
Trabajo Especial de la Licenciatura en Ciencias de la Computación

# Algoritmos de Tableaux para XPath con Datos

*Autor:*
Nahuel SEILER

*Director:*
Dr. Raul FERVARI

Marzo 28, 2018

# Contenido

**Resumen**

En este trabajo se presenta un cálculo correcto y completo para XPath con datos y caminos descendentes, enriquecido con nominales y operadores de satisfacción. Llamaremos HXPath$_{=}(\downarrow)$ al lenguaje híbrido que resulta de agregar operadores híbridos a XPath. Primero se mencionan aspectos básicos de las lógicas modales, híbridas y de XPath, para luego dar el cálculo de tableaux para XPath$_{=}$. Finalmente se demuestra completitud del cálculo.

**Abstract**

This work presents a sound and complete calculus to check satisfiability of downward XPath formulas, enriched with nominal and satisfaction operators. We will call HXPath$_{=}(\downarrow)$ the hybrid language resulting from the addition of the hybrid operators to XPath First, basic aspects of hybrid modal logic and XPath are mentioned, then we give the tableaux calculus for XPath$_{=}$. Finally, we prove the calculus is complete.

# Capítulo 1

# XPath como lenguaje modal híbrido

En muchas aplicaciones, lidiar datos reales es un desafío importante. Por ejemplo, existen aplicaciones que manejan grandes volúmenes de datos médicos o grandes contenidos en la web que requieren, en muchos casos, modelos más complejos que los que pueden ser codificados en bases de datos relacionales clásicas. Un modelo *semi-estructurado* de datos consta de datos organizados en árboles etiquetados o grafos, y lenguajes de búsqueda para acceder y actualizar dichas estructuras.

Muchos lenguajes de búsqueda están enfocados sólo en cómo acceder a la información estructural, sin prestar demasiada atención a los datos. En este trabajo se estudiará un modelo que es capaz de expresar popiedades sobre los datos.

XML (*eXtensible Markup Language*) es posiblemente el lenguaje de marcado de datos más exitoso que captura tanto la información estructural como la de los datos. Un documento XML es un estructura jerárquica representada por un árbol finito donde los nodos están etiquetados (por símbolos de un alfabeto finito, como así también por valores de datos provenientes de un alfabeto infinito).

XPath es probablemente el lenguaje de consulta más usado para XML. Está implementado en XSLT [17] and XQuery [35]. XPath es fundamentalmente un lenguaje de propósito general para la búsqueda, comparación y direccionamiento de las distintas partes de un documento XML. Es un estándar abierto y constituye la World Wide Web Consortium (W3C) Recommendation [18]. Core-XPath [27] es un fragmento de XPath 1.0 que contiene los operadores de XPath correspondientes a la navegación: permite expresar propiedades acerca de la estructura del documento XML pero no puede expresar condiciones de los datos contenidos en los atributos inspeccionados. En otras palabras es esencialmente una lógica modal clásica [11, 12]. Core-XPath ha sido bien estudiado desde un punto de vista de la lógica modal. Por ejemplo el problema de satisfacibilidad es siempre decidible en presencia de DTDs [33, 9]. Su poder expresivo es equivalente a FO2 (lógica de primer orden con dos variables) sobre árboles con las signaturas apropiadas [34], y es estrictamente menos expresivo que PDL con operador inverso sobre árboles ordenados [10]. Axiomatizaciones correctas y completas han sido introducidas en [16, 15]. Sin embargo sin la capacidad de relacionar nodos en base a los valores de datos reales de los atributos, el poder expresivo de Core-XPath es inapropiado para muchas aplicaciones. De hecho, no es posible definir la operación *join*, el principal constructor de un lenguaje de consulta de base de datos. Una extensión de Core-XPath con test de igualdad (desigualdad) de datos es Core-Data-XPath [13]. En este trabajo lo denotaremos $\text{XPath}_{=}$. Desde el punto de vista de la lógica, los modelos de $\text{XPath}_{=}$ son árboles de datos los cuales pueden verse como abstracciones de un documento XML. Un *árbol de datos* es un árbol cuyos

nodos contienen una etiqueta de un alfabeto finito y un dato de un dominio infinito. XPath$_=$ permite fórmulas de la forma $\langle \alpha = \beta \rangle$ y $\langle \alpha \neq \beta \rangle$ donde $\alpha$, $\beta$ son expresiones sobre los caminos que recorren el árbol a través de ejes: descendente, hijo, ancestro, etc. pudiendo realizar tests en los nodos intermedios. La fórmula $\langle \alpha = \beta \rangle$ (respectivamente $\langle \alpha \neq \beta \rangle$) es verdadera en un nodo $x$ de un árbol de datos si existen nodos $y, z$ tal que pueden ser alcanzados por caminos denotados por $\alpha, \beta$ y tal que el dato alojado en el nodo $y$ es igual (respectivamente distinto) al dato alojado en el nodo $z$. Por ejemplo, en la *Figura*1.1, la expresión *"Hay un descendiente accesible en un paso y un descendiente accesible en dos pasos tal que poseen el mismo dato"* (formalmente $\langle \downarrow = \downarrow\downarrow \rangle$) es verdadera en $x$, dada la presencia de $u$ y $z$. La expresión *"Hay dos nodos hijos con distintos datos"* (formalmente $\langle \downarrow \neq \downarrow \rangle$) es también cierta en $x$, porque $y$ y $z$ tienen diferentes datos.

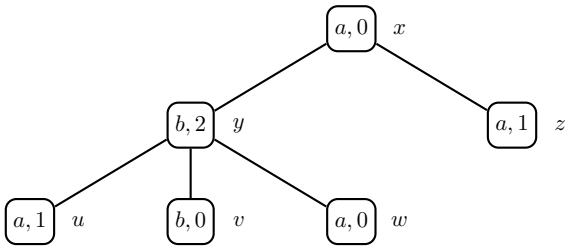


Figura 1.1: Ejemplo de árbol con datos. Las letras representan etiquetas, los números representan datos.

## 1.1 Primer cálculo correcto y completo para XPath con datos

**Trabajo relacionado.** Algunos artículos recientes estudian a XPath$_=$ desde un punto de vista de la lógica modal. Por ejemplo, el problema de decidir si una fórmula de XPath$_=$ es satisfacible ha sido investigado en [21, 25, 22], mientras que la teoría de modelos y expresividad son estudiadas en [1, 23, 24, 3]. En [8], se introduce un cálculo basado en secuentes para un fragmento restringido de XPath, llamado *DataGL*. En *DataGL*, sólo se permite comparar datos entre el punto de evaluación y sus sucesores. En [2] se introduce una extensión del sistema axiomático dado en [15], permitiendo la navegación hacia abajo y los test de igualdad/desigualdad. [4] provee una axiomatización para esta lógica, extendida con navegación ascendente, nominales y operadores de satisfacibilidad. En [5] se introduce una versión refinada del cálculo presentado en esta tesis, donde también es posible demostrar terminación del mismo.

**Motivaciones.** Se ha demostrado que el estudio de XPath desde un punto de vista formal no es meramente un desafío teórico, si no que también tiene aplicaciones concretas. Por ejemplo, el estudio del poder expresivo de algunos de sus fragmentos ayuda a determinar si una fórmula expresando un restricción de integridad puede ser simplificada. Por otro lado los problemas de determinar si el resultado de una consulta está contenido en otra, o si dos consultas son equivalentes[1] son dos tareas de razonamiento fundamentales para la optimización de consultas. Estos problemas pueden ser reducidos al problema de satisfacibilidad de una fórmula (o de manera análoga, a la validez de la misma). Estos resultados tienen impacto directo en seguridad [20], chequeo de tipos [32] y consistencia de especificaciones XML [7], solo por nombrar algunas.

[1] Estos problemas son conocidos como *query containment* y *query equivalence* en la literatura

**Contribuciones.** En este trabajo se introduce un cálculo de tableaux correcto y completo para el lenguaje XPath con navegación descendente y comparación de igualdad y desigualdad de datos, donde las expresiones sobre los nodos serán extendidas con nominales (símbolos de proposición o etiquetas que son verdaderos sólo en un nodo), y las expresiones sobre los caminos serán extendidas con un operador de satisfacibilidad (permitiendo navegar a un nodo en particular). Esta lógica será denominada $\text{XPath}_{=}(\downarrow)$. Vale la pena destacar que no nos limitaremos a trabajar sobre árboles, si no que nuestro enfoque será más general por lo que trabajaremos sobre la clase de todos los modelos.

El cálculo de tableaux que introduciremos permite decidir si una fórmula es satisfacible o no, al mismo tiempo que nos permite construir un modelo de la misma en caso de ser posible. La construcción del modelo es particularmente interesante para la búsqueda de contraejemplos: se niega la fórmula que se presume insatisfacible, se ejecuta el tableaux, y si la fórmula negada es satisfacible podemos extraer un modelo que resulta un contraejemplo de la fórmula original.

# Capítulo 2

# Lógicas Modales y XPath

## 2.1 Una breve introducción a las lógicas modales e híbridas

Las lógicas modales son lenguajes diseñados para describir y razonar sobre estructuras relacionales, es decir estructuras tales como grafos y árboles. A continuación introduciremos la lógica modal básica $\mathsf{K}$ y la lógica híbrida básica $\mathcal{H}(@)$. El objetivo es establecer una notación común, además de destacar la relación entre XPath y estas lógicas.

**Definición 2.1.1.** Sea $\mathsf{PROP}$ un conjunto infinito contable de símbolos proposicionales. El conjunto $\mathsf{Form}$ de fórmulas de $\mathsf{K}$ sobre $\mathsf{PROP}$ está definido como:

$$\mathsf{Form} ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond\varphi,$$

donde $p \in \mathsf{PROP}$ y $\varphi, \psi \in \mathsf{Form}$. Otros operadores se definen de la forma usual: $\bot$ es $p \wedge \neg p$, $\top$ es $\neg\bot$, $\varphi \vee \psi$ es $\neg(\neg\varphi \wedge \neg\psi)$ y $\Box\varphi$ es una abreviación de $\neg\Diamond\neg\varphi$.

Las fórmulas del lenguaje modal básico son interpretadas en estructuras relacionales. Pueden ser vistas como grafos dirigidos etiquetados, los cuales son llamados *Modelos de Kripke* [26].

**Definición 2.1.2.** Un *Modelo de Kripke* es una tupla $\mathcal{M} = \langle M, \rightarrow, V\rangle$, donde:

- $M$ es un conjunto no vacío cuyos elementos llamados puntos, estados o nodos;
- $\rightarrow \ \subseteq W^2$ es la relación de accesibilidad;
- $V : \mathsf{PROP} \rightarrow 2^W$ es la función de valuación.

Sea $w$ un estado en $\mathcal{M}$, el par $(\mathcal{M}, w)$ es un (*pointed model*); usualmente se escribirá $\mathcal{M}, w$ para denotar un pointed model.

A continuación introduciremos la semántica. Los operadores de la lógica $\mathsf{K}$ describen propiedades de los modelos, lo que significa que las fórmulas se evalúan en algún punto específico.

**Definición 2.1.3.** Dado un *pointed model* $\mathcal{M}, w$ y una fórmula $\varphi$ decimos que $\mathcal{M}, w$ satisface $\varphi$ ($\mathcal{M}, w \models \varphi$) cuando:

$$\begin{array}{rll} \mathcal{M}, w \models p & \text{sii} & w \in V(p) \\ \mathcal{M}, w \models \neg\varphi & \text{sii} & \mathcal{M}, w \not\models \varphi \\ \mathcal{M}, w \models \varphi \wedge \psi & \text{sii} & \mathcal{M}, w \models \varphi \text{ y } \mathcal{M}, w \models \psi \\ \mathcal{M}, w \models \Diamond\varphi & \text{sii} & \exists v \in M \text{ t.q. } w \rightarrow v \text{ y } \mathcal{M}, v \models \varphi \end{array}$$

Una fórmula $\varphi$ de $\mathsf{K}$ *es satisfacible* si existe un *pointed model* $\mathcal{M}, w$ tal que $\mathcal{M}, w \models \varphi$.

Una fórmula de $\mathsf{K}$ describe propiedades estructurales de los modelos relacionales, del mismo modo que el fragmento de XPath correspondiente a la navegación. Por este motivo resulta posible aplicar las técnicas provenientes de lógica modal para investigar XPath, y luego extenderlas para el manejo de datos. En nuestro caso definiremos un cálculo de tableaux que permitirá decidir si una fórmula de $\text{XPath}_=$ es satisfacible. Además, el tableaux nos permitirá encontrar un modelo de la fórmula.

De la misma manera que para lógica modal, es posible extender la lógica con *operadores híbridos*, los cuales resultan útiles a lo hora de diseñar el cálculo. La lógica híbrida básica $\mathcal{H}(@)$ [6] es la lógica $\mathsf{K}$ extendida con nominales, y operadores de satisfacción (permitiendo navegar a algún nodo en particular).

**Definición 2.1.4.** Sea $\mathsf{NOM}$ un conjunto infinito contable de nominales tal que $\mathsf{NOM} \cap \mathsf{PROP} = \emptyset$. El conjunto de fórmulas híbridas $\mathsf{HForm}$ está definido como

$$\mathsf{HForm} ::= p \mid i \mid \neg\varphi \mid \varphi \wedge \psi \mid \Diamond\varphi \mid @_i\varphi,$$

donde $i \in \mathsf{NOM}$, $p \in \mathsf{PROP}$ y $\varphi, \psi \in \mathsf{HForm}$.

**Definición 2.1.5.** Un *modelo híbrido* es una tupla $\mathcal{M} = \langle M, \rightarrow, V, nom\rangle$, donde $\langle M, \rightarrow, V\rangle$ es un modelo de *Kripke*, y $nom : \mathsf{NOM} \rightarrow M$.

Sea $\mathcal{M}$ un modelo híbrido y $m$ un estado en $M$, la semántica de los operadores híbridos está dada por:

$$\begin{array}{rcl} \mathcal{M}, m \models i & \text{sii} & nom(i) = m \\ \mathcal{M}, m \models @_i\varphi & \text{sii} & \mathcal{M}, nom(i) \models \varphi. \end{array}$$

En la literatura existen diversas discusiones acerca de por qué agregar operadores híbridos mejora la teoría de prueba de una lógica modal (ver por ejemplo [6]). En particular, los nominales y los operadores de satisfacción son muy útiles para implementar directamente en el lenguaje formal metodologías usadas por *tableaux etiquetados* [29].

## 2.2 XPath en la web

XPath es el resultado de un esfuerzo para proveer una sintaxis y semántica común para las funcionalidades compartidas entre *XSL Transformations (XSLT)* y *XPointer*. El propósito principal de XPath es "direccionar" las partes de un documento XML. Es por ello que provee funcionalidades básicas para la manipulación de cadenas, números y Booleanos. XPath usa una sintaxis compacta, *no-XML*, para facilitar su uso sin necesidad de *URIs* y atributos XML. XPath opera sobre la estructura abstracta y lógica de un documento XML, en lugar de actuar sobre la sintaxis. Además está diseñado para tener un subconjunto natural que se pueda utilizar para testear si un nodo coincide con algún patrón específico. Este uso de describe en XSLT [17]. En la Figura 2.2 se puede visualizar un ejemplo de un documento XML, correpondiente al stock de un comercio de productos electrónicos.

Es importante por lo tanto contar con un lenguaje para hablar de este tipo de estructuras. Por ejemplo, resulta interesante escribir propiedades del tipo:

- El nombre de un producto, no puede ser igual a su categoría.
- Todos los identificadores de producto bajo una categoría son diferentes.
- Cada producto en stock está clasificado bajo una categoría.

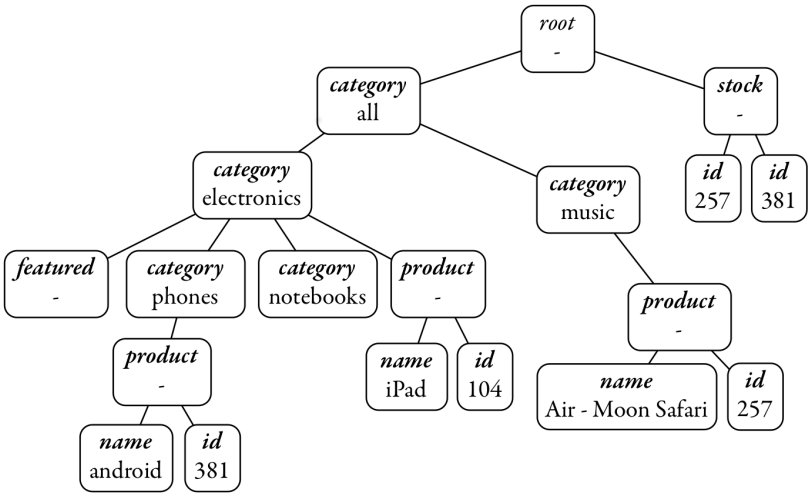


Figura 2.1: Ejemplo de documento XML.

XPath modela un documento XML como un árbol de nodos. Existen diferentes tipos de nodos, incluyendo nodos de elementos, nodos de atributos y nodos de texto. Algunos tipos de nodos también tienen nombres, así, el nombre de un nodo se modela como un par formado por una parte local y un URI posiblemente nulo del espacio de nombres; esto se llama *nombre expandido* [18].

## 2.3 El lenguaje formal HXPath$_=(\downarrow)$

En este capítulo se introduce la sintaxis y semántica para la lógica llamada *Hybrid Downward XPath* (abreviada como HXPath$_=(\downarrow)$).

En esta tesis trabajamos con un fragmento de XPath que corresponde a la parte de navegación de XPath1.0, con igualdad y desigualdad de datos, y extendida con nominales y el operador híbrido @. HXPath$_=(\downarrow)$ es un lenguaje con dos clases de fórmulas. Las *path expressions* o expresiones de caminos (que denotamos con $\alpha, \beta, \gamma$), y las *node expressions* o expresiones de nodos (que denotamos con $\varphi, \psi, \eta$), y se definen por recursión mutua. A continuación introducimos formalmente la sintaxis del lenguaje HXPath$_=(\downarrow)$.

**Definición 2.3.1** (Sintaxis)**.** El conjunto PExp de path expressions y NExp de node expressions del lenguaje HXPath$_=(\downarrow)$ se define recursivamente como sigue:

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

donde $p \in \mathsf{PROP}$, $i \in \mathsf{NOM}$, $\alpha, \beta \in \mathsf{PExp}$, $\varphi, \psi \in \mathsf{NExp}$ y $\lambda$ es la node expression vacía. El conjunto Exp de expresiones o fórmulas de HXPath$_=(\downarrow)$ está definido como $\mathsf{NExp} \cup \mathsf{PExp}$.

Se usarán los símbolos $i$, $j$, $k$, $n$, $m$ para denotar a los nominales; $p$, $q$, $r$, para los símbolos proposicionales; $\alpha$, $\beta$, $\gamma$, $\delta$ para las path expressions; $\varphi$, $\psi$ para las node expressions y $\varepsilon$ para expresiones arbitrarias.

Notar que las path expressions ocurren en las node expressions dentro de *fórmulas de comparación de datos* de la forma $\langle \alpha = \beta \rangle$ y $\langle \alpha \neq \beta \rangle$, mientras que las node expressions ocurren en path expressions en *test de fórmulas* de la forma $[\varphi]$.

Se empleará el símbolo $*$ para denotar a los símbolos $=$ y $\neq$. Los operadores Booleanos que no se muestran se definen de manera usual.

**Definición 2.3.2.** Sea $\alpha$ una path expression, $\varphi$ una node expression, $i \in \mathsf{NOM}$, y $p \in \mathsf{PROP}$, definimos las siguientes equivalencias:

| **Node Expressions** | | |
|---|---|---|
| $\top$ | $\equiv$ | $p \vee \neg p$ |
| $\bot$ | $\equiv$ | $\neg\top$ |
| $\langle\alpha\rangle\varphi$ | $\equiv$ | $\langle\alpha[\varphi] = \alpha[\varphi]\rangle$ |
| $[\alpha]\varphi$ | $\equiv$ | $\neg\langle\alpha\rangle\neg\varphi$ |
| $@_i\varphi$ | $\equiv$ | $\langle@_i\rangle\varphi$ |

Las fórmulas de la forma $@_i\varphi$ para $i \in \mathsf{NOM}$ y $\varphi$ una node expression, serán llamadas *at-fórmulas* o *fórmulas con prefijo* (intuitivamente, expresan que $\varphi$ vale en el nodo denotado por $i$). Las *at-formulas* tendrán un rol importante en el cálculo de tableaux que se introducirá más adelante.

Notar que siguiendo la notación estándar para la lógica en XPath y la lógica modal, el operador [ ] está sobrecargado: para una node expression $\varphi$ y una path expression $\alpha$, las dos expresiones $[\alpha]\varphi$ y $[\varphi]\alpha$ son expresiones válidas; la primera es una node expression donde $[\alpha]$ es la modalidad que representa al box, la segunda es una path expression donde $[\varphi]$ es un test.

Ahora definimos las estructuras que serán usadas para evaluar las fórmulas en este lenguaje.

**Definición 2.3.3.** Sea $\mathsf{PROP}$ conjunto infinito contable de símbolos proposicionales, sea $\mathsf{NOM}$ conjunto infinito contable de nominales tal que $\mathsf{NOM} \cap \mathsf{PROP} = \emptyset$, y sea $\mathsf{ATOM} = \mathsf{PROP} \cup \mathsf{NOM}$ conjunto de fórmulas atómicas (atoms para abreviar).

Un *modelo híbrido de datos concreto* es una tupla $\mathcal{M} = \langle M, D, \rightarrow, label, nom, data\rangle$, donde:

- $M$ es un conjunto no vacío de elementos,
- $D$ es un conjunto finito no vacío de datos,
- $\rightarrow \subseteq M^2$ es una relación de accesibilidad,
- $label : M \rightarrow 2^{\mathsf{PROP}}$ es una función de labeling,
- $nom : \mathsf{NOM} \rightarrow M$ es una función que asigna nominales a determinados elementos,
- $data : M \rightarrow D$ es una función que asigna un dato a cada nodo del modelo.

Un *modelo híbrido de datos abstracto* es una tupla $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom\rangle$, donde:

- $M$ es un conjunto no vacío de elementos,
- $\sim \subseteq M^2$ es una relación de equivalencia entre los elementos de $M$,
- $\rightarrow \subseteq M^2$ es una relación de accesibilidad,
- $label : M \rightarrow 2^{\mathsf{PROP}}$ es una función de etiquetado,
- $nom : \mathsf{NOM} \rightarrow M$ es una función que asigna nominales a determinados elementos.

Un modelo abstracto para HXPath$_=(\downarrow)$ puede verse como un modelo híbrido para $\mathcal{H}(@)$ extendido con una relación de equivalencia $\mathcal{H}(@)$; en la literatura sobre XPath, se denomina a la función de valuación $V$ como *labelling function*.

Los modelos con datos son comunmente usados en aplicaciones donde podemos encontrar datos de un alfabeto infinito (e.g., strings alfabéticos) asociados a los nodos de una base de datos semi-estructurada. Es fácil notar que cada modelo de datos está asociado, equivalentemente el modelo abstracto de datos donde los datos son reemplazados por relaciones de equivalencia que establecen un link de todos los nodos con el mismo dato. Vice-versa, cada modelo abstracto de datos puede ser "concretizado" asignando a cada nodo la clase de equivalencia con el mismo dato.

**Definición 2.3.4** (Semántica)**.** Sea $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom\rangle$ un modelo abstracto con datos, y $x, y \in M$. Definimos la semántica de HXPath$_=(\downarrow)$ de la siguiente manera:

$$
\begin{array}{rll}
\mathcal{M}, x, y \models \downarrow & \text{sii} & x \rightarrow y \\
\mathcal{M}, x, y \models @_i & \text{sii} & nom(i) = y \\
\mathcal{M}, x, y \models [\varphi] & \text{sii} & x = y \text{ y } \mathcal{M}, x \models \varphi \\
\mathcal{M}, x, y \models \alpha\beta & \text{sii} & \text{existe un } z \in M \text{ s.t.} \\
 & & \mathcal{M}, x, z \models \alpha \text{ y } \mathcal{M}, z, y \models \beta \\
\mathcal{M}, x, y \models \alpha \cup \beta & \text{sii} & \mathcal{M}, x, y \models \alpha \text{ o } \mathcal{M}, x, y \models \beta \\
 & & \\
\mathcal{M}, x \models p & \text{sii} & p \in label(x) \\
\mathcal{M}, x \models i & \text{sii} & nom(i) = x \\
\mathcal{M}, x \models \neg\varphi & \text{sii} & \mathcal{M}, x \not\models \varphi \\
\mathcal{M}, x \models \varphi \wedge \psi & \text{sii} & \mathcal{M}, x \models \varphi \text{ y } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \langle \alpha = \beta \rangle & \text{sii} & \text{existen } y, z \in M \text{ t.q.} \\
 & & \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \text{ y } y \sim z \\
\mathcal{M}, x \models \langle \alpha \neq \beta \rangle & \text{sii} & \text{existen } y, z \in M \text{ t.q.} \\
 & & \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \text{ y } y \not\sim z \\
\mathcal{M}, x \models \langle \lambda = \alpha \rangle & \text{sii} & \text{existe } y \in M \text{ t.q. } \mathcal{M}, x, y \models \alpha \text{ y } x \sim y \\
\mathcal{M}, x \models \langle \lambda \neq \alpha \rangle & \text{sii} & \text{existe } y \in M \text{ t.q. } \mathcal{M}, x, y \models \alpha \text{ y } x \not\sim y.
\end{array}
$$

Para una node expression $\varphi$, escribimos $\mathcal{M}, u \models \varphi$ para denotar $u \in [\![\alpha]\!]^{\mathcal{M}}$, y en ese caso decimos que $\mathcal{M}, u$ satisface $\varphi$ o que $\varphi$ es cierta en $\mathcal{M}, u$. En el mismo sentido, para una path expression $\alpha$ , escribimos $\mathcal{M}, u, v \models \alpha$ para denotar $(u, v) \in [\![\alpha]\!]^{\mathcal{M}}$, y decimos que $\mathcal{M}, u, v$ satisface $\alpha$ o que $\alpha$ es cierta en $\mathcal{M}, u, v$. Sean $\varphi, \psi$ node expressions de XPath$_=$, decimos que son equivalentes (notación $\varphi \equiv \psi$) sí y sólo sí $[\![\varphi]\!]^{\mathcal{M}} = [\![\psi]\!]^{\mathcal{M}}$ para todos los modelos $\mathcal{M}$. Del mismo modo, sean $\alpha, \beta$ path expressions de XPath$_=$, decimos que son **equivalentes** (notación $\alpha \equiv \beta$) sí y sólo sí $[\![\alpha]\!]^{\mathcal{M}} = [\![\beta]\!]^{\mathcal{M}}$ para todos los modelos $\mathcal{M}$.

Como corolario de la Definición 2.3.4, las abreviaciones $\langle\alpha\rangle\varphi$, $[\alpha]\varphi$ y $@_i\varphi$ tienen el significado habitual.

$$
\begin{array}{rll}
\mathcal{M}, x \models \langle\alpha\rangle\varphi & \text{sii} & \text{existen } y \in M \text{ tq. } \mathcal{M}, x, y \models \alpha \text{ y } \mathcal{M}, y \models \varphi \\
\mathcal{M}, x \models [\alpha]\varphi & \text{sii} & \text{para todo } y \in M,\ \mathcal{M}, x, y \models \alpha \text{ implica } \mathcal{M}, y \models \varphi \\
\mathcal{M}, x \models @_i\varphi & \text{sii} & \mathcal{M}, nom(i) \models \varphi
\end{array}
$$

**Ejemplo 2.3.1.** A continuación se describen algunos ejemplos de node expressions sobre el árbol $\mathcal{T}$ de la Figura 2.2:

- $\varphi_1 = \langle\downarrow[a] = \downarrow[b]\rangle$: "nodos con un hijo etiquetado con $a$ y un hijo etiquetado con $b$, que tienen el mismo dato", entonces $[\![\varphi_1]\!]^{\mathcal{T}} = \{z\}$.

- $\varphi_2 = \langle \downarrow[a] \neq \downarrow[a] \rangle$: "nodos con dos hijos con etiqueta $a$ pero con diferente valor de dato", entonces $[\![\varphi_2]\!]^{\mathcal{T}} = \{z\}$.

- $\varphi_3 = \langle \downarrow\downarrow = \downarrow\downarrow \rangle$: "nodos con un camino descendente de tamaño 2", entonces $[\![\varphi_3]\!]^{\mathcal{T}} = \{x\}$.

A continuación algunos ejemplos de path expressions sobre el árbol de la Figura 2.1:

- $\alpha_1 = [a]\downarrow\downarrow[b]$: "caminos descendentes de tamaño dos que comienzan en un nodo con etiqueta a y terminan en un nodo con etiqueta b", entonces $[\![\alpha_1]\!]^{\mathcal{T}} = \{(x, w)\}$.

- $\alpha_2 = [a]\downarrow[a] \cup [a]\downarrow\downarrow[a]$: "caminos descendentes de tamaño uno o dos, que comienzan y terminan en un nodo con etiqueta $a$" entonces $[\![\alpha_2]\!]^{\mathcal{T}} = \{(x, y), (x, u), (x, v)\}$.

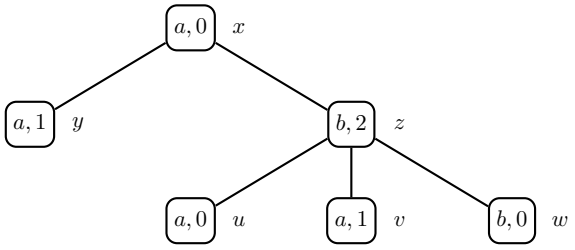


Figura 2.2: Árbol con datos. Las letras representan etiquetas, los números representan datos.

La adición de operadores híbridos a XPath, incrementa su poder expresivo. Los siguientes ejemplos servirán para ilustrar este hecho.

**Ejemplo 2.3.2.** Se listan a continuación algunas expresiones de HXPath$_=(\downarrow)$ junto a su significado intuitivo:

| | |
|---|---|
| $[i]\alpha$ | El nodo actual se denomina $i$ y existe un camino $\alpha$ a partir de tal nodo. |
| $\alpha[i]$ | Existe un camino $\alpha$ entre el nodo actual y el nodo nombrado con $i$. |
| $@_i\alpha$ | Existe un camino $\alpha$ entre el nodo nombrado con $i$ y algún otro nodo. |
| $\alpha@_i$ | Existe un camino $\alpha$ entre el nodo actual y algún otro nodo, la evaluación continúa en el nodo nombrado por $i$. |
| $\langle @_i = @_i \rangle$ | Siempre es válido. |
| $\langle [i] = [i] \rangle$ | El nodo actual está etiquetado con $i$. |
| $\langle @_i = @_j \rangle$ | El nodo nombrado por $i$ tiene el mismo dato que el nodo nombrado por $j$. |
| $\langle \alpha = @_i\beta \rangle$ | Existe un nodo accesible desde el nodo actual por el camino $\alpha$ que tiene el mismo dato que el nodo nombrado con $i$, accesible por el camino $\beta$. |

Cabe destacar la diferencia entre $\langle \alpha \neq \beta \rangle$ y $\neg\langle \alpha = \beta \rangle$. La primera expresión establece que hay un camino $\alpha$ desde el punto actual a algún punto $y$, y un camino $\beta$ a algún punto $z$ tal que $y \not\sim z$ (es decir, tienen diferentes datos). La segunda expresión dice que al menos uno de los caminos $\alpha$, $\beta$ no existe o que existen pero que ningún par de nodos alcanzados almacena el mismo dato. Entonces mientras que en la primera expresión los caminos existen necesariamente, en la segunda puede que esto no ocurra. Sin embargo, estas fórmulas son equivalentes si $\alpha = @_i$ y $\beta = @_j$, con $i, j \in \mathsf{NOM}$.

La siguiente proposición muestra algunas propiedades semánticas que se usarán más adelante en la prueba de completitud. Para probar las mismas basta sólo con usar la definición de $\models$.

**Proposición 2.3.1.** Sea $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom \rangle$ un modelo abstracto con datos, $x, y \in M$, $\alpha, \beta$ path expressions arbitrarias, y $i, j \in \mathsf{NOM}$. Luego

1. $\mathcal{M}, x \models \langle @_i \alpha * @_j \beta \rangle$ sii $\mathcal{M}, y \models \langle @_i \alpha * @_j \beta \rangle$,
2. $\mathcal{M}, x \models i$ y $\mathcal{M}, x \models \langle \alpha * \beta \rangle$ entonces $\mathcal{M}, x \models \langle @_i \alpha * \beta \rangle$,
3. $nom(i) \sim nom(j)$ sii $\mathcal{M}, x \models \langle @_i = @_j \rangle$,
4. $\mathcal{M}, x \models \langle @_i @_j \alpha * \beta \rangle$ sii $\mathcal{M}, x \models \langle @_j \alpha * \beta \rangle$.

Esta proposición establece que:

1. Las path expressions que comienzan con un nominal son globales y no dependen del punto en el que se evalúen.
2. Si el punto de evaluación es la interpretación de un nominal, es posible agregar cada nominal al comienzo de una path expression sin efecto.
3. Dos nominales pertenecen a la misma clase de equivalencia sii tienen el mismo dato.
4. Agregar un nominal a una path expression que comienza con otro nominal no tiene efectos.

# Capítulo 3

# Un cálculo de tableaux para XPath

## 3.1 Introducción al cálculo de tableaux

El método de tableaux es un procedimiento formal de prueba para diferentes tipos de lógicas, lógicas, pero manteniendo ciertas características. Primero, es utilizado de manera habitual como un procedimiento de refutación: para mostrar la validez de una determinada fórmula $\varphi$ comenzamos con alguna expresión sintáctica destinada a asegurar lo contrario. Luego, la expresión que asegura que la fórmula $\varphi$ es falsa se descompone sintácticamente, generalmente separándola en diversos casos. Esta parte del procedimiento del tableaux, la etapa de expansión, generalmente implica pasar de fórmulas a subfórmulas. Por último, hay reglas para el cierre de las ramas: condiciones de imposibilidad basadas en la sintaxis (usualmente conocidas como *condiciones de clash*). Si cada rama se cierra, se dice que el tableaux *está cerrado*. Un tableaux cerrado que empiece con una expresión afirmando que $\varphi$ no es válida es una prueba de la validez de $\varphi$.

Otra manera de pensar a los tableaux, dentro de un enfoque semántico, es como un procedimiento de búsqueda de modelos que cumplan ciertas condiciones. Cada rama del tableaux puede ser considerada como una descripción parcial del modelo. Varios teoremas fundamentales de la teoría de modelos tienen pruebas que pueden ser extraídas como resultado de la aplicación del método de tableaux. Los tableaux se utilizan a veces en pruebas de teoremas automatizadas para la generación de contraejemplos. Es sencillo ver la conexción entre los dos roles de un tableaux, como procedimiento de prueba y como procedimiento de búsqueda de modelo. Si usamos el tableaux para buscar un modelo en el cual la fórmula $\varphi$ es falsa, y obtenemos un tableaux cerrado, tal modelo no existe entonces $\varphi$ es verdadera [19].

## 3.2 Tableaux para HXPath$_=(\downarrow)$

En esta sección definiremos un cálculo de tableaux para el lenguaje HXPath$_=(\downarrow)$. En particular, comenzaremos por introducir la definición de las reglas del tableaux y las condiciones de clash.

Además de las fórmulas de HXPath$_=(\downarrow)$, las reglas de tableaux contienen *fórmulas de accesibilidad* de la forma $nRm$, donde $n, m \in \mathsf{NOM}$. La interpretación de $nRm$ es que el nodo denotado por $m$ es accesible desde el nodo denotado por $n$ por la relación de accesibilidad $\rightarrow$. Se usará el término *fórmula* para denotar una fórmula de HXPath$_=(\downarrow)$ o una fórmula de accesibilidad. El tableaux para este cálculo es un árbol finito donde cada nodo está etiquetado por una fórmula y las aristas representan la aplicación de las reglas del tableaux. Para chequear la satisfacibilidad de una *node expression* $\varphi$ se inicializa el tableaux con $@_i\varphi$,

para $i \notin \mathsf{NOM}(\varphi)$. Para chequear satisfacibilidad de una *path expression* $\alpha$, se inicializa el tableaux con $i\langle\alpha\rangle\top$, para $i \notin \mathsf{NOM}(\alpha)$.

Una rama $B$ de un tableaux contiene un *clash* si se da algunas de las siguientes condiciones:

1. $@_i[\lambda \neq \lambda] \in B$
2. $@_i p,\ @_i \neg p \in B$
3. $@_i\langle\lambda \neq \lambda\rangle \in B$
4. $@_i[\lambda \neq @_i] \in B$
5. $@_i\langle\lambda \neq @_i\rangle \in B$.

Un *tableaux saturado* es un tableaux en el cual no pueden aplicarse más reglas en ninguna de sus ramas. Una *rama saturada* es una rama de un tableaux saturado. Se llamará a una rama de un tableaux *cerrada* cuando contenga una condición de clash, en otro caso se la llamará rama *abierta*. Un *tableaux cerrado* es aquel en el que todas sus ramas están cerradas y un *tableaux abierto* es aquel en el que hay al menos una rama abierta.

En la Figura 3.1 se introducen las reglas correspondientes a los operadores Booleanos. En la Figura 3.2, las reglas determinan que las relaciones inducidas por el operador híbrido @ y por la igualdad de datos son relaciones de equivalencia.

$$\frac{@_i(\varphi \wedge \psi)}{\begin{array}{l}@_i\varphi \\ @_i\psi\end{array}}\ (\wedge) \qquad \frac{@_i(\varphi \vee \psi)}{@_i\varphi \mid @_i\psi}\ (\vee)$$

Figura 3.1: Reglas de Tableau para Operadores Booleanos.

$$\frac{}{\begin{array}{l}@_i\langle\lambda = @_i\rangle \\ @_i\langle\lambda = [i]\rangle\end{array}}\ (Reflex.) \qquad \frac{@_i\langle\lambda = @_j\rangle \quad @_j\langle\lambda = @_k\rangle}{@_i\langle\lambda = @_k\rangle}\ (Transit.)$$

$$\frac{@_i\langle\lambda = [j]\rangle}{@_j\langle\lambda = [i]\rangle}\ (Sim.test) \qquad \frac{@_i\langle\lambda = [j]\rangle \quad @_j\langle\lambda = [k]\rangle}{@_i\langle\lambda = [k]\rangle}\ (Transit.test)$$

$$\frac{@_i\langle\lambda = @_j\rangle \quad @_i\varphi}{@_j\varphi}\ (Copy) \qquad \frac{@_i\langle\lambda = @_j\rangle}{@_j\langle\lambda = @_i\rangle}\ (Simetria) \qquad \frac{@_i j}{@_i\langle\lambda = @_j\rangle}\ (Data_{=})$$

Figura 3.2: Reglas equivalencia para @ y $=$.

A continuación se introducen las reglas correspondientes a operadores modales, es decir las expresiones de caminos. En la Figura 3.3 definimos las reglas para el tratamiento de node expressions del tipo $\langle\alpha \star \beta\rangle$, con $\star \in \{=, \neq\}$. En Figura 3.4 se definen las reglas para node expression del tipo $[\alpha \star \beta]$. Para cada una de ellas se definen reglas dependiente del tipo de la path expression al comienzo de la parte izquierda.

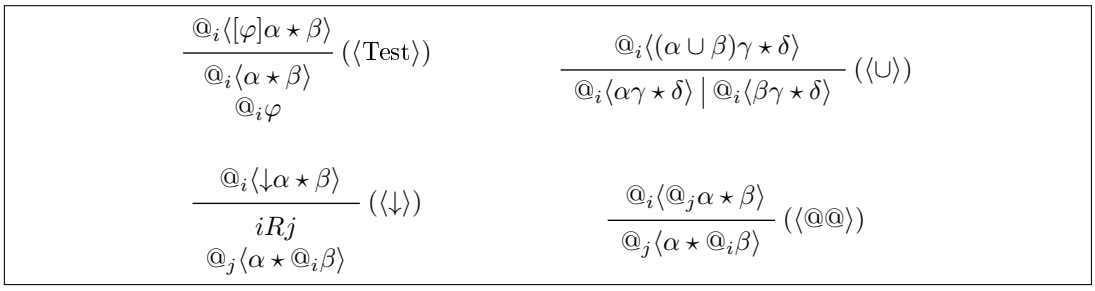


Figura 3.3: Reglas diamante.

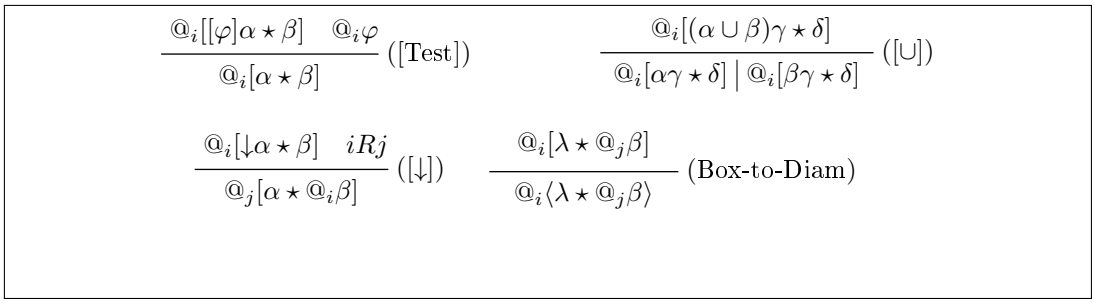


Figura 3.4: Reglas sobre box.

Al finalizar el análisis de la parte izquierda de una node expression, es necesario comenzar con la parte derecha. Las reglas que consideran la parte derecha se definen en la Figura 3.5. Notar que todas las reglas analizan expresiones de la forma $\langle\lambda \star \alpha\rangle$, es decir expresiones con "diamante". Esto es debido a que cuando las expresiones con "box" contienen una parte izquierda vacía (es decir, igual a $\lambda$), se las transforma es una expresión con diamante ya que son equivalentes.

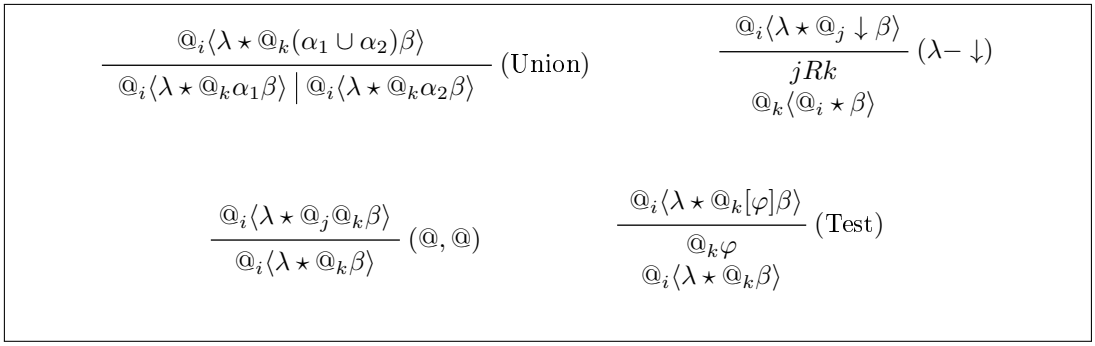


Figura 3.5: Reglas con lado izquierdo $\lambda$.

Realizando una simple inspección en las reglas, es posible determinar que el cálculo es correcto:

**Teorema 1.** *Sea $\varepsilon \in$ Exp. Si $\varepsilon$ es satisfacible entonces todo Tableau($\varepsilon$) contiene una rama abierta.*

A continuación introduciremos algunos ejemplos del uso del cálculo definido previamente.

**Ejemplo 1.** *Tableaux para* $@_i\langle\downarrow\downarrow = @_j\downarrow\rangle$.

| | | |
|---|---|---|
| (0) | $@_i\langle\downarrow\downarrow = @_j\downarrow\rangle$ | |
| (1) | $iRk$ | $(\langle\downarrow\rangle, 0)$ |
| (2) | $@_k\langle\downarrow = @_i@_j\downarrow\rangle$ | $(\langle\downarrow\rangle, 0)$ |
| (3) | $kRl$ | $(\langle\downarrow\rangle, 2)$ |
| (4) | $@_l\langle\lambda = @_k@_i@_j\downarrow\rangle$ | $(\langle\downarrow\rangle, 2)$ |
| (5) | $@_l\langle\lambda = @_i@_j\downarrow\rangle$ | $(@@, 4)$ |
| (6) | $@_l\langle\lambda = @_j\downarrow\rangle$ | $(@@, 5)$ |
| (7) | $jRm$ | $(\lambda - \downarrow, 6)$ |
| (8) | $@_m\langle@_l = \lambda\rangle$ | $(\lambda - \downarrow, 6)$ |
| (9) | $@_l\langle\lambda = @_m\rangle$ | $(\langle@@\rangle, 8)$ |

*La fórmula* $@_i\langle\downarrow\downarrow = @_j\downarrow\rangle$ *establece que en el nominal i vale que el sucesor en dos pasos tiene el mismo dato que el sucesor en un paso en el nominal j. El tableaux obtenido está saturado y abierto, luego la fórmula de entrada es satisfacible.*

**Ejemplo 2.** *Veamos ahora un ejemplo para la fórmula* $@_i\langle\downarrow\downarrow\cup\downarrow@_k = @_j\downarrow\rangle$. *Debemos analizar las fórmulas* $@_i\langle\downarrow\downarrow = @_j\downarrow\rangle$ *y* $@_i\langle\downarrow@_k = @_j\downarrow\rangle$ *que se derivan de aplicar la regla de la unión dada en esta sección. Como vimos en el ejemplo anterior la primera es satisfacible, veamos la segunda fórmula.*

| | | |
|---|---|---|
| (0) | $@_i\langle\downarrow@_k = @_j\downarrow\rangle$ | |
| (1) | $iRl$ | $(\langle\downarrow\rangle, 1)$ |
| (2) | $@_l\langle@_k = @_i@_j\downarrow\rangle$ | $(\langle\downarrow\rangle, 1)$ |
| (3) | $@_l\langle@_k = @_j\downarrow\rangle$ | $(@@, 2)$ |
| (4) | $@_k\langle\lambda = @_l@_j\downarrow\rangle$ | $(\langle@@\rangle, 3)$ |
| (5) | $@_k\langle\lambda = @_j\downarrow\rangle$ | $(@@, 4)$ |
| (6) | $jRm$ | $(\lambda - \downarrow, 5)$ |
| (7) | $@_m\langle@_k = \lambda\rangle$ | $(\lambda - \downarrow, 5)$ |
| (8) | $@_k\langle\lambda = @_m\rangle$ | $(@@, 7)$ |

*Los dos tableaux obtenidos están saturados y abiertos, por lo tanto la fórmula* $@_i\langle\downarrow\downarrow \cup \downarrow@_k = @_j\downarrow\rangle$ *es satisfacible.*

**Ejemplo 3.** *Veamos ahora un ejemplo en el cual el tableaux termina en un clash. Como entrada se tiene la fórmula* $@_ij \wedge @_i\langle\downarrow = @_k\rangle \wedge @_j[\downarrow \neq @_k]$.

| | | |
|---|---|---|
| (0) | $@_ij \wedge @_i\langle\downarrow = @_k\rangle \wedge @_j[\downarrow \neq @_k]$ | |
| (1) | $@_ij$ | $(\wedge, 0)$ |
| (2) | $@_i\langle\downarrow = @_k\rangle$ | $(\wedge, 0)$ |
| (3) | $@_j[\downarrow \neq @_k]$ | $(\wedge, 0)$ |
| (4) | $@_i\langle\lambda = @_j\rangle$ | $(Data_=, 1)$ |
| (5) | $iRl$ | $(\langle\downarrow\rangle, 2)$ |
| (6) | $@_l\langle\lambda = @_i@_k\rangle$ | |
| (7) | $@_l\langle\lambda = @_k\rangle$ | $(@@, 6)$ |
| (8) | $@_j\langle\downarrow = @_k\rangle$ | $(Copy\ 1, 7)$ |
| (9) | $jRm$ | $(\langle\downarrow\rangle, 8)$ |
| (10) | $@_m\langle\lambda = @_k\rangle$ | |
| (11) | $@_m[\lambda \neq @_k]$ | $([\downarrow], 3, 9)$ |
| (12) | $@_m\langle\lambda \neq @_k\rangle$ | $(Box\text{-}to\text{-}diam, 11)$ |
| (13) | $@_k\langle\lambda \neq @_k\rangle$ | $(Copy, 10, 12)$ |

*La línea 13 provoca un clash (condición 5), luego el tableaux es cerrado. Por lo tanto la fórmula* $@_i j \wedge @_i\langle\downarrow = @_k\rangle \wedge @_j[\downarrow \neq @_k]$ *no es satisfacible.*

# Capítulo 4

# Completitud

## 4.1 Construcción del modelo

En esta subsección introduciremos la noción de *modelo extraído*. Dada una rama de tableaux abirta y saturada a partir de una fórmula

**Definición 1** (Modelo extraído)**.** *Sea $B$ una rama abierta saturada. Definimos el modelo sintáctico abstracto $\mathcal{M}^B = \langle M, \sim, \rightarrow, label, nom\rangle$ de la siguiente manera:*

$i \equiv j$ *(también denotado $i \in [j]$) sii* $@_i\langle\lambda = [j]\rangle \in B$

$M = \{[k] \mid k \text{ aparece en } B\}$

$[i] \sim [j]$ *sii* $@_i\langle\lambda = @_j\rangle \in B$

$[i] \rightarrow [j]$ *sii* $iRj \in B$

$a \in label([k])$ *sii* $@_k a \in B$, *para* $a \in \mathsf{PROP}$

$nom(i) = [i]$, *para* $i \in \mathsf{NOM}$.

**Definición 2.** *Sean $\alpha, \beta$ path expressions y sea $\varphi$ una node expression definimos el tamaño de una fórmula como:*

$$\begin{array}{lcllcl} tam_i(@_i) & = & 2 & tam_d(@_i) & = & 1 \\ tam_l(\lambda) & = & 0 & tam_l(\downarrow) & = & 2 \\ tam_l([\varphi]) & = & 3 + tam_l(\varphi) & tam_l(\alpha \cup \beta) & = & tam_l(\alpha) + tam_l(\beta) \\ tam_l(\alpha\beta) & = & tam_l(\alpha) + tam_l(\beta) & tam_l(@_i\varphi) & = & 2 + tam_l(\varphi) \end{array}$$

*donde $l \in \{i, d\}$.*

$$tam(\langle\alpha \star \beta\rangle) = tam_i(\alpha) + tam_d(\beta)$$

Notar que definimos como $tam_i$ a la función tamaño aplicada al lado izquierdo de una expresión de la forma $\langle\alpha \star \beta\rangle$ y análogamente $tam_d$ será aplicada al lado derecho de dicha expresión.

A continuación introduciremos algunos resultados generales que usaremos en la siguiente subsección. Notar que los mismos no dependen del modelo extraído, pero los mismos son introducidos ya que en general se utilizan de manera implícita.

**Teorema 2.** *Si $\varphi$ es satisfacible entonces $@_i\varphi$ es satisfacible, para todo $i$ que no aparece en $\varphi$.*

*Proof.* Como $\varphi$ es satisfacible entonces existe un modelo $\mathcal{M}$ y un punto $w$ en el modelo tal que $\mathcal{M}, w \models \varphi$. Supongamos que $i$ es un nominal que pertenece a $\mathcal{M}$ y que $\mathcal{M}, w \not\models @_i\varphi$. Basta con tomar $\mathcal{M}'$ idéntico a $\mathcal{M}$ excepto que $i$ vale en $W$. Luego, como $i$ no aparece en $\varphi$, $\mathcal{M}', w \models @_i\varphi$, por lo tanto $@_i\varphi$ es satisfacible. □

**Teorema 3.** *Si $@_i\varphi$ es satisfacible entonces $\varphi$ es satisfacible.*

*Proof.* Es claro que si $@_i\varphi$ es satisfacible existe un par $\mathcal{M}, w$ tal que $\mathcal{M}, w \models @_i\varphi$. Luego, por la definición de @ existe un punto etiquetado por $i$ en el modelo tal que vale la fórmula $\varphi$. □

**Teorema 4.** *Si $@_i(\varphi \wedge \psi)$ es satisfacible entonces $\varphi$ y $\psi$ son satisfacibles.*

*Proof.* Sea $\mathcal{M}, w$ tal que $\mathcal{M}, w \models @_i(\varphi \wedge \psi)$. Entonces existe $v$ tal que $nom(i) = v$ y $\mathcal{M}, v \models \varphi \wedge \psi$. Luego $\varphi$ y $\psi$ son satisfacibles. □

**Teorema 5.** *$\varphi \vee \psi$ satisfacible $\Rightarrow @_i(\varphi \vee \psi)$ satisfacible.*

*Proof.* $\varphi \vee \psi$ satisfacible entonces $\varphi$ satisfacible o $\psi$ satisfacible. Si $\varphi$ satisfacible, por Teorema 3 tenemos que $@_i\varphi$ satisfacible. Análogo para $\psi$.

□

**Teorema 6.** *$@_i(\varphi \vee \psi)$ satisfacible $\Rightarrow \varphi$ satisfacible o $\psi$ satisfacible.*

*Proof.* Similar a Teorema 4. □

Trabajaremos con fórmulas en NNF (*Negation Normal Form*), donde la negación ocurre sólo sobre labels y nominales. Una fórmula es equivalente a su NNF (la demostración es una inducción estándar).

**Teorema 7.** *$\neg @_i\varphi$ satisfacible $\Leftrightarrow @_i\neg\varphi$ satisfacible*

*Proof.* Directo del hecho de que @ es self-dual. □

## 4.2 Completitud del cálculo

Este capítulo estará dedicado a la demostración de completitud del cálculo introducido en el Capítulo 3. Decimos que un cálculo de tableaux es *completo* si y solo si a partir de una rama abierta saturada para una fórmula $\varphi$ podemos encontrar un modelo para $\varphi$ (es decir, $\varphi$ es satisfacible). Dicho modelo será el modelo extraído que definimos en la sección anterior. Por este motivo debemos demostrar que el modelo extraído es un modelo híbrido de datos.

**Teorema 8.** *Si $B$ es una rama abierta saturada, entonces $\mathcal{M}^B$ es un modelo híbrido de datos.*

*Proof.* Probaremos que $\sim$ y $\equiv$ son relaciones de equivalencia.

Recordemos que $\sim \subseteq M^2$, definida como:

$$[i] \sim [j] \textit{ sii } @_i\langle\lambda = @_j\rangle \in B$$

Para probar que es una relación de equivalencia debemos mostrar que es reflexiva, simétrica y transitiva.

***Reflexividad***

Supongamos $[i] \sim [i]$. Luego por definición de $\sim$ tenemos que $@_i\langle\lambda = @_i\rangle \in B$, lo cual vale por la regla *Reflex.*

***Simetría***

Supongamos $[i] \sim [j]$, debemos probar que $[j] \sim [i]$

Como vimos anteriormente, $[i] \sim [j]$ sii $@_i\langle\lambda = @_j\rangle \in B$, apliquemos la regla de tableaux para este caso:

$$\frac{@_i\langle\lambda = @_j\rangle}{@_j\langle\lambda = @_i\rangle}\, Simetria$$

Luego por definición de $\sim$ concluimos que:

$$@_j\langle\lambda = @_i\rangle \in B \text{ sii } [j] \sim [i]$$

***Transitividad***

Debemos probar que si $[i] \sim [j]$ y $[j] \sim [k]$ entonces $[i] \sim [k]$. Tenemos entonces que $@_i\langle\lambda = @_j\rangle \in B$ y $@_j\langle\lambda = @_k\rangle \in B$.
Usando las reglas de tableaux:

$$\frac{@_i\langle\lambda = @_j\rangle \quad @_j\langle\lambda = @_k\rangle}{@_i\langle\lambda = @_k\rangle}\, Transitividad$$

Luego por definición de $\sim$ concluimos que:

$$@_i\langle\lambda = @_k\rangle \in B \text{ sii } [i] \sim [k]$$

Veamos ahora que la relación dada por $i \equiv j$ es una relación de equivalencia.

Recordemos que $i \equiv j$ sii $@_i\langle\lambda = [j]\rangle \in B$.

***Reflexividad***

Por la regla *Reflex.* tenemos que $@_i\langle\lambda = [i]\rangle \in B$, por lo que $i \equiv i$.

***Simetría***

Sea $i \equiv j$ probaremos que $j \equiv i$. Suponemos que $i \equiv j$, entonces por definición de modelo, tenemos que $@_i\langle\lambda = [j]\rangle \in B$. Por *Sim.test*, $@_j\langle\lambda = [i]\rangle \in B$. Por lo tanto, $j \equiv i$.

***Transitividad***

Supongamos que $i \equiv j$, $j \equiv k$ y veamos que $i \equiv k$. Por definición del modelo, tenemos $@_i\langle\lambda = [j]\rangle \in B$ y que $@_j\langle\lambda = [k]\rangle \in B$. Pero por *Transit.test*, también tenemos que $@_i\langle\lambda = [k]\rangle \in B$, luego $i \equiv k$. □

Hemos demostrado que el modelo $\mathcal{M}^B$ extraído a partir de una rama abierta y saturada $B$ para una fórmula $\varphi$ es efectivamente un modelo híbrido de datos. Por lo tanto, $\mathcal{M}^B$ servirá como evidencia de la satisfacibilidad de $\varphi$. El siguiente teorema establece la completitud del cálculo.

**Teorema 9.** *Sea $B$ una rama abierta y saturada, y sea $i \in$ NOM. Si $@_i\varphi \in B$ entonces $\mathcal{M}^B, [i] \models \varphi$*

*Proof.* La demostración es por inducción en el tamaño de la fórmula definido por la función *tam*. Los distintos casos son planteados de manera genérica usando el operador $\star$; cuando los casos $=$ y $\neq$ difieran sustancialmente, se expresarán dichas diferencias.

Los casos base, y los Booleanos son directos por una simple inspección en la definición del modelo, y por aplicación de hipótesis inductiva. Demostraremos lo casos inductivos.

*Caso* : $\varphi = \langle\downarrow \alpha \star \beta\rangle$

Si $@_i\langle\downarrow\alpha \star \beta\rangle \in B$, aplicando las reglas de nuestro cálculo obtenemos lo siguiente:

$$\frac{@_i\langle\downarrow\alpha \star \beta\rangle}{\begin{array}{c} i{\downarrow}j \\ @_j\langle\alpha \star @_i\beta\rangle \end{array}}\ (\langle\downarrow\rangle)$$

Veamos que $@_j\langle\alpha \star @_i\beta\rangle \in B$, para eso usaremos la definición de la función *tam*:

$tam(@_i\langle\downarrow\alpha \star \beta\rangle) = 2 + tam_i(\downarrow\alpha) + tam_d(\beta) = 2 + 2 + tam_i(\alpha) + tam_d(\beta)$ $(A)$
Por otro lado:

$tam(@_j\langle\alpha \star @_i\beta\rangle) = 2 + tam_i(\alpha) + tam_d(@_i\beta) = 2 + tam_i(\alpha) + 1 + tam_d(\beta)$ $(B)$

Como $(A) < (B)$, la regla $(\langle\downarrow\rangle)$ disminuye el tamaño de la fórmula, entonces por hipótesis inductiva $\mathcal{M}^B, [j] \models @_j\langle\alpha \star @_i\beta\rangle$.

Aplicando la semántica de nuestro modelo tenemos que existen $y, z \in \mathcal{M}$ tales que:

1. $\mathcal{M}^B, [j], y \models \alpha$,
2. $\mathcal{M}^B, [j], z \models @_i\beta$,
3. $y \sim z$.

Por (2), sabemos que existe un $q$ tal que $\mathcal{M}^B, [j], q \models @_i$ sii $nom(i) = q$ y $\mathcal{M}^B, q, z \models \beta$, entonces podemos decir que $\mathcal{M}^B, [i], z \models \beta$.

Tenemos hasta ahora:

(a) $\mathcal{M}^B, [i], z \models \beta$

(b) $\mathcal{M}^B, [j], y \models \alpha$

(c) $\mathcal{M}^B, [i], [j] \models \downarrow$

Usando (b) y (c) obtenemos $\mathcal{M}^B, [i], y \models \downarrow\alpha$. Luego, con (a) y (3), tenemos que:

$$\mathcal{M}^B, [i] \models \langle\downarrow\alpha = \beta\rangle.$$

Para $\langle\downarrow\alpha \neq \beta\rangle$ la demostración es análoga, cambiando $y \not\sim z$ en (3).

*Caso* : $\varphi = \langle @_j\alpha \star \beta\rangle$

Al igual que en el caso anterior, aplicaremos la regla de tableaux y la definición de tamaño:

$$\frac{@_i\langle @_j\alpha \star \beta\rangle}{@_j\langle\alpha \star @_i\beta\rangle}\,(@,@)$$

Veamos el tamaño de las fórmulas:

- $tam(@_i\langle @_j\alpha \star \beta\rangle) = 2 + 2 + tam_i(\alpha) + tam_d(\beta)$
- $tam(@_j\langle\alpha \star @_i\beta\rangle = 2 + 1 + tam_i(\alpha) + tam_d(\beta))$

Como $(@, @)$ disminuye el tamaño de la fórmula, entonces por hipótesis inductiva $\mathcal{M}^B, [j] \models @_j\langle\alpha \star @_i\beta\rangle$. Como $nom(j) = [j]$, entonces tenemos que $\mathcal{M}^B, [j] \models \langle\alpha \star @_i\beta\rangle$. Por definición de $\models$ tenemos:

1. existe $x$ tal que $\mathcal{M}^B, [j], x \models \alpha$,
2. existe $y$ tal que $\mathcal{M}^B, [j], y \models @_i\beta$, y
3. $x \sim y$.

De 2 obtenemos por un lado que existe $z$ tal que $\mathcal{M}^B, [j], z \models @_i$ sii $nom(i) = z$, y por otro lado $\mathcal{M}^B, z, y \models \beta$. Luego $\mathcal{M}^B, [i], y \models \beta$.

Por 1 y definición de *nom* tenemos que $\mathcal{M}^B, [j], x \models @_j\alpha$. Como @ es un operador global, entonces también $\mathcal{M}^B, [i], x \models @_j\alpha$. Por lo tanto, usando 3, tenemos que:

$$\mathcal{M}^B, [i] \models \langle @_j\alpha \star \beta\rangle \text{ sii (por } nom(i) \star [i]) \; \mathcal{M}^B, [i] \models @_i\langle @_j\alpha \star \beta\rangle.$$

*Caso* : $\varphi = \langle[\varphi]\alpha \star \beta\rangle$

Apliquemos la regla de tableaux correspondiente a este caso:

$$\frac{\begin{array}{c}@_i\langle[\varphi]\alpha \star \beta\rangle \\ \hline @_i\langle\alpha \star \beta\rangle \\ @_i\varphi\end{array}}{}\,(\langle Test\rangle)$$

Aplicando la definición de la función $tam$ tenemos que:

- $tam(\langle[\varphi]\alpha \star \beta\rangle) = 2 + 3 + tam(\varphi) + tam_i(\alpha) + tam_d(\beta)$
- $tam(@_i\langle\alpha \star \beta\rangle) = 2 + tam_i(\alpha) + tam_d(\beta)$
- $tam(@_i\varphi) = 2 + tam(\varphi)$

Por lo tanto la regla $\langle Test\rangle$ disminuye el tamaño de la fórmula. Entonces podemos aplicar hipótesis inductiva:

1. $\mathcal{M}^B, [i] \models \langle\alpha \star \beta\rangle$
2. $\mathcal{M}^B, [i] \models \varphi$

Trabajaremos una vez más con el caso donde $\star$ es $=$, dado que para $\neq$ la demostración es análoga. Queremos ver que $\mathcal{M}^B, [i] \models \langle[\varphi]\alpha = \beta\rangle$, por definición de modelo esto se cumple sii existen $x, y \in \mathcal{M}^B$, con $x \sim y$, tales que:

3. $\mathcal{M}^B, [i], [x] \models [\varphi]\alpha$
4. $\mathcal{M}^B, [i], [y] \models \beta$.

Luego (3) se cumple sii hay un $z \in \mathcal{M}^B$ tq:

5. $\mathcal{M}^B, [i], [z] \models [\varphi]$
6. $\mathcal{M}^B, [z], [y] \models \alpha$.

Tenemos que 5 se cumple sii $[i] = [z]$ y $\mathcal{M}^B, [i] \models \varphi$ lo cual es cierto por (2) y para 6, como $[i] = [z]$ entonces $\mathcal{M}^B, [z], [x] \models \alpha$ queda $\mathcal{M}^B, [i], [x] \models \alpha$.

Por (1) tenemos que existen $a, b \in \mathcal{M}^B$ tales que:

$\mathcal{M}^B, [i], [a] \models \alpha$ y $\mathcal{M}^B, [i], [b] \models \beta$

Entonces eligiendo $a = x$ y $b = y$ aseguramos la existencia de $x, y$.

Probamos entonces que:

$\mathcal{M}^B, [i], [x] \models [\varphi]\alpha$ y $\mathcal{M}^B, [i], [y] \models \beta$, con lo cual por definición de modelo:

$$\mathcal{M}^B, [i] \models \langle[\varphi]\alpha \star \beta\rangle.$$

$\underline{Caso : \varphi = \langle\lambda \star @_j{\downarrow}\beta\rangle}$

Aplicamos la siguiente regla:

$$\frac{@_i\langle\lambda \star @_j{\downarrow}\beta\rangle}{\begin{array}{c} jRk \\ @_k\langle@_i \star \beta\rangle \end{array}} (\downarrow)$$

Aplicando la regla (@,@), tenemos:

$$\frac{@_k\langle@_i \star \beta\rangle}{@_i\langle\lambda \star @_k\beta\rangle} (@, @)$$

Veamos los tamaños de cada fórmula:

- $tam(@_i\langle\lambda \star @_j{\downarrow}\beta\rangle) = 2 + 1 + 2 + tam(\beta)$
- $tam(@_k\langle@_i \star \beta\rangle) = 2 + 2 + tam(\beta)$
- $tam(@_i\langle\lambda \star @_k\beta) = 2 + 1 + tam(\beta)$

Tanto la regla $(\downarrow)$ como $(@, @)$ disminuyen el tamaño de la fórmula,
entonces podemos aplicar hipótesis inductiva y obtener:

$\mathcal{M}^B, [k] \models \langle@_i \star \beta\rangle$

$\mathcal{M}^B, [i] \models \langle\lambda \star @_k\beta\rangle$

Por HI, $\mathcal{M}^B, [i] \models \langle\lambda \star @_k\beta'\rangle$. Como además $jRk$ podemos concluir por definición de modelo que

$$\mathcal{M}^B, [i] \models \langle\lambda \star @_j{\downarrow}\beta'\rangle$$

$\underline{Caso\langle\lambda \star @_j[\varphi]\beta\rangle}$

$$\frac{@_i\langle\lambda \star @_j[\varphi]\beta\rangle}{\begin{array}{c}@_j\varphi\\ @_i\langle\lambda \star @_j\beta\rangle\end{array}}\ (\text{Test})$$

Aplicando función de tamaño:

- $tam(@_i\langle\lambda \star @_j[\varphi]\beta\rangle) = 7 + tam(\varphi) + tam(\beta)$
- $tam(@_j\varphi) = 2 + tam(\varphi)$
- $tam(@_i\langle\lambda \star @_j\beta\rangle) = 3 + tam(\beta)$

Claramente la regla de test disminuye el tamaño de la fórmula, por lo tanto:

1. $\mathcal{M}^B, [j] \models \varphi$
2. $\mathcal{M}^B, [i] \models \langle\lambda \star @_j\beta\rangle$

Luego, por (1) y (2) y definición del modelo:

3. $\mathcal{M}^B, [i], x \models \lambda$
4. $\mathcal{M}^B, [i], y \models @_j\beta$

Por (4) tenemos que:

5. $\mathcal{M}^B, [i], z \models @_j$
6. $\mathcal{M}^B, z, y \models \beta$

Por (1) y (5):

(a) $\mathcal{M}^B, [i], z \models @_j[j]$

Usando $(a)$ y (6):

(b) $\mathcal{M}^B, [i], y \models @_j[\varphi]\beta$

Finalmente por (3) y $(b)$:

$$\mathcal{M}^B, [i] \models \langle \lambda \star @_j[\varphi]\beta \rangle$$

$\underline{Caso \langle \lambda \star @_k(\alpha_1 \cup \alpha_2)\beta \rangle}$

$$\frac{@_i\langle \lambda \star @_k(\alpha_1 \cup \alpha_2)\beta \rangle}{@_i\langle \lambda \star @_k\alpha_1\beta \rangle \mid @_i\langle \lambda \star @_k\alpha_2\beta \rangle} \, (Union)$$

Apliquemos la definición de tamaño:

- $tam(@_i\langle \lambda \star @_k(\alpha_1 \cup \alpha_2)\beta \rangle) = 4 + tam(\alpha_1) + tam(\alpha_2) + tam(\beta)$
- $tam(@_i\langle \lambda \star @_k\alpha_1\beta \rangle) = 4 + tam(\alpha_1) + tam(\beta)$
- $tam(@_i\langle \lambda \star @_k\alpha_2\beta \rangle) = 4 + tam(\alpha_2) + tam(\beta)$

La regla *Union* disminuye el tamaño de la fórmula, entonces podemos aplicar hipótesis inductiva y obtener:

1. $\mathcal{M}^B, [i] \models \langle \lambda \star @_k\alpha_1\beta \rangle$
2. $\mathcal{M}^B, [i] \models \langle \lambda \star @_k\alpha_2\beta \rangle$

Como la regla de *Union* nos deriva en una disyunción, veamos que pasa eligiendo a (1). Aplicando la definición de modelo, tenemos que existen $x, y, z, w \in \mathcal{M}^B$ tales que:

3. $\mathcal{M}^B, [i], x \models \lambda$
4. $\mathcal{M}^B, [i], y \models @_k\alpha_1\beta$
5. $\mathcal{M}^B, [i], z \models @_k$
6. $\mathcal{M}^B, z, y \models \alpha_1\beta$
7. $\mathcal{M}^B, z, w \models \alpha_1$
8. $\mathcal{M}^B, w, y \models \beta$

Haciendo uso de la definición del modelo podemos utilizar (7) para construir $\mathcal{M}^B, z, w \models \alpha_1 \cup \alpha_2$ con $\alpha_2$ una *path expression*.

Con (8) y la expresión anterior formamos:

$$\mathcal{M}^B, z, y \models \langle(\alpha_1 \cup \alpha_2)\beta\rangle$$

usando la fórmula 5 y la expresión anterior obtenemos:

$$\mathcal{M}^B, [i], y \models @_k(\alpha_1 \cup \alpha_2)\beta$$

y finalmente con (3) construimos la fórmula que estábamos buscando:

$$\mathcal{M}^B, [i] \models \langle \lambda \star @_k(\alpha_1 \cup \alpha_2)\beta \rangle$$

Para la fórmula (2) el razonamiento es análogo, pero utilizando $\alpha_2$ en lugar de $\alpha_1$.

$\underline{Caso \langle \lambda \star @_j @_k \beta \rangle}$

$$\frac{@_i \langle \lambda \star @_j @_k \beta \rangle}{@_i \langle \lambda \star @_k \beta \rangle} (@,@)$$

Aplicando definición de tamaño obtenemos:

- $tam(@_i \langle \lambda \star @_j @_k \beta \rangle) = 6 + tam(\beta)$
- $tam(@_i \langle \lambda \star @_k \beta \rangle) = 4 + tam(\beta)$

Como en los casos anteriores observamos que la regla $(@,@)$ disminuye el tamaño de la fórmula,
entonces aplicando hipótesis inductiva podemos concluir que:

$$\mathcal{M}^B, [i] \models \langle \lambda \star @_k \beta \rangle$$

Aplicando la definición del modelo obtenemos:

1. $\mathcal{M}^B, [i], x \models \lambda$
2. $\mathcal{M}^B, [i], y \models @_k \beta$
3. $\mathcal{M}^B, [i], z \models @_k$
4. $\mathcal{M}^B, [i], y \models \beta$

Por definición:

5. $\mathcal{M}^B, [i], [j] \models @_j$ ya que $nom(j) = [j]$
6. $\mathcal{M}^B, j, z \models @_k$ ya que $nom(k) = z$ por (3)

Luego, por (5) y (6):

7. $\mathcal{M}^B, [i], z \models @_j @_k$

con (7) y (4):

8. $\mathcal{M}^B, [i], y \models @_j @_k \beta$

finalmente, con (8) y (1) obtenemos lo que queríamos probar:

$$\mathcal{M}^B, [i] \models \langle \lambda \star @_j @_k \beta \rangle$$

□

# Capítulo 5

# Conclusiones

En esta tesis introducimos el primer cálculo de tableaux correcto y completo para el lenguaje XPath con comparaciones de igualdad y desigualdad de datos. Para ello extendimos con nominales y el operador de la lógica híbrida @. Tal como se argumenta en [14], agregar operadores híbridos es una técnica muy utilizada debido al buen comportamiento de su teoría de prueba.

En el Capítulo 1 presentamos una visión de XPath como lenguaje modal híbrido, un enfoque adaptado en los últimos años, tal como en [33, 16, 15]. También se discute la importancia de XPath como un lenguaje de consulta para bases de datos no relacionales. Intentamos dar una noción general sobre el por qué es provechoso mirar a este lenguaje desde el punto de vista de la lógica modal, y en particular en este trabajo, desde la lógica modal híbrida

En el Capítulo 2 ahondamos en nociones básicas de la lógica modal y la lógica híbrida, con el objeto de ilustrar las similitudes y las diferencias con el lenguaje XPath. Se introduce la sintaxis y la semántica de la lógica modal básica $\mathsf{K}$ y de la lógica híbrida básica $\mathcal{H}(@)$. Luego se motiva el uso del lenguaje XPath en la web, y se introduce su definición como un lenguaje formal: llamamos a este lenguaje HXPath$_{=}(\downarrow)$. Definimos la sintaxis de las *path expressions* y *node expressions*, así como también la noción de modelo híbrido de datos sobre los cuales son interpretadas. Finalmente definimos la semántica de HXPath$_{=}(\downarrow)$ dado un modelo abstracto de datos, como así también algunas propiedades interesantes del lenguaje.

El Capítulo 3 da una introducción breve a los tableaux como técnica para la verificación de satisfacibilidad de fórmulas y como método para construir contraejemplos. Luego introducimos las reglas de tableaux para HXPath$_{=}(\downarrow)$, tanto para el operador $\langle\cdot\rangle$ como para el operador $[\cdot]$. Además se muestran algunos ejemplos de la aplicación de las reglas para algunas fórmulas en particular.

Finalmente en el Capítulo 4 nos centramos en la demostración de completitud del cálculo de tableaux presentado para HXPath$_{=}(\downarrow)$. Definimos el modelo sintáctico abstracto y la función *tam* que devuelve el tamaño de una fórmula. Esta función nos permitirá comprobar que tras la aplicación de las reglas de tableaux a las fórmulas de HXPath$_{=}(\downarrow)$, la complejidad de las mismas no se incrementa. Se habla del significado de la completitud para este cálculo y se brindan los teoremas y demostraciones auxiliares para demotrar que el cálculo es completo.

Como consecuencia de este trabajo, en [5] se presenta una modificación a este cálculo, modificando las reglas para facilitar la demostración de terminación del cálculo. Una consecuencia importante es que esto demuestra que el problema de verificar si una fórmula de HXPath$_{=}(\downarrow)$ es satisfacible es decidible. Más aún, dicho cálculo se utiliza para demostrar que el problema de satisfacibilidad para HXPath$_{=}(\downarrow)$ es PSPACE-completo. Resulta evidente que el cálculo introducido en [5] es más eficiente que el presentado en esta tesis. Sin embargo, el trabajo de esta tesis ha sido el primer cálculo de tableaux para HXPath$_{=}(\downarrow)$, el cual fue

refinado logrando el cálculo de [5].

Como mencionamos anteriormente, de acuerdo a nuestro conocimiento este es el primer cálculo de tableaux para XPath con comparaciones de dato. En [8] se introduce un cálculo de secuentes para una versión de XPath con comparaciones de dato restringidas; en [2] se intrduce una axiomatización ecuacional para XPath con navegación hacia abajo y datos, y en [4] se estudia una axiomatización a la Hilbert para XPath con comparaciones de datos y operadores híbridos. La ventaja de nuestro enfoque, es que los procedimientos basados en tableaux son más apropiados para implementaciones.

**Trabajo futuro.** Existen varias líneas de investigación a ser exploradas en el futuro. La primera es aprovechar las técnicas e implementaciones de procedimientos basados en tableaux para lógicas híbridas existentes, para desarrollar un demostrador para $\mathrm{HXPath}_{=}(\downarrow)$. Sería interesante usar las implementaciones de demostradores como HTab [31, 30] o Spartacus [28], y extenderlos para manejar comparaciones de datos.

Otra línea interesante es explorar procedimientos de tableaux para extensiones de $\mathrm{HXPath}_{=}(\downarrow)$, en particular incluyendo otros operadores de navegación como *descendente* ($\downarrow^*$), ancestro ($\uparrow^*$), padre ($\uparrow$), o hermano ($\twoheadrightarrow$). Para algunos de estos operadores es posible aprovechar las reglas de tableaux utilizadas para operadores modales similares. Por otro lado es posible considerar variantes semánticas, tal como considerar modelos de árbol, que pueden ser vistos como abstracciones de documentos XML.

# Referencias


[1] 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.

[2] S. Abriola, M. E. Descotte, R. Fervari, and S. Figueira. Axiomatizations for downward XPath on data trees. *J. Comput. Syst. Sci.*, 89:209–245, 2017.

[3] S. Abriola, M. E. Descotte, and S. Figueira. Model theory of XPath on data trees. Part II: Binary bisimulation and definability. *Inf. Comput.*, 255:195–223, 2017.

[4] 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, Larnaca, Cyprus, November 9-11, 2016, Proceedings*, volume 10021 of *Lecture Notes in Computer Science*, pages 34–48, 2016.

[5] C. Areces, R. Fervari, and N. Seiler. Tableaux for hybrid XPath with data. In Eugénio C. Oliveira, João Gama, Zita A. Vale, and Henrique Lopes Cardoso, editors, *Progress in Artificial Intelligence - 18th EPIA Conference on Artificial Intelligence, EPIA 2017, Porto, Portugal, September 5-8, 2017, Proceedings*, volume 10423 of *Lecture Notes in Computer Science*, pages 611–623. Springer, 2017.

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

[7] M. Arenas, W. Fan, and L. Libkin. On verifying consistency of xml specifications. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 259–270, New York, NY, USA, 2002. ACM.

[8] 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.

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

[10] M. Benedikt and C. Koch. Xpath leashed. *ACM Computing Surveys*, 41(1):3:1–3:54, January 2009.

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

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

[13] 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.

[14] T. Braüner. Why does the proof-theory of hybrid logic work so well? *Journal of Applied Non-Classical Logics*, 17(4):521–543, 2007.

[15] B. ten Cate, T. Litak, and M. Marx. Complete axiomatizations for XPath fragments. *Journal of Applied Logic*, 8(2):153–172, 2010.

[16] B. ten Cate and M. Marx. Axiomatizing the logical core of XPath 2.0. *Theory of Computing Systems*, 44(4):561–589, 2009.

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

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

[19] M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors. *Handbook of Tableau Methods*. Springer, 1999.

[20] W. Fan, C. Chan, and M. Garofalakis. Secure XML querying with security views. In *Proc. of the 2004 ACM SIGMOD Int. Conf. on Management of Data*, pages 587–598. ACM, 2004.

[21] D. Figueira. *Reasoning on Words and Trees with Data*. PhD thesis, Laboratoire Spécification et Vérification, ENS Cachan, France, 2010.

[22] D. Figueira. Decidability of downward XPath. *ACM Transactions on Computational Logic*, 13(4):34, 2012.

[23] 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.

[24] 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.

[25] 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.

[26] R. Goldblatt. Mathematical modal logic: a view of its evolution. In Dov Gabbay and John Woods, editors, *Handbook of the History of Logic*, volume 7, pages 1–98. Elsevier, 2006.

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

[28] D. Götzmann, M. Kaminski, and G. Smolka. Spartacus: A tableau prover for hybrid logic. *Electronic Notes in Theoretical Computer Science*, 262:127–139, 2010.

[29] G. Governatori. Labelled modal tableaux. In Carlos Areces and Rob Goldblatt, editors, *Advances in Modal Logic*, volume 7, pages 87–110. College Publications, 2008.

[30] G. Hoffmann. *Tâches de raisonnement en logiques hybrides. (Reasoning Tasks for Hybrid Logics)*. PhD thesis, Henri Poincaré University, Nancy, France, 2010.

[31] G. Hoffmann and C. Areces. HTab: A terminating tableaux system for hybrid logic. *Electronic Notes in Theoretical Computer Science*, 231:3–19, March 2009.

[32] W. Martens and F. Neven. Frontiers of tractability for typechecking simple xml transformations. *J. Comput. Syst. Sci.*, 73(3):362–390, 2007.

[33] 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.

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

[35] 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/.