KReach: A Tool for Reachability in Petri Nets

Alex Dixon
Ranko Lazić

The Reachability Problem

INPUT:

- Petri net **N** = (**P, T**)
- Initial marking m_i
- Target marking m_t

OUTPUT:

- **Reachable** if $m_i \rightsquigarrow m_t$ in ***N***
- **NotReachable** otherwise

Complexity

- Lower bound – Not Elementary [1]
- Upper bound - Ackermannian [2]
- Coverability is EXPSPACE-Complete [3]

---

[1] Czerwinski et al., STOC 2019
[2] Leroux & Schmitz, LICS 2019
[3] Rackoff, TCS 1978

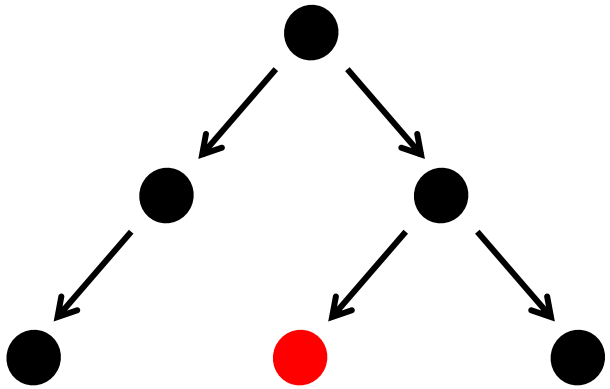
Kosaraju's Algorithm[4]

- Builds on the work of Mayr, Sacerdote, Tenney
- Is a complete algorithm for deciding reachability
- **Can be implemented and tested** (this work)

---

[4] Kosaraju, STOC 1982

The algorithm, quickly



- **Search**
- Through the space of **decompositions**
- Decompositions are computed using a structural predicate called the θ condition
- Eventually:
  - We find a decomposition that fulfils θ (Reachable! ☺)
  - We **exhaust** the tree (Not reachable. ☹)

The Tool

Available at:

https://github.com/dixonary/kosaraju

Implemented in the **Haskell** programming language

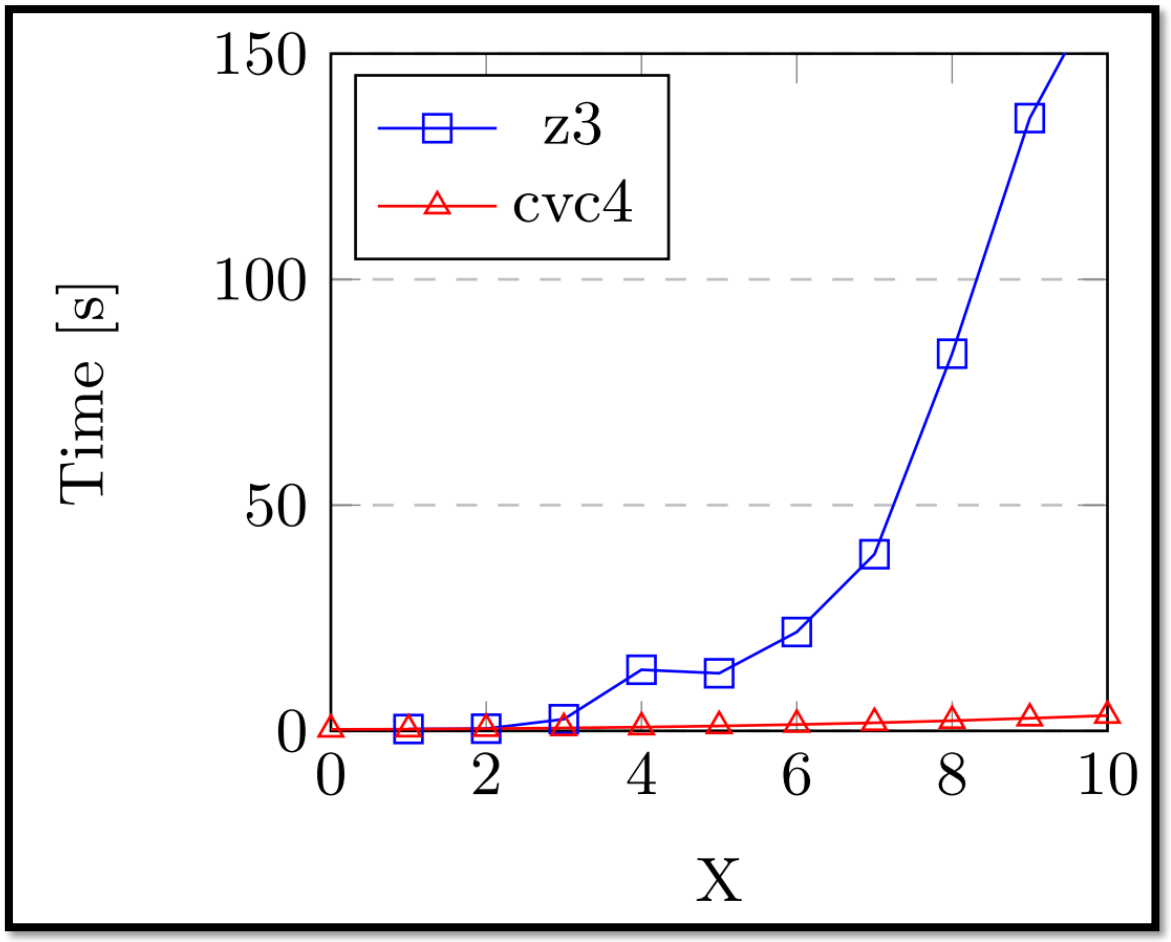
Related VASS programming libraries:

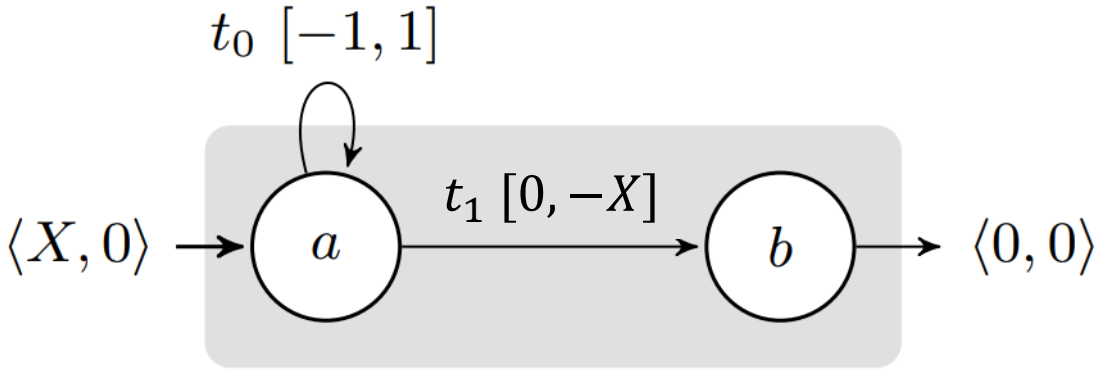
https://github.com/dixonary/vass

https://github.com/dixonary/karp-miller

Results

Parameterised Results





The program ran in exponential time in z3, but cvc4 remained linear

(time to check satisfiability of θ is linear in X)

Coverability → Reachability

Reachability problems are hard to come by

Reduction from coverability: we can cover $\vec{m}$ in some net **iff** we can reach $\vec{0}$ in a modified version of the same net

=> We can test the reachability decision procedure using coverability benchmarks from the literature

The fast and the curious

Notably, KReach was able to rule out coverability on some safe Petri nets as fast, or faster, than several leading coverability solvers.

KReach was able to rule out coverability based on few decompositions.

Instance	Outcome	MIST (s)	Qcover (s)	Icover (s)	**KReach** (s)
Kanban	safe	404	TLE	TLE	1
Bingham_h150	safe	TLE	TLE	TLE	533

Instances which required a lot of decomposition took much longer to evaluate with KReach than other tools.

---

* Within the given time and memory constraints [4GB, 1hr]