated abstract data model where data is replaced by an equivalence relation that links all nodes with the same data. Vice-versa, each abstract data model can be "concretized" by assigning to each node its equivalence data class as data.

Definition 2. *The sets* PExp *of path expressions and* NExp *of node expressions of the language HXPath*$_=(\downarrow)$ *are defined by mutual recursion as follows:*

$$\begin{aligned} \mathsf{PExp} &::= \downarrow \mid i \mid [\varphi] \mid \alpha\beta \mid \alpha \cup \beta \\ \mathsf{NExp} &::= p \mid i \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle \alpha = \beta \rangle \mid \langle \alpha \neq \beta \rangle, \end{aligned}$$

[1] The well-ordered condition will be used to prove termination.
[2] For a detailed introduction, see [3].

where $p \in \mathsf{PROP}$, $i \in \mathsf{NOM}$, $\alpha, \beta \in \mathsf{PExp}$ *and* $\varphi, \psi \in \mathsf{NExp}$. *The set* Exp *of expressions of HXPath*$_=(\downarrow)$ *is defined as* $\mathsf{NExp} \cup \mathsf{PExp}$.

$\mathsf{NOM}(\varepsilon)$ *denotes the set of nominals appearing in an expression* $\varepsilon \in \mathsf{Exp}$. *If* E *is a set of expressions then* $\mathsf{NOM}(E) = \bigcup_{\varepsilon \in E} \mathsf{NOM}(\varepsilon)$.

In the rest of the article we will use the symbols i, j, k, n, m for nominals; p, q, r, for propositional symbols; α, β, γ, δ for path expressions; φ, ψ for node expressions; and ε for an arbitrary expression.

In what follows we will always use $*$ for $=$ and $\neq$. Missing Boolean operators are defined as usual. We define the following operators as abbreviations. Let α be a path expression, φ a node expression, $i \in \mathsf{NOM}$, and $p \in \mathsf{PROP}$:

$$\top \equiv p \vee \neg p \quad \bot \equiv \neg\top \quad \langle\alpha\rangle\varphi \equiv \langle\alpha[\varphi] = \alpha[\varphi]\rangle \quad [\alpha]\varphi \equiv \neg\langle\alpha\rangle\neg\varphi \quad i\varphi \equiv \langle i\rangle\varphi.$$

Formulas of the form $i\varphi$ for $i \in \mathsf{NOM}$ and φ a node expression are called *at-formulas* or *prefixed-formulas* (intuitively, they express that φ holds *at* the state named by i); i is called the prefix of $i\varphi$. At-formulas will play a crucial role in the tableau calculus introduced in the next section.

Notice that we use nominals in satisfaction operators appearing in path expressions (e.g., $\downarrow i\downarrow$) and as atoms (e.g., $i \wedge p$) and prefixes (in $i\varphi$) in node expressions; the intended meaning will always be clear by context.

Also, following the standard notation in XPath logics and in modal logics, the [ ] operation is overloaded: for φ a node expression and α a path expression, both $[\alpha]\varphi$ and $[\varphi]\alpha$ are well-formed expressions; the former is a node expression where $[\alpha]$ is a box modality, the latter is a path expression where $[\varphi]$ is a test.

Definition 3. *Let* $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom\rangle$ *be an abstract data model, and* $x, y \in M$. *We define the semantics of HXPath*$_=(\downarrow)$ *as follows:*

$$\begin{aligned}
\mathcal{M}, x, y &\models \downarrow \text{ iff } x \rightarrow y\\
\mathcal{M}, x, y &\models i \text{ iff } nom(i) = y\\
\mathcal{M}, x, y &\models [\varphi] \text{ iff } x = y \text{ and } \mathcal{M}, x \models \varphi\\
\mathcal{M}, x, y &\models \alpha\beta \text{ iff there is some } z \in M \text{ s.t. } \mathcal{M}, x, z \models \alpha \text{ and } \mathcal{M}, z, y \models \beta\\
\mathcal{M}, x, y &\models \alpha \cup \beta \text{ iff } \mathcal{M}, x, y \models \alpha \text{ or } \mathcal{M}, x, y \models \beta\\
\mathcal{M}, x &\models p \text{ iff } p \in label(x)\\
\mathcal{M}, x &\models i \text{ iff } nom(i) = x\\
\mathcal{M}, x &\models \neg\varphi \text{ iff } \mathcal{M}, x \not\models \varphi\\
\mathcal{M}, x &\models \varphi \wedge \psi \text{ iff } \mathcal{M}, x \models \varphi \text{ and } \mathcal{M}, x \models \psi\\
\mathcal{M}, x &\models \langle\alpha = \beta\rangle \text{ iff there are } y, z \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \text{ and } y \sim z\\
\mathcal{M}, x &\models \langle\alpha \neq \beta\rangle \text{ iff there are } y, z \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \text{ and } y \not\sim z.
\end{aligned}$$

As a corollary of the definition above, the abbreviations $\langle\alpha\rangle\varphi$, $[\alpha]\varphi$ and $i\varphi$ have their classical meaning.

$$\begin{aligned}
\mathcal{M}, x &\models \langle\alpha\rangle\varphi \text{ iff there is } y \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha \text{ and } \mathcal{M}, y \models \varphi\\
\mathcal{M}, x &\models [\alpha]\varphi \text{ iff for all } y \in M\ \mathcal{M}, x, y \models \alpha \text{ implies } \mathcal{M}, y \models \varphi\\
\mathcal{M}, x &\models i\varphi \text{ iff } \mathcal{M}, nom(i) \models \varphi
\end{aligned}$$

The addition of the hybrid operators to XPath increases its expressive power. The following examples should serve as illustration.

Example 1. We list below some HXPath$_=(\downarrow)$ expressions together with their intuitive meaning:

$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.
$i\alpha$	There exists an α path between the node named i and some other node.
αi	There exists an α path between the current node and some other node, evaluation continues at point named i.
$\langle i = i\rangle$	It is always valid.
$\langle [i] = [i]\rangle$	The current node is named i.
$\langle i = j\rangle$	The node named i has the same data as the node named j.
$\langle \alpha = i\beta\rangle$	A node accessible from the current node by an α path has the same data than a node accessible from the point named i by a β path.

It is worth highlighting the difference between $\langle \alpha \neq \beta\rangle$ and $\neg\langle \alpha = \beta\rangle$. The first expression establishes that there is an α path from the evaluation point to some point y, and there is a β path from the evaluation point to some point z, and $y \not\sim z$ (i.e., they have different data value). The second expression says that either at least one of the the α or β paths fail to exist, or they exist but they fail to have the same data value. So while in the first expression paths are necessarily realizable to make the formula true, this is not the case in the second expression. However, these formulas are equivalent if $\alpha = i$ and $\beta = j$, with $i, j \in \mathsf{NOM}$.

The next proposition collects some semantic properties that will be useful in the completeness proof of Section 4. The proof uses only the definition of $\models$.

Proposition 1. *Let $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom\rangle$ be an abstract data model, $x, y \in M$, α, β arbitrary path expressions, and $i, j \in \mathsf{NOM}$. Then*

1. $\mathcal{M}, x \models \langle i\alpha * j\beta\rangle$ *iff* $\mathcal{M}, y \models \langle i\alpha * j\beta\rangle$,
2. $\mathcal{M}, x \models i$ *and* $\mathcal{M}, x \models \langle \alpha * \beta\rangle$ *[resp.* $\neg\langle \alpha * \beta\rangle$*] then* $\mathcal{M}, x \models \langle i\alpha * \beta\rangle$ *[resp.* $\neg\langle i\alpha * \beta\rangle$*]*,
3. $nom(i) \sim nom(j)$ *iff* $\mathcal{M}, x \models \langle i = j\rangle$,
4. $\mathcal{M}, x \models \langle ij\alpha * \beta\rangle$ *iff* $\mathcal{M}, x \models \langle j\alpha * \beta\rangle$.

3 Tableau Calculus

We present a tableau calculus for HXPath$_=(\downarrow)$. We assume basic knowledge of tableau calculi for modal logics [12].

In addition to HXPath$_=(\downarrow)$-formulas, the tableau rules contain *accessibility formulas* of the form $n \rightarrow m$, where $n, m \in \mathsf{NOM}$. The intended interpretation of $n \rightarrow m$ is that the node denoted by m is accessible from the node denoted by n by the accessibility relation $\rightarrow$. In the following we will use the term *formula* to denote either a formula of HXPath$_=(\downarrow)$, or an accessibility formula.

A tableau in this calculus is a well-founded, finitely branching tree in which each node is labeled by a formula, and the edges represent applications of tableau rules, in the usual way. To check satisfiability of a node expression φ, we initialize the tableau with $i\varphi$, for $i \notin \mathsf{NOM}(\varphi)$. To check satisfiability of a path expression

α, we initialize the tableaux with $i\langle\alpha\rangle\top$, for $i \notin \mathsf{NOM}(\alpha)$. Figure 2 introduces the rules of the calculus.

A branch Θ of a tableau contains a *clash* if one of the following conditions holds:
1. $\{na, n\neg a\} \subseteq \Theta$, with $a \in \mathsf{ATOM}$,
2. $\langle n \neq n\rangle \in \Theta$,
3. $\neg\langle n = n\rangle \in \Theta$,
4. $\{\langle n = m\rangle, \langle n \neq m\rangle\} \subseteq \Theta$,
5. $\{\langle n = m\rangle, \neg\langle n = m\rangle\} \subseteq \Theta$,

for some $n,\ m \in \mathsf{NOM}$. Conditions $1-5$ are called *clash conditions*. The rules $(\neg)$, $(\downarrow)$, $(\downarrow_r)$ $(@)$, $(@_r)$, $(\neg@)$ and $(\neg@_r)$ are called *nominal generating rules*. We impose two general constraints on the construction of tableaux:
- C1: A nominal generating rule is never applied twice to the same premise on the same branch.
- C2: A formula is never added to a tableau branch where it already occurs.

A *saturated tableau* is a tableau in which no more rules can be applied that satisfy the constraints. A *saturated branch* is a branch of a saturated tableau. For ε an expression, let *Tableau*(ε) be a saturated tableau for ε. We say that a tableau branch is *closed* if it contains a clash, otherwise it is called *open*. A *closed tableau* is one in which all branches are closed, and an *open tableau* is one in which at least one branch is open. It is easy to show the calculus is sound:[3]

Theorem 1. *If ε is satisfiable then any Tableau(ε) has an open branch.*

4 Completeness

In this section we will prove that the tableau calculus we introduced is *complete*, i.e., if a formula φ appears in an open and saturated branch, then φ is satisfiable.

Definition 4. *Let Θ be a tableau branch. Define the relation $\equiv_\Theta \subseteq \mathsf{NOM}(\Theta)^2$ as $n \equiv_\Theta m$ iff $nm \in \Theta$.*

Lemma 1. *$\equiv_\Theta$ is an equivalence relation.*

Definition 5. *Let Θ be a tableau branch, and let $n \in \mathsf{NOM}(\Theta)$. The* nominal urfather *of n on Θ (denoted $u_\Theta(n)$) is the smallest m such that $m \equiv_\Theta n$. m is called a* nominal urfather *on Θ if $m = u_\Theta(n)$ for some n.*

Lemma 2. *Let Θ be a saturated branch. If $n\varphi$ occurs on Θ then $u_\Theta(n)\varphi$ also occurs on Θ.*

Proof. Assume $n\varphi \in \Theta$. By definition of urfather, $u_\Theta(n) = m$ and $nm \in \Theta$. Since Θ is saturated we have closure under (*Copy*) and (*Ref*). Hence, $n\varphi$, nm, $mm \in \Theta$ and $m\varphi \in \Theta$.

[3] Some readers may call this notion "completeness", and the one introduced in the next section "soundness". However, we use the classical tableaux denomination.

Lemma 3. *Let Θ be a saturated branch, and let n and m be nominals occurring on Θ. $n \equiv_\Theta m$ if and only if n and m make the same nominals true on Θ.*

Lemma 4. *Let Θ be a saturated branch. If $n \equiv_\Theta m$ then $u_\Theta(n) = u_\Theta(m)$.*

Lemma 5. *Let Θ be a saturated branch. Then n is a nominal urfather on Θ if and only if $u_\Theta(n) = n$.*

Proof. The right to left direction is direct from the definition of a nominal urfather. For the other direction, if n is a nominal urfather then $u_\Theta(m) = n$ for some m, and $n \equiv_\Theta m$. By Lemma 4, we have $u_\Theta(n) = u_\Theta(m) = n$.

Definition 6. *Let Θ be an open saturated branch, we define the extracted model $\mathcal{M}^\Theta = \langle M^\Theta, \sim^\Theta, \rightarrow^\Theta, label^\Theta, nom^\Theta \rangle$, as*

$$
\begin{array}{lll}
M^\Theta & = \{n \mid n\varphi \in \Theta\} \cup \{n, n' \mid n \rightarrow n' \in \Theta\} \cup \{n, n' \mid \langle n * n' \rangle \in \Theta\} \\
\sim^\Theta & = \{(n, u_\Theta(n')) \mid \langle n = n' \rangle \in \Theta\} \\
\rightarrow^\Theta & = \{(u_\Theta(n), u_\Theta(n')) \mid n \rightarrow n' \in \Theta\} \\
label^\Theta(n) & = \{p \mid np \in \Theta\} \\
nom^\Theta(i) & = \begin{cases} n_0, & \text{if } i \notin \mathsf{NOM}(\Theta) \\ u_\Theta(i), & \text{if } i \in \mathsf{NOM}(\Theta), \end{cases}
\end{array}
$$

where n_0 is the first prefix introduced in Θ.

Proposition 2. *$\sim^\Theta$ is an equivalence relation.*

Lemma 6. *Let $i \in \mathsf{NOM}(\Theta)$ then $u_\Theta(i) \in M^\Theta$ and $\mathcal{M}^\Theta, u_\Theta(i) \models i$.*

Proof. By definition $u_\Theta(i) = m$ and $mi \in \Theta$. Hence, by definition of M^Θ, $m \in M^\Theta$. That it satisfies i follows from the definition of nom^Θ.

We need a notion of size both for path and node expressions, and a notion of size for any tableau formula:

$$
\begin{array}{lll@{\qquad}lll}
psize(i) & = & 1 \text{ for } i \in \mathsf{NOM} & psize(\alpha\beta) & = & psize(\alpha) + psize(\beta) \\
psize(\downarrow) & = & 2 & psize(\alpha \cup \beta) & = & psize(\alpha) + psize(\beta) + 1 \\
psize([\varphi]) & = & nsize(\varphi) + 1 & & & \\
nsize(a) & = & 1 \text{ for } a \in \mathsf{ATOM} & nsize(\varphi \wedge \psi) & = & nsize(\varphi) + nsize(\psi) + 1 \\
nsize(\neg\varphi) & = & nsize(\varphi) + 1 & nsize(\langle \alpha * \beta \rangle) & = & psize(\alpha) + psize(\beta) + 2 \\
size(n \rightarrow m) & = & 0 & size(\langle \alpha * \beta \rangle) & = & nsize(\langle \alpha * \beta \rangle) \\
size(i\varphi) & = & nsize(\varphi) + 3. & & &
\end{array}
$$

Notice that $size(\varphi)$ induces a well-founded order on the set of HXPath$_=(\downarrow)$-formulas by taking $\varphi < \psi$ if and only if $size(\varphi) < size(\psi)$. The particular notion of *size* introduced will let us prove Theorem 3.

Theorem 2. *Let Θ be an open and saturated branch and $n, m \in \mathsf{NOM}$. Then*

1. $n\varphi \in \Theta$ *implies* $\mathcal{M}^\Theta, u_\Theta(n) \models \varphi$.
2. *(a)* $\langle n\alpha * m\beta\rangle \in \Theta$ *implies* $\mathcal{M}^\Theta, k \models \langle n\alpha * m\beta\rangle$, *for any* $k \in M^\Theta$, *and*
   (b) $\neg\langle n\alpha * m\beta\rangle \in \Theta$ *implies* $\mathcal{M}^\Theta, k \models \neg\langle n\alpha * m\beta\rangle$, *for any* $k \in M^\Theta$.

Proof. We reason by structural induction (as we mentioned, *size* induces a well-founded order on the set of formulas). Let us consider the base cases. We show some of them, the rest can be proved in a similar way:

- Cases np and $n\neg p$, with $p \in \mathsf{PROP}$ are direct from definition of $label^\Theta$.
- $ni \in \Theta$, with $i \in \mathsf{NOM}$. Then $n \equiv_\Theta i$ and by Lemma 4, $u_\Theta(n) = u_\Theta(i)$. By definition of nom^Θ, $nom^\Theta(i) = u_\Theta(i)$, and $\mathcal{M}^\Theta, u_\Theta(n) \models i$.
- $n\neg i \in \Theta$. By $(\neg)$, we have $mi \in \Theta$, for some m, and by definition of $\mathcal{M}^\Theta$ we have $n, m \in M^\Theta$ (and their respective urfathers by Lemma 6). Reasoning as above, $nom^\Theta(i) = u_\Theta(i) = u_\Theta(m)$. By Lemma 2, $u_\Theta(m)i \in \Theta$, but because Θ is an open saturated branch such that $n\neg i,\ mi \in \Theta$, then $n \neq u_\Theta(m)$. Then $nom^\Theta(i) \neq n$, hence $\mathcal{M}^\Theta, n \models \neg i$.
- $\langle n = m\rangle \in \Theta$. By definition of $\mathcal{M}^\Theta$, we have $n, m \in M^\Theta$ and $n \sim^\Theta u_\Theta(m)$. By definition of urfather $nu_\Theta(n) \in \Theta$ and by $(Data)$, $\langle n = u_\Theta(n)\rangle \in \Theta$. By definition of $\sim^\Theta$, $n \sim^\Theta u_\Theta(u_\Theta(n))$, i.e., $n \sim^\Theta u_\Theta(n)$ by Lemma 5. Because, $\sim^\Theta$ is an equivalence relation, $u_\Theta(n) \sim^\Theta u_\Theta(m)$. By definition of nom^Θ, $nom^\Theta(n) \sim^\Theta nom^\Theta(m)$ and, hence, by Proposition 1 (item 3), $\mathcal{M}^\Theta, k \models \langle n = m\rangle$ for any $k \in M^\Theta$.

Now we proceed with the inductive cases:

- $n\langle\alpha * \beta\rangle \in \Theta$: by (Int), we have $\langle n\alpha * n\beta\rangle \in \Theta$. $size(n\langle\alpha * \beta\rangle) = nsize(\langle\alpha * \beta\rangle) + 3 = psize(\alpha) + psize(\beta) + 5$, and $size(\langle n\alpha * n\beta\rangle) = psize(n\alpha) + psize(n\beta) = psize(\alpha) + psize(\beta) + 4$, i.e., (Int) decrements the size of the formula. Then we can apply inductive hypothesis and get $\mathcal{M}^\Theta, x \models \langle n\alpha * n\beta\rangle$, for all $x \in M^\Theta$. In particular $\mathcal{M}^\Theta, u_\Theta(n) \models \langle n\alpha * n\beta\rangle$. Therefore (by Lemma 6 and $\models$), $\mathcal{M}^\Theta, u_\Theta(n) \models n\langle\alpha * \beta\rangle$.
- $\langle n\alpha * k\beta\rangle \in \Theta$: induction on α.
  - $\alpha = \downarrow\alpha'$: $\langle n\downarrow\alpha' * k\beta\rangle \in \Theta$, then by $(\downarrow)$ we have $n \to m$, $\langle m\alpha' * \beta\rangle \in \Theta$ (with m the smallest nominal that has not appeared in the tableau). By definition of $\mathcal{M}^\Theta$, we have $u_\Theta(n) \to^\Theta u_\Theta(m)$ $(\otimes_1)$ and by IH, $\mathcal{M}^\Theta, x \models \langle m\alpha' * k\beta\rangle$ $(\otimes_2)$, for all $x \in M^\Theta$. From $\otimes_2$, we have that in particular $\mathcal{M}^\Theta, u_\Theta(m) \models \langle m\alpha' * k\beta\rangle$, and because $u_\Theta(m)$ is the urfather of m, by Lemma 6, $\mathcal{M}^\Theta, u_\Theta(m) \models m$. Then by $\models$, $\mathcal{M}^\Theta, u_\Theta(m) \models \langle\alpha' * k\beta\rangle$ $(\otimes_3)$. From $\otimes_1$ and $\otimes_3$ we get $\mathcal{M}^\Theta, u_\Theta(n) \models \langle\downarrow\alpha' * k\beta\rangle$, iff $\mathcal{M}^\Theta, u_\Theta(n) \models \langle n\downarrow\alpha' * k\beta\rangle$, iff (by Proposition 1, item 1) $\mathcal{M}^\Theta, x \models \langle n\downarrow\alpha' * k\beta\rangle$, for all $x \in M^\Theta$.
- $\neg\langle n\alpha * k\beta\rangle \in \Theta$: induction on α.
  - $\alpha = \downarrow\alpha'$: $\neg\langle n\downarrow\alpha' * k\beta\rangle \in \Theta$, and suppose $n \to m \in \Theta$, then by $(\neg\downarrow)$ $\neg\langle m\alpha' * k\beta\rangle \in \Theta$. By definition of $\mathcal{M}^\Theta$, we have $u_\Theta(n) \to^\Theta u_\Theta(m)$ $(\otimes_1)$ and by IH, $\mathcal{M}^\Theta, x \models \neg\langle m\alpha' * k\beta\rangle$ $(\otimes_2)$, for all $x \in M^\Theta$. From $\otimes_2$, we have that in particular $\mathcal{M}^\Theta, u_\Theta(m) \models \neg\langle m\alpha' * k\beta\rangle$, and because $u_\Theta(m)$ is the urfather of m, by Lemma 6, $\mathcal{M}^\Theta, u_\Theta(m) \models m$. Then by $\models$, $\mathcal{M}^\Theta, u_\Theta(m) \models \neg\langle\alpha' * k\beta\rangle$ $(\otimes_3)$. From $\otimes_1$ and $\otimes_3$ we get $\mathcal{M}^\Theta, u_\Theta(n) \models \neg\langle\downarrow\alpha' * k\beta\rangle$, iff $\mathcal{M}^\Theta, u_\Theta(n) \models \neg\langle n\downarrow\alpha' * k\beta\rangle$, iff (by Proposition 1, item 1) $\mathcal{M}^\Theta, x \models \neg\langle n\downarrow\alpha' * k\beta\rangle$, for all $x \in M^\Theta$.

5 Termination

In this section we prove that any tableaux obtained by the application of the rules in Section 3 is finite. This proves that satisfiability for HXPath$_=(\downarrow)$ is decidable.

Definition 7. *Let φ, φ' be node expressions. Define $\varphi \prec \varphi'$ if and only if,*

1. *there are ψ, n, m such that $\varphi = n\psi$, $\varphi' = m\psi$, and $n < m$, or*
2. *there are no ψ, n, m such that $\varphi = n\psi$ and $\varphi' = m\psi$, and $size(\varphi) < size(\varphi')$.*

Proposition 3. *The relation $\prec$ from Definition 7 is a well-founded order.*

Proof. As we already mentioned *size* induces a well-founded order over the set of node expressions. The relation $\prec$ orders also at-formulas which are identical except for their prefix, and hence have the same value for *size*. The order used in this case is the order given by NOM. As a result, $\prec$ cannot have infinite descending chains.

Proposition 4. *In every tableaux rule in Section 3 except (Ref), (Sym), (Nom), (DSym), (DTrans) and (Data) the formulas in the consequent are strictly smaller, in terms of $\prec$, than some of the formulas in the antecedent.*

Theorem 3. *Any tableau in the calculus from Section 3 is finite.*

Proof. Suppose, for contradiction, that a tableaux T is infinite. As all rules in Section 3 are finitely branching, T should have an infinite branch Θ. Moreover, by Propositions 3 and 4, together with the general constraints C1 and C2 imposed on the construction of tableaux, there is a point in Θ in which the only rules applied further down the branch are (*DSym*), (*DTrans*), (*Ref*), (*Sym*), (*Nom*) and (*Data*). But these rules only introduce atomic formulas built over symbols that have already appeared in Θ, and hence, at one point, by constraint C2 no further application is possible.

Theorem 4. *The satisfiability problem for formulas of HXPath$_=(\downarrow)$ is* PSPACE*-complete.*

Proof. Hardness follows from the PSPACE satisfiability problem for the basic modal logic K [5]. PSPACE completeness can be proved by designing a backtracking algorithm which uses polynomial space, based on the rules from Section 3. A sketch of a non-deterministic algorithm that uses only polynomial space is shown in Algorithm 1. The algorithm explores a model "depth-first" and allows the expansion of only one $\langle \alpha * \beta \rangle$ formula at a time. The following constraints are, furthermore, assumed by the procedure. Let φ be the input formula:

– Formulas of the form $i\varphi$, $\langle i * j \rangle$ and $\neg\langle i * j \rangle$ for $i, j \in \mathsf{NOM}(\varphi)$ are never removed from the tableau once generated, and they are assumed to be preserved by the Pop operation (e.g., they are copied to the previous instance of T in the stack).

– To allow the exploration of *two* branches in the model needed to check $\langle \alpha * \beta \rangle$ formulas, we assume that $(\downarrow)$ marks the data comparison formula in the consequent using a * (e.g., $\langle m\alpha * \beta \rangle^\star$). All other rules except $(\downarrow_r)$ pass the mark to the data comparison formula in its consequent when applied to a marked formula. $(\downarrow_r)$ never marks a data comparison formula in its consequent. Notation $\varphi^{(\star)}$ indicates that $\star$ may appear or not.

Algorithm 1 PSPACE Tableaux

In: The algorithm is non-deterministic, branching rules are handled in parallel runs. All variables are global. Tableau rules are applied following the constraints described in Section 3.
Out: φ is satisfiable if at least one run returns SAT.

```
 1: T ← {0φ}, 0 not in φ
 2: ST ← [ ]
 3: loop
 4:     Saturate( )
 5:     if ChooseLeft( ) then
 6:         ExploreLeft( )
 7:     else if ChooseRight( ) then
 8:         ExploreRight( )
 9:     else
10:         Pop ST

 1: procedure Saturate( )
 2:     Apply all rules except (↓) and (↓_r)
 3:       till saturation
 4:     if T has a clash then
 5:         exit(FAIL)
 6:     if No formula waits for (↓) or (↓_r) then
 7:         exit(SAT)

 1: function ChooseLeft( )
 2:     Choose from T
 3:       Unexpanded φ = ⟨n↓α * β⟩^(⋆)
 4:         s.t. n → m ∉ T
 5: return φ was found in T?

 1: function ChooseRight( )
 2:     Let h be the highest s.t., ⟨n * h↓α⟩* ∈ T
 3:     Choose from T
 4:       Unexpanded φ = ⟨n * h↓α⟩*
 5:      otherwise
 6:       Unexpanded φ = ⟨n * m↓α⟩
 7:         s.t. m → k ∉ T
 8: return φ was found in T?

 1: procedure ExploreLeft( )
 2:     Push T in ST
 3:     Expand T using (↓)

 1: procedure ExploreRight( )
 2:     Push T in ST
 3:     Expand T using (↓_r)
```

6 Final Remarks

We have introduced a tableau calculus for the logic HXPath$_=(\downarrow)$, i.e., XPath with downward navigation and data comparison (by $=$ and $\neq$), extended with nominals and satisfaction operators. We proved that the calculus is sound, complete, and that it terminates on all inputs. As the tableaux only needs polynominal space, and the satifiability for HXPath$_=(\downarrow)$ problem is PSPACE-hard (because it embeds the satisfiability problem for the basic modal logic K, [5]) a PSPACE-complete bound follows.

Several lines of further research are worth exploring:

– Given that XPath is commonly used as a query language for XML documents, we will consider extending the calculus with rules and clash conditions to restrict the class of models to finite data trees.
– We plan to take advantage of existing techniques and implementations of tableau procedures for hybrid logics to develop a prover for HXPath$_=(\downarrow)$.

– We will investigate extensions of HXPath$_=(\downarrow)$ and consider the inclusion of additional navigation axis like descendant ($\downarrow^*$), ancestor ($\uparrow^*$), father ($\uparrow$), next-sibling ($\twoheadrightarrow$), etc.

Acknowledgements. This work was partially supported by grant ANPCyT-PICT-2013-2011, STIC-AmSud "Foundations of Graph Structured Data (FoG)", SeCyT-UNC, the Laboratoire International Associé "INFINIS", and the European Union's Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 690974 for the project MIREL: MIning and REasoning with Legal texts.

References

1. S. Abriola, M. Descotte, R. Fervari, and S. Figueira. Axiomatizations for downward XPath on data trees. *Journal of Computer and System Sciences. In Press*, 2017.
2. S. Abriola, M. Descotte, and S. Figueira. Model theory of XPath on data trees. Part II: Binary bisimulation and definability. *Information and Computation*. To appear, http://www.glyc.dc.uba.ar/santiago/papers/xpath-part2.pdf.
3. C. Areces and R. Fervari. Hilbert-style axiomatization for hybrid xpath with data. In L. M. and A. Kakas, editors, *Logics in Artificial Intelligence (JELIA'16)*, volume 10021 of *LNCS*, pages 34–48, 2016.
4. D. Baelde, S. Lunel, and S. Schmitz. A sequent calculus for a modal logic on finite data trees. In *25th EACSL Annual Conference on Computer Science Logic, (CSL'16)*, pages 32:1–32:16, 2016.
5. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.
6. P. Blackburn and J. van Benthem. Modal Logic: A Semantic Perspective. In *Handbook of Modal Logic*, pages 1–84. Elsevier, 2006.
7. M. Bojańczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *Journal of the ACM*, 56(3), 2009.
8. T. Bolander and P. Blackburn. Termination for hybrid tableaus. *Journal of Logic and Computation*, 17(3):517–554, 2007.
9. P. Buneman. Semistructured data. In *In ACM Symposium on Principles of Database Systems (PODS'97)*, pages 117–121, 1997.
10. B. t. Cate, T. Litak, and M. Marx. Complete axiomatizations for XPath fragments. *Journal of Applied Logic*, 8(2):153–172, 2010.
11. J. Clark and S. DeRose. XML path language (XPath). Website, 1999. W3C Recommendation. http://www.w3.org/TR/xpath.
12. M. D'Agostino, D. M. Gabbay, R. Hähnle, and J. Posegga, editors. *Handbook of Tableau Methods*. Springer, 1999.
13. D. Figueira. Decidability of downward XPath. *ACM Transactions on Computational Logic*, 13(4):34, 2012.
14. D. Figueira. On XPath with transitive axes and data tests. In W. Fan, editor, *In ACM Symposium on Principles of Database Systems (PODS'13)*, pages 249–260, New York, NY, USA, 2013. ACM Press.
15. D. Figueira, S. Figueira, and C. Areces. Model theory of XPath on data trees. Part I: Bisimulation and characterization. *Journal of Artificial Intelligence Research*, 53:271–314, 2015.
16. G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. *ACM Transactions on Database Systems*, 30(2):444–491, 2005.

Boolean	$\dfrac{n\neg i}{mi}\ (\neg^1)$	$\dfrac{n\neg\neg\varphi}{n\varphi}\ (\neg\neg)$	$\dfrac{n(\varphi\wedge\psi)}{\begin{array}{c}n\varphi\\ n\psi\end{array}}\ (\wedge)$	$\dfrac{n\neg(\varphi\wedge\psi)}{n\neg\varphi \quad \mid \quad n\neg\psi}\ (\neg\wedge)$
Prefix	$\dfrac{n\langle\alpha * \beta\rangle}{\langle n\alpha * n\beta\rangle}\ (Int)$	$\dfrac{n\neg\langle\alpha * \beta\rangle}{\neg\langle n\alpha * n\beta\rangle}\ (\neg Int)$		
Data	$\dfrac{\neg\langle n \neq m\rangle}{\langle n = m\rangle}\ (DEq)$	$\dfrac{\langle n = m\rangle}{\langle m = n\rangle}\ (DSym)$	$\dfrac{\langle n = m\rangle \ \ \langle m = k\rangle}{\langle n = k\rangle}\ (DTrans)$	
Nominal	$\dfrac{}{ii}\ (Ref^2)$	$\dfrac{ni}{in}\ (Sym)$	$\dfrac{nj \ \ ni \ \ mi}{mj}\ (Nom)$	$\dfrac{n\varphi \ \ ni \ \ mi}{m\varphi}\ (Copy^3)$ $\quad \dfrac{nm}{\langle n = m\rangle}\ (Data)$
XPath	$\dfrac{\langle n[\varphi]\alpha * \beta\rangle}{\begin{array}{c}n\varphi\\ \langle n\alpha * \beta\rangle\end{array}}\ (?)$	$\dfrac{\neg\langle n[\varphi]\alpha * \beta\rangle}{n\neg\varphi \ \mid \ \neg\langle n\alpha * \beta\rangle}\ (\neg?)$	$\dfrac{\langle n * m[\varphi]\alpha\rangle}{\begin{array}{c}m\varphi\\ \langle n * m\alpha\rangle\end{array}}\ (?_r)$	$\dfrac{\neg\langle n * m[\varphi]\alpha\rangle}{m\neg\varphi \ \mid \ \neg\langle n * m\alpha\rangle}\ (\neg?_r)$
	$\dfrac{\langle n\downarrow\alpha * \beta\rangle}{\begin{array}{c}n \to m\\ \langle m\alpha * \beta\rangle\end{array}}\ (\downarrow^1)$	$\dfrac{\neg\langle n\downarrow\alpha * \beta\rangle \ \ n \to m}{\neg\langle m\alpha * \beta\rangle}\ (\neg\downarrow)$	$\dfrac{\langle n * m\downarrow\alpha\rangle}{\begin{array}{c}m \to k\\ \langle n * k\alpha\rangle\end{array}}\ (\downarrow^4_r)$	$\dfrac{\neg\langle n * m\downarrow\alpha\rangle \ \ m \to k}{\neg\langle n * k\alpha\rangle}\ (\neg\downarrow_r)$
	$\dfrac{\langle n(\alpha\cup\beta)\gamma * \delta\rangle}{\langle n\alpha\gamma * \delta\rangle \ \mid \ \langle n\beta\gamma * \delta\rangle}\ (\cup)$	$\dfrac{\neg\langle n(\alpha\cup\beta)\gamma * \delta\rangle}{\begin{array}{c}\neg\langle n\alpha\gamma * \delta\rangle\\ \neg\langle n\beta\gamma * \delta\rangle\end{array}}\ (\neg\cup)$	$\dfrac{\langle n * m(\alpha\cup\beta)\gamma\rangle}{\langle n * m\alpha\gamma\rangle \ \mid \ \langle n * m\beta\gamma\rangle}\ (\cup_r)$	$\dfrac{\neg\langle n * m(\alpha\cup\beta)\gamma\rangle}{\begin{array}{c}\neg\langle n * m\alpha\gamma\rangle\\ \neg\langle n * m\beta\gamma\rangle\end{array}}\ (\neg\cup_r)$
	$\dfrac{\langle ni\alpha * \beta\rangle}{\begin{array}{c}mi\\ \langle m\alpha * \beta\rangle\end{array}}\ (@^1)$	$\dfrac{\neg\langle ni\alpha * \beta\rangle}{\begin{array}{c}mi\\ \neg\langle m\alpha * \beta\rangle\end{array}}\ (\neg@^1)$	$\dfrac{\langle n * mi\alpha\rangle}{\begin{array}{c}ki\\ \langle n * k\alpha\rangle\end{array}}\ (@^4_r)$	$\dfrac{\neg\langle n * mi\alpha\rangle}{\begin{array}{c}ki\\ \neg\langle n * k\alpha\rangle\end{array}}\ (\neg@^4_r)$

[1] m is the smallest nominal that has not appeared in the tableau.
[2] i appears in the tableau.
[3] m is the smallest nominal making i true in the tableau.
[4] k is the smallest nominal that has not appeared in the tableau.

Fig. 2. Tableaux Rules