Verifying recursive active documents with positive data tree rewriting

Anca Muscholl

Bordeaux

with B. Genest (Rennes), Z. Wu (Bordeaux)
Active documents (Abiteboul & co)

**Document trees**
- XML: unranked, unordered, labeled trees (tags on inner nodes, data on leaves)
- Active XML (AXML): extended by service nodes

**AXML trees**
```
book
|--- title
|   |--- HP7
|--- author
|   |--- JRK
|--- ISBN
|   |--- 207-...
|--- translations
|   |--- french
|   |--- hindi
|--- sales
|   |--- 3 Mio.
```

upd@publishers: update translations and number of sold items
**Service calls**

- query document on given peer (example: service *upd* at peer *publishers*)
- add query result to the original tree at a designated node (materialization of call)

**System**

Active documents evolve over time (branching, since calls are not confluent). They can be viewed as **tree/graph rewriting systems**, with RHS described implicitly by queries.
**Service calls**
- query document on given peer (example: service \textit{upd} at peer \textit{publishers})
- add query result to the original tree at a designated node (materialization of call)

**System**
Active documents evolve over time (branching, since calls are not confluent). They can be viewed as \textit{tree/graph rewriting systems}, with RHS described implicitly by queries.

**Remark**
- We consider here w.l.o.g. a single peer (i.e., a single document tree).
- Query results may contain further service calls (recursion). Order in which services are called can be relevant.
- Particular setting for \textit{infinite-state} systems. \textit{“Infinite”} stems from recursion \textit{and data}. 

Anca Muscholl (Bordeaux)
DOTS, Mar. 2010
Questions and background

Questions

- **Abstract model**: give a “simple” model for such systems including workflow reasoning.
- **Verification**: determine the frontier(s) for deciding properties on the evolution of such systems.
Questions and background

Questions

- **Abstract model**: give a “simple” model for such systems including workflow reasoning.
- **Verification**: determine the frontier(s) for deciding properties on the evolution of such systems.

Background

- **Monotonous AXML documents** (Abiteboul/Benjelloun/Milo’04): service calls are always enabled. Termination decidable. No data.
- **Tree pattern rewriting systems (TPRS)** (Genest/M./Serre/Zeitoun’08): guarded tree rewriting model. No data. Decidability for termination and pattern reachability for positive (recursive) TPRS.
- **Guarded AXML (GAXML)** (Abiteboul/Segoufin/Vianu’08): AXML + guards. Verification of temporal properties decidable for recursion-free systems.
Questions and background

Questions

- Abstract model: give a “simple” model for such systems including workflow reasoning.
- Verification: determine the frontier(s) for deciding properties on the evolution of such systems.

Background

- Monotonous AXML documents (Abiteboul/Benjelloun/Milo’04): service calls are always enabled. Termination decidable. No data.
- Tree pattern rewriting systems (TPRS) (Genest/M./Serre/Zeitoun’08): guarded tree rewriting model. No data. Decidability for termination and pattern reachability for positive (recursive) TPRS.
- Guarded AXML (GAXML) (Abiteboul/Segoufin/Vianu’08): AXML + guards. Verification of temporal properties decidable for recursion-free systems.
- Here we extend TPRS by adding data: DTPRS.
Abstract model

Data trees

Trees with inner nodes labeled by tags (finite alphabet) and leaves labeled by data (infinite alphabet) or tags.
Abstract model

Data trees
Trees with inner nodes labeled by tags (finite alphabet) and leaves labeled by data (infinite alphabet) or tags.

Data tree patterns (DTP)
- Tree $P$ with node labels from $\Sigma \cup \{\text{any}\}$, child/descendant edges and data (in)equality constraints on leaves.
- Match a pattern $P$ against a document $T$: (injective) mapping from $P$ into $T$, preserving the root, the labels, the edge relations and the data constraints.
- Relative DTP, boolean combination of (relative) DTP.
**Abstract model**

**Data trees**
Trees with inner nodes labeled by *tags* (finite alphabet) and leaves labeled by *data* (infinite alphabet) or tags.

**Data tree patterns (DTP)**
- Tree \( P \) with node labels from \( \Sigma \cup \{ \text{any} \} \), child/descendant edges and data (in)equality constraints on leaves.
- Match a pattern \( P \) against a document \( T \): (injective) mapping from \( P \) into \( T \), preserving the root, the labels, the edge relations and the data constraints.
- Relative DTP, boolean combination of (relative) DTP.

**DTP queries**
- Already in GAXML (and TPRS): DTP queries (DTPQ).
- DTPQ: \( \text{body} \rightsquigarrow \text{head} \), with \( \text{body} \) DTP and \( \text{head} \) “collecting” results. Query returns the forest of all instantiations of \( \text{head} \) (by matching \( \text{body} \) against data tree).
Example

$P$ matches $T$: 

$P$: any

- editor
  - ISBN
  - country
  - $X$
  - $Y$

- editor
  - ISBN
  - country
  - $X$
  - $Z$

$T$: amazon
europe

- editor
  - name
  - country
  - Egmont
  - Romania
  - ISBN
  - 207-...

- editor
  - country
  - Russia
  - ISBN
  - 207-...
**Abstract model**

**Data tree rewriting systems (DTPRS)**

DTP rule:
- **Locator**: relative DTP
  - “context” of the rule
  - nodes labeled by “actions”: rename/delete/append
- **Guard**: boolean combination of (relative) DTPs
- **Queries**: DTPQs used by *append*

**Example**

```
Play.com

CCatalog
  Customer
    CId Name Email
      Serge

PCatalog
  Product
    PLId Name Price token
      9221 Rolex $400
```
Example (contd.)

Rule “create cart” (connect to Play.com):

Locator:

\[
\text{Play.com} \quad \text{append}(F)
\]

\[
\text{CCatalog} \quad \text{Customer} \quad \text{CId}
\]

\[
X
\]

Appended forest \( F \):

\[
\text{Cart} \quad \log \quad \text{products} \quad \text{select}
\]

\[
\text{CId} \quad X
\]
DTP rules

Example (contd.)

Add-product:

Play.com

Cart

PCatalog

Product

products

select

Customer

log

CCatalog

Cart

PCatalog

Product

CCatalog

Cart

Customer

log

products

select

CId

Name

Email

CIId

Name

Email

CIId

Name

Email

CIId

Name

Email

F: PIId

X

del

Serge

Rolex

$ 400

Serge

Anca Muscholl (Bordeaux)
Finite set of DTP rules. DTP rule = (locator, guard, queries, forests).

Static invariant: DTD + data constraints.

For termination, fix the initial data tree. For pattern reachability, given set of initial data trees (conjunction of DTD + boolean combination of DTPs).
DTPRS

- Finite set of DTP rules. DTP rule = (locator, guard, queries, forests).
- Static invariant: DTD + data constraints.
- For termination, fix the initial data tree. For pattern reachability, given set of initial data trees (conjunction of DTD + boolean combination of DTPs).

**Applying DTP rules**

- Match (injectively) locator against data tree.
- All variables occurring in the appended forests, but not in locator, get fresh data values. Queries labeling nodes of forests are evaluated.
- Rewriting (if invariant true): delete nodes (and their subtrees) labeled del, rename nodes labeled ren, and attach forests to nodes labeled append.
DTPRS

**Finite set of DTP rules.** DTP rule = (locator, guard, queries, forests).

**Static invariant:** DTD + data constraints.

For termination, fix the initial data tree. For pattern reachability, given set of initial data trees (conjunction of DTD + boolean combination of DTPs).

**Applying DTP rules**

- Match (injectively) locator against data tree.

- All variables occurring in the appended forests, but not in locator, get fresh data values. Queries labeling nodes of forests are evaluated.

- Rewriting (if invariant true): delete nodes (and their subtrees) labeled `del`, rename nodes labeled `ren`, and attach forests to nodes labeled `append`.

**Rem.**

Main difference with previous TPRS (ATVA’08): no “move” of subtrees here. Just for convenience.
Unsurprisingly, DTPRS are undecidable:

- when DTD is recursive (without data),
- when negation of DTPs in guards or invariant (without data).

Restricting both items above still does not help:

- Both termination and pattern reachability are undecidable for DTPRS such that the DTD is non-recursive and guards/static invariant use only positive DTPs.
**Undecidability and Restrictions**

**Undecidability**

Unsurprisingly, DTPRS are undecidable:
- when DTD is recursive (without data),
- when negation of DTPs in guards or invariant (without data).

Restricting both items above still **does not help**:
- Both termination and pattern reachability are undecidable for DTPRS such that the DTD is non-recursive and guards/static invariant use only positive DTPs.

**Horizontal Orders**

```
                    root
                   /   \
                  a     a
                 /   \     \
                a     a     a
               /   \    /   \  
              D_1  D_2 D_2 D_3
                     /   \    \
                    D_N-1 D_N D_N  D_N+1
```
## Data tree $T$ and associated graph $G(T)$

- one additional node for each data value $d$ occurring in $T$,
- additional edges from each leaf with data value $d$ to node $d$.
- $G(T)$ is finitely labeled.

A DTPRS is **path-bounded** if for some $K > 0$, every reachable data tree $T$ is such that simple paths in $G(T)$ are of length at most $K$. 

---

**Positive DTPRS**
## Positive DTPRS

### Data tree $T$ and associated graph $G(T)$
- one additional node for each data value $d$ occurring in $T$,
- additional edges from each leaf with data value $d$ to node $d$.
- $G(T)$ is finitely labeled.

A DTPRS is **path-bounded** if for some $K > 0$, every reachable data tree $T$ is such that simple paths in $G(T)$ are of length at most $K$.

## Positive DTPRS

A DTPRS is **positive** if
- non-recursive & positive DTD in the static invariant,
- no negation of DTPs in guards/static invariant,
- path-bounded.
**Thm.**

Pattern reachability and termination are **decidable** for positive DTPRS.

**Proof**

Positive DTPRS are **well-structured transition systems (WSTS)** + assumptions needed for deciding reachability/termination.
well-quasi-order (wqo) \leq on the set of states

upwards compatibility: for any $T_1 \rightarrow^* T_2$ and $T_1 \leq T_1'$ there is some $T_2'$ with $T_2 \leq T_2'$ and $T_1' \rightarrow^* T_2'$. 
well-quasi-order (wqo) $\leq$ on the set of states

upwards compatibility: for any $T_1 \xrightarrow{*} T_2$ and $T_1 \leq T_1'$ there is some $T_2'$ with $T_2 \leq T_2'$ and $T_1' \xrightarrow{*} T_2'$.

Positive DTPRS are WSTS

Existence of wqo ensured by the uniform bound on depth and simple paths. Upwards compatibility ensured by positive guards/invariant.
well-quasi-order (wqo) \leq on the set of states

upwards compatibility: for any $T_1 \xrightarrow{*} T_2$ and $T_1 \leq T_1'$ there is some $T_2'$ with $T_2 \leq T_2'$ and $T_1' \xrightarrow{*} T_2'$.

Positive DTPRS are WSTS

Existence of wqo ensured by the uniform bound on depth and simple paths. Upwards compatibility ensured by positive guards/invariant.

Wqo \leq

$T \leq T'$ if $\leq$ injective and preserves:

- root and parent-child relation,
- tags and data (in)equality.

$\leq$ is well-founded preorder.
\( \preceq \) is \textit{wqo} on \textit{finitely} labeled, unordered trees of \( B \)-bounded depth (\( B \) fixed).

Let \( G \) be an (undirected) graph s.t. every simple path is of length at most \( K \). Then \( G \) has a \textit{tree decomposition} of depth and width at most \( K \).

Extend the labeling of a \textit{data tree} \( T \), resp. associated graph \( G(T) \), by the depth of each node.

Let \( \tilde{T} \) be (the) tree decomposition of \( G(T) \) of width at most \( K \) (including node labels and edge relations).

\( \tilde{T} \) is finitely labeled and of depth at most \( K \).

Since

\[ \tilde{T}_1 \preceq \tilde{T}_2 \implies T_1 \preceq T_2, \]

the quasi-order \( \preceq \) on data trees is a \textit{wqo}. 
How WSTS works

PROBLEMS

- Termination: given data tree $T$, does an infinite derivation chain exist from $T$?
- Pattern reachability: given a DTP $P$ and an initial set $Init$ of data trees, is there some $T$ reachable from $Init$ that satisfies $P$?
How WSTS works

Problems

- Termination: given data tree $T$, does an infinite derivation chain exist from $T$?
- Pattern reachability: given a DTP $P$ and an initial set $Init$ of data trees, is there some $T$ reachable from $Init$ that satisfies $P$?

Termination

- Forward algorithm: compute the finite reachability tree (stop when current $T$ is s.t. $T' \preceq T$ for some ancestor $T'$).
- The successor relation is effectively computable.
How WSTS works

PROBLEMS

- Termination: given data tree $T$, does an infinite derivation chain exist from $T$?
- Pattern reachability: given a DTP $P$ and an initial set $Init$ of data trees, is there some $T$ reachable from $Init$ that satisfies $P$?

TERMINATION

- Forward algorithm: compute the finite reachability tree (stop when current $T$ is s.t. $T' \preceq T$ for some ancestor $T'$).
- The successor relation is effectively computable.

PATTERN REACHABILITY

- Backwards algorithm: start with upward-closed set of states $I$ and compute iteratively $Pred(I)$ (upward-closed, too). Check whether $I \cap Init \neq \emptyset$.
- Each upward-closed set $I$ can be represented by a finite basis (because of wqo).
- The finite basis is effectively computable.
- Works also for reachability of a positive combination of DTPs.
Beyond pattern reachability?

**Thm.**

Model-checking Tree-LTL properties on (positive) DTPRS is undecidable (even without data). Becomes decidable for positive Tree-LTL (EX, EU, positive boolean combinations of DTPs).
Beyond pattern reachability?

**Tree-LTL** [Abiteboul et al.] LTL with DTPs as atomic props.

**Thm.**

Model-checking Tree-LTL properties on (positive) DTPRS is undecidable (even without data). Becomes decidable for positive Tree-LTL (EX, EU, positive boolean combinations of DTPs).

**Proof**

- Checking whether $T_1 \xrightarrow{*} T_2$ in a TPRS is undecidable (reduction from reachability in reset Petri nets).

- Tree-LTL formula $G(\gamma \rightarrow F\delta)$, initial tree and add rules:
Results (2): Bounded model-checking

Since complexity in general bad when using WSTS (e.g. non-elementary for TPRS) we consider bounded model-checking:

Bounded model-checking

Given DTPRS, DTP $P$ initial set $Init$ and bound $N$ (in unary) ask whether some data tree $T$ exists, matching $P$ and s.t. $T_0 \xrightarrow{\leq N} T$. 
Since complexity in general bad when using WSTS (e.g. non-elementary for TPRS) we consider **bounded model-checking**:

**Bounded model-checking**

Given DTPRS, DTP $P$ initial set $Init$ and bound $N$ (in unary) ask whether some data tree $T$ exists, matching $P$ and s.t. $T_0 \xrightarrow{\leq N} T$.

**Thm.**

Bounded model-checking for (not necessarily positive) DTPRS is $\text{NexpTime}$-complete.

**Proof**

- Upper bound: encode into recursion-free GAXML and use upper bound from [Abiteboul et al’08].
- Lower bound: adapt ideas from recursion-free GAXML lower bound. Main idea: create and check lists of length $2^n$. Encode lists with data, horizontally. Using transitive closure ($n$ queries) check length $2^n$. 
Conclusion, outlook

Results

- We identified **simple-path-boundedness** as key property for obtaining decidability of guarded tree rewriting with data. Positiveness is still needed, since we allow recursion.
- Decidability of reaching (a positive combination of) patterns, but undecidability of very simple Tree-LTL formulas (reason: negation).
- High complexity.

Outlook

- What happens when there is no assumption on fresh data? might be decidable without simple-path-boundedness condition.
- Other tree-like decompositions, compatible with rules?
- Fragments with reasonable complexity for BMC?
- Weaken recursion by putting identical, acyclic processes in parallel.
Conclusions and Outlook

Results

- We identified simple-path-boundedness as key property for obtaining decidability of guarded tree rewriting with data. Positiveness is still needed, since we allow recursion.
- Decidability of reaching (a positive combination of) patterns, but undecidability of very simple Tree-LTL formulas (reason: negation).
- High complexity.

Outlook

- What happens when there is no assumption on fresh data? might be decidable without simple-path-boundedness condition.
- Other tree-like decompositions, compatible with rules?
- Fragments with reasonable complexity for BMC?
- Weaken recursion by putting identical, acyclic processes in parallel.

Merci!