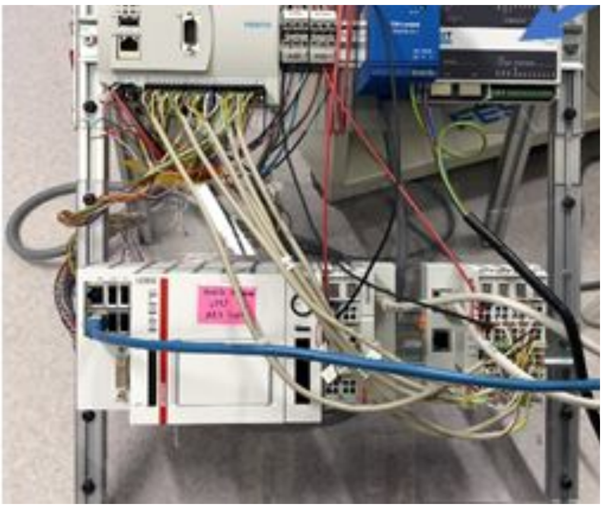
Towards automatic state machine reconstruction from legacy PLC using data collection

Daniil Chivilikhin, Sandeep Patil,
Anthony Cordonnier, Valeriy Vyatkin

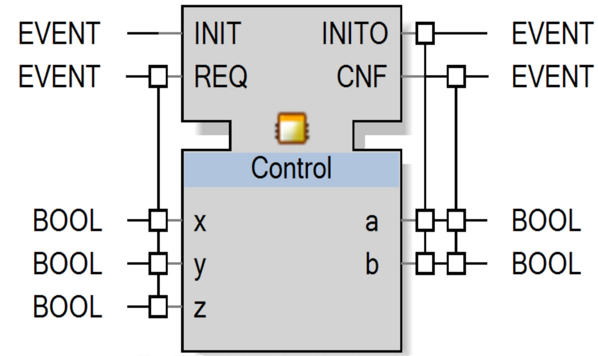
IEEE INDIN 2019, Helsinki, Finland

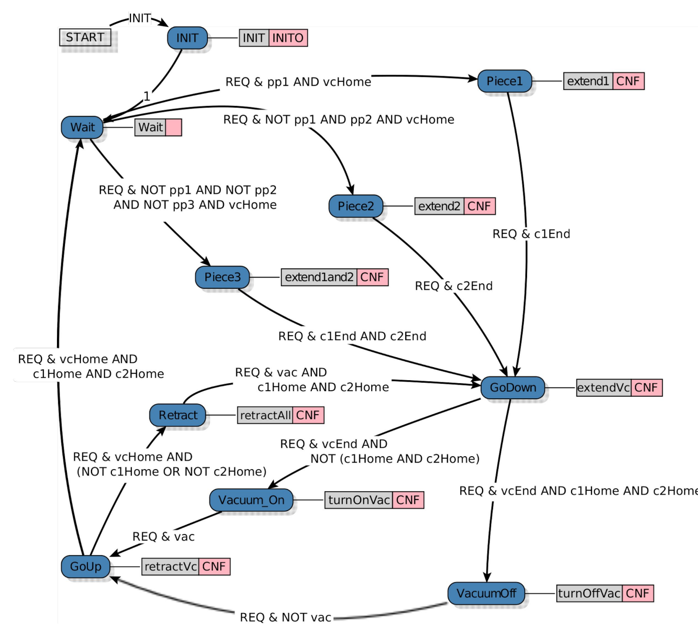
24 July 2019

Goal



Legacy PLC
IEC 61131-3
(black box)





IEC 61499 state machine

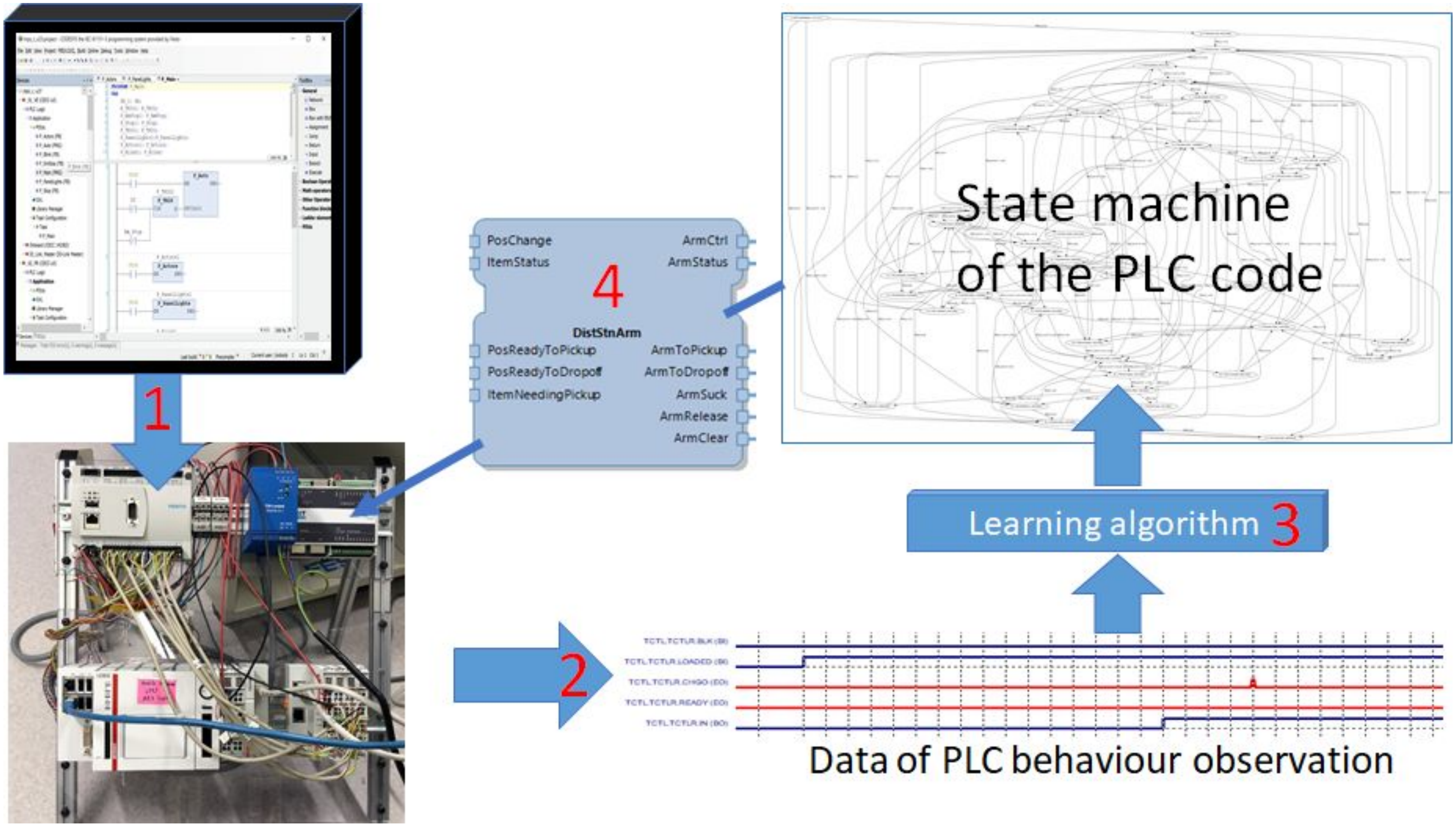
Contributions

1. **Hardware** and **software architecture** for collecting behavioral data from legacy PLCs in production

2. **Algorithm** based on translation to Boolean satisfiability problem (SAT) for **reconstructing controller** logic in the form of a state machine from data collected from PLC

3. **Demonstration** of the proposed solution on an example of a laboratory scale model of a distribution station

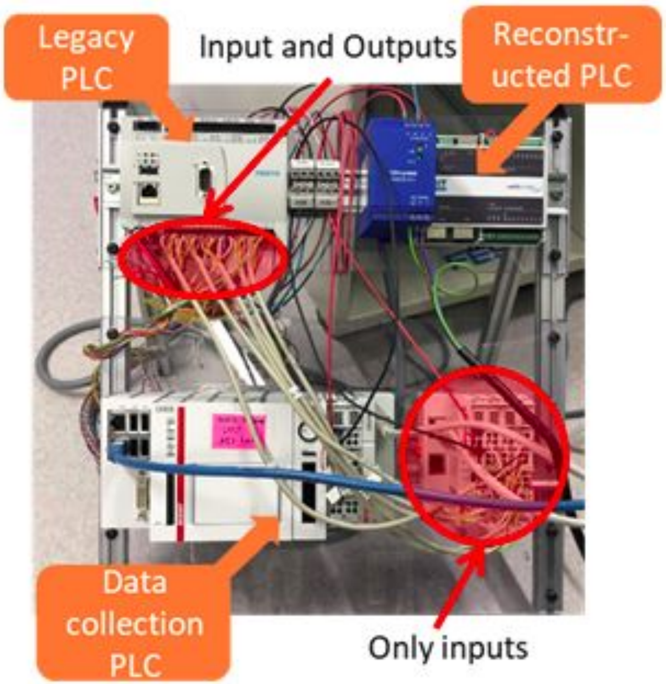
Overview of the proposed approach



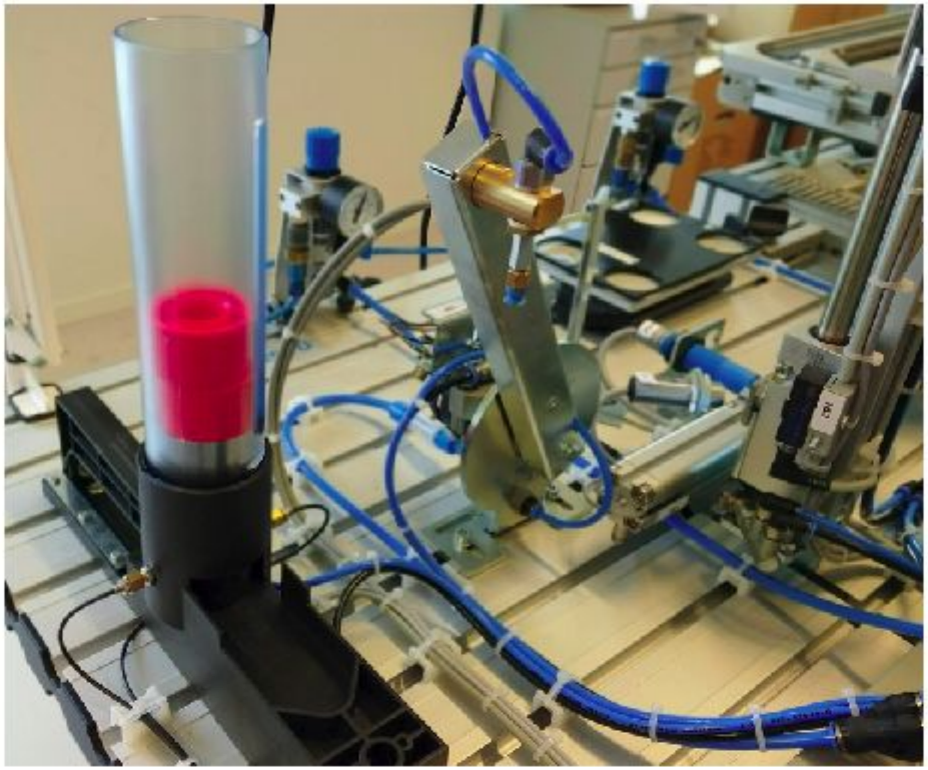
Data collection

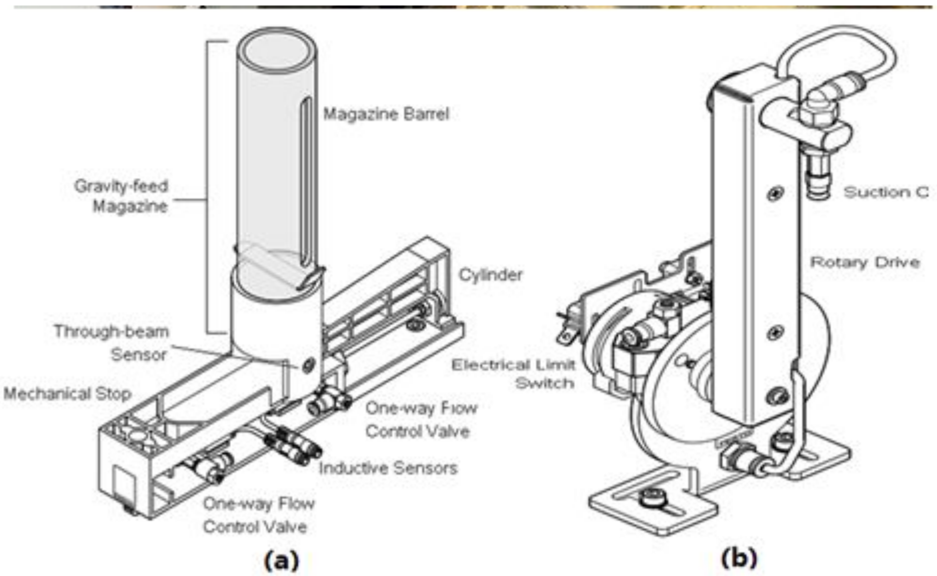
Hardware architecture for data collection

- Black-box PLC
- Data collection PLC running IEC 61499 app
- Reconstructed PLC



Example system: Festo distribution station





Data preprocessing (1/4)

Raw data:

Input=[0100010101001] Output=[10000010]
Input=[0100010101001] Output=[10000010]
Input=[0100010101001] Output=[10000010]
Input=[0100010101001] Output=[10000010]
Input=[1100010101001] Output=[00000010]
Input=[1100010101001] Output=[00000010]
Input=[1100010101001] Output=[00000010]
Input=[1100010101001] Output=[00000010]

Data preprocessing (2/4)

Raw data:

Input=[01000101001001] Output=[10000010]
Input=[01000101001001] Output=[10000010]
Input=[01000101001001] Output=[10000010]
Input=[01000101001001] Output=[10000010]
Input=[11000101001001] Output=[00000010]
Input=[11000101001001] Output=[00000010]
Input=[11000101001001] Output=[00000010]
Input=[11000101001001] Output=[00000010]

Data preprocessing (3/4)

Input=[01000101001001] Output=[10000010]
Input=[11000101001001] Output=[00000010]

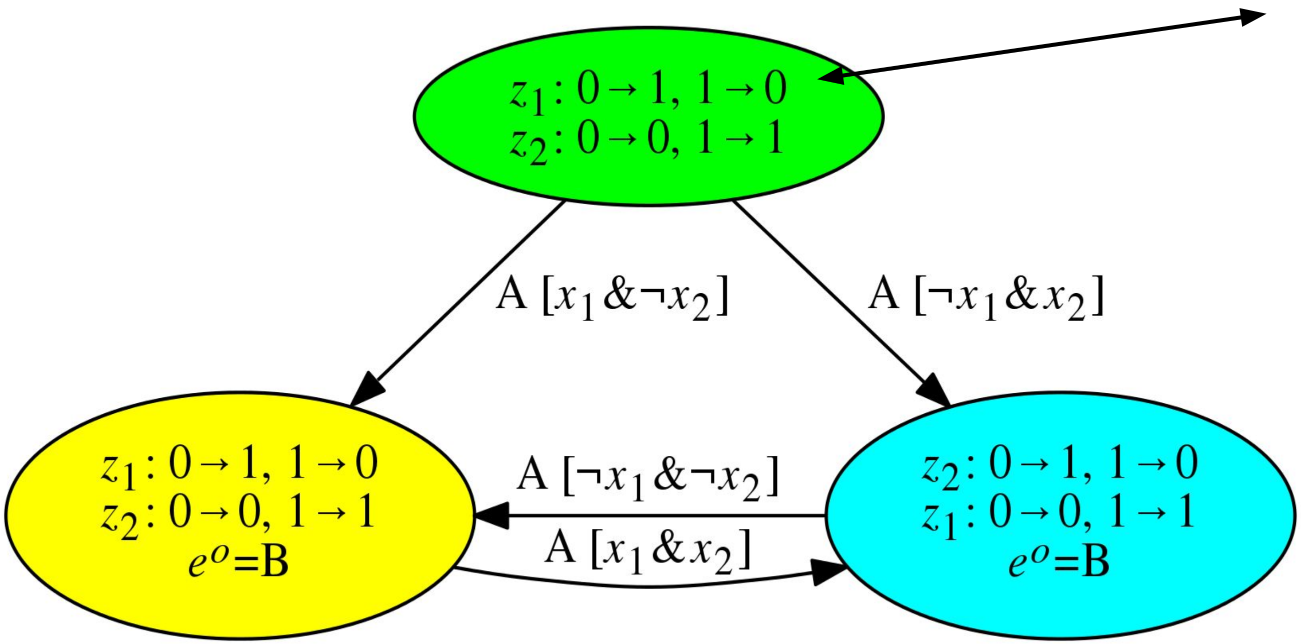
Data preprocessing (4/4)

<REQ[01000101001001], CNF[10000010]>;

<REQ[11000101001001], CNF[00000010]>

Basic function block model

Boolean input/output vars



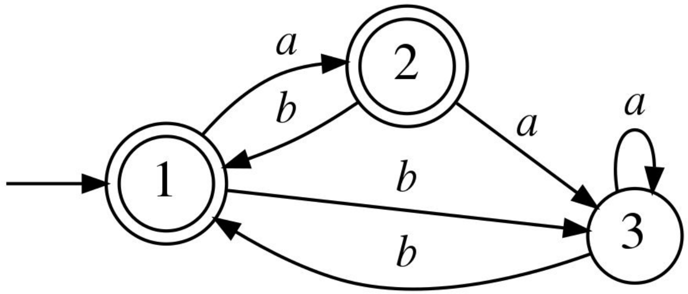

```
IF NOT z1 THEN
    z1 := TRUE
ELSE
    z1 := FALSE
END_IF;
```

State machine reconstruction

Background

- **Minimum** deterministic finite automaton construction from labeled data is NP-complete [Gold, 1978]

T_+={ab, b, ba, bbb}
T_-={abbb, baba}

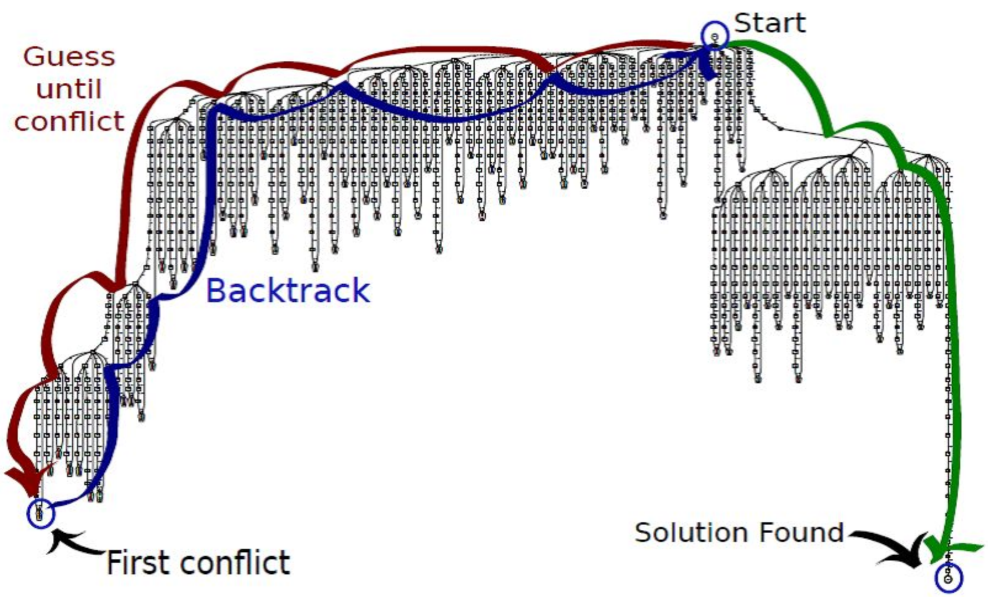


Heuristics, e.g. state merging, k-tails

Metaheuristic, e.g. genetic algorithms

SAT-based

SAT-based state machine synthesis (1/3)

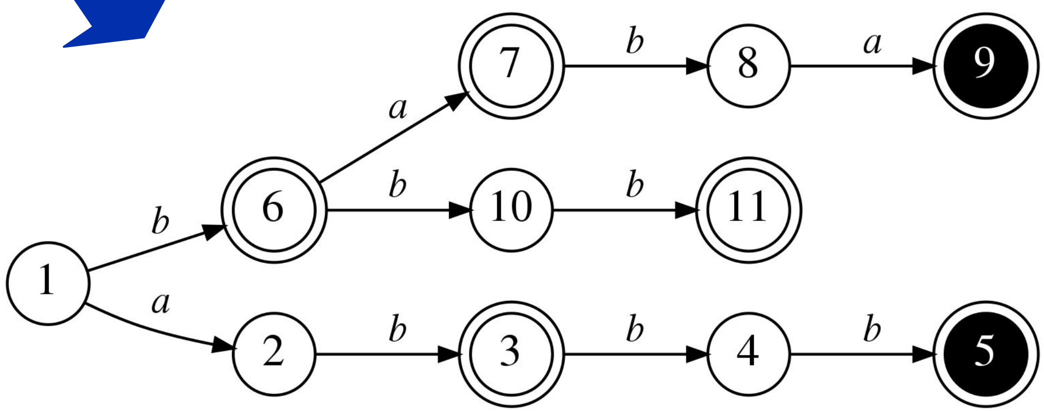


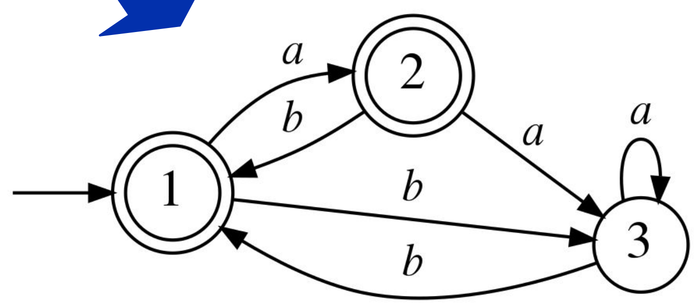
- Heule et al. Exact DFA Identification Using SAT Solvers [ICGI'10]
- ...
- Ulyantsev et al. Exact finite-state machine identification from scenarios and temporal properties [STTT'18]
- Chivilikhin et al. Function block finite-state model identification using SAT and CSP solvers [TII'19]

https://srlabs.de/bites/minisat-intro/

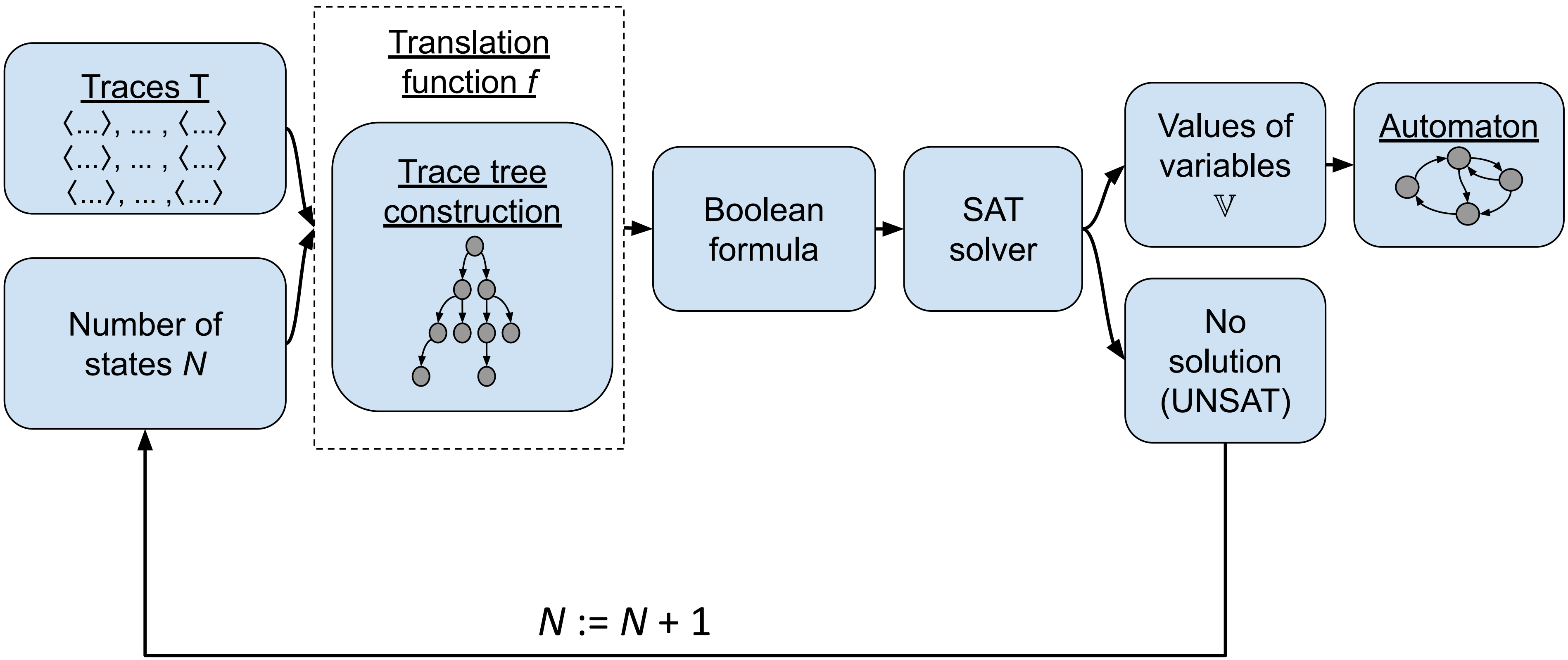
SAT-based state machine synthesis (2/3)

T_+={ab, b, ba, bbb}
T_-={abbb, baba}

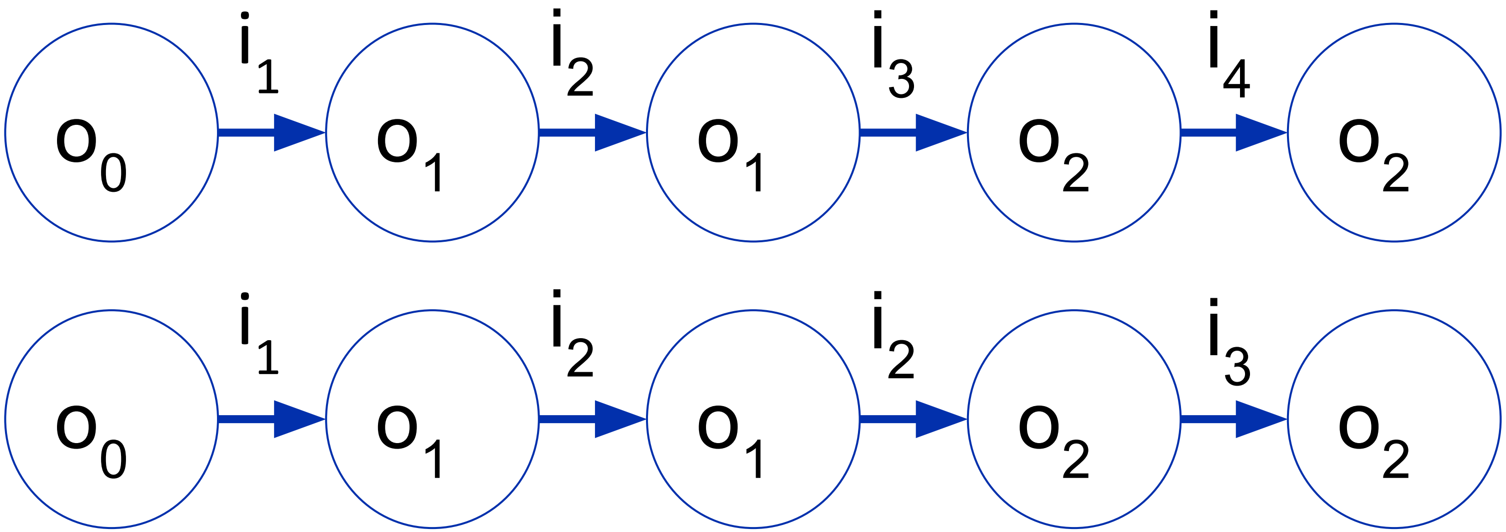




SAT-based state machine synthesis (3/3)

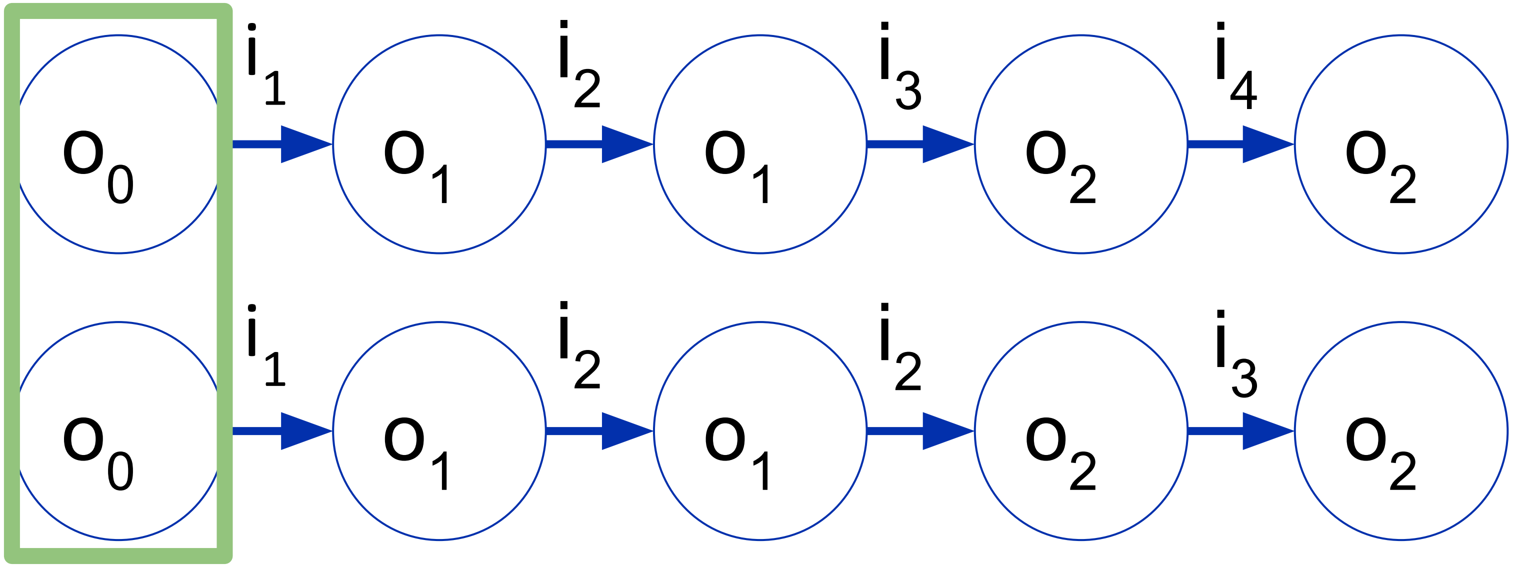


Example

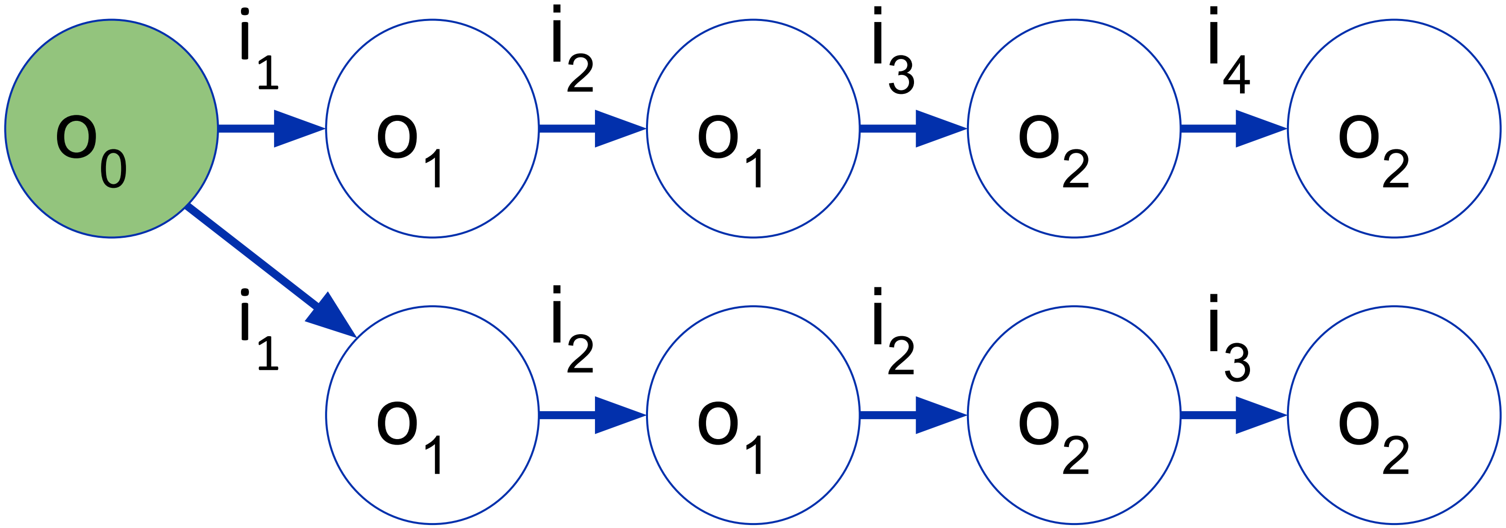


$o_0 \xrightarrow{i_1} o_1 \xrightarrow{i_2} o_1 \xrightarrow{i_3} o_2 \xrightarrow{i_4} o_2$
$o_0 \xrightarrow{i_1} o_1 \xrightarrow{i_2} o_1 \xrightarrow{i_2} o_2 \xrightarrow{i_3} o_2$

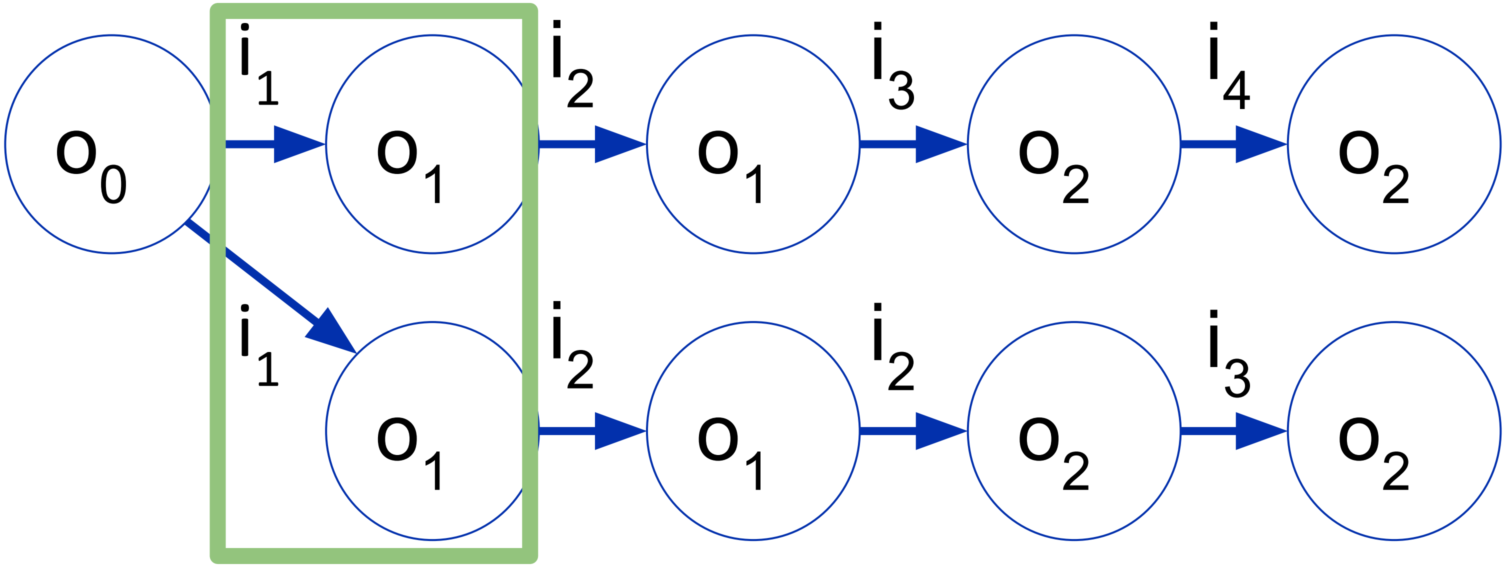
Example



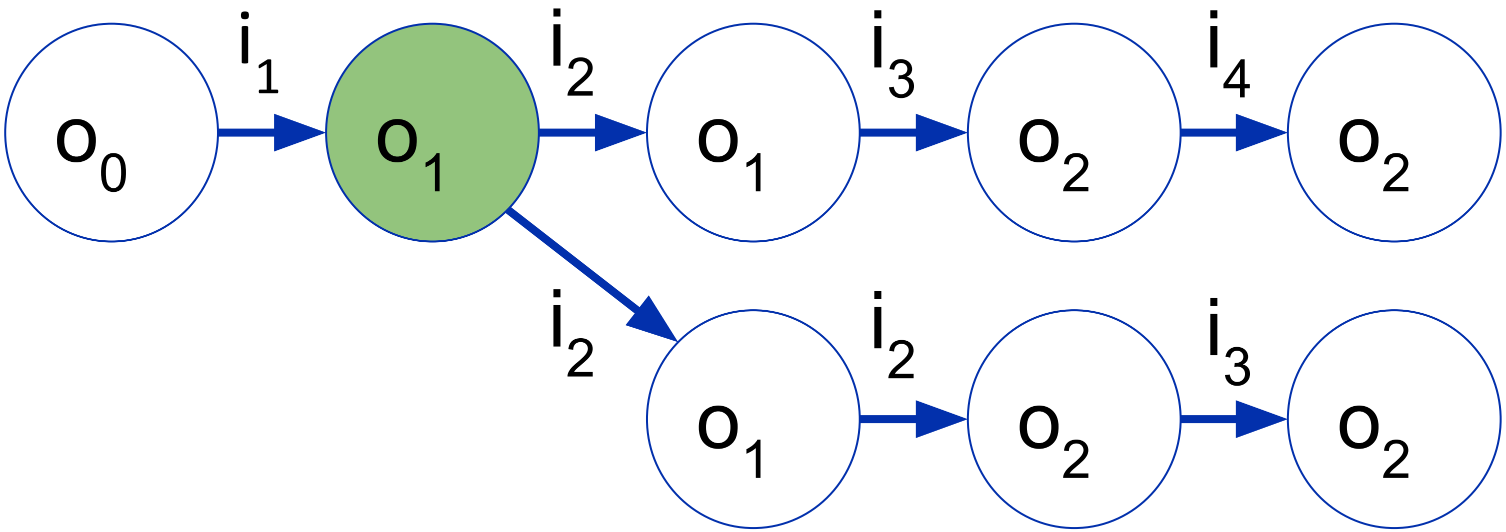
Example



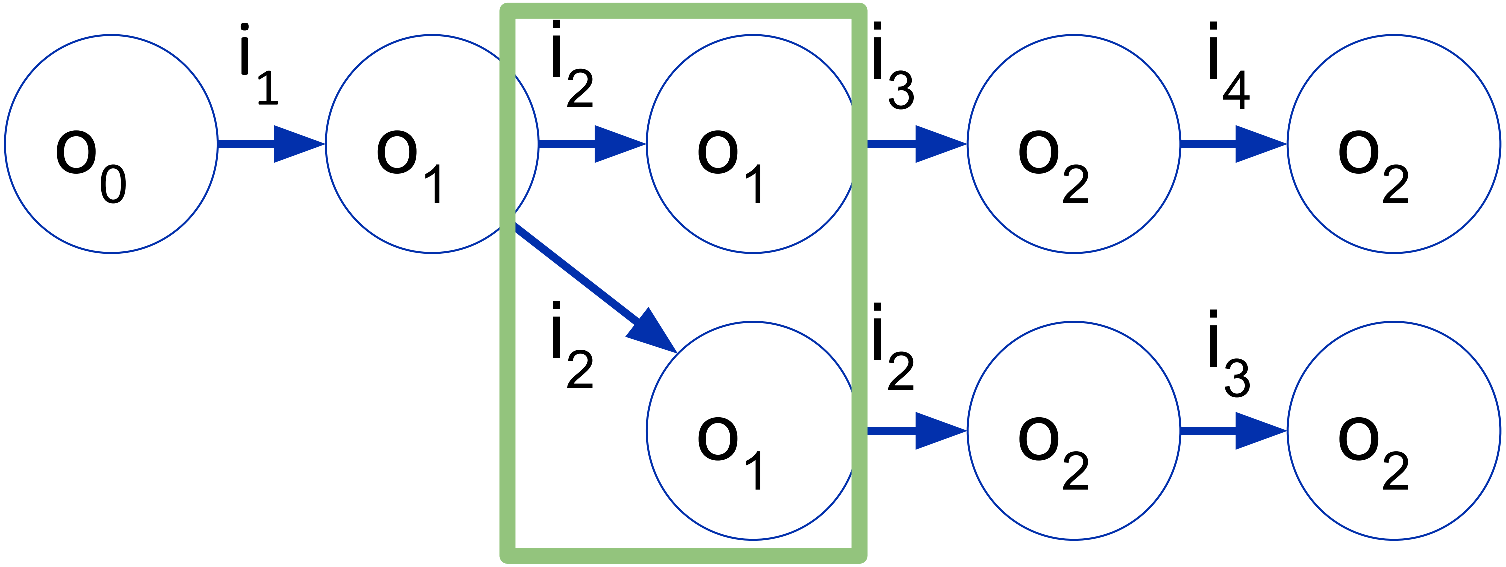
Example



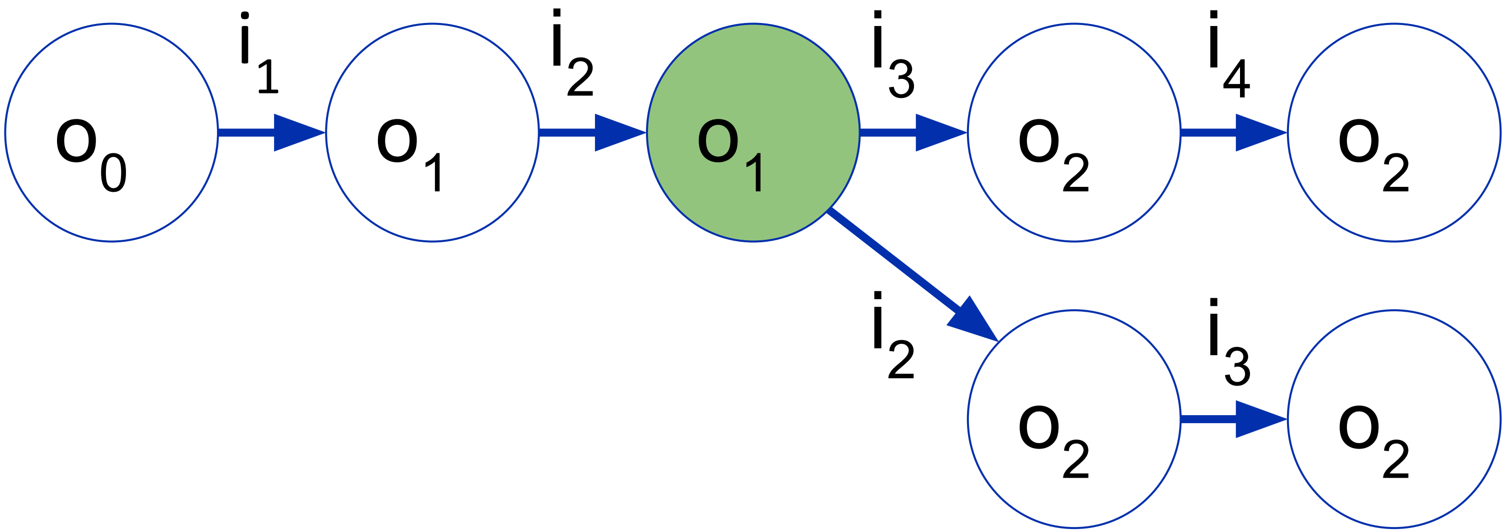
Example



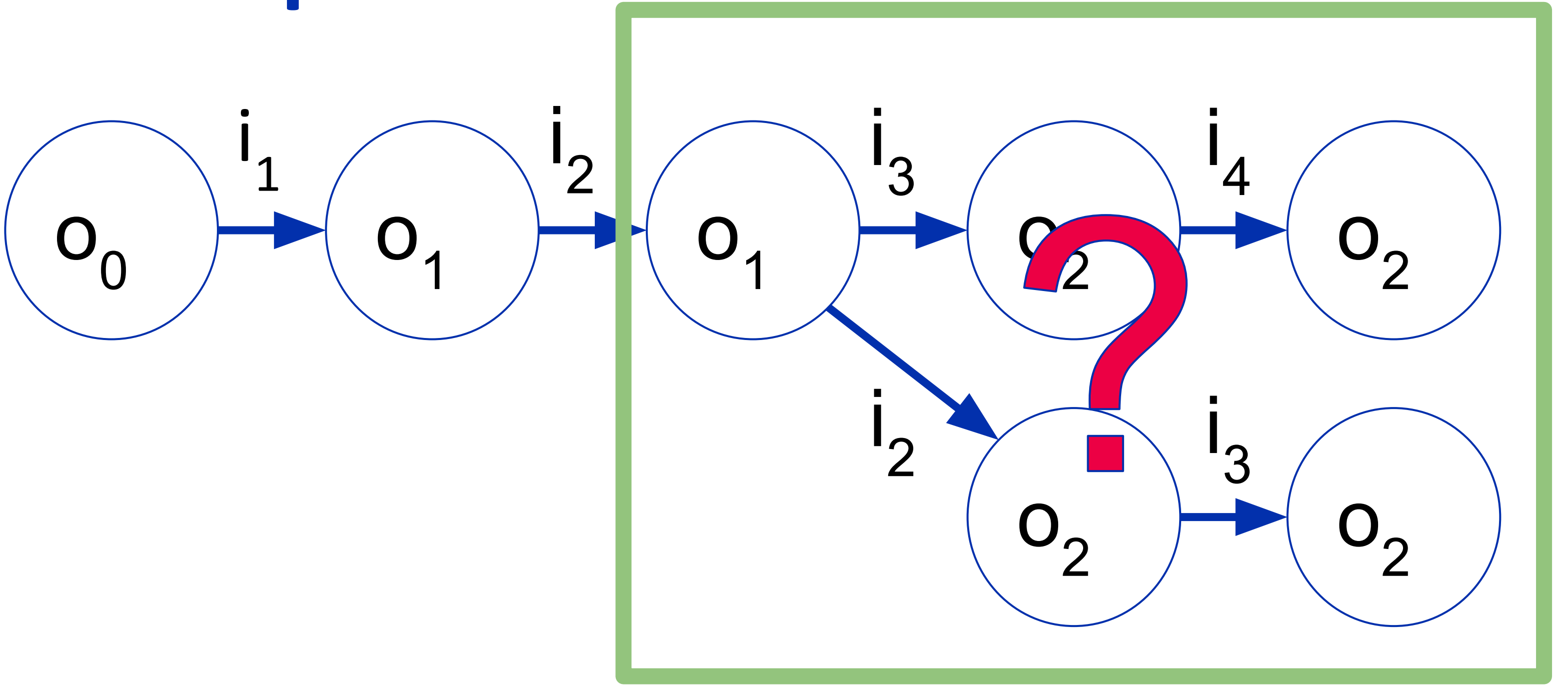
Example



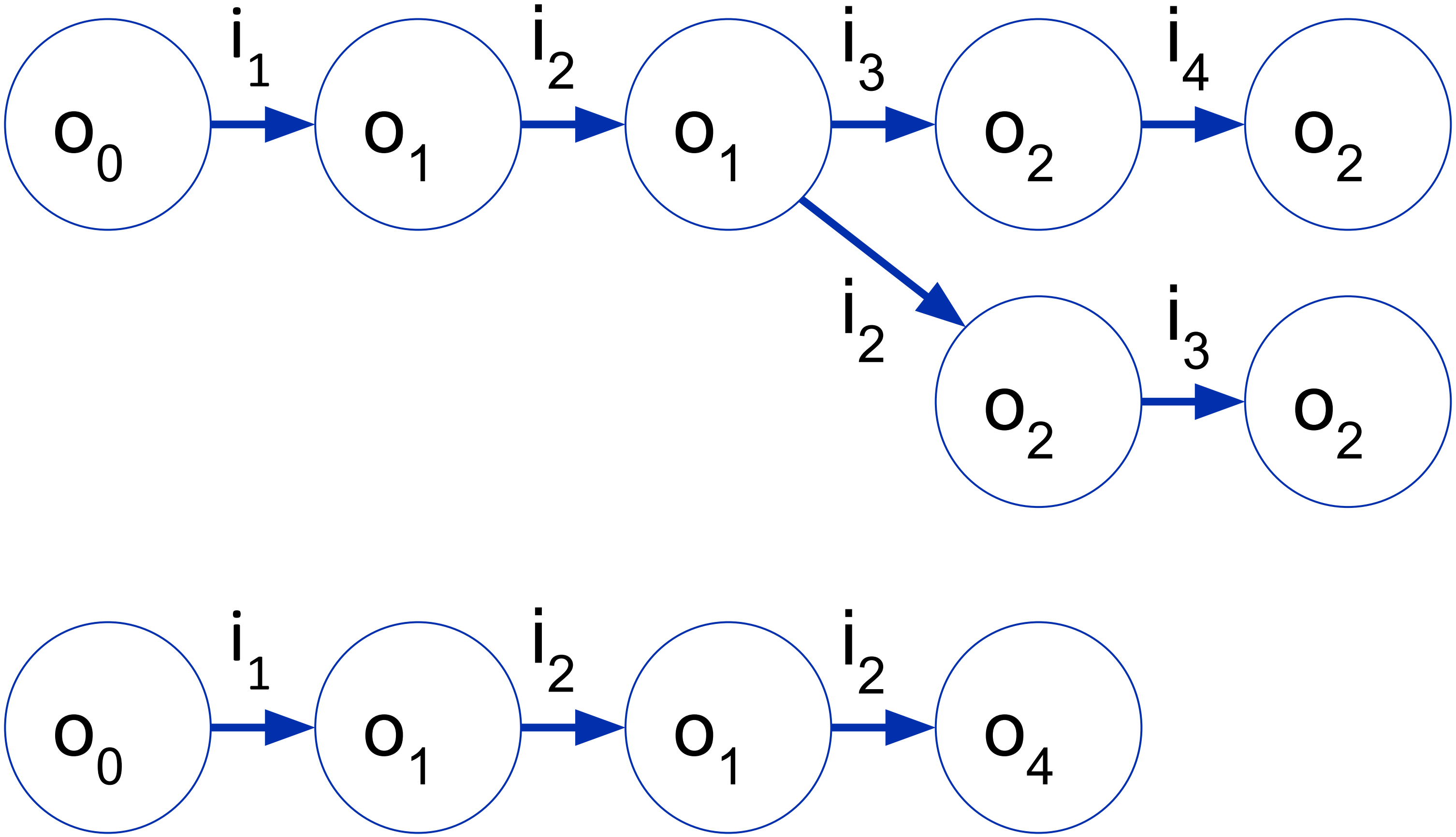
Example



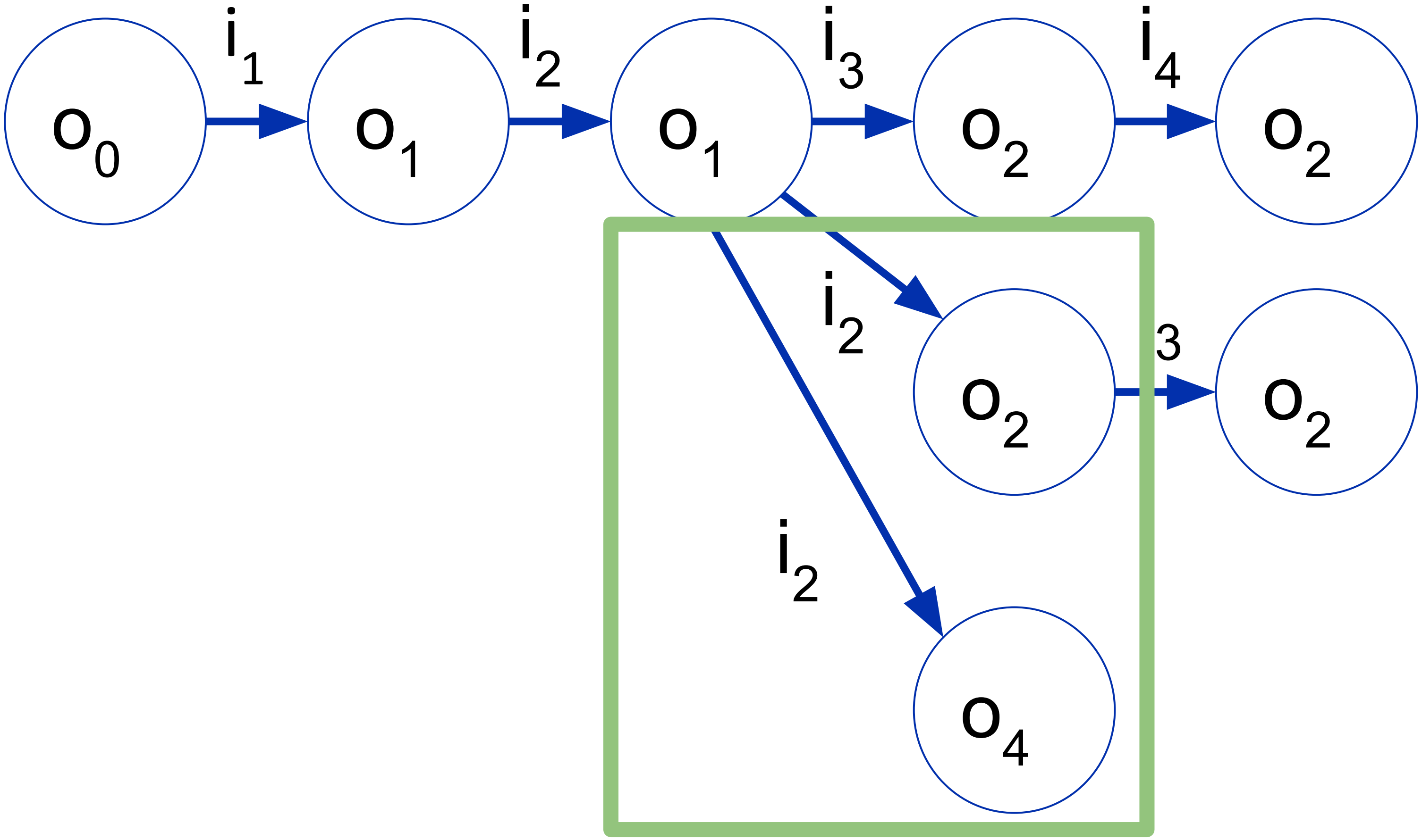
Example



Example



Example: fail!



Difference in scan cycles of PLCs leads to inconsistent traces!

Challenges & approach

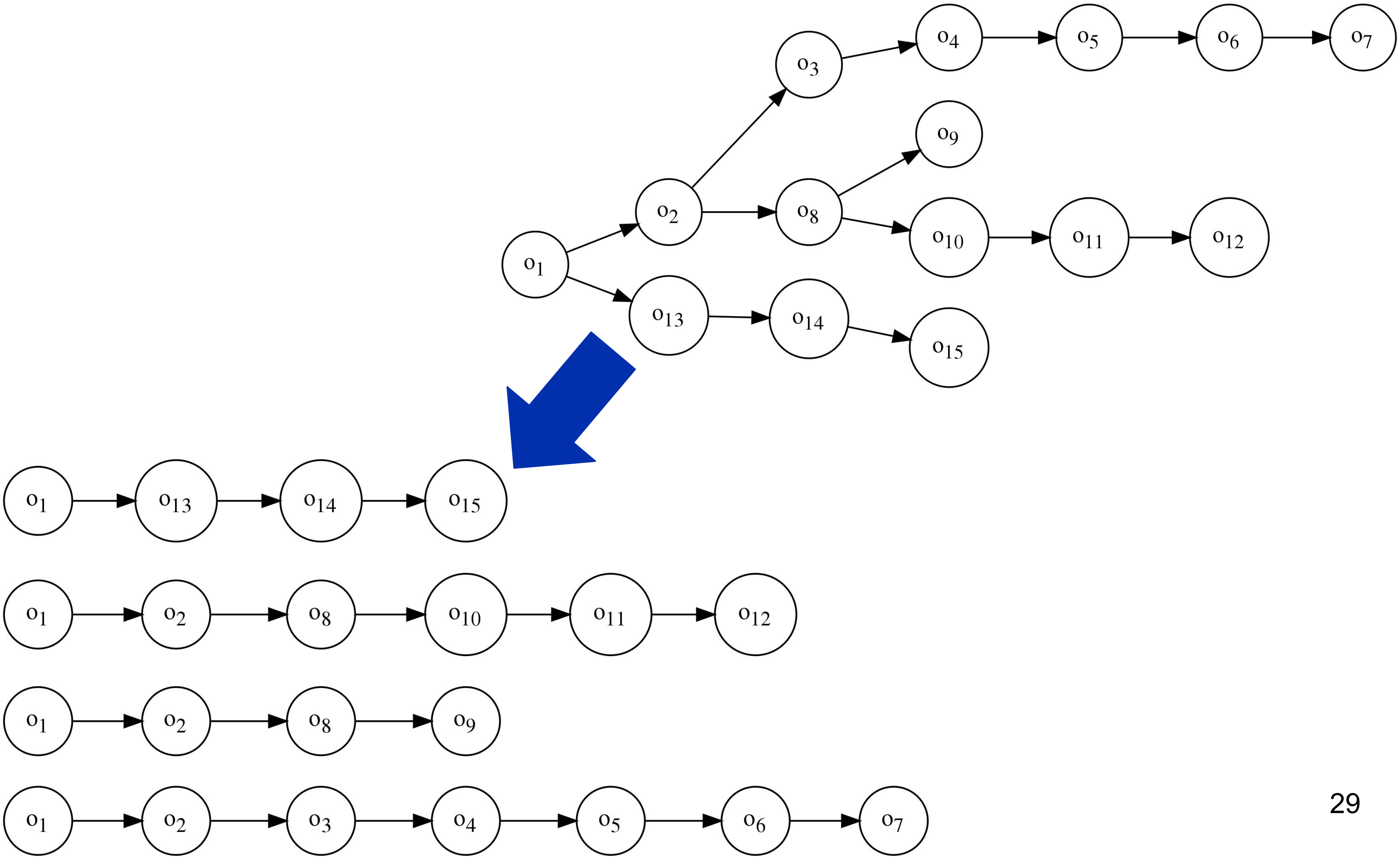
Challenges

1. Traces contain errors due to trace collection procedure
2. We do not know the ground truth

Approach

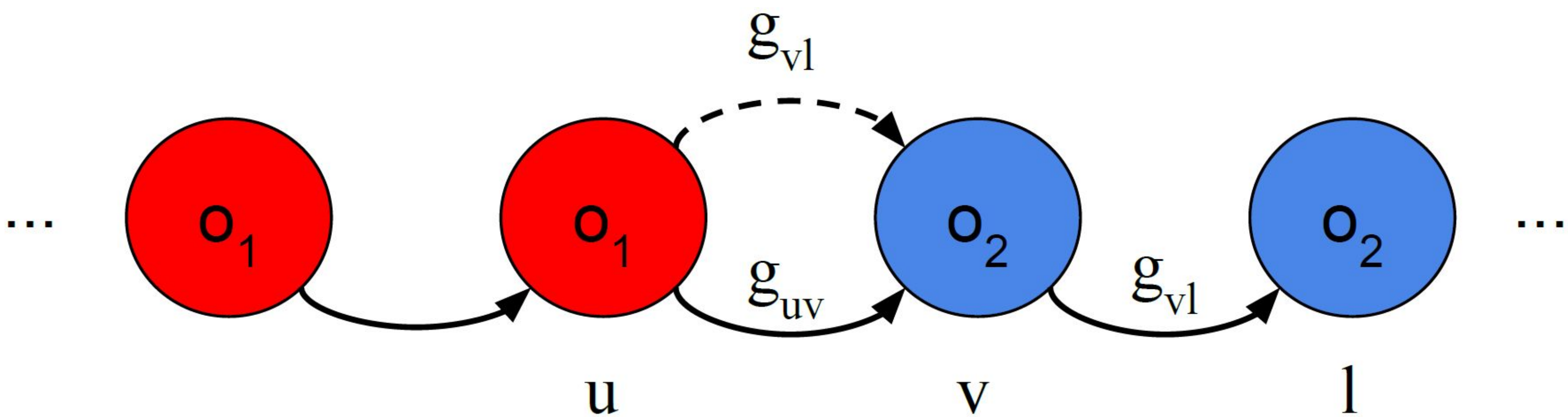
1. Account for errors in the SAT reduction
2. Enumerate all possible solutions

Trace tree → Trace graph



Error model for trace graph

Add multi-edges on the interface between different outputs



- Simple model, richer models are possible
  - Up to fully connected graph in the worst case

Constraints...

$$\mathrm{ALO}_i(c_{v,i}) \leftrightarrow \bigvee_i c_{v,i}$$

$$\mathrm{AMO}_i(c_{v,i}) \leftrightarrow \bigwedge_{i_1<i_2} (\neg c_{v,i_1} \vee \neg c_{v,i_2})$$

$$\bigwedge_{\mathrm{isRoot}(v)} c_{v,0} \qquad \bigwedge_{(u,v,g)\in E} \mathrm{ONE}_g(e_{u,v,g})$$

$$\bigwedge_{(u,v,g)\in E} c_{u,n_1} \wedge c_{v,n_2} \wedge e_{u,v,g} \to y_{n_1,g,n_2}$$

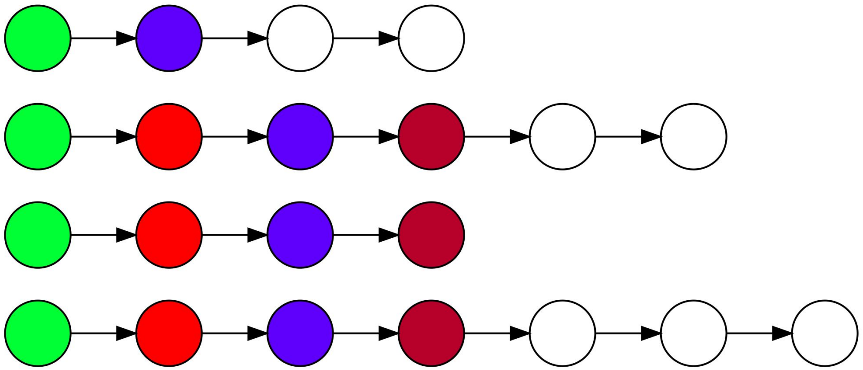
$$\bigwedge_{0\le n<N} \bigwedge_{0\le j<|Z|} \neg d^0_{n,j} \vee \neg d^1_{n,j}$$

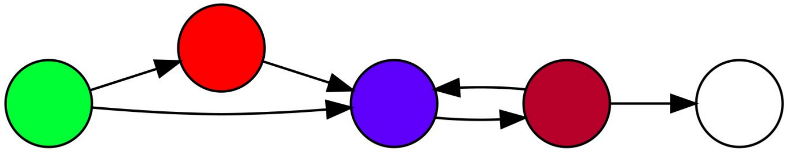
$$\bigwedge_{(u,v,g)\in E} \bigwedge_{0\le n_1,n_2<N} \bigwedge_{0\le j<|Z|} c_{u,n_1} \wedge c_{u,n_2} \wedge \wedge e_{u,v,g} \wedge y_{n_1,g,n_2} \to \begin{cases} d^0_{n_2,j}, & \text{if } z_{u,j} \wedge \neg z_{v,j} \\ \neg d^0_{n_2,j}, & \text{if } z_{u,j} \wedge z_{v,j} \\ d^1_{n_2,i}, & \text{if } \neg z_{u,i} \wedge \neg z_{v,i} \\ \neg d^1_{n_2,i}, & \text{if } \neg z_{u,i} \wedge \neg z_{v,i} \end{cases}$$

$$\bigwedge_{0\le j<i} \left(\neg \bigwedge_{(u,v,g)\in E} e^j_{u,v,g} \right) \qquad \bigwedge_{n_1,g} t_{n_1,g} \leftrightarrow \bigvee_{n_2} y_{n_1,g,n_2} \qquad \bigwedge_{0\le i<N|G|} \neg\tau_{i,R+1}$$

$$\bigwedge_{0\le i\le N|G|} \tau_{i,0} \qquad \bigwedge_{\substack{0\le i\le N \\ 0\le g<|G| \\ 0\le j<N|G|}} \begin{cases} t_{n_1,g} \wedge \tau_{n_1|G|+g-1,j} \to \tau_{n_1|G|+g,j+1} \\ \neg t_{n_1,g} \wedge \tau_{n_1|G|+g-1,j} \to \tau_{n_1|G|+g,j} \end{cases}$$

Color graph nodes in N colors = map graph nodes to automaton states





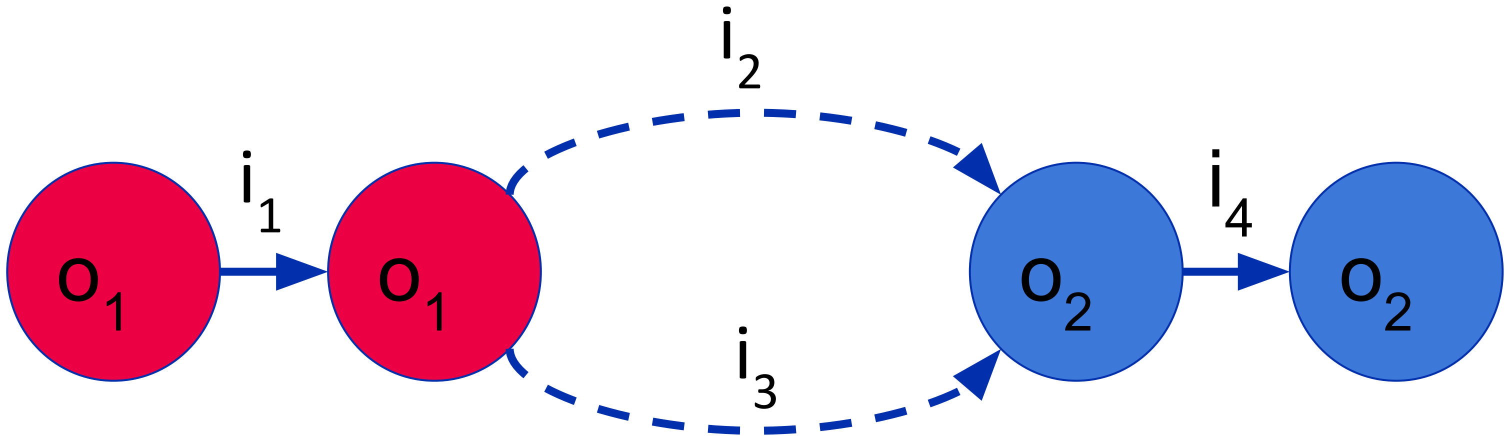
$$\bigwedge_{\mathrm{isRoot}(v)} c_{v,0}$$

$$\mathrm{ALO}_i(c_{v,i}) \leftrightarrow \bigvee_i c_{v,i}$$

$$\mathrm{AMO}_i(c_{v,i}) \leftrightarrow \bigwedge_{i_1<i_2} (\neg c_{v,i_1} \vee \neg c_{v,i_2})$$

$$\bigwedge_{(u,v,g)\in E} c_{u,n_1} \wedge c_{v,n_2} \wedge e_{u,v,g} \rightarrow y_{n_1,g,n_2}$$

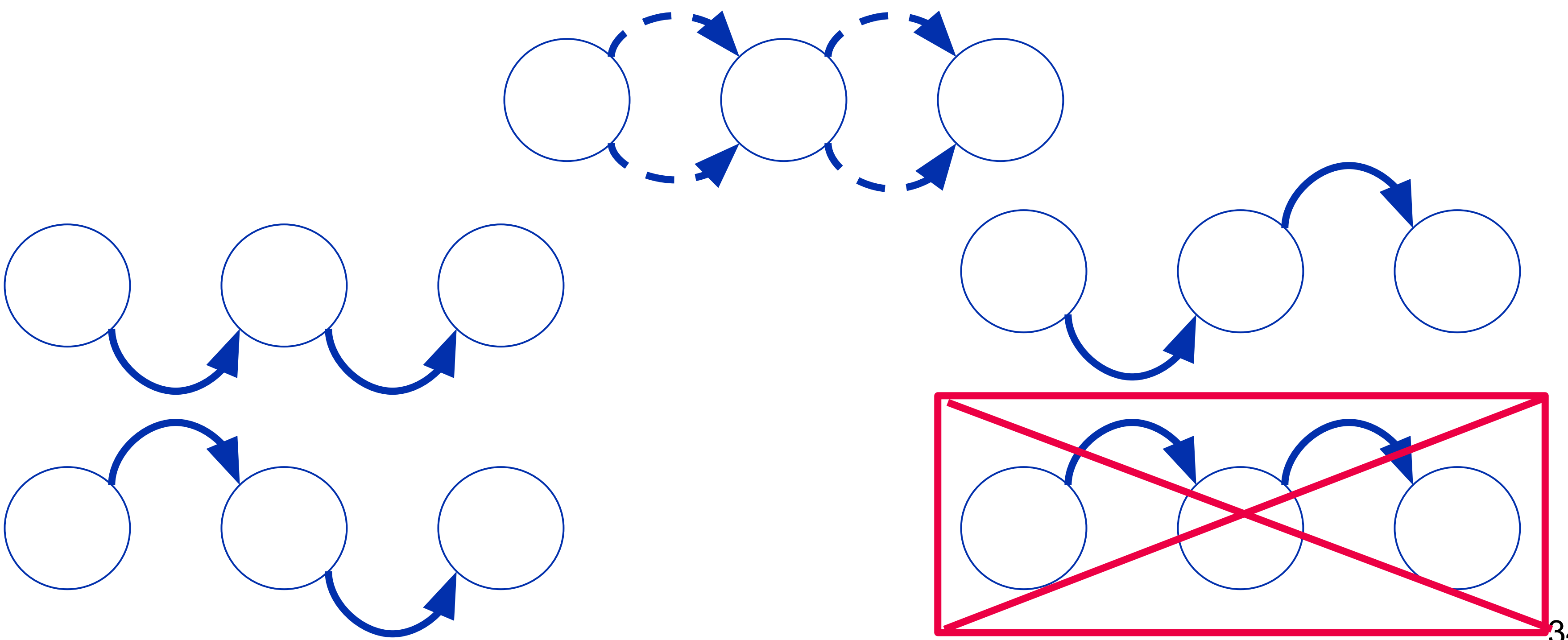
Only one of the multi-edges may be used for each pair of nodes



$$\bigwedge_{(u,v,g)\in E} \mathrm{ONE}_g(e_{u,v,g})$$

Find all solutions with different alternative edge choices

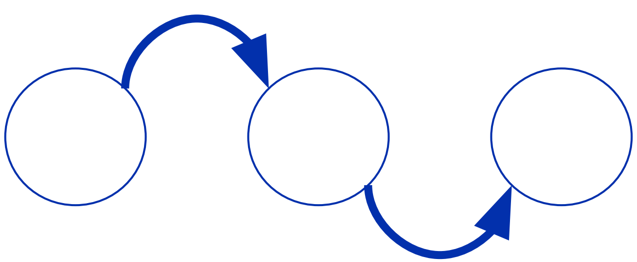
$$\bigwedge_{0 \leq j < i} \left(\neg \bigwedge_{(u,v,g) \in E} e^{j}_{u,v,g} \right)$$

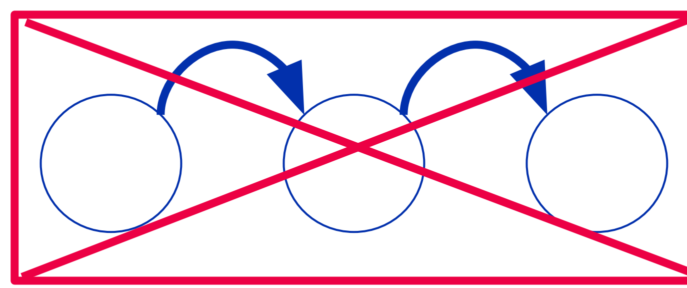


Find all solutions with different alternative edge choices

$$\bigwedge \left(\neg \bigwedge e^{j}_{u,v,g} \right)$$

Still, exponential number of solutions!





Coping with exponential number of solutions

- Zakirzyanov et al. Efficient Symmetry Breaking for SAT-Based Minimum DFA Inference [LATA'19]
- Minimize parameters of state machine
  - N – number of states
  - K – outgoing degree of states
  - R – number of transitions

$$\bigwedge_{n_1,g} t_{n_1,g} \leftrightarrow \bigvee_{n_2} y_{n_1,g,n_2}$$

$$\bigwedge_{\substack{0\le i\le N\\0\le g<|G|\\0\le j<N|G|}} \begin{cases} t_{n_1,g} \wedge \tau_{n_1|G|+g-1,j} \to \tau_{n_1|G|+g,j+1} \\ \neg t_{n_1,g} \wedge \tau_{n_1|G|+g-1,j} \to \tau_{n_1|G|+g,j} \end{cases}$$

$$\bigwedge_{0\le i\le N|G|} \tau_{i,0} \qquad \bigwedge_{0\le i<N|G|} \neg\tau_{i,R+1}$$

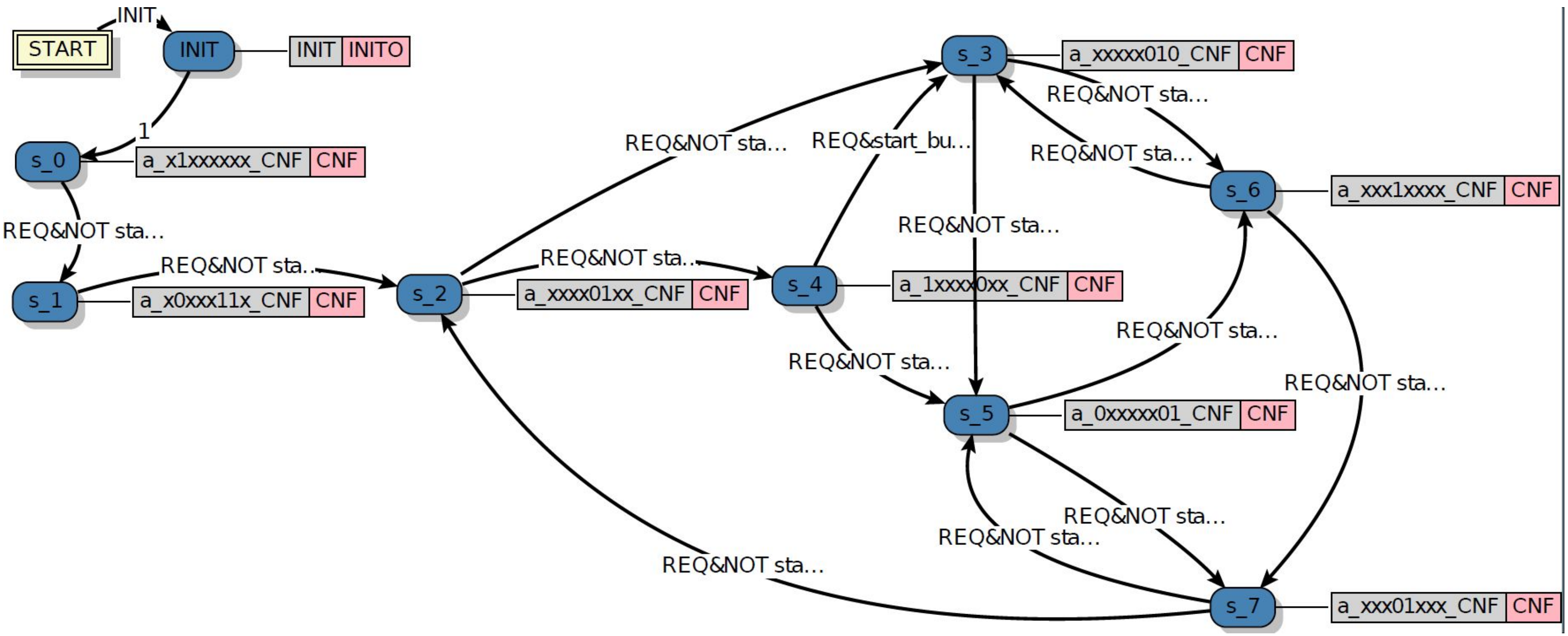
Algorithm

```
Data: PLC execution traces 𝕋
/* search for minimal number N                                  */
N ← 1
while findModel(𝕋, N) = ∅ do
  └ N ← N + 1
/* search for minimal number K                                  */
K ← 1
while findModel(𝕋, N, K) = ∅ do
  └ K ← K + 1
/* search for minimal number of
   transitions R                                                */
R ← 1
while findModel(𝕋, N, K, R) = ∅ do
  └ R ← R + 1
/* find all minimal solutions
   consistent with traces                                       */
S ← list()
while True do
  | A ← findModel(𝕋, N, K, R)
  | if A ≠ ∅ then S.add(A)
  | else
  |   └ break
return S
```

Experiment with distribution station

- 12 inputs, 8 outputs
- Six logs for different use cases with varying complexity and length of runs
- Algorithm found 63 different state machines that satisfy the traces with respect to the error model
- Launch simulation of use cases in NxtStudio
- Only one (!) state machine was truly correct

Generated state machine



Conclusion & Future work

- Developed hardware and software architecture for data collection from PLC
- Developed algorithm for reconstructing state machine from (noisy) PLC traces

Future work

- Improve synthesis algorithm
- Automate validation against legacy system (model)
- Improve data collection, add time synchronization
- Target distributed controller reconstruction
- Move data storage and synthesis to the cloud

ITMO UNIVERSITY

Thank you!

Daniil Chivilikhin, chivdan@itmo.ru