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\textsuperscript{1}, Michael Blondin\textsuperscript{12} & Pierre McKenzie\textsuperscript{12}

\textsuperscript{1}DIRO, Université de Montréal, Canada

\textsuperscript{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 $\alpha \in \mathbb{R}_{\geq 0}$
Continuous Petri nets (CPN)

Transitions fired by an amount $\alpha \in \mathbb{R}_{\geq 0}$
Continuous Petri nets (CPN)

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

$\alpha = 1, \frac{1}{2}$
Continuous Petri nets (CPN)

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

$\alpha = 1, \frac{1}{2}, \frac{1}{4}$
Continuous 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

\[ p \]

\[ q \]

-10

2

5 / 16
CVASS with “unique states”
CVASS with “unique states”

Graph: Two nodes labeled p and q, connected by edges labeled 2 and -10.
CVASS with “unique states”
CVASS with “unique states”

-10

3

2

p

q
CVASS with “unique states”

Motivation
Continuous VASS
Tool
Conclusion

Unique states
Multiple states
Equivalence with CPN

$p$
$q$
\[ \frac{1}{10} \quad -10 \]
2

$n - 2$
CVASS with “unique states”
CVASS with “multiple states”

\[ t_1 : (2, -2) \]
\[ t_2 : (0, 4) \]
\[ t_3 : (0, 0) \]
CVASS with “multiple states”

$p$ with states and transitions:

- $t_1 : (2, -2)$
- $t_2 : (0, 4)$
- $t_3 : (0, 0)$

$p$ transitions to $q$:

$q$ with states:

$p q r$

$(1, 0, 0, 0, 1)$
CVASS with "multiple states"

$p$ \(\xrightarrow{t_1} (2, -2)\) $q$

$q$ \(\xrightarrow{t_3} (0, 0)\) $r$

$t_2 : (0, 4)$

$p$ \(\xrightarrow{\frac{1}{2}t_1} (1, 0, 0, 0, 1)\)

$p$ \(\xrightarrow{t_2} (0, 0, 0, 1, 0)\)

$p$ \(\xrightarrow{t_3} (\frac{1}{2}, \frac{1}{2}, 0, 1, 0)\)
CVASS with “multiple states”

$p \ x \ r$

$t_1 : (2, -2)$

$t_2 : (0, 4)$

$t_3 : (0, 0)$

$p q r$

$(1, 0, 0, 0, 1) \xrightarrow{\frac{1}{2} t_1} (\frac{1}{2}, \frac{1}{2}, 0, 1, 0) \xrightarrow{\frac{1}{2} t_2} (0, \frac{1}{2}, \frac{1}{2}, 1, 2)$
CVASS with “multiple states”

$\begin{align*}
  t_1 & : (2, -2) \\
  t_2 & : (0, 4) \\
  t_3 & : (0, 0)
\end{align*}$

$p \ x \ q \ x \ r$

$p \ x \ q \ x \ r$

\begin{align*}
  & (1, 0, 0, \rightarrow 0, 1) \quad \frac{1}{2} t_1 \\
  & (\frac{1}{2}, \frac{1}{2}, 0, \rightarrow 1, 0) \quad \frac{1}{2} t_2 \\
  & (0, \frac{1}{2}, \frac{1}{2}, \rightarrow 1, 2) \quad \frac{1}{2} t_3 \\
  & (0, 0, 1, \rightarrow 1, 2)
\end{align*}
CVASS with “multiple states” ≤ CPN

Usual transformation, straightforward proof

\[ t_1 : (2, -2) \]
\[ t_2 : (0, 4) \]
\[ t_3 : (0, 0) \]
CPN $\leq$ CVASS with “multiple states”

Usual transformation, less straightforward proof
Our implementation

\( T' \leftarrow T \)
\[
\text{while } T' \neq \emptyset \text{ do } \\
\quad \text{nbsol} \leftarrow 0; \ \text{sol} \leftarrow 0 \\
\quad \text{for } t \in T' \text{ do } \\
\quad \quad \text{solve } \exists v \ v \geq 0 \land v[t] > 0 \land C_{P \times T'} v = m - m_0 \\
\quad \quad \quad \text{if } \exists v \text{ then } \text{nbsol} \leftarrow \text{nbsol} + 1; \ \text{sol} \leftarrow \text{sol} + v \\
\quad \quad \text{end} \\
\quad \text{if } \text{nbsol} = 0 \text{ then return false else sol} \leftarrow \frac{1}{\text{nbsol}} \text{sol} \\
\]

- Fraca & Haddad PN’13
Our implementation

\[ T' \leftarrow T \]

while \( T' \neq \emptyset \) do

\[ nbsol \leftarrow 0; \ sol \leftarrow 0 \]

for \( t \in T' \) do

solve \( \exists \mathbf{v} \mathbf{v} \geq 0 \cap \mathbf{v}[t] \geq 0 \cap C_{P \times T'} \mathbf{v} = \mathbf{m} - \mathbf{m}_0 \)

if \( \exists \mathbf{v} \) then \( nbsol \leftarrow nbsol + 1; \ sol \leftarrow sol + \mathbf{v} \)

end

if \( nbsol = 0 \) then return false else \( sol \leftarrow \frac{1}{nbsol} \cdot sol \)

\[ \text{...} \]

- Fraca & Haddad PN’13
- Reachability in CPN \( \in P \)
whose size is possibly exponential. For each item, say $T'$, the algorithm would check in polynomial time (1) whether $m \in FS(\mathcal{M})$ and $C_{P \times T'} \cdot v \leq m - m_0$ holds. By definition, vector sol which is a barycenter of solutions is also a solution.

Our implementation

```
T' ← T
while T' ≠ ∅ do
  nbsol ← 0; sol ← 0
  for t ∈ T' do
    solve ∃v v ≥ 0 ∧ v[t] > 0 ∧ C_{P \times T'} v = m - m_0
    if ∃v then nbsol ← nbsol + 1; sol ← sol + v
  end
  if nbsol = 0 then return false else sol ← \frac{1}{nbsol} \cdot 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 ∈ P
- Python with NumPy
Our implementation

\[
T' \leftarrow T
\]

while \( T' \neq \emptyset \) do

\[
nbsol \leftarrow 0; \ sol \leftarrow 0
\]

for \( t \in T' \) do

\[
solve \exists v \ v \geq 0 \land v[t] > 0 \land C_{P \times T'} v = m - m_0
\]

if \( \exists v \) then \( nbsol \leftarrow nbsol + 1; \ sol \leftarrow sol + v \)

end

if \( nbsol = 0 \) then return false else \( sol \leftarrow \frac{1}{nbsol} \ sol \)

\[
\]

\[
t1 = \text{np.array}(\text{range}(0, n2))
\]

\[
b_{eq} = \text{np.array}(m - m_0)
\]

while \( t1.\text{size} \neq 0 \) do

\[
l = t1.\text{size}
\]

\[
nbsol, sol = 0, \text{np.zeros}(l, \text{dtype} = \text{Fraction})
\]

A_eq = \text{incident} (\text{subnet}(\text{net}, t1))

for \( t \) in \( t1 \):

\[
\text{objective_vector} = [\text{objective}(t, x) \text{ for } x \text{ in } \text{range}(0, l)]
\]

\[
\text{result} = \text{solve_qsopt} (\text{objective_vector}, A_{eq}, b_{eq}, t)
\]

if result is not None:

\[
nbsol += 1
\]

\[
sol += \text{result}
\]

\[
\]

- Fraca & Haddad PN’13
- Reachability in CPN \( \in \mathbb{P} \)
- \( \mathbb{Python} \) with NumPy
- 299 lines of code
  (215 code + 84 docstring)
Algorithm 2: Decision algorithm for reachability

```
Algorithm 2: Decision algorithm for reachability

Reachable(⟨N, m0⟩, m): status
Input: a CPN system ⟨N, m0⟩, a marking m
Output: the reachability status of m
Output: the Parikh image of a witness in the positive case
Data: nbsol: integer; v, sol: vectors; T′: subset of transitions

1 if m = m0 then return (true, 0)
2 T′ ← T
3 while T′ ≠ ∅ do
4     nbsol ← 0; sol ← 0
5     for t ∈ T′ do
6         solve ∃?v v ≥ 0 ∧ v[t] > 0 ∧ C_{P×T′}v = m − m0
7         if ∃v then nbsol ← nbsol + 1; sol ← sol + v
8     end
9     if nbsol = 0 then return false else sol ← 1/nbsol, sol
10    T′ ← [sol]
11    T′ ← T′ ∩ maxFS(N_{T′}, m_0[*T′*])
12    T′ ← T′ ∩ maxFS(N_{T′}^{-1}, m[*T′*]) /* deleted for lim-reachability */
13    if T′ = [sol] then return (true, sol)
14 end
15 return false
```
Polynomial time algorithm (Fraca & Haddad PN’13)
Polynomial time algorithm (Fraca & Haddad PN’13)

**Algorithm 2:** Decision algorithm for reachability

```plaintext
Reachable(⟨N, m₀⟩, m): status

Input: a CPN system ⟨N, m₀⟩, a marking m
Output: the reachability status of m

Data: nbsol: integer; v, sol: vectors; T': subset of transitions

1. if m = m₀ then return (true, 0)
2. T' ← T
3. while T' ≠ ∅ do
4.   nbsol ← 0; sol ← 0
5.   for t ∈ T' do
6.     if ∃v v ≥ 0 ∧ v[t] > 0 ∧ C_{P×T'}v = m - m₀
7.     then solve ∃?v v ≥ 0 ∧ v[t] > 0 ∧ C_{P×T'}v = m - m₀
8.     if ∃v then nbsol ← nbsol + 1; sol ← sol + v
9.   end
10. if nbsol = 0 then return false else sol ← \frac{1}{nbsol}[sol]
11. T' ← [sol]
12. T' ← T' ∩ maxFS(N_{T'}, m₀[•T'*])
13. if T' = [sol] then return (true,sol)
14. end
15. return false
```

A bit trickier
System of linear inequalities

\[ \exists x \in \mathbb{R}^k \text{ such that } x \geq 0, \ x_t > 0 \text{ and } Ax = b? \]
### System of linear inequalities

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( \exists \mathbf{x} \in \mathbb{R}^k ) such that ( \mathbf{x} \geq 0, \ x_t &gt; 0 ) and ( A\mathbf{x} = \mathbf{b} )?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Without **this condition**, could simply use simplex
Handling the strict inequality

1. Solve

$$\text{Maximize} \quad x_t$$

$$\text{Subject to} \quad Ax = b, \quad x \geq 0$$
Handling the strict inequality

1. Solve

   Maximize $x_t$
   Subject to $Ax = b$, $x \geq 0$

2. If $x_t > 0$, return $x$
Handling the strict inequality

1. Solve

Maximize $x_t$
Subject to $Ax = b$, $x \geq 0$

2. If $x_t > 0$, return $x$
   If $x_t = 0$, return "no solution"
Handling the strict inequality

1. Solve
   \[
   \begin{align*}
   \text{Maximize} & \quad x_t \\
   \text{Subject to} & \quad Ax = b, \ x \geq 0
   \end{align*}
   \]

2. If \( x_t > 0 \), return \( x \)
   If \( x_t = 0 \), return \( \text{"no solution"} \)
   If no solution, return \( \text{"no solution"} \)
Handling the strict inequality

1. Solve

   Maximize \( x_t \)
   Subject to \( Ax = b, \ x \geq 0 \)

2. If \( x_t > 0 \), return \( x \)
   - If \( x_t = 0 \), return "no solution"
   - If no solution, return "no solution"
   - If unbounded, continue
Handling the strict inequality

3. Solve

\begin{align*}
\text{Minimize} & \quad x_t \\
\text{Subject to} & \quad Ax = b, \ x \geq 0, \ x_t \geq 1
\end{align*}
Handling the strict inequality

3. Solve

\[
\begin{align*}
\text{Minimize} & \quad x_t \\
\text{Subject to} & \quad Ax = b, \ x \geq 0, \ x_t \geq 1
\end{align*}
\]

return 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\textsuperscript{a} William Cook\textsuperscript{b} Sanjeeb Dash\textsuperscript{c} Daniel G. Espinoza\textsuperscript{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!