Using Logic to Escape the Jungle of Data-aware Process Verification

Sarah Winkler
Free University of Bozen-Bolzano, Italy

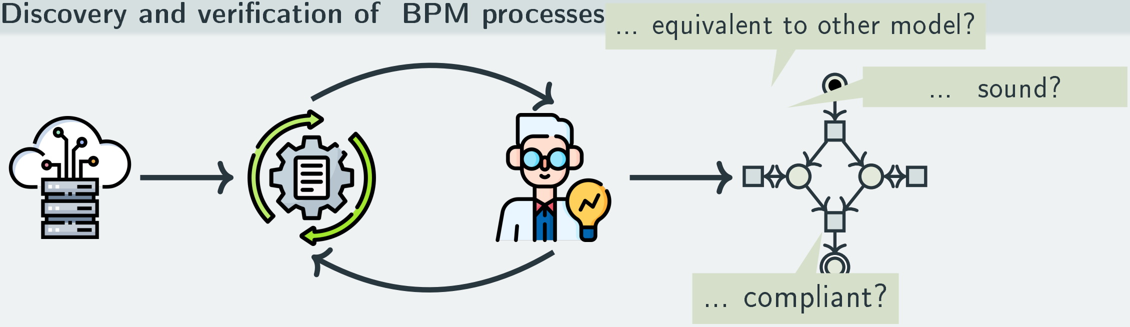
seminar @ DTU Compute, 7.9.2023

Motivation

Discovery and verification of BPM processes

Motivation

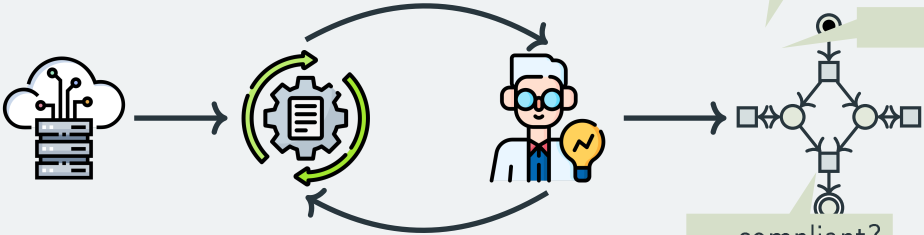
Discovery and verification of BPM processes



e.g. every order is eventually shipped

Motivation

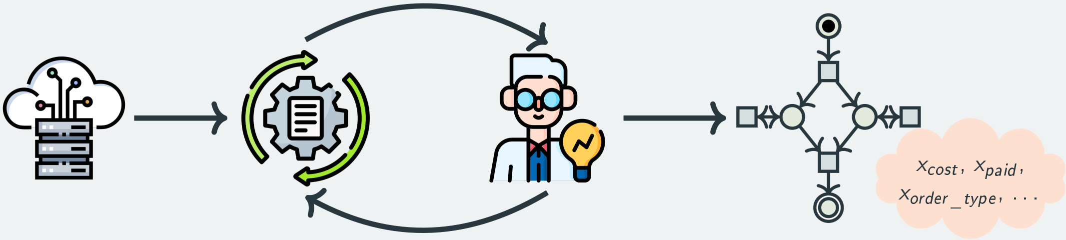
Discovery and verification of BPM processes



in Petri nets for typical BPM processes, verification tasks can be effectively decided

Motivation

Discovery and verification of data-aware BPM processes

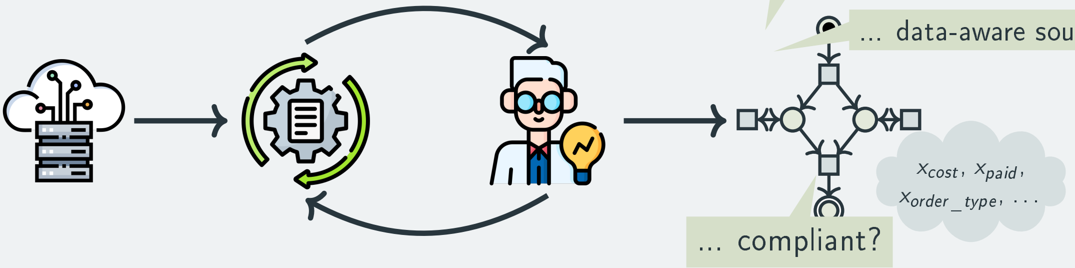


Assumption

data is represented by numeric variables, can be read and written by transitions

Motivation

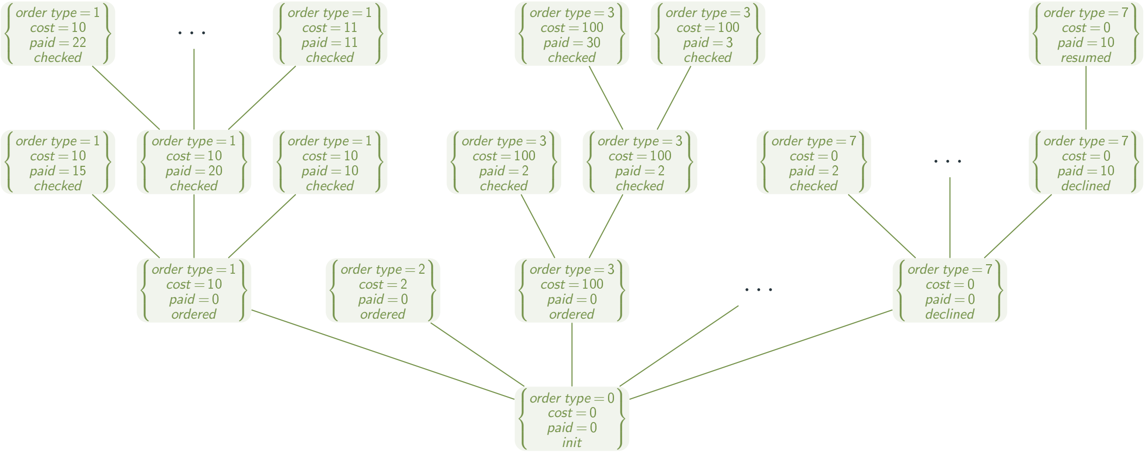
Discovery and verification of data-aware BPM



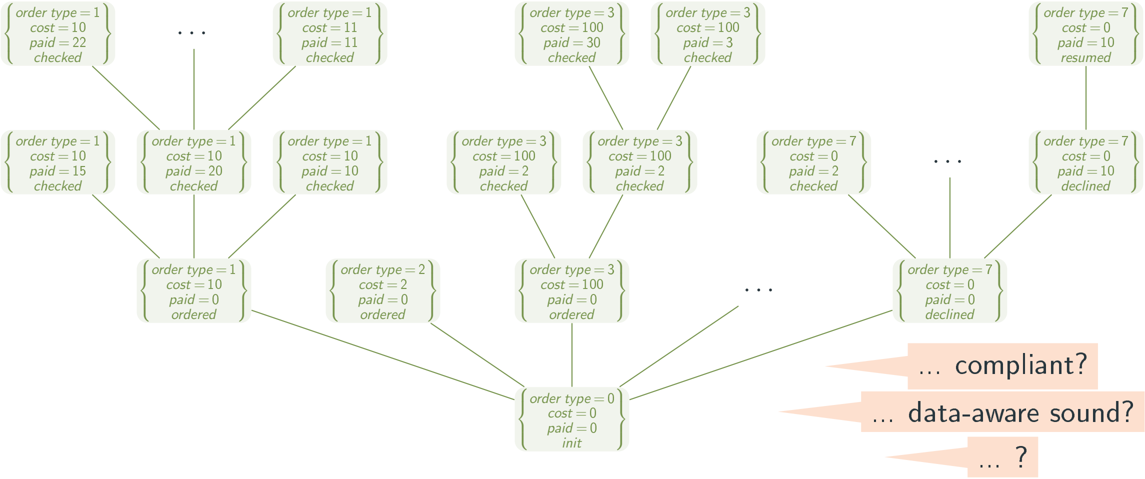
Assumption

data is represented by numeric variables, can be read and written by transitions

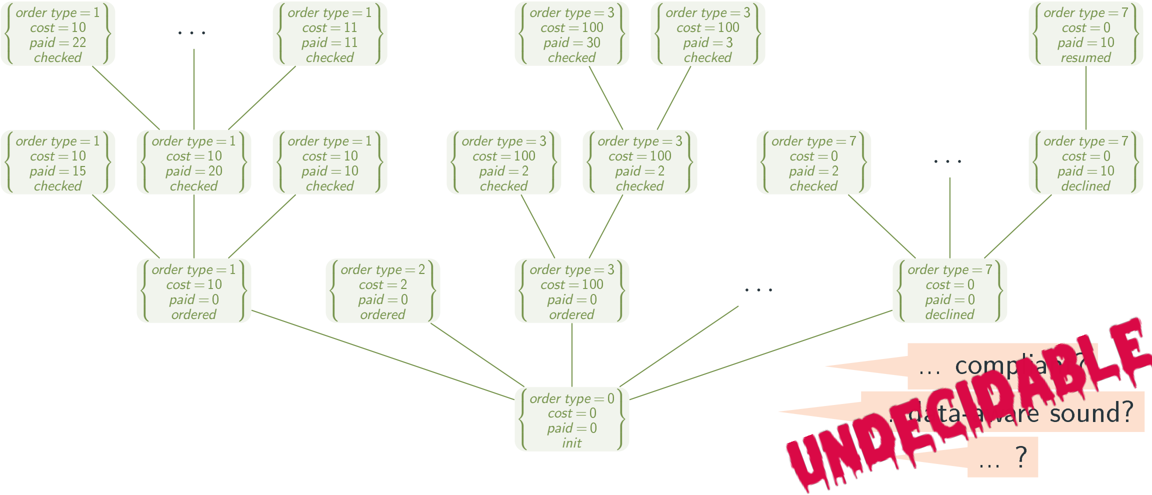
The Infinite State Space Jungle



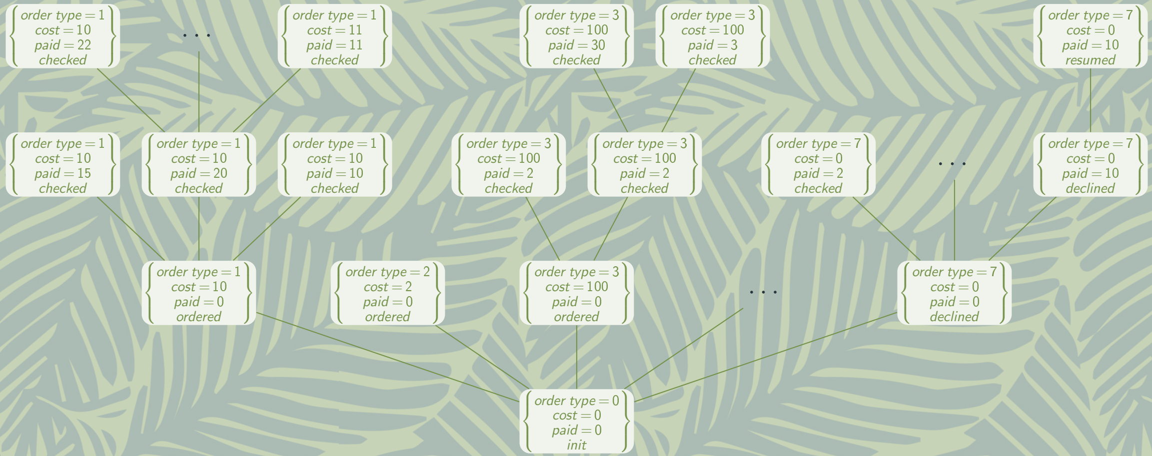
The Infinite State Space Jungle



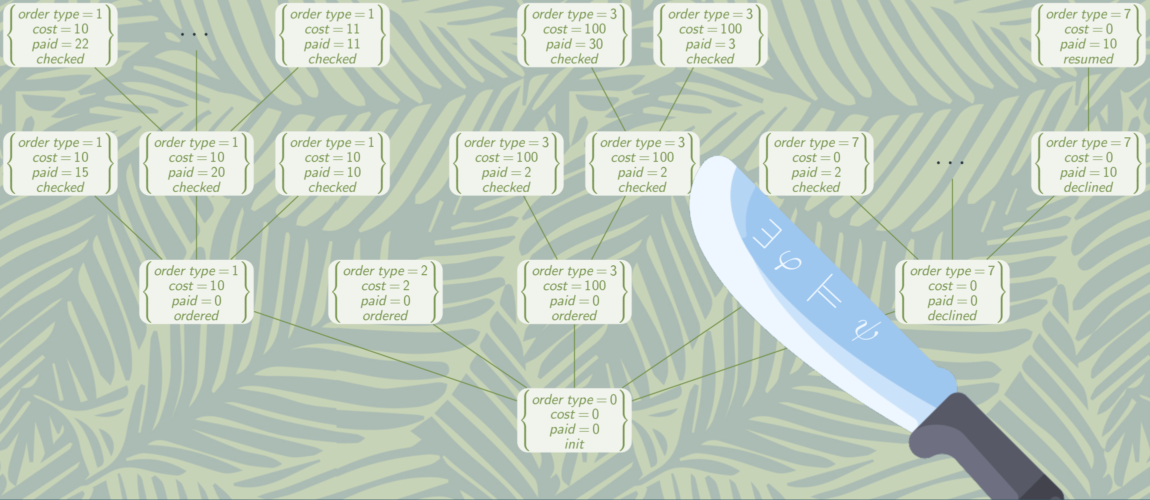
The Infinite State Space Jungle



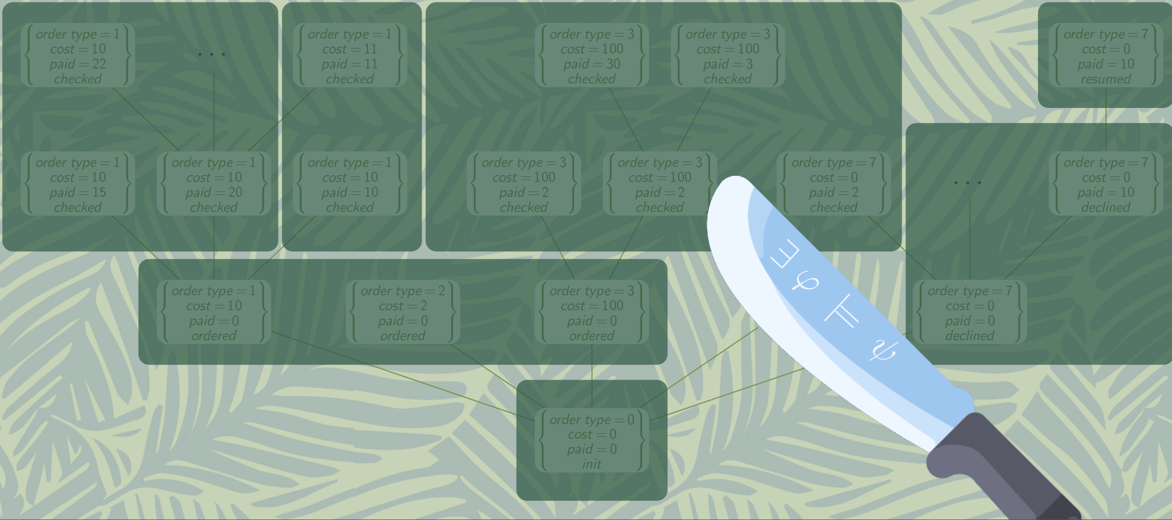
The Infinite State Space Jungle



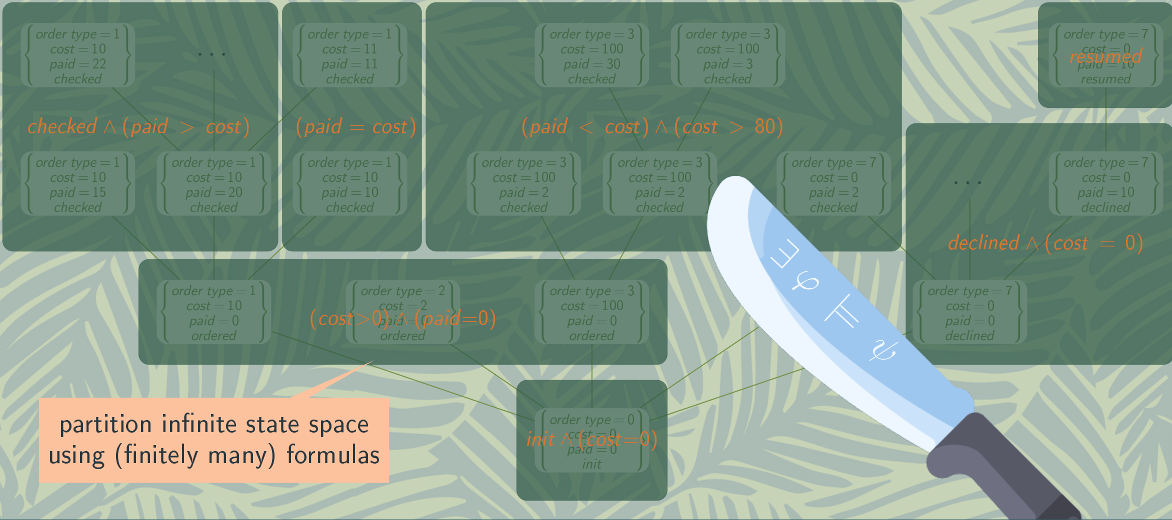
The Infinite State Space Jungle



The Infinite State Space Jungle



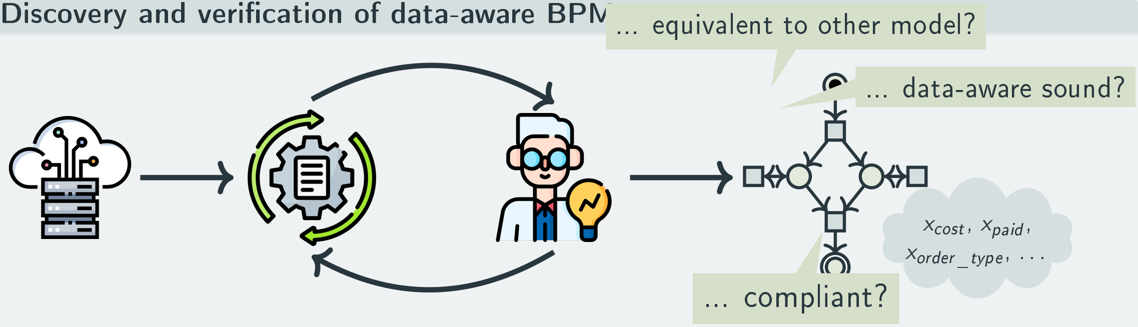
The Infinite State Space Jungle



partition infinite state space using (finitely many) formulas

Motivation

Discovery and verification of data-aware BPM



This talk

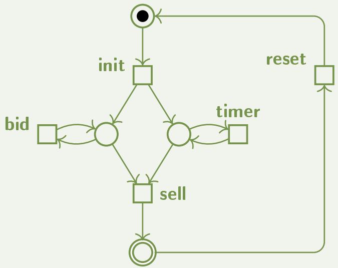
identify classes of data-aware models where verification tasks are decidable

Outline

Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F

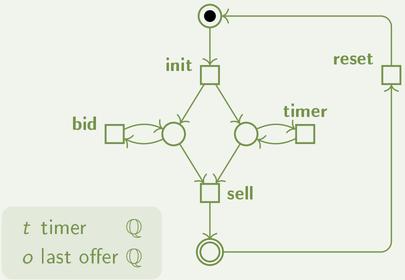
Example (Auction model)



Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F
- data: set of “global” variables V with numeric domain ($\mathbb{Q}$ or $\mathbb{Z}$)

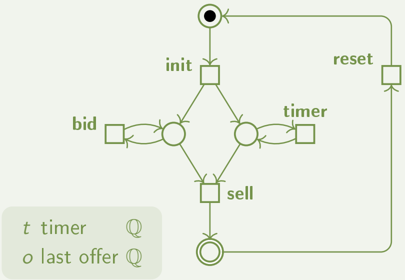
Example (Auction model)



Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F
- data: set of "global" variables V
  with numeric domain ($\mathbb{Q}$ or $\mathbb{Z}$)
- initial values of V are fixed by valuation α_0

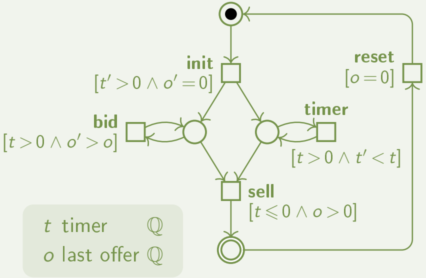
Example (Auction model)



Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F
- data: set of "global" variables V with numeric domain ($\mathbb{Q}$ or $\mathbb{Z}$)
- initial values of V are fixed by valuation α_0
- transitions have guards that read and write variables: linear arithmetic expressions over V and V'

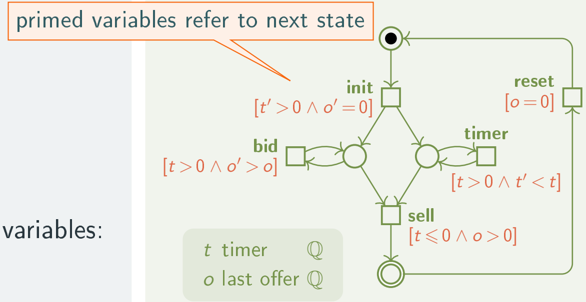
Example (Auction model)



Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F
- data: set of "global" variables V
  with numeric domain ($\mathbb{Q}$ or $\mathbb{Z}$)
- initial values of V are fixed by valuation α_0
- transitions have guards that read and write variables:
  linear arithmetic expressions over V and V'

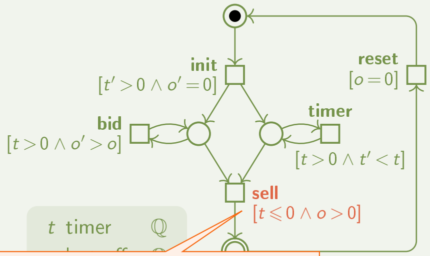
Example (Auction model)



Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F
- data: set of "global" variables V with numeric domain ($\mathbb{Q}$ or $\mathbb{Z}$)
- initial values of V are fixed by valuation α_0
- transitions have guards that read and write variables: linear arithmetic expressions over V and V'

Example (Auction model)

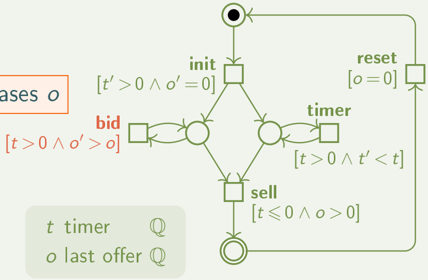


requires that $t \leqslant 0$ and o positive

Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F
- data: set of “global” variables requires that t positive, increases o
  with numeric domain ($\mathbb{Q}$ or $\mathbb{Z}$)
- initial values of V are fixed by valuation α_0
- transitions have guards that read and write variables:
  linear arithmetic expressions over V and V'

Example (Auction model)

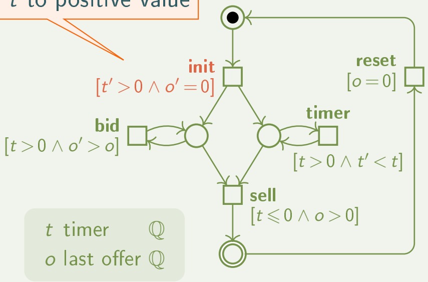


Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F
- data: set of "global" variables V with numeric domain ($\mathbb{Q}$ or $\mathbb{Z}$)
- initial values of V are fixed by valuation α_0
- transitions have guards that read and write variables: linear arithmetic expressions over V and V'

Example (Auction model)

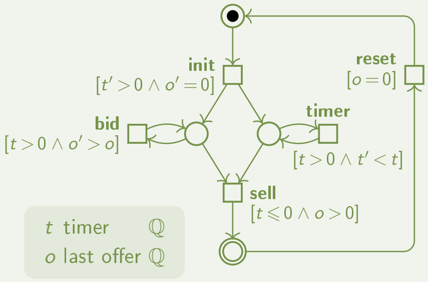
set o to 0 and t to positive value



Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F
- data: set of "global" variables V with numeric domain ($\mathbb{Q}$ or $\mathbb{Z}$)
- initial values of V are fixed by valuation α_0
- transitions have guards that read and write variables: linear arithmetic expressions over V and V'

Example (Auction model)



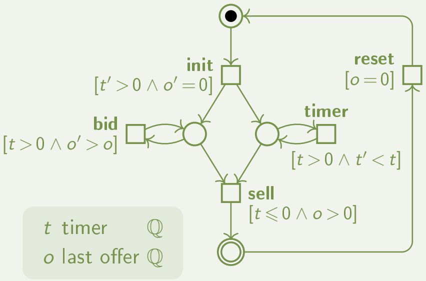
Background logic

- propositional logic + theory of linear arithmetic over integers and rationals
- satisfiability is decidable (SMT solvers), quantifiers can be eliminated

Data Petri net (DPN)

- based on Petri net
- initial and final markings M_I and M_F
- data: set of "global" variables V with numeric domain ($\mathbb{Q}$ or $\mathbb{Z}$)
- initial values of V are fixed by valuation α_0
- transitions have guards that read and write variables: linear arithmetic expressions over V and V'

Example (Auction model)



Remark

- DPNs can be mined automatically from data [Mannhardt et al 2016, de Leoni 2013]
- used to model BPM processes from various domains [Mannhardt et al 2016, Mannhardt 2018]

Observation

if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Observation

if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

Observation

if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN

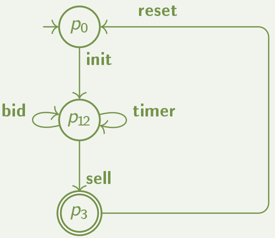
Observation

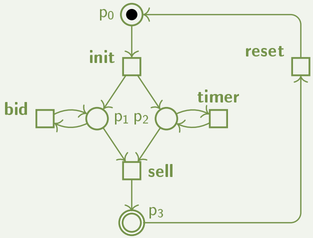
if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN

Example (DDSA for DPN)





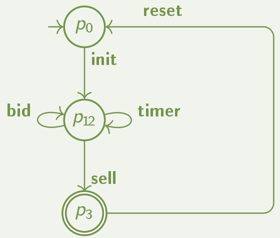
Observation

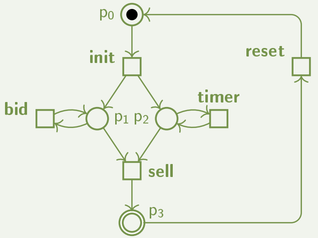
if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN
- data variables V like in DPN, with α_0

Example (DDSA for DPN)





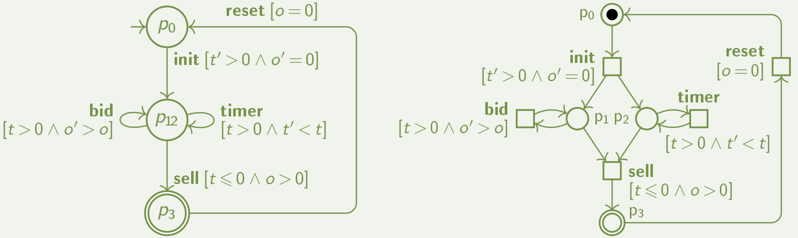
Observation

if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0

Example (DDSA for DPN)



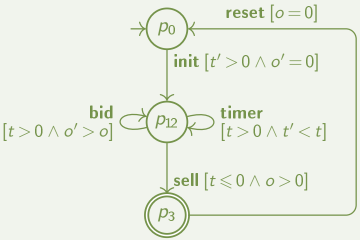
Observation

if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)



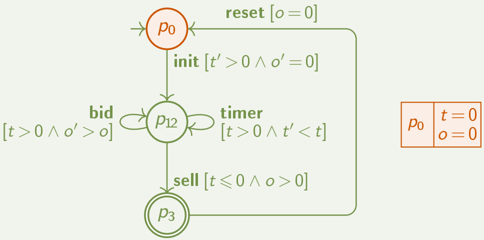
Observation

if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)



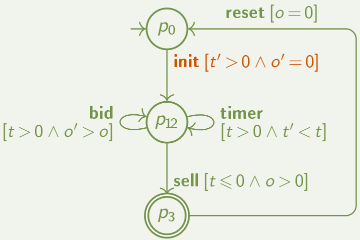
Observation

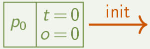
if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)





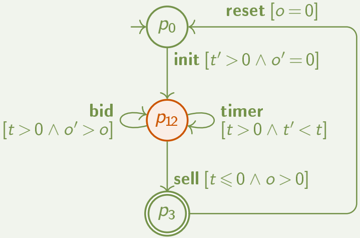
Observation

if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)



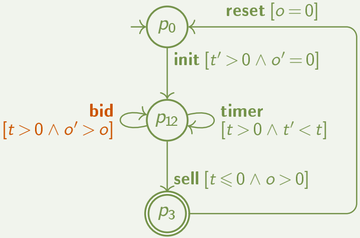
Observation

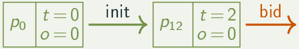
if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)





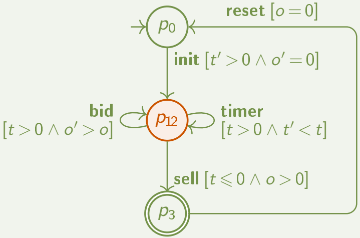
Observation

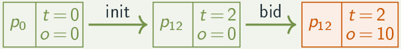
if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)





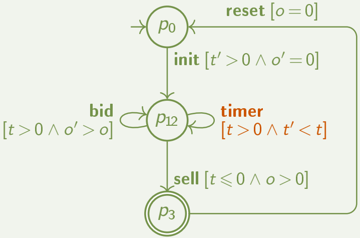
Observation

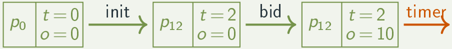
if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)





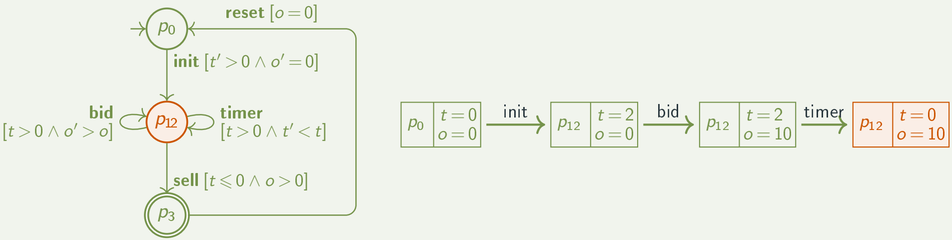
Observation

if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)



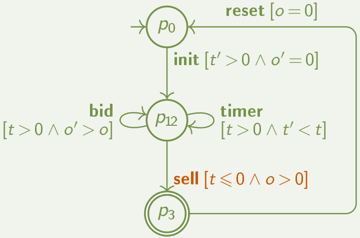
Observation

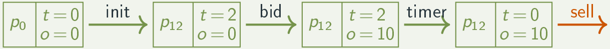
if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)





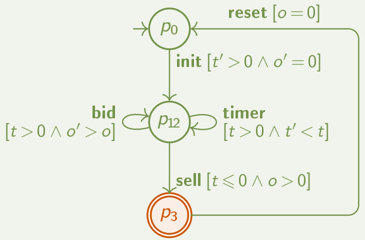
Observation

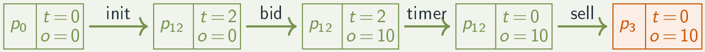
if underlying Petri net is bounded, DPN has finite set of markings: can be unfolded

Data-aware Dynamic System with Arithmetic (DDSA)

- labeled transition system
  - control states: markings of DPN
  - transitions: reflect firings in DPN, with same guards
- data variables V like in DPN, with α_0
- run is sequence of states and valuations of V

Example (DDSA for DPN)





State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V

Example

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V

Example

- state

p_{12}	$t = 2$ $o = 0$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V

Example

- state

p_{12}	$t = 2$ $o = 0$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V

Example

- state $\boxed{p_{12}} \boxed{\begin{array}{c} t = 2 \\ o = 0 \end{array}}$
- abstract state (p_{12}, φ)

$\varphi = (o = 0) \wedge (t > 0)$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

- state $\boxed{p_{12}}\boxed{\begin{array}{c} t=2 \\ o=0 \end{array}}$
- abstract state (p_{12}, φ)

$\varphi = (o=0) \wedge (t>0)$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

- state $\boxed{p_{12} \;\Big|\; \begin{array}{c} t = 2 \\ o = 0 \end{array}}$

$$\Big\downarrow \text{matches}$$

- abstract state (p_{12}, φ)

$\varphi = (o = 0) \wedge (t > 0)$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

- state

$$\boxed{p_{12} \;\Big|\; \begin{array}{l} t = 2 \\ o = 0 \end{array}} \xrightarrow[t > 0 \wedge o' > o]{\text{bid}} \boxed{p_{12} \;\Big|\; \begin{array}{l} t = 2 \\ o = 10 \end{array}}$$

matches

- abstract state (p_{12}, φ)

$\varphi = (o = 0) \wedge (t > 0)$

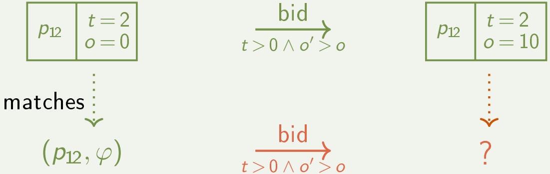
State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

- state
- abstract state



$\varphi = (o = 0) \land (t > 0)$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

- state
- abstract state

$$\begin{tikzcd}[column sep=huge]
\boxed{p_{12} \mid \begin{smallmatrix} t=2 \\ o=0 \end{smallmatrix}} \arrow[r, "\text{bid}", "t>0 \wedge o'>o"'] \arrow[d, dotted, "\text{matches}"'] & \boxed{p_{12} \mid \begin{smallmatrix} t=2 \\ o=10 \end{smallmatrix}} \arrow[d, dotted] \\
(p_{12}, \varphi) \arrow[r, "\text{bid}", "t>0 \wedge o'>o"'] & (p_{12}, \varphi')
\end{tikzcd}$$

$\varphi = (o = 0) \wedge (t > 0)$

$\varphi' = ?$

Definition (Update)

describes how formula φ changes after transition a

for formula φ and transition a in DDSA

$$update(\varphi, a) =$$

Definition (Update)

for formula φ and transition a i

rename variables in formula to auxiliary $\widehat{V} = \{\widehat{v} \mid v \in V\}$

$$update(\varphi, a) = \qquad \varphi(\widehat{V})$$

Definition (Update)

for formula φ and transition a in DDSA

guard must hold, propagate variables that are not written

$$update(\varphi, a) = \quad \varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v$$

Definition (Update)

∃ quantification to get formula with free variables V

for formula φ and transition a in DDSA

$$update(\varphi, a) = \exists \widehat{V}.\ (\varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v)$$

Definition (Update)

for formula φ and transition a in DDSA

can get equivalent quantifier-free formula by quantifier elimination

$$update(\varphi, a) = \exists \widehat{V}.\ (\varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v)$$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

- state
- abstract state

$$\begin{tikzcd}
\boxed{p_{12} \;\Big|\; \begin{matrix} t=2 \\ o=0 \end{matrix}} \arrow[r, "\text{bid}", "t>0 \wedge o'>o"'] \arrow[d, dotted, "\text{matches}"'] & \boxed{p_{12} \;\Big|\; \begin{matrix} t=2 \\ o=10 \end{matrix}} \arrow[d, dotted] \\
(p_{12}, \varphi) \arrow[r, "\text{bid}", "t>0 \wedge o'>o"'] & (p_{12}, \varphi')
\end{tikzcd}$$

$\varphi = (o=0) \wedge (t>0)$

$\varphi' = \mathit{update}(\varphi, \text{bid})$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

- state

$$\boxed{p_{12} \;\Big|\; \begin{array}{l} t=2 \\ o=0 \end{array}} \xrightarrow[t>0 \wedge o'>o]{\text{bid}} \boxed{p_{12} \;\Big|\; \begin{array}{l} t=2 \\ o=10 \end{array}}$$

matches

- abstract state

$$(p_{12}, \varphi) \xrightarrow[t>0 \wedge o'>o]{\text{bid}} (p_{12}, \varphi')$$

$$\varphi = (o=0) \wedge (t>0)$$

$$\begin{aligned} \varphi' &= \mathit{update}(\varphi, \text{bid}) \\ &= \exists \hat{o}\, \hat{t}.\ (\hat{o}=0) \wedge (\hat{t}>0) \end{aligned}$$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

$$
\begin{tikzcd}
\text{▶ state} & \boxed{p_{12} \;\middle|\; \begin{matrix} t=2 \\ o=0 \end{matrix}} \arrow[rr, "\text{bid}", "t>0 \land o'>o"'] \arrow[d, dotted, "\text{matches}"'] & & \boxed{p_{12} \;\middle|\; \begin{matrix} t=2 \\ o=10 \end{matrix}} \arrow[d, dotted] \\
\text{▶ abstract state} & (p_{12}, \varphi) \arrow[rr, "\text{bid}", "t>0 \land o'>o"'] & & (p_{12}, \varphi')
\end{tikzcd}
$$

$\varphi = (o=0) \land (t>0)$

$$
\begin{aligned}
\varphi' &= \mathit{update}(\varphi, \text{bid}) \\
&= \exists \hat{o}\,\hat{t}.\ (\hat{o}=0) \land (\hat{t}>0) \land (\hat{t}>0) \land (o>\hat{o})
\end{aligned}
$$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

- state
- abstract state

$$\begin{tikzcd}[column sep=huge, row sep=large]
\boxed{p_{12}}\boxed{\begin{smallmatrix} t=2 \\ o=0 \end{smallmatrix}} \arrow[r, "\text{bid}", "t>0 \wedge o'>o"'] \arrow[d, dotted, "\text{matches}"'] & \boxed{p_{12}}\boxed{\begin{smallmatrix} t=2 \\ o=10 \end{smallmatrix}} \arrow[d, dotted] \\
(p_{12}, \varphi) \arrow[r, "\text{bid}", "t>0 \wedge o'>o"'] & (p_{12}, \varphi')
\end{tikzcd}$$

$$\varphi = (o = 0) \wedge (t > 0)$$

$$\begin{aligned} \varphi' &= \mathit{update}(\varphi, \text{bid}) \\ &= \exists \hat{o}\, \hat{t}.\ (\hat{o} = 0) \wedge (\hat{t} > 0) \wedge (\hat{t} > 0) \wedge (o > \hat{o}) \wedge (\hat{t} = t) \end{aligned}$$

State Space Abstraction

Definitions

- state of DDSA is tuple (s, α) of control state s and assignment α to data variables V
- abstract state is tuple (s, φ) of control state s and formula φ with free variables V
- (s, α) matches (s, φ) if $\alpha \models \varphi$

Example

- state
- abstract state

$$\begin{tikzcd} \boxed{p_{12} \;\middle|\; \begin{matrix} t=2 \\ o=0 \end{matrix}} \arrow[r, "\text{bid}", "t>0 \wedge o'>o"'] \arrow[d, dotted, "\text{matches}"'] & \boxed{p_{12} \;\middle|\; \begin{matrix} t=2 \\ o=10 \end{matrix}} \arrow[d, dotted] \\ (p_{12}, \varphi) \arrow[r, "\text{bid}", "t>0 \wedge o'>o"'] & (p_{12}, \varphi') \end{tikzcd}$$

$\varphi = (o = 0) \wedge (t > 0)$

$$\begin{aligned} \varphi' &= \mathit{update}(\varphi, \text{bid}) \\ &= \exists \hat{o}\, \hat{t}.\ (\hat{o} = 0) \wedge (\hat{t} > 0) \wedge (\hat{t} > 0) \wedge (o > \hat{o}) \wedge (\hat{t} = t) \\ &\equiv (o > 0) \wedge (t > 0) \end{aligned}$$

Definition (Update)

for formula φ and transition a in DDSA

$$update(\varphi, a) = \exists \widehat{V}. \, (\varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v)$$

Definition (Constraint graph)

is graph with node set of abstract states such that

Definition (Update)

for formula φ and transition a in DDSA

$$update(\varphi, a) = \exists \widehat{V}. \left(\varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v\right)$$

Definition (Constraint graph)

is graph with node set of abstract states such that

- initial node is (s_0, φ_0), with $\varphi_0 = \bigwedge_{v \in V} v = \alpha_0(v)$

Definition (Update)

for formula φ and transition a in DDSA

$$update(\varphi, a) = \exists \widehat{V}.\, (\varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v)$$

Definition (Constraint graph)

is graph with node set of abstract states such that

- initial node is (s_0, φ_0), with $\varphi_0 = \bigwedge_{v \in V} v = \alpha_0(v)$
- for every node (s, φ)

Definition (Update)

for formula φ and transition a in DDSA

$$update(\varphi, a) = \exists \widehat{V}.\,(\varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v)$$

Definition (Constraint graph)

is graph with node set of abstract states such that

- initial node is (s_0, φ_0), with $\varphi_0 = \bigwedge_{v \in V} v = \alpha_0(v)$
- for every node (s, φ) where $s \xrightarrow{a} s'$

Definition (Update)

for formula φ and transition a in DDSA

$$update(\varphi, a) = \exists \widehat{V}.\, (\varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v)$$

Definition (Constraint graph)

is graph with node set of abstract states such that

- initial node is (s_0, φ_0), with $\varphi_0 = \bigwedge_{v \in V} v = \alpha_0(v)$
- for every node (s, φ) where $s \xrightarrow{a} s'$ and $update(\varphi, a)$ is satisfiable

Definition (Update)

for formula φ and transition a in DDSA

$$update(\varphi, a) = \exists \widehat{V}. \left(\varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v\right)$$

Definition (Constraint graph)

is graph with node set of abstract states such that

- initial node is (s_0, φ_0), with $\varphi_0 = \bigwedge_{v \in V} v = \alpha_0(v)$
- for every node (s, φ) where $s \xrightarrow{a} s'$ and $update(\varphi, a)$ is satisfiable there is node (s', φ') such that $\varphi' \equiv update(\varphi, a)$

Definition (Update)

for formula φ and transition a in DDSA

$$\mathit{update}(\varphi, a) = \exists \widehat{V}.\, (\varphi(\widehat{V}) \wedge \mathit{guard}_a(\widehat{V}, V) \wedge \bigwedge_{v \notin \mathit{write}(a)} \widehat{v} = v)$$

Definition (Constraint graph)

is graph with node set of abstract states such that

- initial node is (s_0, φ_0), with $\varphi_0 = \bigwedge_{v \in V} v = \alpha_0(v)$
- for every node (s, φ) where $s \xrightarrow{a} s'$ and $\mathit{update}(\varphi, a)$ is satisfiable there is node (s', φ') such that $\varphi' \equiv \mathit{update}(\varphi, a)$, and edge $(s, \varphi) \xrightarrow{a} (s', \varphi')$

Definition (Update)

for formula φ and transition a in DDSA

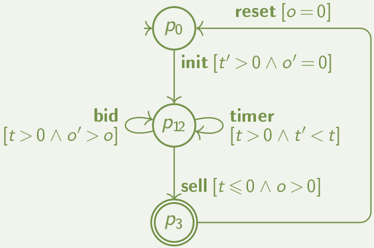
$$update(\varphi, a) = \exists \widehat{V}. \, (\varphi(\widehat{V}) \wedge guard_a(\widehat{V}, V) \wedge \bigwedge_{v \notin write(a)} \widehat{v} = v)$$

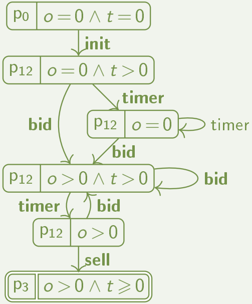
Definition (Constraint graph)

is graph with node set of abstract states such that

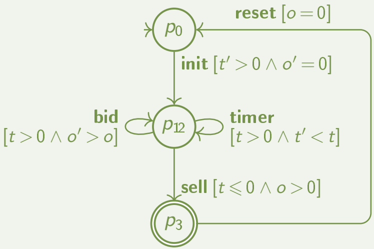
- initial node is (s_0, φ_0), with $\varphi_0 = \bigwedge_{v \in V} v = \alpha_0(v)$
- for every node (s, φ) where $s \xrightarrow{a} s'$ and $update(\varphi, a)$ is satisfiable
  there is node (s', φ') such that $\varphi' \equiv update(\varphi, a)$, and edge $(s, \varphi) \xrightarrow{a} (s', \varphi')$
- $(s, \dots) \in N$ is final if s is final in DDSA

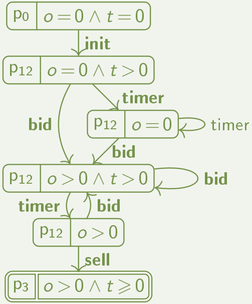
Example (Constraint graph for auction model)





Example (Constraint graph for auction model)

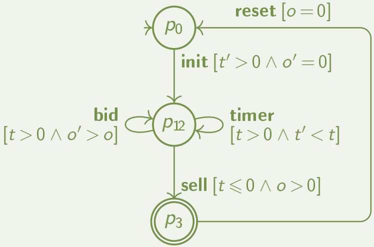


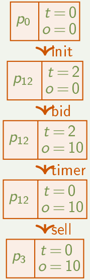


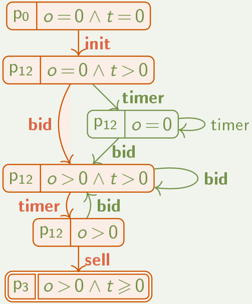
Key Lemma

$\exists$ DDSA run $(s_0, \alpha_0) \xrightarrow{*} (s, \alpha)$ $\iff$ $\exists$ path in CG $(s_0, \varphi_0) \xrightarrow{*} (s, \varphi)$ with $\alpha \models \varphi$

Example (Constraint graph for auction model)



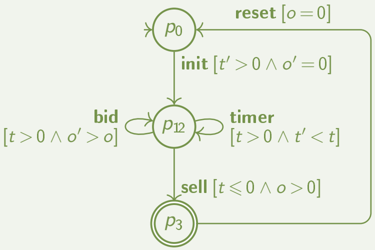


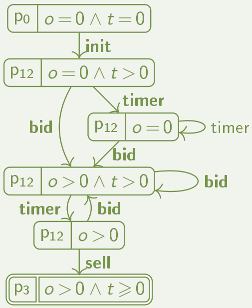


Key Lemma

$\exists$ DDSA run $(s_0, \alpha_0) \xrightarrow{*} (s, \alpha)$ $\iff$ $\exists$ path in CG $(s_0, \varphi_0) \xrightarrow{*} (s, \varphi)$ with $\alpha \models \varphi$

Example (Constraint graph for auction model)





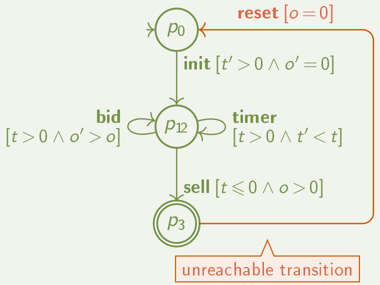
Key Lemma

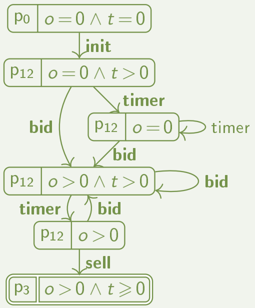
$\exists$ DDSA run $(s_0, \alpha_0) \xrightarrow{*} (s, \alpha)$ $\iff$ $\exists$ path in CG $(s_0, \varphi_0) \xrightarrow{*} (s, \varphi)$ with $\alpha \models \varphi$

Observation

control state or transition of DDSA are reachable iff they appear in the constraint graph

Example (Constraint graph for auction model)





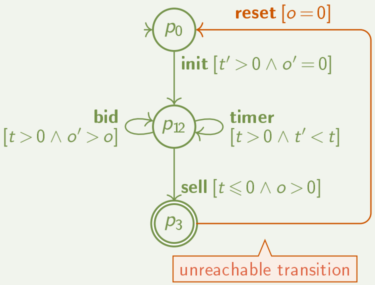
Key Lemma

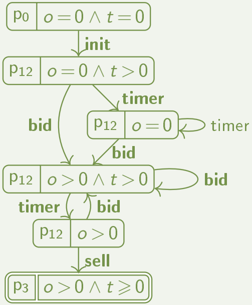
$\exists$ DDSA run $(s_0, \alpha_0) \xrightarrow{*} (s, \alpha)$ $\iff$ $\exists$ path in CG $(s_0, \varphi_0) \xrightarrow{*} (s, \varphi)$ with $\alpha \models \varphi$

Observation

control state or transition of DDSA are reachable iff they appear in the constraint graph

Example (Constraint graph for auction model)





Key Lemma

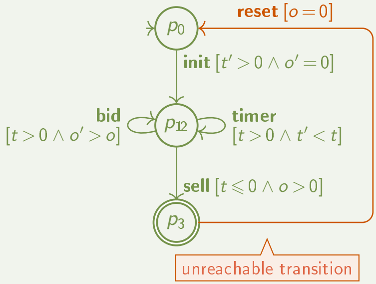
$\exists$ DDSA run $(s_0, \alpha_0) \xrightarrow{*} (s, \alpha)$ $\iff$ $\exists$ path in CG $(s_0, \varphi_0) \xrightarrow{*} (s, \varphi)$ with $\alpha \models \varphi$

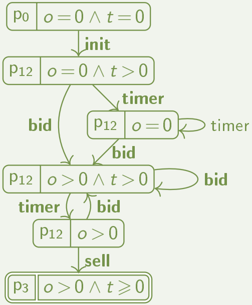
if CG is finite, reachability is decidable

Observation

control state or transition of DDSA are reachable iff they appear in the constraint graph

Example (Constraint graph for auction model)





Key Lemma

$\exists$ DDSA run $(s_0, \alpha_0) \xrightarrow{*} (s, \alpha)$ $\iff$ $\exists$ path in CG $(s_0, \varphi_0) \xrightarrow{*} (s, \varphi)$ with $\alpha \models \varphi$

Caveat

constraint graph can be infinite

Decidability Conditions

formulas in CG are history constraints:
$\exists \ldots \exists$ (conjunctions of renamed transition guards)

Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Decidability Conditions

Definition (Finite summary) abstract decidability condition

DDSA has finite summary if set of history constraints is finite

Decidability Conditions

Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

[Felli, Montali & W, AAAI 2022]

Decidability Conditions

Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

[Felli, Montali & W, AAAI 2022]

- restrict shape of transition guards:
  - variable-to-variable/constant comparisons over $\mathbb{Q}$: $x < y$, $x \geqslant \frac{1}{2}$, $y = 0$ [Demri & de Souza 2006]

Decidability Conditions

Definition (Finite summary)

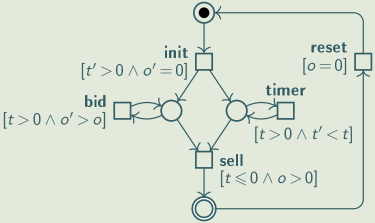
DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

[Felli, Montali & W, AAAI 2022]

- restrict shape of transition guards:
  - variable-to-variable/constant comparisons over $\mathbb{Q}$: $x < y$, $x \geq \frac{1}{2}$, $y = 0$ [Demri & de Souza 2006]

applies to all DPNs mined according to [Mannhardt et al 2016]



Decidability Conditions

Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

[Felli, Montali & W, AAAI 2022]

- restrict shape of transition guards:
  - variable-to-variable/constant comparisons over $\mathbb{Q}$: $x < y$, $x \geqslant \frac{1}{2}$, $y = 0$ [Demri & de Souza 2006]
  - integer periodicity constraints over $\mathbb{Z}$: $x = y$, $x < 3$, $y \equiv_5 3$ [Demri 2006, Gascon 2009]

Decidability Conditions

Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

[Felli, Montali & W, AAAI 2022]

- restrict shape of transition guards:
  - variable-to-variable/constant comparisons over $\mathbb{Q}$: $x < y$, $x \geqslant \frac{1}{2}$, $y = 0$ [Demri & de Souza 2006]
  - integer periodicity constraints over $\mathbb{Z}$: $x = y$, $x < 3$, $y \equiv_5 3$ [Demri 2006, Gascon 2009]
  - gap-order constraints: $x - y \geqslant 2$ [Cerans 1995, Bozzelli & Pinchinat 2014]

Decidability Conditions

Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

[Felli, Montali & W, AAAI 2022]

- restrict shape of transition guards:
  - variable-to-variable/constant comparisons over $\mathbb{Q}$: $x < y$, $x \geqslant \frac{1}{2}$, $y = 0$ [Demri & de Souza 2006]
  - integer periodicity constraints over $\mathbb{Z}$: $x = y$, $x < 3$, $y \equiv_5 3$ [Demri 2006, Gascon 2009]
  - gap-order constraints: $x - y \geqslant 2$ [Cerans 1995, Bozzelli & Pinchinat 2014]
- restrict control flow:
  - feedback freedom [Damaggio, Deutsch & Vianu 2012]

Decidability Conditions

Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

[Felli, Montali & W, AAAI 2022]

- restrict shape of transition guards:
  - variable-to-variable/constant comparisons over $\mathbb{Q}$: $x < y$, $x \geqslant \frac{1}{2}$, $y = 0$ [Demri & de Souza 2006]
  - integer periodicity constraints over $\mathbb{Z}$: $x = y$, $x < 3$, $y \equiv_5 3$ [Demri 2006, Gascon 2009]
  - gap-order constraints: $x - y \geqslant 2$ [Cerans 1995, Bozzelli & Pinchinat 2014]
- restrict control flow:
  - feedback freedom [Damaggio, Deutsch & Vianu 2012]
  - bounded memory

idea: behaviour depends on bounded number of past steps

Decidability Conditions

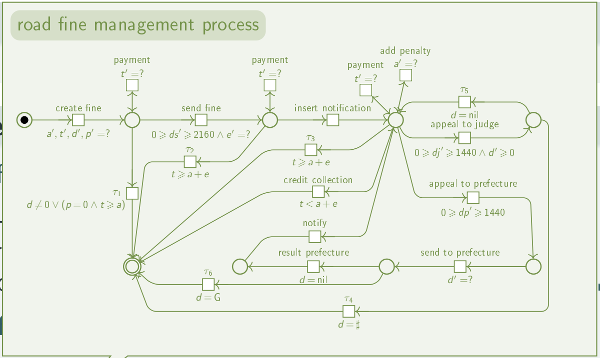
Definition (Finite

DDSA has finite sum

Concrete instance

[Felli, Montali & W, AAAI 2022]

- restrict shape of
  - variable-to- ... $y = 0$ [Demri & de Souza 2006]
  - integer per ... [Demri 2006, Gascon 2009]
  - gap-order c ... rans 1995, Bozzelli & Pinchinat 2014]
- restrict control f
  - feedback freedom [Damaggio, Deutsch & Vianu 2012]
  - bounded memory



idea: behaviour depends on bounded number of past steps

Decidability Conditions

Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

[Felli, Montali & W, AAAI 2022]

- restrict shape of transition guards:
  - variable-to-variable/constant comparisons over $\mathbb{Q}$: $x < y$, $x \geqslant \frac{1}{2}$, $y = 0$ [Demri & de Souza 2006]
  - integer periodicity constraints over $\mathbb{Z}$: $x = y$, $x < 3$, $y \equiv_5 3$ [Demri 2006, Gascon 2009]
  - gap-order constraints: $x - y \geqslant 2$ [Cerans 1995, Bozzelli & Pinchinat 2014]
- restrict control flow:
  - feedback freedom [Damaggio, Deutsch & Vianu 2012]
  - bounded memory
- DDSA can be decomposed into subsystems that have finite summary
  - into sequential process parts
  - by splitting variables

Decidability Conditions

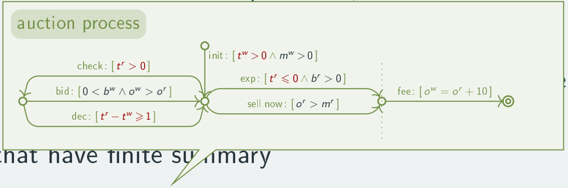
Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

[Felli, Montali & W, AAAI 2022]

- restrict shape of transition guards:
  - variable-to-variable/constant comparisons over $\mathbb{Q}$: $x < y$, $x \geqslant \frac{1}{2}$, $y = 0$ [Demri & de Souza 2006]
  - integer periodicity constraints over $\mathbb{Z}$: $x = y$, $x < 3$, $y \equiv_5 3$ [Demri 2006, Gascon 2009]
  - gap-order constraints: $x - y \geqslant 2$ [Cerans 1995, Bozzelli & Pinchinat 2014]
- restrict control flow:
  - feedback freedom
  - bounded memory



- DDSA can be decomposed into subsystems that have finite summary
  - into sequential process parts
  - by splitting variables

Decidability Conditions

Definition (Finite summary)

DDSA has finite summary if set of history constraints is finite

Concrete instances of finite summary

capture wide range of DPNs from literature

- restrict shape of transition guards:
  - variable-to-variable/constant comparisons over $\mathbb{Q}$: $x < y$, $x \geqslant \frac{1}{2}$, $y = 0$ [Demri & de Souza 2006]
  - integer periodicity constraints over $\mathbb{Z}$: $x = y$, $x < 3$, $y \equiv_5 3$ [Demri 2006, Gascon 2009]
  - gap-order constraints: $x - y \geqslant 2$ [Cerans 1995, Bozzelli & Pinchinat 2014]
- restrict control flow:
  - feedback freedom [Damaggio, Deutsch & Vianu 2012]
  - bounded memory
- DDSA can be decomposed into subsystems that have finite summary
  - into sequential process parts
  - by splitting variables

Outline

Linear-Time Model Checking

Verification problem: Compliance

given DDSA and LTL$_f$ formula ψ with arithmetic constraints:

$$\mathit{constraint} \;\mid\; \mathit{control\ state} \;\mid\; \psi \wedge \psi \;\mid\; \psi \vee \psi \;\mid\; \langle \mathit{action} \rangle \psi \;\mid\; \mathsf{X}\,\psi \;\mid\; \mathsf{F}\,\psi \;\mid\; \mathsf{G}\,\psi \;\mid\; \psi\,\mathsf{U}\,\psi$$

is there a witness run of DDSA that satisfies ψ?

Linear-Time Model Checking

evaluated over finite traces

Verification problem: Compliance

given DDSA and LTL$_f$ formula ψ with arithmetic constraints:

$$\textit{constraint} \;\mid\; \textit{control state} \;\mid\; \psi \wedge \psi \;\mid\; \psi \vee \psi \;\mid\; \langle \textit{action} \rangle \psi \;\mid\; \mathsf{X}\,\psi \;\mid\; \mathsf{F}\,\psi \;\mid\; \mathsf{G}\,\psi \;\mid\; \psi\,\mathsf{U}\,\psi$$

is there a witness run of DDSA that satisfies ψ?

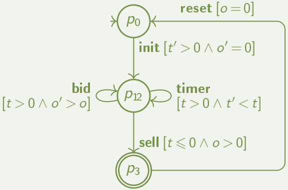
Verification problem: Compliance

given DDSA and LTL_f formula ψ with arithmetic constraints:

$$\mathit{constraint} \mid \mathit{control\ state} \mid \psi \wedge \psi \mid \psi \vee \psi \mid \langle \mathit{action} \rangle \psi \mid \mathsf{X}\,\psi \mid \mathsf{F}\,\psi \mid \mathsf{G}\,\psi \mid \psi\,\mathsf{U}\,\psi$$

is there a witness run of DDSA that satisfies ψ?

Example



- $\mathsf{F}\,((o=100) \wedge \mathsf{G}\,(p_3 \rightarrow o \neq 100))$: witness exists
  it is possible that bid of 100€ does not win

Linear-Time Model Checking

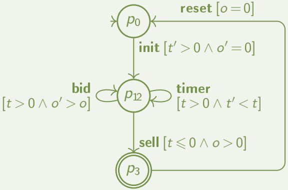
Verification problem: Compliance

given DDSA and LTL$_f$ formula ψ with arithmetic constraints:

$$\mathit{constraint} \mid \mathit{control\ state} \mid \psi \land \psi \mid \psi \lor \psi \mid \langle \mathit{action} \rangle \psi \mid \mathsf{X}\,\psi \mid \mathsf{F}\,\psi \mid \mathsf{G}\,\psi \mid \psi\,\mathsf{U}\,\psi$$

is there a witness run of DDSA that satisfies ψ?

Example



- $\mathsf{F}\,((o = 100) \land \mathsf{G}\,(p_3 \rightarrow o \neq 100))$: witness exists
  it is possible that bid of 100€ does not win
- $\mathsf{F}\,(\langle \textbf{bid} \rangle \langle \textbf{sell} \rangle \top)$: no witness exists
  it is possible that a sell happens right after a bid

Linear-Time Model Checking

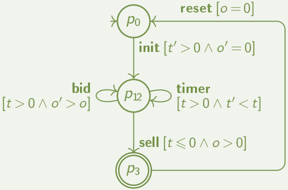
Verification problem: Compliance

given DDSA and LTL$_f$ formula ψ with arithmetic constraints:

$$\textit{constraint} \mid \textit{control state} \mid \psi \wedge \psi \mid \psi \vee \psi \mid \langle \textit{action} \rangle \psi \mid \mathsf{X}\,\psi \mid \mathsf{F}\,\psi \mid \mathsf{G}\,\psi \mid \psi\,\mathsf{U}\,\psi$$

is there a witness run of DDSA that satisfies ψ?

Example

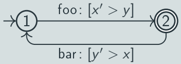


- $\mathsf{F}\,((o = 100) \wedge \mathsf{G}\,(p_3 \to o \neq 100))$: witness exists
  it is possible that bid of 100€ does not win
- $\mathsf{F}\,(\langle \textbf{bid} \rangle \langle \textbf{sell} \rangle \top)$: no witness exists
  it is possible that a sell happens right after a bid

Fact

can construct finite automaton (NFA) accepting exactly those runs that satisfy LTL$_f$ property

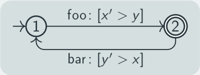
Model checking approach



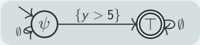
DDSA

Model checking approach

check $\mathsf{F}\,(y > 5)$



$+$

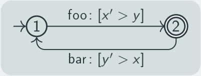


DDSA

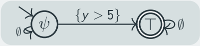
NFA for ψ

Model checking approach

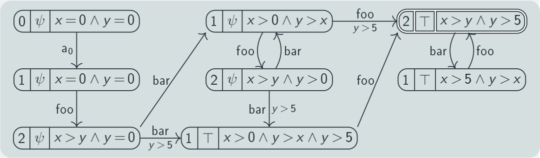
check F $(y > 5)$



$+$



$\mapsto$

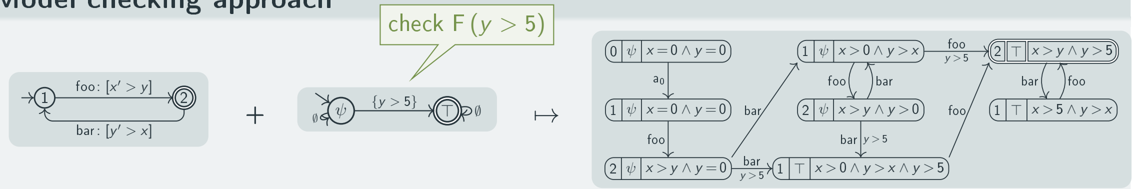


DDSA

NFA for ψ

product automaton $\mathcal{N}$

Model checking approach



DDSA

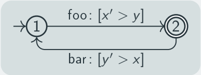
NFA for ψ

product automaton $\mathcal{N}$

Product automaton

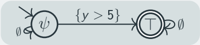
- **nodes** are triples (DDSA state, NFA state, formula)

Model checking approach



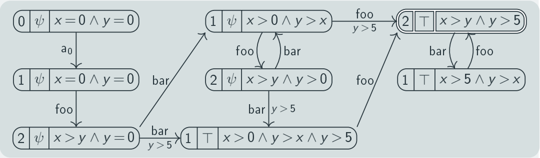
DDSA

+



NFA for ψ

$\mapsto$

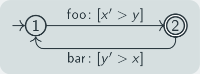


product automaton $\mathcal{N}$

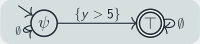
Product automaton

- nodes are triples (DDSA state, NFA state, formula)
- construction is similar as for constraint graph, but edges combine DDSA and NFA steps

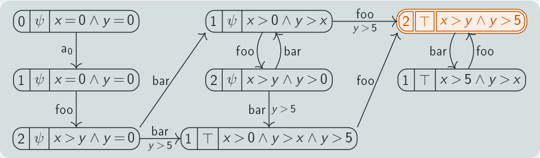
Model checking approach



+



↦



DDSA

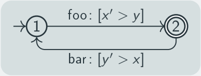
NFA for ψ

product automaton $\mathcal{N}$

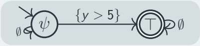
Product automaton

- nodes are triples (DDSA state, NFA state, formula)
- construction is similar as for constraint graph, but edges combine DDSA and NFA steps
- final nodes are those that combine final DDSA and NFA state

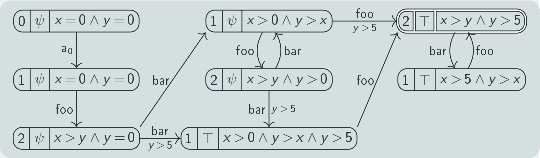
Model checking approach



$+$



$\mapsto$



DDSA

NFA for ψ

product automaton $\mathcal{N}$

Product automaton

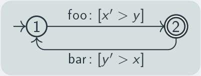
- nodes are triples (DDSA state, NFA state, formula)
- construction is similar as for constraint graph, but edges combine DDSA and NFA steps
- final nodes are those that combine final DDSA and NFA state

Theorem

- *product automaton has final node iff DDSA admits witness for* ψ

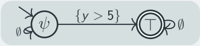
[Felli, Montali & W, AAAI 2022]

Model checking approach



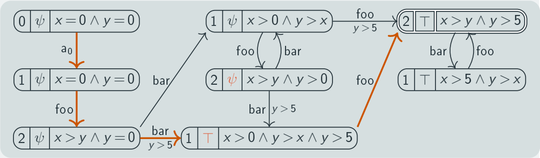
DDSA

+



NFA for ψ

$\mapsto$



product automaton $\mathcal{N}$

Product automaton

- nodes are triples (DDSA state, NFA state, formula)
- construction is similar as for constraint graph, but edges combine DDSA and NFA steps
- final nodes are those that combine final DDSA and NFA state

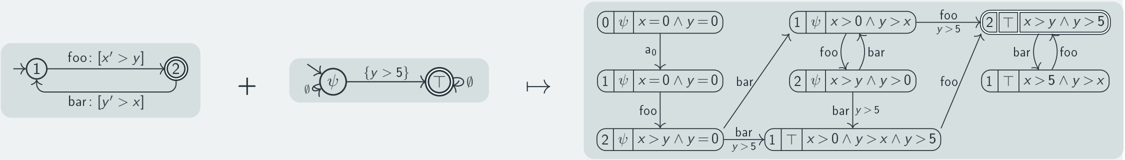
Theorem

can use SMT solver to extract witness from accepting path

- *product automaton has final node* *iff* *DDSA admits witness for* ψ

[Felli, Montali & W, AAAI 2022]

Model checking approach



DDSA | NFA for ψ | product automaton $\mathcal{N}$

Product automaton

- nodes are triples (DDSA state, NFA state, formula)
- construction is similar as for constraint graph, but edges combine DDSA and NFA steps
- final nodes are those that combine final DDSA and NFA state

Theorem

- *product automaton has final node iff DDSA admits witness for ψ*
- *LTL$_f$ model checking is decidable if DDSA has finite summary with respect to ψ*

[Felli, Montali & W, AAAI 2022]

Data-aware Soundness

Definition

DPN is data-aware sound if

Data-aware Soundness

Definition

DPN is data-aware sound if

1. from any reachable state, a final state can be reached

Data-aware Soundness

Definition

DPN is data-aware sound if

1. from any reachable state, a final state can be reached
2. termination is clean (no reachable marking M is such that $M \supset M_F$)

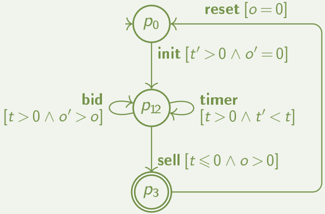
Data-aware Soundness

Definition

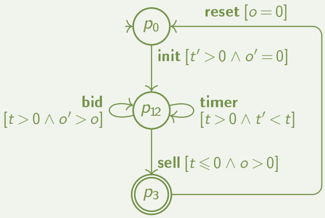
DPN is data-aware sound if

1. from any reachable state, a final state can be reached
2. termination is clean (no reachable marking M is such that $M \supset M_F$)
3. all transitions are reachable

Example (Auction)



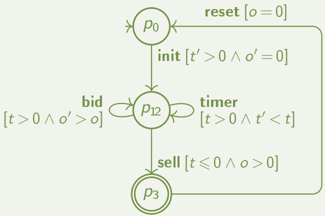
Example (Auction)



not data-aware sound because

- transition **reset** is unreachable

Example (Auction)



not data-aware sound because

- transition **reset** is unreachable
- deadlocks exist, e.g. after $\boxed{p_0 \mid \begin{smallmatrix} t=0 \\ o=0 \end{smallmatrix}} \xrightarrow{\text{init}} \boxed{p_{12} \mid \begin{smallmatrix} t=1 \\ o=0 \end{smallmatrix}} \xrightarrow{\text{timer}} \boxed{p_{12} \mid \begin{smallmatrix} t=0 \\ o=0 \end{smallmatrix}}$

Data-aware Soundness

Definition

can be checked on corresponding DDSA

DPN is data-aware sound if

1. from any reachable state, a final state can be reached
2. termination is clean (no reachable marking M is such that $M \supset M_F$)
3. all transitions are reachable

Data-aware Soundness

Definition

DPN is data-aware sound if

can be checked on corresponding DDSA

1. from any reachable state, a final state can be reached
2. termination is clean (no reachable marking M is such that $M \supset M_F$)
3. all transitions are reachable

Soundness checking approach

[Felli, Montali & W, CAiSE 2022]

- can check 2 and 3 directly on CG

Data-aware Soundness

Definition

can be checked on corresponding DDSA

DPN is data-aware sound if

1. from any reachable state, a final state can be reached
2. termination is clean (no reachable marking M is such that $M \supset M_F$)
3. all transitions are reachable

Soundness checking approach

[Felli, Montali & W, CAiSE 2022]

- can check 2 and 3 directly on CG
- for 1: for each non-final CG node (s, φ):

Data-aware Soundness

Definition

DPN is data-aware sound if

can be checked on corresponding DDSA

1. from any reachable state, a final state can be reached
2. termination is clean (no reachable marking M is such that $M \supset M_F$)
3. all transitions are reachable

Soundness checking approach

[Felli, Montali & W, CAiSE 2022]

- can check 2 and 3 directly on CG
- for 1: for each non-final CG node (s, φ):
  - compute CG_s starting from s and unknown initial values V_0

Data-aware Soundness

Definition

can be checked on corresponding DDSA

DPN is data-aware sound if

1. from any reachable state, a final state can be reached
2. termination is clean (no reachable marking M is such that $M \supset M_F$)
3. all transitions are reachable

Soundness checking approach

[Felli, Montali & W, CAiSE 2022]

- can check 2 and 3 directly on CG
- for 1: for each non-final CG node (s, φ):
  - compute CG_s starting from s and unknown initial values V_0
  - extract formula $\mathit{reach\_final}(s)$ expressing conditions on V_0 that guarantee reachability of final state from s

Data-aware Soundness

Definition

can be checked on corresponding DDSA

DPN is data-aware sound if

1. from any reachable state, a final state can be reached
2. termination is clean (no reachable marking M is such that $M \supset M_F$)
3. all transitions are reachable

Soundness checking approach

[Felli, Montali & W, CAiSE 2022]

- can check 2 and 3 directly on CG
- for 1: for each non-final CG node (s, φ):
  - compute CG_s starting from s and unknown initial values V_0
  - extract formula $reach\_final(s)$ expressing conditions on V_0 that guarantee reachability of final state from s
  - $\varphi \models reach\_final(s)$ iff final state is always reachable from (s, φ)

Data-aware Soundness

Definition

can be checked on corresponding DDSA

DPN is data-aware sound if

1. from any reachable state, a final state can be reached
2. termination is clean (no reachable marking M is such that $M \supset M_F$)
3. all transitions are reachable

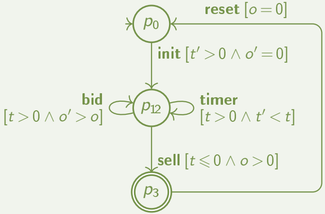
Soundness checking approach

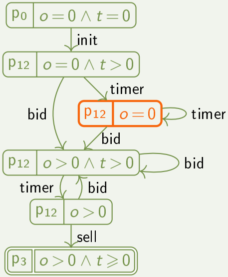
[Felli, Montali & W, CAiSE 2022]

if DDSA has finite summary, data-aware soundness is decidable

- can check 2 and 3 directly on CG
- for 1: for each non-final CG node (s, φ):
  - compute CG_s starting from s and unknown initial values V_0
  - extract formula $\mathit{reach\_final}(s)$ expressing conditions on V_0 that guarantee reachability of final state from s
  - $\varphi \models \mathit{reach\_final}(s)$ iff final state is always reachable from (s, φ)

Example (Auction)

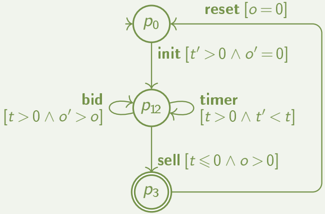


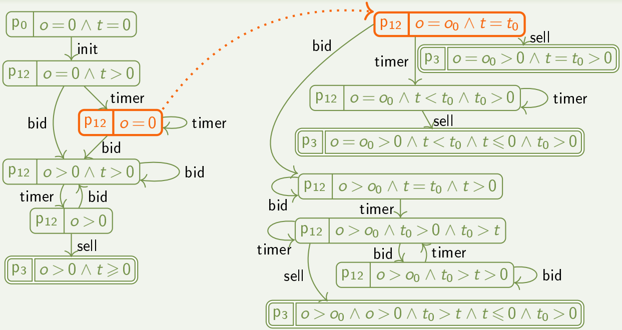


not data-aware sound because

- transition **reset** is unreachable
- deadlocks exist, e.g. after $\boxed{p_0 \;\big|\; \begin{matrix} t=0 \\ o=0 \end{matrix}} \xrightarrow{\text{init}} \boxed{p_{12} \;\big|\; \begin{matrix} t=1 \\ o=0 \end{matrix}} \xrightarrow{\text{timer}} \boxed{p_{12} \;\big|\; \begin{matrix} t=0 \\ o=0 \end{matrix}}$

Example (Auction)

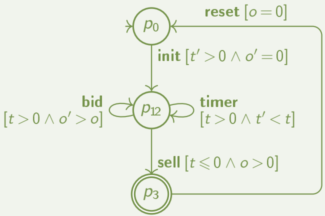


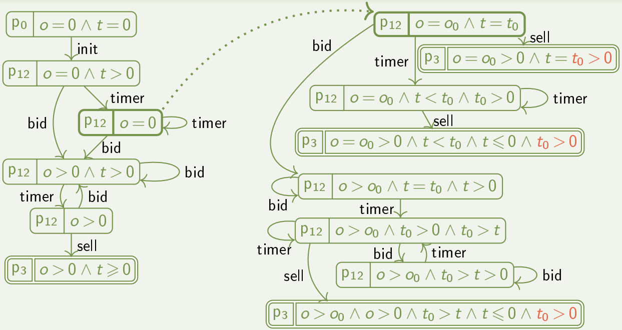


not data-aware sound because

- transition **reset** is unreachable
- deadlocks exist, e.g. after $\boxed{p_0 \;\big|\; \begin{array}{c} t = 0 \\ o = 0 \end{array}} \xrightarrow{\text{init}} \boxed{p_{12} \;\big|\; \begin{array}{c} t = 1 \\ o = 0 \end{array}} \xrightarrow{\text{timer}} \boxed{p_{12} \;\big|\; \begin{array}{c} t = 0 \\ o = 0 \end{array}}$

Example (Auction)



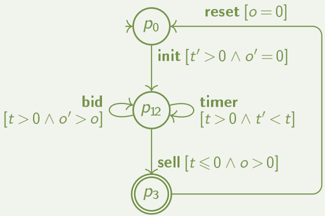


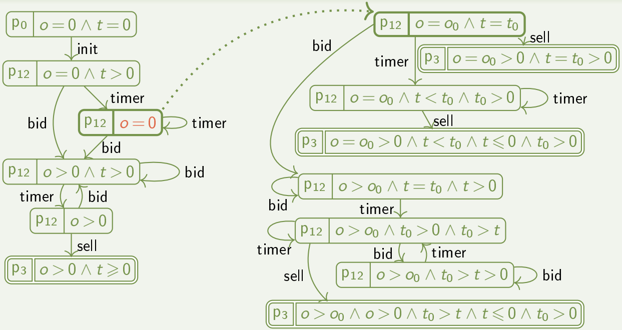
not data-aware sound because

- transition **reset** is unreachable
- deadlocks exist, e.g. after $\boxed{p_0 \mid t=0,\ o=0} \xrightarrow{\text{init}} \boxed{p_{12} \mid t=1,\ o=0} \xrightarrow{\text{timer}} \boxed{p_{12} \mid t=0,\ o=0}$

  have $reach\_final(p_{12}) = (t > 0)$

Example (Auction)



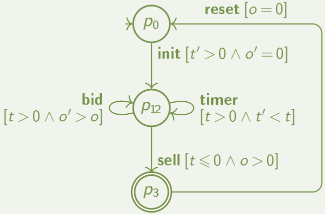


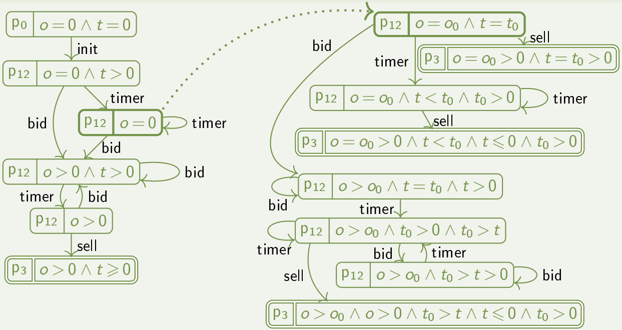
not data-aware sound because

- transition **reset** is unreachable
- deadlocks exist, e.g. after $\boxed{p_0 \,\big|\, \begin{smallmatrix} t=0 \\ o=0 \end{smallmatrix}} \xrightarrow{\text{init}} \boxed{p_{12} \,\big|\, \begin{smallmatrix} t=1 \\ o=0 \end{smallmatrix}} \xrightarrow{\text{timer}} \boxed{p_{12} \,\big|\, \begin{smallmatrix} t=0 \\ o=0 \end{smallmatrix}}$

  have $reach\_final(p_{12}) = (t > 0)$, and $(o = 0) \not\models (t > 0)$

Example (Auction)





not data-aware sound because

- transition **reset** is unreachable
- deadlocks exist, e.g. after $\boxed{p_0 \,\Big|\, \begin{matrix} t=0 \\ o=0 \end{matrix}} \xrightarrow{\text{init}} \boxed{p_{12} \,\Big|\, \begin{matrix} t=1 \\ o=0 \end{matrix}} \xrightarrow{\text{timer}} \boxed{p_{12} \,\Big|\, \begin{matrix} t=0 \\ o=0 \end{matrix}}$

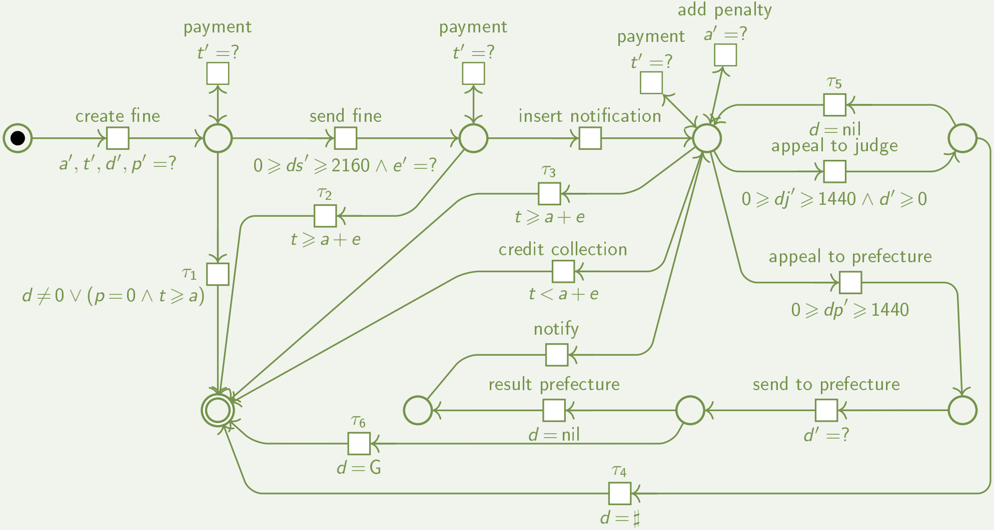
  have $\mathit{reach\_final}(p_{12}) = (t > 0)$, and $(o = 0) \not\models (t > 0)$

Branching-time model checking

use similar approach to obtain CTL* model checking procedure

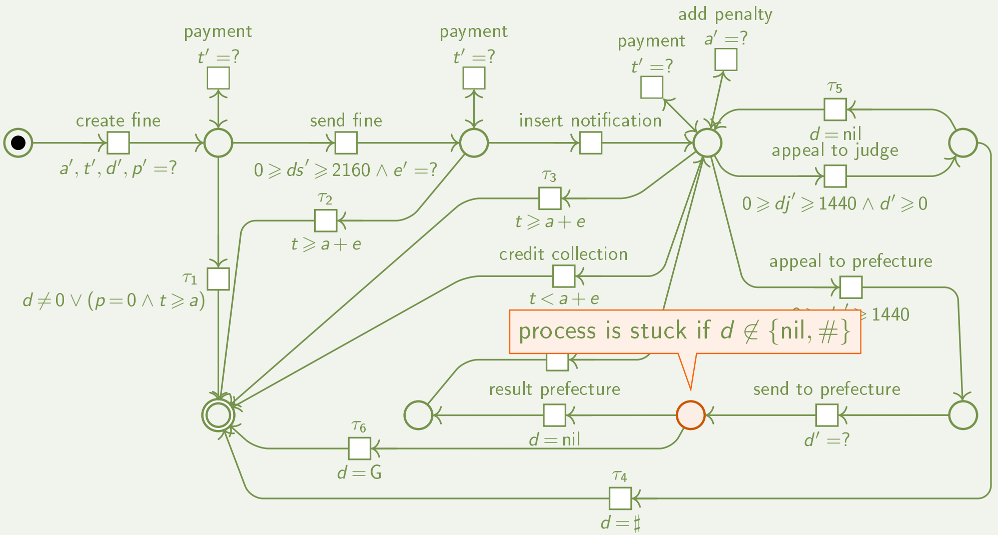
[Felli, Montali & W, IJCAR 2022]

Example (Road fine management process)



[Mannhardt et al 2016]

Example (Road fine management process) not data-aware sound



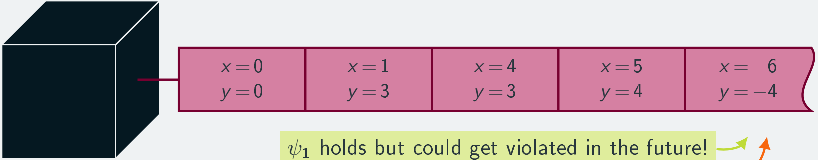
[Mannhardt et al 2016]

Monitoring Arithmetic Temporal Properties

given a trace of values, check current and possible future satisfaction of LTL_f properties like

- $\psi_1 = (y \geqslant 0)\ \mathsf{U}\ (x > y \wedge \mathsf{G}\,(x > y))$
- $\psi_2 = \mathsf{G}\,(x' > x) \wedge \mathsf{F}\,(x = 2)$

x' is value of x looking one trace instant ahead



ψ_1 holds but could get violated in the future!

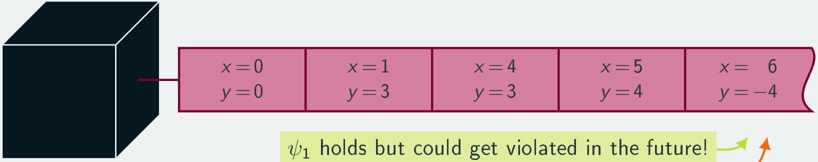
ψ_2 does not hold and will never hold in the future!

Monitoring Arithmetic Temporal Properties

given a trace of values, check current and possible future satisfaction of LTL_f properties like

- $\psi_1 = (y \geqslant 0)\ \mathsf{U}\ (x > y \wedge \mathsf{G}\,(x > y))$
- $\psi_2 = \mathsf{G}\,(x' > x) \wedge \mathsf{F}\,(x = 2)$

x' is value of x looking one trace instant ahead



ψ_1 holds but could get violated in the future!

ψ_2 does not hold and will never hold in the future!

Results

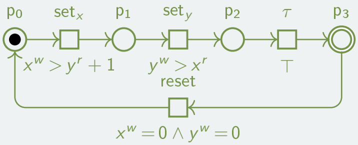
- developed monitoring procedure
- use finite summary approach to identify classes of properties where problem is decidable

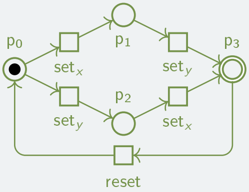
[Felli, Montali, Patrizi & W, AAAI 2023]

Process Equivalence

Verification problem

given two DPNs, do they have the same sets of configurations, and/or the same language?





Results

- can be determined using constraint graphs
- decidable for finite summary systems

talk at BPFM workshop at BPM next week!

Implementation

Arithmetic DDS Analyzer (ada)

- input DPN (+ LTL_f or CTL_f^* property)
- checks for decidability conditions, visualizes CG/product automaton
- performs LTL_f, CTL_f^* model checking, soundness checking, and monitoring
- computes witness/counterexample
- written in Python, using Z3/Yices/CVC5 for SMT solving and quantifier elimination

`https://ltl.adatool.dev` `https://soundness.adatool.dev`

`https://ctlstar.adatool.dev`

Implementation

Arithmetic DDS Analyzer (ada)

- input DPN (+ LTL_f or CTL_f^* property)
- checks for decidability conditions, visualizes CG/product automaton
- performs LTL_f, CTL_f^* model checking, soundness checking, and monitoring
- computes witness/counterexample
- written in Python, using Z3/Yices/CVC5 for SMT solving and quantifier elimination

`https://ltl.adatool.dev` `https://soundness.adatool.dev`

`https://ctlstar.adatool.dev`

Experiments

- about 60 DPNs (20 from literature, 40 artificial)
- all DPNs from literature are in some decidable class for LTL_f (but not CTL_f^*) model checking

Implementatic

Arithmetic DD

- input DPN (
- checks for d
- performs LT
- computes wi
- written in P

I

Experiments

- about 60 DF
- all DPNs fro

process	property	sat	time	checks	$\lvert\mathcal{B}\rvert$	$\lvert\mathcal{N}^{\psi}_{\mathcal{B},b}\rvert$
road fines (1)	no deadlock	✗	7.0s	8161	9	2052
	$\mathsf{AG}\,(p_7 \to \mathsf{E\,F}\,\mathit{end})$	✓	7.6s	7655		1987
road fines (2)	no deadlock	✓	15m27s	247563	9	4927
	$\mathsf{AG}\,(p_7 \to \mathsf{E\,F}\,\mathit{end})$	✓	16m7s	246813		4927
road fines (3)	no deadlock	✗	9s	9179	9	1985
	$\mathsf{AG}\,(p_7 \to \mathsf{E\,F}\,\mathit{end})$	✓	6.6s	6382		1597
	$\mathsf{EF}\,(dS \geqslant 2160)$	✗	11.5s	17680		1280
hospital billing	no deadlock	✓	20m59s	1234928	17	23147
	$\mathsf{EF}\,(p16 \wedge \neg \mathit{closed})$	✓	10m20s	669379		10654
sepsis (1)	no deadlock	✓	1m36s	139	301	44939
	$\mathsf{AG}\,(\mathit{sink} \to t_{tr} < t_{ab})$	✗	30.1s	170		22724
	$\mathsf{AG}\,(\mathit{sink} \to t_{tr} + 60 \geqslant t_{ab})$	✓	32s	153		22538
sepsis (2)	no deadlock	✓	7m24	4524	301	161242
	$\mathsf{A}\,(\neg \mathit{lacticAcid}\ \mathsf{U}\ \langle \mathrm{diagnostic}\rangle \top)$	✓	3m53s	5734		74984
board: register	no deadlock	✓	1.4s	12	7	27
board: transfer	no deadlock	✓	1.4s	27	7	51
board: discharge	no deadlock	✓	1.5s	25	6	67
	$\mathsf{AG}\,(p_2 \wedge o_1{=}207 \to \mathsf{AG}\ o_1{=}207)$	✓	1.5s	94		91
	$\mathsf{AG}\,(\mathsf{EF}\,\langle \mathrm{tra}\rangle \top \wedge \mathsf{EF}\,\langle \mathrm{his}\rangle \top)$	✓	1.5s	27		98
	$\neg \mathsf{E}\,(\mathsf{F}\,\langle \mathrm{tra}\rangle \top \wedge \mathsf{F}\,\langle \mathrm{his}\rangle \top)$	✓	1.4s	56		43
credit approval	no deadlock	✓	1.7s	470	6	230
	$\mathsf{AG}\,(\langle \mathrm{openLoan}\rangle \top \to \mathit{ver} \wedge \mathit{dec})$	✓	13.2s	14156		645
	$\mathsf{A}\,(\mathsf{F}\,(\mathit{ver} \wedge \mathit{dec}) \to \mathsf{F}\,\langle \mathrm{openLoan}\rangle \top)$	✗	3.7s	3128		316
package handling	no deadlock	✓	2.7ss	1025	16	693
	no deadlock (τ_1)	✓	2.5s	1079		398
	$\psi_{k1} = \mathsf{EF}\,\langle \mathrm{fetch}\rangle \top$	✗	2.6s	850		343
	$\psi_{k2} = \mathsf{EF}\,\langle \tau_6\rangle \top$	✗	2.4s	875		336
auction	no deadlock	✗	10.8s	1683	5	186
	$\mathsf{EF}\,(\mathrm{sold} \wedge d > 0 \wedge o \leqslant t)$	✗	6.4s	1180		79
	$\mathsf{EF}\,(b{=}1 \wedge o > t \wedge \mathsf{F}\,(\mathrm{sold} \wedge b > 1))$	✓	26.5s	4000		263

Conclusion

Summary

- for Data Petri nets with arithmetic constraints:
  verification procedures for LTL_f, CTL^*_f, data-aware soundness
- decision procedure if DPN satisfies finite summary property: new decidability results
- implemented and tested on processes from BPM

Conclusion

Summary

- for Data Petri nets with arithmetic constraints:
  verification procedures for LTL_f, CTL^*_f, data-aware soundness
- decision procedure if DPN satisfies finite summary property: new decidability results
- implemented and tested on processes from BPM

Take-home message

- finite constraint graphs are powerful tool for verification
- many relevant verification tasks are decidable for "practical" Data Petri nets

Conclusion

Summary

- for Data Petri nets with arithmetic constraints:
  verification procedures for LTL_f, CTL^*_f, data-aware soundness
- decision procedure if DPN satisfies finite summary property: new decidability results
- implemented and tested on processes from BPM

Take-home message

- finite constraint graphs are powerful tool for verification
- many relevant verification tasks are decidable for "practical" Data Petri nets

Future work

- further SMT theories, e.g. allow guards to refer to database
- discover more expressive transition guards for DPNs :)

Marco Montali

Paolo Felli

Fabio Patrizi

Bibliography: DPN Toolbox

P. Felli, M. Montali, S. Winkler
Linear-time verification of data-aware dynamic systems with arithmetic
AAAI-36, 5642-5650, 2022

P. Felli, M. Montali, S. Winkler
Soundness of data-aware processes with arithmetic conditions
CAiSE-34, LNCS 13295, 389–406, 2022

P. Felli, M. Montali, S. Winkler
CTL* model checking for data-aware dynamic systems with arithmetic
IJCAR-11, LNCS 13385, 36–56, 2022

P. Felli, M. Montali, F. Patrizi, S. Winkler
Monitoring Arithmetic Temporal Properties on Finite Traces
AAAI-37, 6346-6354, 2023

M. Montali, S. Winkler
Equivalence of Data Petri Nets with Arithmetic
FM-BPM 2023, to appear

Bibliography: Related Work

F. Mannhardt
Multi-perspective Process Mining
Ph.D. thesis, Technical University of Eindhoven, 2018

F. Mannhardt, M. de Leoni, H. A. Reijers, W. van der Aalst
Decision mining revisited: Discovering overlapping rules
CAiSE-28, LNCS 9694, 377–392, 2016