The Reachability Problem for Petri Nets Is Not Elementary

Lejun Min
Ziyi Cai

ACM Class 2019, Zhiyuan College, Shanghai Jiao Tong University

June 3, 2021

Intro

Intro

Petri net and vector addition system

Vector addition system

Vector addition system (VAS) is an alternative presentation of Petri net.
For ease of presentation, we will work with VAS instead of Petri net.
A VAS is a finite set $V \subseteq \mathbb{Z}^d$ for $d \geq 1$.
Given a vector $u \in \mathbb{N}^d$, the vector $u + v$ can be reached, in one transition, if $v \in V$ and $u + v \in \mathbb{N}^d$.

Intro

Reachability: lower bound and upper bound

Reachability and coverability problem

Reachability of VAS

Given a VAS V, two vectors u and v in $\mathbb{N}^d$.
Decide that if v is reachable from u by finite steps of transitions.

Coverability of VAS

Given a VAS V, two vectors u and v in $\mathbb{N}^d$.
Decide that if there exists a vector w with $w_i \geq v_i \; \forall 1 \leq i \leq d$, so that w is reachable from u by finite steps of transitions.

ELEMENTARY

$$\begin{aligned}\text{ELEMENTARY} &= \bigcup_{k\in\mathbb{N}} k - \text{EXP} \\ &= \text{DTIME}\left(2^n\right) \cup \text{DTIME}\left(2^{2^n}\right) \cup \text{DTIME}\left(2^{2^{2^n}}\right) \cdots\end{aligned}$$

ELEMENTARY is far beyond EXPSPACE and some other complexity classes like NP which we might be familiar with.

Fast-growing complexity classes

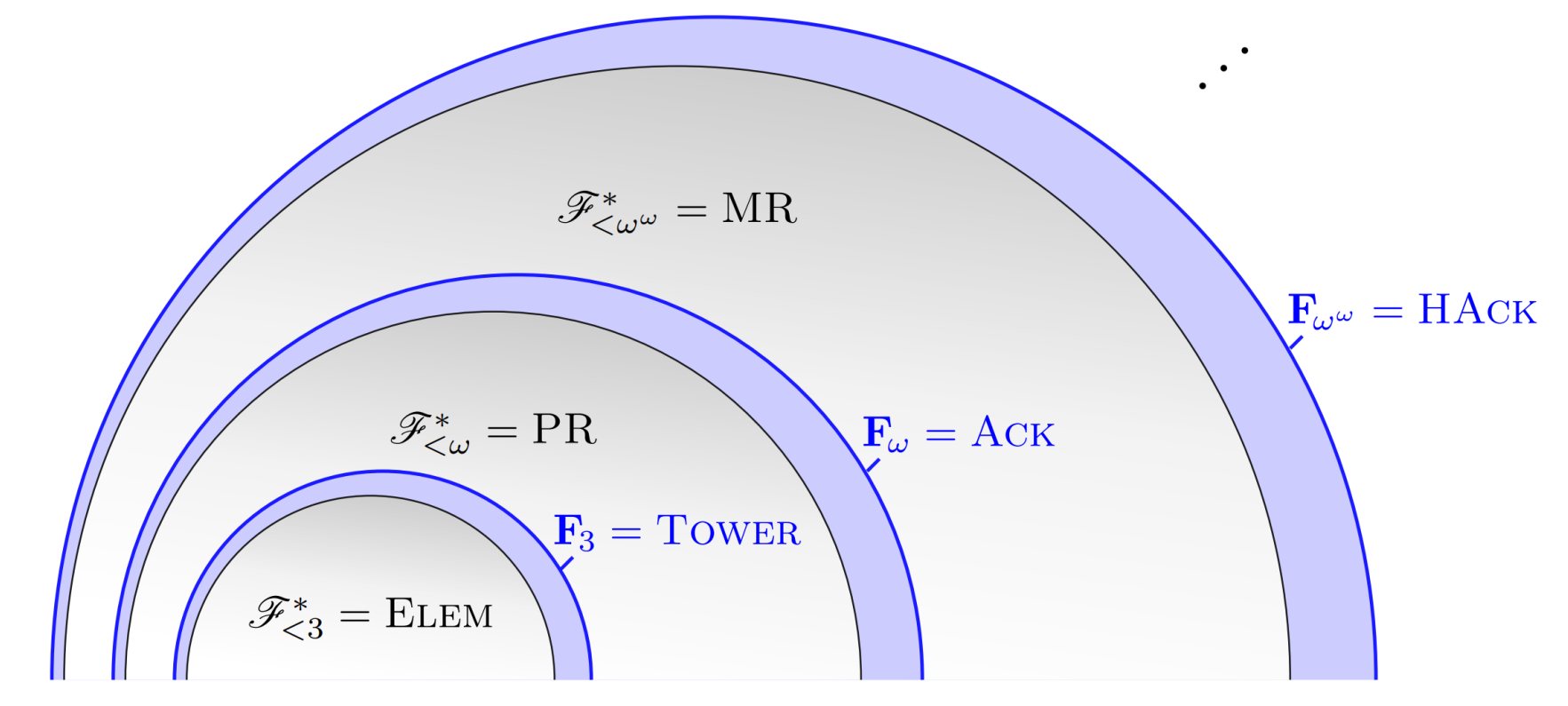

Figure: Sylvain Schmitz. Complexity hierarchies beyond elementary.

$$Tower(n) = \left(\underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n} \right), \mathrm{TOWER} = \bigcup_{p \in \mathscr{F}_{<3}} \mathrm{DTIME}\left(Tower(p(n))\right)$$

Reachability: lower bound and upper bound

- 1976 EXPSPACE lower bound [Lipton]
- 1977 (incomplete) decidability [Sacerdote, Tenney]
- 1981 decidability [Mayr]
- 1982 decidability - simplified proof [Kosaraju]
- 2015 first upper bound $\mathbb{F}_{\omega^3}$ [Leroux, Schmitz]
- 2019 Ackermannian upper bound $\mathbb{F}_{\omega}$ [Leroux, Schmitz]
- 2019 *Tower* lower bound $\mathbb{F}_3$ [Czerwiński, Lasota, Lazić, Leroux, Mazowiecki]
- 2021 Ackermannian lower bound $\mathbb{F}_{\omega}$ [Czerwiński, Orlikowski]

Intro

The crux of Lipton's construction

Obtaining the EXPSPACE lower bound

If we reduce an EXPSPACE-complete problem to the reachability problem of VAS, then the reachability is at least EXPSPACE hard,
The EXPSPACE-complete problem we chose is a restricted halting problem of a finite-state machine with counters. So we use VAS to simulate such a machine by supporting the following two operations:

- Increase x by 1;
- Decrease x by 1; however if $x = 0$, then the non-deterministic computation dies.

Bad news: VAS does not directly support **zero?** x, i.e. testing if a counter is 0, which is supported by the FSM.

Main idea of Lipton's construction

In the EXPSPACE-complete problem we chose, all counters are bounded by R.

Simulating **zero?** x

Introduce a counter $\hat{x}$, set up and maintain the invariant $x + \hat{x} = R$.
Implement a macro $\mathbf{Dec}_n\ x$ that decrements x and increments $\hat{x}$ exactly R times.
Then $\mathbf{Dec}_n\hat{x}$, $\mathbf{Dec}_n x$ is a zero test for x.

Intro

Obtaining the Tower lower bound

Tightness of Lipton's construction

Since there is an efficient zero-test for all counters, Lipton's construction applies also to the coverability problem, which has been proved to be EXPSPACE-complete.
This led to the conjecture which was common in the community that the reachability problem is EXPSPACE-complete.

To achieve a lower bound beyond EXPSPACE, we cannot follow the same pattern.

Main Technique

Main Technique

Enhanced counter program structure

Commands of Counter Program

Normal commands:

- x $+= 1$
- x $-= 1$
- **goto** L **or** $L\prime$
- **zero?** x
- **max?** x

Last command:

- **halt if** $x_1, \cdots, x_l = 0$

Two kinds of counters

Tested counters

A counter is a tested counter iff. it occurs in a **zero?** or **max?** command in the program, meaning that it is bounded from 0 to B.

Untested counters

Counters that are unbounded and the testing commands may not be applied to them.

Notice that untested counters may also appear in the last **halt if** command.

Some terminologies

B-run

A run where the **max** tests are interpreted as checks for equality with B.

Complete run

A run is *complete* iff.

- it's *halted*: successfully executes the last **halt** command;
- its initial valuation assigns 0 to every counter.

Main Technique

Find the reduction

Reduction

Petri nets reachability problem

Input A counter program without *tested counters*.
Question Does it have a *complete run*?

What is coverability then? Just change the last **halt if** ...=0 command to an unconditional **halt**.

A TOWER-complete problem

Input A counter program of size n, without *untested counters*.
Question Does it have a *complete* $3!^n$*-run*?

Once we reduce the TOWER-complete problem to our Petri Nets reachability problem, the lower bound is updated to TOWER!

Main Technique

Amplifiers

Purpose

To reduce the TOWER-Complete problem into the reachability problem of Petri Net.

Obstacles:

- changing tested counters to untested counters;
- modifying the $3!^n$ bound to some arbitrary bound.

Definition

Amplifier

A *B*-amplifier by *R* is a program that maintains 3 counters b, c, d such that

$$b = R, c > 0, d = c \cdot b.$$

B is the bound of this program. *R* is called the *ratio* of the amplifier.

A *B*-amplifier by *R* is designed to connect to a *R*-program and change the bound to *B*. Let's denote $\triangleright$ as "connect", then

$$(B\text{-amplifier by } R) \triangleright (R\text{-program}) \to (B\text{-program}).$$

The $\triangleright$ operator

$\mathcal{A} \triangleright \mathcal{P}$: concatenate an amplifier with a counter program. It does two things:

- eliminate tested counters in $\mathcal{P}$;
- change the bound.

In order to eliminate tested counters, how to simulate **zero?** x and **max?** x tests?
Main idea:

- Instead of using a deterministic macro **Dec** x in Lipton's work, we use non-deterministic **loop** to construct **zero?** x and **max?** x test.
- Then we add an additional **halt** condition to ensure that each simulating **loop** runs exactly R times.

The $\triangleright$ operator

Simulating **zero?** x

Every **zero?** x_i command in $\mathcal{P}$ is simulated by

loop
    x_i += 1, $\hat{x}_i$ -= 1
    d -= 1
c -= 1
loop
    x_i -= 1, $\hat{x}_i$ += 1
    d -= 1
c -= 1

Merged **halt**

halt if {counters in A's **halt**}, {counters in P's **halt**}, d = 0

The $\triangleright$ operator

Proposition

For every valuation of counters of $\mathcal{P}$, it occurs after a complete B-run of $\mathcal{A} \triangleright \mathcal{P}$ iff. it occurs after a complete R-run of $\mathcal{P}$.

It means that $\mathcal{A} \triangleright \mathcal{P}$ can faithfully simulate any complete R-run of $\mathcal{P}$.

Observation

Applying the operator to a B-amplifier by B', and B'-amplifier by B'', will produce a B-amplifier by B''.

It means that we can apply $\triangleright$ multiple times to construct an even "stronger" amplifier!

Two kinds of amplifiers

Trivial Amplifier $\mathcal{A}$

- A B-amplifier by R, where B is arbitrary and R a constant.
- It will not introduce new tested counters.

Factorial Amplifier $\mathcal{F}$

- A k-amplifier by $k!$.
- The bound is reduced factorially here!
- It will introduce some new tested counters.

Idea: we can combine these two amplifiers to construct the needed amplifier for reduction.

Trivial Amplifier

$\mathcal{A}$'s code:

```
b += R
c += 1, d += R
loop
    c += 1, d += R
halt.
```

Notice that it has no tested counters. This feature ensures that B is arbitrary in $\mathcal{A}$.

Factorial Amplifier

What we want to do is to construct a k-amplifier by $k!$, i.e. a k-program such that the relation it k-computes in counters b, c, d is

$$b = k!, c > 0, d = c \cdot b.$$

Factorial Amplifier

```
// Initialize.
i+=1, b+=1, c+=1, d+=1,
x+=1, y+=1
loop
    c+=1, d+=1, x+=1, y+=1
.
.
.
.
.
.
.
// Check correctness.
max? i
loop
    x -= i, y -= 1
halt if y = 0
```

```
// Construct b, c, d.
loop
    loop
        c -= i, c' += 1
        loop at most b times
            d -= i, x -= i, d' += i+1
    loop
        b -= i, b' += i+1
    loop
        b' -= 1, b += 1
    loop
        c' -= 1, c += 1
        loop at most b times
            d' -= 1, d += 1, x += 1
    i += 1
```

Main Technique

Combining all together

Final Amplifier

Now we can construct a powerful amplifier by $3!^n$.

Lemma

An amplifier by $3!^n$ without tested counters is computable in time $O(n)$.

Proof.

Letting $\mathcal{A}$ be a trivial amplifier by 3, The program

$$\mathcal{T} := \underbrace{((\mathcal{A} \rhd F) \rhd F) \rhd \cdots \rhd F}_{n \text{ compositions}}$$

is an ampifier by $3!^n$ without tested counters, and it is computable in time $O(n)$ by the definition of our composition operator. □

Final Result

Theorem

The Petri nets reachability problem is TOWER*-hard.*

Proof.

Let $\mathcal{M}$ be a program solving the TOWER-complete halting problem. Let $\mathcal{T}$ be an amplifier by $3!^n$ without tested counters which is computable in time $O(n)$. Then the composite program

$$\mathcal{T} \rhd \mathcal{M}$$

has no tested counters and is reduced to the Petri net reachability problem. □

Thanks for Listening!