Time-Frame Folding: Back to the Sequentiality

ICCAD 05.11.2019

Po-Chun Chien[†], Jie-Hong Roland Jiang[†‡]

ALCom Lab

[‡]Department of Electrical Engineering,
[†]Graduate Institute of Electronics Engineering
National Taiwan University

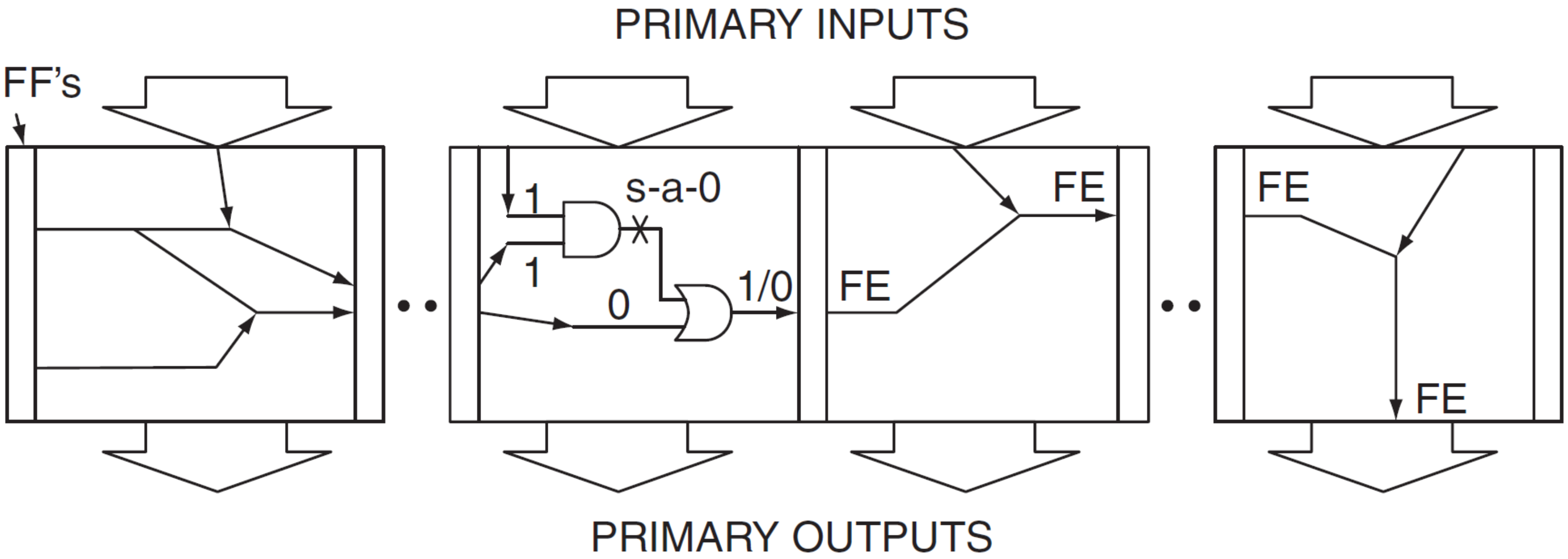
Outline

- Introduction
  - Time-Frame Unfolding vs. Folding
- Algorithm
  - State Identification
  - Transition Reconstruction
- Experiments
  - Setup
  - Results & Discussion
- Conclusions

INTRODUCTION

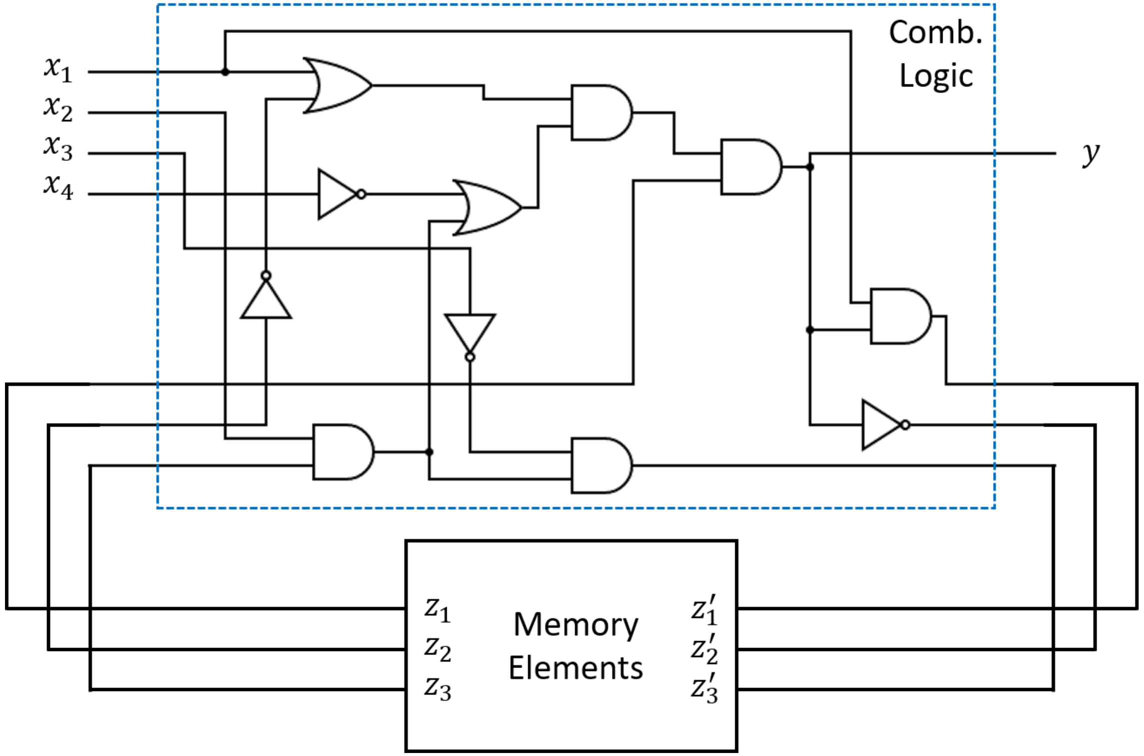
Time-Frame Unfolding

- TFU, or time-frame expansion
- A technique often used in ATPG, BMC



Time-Frame Unfolding

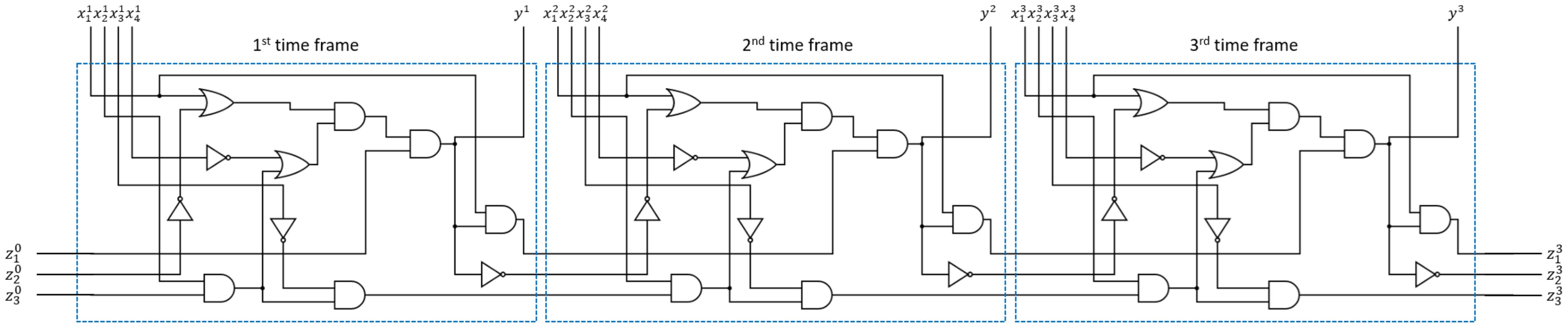
- An example sequential circuit



Sequential circuit $s27$

Time-Frame Unfolding

- Expand 3 time-frames

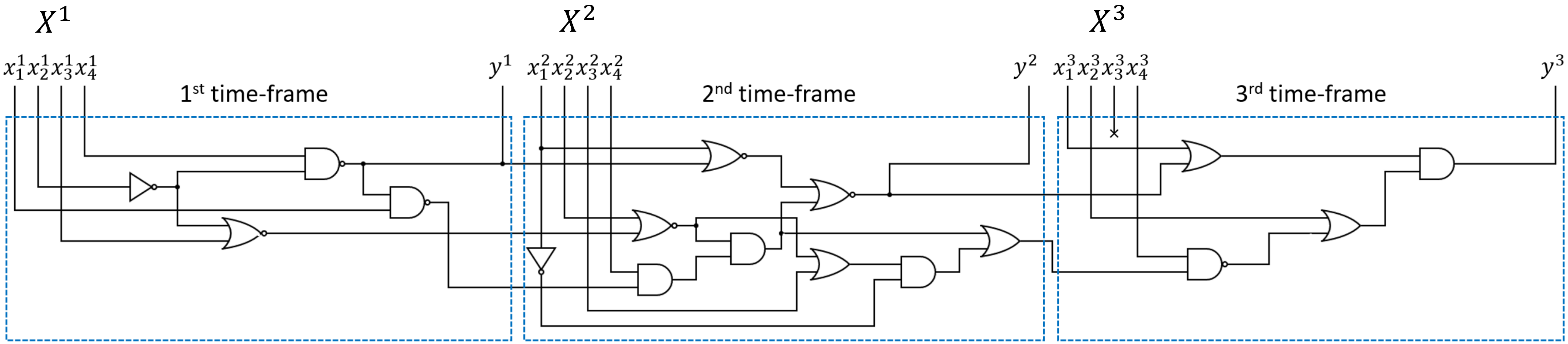


Regular duplication

with flip-flops from consecutive time-frames connected

Time-Frame Unfolding

- Expand 3 time-frames



with initial state propagation and simplification

$$y^1 = f(X^1) \qquad y^2 = g(X^1, X^2) \qquad y^3 = h(X^1, X^2, X^3)$$

Can we reverse it?

Iterative form

Motivation

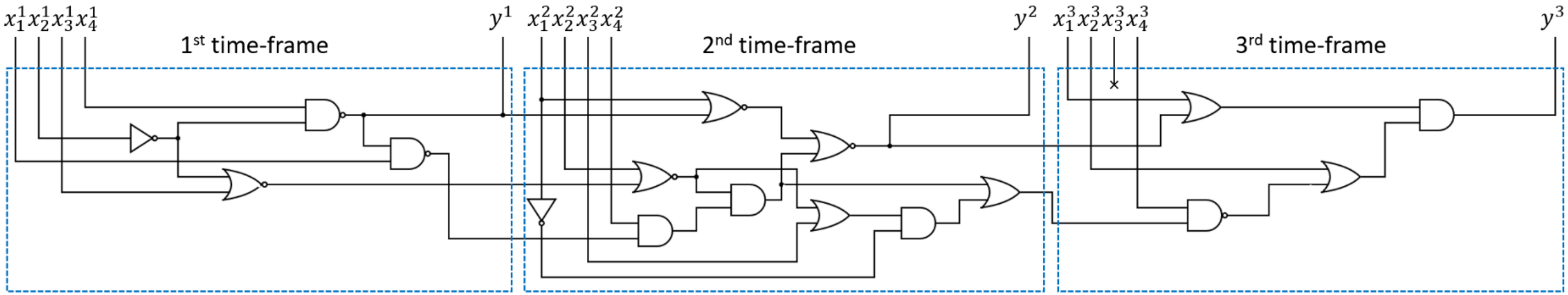
- In model-based testing of software systems [1, 2], one may be asked to compute synchronizing, distinguishing, or homing sequences.
- These problems can be formulated as quantified Boolean formula (QBF) [3, 4] solving of strategy derivation.
- The derived strategy corresponds to **a large (iterative) combinational circuit.** However, it can be alternatively represented **more compactly by a sequential circuit**.
- How can one reconstruct a sequential circuit from an iterative combinational circuit?

[1] Sandberg et al., 2005. [2] Kushiket al., 2016. [3] Bieree t al., 2009. [4] Wang et al., 2017.

Time-Frame Folding

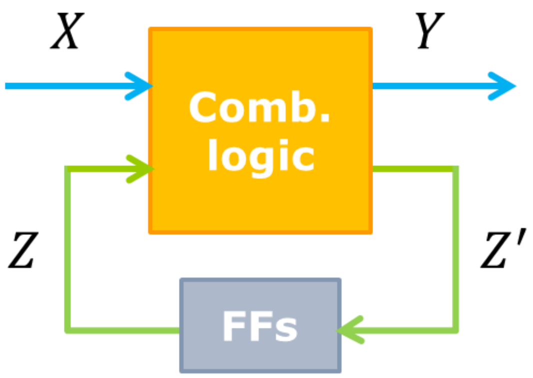
- TFF is a reverse operation of TFU

Given: a k-iterative combinational circuit



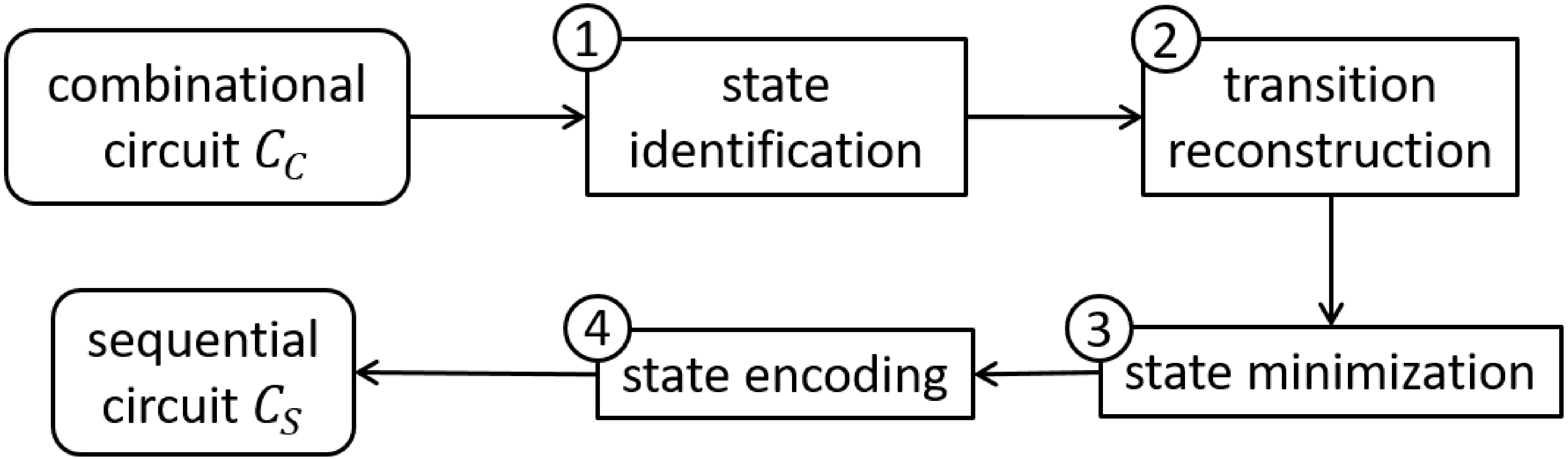
Goal: obtain a sequential circuit that
- is equivalent within bounded k time-frames
- has minimized state transition graph (STG)

(no assumption is made on the circuit structure except for the iterative form)



ALGORITHM

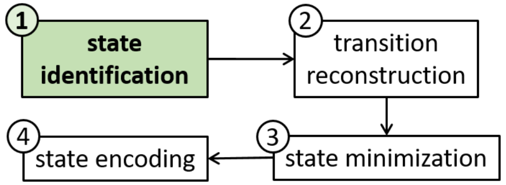
Computation Flow



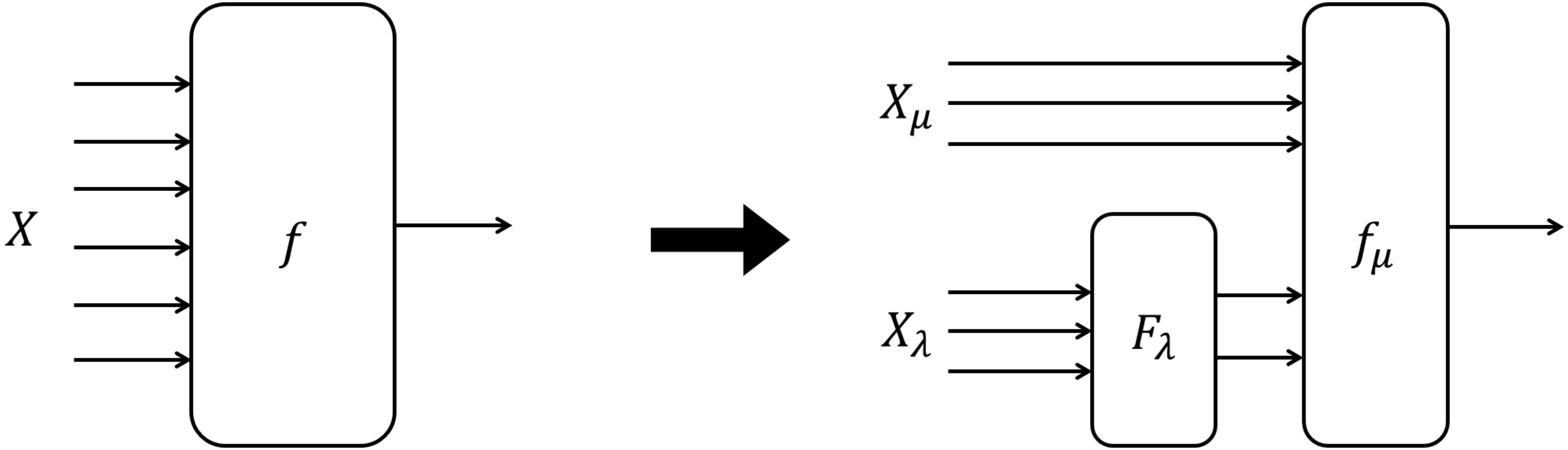
Notations

- $X^t = \{x_1^t, x_2^t, \dots, x_n^t\}$
  - X^t: the set of inputs at t^{th} time-frame
  - x_i^t: the i^{th} input at t^{th} time-frame
- $Y^t = \{y_1^t, y_2^t, \dots, y_m^t\}$
  - Y^t: the set of outputs at t^{th} time-frame
  - y_1^t: the i^{th} output at t^{th} time-frame
- $S^t = \left\{\left(q_1^t, \tau_{q_1^t}\right), \left(q_2^t, \tau_{q_2^t}\right), \dots, \left(q_k^t, \tau_{q_k^t}\right)\right\}$
  - S^t: the set of states at t^{th} time-frame
  - q_i^t: the symbol of the i^{th} state at t^{th} time-frame
  - $\tau_{q_i^t}$: transition condition of q_i^t

State Identification

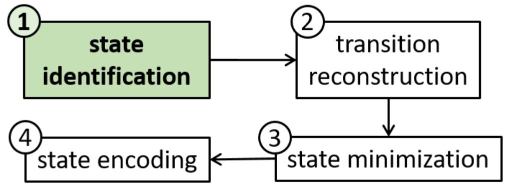


- Functional decomposition [5, 6]



X_λ: bound set, X_μ: free set

[5] Lai et al., 1993. [6] Chang et al., 1996.



State Identification

BDD-based decomposition

Decomposition chart

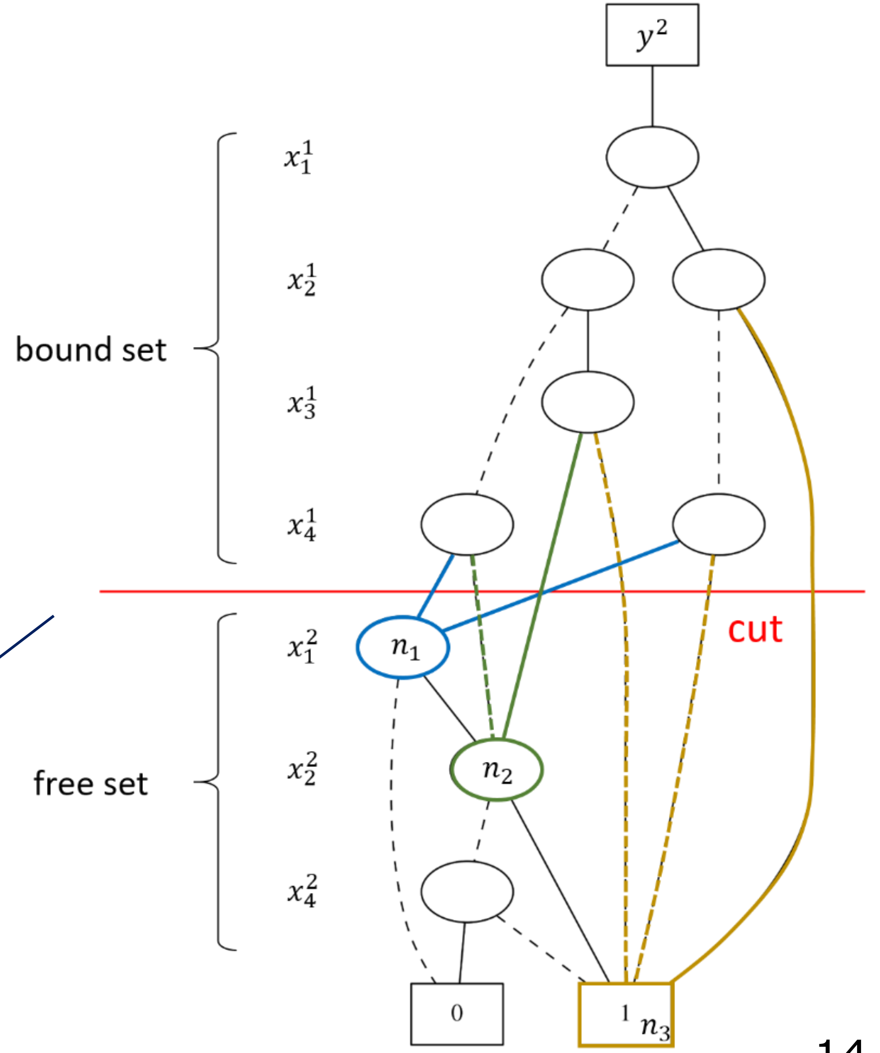
free $x_1^2x_2^2x_3^2x_4^2$ \ bound $x_1^1x_2^1x_3^1x_4^1$	-0-1	011-	00-0	11--	10-0	010-
0000	0	1	1	1	1	1
0001	0	0	0	1	1	1
0010	0	1	1	1	1	1
0011	0	0	0	1	1	1
⋮	⋮	⋮		⋮		

column patterns

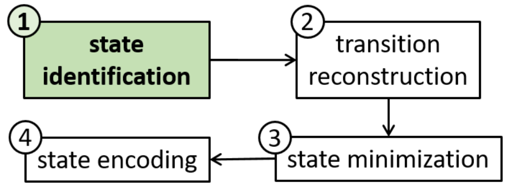
3 equivalence classes

$$\{\overline{x_2^1}x_4^1, \overline{x_1^1}(x_2^1x_3^1 + \overline{x_2^1}\,\overline{x_4^1}), x_1^1(\overline{x_2^1} + \overline{x_4^1}) + \overline{x_1^1}x_2^1\overline{x_3^1}\}$$

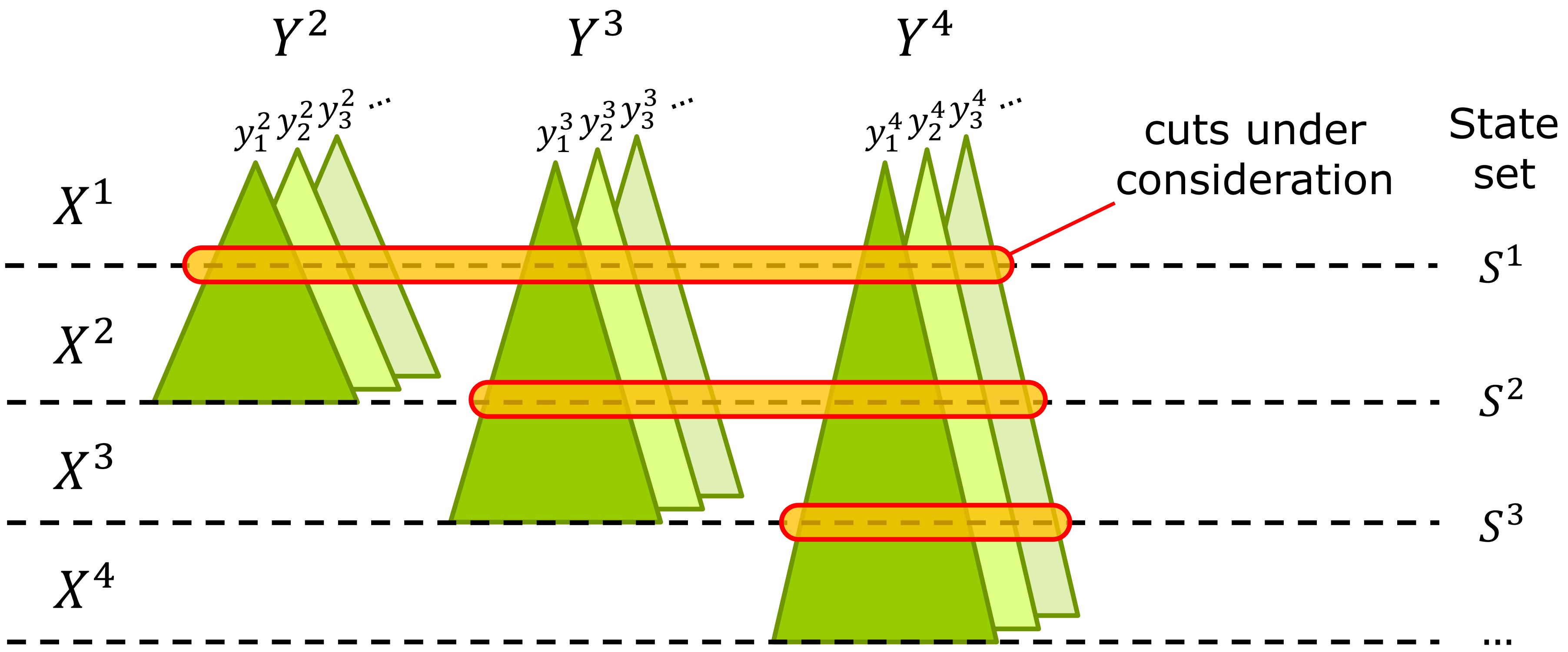
forming a partition on $\mathbb{B}^{|X^1|}$



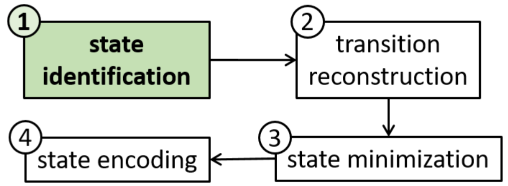
State Identification



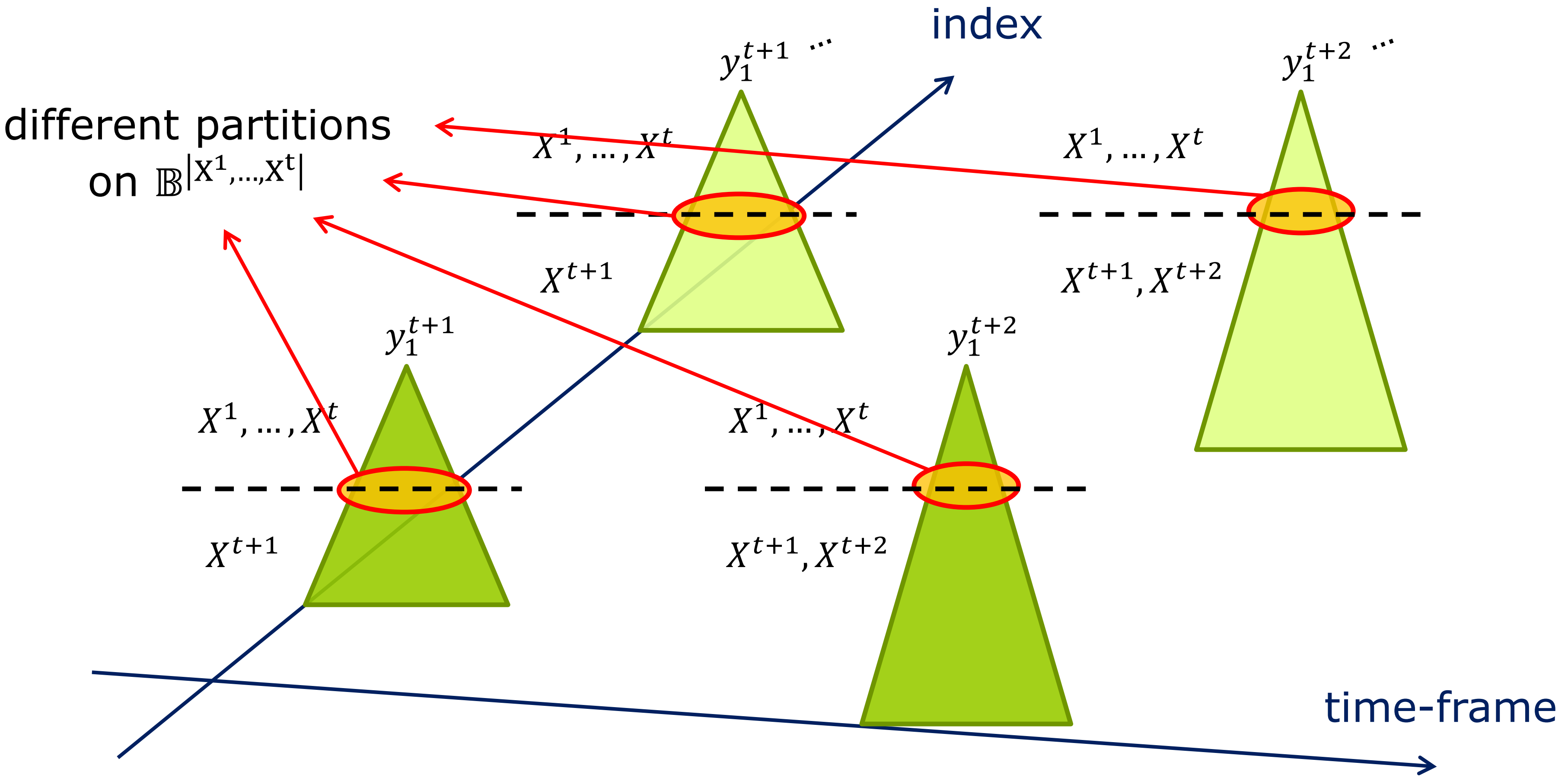
- State set S^t reached at t^{th} time-frame is determined by $Y^{t+1}, Y^{t+2}, \ldots, Y^T$



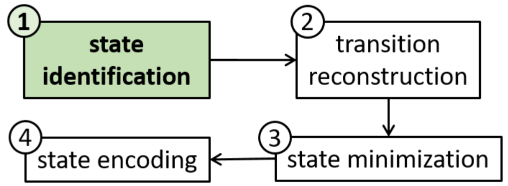
State Identification



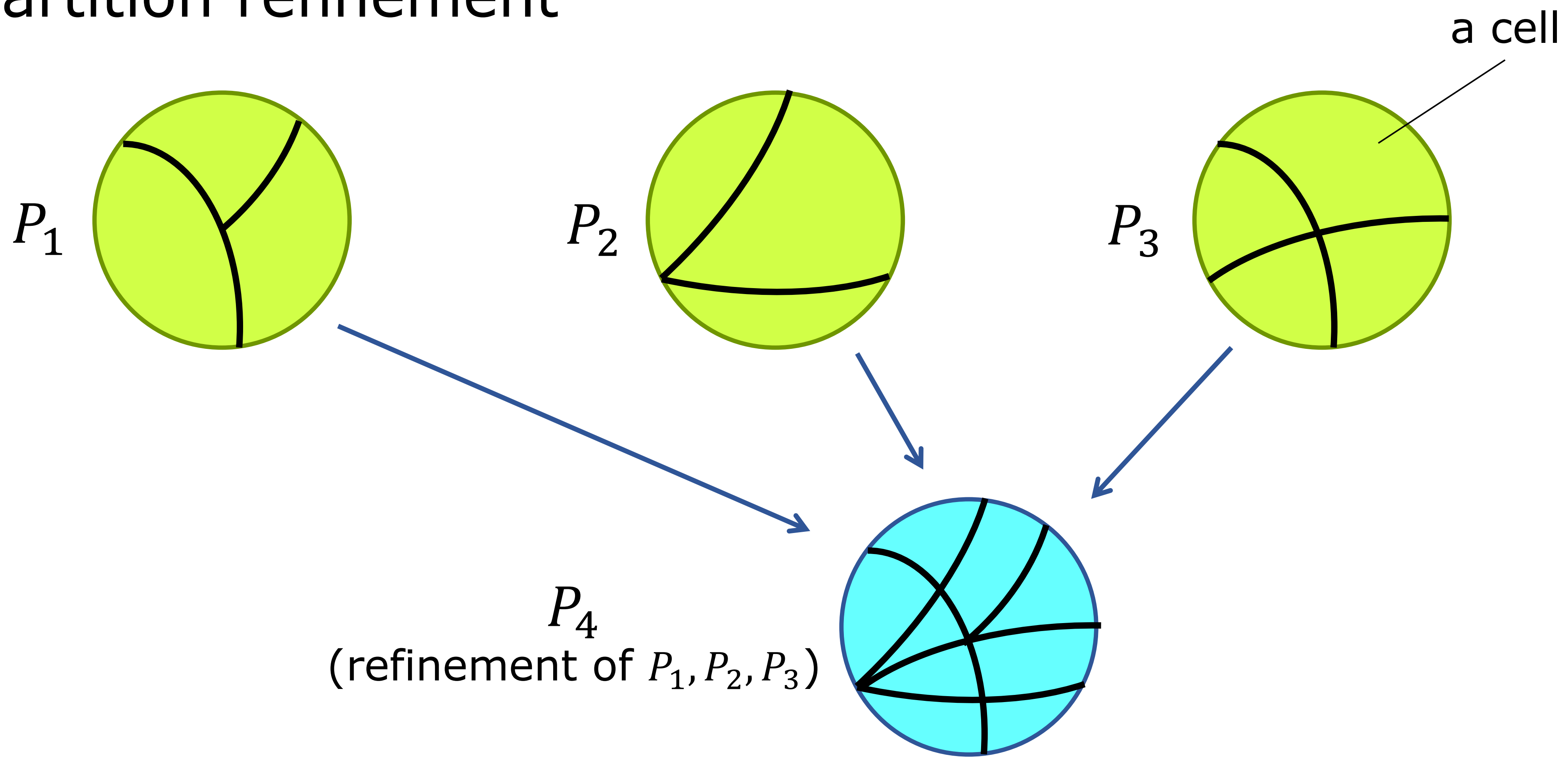
- s^t derivation



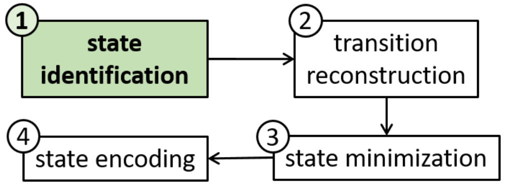
State Identification



- Partition refinement



State Identification



- Hyper-function encoding [7]:
  E.g. for a multi-output function

$$F(X) = \{f_1(X), f_2(X), f_3(X), f_4(X)\}$$

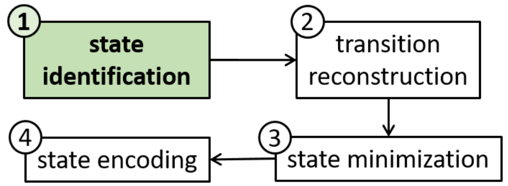
introduce $A = \{\alpha_1, \alpha_2\}$ to encode F into

$$h(X, A) = \overline{\alpha_1}\,\overline{\alpha_2} f_1 + \overline{\alpha_1}\alpha_2 f_2 + \alpha_1\overline{\alpha_2} f_3 + \alpha_1\overline{\alpha_2} f_4$$

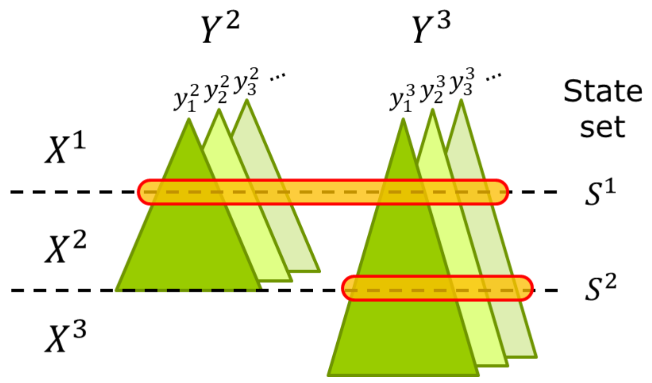
single-output functional decomposition algorithm can be applied

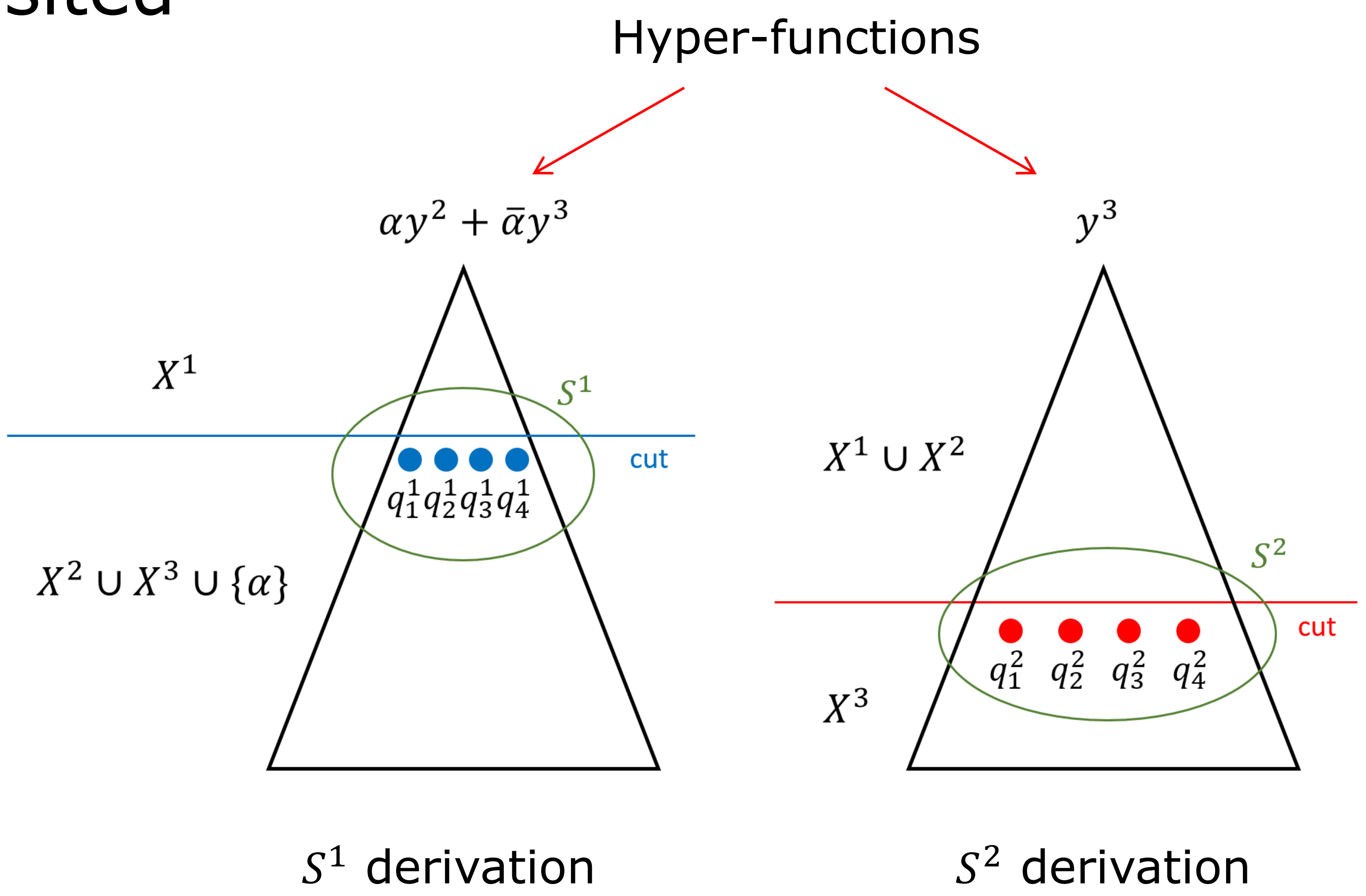
[7] Jiang et al., 1998.

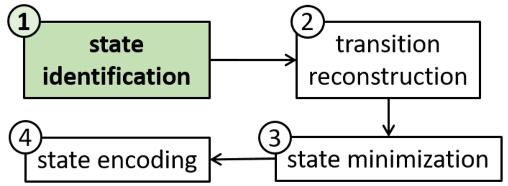
State Identification



- $s27$ example revisited





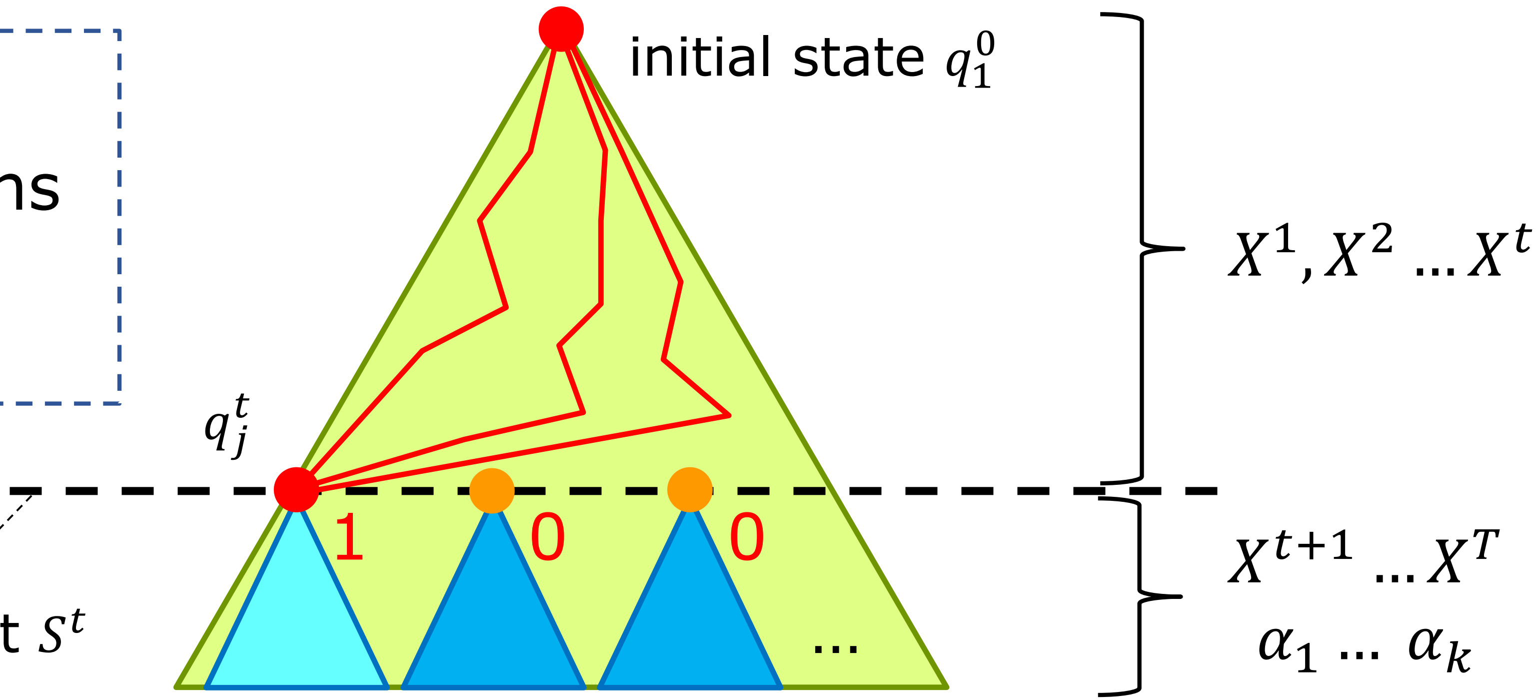


State Identification

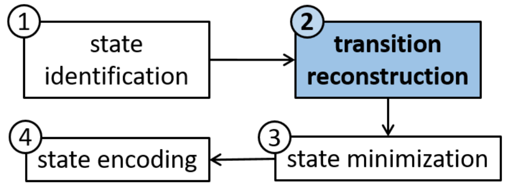
- Transition condition $\tau_{q_j^t}$ of state q_j^t

$hyper\text{-}function\ of\ \{Y^{t+1}, \dots, Y^T\}$

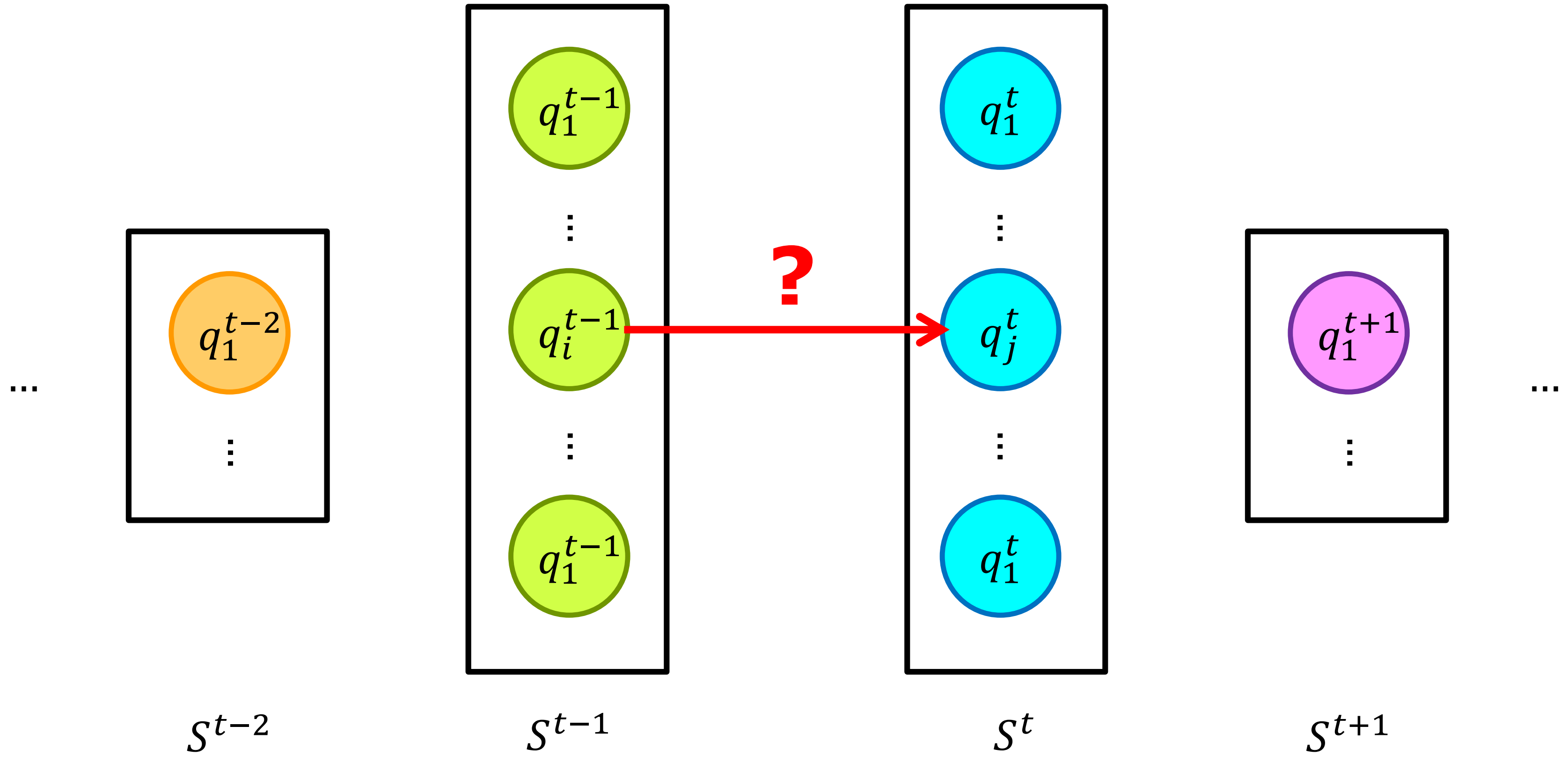
$\tau_{q_j^t}(X^1, X^2 \dots X^t)$:
collection of paths
starting from q_1^0
leading to q_j^t



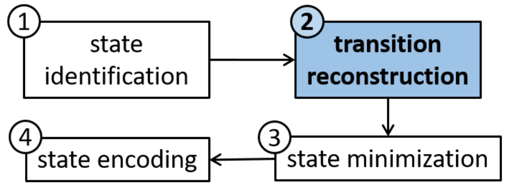
Transition Reconstruction



- Find the transition between state pairs



Transition Reconstruction



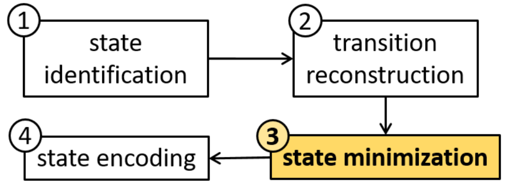
- For each pair of state (q_i^{t-1}, q_j^t) in adjacent 2 time-frames:
  - Input transition condition:

$$\varphi_{i,j}^t = \underbrace{\exists X^1, \ldots, X^{t-1}}_{\text{global} \rightarrow \text{local info.}} . \underbrace{\tau_{q_i^{t-1}} \wedge \tau_{q_j^t}}_{\text{paths to } q_j^t \text{ through } q_i^{t-1}}$$

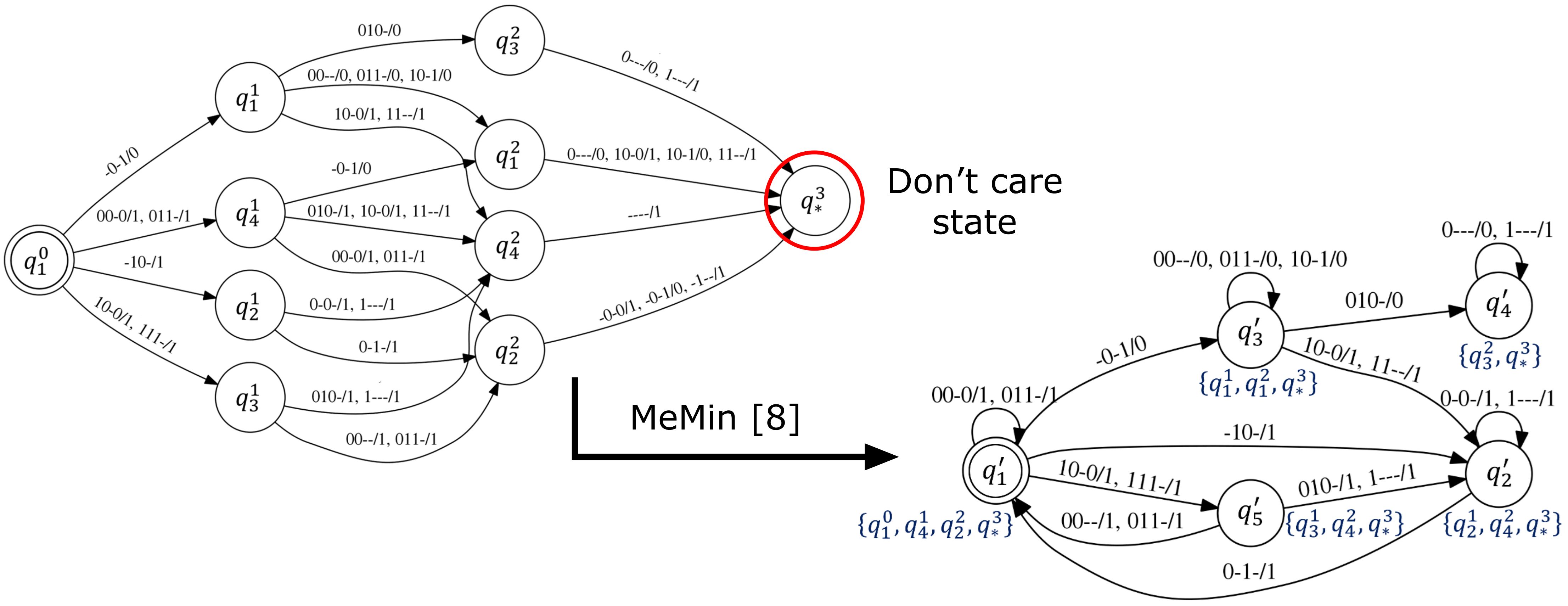
  - Output transition response

$$\psi_{i,k}^t = \exists X^1, \ldots, X^{t-1} . \tau_{q_i^t} \wedge y_k^t$$

State Minimization

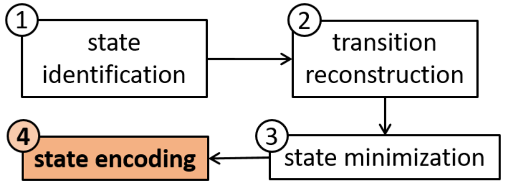


- $s27$ example revisited



[8] Abel et al., 2015.

State Encoding



- Encode each state in the state set Q with actual bits, 2 schemes are applied:
  - Natural Encoding with $\lceil \log(|Q|) \rceil$ bits
  - One-hot encoding with $|Q|$ bits, each of which represents a state in Q.

EXPERIMENTS

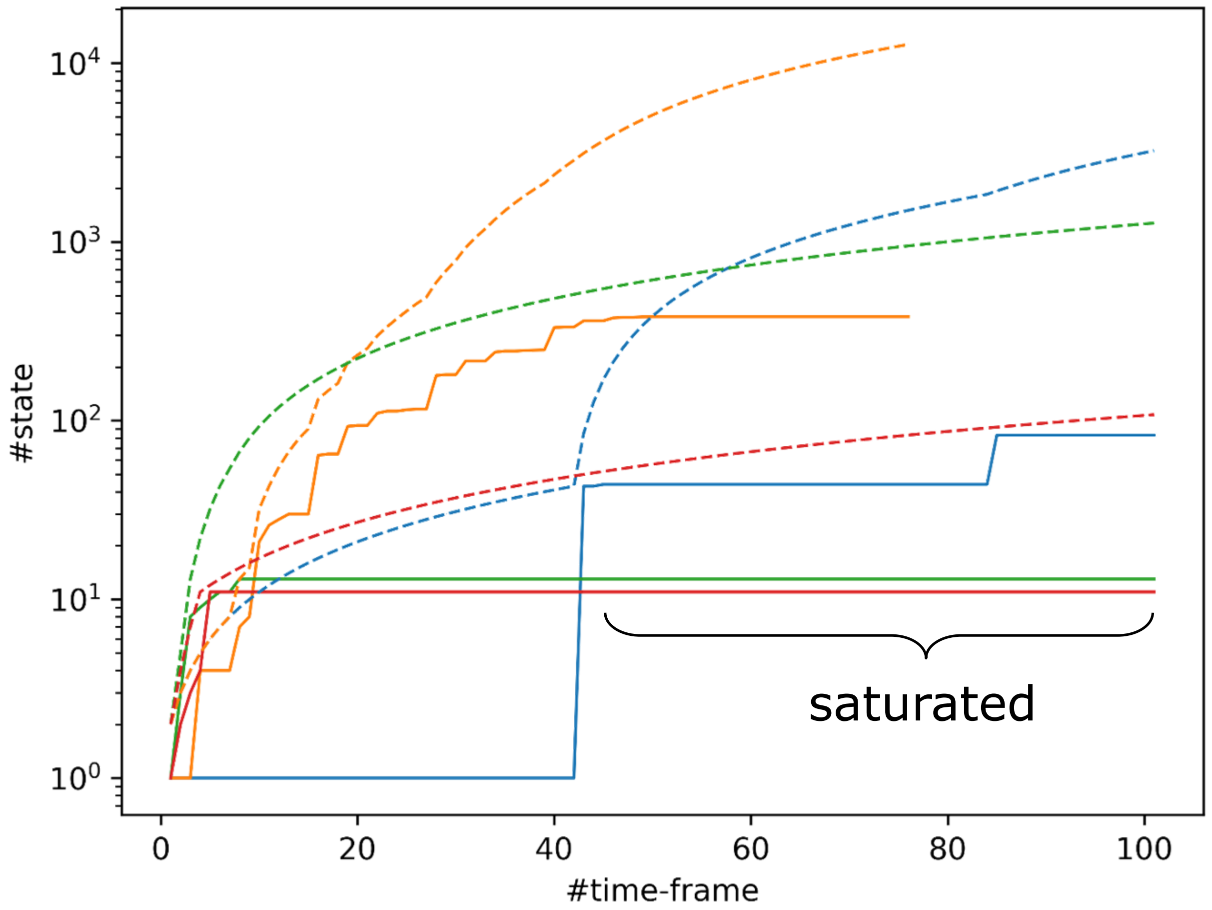
Setup

- Implemented in C++ within ABC [9] and used CUDD [10] as the underlying BDD package.
- Environment: Intel(R) Xeon(R) CPU E5-2620 v4 of 2.10 GHz and 126 GB RAM
- Benchmark circuits
  - Unfolded ISCAS/ITC circuits
  - QBF solving of homing sequence [4]
- 300s timeout limit

[4] Wang et al., 2017. [9] Brayton et al., 2010. [10] Somenzi et al., 2005.

Results

- Number of states



b07
b18
s386
s15850

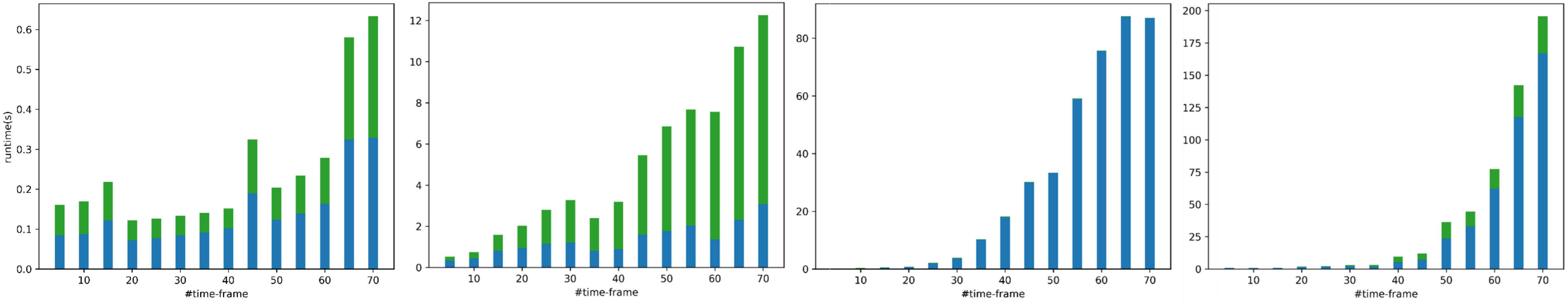
------- : #state **before** minimization
—— : #state **after** minimization

#state vs. #time-frame.

Results

- Total runtime

b07 s15850 s386 b18



runtime vs. #time-frame.

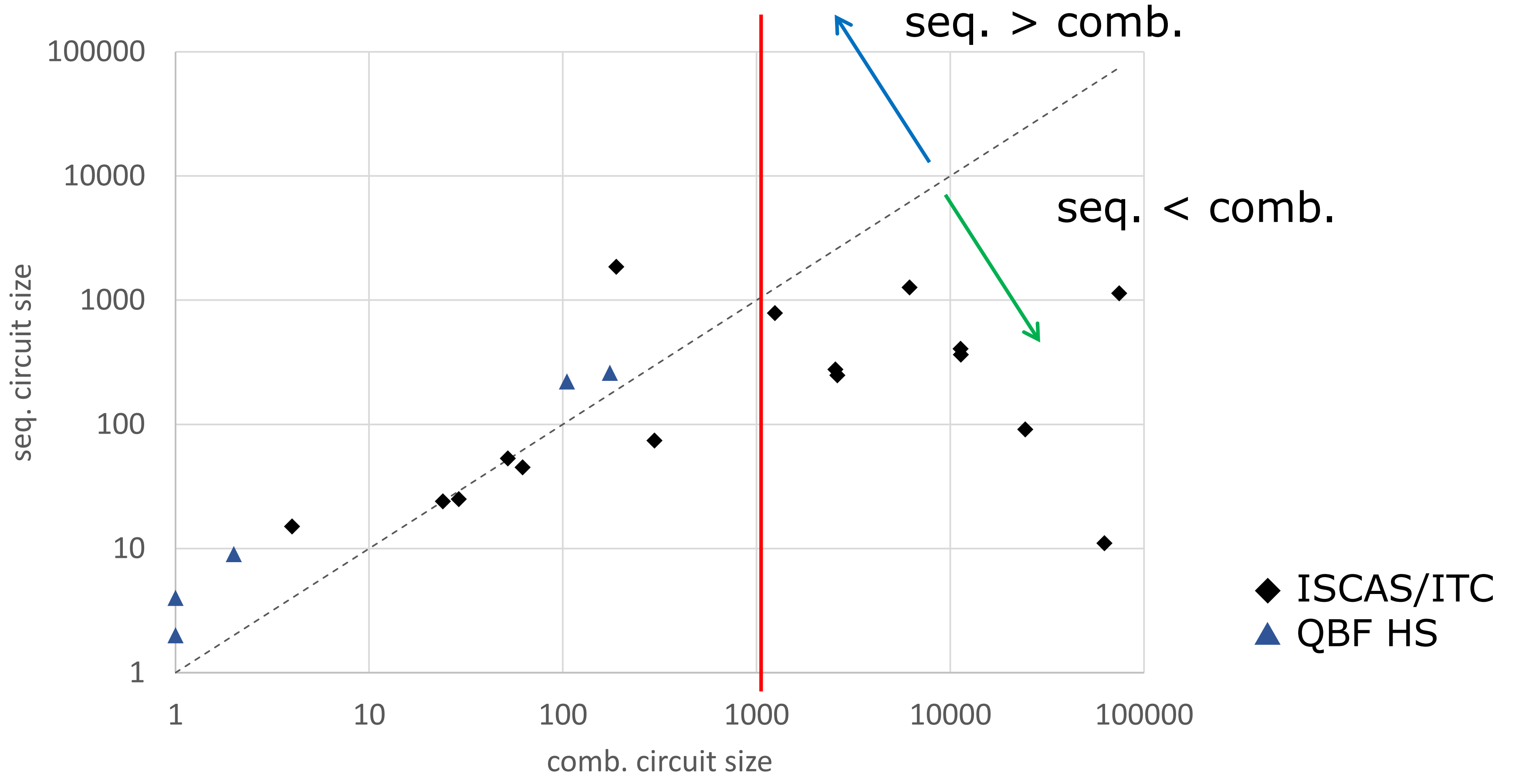
Time-Fold
MeMin

Results

circuit	#time-frame		expanded circuit	natural encoding		one-hot encoding	
	state saturate	fixed point	#gate	#FF	#gate	#FF	#gate
b01	9	9	52	5	109	18	53
b02	6	10	4	3	16	8	15
b03	14	14	189	10	8947	631	1848
b05	69	133	62635	7	52	69	11
b06	6	7	62	4	82	13	45
b07	85	85	24438	7	91	83	94
b08	55	55	6173	10	3395	798	1265
b18	50	50	74461	9	2516	382	1134
s27	3	5	29	3	25	5	42
s298	20	23	1243	8	1489	135	785
s386	8	9	297	4	124	13	74
s820	12	13	2558	5	276	24	8639
s832	12	13	2612	5	248	24	10075
s1488	23	23	11298	6	578	48	406
s1494	23	23	11367	6	526	48	364
s15850	5	5	24	4	28	11	24

Results on folding with "fixed points" reached.

Results



Circuit size comparison.

Conclusions

- We have formulated the time-frame folding problem, and provided a computational solution based on functional decomposition.
- Our method guarantees the folded sequential circuit is state minimized.
- Experimental results demonstrated the benefit of our method in circuit compaction from an iterative combinational circuit to its sequential counterpart.
- Our method can be useful in testbench generation, sequential synthesis of bounded strategies, and other applications.

THE END