DATA-AWARE HYBRID TABLEAUX

CARLOS ARECES [a,b], VALENTIN CASSANO [a,b,c], AND RAUL FERVARI [a,b]

[a] Universidad Nacional de Córdoba, Argentina
e-mail address: carlos.areces@unc.edu.ar, rfervari@unc.edu.ar

[b] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina

[c] Universidad Nacional de Río Cuarto, Argentina
e-mail address: valentin@dc.exa.unrc.edu.ar

Abstract. Labelled tableaux have been a traditional approach to define satisfiability checking procedures for Modal Logics. In many cases, they can also be used to obtain tight complexity bounds and lead to efficient implementations of reasoning tools. More recently, it has been shown that the expressive power provided by the operators characterizing Hybrid Logics (nominals and satisfiability modalities) can be used to *internalize* labels, leading to well-behaved inference procedures for fairly expressive logics. The resulting procedures are attractive because they do not use external mechanisms outside the language of the logic at hand, and have good logical and computational properties.

Many tableau systems based on Hybrid Logic have been investigated, with more recent efforts concentrating on Modal Logics that support data comparison operators. Here, we introduce an internalized tableau calculus for XPath, arguably one of the most prominent approaches for querying semistructured data. More precisely, we define data-aware tableaux for XPath featuring data comparison operators and enriched with nominals and the satisfiability modalities from Hybrid Logic. We prove that the calculus is sound, complete and terminating. Moreover, we show that tableaux can be explored in polynomial space, therefore establishing that the satisfiability problem for the logic is PSpace-complete. Finally, we explore different extensions of the calculus, in particular how to handle data trees and other frame classes.

1. Introduction

In many data-centric applications –such as those handling large volumes of web or medical information– explicit reasoning about both structure and content is essential. Classical relational databases often fall short in expressiveness for such use cases, leading to the development of *semi-structured data models*, typically based on labeled trees or graphs [Bun97, ABS99]. A prominent example of such semi-structured data models is XML (eXtensible Markup Language), which encodes structural information via labels from a finite alphabet, and data content using values drawn from an infinite domain. XML documents are commonly queried using XPath, a standard language widely used in tools like XSLT [Cla99] and XQuery [RCDS14].

XPath is, fundamentally, a general purpose language for addressing, searching, and matching pieces of an XML document. It is an open standard World Wide Web Consortium

Key words and phrases: XPath, data graphs, axiomatizations, data trees, decidability.

(W3C) Recommendation [CD99]. Core-XPath, the fragment of XPath 1.0 that captures its navigational capabilities [GKP05], can express structural properties (e.g., tag names, tree traversal), but not conditions involving the actual data values. Logically, Core-XPath corresponds to a classical modal logic [BdRV01, BvB06], and its satisfiability problem is decidable, even under DTD constraints [Mar04, BFG08]. Its expressive power has been precisely characterized: it is equivalent to two-variable first-order logic (FO^2) on trees [MdR05] and it is strictly less expressive than PDL with converse [BK09]. Sound and complete axiomatizations are available in [CM09, CLM10].

However, Core-XPath's inability to compare data values across nodes limits its applicability, notably preventing the expression of joins –an essential construct in query languages. To address this, the logic $\mathsf{XPath}_{\mathsf{D}}$ (also known as Core-Data-XPath [BMSS09]) extends Core-XPath with data-aware features, allowing for expressions such as $\langle \alpha =_{\mathsf{c}} \beta \rangle$ or $\langle \alpha \neq_{\mathsf{c}} \beta \rangle$, where α and β are path expressions navigating the XML tree via axes like child, descendant, or sibling. Such expressions enable comparisons of data values for c in nodes reachable via different paths, allowing for significantly richer queries.

By way of example, an expression of the form $\langle \alpha =_{\mathsf{c}} \beta \rangle$ is true at a node x in a data tree if there exist two nodes reachable from x via α and β paths respectively, such that the data values for c of y and z are equal. This situation is illustrated in Fig. 1 depicting how the expression *"there is a one-step descendant via* e *and a two-steps descendant via* e *with the same data value for* price*"* (written $\langle \mathsf{e} =_{\mathsf{price}} \mathsf{ee} \rangle$) holds at x, given the presence of u and z. The expression *"there are two two-steps descendant with different data value for* price*"* (written $\langle \mathsf{ee} \neq_{\mathsf{price}} \mathsf{ee} \rangle$) is also true at x, because v and w have different data for price.

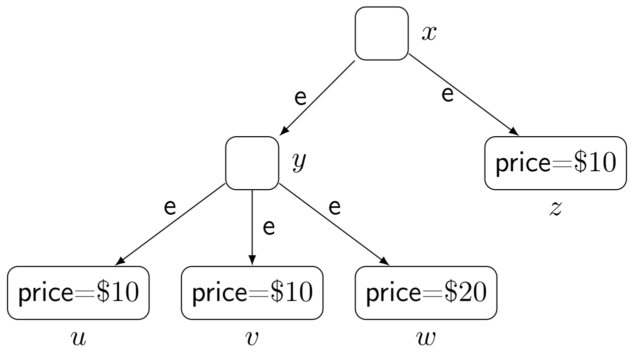

FIGURE 1. An example of a data tree.

The language of $\mathsf{XPath}_{\mathsf{D}}$ allows us 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. This makes it 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 linking nodes with the same value. In the data tree from Fig. 1, the relation consists of four equivalence classes: $\{u, v, z\}$, $\{w\}$, $\{x\}$ and $\{y\}$. This approach treats uniformly nodes with unique data values of a certain type and nodes where data of that type is undefined (see [ACDF23] for a treatment of incomplete information).

Recent articles investigate $\mathsf{XPath}_{\mathsf{D}}$ from a modal logic perspective. For example, the complexity of the satisfiability and evaluation problems of different fragments are discussed in [Fig10, FS11, Fig12, Fig13], while model theory and expressivity are studied in [ADF14, FFA14, FFA15, ADF17, ABFF16, GA21].

In this article, we will focus on reasoning methods for $\mathsf{XPath}_{\mathsf{D}}$. In this line of work, a Gentzen-style sequent calculus is given for a very restricted fragment of $\mathsf{XPath}_{\mathsf{D}}$, named DataGL, in [BLS16]. In DataGL, data comparisons are allowed only between the evaluation point and its successors. An extension of the equational axiomatic system from [CLM10] is introduced in [ADFF17] that is proved sound and complete for $\mathsf{XPath}_{\mathsf{D}}$. This extension allows downward navigation and equality/inequality tests. More recently, [AF16, AF21] provide an axiomatization for this logic, extended with upward navigation, and with nominals and satisfiability modalities from Hybrid Logic [AtC06]. As argued in, e.g., [Bra07], features from Hybrid Logic can be effectively leveraged in the design of proof methods for modal logics. In [AFS17], we introduced sound and complete tableau calculus for hybrid $\mathsf{XPath}_{\mathsf{D}}$.

Contributions. In this article we start by introducing a sound, complete and terminating tableau calculus for $\mathsf{XPath}_{\mathsf{D}}$ 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 satisfiability modality (allowing the navigation to some particular named node), over the class of all data-graphs. We call this logic $\mathsf{HXPath}_{\mathsf{D}}$.

Taking inspiration from [BB07], we use the hybrid logic operators to provide internalized tableau rules, which we prove are sound and complete for $\mathsf{HXPath}_{\mathsf{D}}$. The main intuition is that nominals and satisfiability modalities can be used in tableaux to keep track of the evaluation of an expression during an attempt to build a model. This enables us to define tableaux without any help of external features, like labels or relational annotations, as they can be internalized into the language of $\mathsf{HXPath}_{\mathsf{D}}$. Crucially, we show how the hybrid machinery can also be used to help with data comparisons. We show that the tableau calculus can be ran in polynomial space, without compromising completeness, establishing in this way that the satisfiability problem for $\mathsf{HXPath}_{\mathsf{D}}$ is PSpace-complete.

We then extend our tableaux to handle different structures. First, we show how to enforce certain reachability conditions over the accessibility relations. With this at hand, we provide a method to check satisfiability of $\mathsf{HXPath}_{\mathsf{D}}$ over the class of data trees, and show that the complexity of the satisfiability problem remains in PSpace. Then, we extend the calculus with so-called pure axioms and node creating rules. These two extensions can be used to characterize many other model classes, and introduce new operators. We show that soundness and completeness of the resulting tableau calculi follow automatically (even though termination can be compromised).

This work is an extension of [AFS17] and improves on it in several ways. Most importantly we provide details on the termination argument, which was only sketched in the previous article, and show that the PSpace upper bound holds also for trees. To this end, we start by introducing changes to the language that simplify proofs. Concretely, we extend the base language with primitive expressions of the form $\langle \mathsf{a} \rangle \varphi$. This enables us to obtain internalized tableaux, and consequently to prove termination in a simpler way. Second, we reduce the number of rules by exploring only the left side in a data comparison and adding a commutativity rule. Finally, the current presentation includes a detailed algorithm suitable for implementation.

The contributions in this article serve as a starting point for research and mechanization of other data-aware logics. For example our approach can be use to investigate the logical properties of languages like GQL [FGG$^+$23a, FGG$^+$23b] and SHACL [Ort23, AOOS23], Modal Logics with Concrete Domains (see, e.g., [DQ21]), and to study connections with Register Automata (see, e.g., [KF94]).

Organization. Section 2 introduces the syntax and semantics of $\mathsf{HXPath_D}$. Section 3 deals with a tableau calculus for $\mathsf{HXPath_D}$, and with examples of how to build tableaux for concrete expressions. This section also shows the calculus is sound and complete with respect to the class of all data models. Section 4 deals with the proof of termination of the calculus, whereas Section 5 shows that the satisfiability problem for $\mathsf{HXPath_D}$ is PSpace-complete. Section 6 extends the calculus to work over the class of forest and tree models, and discusses completeness and complexity, showing that the complexity of the satisfiability problem over these classes remains PSpace-complete. Section 7 introduces extensions of the calculus with pure axioms and node creating rules. These new rules permits the characterization of new classes of models and the definition of new operators. we argue that the resulting tableau calculi remain sound and complete (but termination might be compromised). Section 8 includes final remarks and discusses future lines of research.

2. Preliminaries

We begin by introducing the syntax and the semantics of $\mathsf{HXPath_D}$. We assume a fixed *signature* of pairwise disjoint sets: $\mathsf{Prop} = \{p, q, r, \dots\}$ of *propositional* symbols, $\mathsf{Nom} = \{i, j, k, \dots\}$ of *nominals*, $\mathsf{Rel} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \dots\}$ of symbols for *accessibility relations*, and $\mathsf{Eq} = \{\mathsf{e}, \mathsf{f}, \mathsf{g}, \dots\}$ of *comparisons by data (in)equality*. We further assume that Prop and Nom are countably infinite, and that Rel and Eq are finite.

Definition 2.1. The *language* of $\mathsf{HXPath_D}$ consists of *path* expressions α, β, γ, $\dots$, and *node* expressions φ, ψ, χ, $\dots$. Path and node expressions are defined by mutual recursion according to the grammar:

$$\alpha, \beta ::= \mathsf{a} \mid @_i \mid \varphi? \mid \alpha\beta \mid \alpha \cup \beta$$
$$\varphi, \psi ::= p \mid i \mid \neg\varphi \mid \varphi \wedge \psi \mid i{:}\varphi \mid \langle \mathsf{a} \rangle \varphi \mid \langle \alpha =_{\mathsf{e}} \beta \rangle \mid \langle \alpha \neq_{\mathsf{e}} \beta \rangle,$$

where $p \in \mathsf{Prop}$, $i \in \mathsf{Nom}$, $\mathsf{a} \in \mathsf{Rel}$, and $\mathsf{e} \in \mathsf{Eq}$. By way of notation, we use $\blacktriangle_{\mathsf{e}}$ when there is no need to distinguish between $=_{\mathsf{e}}$ and $\neq_{\mathsf{e}}$. We sometimes refer to node expressions of the form $i{:}\varphi$, $\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_j\beta \rangle$, or $\neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_j\beta \rangle$, as *prefixed expressions.*

In the syntax above we find nominals and satisfiability statements. The latter come in two flavours: as path expressions of the form $@_i\alpha$, and as node expressions of the form $i{:}\varphi$. The modal diamond $\langle \mathsf{a} \rangle \varphi$ is included as primitive (as we will see, this has a positive impact in the definition of tableaux in Section 3). Notice that we are in a multi-modal system, as we have one modal diamond for each $\mathsf{a} \in \mathsf{Rel}$. We also allow for different (in)equality comparisons, one for each $\mathsf{e} \in \mathsf{Eq}$. Finally, we make use the following abbreviations for node expressions:

$$\varphi \rightarrow \psi := \neg\varphi \vee \psi \qquad \varphi \leftrightarrow \psi := (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) \qquad \varphi \vee \psi := \neg(\neg\varphi \wedge \neg\psi)$$
$$\langle \alpha \rangle \varphi := \langle \alpha\varphi? = \alpha\varphi? \rangle \qquad [\alpha]\varphi := \neg\langle \alpha \rangle \neg\varphi \qquad [\alpha \blacktriangle_{\mathsf{e}} \beta] := \neg\langle \alpha \blacktriangledown_{\mathsf{e}} \beta \rangle.$$

The symbol $\blacktriangledown_{\mathsf{e}}$ above is $\neq_{\mathsf{e}}$ if $\blacktriangle_{\mathsf{e}}$ is $=_{\mathsf{e}}$, and is $=_{\mathsf{e}}$ if $\blacktriangle_{\mathsf{e}}$ is $\neq_{\mathsf{e}}$.

The semantics of path and node expressions is given with respect to *hybrid data models.*

Definition 2.2. A *(hybrid data) model* is a tuple $\mathfrak{M} = \langle N, \{R_{\mathsf{a}}\}_{\mathsf{a} \in \mathsf{Rel}}, \{\approx_{\mathsf{e}}\}_{\mathsf{e} \in \mathsf{Eq}}, V, \mathrm{nom} \rangle$ where:

(1) N is a non-empty set of elements called *nodes*,
(2) each $R_{\mathsf{a}} \subseteq N \times N$ is a binary *accessibility* relation,

(3) each $\approx_{\mathsf{e}} \subseteq N \times N$ is an *equivalence* relation,
(4) $V : \mathsf{Prop} \to 2^N$ is a *valuation* function, and
(5) $\mathrm{nom} : \mathsf{Nom} \to N$ is a *naming* function.

In brief, the models of $\mathsf{HXPath_D}$ can be understood as the typical models of Hybrid Modal Logic, see, e.g., [AtC06], augmented with a collection of equivalence relations intended to capture equality criteria between data values in nodes. The *satisfiability* relation is defined below. Notice how satisfiability of a path expression on a model is relative to two nodes (as it describes the accessibility from one to the other), whereas satisfiability of a node expression is at a particular node (stating truth at that node).

Precisely, the path expression a can be understood as traversing an edge labelled by a in the graph. In turn, $@_i$ can be understood as a "go to" statement, i.e., jump to *the* node named by the nominal i. Then, $\varphi?$ can be understood as a "test on the truth of φ". Finally, $\alpha\beta$ can be understood as the "concatenation" of α and β, and $\alpha \cup \beta$ as the "non-deterministic choice" between α and β. The intuitive semantics of the node expressions of the form p, i, $\neg\varphi$, $\varphi \wedge \psi$, $i{:}\varphi$, and $\langle \mathsf{a} \rangle\varphi$ are the same as in Hybrid Modal Logic (see, e.g., [AtC06] for details). Specifically, $i{:}\varphi$ can be understood as "φ is true at the node named by i", while $\langle \mathsf{a} \rangle\varphi$ is read as "φ is true after traversing some a-edge". Finally, the intuitive semantics of data equalities $\langle \alpha =_{\mathsf{e}} \beta \rangle$ (and data inequalities $\langle \alpha \neq_{\mathsf{e}} \beta \rangle$) is "there are α and β paths with a common starting point whose end points have data values that are (not) equal according to the criteria determined by e".

Definition 2.3. Let $\mathfrak{M} = \langle N, \{R_{\mathsf{a}}\}_{\mathsf{a} \in \mathsf{Rel}}, \{\approx_{\mathsf{e}}\}_{\mathsf{e} \in \mathsf{Eq}}, V, \mathrm{nom} \rangle$ be a model, and $n, n' \in N$. Then:

$$\begin{array}{lll}
\mathfrak{M}, n, n' \models \mathsf{a} & \textit{iff} & nR_{\mathsf{a}}n' \\
\mathfrak{M}, n, n' \models @_i & \textit{iff} & \mathrm{nom}(i) = n' \\
\mathfrak{M}, n, n' \models \varphi? & \textit{iff} & n = n' \text{ and } \mathfrak{M}, n \models \varphi \\
\mathfrak{M}, n, n' \models \alpha\beta & \textit{iff} & \text{exists } z \text{ s.t. } \mathfrak{M}, n, z \models \alpha \text{ and } \mathfrak{M}, z, n' \models \beta \\
\mathfrak{M}, n, n' \models \alpha \cup \beta & \textit{iff} & \mathfrak{M}, n, n' \models \alpha \text{ or } \mathfrak{M}, n, n' \models \beta \\
\mathfrak{M}, n \models p & \textit{iff} & n \in V(p) \\
\mathfrak{M}, n \models i & \textit{iff} & \mathrm{nom}(i) = n \\
\mathfrak{M}, n \models \neg\varphi & \textit{iff} & \mathfrak{M}, n \not\models \varphi \\
\mathfrak{M}, n \models \varphi \wedge \psi & \textit{iff} & \mathfrak{M}, n \models \varphi \text{ and } \mathfrak{M}, n \models \psi \\
\mathfrak{M}, n \models \langle \mathsf{a} \rangle\varphi & \textit{iff} & \text{exists } z \text{ s.t. } nR_{\mathsf{a}}z \text{ and } \mathfrak{M}, z \models \varphi \\
\mathfrak{M}, n \models i{:}\varphi & \textit{iff} & \mathfrak{M}, \mathrm{nom}(i) \models \varphi \\
\mathfrak{M}, n \models \langle \alpha =_{\mathsf{e}} \beta \rangle & \textit{iff} & \text{exist } z, z' \text{ s.t. } \mathfrak{M}, n, z \models \alpha,\ \mathfrak{M}, n, z' \models \beta, \text{ and } z \approx_{\mathsf{e}} z' \\
\mathfrak{M}, n \models \langle \alpha \neq_{\mathsf{e}} \beta \rangle & \textit{iff} & \text{exist } z, z' \text{ s.t. } \mathfrak{M}, n, z \models \alpha,\ \mathfrak{M}, n, z' \models \beta, \text{ and } z \not\approx_{\mathsf{e}} z'.
\end{array}$$

A node expression φ is satisfiable iff there is $\mathfrak{M}, n$ s.t., $\mathfrak{M}, n \models \varphi$. In turn, φ is *valid*, written $\models \varphi$ iff $\mathfrak{M}, n \models \varphi$, for all $\mathfrak{M}, n$. These notions are analogously defined for path expressions.

We conclude this section by listing some immediate properties for node expressions. These properties are useful in the justification of tableaux rules in Def. 3.1.

Proposition 2.4. *Let $\mathfrak{M}$ be any model. The following node expressions are valid in $\mathfrak{M}$:*

$$\begin{array}{cc}
i{:}\langle \alpha \blacktriangle_{\mathsf{e}} \beta \rangle \leftrightarrow \langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta \rangle & i{:}\neg\langle \alpha \blacktriangle_{\mathsf{e}} \beta \rangle \leftrightarrow \neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta \rangle \\
\langle \varphi?\alpha \blacktriangle_{\mathsf{e}} \beta \rangle \leftrightarrow \varphi \wedge \langle \alpha \blacktriangle_{\mathsf{e}} \beta \rangle & \neg\langle \varphi?\alpha \blacktriangle_{\mathsf{e}} \beta \rangle \leftrightarrow \neg\varphi \vee \neg\langle \alpha \blacktriangle_{\mathsf{e}} \beta \rangle \\
\langle (\alpha \cup \beta)\gamma \blacktriangle_{\mathsf{e}} \eta \rangle \leftrightarrow \langle \alpha\gamma \blacktriangle_{\mathsf{e}} \eta \rangle \vee \langle \beta\gamma \blacktriangle_{\mathsf{e}} \eta \rangle & \neg\langle (\alpha \cup \beta)\gamma \blacktriangle_{\mathsf{e}} \eta \rangle \leftrightarrow \neg\langle \alpha\gamma \blacktriangle_{\mathsf{e}} \eta \rangle \wedge \neg\langle \beta\gamma \blacktriangle_{\mathsf{e}} \eta \rangle.
\end{array}$$

3. Tableaux for $\mathsf{HXPath}_{\mathsf{D}}$

In this section we present the notion of tableaux for $\mathsf{HXPath}_{\mathsf{D}}$, comment on the view of these tableaux as a calculus for satisfiability, and prove the soundness and completeness of this calculus. Our ideas and results build on the terminating tableau calculus for Hybrid Modal Logic in [BB07]. In brief, we incorporate to such a calculus rules for handling data (in)equalities $\langle \alpha \blacktriangle_{\mathsf{e}} \beta \rangle$ and $[\alpha \blacktriangle_{\mathsf{e}} \beta]$. We consider our presentation of tableaux for $\mathsf{HXPath}_{\mathsf{D}}$ simpler and more elegant than the one presented in [AFS17]. First, the internalization of rules in our calculus simplifies their manipulation. Second, our way of handling the left-hand-side of data (in)equalities together with a commutation rule significantly reduces the number of rules. For the remainder of the article, we assume that Nom is the set of natural numbers.

3.1. Tableau Calculus.

We begin with the definition of a tableau for a node expression. We assume familiarity with the usual terminology on labelled trees. In particular, the notion of a root, a node, a successor of a node, a branch, a leaf, and the label of a node. Moreover, we assume some familiarity with tableaux for Hybrid Logic (see [BB07]).

Definition 3.1. Let φ be a node expression and i a new nominal. A *tableau* for φ is a labelled tree constructed from an initial single node tree labelled with $i{:}\varphi$ using the branch *extension* rules in Figs. 2 to 5. Crucially, rules can be applied only once to its premisses. In this definition, a *new* nominal is the smallest nominal that is larger than all other nominals appearing in the tableau (thus far constructed). In turn, a *root* nominal is one appearing in φ. Finally, a node expression $i{:}\langle \mathsf{a} \rangle j$ is an *accessibility constraint* iff it appears in the conclusion of the rules $(\Diamond)$ and (child).

Let us briefly discuss how tableaux rules are meant to be understood. In general, all rules are taken to be read as: if the premisses of the rule are in the branch, then, extend the branch by adding to its leaf node successors labelled with the conclusions of the rule. Each rule is applied only once to a given set of premisses present in a branch. Fig. 2 presents the usual rules for Boolean and modal operators of Basic Modal Logic, with the distinguishing characteristic of accessibility constraints being internalized in the language using nominals. Fig. 3 presents standard rules for handling nominals, similar to those found in [BB07, Bra11]. The only difference is the rule (copy) whose side conditions are in place to guarantee termination of the calculus. Fig. 4 introduces rules for data comparison, they internalize nominals into data (in)equalities. More precisely, notice how the rule (int_1) takes a satisfiability statement $i{:}\langle \alpha \blacktriangle_{\mathsf{e}} \beta \rangle$, and internalizes $i{:}$ as an $@_i$-prefix in α and β resulting in $\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta \rangle$. In doing this, we anchor the point of evaluation of the node expression at the beginning of each path in the data comparison. This enables us to work out paths individually using the rules in Fig. 5. The rule (int_2) does the same job in presence of a negation. In turn, $(\neg\blacktriangle_{\mathsf{e}})$ internalizes negated equalities (inequalities) into inequalities (equalities). Finally, (dRef) tells us that nodes with the same name must have the same data values. Fig. 5 depicts the last set of rules in our tableaux. These rules tell us how to deal with path expressions in data comparisons. Essentially, (child) mimics what is done in $(\Diamond)$ but in the context of a data comparison, whereas $(\neg\text{child})$ is analogous to $(\neg\Diamond)$. The rule (test) eliminates φ from the path expression and evaluates it as a standalone node expression, while the case for $(\neg\text{test})$ is dual. The rules $(@)$ and $(\neg@)$ can be understood in analogy with (nom) and $(\neg\text{nom})$. The rationale behind $(\cup)$ and $(\neg\cup)$ is immediately understood from Prop. 2.4. A novelty of our calculus is that rules only decompose, step-by-step, path

Basic

$$\frac{i{:}\neg\neg\varphi}{i{:}\varphi}\,(\neg\neg) \qquad \frac{i{:}(\varphi\wedge\psi)}{\begin{array}{c}i{:}\psi\\ i{:}\varphi\end{array}}\,(\wedge) \qquad \frac{i{:}\neg(\varphi\wedge\psi)}{i{:}\neg\varphi \quad i{:}\neg\psi}\,(\neg\wedge) \qquad \frac{i{:}\langle\mathsf{a}\rangle\varphi}{\begin{array}{c}i{:}\langle\mathsf{a}\rangle j\\ j{:}\varphi\end{array}}\,(\Diamond^1) \qquad \frac{\begin{array}{c}i{:}\neg\langle\mathsf{a}\rangle\varphi\\ i{:}\langle\mathsf{a}\rangle j\end{array}}{j{:}\neg\varphi}\,(\neg\Diamond^2)$$

[1] $i{:}\langle\mathsf{a}\rangle\varphi$ is not an accessibility constraint. j is a new nominal.
[2] $i{:}\langle\mathsf{a}\rangle j$ is an accessibility constraint.

Figure 2. Basic Rules

Satisfiability and Nominals

$$\frac{i{:}(j{:}\varphi)}{j{:}\varphi}\,(\text{nom}) \qquad \frac{i{:}\neg(j{:}\varphi)}{j{:}\neg\varphi}\,(\neg\text{nom}) \qquad \frac{\begin{array}{c}i{:}\varphi\\ i{:}j\end{array}}{j{:}\varphi}\,(\text{copy}^1) \qquad \frac{}{i{:}i}\,(\text{ref}^2) \qquad \frac{j{:}i}{i{:}j}\,(\text{sym}) \qquad \frac{\begin{array}{c}i{:}k\\ i{:}l\\ j{:}l\end{array}}{j{:}k}\,(\text{trans})$$

[1] φ is not a nominal, j is a root nominal, $j < i$.
[2] i is a nominal in the branch.

Figure 3. Rules for Nominals

Internalization into Data (In)Equalities

$$\frac{i{:}\langle\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}{\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta\rangle}\,(\text{int}_1) \qquad \frac{i{:}\neg\langle\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}{\neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta\rangle}\,(\text{int}_2) \qquad \frac{\neg\langle @_i \blacktriangle_{\mathsf{e}} @_j\rangle}{\langle @_i \blacktriangledown_{\mathsf{e}} @_j\rangle}\,(\neg\blacktriangle_{\mathsf{e}}) \qquad \frac{i{:}j}{\langle @_i =_{\mathsf{e}} @_j\rangle}\,(\text{dRef})$$

Figure 4. Rules for Internalizing Nominals and Negations into Data (In)Equalities

Path Expressions in Data (In)Equalities

$$\frac{\langle @_i\mathsf{a}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}{\begin{array}{c}i{:}\langle\mathsf{a}\rangle j\\ \langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle\end{array}}\,(\text{child}^1) \qquad \frac{\begin{array}{c}\neg\langle @_i\mathsf{a}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle\\ i{:}\langle\mathsf{a}\rangle j\end{array}}{\neg\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}\,(\neg\text{child}) \qquad \frac{\langle @_i \blacktriangle_{\mathsf{e}} \beta\rangle}{\langle \beta \blacktriangle_{\mathsf{e}} @_i\rangle}\,(\text{com}_1)$$

$$\frac{\langle @_i\varphi?\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}{\begin{array}{c}i{:}\varphi\\ \langle @_i\alpha \blacktriangle_{\mathsf{e}} \beta\rangle\end{array}}\,(\text{test}) \qquad \frac{\neg\langle @_i\varphi?\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}{i{:}\neg\varphi \quad \neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}\,(\neg\text{test}) \qquad \frac{\neg\langle @_i \blacktriangle_{\mathsf{e}} \beta\rangle}{\neg\langle \beta \blacktriangle_{\mathsf{e}} @_i\rangle}\,(\text{com}_2)$$

$$\frac{\langle @_i @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}{\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}\,(@) \qquad \frac{\neg\langle @_i @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}{\neg\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}\,(\neg @) \qquad \frac{\begin{array}{c}\langle @_i =_{\mathsf{e}} @_k\rangle\\ \langle @_k =_{\mathsf{e}} @_j\rangle\end{array}}{\langle @_i =_{\mathsf{e}} @_j\rangle}\,(\text{dTrans})$$

$$\frac{\langle @_i(\alpha\cup\beta)\gamma \blacktriangle_{\mathsf{e}} \eta\rangle}{\langle @_i\alpha\gamma \blacktriangle_{\mathsf{e}} \eta\rangle \quad \langle @_i\beta\gamma \blacktriangle_{\mathsf{e}} \eta\rangle}\,(\cup) \qquad \frac{\neg\langle @_i(\alpha\cup\beta)\gamma \blacktriangle_{\mathsf{e}} \eta\rangle}{\begin{array}{c}\neg\langle @_i\alpha\gamma \blacktriangle_{\mathsf{e}} \eta\rangle\\ \neg\langle @_i\beta\gamma \blacktriangle_{\mathsf{e}} \eta\rangle\end{array}}\,(\neg\cup)$$

[1] j is a new nominal.

Figure 5. Rules for Handling Path Expressions in Data (In)Equalities

1.	$4{:}\langle \mathsf{a}\rangle\langle @_2\mathsf{b}2? =_{\mathsf{e}} \mathsf{b}(q \wedge 3)?\rangle$	root
2.	$4{:}\langle \mathsf{a}\rangle 5$	$1\ (\Diamond)$
3.	$5{:}\langle @_2\mathsf{b}2? =_{\mathsf{e}} \mathsf{b}(q \wedge 3)?\rangle$	$1\ (\Diamond)$
4.	$\langle @_5 @_2\mathsf{b}2? =_{\mathsf{e}} @_5\mathsf{b}(q \wedge 3)?\rangle$	$3\ (\mathrm{int}_1)$
5.	$\langle @_2\mathsf{b}2? =_{\mathsf{e}} @_5\mathsf{b}(q \wedge 3)?\rangle$	$4\ (@)$
6.	$2{:}\langle \mathsf{b}\rangle 6$	5 (child)
7.	$\langle @_6 2? =_{\mathsf{e}} @_5\mathsf{b}(q \wedge 3)?\rangle$	6 (child)
8.	$6{:}2$	7 (test)
9.	$2{:}6$	8 (sym)
10.	$\langle @_6 =_{\mathsf{e}} @_5\mathsf{b}(q \wedge 3)?\rangle$	7 (test)
11.	$\langle @_5\mathsf{b}(q \wedge 3)? =_{\mathsf{e}} @_6\rangle$	$10\ (\mathrm{com}_1)$
12.	$5{:}\langle \mathsf{b}\rangle 7$	11 (child)
13.	$\langle @_7(q \wedge 3)? =_{\mathsf{e}} @_6\rangle$	11 (child)
14.	$7{:}(q \wedge 3)$	13 (test)
15.	$\langle @_7 =_{\mathsf{e}} @_6\rangle$	13 (test)
16.	$\langle @_6 =_{\mathsf{e}} @_7\rangle$	$15\ (\mathrm{com}_1)$
17.	$7{:}q$	$14\ (\wedge)$
18.	$7{:}3$	$14\ (\wedge)$
19.	$3{:}7$	18 (sym)
20.	$3{:}q$	17, 18 (copy)

FIGURE 6. A tableau for $\langle \mathsf{a}\rangle\langle @_2\mathsf{b}2? =_{\mathsf{e}} \mathsf{b}(q \wedge 3)?\rangle$

expressions on the left side of a data comparison. The treatment of the right side takes over after applying (com_1) or (com_2). Finally, (dTrans) corresponds to transitivity of equalities.

Example 3.2. Fig. 6 depicts a tableau for $\langle \mathsf{a}\rangle\langle @_2\mathsf{b}2? =_{\mathsf{e}} \mathsf{b}(q \wedge 3)?\rangle$, starting from an initial node labelled by $4{:}\langle \mathsf{a}\rangle\langle @_2\mathsf{b}2? =_{\mathsf{e}} \mathsf{b}(q \wedge 3)?\rangle$. The annotations on the right indicate the rules we applied and the nodes we applied them to. The tableau has a single branch, which becomes saturated if we apply the rules (ref) and (dRef) to all the nominals it contains –e.g., if we apply (ref) to 2, we obtain a node 18 with label 2:2, and if we apply (dRef) to the new node, we obtain a node 19 with label $\langle @_2 =_{\mathsf{e}} @_2\rangle$; and so on for all the nominals in the branch.

We consider our tableaux to be simpler and more elegant than the proposal in [AFS17]. Treating only the left side of data comparisons significantly reduces the number of rules. It is also clear that our tableaux are *internalized*; they have (only) node expressions as labels. Internalized tableaux have important advantages. In particular, it will enable us to define extended calculi to handle many different frame conditions, and obtain completeness automatically as it is done in [Bla00, BtC02] (see Section 7). Our tableaux are also *analytic* in the sense of Def. 3.5 below (see, e.g., [Smu68] for background on analytic tableaux). This property is helpful to provide a termination argument (see Section 4).

Definition 3.3. The *subcomponents* of a node expression φ are defined as

$$\begin{aligned} \mathrm{sub}(p) &= \{p\} \\ \mathrm{sub}(i) &= \{i\} \\ \mathrm{sub}(\neg\varphi) &= \{\neg\varphi\} \cup \mathrm{sub}(\varphi) \end{aligned}$$

$$\begin{aligned}
\mathrm{sub}(\varphi \wedge \psi) &= \{\varphi \wedge \psi\} \cup \mathrm{sub}(\varphi) \cup \mathrm{sub}(\psi)\\
\mathrm{sub}(i{:}\varphi) &= \{i,\, i{:}\varphi\} \cup \mathrm{sub}(\varphi)\\
\mathrm{sub}(\langle \mathsf{a}\rangle\varphi) &= \{\langle \mathsf{a}\rangle\varphi\} \cup \mathrm{sub}(\varphi)\\
\mathrm{sub}(\langle \mathbf{s} \blacktriangle_{\mathsf{e}} \mathbf{b}\rangle) &= \{\langle \mathbf{s} \blacktriangle_{\mathsf{e}} \mathbf{b}\rangle\} \cup \mathrm{sub}'(\mathbf{s}) \cup \mathrm{sub}'(\mathbf{b})\\
\mathrm{sub}(\langle \mathbf{s} \blacktriangle_{\mathsf{e}} \mathbf{b}\beta\rangle) &= \{\langle \mathbf{s} \blacktriangle_{\mathsf{e}} \mathbf{b}\beta\rangle\} \cup \mathrm{sub}(\langle \mathbf{b}\beta \blacktriangle_{\mathsf{e}} \mathbf{s}\rangle)\\
\mathrm{sub}(\langle \mathbf{s}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle) &= \{\langle \mathbf{s}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle\} \cup \mathrm{sub}(\langle \alpha \blacktriangle_{\mathsf{e}} \beta\rangle) \cup \mathrm{sub}'(\mathbf{s})\\
\mathrm{sub}(\langle (\alpha \cup \beta)\gamma \blacktriangle_{\mathsf{e}} \xi\rangle) &= \{\langle (\alpha \cup \beta)\gamma \blacktriangle_{\mathsf{e}} \xi\rangle\} \cup \mathrm{sub}\left(\langle \alpha\gamma \blacktriangle_{\mathsf{e}} \xi\rangle\right) \cup \mathrm{sub}\left(\langle \beta\gamma \blacktriangle_{\mathsf{e}} \xi\rangle\right).
\end{aligned}$$

where $\mathbf{s} \in \{@_i, \mathsf{a}, \varphi?\}$. $\mathrm{sub}'(@_i) = \{i\}$, $\mathrm{sub}'(\mathsf{a}) = \emptyset$, and $\mathrm{sub}'(\varphi?) = \mathrm{sub}(\varphi)$.

Subcomponents are the basic parts tableau rules operate on as they break down complex expressions. During this process, the rules often prefix them with nominals, associating parts of expressions with specific points in a model. The definition of quasicomponents below formalizes which combinations of nominals and subcomponents can arise during this process.

Definition 3.4. The set of quasi subcomponents of a node expression $i{:}\varphi$ is defined as

$$\begin{aligned}
\mathrm{qsub}(i{:}\varphi) = \{&j{:}\psi, j{:}\neg\psi \mid \psi \in \mathrm{sub}(i{:}\psi), j \in \mathsf{Nom}\}\\
&\cup \{\langle @_j\alpha \blacktriangle_{\mathsf{e}} @_k\beta\rangle, \langle @_j \blacktriangle_{\mathsf{e}} @_k\beta\rangle, \langle @_j\alpha \blacktriangle_{\mathsf{e}} @_k\rangle, \langle @_j \blacktriangle_{\mathsf{e}} @_k\rangle\\
&\qquad \mid \langle \alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in \mathrm{sub}(i{:}\varphi), j \in \mathsf{Nom}\}\\
&\cup \{\neg\langle @_j\alpha \blacktriangle_{\mathsf{e}} @_k\beta\rangle, \neg\langle @_j \blacktriangle_{\mathsf{e}} @_k\beta\rangle, \neg\langle @_j\alpha \blacktriangle_{\mathsf{e}} @_k\rangle, \neg\langle @_j \blacktriangle_{\mathsf{e}} @_k\rangle, \langle @_j \blacktriangledown_{\mathsf{e}} @_k\rangle\\
&\qquad \mid \neg\langle \alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in \mathrm{sub}(i{:}\varphi), j \in \mathsf{Nom}\}.
\end{aligned}$$

The cases where the quasi subcomponent are $j{:}\psi$, or $j{:}\neg\psi$ arise naturally from the basic tableau rules. The other cases, correspond to applications of commutativity rules and for handling data equalities. For instance, $\langle @_j\alpha \blacktriangle_{\mathsf{e}} @_k\beta\rangle$ and $\neg\langle @_j\alpha \blacktriangle_{\mathsf{e}} @_k\beta\rangle$ may arise as the result of applying one of the internalization rules in Fig. 4. The case $\langle @_j \blacktriangle_{\mathsf{e}} @_k\beta\rangle$ corresponds to a situation in which the left-hand side of a data (in)equality has been fully decomposed. Similarly, $\langle @_j \blacktriangle_{\mathsf{e}} @_k\rangle$ corresponds to a situation in which both sides of a data equality have been fully decomposed, and $\langle @_j \blacktriangledown_{\mathsf{e}} @_k\rangle$ may arise from applying $(\neg\blacktriangle_{\mathsf{e}})$.

Definition 3.5. A tableau for a node expression φ is *analytic* iff the label of any node in the tableau is one of the following:

(1) a quasi subcomponent of the root,
(2) an accessibility constraint,
(3) an equality constraint $\langle @_i =_{\mathsf{e}} @_j\rangle$ resulting from the rules (dRef) and (dTrans), or
(4) a satisfiability constraint $i{:}j$ resulting from the rules (ref), (sym), or (trans).

Lemma 3.6. *The tableaux in Def. 3.1 are analytic.*

Proof. Follows immediately from the tableaux rules. □

Def. 3.7 is auxiliary to seeing tableaux rules as a calculus for satisfiability.

Definition 3.7. A branch Θ of a tableau has a *clash* iff one of the following conditions hold:

(1) $\{i{:}\varphi, i{:}\neg\varphi\} \subseteq \Theta$,
(2) $\{\langle @_i =_{\mathsf{e}} @_j\rangle, \langle @_i \neq_{\mathsf{e}} @_j\rangle\} \subseteq \Theta$.

We say that Θ is *closed* if it contains a clash, otherwise it is *open.* The tableau is *closed* iff all its branches are closed, otherwise it is *open.* Finally, we say that Θ is *saturated* iff it is fully expanded according to the rules of the tableau system.

The notion of clash tells us that a contradiction has been found in a branch of the tableau. At the same time, the notion of an open and saturated branch tells us that such a branch is free of contradictions (i.e., contradictions are impossible in the branch). To check whether a node expression φ is satisfiable, we begin with a single node tableaux having $i{:}\varphi$ as its root and successively apply the rules in Figs. 2 to 5. We answer the question negatively if at some stage the tableau closes, and answer the question positively if we find an open and saturated branch. The correctness of Def. 3.7 as a calculus for satisfiability follows from the results in Sections 3.2 and 3.3. Moreover, the results in Section 4 tell us that this calculus yields a decision procedure for the satisfiability problem. Even further, the results in Section 5 tells us that the computational complexity of this calculus is relatively low, insofar as it is in the same class as the satisfiability problem for Basic Modal Logic [BWvB06].

3.2. **Soundness.** In Thm. 3.11 we present the proof of soundness for tableaux as a calculus for satisfiability. Our proof follows the same strategy used in [Fit83, Fit96, Pri08]. We start by introducing a well-known satisfiability preserving result regarding the use of new nominals.

Lemma 3.8. *Let φ be a node expression and let j be a nominal not in φ. Then, (1) φ is satisfiable iff $j{:}\varphi$ is satisfiable, (2) $\varphi = \psi \wedge i{:}\langle \mathsf{a}\rangle\chi$ is satisfiable iff $\varphi \wedge i{:}\langle \mathsf{a}\rangle j \wedge j{:}\chi$ is satisfiable, and (3) $\varphi = \psi \wedge \langle @_i \mathsf{a}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle$ is satisfiable iff $\varphi \wedge i{:}\langle \mathsf{a}\rangle j \wedge (\langle @_j \alpha \blacktriangle_{\mathsf{e}} \beta\rangle)$ is satisfiable.*

Proof. In all cases, the implication from right to left is direct. For (1), let $\mathfrak{M}, n \vDash \varphi$. Since j does not appear in φ, we can build a new model $\mathfrak{M}'$ that is exactly like $\mathfrak{M}$ with the exception that in $\mathfrak{M}'$ we set $\mathrm{nom}(j) = n$. Thus, $\mathfrak{M}', n \vDash j{:}\varphi$. For (2) and (3), similarly, if we choose j not in φ, we can use the same idea and set $\mathrm{nom}(j) = m$ for m such that $(\mathrm{nom}(i), m) \in R_{\mathsf{a}}$. □

Now we proceed to prove some properties relating the extension of a branch using tableaux rules, and the satisfiability of the node expressions contained in such a branch.

Definition 3.9. Let Θ be branch of a tableau. We say that Θ is *faithful* to a model $\mathfrak{M}$ iff every node expression in Θ is satisfiable in $\mathfrak{M}$. In turn, Θ is satisfiable iff there is a model $\mathfrak{M}$ such that Θ is faithful to $\mathfrak{M}$.

The following lemma makes clear the way in which tableau rules work: they only introduce a contradiction (a clash) if such a contradiction is already present in the node expression we are building the tableau for. Stated otherwise, tableau rules preserve satisfiability.

Lemma 3.10. *The application of any one of the tableau rules in Figs. 2 to 5 to a satisfiable branch Θ yields another satisfiable branch.*

Proof. Let Θ be a satisfiable branch of tableau, i.e., Θ is faithful to a model $\mathfrak{M}$. The proof is by a case analysis of tableau rules. We develop some relevant cases. For the rules ($\Diamond$) and (child), the result follows from Lemma 3.8. For the rule (copy), notice that we have $\{i{:}\varphi, i{:}j\} \subseteq \Theta$ and that the extension after the application of (copy) is $\Theta' = \Theta \cup \{j{:}\varphi\}$. Moreover, we have $\mathfrak{M} \vDash i{:}j$ and $\mathfrak{M} \vDash i{:}\varphi$. From the first conjunct, we obtain $\mathrm{nom}(i) = \mathrm{nom}(j)$, and from the second $\mathfrak{M}, \mathrm{nom}(i) \vDash \varphi$, then $\mathfrak{M} \vDash j{:}\varphi$. Therefore, Θ' is satisfiable. For the rule (dRef), we have $i{:}j \in \Theta$ and that the extension after the application of (dRef) is $\Theta' = \Theta \cup \{\langle @_i =_{\mathsf{e}} @_j\rangle\}$. Moreover, we have $\mathfrak{M} \vDash i{:}j$, and thus $\mathrm{nom}(i) = \mathrm{nom}(j)$. Since $\approx_{\mathsf{e}}$ is an equivalence relation, $(\mathrm{nom}(i), \mathrm{nom}(j)) \in \approx_{\mathsf{e}}$, then $\mathfrak{M} \vDash \langle @_i =_{\mathsf{e}} @_j\rangle$. Therefore, Θ' is satisfiable. Finally, notice that the cases (test), ($\neg$test), ($\cup$), ($\neg\cup$), (int_1), and (int_2) are immediate from Prop. 2.4. □

The result in Lemma 3.10 enables us to state and prove the main result of this section.

Theorem 3.11 (Soundness)**.** *Let φ be a node expression. If φ is satisfiable, then there is a saturated branch of a tableau for φ that is open.*

Proof. Aiming for a contradiction, suppose all saturated tableaux for φ are closed, but φ is satisfiable. Any tableau for φ starts with $i{:}\varphi$, for some $i \in \mathsf{Nom}$ not appearing in φ. From Lemma 3.8, we know $i{:}\varphi$ is also satisfiable, thus the initial branch $\Theta = \{i{:}\varphi\}$ of the tableaux is satisfiable. From Lemma 3.10, we know that the application of the rules in Figs. 2 to 5 always yields a satisfiable branch from Θ. Eventually, we would obtain a saturated branch. This contradicts the assumption that all saturated tableaux built from φ are closed. □

3.3. **Completeness.** Let us now turn our attention to proving the completeness of tableaux as a calculus for satisfiability. The main idea behind the proof is to look at an open and saturated branch of tableaux and to build a model from it (cf., e.g., [Fit83, Fit96, Pri08]). We begin with some preliminary definitions and results.

Definition 3.12. We use $\mathsf{Nom}(\Theta)$ for the set of nominals appearing in a branch Θ of a tableau. If Θ is saturated, for every $\{i, j\} \subseteq \mathrm{Nom}(\Theta)$ define $i \equiv_\Theta j$ iff $i{:}j \in \Theta$, and let $[i]_{\equiv_\Theta} = \{j \mid i \equiv_\Theta j\}$.

Intuitively, the relation $\equiv_\Theta$ groups together all the nominals in the branch Θ representing the same node in the model we want to build. The following lemma states $\equiv_\Theta$ is an equivalence relation as desired.

Lemma 3.13. *$\equiv_\Theta$ is an equivalence relation.*

Proof. Immediate from (ref), (sym) and (trans). □

Let us proceed to introduce key elements in our proof completeness: *nominal urfathers*.

Definition 3.14. Let Θ be a saturated tableau branch of a tableau. Define $\mathrm{uf}_\Theta(i) = \min([i]_{\equiv_\Theta})$; and we say that $\mathrm{uf}_\Theta(i)$ is the *urfather* of i. The set $\mathrm{uf}(\Theta) = \{\mathrm{uf}_\Theta(i) \mid i \in \mathsf{Nom}(\Theta)\}$ is the set of urfathers in the branch.

The following is an important property of nominal urfathers. Intuitively, as pointed out in [Bra11], this property tells us that non-trivial reasoning involving non-singleton equivalence classes is carried out only in connection with nominals appearing in the root node expression.

Lemma 3.15. *If Θ is a saturated branch of a tableau and $\mathrm{uf}_\Theta(i) \neq i$, $\mathrm{uf}_\Theta(i)$ is a root nominal.*

Proof. Let $\mathrm{uf}_\Theta(i) \neq i$ and $\mathrm{uf}_\Theta(i) = j$. The proof proceeds by cases.

(i is a root nominal): Recall that a root nominal is a nominal appearing in the root of our tableau. Any new nominal added by the application of a rule from Section 3.1 is greater than any root nominal. Since $j < i$ and i is a root nominal, j must be also a root nominal.

(i is not a root nominal): If $i{:}j$ is the root of the tableau, then, we are done. If $i{:}j$ is not the root of the tableau, then, it was introduced by a tableau expansion rule. It is clear that j is a root nominal if it was introduced by ($\neg\neg$), ($\wedge$), (nom), and (test). Notice that in any of these cases, j is a component of the node expression labelling the root of the tableau. Otherwise, $i{:}j$ is introduced into the tableau by (sym) or (trans). In these two last cases, it must be that there is a root nominal k such that $\{i{:}k, k{:}k\} \subset \Theta$. That j is a root nominal follows from this fact and the minimality of being an urfather. $\square$

From their definition, it follows that nominal urfathers can be taken as canonical representatives of the equivalence classes in $\mathsf{Nom}(\Theta)/\!\equiv \Theta$. This characteristic makes them key elements in our proof of completeness. Moreover, as made clear in Lemma 3.16, nominal urfathers collect the information about all members of the equivalence class.

Lemma 3.16. *If Θ is a saturated branch of a tableau and $i{:}\varphi \in \Theta$, then,* $\mathrm{uf}_\Theta(i){:}\varphi \in \Theta$.

Proof. Suppose that $i{:}\varphi \in \Theta$ and $\mathrm{uf}_\Theta(i) = j$. If $i = j$, then, we are done. If $i \neq j$, from Lemma 3.15, we know j is a root nominal. Moreover, from Def. 3.12, $i{:}j \in \Theta$, and, from Def. 3.14, we know $j < i$. Since Θ is saturated it is closed under (copy), $j{:}\varphi \in \Theta$. Thus, $\mathrm{uf}_\Theta(i) : \varphi \in \Theta$. $\square$

The results in Lemmas 3.15 and 3.16 pave the way to define the intended model from an open branch. Intuitively, they tell us that if our goal is to build a model that satisfies a node expression φ from an open and saturated branch of a tableau, then, we can restrict our attention to nominal urfathers. In comparison with what is done in [AFS17], the use of urfathers simplifies the definition of the model, and aid in the proof of Lemma 3.20.

Definition 3.17 (Extracted Model)**.** Let Θ be an open and saturated branch of a tableau. Define a structure $\mathfrak{M}^\Theta = \langle N^\Theta, \{R^\Theta_{\mathsf{a}}\}_{\mathsf{a}\in\mathsf{Rel}}, \{\approx^\Theta_{\mathsf{e}}\}_{\mathsf{e}\in\mathsf{Eq}}, V^\Theta, \mathrm{nom}^\Theta\rangle$ where

$$\begin{aligned}
N^\Theta &= \mathrm{uf}(\Theta)\\
R^\Theta_{\mathsf{a}} &= \{(\mathrm{uf}_\Theta(i), \mathrm{uf}_\Theta(j)) \mid i{:}\langle \mathsf{a}\rangle j \in \Theta\}\\
\approx^\Theta_{\mathsf{e}} &= \{(\mathrm{uf}_\Theta(i), \mathrm{uf}_\Theta(j)) \mid \langle @_i =_{\mathsf{e}} @_j\rangle \in \Theta\}\\
V^\Theta(p) &= \{\mathrm{uf}_\Theta(i) \mid i{:}p \in \Theta\}\\
\mathrm{nom}^\Theta(i) &= \begin{cases} \min(\mathrm{uf}(\Theta)) & \text{if } i \notin \mathrm{Nom}(\Theta)\\ \mathrm{uf}_\Theta(i) & \text{if } i \in \mathrm{Nom}(\Theta).\end{cases}
\end{aligned}$$

Proposition 3.18. *The structure $\mathfrak{M}^\Theta$ in Def. 3.17 is a model of* $\mathsf{HXPath_D}$.

Proof. The only challenge is proving $\approx^\Theta_{\mathsf{e}}$ to be an equivalence relation. Reflexivity follows from (ref) and (dRef), symmetry follows from (com$_1$), and transitivity follows from (dTrans). $\square$

We refer to $\mathfrak{M}^\Theta$ as the *extracted model* for Θ. This is the main ingredient in the proof of completeness of tableaux as a calculus for satisfiability. The proof of this result is carried out by induction on the *size* of a node expression. We introduce this definition of size below and comment on it immediately after.

Definition 3.19. The function size is defined by mutual recursion on node and path expressions.

$$\begin{aligned}
\mathrm{size}(i) &= 1 & \mathrm{size}(\mathsf{a}) &= 1\\
\mathrm{size}(p) &= 1 & \mathrm{size}(@_i) &= 1\\
\mathrm{size}(\varphi \wedge \psi) &= 1 + \mathrm{size}(\varphi) + \mathrm{size}(\psi) & \mathrm{size}(\varphi?) &= 1 + \mathrm{size}(\varphi)\\
\mathrm{size}(\neg\varphi) &= 1 + \mathrm{size}(\varphi) & \mathrm{size}(\alpha\beta) &= \mathrm{size}(\alpha) + \mathrm{size}(\beta)\\
\mathrm{size}(\langle \mathsf{a}\rangle\varphi) &= 1 + \mathrm{size}(\varphi) & \mathrm{size}(\alpha \cup \beta) &= 1 + \mathrm{size}(\alpha) + \mathrm{size}(\beta)\\
\mathrm{size}(i{:}\varphi) &= 3 + \mathrm{size}(\varphi)\\
\mathrm{size}(\langle \alpha \blacktriangle_{\mathsf{e}} \beta\rangle) &= 5 + \mathrm{size}(\alpha) + \mathrm{size}(\beta).
\end{aligned}$$

The definition of size is necessary because we cannot rely on structural induction over node expressions. This is due to the fact that rule applications may generate expressions not structurally contained in the original input. For example, applying $(\neg\text{int})$ to $i{:}\neg\langle\alpha \blacktriangle_{\mathsf{e}} \beta\rangle$ introduces $\neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta\rangle$ in the branch, which is not a part of the original expression. To handle this, we define a size measure on node and path expressions and carry out induction on that measure instead. The constants 3 for $i{:}\varphi$ and 5 for $\langle\alpha \blacktriangle_{\mathsf{e}} \beta\rangle$ are chosen to ensure that rule applications strictly reduce size. This guarantees that the induction is well-founded. For instance, in the case of $(\neg\text{int})$, we have $\text{size}(\neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta\rangle) < \text{size}(i{:}\neg\langle\alpha \blacktriangle_{\mathsf{e}} \beta\rangle)$.

Lemma 3.20. *Let Θ be an open and saturated branch, and $\mathfrak{M}^\Theta$ be the model extracted from it. In addition, let $\mathfrak{M}^\Theta \models \varphi$ indicate $\mathfrak{M}^\Theta, n \models \varphi$ for all N^Θ. Then, $\varphi \in \Theta$ implies $\mathfrak{M}^\Theta \models \varphi$.*

Proof. Observe that every $\varphi \in \Theta$ has one of the following forms:

$$(1)\ i{:}\psi \qquad (2)\ \langle @_i\alpha \blacktriangle_{\mathsf{e}} @_j\beta\rangle \qquad (3)\ \neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_j\beta\rangle$$

The proof is by induction on the size of $\varphi \in \Theta$. We focus on some of the cases in the proof. For the base cases we consider**:**

($\varphi = i{:}j$**):** We need to prove $\mathfrak{M}^\Theta \models i{:}j$, or equivalently, $\text{uf}_\Theta(i) = \text{uf}_\Theta(j)$. The result is immediate from the definition of nom^Θ.

($\varphi = i{:}\neg j$**):** We need to prove $\mathfrak{M}^\Theta \models i{:}\neg j$, or equivalently $\text{uf}_\Theta(i) \neq \text{uf}_\Theta(j)$. The proof is by contradiction. Suppose $\text{uf}_\Theta(i) = \text{uf}_\Theta(j)$. By Defs. 3.12 and 3.14, $i{:}j \in \Theta$. This implies that Θ contains a clash, contradicting that Θ is open. Thus, $\mathfrak{M}^\Theta \models i{:}\neg j$.

($\varphi = \langle @_i =_{\mathsf{e}} @_j\rangle$**):** We neet to prove $\mathfrak{M}^\Theta \models \langle @_i =_{\mathsf{e}} @_j\rangle$, or equivalently that the pair $(\text{uf}_\Theta(i), \text{uf}_\Theta(j)) \in \approx^\Theta_{\mathsf{e}}$. The result is immediate from the definition of $\approx^\Theta_{\mathsf{e}}$.

($\varphi = \langle @_i \neq_{\mathsf{e}} @_j\rangle$**):** We need to prove $\mathfrak{M}^\Theta \models \langle @_i \neq_{\mathsf{e}} @_j\rangle$, or equivalently that the pair $(\text{uf}_\Theta(i), \text{uf}_\Theta(j)) \notin \approx^\Theta_{\mathsf{e}}$. The proof is by contradiction. Suppose that the pair $(\text{uf}_\Theta(i), \text{uf}_\Theta(j)) \in \approx^\Theta_{\mathsf{e}}$. By definition of $\approx^\Theta_{\mathsf{e}}$, $\langle @_i =_{\mathsf{e}} @_j\rangle \in \Theta$. But this means the branch contains a clash, contradicting that Θ is open. Thus, $\mathfrak{M}^\Theta \models \langle @_i \neq_{\mathsf{e}} @_j\rangle$.

The inductive hypothesis (IH) is**:** for all $\psi \in \Theta$ such that $\text{size}(\psi) < \text{size}(\varphi)$, $\mathfrak{M}^\Theta \models \psi$.
For the inductive cases we consider**:**

($\varphi = i{:}\neg\neg\psi$**):** We need to prove $\mathfrak{M}^\Theta \models i{:}\neg\neg\psi$, or equivalently $\mathfrak{M}^\Theta \models i{:}\psi$. From $(\neg\neg)$, $i{:}\psi \in \Theta$. By the IH, $\mathfrak{M}^\Theta \models i{:}\psi$.

($\varphi = i{:}\neg(\psi \wedge \chi)$**):** We need to prove $\mathfrak{M}^\Theta \models i{:}\neg(\psi \wedge \chi)$, or equivalently $\mathfrak{M}^\Theta \models i{:}\neg\psi$ or $\mathfrak{M}^\Theta \models i{:}\neg\chi$. From $(\neg\wedge)$, $i{:}\neg\psi \in \Theta$ or $i{:}\neg\chi \in \Theta$. W.l.o.g., let $i{:}\neg\psi \in \Theta$. By the IH, $\mathfrak{M}^\Theta \models i{:}\neg\psi$.

($\varphi = i{:}\neg j{:}\psi$**):** We need to prove $\mathfrak{M}^\Theta \models i{:}\neg(j{:}\psi)$, or equivalently $\mathfrak{M}^\Theta \models j{:}\neg\psi$. From $(\neg\text{nom})$, $j{:}\neg\psi \in \Theta$. Then, by the IH, $\mathfrak{M}^\Theta \models j{:}\neg\psi$.

($\varphi = i{:}\neg\langle\mathsf{a}\rangle\psi$**):** We must prove $\mathfrak{M}^\Theta \models i{:}\neg\langle\mathsf{a}\rangle\psi$, or equivalently, for all $(\text{uf}_\Theta(i), n) \in R^\Theta_{\mathsf{a}}$, $\mathfrak{M}^\Theta, n \models \neg\psi$. Let $(\text{uf}_\Theta(i), n) \in R^\Theta_{\mathsf{a}}$. The definition of R^Θ_{a} tells us that exists $i{:}\langle\mathsf{a}\rangle j \in \Theta$ s.t. $\text{uf}_\Theta(j) = n$. Since $\{i{:}\neg\langle\mathsf{a}\rangle\psi, i{:}\langle\mathsf{a}\rangle j\} \subseteq \Theta$, using $(\neg\Diamond)$, we get $j{:}\neg\psi \in \Theta$. By IH, $\mathfrak{M}^\Theta \models j{:}\neg\psi$, i.e., $\mathfrak{M}^\Theta, n \models \neg\psi$.

($\varphi = i{:}\neg\langle\alpha \blacktriangle_{\mathsf{e}} \beta\rangle$**):** We neet to prove $\mathfrak{M}^\Theta \models i{:}\neg\langle\alpha \blacktriangle_{\mathsf{e}} \beta\rangle$. Equivalently, we prove $\mathfrak{M}^\Theta \models \neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta\rangle$. From (int_2), $\neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta\rangle \in \Theta$. Then, we have $\text{size}(\neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta\rangle) < \text{size}(i{:}\neg\langle\alpha \blacktriangle_{\mathsf{e}} \beta\rangle)$. This enables us to apply the IH and get $\mathfrak{M}^\Theta \models \neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_i\beta\rangle$.

($\varphi = i{:}\langle\mathsf{a}\rangle\psi$**):** We need to prove $\mathfrak{M}^\Theta \models i{:}\langle\mathsf{a}\rangle\psi$. From $(\Diamond)$, $\{i{:}\langle\mathsf{a}\rangle j, j{:}\psi\} \subseteq \Theta$. Using the IH, we can prove $\mathfrak{M}^\Theta \models i{:}\langle\mathsf{a}\rangle j \wedge j{:}\psi$. Lemma 3.8 gives us the desired result.

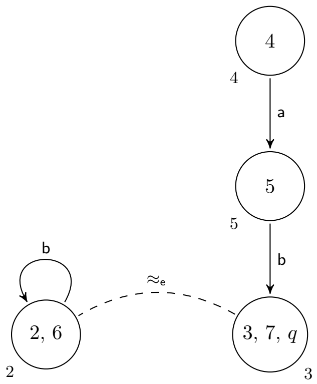

FIGURE 7. Extracted model for $\langle \mathsf{a}\rangle\langle @_2\mathsf{b}2? =_{\mathsf{e}} \mathsf{b}(q \wedge 3)?\rangle$

$(\varphi = \langle @_i(\alpha \cup \beta)\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle)$: We need to prove that $\mathfrak{M}^\Theta \vDash \langle @_i(\alpha\cup\beta)\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle$, equivalently $\mathfrak{M}^\Theta \vDash \langle @_i\alpha\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle$ or $\mathfrak{M}^\Theta \vDash \langle @_i\beta\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle$. From $(\cup)$, $\langle @_i\alpha\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle \in \Theta$ or $\langle @_i\beta\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle \in \Theta$. W.l.o.g. let $\langle @_i\alpha\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle \in \Theta$, by the IH, $\mathfrak{M}^\Theta \vDash \langle @_i\alpha\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle$.

$(\varphi = \neg\langle @_i(\alpha \cup \beta)\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle)$: We need to prove $\mathfrak{M}^\Theta \vDash \neg\langle @_i(\alpha \cup \beta)\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle$, equivalently $\mathfrak{M}^\Theta \vDash \neg\langle @_i\alpha\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle$ and $\mathfrak{M}^\Theta \vDash \neg\langle @_i\beta\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle$. From $(\neg\cup)$, $\neg\langle @_i\alpha\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle \in \Theta$ and $\neg\langle @_i\beta\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle \in \Theta$. By the IH, $\mathfrak{M}^\Theta \vDash \neg\langle @_i\alpha\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle$ and $\mathfrak{M}^\Theta \vDash \langle @_i\beta\gamma \blacktriangle_{\mathsf{e}}\ \eta\rangle$.

□

The lemma above is the fundamental piece to state the intended completeness result.

Theorem 3.21 (Completeness)**.** *Let φ be a node expression. If there is a tableau for φ with an open and saturated branch, then, φ is satisfiable.*

Proof. Suppose that there is a tableau for φ with an open and saturated branch Θ. Let $\mathfrak{M}^\Theta$ be the extracted model for Θ. From Lemma 3.20, we know $\mathfrak{M}^\Theta \vDash i{:}\varphi$ where $i{:}\varphi$ is the root of the tableau. Immediately, $\mathfrak{M}^\Theta, \mathrm{nom}(i) \vDash \varphi$. Thus, φ is satisfiable. □

As it is usual, our tableaux-based calculus not only provides a way to check the satisfiability of a node expression, but also acts as a *model building procedure*, since for every satisfiable node expression it is possible to obtain a model (Def. 3.17). We finish this section by showing how the extracted model is built for a given node expression.

Example 3.22. Consider the tableau for $\varphi = \langle \mathsf{a}\rangle\langle @_2\mathsf{b}2? =_{\mathsf{e}} \mathsf{b}(q \wedge 3)?\rangle$ in Ex. 3.2. This tableau has a sole open and saturated branch Θ. Thus, we can conclude φ is satisfiable. We depict the extracted model from Θ in Fig. 7. In this figure, nodes are labelled by the nominals and propositional symbols satisfied by them. We omit the reflexive edges for $\approx_{\mathsf{e}}$. Notice that in the extracted model nodes correspond to urfathers; in this respect, we have $\mathrm{uf}_\Theta(7) = 3$, $\mathrm{uf}_\Theta(6) = 2$, and for any other nominal k, $\mathrm{uf}_\Theta(k) = k$.

Up to now we focused on node expressions. Handling satisfiability of path expressions is simple, given the following proposition.

Proposition 3.23. *Let α be a path expression of $\mathsf{HXPath}_{\mathsf{D}}$, then α is satisfiable if and only if $i{:}\langle\alpha\rangle\top$ is satisfiable, for i a nominal not in α.*

Proof. The implication from right to left is obvious. For the other direction, any model satisfying α can be turn into a model of $i{:}\langle\alpha\rangle\top$ by setting $\mathrm{nom}(i)$ to the starting point of the path satisfying α. □

4. Termination

In the previous section we show how tableaux yield a satisfiability calculus. In this section, we prove that we can restrict these tableaux to be finite, without loosing completeness. Such a result establishes that the satisfiability problem for $\mathsf{HXPath_D}$ is decidable.

We begin by observing that the analyticity of the tableau ensures that all expressions introduced during expansion are constructed from subcomponents of the root, reducing the search space. The main technical challenge lies in managing the introduction of new nominals and new accessibility constraints through applications of the ($\Diamond$) and (child) rules. To resolve this, we adapt the termination argument for Hybrid Logic tableaux in [BB07]. In particular, we impose a controlled strategy for applying the ($\Diamond$) and (child) rules to ensure that the resulting structure remains tree-like.

Definition 4.1. Let Θ be a branch of a tableau with root $i{:}\varphi$. For $j \in \mathsf{Nom}(\Theta)$, define:

$$\mathrm{at}^{\Theta}_{\mathrm{N}}(j) = \{\psi \mid j{:}\psi \in (\Theta \cap \mathrm{qsub}(i{:}\varphi))\}$$
$$\mathrm{at}^{\Theta}_{\Pi}(j) = \{\alpha \mid \pm \langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in (\Theta \cap \mathrm{qsub}(i{:}\varphi))\}.$$

where $\pm$ indicates that $\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle$ may or may not be preceded by a negation. In addition, define $\mathrm{at}^{\Theta}(j) = \mathrm{at}^{\Theta}_{\mathrm{N}}(j) \cup \mathrm{at}^{\Theta}_{\Pi}(j)$.

Intuitively, the set $\mathrm{at}^{\Theta}_{\mathrm{N}}(j)$ captures the subcomponents of the root of the tableau true at j in the branch. In turn, the set $\mathrm{at}^{\Theta}_{\Pi}(j)$ captures the subpaths present in the data (in)equalities in the root of the tableau that start at j in the branch.

Example 4.2. Let Θ be the sole branch in the tableau in Ex. 3.2. Then

$$\mathrm{at}^{\Theta}(5) = \mathrm{at}^{\Theta}_{\mathrm{N}}(5) \cup \mathrm{at}^{\Theta}_{\Pi}(5) = \{\langle @_2\mathsf{b}2? =_{\mathsf{e}} \mathsf{b}(q \wedge 3)?\rangle\} \cup \{@_2\mathsf{b}2?, \mathsf{b}(q \wedge 3)?\}.$$

Notice how the set $\mathrm{at}^{\Theta}(5)$ gives us a hint of the rules which can be applied to the node expressions having 5 as a prefix. We retake this idea later on.

The set $\mathrm{qsub}(i{:}\varphi)$ of quasi subcomponents of the label of the root of a tableau is infinite, since its considers all possible ways of prefixing it with nominals. However, we can establish that in fact at^{Θ} only considers those quasicomponents that are made of either a subcomponent or a negated subcomponent, i.e., it is restricted to nominals appearing in a branch of the tableau. This is stated in the following result.

Proposition 4.3. *For every branch Θ, the set $\mathrm{at}^{\Theta}(j)$ is finite.*

Proof. Let Θ be a branch of a tableau with root $i{:}\varphi$. From Lemma 3.6, we get:

$$\mathrm{at}^{\Theta}_{\mathrm{N}}(j) \subseteq \{\psi, \neg\psi \mid \psi \in \mathrm{sub}(i{:}\varphi)\}$$
$$\mathrm{at}^{\Theta}_{\Pi}(j) \subseteq \{\alpha \mid \pm\langle \alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in \mathrm{sub}(i{:}\varphi)\}.$$

Since the set $\mathrm{sub}(i{:}\varphi)$ is finite, we immediately get that the set $\mathrm{at}^{\Theta}(j)$ is finite. □

Prop. 4.3 tells us that the set of expressions that are true at a given node must be finite. Still, there is no guarantee that every tableau must be finite. It could be the case that the tableau is infinitely branching, or that it has a branch of infinite size. We prove that neither is the case for our tableaux.

Definition 4.4. Let Θ be a branch of a tableau, and $\{i,j\} \subseteq \mathsf{Nom}(\Theta)$. We say that j *is generated by* i, and write $i \prec_\Theta j$, iff there is $\mathsf{a} \in \mathsf{Rel}$ such that $i{:}\langle\mathsf{a}\rangle j \in \Theta$ is an accessibility constraint. Finally, we say that i is an *ancestor* of j, and write $i \prec_\Theta^+ j$, iff (i,j) is in the transitive closure of $\prec_\Theta$.

The following lemma tells us that every nominal in the branch of a tableau generates a finite number of new nominals. More precisely, this result tells us that applications of $(\Diamond)$ and (child) result in a finite set of trees, each of which has a finite number of branches.

Lemma 4.5. *Let Θ be a branch of a tableau. The graph $G_\Theta = (\mathsf{Nom}(\Theta), \prec_\Theta)$ is a finite set of finitely branching trees.*

Proof. That G_Θ is a finite set of trees follows from the following three facts. First, for every $\{i,j,k\} \subseteq \mathsf{Nom}(\Theta)$, if $j \prec_\Theta i$ and $k \prec_\Theta i$, then, $j = k$, i.e., if a nominal is generated by another nominal, then it is generated by at most one other nominal. Second, the set of ancestors of a nominal $i \in \mathsf{Nom}(\Theta)$ that is generated by another nominal has a least element. Third, for any $i \in \mathsf{Nom}(\Theta)$, the set $\mathrm{at}^\Theta(i)$ is finite (see Prop. 4.3), this implies that there is at most $\mathrm{size}(\mathrm{at}^\Theta(i))$ new nominals generated from i. □

Now that we know that applications of $(\Diamond)$ and (child) always result in a finite set of finitely branching trees, we can move on to show that every branch in such trees is of finite size. Such a result establishes that tableaux are finite.

Lemma 4.6. *A branch of a tableau is infinite iff there is an infinite chain $i \prec_\Theta j \prec_\Theta \dots$.*

Proof. The right-to-left direction is trivial. To prove the left-to-right direction, let Θ be an infinite branch. From Prop. 4.3, the branch Θ must contain infinitely many distinct nominals. This implies that $\mathsf{Nom}(\Theta)$ in Lemma 4.5 must be infinite. Since, from Lemma 4.5, we know that G_Θ is a finite set of finitely branching trees, it must be that some tree in G_Θ contains an infinite chain $i \prec_\Theta j \prec_\Theta \dots$. □

Def. 4.7 introduces the last piece needed to prove termination.

Definition 4.7. Let Θ be a branch of a tableau, and let $i \in \mathsf{Nom}(\Theta)$. Define

$$\begin{aligned}
\mathrm{m}^\Theta_{\mathrm{N}}(i) &= \max\{\mathrm{size}(\varphi) \mid \varphi \in \mathrm{at}^\Theta_{\mathrm{N}}(i)\}\\
\mathrm{m}^\Theta_{\Pi}(i) &= \begin{cases} \max\{\mathrm{size}(\alpha) \mid \alpha \in \mathrm{at}^\Theta_{\Pi}(i)\} & \text{if } \mathrm{at}^\Theta_{\Pi}(i) \neq \emptyset \\ 0 & \text{otherwise} \end{cases}\\
\mathrm{m}^\Theta(i) &= \max(\mathrm{m}^\Theta_{\mathrm{N}}(i), \mathrm{m}^\Theta_{\Pi}(i)).
\end{aligned}$$

Intuitively, $\mathrm{m}^\Theta(i)$ indicates the size of the largest node expression true at i in Θ, or the length of the longest path expression starting at i. The idea is to prove that such a measure of size decreases after the application of the tableaux rules.

Lemma 4.8. *Let Θ be a saturated branch. If $i \prec_\Theta j$, then $\mathrm{m}^\Theta(i) > \mathrm{m}^\Theta(j)$.*

Proof. Let $i \prec_\Theta j$. We have three possible cases: (1) $\mathrm{m}^\Theta(j) = 0$, (2) $\mathrm{m}^\Theta(j) = \mathrm{size}(\psi)$ for some node expression ψ, and (3) $\mathrm{m}^\Theta(j) = \mathrm{size}(\alpha)$ for some path expression α. The latter has two possible sub-cases: (a) $\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in \Theta$ and $\neg\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle \notin \Theta$, and (b) $\neg\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in \Theta$ and $\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle \notin \Theta$. The result is immediate for (1). We prove the remaining cases below.

(Case 2): Let $\mathrm{m}^{\Theta}(j) = \mathrm{size}(\psi)$ for some node expression ψ. We need to prove $\mathrm{m}^{\Theta}(i) > \mathrm{size}(\psi)$. It is clear that $j{:}\psi$ can only result from ($\Diamond$), ($\neg\Diamond$), (ref), (sym), or (trans). If $j{:}\psi$ results from ($\Diamond$), using Lemma 4.5, we know $i{:}\langle \mathsf{a}\rangle\psi \in \Theta$. This implies, $\mathrm{m}^{\Theta}(i) \geq \mathrm{size}(\langle \mathsf{a}\rangle\psi) > \mathrm{size}(\psi) = \mathrm{m}^{\Theta}(j)$ as needed. In turn, if $j{:}\psi$ results from ($\neg\Diamond$), we know $\psi = \neg\psi'$. Using Lemma 4.5 again, we know $\{i{:}\neg\langle \mathsf{a}\rangle\psi', i{:}\langle \mathsf{a}\rangle j\} \subseteq \Theta$. This implies, $\mathrm{m}^{\Theta}(i) \geq \mathrm{size}(\neg\langle \mathsf{a}\rangle\psi') > \mathrm{size}(\psi) = \mathrm{m}^{\Theta}(j)$ as needed. Lastly, if $j{:}\psi$ results from (ref), (sym), or (trans), we have $\mathrm{size}(\psi) = 1$. At the same time, from $i \prec_{\Theta} j$, we have $\{i{:}\langle \mathsf{a}\rangle\chi, \langle @_i \mathsf{a}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle\} \cap \Theta \neq \emptyset$. Immediately, $\mathrm{m}^{\Theta}(i) > 1 = \mathrm{size}(\psi)$ as needed.

(Case 3.a): Let $\mathrm{m}^{\Theta}(j) = \mathrm{size}(\alpha)$, $\langle @_j \alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in \Theta$, and $\neg\langle @_j \alpha \blacktriangle_{\mathsf{e}} \beta\rangle \notin \Theta$. We need to prove $\mathrm{m}^{\Theta}(i) > \mathrm{size}(\alpha)$. It is clear that $\langle @_j \alpha \blacktriangle_{\mathsf{e}} \beta\rangle$ can only result from (child). Using Lemma 4.5, we know that $\langle @_i \mathsf{a}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in \Theta$. Then, $\mathrm{m}^{\Theta}(i) \geq \mathrm{size}(\langle @_i \mathsf{a}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle) > \mathrm{size}(\langle \alpha \blacktriangle_{\mathsf{e}} \beta\rangle) = \mathrm{m}^{\Theta}(j)$ as needed.

(Case 3.b): Let $\mathrm{m}^{\Theta}(j) = \mathrm{size}(\alpha)$, $\neg\langle @_j \alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in \Theta$, and $\langle @_j \alpha \blacktriangle_{\mathsf{e}} \beta\rangle \notin \Theta$. We need to prove $\mathrm{m}^{\Theta}(i) > \mathrm{size}(\alpha)$. It is clear that $\neg\langle @_j \alpha \blacktriangle_{\mathsf{e}} \beta\rangle$ can only result from ($\neg$child). Using Lemma 4.5, we know that $\neg\langle @_i \mathsf{a}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle \in \Theta$. Then, $\mathrm{m}^{\Theta}(i) \geq \mathrm{size}(\neg\langle @_i \mathsf{a}\alpha \blacktriangle_{\mathsf{e}} \beta\rangle) > \mathrm{size}(\neg\langle \alpha \blacktriangle_{\mathsf{e}} \beta\rangle) = \mathrm{m}^{\Theta}(j)$ as needed.

□

Lemma 4.9. *Let φ be a node expression. Any tableau for φ is finite.*

Proof. The proof is by contradiction. Suppose there is an infinite tableau for φ. Then it must be that this tableau has an infinite branch Θ. By Lemma 4.6, there exists an infinite chain $i \prec_{\Theta} j \prec_{\Theta} \ldots$ At the same time, by Lemma 4.8, we have $\mathrm{m}^{\Theta}(i) > \mathrm{m}^{\Theta}(j) > \ldots$ This yields a contradiction; since for any nominal i, it its trivial that $\mathrm{m}^{\Theta}(i) \geq 0$. □

Combining the result of Lemma 4.9 with the completeness of our tableau-based satisfiability procedure, we obtain the main result of this section.

Theorem 4.10. *The satisfiability problem for* $\mathsf{HXPath}_{\mathsf{D}}$ *is decidable.*

In Section 5 we refine the result in Thm. 4.10 and characterize the exact complexity of the satisfiability problem for $\mathsf{HXPath}_{\mathsf{D}}$.

5. Complexity of the Satisfiability Problem for $\mathsf{HXPath}_{\mathsf{D}}$

In this section, we prove that the satisfiability problem for $\mathsf{HXPath}_{\mathsf{D}}$ is PSpace-complete. PSpace-hardness follows from PSpace-hardness of the Basic Modal Logic [BdRV01, BvB06], a proper fragment of $\mathsf{HXPath}_{\mathsf{D}}$. For completeness, we present an algorithm that applies the tableau rules in a controlled manner, ensuring that satisfiability checking can be performed in polynomial space. Our algorithm explores the model induced by the tableau in a depth-first fashion. This requires particular care, as this model is not necessarily tree-shaped and may contain cycles or shared substructures. Furthermore, we make the necessary mechanisms explicit, making our algorithm suitable for implementation, a contribution by its own sake.

We begin by introducing some preliminary concepts. First, recall that without loss of generality, we assume that Nom is the set of natural numbers. In what follows we use [ ] for the empty list, [e] for a list with a single element e, $\mathrm{L}_1 + \mathrm{L}_2$ for the concatenation of lists L_1 and L_2, and $|\mathrm{L}|$ for the length of a list L. Moreover, we use $\mathrm{L} = [\mathrm{e}] + \mathrm{L}'$ to indicate a non-empty list L with first element e and tail L', and L_{n} to indicate L's initial segment of

Algorithm 1: Sat

```
def Sat($\varphi$ : node expression):
    require: true
    n ← a new nominal;
    $\Phi \leftarrow [\mathrm{n}{:}\varphi]$;
    return Sat($\Phi, \mathrm{n}+1$);
    ensure: Sat($\varphi$) iff $\varphi$ is satisfiable
```

length n. Finally, we assume that parameter variables are pass-by-reference and assignments are copy constructors.

The algorithmic application of tableau rules starts by calling the function Sat in Alg. 1 with a node expression φ. In this function, we set up the context for the application of tableau rules. The actual application of rules is implemented in the overloaded Sat in Alg. 2.

To keep the space complexity polynomial in Alg. 2, we explore branches one at a time. The information needed to explore branches is kept in local lists B and Σ. Intuitively, B stores the current branch of the tableau. In turn, Σ behaves as a stack storing pairs (b, χ) where b is a number indicating the position to backtrack in B to explore an alternative branch, and χ is the alternative node expression to be explored.

Turning to branch expansions, we classify tableau rules into types 1, 2, and 3. The rules of type 1 are: the rules $(\neg\neg)$, $(\wedge)$, and $(\neg\Diamond)$ in Fig. 2; all the rules in Figs. 3 and 4; the rules (test), (@), ($\neg$child), ($\neg$@), $(\neg\cup)$, and (com$_1$), (com$_2$), and (dTrans) in Fig. 5; and the rules

$$\frac{\begin{array}{c} k{:}\langle \mathsf{a}\rangle i \\ i{:}j \end{array}}{k{:}\langle \mathsf{a}\rangle j}\ (\mathrm{copy}_0) \qquad \frac{\begin{array}{c} \langle @_i\alpha \blacktriangle_{\mathsf{e}} \beta\rangle \\ i{:}j \end{array}}{\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}\ (\mathrm{copy}_1) \qquad \frac{\begin{array}{c} \neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} \beta\rangle \\ i{:}j \end{array}}{\neg\langle @_j\alpha \blacktriangle_{\mathsf{e}} \beta\rangle}\ (\mathrm{copy}_2)$$

These rules have side conditions $j < i$ and j a root nominal. In brief, these rules may be understood in analogy with (copy) in Fig. 3. These rules are admissible and do not affect termination. They are introduced here only to cope with space complexity. The rules of type 2 are: the rule $(\neg\wedge)$ in Fig. 2, and the rules ($\neg$test), and $(\cup)$ in Fig. 5. Finally, the rules of type 3 are: the rule $(\Diamond)$ in Fig. 2, and the rule (child) in Fig. 5. The rationale behind the classification of tableau rules is that rules of type 1 extend the branch linearly, rules of type 2 split the branch into two, and rules of type 3 handle diamonds.

We are now in a position to explain how Alg. 2 applies tableau rules to the current branch, i.e., to the list B. It is clear that the application of a rule should expand B by adding node expressions at its end. This expansion is actually dealt with in two blocks of code:

(expand): Lines 9 to 18. This block of code applies rules of type 1 or of type 2 until there are no untreated rules. The case of a rule R of type 1 is straightforward. Let Ψ be the set of node expressions in the consequent of R. We say that R is untreated iff $\Psi \nsubseteq \mathrm{B}$. If R is untreated, we add all the node expressions in Ψ at the end of B. In turn, let R be a rule of type 2 and let $\Psi = \{\psi, \chi\}$ be the node expressions in the consequent of R. We say that R is untreated iff $\Psi \cap \mathrm{B} = \emptyset$. If R is untreated, we add ψ at the end of B, and store in Σ the alternative χ together with the point in B at which ψ is introduced.

(explore): Lines 23 to 33. This block deals with the exploration of "diamonds", i.e., node expressions of the form $i{:}\langle \mathsf{a}\rangle\psi$ that are not accesibility constraints, or node expressions

Algorithm 2: SAT

```
def SAT($\Phi$ : list of node expressions, n : nominal):
    require: the nominals in $\Phi$ are in $[0 \dots n)$
    if $\Phi = [\,]$ then
        return true
    $B, \Sigma \leftarrow \Phi, [\,]$;
    repeat
        if $\Sigma = [(b, \chi)] + L$ then                         ▷ explore an alternative branch ◁
            $B, \Sigma \leftarrow B_b + [\chi], L$;
        expand:                          ▷ close B under the application of rules of type 1 ◁
            if there is a rule R of type 1 that is untreated in B then
                let $\Psi$ be the consequent of R
                $B \leftarrow B + \Psi$;                                      ▷ expand the branch ◁
                goto expand
            if there is a rule R of type 2 that is untreated in B then
                let $\{\psi, \chi\}$ be the consequent of R
                $B \leftarrow B + [\psi]$;                                    ▷ expand the branch ◁
                $\Sigma \leftarrow [(|B|, \chi)] + \Sigma$;   ▷ store the alternative and its location in the branch ◁
                goto expand
        if there is a clash in B and $\Sigma \neq [\,]$ then               ▷ try an alternative branch ◁
            continue;
        else if there is a clash in B then                        ▷ no alternative branches to try ◁
            return false;
        explore:                                            ▷ explore diamonds in a DFS manner ◁
            $r \leftarrow$ true;
            forall rules R of type 3 applicable in B do
                let $\psi$ be the premiss of R and $\psi \in \{i{:}\langle a \rangle \chi, \langle @_i a\alpha \blacktriangle_e \beta \rangle\}$;
                $\Phi \leftarrow \Phi$ + B minus all diamonds in B + $[i{:}\langle a \rangle n]$;
                if $\psi = i{:}\langle a \rangle \chi$ then
                    $\Phi \leftarrow \Phi + [n{:}\chi]$
                if $\psi = \langle @_i a\alpha \blacktriangle_e \beta \rangle$ then
                    $\Phi \leftarrow \Phi + [\langle @_n \alpha \blacktriangle_e \beta \rangle]$
                $r \leftarrow r$ and SAT($\Phi, n + 1$);
                $\Phi \leftarrow$ list of root literals and rooted boxes in $\Phi$;
        if r then return true;                                 ▷ the branch is open and saturated ◁
    until $\Sigma = [\,]$;
    return false;
    ensure: SAT($[n{:}\varphi], n + 1$) iff $\varphi$ is satisfiable
```

of the form $\langle @_i a\alpha \blacktriangle_e \beta \rangle$. To do so, we take some ideas from [Lad77, Mar06] of exploring diamonds in a depth first search (DFS) manner and adapt them to our setting.

First, in our case, the exploration of diamonds corresponds to the application of rules of type 3. A rule R of type 3 is applicable iff its premiss belongs to B. If R is applicable, its application involves: creating a new nominal n, creating the accesibility constraint $i{:}\langle \mathsf{a}\rangle \mathrm{n}$, and adding the node expression in the consequent of R to the branch.

Second, exploring diamonds in a DFS manner involves recursive calls to SAT. These recursive calls take into account all alternative explorations of diamonds and put a temporary pause on them. Notice that only one diamond must be treated at each step, as they can be individually satisfied and only can find contradictions with box expressions. Once the satisfiability of one particular diamond is checked, the algorithm backtracks and all diamonds are restored on B. This is implemented by removing all alternative diamonds from the branch in line 27 of Alg. 2, before the recursive call.

Lastly, before backtracking in the DFS strategy we may need to gather *global* information to be passed along to alternative calls to SAT. This global information is mediated by root nominals in a tableau branch and corresponds to *root literals* and *rooted boxes*. By a literal, we mean a node expression of the form $i{:}j$, $i{:}\neg j$, $i{:}p$, $i{:}\neg p$, $i{:}\langle \mathsf{a}\rangle j$, $\langle @_i \blacktriangle_{\mathsf{e}} @_j\rangle$ where $\{i, j\} \subseteq \mathsf{Nom}$ and $p \in \mathsf{Prop}$. A root literal is one whose nominals are root nominals. Root literals contain the relevant information stored in urfathers of root nominals, thus they need to be passed to the next recursive call. A rooted box is a node expression of the form $i{:}\neg\langle \mathsf{a}\rangle\psi$ or $\neg\langle @_i\alpha \blacktriangle_{\mathsf{e}} @_j\beta\rangle$ where i and j are root nominals. Global information to the branch is recorded in line 33 of Alg. 2.

Correctness of Alg. 2 follows from the correctness of tableau rules, while termination is ensured since each loop and recursive call is bounded. The next result shows the algorithm uses only polynomial space.

Theorem 5.1. *The satisfiability problem for* $\mathsf{HXPath_D}$ *is* PSPACE*-complete.*

Proof. Hardness follows from the fact that the Basic Modal Logic K is PSPACE-hard [BdRV01, BvB06], and $\mathsf{HXPath_D}$ subsumes it. Completeness follows from the fact that Alg. 2 uses at most polynomial space with respect to the size of the input node expression, as analyzed in what follows.

The local variables B and Σ occupy at most a polynomial amount of memory (bounded by the number of untreated rules of type 1 and 2). The number of recursive calls, in a branch, is also polynomial (bounded by the number of nested diamonds, including path expressions of the form a). The only information that is global to the branch is that on the list Φ. But this list is also of polynomial size, as the only persistent information that is passed along between different recursive calls refers only to root literals, and this information is polynomial.

When put together, these facts tell us that Alg. 2 uses at most a polynomial amount of memory with respect to the size of the input node expression. □

We close this section with a run of the function SAT. This run illustrates the inner workings of the algorithm it implements. Moreover, it allows us to recognize how it uses only polynomial space.

Example 5.2. Let us suppose that we start SAT in Alg. 1 with the node expression

$$\varphi = \langle \mathsf{a}(1 \wedge p)? =_{\mathsf{e}} \mathsf{b}\rangle \wedge \langle \mathsf{c}@_1(\neg p)? =_{\mathsf{e}} \mathsf{b}\rangle \wedge \neg\langle \mathsf{b} \neq_{\mathsf{e}} @_1\rangle.$$

The call to SAT(φ) invokes SAT(Φ, n) with $\Phi = [2{:}\varphi]$ and $\mathrm{n} = 3$. We proceed to list the different steps in Alg. 2 on this input. Sometimes we group several steps together and omit some node expressions to make the presentation more succinct.

Initialization: We set $\mathrm{B} = \Phi$ and $\Sigma = [\,]$. Notice that the sole root nominal in φ is 1.

Step 1: We reach the 'expand' block and apply rules of type 1 until no such rule is left untreated. The list B is updated during this process (the rules applied to obtain the node expressions are indicated on the right):

$$\begin{aligned} \mathrm{B} = {} & [2{:}\langle \mathsf{a}(1 \wedge p)? =_{\mathsf{e}} \mathsf{b}\rangle, 2{:}\langle \mathsf{c}@_1(\neg p)? =_{\mathsf{e}} \mathsf{b}\rangle, 2{:}\neg\langle \mathsf{b} \neq_{\mathsf{e}} @_1\rangle,] + \cdots + & (\wedge) \\ & [\langle @_2\mathsf{a}(1 \wedge p)? =_{\mathsf{e}} @_2\mathsf{b}\rangle, \langle @_2\mathsf{c}@_1(\neg p)? =_{\mathsf{e}} @_2\mathsf{b}\rangle, \neg\langle @_2\mathsf{b} \neq_{\mathsf{e}} @_2@_1\rangle] & (\mathrm{int}_1), (\mathrm{int}_2) \end{aligned}$$

Step 2: We reach the 'explore' block and choose to apply the rule of type 3 with premiss $\langle @_2\mathsf{a}(1 \wedge p)? =_{\mathsf{e}} @_2\mathsf{b}\rangle$. Before the recursive call we get

$$\Phi = [\neg\langle @_2\mathsf{b} \neq_{\mathsf{e}} @_2@_1\rangle, 2{:}\langle \mathsf{a}\rangle 3, \langle @_3(1 \wedge p)? =_{\mathsf{e}} @_2\mathsf{b}\rangle]$$

Step 3: We call $\mathrm{S}\textsc{at}(\Phi, \mathrm{n})$ recursively for a first time, with $\mathrm{n} = 4$. In the 'expand' block of the first recursive call, we apply all rules of type 1 and get

$$\begin{aligned} \mathrm{B} = {} & [\neg\langle @_2\mathsf{b} \neq_{\mathsf{e}} @_2@_1\rangle, 2{:}\langle \mathsf{a}\rangle 3, \langle @_3(1 \wedge p)? =_{\mathsf{e}} @_2\mathsf{b}\rangle,] + \cdots + \\ & [3{:}1, 3{:}p, 1{:}3, 1{:}p, 2{:}\langle \mathsf{a}\rangle 1, \langle @_3 =_{\mathsf{e}} @_2\mathsf{b}\rangle, \langle @_1 =_{\mathsf{e}} @_2\mathsf{b}\rangle, \langle @_2\mathsf{b} =_{\mathsf{e}} @_1\rangle] \end{aligned}$$

Notice that $1{:}p$ is obtained from (copy), $2{:}\langle \mathsf{a}\rangle 1$ from (copy$_0$) and $\langle @_1 =_{\mathsf{e}} @_2\mathsf{b}\rangle$ from (copy$_1$). This way, the nominal 3 is replaced by the root nominal 1. Finally, the last node expression in B is obtained by applying the rule (com$_1$).

Step 4: In the 'explore' block of the first recursive call we choose to apply the rule of type 3 with premiss $\langle @_2\mathsf{b} =_{\mathsf{e}} @_1\rangle$. Before the recursive call we get

$$\Phi = [\neg\langle @_2\mathsf{b} \neq_{\mathsf{e}} @_2@_1\rangle, 1{:}p, 2{:}\langle \mathsf{a}\rangle 1, 2{:}\langle \mathsf{b}\rangle 4, \langle @_4 =_{\mathsf{e}} @_1\rangle]$$

Step 5: We call $\mathrm{S}\textsc{at}(\Phi, \mathrm{n})$ recursively for a second time (with $\mathrm{n} = 5$). In the 'expand' block of the second recursive call, we apply all rules of type 1 and get

$$\begin{aligned} \mathrm{B} = {} & [\neg\langle @_2\mathsf{b} \neq_{\mathsf{e}} @_2@_1\rangle, 2{:}\langle \mathsf{b}\rangle 4, \langle @_4 =_{\mathsf{e}} @_2@_1\rangle] + \cdots + & \\ & [\neg\langle @_4 \neq_{\mathsf{e}} @_2@_1\rangle] + \cdots + & (\neg\mathrm{child}) \\ & [\neg\langle @_4 \neq_{\mathsf{e}} @_1\rangle] & (\neg @) \end{aligned}$$

Step 6: Since no rules of type 2 are triggered, no clash is found, and there are some untreated node expressions, the algorithm proceeds to the 'explore' phase again. A new rule of type 3 is triggered, this time with premise $\psi = \langle @_2\mathsf{c}@_1(\neg p)? =_{\mathsf{e}} @_2\mathsf{b}\rangle$. Before the new recursive call, we get

$$\Phi = [\neg\langle @_2 \neq_{\mathsf{e}} @_1\rangle, 1{:}p] + \cdots + [2{:}\langle \mathsf{c}\rangle 5, \langle @_5@_1(\neg p)? =_{\mathsf{e}} @_2\mathsf{b}\rangle]$$

Step 7: The recursive call invokes $\mathrm{S}\textsc{at}(\Phi, \mathrm{n})$, with $\mathrm{n} = 6$. In the 'expand' block, we get

$$\begin{aligned} \mathrm{B} = {} & [\neg\langle @_2 \neq_{\mathsf{e}} @_1\rangle, 1{:}p, 2{:}\langle \mathsf{c}\rangle 5, \langle @_5@_1(\neg p)? =_{\mathsf{e}} @_2\mathsf{b}\rangle] + \cdots + & \\ & [\langle @_1(\neg p)? =_{\mathsf{e}} @_2\mathsf{b}\rangle, \langle @_1 =_{\mathsf{e}} @_2\mathsf{b}\rangle, 1{:}\neg p] & (@), (?) \end{aligned}$$

Step 8: After executing the Step 7, the algorithm finds a clash in line 21, since both $1{:}p$ and $1{:}\neg p$ belong to the sole tableau branch. Therefore, Alg. 2 returns that φ is unsatisfiable.

Notice how in each step the algorithm keeps track of the list of accesibility constraints created in the exploration of diamonds in a DFS manner. The length of these lists is bounded by the lengths of the path expressions traversed in each diamond. Moreover, we only need to keep track of a polynomially bounded number of node expressions at each point in these paths. This example illustrates the polynomial space bound in Thm. 5.1.

Reachability Rules

$$\frac{i{:}\langle \mathsf{a}\rangle j}{i{:}\langle + \rangle j}\ (\text{+intro}) \qquad \frac{\begin{array}{c} i{:}\langle + \rangle j \\ j{:}\langle + \rangle k \end{array}}{i{:}\langle + \rangle k}\ (\text{trans}) \qquad \frac{\begin{array}{c} i{:}\langle \mathsf{a}\rangle j \\ j{:}k \end{array}}{i{:}\langle + \rangle k}\ (\text{+copy})$$

Figure 8. Tableau Rules for Forests and Trees.

6. Forest and Tree Models

In this section, we extend our tableau calculus to ensure that extracted models are *data trees.* Besides their theoretical interest, date trees are relevant as they are the underlying structure of XML documents in practice. In this extension of the calculus, we take advantage of the expressive power of nominals and the possibility to "jump" to evaluation points to characterize *forests*, i.e., disjoint unions of trees, and trees as a particular case. We introduce rules and clash conditions to determine whether nodes have at most one predecessor, and to avoid loops via accessibility relations. We then extend our soundness and completeness results to deal with the class of forests. Finally, we show how to get completeness with respect to the class of trees by checking a single constraint on open branches.

To work with forests and trees, we extend our tableaux rules with a new type of accessibility constraint called *transitivity constraint.* Transitivity constraints are expressions of the form $i{:}\langle + \rangle j$. They are introduced into the calculus to represent reachability navigating accessibility relations. We introduce transitivity constraints and the new rules of the calculus in Fig. 8. In this figure, the rule (+intro) ensures that the relation described by + contains the relation described by a, whereas the rule (trans) indicates that the relation described by + is a transitive relation. Finally, the rule (+copy) takes care of dealing with urfathers, copying transitivity constraints where they need to be copied.

Remark 6.1. It is important to point out that even though transitivity constraints can be mistaken as characterizing the transitive closure of $\bigcup_{\mathsf{a}\in\mathsf{Rel}} R_{\mathsf{a}}$, this is not the case. The relation associated to $\langle + \rangle$ in the extracted model may not be the smallest transitive relation containing $\bigcup_{\mathsf{a}\in\mathsf{Rel}} R_{\mathsf{a}}$. In any case, transitivity constraints suffice for handling forests (and trees as a particular case).

Furthermore, we can always assume that the symbol + is not part of the set Rel, and then does not occur in the node expression at the root of the tableau. Thus, the obtained tableau is still internalized, but over the extended signature $\mathsf{Rel}' = \mathsf{Rel} \cup \{+\}$.

In addition to the rules in Fig. 8, we introduce new clash conditions for forests and trees.

Definition 6.2. A branch Θ is said to contain a *clash* if it satisfies any of the conditions listed in Def. 3.7, or if any of the following situations arise:
(3) $i{:}\langle + \rangle i \in \Theta$,
(4) $\{j{:}\langle + \rangle i, k{:}\langle + \rangle i\} \subseteq \Theta$, and $\{j{:}k, j{:}\langle + \rangle k\} \cap \Theta = \emptyset$.

Notice how condition (3) in Def. 6.2 is an indication that no node can be reached by itself; whereas condition (4) in Def. 6.2 tells us that a node cannot have more than one predecessor. More precisely, condition (4) tells us that if a nominal is reachable from two others, then, they either indicate the same node, or one must be reachable from the other.

The following example demonstrates how the introduction of new rules and clash conditions restricts the range of models that satisfy a given node expression.

Example 6.3. Fig. 9 depicts a tableau for $\langle @_0\mathsf{a}0? =_{\mathsf{e}} p?\rangle$. In brief, $\langle @_0\mathsf{a}0? =_{\mathsf{e}} p?\rangle$ forces a self loop at the node named by 0. Applying only the rules in Figs. 2 to 5 from Section 3 yields the first 11 nodes of the tableau. Notice that if we restrict our attention only to those 11 nodes (i.e., if we truncate the tableau at the node 11), we obtain a branch that is open and saturated according to Def. 3.7 (modulo the application of reflexivity and symmetry rules, which are omitted). This indicates that $\langle @_0\mathsf{a}0? =_{\mathsf{e}} p?\rangle$ is satisfiable in the class of all models. The nodes 12 and 13 show the effect of the new rules, and the way in which they allow us to talk about reachability. Notice also that node 13 introduces a clash condition as per Def. 6.2; i.e., a node is reachable from itself. This clash condition tells that $\langle @_0\mathsf{a}0? =_{\mathsf{e}} p?\rangle$ is unsatisfiable in the class of tree models; and thus also unsatisfiable in the class of forests.

1.	$1{:}\langle @_0\mathsf{a}0? =_{\mathsf{e}} p?\rangle$	root
2.	$\langle @_1 @_0\mathsf{a}0? =_{\mathsf{e}} @_1 p?\rangle$	1 (int_1)
3.	$\langle @_0\mathsf{a}0? =_{\mathsf{e}} @_1 p?\rangle$	2 (@)
4.	$0{:}\langle \mathsf{a}\rangle 2$	3 (child)
5.	$\langle @_2 0? =_{\mathsf{e}} @_1 p?\rangle$	3 (child)
6.	$2{:}0$	5 (test)
7.	$\langle @_2 =_{\mathsf{e}} @_1 p?\rangle$	5 (test)
8.	$\langle @_1 p? =_{\mathsf{e}} @_2\rangle$	7 (com_1)
9.	$1{:}p$	8 (test)
10.	$\langle @_1 =_{\mathsf{e}} @_2\rangle$	8 (test)
11.	$\langle @_2 =_{\mathsf{e}} @_1\rangle$	10 (com_1)
12.	$0{:}\langle +\rangle 2$	4 (+intro)
13.	$0{:}\langle +\rangle 0$	12, 6 (+copy)
	$\underset{12}{\otimes}$	

FIGURE 9. A closed tableau for $\langle @_0\mathsf{a}0? =_{\mathsf{e}} p?\rangle$

We are now in a position to prove that the new rules and clash conditions indeed force forest-like structures.

Proposition 6.4. *Let Θ be an open saturated branch in the extended calculus, and let $\mathfrak{M}^\Theta$ be its extracted model. Then, $\mathfrak{M}^\Theta$ is a forest.*

Proof. The proof is concluded if $F = (\bigcup_{\mathsf{a}\in\mathsf{Rel}} R^\Theta_{\mathsf{a}})^+$ is a forest. To this end, we show that:

(a) it is not the case that $(n,n) \in F$, and
(b) if $(n,m) \in F$ and $(n',m) \in F$, then $n = n'$ or $(n,n') \in F$.

The proof of (a) is by contradiction. Suppose that $(n,n) \in F$. Then, exist $\mathsf{a}_1,\dots,\mathsf{a}_\ell$ such that $(n,i) \in R_{\mathsf{a}_1}, (i,j) \in R_{\mathsf{a}_2} \dots (k,n) \in R_{\mathsf{a}_\ell}$. From the definition of R^Θ_{a}, we have that $\{n{:}\langle \mathsf{a}_1\rangle i, i{:}\langle \mathsf{a}_2\rangle j, \dots, k{:}\langle \mathsf{a}_\ell\rangle n\} \subseteq \Theta$. Since Θ is closed under (+intro), we also have that $\{n{:}\langle +\rangle i, i{:}\langle +\rangle j, \dots, k{:}\langle +\rangle n\} \subseteq \Theta$. In turn, closure under (trans), gives us $n{:}\langle +\rangle n \in \Theta$. This yields a clash using condition (3) from Def. 6.2, contradicting the assumption that Θ is open.

The proof of (b) is also by contradiction. Suppose $(n,m) \in F$, $(n',m) \in F$, $n \neq n'$, and $(n,n') \notin F$. From $(n,m) \in F$ and $(n',m) \in F$, we know that $\{n{:}\langle +\rangle m, n'{:}\langle +\rangle m\} \subseteq \Theta$. In addition, from $(n,n') \notin F$, we know $n{:}\langle +\rangle n' \notin \Theta$. Using the definition of urfather, we know that $j{:}k \notin \Theta$. When put together, these facts yield a clash using condition (4) from Def. 6.2, contradicting the assumption that Θ is open. □

To further ensure that the resulting model is a tree we only need to verify that every point in the extracted model is reachable from the root.

Definition 6.5. Let Θ be a branch of a tableau for $i{:}\varphi$ in the extended calculus. We say that Θ is *connected* iff for all $j \in \Theta$, if $i{:}j \notin \Theta$, then, $i{:}\langle + \rangle j \in \Theta$.

Proposition 6.6. *Let Θ be an open, saturated, and connected branch. Then, $\mathfrak{M}^{\Theta}$ is a tree.*

Proof. The proof is concluded if $T = (\bigcup_{\mathsf{a} \in \mathsf{Rel}} R^{\Theta}_{\mathsf{a}})^{+}$ is a tree. We already know that T is a forest. Thus, we only need to prove that for all $j \in N^{\Theta}$, $(i, j) \in T$. This is straightforward since Θ is connected. □

We conclude this section by showing that the satisfiability problem for forests and data trees remains decidable and stays within PSpace.

Theorem 6.7. *The satisfiability problem for forests and data trees in* $\mathsf{HXPath_D}$ *is decidable and PSpace-complete.*

Proof. Notice that for forests and trees, we only add three new rules which are applied over a finite set of node expressions (by Lemma 4.9). Since we apply rules only once, this yields again a finite tableau. Moreover, the new rules are clearly of type 1, thus applying them in Alg. 2 yields similar bounds as in Thm. 5.1. Checking the clash conditions can also be done using polynomial space. Therefore, the extended algorithm runs in polynomial space. □

7. Pure Axioms and Node Creating Rules

A common advantage of proof systems for Hybrid Logics is their ability to establish a very general completeness result, in which it is shown that certain extensions of the base proof system are automatically complete for the corresponding classes of models. This is in sharp contrast with many proof systems for other Modal Logics, in which simple extensions of a complete proof system might be incomplete.

This property of Hybrid Logics is particularly useful when designing proof systems for specific classes of models. For instance, certain applications may require structural constraints —such as transitivity or irreflexivity— on accessibility relations. This is specially relevant for XPath, as the different axes are usually associated with semantics (e.g., price, sub-category, sub-menu, max-level, etc.) which might need to obey structural restrictions. Extensions using pure axioms and node-creating rules address these requirements naturally, while still yielding automatic completeness results.

Extensions for $\mathsf{HXPath_D}$ that preserve completeness have been investigated in [AF21] for Hilbert-style axiom systems. In this section, we adopt a similar approach to develop complete tableau systems for extensions of $\mathsf{HXPath_D}$. To this end, we follow the methodology introduced in [Bla00, BtC02, BtC06] for Hybrid Logics. We will consider two kinds of extension: pure axioms, and node creating rules. These extensions ensure the soundness and completeness of the resulting calculi, but do not generally preserve decidability or complexity bounds. Such results depend on carefully controlling the flow of information within the tableaux and must be established on a case-by-case basis. While some structural properties can be managed using the mechanisms introduced earlier, others may fall outside the scope of such approach.

7.1. **Pure Axioms.** One of the most standard extensions of hybridized proof systems is via pure axioms. This extension enables us to characterize several interesting classes of models, including some that are not definable by so-called "orthodox" node expressions (i.e., expressions containing only propositional symbols).

Definition 7.1. A node expression is *pure* iff it does not contain propositional symbols.

Example 7.2. Below we list some pure node expressions defining interesting frame conditions:

Irreflexivity	$i{:}\neg\langle \mathsf{a}\rangle i$
Transitivity	$i{:}(\langle \mathsf{a}\rangle\langle \mathsf{a}\rangle j \to \langle \mathsf{a}\rangle j)$
Trichotomy	$i{:}\langle \mathsf{a}\rangle j \vee i{:}j \vee j{:}\langle \mathsf{a}\rangle i$
Data Uniqueness	$\langle @_i =_{\mathsf{e}} @_j\rangle \to i{:}j.$

It is interesting to notice that transitivity is definable without the use of nominals, but trichotomy and irreflexivity are not (see [Bla00]). The latter is also true for data uniqueness.

It has been shown in [AF21] that pure expressions correspond to first-order properties when taken as axioms. This makes them suitable to characterize interesting classes of models. Below we formally introduce such a notion.

Definition 7.3. A *(data) frame* is a tuple $\mathfrak{F} = \langle N, \{R_{\mathsf{a}}\}_{\mathsf{a}\in\mathsf{Rel}}, \{\approx_{\mathsf{e}}\}_{\mathsf{e}\in\mathsf{Eq}}\rangle$, where each component is as in Def. 2.2. We write $\mathfrak{F} \vDash \varphi$ iff for all $\mathfrak{M}$ based on $\mathfrak{F}$, and $n \in N$, we have $\mathfrak{M}, n \vDash \varphi$. We say that a node expression φ *defines a class of frames* $\mathbb{F}$ iff $\mathfrak{F} \vDash \varphi$ is equivalent to $\mathfrak{F} \in \mathbb{F}$.

Our tableaux receive as input node expressions that are prefixed by some $i{:}$, so in order to handle pure axioms we need to make sure they are also prefixed. Following [Bla00, BtC02], in the following lemma we show that this is not a real restriction.

Lemma 7.4. *If a pure node expression φ defines a class of frames $\mathbb{F}$ and i a nominal not in φ, then, $i{:}\varphi$ also defines $\mathbb{F}$.*

Proof. The result follows immediately from the requisite that i does not appear in φ, as in this case i does not affect the meaning of φ. $\square$

We are now in position to state a general completeness result. The idea is to incorporate (prefixed) pure expressions to check satisfiability of a node expression in a particular class of models. In other words, if we look for a model of a given node expression, pure axioms will guarantee that this model will belong to the corresponding class of models. Given a set of pure expressions `Axiom` and $\psi \in$ `Axiom`, we need to instantiate nominals appearing in ψ, with nominals appearing in the branch of the tableaux in which ψ is being incorporated. For instance, for a branch Θ with $k{:}\varphi \in \Theta$, we incorporate irreflexivity as formulated in Ex. 7.2 with the instance $k{:}\neg\langle \mathsf{a}\rangle k$. This kind of remark applies also to data comparisons.

Theorem 7.5 (Strong Completeness for Pure Axioms)**.** *Let* `Axiom` *be a set of pure node expressions prefixed by some $i{:}$. Extend the tableau calculus from Section 3 with the rule (Pure) which adds, at any stage, any $\psi \in$* `Axiom` *instantiated with nominals already appearing in the branch. The obtained tableau is complete with respect to the class of models based on frames defined by $\bigwedge_{\psi\in\texttt{Axiom}} \psi$.*

Proof. The argument for this proof is the same as the one in [Bla00, Th. 7.2] and [BtC02, Prop. 3.2] for tableau calculi for hybrid logics, and [AF21, Th. 3.23] for axiomatizations of hybrid XPath. Intuitively, the rule (Pure) ensures that pure axioms in `Axiom` are fully instantiated in a suitable way, in an open and saturated branch Θ. As a consequence, the extracted model $\mathfrak{M}^\Theta$ globally satisfies all relevant instances in `Axiom`. This is sufficient to ensure that the frame of $\mathfrak{M}^\Theta$ satisfies the structural properties associated with `Axiom`. In other words, the extracted model $\mathfrak{M}^\Theta$, satisfying the root formula in the branch, will belong to the required model class. □

By choosing a suitable set of pure axioms it is possible to characterize interesting XPath features. We introduce some of them in the following example.

Example 7.6. As our first example of the use of pure actions, we begin by considering the *inverse* relation, a distinctive feature in XPath fragments which allows for backwards navigation. To this end, we extend the set of relation symbols in the signature to $\mathsf{Rel}' = \mathsf{Rel} \cup \{\mathsf{a}^- \mid \mathsf{a} \in \mathsf{Rel}\}$. Then, we define

$$\texttt{Inverse} = \{i{:}(j \to [\mathsf{a}]\langle \mathsf{a}^- \rangle j), i{:}(j \to [\mathsf{a}^-]\langle \mathsf{a} \rangle j) \mid \mathsf{a} \in \mathsf{Rel}\}.$$

By incorporating `Inverse`, our tableaux yields a complete calculus for $\mathsf{HXPath_D}$ with inverse relations (over the new signature). More precisely, pure axioms in `Inverse` ensure that, in any model considered, $R_\mathsf{a} = (R_{\mathsf{a}^-})^-$. Hence, a^- can be used to refer to the inverse of a.

For a second example, fix now $\mathsf{Rel}'' = \mathsf{Rel}' \cup \{\mathsf{s}\}$, and define

$$\texttt{Sibling} = \texttt{Inverse} \cup \{i{:}(\langle \mathsf{s} \rangle j \to \langle \mathsf{a}^- \rangle \langle \mathsf{a} \rangle j), i{:}(\neg j \wedge \langle \mathsf{a}^- \rangle \langle \mathsf{a} \rangle j \to \langle \mathsf{s} \rangle j), i{:}(j{:}\neg \langle \mathsf{s} \rangle j) \mid \mathsf{a} \in \mathsf{Rel}\}.$$

By incorporating `Sibling` we get, additionally, completeness for $\mathsf{HXPath_D}$ with R_s being the *sibling* relation. The first node expression states that a sibling is one node found going backwards and then forwards in the model. The second node expression states the converse of the first expression, in case j is not the current node. Finally, the last expression expresses that a node is not its own sibling.

As a final example, define

$$\texttt{s-Irreflexivity} = \texttt{Sibling} \cup \{i{:}(\langle \mathsf{s} \rangle j \to (\langle @_i \neq_\mathsf{e} @_j \rangle))\}.$$

`s-Irreflexivity` gives us completeness with respect to the class of models in which siblings have different data values for the comparison criteria e. This could be useful, for example, if when the siblings in the data tree represent categories with different prize ranges (e.g., economy, business, first-class).

We stress that the Strong Completeness Theorem (Thm. 7.5) guarantees that the resulting calculi are complete for the intended model classes. All the extensions discussed in Ex. 7.6 correspond to common features found in various versions of XPath. It is therefore desirable to have tableau calculi that are not only sound but also complete for these extensions. As it turns out, in all these cases, completeness is automatically preserved due to the specific form of the axioms being added. This being said, adding such axioms does not imply complexity bounds are preserved. For instance, the satisfiability problem for Hybrid Logic $\mathcal{H}(:)$ extended with the inverse operator is already known to be ExpTime-complete [ABM00]. Since $\mathcal{H}(:)$ with inverses is a proper fragment of $\mathsf{HXPath_D}$ extended with inverses, this immediately yields an ExpTime lower bound for the satisfiability problem of $\mathsf{HXPath_D}$ with inverses.

7.2. **Node Creating Rules.** We now turn our attention to introducing so-called *node creation rules* into our tableaux. To precisely characterize the expressivity provided by these new rules, we need to extend the basic language $\mathsf{HXPath_D}$ to a strong version with quantification over nominals, as is done for Hybrid Logic [BS95]. It is important to clarify that the strong hybrid language for XPath introduced in Def. 7.7 is used solely to characterize the expressive power of the node-creating rules defined below. It is also worth noting that the satisfiability problem for the strong hybrid language for XPath is undecidable [AtC06]. Therefore, node-creating rules must be used with caution when aiming for a tableau calculus with good computational properties.

Definition 7.7 (Strong Hybrid Language for XPath)**.** The *strong hybrid language* for XPath is obtained by allowing expressions of the form $\exists x.\varphi$ (where $x \in \mathsf{Nom}$, and φ is a node expression) as node expressions. The semantics of these new node expressions over models $\mathfrak{M} = \langle N, \{R_{\mathsf{a}}\}_{\mathsf{a}\in\mathsf{Rel}}, \{\approx_{\mathsf{e}}\}_{\mathsf{e}\in\mathsf{Eq}}, V, \mathrm{nom}\rangle$ is given as:

$$\mathfrak{M}, n \vDash \exists x.\varphi \quad \textit{iff} \quad \text{there is nom}' \text{ s.t. } \mathfrak{M}', n \vDash \varphi,$$

where $\mathfrak{M}' = \langle N, \{R_{\mathsf{a}}\}_{\mathsf{a}\in\mathsf{Rel}}, \{\approx_{\mathsf{e}}\}_{\mathsf{e}\in\mathsf{Eq}}, V, \mathrm{nom}'\rangle$ for nom′ defined exactly as nom except, perhaps, for x. The expression $\forall x.\varphi$ is defined as $\neg\exists.\neg\varphi$. We will use $\exists x, y, \ldots$ and $\forall x, y, \ldots$ as shortcuts for $\exists x.\exists y \ldots$ and $\forall x.\forall y \ldots$, respectively.

The strong hybrid language helps highlight a key limitation of pure axioms and motivates the need for node-creating rules. Pure axioms correspond to universally quantified expressions in the strong hybrid language (see [BtC02, Prop. 3.2] for details). However, certain natural frame properties cannot be captured using pure axioms alone. A well-known example is the Church-Rosser class of frames, or the property of right-directness. The former property can be expressed as

$$\forall i, j, k.\exists x.(i{:}\langle\mathsf{a}\rangle j \wedge i{:}\langle\mathsf{a}\rangle x \rightarrow j{:}\langle\mathsf{a}\rangle x \wedge k{:}\langle\mathsf{a}\rangle x),$$

while the latter can be expressed as

$$\forall i, j.\exists k.(i{:}\langle\mathsf{a}\rangle k \wedge j{:}\langle\mathsf{a}\rangle k).$$

Both expressions involve a $\forall\exists$ quantification pattern, which goes beyond the purely universal pattern allowed by pure axioms. Node-creating rules provide a natural way to handle such $\forall\exists$ properties: the premises correspond to the universally quantified parts, while the conclusions create new nodes serving as witnesses for the existential quantifiers.

For example,

$$\forall i, j.\exists k.\langle @_i \neq_{\mathsf{e}} @_k\rangle \wedge \langle @_j \neq_{\mathsf{e}} @_k\rangle$$

can be casted into a node generating rule as

$$\frac{i, j \text{ in the branch}}{\begin{array}{c}\langle @_j \neq_{\mathsf{e}} @_k\rangle \\ \langle @_i \neq_{\mathsf{e}} @_k\rangle\end{array}} \text{ with } k \text{ a new nominal.}$$

Incorporating this rule into our tableaux ensures that for every pair of nodes i and j in a branch, there is a node k whose data value differs from both.

Let us now move to the general form of these rules and how to represent them. Let $\forall x_1, \ldots, x_\ell.\exists y_1, \ldots y_{\ell'}.\varphi$ be a node expression from the strong hybrid language, such that φ is quantifier free. For every such a node expression we define the rule

$$\frac{i_1, \ldots, i_\ell \text{ in the branch}}{\varphi[x_1 \leftarrow i_1, \ldots, x_\ell \leftarrow i_\ell, y_1 \leftarrow k_1 \ldots y_{\ell'} \leftarrow k_{\ell'}]} \text{ with } k_1, \ldots, k_{\ell'} \text{ new nominals.}$$

Once more, we can prove a Strong Completeness Theorem for Node Creating Rules.

Theorem 7.8 (Strong Completeness for Node Creating Rules)**.** *Let S be a set of node expressions from the strong hybrid language of the form $\forall x_1, \ldots, x_\ell.\exists y_1, \ldots y_{\ell'}.\varphi$, where φ is quantifier free. Extend the tableau calculus from Section 3 with the associated rules as defined* *above. The obtained tableaux is complete with respect to the class of models whose frames are defined by* $\bigwedge_{\psi \in S} \psi$.

Proof. The argument is as in the proof of Thm. 7.5. □

8. Final Remarks

We introduced tableau calculi for data-aware logics, with particular attention to variants of the query language XPath extended with data comparisons. Precisely, we presented an internalized tableau calculus for the logic called $\mathsf{HXPath_D}$. This logic extends the standard XPath formalism as studied in the context of Modal Logic by incorporating data comparisons, along with nominals and satisfiability modalities from Hybrid Logic. We proved that our tableau calculus is sound, complete, and terminating, making it a practical tool for checking the satisfiability of $\mathsf{HXPath_D}$ node and path expressions. Additionally, we showed that rule applications can be systematically restricted to ensure that tableau construction runs in polynomial space, without sacrificing completeness. This yields a PSpace-completeness result for the satisfiability problem in $\mathsf{HXPath_D}$.

Our work builds upon [AFS17] and offers several enhancements. Most notably, it provides a detailed account of the termination argument, which was only briefly outlined in the earlier article. The presence of nominals and satisfiability modalities in the language implies that the tableau calculi need to perform equality reasoning, which complicates termination. In particular, the introduction of new nominals needs to be carefully handled. In addition, we demonstrate that the PSpace upper bound also applies in the case of trees. The current proposal is, arguably, simpler and more elegant than the presentation of the calculus given in [AFS17]. In particular, the PSpace upper bound for satisfiability is presented in full, at a level of detail suitable for implementation. The tableau calculi introduced in this article are *internalized*, i.e., rules only contain expressions of $\mathsf{HXPath_D}$. It is known that internalized proof systems for Hybrid Logics enjoy some nice properties [Bla00, BtC02]. We exploited these properties here, demonstrating how to extend the tableau calculus with pure axioms and node creating rules to characterize interesting model classes. Moreover, we showed that strong completeness theorems holds for these particular kind of tableau rules, ensuring that the resulting calculi are automatically complete for the corresponding model classes. We illustrated how these rules can be used to capture new operators and model classes relevant for applications based on semi-structured data.

The results presented in this article pave the way to a modal investigation of other data-aware logics that are receiving considerable attention recently, such as GQL [FGG$^+$23a, FGG$^+$23b] and SHACL [Ort23, AOOS23]. It also makes it possible to consider Modal Logics built over more complex data structures, such as Concrete Domains [DQ21]. It would be interesting to explore futher connections with Register Automata [KF94] and similar structures. Finally, implementing a tableaux-based procedure for data-aware languages over the hybrid prover HTab [HA09] is also part of our agenda.

Acknowledgments. This work is supported by projects ANPCyT-PICT-2020-3780, PICT-2021-00400, CONICET projects PIBAA-28720210100428CO, PIBAA28720210100165CO, Stic-AmSud 23-STIC-07 'DL(R)', Secyt-UNC, the EU Grant Agreement 101008233 (MISSION) and by the Laboratoire International Associé SINFIN.

References

[ABFF16] S. Abriola, P. Barceló, D. Figueira, and S. Figueira. Bisimulations on data graphs. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR*, pages 309–318, 2016.

[ABM00] C. Areces, P. Blackburn, and M. Marx. The computational complexity of hybrid temporal logics. *Logic Journal of the IGPL*, 8(5):653–679, 2000.

[ABS99] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[ACDF23] C. Areces, V. Cassano, D. Dutto, and R. Fervari. Data graphs with incomplete information (and a way to complete them). In *Logics in Artificial Intelligence. 18th European Conference, JELIA 2023*, Lecture Notes in Computer Science, pages 729–744. Springer, 2023.

[ADF14] S. Abriola, M. Descotte, and S. Figueira. Definability for downward and vertical XPath on data trees. In *21th Workshop on Logic, Language, Information and Computation*, volume 6642 of *LNCS*, pages 20–34, 2014.

[ADF17] S. Abriola, M. Descotte, and S. Figueira. Model theory of XPath on data trees. Part II: Binary bisimulation and definability. *Information and Computation*, 255:195–223, 2017. URL: `https://www.sciencedirect.com/science/article/pii/S0890540117300032`, `doi:10.1016/j.ic.2017.01.002`.

[ADFF17] S. Abriola, M. Descotte, R. Fervari, and S. Figueira. Axiomatizations for downward XPath on data trees. *Journal of Computer and System Sciences*, 89:209–245, 2017.

[AF16] C. Areces and R. Fervari. Hilbert-style axiomatization for hybrid XPath with data. In Loizos M. and A. Kakas, editors, *Logics in Artificial Intelligence - 15th European Conference, JELIA 2016*, volume 10021 of *LNCS*, pages 34–48, 2016.

[AF21] C. Areces and R. Fervari. Axiomatizing hybrid XPath with data. *Logical Methods in Computer Science*, 17(3), 2021. URL: `https://doi.org/10.46298/lmcs-17(3:5)2021`, `doi:10.46298/LMCS-17(3:5)2021`.

[AFS17] C. Areces, R. Fervari, and N. Seiler. Tableaux for hybrid XPath with data. In *Progress in Artificial Intelligence - 18th EPIA Conference on Artificial Intelligence, EPIA 2017*, volume 10423 of *LNCS*, pages 611–623. Springer, 2017. `doi:10.1007/978-3-319-65340-2\_50`.

[AOOS23] S. Ahmetaj, M. Ortiz, A. Oudshoorn, and M. Simkus. Reconciling SHACL and ontologies: Semantics and validation via rewriting. In *ECAI 2023 - 26th European Conference on Artificial Intelligence*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pages 27–35. IOS Press, 2023. `doi:10.3233/FAIA230250`.

[AtC06] C. Areces and B. ten Cate. Hybrid logics. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logic*, pages 821–868. Elsevier, 2006.

[BB07] T. Bolander and P. Blackburn. Termination for hybrid tableaus. *Journal of Logic and Computation*, 17(3):517–554, 2007.

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[BFG08] M. Benedikt, W. Fan, and F. Geerts. XPath satisfiability in the presence of DTDs. *Journal of the ACM*, 55(2):1–79, 2008. `doi:10.1145/1346330.1346333`.

[BK09] M. Benedikt and C. Koch. XPath leashed. *ACM Computing Surveys*, 41(1):3:1–3:54, January 2009. URL: `http://doi.acm.org/10.1145/1456650.1456653`, `doi:10.1145/1456650.1456653`.

[Bla00] P. Blackburn. Internalizing labelled deduction. *Journal of Logic and Computation*, 10(1):137–168, 2000. URL: `https://doi.org/10.1093/logcom/10.1.137`, `doi:10.1093/LOGCOM/10.1.137`.

[BLS16] 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 2016*, pages 32:1–32:16, 2016.

[BMSS09] 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. URL: `http://doi.acm.org/10.1145/1516512.1516515`, `doi:10.1145/1516512.1516515`.

[Bra07] T. Braüner. Why does the proof-theory of hybrid logic work so well? *Journal of Applied Logic*, 17(4):521–543, 2007. URL: `https://doi.org/10.3166/jancl.17.521-543`, `doi:10.3166/JANCL.17.521-543`.

[Bra11] T. Braüner. *Hybrid Logic and its Proof-Theory*, volume 37 of *Applied Logics Series*. Springer, 2011.

[BS95] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4(3):251–272, 1995. doi:10.1007/BF01049415.

[BtC02] P. Blackburn and B. ten Cate. Beyond pure axioms: Node creating rules in hybrid tableaux. In *4th Workshop on Hybrid Logics (Hylo@LICS)*, 2002.

[BtC06] P. Blackburn and B. ten Cate. Pure extensions, proof rules, and hybrid axiomatics. *Studia Logica*, 84(2):277–322, 2006.

[Bun97] P. Buneman. Semistructured data. In *In ACM Symposium on Principles of Database Systems (PODS'97)*, pages 117–121, 1997.

[BvB06] P. Blackburn and J. van Benthem. Modal Logic: A Semantic Perspective. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logic*, pages 1–84. Elsevier, 2006.

[BWvB06] P. Blackburn, F. Wolter, and J. van Benthem, editors. *Handbook of Modal Logic*. Elsevier, 2006.

[CD99] J. Clark and S. DeRose. XML path language (XPath). Website, 1999. W3C Recommendation. http://www.w3.org/TR/xpath.

[Cla99] J Clark. XSL transformations (XSLT). Website, 1999. W3C Recommendation. http://www.w3.org/TR/xslt.

[CLM10] B. ten Cate, T. Litak, and M. Marx. Complete axiomatizations for XPath fragments. *Journal of Applied Logic*, 8(2):153–172, 2010. URL: http://dx.doi.org/10.1016/j.jal.2009.09.002, doi:10.1016/j.jal.2009.09.002.

[CM09] B. ten Cate and M. Marx. Axiomatizing the logical core of XPath 2.0. *Theory of Computing Systems*, 44(4):561–589, 2009. URL: http://dx.doi.org/10.1007/s00224-008-9151-9, doi:10.1007/s00224-008-9151-9.

[DQ21] S. Demri and K. Quaas. Concrete domains in logics: a survey. *ACM SIGLOG News*, 8(3):6–29, 2021. doi:10.1145/3477986.3477988.

[FFA14] D. Figueira, S. Figueira, and C. Areces. Basic model theory of XPath on data trees. In *International Conference on Database Theory*, pages 50–60, 2014.

[FFA15] 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.

[FGG$^+$23a] N. Francis, A. Gheerbrant, P. Guagliardo, L. Libkin, V. Marsault, W. Martens, F. Murlak, L. Peterfreund, A. Rogova, and D. Vrgoc. GPC: A pattern calculus for property graphs. In *42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, (PODS'23)*, pages 241–250. ACM, 2023. doi:10.1145/3584372.3588662.

[FGG$^+$23b] N. Francis, A. Gheerbrant, P. Guagliardo, L. Libkin, V. Marsault, W. Martens, F. Murlak, L. Peterfreund, A. Rogova, and D. Vrgoc. A researcher's digest of GQL (invited talk). In *26th International Conference on Database Theory, ICDT 2023*, volume 255 of *LIPIcs*, pages 1:1–1:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.ICDT.2023.1.

[Fig10] D. Figueira. *Reasoning on Words and Trees with Data*. PhD thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, 2010. URL: http://www.lsv.ens-cachan.fr/~figueira/phd/latest/thesisFigueira.pdf.

[Fig12] D. Figueira. Decidability of downward XPath. *ACM Transactions on Computational Logic*, 13(4):34, 2012. URL: http://doi.acm.org/10.1145/2362355.2362362, doi:10.1145/2362355.2362362.

[Fig13] D. Figueira. On XPath with transitive axes and data tests. In Wenfei Fan, editor, *In ACM Symposium on Principles of Database Systems (PODS'13)*, pages 249–260, New York, NY, USA, 2013. ACM Press. doi:10.1145/2463664.2463675.

[Fit83] M. Fitting. *Proof Methods for Modal and Intuitionistic Logics*. Synthese Library Volume. Springer Netherlands, 1983. URL: https://books.google.com.ar/books?id=FfQOdQssjCAC.

[Fit96] M. Fitting. *First-Order Logic and Automated Theorem Proving, Second Edition*. Graduate Texts in Computer Science. Springer, 1996. doi:10.1007/978-1-4612-2360-3.

[FS11] D. Figueira and L. Segoufin. Bottom-up automata on data trees and vertical XPath. In *28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, pages 93–104, 2011. URL: http://dx.doi.org/10.4230/LIPIcs.STACS.2011.93, doi:10.4230/LIPIcs.STACS.2011.93.

[GA21] N. González and S. Abriola. Characterizations for XPath$_{\mathbf{R}}(\downarrow)$. In A. Silva, R. Wassermann, and R. de Queiroz, editors, *Logic, Language, Information, and Computation - 27th International*

Workshop, WoLLIC 2021, volume 13038 of *LNCS*, pages 319–336. Springer, 2021. `doi:10.1007/978-3-030-88853-4\_20`.

[GKP05] G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing XPath queries. *ACM Transactions on Database Systems*, 30(2):444–491, 2005. `doi:10.1145/1071610.1071614`.

[HA09] G. Hoffmann and C. Areces. HTab: A terminating tableaux system for hybrid logic. *Electronic Notes in Theoretical Computer Science*, 231:3–19, March 2009. `doi:10.1016/j.entcs.2009.02.026`.

[KF94] M. Kaminski and N. Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994. `doi:10.1016/0304-3975(94)90242-9`.

[Lad77] R. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM Journal of Computing*, 6(3):467–480, 1977. `doi:10.1137/0206033`.

[Mar04] M. Marx. XPath with conditional axis relations. In *International Conference on Extending Database Technology (EDBT'04)*, volume 2992 of *LNCS*, pages 477–494. Springer, 2004. `doi:10.1007/b95855`.

[Mar06] M. Marx. Complexity of modal logic. In P. Blackburn, F. Wolter, and J. van Benthem, editors, *Handbook of Modal Logic*, pages 139–179. Elsevier, 2006.

[MdR05] M. Marx and M. de Rijke. Semantic characterizations of navigational XPath. *ACM SIGMOD Record*, 34(2):41–46, 2005.

[Ort23] M. Ortiz. A short introduction to SHACL for logicians. In *Logic, Language, Information, and Computation - 29th International Workshop, WoLLIC 2023*, volume 13923 of *LNCS*, pages 19–32. Springer, 2023. `doi:10.1007/978-3-031-39784-4\_2`.

[Pri08] Graham Priest. *An Introduction to Non-Classical Logic: From If to Is*. Cambridge University Press, 2 edition, 2008.

[RCDS14] J. Robie, D. Chamberlin, M. Dyck, and J. Snelson. XQuery 3.0: An XML query language. Website, 2014. W3C Recommendation. `https://www.w3.org/TR/xquery-30/`.

[Smu68] R. Smullyan. Analytic tableaux. In *First-Order Logic*, pages 15–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 1968. `doi:10.1007/978-3-642-86718-7_2`.