The Typechecking Problem for XML Transformations: Methods and Formal Models

Wim Martens

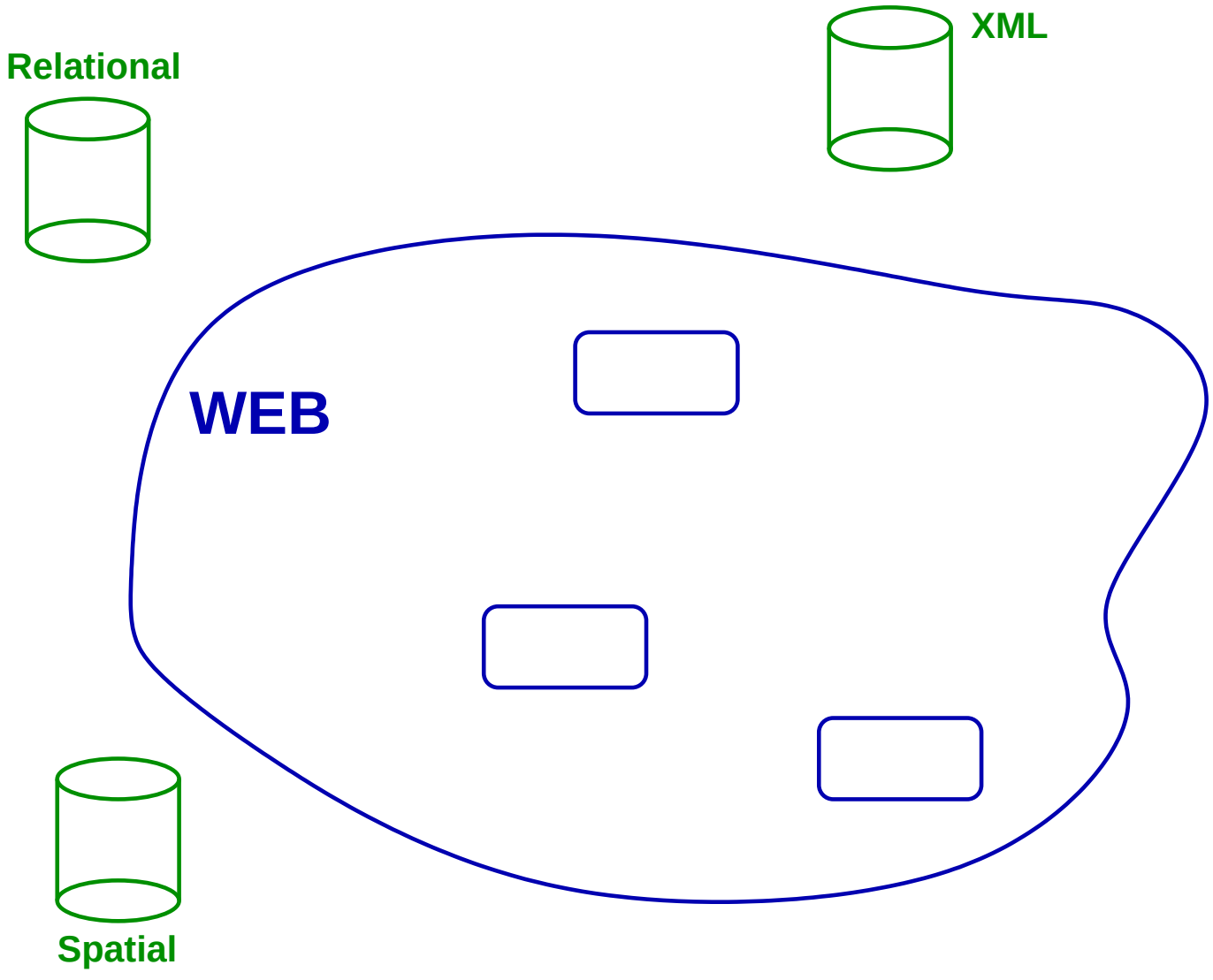
Hasselt University

Belgium

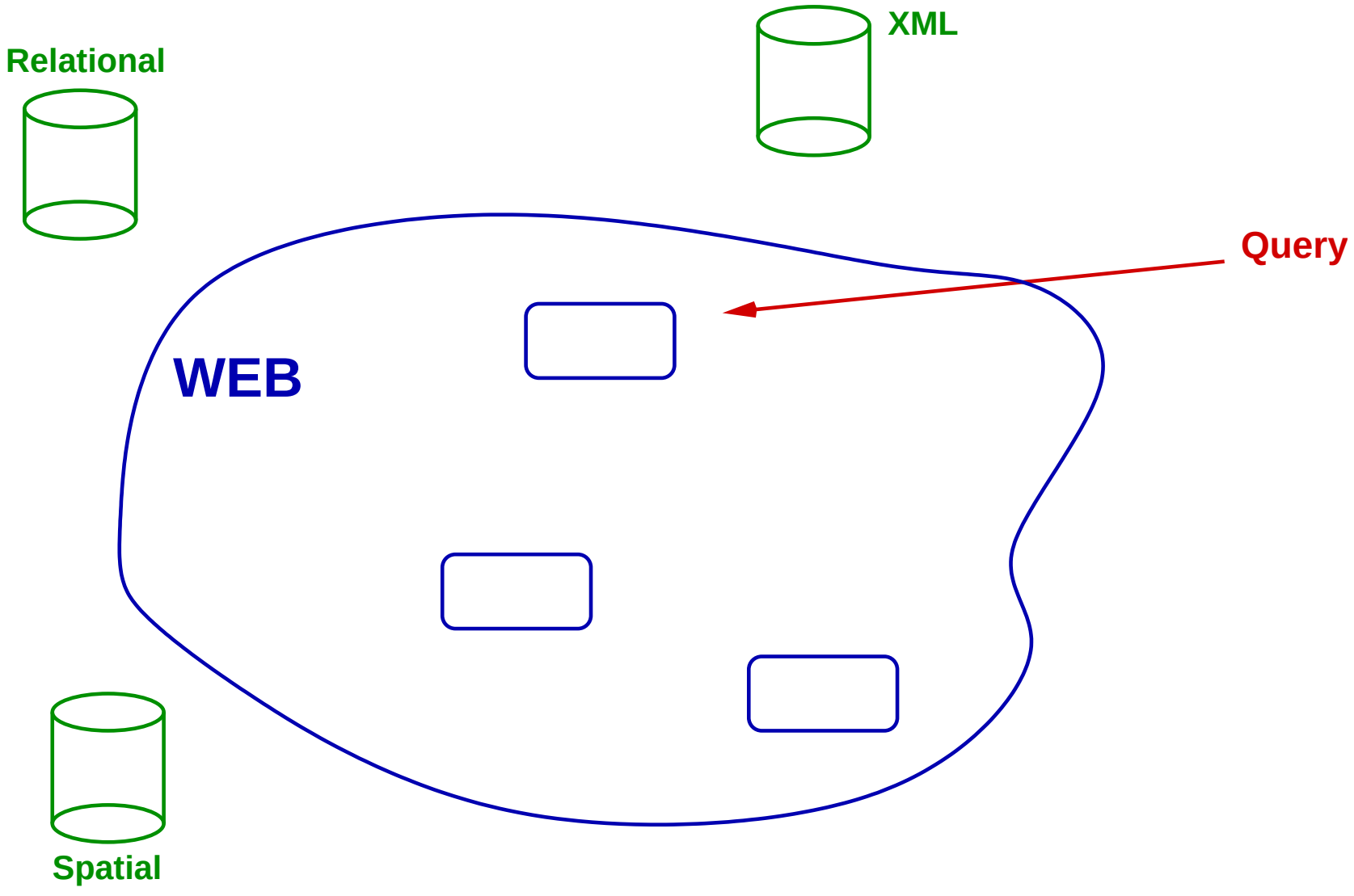
Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

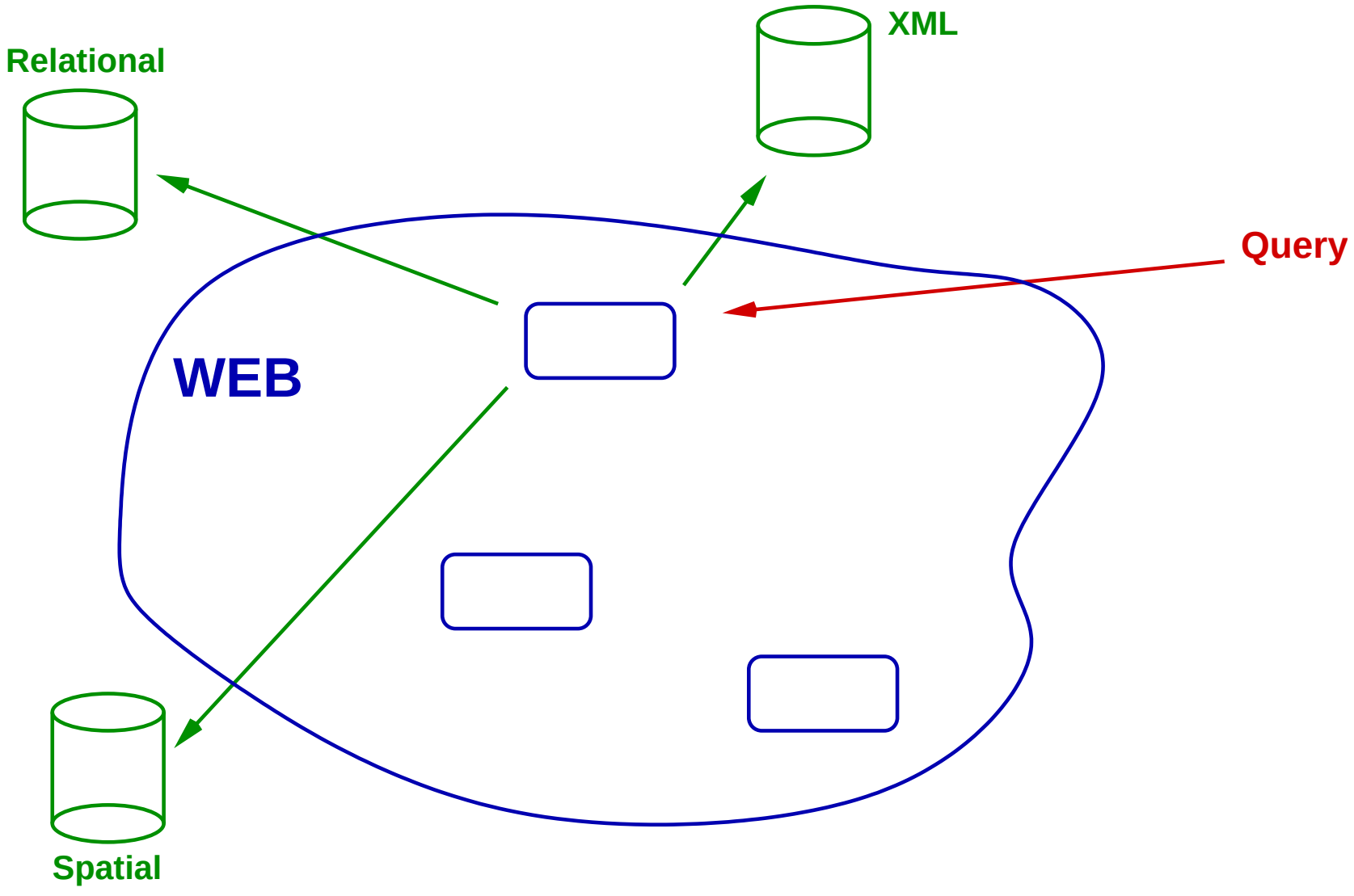
Data Integration on the Web



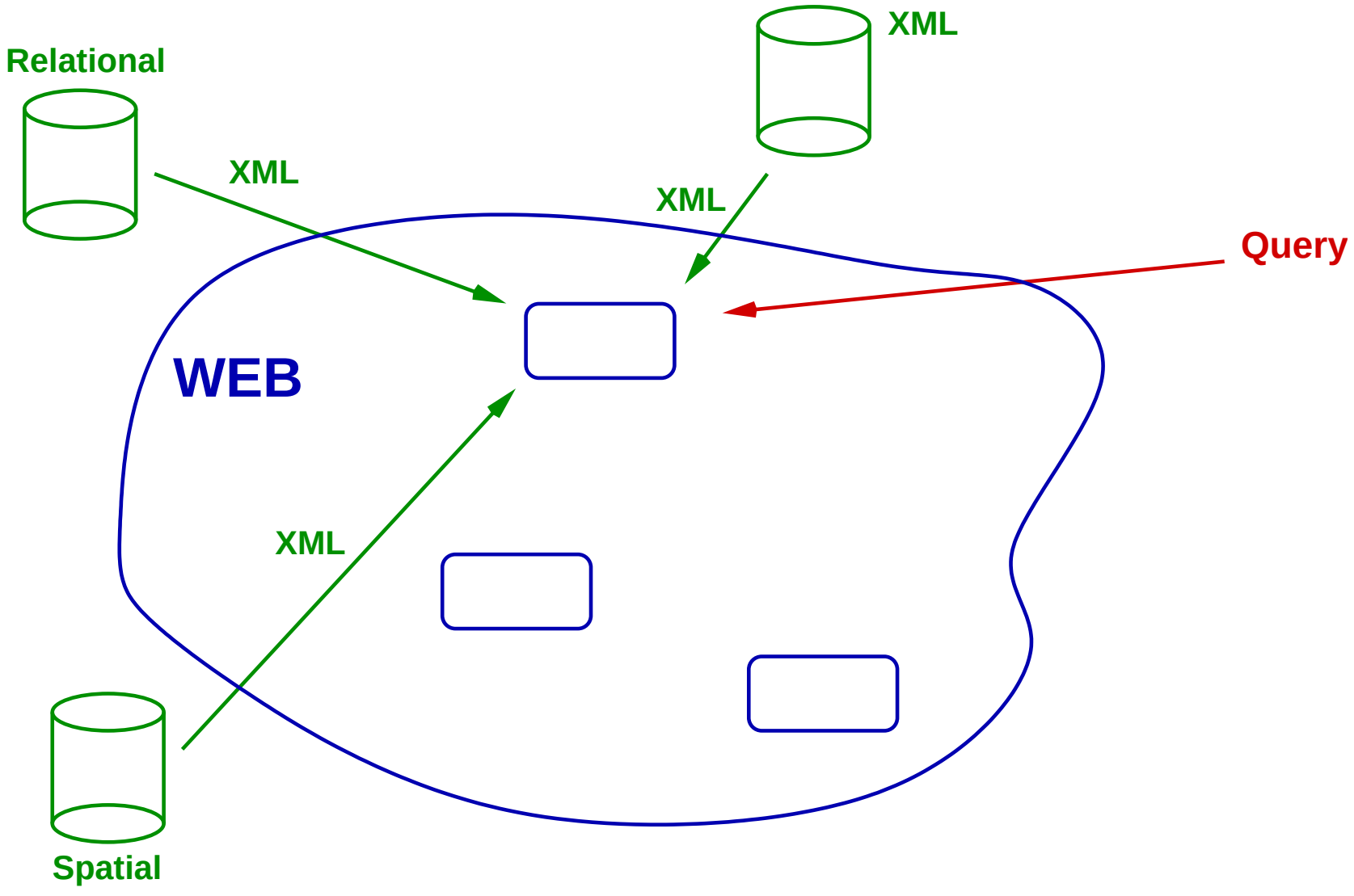
Data Integration on the Web



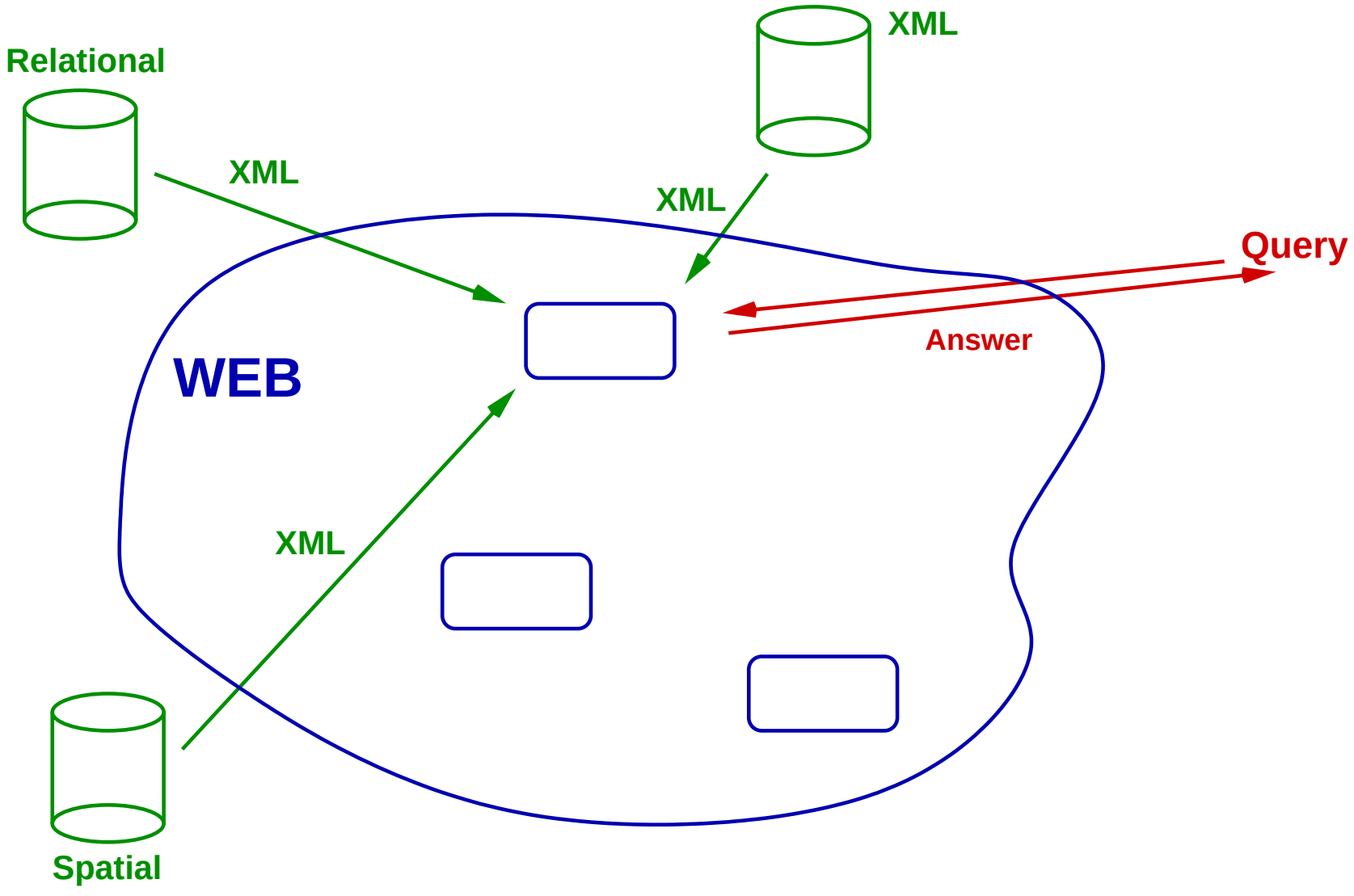
Data Integration on the Web



Data Integration on the Web



Data Integration on the Web



What is Typechecking?

Typechecking:

$$\text{is } \forall t \in S_{\text{in}} : T(t) \in S_{\text{out}}?$$

Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

Which varieties have been investigated?

Complete vs. Incomplete:

Today, typecheckers are sound but incomplete.
Excellent tutorial on incomplete typecheckers:
[Møller, Schwartzbach, ICDT 05]

Data Values vs. No Data Values:

Complete typechecking quickly turns undecidable when data values are incorporated [Alon et al. 2001]

We focus on complete typechecking where data values are not taken into account

Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

What do I want to discuss?

Formal Models

. . . for modelling

- XML schema languages and
- XML transformations

Methods

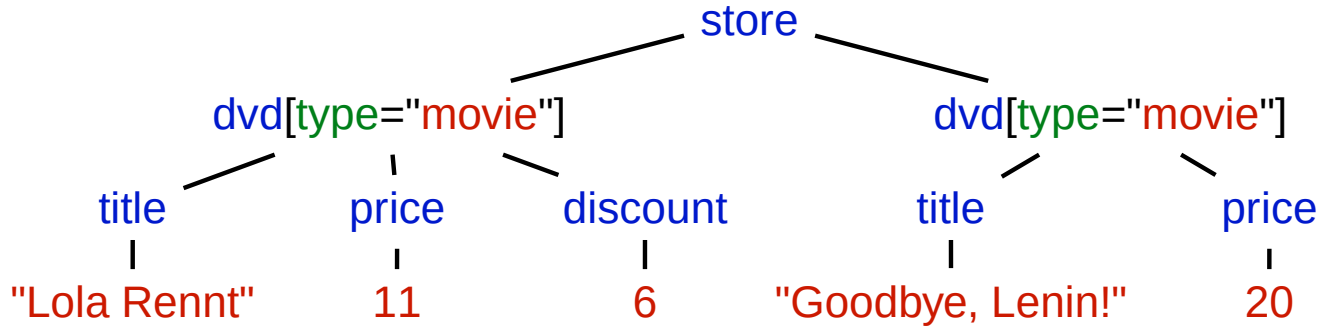
. . . for obtaining upper and lower bounds on the complexity of the typechecking problem

Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

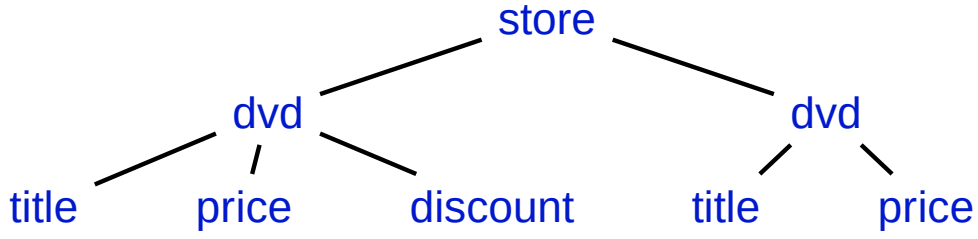
XML Documents are Trees

```
<store>
    <dvd type="movie">
        <title> "Lola Rennt" </title>
        <price> 11 </price>
        <discount> 6 </discount>
    </dvd>
    <dvd type="movie">
        <title> "Goodbye, Lenin!" </title>
        <price> 20 </price>
    </dvd>
</store>
```



XML Documents are Trees

```
<store>
    <dvd type="movie">
        <title> "Lola Rennt" </title>
        <price> 11 </price>
        <discount> 6 </discount>
    </dvd>
    <dvd type="movie">
        <title> "Goodbye, Lenin!" </title>
        <price> 20 </price>
    </dvd>
</store>
```



Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

XML Schema Languages: DTDs

- DTDs (Document Type Definitions):

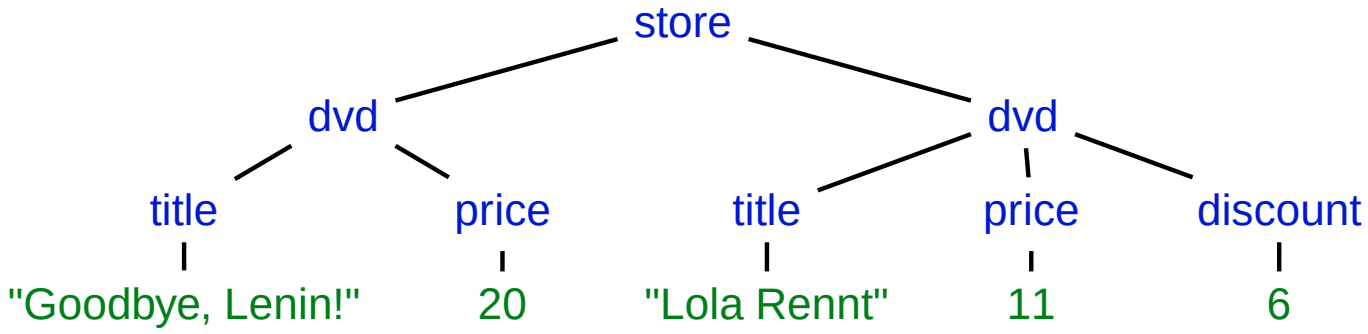
$$\begin{aligned} \text{store} &\longrightarrow \text{dvd dvd}^* \\ \text{dvd} &\longrightarrow \text{title price discount?} \end{aligned}$$

XML Schema Languages: DTDs

- DTDs (Document Type Definitions):

store → dvd dvd*

dvd → title price discount?



Regular Tree Languages

Extended DTDs (EDTDs) [Papak., Vianu 2000]:
$\equiv$ tree automata on unranked trees

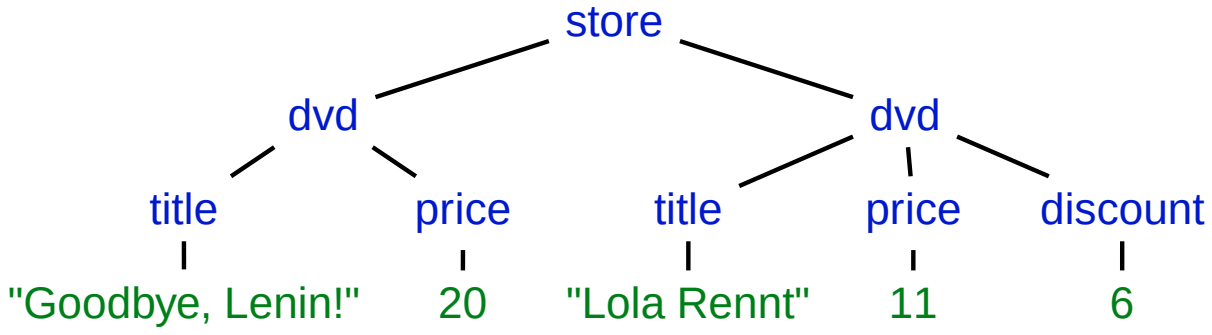
$$\begin{aligned} \text{store} &\rightarrow (\text{dvd}^1)^* \ \text{dvd}^2 \ (\text{dvd}^2)^* \\ \text{dvd}^1 &\rightarrow \text{title price} \\ \text{dvd}^2 &\rightarrow \text{title price discount} \end{aligned}$$

Regular Tree Languages

Extended DTDs (EDTDs) [Papak., Vianu 2000]:

$\equiv$ tree automata on unranked trees

$$\begin{aligned} \text{store} &\rightarrow (\text{dvd}^1)^* \ \text{dvd}^2 \ (\text{dvd}^2)^* \\ \text{dvd}^1 &\rightarrow \text{title price} \\ \text{dvd}^2 &\rightarrow \text{title price discount} \end{aligned}$$

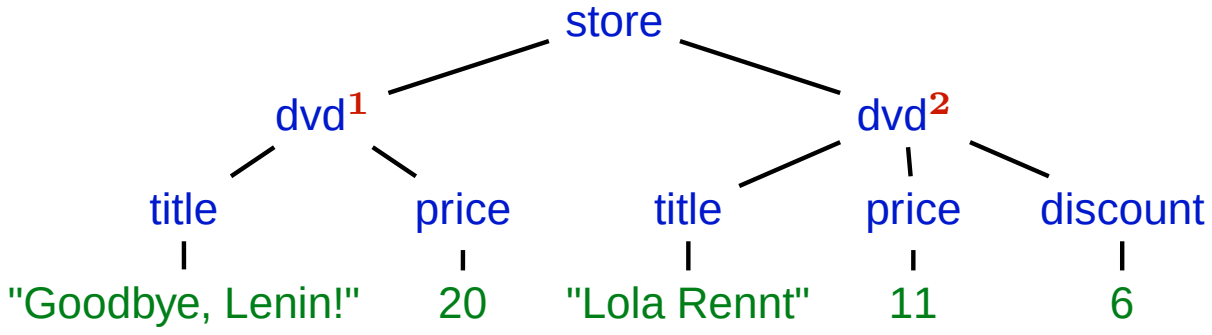


Regular Tree Languages

Extended DTDs (EDTDs) [Papak., Vianu 2000]:

$\equiv$ tree automata on unranked trees

$$\begin{aligned} \text{store} &\rightarrow (\text{dvd}^1)^* \ \text{dvd}^2 \ (\text{dvd}^2)^* \\ \text{dvd}^1 &\rightarrow \text{title price} \\ \text{dvd}^2 &\rightarrow \text{title price discount} \end{aligned}$$



XML Schema

Single-type EDTDs (stEDTDs) [Murata et al., 2001]:

. . . capture the expressive power of XML Schema

Different types for a label in the same rhs are not allowed!

Example:

$\text{store} \longrightarrow (\text{dvd}^{\mathbf{1}})^* \ \text{dvd}^{\mathbf{2}} \ (\text{dvd}^2)^*$	not allowed
$\text{dvd}^1 \longrightarrow \text{title}^2 \ \text{price}^3$	allowed

XML Schema

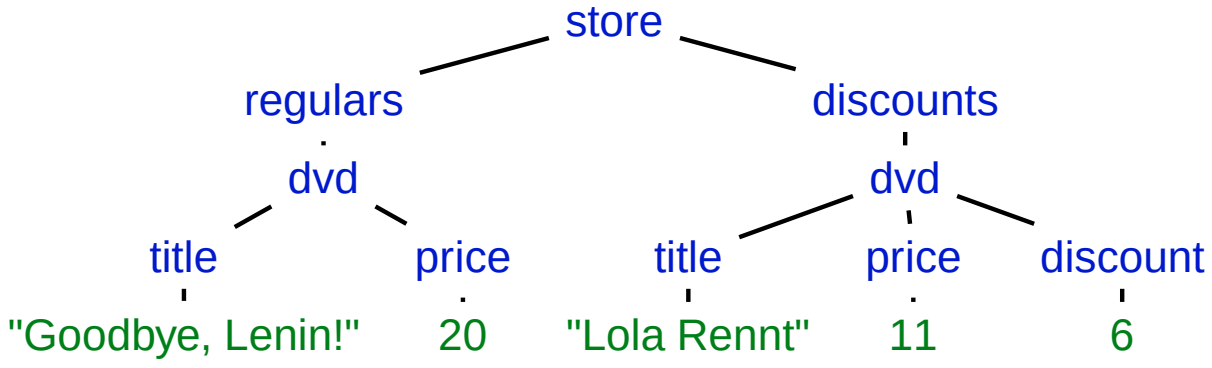
Single-type EDTDs (stEDTDs) [Murata et al., 2001]:

$$
\begin{array}{lcl}
\text{store} & \longrightarrow & \text{regulars}^* \ \text{discounts} \ \text{discounts}^* \\
\text{regulars} & \longrightarrow & \text{dvd}^1 \\
\text{discounts} & \longrightarrow & \text{dvd}^2 \\
\text{dvd}^1 & \longrightarrow & \text{title price} \\
\text{dvd}^2 & \longrightarrow & \text{title price discount}
\end{array}
$$

XML Schema

Single-type EDTDs (stEDTDs) [Murata et al., 2001]:

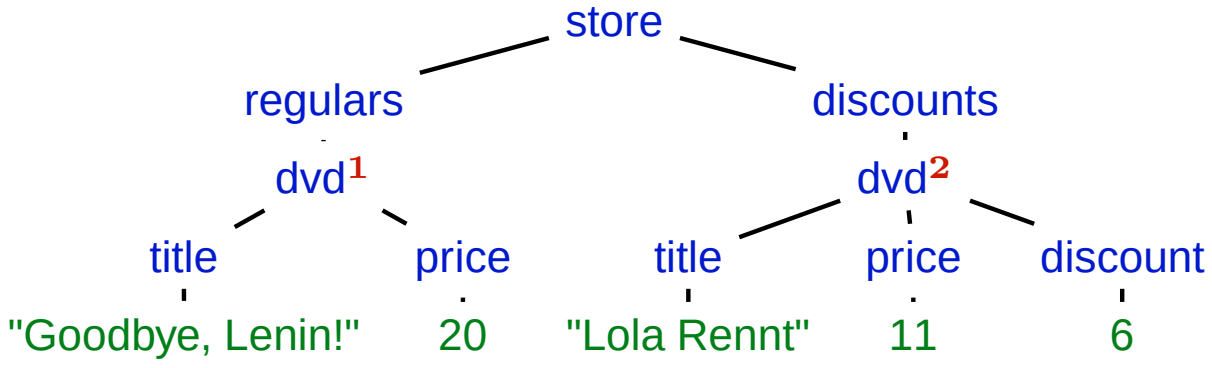
$$
\begin{array}{lcl}
\text{store} & \longrightarrow & \text{regulars}^* \ \text{discounts} \ \text{discounts}^* \\
\text{regulars} & \longrightarrow & \text{dvd}^1 \\
\text{discounts} & \longrightarrow & \text{dvd}^2 \\
\text{dvd}^1 & \longrightarrow & \text{title price} \\
\text{dvd}^2 & \longrightarrow & \text{title price discount}
\end{array}
$$



XML Schema

Single-type EDTDs (stEDTDs) [Murata et al., 2001]:

$$
\begin{array}{lcl}
\text{store} & \longrightarrow & \text{regulars}^* \ \text{discounts} \ \text{discounts}^* \\
\text{regulars} & \longrightarrow & \text{dvd}^1 \\
\text{discounts} & \longrightarrow & \text{dvd}^2 \\
\text{dvd}^1 & \longrightarrow & \text{title price} \\
\text{dvd}^2 & \longrightarrow & \text{title price discount}
\end{array}
$$

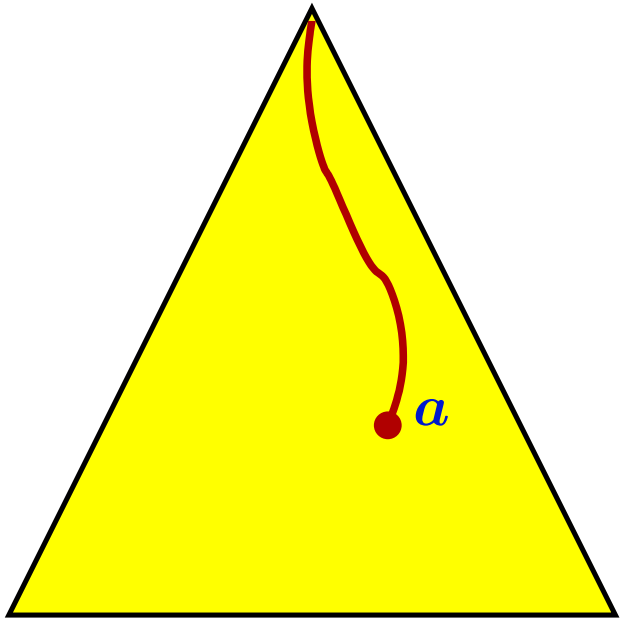


Note: DTD $\subsetneq$ single-type EDTD $\subsetneq$ EDTD

stEDTDs: Alternative Characterizations

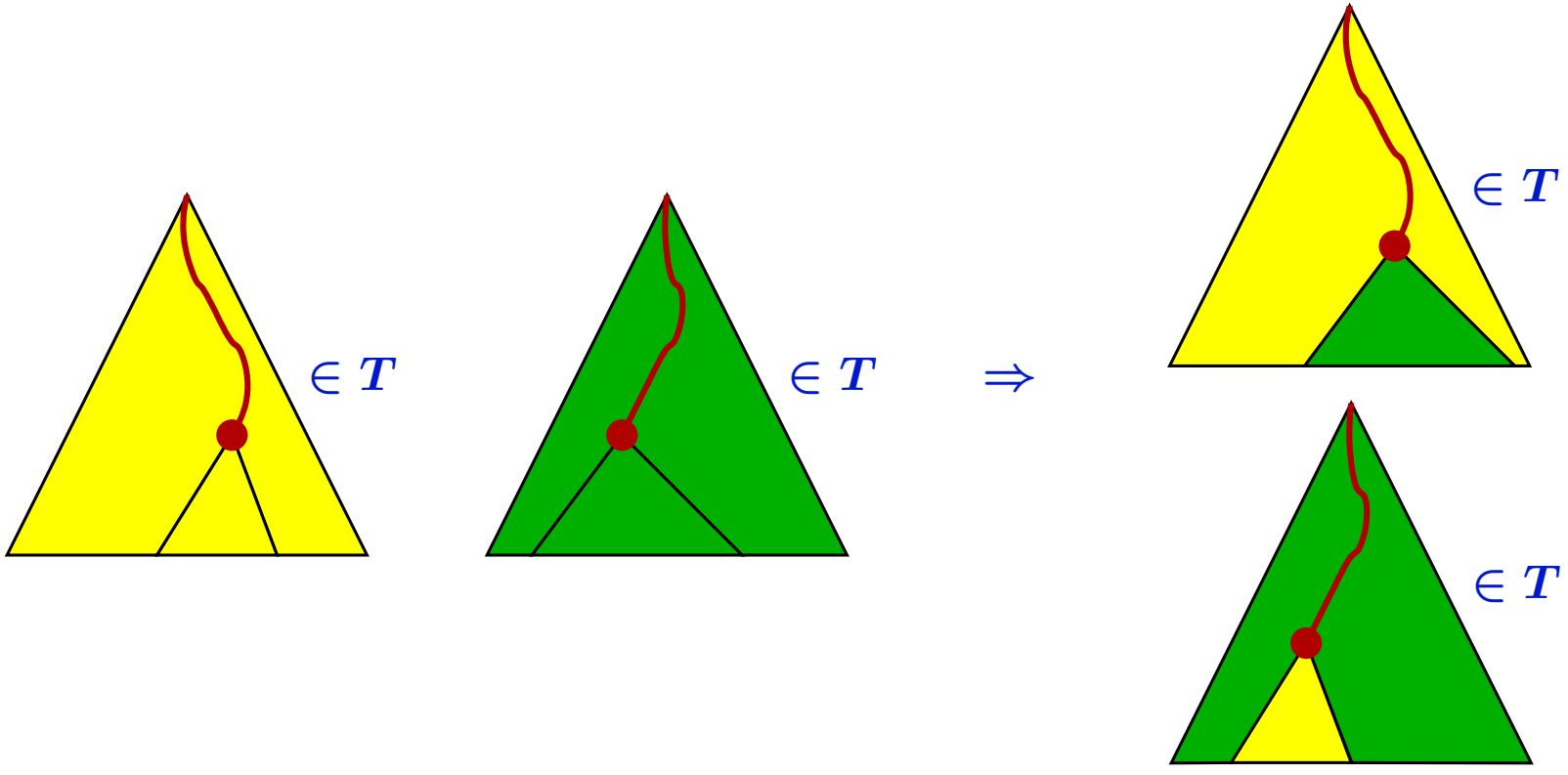
stEDTDs: Alternative Characterizations

The Ancestor-string:

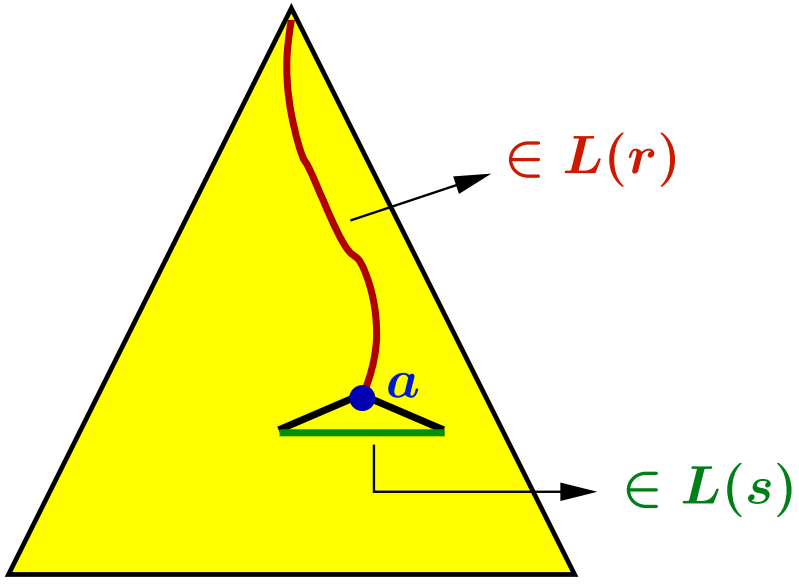


Ancestor-Guarded Subtree Exchange

T a regular tree language



Ancestor-Guarded DTDs



Ancestor-guarded DTD consists of triples $(r, a) \longrightarrow s$

The Equivalence

Let T be a regular tree language

The following are equivalent:

- T is definable by a single-type EDTD
- T is closed under ancestor-guarded subtree exchange
- T is definable by an ancestor-guarded DTD

The Equivalence

Let T be a regular tree language

The following are equivalent:

- T is definable by a single-type EDTD
- T is closed under ancestor-guarded subtree exchange
- T is definable by an ancestor-guarded DTD

DTDs can be characterized analogously

Closure Properties

Are these schema languages closed under union, difference, complement or intersection?

	union	difference	complement	intersection
DTD	no	no	no	yes
Single-type	no	no	no	yes
EDTD	yes	yes	yes	yes

Non-closure proofs are easy by the presented subtree-exchange properties

Closure proofs are by direct construction

Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

Simple Tree Transducers

[M., Neven 2003]

"A finite automaton running on all paths from root to leaf", while producing output

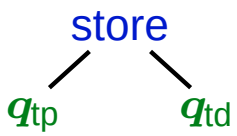
At each point in the computation it either

- stops; or
- continues in all the children of the current node

It can continue computation in multiple states (i.e. copying).

GOAL: Model simple restructuring transformations

Simple Tree Transducers: Example

$(q_o, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

$(q_{td}, \text{dvd}) \rightarrow$ 

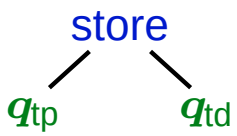
$(q_{tp}, \text{t}) \rightarrow \text{t}$

$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

$(q_{td}, \text{d}) \rightarrow \text{d}$

Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

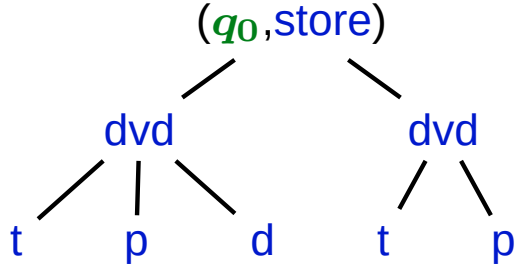
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

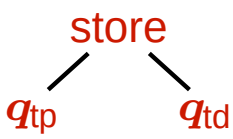
$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

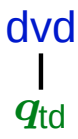
$(q_{td}, \text{d}) \rightarrow \text{d}$



Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

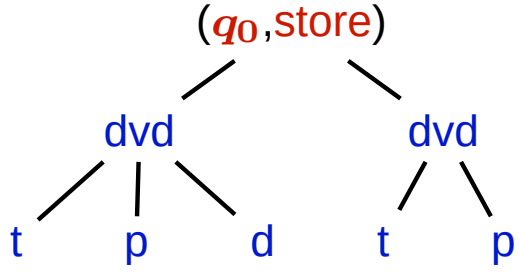
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

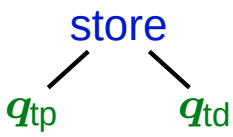
$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

$(q_{td}, \text{d}) \rightarrow \text{d}$



Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

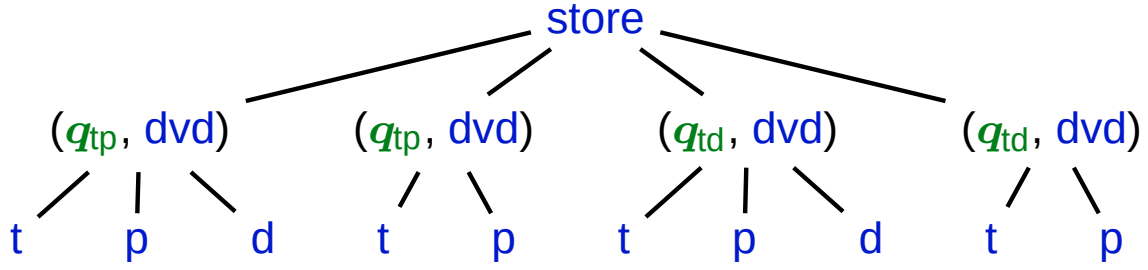
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

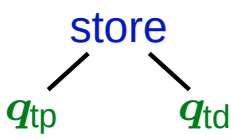
$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

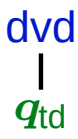
$(q_{td}, \text{d}) \rightarrow \text{d}$



Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

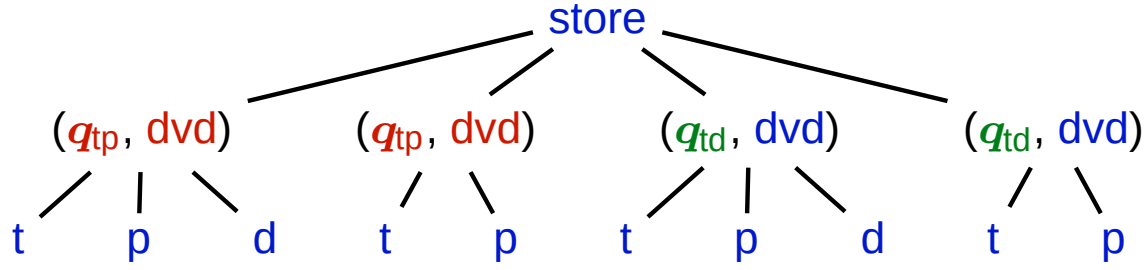
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

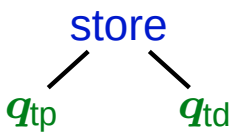
$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

$(q_{td}, \text{d}) \rightarrow \text{d}$



Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

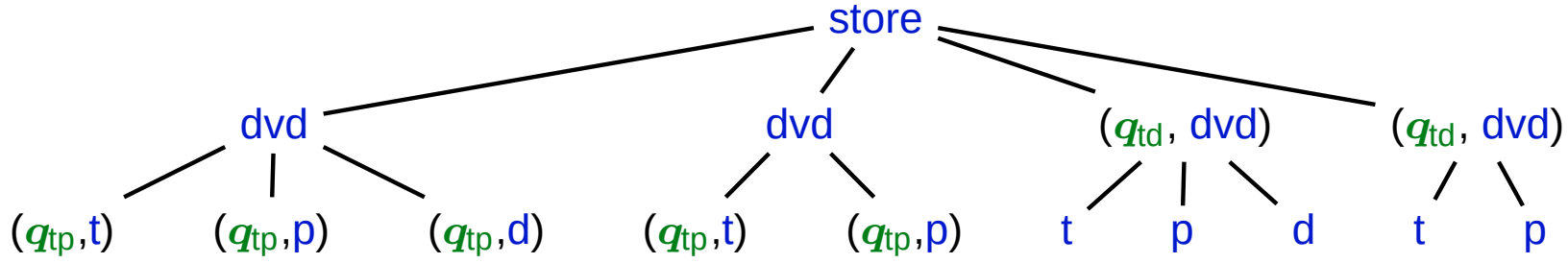
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

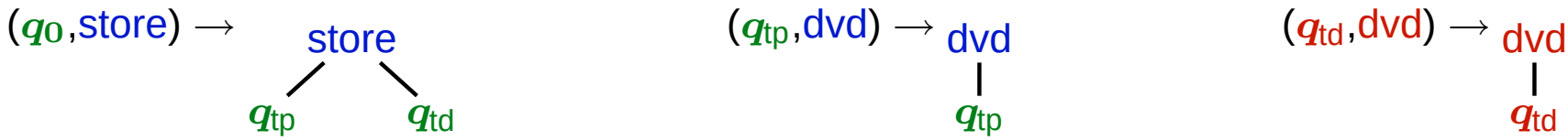
$(q_{tp}, \text{p}) \rightarrow \text{p}$

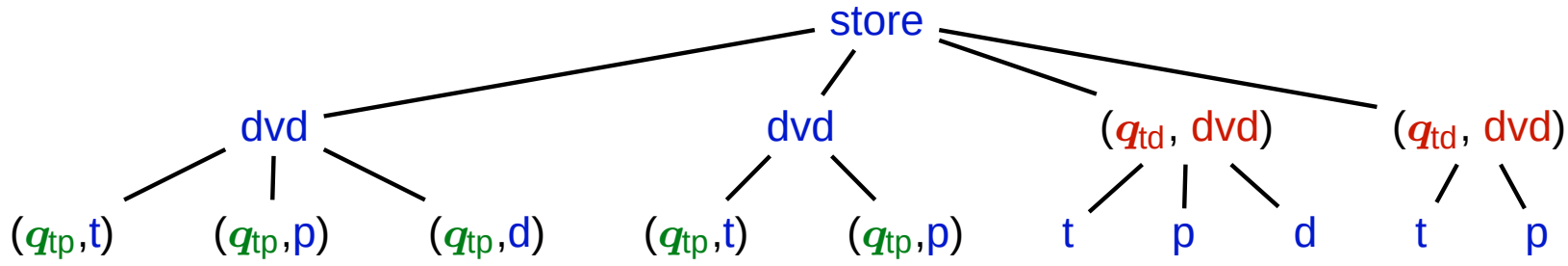
$(q_{td}, \text{t}) \rightarrow \text{t}$

$(q_{td}, \text{d}) \rightarrow \text{d}$

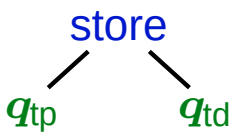


Simple Tree Transducers: Example





Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

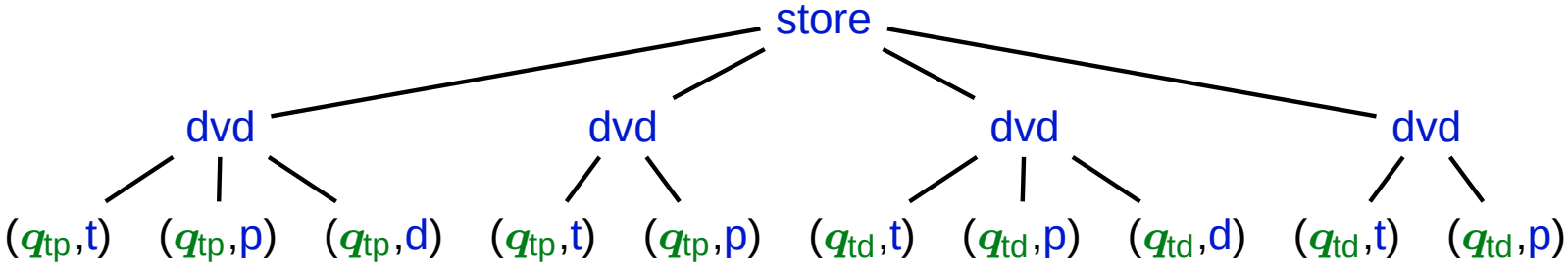
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

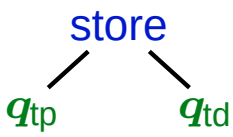
$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

$(q_{td}, \text{d}) \rightarrow \text{d}$



Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

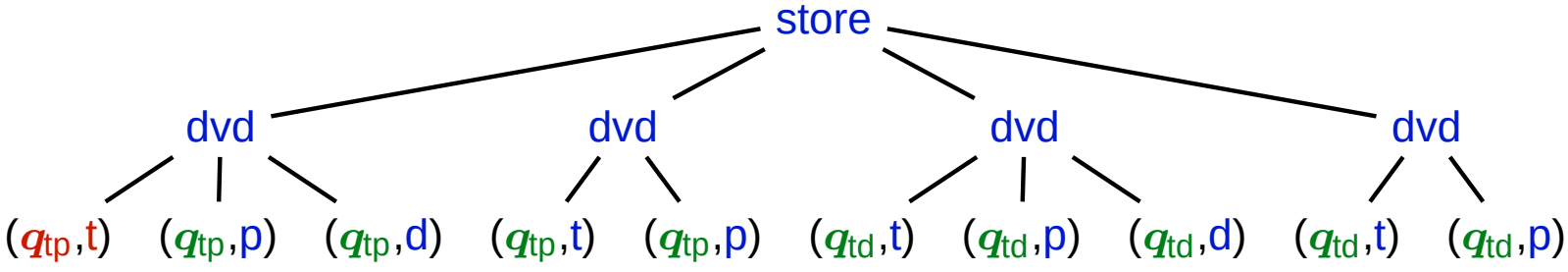
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

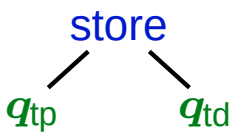
$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

$(q_{td}, \text{d}) \rightarrow \text{d}$



Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

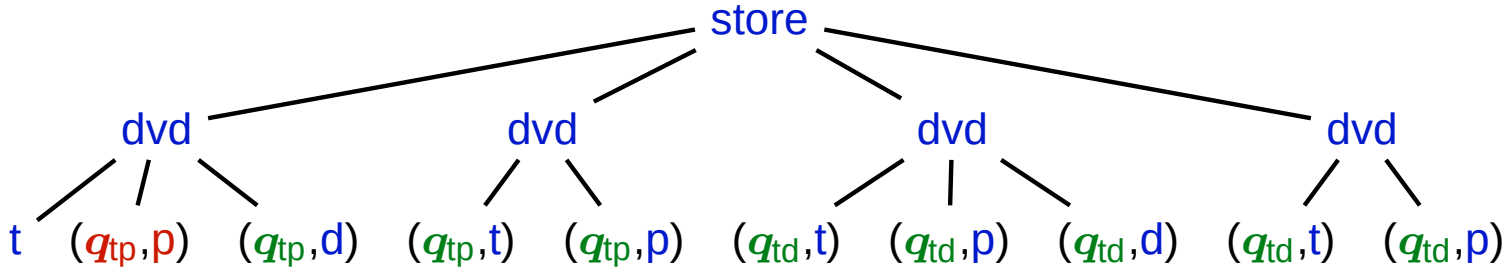
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, t) \rightarrow t$

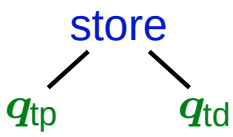
$(q_{tp}, p) \rightarrow p$

$(q_{td}, t) \rightarrow t$

$(q_{td}, d) \rightarrow d$



Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

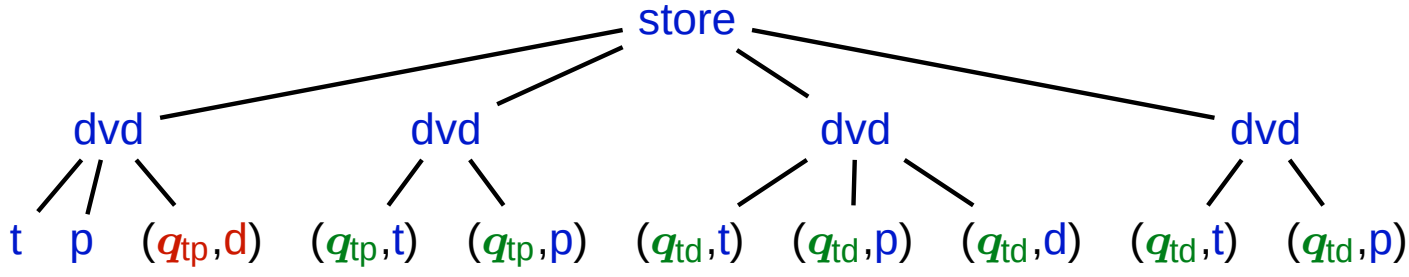
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

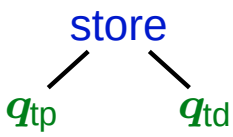
$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

$(q_{td}, \text{d}) \rightarrow \text{d}$



Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

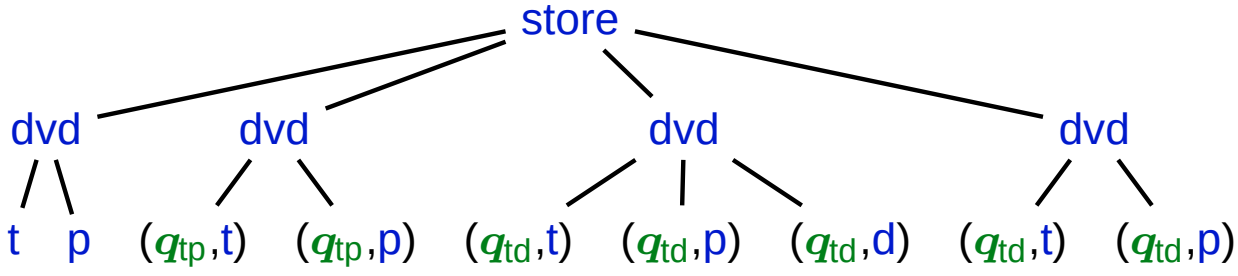
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

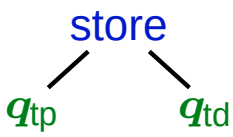
$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

$(q_{td}, \text{d}) \rightarrow \text{d}$



Simple Tree Transducers: Example

$(q_0, \text{store}) \rightarrow$ 

$(q_{tp}, \text{dvd}) \rightarrow$ 

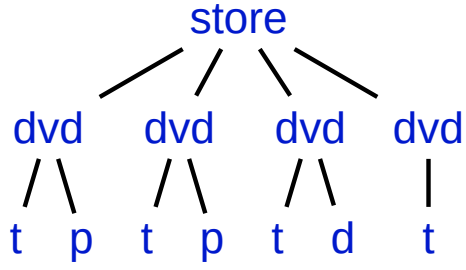
$(q_{td}, \text{dvd}) \rightarrow$ 

$(q_{tp}, \text{t}) \rightarrow \text{t}$

$(q_{tp}, \text{p}) \rightarrow \text{p}$

$(q_{td}, \text{t}) \rightarrow \text{t}$

$(q_{td}, \text{d}) \rightarrow \text{d}$



Macro Tree Transducers

Macro tree transducers [Engelfriet, Vogler 1985]

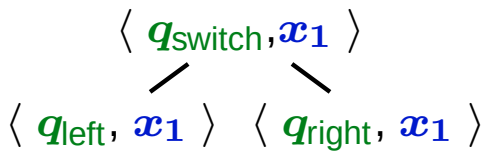
. . . are much more powerful than Simple Tree Transducers

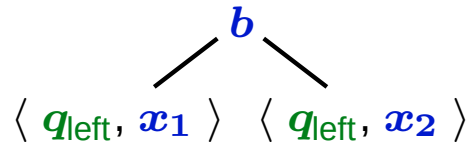
. . . normally work on ranked instead of unranked trees

. . . can be seen as a term rewriting system with input variables and output variables

. . . are also a useful tool to obtain complexity upper bounds

Macro Tree Transducers: Example

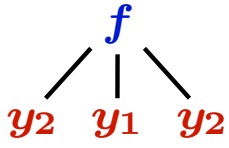
$\langle\, q_{\text{copy}}, a(x_1) \,\rangle \to$ 

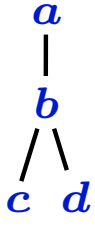
$\langle\, q_{\text{left}}, b(x_1, x_2) \,\rangle \to$ 

$\langle\, q_{\text{left}}, c \,\rangle \to c$

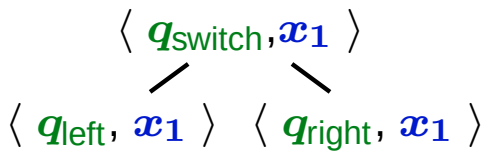
$\langle\, q_{\text{left}}, d \,\rangle \to d$

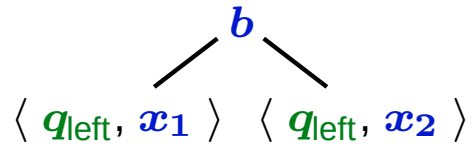
$\langle\, q_{\text{right}}, b(x_1, x_2) \,\rangle \to e$

$\langle\, q_{\text{switch}}, b(x_1, x_2) \,\rangle\, (y_1, y_2) \to$ 



Macro Tree Transducers: Example

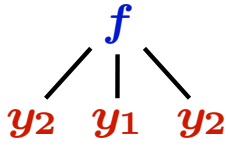
$\langle q_{\text{copy}}, a(x_1) \rangle \to$ 

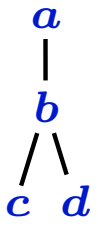
$\langle q_{\text{left}}, b(x_1, x_2) \rangle \to$ 

$\langle q_{\text{left}}, c \rangle \to c$

$\langle q_{\text{left}}, d \rangle \to d$

$\langle q_{\text{right}}, b(x_1, x_2) \rangle \to e$

$\langle q_{\text{switch}}, b(x_1, x_2) \rangle (y_1, y_2) \to$ 

$\langle q_{\text{copy}},$  $\rangle$

Macro Tree Transducers: Example

$\langle\, q_{\text{copy}}, a(x_1) \,\rangle \to$ $\langle\, q_{\text{switch}}, x_1 \,\rangle$ with children $\langle\, q_{\text{left}}, x_1 \,\rangle$ and $\langle\, q_{\text{right}}, x_1 \,\rangle$

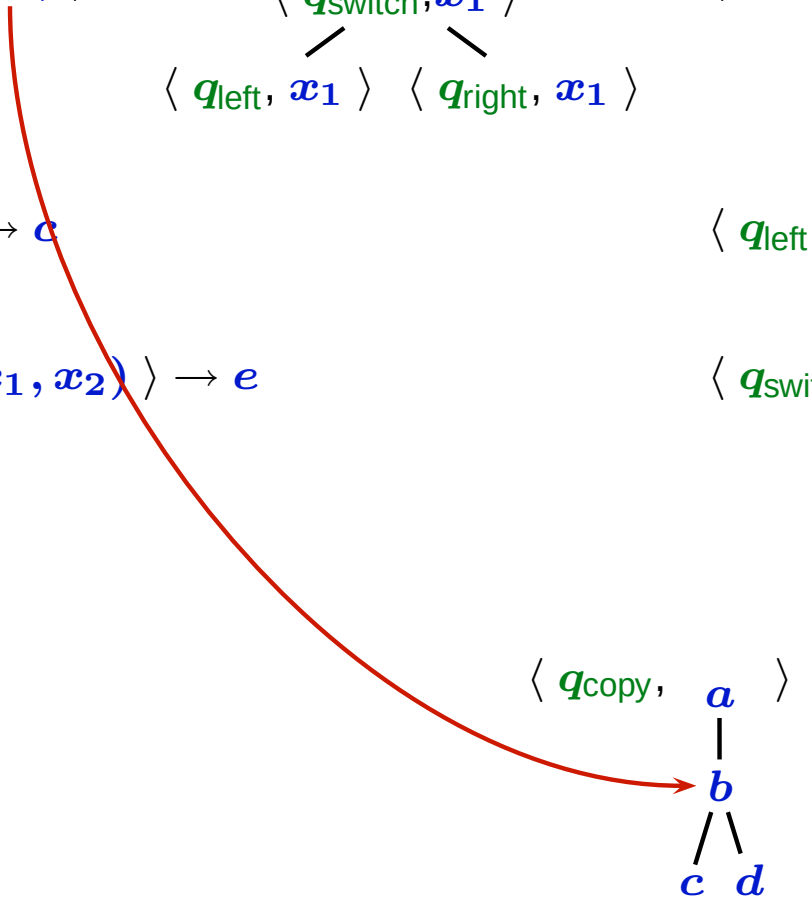
$\langle\, q_{\text{left}}, b(x_1, x_2) \,\rangle \to$ b with children $\langle\, q_{\text{left}}, x_1 \,\rangle$ and $\langle\, q_{\text{left}}, x_2 \,\rangle$

$\langle\, q_{\text{left}}, c \,\rangle \to c$

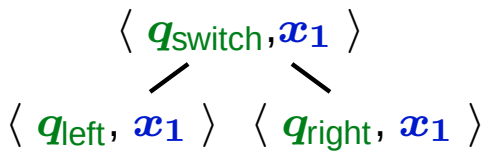
$\langle\, q_{\text{left}}, d \,\rangle \to d$

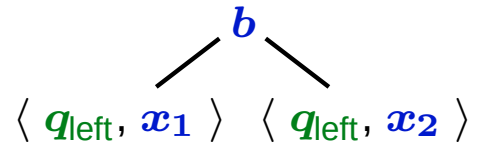
$\langle\, q_{\text{right}}, b(x_1, x_2) \,\rangle \to e$

$\langle\, q_{\text{switch}}, b(x_1, x_2) \,\rangle\, (y_1, y_2) \to$ f with children y_2, y_1, y_2



Macro Tree Transducers: Example

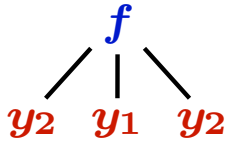
$\langle\, q_{\text{copy}}, a(x_1) \,\rangle \to$ 

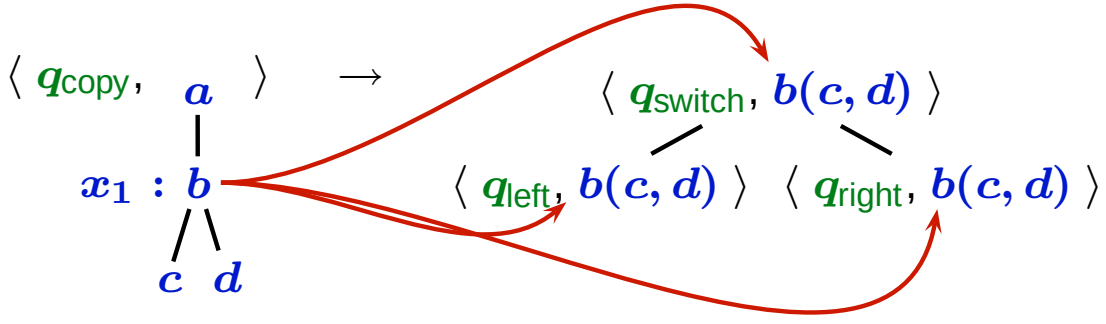
$\langle\, q_{\text{left}}, b(x_1, x_2) \,\rangle \to$ 

$\langle\, q_{\text{left}}, c \,\rangle \to c$

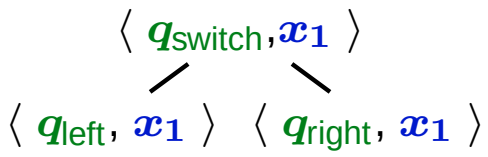
$\langle\, q_{\text{left}}, d \,\rangle \to d$

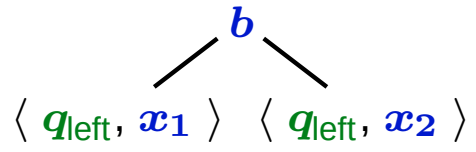
$\langle\, q_{\text{right}}, b(x_1, x_2) \,\rangle \to e$

$\langle\, q_{\text{switch}}, b(x_1, x_2) \,\rangle\, (y_1, y_2) \to$ 



Macro Tree Transducers: Example

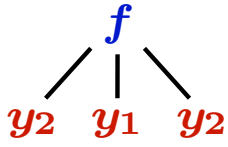
$\langle\, q_{\text{copy}}, a(x_1) \,\rangle \to$ 

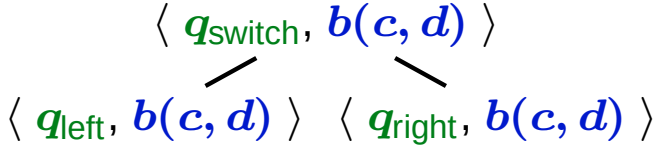
$\langle\, q_{\text{left}}, b(x_1, x_2) \,\rangle \to$ 

$\langle\, q_{\text{left}}, c \,\rangle \to c$

$\langle\, q_{\text{left}}, d \,\rangle \to d$

$\langle\, q_{\text{right}}, b(x_1, x_2) \,\rangle \to e$

$\langle\, q_{\text{switch}}, b(x_1, x_2) \,\rangle\, (y_1, y_2) \to$ 



Macro Tree Transducers: Example

$\langle q_{\text{copy}}, a(x_1) \rangle \to \langle q_{\text{switch}}, x_1 \rangle$ with children $\langle q_{\text{left}}, x_1 \rangle$ and $\langle q_{\text{right}}, x_1 \rangle$

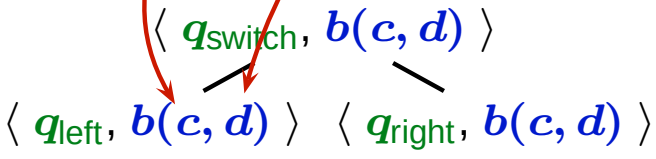
$\langle q_{\text{left}}, b(x_1, x_2) \rangle \to b$ with children $\langle q_{\text{left}}, x_1 \rangle$ and $\langle q_{\text{left}}, x_2 \rangle$

$\langle q_{\text{left}}, c \rangle \to c$

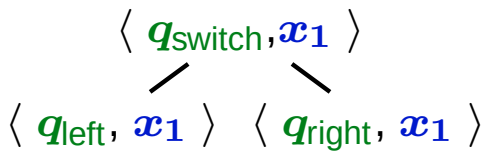
$\langle q_{\text{left}}, d \rangle \to d$

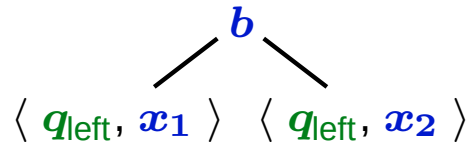
$\langle q_{\text{right}}, b(x_1, x_2) \rangle \to e$

$\langle q_{\text{switch}}, b(x_1, x_2) \rangle (y_1, y_2) \to f$ with children y_2, y_1, y_2



Macro Tree Transducers: Example

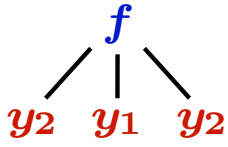
$\langle\, q_{\text{copy}}, a(x_1) \,\rangle \to$ 

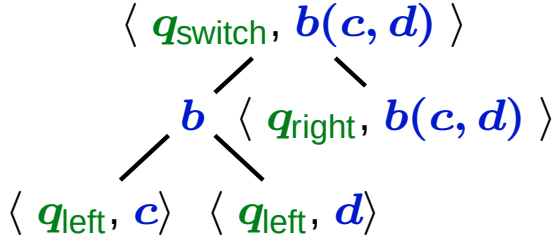
$\langle\, q_{\text{left}}, b(x_1, x_2) \,\rangle \to$ 

$\langle\, q_{\text{left}}, c \,\rangle \to c$

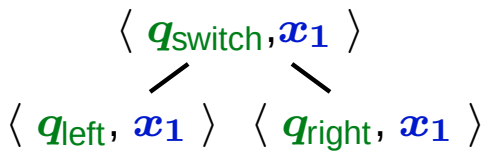
$\langle\, q_{\text{left}}, d \,\rangle \to d$

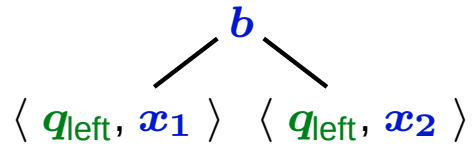
$\langle\, q_{\text{right}}, b(x_1, x_2) \,\rangle \to e$

$\langle\, q_{\text{switch}}, b(x_1, x_2) \,\rangle\, (y_1, y_2) \to$ 



Macro Tree Transducers: Example

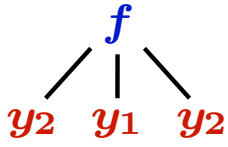
$\langle\, q_{\text{copy}}, a(x_1) \,\rangle \to$ 

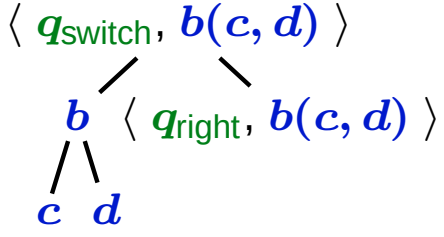
$\langle\, q_{\text{left}}, b(x_1, x_2) \,\rangle \to$ 

$\langle\, q_{\text{left}}, c \,\rangle \to c$

$\langle\, q_{\text{left}}, d \,\rangle \to d$

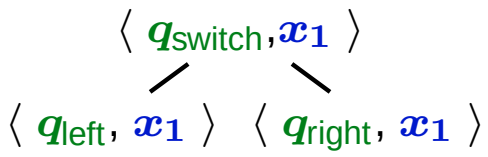
$\langle\, q_{\text{right}}, b(x_1, x_2) \,\rangle \to e$

$\langle\, q_{\text{switch}}, b(x_1, x_2) \,\rangle\, (y_1, y_2) \to$ 

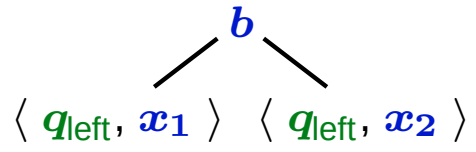


Macro Tree Transducers: Example

$\langle\, q_{\text{copy}}, a(x_1) \,\rangle \to$



$\langle\, q_{\text{left}}, b(x_1, x_2) \,\rangle \to$

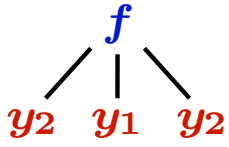


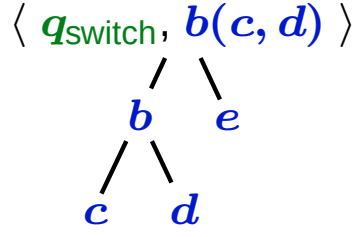
$\langle\, q_{\text{left}}, c \,\rangle \to c$

$\langle\, q_{\text{left}}, d \,\rangle \to d$

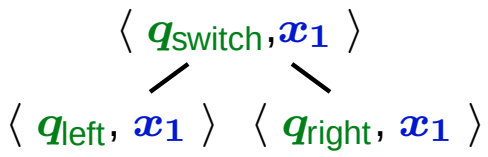
$\langle\, q_{\text{right}}, b(x_1, x_2) \,\rangle \to e$

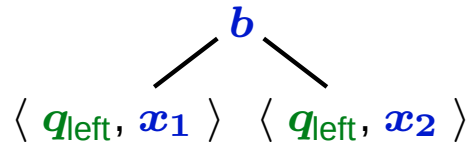
$\langle\, q_{\text{switch}}, b(x_1, x_2) \,\rangle\, (y_1, y_2) \to$





Macro Tree Transducers: Example

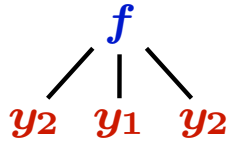
$\langle\, q_{\text{copy}}, a(x_1) \,\rangle \rightarrow$ 

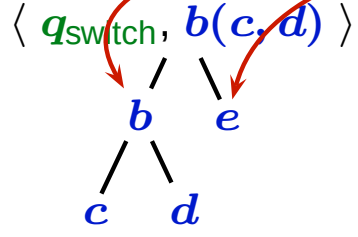
$\langle\, q_{\text{left}}, b(x_1, x_2) \,\rangle \rightarrow$ 

$\langle\, q_{\text{left}}, c \,\rangle \rightarrow c$

$\langle\, q_{\text{left}}, d \,\rangle \rightarrow d$

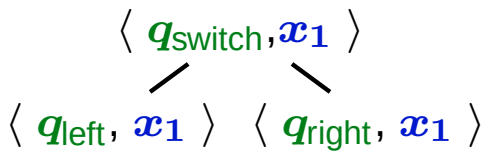
$\langle\, q_{\text{right}}, b(x_1, x_2) \,\rangle \rightarrow e$

$\langle\, q_{\text{switch}}, b(x_1, x_2) \,\rangle\ (y_1, y_2) \rightarrow$ 

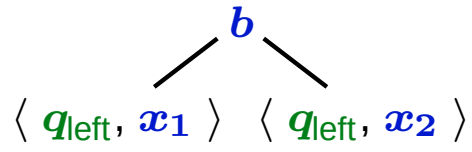


Macro Tree Transducers: Example

$\langle\, q_{\text{copy}}, a(x_1) \,\rangle \rightarrow$



$\langle\, q_{\text{left}}, b(x_1, x_2) \,\rangle \rightarrow$

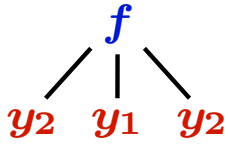


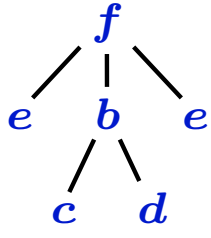
$\langle\, q_{\text{left}}, c \,\rangle \rightarrow c$

$\langle\, q_{\text{left}}, d \,\rangle \rightarrow d$

$\langle\, q_{\text{right}}, b(x_1, x_2) \,\rangle \rightarrow e$

$\langle\, q_{\text{switch}}, b(x_1, x_2) \,\rangle\, (y_1, y_2) \rightarrow$





More Models. . .

. . . are briefly described in the paper:

k-pebble tree transducers: [Milo, Suciu, Vianu 2000]

- Very powerful, designed to model many XML transformation languages
- Typechecking is decidable, non-elementary

TL transformers: [Maneth et al. 2005]

- Strict extension of Simple Tree Transducers
- Typechecking is decidable, space $O(2^{2^{2^{2^{O(n)}}}})$

the query language QL: [Alon et al. 2001]

- Models transformations that compare data values
- Typechecking is undecidable in very restricted cases

Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

Emptiness Test

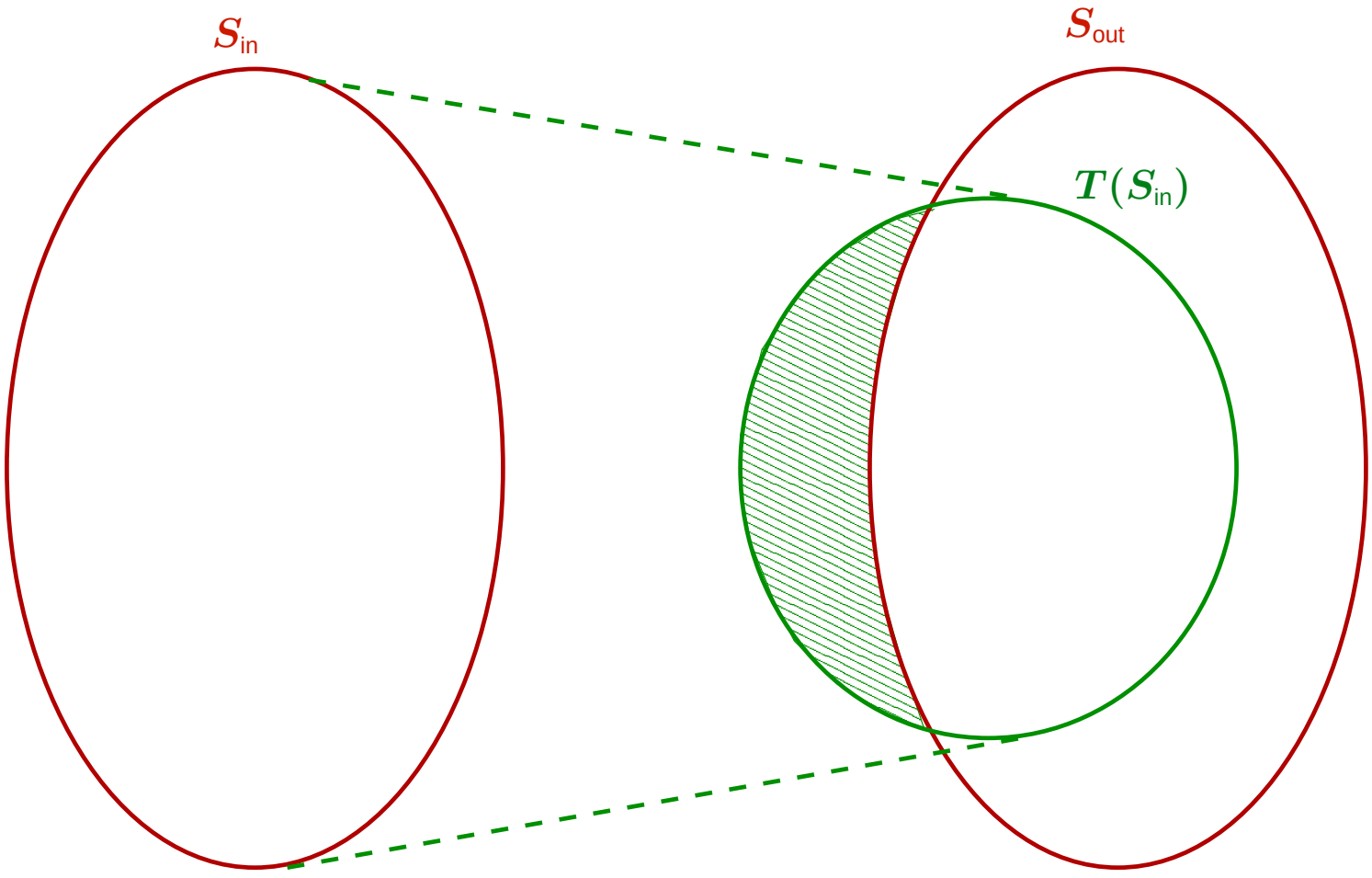
Given a schema S, is $L(S) = \emptyset$?

Reduce entire typechecking problem to
emptiness of one big EDTD (or unranked tree automaton)

Advantage: Emptiness of EDTDs is in **PTIME**

So, the size of the EDTD approximates the time complexity!

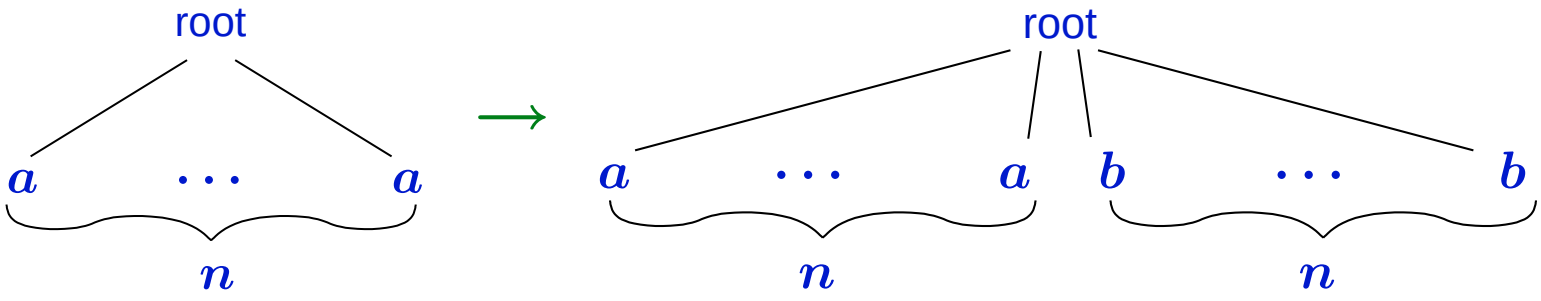
Type Inference



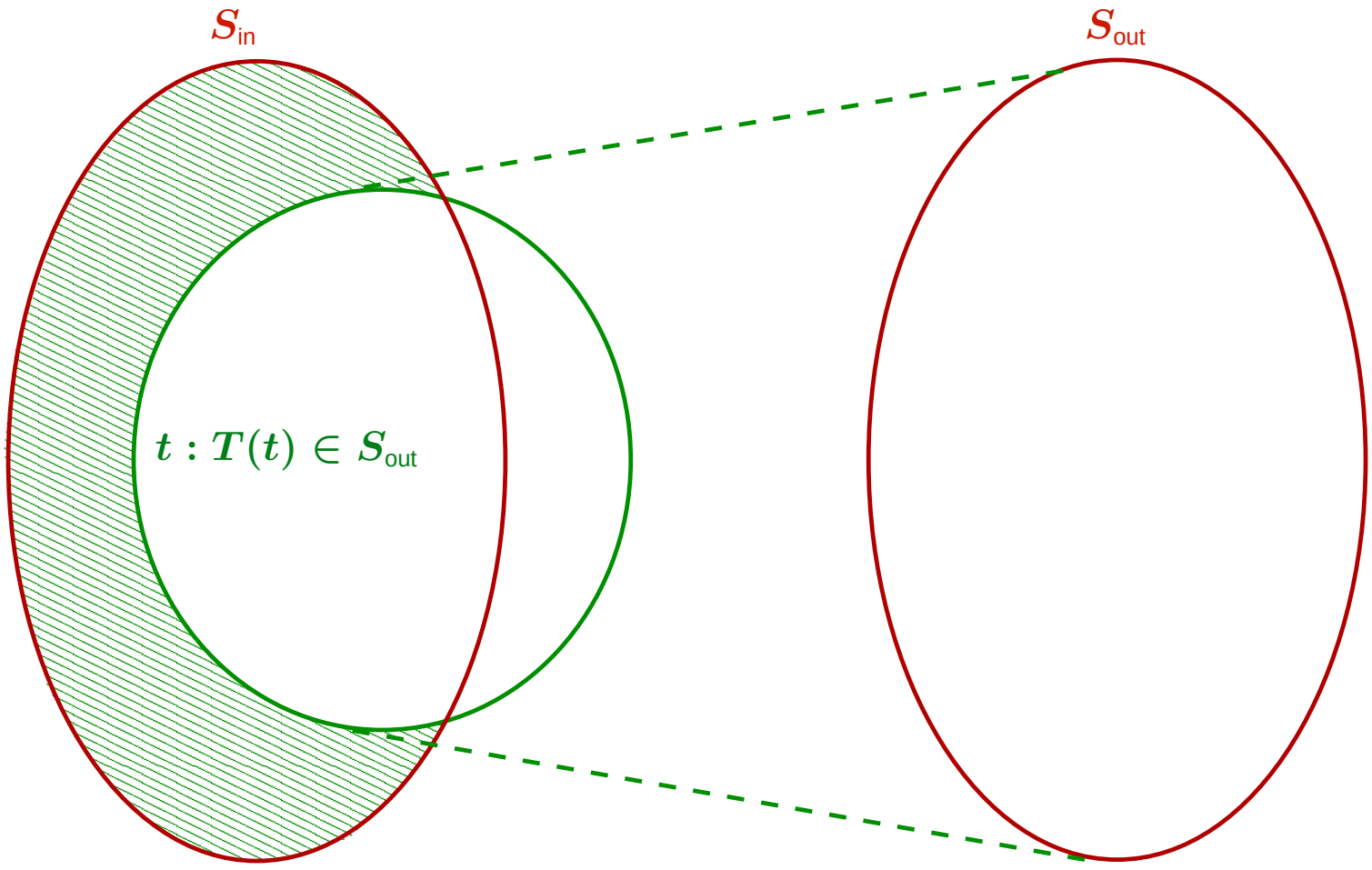
Possible problem: Image is not always regular!

Type Inference:Problem

Image is not always regular: simple example.



Inverse Type Inference



Good news: Pre-image is regular in many cases!

Reduce to Compositions

Write transformation as a composition of MTTs

Inverse type inference through a MTT is doubly exponential

Write transformation as a composition of kPTTs

Inverse type inference through kPTT hyperexponential in k

Seems more aimed towards decidability results

Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

Inclusion

Is $L(S_{\text{in}}) \subseteq L(S_{\text{out}})$?

Sounds trivial, but is still useful when. . .

. . . input schema is fixed

Universality: $\mathcal{T}_\Sigma \subseteq L(S_{\text{out}})$

. . . output schema is fixed

Emptiness: $L(S_{\text{in}}) \subseteq \emptyset$

Simulation

Simulate automata by the tree transducer

Interesting problem: Intersection Emptiness

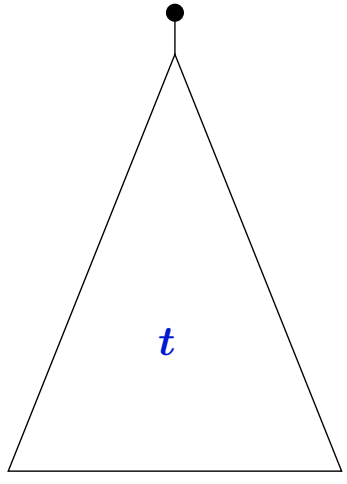
Given n automata $A_1, \ldots, A_n$, is

$$L(A_1) \cap \cdots \cap L(A_n) = \emptyset?$$

Simulation

Given n automata $A_1, \ldots, A_n$, is

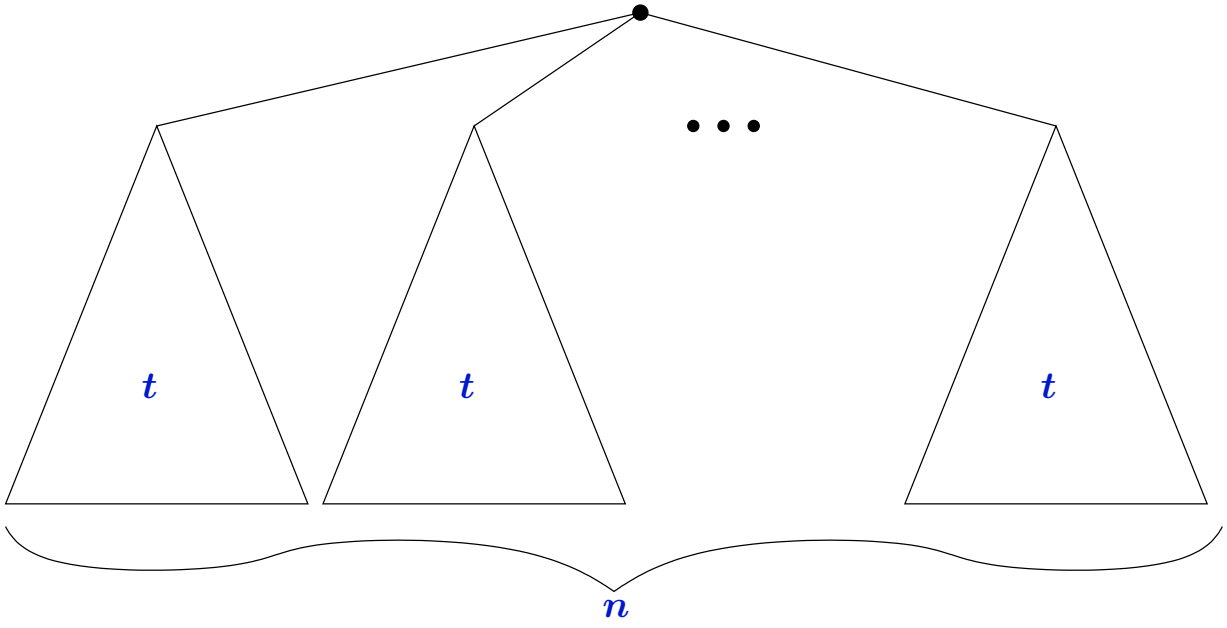
$$L(A_1) \cap \cdots \cap L(A_n) = \emptyset?$$



Simulation

Given n automata $A_1, \ldots, A_n$, is

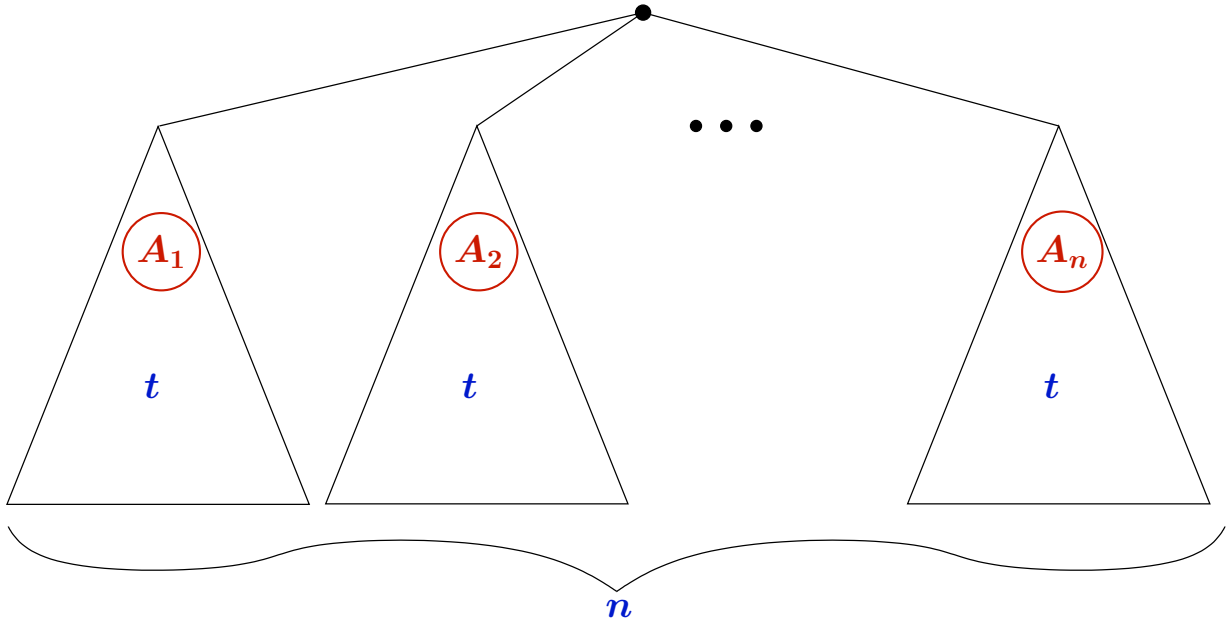
$$L(A_1) \cap \cdots \cap L(A_n) = \emptyset?$$



Simulation

Given n automata $A_1, \ldots, A_n$, is

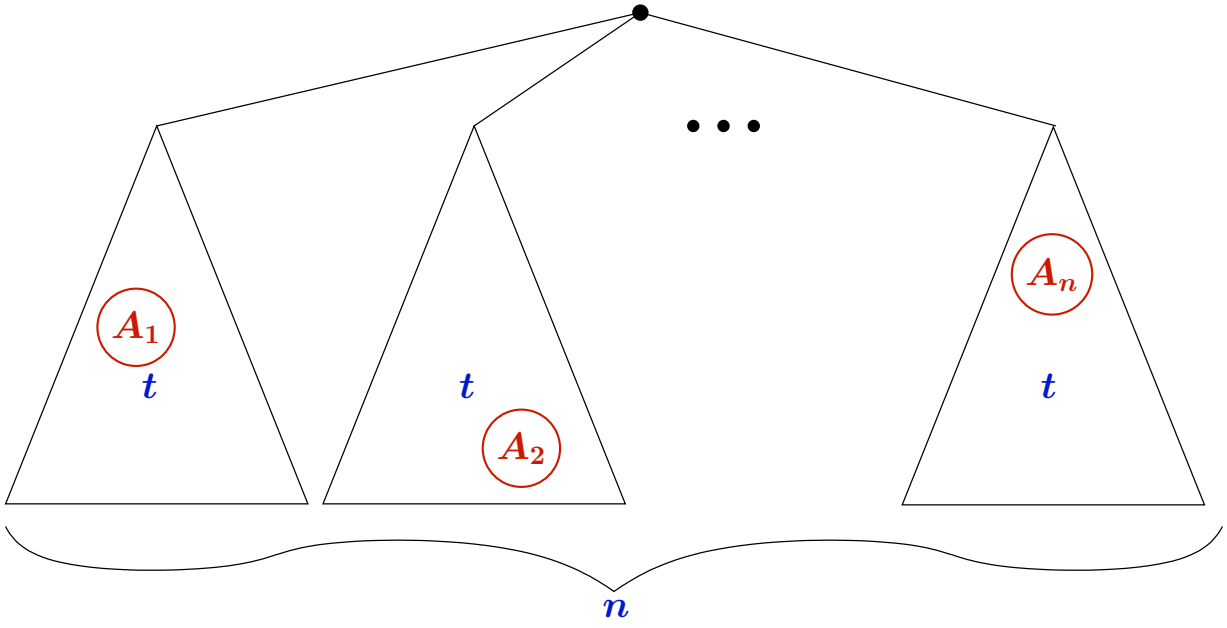
$$L(A_1) \cap \cdots \cap L(A_n) = \emptyset?$$



Simulation

Given n automata $A_1, \ldots, A_n$, is

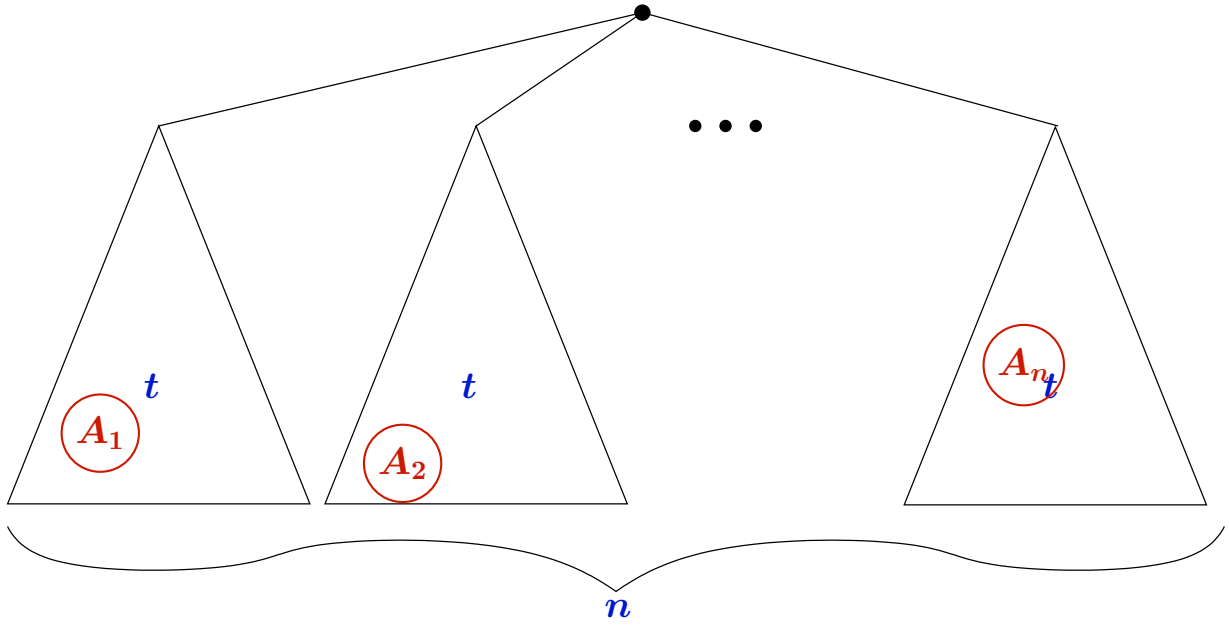
$$L(A_1) \cap \cdots \cap L(A_n) = \emptyset?$$



Simulation

Given n automata $A_1, \ldots, A_n$, is

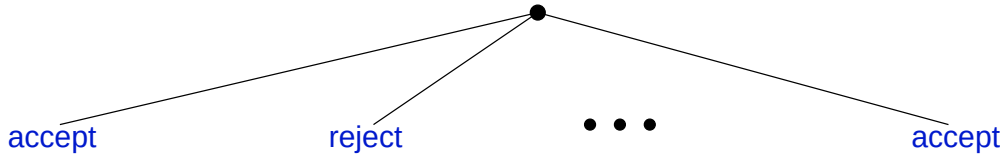
$$L(A_1) \cap \cdots \cap L(A_n) = \emptyset?$$



Simulation

Given n automata $A_1, \ldots, A_n$, is

$$L(A_1) \cap \cdots \cap L(A_n) = \emptyset?$$



Outline

- What is typechecking?
- Which varieties have been investigated?
- What do I want to discuss?
- Formal models for the typechecking problem
  - XML schema languages
  - XML transformations
- Methods for the typechecking problem
  - Proving upper bounds
  - Proving lower bounds

What is Left to do?

- Quest for large tractable fragments
- Quest for large decidable fragments
- Settings where input and/or output schemas are fixed are also relevant in practice
- Building a complete typechecker
- Building a hybrid complete/incomplete typechecker