

Specification mining for asynchronous controllers

Javier de San Pedro[†]

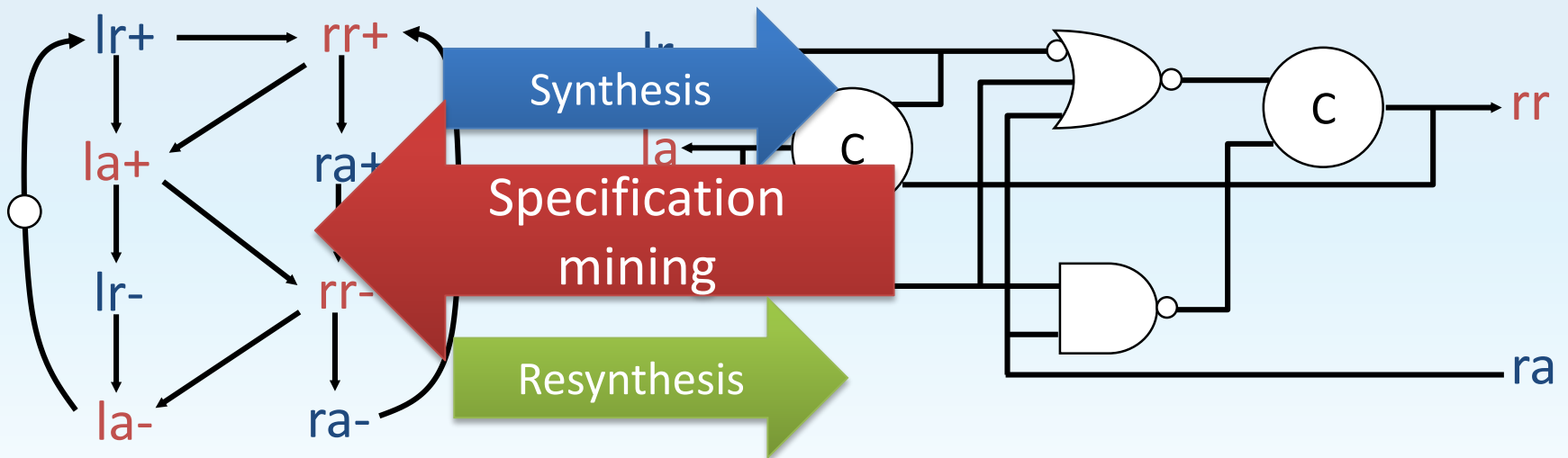
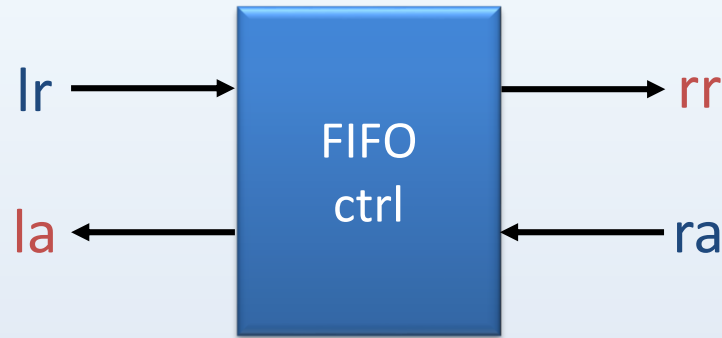
Thomas Bourgeat [‡]

Jordi Cortadella[†]

[†] Universitat Politecnica de Catalunya

[‡] Massachusetts Institute of Technology

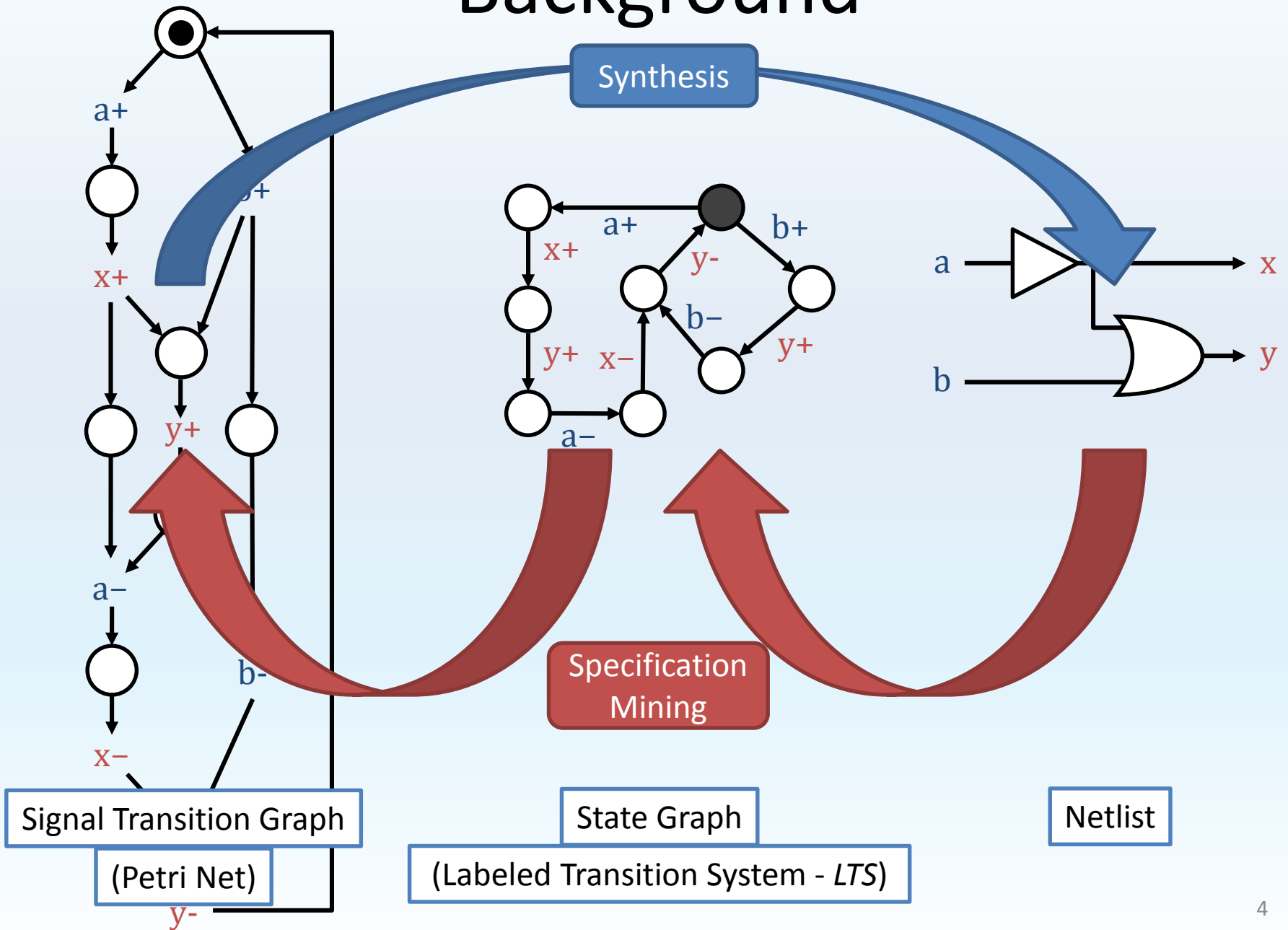
Specification mining



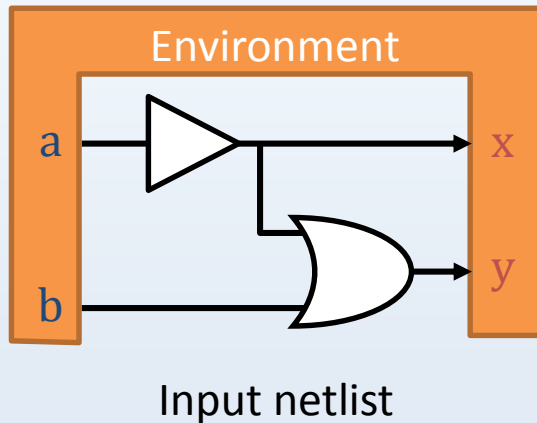
Outline

- Overview of specification mining
- Modeling behavioral properties
- Mining algorithm
- Results and conclusions

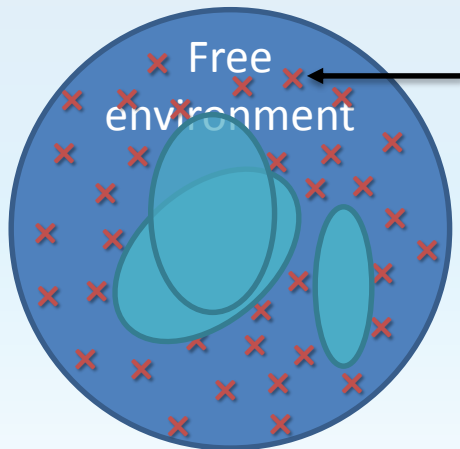
Background



Specifications with properties

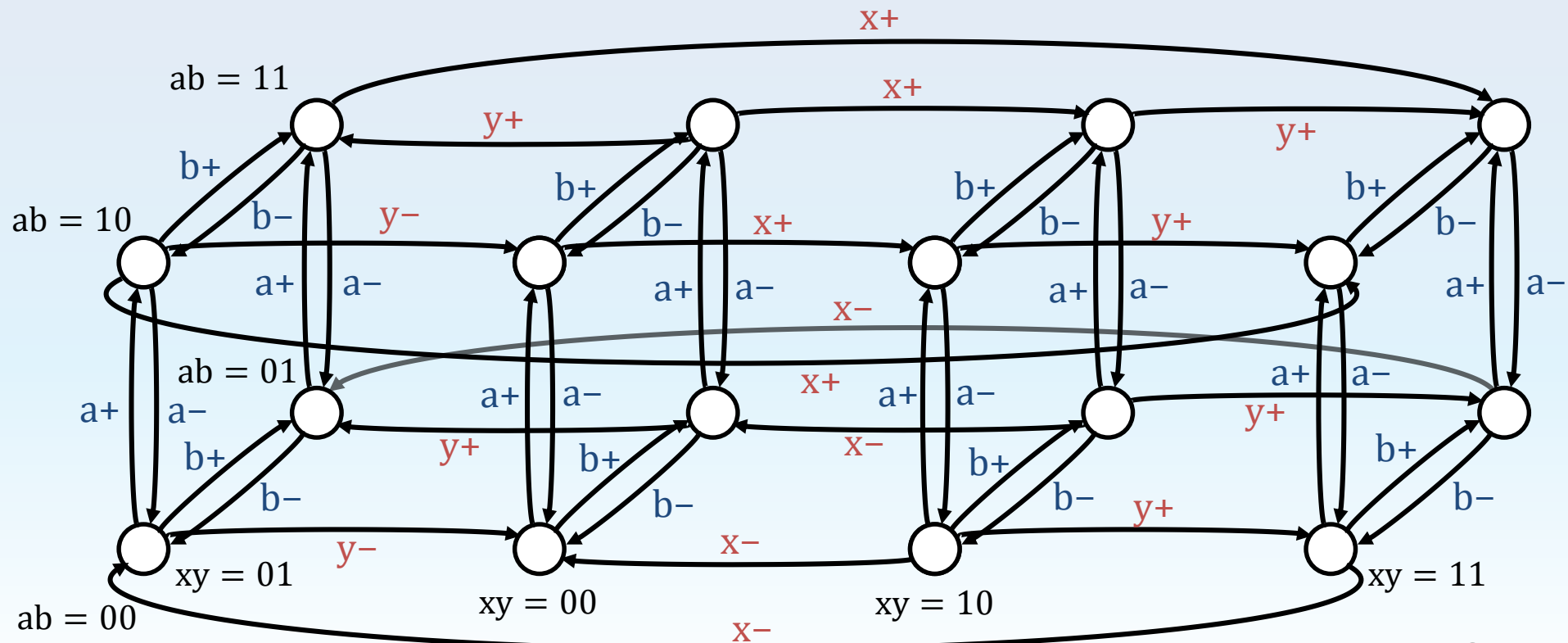
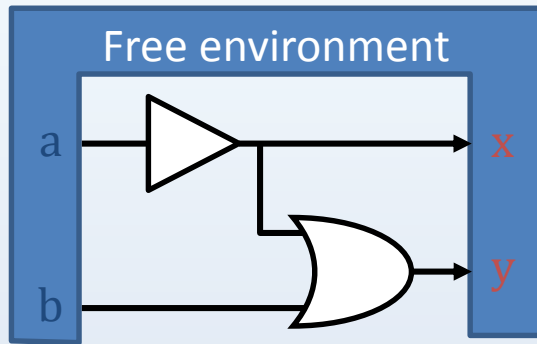


- Mined specifications must satisfy a set of behavioral *properties*
 - E.g., only show hazard-free behavior
- Nothing is known about the environment
 - Initially assume free (unrestricted) env.
 - Circuit may exhibit hazards under free env.

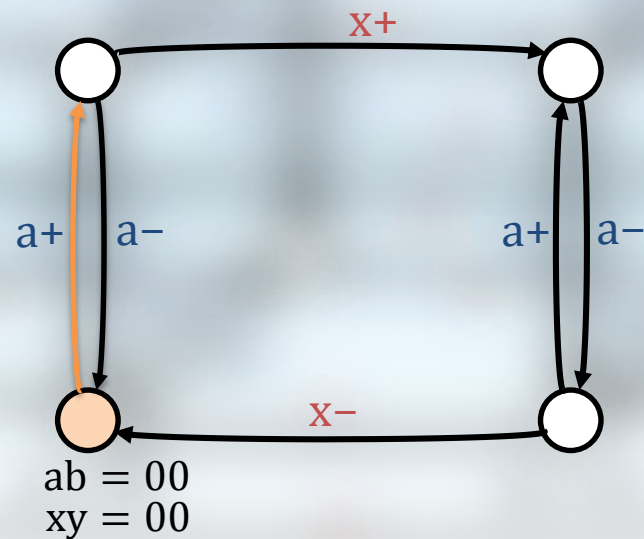
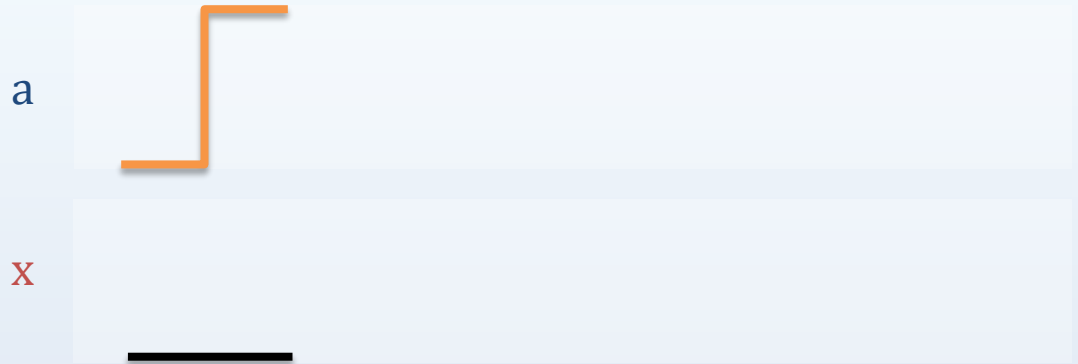
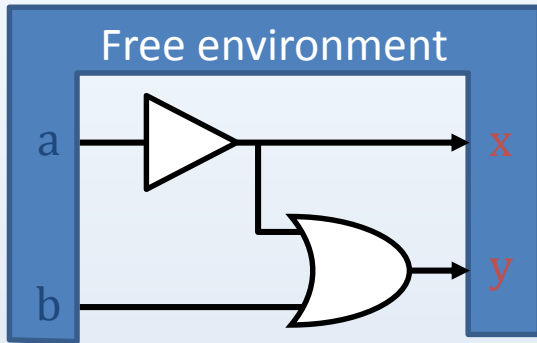


- **Hazard** To ensure the specification satisfies the set of properties...
 - Circuit behavior cannot be changed
 - Discover *constrained* environment where the circuit satisfies all properties

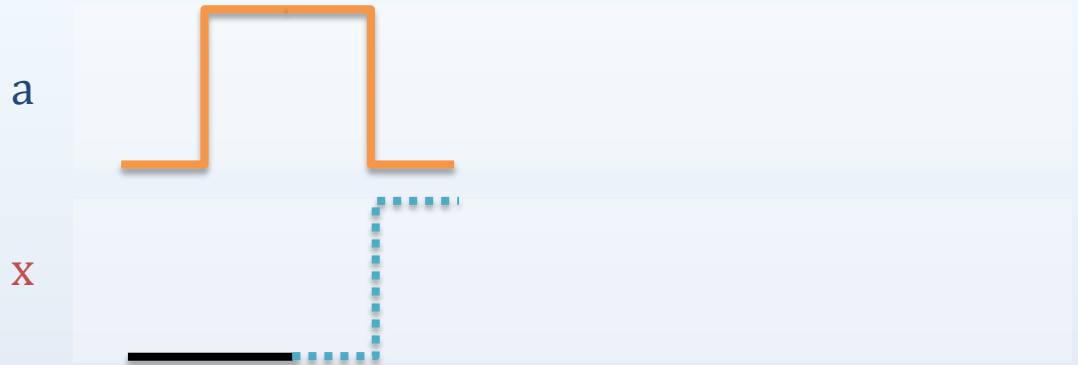
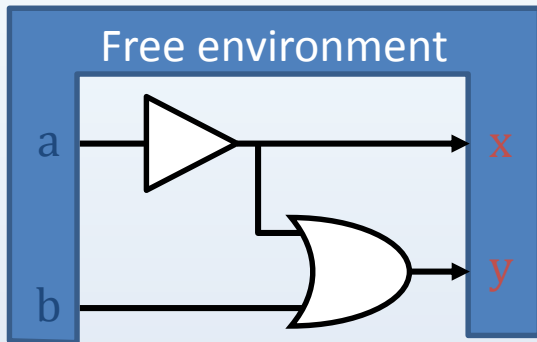
LTS under free environment



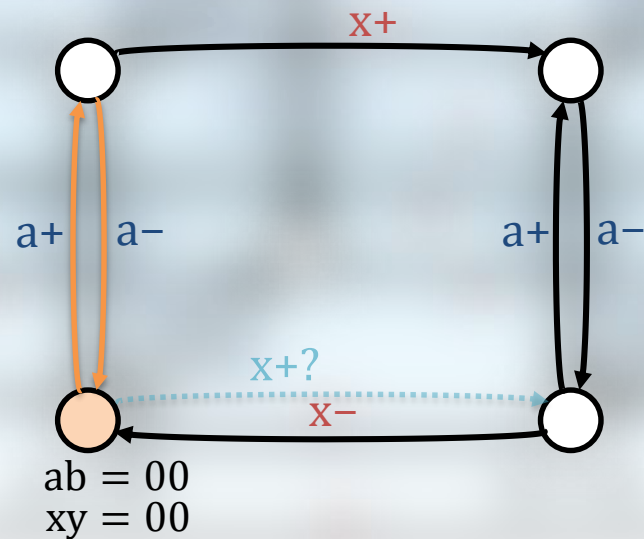
LTS under free environment



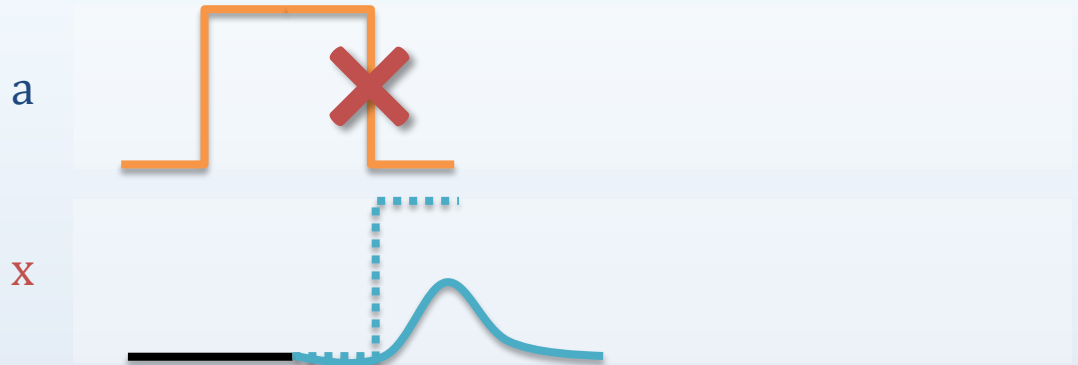
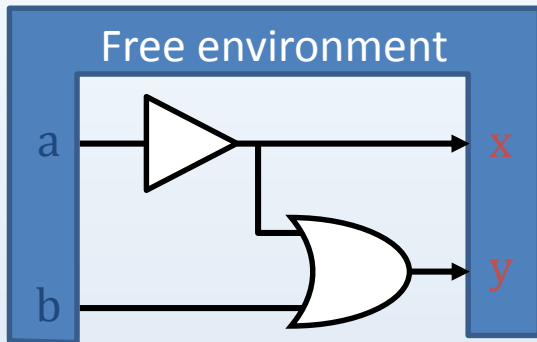
LTS under free environment



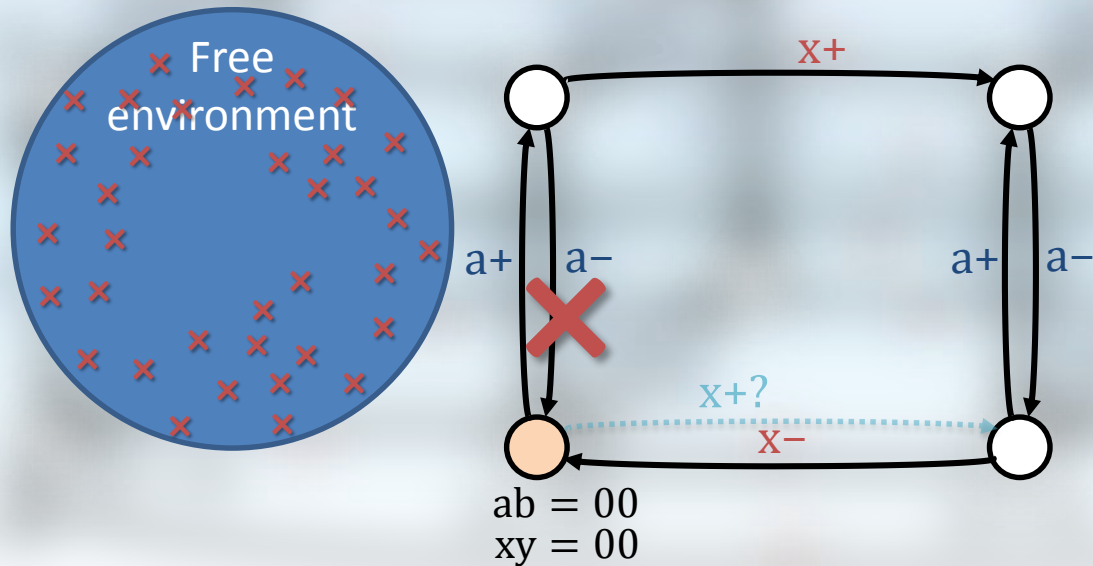
$x+$ is not *persistent* under free environment



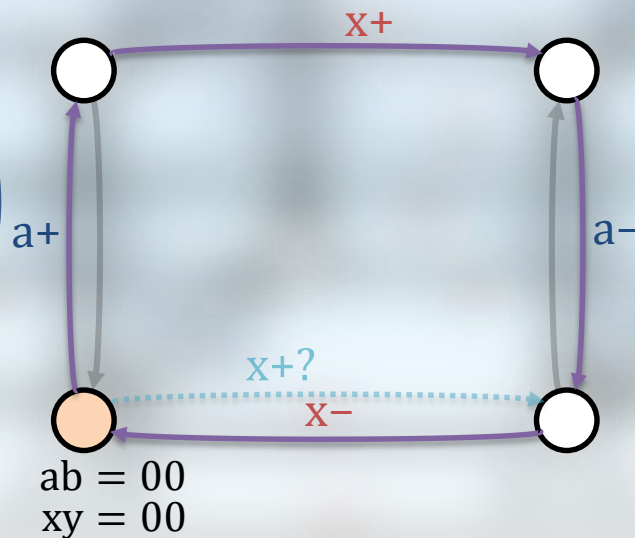
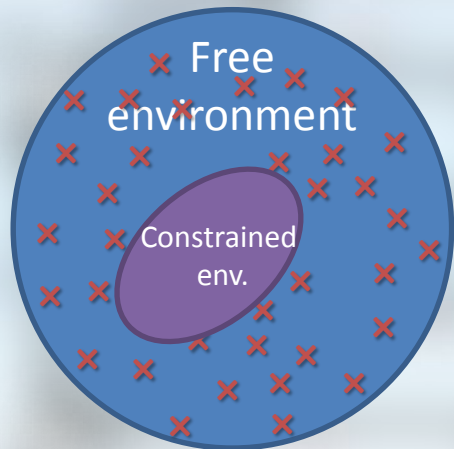
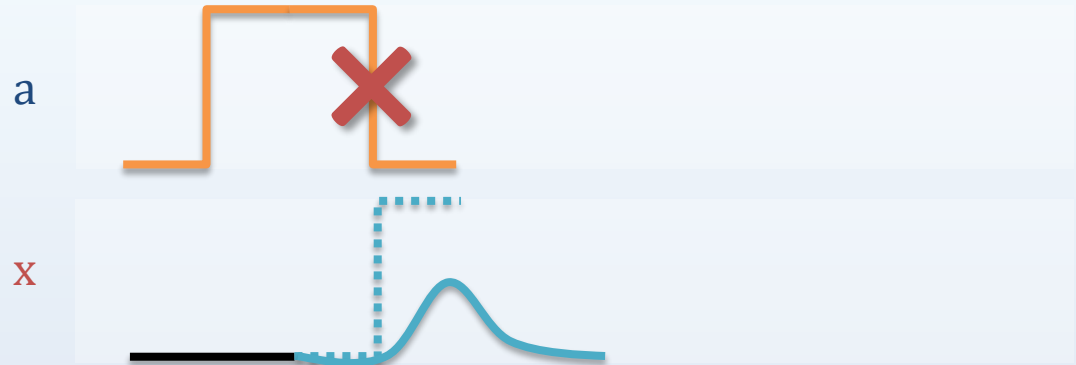
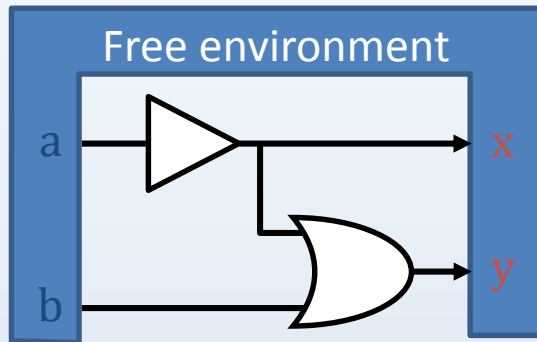
LTS under free environment



$x+$ is not *persistent* under free environment

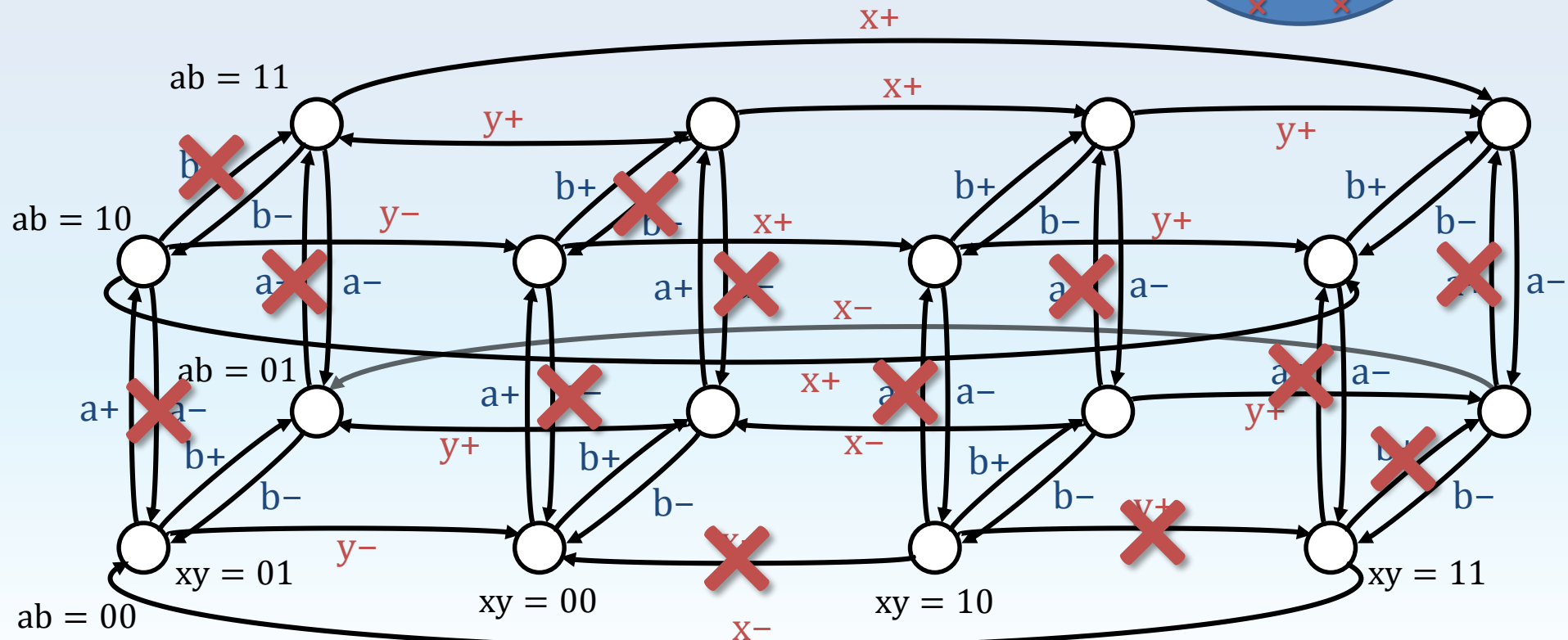
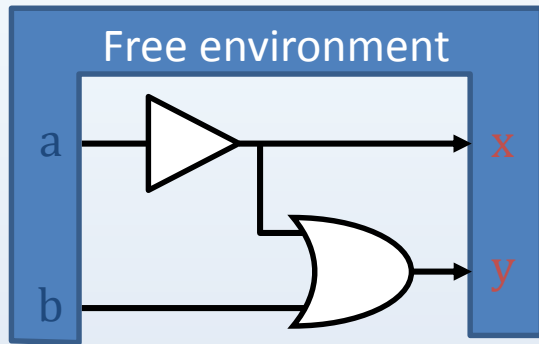


LTS under free environment

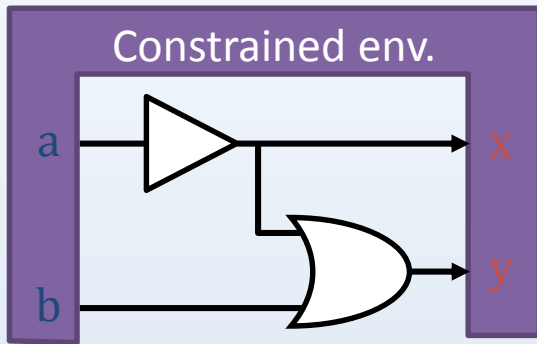


Select a **subset** of *input transition* arcs that satisfy all behavioral properties

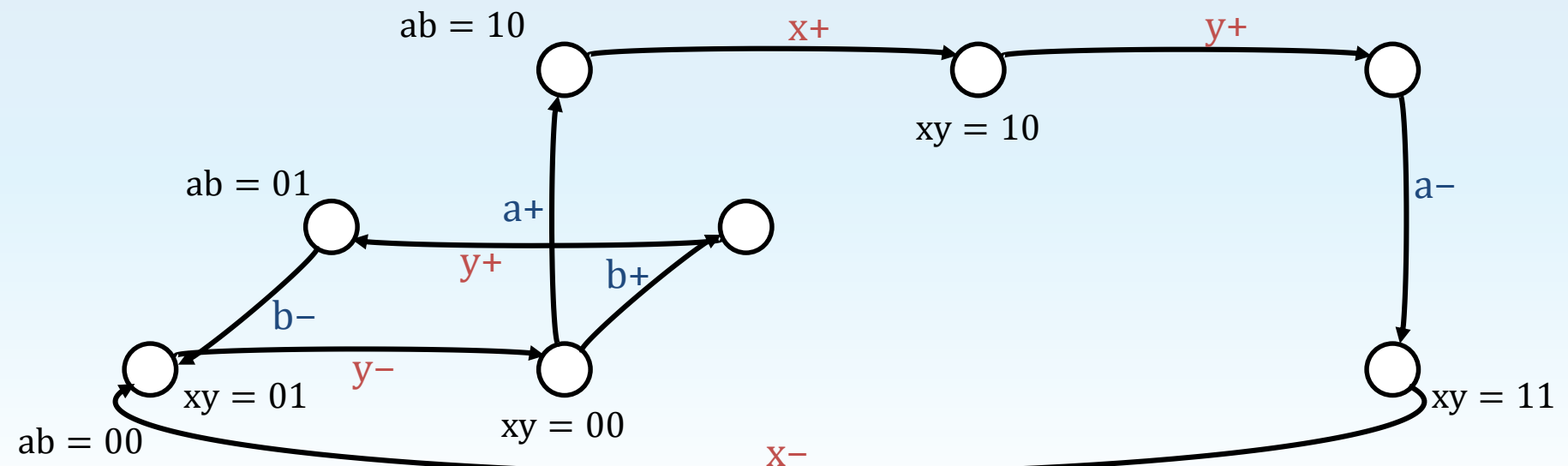
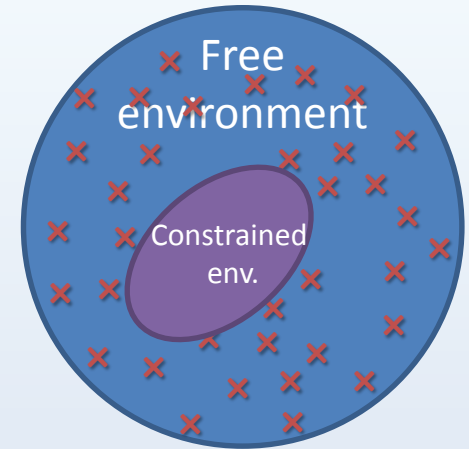
Hazards under free environment



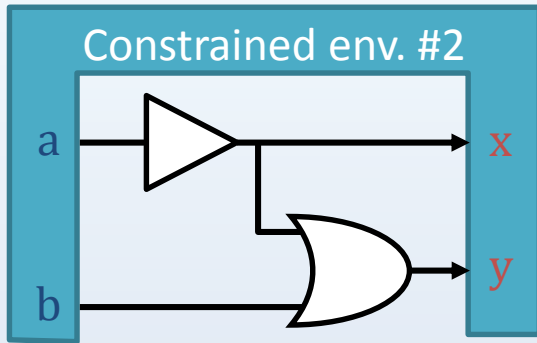
Constraining the environment



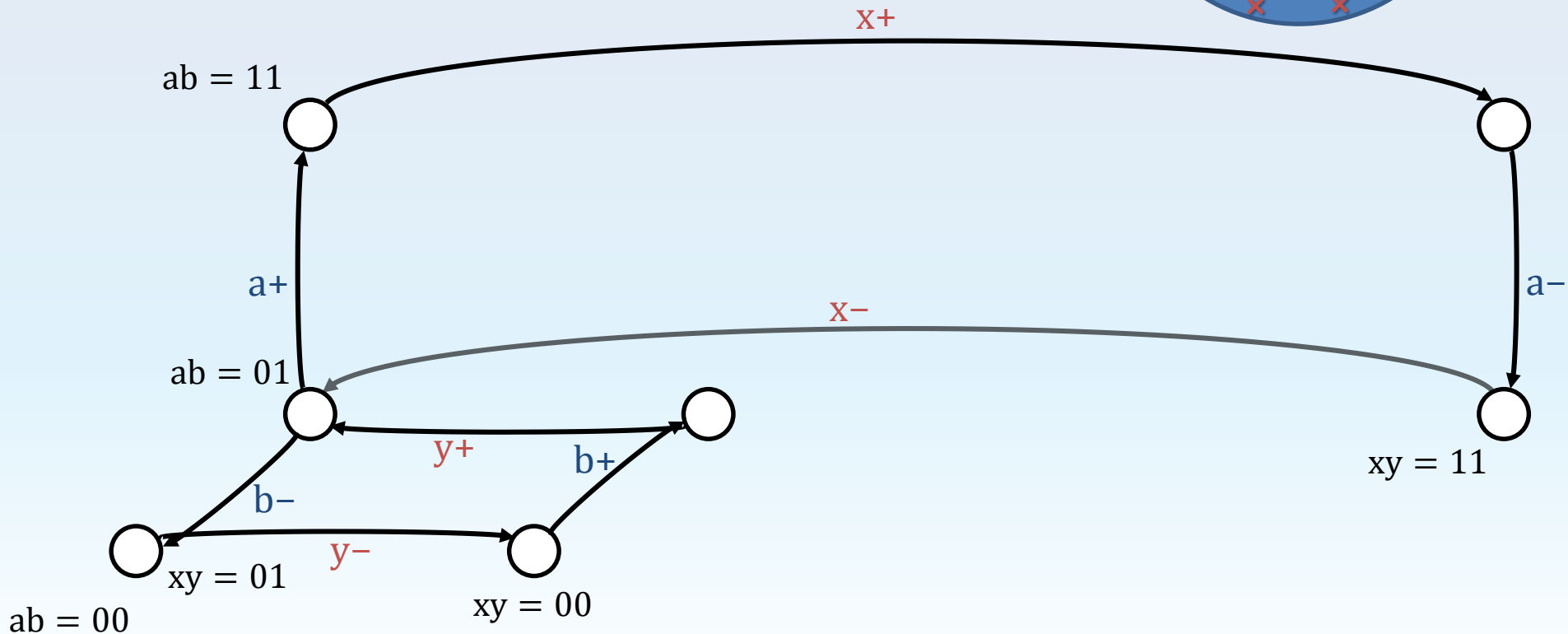
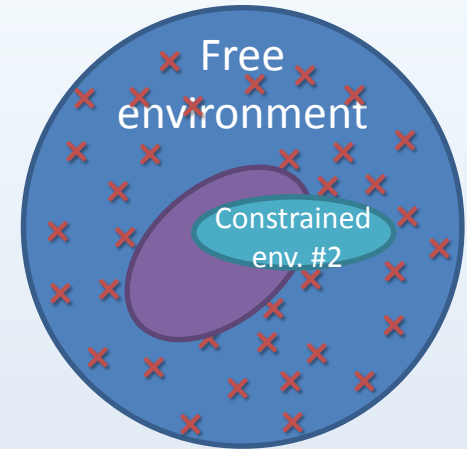
Select a **subset** of *input transition* arcs that satisfy all behavioral properties



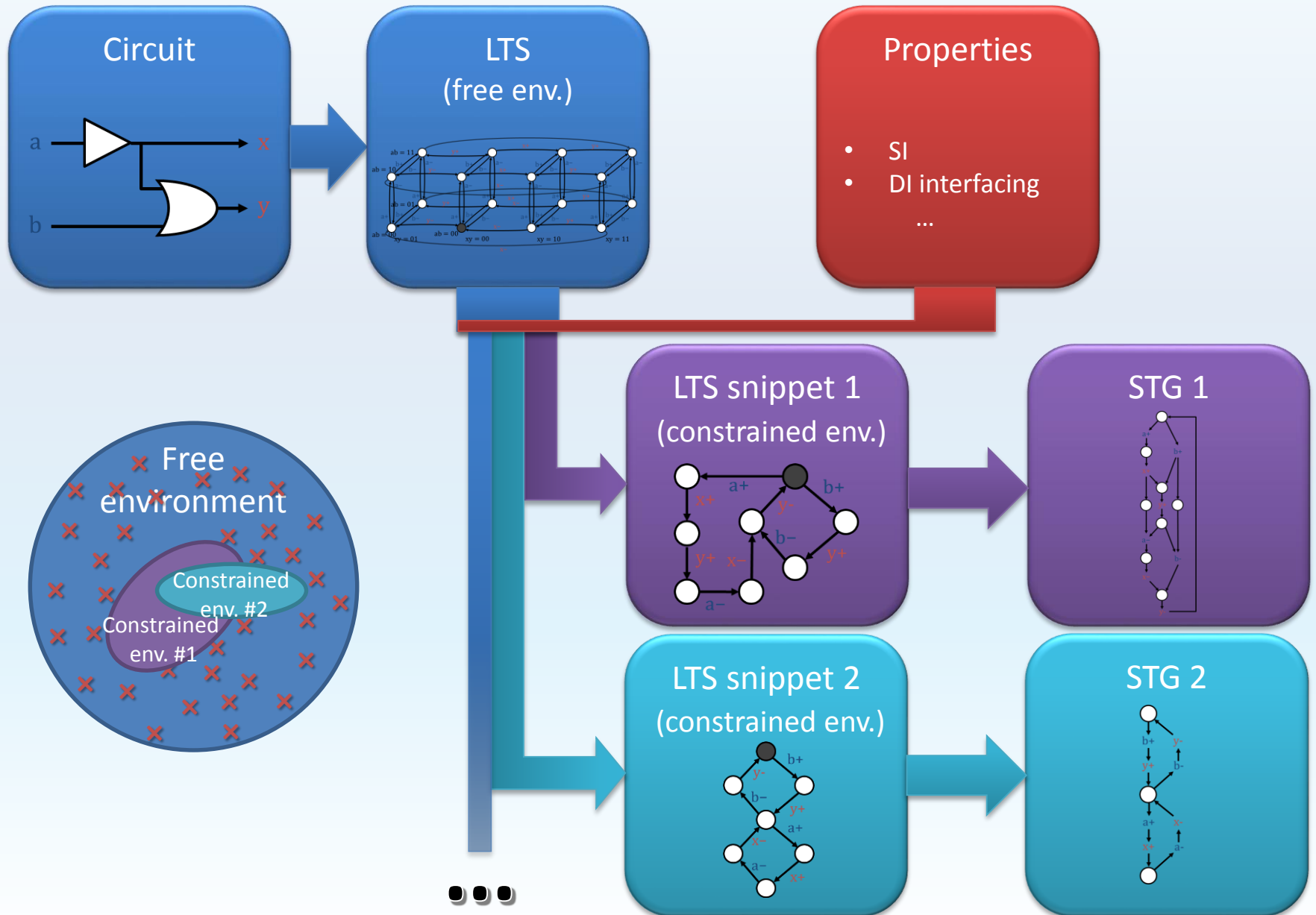
More than one specification



There might more than one environment that satisfies the properties



Overview of mining flow



Outline

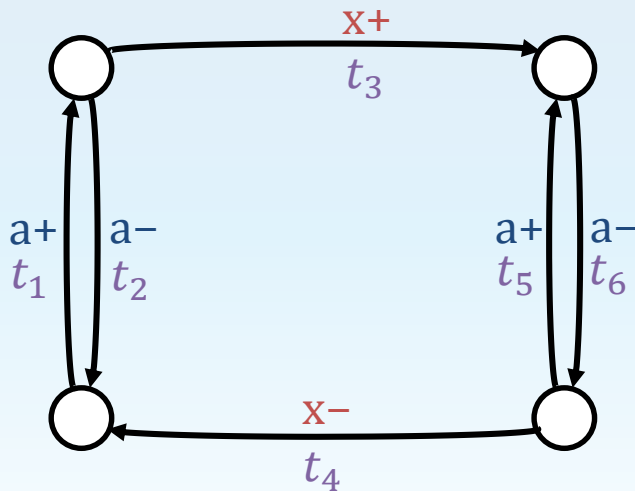
- Overview of specification mining
- **Modeling behavioral properties**
- Mining algorithm
- Results and conclusions

Properties

- Behavioral properties
 - Speed-independence
 - Delay-insensitive interfacing
 - Multiple independent environments
 - ...
 - Properties of the specification model
 - Marked Graph
 - Free Choice
 - ...
- All properties modeled using SAT

Modeling properties using SAT

- Finding a *subset* of the LTS which satisfies properties
- A Boolean variable t_i for every *transition* of the LTS
 - Indicates if transition is part of subset
- Desired properties modeled as SAT formulas

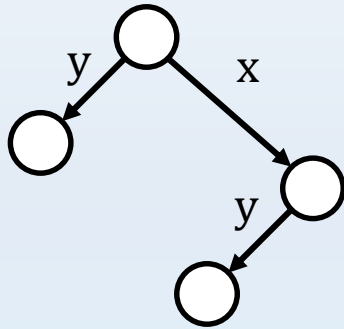


- Example property:
 - Cannot remove output transition arcs except if state is unreachable

➡ $t_1 \Rightarrow t_3$

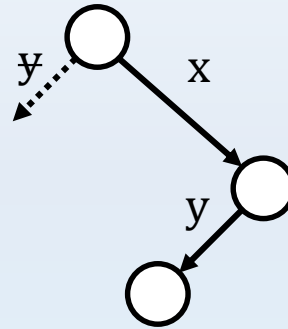
➡ $t_6 \Rightarrow t_4$

Causality patterns in LTS



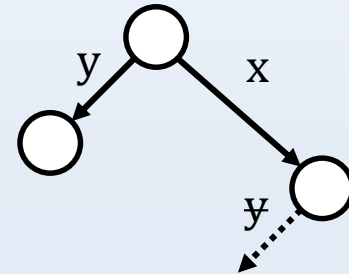
y is persistent

Concurrency



x triggers y

Order



x disables y

Conflict/choice

Behavioral properties

- Speed-independence (SI)
 - Insensitive to gate delays
 - **Property:** Only inputs can disable inputs
- Delay insensitive interfacing (DII)
 - Insensitive to arrival order *of input transitions*
 - **Property:** Inputs cannot trigger other inputs

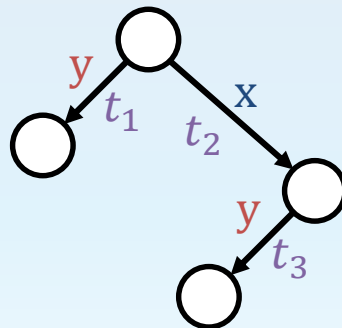
x	y	x triggers y	x disables y
Input	Input	Violates DII	
Output	Output		Violates SI
Input	Output		Violates SI
Output	Input		Violates SI

H. Saito, A. Kondratyev, J. Cortadella,
L. Lavagno and A. Yakovlev,
What is the cost of delay insensitivity?
ICCAD 1999

Example: speed-independence

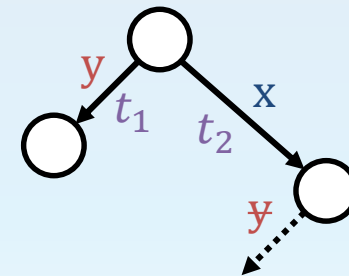
x	y	x triggers y	x disables y
Input	Input	Violates DII	
Output	Output		Violates SI
Input	Output		Violates SI
Output	Input		Violates SI

- For every state, for every pair (x, y) where **x** input, **y** output:



y is persistent

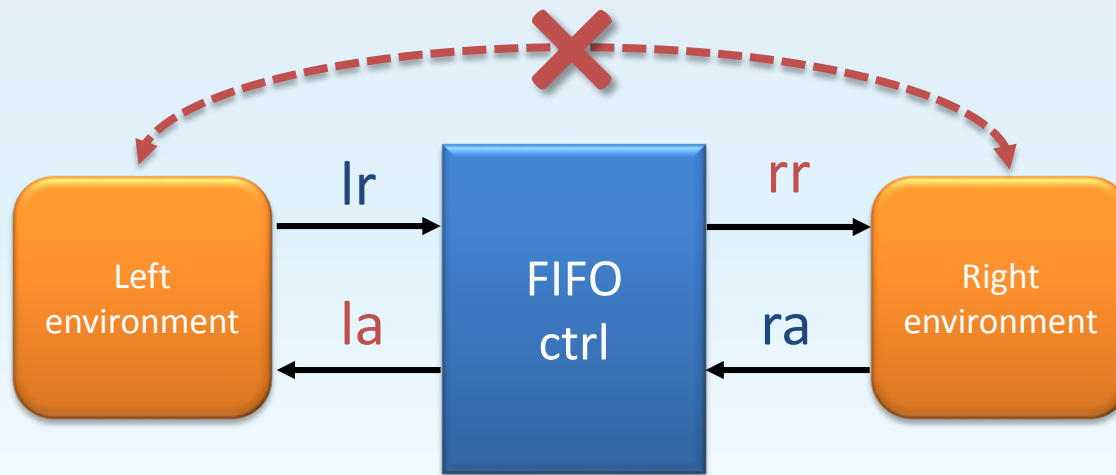
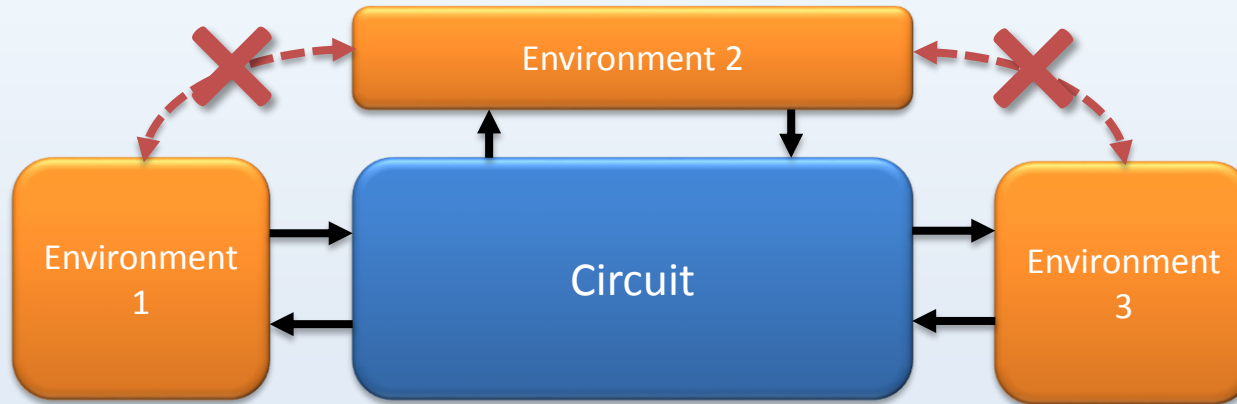
➡ $t_1 \wedge t_2 \Rightarrow t_3$



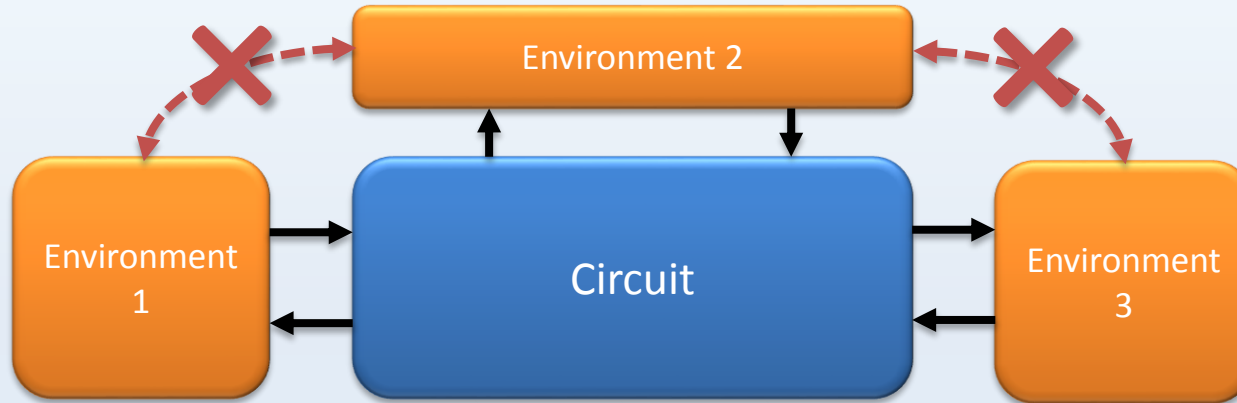
x disables y

➡ $\neg t_1 \vee \neg t_2$

Multi-environments



Multi-environments



x	y	x triggers y	x disables y
Input	Input	Violates DII	Only if same env.
Output	Output		Violates SI
Input	Output		Violates SI
Output	Input	Only if same env.	Violates SI

Properties of the specification model

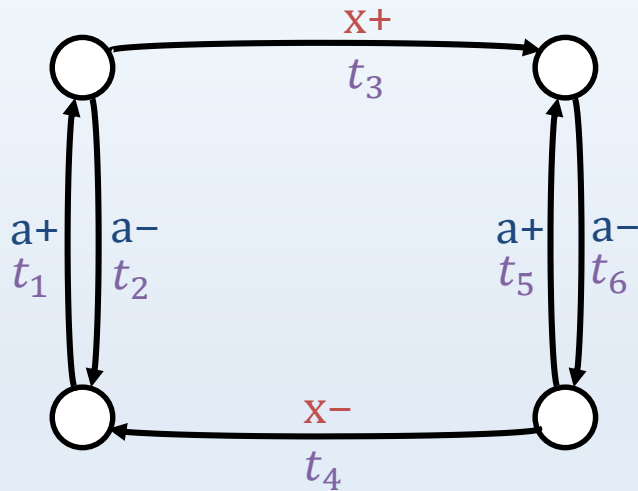
- Enforce specific structural types of STGs
 - Enhances visualization, computational cost, ...
- Marked graph
 - **Property:** No two signals are in conflict
- Free choice
 - **Property:** Signals in conflict must have the same excitation set (see article)

E. Best and R. Devillers,
*Characterization of the state spaces of live
and bounded marked graph Petri nets,*
Language and Automata Theory 2014.

Outline

- Overview of specification mining
- Modeling behavioral properties
- **Mining algorithm**
- Results and conclusions

SAT model



- SAT constraints:

$$t_1 \Rightarrow t_3$$

$$t_6 \Rightarrow t_4$$

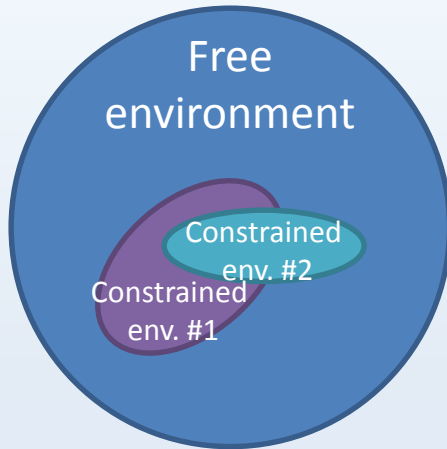
$$\neg t_2 \vee \neg t_3$$

$$\neg t_2 \vee \neg t_4$$

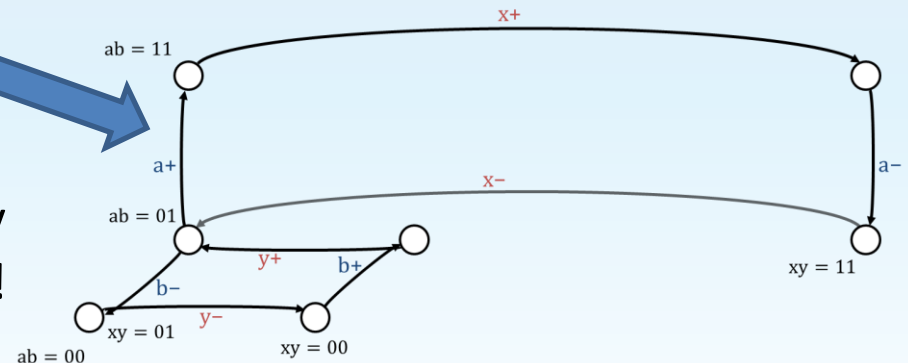
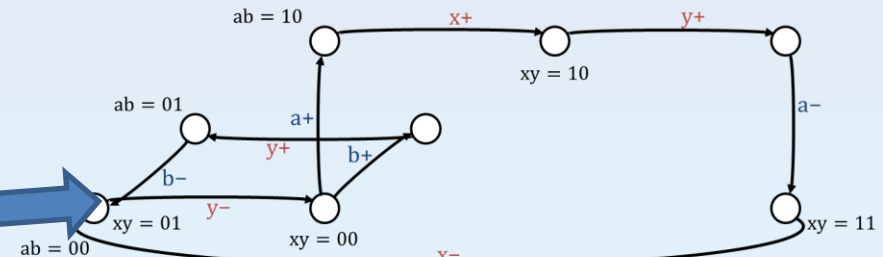
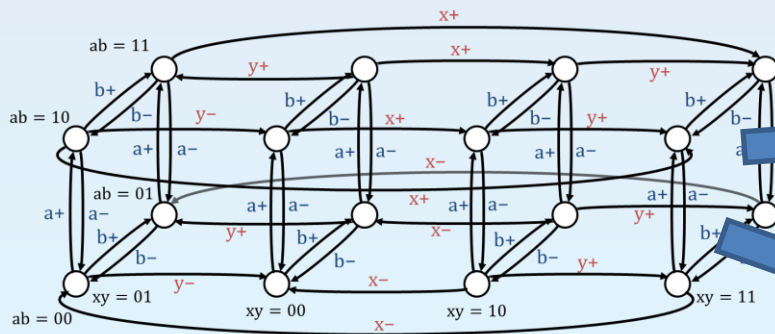
...

- Constraints ensure properties are satisfied *in every state*
- Each assignment = LTS subset that satisfies all properties
 - $\{t_1, t_3, t_6, t_4\}$
 - $\{\}$
- *MaxSAT*: Maximize $\sum t_i$
 - Only consider *maximal* subsets

Mining algorithm: MaxSAT

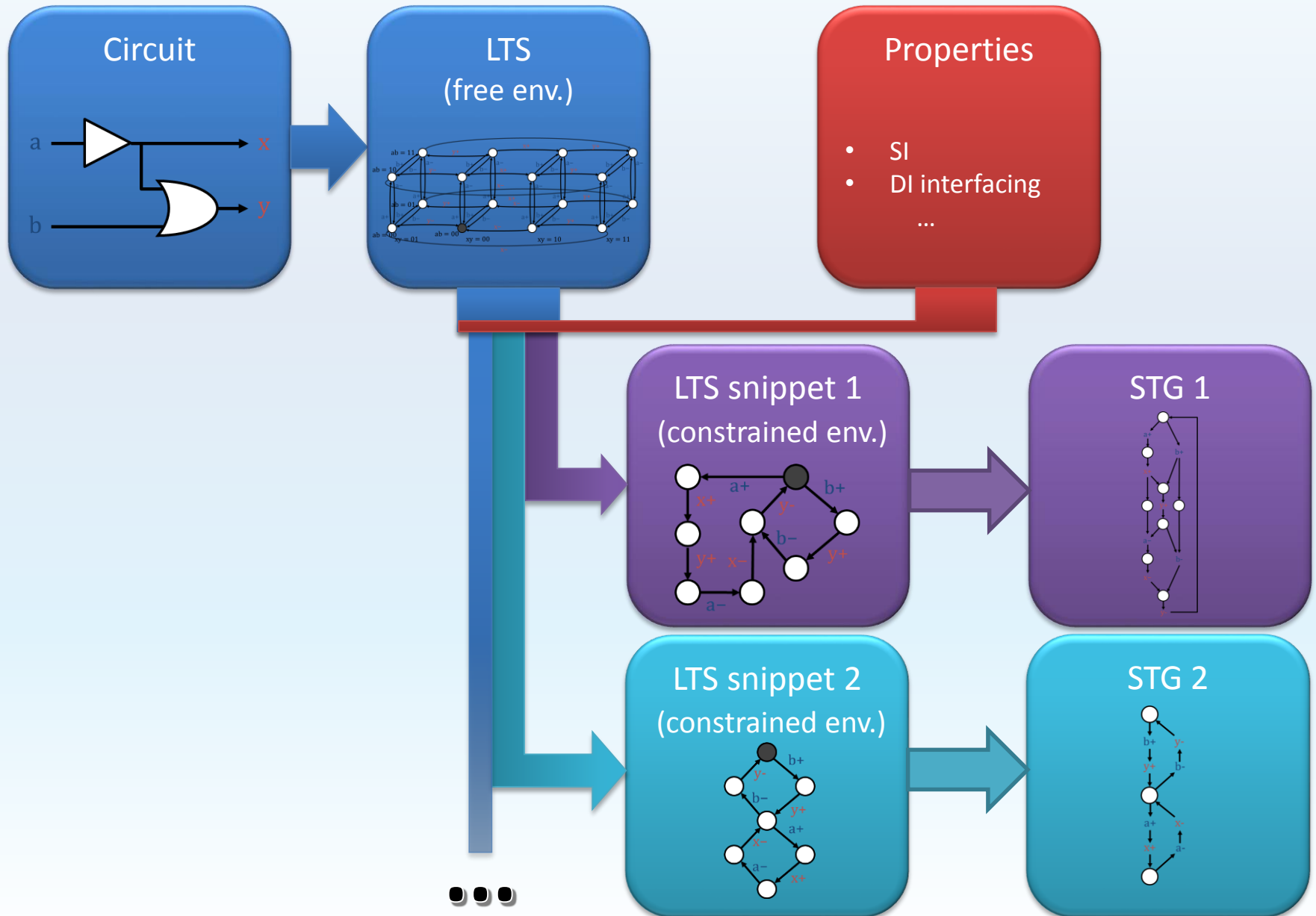


- No single specification may cover all behavior without violating some property
 - E.g., marked graphs
- Discover *a set of maximal* specifications



Maximize $\sum t_i$
 s.t. *properties* Except those already
 in previous snippets!

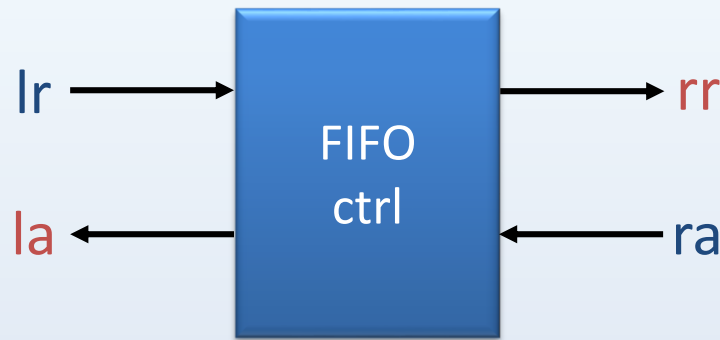
Overview of mining flow



Outline

- Overview of specification mining
- Modeling behavioral properties
- Mining algorithm
- **Results and conclusions**

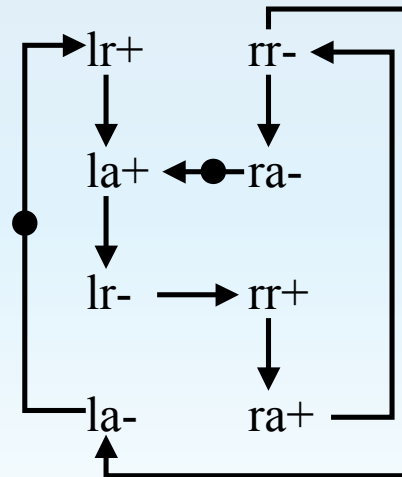
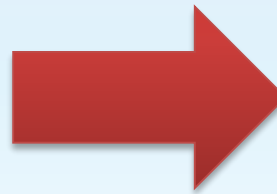
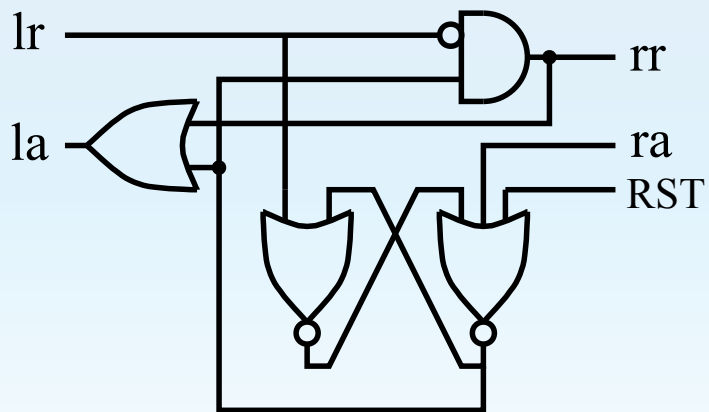
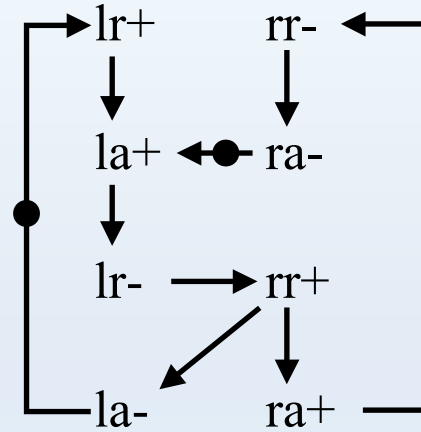
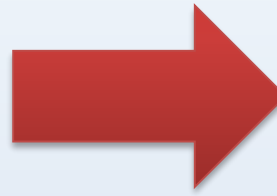
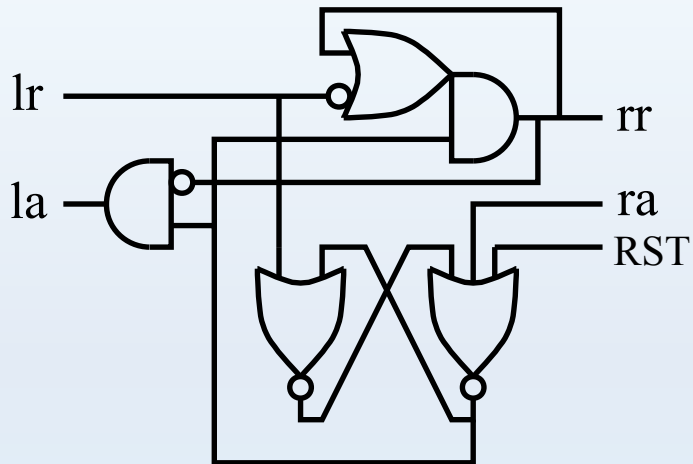
Case study: 4-phase latch controller



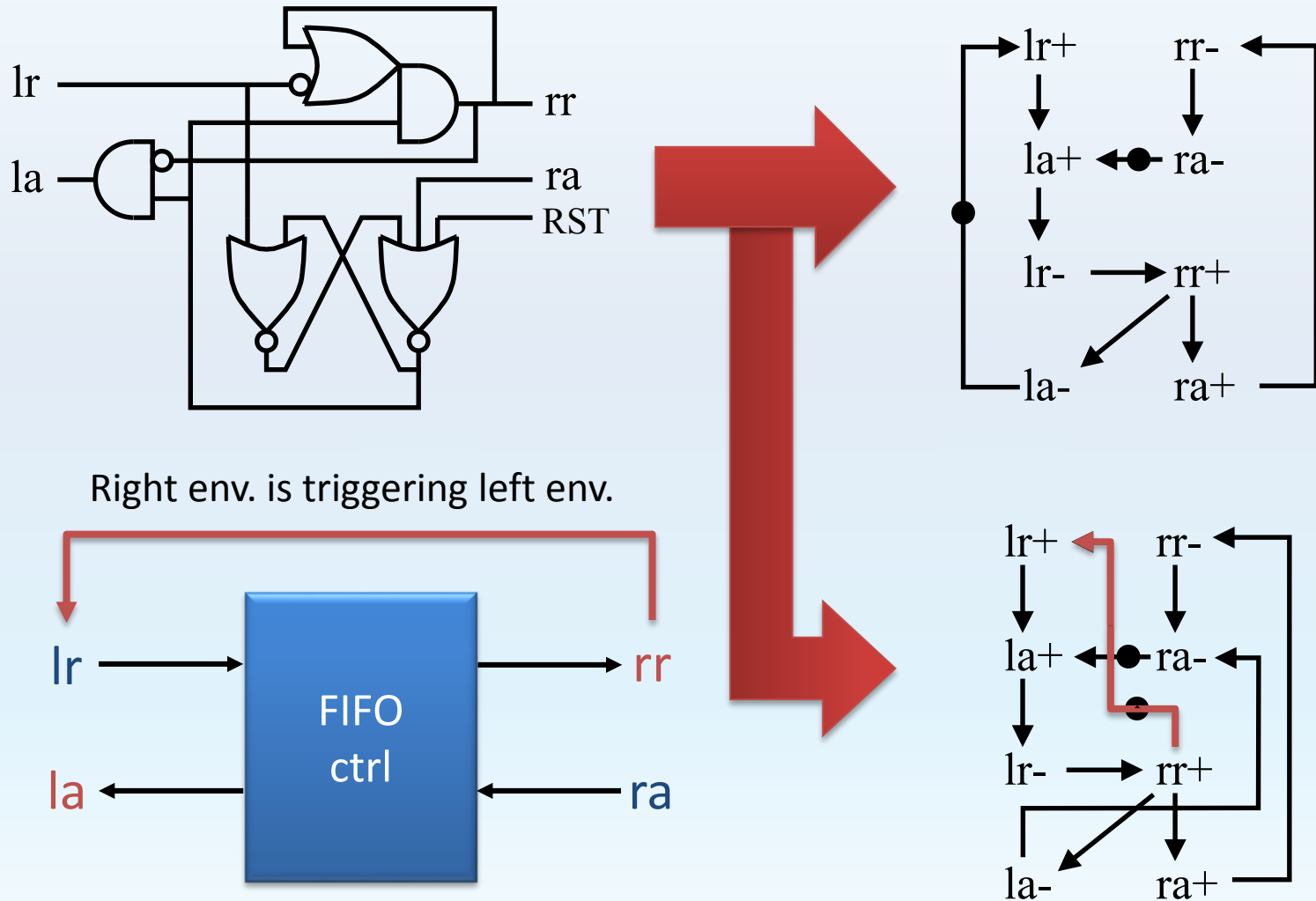
G. Birtwistle and K. Stevens
*Modelling mixed 4-phase pipelines:
structures and patterns,*
ASYNC 2014.

- 137 possible speed-independent specifications
- Synthesized with *Petrify*, then STGs mined assuming:
 - *Speed-independence*
 - *Delay insensitive interfacing*
 - *Multi-environment (L, R)*
- All mined STGs identical to original ones


Some 4-phase examples



Why multi-environment constraint?



Controllers with choice

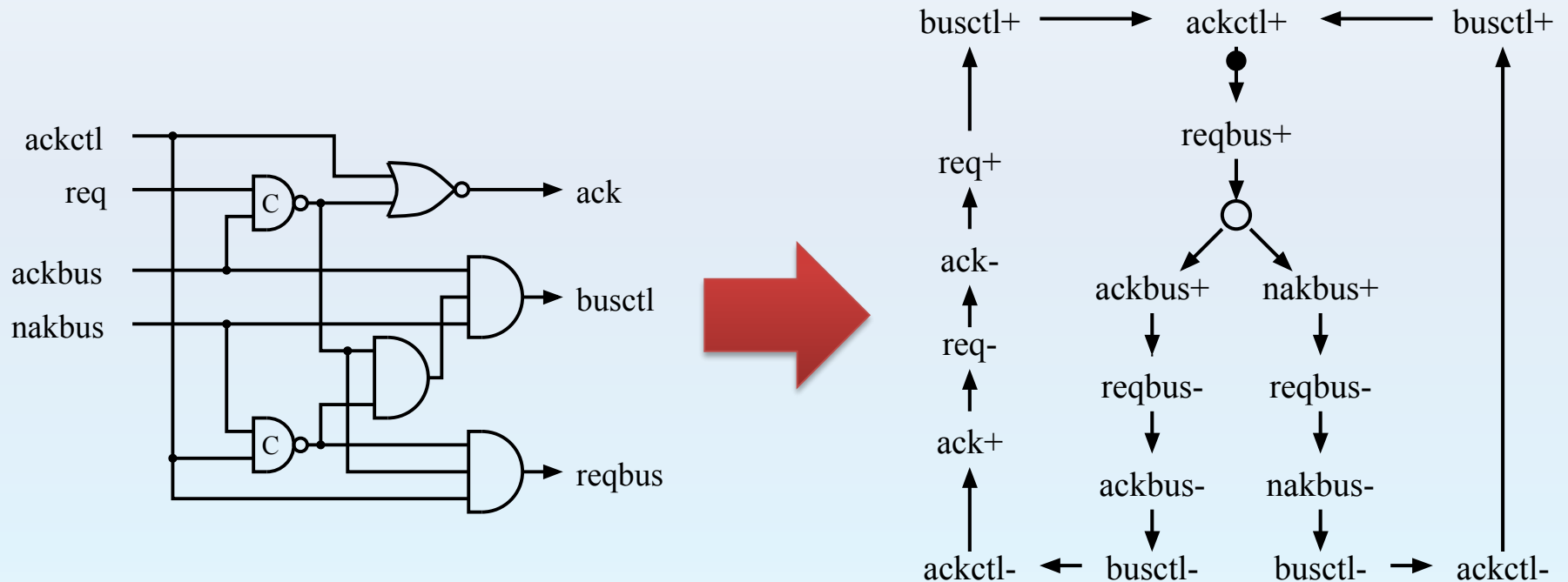


Benchmark	Num. of snippets	Gurobi runtime [seconds]	Num. of signals in orig. circuit
SM-latch	1	0.10	3
RLM	4	0.14	6
1-bit variable	1	0.31	6 i/o + 2 internal
alloc-outbound	3	0.73	7 i/o + 2 internal
vmebus	4	0.12	6 i/o + 1 internal
A/D converter ctrl.	1	0.43	7
tsend-csm	-	> 1 hour	9 i/o + 2 internal



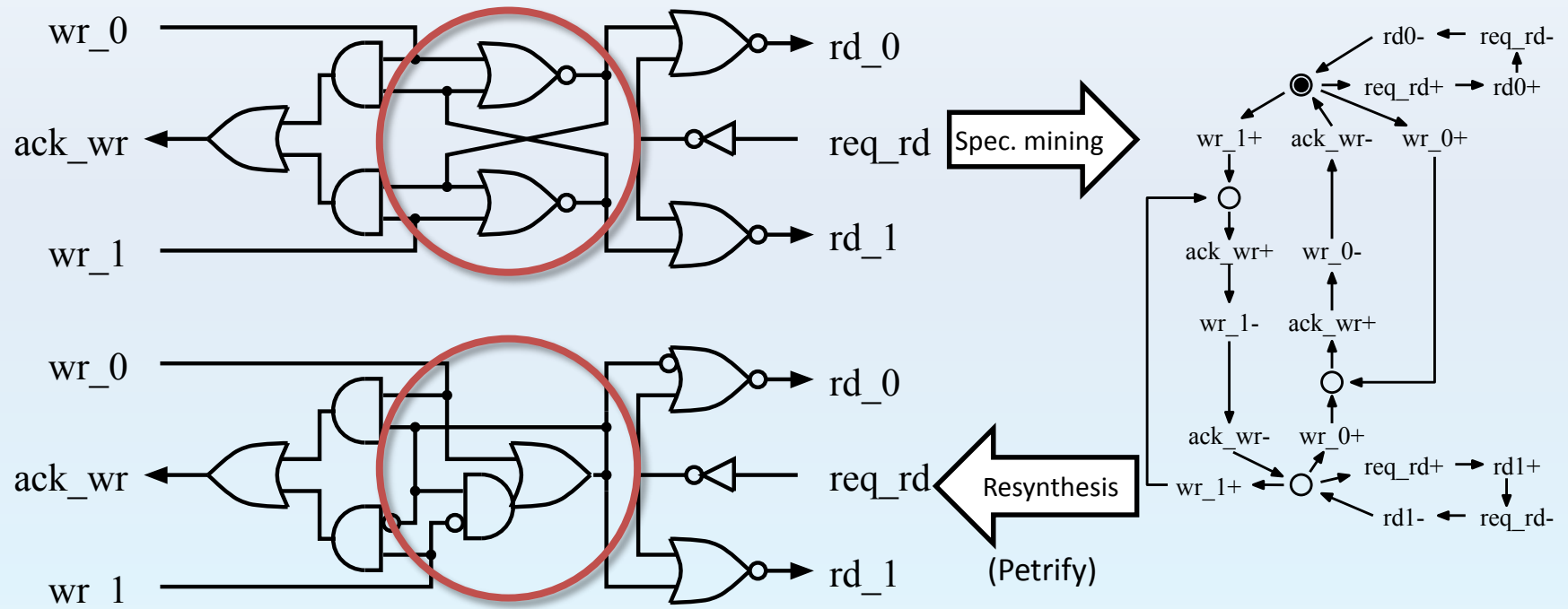
For each benchmark one snippet was identical to the original specification

Example: alloc-outbound



Resynthesis: 1-bit variable

K. v. Berkel. *Handshake Circuits: an Asynchronous Architecture for VLSI Programming*
Cambridge University Press, 1993.



Conclusions

- Specifications can be successfully recovered for several types of controllers
- Useful for reverse engineering, resynthesis, compositional verification, ...
- Future work:
 - Heuristics to handle large exploration spaces
 - Circuits with bounded delays