Reachability in continuous vector addition systems: from theory to practice

Michael Blondin

DIRO, Université de Montréal, Canada

LSV, ENS Cachan & CNRS, France

May 13, 2015

Reachability in continuous vector addition systems: from theory to practice

Vincent Antaki[1], Michael Blondin[1,2] & Pierre McKenzie[1,2]

[1]DIRO, Université de Montréal, Canada

[2]LSV, ENS Cachan & CNRS, France

May 13, 2015

Project

- Tool for reachability in VASS

Project

- Tool for reachability in VASS
- Relaxations to decide non reachability

Project

- Tool for reachability in VASS
- Relaxations to decide non reachability
  - Coverability: EXPSPACE/PSPACE-complete

Project

- Tool for reachability in VASS
- Relaxations to decide non reachability
  - Coverability: EXPSPACE/PSPACE-complete
  - 2-VASS: PSPACE-complete

Project

- Tool for reachability in VASS
- Relaxations to decide non reachability
  - Coverability: EXPSPACE/PSPACE-complete
  - 2-VASS: PSPACE-complete
  - 1-VASS, $\mathbb{Z}$-VASS: NP-complete

Project

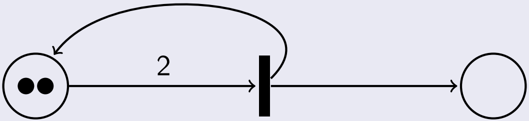
- Tool for reachability in VASS
- Relaxations to decide non reachability
  - Coverability: EXPSPACE/PSPACE-complete
  - 2-VASS: PSPACE-complete
  - 1-VASS, $\mathbb{Z}$-VASS: NP-complete
  - Continuous Petri nets: P-complete

Project

- Tool for reachability in VASS
- Relaxations to decide non reachability
  - Coverability: EXPSPACE/PSPACE-complete
  - 2-VASS: PSPACE-complete
  - 1-VASS, $\mathbb{Z}$-VASS: NP-complete
  - **Continuous Petri nets**: **P-complete**

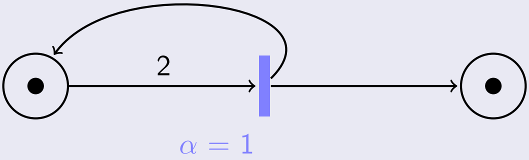
Continous Petri nets (CPN)

Transitions fired by an amount $\alpha \in \mathbb{R}_{\geq 0}$



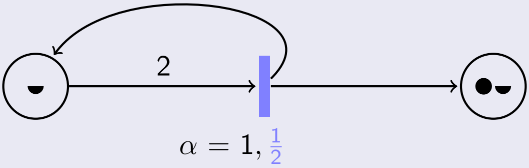
Continous Petri nets (CPN)

Transitions fired by an amount $\alpha \in \mathbb{R}_{\geq 0}$



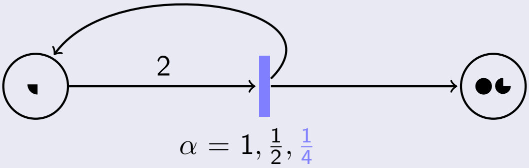
Continous Petri nets (CPN)

Transitions fired by an amount $\alpha \in \mathbb{R}_{\geq 0}$



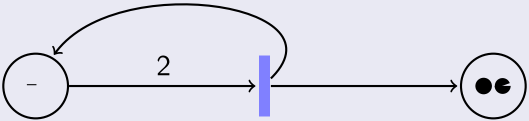
Continous Petri nets (CPN)

Transitions fired by an amount $\alpha \in \mathbb{R}_{\geq 0}$



Continous Petri nets (CPN)

Transitions fired by an amount $\alpha \in \mathbb{R}_{\geq 0}$



$\alpha = 1, \frac{1}{2}, \frac{1}{4}, \ldots, \frac{1}{2^n}$

Continuous vector addition systems with states (CVASS)

- What is a continuous VASS?

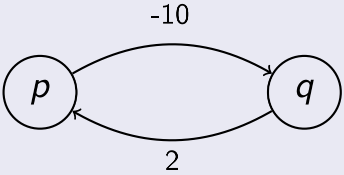
Continuous vector addition systems with states (CVASS)

- What is a continuous VASS?
- Not defined in the literature

Continuous vector addition systems with states (CVASS)

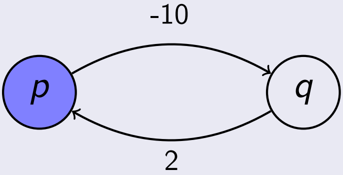
- What is a continuous VASS?
- Not defined in the literature
- Two possible definitions

CVASS with “unique states”

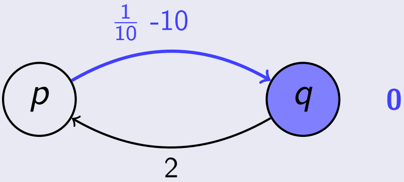


CVASS with “unique states”

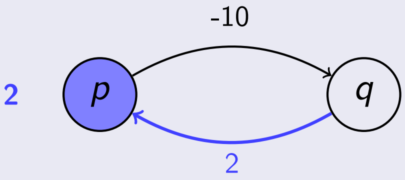
1



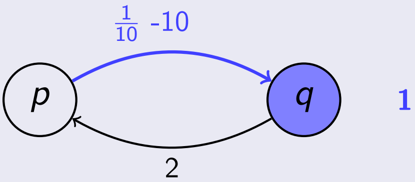
CVASS with “unique states”



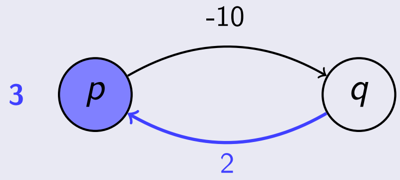
CVASS with “unique states”



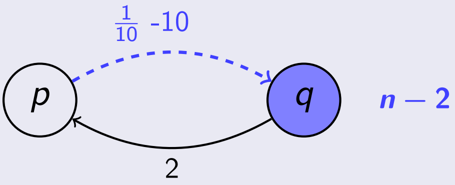
CVASS with "unique states"



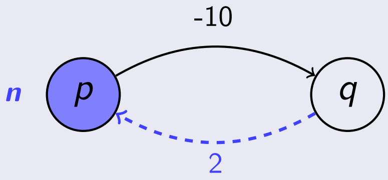
CVASS with “unique states”



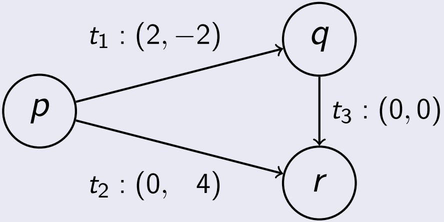
CVASS with “unique states”



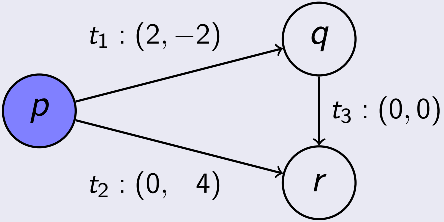
CVASS with "unique states"



CVASS with “multiple states”

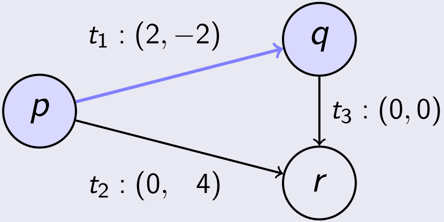


CVASS with “multiple states”



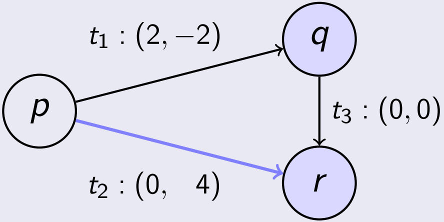
$$\begin{array}{ccc|cc} \boldsymbol{p} & \boldsymbol{q} & \boldsymbol{r} & & \\ (1, & 0, & 0, & 0, & 1) \end{array}$$

CVASS with “multiple states”



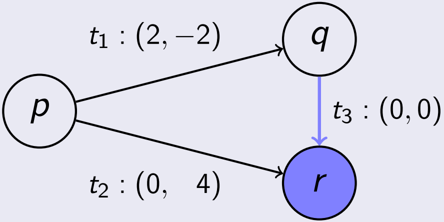
$$\begin{array}{l}
\quad \boldsymbol{p} \;\; \boldsymbol{q} \;\; \boldsymbol{r} \\
(1, 0, 0, \;\vdots\; 0, 1) \xrightarrow{\frac{1}{2} t_1} \\
(\frac{1}{2}, \frac{1}{2}, 0, \;\vdots\; 1, 0)
\end{array}$$

CVASS with “multiple states”



$$
\begin{array}{l}
\quad \boldsymbol{p} \;\; \boldsymbol{q} \;\; \boldsymbol{r} \\
(1, 0, 0, \;\vdots\; 0, 1) \xrightarrow{\frac{1}{2} t_1} \\
(\frac{1}{2}, \frac{1}{2}, 0, \;\vdots\; 1, 0) \xrightarrow{\frac{1}{2} t_2} \\
(0, \frac{1}{2}, \frac{1}{2}, \;\vdots\; 1, 2)
\end{array}
$$

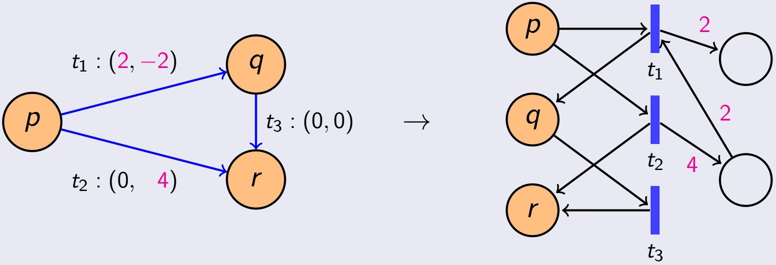
CVASS with “multiple states”



$$
\begin{array}{l}
\quad\;\; \boldsymbol{p} \;\; \boldsymbol{q} \;\; \boldsymbol{r} \\
(1, 0, 0, \mid 0, 1) \xrightarrow{\frac{1}{2} t_1} \\
(\tfrac{1}{2}, \tfrac{1}{2}, 0, \mid 1, 0) \xrightarrow{\frac{1}{2} t_2} \\
(0, \tfrac{1}{2}, \tfrac{1}{2}, \mid 1, 2) \xrightarrow{\frac{1}{2} t_3} \\
(0, 0, 1, \mid 1, 2)
\end{array}
$$

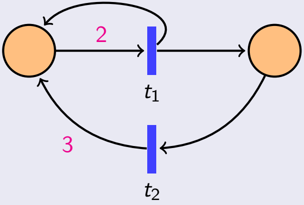
CVASS with "multiple states" $\leq$ CPN

Usual transformation, straightforward proof

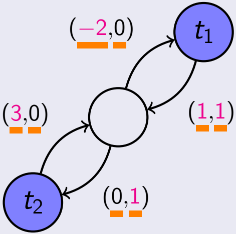


CPN $\leq$ CVASS with "multiple states"

Usual transformation, less straightforward proof



$\longrightarrow$



Our implementation

$\vdots$

$T' \leftarrow T$
while $T' \neq \emptyset$ **do**
  $nbsol \leftarrow 0$; **sol** $\leftarrow \mathbf{0}$
  for $t \in T'$ **do**
    solve $\exists ? \mathbf{v}\ \mathbf{v} \geq \mathbf{0} \wedge \mathbf{v}[t] > 0 \wedge \boldsymbol{C}_{P \times T'} \mathbf{v} = \boldsymbol{m} - \boldsymbol{m}_0$
    if $\exists \mathbf{v}$ **then** $nbsol \leftarrow nbsol + 1$; **sol** $\leftarrow$ **sol** $+ \mathbf{v}$
  end
  if $nbsol = 0$ **then return false else sol** $\leftarrow \frac{1}{nbsol}$**sol**

$\vdots$

- Fraca & Haddad PN'13

Our implementation

$\vdots$

$T' \leftarrow T$
while $T' \neq \emptyset$ **do**
  $nbsol \leftarrow 0$; **sol** $\leftarrow \mathbf{0}$
  for $t \in T'$ **do**
    solve $\exists ?\mathbf{v}\ \mathbf{v} \geq \mathbf{0} \wedge \mathbf{v}[t] > 0 \wedge \boldsymbol{C}_{P \times T'}\mathbf{v} = \boldsymbol{m} - \boldsymbol{m}_0$
    if $\exists \mathbf{v}$ **then** $nbsol \leftarrow nbsol + 1$; **sol** $\leftarrow$ **sol** $+ \mathbf{v}$
  end
  if $nbsol = 0$ **then return false else sol** $\leftarrow \frac{1}{nbsol}$**sol**

$\vdots$

- Fraca & Haddad PN'13
- Reachability in CPN $\in$ P

Our implementation

$\vdots$

$T' \leftarrow T$
while $T' \neq \emptyset$ **do**
  $nbsol \leftarrow 0$; $\mathbf{sol} \leftarrow \mathbf{0}$
  for $t \in T'$ **do**
    solve $\exists ?\mathbf{v}\ \mathbf{v} \geq \mathbf{0} \wedge \mathbf{v}[t] > 0 \wedge \boldsymbol{C}_{P \times T'}\mathbf{v} = \boldsymbol{m} - \boldsymbol{m}_0$
    if $\exists\mathbf{v}$ **then** $nbsol \leftarrow nbsol + 1$; $\mathbf{sol} \leftarrow \mathbf{sol} + \mathbf{v}$
  end
  if $nbsol = 0$ **then return false else** $\mathbf{sol} \leftarrow \frac{1}{nbsol}\mathbf{sol}$

$\vdots$

```
t1   = np.array(range(0, n2))
b_eq = np.array(m - m0)

while t1.size != 0:
  l           = t1.size
  nbsol, sol = 0, np.zeros(l, dtype=Fraction)
  A_eq        = incident(subnet(net, t1))

  for t in t1:
    objective_vector = [objective(t, x) for x in range(0, l)]
    result           = solve_qsopt(objective_vector, A_eq, b_eq, t)

    if result is not None:
      nbsol += 1
      sol   += result
```

- Fraca & Haddad PN'13
- Reachability in CPN $\in$ P
- python™ with NumPy

Our implementation

⋮

$T' \leftarrow T$
while $T' \neq \emptyset$ **do**
  $nbsol \leftarrow 0$; $\mathbf{sol} \leftarrow \mathbf{0}$
  for $t \in T'$ **do**
    solve $\exists ?\mathbf{v}\ \mathbf{v} \geq \mathbf{0} \wedge \mathbf{v}[t] > 0 \wedge \boldsymbol{C}_{P \times T'}\mathbf{v} = \boldsymbol{m} - \boldsymbol{m}_0$
    if $\exists\mathbf{v}$ **then** $nbsol \leftarrow nbsol + 1$; $\mathbf{sol} \leftarrow \mathbf{sol} + \mathbf{v}$
  end
  if $nbsol = 0$ **then return false else** $\mathbf{sol} \leftarrow \frac{1}{nbsol}\mathbf{sol}$

⋮

```
t1   = np.array(range(0, n2))
b_eq = np.array(m - m0)

while t1.size != 0:
  l          = t1.size
  nbsol, sol = 0, np.zeros(l, dtype=Fraction)
  A_eq       = incident(subnet(net, t1))

  for t in t1:
    objective_vector = [objective(t, x) for x in range(0, l)]
    result           = solve_qsopt(objective_vector, A_eq, b_eq, t)

    if result is not None:
      nbsol += 1
      sol   += result
```

- Fraca & Haddad PN'13
- Reachability in CPN $\in$ P
- python™ with NumPy
- 299 lines of code

  (215 code + 84 docstring)

Polynomial time algorithm (Fraca & Haddad PN'13)

Algorithm 2: Decision algorithm for reachability

`Reachable`$(\langle \mathcal{N}, \boldsymbol{m}_0\rangle, \boldsymbol{m})$: status
Input: a CPN system $\langle \mathcal{N}, \boldsymbol{m}_0\rangle$, a marking $\boldsymbol{m}$
Output: the reachability status of $\boldsymbol{m}$
Output: the Parikh image of a witness in the positive case
Data: *nbsol*: integer; $\mathbf{v}, \mathbf{sol}$: vectors; T': subset of transitions

if $\boldsymbol{m} = \boldsymbol{m}_0$ **then return** (**true**,**0**)
$T' \leftarrow T$
while $T' \neq \emptyset$ **do**
    $nbsol \leftarrow 0$; $\mathbf{sol} \leftarrow \mathbf{0}$
    for $t \in T'$ **do**
        solve $\exists ? \mathbf{v}\ \mathbf{v} \geq \mathbf{0} \wedge \mathbf{v}[t] > 0 \wedge \boldsymbol{C}_{P \times T'}\mathbf{v} = \boldsymbol{m} - \boldsymbol{m}_0$
        if $\exists \mathbf{v}$ **then** $nbsol \leftarrow nbsol + 1$; $\mathbf{sol} \leftarrow \mathbf{sol} + \mathbf{v}$
    end
    if $nbsol = 0$ **then return false else** $\mathbf{sol} \leftarrow \frac{1}{nbsol}\mathbf{sol}$
    $T' \leftarrow [\![\mathbf{sol}]\!]$
    $T' \leftarrow T' \cap$ `maxFS`$(\mathcal{N}_{T'}, \boldsymbol{m}_0[{}^\bullet T'^\bullet])$
    $T' \leftarrow T' \cap$ `maxFS`$(\mathcal{N}^{-1}_{T'}, \boldsymbol{m}[{}^\bullet T'^\bullet])$ `/* deleted for lim-reachability */`
    if $T' = [\![\mathbf{sol}]\!]$ **then return** (**true**,**sol**)
end
return false

Polynomial time algorithm (Fraca & Haddad PN'13)

Algorithm 2: Decision algorithm for reachability

`Reachable`$(\langle \mathcal{N}, \boldsymbol{m}_0 \rangle, \boldsymbol{m})$: status

Input: a CPN system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, a marking $\boldsymbol{m}$

Output: the reachability status of $\boldsymbol{m}$

Output: the Parikh image of a witness in the positive case

Data: *nbsol*: integer; $\mathbf{v}, \mathbf{sol}$: vectors; T': subset of transitions

if $\boldsymbol{m} = \boldsymbol{m}_0$ **then return** (**true**,**0**)
$T' \leftarrow T$
while $T' \neq \emptyset$ **do**
    $nbsol \leftarrow 0$; $\mathbf{sol} \leftarrow \mathbf{0}$
    for $t \in T'$ **do**
        solve $\exists ?\mathbf{v}\ \mathbf{v} \geq \mathbf{0} \wedge \mathbf{v}[t] > 0 \wedge \boldsymbol{C}_{P \times T'} \mathbf{v} = \boldsymbol{m} - \boldsymbol{m}_0$
        if $\exists \mathbf{v}$ **then** $nbsol \leftarrow nbsol + 1$; $\mathbf{sol} \leftarrow \mathbf{sol} + \mathbf{v}$
    end
    if $nbsol = 0$ **then return false else** $\mathbf{sol} \leftarrow \frac{1}{nbsol}\mathbf{sol}$
    $T' \leftarrow [\![\mathbf{sol}]\!]$
    $T' \leftarrow T' \cap \mathtt{maxFS}(\mathcal{N}_{T'}, \boldsymbol{m}_0[{}^\bullet T'^\bullet])$
    $T' \leftarrow T' \cap \mathtt{maxFS}(\mathcal{N}^{-1}_{T'}, \boldsymbol{m}[{}^\bullet T'^\bullet])$ `/* deleted for lim-reachability */`
    if $T' = [\![\mathbf{sol}]\!]$ **then return** (**true**,**sol**)
end
return false

Relatively easy with `NumPy`

Polynomial time algorithm (Fraca & Haddad PN'13)

Algorithm 2: Decision algorithm for reachability

`Reachable`$(\langle \mathcal{N}, \boldsymbol{m}_0 \rangle, \boldsymbol{m})$: status
Input: a CPN system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, a marking $\boldsymbol{m}$
Output: the reachability status of $\boldsymbol{m}$
Output: the Parikh image of a witness in the positive case
Data: *nbsol*: integer; $\mathbf{v}, \mathbf{sol}$: vectors; T': subset of transitions

if $\boldsymbol{m} = \boldsymbol{m}_0$ **then return** (**true**,**0**)
$T' \leftarrow T$
while $T' \neq \emptyset$ **do**
  $nbsol \leftarrow 0$; $\mathbf{sol} \leftarrow \mathbf{0}$
  for $t \in T'$ **do**
    $\boxed{\textbf{solve } \exists ? \mathbf{v}\ \mathbf{v} \geq \mathbf{0} \wedge \mathbf{v}[t] > 0 \wedge \boldsymbol{C}_{P \times T'} \mathbf{v} = \boldsymbol{m} - \boldsymbol{m}_0}$ A bit trickier
    if $\exists \mathbf{v}$ **then** $nbsol \leftarrow nbsol + 1$; $\mathbf{sol} \leftarrow \mathbf{sol} + \mathbf{v}$
  end
  if $nbsol = 0$ **then return false else** $\mathbf{sol} \leftarrow \frac{1}{nbsol}\mathbf{sol}$
  $T' \leftarrow [\![\mathbf{sol}]\!]$
  $T' \leftarrow T' \cap$ `maxFS`$(\mathcal{N}_{T'}, \boldsymbol{m}_0[{}^\bullet T'^\bullet])$
  $T' \leftarrow T' \cap$ `maxFS`$(\mathcal{N}_{T'}^{-1}, \boldsymbol{m}[{}^\bullet T'^\bullet])$ `/* deleted for lim-reachability */`
  if $T' = [\![\mathbf{sol}]\!]$ **then return** (**true**,**sol**)
end
return false

System of linear inequalities

$\exists \mathbf{x} \in \mathbb{R}^k$ such that $\mathbf{x} \geq \mathbf{0}$, $\mathbf{x}_t > 0$ and $A\mathbf{x} = \mathbf{b}$?

System of linear inequalities

$\vdots$

$\exists \mathbf{x} \in \mathbb{R}^k$ such that $\mathbf{x} \geq \mathbf{0}$, $\mathbf{x}_t > 0$ and $A\mathbf{x} = \mathbf{b}$?

$\vdots$

Without this condition, could simply use simplex

Handling the strict inequality

1. Solve

$$\begin{array}{ll} \text{Maximize} & \mathbf{x}_t \\ \text{Subject to} & A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \end{array}$$

Handling the strict inequality

1. Solve

$$\begin{array}{ll} \text{Maximize} & \mathbf{x}_t \\ \text{Subject to} & A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \end{array}$$

2. - If $\mathbf{x}_t > 0$, return $\mathbf{x}$

Handling the strict inequality

1. Solve

$$\begin{array}{ll} \text{Maximize} & \mathbf{x}_t \\ \text{Subject to} & A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \end{array}$$

2.
   - If $\mathbf{x}_t > 0$, return $\mathbf{x}$
   - If $\mathbf{x}_t = 0$, return `"no solution"`

Handling the strict inequality

1. Solve

$$\begin{array}{ll} \text{Maximize} & \mathbf{x}_t \\ \text{Subject to} & A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \end{array}$$

2.
   - If $\mathbf{x}_t > 0$, return $\mathbf{x}$
   - If $\mathbf{x}_t = 0$, return `"no solution"`
   - If no solution, return `"no solution"`

Handling the strict inequality

1. Solve

$$\begin{array}{ll} \text{Maximize} & \mathbf{x}_t \\ \text{Subject to} & A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0} \end{array}$$

2.
   - If $\mathbf{x}_t > 0$, return $\mathbf{x}$
   - If $\mathbf{x}_t = 0$, return `"no solution"`
   - If no solution, return `"no solution"`
   - If unbounded, continue

Handling the strict inequality

3. Solve

$$\begin{array}{ll} \text{Minimize} & \mathbf{x}_t \\ \text{Subject to} & A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0},\ \mathbf{x}_t \geq 1 \end{array}$$

Handling the strict inequality

3. Solve

$$\begin{array}{ll} \text{Minimize} & \mathbf{x}_t \\ \text{Subject to} & A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0},\ \mathbf{x}_t \geq 1 \end{array}$$

return $\mathbf{x}$

Simplex implementations

- Usually in floating-point arithmetic

Simplex implementations

- Usually in floating-point arithmetic
- Error-prone, even worse with $2|T|^2$ resolutions

Simplex implementations

- Usually in floating-point arithmetic
- Error-prone, even worse with $2|T|^2$ resolutions
- Interested in non reachability, no certificate to verify answer

Simplex implementations

- Usually in floating-point arithmetic
- Error-prone, even worse with $2|T|^2$ resolutions
- Interested in non reachability, no certificate to verify answer

Current solution

`QSopt-Exact`: exact solver from

Exact solutions to linear programming problems

David L. Applegate [a] William Cook [b] Sanjeeb Dash [c] Daniel G. Espinoza [d,*]

Open questions

- Floating-point solver + testing certificates (Farkas' lemma, reconstruct simplex tableaux in $\mathbb{Q}$)

Open questions

- Floating-point solver + testing certificates (Farkas' lemma, reconstruct simplex tableaux in $\mathbb{Q}$)
- Reachability in CVASS with “unique states”?

Open questions

- Floating-point solver + testing certificates (Farkas' lemma, reconstruct simplex tableaux in $\mathbb{Q}$)
- Reachability in CVASS with "unique states"?
- Any use for CVASS with "unique states"?

Further work

- Test other solvers

Further work

- Test other solvers
- Benchmarks

Further work

- Test other solvers
- Benchmarks
- Next modules

Thank you! Merci! Danke!