Satisfiability checking in presence of DTDs capturing well-typed references

Nicole Bidoit
LRI-Université Paris XI, Orsay
bidoit@lri.fr

References for this talk

- Preliminary study in collaboration with S. Cerrito and V. Thion.
  - `A first step towards modeling semistructured data in hybrid multimodal logic`, *Journal of Applied Non-Classical logic*, Volume 14, No 4/2004.
- Recent results are joint work with **Dario Colazzo**
  - `Capturing well typed references in DTDs`,
  - `Testing XML constraint satisfiability`,

Motivation

• few investigations for typing references of semistructured data and XML documents.

▶ REF and IDREF attributes

▶ key and foreign-key constraints

▶ XML Schema uses XPath to specify typed references

↪ requires a good amount of expertize to be used correctly

↪ reasoning about constraints defined with XPath is highly intricate, if not impossible.

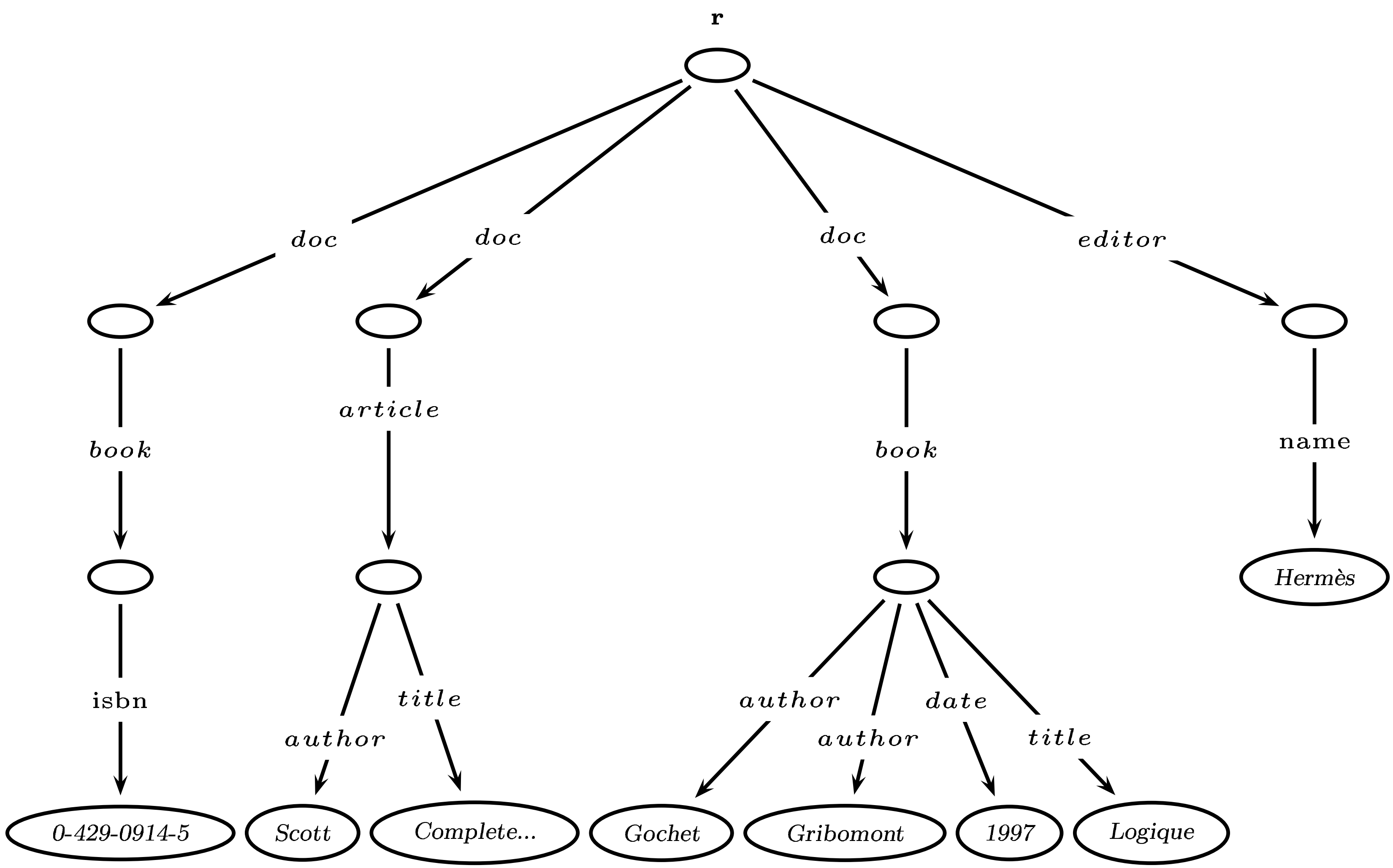


Figure 1: The well-known library example **without references**

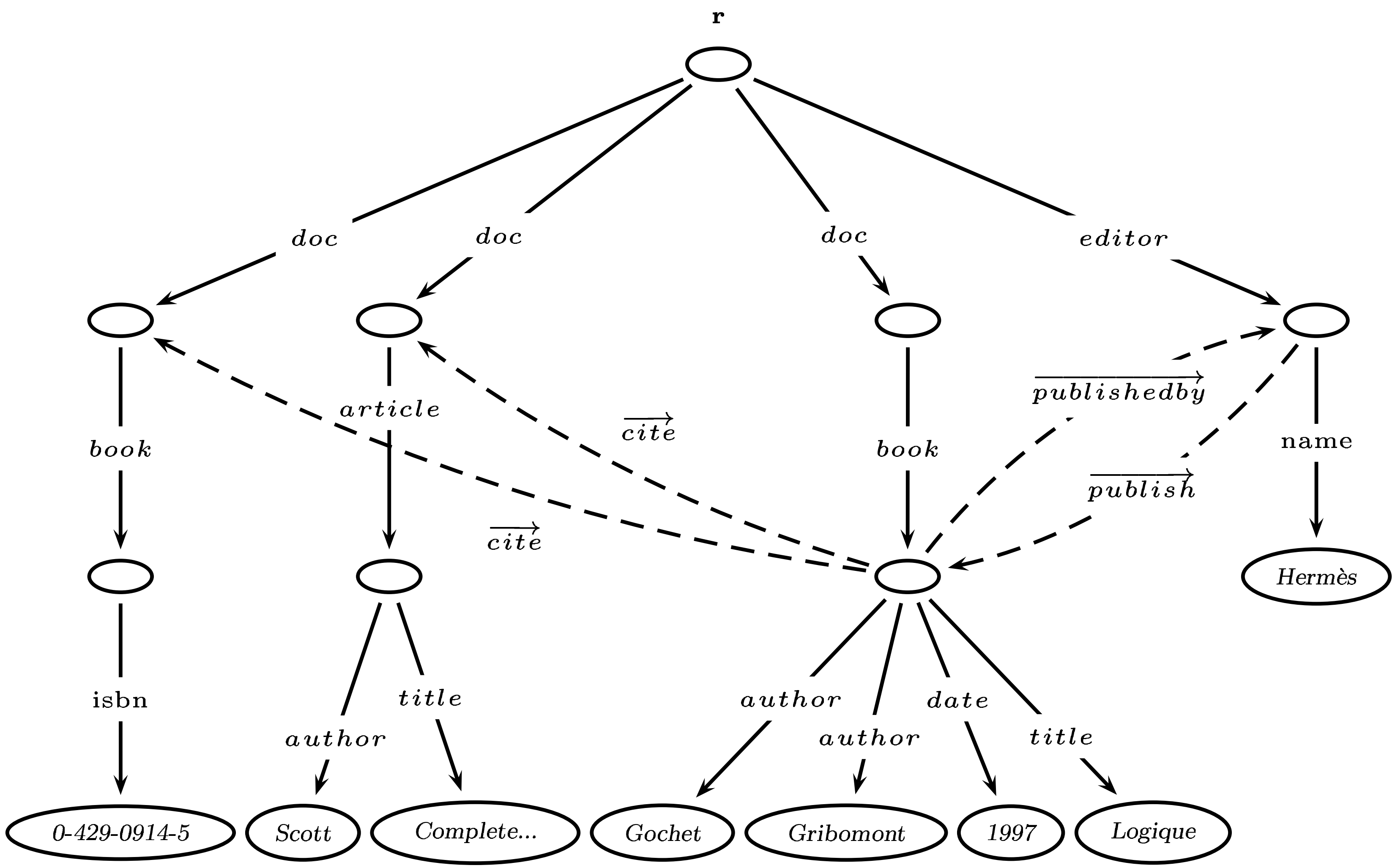


Figure 2: The well-known library example **with references**

Goal and Approach

- extension of DTDs / schemas to capture well typed references
- a unique formalism for schemas, constraints and queries:
  Hybrid Modal Logic

Why a unique formalism ?

- ▶ subtyping, constraint implication and satisfiability,
- ▶ query correctness, optimization

Why Modal logic ?

Why modal logic ?

- Modal propositional logics

  simple languages for talking about any kind of graphs

  tree-structures, transition networks, parse trees,

  networks of properties, ontologies, flows of time, ...

  possible worlds

- Usefuf in a wide range of applications

  (simple syntax, often decidable)

  logics of time, computation, parsing, ... linguistics

- relational structures are ubiquitous

- relational structures are models of classical model theory

  Modal logic is a (decidable) fragment of classical logic

Semistructured document

- a document is a labelled graph (labels over edges)
- a Kripke model is a labelled graph (interpretation for modal logic)

- a document is a Kripke model

A *model* (**document**) $\mathfrak{M}$ is a **Kripke structure** (S, r, R, V):

▶ S is a finite set of *states* (nodes of the document)

▶ r is a distinguished element (root of the document),

▶ $R = \{r_e | e \in \mathcal{E}\}$ is a set of binary *accessibility relations* on S (labelled links of the document),

▶ $V{:}PROP \rightarrow Pow(S)$ assignes to each proposition p the set of states where p holds (data component);

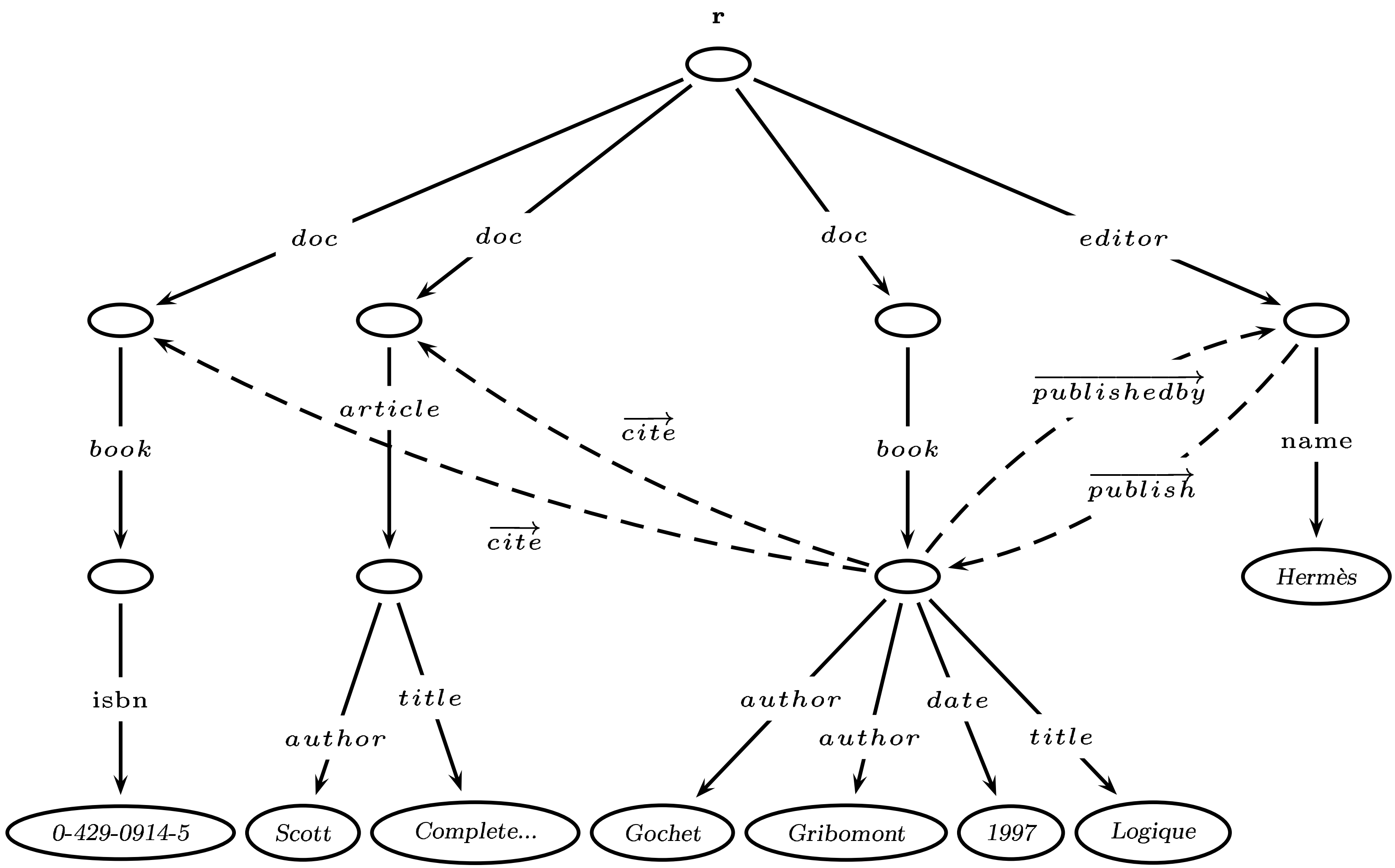


Figure 3: The well-known library example

Modal logic and Semistructured Data: Related works

- Schemes subsumption
  - ▶ hybrid modal logic [Alechina 97]
  - ▶ description logic [Calvanese&all 98]
- DTDs encoded into a PDL-like description logic [Calvanese&all 99]
- Query languages
  - ▶ TQL based on ambient logic [Cardelli,Ghelli 01]
  - ▶ Xpath fragments vs CTL [Miklau,Suciu 02] [Gottlob, Koch 02]
  - ▶ Xpath queries equivalence vs PDL [Marx 03]
- Constraints
  - ▶ Path constraints vs Converse PDL [Alechina 03]
  - ▶ Path constraints vs HML [Franceschet, de Rijke 03]

(Modal) Logics for Semistructured data [Demri 03]

(invited talk at M4M-3)

Organization of this talk

1. **Ref-schema** capturing well typed references
2. **Hybrid Modal Logic** (HML): an introduction.
3. How **ref-schema** are expressed in **HML**
4. Checking **constraint satisfiability** in presence of ref-schemas.
5. Discussion and further research directions

Schema capturing well-typed references : an example

$$
\begin{array}{lcl}
\textbf{Start} & ::= & (doc\ Doc)^*, (editor\ Editor)^* \\
Editor & ::= & (name\ Name)^!, (\overrightarrow{publish}\ Book)^* \\
Doc & ::= & (article\ Art)^! + (book\ Book)^! \\
Art & ::= & (author\ Name)^+, (title\ Name)^!, (date\ Dat)^?, (\overrightarrow{cite}\ Doc)^* \\
Book & ::= & (isbn\ Isb)^!, (\overrightarrow{cite}\ Doc)^* + ((author\ Name)^+, (date\ Dat)^! \\
 & & (title\ Name)^!, (\overrightarrow{cite}\ Doc)^*, (\overrightarrow{publisedby}\ Editor)^!) \\
Name & ::= & \Lambda \\
Dat & ::= & \Lambda \\
Isb & ::= & \Lambda
\end{array}
$$

↑ non terminal symbols

↑ regular expression

Non terminal symbols: Start, *Editor*, *Doc*, *Art*, ...

Labels : *name*, *article*, *author*, ... — child labels

$\overrightarrow{publish}$, $\overrightarrow{cite}$, $\overrightarrow{publisedby}$ — references

Schema capturing well-typed references : ref-schema

- The set of labels $\mathcal{E}$ is partionned:
  - ▶ labels in E are called *child* labels
  - ▶ labels in $\overrightarrow{E}$ are called *references*.
- $\mathcal{V}$ is a set of non terminal symbols among which $Start$ and Λ.

- A ref-schema $\mathcal{G}$ is given by a typing function θ:
  - ▶ $\theta(X)$ is a regular expression of the form:

$$R ::= \tilde{e}X \mid R + R \mid R, R \mid R* \mid \Lambda.$$

  - ▶ θ satisfies that:

  (1) if e is a child-label, then there exists a unique non terminal symbol X such that the pattern $(e\ X)$ appears in $\mathcal{G}$

  (2) for each non terminal $X \neq$Start, $Start \Rightarrow_{\mathcal{G}}^{*} X$ holds.

Schema capturing well-typed references : a second example

$$Start := (e\ X)^*,$$
$$X := (\ p\ Y\ +\ o\ Z)^*, (\overrightarrow{r}\ Y\ +\ \overrightarrow{r}\ Z)^*$$
$$Y := (e\ X, e\ X)^*$$
$$Z := (e\ X, e\ X)^*,\ e\ X$$

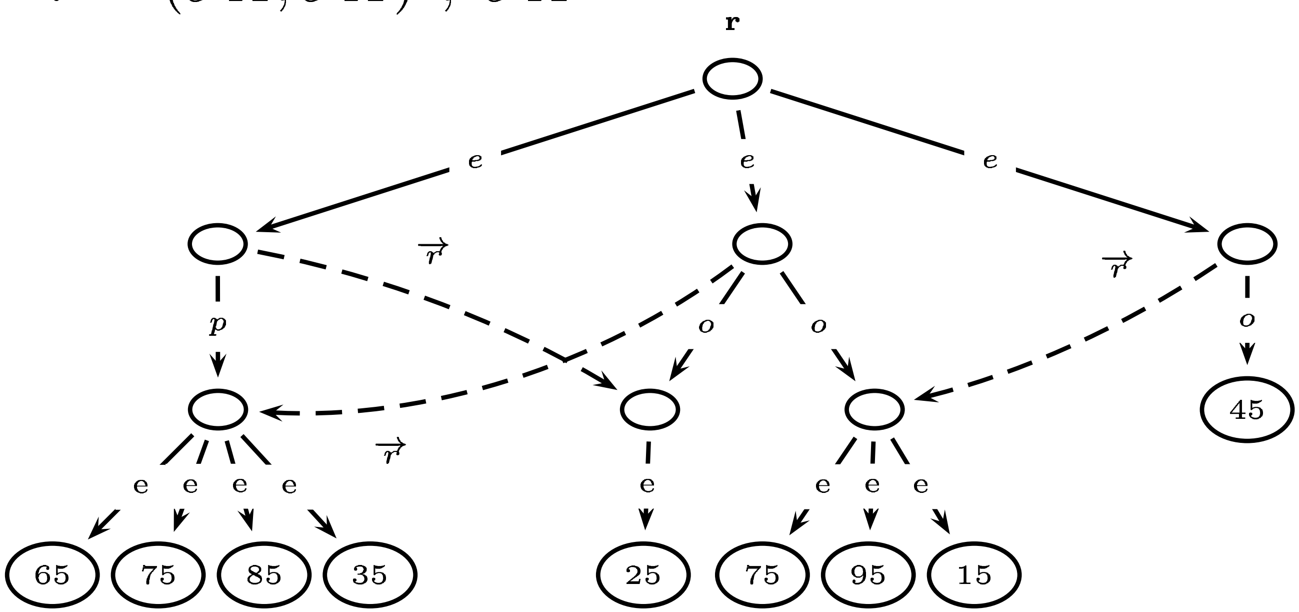


Figure 4: A document conforming to the ref-schema

Ref-Schema validation

A document $\mathfrak{M} = (S, r, R, V, \mathcal{I}_{nom})$ satisfies the ref-schema $\mathcal{G} = (\mathcal{E}, \mathcal{V}, Start, \theta)$, denoted $\mathfrak{M} : \mathbf{G}$, if:

(1) $\mathfrak{M}$ "restricted to" the child label edges e is a tree;

(2) there exists a total mapping $\vartheta : S \to \mathcal{V}$ such that:

(a) $\vartheta(r) = Start$,

(b) forall $n \in S$ if $\vartheta(n) = X$ and $\theta(X) = R$ then

$$\{\{\tilde{e}Y \mid (n, n') \in R_{\tilde{e}} \; and \; Y = \vartheta(n')\}\} \in [\![R]\!]$$

Extension of regular expressions:

$$
\begin{array}{lcl}
[\![\tilde{e}X]\!] & = & \{\{\{\tilde{e}X\}\}\} \\
[\![R_1 + R_2]\!] & = & [\![R_1]\!] \cup [\![R_2]\!] \\
[\![R_1, R_2]\!] & = & \{\{b_1 \uplus b_2 \mid b_i \in [\![R_i]\!]\}\} \\
[\![R*]\!] & = & \{\{b_1 \uplus \ldots \uplus b_n \mid b_i \in [\![R]\!],\; n > 0\}\} \cup \{\{\{\}\}\} \\
[\![\Lambda]\!] & = & \{\{\{\}\}\}
\end{array}
$$

Organization of this talk

1. **Ref-schema** capturing well typed references
2. **Hybrid Modal Logic** (HML): an introduction.
3. How **ref-schema** are expressed in **HML**
4. Checking **constraint satisfiability** in presence of ref-schemas.
5. Discussion and further research directions

From modal to hybrid modal logic

- Syntax

  a set of propositional symbols $p, q, \ldots,$

  conjunction $\wedge$, negation $\neg$,

  the modality $[e]$ where $e \in \mathcal{E}$

- Semantics : an internal and local perspective

  To evaluate satisfaisability of a formula

  ▶ choose a node s inside the model $\mathfrak{M}$

  ▶ navigate from this node to the accessible ones

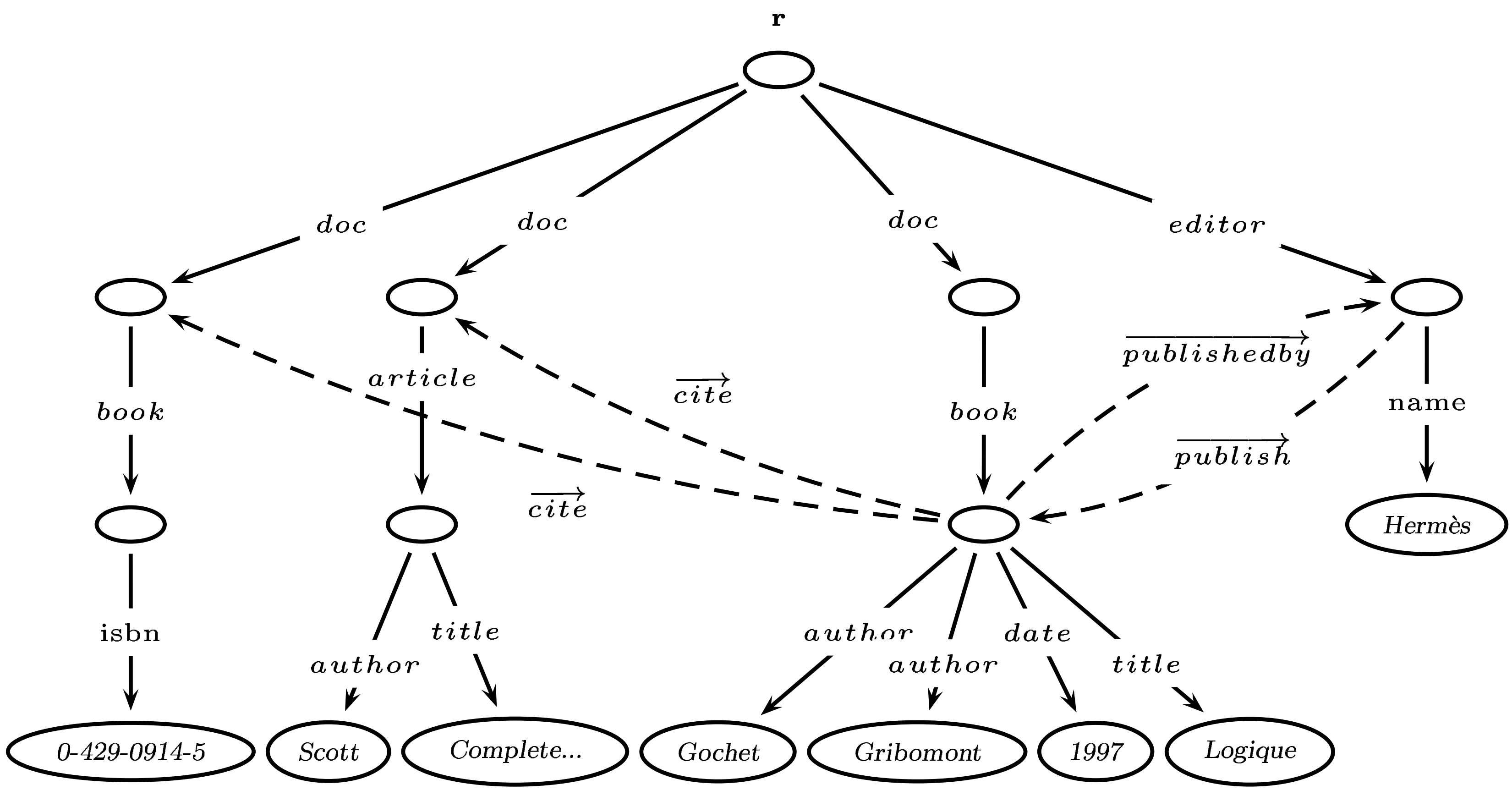
  $\mathfrak{M}, g, s \models [e]\psi$ iff $\forall s'$ such that $(s, s') \in r_e$ we have $\mathfrak{M}, g, s' \models \psi$

  $\mathfrak{M}, g, s \models \langle e \rangle\psi$ iff $\exists s'$ such that $(s, s') \in r_e$ with $\mathfrak{M}, g, s' \models \psi$

- Other modalities (behond first order)

  G : accessibility via all path **F** : accessibility via one path

Modal logic: an example



$$\mathfrak{M}_{library}, \quad r \quad \models \quad [doc]\ [book]\ ([\overrightarrow{publishedby}]\ \langle\overrightarrow{publish}\rangle)$$

From modal to hybrid modal logic

- Modal Logics : What exactly is missing ?
  - ▶ Nodes (states) are at the heart of modal logic
  - ▶ But not really ... nothing to grip with them

Example : No e-labelled edge from the node s to itself

$\neg\langle e\rangle$??

- One need to deal with nodes **explicitly**

  $\hookrightarrow$ Hybrid Modal Logics [Blackburn]

- Syntax

nominals: names for nodes	atomic formulas
state variables: capturing nodes	atomic formulas
binder $\downarrow x$: binds x to the current node	new modality
at operator $@_x$: move to the node x	new modality

Example : No e-labelled edge from the node s to itself

$\downarrow x \ \ \neg\langle e\rangle x$

From modal to hybrid modal logic

- Back to Kripke structure

▶ a unique nominal *root* to name the root r of a document

A *model* (**document**) $\mathfrak{M}$ is a **Kripke structure** $(S, r, R, V, \mathcal{I}_{nom})$

▶ $\mathcal{I}_{nom}(root) = r$ is the interpretation for nominals.

- Semantics of hybrid features

  g is a valuation of state variables

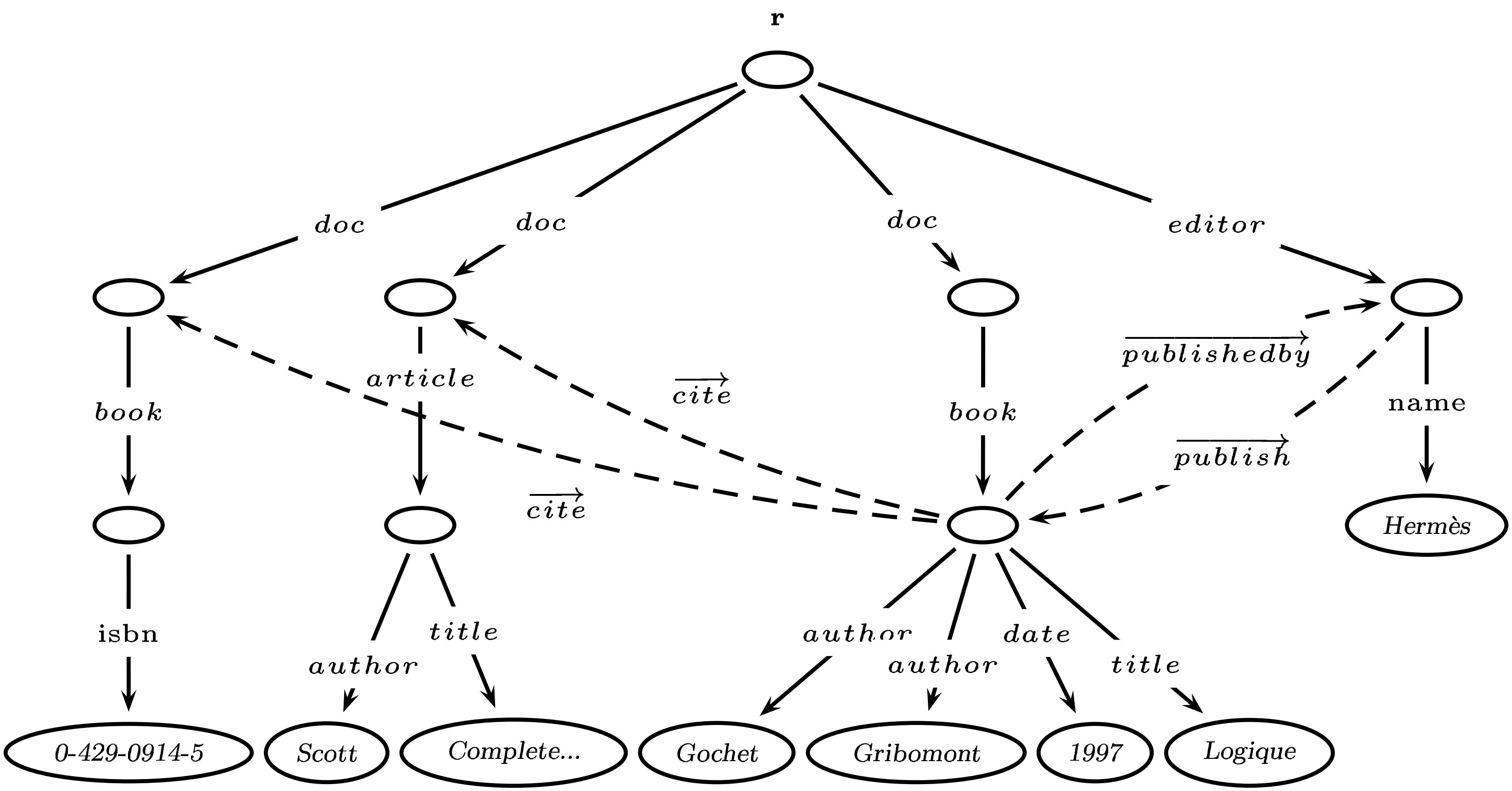
  $\mathfrak{M}, g, s \models a$ iff $I_{nom}(a) = s$ (a is a nominal)

  $\mathfrak{M}, g, s \models x$ iff $g(x) = s$ (x is a state variable)

  $\mathfrak{M}, g, s \models \downarrow x\ \psi$ iff $\mathfrak{M}, g', s \models \psi$ with $g \stackrel{x}{\sim} g'$ and $g'(x) = s$

  $\mathfrak{M}, g, s \models @_x \psi$ iff $\mathfrak{M}, g, g(x) \models \psi$

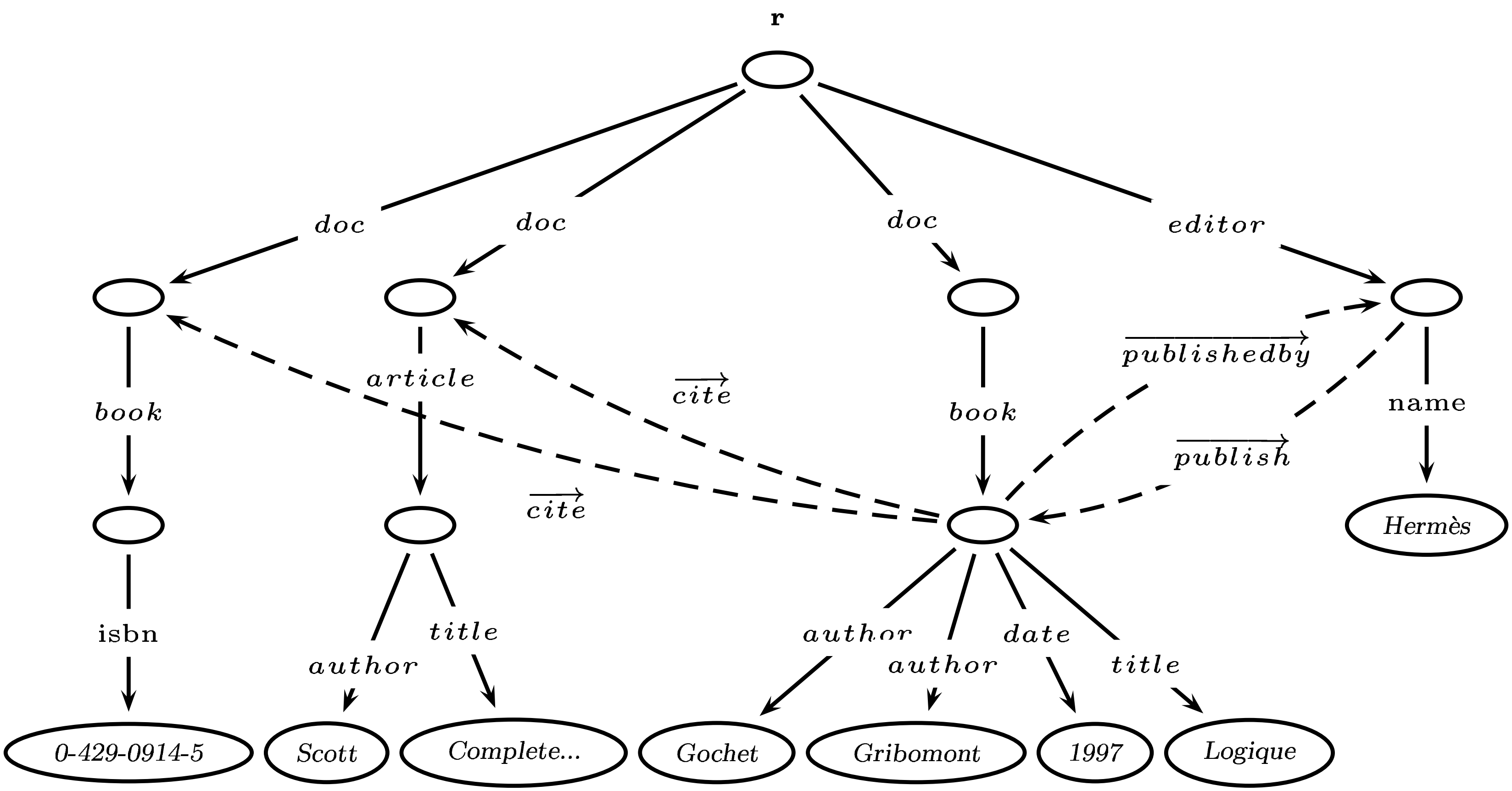
HML: an example



$$\mathfrak{M}_{library}, g, \quad r \quad \models \quad @_{root}[doc][book]{\downarrow}x \ ([\overrightarrow{publishedby}]\langle\overrightarrow{publish}\rangle x)$$

given any book x, if x is published by y then y publishes x

Modal logic: an example



$$\mathfrak{M}_{library}, g, \quad r \quad \models \quad @_{root}[doc][book]{\downarrow}x\ (\langle isbn\rangle{\downarrow}y\ (@_x[isbn]y))$$

a book has exactly one isbn number.

Constraints and Hybrid Modal Logic

- Result: HML is strictly more expressive than the language $\mathcal{P}$ devised to define **forward and backward constraints**.

Example: `‘‘given any book` x`, if` x `is published by` y `then` y `publishes` x`’’`.

$@_{root}[doc][book]{\downarrow}x\ ([\overrightarrow{publishedby}]\langle\overrightarrow{publish}\rangle x)$

expressible in $\mathcal{P}$ $\Longrightarrow$ expressible in HML

Example: `a book has exactly one isbn number`.

$@_{root}[doc][book]{\downarrow}x\ (\langle isbn\rangle{\downarrow}y\ (@_x[isbn]y))$

expressible in HML but not in $\mathcal{P}$

Organization of this talk

1. **Ref-schema** capturing well typed references
2. **Hybrid Modal Logic** (HML): an introduction.
3. How **ref-schema** are expressed in **HML**
4. Checking **constraint satisfiability** in presence of ref-schemas.
5. Discussion and further research directions

Normalized ref-schema in HML

- **Motivation**
  - ▶ historical reason, ▶ sake of simplicity
  - ▶ without loss of generality

- A **normalized ref-schema** $(\mathcal{E}, \mathcal{V}, Start, \theta)$ is a (marked) ref-schema based on the normalized regular expressions defined by:

  $R := B \mid R + R$, and

  $B := \Lambda \mid ((\tilde{e}, p)X)^{op} \mid B, B$ where op is either ! or $*$.

the library schema is normalized

$$
\begin{array}{rcl}
\textbf{Start} & ::= & (doc\ Doc)^*, (editor\ Editor)^* \\
Editor & ::= & (name\ Name)^!, (\overrightarrow{publish}\ Book)^* \\
Doc & ::= & (article\ Art)^!\ +\ (book\ Book)^! \\
Art & ::= & (author\ Name)^+, (title\ Name)^!, (date\ Dat)^?, (\overrightarrow{cite}\ Doc)^* \\
Book & ::= & (isbn\ Isb)^!, (\overrightarrow{cite}\ Doc)^*\ +\ (\ (author\ Name)^+, (date\ Dat)^!, \\
 & & (title\ Name)^!, (\overrightarrow{cite}\ Doc)^*, (\overrightarrow{publisedby}\ Editor)^!\ ) \\
Name & ::= & \Lambda \\
Dat & ::= & \Lambda \\
Isb & ::= & \Lambda
\end{array}
$$

the odd-even tree schema is not normalized

$$
\begin{array}{lcl}
Start & := & (e\ X)^*, \\
X & := & (\ p\ Y\ +\ o\ Z)^*, (\overrightarrow{r}\ Y\ +\ \overrightarrow{r}\ Z)^* \\
Y & := & (e\ X, e\ X)^* \\
Z & := & (e\ X, e\ X)^*,\ e\ X
\end{array}
$$

Expressing normalized ref-schema in HML

- **Result :**

Schema $\mathcal{G}$ (normalized)	$\longrightarrow$	$\tau_{\mathcal{G}}$ (HML Formula)
$\downarrow$ instance		$\downarrow$ satisfied by ($\models$)
Document (Semistructured data)	$=$	Kripke Model

$$\mathfrak{M} : \mathcal{G} \quad \text{iff} \quad \mathfrak{M}, r \models \tau_{\mathcal{G}}$$

Expressing normalized ref-schema in HML

$\tau_{\mathcal{G}}$ is the conjunction of 3 formulas :

▶ *tree* enforces that the "subframe" of the document generated by child labels is a tree.

▶ $\tau_{\mathcal{G}}^{E}$ checks that, given a state x reachable by an e child edge, the edges (child edges as well as references) outgoing from e are the ones allowed by the schema.

$$@_{root}\left(\tau_{Start} \wedge \bigwedge_{e \in E} G^{*}[e]\tau_{Type(e)}\right)$$

▶ $\tau_{\mathcal{G}}^{\overrightarrow{E}}$ checks the type of the nodes which are targets of references.

$$@_{root}\left(\bigwedge_{\overrightarrow{e} \in \overrightarrow{E}} G^{*}[\overrightarrow{e}]{\downarrow}x \left(\bigvee_{e \in child(\overrightarrow{e})} @_{root}F^{*}\langle e \rangle x\right)\right).$$

Expressing normalized ref-schema in HML

▶ For each non terminal X, the formula τ_X checks that the nodes of type X are the sources of allowed edges.

$$\tau_X = \Psi(\theta(X)) \text{ where:}$$

1. $\Psi(\Lambda) = \bigwedge_{\tilde{e} \in \mathcal{E}} \neg\langle\tilde{e}\rangle\top$
2. $\Psi(R_1 + R_2) = \Psi(R_1) \vee \Psi(R_2)$
3. If R is $((\tilde{e_1}, p_1)X_1)^{op_1}, \cdots, ((\tilde{e_k}, p_k)X_k)^{op_k}$ then

$$\Psi(R) = \bigwedge_{i=1\cdots k} \tau_i \quad \wedge \quad \bigwedge_{e \ not \ in \ R} \neg\langle e\rangle\top$$

where if op_i is ! then $\tau_i = \downarrow x \ \langle\tilde{e_i}\rangle\downarrow y \ (p_i \ \wedge \ @_x[\tilde{e_i}](p_i \to y))$

if op_i is $*$ then $\tau_i = \langle\tilde{e_i}\rangle(\top \to \bigvee_{p \in Prop_{\tilde{e_i}}} p)$

with $Prop_{\tilde{e_i}} = \{p \mid ((\tilde{e_i}, p)Y)^{op} \ in \ \theta(X)\}$

Expressing normalized ref-schema in HML: the library example

$$@_{root} \quad (\varphi_{Root} \wedge G^*[doc]\varphi_{Doc} \wedge G^*[editor]\varphi_{Editor} \wedge$$
$$G^*[Name]\varphi_{Name} \wedge G^*[article]\varphi_{Art} \wedge G^*[book]\varphi_{Book} \wedge$$
$$G^*[auteur]\varphi_{Name} \wedge G^*[title]\varphi_{Name} \wedge G^*[date]\varphi_{Dat} \wedge G^*[isbn]\varphi_{Isb})$$

$$\varphi_{Root} \quad \equiv_{def} \quad \bigwedge_{e \in \mathcal{E}-\{doc,editor\}} \neg\langle e \rangle \top$$

$$\varphi_{Editor} \quad \equiv_{def} \quad \downarrow x \ \langle Name \rangle \downarrow y \ (@_x[Name]y) \wedge \ \bigwedge_{e \in \mathcal{E}-\{Name,\overrightarrow{publish}\}} \neg\langle e \rangle \top$$

$$\varphi_{Doc} \quad \equiv_{def} \quad \downarrow x \ \langle book \rangle \downarrow y \ (@_x[book]y)$$
$$\wedge \downarrow x \ \langle article \rangle \downarrow y \ (@_x[article]y)$$
$$\wedge \ \bigwedge_{e \in \mathcal{E}-\{book,article\}} \neg\langle e \rangle \top$$

$$@_{root} \quad (G^*[\overrightarrow{cite}]\downarrow x \ (@_{root} F^* \langle doc \rangle x) \wedge$$
$$G^*[\overrightarrow{publish}]\downarrow x \ (@_{root} F^* \langle book \rangle x) \wedge$$
$$G^*[\overrightarrow{publishedby}]\downarrow x \ (@_{root} F^* \langle editor \rangle x))$$

Expressing general ref-schema in HML

$$\text{Ref-schema} \equiv \text{Normalized Ref-schema} + \text{constraints}$$

Result:

Let $\mathcal{G}$ be a ref-schema. Then there exists a normalized ref-schema $\mathcal{G}_{norm}$ and an HML constraint $\mathcal{C}_{\mathcal{G}}$ such that:

1. for each model $\mathfrak{M}$ there exists a model $\mathfrak{M}_{norm}$ such that $\mathfrak{M} : \mathcal{G}$ iff $\mathfrak{M}_{norm} : \mathcal{G}_{norm}$ and $\mathfrak{M}_{norm}, g, r \models \mathcal{C}_{\mathcal{G}}$.
2. for each model $\mathfrak{M}_{norm}$ there exists a model $\mathfrak{M}$ such that $\mathfrak{M} : \mathcal{G}$ iff $\mathfrak{M}_{norm} : \mathcal{G}_{norm}$ and $\mathfrak{M}_{norm}, g, r \models \mathcal{C}_{\mathcal{G}}$.

Expressing general ref-schema in HML: the odd-even tree example

The initial non normalized ref-schema (odd-even tree)

$$
\begin{array}{lcl}
Start & := & (e\ X)^*, \\
X & := & (\ p\ Y\ +\ o\ Z)^*, (\overrightarrow{r}\ Y\ +\ \overrightarrow{r}\ Z)^* \\
Y & := & (e\ X, e\ X)^* \\
Z & := & (e\ X, e\ X)^*,\ e\ X
\end{array}
$$

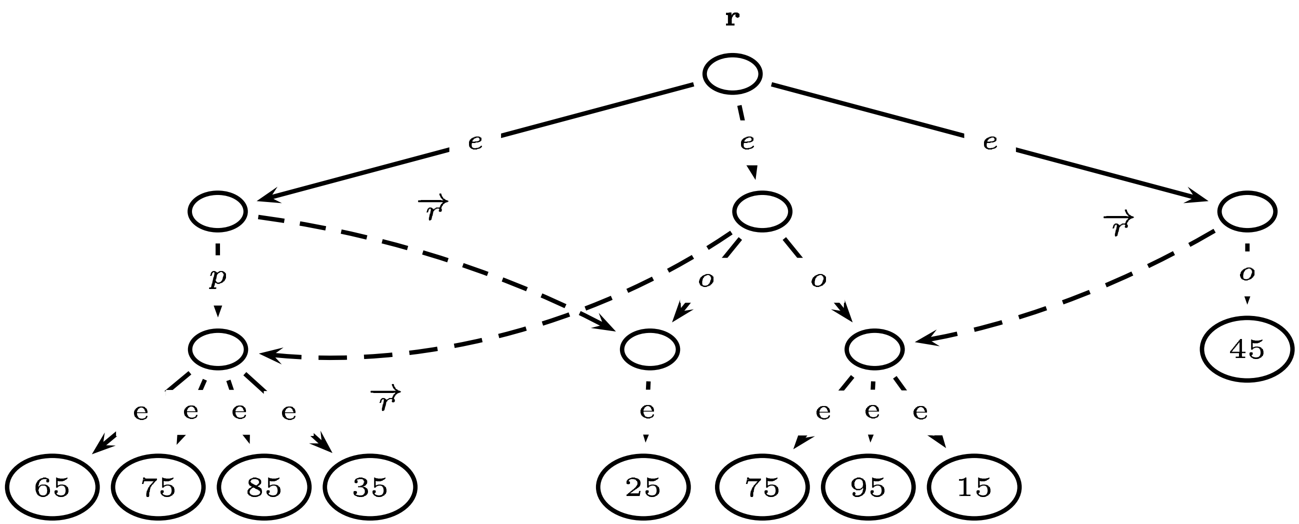
The normalized schema + constraint associated with the odd-even tree ref-schema

$$
\begin{array}{lcl}
Start & := & ((e, p_0)X)*, \\
X & := & (p\ Y)*, (o\ Z)*, (\overrightarrow{r}\ Y)*, (\overrightarrow{r}\ Z)*(\overrightarrow{c}\,X)* \\
Y & := & ((e, p_1)X)*, ((e, p_2)X)* \\
Z & := & ((e, p_3)X)*, ((e, p_4)X)*,\ (e, p_5)X!
\end{array}
$$

$$
\begin{array}{lcl}
\mathcal{C}_{\mathcal{G}} & = & G^*{\downarrow}x\ \langle e\rangle{\downarrow}y\ (p_1 \wedge\ @_y\langle\overrightarrow{c}\rangle{\downarrow}z\ (p_2\ \wedge\ @_y[\overrightarrow{c}]z \wedge\ @_x\langle e\rangle z \wedge \top)) \wedge \\
 & & G^*{\downarrow}x\ \langle e\rangle{\downarrow}y\ (p_3 \wedge\ @_y\langle\overrightarrow{c}\rangle{\downarrow}z\ (p_4\ \wedge\ @_y[\overrightarrow{c}]z \wedge\ @_x\langle e\rangle z \wedge \top))
\end{array}
$$

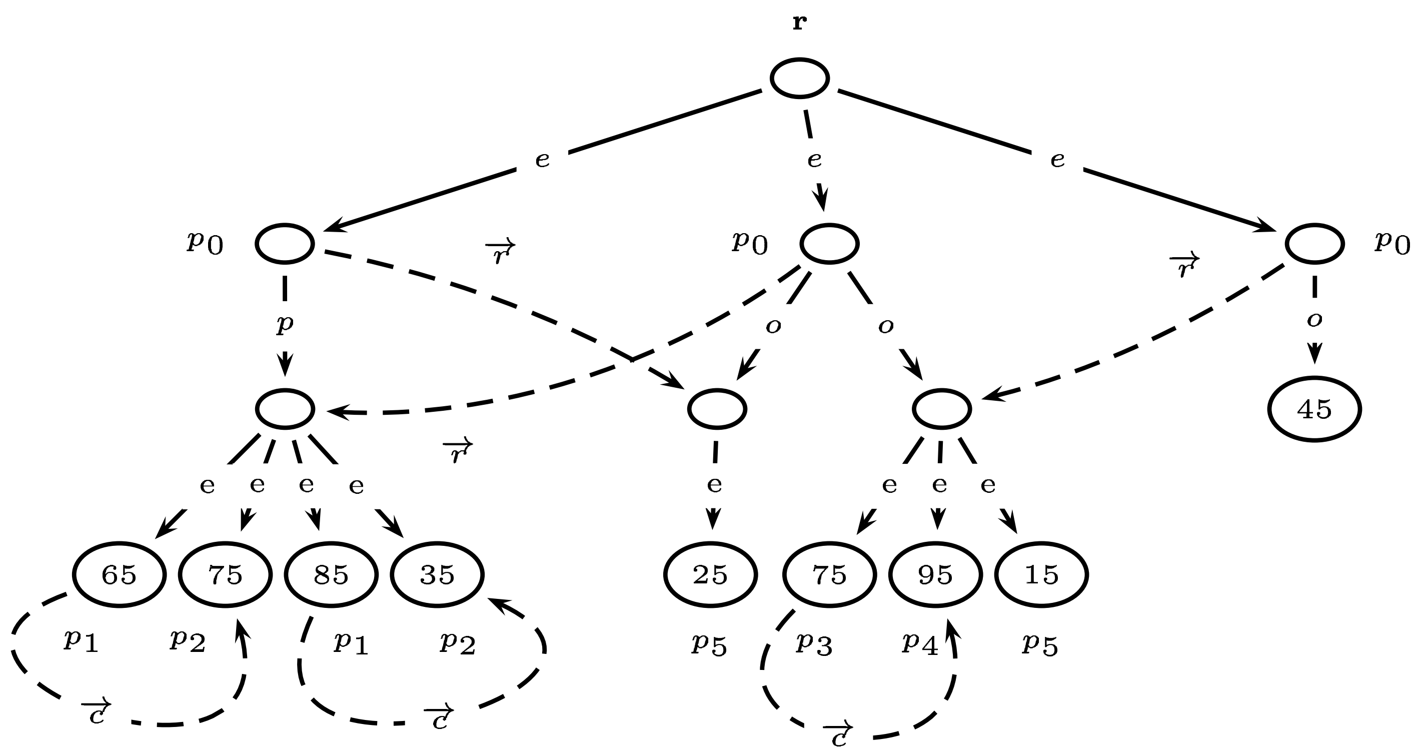
Expressing general ref-schema in HML: the odd-even tree example

The initial document conforming to the odd-even tree ref-schema



Expressing general ref-schema in HML: the odd-even tree example

The "correponding" document conforming to the normalization of the odd-even tree ref-schema



Organization of this talk

1. **Ref-schema** capturing well typed references
2. **Hybrid Modal Logic** (HML): an introduction.
3. How **ref-schema** are expressed in **HML**
4. Checking **constraint satisfiability** in presence of ref-schemas
5. Discussion and further research directions

Constraint Satisfiability in presence of ref-schema

- Statement of the problem

  Given a schema $\mathcal{G}$ and a constraint $\mathcal{C}$,
  does a document $\mathfrak{M}$ conforming to $\mathcal{G}$ exists such that
  $\mathfrak{M}$ satisfies C ?

- Formal context

  The schema "is" a HML formula $\tau_{\mathcal{G}}$
  The constraint is a HML formula $\mathcal{C}$

$\hookrightarrow$ Re-Statement of the problem

  Is $\mathcal{G} \wedge \mathcal{C}$ (finitely) satisfiable ?

- Goal:

  (terminating) proof system

Constraint Satisfiability: restriction

- normalized ref-schemas without markers (WLOG)
- HML is not decidable

  $\hookrightarrow$ non recursive schemas

  $\Longrightarrow$ the depth of models of $\mathcal{G} \wedge \mathcal{C}$ are bounded

- not sufficient to enforce the finite model property

  an example is coming next

  $\hookrightarrow$ relax "finite" satisfiability in a first step

  see concluding discussion

Non Recursive ref-schema + constraint having no finite models

Example

Schema: $Start := (e\ E)^*$ and $E := (\overrightarrow{e}\ E)^*$.

Constraint: $\psi_1 \wedge \psi_2 \wedge \psi_3 \wedge \psi_4$ where

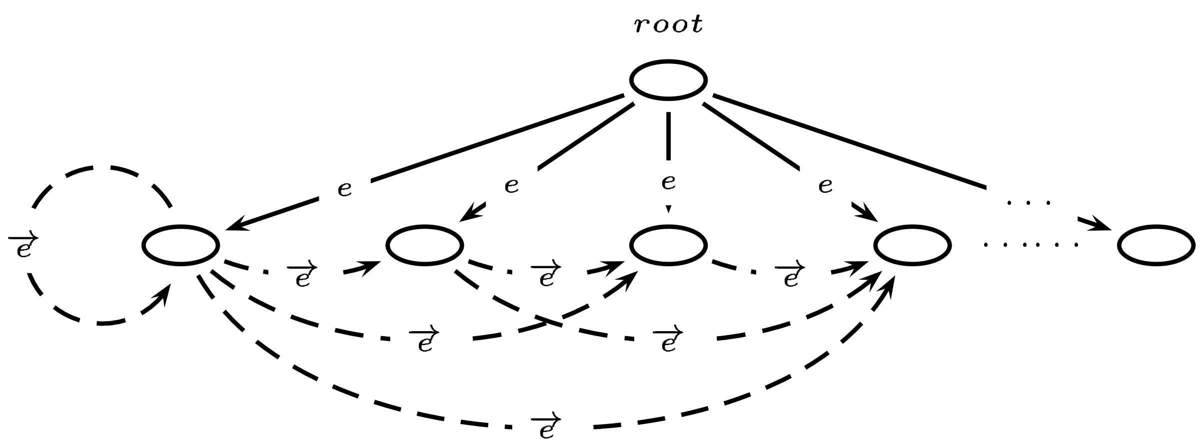
ψ_1 is $\langle e \rangle {\downarrow} y\ (\langle \overrightarrow{e} \rangle y)$

ψ_2 is $[e][\overrightarrow{e}]{\downarrow} y\ (@_{root} \langle e \rangle {\downarrow} z\ (\neg root \wedge \neg y \wedge @_y \langle \overrightarrow{e} \rangle z))$

ψ_3 is $[e]{\downarrow} x\ [\overrightarrow{e}][\overrightarrow{e}]{\downarrow} y\ @_x \langle \overrightarrow{e} \rangle y$

ψ_4 is $[e]{\downarrow} x\ [\overrightarrow{e}]{\downarrow} y\ (@_x y \vee @_y [\overrightarrow{e}] \neg x)$

The constraint is satisfied by the following infinite instance of $\mathcal{G}$



The tableau system

- geared to model building rather than refutation.
- the modalities G and F are not considered
- the schema formula $\mathcal{G}$ not used directly

  the formulas τ_X associated to types X are used
- a prefixed tableau system
  - ▶ prefixes are naming nodes (states)
  - ▶ prefixed formulas $n : \varphi$ capture that

    φ has to be satisfied at the node named by n.

  $\hookrightarrow$ encapsulation of the frame by prefixed formulas $n : \langle \tilde{e} \rangle m$
- closed rectified formula in negation normal form
- shape of rules of the tableau system are as usual
  - ▶ propositionnal rules
  - ▶ state variables and hybrid rules
  - ▶ transition rules.

Propositional rules:

$$(\alpha)\quad \frac{n:\varphi\wedge\psi,\quad\Phi}{n:\varphi,\quad n:\psi,\quad\Phi}\qquad\qquad (\beta)\quad \frac{n:\varphi\vee\psi,\quad\Phi}{n:\varphi,\quad\Phi\quad\mid\quad n:\psi,\quad\Phi}$$

State variable rule:

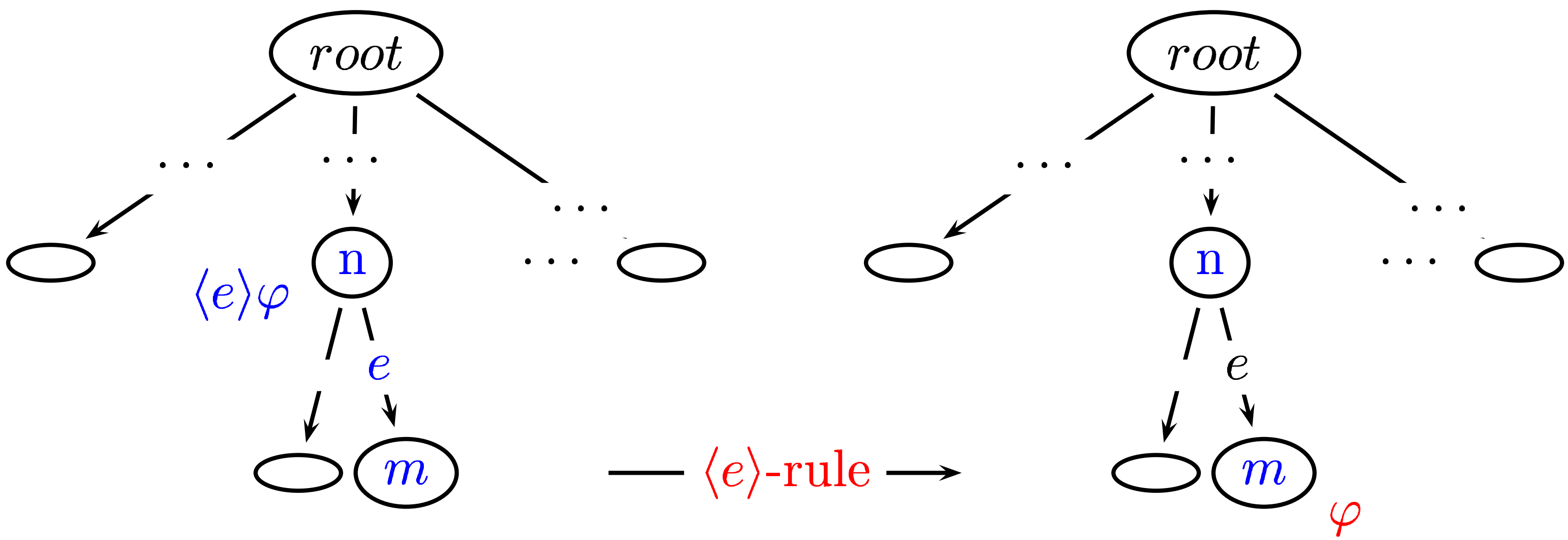
$$(Ref)\quad \frac{\Phi}{n:n,\quad\Phi}\qquad\qquad \text{if}\quad \text{n}\quad \text{occurs}\quad \text{in}\quad \Phi$$

Hybrid rules:

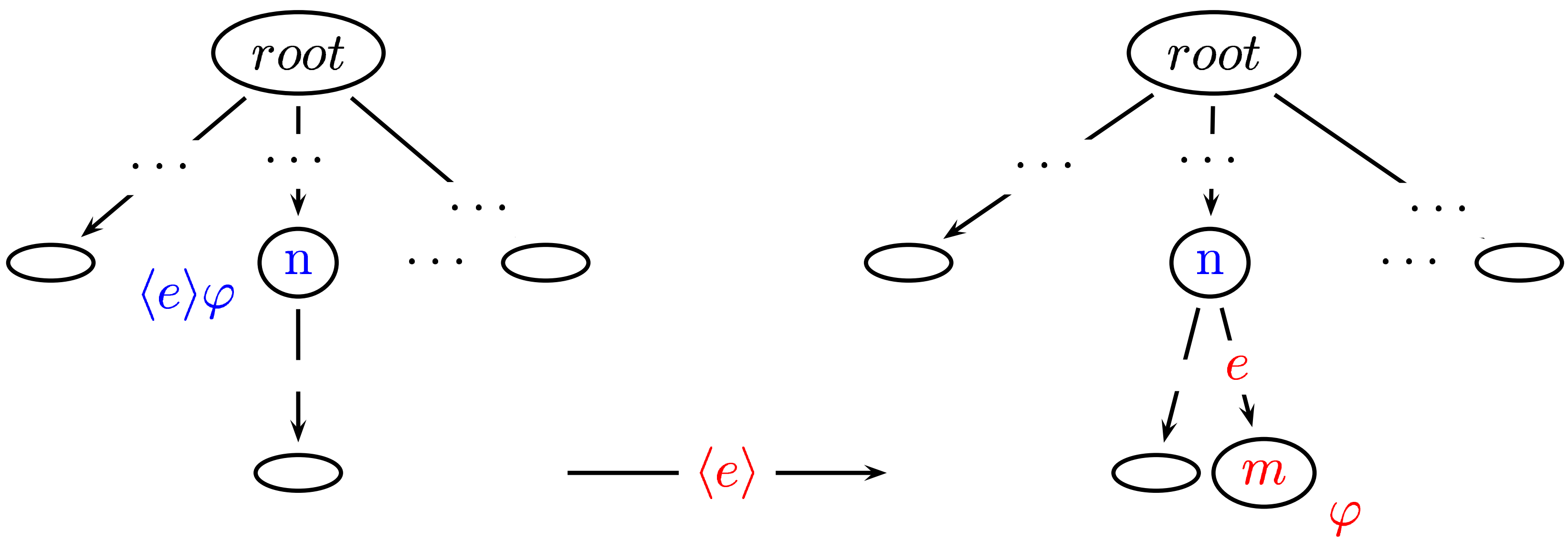
$$(@)\quad \frac{n:@_m\varphi,\quad\Phi}{m:\varphi,\quad\Phi}\qquad\qquad (\downarrow)\quad \frac{n:\downarrow x\ \varphi,\quad\Phi}{n:\varphi[x\backslash n],\quad\Phi}$$

$$\frac{n : \langle e \rangle \varphi, \quad \Phi}{m : \varphi, \; \Phi \quad \text{for } n : \langle e \rangle m \in \Phi \quad \Big| \quad n : \langle e \rangle m, \; m : \tau_{Type(e)}, \; m : \varphi, \; \Phi \quad \text{for a } \mathbf{new} \ m}$$

$Type(e)$ is the unique non terminal symbol such that the pattern (e X) occurs in the normalized schema



root
...
...
...
...
n
$\langle e \rangle\varphi$
e
m
— $\langle e \rangle$-rule →
root
...
...
...
...
n
e
m
φ

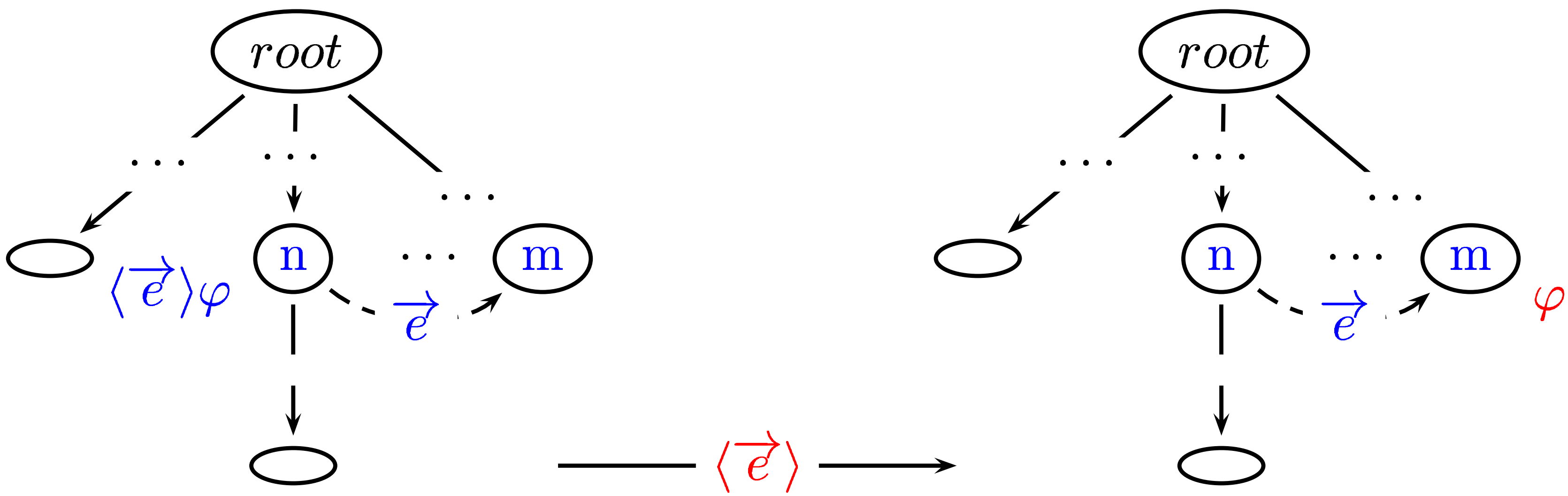

root
n
$\langle e \rangle \varphi$
root
n
e
m
φ
$\langle e \rangle$

$$\frac{n : \langle \overrightarrow{e} \rangle \varphi, \quad \Phi}{m : \varphi, \ \Phi \quad \Bigg| \quad n : \langle \overrightarrow{e} \rangle m, \ m : \varphi, \ \Phi \quad \Bigg| \quad root : \pi(\overrightarrow{e}, m), \ m : \varphi \ \ n : \langle \overrightarrow{e} \rangle m, \ \Phi}$$

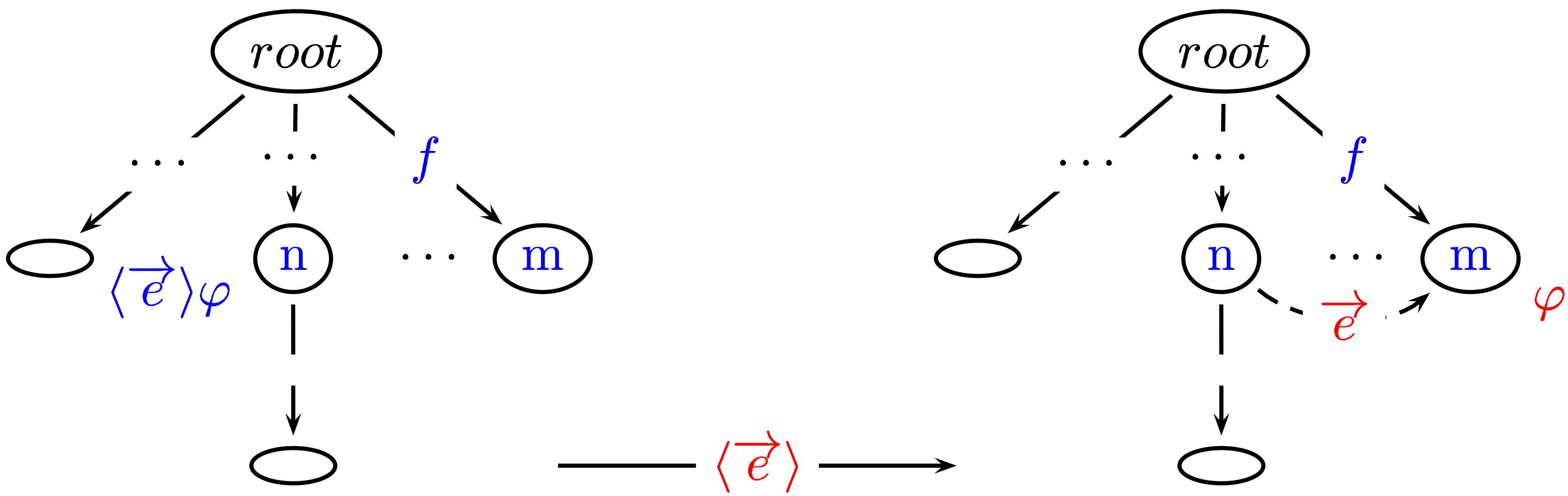
for $n : \langle \overrightarrow{e} \rangle m \in \Phi$	for $p : \langle f \rangle m \in \Phi$ $f \in Lab(Type(\overrightarrow{e}))$	for a new m and $\pi(\overrightarrow{e}, m)$ defined below

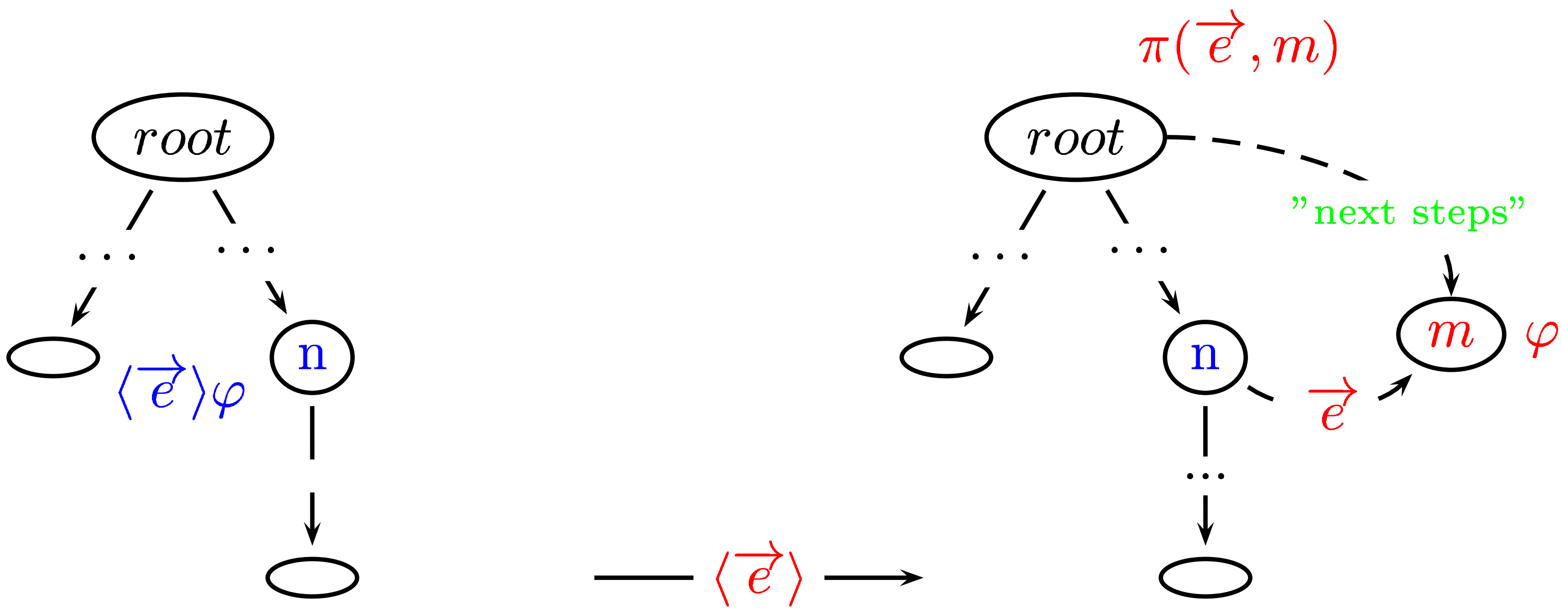
$\pi(\overrightarrow{e}, m)$ is the formula $\bigvee_{e \in Lab(Type(\overrightarrow{e})) \cap E} (\bigvee_{ph \in Path(e)} @_{root} \ \Diamond(ph) \ m)$
$Path(e)$ is teh set of paths starting from $Start$ ending with an edge labelled by e in the dependency graph associated with $\mathcal{G}$
$\Diamond(ph)$ is the modal fragment $\langle e_1 \rangle \cdots \langle e_n \rangle$ when ph is the path $e_1, \ldots, e_n$.

For instance, for our running example, $\pi(\overrightarrow{publish}, m)$ is $@_{root} \langle doc \rangle \langle book \rangle m$.



root
. . .
n
. . .
m
$\langle \overrightarrow{e} \rangle \varphi$
$\overrightarrow{e}$
$\langle \overrightarrow{e} \rangle$
root
n
m
$\overrightarrow{e}$
φ

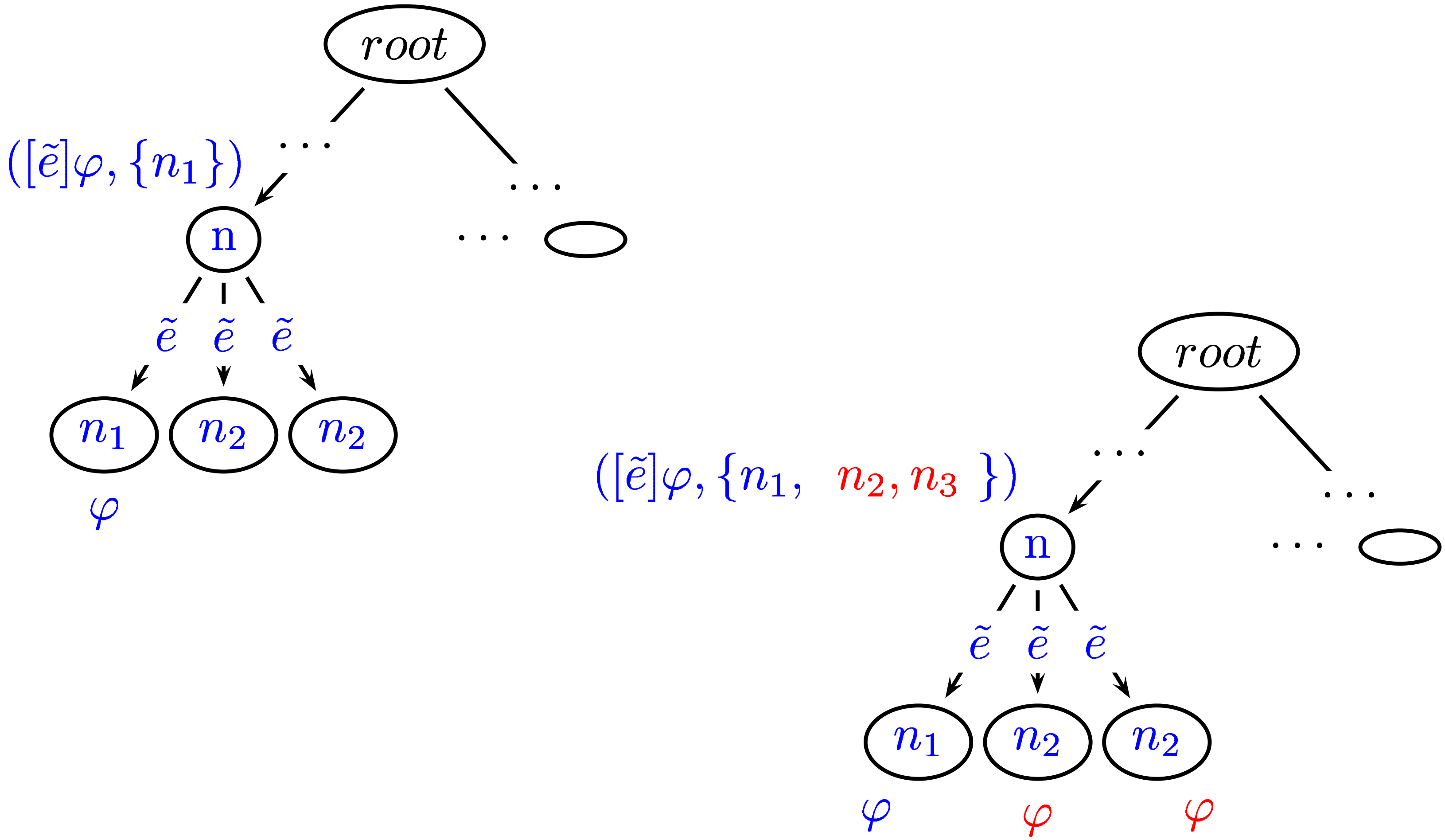

root
. . .
. . .
f
n
. . .
m
$\langle\overrightarrow{e}\rangle\varphi$
$\langle\overrightarrow{e}\rangle$
root
. . .
. . .
f
n
. . .
m
$\overrightarrow{e}$
φ


$\pi(\overrightarrow{e}, m)$
$root$
$root$
"next steps"
n
$\langle\overrightarrow{e}\rangle\varphi$
n
m
φ
$\overrightarrow{e}$
$\langle\overrightarrow{e}\rangle$

$$\frac{n : [\tilde{e}]\varphi,\ \ \Phi}{n : ([\tilde{e}]\varphi, \emptyset),\ \ \Phi}$$

$$\frac{n : ([\tilde{e}]\varphi, N),\ \ \ \Phi}{\bigcup_{m \in N'} m : \varphi,\ \ n : ([\tilde{e}]\varphi, N \cup N'),\ \ \Phi}$$

for $N' \neq \emptyset$ where $N' = \{m | n : \langle\tilde{e}\rangle m \in \Phi\} - N$


root
$([\tilde{e}]\varphi, \{n_1\})$
n
$\tilde{e}$ $\tilde{e}$ $\tilde{e}$
n_1 n_2 n_2
φ
root
$([\tilde{e}]\varphi, \{n_1, n_2, n_3\})$
n
$\tilde{e}$ $\tilde{e}$ $\tilde{e}$
n_1 n_2 n_2
φ φ φ

Systematic construction of a $\mathcal{G}$-tableau T for $\mathcal{C}$

- **Stage** 1 - Begin with $root : \tau_{Start} \wedge \mathcal{C}$

- **Stage** $i+1$ - Choose a leaf node L of the tableau as closed as possible to the root of the tableau. Choose in L a (generalized) prefixed formula $n : \varphi$ in order to apply one of the tableau rules defined above with the following priority :

  (1) propositional rules, state variable rule, hybrid rules
  (2) $\langle e \rangle$ rules, (3) $\langle \overrightarrow{e} \rangle$ rules, (4) $[e]$ rules.

  Expand L by applying the corresponding rule with respect to $n : \varphi$ in all manners.

Fairness of the systematic tableau construction

Infinite tableau (see previous example)

Open/Closed $\mathcal{G}$-Tableau

A branch $\mathcal{B}$ of T is closed iff one of its nodes contains either

some prefixed formula $n : \varphi$ and "its negation" $n : \neg\varphi$,

or some statement $n : m$ for $n \neq m$.

A branch which is not closed is open and the tableau T is open iff one of its branches is open (otherwise it is closed).

Correctness and completness of the tableau system

The proofs rest on the notion of **$\mathcal{G}$-Hintikka set**.

– set of prefixed formulas "weakly closed" under the rules of the tableau system

– link between a model $\mathfrak{M}$ of $\mathcal{G} \wedge \mathcal{C}$ and an open branch $\mathcal{B}$ of T

$\mathcal{G}$-tableau T for $\mathcal{C}$

Soundness

Let T be a $\mathcal{G}$-tableau (systematic proof tree) build for the formula $root : \tau_{Start} \wedge \mathcal{C}$.

if $\mathcal{B}$ is an open branch of the tableau T
then
$H_{\mathcal{B}}$ is a $\mathcal{G}$-Hintikka set (and satisfies $\mathcal{C}$).

$H_{\mathcal{B}}$ is the set of prefixed formula "gathered" all along the branch $\mathcal{B}$

$\mathcal{G}$-tableau T for $\mathcal{C}$

Completness Given a schema $\mathcal{G}$ and a constraint $\mathcal{C}$.

if there exists an instance $\mathfrak{M}$ of $\mathcal{G}$ satisfying the constraint $\mathcal{C}$
then
the $\mathcal{G}$-tableau T for $\mathcal{C}$ has at least one open branch $\mathcal{B}$.

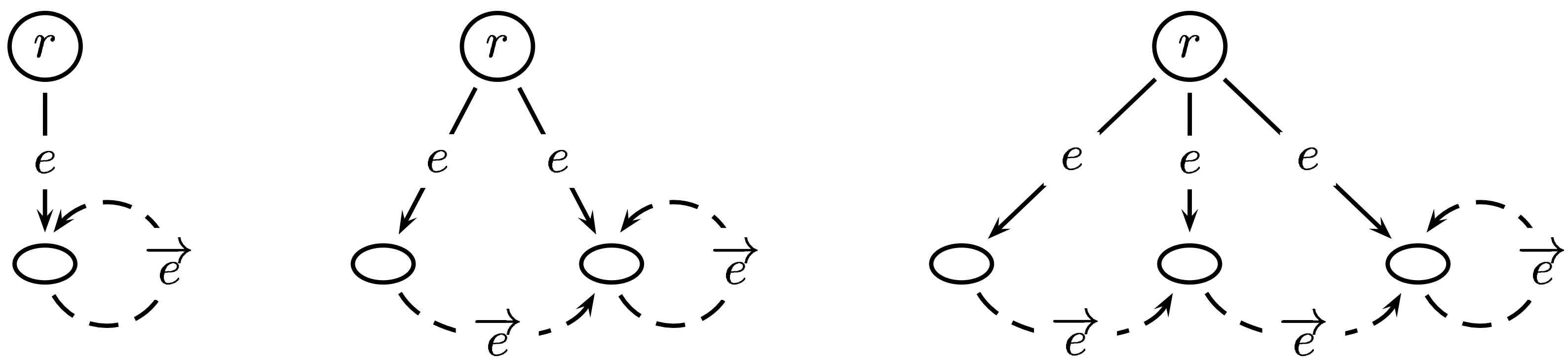
Remark

The $\mathcal{G}$-tableau T for $\mathcal{C}$ "constructs" some of the instances of $\mathcal{G}$ satisfying the constraint $\mathcal{C}$, not all of them.

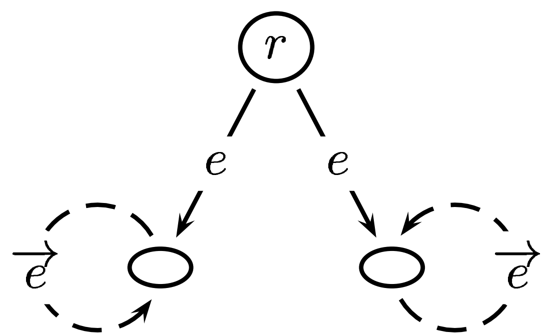
and of course T may not build $\mathfrak{M}$ at all.

$\mathcal{G}$-tableau T for $\mathcal{C}$ — Completeness

Example $\quad Start := (eE)^+ \quad E := (\overrightarrow{e}E)^+$



Instances generated by the tableau system



Instance not generated by the tableau system

Future Work

Working on the tableau system

- Finite satisfiability

  syntaxic restriction (interleaving of $\downarrow x$ and $@_x$ operators)

  bissimulation

- Implementation

Extending schema definition versus HML

- unordered elements and ordered elements
- using proposition over internal nodes (Colorful XML)

Working on optimisation

- investigating HML as a query language
- expressivity / complexity / automata ?
- optimization