Proof Theory for XPath using Hybrid Logic tools

Raul Fervari

FaMAF-UNC and CONICET, Argentina

work in collaboration with C. Areces

LSV-ENS Cachan, France,
July 11, 2017

In this talk

- Introduce the language HXPath, an extension of XPath with hybrid operators
- Introduce an axiomatic system
- Prove completeness via a Henkin-style construction
- A tableaux calculus for HXPath
- Discuss future work

XPath as a modal language

- XPath is one of the most used query languages for XML documents.
- XML documents are trees (relational structures).
- Core-XPath: fragment that can express properties on the underlying tree structure
- It is essentially a modal language (such as LTL, PDL).
- Sometimes not expressive enough, e.g.: the *join* in database theory, cannot be implemented without access to the data attributes.
- Core-Data-XPath (here $\mathrm{XPath}_{=}$) extends Core-XPath with $=$ and $\neq$ comparisons for data.

Data Trees

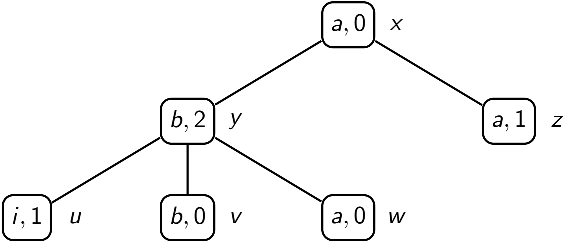

Figure: An example of a data tree.

Data Trees and structural properties

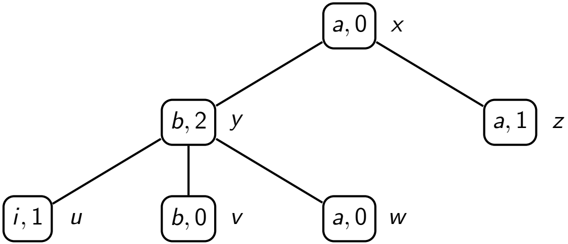

Figure: An example of a data tree.

- x has a child labeled by a (in modal logic, $\Diamond a$).

Data Trees and structural properties

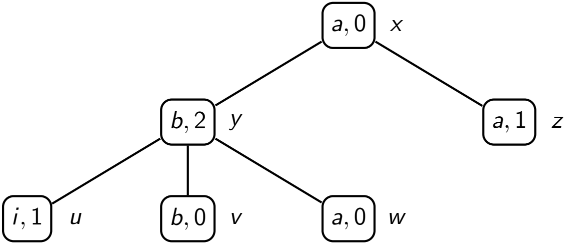

Figure: An example of a data tree.

- x has a child labeled by a (in modal logic, $\Diamond a$).
- x has a two-steps descendant labeled by b and no child labeled by c ($\Diamond\Diamond b \wedge \neg\Diamond c$).

Data Trees and structural properties

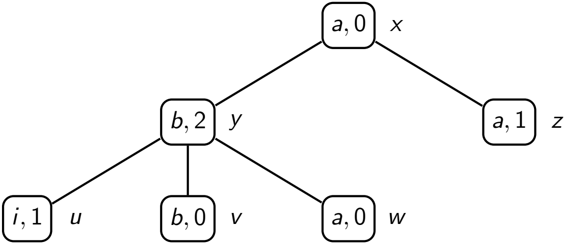

Figure: An example of a data tree.

- x has a child labeled by a (in modal logic, $\Diamond a$).
- x has a two-steps descendant labeled by b and no child labeled by c ($\Diamond\Diamond b \wedge \neg\Diamond c$).
- Node named by i has no successors (in hybrid logic, $@_i \neg\Diamond\top$).

Data Trees and structural properties

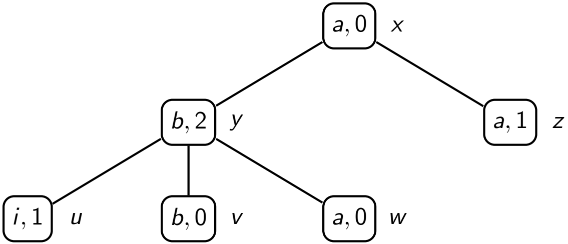

Figure: An example of a data tree.

- x has a child labeled by a (in modal logic, $\Diamond a$).
- x has a two-steps descendant labeled by b and no child labeled by c ($\Diamond\Diamond b \wedge \neg\Diamond c$).
- Node named by i has no successors (in hybrid logic, $@_i \neg\Diamond\top$).
- We cannot talk about data values.

Data Trees and structural properties

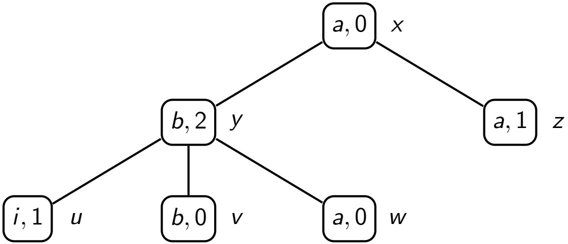

Figure: An example of a data tree.

- x has a child labeled by a (in Core-XPath, $\langle\downarrow[a]\rangle$).
- x has a two-steps descendant labeled by b and no child labeled by c ($\langle\downarrow\downarrow[b]\rangle \wedge \neg\langle\downarrow[c]\rangle$).
- Node named by i has no successors (in Core-XPath + @, $\langle @_i[\neg\langle\downarrow\rangle]\rangle$).
- We cannot talk about data values.

Data Trees and data properties

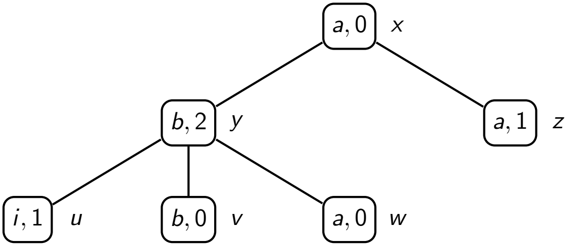

Figure: An example of a data tree.

If we evaluate the expressions at x, we have:

- There is a one-step succesor, and a two-steps succesor, with the same data value (in $\text{XPath}_=$ $\langle \downarrow = \downarrow\downarrow \rangle$).

Data Trees and data properties

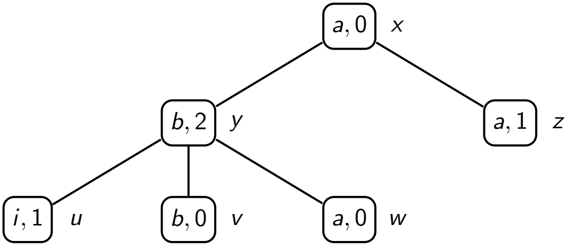

Figure: An example of a data tree.

If we evaluate the expressions at x, we have:

- There is a one-step succesor, and a two-steps succesor, with the same data value (in XPath$_=$ $\langle \downarrow = \downarrow\downarrow \rangle$).
- There is a child labeled by a and a child labeled by b, which have different data values: $\langle \downarrow[a] \neq \downarrow[b] \rangle$.

Data Trees and data properties

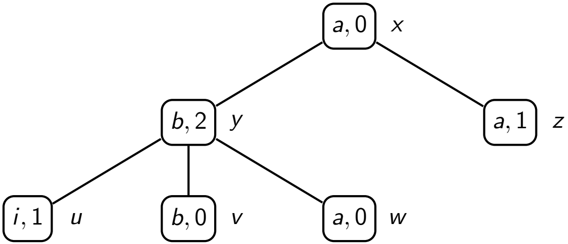

Figure: An example of a data tree.

If we evaluate the expressions at x, we have:

- There is a one-step succesor, and a two-steps succesor, with the same data value (in XPath$_=$ $\langle\downarrow = \downarrow\downarrow\rangle$).
- There is a child labeled by a and a child labeled by b, which have different data values: $\langle\downarrow[a] \neq \downarrow[b]\rangle$.
- Notice we cannot say something like $\langle\downarrow = 2\rangle$!.

Data Trees and data properties

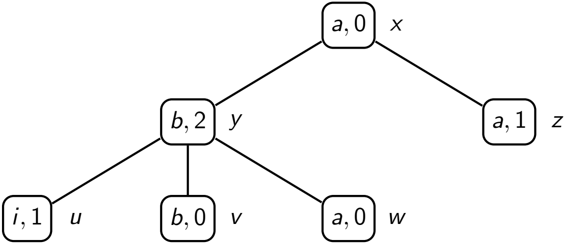

Figure: An example of a data tree.

If we evaluate the expressions at x, we have:

- There is a one-step succesor, and a two-steps succesor, with the same data value (in XPath$_=$ $\langle \downarrow = \downarrow\downarrow \rangle$).
- There is a child labeled by a and a child labeled by b, which have different data values: $\langle \downarrow[a] \neq \downarrow[b] \rangle$.
- Notice we cannot say something like $\langle \downarrow = 2 \rangle$!.
- But, with HXPath$_=(\uparrow\downarrow)$ we will able to say that there is a child with the same data as the node named i: $\langle \downarrow = @_i \rangle$.

Hybrid Data Models

Definition

Let LAB and NOM be two infinite, disjoint countable sets.
A concrete hybrid data model is a tuple $\mathcal{M} = \langle M, D, \rightarrow, label, nom, data\rangle$, where

- M is a non-empty set of elements
- D is a non-empty set of data
- $\rightarrow \subseteq M \times M$ is the accessibility relation
- $label : M \rightarrow 2^{\mathrm{LAB}}$ is the labeling function,
- $nom : \mathrm{NOM} \rightarrow M$ is a function which names some nodes
- $data : M \rightarrow D$ is the function which assigns a data value to each node.

Hybrid Data Models

Definition

Let LAB and NOM be two infinite, disjoint countable sets.
A concrete hybrid data model is a tuple $\mathcal{M} = \langle M, D, \rightarrow, label, nom, data \rangle$, where

- M is a non-empty set of elements
- D is a non-empty set of data
- $\rightarrow \subseteq M \times M$ is the accessibility relation
- $label : M \rightarrow 2^{\mathrm{LAB}}$ is the labeling function,
- $nom : \mathrm{NOM} \rightarrow M$ is a function which names some nodes
- $data : M \rightarrow D$ is the function which assigns a data value to each node.

An abstract hybrid data model is a tuple $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom \rangle$, where $\sim \subseteq M \times M$ is an equivalence relation between elements of M.

Syntax

Definition

The set of path expressions and node expressions of $\mathrm{HXPath}_{=}(\uparrow\downarrow)$ are defined by mutual recursion as follows:

$$\alpha, \beta \ ::= \ \downarrow \mid \uparrow \mid @_i \mid [\varphi] \mid \alpha\beta$$
$$\varphi, \psi \ ::= \ a \mid i \mid \neg\varphi \mid \varphi \land \psi \mid \langle \alpha = \beta \rangle \mid \langle \alpha \neq \beta \rangle, \quad a \in \mathrm{LAB}, i \in \mathrm{NOM}.$$

Syntax

Definition

The set of path expressions and node expressions of HXPath$_=(\uparrow\downarrow)$ are defined by mutual recursion as follows:

$$\alpha, \beta ::= \downarrow \mid \uparrow \mid @_i \mid [\varphi] \mid \alpha\beta$$
$$\varphi, \psi ::= a \mid i \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle \alpha = \beta \rangle \mid \langle \alpha \neq \beta \rangle, \quad a \in \mathsf{LAB}, i \in \mathsf{NOM}.$$

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

Syntax

Definition

The set of path expressions and node expressions of HXPath$_=(\uparrow\downarrow)$ are defined by mutual recursion as follows:

$$\begin{aligned}\alpha, \beta &::= \downarrow \mid \uparrow \mid @_i \mid [\varphi] \mid \alpha\beta \\ \varphi, \psi &::= a \mid i \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle \alpha = \beta \rangle \mid \langle \alpha \neq \beta \rangle, \quad a \in \mathsf{LAB}, i \in \mathsf{NOM}.\end{aligned}$$

Node Expressions		
$\top$	$\equiv$	$p \vee \neg p$
$\bot$	$\equiv$	$\neg\top$
$\langle\alpha\rangle\varphi$	$\equiv$	$\langle \alpha[\varphi] = \alpha[\varphi] \rangle$
$[\alpha]\varphi$	$\equiv$	$\neg\langle\alpha\rangle\neg\varphi$
$@_i\varphi$	$\equiv$	$\langle @_i \rangle\varphi$
Path Expressions		
ϵ	$\equiv$	$[\top]$
$\langle\gamma_1(\alpha \cup \beta)\gamma_2 * \gamma_3\rangle$	$\equiv$	$\langle\gamma_1\alpha\gamma_2 * \gamma_3\rangle \vee \langle\gamma_1\beta\gamma_2 * \gamma_3\rangle$
$\langle\gamma_1 * \gamma_2(\alpha \cup \beta)\gamma_3\rangle$	$\equiv$	$\langle\gamma_1 * \gamma_2\alpha\gamma_3\rangle \vee \langle\gamma_1 * \gamma_2\beta\gamma_3\rangle$

Semantics

Definition

Let $\mathcal{M} = \langle M, \sim, \to, \mathit{label}, \mathit{nom} \rangle$, and $x, y \in M$.

$$\mathcal{M}, x, y \models \downarrow \text{ iff } x \to y$$
$$\mathcal{M}, x, y \models \uparrow \text{ iff } y \to x$$

Semantics

Definition

Let $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom \rangle$, and $x, y \in M$.

$$\begin{aligned} \mathcal{M}, x, y &\models \downarrow \text{ iff } x \rightarrow y \\ \mathcal{M}, x, y &\models \uparrow \text{ iff } y \rightarrow x \\ \mathcal{M}, x, y &\models @_i \text{ iff } nom(i) = y \end{aligned}$$

Semantics

Definition

Let $\mathcal{M} = \langle M, \sim, \rightarrow, \mathit{label}, \mathit{nom} \rangle$, and $x, y \in M$.

$$\begin{array}{rll}
\mathcal{M}, x, y \models \downarrow & \text{iff} & x \rightarrow y \\
\mathcal{M}, x, y \models \uparrow & \text{iff} & y \rightarrow x \\
\mathcal{M}, x, y \models @_i & \text{iff} & \mathit{nom}(i) = y \\
\mathcal{M}, x, y \models [\varphi] & \text{iff} & x = y \text{ and } \mathcal{M}, x \models \varphi \\
\mathcal{M}, x, y \models \alpha\beta & \text{iff} & \exists z \in M \text{ s.t. } \mathcal{M}, x, z \models \alpha \text{ and } \mathcal{M}, z, y \models \beta
\end{array}$$

Semantics

Definition

Let $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom \rangle$, and $x, y \in M$.

$$
\begin{array}{rll}
\mathcal{M}, x, y \models \downarrow & \text{iff} & x \rightarrow y \\
\mathcal{M}, x, y \models \uparrow & \text{iff} & y \rightarrow x \\
\mathcal{M}, x, y \models @_i & \text{iff} & nom(i) = y \\
\mathcal{M}, x, y \models [\varphi] & \text{iff} & x = y \text{ and } \mathcal{M}, x \models \varphi \\
\mathcal{M}, x, y \models \alpha\beta & \text{iff} & \exists z \in M \text{ s.t. } \mathcal{M}, x, z \models \alpha \text{ and } \mathcal{M}, z, y \models \beta \\
\mathcal{M}, x \models a & \text{iff} & a \in label(x)
\end{array}
$$

Semantics

Definition

Let $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom \rangle$, and $x, y \in M$.

$$
\begin{array}{rcl}
\mathcal{M}, x, y \models \downarrow & \text{iff} & x \rightarrow y \\
\mathcal{M}, x, y \models \uparrow & \text{iff} & y \rightarrow x \\
\mathcal{M}, x, y \models @_i & \text{iff} & nom(i) = y \\
\mathcal{M}, x, y \models [\varphi] & \text{iff} & x = y \text{ and } \mathcal{M}, x \models \varphi \\
\mathcal{M}, x, y \models \alpha\beta & \text{iff} & \exists z \in M \text{ s.t. } \mathcal{M}, x, z \models \alpha \text{ and } \mathcal{M}, z, y \models \beta \\
\mathcal{M}, x \models a & \text{iff} & a \in label(x) \\
\mathcal{M}, x \models i & \text{iff} & nom(i) = x
\end{array}
$$

Semantics

Definition

Let $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom \rangle$, and $x, y \in M$.

$$
\begin{array}{rcl}
\mathcal{M}, x, y \models \downarrow & \text{iff} & x \rightarrow y \\
\mathcal{M}, x, y \models \uparrow & \text{iff} & y \rightarrow x \\
\mathcal{M}, x, y \models @_i & \text{iff} & nom(i) = y \\
\mathcal{M}, x, y \models [\varphi] & \text{iff} & x = y \text{ and } \mathcal{M}, x \models \varphi \\
\mathcal{M}, x, y \models \alpha\beta & \text{iff} & \exists z \in M \text{ s.t. } \mathcal{M}, x, z \models \alpha \text{ and } \mathcal{M}, z, y \models \beta \\
\mathcal{M}, x \models a & \text{iff} & a \in label(x) \\
\mathcal{M}, x \models i & \text{iff} & nom(i) = x \\
\mathcal{M}, x \models \neg\varphi & \text{iff} & \mathcal{M}, x \not\models \varphi \\
\mathcal{M}, x \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, x \models \varphi \text{ and } \mathcal{M}, x \models \psi
\end{array}
$$

Semantics

Definition

Let $\mathcal{M} = \langle M, \sim, \to, label, nom \rangle$, and $x, y \in M$.

$$
\begin{array}{rll}
\mathcal{M}, x, y \models \downarrow & \text{iff} & x \to y \\
\mathcal{M}, x, y \models \uparrow & \text{iff} & y \to x \\
\mathcal{M}, x, y \models @_i & \text{iff} & nom(i) = y \\
\mathcal{M}, x, y \models [\varphi] & \text{iff} & x = y \text{ and } \mathcal{M}, x \models \varphi \\
\mathcal{M}, x, y \models \alpha\beta & \text{iff} & \exists z \in M \text{ s.t. } \mathcal{M}, x, z \models \alpha \text{ and } \mathcal{M}, z, y \models \beta \\
\mathcal{M}, x \models a & \text{iff} & a \in label(x) \\
\mathcal{M}, x \models i & \text{iff} & nom(i) = x \\
\mathcal{M}, x \models \neg\varphi & \text{iff} & \mathcal{M}, x \not\models \varphi \\
\mathcal{M}, x \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, x \models \varphi \text{ and } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \langle \alpha = \beta \rangle & \text{iff} & \exists y, z \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \text{ and } y \sim z
\end{array}
$$

Semantics

Definition

Let $\mathcal{M} = \langle M, \sim, \rightarrow, label, nom \rangle$, and $x, y \in M$.

$$
\begin{array}{rcl}
\mathcal{M}, x, y \models \downarrow & \text{iff} & x \rightarrow y \\
\mathcal{M}, x, y \models \uparrow & \text{iff} & y \rightarrow x \\
\mathcal{M}, x, y \models @_i & \text{iff} & nom(i) = y \\
\mathcal{M}, x, y \models [\varphi] & \text{iff} & x = y \text{ and } \mathcal{M}, x \models \varphi \\
\mathcal{M}, x, y \models \alpha\beta & \text{iff} & \exists z \in M \text{ s.t. } \mathcal{M}, x, z \models \alpha \text{ and } \mathcal{M}, z, y \models \beta \\
\mathcal{M}, x \models a & \text{iff} & a \in label(x) \\
\mathcal{M}, x \models i & \text{iff} & nom(i) = x \\
\mathcal{M}, x \models \neg\varphi & \text{iff} & \mathcal{M}, x \not\models \varphi \\
\mathcal{M}, x \models \varphi \wedge \psi & \text{iff} & \mathcal{M}, x \models \varphi \text{ and } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \langle \alpha = \beta \rangle & \text{iff} & \exists y, z \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \text{ and } y \sim z \\
\mathcal{M}, x \models \langle \alpha \neq \beta \rangle & \text{iff} & \exists y, z \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \text{ and } y \not\sim z.
\end{array}
$$

Semantics

Definition

Let $\mathcal{M} = \langle M, \sim, \to, label, nom \rangle$, and $x, y \in M$.

$$
\begin{array}{rcl}
\mathcal{M}, x, y \models \downarrow & \text{iff} & x \to y \\
\mathcal{M}, x, y \models \uparrow & \text{iff} & y \to x \\
\mathcal{M}, x, y \models @_i & \text{iff} & nom(i) = y \\
\mathcal{M}, x, y \models [\varphi] & \text{iff} & x = y \text{ and } \mathcal{M}, x \models \varphi \\
\mathcal{M}, x, y \models \alpha\beta & \text{iff} & \exists z \in M \text{ s.t. } \mathcal{M}, x, z \models \alpha \text{ and } \mathcal{M}, z, y \models \beta \\
\mathcal{M}, x \models a & \text{iff} & a \in label(x) \\
\mathcal{M}, x \models i & \text{iff} & nom(i) = x \\
\mathcal{M}, x \models \neg\varphi & \text{iff} & \mathcal{M}, x \not\models \varphi \\
\mathcal{M}, x \models \varphi \land \psi & \text{iff} & \mathcal{M}, x \models \varphi \text{ and } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \langle \alpha = \beta \rangle & \text{iff} & \exists y, z \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \text{ and } y \sim z \\
\mathcal{M}, x \models \langle \alpha \neq \beta \rangle & \text{iff} & \exists y, z \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \text{ and } y \not\sim z.
\end{array}
$$

Notice:

$$
\begin{array}{rcl}
\mathcal{M}, x \models @_i\varphi & \text{iff} & \mathcal{M}, nom(i) \models \varphi \\
\mathcal{M}, x \models \langle\delta\rangle\varphi & \text{iff} & \exists y \in M \text{ s.t. } x\delta y \text{ and } \mathcal{M}, y \models \varphi \\
\mathcal{M}, x \models [\delta]\varphi & \text{iff} & \forall y \in M,\ x\delta y \text{ then } \mathcal{M}, y \models \varphi.
\end{array}
$$

Examples

Example

Some HXPath$_=(\uparrow\downarrow)$ expressions together with their intuitive meaning:

$\alpha[i]$	There exists an α path between the current point of evaluation and the node named i.

Examples

Example

Some $\mathrm{HXPath}_=(\uparrow\downarrow)$ expressions together with their intuitive meaning:

$\alpha[i]$	There exists an α path between the current point of evaluation and the node named i.
$@_i\alpha$	There exists an α path between the node named i and some other node.

Examples

Example

Some $\mathrm{HXPath}_{=}(\uparrow\downarrow)$ expressions together with their intuitive meaning:

$\alpha[i]$	There exists an α path between the current point of evaluation and the node named i.
$@_i\alpha$	There exists an α path between the node named i and some other node.
$\langle @_i = @_j \rangle$	The node named i has the same data than the node named j.

Examples

Example

Some $\mathrm{HXPath}_{=}(\uparrow\downarrow)$ expressions together with their intuitive meaning:

$\alpha[i]$	There exists an α path between the current point of evaluation and the node named i.
$@_i\alpha$	There exists an α path between the node named i and some other node.
$\langle @_i = @_j \rangle$	The node named i has the same data than the node named j.
$\langle \alpha = @_i\beta \rangle$	There exists a node accessible from the current point of evaluation by an α path that has the same data than a node accessible from the point named i by a β path.

Axiomatization

In addition to an arbitrary set of axiom and rule schemes for propositional logic, we include generalizations of the K axiom and the *Necessitation* rule for the basic modal logic to handle modalities with arbitrary path expressions. We call the system HXP.

Axiom and rule for classical modal logic

K $\quad [\alpha](\varphi \to \psi) \to ([\alpha]\varphi \to [\alpha]\psi)$

$$\frac{\vdash \varphi}{\vdash [\alpha]\varphi}\ \textit{Nec}$$

We include also the standard axioms for future and past operators.

Axioms for $\downarrow, \uparrow$-interaction	
down-up	$\varphi \to [\downarrow]\langle\uparrow\rangle\varphi$
up-down	$\varphi \to [\uparrow]\langle\downarrow\rangle\varphi$

Rules for hybrid operators

Then we introduce generalizations of the rules for the hybrid logic HL(@).

Hybrid rules

$$\frac{\vdash j \to \varphi}{\vdash \varphi}\ \textit{name} \qquad \frac{\vdash @_i\langle\gamma\rangle j \wedge \langle @_j\alpha * \beta\rangle \to \theta}{\vdash \langle @_i\gamma\alpha * \beta\rangle \to \theta}\ \textit{paste}$$

j is a nominal different from i that does not occur in $\varphi, \theta, \alpha, \beta, \gamma$.

Axioms for @

Now we introduce axioms that handle @. Notice that $@_i$ is a path expression of $\mathrm{HXPath}_=(\uparrow\downarrow)$ and as a result, some of the standard hybrid axioms for @ have been generalized. In particular, the K axiom and *Nec* rule above also apply to $@_i$. In addition, we provide axioms to ensure that the relation induced by @ is a congruence.

Axioms for @		**Congruence for @**	
		@-refl.	$@_i i$
		@-sym.	$@_i j \to @_j i$
@-self-dual	$\neg @_i \varphi \leftrightarrow @_i \neg \varphi$	nom	$@_i j \wedge \langle @_i \alpha * \beta \rangle \to \langle @_j \alpha * \beta \rangle$
@-intro	$i \wedge \varphi \to @_i \varphi$	agree	$\langle @_j @_i \alpha * \beta \rangle \leftrightarrow \langle @_i \alpha * \beta \rangle$
		back	$\langle \gamma @_i \alpha * \beta \rangle \to \langle @_i \alpha * \beta \rangle$

Axioms for XPath

We introduce axioms to handle complex path expressions in data comparisons. Finally, we introduce axioms to handle data tests.

Axioms for paths	
comp-assoc	$\langle(\alpha\beta)\gamma * \eta\rangle \leftrightarrow \langle\alpha(\beta\gamma) * \eta\rangle$
comp-neutral	$\langle\alpha\beta * \gamma\rangle \leftrightarrow \langle\alpha\epsilon\beta * \gamma\rangle$ (α or β can be empty)
comp-dist	$\langle\alpha\beta\rangle\varphi \leftrightarrow \langle\alpha\rangle\langle\beta\rangle\varphi$

Axioms for XPath

We introduce axioms to handle complex path expressions in data comparisons. Finally, we introduce axioms to handle data tests.

Axioms for paths	
comp-assoc	$\langle (\alpha\beta)\gamma * \eta \rangle \leftrightarrow \langle \alpha(\beta\gamma) * \eta \rangle$
comp-neutral	$\langle \alpha\beta * \gamma \rangle \leftrightarrow \langle \alpha\epsilon\beta * \gamma \rangle$ (α or β can be empty)
comp-dist	$\langle \alpha\beta \rangle\varphi \leftrightarrow \langle \alpha \rangle\langle \beta \rangle\varphi$
Axioms for data	
equal	$\langle \epsilon = \epsilon \rangle$
distinct	$\neg\langle \epsilon \neq \epsilon \rangle$
@-data	$\neg\langle @_i = @_j \rangle \leftrightarrow \langle @_i \neq @_j \rangle$
ϵ-trans	$\langle \epsilon = \alpha \rangle \wedge \langle \epsilon = \beta \rangle \rightarrow \langle \alpha = \beta \rangle$
$*$-comm	$\langle \alpha * \beta \rangle \leftrightarrow \langle \beta * \alpha \rangle$
$*$-test	$\langle [\varphi]\alpha * \beta \rangle \leftrightarrow \varphi \wedge \langle \alpha * \beta \rangle$
@$*$-dist	$\langle @_i\alpha * @_i\beta \rangle \rightarrow @_i\langle \alpha * \beta \rangle$
subpath	$\langle \alpha\beta * \gamma \rangle \rightarrow \langle \alpha \rangle\top$
comp$*$-dist	$\langle \alpha \rangle\langle \beta * \gamma \rangle \rightarrow \langle \alpha\beta * \alpha\gamma \rangle$

Some natural theorems

Proposition

The following formulas are theorems in HXP.

1. *test-dist* $\vdash \langle [\varphi] = [\psi] \rangle \leftrightarrow \varphi \land \psi$
2. *test-*$\bot$ $\vdash \langle [\varphi] \neq [\psi] \rangle \leftrightarrow \bot$
3. *@-swap* $\vdash @_i \langle \alpha * @_j \beta \rangle \leftrightarrow @_j \langle \beta * @_i \alpha \rangle$
4. *bridge* $\vdash \langle \alpha \rangle i \land @_i \varphi \to \langle \alpha \rangle \varphi$

The Completeness Proof

The completeness argument follows the lines of the completeness proof for HL(@), which is a Henkin-style proof with nominals playing the role of first-order constants.
In what follows, we will write $\Gamma \vdash \varphi$ if and only if φ can be obtained from a set of formulas Γ by applying the inference rules of HXP.

Definition

Let Γ be a set of formulas, we say that Γ is an HXP maximal consistent set (HXP-MCS, or MCS for short) if and only if $\Gamma \nvdash \bot$ and for all $\varphi \notin \Gamma$ we have $\Gamma \cup \{\varphi\} \vdash \bot$.

Each MCS is a full model description

In the same way as for hybrid logic, inside every MCS there are a collection of MCSs with some desirable properties:

Lemma

Let Γ *be an* HXP-*MCS. For any nominal* $i \in \Gamma$*, let us define* $\Delta_i = \{\varphi \mid @_i\varphi \in \Gamma\}$*. Then*

1. Δ_i *is an* HXP-*MCS.*
2. *For all nominals* i, j*, if* $i \in \Delta_j$ *then* $\Delta_i = \Delta_j$*.*
3. *For all nominals* i, j*, we have* $@_i\varphi \in \Delta_j$ *iff* $@_i\varphi \in \Gamma$*.*
4. *If* $k \in \Gamma$ *then* $\Gamma = \Delta_k$*.*

Naming and Pasting MCSs

Definition (Named and Pasted MCS)

Let Γ be an HXP-MCS. We say that Γ is named if for some nominal i we have that $i \in \Gamma$ (and we will say that Γ is named by i).

Naming and Pasting MCSs

Definition (Named and Pasted MCS)

Let Γ be an HXP-MCS. We say that Γ is named if for some nominal i we have that $i \in \Gamma$ (and we will say that Γ is named by i).

We say that Γ is pasted if the following holds:

1. $\langle @_i \delta\alpha = \beta \rangle \in \Gamma$ implies that $\exists j$, $@_i \langle \delta \rangle j \wedge \langle @_j \alpha = \beta \rangle \in \Gamma$
2. $\langle @_i \delta\alpha \neq \beta \rangle \in \Gamma$ implies that $\exists j$, $@_i \langle \delta \rangle j \wedge \langle @_j \alpha \neq \beta \rangle \in \Gamma$.

Lindenbaum

Lemma (Extended Lindenbaum Lemma)

Let NOM$'$ *be a (countably) infinite set of nominals disjoint from* NOM*, and let* $HXPath_=(\uparrow\downarrow)'$ *be the language obtained by adding these new nominals to* $HXPath_=(\uparrow\downarrow)$*. Then, every* HXP*-consistent set of formulas in* $HXPath_=(\uparrow\downarrow)$ *can be extended to a named and pasted* HXP*-MCS in* $HXPath_=(\uparrow\downarrow)'$*.*

Extracted Model

Definition

Let Γ be a named and pasted HXP-MCS, then we define the *extracted model* from Γ, $\mathcal{M}_\Gamma = \langle M, \sim, \rightarrow, label, nom \rangle$ as:

- $M = \{\Delta_i \mid \Delta_i \text{ was obtained from } \Gamma\}$
- $\Delta_i \rightarrow \Delta_j$ iff $\langle\downarrow\rangle j \in \Delta_i$
- $a \in label(\Delta_i)$ iff $a \in \Delta_i$
- $nom(i) = \Delta_i$
- $\Delta_i \sim \Delta_j$ iff $\langle \epsilon = @_j \rangle \in \Delta_i$.

Existence Lemma

Lemma

Let Γ *be an* HXP-*MCS and let* $\mathcal{M}_\Gamma = \langle M, \sim, \rightarrow, \mathit{label}, \mathit{nom} \rangle$ *be the extracted model from* Γ*. Suppose* $\Delta \in M$ *and* $i \in \Delta$*. Then*

1. $\langle \delta\alpha = \beta \rangle \in \Delta$ *implies* $\exists \Sigma \in M$ *s.t.* $\Delta\delta\Sigma$ *and* $\langle \alpha = @_i\beta \rangle \in \Sigma$,
2. $\langle \delta\alpha \neq \beta \rangle \in \Delta$ *implies* $\exists \Sigma \in M$ *s.t.* $\Delta\delta\Sigma$ *and* $\langle \alpha \neq @_i\beta \rangle \in \Sigma$,

Existence Lemma

Lemma

Let Γ *be an* HXP-*MCS and let* $\mathcal{M}_\Gamma = \langle M, \sim, \rightarrow, label, nom\rangle$ *be the extracted model from* Γ. *Suppose* $\Delta \in M$ *and* $i \in \Delta$. *Then*

1. $\langle \delta\alpha = \beta\rangle \in \Delta$ *implies* $\exists \Sigma \in M$ *s.t.* $\Delta\delta\Sigma$ *and* $\langle \alpha = @_i\beta\rangle \in \Sigma$,
2. $\langle \delta\alpha \neq \beta\rangle \in \Delta$ *implies* $\exists \Sigma \in M$ *s.t.* $\Delta\delta\Sigma$ *and* $\langle \alpha \neq @_i\beta\rangle \in \Sigma$,
3. $\langle @_j\alpha = @_k\beta\rangle \in \Delta$ *implies there exists* $\Sigma \in M$ *s.t.* $\langle \alpha = @_k\beta\rangle \in \Sigma$,
4. $\langle @_j\alpha \neq @_k\beta\rangle \in \Delta$ *implies there exists* $\Sigma \in M$ *s.t.* $\langle \alpha \neq @_k\beta\rangle \in \Sigma$.

Truth Lemma

We can prove the Truth Lemma that states that membership in an MCS of the extracted model is equivalent to being true in that MCS.

Lemma

Let $\mathcal{M}_\Gamma = \langle M, \sim, \rightarrow, label, nom \rangle$ be the extracted model from a MCS Γ, and let $\Delta_i \in M$. Then, for any formula φ,

$$\mathcal{M}_\Gamma, \Delta_i \models \varphi \;\; \textit{iff} \;\; \varphi \in \Delta_i.$$

Truth Lemma

We can prove the Truth Lemma that states that membership in an MCS of the extracted model is equivalent to being true in that MCS.

Lemma

Let $\mathcal{M}_\Gamma = \langle M, \sim, \to, label, nom \rangle$ be the extracted model from a MCS Γ, and let $\Delta_i \in M$. Then, for any formula φ,

$$\mathcal{M}_\Gamma, \Delta_i \models \varphi \text{ iff } \varphi \in \Delta_i.$$

As a result we obtain the completeness result.

Theorem

The axiomatic system HXP *is complete for abstract hybrid data models.*

Other systems

- We introduce an axiom system which is a theorem generation machine.
- It's very elegant, but not very handy computationally.
- A tableaux calculus is more appropriate to get implementations.
- We follow similar ideas: nominals and satisfaction operators can be used in tableaux to keep track of the evaluation of a formula during an attempt to build a model.
- We obtained a terminating PSPACE algorithm.

Some tableaux rules

Prefix Internalization Rules

$$\frac{@_n\langle\alpha * \beta\rangle}{\langle @_n\alpha * @_n\beta\rangle}\,(Int) \qquad \frac{@_n\neg\langle\alpha * \beta\rangle}{\neg\langle @_n\alpha * @_n\beta\rangle}\,(\neg Int)$$

Some tableaux rules

Prefix Internalization Rules

$$\frac{@_n\langle \alpha * \beta\rangle}{\langle @_n\alpha * @_n\beta\rangle}\,(Int) \qquad \frac{@_n\neg\langle \alpha * \beta\rangle}{\neg\langle @_n\alpha * @_n\beta\rangle}\,(\neg Int)$$

Some XPath rules

$$\frac{\langle @_n{\downarrow}\alpha * \beta\rangle}{\begin{array}{c} n \to m \\ \langle @_m\alpha * \beta\rangle \end{array}}\,(\downarrow)\ \ {\scriptstyle m \text{ is new}} \qquad \frac{\neg\langle @_n{\downarrow}\alpha * \beta\rangle \quad n \to m}{\neg\langle @_m\alpha * \beta\rangle}\,(\neg\downarrow)$$

Some tableaux rules

Prefix Internalization Rules

$$\frac{@_n\langle \alpha * \beta \rangle}{\langle @_n\alpha * @_n\beta \rangle}\,(Int) \qquad \frac{@_n\neg\langle \alpha * \beta \rangle}{\neg\langle @_n\alpha * @_n\beta \rangle}\,(\neg Int)$$

Some XPath rules

$$\frac{\langle @_n{\downarrow}\alpha * \beta \rangle}{\begin{array}{c} n \to m \\ \langle @_m\alpha * \beta \rangle \end{array}}\,(\downarrow) \quad m \text{ is new} \qquad \frac{\neg\langle @_n{\downarrow}\alpha * \beta \rangle \quad n \to m}{\neg\langle @_m\alpha * \beta \rangle}\,(\neg\downarrow)$$

$$\frac{\langle @_n * @_m{\downarrow}\alpha \rangle}{\begin{array}{c} m \to k \\ \langle @_n * @_k\alpha \rangle \end{array}}\,(\downarrow_r) \quad k \text{ is new} \qquad \frac{\neg\langle @_n * @_m{\downarrow}\alpha \rangle \quad m \to k}{\neg\langle @_n * @_k\alpha \rangle}\,(\neg\downarrow_r)$$

Future Work

- Take advantage of the hybridization of $\mathrm{XPath}_=$ to obtain general axiomatizations: define minimal proof systems that are complete when extended with additional axioms that are *pure*.

Future Work

- Take advantage of the hybridization of $\mathsf{XPath}_=$ to obtain general axiomatizations: define minimal proof systems that are complete when extended with additional axioms that are *pure*.
- Explore this general framework and obtain complete axiomatic systems for natural extensions of $\mathsf{HXPath}_=(\uparrow\downarrow)$:
  - $\mathsf{HXPath}_=(\uparrow\downarrow)$ with reflexive-transitive closure for downward/upward navigation (i.e., allowing $\downarrow^*$ and $\uparrow^*$), and sibling navigation.
  - Exploring new kind of data comparisons, for instance, including the relation $<$ in addition to $=$ and $\neq$.

Future Work

- Take advantage of the hybridization of $\mathrm{XPath}_=$ to obtain general axiomatizations: define minimal proof systems that are complete when extended with additional axioms that are *pure*.
- Explore this general framework and obtain complete axiomatic systems for natural extensions of $\mathrm{HXPath}_=(\uparrow\downarrow)$:
  - $\mathrm{HXPath}_=(\uparrow\downarrow)$ with reflexive-transitive closure for downward/upward navigation (i.e., allowing $\downarrow^*$ and $\uparrow^*$), and sibling navigation.
  - Exploring new kind of data comparisons, for instance, including the relation $<$ in addition to $=$ and $\neq$.
- Extend tableaux calculus for:
  - Handling data trees.
  - Additional navigation axis: descendant ($\downarrow^*$), ancestor ($\uparrow^*$), father ($\uparrow$), next-sibling ($\twoheadrightarrow$), etc.

Future Work

- Take advantage of the hybridization of $\mathrm{XPath}_=$ to obtain general axiomatizations: define minimal proof systems that are complete when extended with additional axioms that are *pure*.
- Explore this general framework and obtain complete axiomatic systems for natural extensions of $\mathrm{HXPath}_=(\uparrow\downarrow)$:
  - $\mathrm{HXPath}_=(\uparrow\downarrow)$ with reflexive-transitive closure for downward/upward navigation (i.e., allowing $\downarrow^*$ and $\uparrow^*$), and sibling navigation.
  - Exploring new kind of data comparisons, for instance, including the relation $<$ in addition to $=$ and $\neq$.
- Extend tableaux calculus for:
  - Handling data trees.
  - Additional navigation axis: descendant ($\downarrow^*$), ancestor ($\uparrow^*$), father ($\uparrow$), next-sibling ($\twoheadrightarrow$), etc.
- Get implementations: extending the Hybrid Logic prover HTab to handle data.