State-Boundedness in Data-Aware Dynamic Systems

Marco Montali

KRDB Research Centre for Knowledge and Data
Free University of Bozen-Bolzano

Joint work with: B. Bagheri Hariri, D. Calvanese, A. Deutsch

KR 2014
Data-Aware Dynamic System

A dynamic system that manipulate data over time.

Recall the VSL keynote by F. Baader.

- **Data layer**: maintains data of interest.
  - Relational database.
  - (Description logic) ontology.

- **Process layer**: evolves the extensional part of the data layer.
  - Control-flow component: determines when actions can be executed.
  - Actions: atomic units of work that update the data.
    - Interact with the external world to inject *fresh data* into the system.
Data-Centric Dynamic Systems (DCDSs)

- Data layer: relational database with FO constraints.
- Actions: specified by (parallel) effects that query the current state and determine the next state.
  - Service calls can be invoked to get new data.

Example

- Data layer: schema \{R/2, Q/1, S/1\}, no constraint.
- Process: \( \exists y. R(x, y) \mapsto t(x) \).
- Action: \( t(p) : \)
  \begin{align*}
  R(x, y) \land x \neq p & \mapsto R(x, y) \\
  R(p, y) & \mapsto R(p, f(y)) \\
  R(p, y) & \mapsto Q(p)
  \end{align*}
Data-Centric Dynamic Systems (DCDSs)

- Data layer: relational database with FO constraints.
- Actions: specified by (parallel) effects that query the current state and determine the next state.
  - Service calls can be invoked to get new data.

Example

- Data layer: schema \{R/2, Q/1, S/1\}, no constraint.
- Process: \( \exists y. R(x, y) \mapsto t(x) \).
- Action: \( t(p) : \begin{cases} R(x, y) \land x \neq p \mapsto R(x, y) \\ R(p, y) \mapsto R(p, f(y)) \\ R(p, y) \mapsto Q(p) \end{cases} \)

\( R(a,b), R(a,c), R(c,d), Q(d), S(b) \)
Data-Centric Dynamic Systems (DCDSs)

- Data layer: relational database with FO constraints.
- Actions: specified by (parallel) effects that query the current state and determine the next state.
  ▶ Service calls can be invoked to get new data.

**Example**

- Data layer: schema \{R/2, Q/1, S/1\}, no constraint.
- Process: \( \exists y. R(x, y) \mapsto t(x) \).
- Action: \( t(p) : \) 
  \[
  \begin{align*}
  R(x, y) \land x \neq p & \implies R(x, y) \\
  R(p, y) & \implies R(p, f(y)) \\
  R(p, y) & \implies Q(p)
  \end{align*}
  \]
Data-Centric Dynamic Systems (DCDSs)

- Data layer: relational database with FO constraints.
- Actions: specified by (parallel) effects that query the current state and determine the next state.
  - Service calls can be invoked to get new data.

Example

- Data layer: schema \{R/2, Q/1, S/1\}, no constraint.
- Process: \( \exists y. R(x, y) \mapsto t(x) \).
- Action: \( t(a) : \begin{cases} 
  R(x, y) \land x \neq a \mapsto R(x, y) \\
  R(a, y) \mapsto R(a, f(y)) \\
  R(a, y) \mapsto Q(a) 
\end{cases} \)

\[ R(a,b), R(a,c), R(c,d), Q(d), S(b) \]

\[ t(a) \]

\[ R(c,d), Q(a), R(a,f(b)), R(a,f(c)) \]
Data-Centric Dynamic Systems (DCDSs)

- Data layer: relational database with FO constraints.
- Actions: specified by (parallel) effects that query the current state and determine the next state.
  - Service calls can be invoked to get new data.

Example

- Data layer: schema \( \{R/2, Q/1, S/1\} \), no constraint.
- Process: \( \exists y. R(x, y) \rightarrow t(x) \).
- Action: \( t(a) : \)
  \[
  \begin{align*}
  &R(x, y) \land x \neq a \quad \leadsto \quad R(x, y) \\
  &R(a, y) \quad \leadsto \quad R(a, f(y)) \\
  &R(a, y) \quad \leadsto \quad Q(a)
  \end{align*}
  \]

\[
\begin{align*}
&\text{R(a,b), R(a,c),} \\
&\text{R(c,d),} \\
&\text{Q(d),} \\
&\text{S(b)} \\
&\text{t(a)} \\
&\text{R(c,d), Q(a),} \\
&\text{R(a,f(b)) R(a,f(c))} \\
&\text{f(b) = f(c) = a} \\
&\text{R(c,d), Q(a),} \\
&\text{R(a,a)}
\end{align*}
\]
Data-Centric Dynamic Systems (DCDSs)

- Data layer: relational database with FO constraints.
- Actions: specified by (parallel) effects that query the current state and determine the next state.
  - Service calls can be invoked to get new data.

Example

- Data layer: schema \( \{ R/2, Q/1, S/1 \} \), no constraint.
- Process: \( \exists y. R(x, y) \mapsto t(x) \).
- Action: \( t(a) : \begin{cases} 
  R(x, y) \land x \neq a \leadsto R(x, y) \\
  R(a, y) \leadsto R(a, f(y)) \\
  R(a, y) \leadsto Q(a)
\end{cases} \)
Data-Centric Dynamic Systems (DCDSs)

- Data layer: relational database with FO constraints.
- Actions: specified by (parallel) effects that query the current state and determine the next state.
  - Service calls can be invoked to get new data.

Example

- Data layer: schema \( \{ R/2, Q/1, S/1 \} \), no constraint.
- Process: \( \exists y. R(x, y) \mapsto t(x) \).
- Action: \( t(a) : \begin{cases} 
  R(x, y) \land x \neq a & \Rightarrow R(x, y) \\
  R(a, y) & \Rightarrow R(a, f(y)) \\
  R(a, y) & \Rightarrow Q(a) 
\end{cases} \)

\[
R(a,b), R(a,c), \\
R(c,d), \\
Q(d), \\
S(b)
\]

\[
R(c,d), Q(a), \\
R(a,f(b)), R(a,f(c))
\]
Verification

Execution semantics: a relational transition system, possibly with infinitely many states.

Verification Problem

Check whether the dynamic system guarantees a desired property, expressed in some variant of a first-order temporal logic.
Verification

Execution semantics: a relational transition system, possibly with infinitely many states.

Verification Problem

Check whether the dynamic system guarantees a desired property, expressed in some variant of a first-order temporal logic.

Undecidable in general!
(Even for propositional temporal logics)
State-boundedness

State-Bounded System
Has an overall bound on the number of individuals stored in each single state of the system.

Structurally State-Bounded
State-Bounded for any given initial database.

Decidability under state-boundedness
Shown in a plethora of recent works, for a variety of dynamic systems:

• Artifact-centric MASs, and FO-CTLK [BelardinelliEtAl-KR12].
• DCDSs and $\mu$-Lp.
• DL-based Dynamic Systems, and $\mu$-LECQ [CalvaneseEtAl-RR13].
• Data-aware MASs with commitments, and $\mu$-LECQ [MontaliEtAl-AAMAS14].
• Bounded situation calculus action theories, and $\mu$-L [DeGiacomoEtAl-KR12].
State-boundedness

State-Bounded System
Has an overall bound on the number of individuals stored in each single state of the system.

Structurally State-Bounded
State-Bounded for any given initial database.

Decidability under state-boundedness
Shown in a plethora of recent works, for a variety of dynamic systems:

- Artifact-centric MASs, and FO-CTLK [BelardinelliEtAl-KR12].
- DCDSs and $\mu\mathcal{L}_p$.
- DL-based Dynamic Systems, and $\mu\mathcal{L}_p^{ECQ}$ [CalvaneseEtAl-RR13].
- Data-aware MASs with commitments, and $\mu\mathcal{L}_p^{ECQ}$ [MontaliEtAl-AAMAS14].
- Bounded situation calculus action theories, and $\mu\mathcal{L}$ [DeGiacomoEtAl-KR12].
State-boundedness is a semantic property
Typically, assumed to hold.

Theorem ([BagheriHaririEtAl-PODS13])
Checking whether a DCDS is state-bounded is **undecidable**.

.study of sufficient, syntactic conditions guaranteeing state-boundedness.
State-boundedness is a semantic property
Typically, assumed to hold.

**Theorem ([BagheriHaririEtAl-PODS13])**

*Checking whether a DCDS is state-bounded is undecidable.*

⇝ study of sufficient, syntactic conditions guaranteeing state-boundedness.

**Central question**

Do there exist significant classes of data-aware dynamic systems for which checking state-boundedness is decidable?
State-boundedness is a semantic property. Typically, assumed to hold.

**Theorem ([BagheriHaririEtAl-PODS13])**

*Checking whether a DCDS is state-bounded is undecidable.*

study of sufficient, syntactic conditions guaranteeing state-boundedness.

**Central question**

*Do there exist significant classes of data-aware dynamic systems for which checking state-boundedness is decidable?*

A similar question has been extensively studied in a different setting . . .
P/T nets

- Extensively used for modelling concurrent systems:
  ▶ Distributed systems, workflows, business processes, ...
- Study of several formal properties: reachability, deadlock freedom, boundedness.

![Diagram of P/T nets with nodes and transitions]
P/T nets - The Good

- Extensively used for modelling **concurrent systems**:  
  - Distributed systems, workflows, business processes, ... 
- Study of several formal properties: reachability, deadlock freedom, **boundedness**.

```
  a  b  c  d  e  f  g
  |   |   |   |   |   |   |
  ▼   ▼   ▼   ▼   ▼   ▼   ▼
```

Marco Montali (unibz)
P/T nets - The Good

- Extensively used for modelling **concurrent systems**: Distributed systems, workflows, business processes, …
- Study of several formal properties: reachability, deadlock freedom, **boundedness**.
P/T nets - The Good

- Extensively used for modelling concurrent systems:
  - Distributed systems, workflows, business processes, ...
- Study of several formal properties: reachability, deadlock freedom, boundedness.
P/T nets - The Good

- Extensively used for modelling concurrent systems:
  - Distributed systems, workflows, business processes, …
- Study of several formal properties: reachability, deadlock freedom, boundedness.
P/T nets - The Good

- Extensively used for modelling concurrent systems:
  - Distributed systems, workflows, business processes, ...
- Study of several formal properties: reachability, deadlock freedom, *boundedness*.
P/T nets - The Good

- Extensively used for modelling **concurrent systems**:
  - Distributed systems, workflows, business processes, ... 
- Study of several formal properties: reachability, deadlock freedom, **boundedness**.

![Petri Net Diagram]
P/T nets - The Good

- Extensively used for modelling concurrent systems:
  - Distributed systems, workflows, business processes, ...
- Study of several formal properties: reachability, deadlock freedom, boundedness.
Not all marked Petri nets are bounded.
P/T Nets - The Bad

Not all marked Petri nets are bounded.
Not all marked Petri nets are bounded.
Not all marked Petri nets are bounded.
Not all marked Petri nets are bounded.
Not all marked Petri nets are bounded.
P/T Nets - The Bad

Not all marked Petri nets are bounded.

\[ a \rightarrow b \rightarrow c \rightarrow a \]
P/T Nets - The Bad

Not all marked Petri nets are bounded.
Forms of Boundedness

**Boundedness**

A **marked Petri net** is bounded if all executions starting from the given marking do not produce an unbounded amount of tokens.

**Structural boundedness**

A **Petri net** is structurally bounded if for all possible initial markings the resulting marked net is bounded.
**Forms of Boundedness**

**Boundedness**
A *marked Petri net* is bounded if all executions starting from the given marking do not produce an unbounded amount of tokens.

**Structural boundedness**
A *Petri net* is structurally bounded if for all possible initial markings the resulting marked net is bounded.
Reset Transfer Nets - The Ugly

\[ \text{P/T nets} \subset \text{RT nets} \subset \text{Reset Post G-nets [DufourEtAllICALP98]} \]

When \( t \) fires, all tokens in \( P \) are removed.

When \( t \) fires, all tokens in \( P \) are transferred to \( Q \).
Boundedness Spectrum

boundedness

RT nets

R nets

T nets

P/T nets

structural boundedness

RT nets

T nets

R nets

P/T nets
Boundedness Spectrum

boundedness

RT nets

R nets

T nets

P/T nets

structural boundedness

RT nets

T nets

R nets

P/T nets

Undec.
Boundedness Spectrum

- **boundedness**
  - RT nets
  - R nets
  - T nets
  - P/T nets

- **structural boundedness**
  - RT nets
  - T nets
  - R nets
  - P/T nets

- **Undec.**
- **Dec.**
- **ExpSpace**
- **PTime**
Understanding State-Boundedness

Goal
Devise a connection between RT nets and DCDSs so as to understand the state-boundedness spectrum in data-aware dynamic systems.

Main issue: set vs bag semantics.
Understanding State-Boundedness

Goal
Devise a connection between RT nets and DCDSs so as to understand the state-boundedness spectrum in data-aware dynamic systems.

Main issue: set vs bag semantics.
Understanding State-Boundedness

Goal

Devise a connection between RT nets and DCDSs so as to understand the state-boundedness spectrum in data-aware dynamic systems.

Main issue: set vs bag semantics.
Idea

Tokens as distinct identifiers distributed over place relations. Only cardinalities matter, not the data values.

Data layer.

- Unary relations for places: \( \{ P_i/1 \mid i \in \{1, \ldots, 4\} \} \)
- No constraints.

Process layer: each transition becomes an action + condition-action rule.

- Condition: gets tokens from input places; feeds the action with them.
- Action: moves tokens according to the firing semantics of the net.
  - Service calls to generate identifiers for new tokens.
From RT Nets to DCDSs - The P/T Case

\[ P_0(x_1) \leftrightarrow a(x_1) \]
From RT Nets to DCDSs - The P/T Case

\[ P_0(x_1) \mapsto a(x_1) \]

\[
\begin{cases}
  P_0 & P_0(y) \land y \neq x_1 \leadsto P_0(f(y)) \\
  P_1 & \\
  P_2 & \\
  P_3 & \\
  P_4 & 
\end{cases}
\]

Marco Montali (unibz)
From RT Nets to DCDSs - The P/T Case

\[ P_0(x_1) \mapsto a(x_1) \]

\[
\begin{align*}
P_0 & : P_0(y) \land y \neq x_1 \leadsto P_0(f(y)) \\
P_1 & : P_1(y) \leadsto P_1(h_1(y)) \\
true & \leadsto P_1(g_1()) \\
P_2 & : P_2(y) \leadsto P_2(h_2(y)) \\
true & \leadsto P_2(g_2()) \\
P_3 & \\
P_4 &
\end{align*}
\]
From RT Nets to DCDSs - The P/T Case

\[ P_0(x_1) \mapsto a(x_1) \]

\[
\begin{align*}
\text{a}(x_1) : & \\
\text{P}_0 & P_0(y) \land y \neq x_1 \quad \leadsto \quad P_0(f(y)) \\
\text{P}_1 & P_1(y) \quad \leadsto \quad P_1(h_1(y)) \\
& \text{true} \quad \leadsto \quad P_1(g_1()) \\
\text{P}_2 & P_2(y) \quad \leadsto \quad P_2(h_2(y)) \\
& \text{true} \quad \leadsto \quad P_2(g_2()) \\
\text{P}_3 & P_3(y) \quad \leadsto \quad P_3(h_3(y)) \\
\text{P}_4 & P_4(y) \quad \leadsto \quad P_4(h_4(y))
\end{align*}
\]
From RT Nets to DCDSs - The Reset Case

\[ P_1(x_1) \land P_3(x_2) \rightarrow b(x_1, x_2) \]
From RT Nets to DCDSs - The Reset Case

\[ P_1(x_1) \land P_3(x_2) \mapsto b(x_1, x_2) \]

\[ b(x_1, x_2) : \begin{cases} 
P_0 & P_0(y) \\
& \sim P_0(h_0(y)) \\
P_1 \\
P_2 \\
P_3 \\
P_4 
\end{cases} \]
From RT Nets to DCDSs - The Reset Case

\[ P_1(x_1) \land P_3(x_2) \mapsto b(x_1, x_2) \]

\[ b(x_1, x_2) : \begin{cases} 
P_0 & P_0(y) \leadsto P_0(h_0(y)) \\
P_1 & P_1(y) \land y \neq x_1 \leadsto P_1(h_1(y)) \\
P_2 & \\
P_3 & P_3(y) \land y \neq x_2 \leadsto P_3(h_3(y)) \\
P_4 & 
\end{cases} \]
From RT Nets to DCDSs - The Reset Case

\[ P_1(x_1) \land P_3(x_2) \mapsto b(x_1, x_2) \]

\[
\begin{align*}
\begin{array}{l}
P_0 & P_0(y) \mapsto P_0(h_0(y)) \\
P_1 & P_1(y) \land y \neq x_1 \mapsto P_1(h_1(y)) \\
P_2 & - \\
P_3 & P_3(y) \land y \neq x_2 \mapsto P_3(h_3(y)) \\
P_4 & \\
\end{array}
\end{align*}
\]

\[ b(x_1, x_2) : \]

\[ \begin{cases} 
P_0 & P_0(y) \mapsto P_0(h_0(y)) \\
P_1 & P_1(y) \land y \neq x_1 \mapsto P_1(h_1(y)) \\
P_2 & - \\
P_3 & P_3(y) \land y \neq x_2 \mapsto P_3(h_3(y)) \\
P_4 & \\
\end{cases} \]
From RT Nets to DCDSs - The Reset Case

\[ P_1(x_1) \land P_3(x_2) \mapsto b(x_1, x_2) \]

**b(x_1, x_2):**

\[
\begin{align*}
\text{P}_0 & : P_0(y) \mapsto P_0(h_0(y)) \\
\text{P}_1 & : P_1(y) \land y \neq x_1 \mapsto P_1(h_1(y)) \\
\text{P}_2 & : - \\
\text{P}_3 & : P_3(y) \land y \neq x_2 \mapsto P_3(h_3(y)) \\
\text{P}_4 & : P_4(y) \mapsto P_4(h_4(y)) \text{ true} \mapsto P_4(g_4())
\end{align*}
\]
From RT Nets to DCDSs - The Reset Case

\[ P_1(x_1) \land P_3(x_2) \mapsto b(x_1, x_2) \]

\[
\begin{align*}
\text{b}(x_1, x_2) \colon & \\
\text{P}_0 & \text{P}_0(y) & \mapsto & \text{P}_0(h_0(y)) \\
\text{P}_1 & \text{P}_1(y) \land y \neq x_1 & \mapsto & \text{P}_1(h_1(y)) \\
\text{P}_2 & \text{true} & \mapsto & \text{P}_4(g_4()) \\
\text{P}_3 & \text{P}_3(y) \land y \neq x_2 & \mapsto & \text{P}_3(h_3(y)) \\
\text{P}_4 & \text{P}_4(y) & \mapsto & \text{P}_4(h_4(y))
\end{align*}
\]

A relation \((P_2)\) does not appear in the action!
From RT Nets to DCDSs - The Transfer Case

\[ P_1(x_1) \mapsto c(x_1) \]
From RT Nets to DCDSs - The Transfer Case

There is an effect involving two different relations ($P_2$ and $P_4$)!

\[
P_1(x_1) \leftrightarrow c(x_1)
\]

\[
c(x_1) : \begin{cases} 
P_0 & P_0(y) \leadsto P_0(h_0(y)) \\
P_1 & \\
P_2 & \\
P_3 & P_3(y) \leadsto P_3(h_3(y)) \\
P_4 & 
\end{cases}
\]
From RT Nets to DCDSs - The Transfer Case

\[ P_1(x_1) \leftrightarrow c(x_1) \]

\[
\begin{align*}
\text{c}(x_1) : & \quad \left\{ \begin{array}{l}
\text{P}_0 \quad P_0(y) \quad \sim \quad P_0(h_0(y)) \\
\text{P}_1 \quad P_1(y) \land y \neq x_1 \quad \sim \quad P_1(h_1(y)) \\
\text{P}_2 \quad \text{P}_3(y) \quad \sim \quad P_3(h_3(y)) \\
\text{P}_4 \quad \text{P}_4(y) \quad \sim \quad P_4(h_4(y))
\end{array} \right. 
\end{align*}
\]
From RT Nets to DCDSs - The Transfer Case

There is an effect involving two different relations ($P_2$ and $P_4$)!

$$P_1(x_1) \leftrightarrow c(x_1)$$

$c(x_1) : \begin{cases}
P_0 & P_0(y) \leadsto P_0(h_0(y)) \\
P_1 & P_1(y) \land y \neq x_1 \leadsto P_1(h_1(y)) \\
P_2 & P_2(y) \leadsto P_4(h_2(y)) \\
P_3 & P_3(y) \leadsto P_3(h_3(y)) \\
P_4 &
\end{cases}$
From RT Nets to DCDSs - The Transfer Case

\[ P_1(x_1) \leftrightarrow c(x_1) \]

\[
\begin{align*}
\mathbf{c}(x_1) : & \\
\mathbf{P}_0 & \quad P_0(y) & \sim & P_0(h_0(y)) \\
\mathbf{P}_1 & \quad P_1(y) \land y \neq x_1 & \sim & P_1(h_1(y)) \\
\mathbf{P}_2 & \quad P_2(y) & \sim & P_4(h_2(y)) \\
\mathbf{P}_3 & \quad P_3(y) & \sim & P_3(h_3(y)) \\
\mathbf{P}_4 & \quad P_4(y) & \sim & P_4(h_4(y))
\end{align*}
\]
From RT Nets to DCDSs - The Transfer Case

\[ P_1(x_1) \mapsto c(x_1) \]

- \( P_0 \): \( P_0(y) \) \( \mapsto \) \( P_0(h_0(y)) \)
- \( P_1 \): \( P_1(y) \land y \neq x_1 \) \( \mapsto \) \( P_1(h_1(y)) \)
- \( P_2 \): \( P_2(y) \) \( \mapsto \) \( P_4(h_2(y)) \)
- \( P_3 \): \( P_3(y) \) \( \mapsto \) \( P_3(h_3(y)) \)
- \( P_4 \): \( P_4(y) \) \( \mapsto \) \( P_4(h_4(y)) \)

There is an effect involving two different relations \( (P_2 \text{ and } P_4) \)!
Is the Translation Correct?

**NO!** The resulting DCDS has a *lossy behavior*.
Is the Translation Correct?

**NO!** The resulting DCDS has a **lossy behavior**.

**Petri net**

\[ P_0 \xrightarrow{t} t \xrightarrow{t} P_0 \]

**DCDS**

\[ t(\ldots) : \{ \ldots, P_0(y) \leadsto P_0(h_0(y)), \ldots \} \]

\[ h_0(a) = a, \ h_0(b) = b, \ h_0(c) = c \]

\[ P_0 \xrightarrow{t} t \xrightarrow{t} P_0 \]

\[ h_0(a) = a, \ h_0(b) = a, \ h_0(c) = a \]

\[ P_0 \xrightarrow{t} t \xrightarrow{t} P_0 \]

\[ h_0(a) = a, \ h_0(b) = a, \ h_0(c) = c \]

\[ P_0 \xrightarrow{t} t \xrightarrow{t} P_0 \]

\[ \ldots \]
Is the Translation Correct?

However...

The resulting DCDS reproduces all behaviors of the net (and more).
Is the Translation Correct?

However...

The resulting DCDS reproduces all behaviors of the net (and more).

Theorem

An RT net is (structurally) bounded if and only if the corresponding DCDS is (structurally) state-bounded.
Data Layer

Schema $\mathcal{R}$ with unary relations only, and no constraint.

Process

Only one rule per action, of the form $Q(\overline{x}) \mapsto \alpha(\overline{x})$, where

$$Q(x_1, \ldots, x_n) = \bigwedge_{i \in \{1, \ldots, n\}, \ P_i \neq P_j \text{ for } i \neq j} P_i(x_i)$$

Shape of action $\alpha(\overline{x})$

For each $P_i \in \text{RELS}(Q)$, $\alpha$ must contain:

- $P_i(y) \land y \neq x_i \leadsto P_i(f_i(y))$ and may contain:
  - true $\leadsto P_i(g_i())$

For each $P_l \in \mathcal{R} \setminus \text{RELS}(Q)$, $\alpha$ may contain:

- $P_l(y) \leadsto P_l(h_l(y))$
- either true $\leadsto P_j(g_j())$, or $P_j'(y) \leadsto P_j(h_j(y))$
  for some $P_j' \in \mathcal{R} \setminus (\text{RELS}(Q) \cup P_j)$. 
Consider schema $\mathcal{R} = \{P_0, P_1, P_2, P_3, P_4\}$, and action $t$ with:

- process condition-action rule $P_0(x_0) \land P_1(x_1) \mapsto t(x_0, x_1)$
  
  \[
  \left\{
  \begin{array}{ll}
  P_0(y) \land y \neq x_0 & \mapsto P_0(f_0(y)) \\
  P_1(y) \land y \neq x_1 & \mapsto P_1(f_1(y)) \\
  true & \mapsto P_1(g_1()) \\
  true & \mapsto P_2(g_2()) \\
  P_3(y) & \mapsto P_4(h_3(y)) \\
  P_4(y) & \mapsto P_4(h_4(y))
  \end{array}
  \right.
  \]

- action $t(x_0, x_1)$:
Consider schema $\mathcal{R} = \{P_0, P_1, P_2, P_3, P_4\}$, and action $t$ with:

- process condition-action rule $P_0(x_0) \land P_1(x_1) \mapsto t(x_0, x_1)$

$$
\begin{cases}
P_0(y) \land y \neq x_0 \leadsto P_0(f_0(y)) \\
P_1(y) \land y \neq x_1 \leadsto P_1(f_1(y)) \\
true \leadsto P_1(g_1()) \\
true \leadsto P_2(g_2()) \\
P_3(y) \leadsto P_4(h_3(y)) \\
P_4(y) \leadsto P_4(h_4(y))
\end{cases}
$$

- action $t(x_0, x_1)$:

```plaintext
\begin{align*}
P_0 & \quad \text{false} \quad \text{false} \quad \text{false} \quad \text{false} \\
P_1 & \quad \text{false} \quad \text{false} \quad \text{false} \quad \text{false} \\
P_2 & \quad \text{false} \quad \text{false} \quad \text{false} \quad \text{false} \\
P_3 & \quad \text{false} \quad \text{false} \quad \text{false} \quad \text{false} \\
P_4 & \quad \text{false} \quad \text{false} \quad \text{false} \quad \text{false}
\end{align*}
```
Consider schema \( R = \{ P_0, P_1, P_2, P_3, P_4 \} \), and action \( t \) with:

- process condition-action rule \( P_0(x_0) \land P_1(x_1) \rightarrow t(x_0, x_1) \)

\[
\begin{cases}
   P_0(y) \land y \neq x_0 \quad \leadsto \quad P_0(f_0(y)) \\
   P_1(y) \land y \neq x_1 \quad \leadsto \quad P_1(f_1(y)) \\
   \text{true} \quad \leadsto \quad P_1(g_1()) \\
   \text{true} \quad \leadsto \quad P_2(g_2()) \\
   P_3(y) \quad \leadsto \quad P_4(h_3(y)) \\
   P_4(y) \quad \leadsto \quad P_4(h_4(y))
\end{cases}
\]

- action \( t(x_0, x_1) \):

\[ P_0 \quad P_1 \quad t \quad P_2 \quad P_3 \quad P_4 \]
Consider schema $\mathcal{R} = \{P_0, P_1, P_2, P_3, P_4\}$, and action $t$ with:

- process condition-action rule $P_0(x_0) \land P_1(x_1) \mapsto t(x_0, x_1)$

- action $t(x_0, x_1)$:

$$
\begin{align*}
P_0(y) &\land y \neq x_0 \quad \Rightarrow \quad P_0(f_0(y)) \\
\quad &\quad \quad \downarrow \downarrow \\
P_1(y) &\land y \neq x_1 \quad \Rightarrow \quad P_1(f_1(y)) \\
\quad &\quad \quad \downarrow \downarrow \\
\text{true} &\quad \Rightarrow \quad P_1(g_1()) \\
\quad &\quad \quad \downarrow \downarrow \\
\quad &\quad \quad \downarrow \downarrow \\
P_3(y) &\quad \Rightarrow \quad P_4(h_3(y)) \\
P_4(y) &\quad \Rightarrow \quad P_4(h_4(y))
\end{align*}
$$

Theorem
An LRT DCDS is (structurally) state-bounded if and only if the corresponding RT net is (structurally) bounded.
From LRT DCDSs to RT Nets

Consider schema \( \mathcal{R} = \{P_0, P_1, P_2, P_3, P_4\} \), and action \( t \) with:

- process condition-action rule \( P_0(x_0) \land P_1(x_1) \mapsto t(x_0, x_1) \)

\[
\begin{align*}
P_0(y) \land y \neq x_0 & \mapsto P_0(f_0(y)) \\
P_1(y) \land y \neq x_1 & \mapsto P_1(f_1(y)) \\
\text{true} & \mapsto P_1(g_1()) \\
\text{true} & \mapsto P_2(g_2()) \\
P_3(y) & \mapsto P_4(h_3(y)) \\
P_4(y) & \mapsto P_4(h_4(y))
\end{align*}
\]

- action \( t(x_0, x_1) \):

Theorem
An LRT DCDS is (structurally) state-bounded if and only if the corresponding RT net is (structurally) bounded.
From LRT DCDSs to RT Nets

Consider schema $\mathcal{R} = \{P_0, P_1, P_2, P_3, P_4\}$, and action $t$ with:

- process condition-action rule $P_0(x_0) \land P_1(x_1) \leadsto t(x_0, x_1)$

  \[
  \begin{cases}
  P_0(y) \land y \neq x_0 & \leadsto P_0(f_0(y)) \\
  P_1(y) \land y \neq x_1 & \leadsto P_1(f_1(y)) \\
  \text{true} & \leadsto P_1(g_1()) \\
  \text{true} & \leadsto P_2(g_2()) \\
  P_3(y) & \leadsto P_4(h_3(y)) \\
  P_4(y) & \leadsto P_4(h_4(y))
  \end{cases}
  \]

- action $t(x_0, x_1)$:
Consider schema $\mathcal{R} = \{P_0, P_1, P_2, P_3, P_4\}$, and action $t$ with:

- process condition-action rule $P_0(x_0) \land P_1(x_1) \leftrightarrow t(x_0, x_1)$

\[
\begin{align*}
P_0(y) \land y \neq x_0 & \rightarrow P_0(f_0(y)) \\
P_1(y) \land y \neq x_1 & \rightarrow P_1(f_1(y)) \\
\text{true} & \rightarrow P_1(g_1()) \\
\text{true} & \rightarrow P_2(g_2()) \\
P_3(y) & \rightarrow P_4(h_3(y)) \\
P_4(y) & \rightarrow P_4(h_4(y))
\end{align*}
\]

- action $t(x_0, x_1)$:

Theorem

An LRT DCDS is (structurally) state-bounded if and only if the corresponding RT net is (structurally) bounded.
State-Boundedness Spectrum

- **Undec.**
  - LRT DCDSs
  - LT DCDSs
  - LP DCDSs

- **Dec.**
  - LR DCDSs
  - LT DCDSs
  - LR DCDSs

- **ExpSpace**
  - LP DCDSs

- **PTime**
  - LP DCDSs
Take Home Message

LRT DCDSs are weak:

- Only unary relations.
- Only conjunctions without joins in conditions.
- Only atomic queries inside effects (possibly with a value inequality).
- Very limited use of negation (inequalities).
- No direct transfer of values from one state to the other.
Take Home Message

LRT DCDSs are weak:

- Only unary relations.
- Only conjunctions without joins in conditions.
- Only atomic queries inside effects (possibly with a value inequality).
- Very limited use of negation (inequalities).
- No direct transfer of values from one state to the other.

Still, to ensure that (structural) state-boundedness is decidable . . .

**Boundedness**

All relations must appear on the left-hand side of action effects, i.e., contribute to form the new state.

**Structural Boundedness**

Each action must be such that only a fixed amount of tuples is added to/removed from each relation in the schema.
Central question

Do there exist significant classes of data-aware dynamic systems for which checking state-boundedness is decidable?

Answer

NO

Hence, it becomes important to provide significant sufficient, checkable syntactic conditions that guarantee structural state-boundedness.
Do there exist significant classes of data-aware dynamic systems for which checking state-boundedness is decidable?

Answer

NO

Hence, it becomes important to provide significant sufficient, checkable syntactic conditions that guarantee structural state-boundedness. We follow this line, focusing on DCDSs and starting from [BagheriHaririEtAl-PODS13].
Consider a DCDS with process \( \{ \text{true} \mapsto \alpha() \} \), and

\[
\alpha() : \begin{cases} 
P(x) \leadsto P(x) \\
P(x) \leadsto Q(f(x)) \\
Q(x) \leadsto Q(x)
\end{cases}
\]
Example
Consider a DCDS with process \( \{ \text{true} \mapsto \alpha() \} \), and
\[
\alpha() : \begin{cases}
    P(x) \leadsto P(x) \\
    P(x) \leadsto Q(f(x)) \\
    Q(x) \leadsto Q(x)
\end{cases}
\]

We approximate the DCDS data-flow through a dependency graph.

The system is not state-bounded, due to:
- a generate cycle that continuously feeds a path issuing service calls;
- a recall cycle that accumulates the obtained results.
- (+ the fact that both cycles are simultaneously active)

**GR-acyclicity** detects exactly these undesired situations.
Our Contribution

\[ GR^+ \quad \text{in} \quad \text{CoNP} \quad \text{in} \quad \Sigma_2^p \]
Our Contribution

- $GR^+$
- $GR$
- $GR^{++}$
- Safe-$GR^{++}$
- Safe-$∃GR^{++}$
- Stratified-$∃GR^{++}$
- Stratified-$∃GR^{++}$

in $\Sigma_2^p$
## Conclusion

1. No significant decidable classes of data-aware dynamic systems for which state-boundedness is decidable.

2. It becomes crucial to provide checkable, sufficient conditions.
   - We have built on results on chase termination for tuple-generating dependencies, providing a family of conditions for DCDSs.

### Ongoing and future work

- Refine the syntactic conditions to handle if-then-else effects.
- Follow a different approach: provide modelling guidelines towards systems that are structurally state bounded by design.
  - Preliminary results in [SolomakhinEtAl-ICSOC13].