XML Schemas Admitting 1-Pass Preorder Typing

Wim Martens    Frank Neven    Thomas Schwentick
Outline

- XML Schema Languages
- Single-Type SDTDs
- 1-Pass Preorder Typing
- Restrained Competition SDTDs
- Unique Particle Attribution vs 1PPT
- Conclusion
XML Schema Languages: DTDs

DTDs (Document Type Definitions):

store → guitar guitar*
guitar → maker price
XML Schema Languages: DTDs

DTDs (Document Type Definitions):

```
store → guitar guitar*
guitar → maker price
```

```
store
  ↓
guitar
    ↓
maker
      ↓
"Fender"
    ↓
1000
  ↓
maker
    ↓
"Gibson"
    ↓
price
      ↓
2500
```
XML Schema Languages: DTDs

Regular expressions should be deterministic

- Backward compatibility with SGML:
  "It is an error if an element in the document can match more than one occurrence of an element type in the content model [without looking ahead]."

- Example:
  \[bc + bd\]. Where do we match \(b\) in the string \(bd\)? \(b(c + d)\) is deterministic.

Purpose: facilitate validation!
Regular expressions should be deterministic

- Backward compatibility with SGML:
  “It is an error if an element in the document can match more than one occurrence of an element type in the content model [without looking ahead].”

- Example:
  $bc + bd$. Where do we match $b$ in the string $bd$?
  $b(c + d)$ is deterministic.

Purpose: facilitate validation!

In which way does this constrain the schemas?
XML Schema Languages: DTDs

[Brüggemann-Klein, Wood 1998]:

- Can you recognize deterministic regular expressions?

- What are the properties of deterministic regular expressions?

- Is it decidable whether a regexp is equivalent to a deterministic one?
Can you recognize deterministic regular expressions?
A regular expression is **deterministic** (one-unambiguous) iff its Glushkov automaton is deterministic (PTIME).

What are the properties of deterministic regular expressions?
- Not every regular language can be denoted by a deterministic regular expression. E.g., $(a + b)^* a(a + b)$.
- **Semantical characterization** in terms of orbits

Is it decidable whether a regexp is equivalent to a deterministic one?
Yes
Specialized DTDs (SDTDs) [Papak., Vianu, 2000]:
≡ tree automata on unranked trees

\[
\begin{align*}
\text{store} & \rightarrow (\text{guitar}^1)^* \text{ guitar}^2 (\text{guitar}^2)^* \\
\text{guitar}^1 & \rightarrow \text{ maker price} \\
\text{guitar}^2 & \rightarrow \text{ maker price discount}
\end{align*}
\]
XML Schema: Regular Tree Languages

Specialized DTDs (SDTDs) [Papak., Vianu, 2000]:
≡ tree automata on unranked trees

store → (guitar\(^1\))\(^*\) guitar\(^2\) (guitar\(^2\))\(^*\)
guitar\(^1\) → maker price
guitar\(^2\) → maker price discount

```
store
  └── guitar
    └── maker
        ├── price
        │   └── "Fender"
        │       └── 1000
    └── guitar
        └── maker
            └── price
                └── "Gibson"
                    └── 2500
```

```
store
  └── guitar
    └── maker
        └── price
            └── "Paul Reed Smith"
                └── 3500
```

```
  └── discount
      └── "10%"
```
Specialized DTDs (SDTDs) [Papak., Vianu, 2000]:
≡ tree automata on unranked trees

store \rightarrow (guitar^1)^* \text{guitar}^2 (guitar^2)^*

\text{guitar}^1 \rightarrow \text{maker price}
\text{guitar}^2 \rightarrow \text{maker price discount}

Typing: associating the right types to nodes
XML Schema

To facilitate validation/typing:

Element Declarations Consistent Rule (EDC):

“It is illegal to have two elements of the same name […] but different types in a content model”

[XML Schema Part 0: Primer]

XML Schemas are SDTDs with a single-type restriction

[Murata, Lee, Mani 2001]
Outline

- XML Schema Languages
- Single-Type SDTDs
- 1-Pass Preorder Typing
- Restrained Competition SDTDs
- Unique Particle Attribution vs 1PPT
- Conclusion
EDC in SDTDs

Single-type SDTDs:

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

Example:

\[
\begin{align*}
\text{store} & \rightarrow (\text{guitar}^1)^* \text{guitar}^2 (\text{guitar}^2)^* \\
\text{guitar}^1 & \rightarrow \text{maker}^2 \text{price}^3
\end{align*}
\]

not allowed

allowed
EDC in SDTDs

Single-type SDTDs:

- store → regulars* discounts discounts*
- regulars → guitar¹
- discounts → guitar²
- guitar¹ → maker price
- guitar² → maker price discount
EDC in SDTDs

Single-type SDTDs:

store → regulars* discounts discounts*
regulars → guitar¹
discounts → guitar²
guitar¹ → maker price
guitar² → maker price discount

- store: "Fender" 1000
- store: "Gibson" 2500
- store: "Paul Reed Smith" 3500
- discount: "10%"
EDC in SDTDs

Single-type SDTDs:

- store $\rightarrow$ regulars* discounts discounts*
- regulars $\rightarrow$ guitar$^1$
- discounts $\rightarrow$ guitar$^2$
- guitar$^1$ $\rightarrow$ maker price
- guitar$^2$ $\rightarrow$ maker price discount

Note: DTD $\subseteq$ single-type SDTD $\subseteq$ SDTD
Questions

- Can you recognize single-type SDTDs?
  *Trivial*

- What kind of languages can be defined by single-type SDTDs?
  *???

- Is it decidable whether an SDTD is equivalent to a single-type SDTD?
  *???
The Ancestor-String
Ancestor-Guarded Subtree Exchange

\( T \) a regular tree language

\( \in T \) \( \in T \) \( \Rightarrow \) \( \in T \)
The Equivalence

Let $T$ be a regular tree language

**THEOREM:** The following are equivalent:

- $T$ is definable by a single-type SDTD
- $T$ is closed under ancestor-guarded subtree exchange
Ancestor-guarded DTD consists of triples \((r, a) \rightarrow s\)
Ancestor-Guarded DTDs

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

Example:

```
(\epsilon, \text{store}) \rightarrow \text{regulars}^* \text{ discounts discounts}^*
(store, regulars) \rightarrow \text{guitar}
(store, discounts) \rightarrow \text{guitar}
(store regulars, guitar) \rightarrow \text{maker price}
(store discounts, guitar) \rightarrow \text{maker price discount}
```

```
store
  -- regulars
    -- guitar\textsuperscript{1}
      -- maker price
        -- "Fender" price 1000
    -- guitar\textsuperscript{1}
      -- maker price
        -- "Gibson" price 2500
  -- discounts
    -- guitar\textsuperscript{2}
      -- maker price discount
        -- "Paul Reed Smith" price 3500 discount "10%"
```
The Equivalence

Let $T$ be a regular tree language

**THEOREM:** The following are equivalent:

- $T$ is definable by a **single-type** SDTD
- $T$ is closed under **ancestor-guarded subtree exchange**
- $T$ is definable by an **ancestor-guarded** DTD
Questions

Can you recognize single-type SDTDs?
Trivial

What kind of languages can be defined by single-type SDTDs?

Semantical characterizations:
- ancestor-guarded subtree exchange
- ...

Syntactical characterizations:
- ancestor-guarded DTDs
- ...

Is it decidable whether an SDTD is equivalent to a single-type SDTD?
Questions

- Can you recognize single-type SDTDs?
  *Trivial*

- What kind of languages can be defined by single-type SDTDs?
  - *Semantical characterizations:*
    - ancestor-guarded subtree exchange
    - ...
  - *Syntactical characterizations:*
    - ancestor-guarded DTDs
    - ...

- Is it decidable whether an SDTD is equivalent to a single-type SDTD?
  *Yes, EXPTIME-complete*
Outline

- XML Schema Languages
- Single-Type SDTDs
- 1-Pass Preorder Typing
- Restrained Competition SDTDs
- Unique Particle Attribution vs 1PPT
- Conclusion
Optimize EDC?

EDC: type of a node only depends on ancestor-string!
Optimize EDC?

EDC: type of a node **only depends on ancestor-string**!

XML streaming: **determine the type** of a node when its opening tag is met

We call this **1-Pass Preorder Typing**
Optimize EDC?

EDC: type of a node only depends on ancestor-string!

XML streaming: determine the type of a node when its opening tag is met

We call this 1-Pass Preorder Typing

≠ 1-Pass Preorder Validation

\[
\begin{align*}
    a & \rightarrow b^1 + b^2 \\
    b^1 & \rightarrow c \\
    b^2 & \rightarrow d
\end{align*}
\]

defines trees

\[
\begin{align*}
    a & \quad a \\
    b^1 & \quad b^2 \\
    c & \quad d
\end{align*}
\]
Questions

Can you recognize 1PPT SDTDs?

What kind of languages can be defined by 1PPT SDTDs?

Is it decidable whether an SDTD is equivalent to a 1PPT SDTD?
Outline

- XML Schema Languages
- Single-Type SDTDs
- 1-Pass Preorder Typing
- Restrained Competition SDTDs
- Unique Particle Attribution vs 1PPT
- Conclusion
Restrained Competition SDTDs

[Murata et al., 2001]: A regular expression $r$ over types restrains competition iff there are no strings $wa^i v$ and $wa^j v'$ in $L(r)$ with $i \neq j$

I.e. the type of $a$ is determined by its left siblings

An SDTD is restrained competition iff every regular expression restrains competition

\[
\begin{align*}
\text{store} & \rightarrow (\text{guitar}^1)\ast\text{discounts}\ (\text{guitar}^2)^+ \\
\text{discounts} & \rightarrow \varepsilon \\
\text{guitar}^1 & \rightarrow \text{maker price} \\
\text{guitar}^2 & \rightarrow \text{maker price discount}
\end{align*}
\]

Note: $\text{DTD} \subseteq \text{stSDTD} \subseteq \text{rcSDTD} \subseteq \text{SDTD}$
The Ancestor-Sibling-String

α
Anc-Sib-Guarded Subtree Exchange

$T$ a regular tree language
Ancestor-Sibling-Guarded DTDs

Anc-Sib guarded DTD consists of triples \((r, a) \rightarrow s\)
Ancestor-Sibling-Guarded DTDs

Anc-Sib guarded DTD consists of triples \((r, a) \rightarrow s\)

Example:

\[(\varepsilon, \text{store}) \rightarrow \text{guitar}^* \text{ discounts guitar}^+\]

\[(\text{store} \# \text{guitar}^*, \text{discounts}) \rightarrow \varepsilon\]

\[(\text{store} \# \text{guitar}^*, \text{guitar}) \rightarrow \text{maker price}\]

\[(\text{store} \# \text{guitar}^*, \text{discounts guitar}^*, \text{guitar}) \rightarrow \text{maker price discount}\]
The Equivalence

Let \( T \) be a regular tree language.

**THEOREM:** The following are equivalent:

- \( T \) is definable by a **restrained competition** SDTD
- \( T \) is closed under **ancestor-sibling-guarded subtree exchange**
- \( T \) is definable by an **ancestor-sibling-guarded DTD**
The Equivalence

Let $T$ be a regular tree language.

**THEOREM:** The following are equivalent:

- $T$ is definable by a restrained competition SDTD
- $T$ is closed under ancestor-sibling-guarded subtree exchange
- $T$ is definable by an ancestor-sibling-guarded DTD
- $T$ allows 1-Pass Preorder Typing

1-Pass Preorder Typeable SDTDs are exactly the rcSDTDs!
The Equivalence: 1PPT vs rcSDTD

Intuition: not rcSDTD implies not 1PPT
Suppose we have $x \rightarrow r$
where $w_a^i v$ and $w_a^j v'$ in $L(r)$ and $i \neq j$
The Equivalence: 1PPT vs rcSDTD

Intuition: not rcSDTD implies not 1PPT
Suppose we have $x \rightarrow r$
where $wa^i_v$ and $wa^j_{v'}$ in $L(r)$ and $i \neq j$

Then we can make trees

![Diagram](image)
The Equivalence: 1PPT vs rcSDTD

Intuition: not rcSDTD implies not 1PPT
Suppose we have $x \rightarrow r$

where $wa^i v$ and $wa^j v'$ in $L(r)$ and $i \neq j$

Then we can make trees

![Diagram showing trees with nodes labeled x, w, a_i, v, a_j, v', and connections to demonstrate the equivalence between 1PPT and rcSDTD.](image-url)
Questions

- Can you recognize 1PPT SDTDs?

- What kind of languages can be defined by 1PPT SDTDs?
  - Semantical characterizations:
    - restrained competition SDTDs
    - ancestor-sibling guarded subtree-exchange
    - ...
  - Syntactical characterizations:
    - ancestor-sibling guarded DTD
    - ...

- Is it decidable whether an SDTD is equivalent to a 1PPT SDTD?
Questions

- Can you recognize 1PPT SDTDs?
  Yes, in NLOGSPACE

- What kind of languages can be defined by 1PPT SDTDs?
  - **Semantical characterizations:**
    - restrained competition SDTDs
    - ancestor-sibling guarded subtree-exchange
    - ...
  
  - **Syntactical characterizations:**
    - ancestor-sibling guarded DTD
    - ...

- Is it decidable whether an SDTD is equivalent to a 1PPT SDTD?
  Yes, EXPTIME-complete
Outline

- XML Schema Languages
- Single-Type SDTDs
- 1-Pass Preorder Typing
- Restrained Competition SDTDs
- Unique Particle Attribution vs 1PPT
- Conclusion
Unique Particle Attribution vs 1PPT

Unique Particle Attribution

New name for one-unambiguous or determinism constraint

Unique Particle Attribution $\Rightarrow$ 1PPT

Intuition:

\[
\begin{align*}
\begin{array}{cccc}
a^1 ? & b^1 & (b^2 + c^1)^* & a^2 & c^1 \\
a_1 ? & b_2 & (b_3 + c_4)^* & a_5 & c_6 \\
\end{array}
\end{align*}
\]

rc: $w^i a^i v \in L(r), w^j a^j v' \in L(r) \Rightarrow i = j$

deterministic: $w^i a^i v \in L(r), w^j a^j v' \in L(r) \Rightarrow i = j$
Unique Particle Attribution

New name for one-unambiguous or determinism constraint

Unique Particle Attribution $\Rightarrow$ 1PPT

Intuition:

\[
\begin{align*}
\mathit{a} \mathit{1} & \mathit{?} \quad \mathit{b} \mathit{1} \quad (\mathit{b} \mathit{2} & \quad + \quad \mathit{c} \mathit{1})^* \quad \mathit{a} \mathit{2} \quad \mathit{c} \mathit{1} \\
\mathit{a} \mathit{1} & \mathit{?} \quad \mathit{b} \mathit{2} \quad (\mathit{b} \mathit{3} & \quad + \quad \mathit{c} \mathit{4})^* \quad \mathit{a} \mathit{5} \quad \mathit{c} \mathit{6}
\end{align*}
\]

1PPT $\nlessdot$ Unique Particle Attribution

Example: \((\mathit{a} \mathit{1} + \mathit{b} \mathit{1})^* \mathit{a} \mathit{1}(\mathit{a} \mathit{1} + \mathit{b} \mathit{1})\)
Outline

- XML Schema Languages
- Single-Type SDTDs
- 1-Pass Preorder Typing
- Restrained Competition SDTDs
- Unique Particle Attribution vs 1PPT
- Conclusion
Conclusions

- EDC, 1PPT have elegant *semantical* characterizations
- Characterizations provide a *toolbox* for proofs
Conclusions

- 1PPT is strictly larger than EDC
- 1PPT is a robust notion
- 1PPT has a syntactical counterpart
- Validation and typing against 1PPT essentially not harder than against EDC
- When content models are deterministic:
  - inclusion/equivalence of 1PPT SDTDs in \text{PTIME}
  - minimizing is in \text{PTIME}, unique minimal 1PPT SDTD

These are the same complexities as for EDC SDTDs!
Conclusions

- 1PPT is strictly larger than EDC
- 1PPT is a robust notion
- 1PPT has a syntactical counterpart
- Validation and typing against 1PPT essentially not harder than against EDC
- When content models are deterministic:
  - inclusion/equivalence of 1PPT SDTDs in \( \text{PTIME} \)
  - minimizing is in \( \text{PTIME} \), unique minimal 1PPT SDTD
- These are the same complexities as for EDC SDTDs!
- EDC is currently in XML Schema specification!