Extensions to the CEGAR Approach on Petri Nets

Ákos Hajdu[1], András Vörös[1], Tamás Bartha[2], Zoltán Mártonka[1]

[1] *Department Of Measurement and Information Systems*

Budapest University of Technology and Economics, Budapest, Hungary

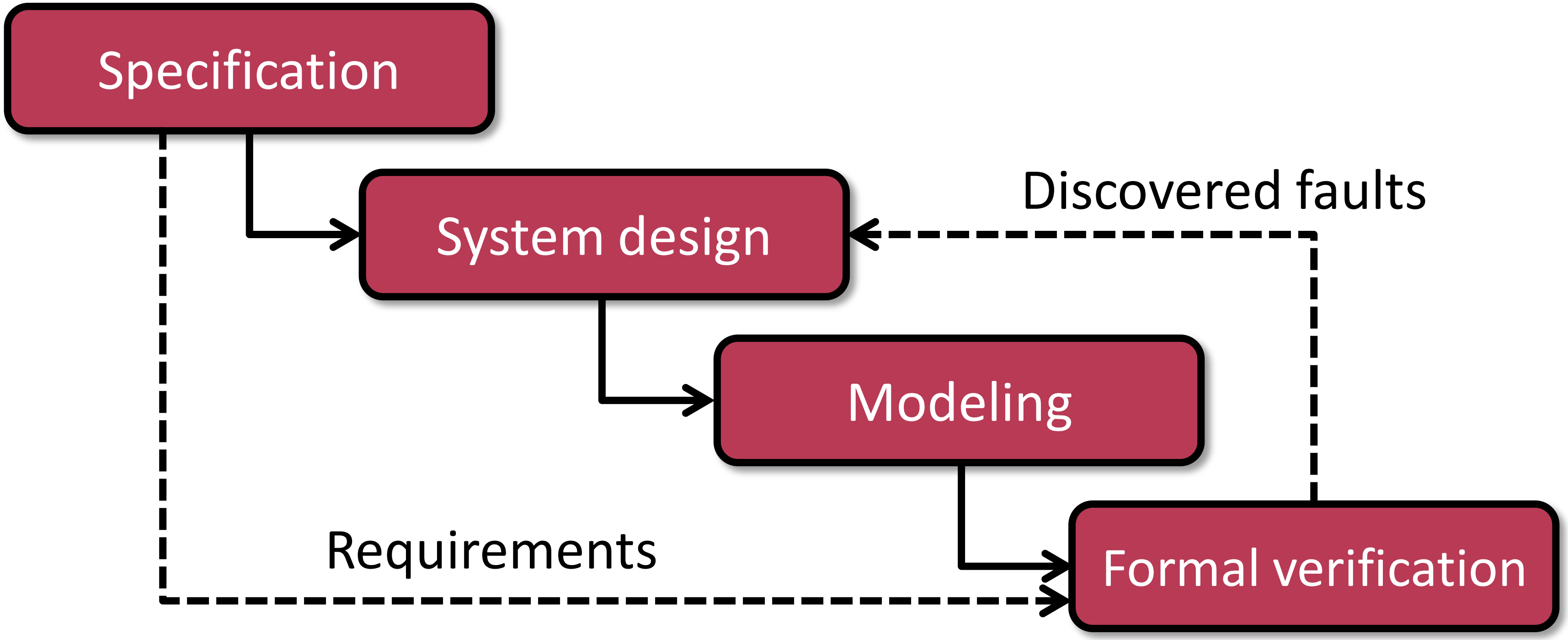
[2] *MTA SZTAKI, Budapest, Hungary*

Outline of the talk

1. Introduction
2. CEGAR approach on Petri nets
3. Theoretical results
4. Algorithmic contributions
5. Evaluation
6. Conclusion

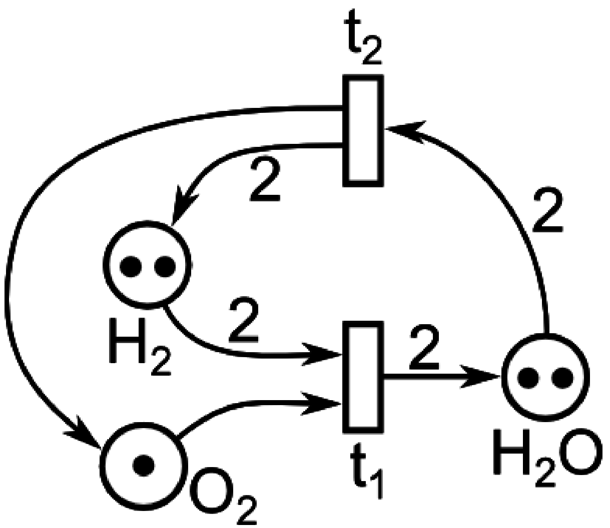
Introduction – Formal methods

- Complex, distributed, safety-critical systems
  - Suitability and faultlessness is important
  - Mathematically precise verification is required
  - Formal modeling and analysis can provide such tools



Introduction – Petri nets

- Petri nets
  - Widely used modeling formalism
  - Asynchronous, distributed, parallel, non-deterministic systems
- Structure
  - Bipartite graph
    - Places, transitions, weighted arcs, tokens
- Dynamic behavior
  - State: distribution of tokens (marking)
  - Firing of a transition: consumes and produces tokens



Introduction – Reachability analysis

- Reachability analysis
  - Formal verification technique
  - Checks, if a given state is reachable from the initial state
  - Drawback: complexity
- Complexity
  - State space can be large or infinite
  - Reachability is decidable, but at least EXPSPACE-hard
  - A possible solution is to use <u>abstraction</u>

CEGAR approach

- **C**ounter**E**xample **G**uided **A**bstraction **R**efinement
  - General approach
    - Can handle large or infinite state space
  - Works on an abstraction of the original model
    - Less detailed state space
    - Finite, smaller representation
  - Abstraction refinement is required
    - An action in the abstract model may not be realizable in the original model
    - Refine the abstraction using the information from the explored part of the state space

CEGAR approach on Petri nets

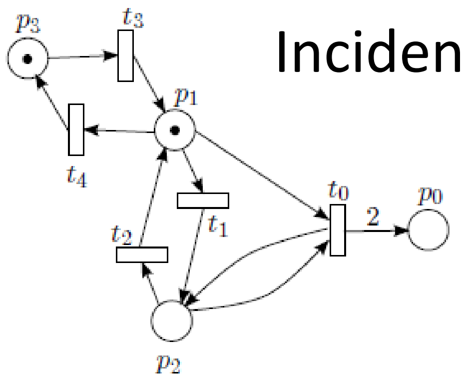
- Abstraction of Petri nets: state equation

Initial state

$$m_0 + Cx = m_1$$

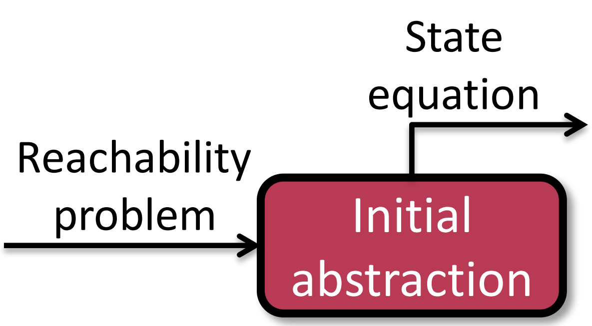
Target state

Incidence matrix



$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 1 & -1 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

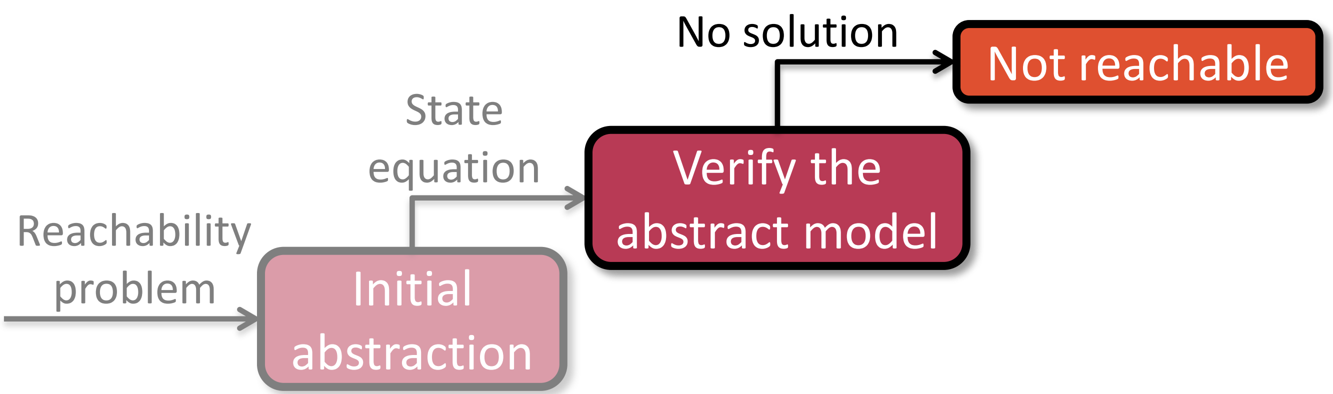
Firing count of transitions *(unknown)*



CEGAR approach on Petri nets

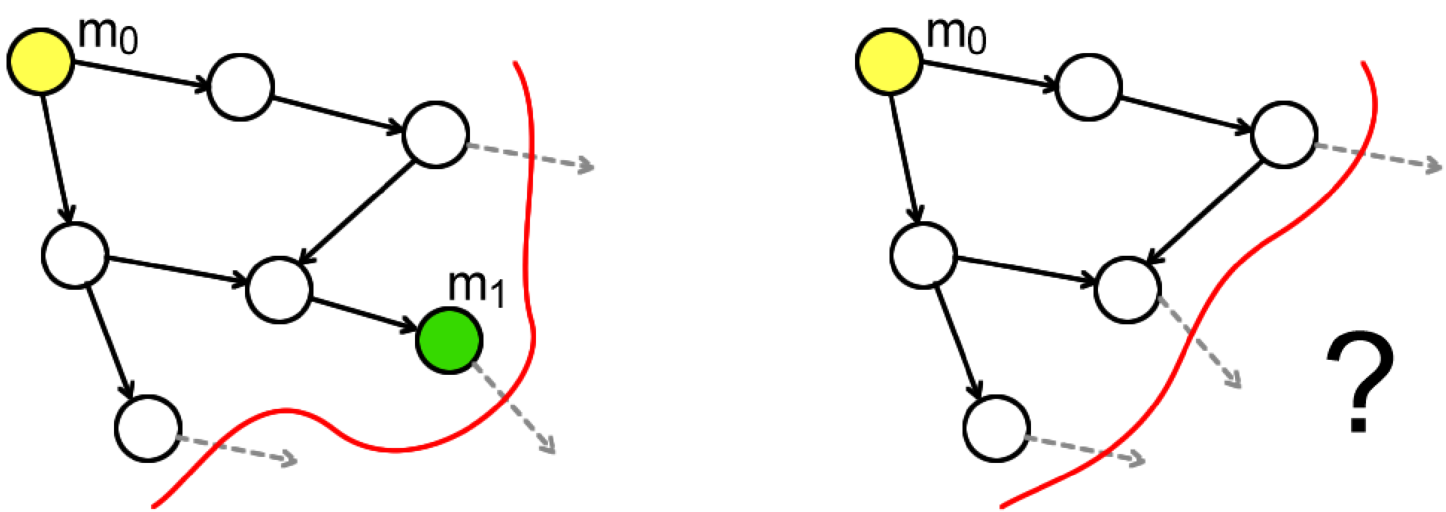
- Verification of the abstract model
  - Solving the state equation for the firing count of transitions
  - Integer Linear Programming problem
  - Necessary, but not sufficient criterion for reachability

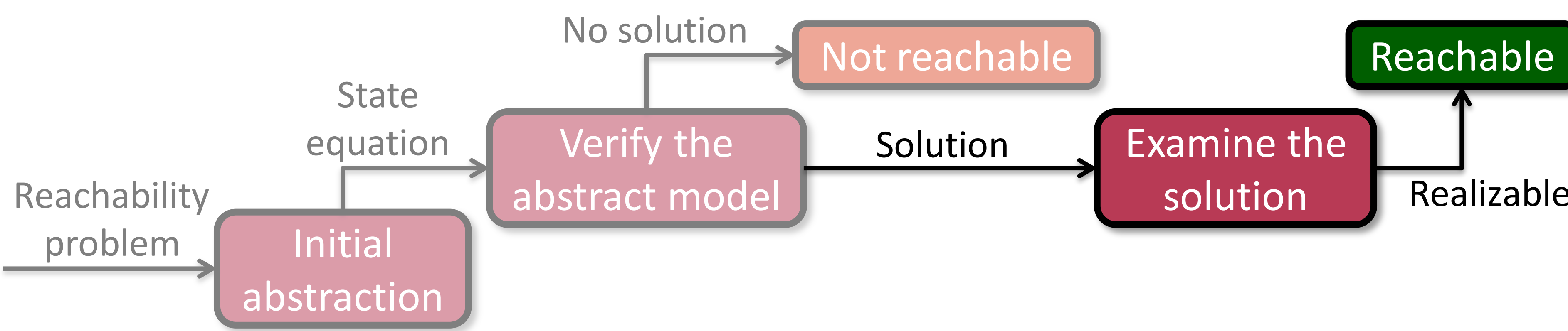
$$m_0 + C\mathbf{x} = m_1$$



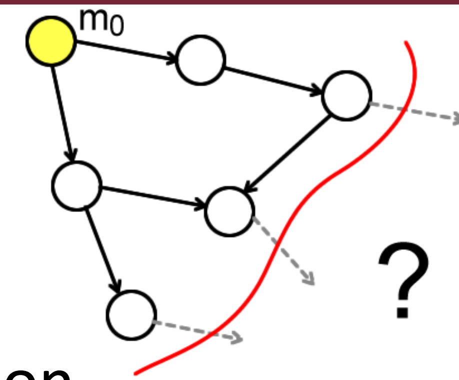
CEGAR approach on Petri nets

- Examining the solution
  - Bounded exploration of the state space

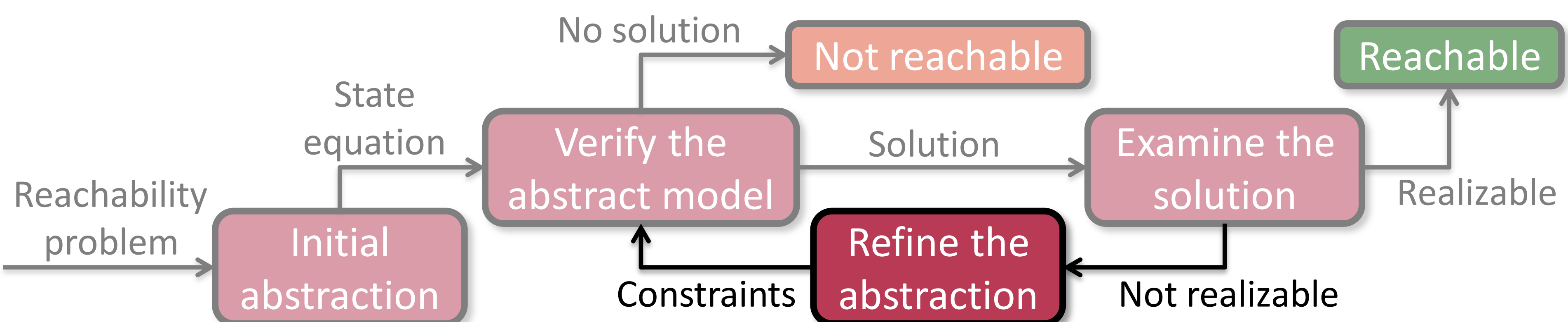




CEGAR approach on Petri nets

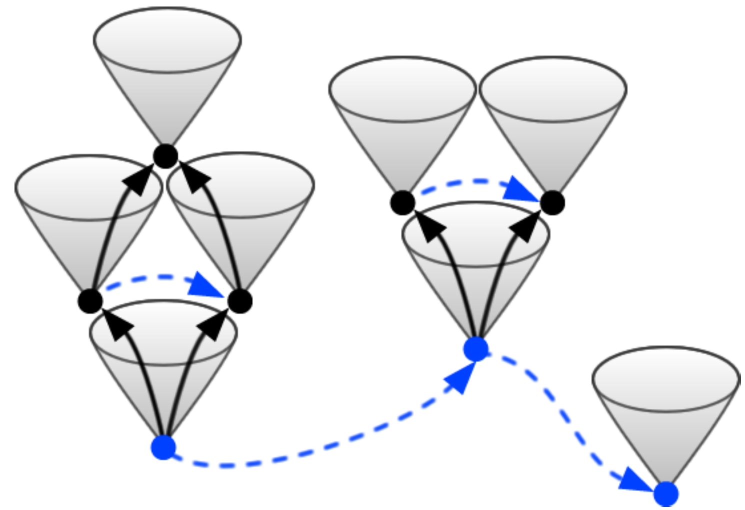


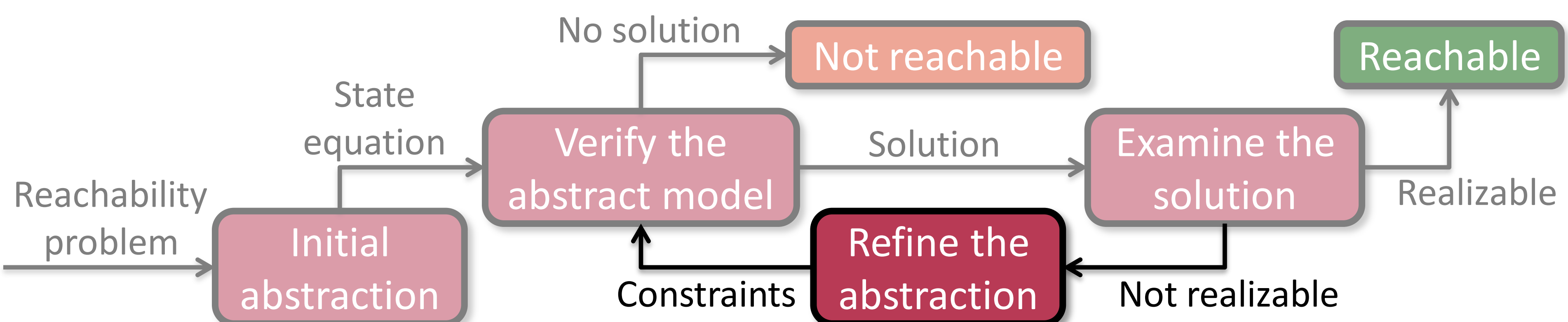
- Abstraction refinement
  - Exclude the counterexample without losing any realizable solution
  - Constraints can be added to the state equation
    - The state equation may become infeasible
    - A new solution can be obtained
  - Traversing the solution space instead of the state space



CEGAR approach on Petri nets

- Traversing the solution space
  - Base vector + linear combination of invariants
  - Constraint types:
    - Jump: obtain another base vector
    - Increment: obtain non-minimal solutions



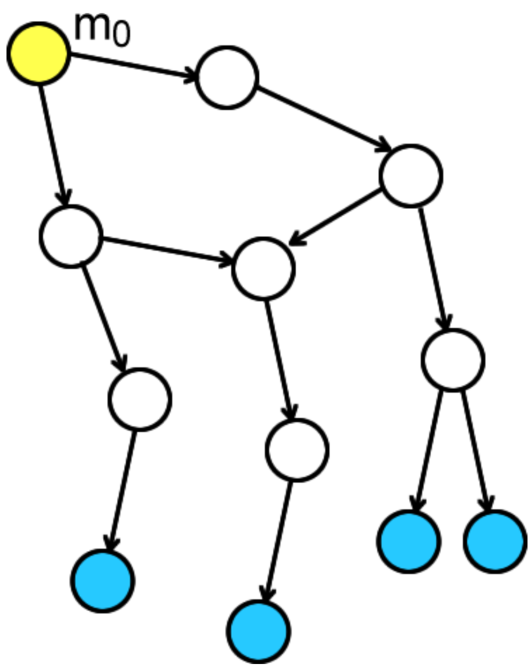


Our work

- Examination of the algorithm
  - **Correctness**
  - Completeness
- Improving the algorithm
  - Extending the set of decidable problems
  - New criterion for termination
- Extending the algorithm
  - Solving submarking coverability
  - Handling inhibitor arcs

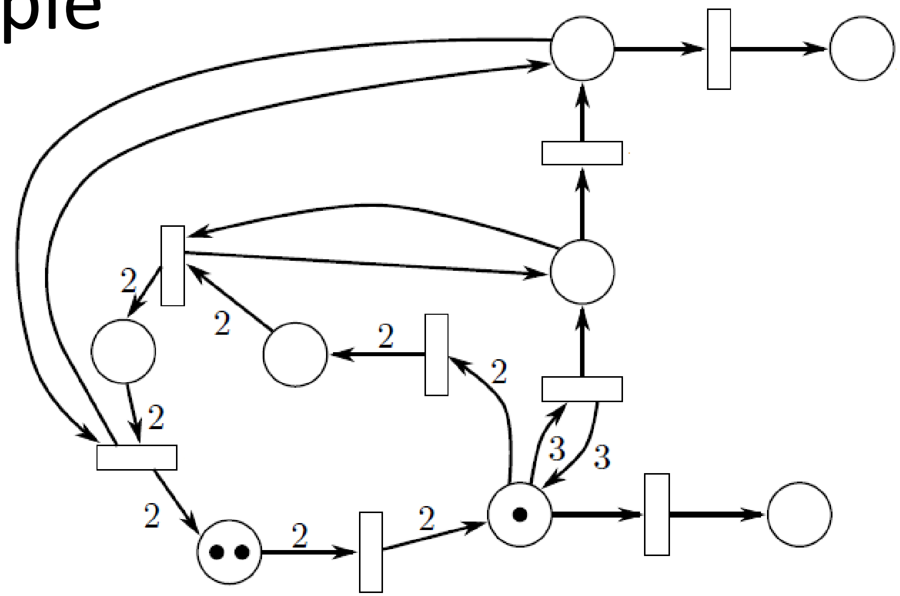
Correctness of the algorithm

- The CEGAR method is proved to be correct, but:
- A heuristic is used to calculate the constraints
  - Uses only information from maximal firing sequences
    - This information may not be enough
  - The algorithm may over-estimate the constraints
    - … and give an incorrect answer



Correctness of the algorithm

- We proved the incorrectness of the heuristic by a counterexample

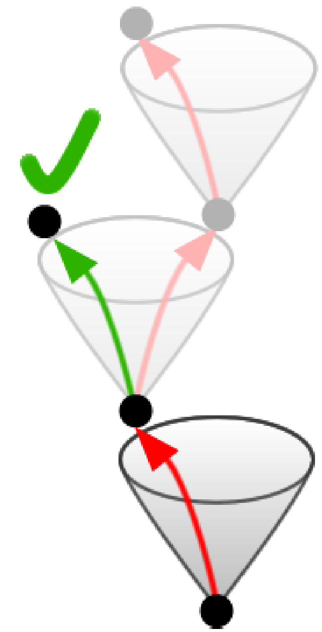


- In our improved algorithm:
  - We use the information of intermediate states
  - Detect the over-estimation and do not give an incorrect answer

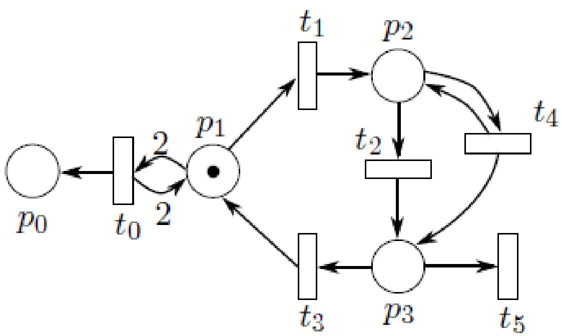
Our work

- Examination of the algorithm
  - Correctness
  - **Completeness**
- Improving the algorithm
  - Extending the set of decidable problems
  - New criterion for termination
- Extending the algorithm
  - Solving submarking coverability
  - Handling inhibitor arcs

Completeness of the algorithm



- The algorithm is incomplete due to its iteration strategy
  - An invariant is added to the solution which can be fired, but does not help
    - The same states are reached
      → the same invariant is added again
  - This solution is skipped to avoid non-termination
    - In some cases using another invariant would help
  - We proved this by counterexamples

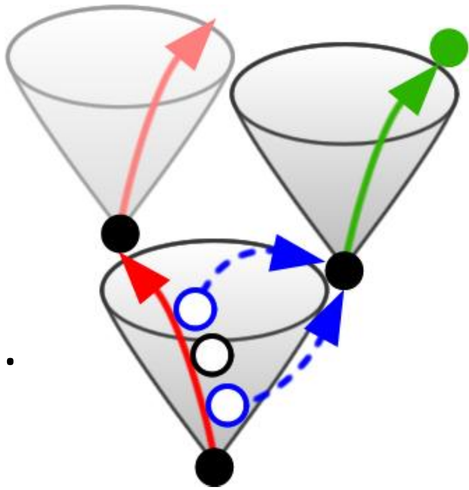


Our work

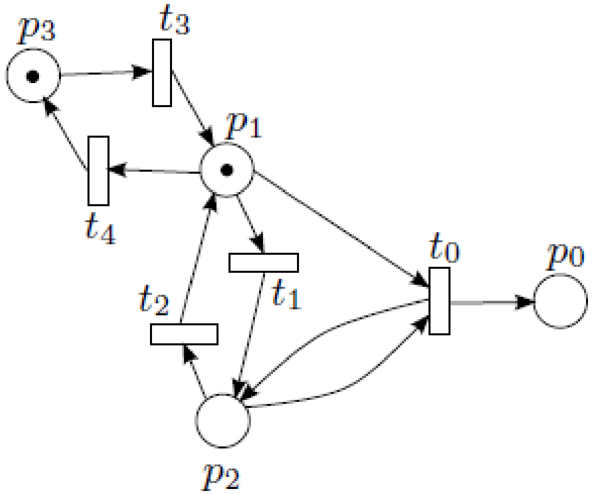
- Examination of the algorithm
  - Correctness
  - Completeness
- Improving the algorithm
  - **Extending the set of decidable problems**
  - New criterion for termination
- Extending the algorithm
  - Solving submarking coverability
  - Handling inhibitor arcs

Algorithmic contributions - Improvements

- When a solution is skipped...
  - ...the algorithm checks if any intermediate state can lead to a realizable solution
  - Not all important states are recognized, due to ...
    - ... the ordering of the states
    - ... optimizations



- We defined a total ordering on the intermediate states
  - Every state closer to a realizable solution is recognized
  - → Extends the set of decidable problems

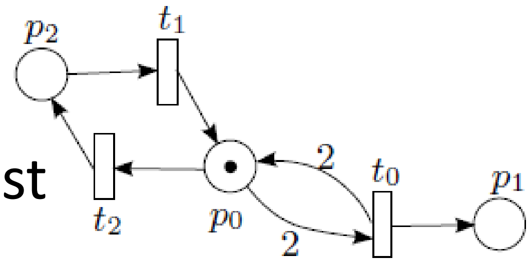


Our work

- Examination of the algorithm
  - Correctness
  - Completeness
- Improving the algorithm
  - Extending the set of decidable problems
  - **New criterion for termination**
- Extending the algorithm
  - Solving submarking coverability
  - Handling inhibitor arcs

Algorithmic contributions - Improvements

- We developed a new termination criterion
  - Before applying a constraint, a modified form of the state equation is checked
  - Constraints that cannot help, can be detected before applying them
- Advantages
  - Cuts the search space efficiently
  - Can prevent non-termination
    - We proved that no realizable solution is lost
    - → Extends the set of decidable problems



Our work

- Examination of the algorithm
  - Correctness
  - Completeness
- Improving the algorithm
  - Extending the set of decidable problems
  - New criterion for termination
- Extending the algorithm
  - **Solving submarking coverability**
  - Handling inhibitor arcs

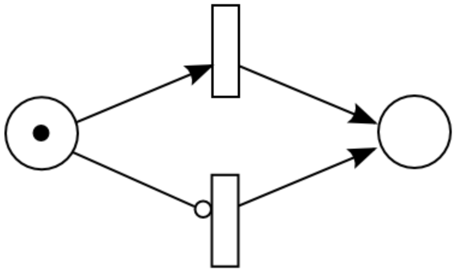
Algorithmic contributions - Extensions

- Submarking coverability problem
  - Linear conditions are given (instead of the target state)
  - Checks, if a state can be reached, for which the given conditions hold
- Solving submarking coverability using CEGAR
  - We transformed the conditions on the state to conditions on transitions → ILP problem
- New types of problems can be analyzed
  - Analysis of subsystems
  - Reachability of infinite set of states

Our work

- Examination of the algorithm
  - Correctness
  - Completeness
- Improving the algorithm
  - Extending the set of decidable problems
  - New criterion for termination
- Extending the algorithm
  - Solving submarking coverability
  - **Handling inhibitor arcs**

Algorithmic contributions - Extensions



- Inhibitor arcs
  - Petri nets extended with inhibitor arcs are Turing complete
  - The reachability problem in general is undecidable
- Handling inhibitor arcs in the CEGAR approach
  - We introduced new types of constraints
  - We extended the total ordering on intermediate states
  - Completenes cannot be guaranteed, but we tested the algorithm on several Petri nets

Evaluation

Evaluation

- We implemented our algorithm
  - PetriDotNet framework
    - Modeling and analysis of Petri nets
    - Supports add-ins
- We compared our algorithm to…
  - … the original implementation
    - SARA tool
    - Won several categories at the model checking contest 2013
  - … the saturation algorithm
    - Other type of reachability analysis method
    - Implemented in the PetriDotNet framework

Evaluation

For certain models, the algorithmic contributions reduce the computational effort and **our algorithm can solve problems that SARA cannot**

The ILP solver can produce results efficiently

Model	SARA tool	Our algorithm
CP_NR 10	0,2 s	0,5 s
CP_NR 25	111 s	2 s
CP_NR 50	TO	16 s
Kanban 1000	0,2 s	1 s
FMS 1500	0,5 s	5 s
MAPK	0,2 s	1 s

Model	Saturation	Our algorithm
Kanban 1000	TO	1 s
FMS 1500	TO	5 s
SlottedRing 50	4 s	433 s
Dphil 50	0,5 s	45 s

Solving the ILP problem dominates run-time

Constant speed penalty due to the managed environment and the overhead of the algorithmic contributions

TO: Time out (runtime > 600 s)

Conclusion

- Theoretical results
  - The iteration strategy of the CEGAR approach is incomplete
  - A heuristic in the original algorithm is incorrect
    - We detect such situations
- Algorithmic contributions
  - Extend the set of decidable problems
  - Reduce the search space
  - Solve new classes of problems
    - Submarking coverability
    - Petri nets with inhibitor arcs

Thank you for your attention!

Questions?