A Tableau Calculus for Hybrid XPath with Data

April 2026

Rio de Janeiro, Brasil

Carlos Areces

FAMAF – Universidad Nacional de Córdoba

carlos.areces@unc.edu.ar

In This Talk

- Well, you just read the title. . . . I am going to introduce
  - A Tableaux Calculus
  - for Hybrid
  - XPath
  - with Data

In This Talk

- Well, you just read the title. . . . I am going to introduce
  - A Tableaux Calculus
  - for Hybrid
  - XPath
  - with Data
- To get there we have some work ahead
  - First, introduce (propositional) Modal Logics, and their tableaux,

In This Talk

- Well, you just read the title. . . . I am going to introduce
  - A Tableaux Calculus
  - for Hybrid
  - XPath
  - with Data
- To get there we have some work ahead
  - First, introduce (propositional) Modal Logics, and their tableaux,
  - then, introduce Hybrid Logics, and their tableaux,

In This Talk

- Well, you just read the title. . . . I am going to introduce
  - A Tableaux Calculus
  - for Hybrid
  - XPath
  - with Data
- To get there we have some work ahead
  - First, introduce (propositional) Modal Logics, and their tableaux,
  - then, introduce Hybrid Logics, and their tableaux,
  - then, introduce XPath,

In This Talk

- Well, you just read the title. . . . I am going to introduce
  - A Tableaux Calculus
  - for Hybrid
  - XPath
  - with Data
- To get there we have some work ahead
  - First, introduce (propositional) Modal Logics, and their tableaux,
  - then, introduce Hybrid Logics, and their tableaux,
  - then, introduce XPath,
  - and XPath with Data,

In This Talk

- Well, you just read the title. . . . I am going to introduce
  - A Tableaux Calculus
  - for Hybrid
  - XPath
  - with Data
- To get there we have some work ahead
  - First, introduce (propositional) Modal Logics, and their tableaux,
  - then, introduce Hybrid Logics, and their tableaux,
  - then, introduce XPath,
  - and XPath with Data,
  - and Hybrid XPath with Data,

In This Talk

- Well, you just read the title. . . . I am going to introduce
  - A Tableaux Calculus
  - for Hybrid
  - XPath
  - with Data
- To get there we have some work ahead
  - First, introduce (propositional) Modal Logics, and their tableaux,
  - then, introduce Hybrid Logics, and their tableaux,
  - then, introduce XPath,
  - and XPath with Data,
  - and Hybrid XPath with Data,
  - and then, introduce the promised Tableau Calculus

Tableaux for Modal Logics

Start with a tableaux for propositional logic.

$$\frac{(\varphi \wedge \psi)}{\begin{array}{c}\varphi \\ \psi\end{array}} \ (\wedge)$$

$$\frac{\neg(\varphi \wedge \psi)}{\neg\varphi \qquad \neg\psi} \ (\neg\wedge)$$

$$\frac{\neg\neg\varphi}{\varphi} \ (\neg\neg)$$

Tableaux for Modal Logics

Start with a tableaux for propositional logic.

$$\frac{(\varphi \wedge \psi)}{\begin{array}{c}\varphi \\ \psi\end{array}}\ (\wedge)$$

$$\frac{\neg(\varphi \wedge \psi)}{\neg\varphi \qquad \neg\psi}\ (\neg\wedge)$$

$$\frac{\neg\neg\varphi}{\varphi}\ (\neg\neg)$$

- Neat!: 3 rules for an NP-complete problem!

Tableaux for Modal Logics

Start with a tableaux for propositional logic.

$$\frac{(\varphi \wedge \psi)}{\begin{array}{c}\varphi \\ \psi\end{array}} \ (\wedge)$$

$$\frac{\neg(\varphi \wedge \psi)}{\neg\varphi \qquad \neg\psi} \ (\neg\wedge)$$

$$\frac{\neg\neg\varphi}{\varphi} \ (\neg\neg)$$

- Neat!: 3 rules for an NP-complete problem!
- But now we have to perform boolean reasoning in each point of the model.

Tableaux for Modal Logics

Start with a tableaux for propositional logic.

$$\frac{s{:}(\varphi \wedge \psi)}{\begin{array}{l} s{:}\varphi \\ s{:}\psi \end{array}} \ (\wedge)$$

$$\frac{s{:}\neg(\varphi \wedge \psi)}{s{:}\neg\varphi \qquad\qquad s{:}\neg\psi} \ (\neg\wedge)$$

$$\frac{s{:}\neg\neg\varphi}{s{:}\varphi} \ (\neg\neg)$$

- Neat!: 3 rules for an NP-complete problem!
- But now we have to perform boolean reasoning in each point of the model.
- The solution is: labels

Tableaux for Modal Logics

Remember the semantics for $\langle R \rangle$

$$\mathcal{M}, w \models \langle R \rangle \varphi \text{ iff there is } w' \text{ s.t. } wRw' \text{ and } \mathcal{M}, w' \models \varphi.$$

Tableaux for Modal Logics

Remember the semantics for $\langle R \rangle$

$$\mathcal{M}, w \models \langle R \rangle \varphi \text{ iff there is } w' \text{ s.t. } wRw' \text{ and } \mathcal{M}, w' \models \varphi.$$

- Start with the labelled formula $\langle R \rangle \varphi$ true at s.

Tableaux for Modal Logics

Remember the semantics for $\langle R \rangle$

$$\mathcal{M}, w \models \langle R \rangle \varphi \text{ iff there is } w' \text{ s.t. } wRw' \text{ and } \mathcal{M}, w' \models \varphi.$$

- Start with the labelled formula $\langle R \rangle \varphi$ true at s. $\dfrac{s{:}\langle R \rangle \varphi}{} \; (\langle R \rangle)$

Tableaux for Modal Logics

Remember the semantics for $\langle R \rangle$

$$\mathcal{M}, w \models \langle R \rangle \varphi \text{ iff there is } w' \text{ s.t. } wRw' \text{ and } \mathcal{M}, w' \models \varphi.$$

- Start with the labelled formula $\langle R \rangle \varphi$ true at s. If this formula is satisfiable, it is because there is an R-sucessor t where φ holds.

$$\frac{s{:}\langle R \rangle \varphi}{\begin{array}{l} t{:}\varphi \\ sRt \end{array}} \; (\langle R \rangle)$$

(for t a new label)

Tableaux for Modal Logics

Remember the semantics for $\langle R \rangle$

$$\mathcal{M}, w \models \langle R \rangle \varphi \text{ iff there is } w' \text{ s.t. } wRw' \text{ and } \mathcal{M}, w' \models \varphi.$$

- Start with the labelled formula $\langle R \rangle \varphi$ true at s. If this formula is satisfiable, it is because there is an R-sucessor t where φ holds.

$$\frac{s{:}\langle R \rangle \varphi}{\begin{array}{l} t{:}\varphi \\ sRt \end{array}} \ (\langle R \rangle)$$

(for t a new label)

- Start with the labelled formula $s{:}\neg\langle R \rangle \varphi$.

$$s{:}\neg\langle R \rangle \varphi$$

$$\text{---------} \ (\neg\langle R \rangle)$$

Tableaux for Modal Logics

Remember the semantics for $\langle R \rangle$

$$\mathcal{M}, w \models \langle R \rangle \varphi \text{ iff there is } w' \text{ s.t. } wRw' \text{ and } \mathcal{M}, w' \models \varphi.$$

- Start with the labelled formula $\langle R \rangle \varphi$ true at s.
  If this formula is satisfiable, it is because there is an R-sucessor t where φ holds.

$$\frac{s{:}\langle R \rangle \varphi}{\begin{array}{l} t{:}\varphi \\ sRt \end{array}} \ (\langle R \rangle)$$

(for t a new label)

- Start with the labelled formula $s{:}\neg\langle R \rangle \varphi$.
  If there is an R-successor t, then φ should not hold at t.

$$\frac{\begin{array}{l} s{:}\neg\langle R \rangle \varphi \\ sRt \end{array}}{t{:}\neg\varphi} \ (\neg\langle R \rangle)$$

Tableaux for Modal Logics

- Clash: $s{:}p$ and $s{:}\neg p$ in a branch.

Tableaux for Modal Logics

- Clash: $s{:}p$ and $s{:}\neg p$ in a branch.
- To check satisfiability of φ, start the tableaux with the labelled formula $s{:}\varphi$.

Tableaux for Modal Logics

- Clash: $s{:}p$ and $s{:}\neg p$ in a branch.
- To check satisfiability of φ, start the tableaux with the labelled formula $s{:}\varphi$.
- The previous 5 rules provide a sound and complete calculus for the basic modal logic
- It is actually terminating, and by imposing some restrictions on applications it can run in PSPACE, so it is optimal.
- But what is the status of labeled formulas $s{:}\varphi$ and accessibility statements sRt?
- Enter Hybrid Logics ...

Hybrid Logics

- The first ideas behind hybrid logics go back to Arthur Prior, but its modern development flourish with the work of Blackburn and Seligman.
- For this talk we can think of hybrid logics as a way to add constants (and equality) to modal languages, so that we can internalize labelled deduction.

Blackburn, P.
Internalizing Labelled Deduction
Journal of Logic and Computation, 10, 137-168, 2000.

Nominals and Satisfaction Operators

- Extend the language with a new set of propositional symbols called nominals NOM $= \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.

Nominals and Satisfaction Operators

- Extend the language with a new set of propositional symbols called nominals NOM $= \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.
- Nominals behave as constants because we demand they be true at a unique point in a model (i.e., $V(i)$ is a singleton).

Nominals and Satisfaction Operators

- Extend the language with a new set of propositional symbols called nominals NOM $= \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.
- Nominals behave as constants because we demand they be true at a unique point in a model (i.e., $V(i)$ is a singleton).
- $\mathcal{M}, w \models i$ iff $w \in V(i)$

Nominals and Satisfaction Operators

- Extend the language with a new set of propositional symbols called nominals NOM $= \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.
- Nominals behave as constants because we demand they be true at a unique point in a model (i.e., $V(i)$ is a singleton).
- $\mathcal{M}, w \models i$ iff $w \in V(i)$ (equivalently $\{w\} = V(i)$).

Nominals and Satisfaction Operators

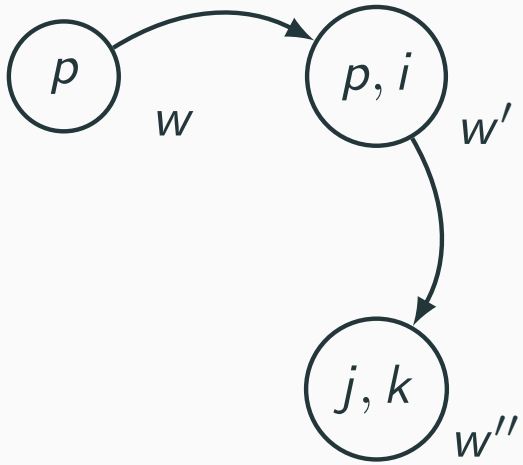
- Extend the language with a new set of propositional symbols called nominals NOM $= \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.
- Nominals behave as constants because we demand they be true at a unique point in a model (i.e., $V(i)$ is a singleton).
- $\mathcal{M}, w \models i$ iff $w \in V(i)$ (equivalently $\{w\} = V(i)$).
  $\mathcal{M}, w \models i{:}\varphi$ iff $\mathcal{M}, v \models \varphi$ for $\{v\} = V(i)$.

Nominals and Satisfaction Operators

- Extend the language with a new set of propositional symbols called nominals $\mathrm{NOM} = \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.
- Nominals behave as constants because we demand they be true at a unique point in a model (i.e., $V(i)$ is a singleton).
- $\mathcal{M}, w \models i$ iff $w \in V(i)$ (equivalently $\{w\} = V(i)$).
  $\mathcal{M}, w \models i{:}\varphi$ iff $\mathcal{M}, v \models \varphi$ for $\{v\} = V(i)$.

Examples: $i, j, k \in \mathrm{NOM}, p \in \mathrm{PROP}$

- $\mathcal{M}, w \models i{:}p$

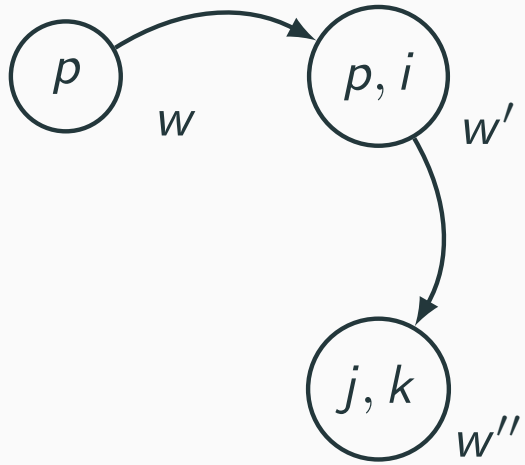


Nominals and Satisfaction Operators

- Extend the language with a new set of propositional symbols called nominals $\mathrm{NOM} = \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.
- Nominals behave as constants because we demand they be true at a unique point in a model (i.e., $V(i)$ is a singleton).
- $\mathcal{M}, w \models i$ iff $w \in V(i)$ (equivalently $\{w\} = V(i)$).
  $\mathcal{M}, w \models i{:}\varphi$ iff $\mathcal{M}, v \models \varphi$ for $\{v\} = V(i)$.

Examples: $i, j, k \in \mathrm{NOM}, p \in \mathrm{PROP}$

- $\mathcal{M}, w \models i{:}p$
- $\mathcal{M}, w \models j{:}i{:}p$

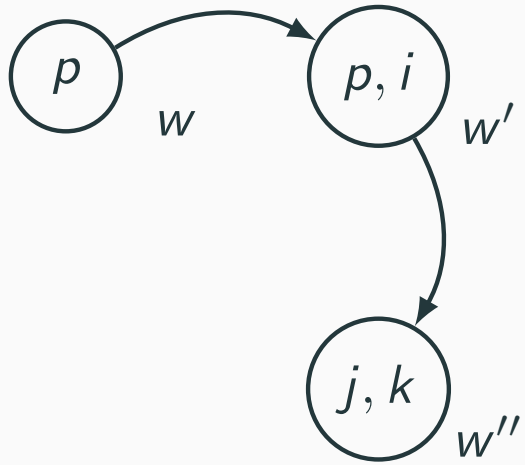


Nominals and Satisfaction Operators

- Extend the language with a new set of propositional symbols called nominals $\mathrm{NOM} = \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.
- Nominals behave as constants because we demand they be true at a unique point in a model (i.e., $V(i)$ is a singleton).
- $\mathcal{M}, w \models i$ iff $w \in V(i)$ (equivalently $\{w\} = V(i)$).
  $\mathcal{M}, w \models i{:}\varphi$ iff $\mathcal{M}, v \models \varphi$ for $\{v\} = V(i)$.

Examples: $i, j, k \in \mathrm{NOM}, p \in \mathrm{PROP}$

- $\mathcal{M}, w \models i{:}p$
- $\mathcal{M}, w \models j{:}i{:}p$
- $\mathcal{M}, w \models j{:}k$

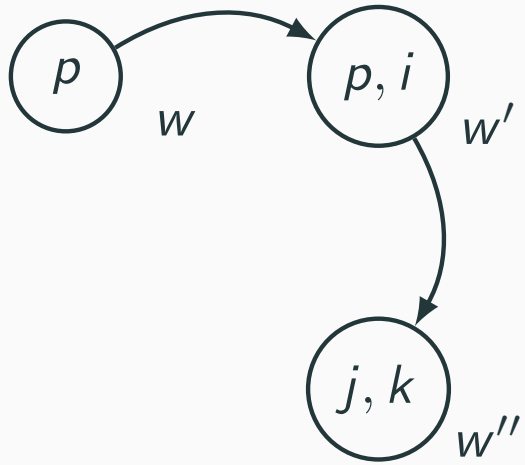


Nominals and Satisfaction Operators

- Extend the language with a new set of propositional symbols called nominals NOM $= \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.
- Nominals behave as constants because we demand they be true at a unique point in a model (i.e., $V(i)$ is a singleton).
- $\mathcal{M}, w \models i$ iff $w \in V(i)$ (equivalently $\{w\} = V(i)$).
  $\mathcal{M}, w \models i{:}\varphi$ iff $\mathcal{M}, v \models \varphi$ for $\{v\} = V(i)$.

Examples: $i, j, k \in$ NOM, $p \in$ PROP

- $\mathcal{M}, w \models i{:}p$
- $\mathcal{M}, w \models j{:}i{:}p$
- $\mathcal{M}, w \models j{:}k$
- $i \wedge p \rightarrow i{:}p$ is a tautology

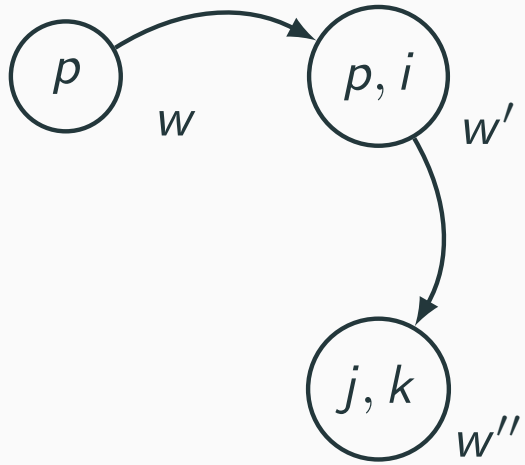


Nominals and Satisfaction Operators

- Extend the language with a new set of propositional symbols called nominals NOM $= \{i, j, k, \ldots\}$, and for each nominal i, a modal operator i: (or $@_i$) called a satisfaction operator.
- Nominals behave as constants because we demand they be true at a unique point in a model (i.e., $V(i)$ is a singleton).
- $\mathcal{M}, w \models i$ iff $w \in V(i)$ (equivalently $\{w\} = V(i)$).
  $\mathcal{M}, w \models i{:}\varphi$ iff $\mathcal{M}, v \models \varphi$ for $\{v\} = V(i)$.

Examples: $i, j, k \in$ NOM, $p \in$ PROP

- $\mathcal{M}, w \models i{:}p$
- $\mathcal{M}, w \models j{:}i{:}p$
- $\mathcal{M}, w \models j{:}k$
- $i \wedge p \rightarrow i{:}p$ is a tautology
- $i{:}p \rightarrow (i \wedge p)$ is not



Hybrid Rules

- The basic hybrid language $\mathcal{H}$ is more expressive than the basic modal language. E.g., $i \wedge \langle R \rangle i$ forces the point of evaluation to be reflexive.

Hybrid Rules

- The basic hybrid language $\mathcal{H}$ is more expressive than the basic modal language. E.g., $i \wedge \langle R \rangle i$ forces the point of evaluation to be reflexive. With the same complexity: SAT is PSPACE-complete for both.

Hybrid Rules

- The basic hybrid language $\mathcal{H}$ is more expressive than the basic modal language. E.g., $i \wedge \langle R \rangle i$ forces the point of evaluation to be reflexive. With the same complexity: SAT is PSPACE-complete for both.
- We can internalize labelled deduction.

$$\frac{i{:}\langle R \rangle \varphi}{\begin{array}{c} i{:}\langle R \rangle j \\ j{:}\varphi \end{array}} (\langle R \rangle) \qquad (j) \text{ a new nominal} \qquad \frac{\begin{array}{c} i{:}\neg\langle R \rangle \varphi \\ i{:}\langle R \rangle j \end{array}}{j{:}\neg\varphi} (\neg\langle R \rangle)$$

Hybrid Rules

- The basic hybrid language $\mathcal{H}$ is more expressive than the basic modal language. E.g., $i \wedge \langle R \rangle i$ forces the point of evaluation to be reflexive. With the same complexity: SAT is PSPACE-complete for both.
- We can internalize labelled deduction.

$$\frac{i{:}\langle R \rangle \varphi}{\begin{array}{c} i{:}\langle R \rangle j \\ j{:}\varphi \end{array}} (\langle R \rangle) \quad (j) \text{ a new nominal} \qquad \frac{\begin{array}{c} i{:}\neg\langle R \rangle \varphi \\ i{:}\langle R \rangle j \end{array}}{j{:}\neg\varphi} (\neg\langle R \rangle)$$

- Rules for : $\frac{i{:}j{:}\varphi}{j{:}\varphi}$ $(:)$ $\frac{i{:}\neg j{:}\varphi}{j{:}\neg\varphi}$ $(\neg{:})$

Hybrid Rules

- The basic hybrid language $\mathcal{H}$ is more expressive than the basic modal language. E.g., $i \wedge \langle R \rangle i$ forces the point of evaluation to be reflexive. With the same complexity: SAT is PSPACE-complete for both.
- We can internalize labelled deduction.

$$\frac{i{:}\langle R \rangle \varphi}{\begin{array}{c} i{:}\langle R \rangle j \\ j{:}\varphi \end{array}} (\langle R \rangle) \qquad (j) \text{ a new nominal} \qquad \frac{\begin{array}{c} i{:}\neg\langle R \rangle \varphi \\ i{:}\langle R \rangle j \end{array}}{j{:}\neg\varphi} (\neg\langle R \rangle)$$

- Rules for : $\frac{i{:}j{:}\varphi}{j{:}\varphi}$ $(:)$ $\frac{i{:}\neg j{:}\varphi}{j{:}\neg\varphi}$ $(\neg{:})$
- As a plus, the language has (limited) equality

$$\frac{}{i{:}i} (Ref) \qquad \frac{i{:}j}{j{:}i} (Sym) \qquad \frac{i{:}\varphi \quad i{:}k \quad j{:}k}{j{:}\varphi} (Cong)$$

Hybrid Rules

- The basic hybrid language $\mathcal{H}$ is more expressive than the basic modal language. E.g., $i \wedge \langle R \rangle i$ forces the point of evaluation to be reflexive. With the same complexity: SAT is PSPACE-complete for both.
- We can internalize labelled deduction.

$$\frac{i{:}\langle R \rangle \varphi}{\begin{array}{c} i{:}\langle R \rangle j \\ j{:}\varphi \end{array}} (\langle R \rangle) \qquad (j) \text{ a new nominal} \qquad \frac{\begin{array}{c} i{:}\neg\langle R \rangle \varphi \\ i{:}\langle R \rangle j \end{array}}{j{:}\neg\varphi} (\neg\langle R \rangle)$$

- Rules for : $\frac{i{:}j{:}\varphi}{j{:}\varphi}$ $(:)$ $\frac{i{:}\neg j{:}\varphi}{j{:}\neg\varphi}$ $(\neg{:})$
- As a plus, the language has (limited) equality

$$\frac{}{i{:}i} (Ref) \qquad \frac{i{:}j}{j{:}i} (Sym) \qquad \frac{i{:}\varphi \quad i{:}k \quad j{:}k}{j{:}\varphi} (Cong)$$

- (Start the tableau with $i{:}\varphi$ for i not in φ).

Hybrid Termination

- Once nominals and satisfaction operators are introduced, ensuring termination is more difficult.

Hybrid Termination

- Once nominals and satisfaction operators are introduced, ensuring termination is more difficult.
- An obvious problem is the *Cong* rule: $\dfrac{i{:}\varphi \quad i{:}k \quad j{:}k}{j{:}\varphi}\,(\mathit{Cong})$

Hybrid Termination

- Once nominals and satisfaction operators are introduced, ensuring termination is more difficult.
- An obvious problem is the *Cong* rule: $\dfrac{i{:}\varphi \;\; i{:}k \;\; j{:}k}{j{:}\varphi}\,(Cong)$
- The solution is to impose a "direction": Only one nominal in the equivalence class is saturated. But equality should still be an equivalence, so an irrestricted version for nominals is introduced.

$$\frac{i{:}\varphi \;\; i{:}k \;\; j{:}k}{j{:}\varphi}\,(wCong^{1}) \qquad \frac{i{:}n \;\; i{:}k \;\; j{:}k}{j{:}n}\,(Nom)$$

[1] j is the earliest introduced nominal making k true

Hybrid Termination

- Once nominals and satisfaction operators are introduced, ensuring termination is more difficult.
- An obvious problem is the *Cong* rule: $\dfrac{i{:}\varphi \;\; i{:}k \;\; j{:}k}{j{:}\varphi}\,(Cong)$
- The solution is to impose a "direction": Only one nominal in the equivalence class is saturated. But equality should still be an equivalence, so an irrestricted version for nominals is introduced.

$$\frac{i{:}\varphi \;\; i{:}k \;\; j{:}k}{j{:}\varphi}\,(wCong^{1}) \qquad \frac{i{:}n \;\; i{:}k \;\; j{:}k}{j{:}n}\,(Nom)$$

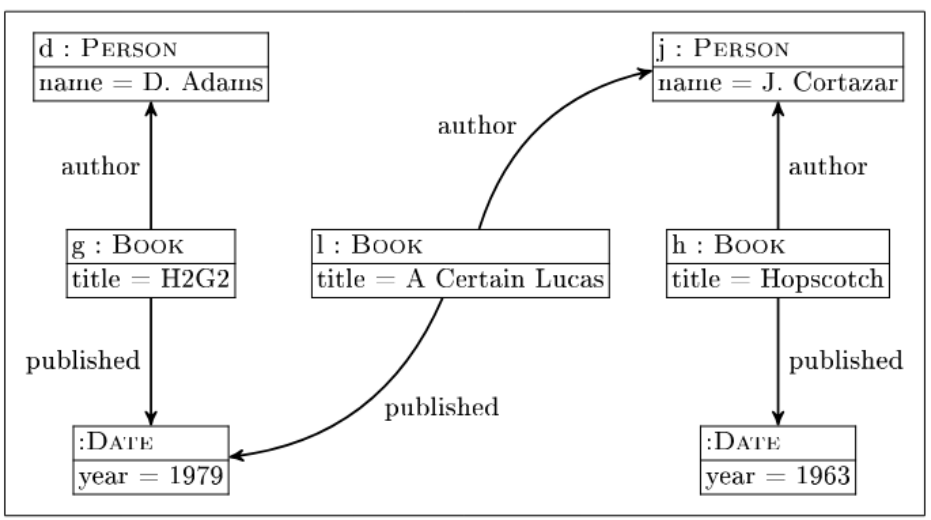
[1] j is the earliest introduced nominal making k true

Bolander, T. and Blackburn, P..
Termination for Hybrid Tableaus
Journal of Logic and Computation, 17, 517-554, 2007.

Semi-Structured Databases

- Many data-intensive applications use complex data that is not naturally encoded in a relational database.
  - Web data or biological data are better described using semi-structured data models, that organize information as labeled trees or graphs.
  - They contain labels from a finite alphabet (the structural information), and from an infinite alphabet (the actual data).
- A well known example is XML (eXtensible Markup Language), the most successful data model of this kind.
- XPath is the most widely used XML query language.

Consider the following picture that shows the information on a part of a semi-structured database containing details about books.



From the information in the picture we can infer that D. Addams and J. Cortazar published a book the same year (on 1979).

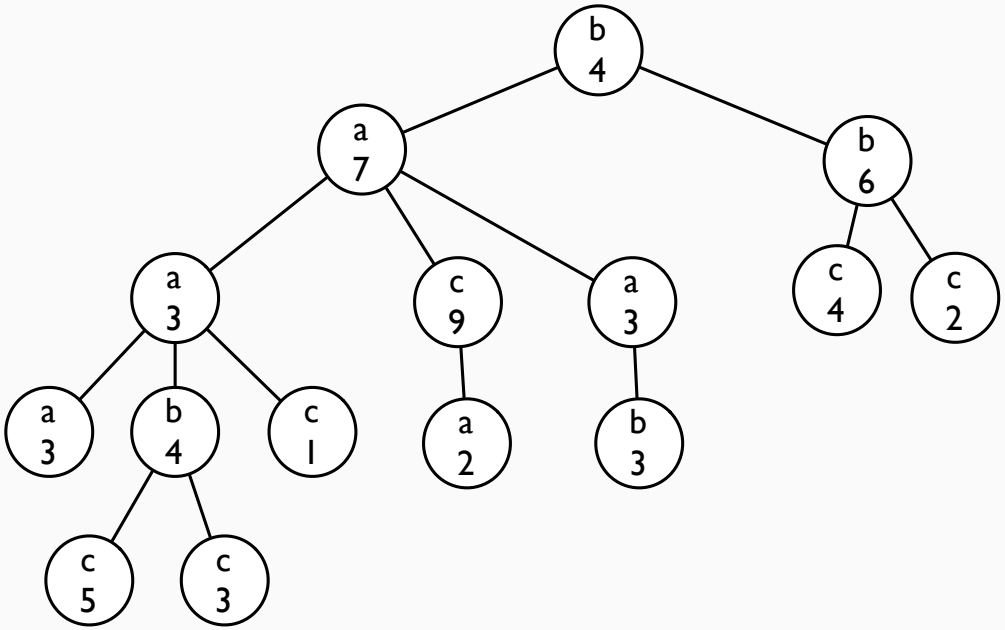
XPath with Data Tests

Data tree

- ordered tree (finite or infinite)
- nodes have
  - a label (finite alphabet)
  - a data (infinite domain)

Path expressions

- go to ancestor, check *a*, go to child, check *c*, go right sibling, go to descendant, check *b*



Node expressions

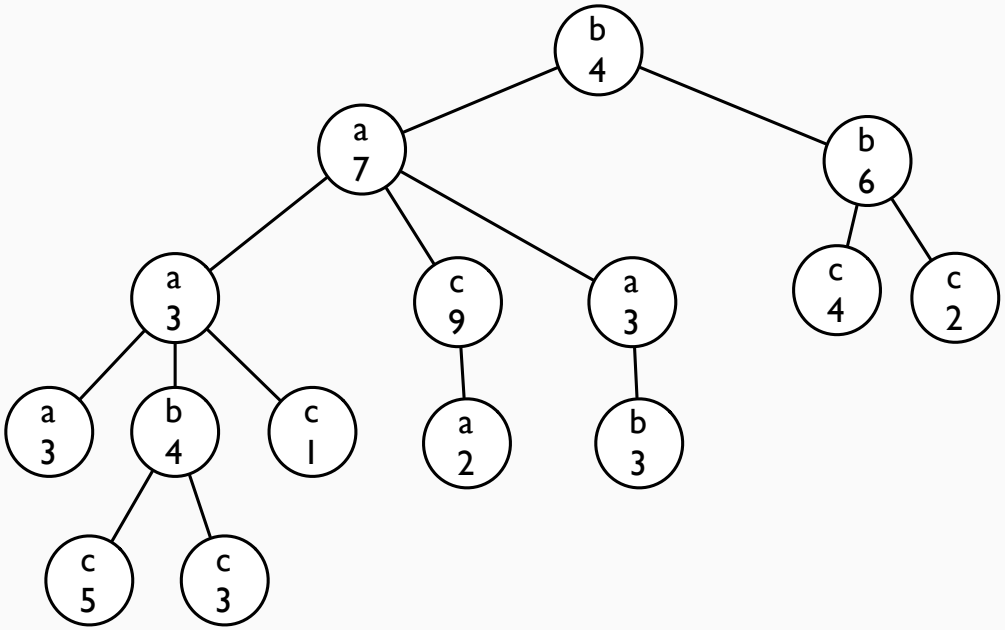
XPath with Data Tests

Data tree

- ordered tree (finite or infinite)
- nodes have
  - a label (finite alphabet)
  - a data (infinite domain)

Path expressions

- go to ancestor, check a, go to child, check c, go right sibling, go to descendant, check b
- $\uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b]$



Node expressions

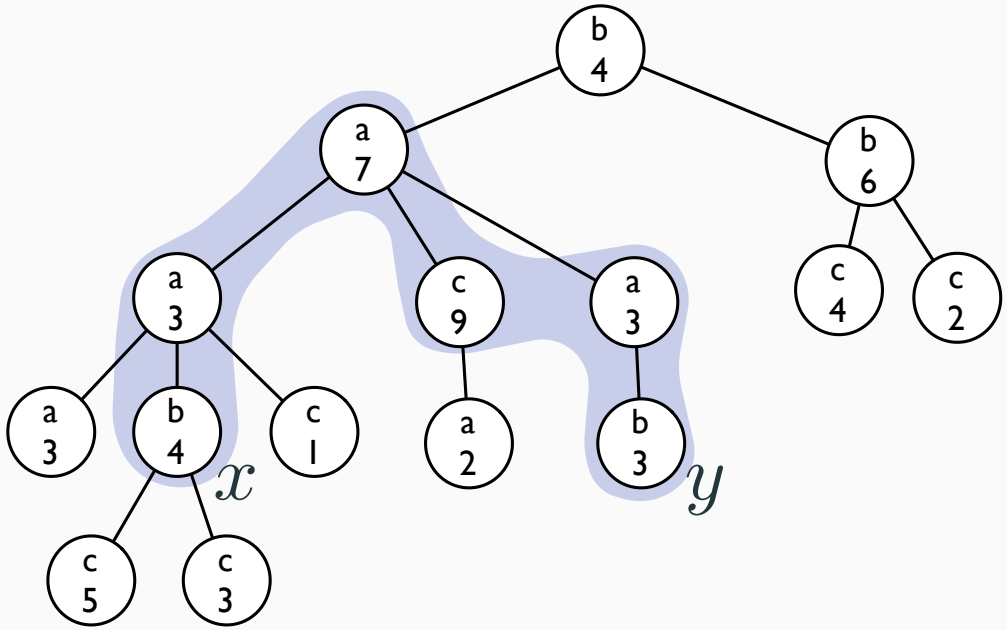
XPath with Data Tests

Data tree

- ordered tree (finite or infinite)
- nodes have
  - a label (finite alphabet)
  - a data (infinite domain)

Path expressions

- go to ancestor, check a, go to child, check c, go right sibling, go to descendant, check b
- $\uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b]$
- $\mathcal{T}, x, y \models \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b]$

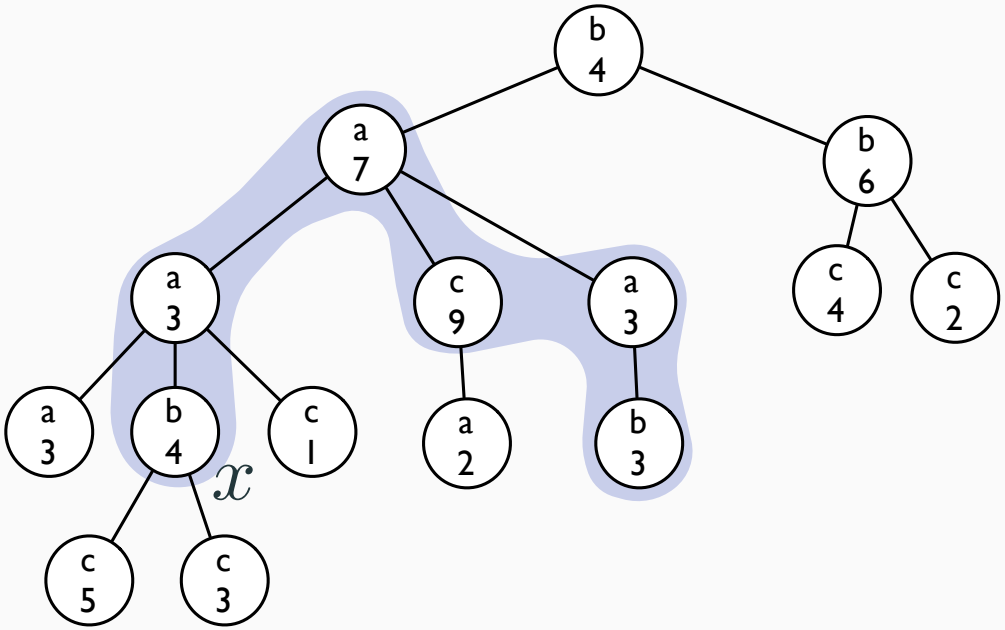


Node expressions

XPath with Data Tests

Data tree

- ordered tree (finite or infinite)
- nodes have
  - a label (finite alphabet)
  - a data (infinite domain)



Path expressions

- go to ancestor, check a, go to child, check c, go right sibling, go to descendant, check b
- $\uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b]$
- $\mathcal{T}, x, y \models \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b]$

Node expressions

- $\mathcal{T}, x \models \langle \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b] \rangle \top$

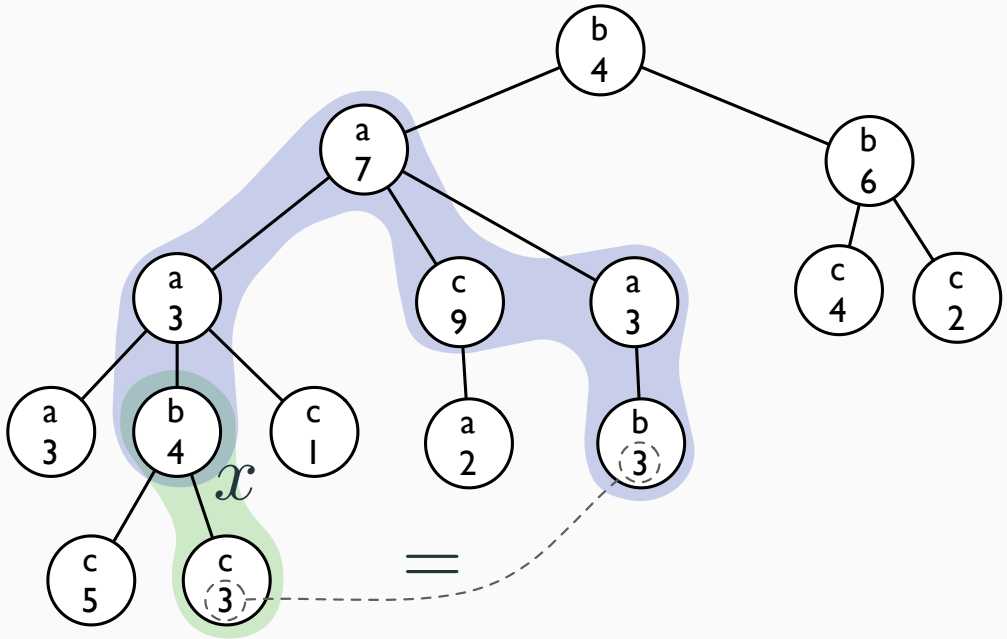
XPath with Data Tests

Data tree

- ordered tree (finite or infinite)
- nodes have
  - a label (finite alphabet)
  - a data (infinite domain)

Path expressions

- go to ancestor, check a, go to child, check c, go right sibling, go to descendant, check b
- $\uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b]$
- $\mathcal{T}, x, y \models \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b]$



Node expressions

- $\mathcal{T}, x \models \langle \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b] \rangle \top$
- $\mathcal{T}, x \models \langle \downarrow [c] = \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b] \rangle$

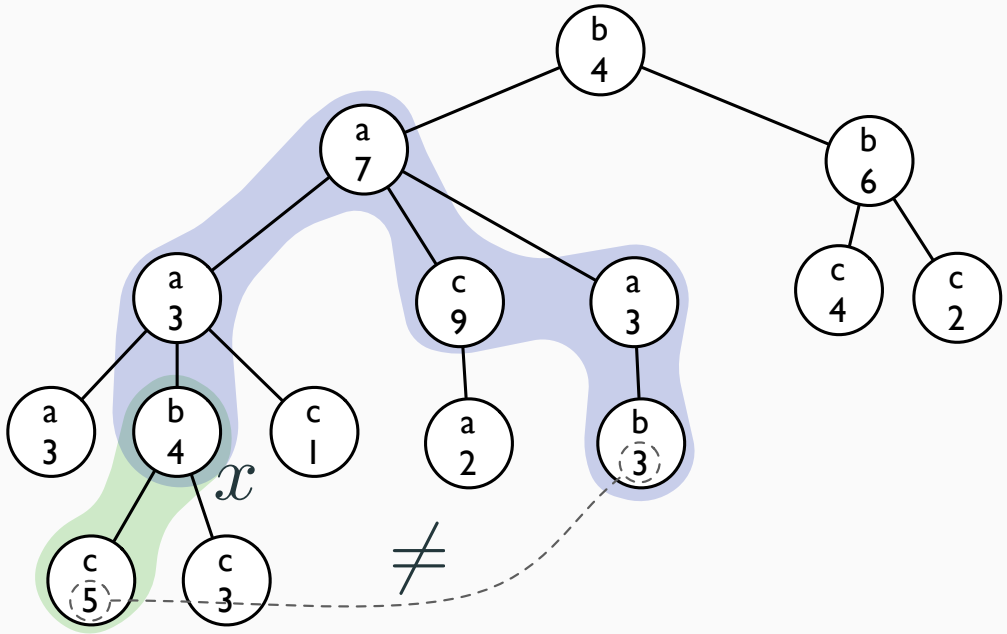
XPath with Data Tests

Data tree

- ordered tree (finite or infinite)
- nodes have
  - a label (finite alphabet)
  - a data (infinite domain)

Path expressions

- go to ancestor, check a, go to child, check c, go right sibling, go to descendant, check b
- $\uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b]$
- $\mathcal{T}, x, y \models \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b]$



Node expressions

- $\mathcal{T}, x \models \langle \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b] \rangle \top$
- $\mathcal{T}, x \models \langle \downarrow [c] = \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b] \rangle$
- $\mathcal{T}, x \models \langle \downarrow [c] \neq \uparrow^* [a] \downarrow [c] \rightarrow \downarrow_* [b] \rangle$

Diamonds in $\mathbf{XPath}_{=}$

Diamonds in $\mathrm{XPath}_{=}$

- $\mathcal{T}, x \models \langle \alpha = \beta \rangle$ iff there are nodes v and w such that
  - $\mathcal{T}, x, v \models \alpha$
  - $\mathcal{T}, x, w \models \beta$
  - $\mathit{data}(v) = \mathit{data}(w)$

Diamonds in $\mathrm{XPath}_{=}$

- $\mathcal{T}, x \models \langle \alpha = \beta \rangle$ iff there are nodes v and w such that
  - $\mathcal{T}, x, v \models \alpha$
  - $\mathcal{T}, x, w \models \beta$
  - $\mathit{data}(v) = \mathit{data}(w)$
- $\mathcal{T}, x \models \langle \alpha \neq \beta \rangle$ iff there are nodes v and w such that
  - $\mathcal{T}, x, v \models \alpha$
  - $\mathcal{T}, x, w \models \beta$
  - $\mathit{data}(v) \neq \mathit{data}(w)$

Formally. . .

- Node and Path Expressions

$$\alpha, \beta \;:=\; \delta \mid [\varphi] \mid \alpha\beta \mid \alpha \cup \beta$$
$$\varphi, \psi \;:=\; p \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle \alpha * \beta \rangle,$$

where $p \in \mathsf{Prop}$, $\delta \in \{\downarrow, \uparrow, \rightarrow, \downarrow^*, \uparrow^*\}$, $* \in \{=, \neq\}$.

Formally. . .

- Node and Path Expressions

$$\alpha, \beta \;:=\; \delta \mid [\varphi] \mid \alpha\beta \mid \alpha \cup \beta$$
$$\varphi, \psi \;:=\; p \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle \alpha * \beta \rangle,$$

  where $p \in \mathsf{Prop}$, $\delta \in \{\downarrow, \uparrow, \rightarrow, \downarrow^*, \uparrow^*\}$, $* \in \{=, \neq\}$.

- Semantics

$$\begin{array}{rcl}
\mathcal{M}, x \models p & \iff & p \in \mathit{label}(x) \\
\mathcal{M}, x \models \neg\varphi & \iff & \mathcal{M}, x \not\models \varphi \\
\mathcal{M}, x \models \varphi \wedge \psi & \iff & \mathcal{M}, x \models \varphi \text{ and } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \langle \alpha * \beta \rangle & \iff & \text{there are } y, z \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \\
 & & \text{and } \mathit{data}(y) * \mathit{data}(z)
\end{array}$$

Formally. . .

- Node and Path Expressions

$$\alpha, \beta \;:=\; \delta \mid [\varphi] \mid \alpha\beta \mid \alpha \cup \beta$$
$$\varphi, \psi \;:=\; p \mid \neg\varphi \mid \varphi \wedge \psi \mid \langle \alpha * \beta \rangle,$$

  where $p \in \mathsf{Prop}$, $\delta \in \{\downarrow, \uparrow, \rightarrow, \downarrow^*, \uparrow^*\}$, $* \in \{=, \neq\}$.

- Semantics

$$\begin{aligned}
\mathcal{M}, x \models p &\iff p \in \mathit{label}(x) \\
\mathcal{M}, x \models \neg\varphi &\iff \mathcal{M}, x \not\models \varphi \\
\mathcal{M}, x \models \varphi \wedge \psi &\iff \mathcal{M}, x \models \varphi \text{ and } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \langle \alpha * \beta \rangle &\iff \text{there are } y, z \in M \text{ s.t. } \mathcal{M}, x, y \models \alpha,\ \mathcal{M}, x, z \models \beta \\
&\qquad \text{and } \mathit{data}(y) * \mathit{data}(z) \\
\mathcal{M}, x, y \models \delta &\iff x \delta y \\
\mathcal{M}, x, y \models [\varphi] &\iff x = y \text{ and } \mathcal{M}, x \models \varphi \\
\mathcal{M}, x, y \models \alpha\beta &\iff \text{there is } z \in M \text{ s.t. } \mathcal{M}, x, z \models \alpha \text{ and } \mathcal{M}, z, y \models \beta \\
\mathcal{M}, x, y \models \alpha \cup \beta &\iff \mathcal{M}, x, y \models \alpha \text{ or } \mathcal{M}, x, y \models \beta
\end{aligned}$$

An Extension of the Basic Modal Language

- The basic modal operators are definable

$$\langle \alpha \rangle \varphi \leftrightarrow \langle \alpha[\varphi] = \alpha[\varphi] \rangle$$

An Extension of the Basic Modal Language

- The basic modal operators are definable

$$\langle \alpha \rangle \varphi \leftrightarrow \langle \alpha[\varphi] = \alpha[\varphi] \rangle$$

  In particular

$$\langle R \rangle \varphi \leftrightarrow \langle \downarrow[\varphi] = \downarrow[\varphi] \rangle$$

An Extension of the Basic Modal Language

- The basic modal operators are definable

$$\langle \alpha \rangle \varphi \leftrightarrow \langle \alpha[\varphi] = \alpha[\varphi] \rangle$$

In particular

$$\langle R \rangle \varphi \leftrightarrow \langle \downarrow[\varphi] = \downarrow[\varphi] \rangle$$

(But, of course, the language is more expressive).

Tableau Rules for XPath$_=$

- I will focus now in the fragment with only the $\downarrow$ axe (Xpath$_=(\downarrow)$).
- Consider a labelled tablea rule for $\langle\downarrow\alpha = \beta\rangle$

Tableau Rules for XPath$_=$

- I will focus now in the fragment with only the $\downarrow$ axe (Xpath$_=(\downarrow)$).
- Consider a labelled tablea rule for $\langle\downarrow\alpha = \beta\rangle$

$$\frac{i{:}\langle\downarrow\alpha = \beta\rangle}{\begin{array}{c} i \downarrow j \\ j{:}\langle\alpha = \beta\rangle \end{array}}$$

- Is this correct?

Tableau Rules for XPath$_=$

- I will focus now in the fragment with only the $\downarrow$ axe (Xpath$_=(\downarrow)$).
- Consider a labelled tablea rule for $\langle\downarrow\alpha = \beta\rangle$

$$\frac{i{:}\langle\downarrow\alpha = \beta\rangle}{\begin{array}{c} i \downarrow j \\ j{:}\langle\alpha = \beta\rangle \end{array}}$$

- Is this correct? No. In $j{:}\langle\alpha = \beta\rangle$ the β path also moved. And it should remain at i.

Tableau Rules for XPath$_=$

- I will focus now in the fragment with only the $\downarrow$ axe (Xpath$_=(\downarrow)$).
- Consider a labelled tablea rule for $\langle\downarrow\alpha = \beta\rangle$

$$\frac{i{:}\langle\downarrow\alpha = \beta\rangle}{\begin{array}{c} i \downarrow j \\ j{:}\langle\alpha = \beta\rangle \end{array}}$$

- Is this correct? No. In $j{:}\langle\alpha = \beta\rangle$ the β path also moved. And it should remain at i.
- We need to anchor β at i, i.e.,

$$\frac{i{:}\langle\downarrow\alpha = \beta\rangle}{\begin{array}{c} i \downarrow j \\ j{:}\langle\alpha = i{:}\beta\rangle \end{array}}$$

Tableau Rules for XPath$_=$

- I will focus now in the fragment with only the $\downarrow$ axe (Xpath$_=(\downarrow)$).
- Consider a labelled tablea rule for $\langle\downarrow\alpha = \beta\rangle$

$$\frac{i{:}\langle\downarrow\alpha = \beta\rangle}{\begin{array}{c} i \downarrow j \\ j{:}\langle\alpha = \beta\rangle \end{array}}$$

- Is this correct? No. In $j{:}\langle\alpha = \beta\rangle$ the β path also moved. And it should remain at i.
- We need to anchor β at i, i.e.,

$$\frac{i{:}\langle\downarrow\alpha = \beta\rangle}{\begin{array}{c} i \downarrow j \\ j{:}\langle\alpha = i{:}\beta\rangle \end{array}}$$

(notice that the second i: is different from the first one).

Hybrid XPath$_=$

- Add nominals and : to XPath$_=$, and call the resulting language HXPath$_=$.

Hybrid XPath$_=$

- Add nominals and : to XPath$_=$, and call the resulting language HXPath$_=$.
- Nominals are as before. They are propositional variables interpreted as singleton subsets of the domain.

$$\mathcal{M}, x \models i \iff i \in \mathit{label}(x)$$

Hybrid XPath$_=$

- Add nominals and : to XPath$_=$, and call the resulting language HXPath$_=$.
- Nominals are as before. They are propositional variables interpreted as singleton subsets of the domain.

$$\mathcal{M}, x \models i \iff i \in \mathit{label}(x)$$

- What is the : operator?

$$\mathcal{M}, x \models i{:}\varphi \iff \mathcal{M}, y \models \varphi \text{ for } y \text{ the extension of } i$$

Hybrid XPath$_=$

- Add nominals and : to XPath$_=$, and call the resulting language HXPath$_=$.
- Nominals are as before. They are propositional variables interpreted as singleton subsets of the domain.

$$\mathcal{M}, x \models i \iff i \in \mathit{label}(x)$$

- What is the : operator?

$$\mathcal{M}, x \models i{:}\varphi \iff \mathcal{M}, y \models \varphi \text{ for } y \text{ the extension of } i$$

- But we need a path version of :

$$\mathcal{M}, x, y \models @_i \iff i \in \mathit{label}(y)$$

Hybrid XPath$_=$

- Add nominals and : to XPath$_=$, and call the resulting language HXPath$_=$.
- Nominals are as before. They are propositional variables interpreted as singleton subsets of the domain.

$$\mathcal{M}, x \models i \iff i \in \mathit{label}(x)$$

- What is the : operator?

$$\mathcal{M}, x \models i{:}\varphi \iff \mathcal{M}, y \models \varphi \text{ for } y \text{ the extension of } i$$

- But we need a path version of :

$$\mathcal{M}, x, y \models @_i \iff i \in \mathit{label}(y)$$

- And actually

$$i{:}\varphi \leftrightarrow \langle @_i[\varphi] = @_i[\varphi] \rangle$$

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths	Formulas
$@_i\alpha$	There exists an α path between the node named i and some other node.

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths	Formulas
$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

	Paths Formulas
$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths Formulas

$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths Formulas

$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.

Node Formulas

$\langle @_i = @_i \rangle$

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths Formulas

$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.

Node Formulas

$\langle @_i = @_i \rangle$	Always valid.

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths Formulas

$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.

Node Formulas

$\langle @_i = @_i \rangle$	Always valid.
$\langle [i] = [i] \rangle$	

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths Formulas

$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.

Node Formulas

$\langle @_i = @_i \rangle$	Always valid.
$\langle [i] = [i] \rangle$	The current node is named i.

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths Formulas

$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.

Node Formulas

$\langle @_i = @_i \rangle$	Always valid.
$\langle [i] = [i] \rangle$	The current node is named i.
$\langle @_i = @_j \rangle$	

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths Formulas

$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.

Node Formulas

$\langle @_i = @_i \rangle$	Always valid.
$\langle [i] = [i] \rangle$	The current node is named i.
$\langle @_i = @_j \rangle$	The node named i has the same data than the node named j.

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths Formulas

$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.

Node Formulas

$\langle @_i = @_i \rangle$	Always valid.
$\langle [i] = [i] \rangle$	The current node is named i.
$\langle @_i = @_j \rangle$	The node named i has the same data than the node named j.
$\langle \alpha = @_i\beta \rangle$	

Some Examples

Some HXPath$_=$ expressions together with their intuitive meaning:

Paths Formulas

$@_i\alpha$	There exists an α path between the node named i and some other node.
$\alpha@_i$	There exists an α path between the current node and some other node, evaluation continues at point named i.
$[i]\alpha$	The current node is named i and there exists an α path to some node.
$\alpha[i]$	There exists an α path between the current node and the node named i.

Node Formulas

$\langle @_i = @_i \rangle$	Always valid.
$\langle [i] = [i] \rangle$	The current node is named i.
$\langle @_i = @_j \rangle$	The node named i has the same data than the node named j.
$\langle \alpha = @_i\beta \rangle$	A node accessible from the current node by an α path has the same data than a node accessible from the point named i by a β path.

Tableau Rules: Boolean, @ and Nominals

$$\frac{n{:}\neg\neg\varphi}{n{:}\varphi}\ (\neg\neg) \qquad \frac{n{:}(\varphi \wedge \psi)}{\begin{array}{l} n{:}\varphi \\ n{:}\psi \end{array}}\ (\wedge) \qquad \frac{n{:}\neg(\varphi \wedge \psi)}{n{:}\neg\varphi \quad \mid \quad n{:}\neg\psi}\ (\neg\wedge)$$

Tableau Rules: Boolean, @ and Nominals

$$\frac{n{:}\neg\neg\varphi}{n{:}\varphi}\ (\neg\neg) \qquad \frac{n{:}(\varphi \wedge \psi)}{\begin{array}{l} n{:}\varphi \\ n{:}\psi \end{array}}\ (\wedge) \qquad \frac{n{:}\neg(\varphi \wedge \psi)}{n{:}\neg\varphi \quad \mid \quad n{\neg}\psi}\ (\neg\wedge)$$

$$\frac{i{:}@_j\varphi}{j{:}\varphi}\ (@) \qquad \frac{i{:}\neg@_j\varphi}{j{:}\neg\varphi}\ (\neg@)$$

$$\frac{}{n{:}n}\ (Ref^2) \qquad \frac{n{:}i}{i{:}n}\ (Sym)$$

$$\frac{n{:}\varphi \quad n{:}i \quad m{:}i}{m{:}\varphi}\ (wCong^3) \qquad \frac{n{:}j \quad n{:}i \quad m{:}i}{m{:}j}\ (Nom)$$

[2] n appears in the tableau.

[3] m is the smallest nominal making i true in the tableau.

Tableau Rules: Prefix

$$\frac{n{:}\langle \alpha * \beta \rangle}{\langle @_n \alpha * @_n \beta \rangle}\ (\mathit{Int}) \qquad \frac{n{:}\neg\langle \alpha * \beta \rangle}{\neg\langle @_n \alpha * @_n \beta \rangle}\ (\neg\mathit{Int}) \qquad \frac{n{:}m}{\langle @_n = @_m \rangle}\ (\mathit{IData})$$

- The satisfaction operator is moved from the node formula to the path formula to anchor each path.
- Notice that the result is an (unlabelled) XPath diamond.
- It only remains to give rules for the different path operators that can occur inside a diamond.

Tableau Rules: Down

$$\frac{\langle @_n{\downarrow}\alpha * \beta\rangle}{\begin{array}{c} n \to m \\ \langle @_m\alpha * \beta\rangle \end{array}} (\downarrow^1) \qquad \frac{\neg\langle @_n{\downarrow}\alpha * \beta\rangle \quad n \to m}{\neg\langle @_m\alpha * \beta\rangle} (\neg\downarrow)$$

[1] m is the smallest nominal that has not appeared in the tableau.

Tableau Rules: Tests

$$\frac{\langle @_n[\varphi]\alpha * \beta\rangle}{\begin{array}{c} n{:}\varphi \\ \langle @_n\alpha * \beta\rangle \end{array}}\,([\,]) \qquad \frac{\neg\langle @_n[\varphi]\alpha * \beta\rangle}{n{:}\neg\varphi \;\mid\; \neg\langle @_n\alpha * \beta\rangle}\,(\neg[\,])$$

Tableau Rules: Union

$$\frac{\langle @_n(\alpha \cup \beta)\gamma * \delta\rangle}{\langle @_n\alpha\gamma * \delta\rangle \mid \langle @_n\beta\gamma * \delta\rangle}\ (\cup) \qquad \frac{\neg\langle @_n(\alpha \cup \beta)\gamma * \delta\rangle}{\begin{array}{c}\neg\langle @_n\alpha\gamma * \delta\rangle \\ \neg\langle @_n\beta\gamma * \delta\rangle\end{array}}\ (\neg\cup)$$

Tableau Rules: At

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

Tableau Rules: Data

$$\frac{\neg\langle @_n \neq @_m \rangle}{\langle @_n = @_m \rangle}\,(DEq)$$

$$\frac{\langle @_n = @_m \rangle}{\langle @_m = @_n \rangle}\,(DSym)$$

$$\frac{\langle @_n = @_m \rangle \;\; \langle @_m = @_k \rangle}{\langle @_n = @_k \rangle}\,(DTrans)$$

Tableau Rules: Commutativity

$$\frac{\langle @_n * \varphi \rangle}{\langle \varphi * @_n \rangle}\ (\text{COM})$$

$$\frac{\neg\langle @_n * \varphi \rangle}{\neg\langle \varphi * @_n \rangle}\ (\neg\text{-COM})$$

Starting and Clashing

- To check the satisfiablity of φ start the tableaux with $i{:}\varphi$.

Starting and Clashing

- To check the satisfiablity of φ start the tableaux with $i{:}\varphi$.
  To check the satisfiability of α start the tableaux with $i{:}\langle\alpha\rangle\top$

Starting and Clashing

- To check the satisfiablity of φ start the tableaux with $i{:}\varphi$.
  To check the satisfiability of α start the tableaux with $i{:}\langle\alpha\rangle\top$
- A branch Θ of a tableau contains a clash if one of the following conditions holds:
  1. $\{n{:}a, n{:}\neg a\} \subseteq \Theta$
  2. $\langle @_n \neq @_n\rangle \in \Theta$,
  3. $\neg\langle @_n = @_n\rangle \in \Theta$,
  4. $\{\langle @_n = @_m\rangle, \langle @_n \neq @_m\rangle\} \subseteq \Theta$,
  5. $\{\langle @_n = @_m\rangle, \neg\langle @_n = @_m\rangle\} \subseteq \Theta$,

  for some $n,\ m \in \mathrm{NOM}$, $a \in \mathrm{NOM} \cup \mathrm{PROP}$.

Final Comments

- It is not difficult to prove that the calculus is sound, complete and terminating.

Final Comments

- It is not difficult to prove that the calculus is sound, complete and terminating.
- It is also possible to define strategies that ensure it runs in PSPACE. (But things are more complicated because we have to do DFS on two branches at the same time.)

Final Comments

- It is not difficult to prove that the calculus is sound, complete and terminating.
- It is also possible to define strategies that ensure it runs in PSPACE. (But things are more complicated because we have to do DFS on two branches at the same time.)
- Applications: Query rewriting and optimization.

Final Comments

- It is not difficult to prove that the calculus is sound, complete and terminating.
- It is also possible to define strategies that ensure it runs in PSPACE. (But things are more complicated because we have to do DFS on two branches at the same time.)
- Applications: Query rewriting and optimization.
- We are investigating extensions:
  - Handling $\uparrow$ is easy for soundness and completeness. Termination needs blocking. Exact complexity is open (conjecture: ExpTime-complete).
  - Now looking at transitive closure, siblings, and other data comparison (e.g., $\leq$).

Final Comments

- It is not difficult to prove that the calculus is sound, complete and terminating.
- It is also possible to define strategies that ensure it runs in PSPACE. (But things are more complicated because we have to do DFS on two branches at the same time.)
- Applications: Query rewriting and optimization.
- We are investigating extensions:
  - Handling $\uparrow$ is easy for soundness and completeness. Termination needs blocking. Exact complexity is open (conjecture: ExpTime-complete).
  - Now looking at transitive closure, siblings, and other data comparison (e.g., $\leq$).
- Implementations.

Future Work

- Interpolation.
- Probabilistic models.
- Data aggregation (max, avg, sum)
- We have a nice sequent calculus, what about natural deduction?
- Transitive closure over trees.