Efficient State Estimation of Discrete-Timed Automata

International Conference on Formal Engineering Methods

December 4, 2024

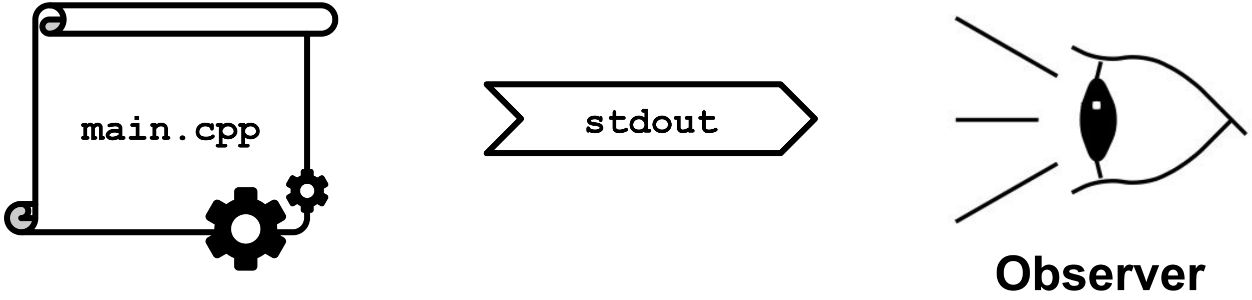
Julian Klein, Paul Kogel, Sabine Glesner

SOFTWARE AND EMBEDDED SYSTEMS ENGINEERING

berlin

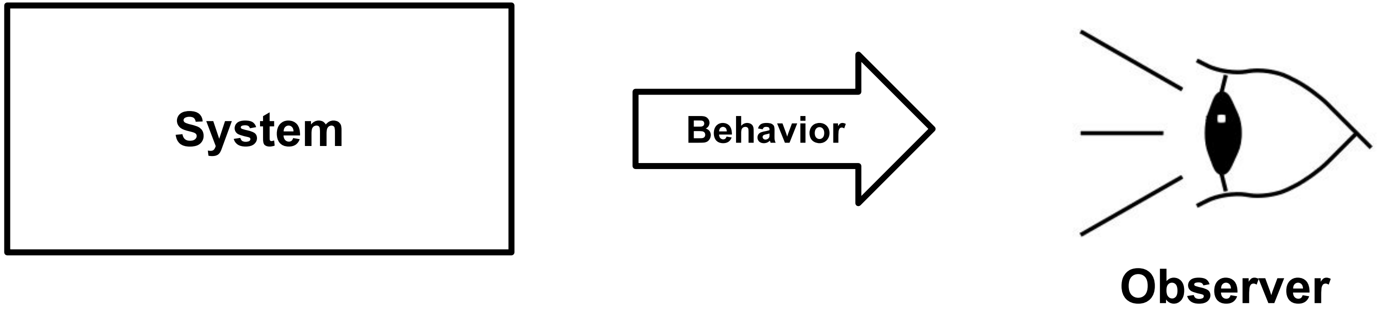
Introduction: State Estimation

- **State estimation**: which state could currently be active?



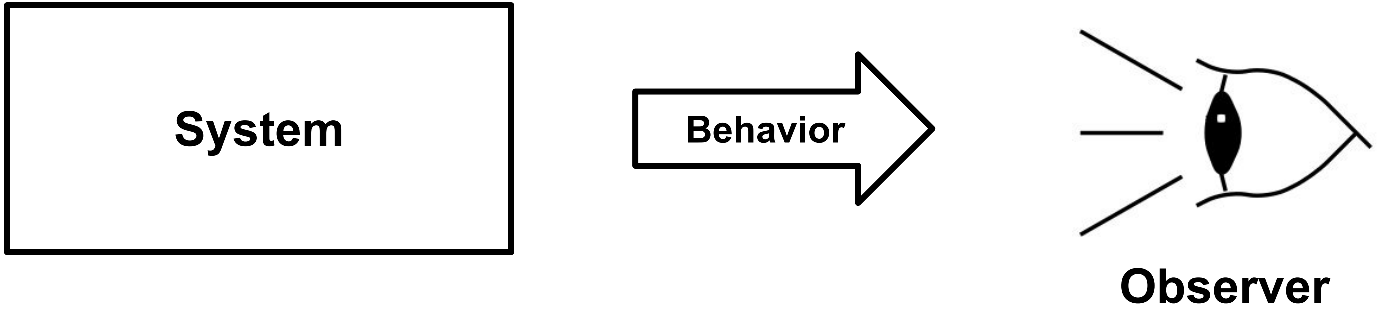
Introduction: State Estimation

- **State estimation**: which state could currently be active?



Introduction: State Estimation

- **State estimation**: which state could currently be active?



- **Applications** in safety, security, fault diagnosis, …
- **Timed models** required to model real-world systems

➔ State estimation for discrete-**timed automata** (TA)

Literature

Continuous time model: state estimation is undecidable (**Baier et al. 2009**)

Discrete time model: existing approaches consider

- Weighted automata (**Lai et al. 2020**, **Li et al. 2021**)
- Automata over monoids (**Zhang 2022**)
- Max-plus automata (**Lai et al. 2019**)
- Tick automata (**Klein et al. 2024**)

Finite automata (FA)

Literature

Continuous time model: state estimation is undecidable (**Baier et al. 2009**)
Discrete time model: existing approaches consider

- Weighted automata (**Lai et al. 2020**, **Li et al. 2021**)
- Automata over monoids (**Zhang 2022**)
- Max-plus automata (**Lai et al. 2019**)
- Tick automata (**Klein et al. 2024**)

Finite automata (FA)

Problem

- Discrete states always considered **individually**
- State estimation generally **scales exponential** with the number of states

➔ **Limited scalability** for realistic systems

Problem Statement

Goal:

Efficient state estimation method for (discrete) timed automata

Key idea:

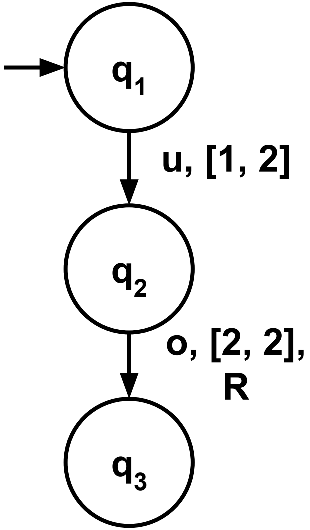
- Group states in equivalence classes (when possible)

➔ Evaluate only one representative state for each class

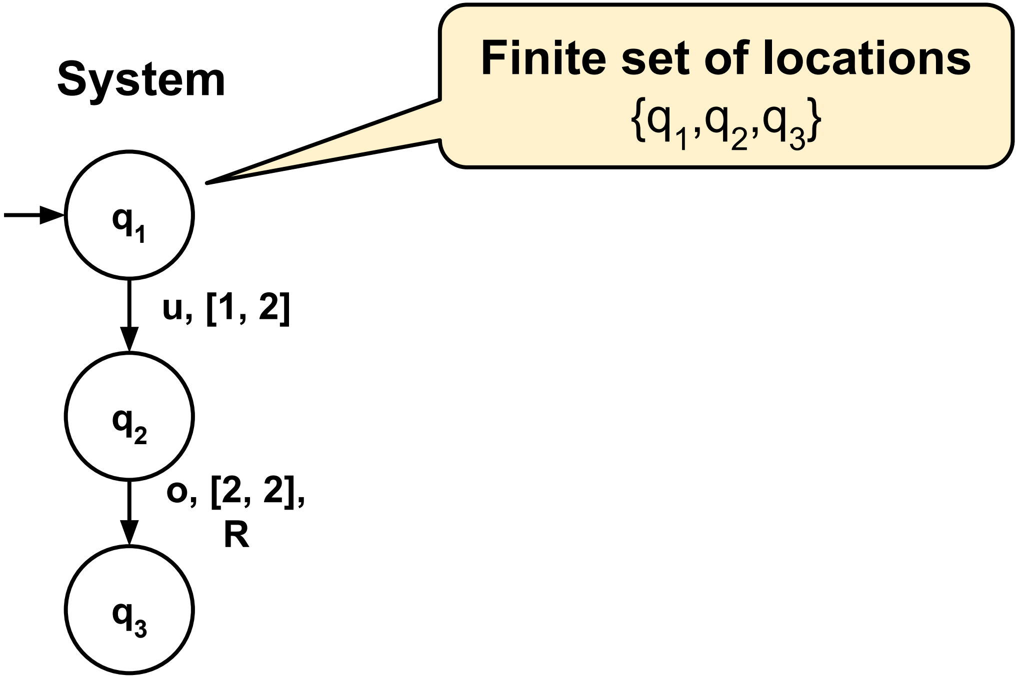
Background

Timed Automata

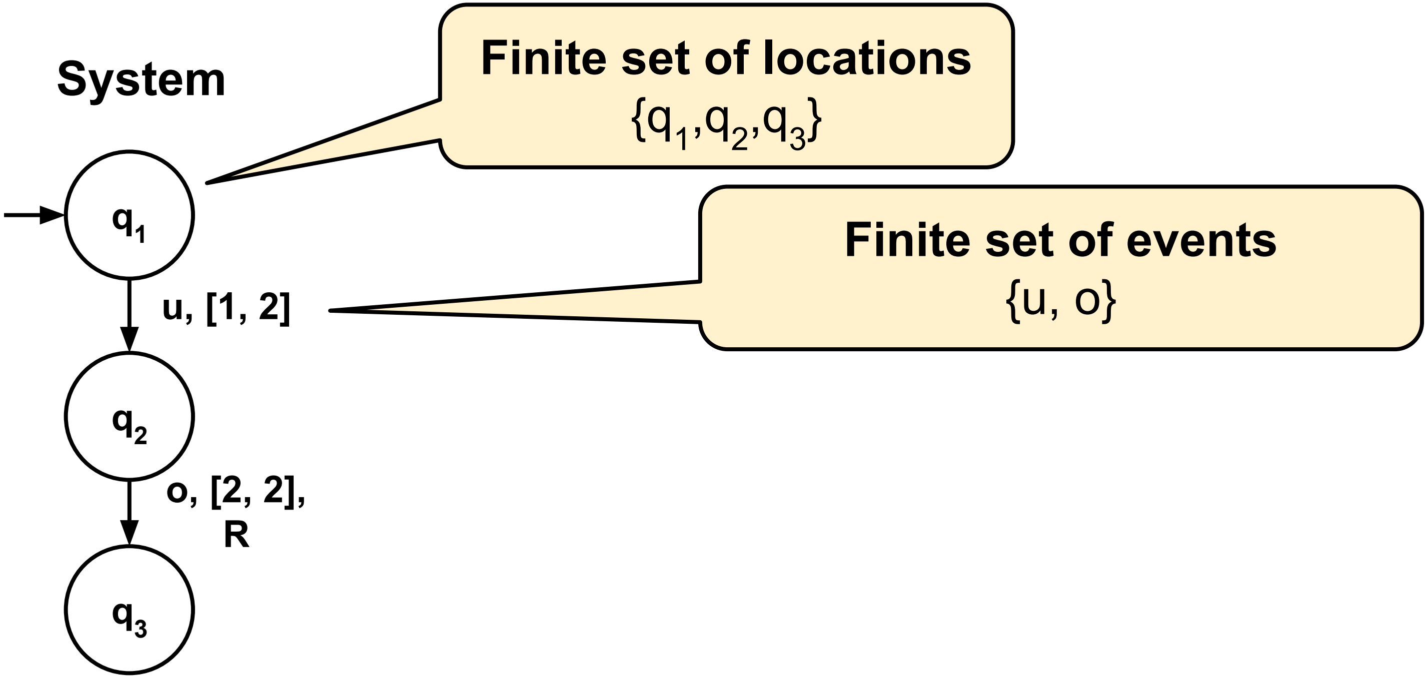
System



Timed Automata

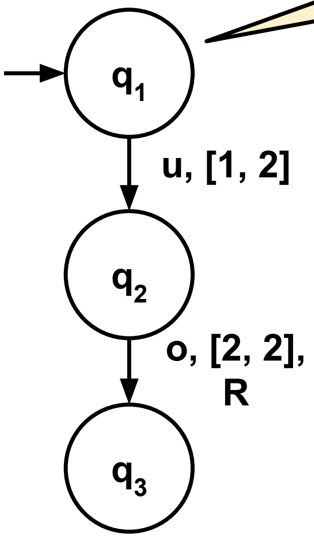


Timed Automata



Timed Automata

System



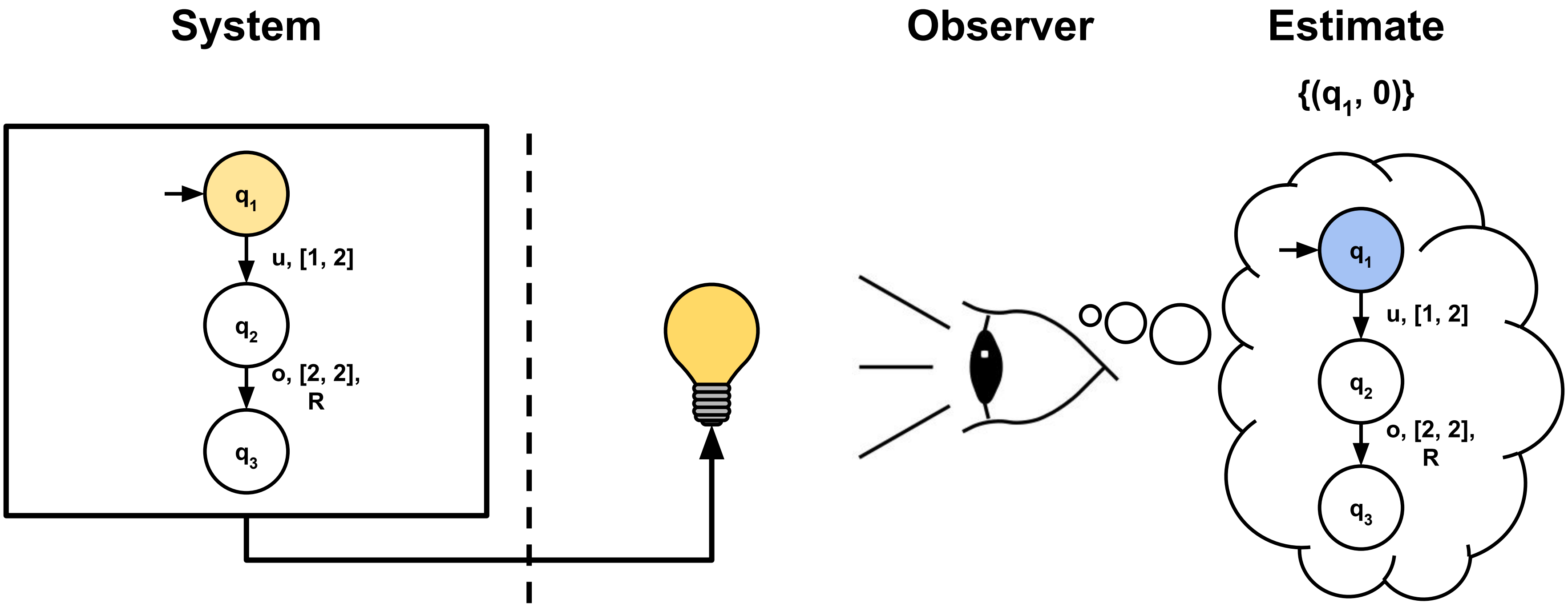
Finite set of locations
$\{q_1, q_2, q_3\}$

Finite set of events
$\{u, o\}$

A single clock
- can be reset on transitions
- can enable/disable transitions

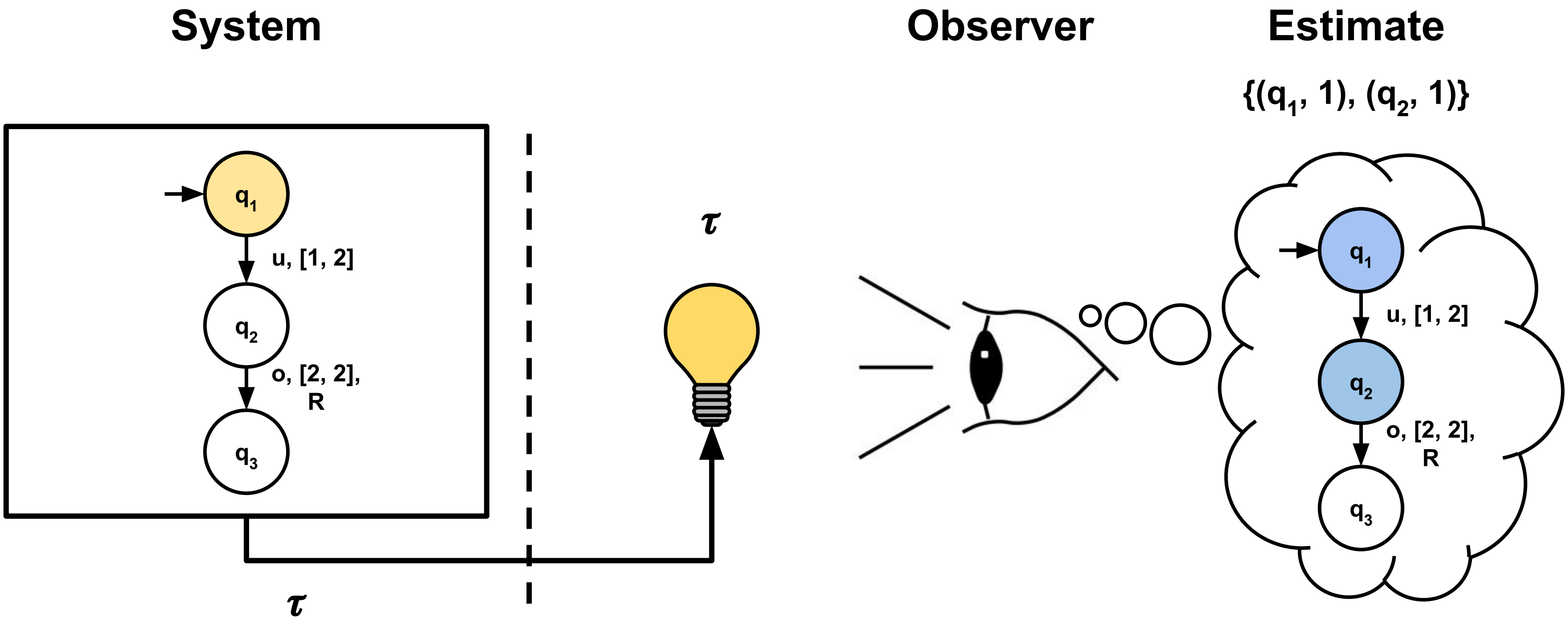
(Online) State Estimation

- Assume **u** is **u**nobservable and **o** is **o**bservable



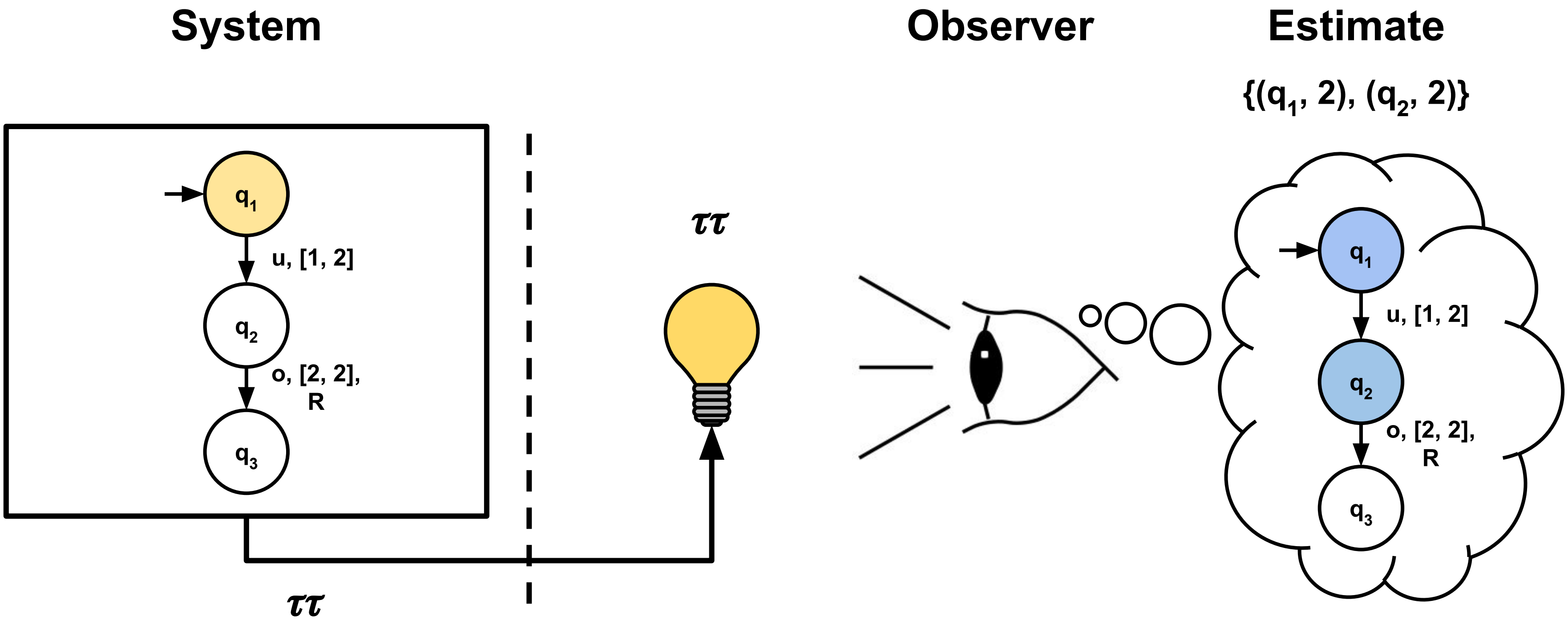
(Online) State Estimation

- Assume **u** is **u**nobservable and **o** is **o**bservable



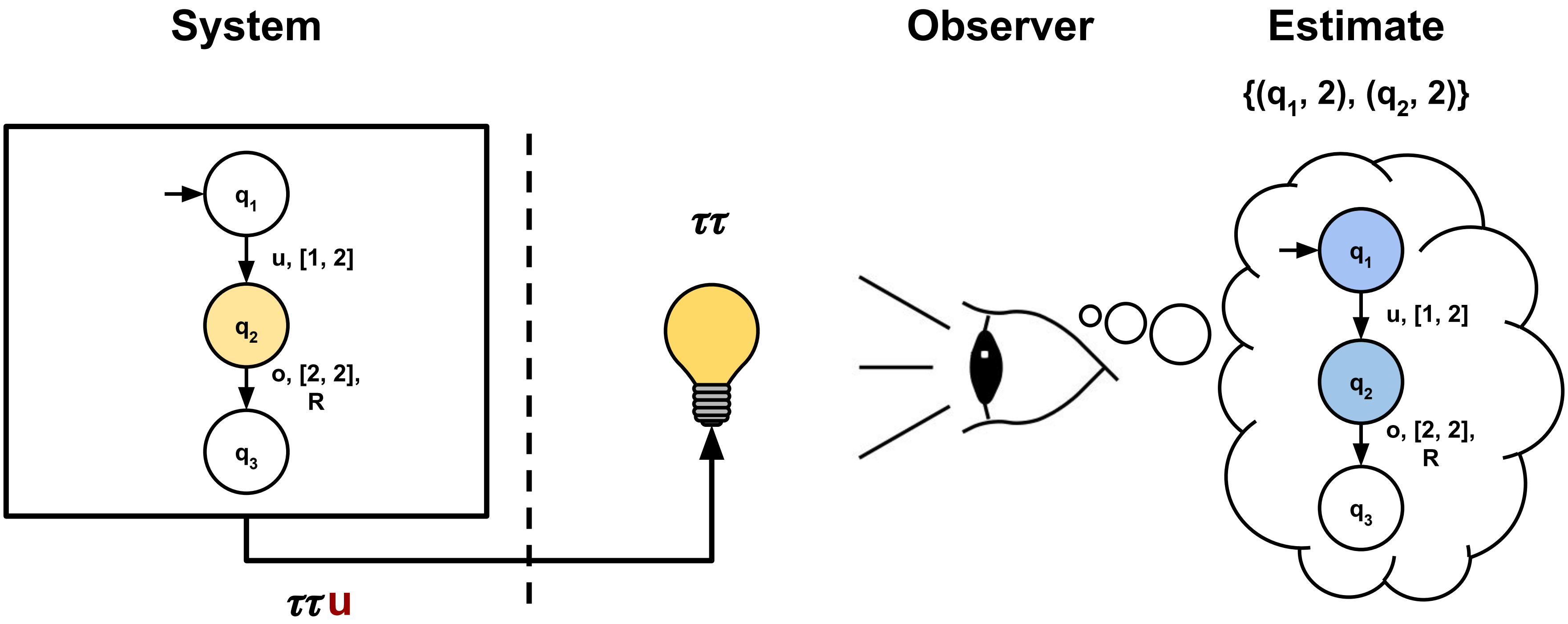
(Online) State Estimation

- Assume **u** is **u**nobservable and **o** is **o**bservable



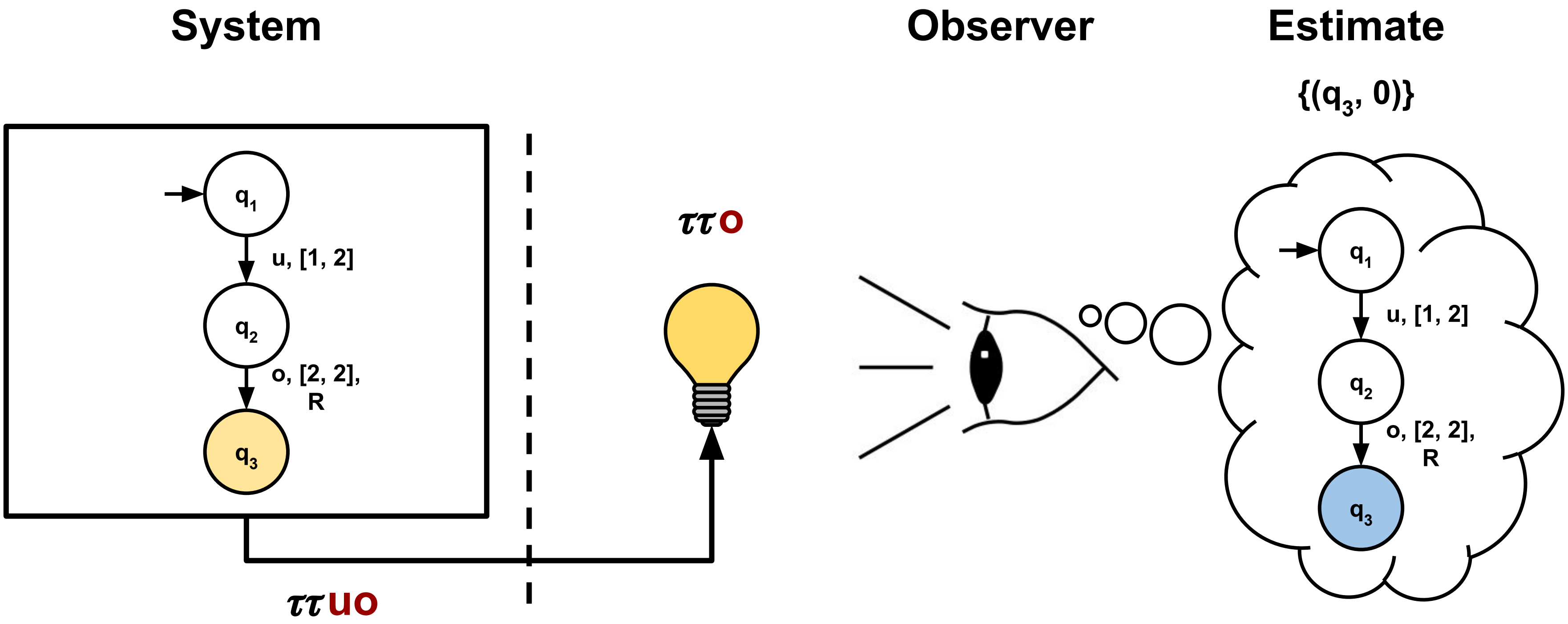
(Online) State Estimation

- Assume **u** is **u**nobservable and **o** is **o**bservable



(Online) State Estimation

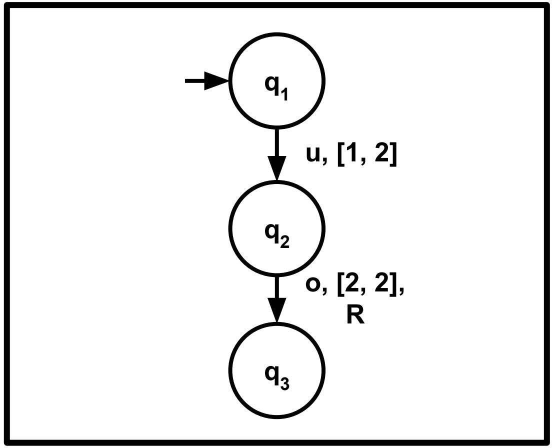
- Assume **u** is **u**nobservable and **o** is **o**bservable



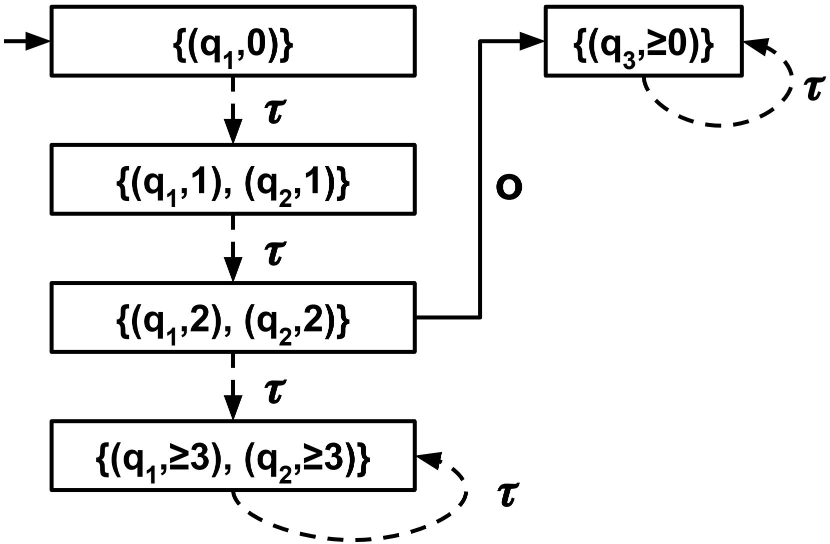
(Offline) State Estimation

- Assume **u** is **u**nobservable and **o** is **o**bservable

System



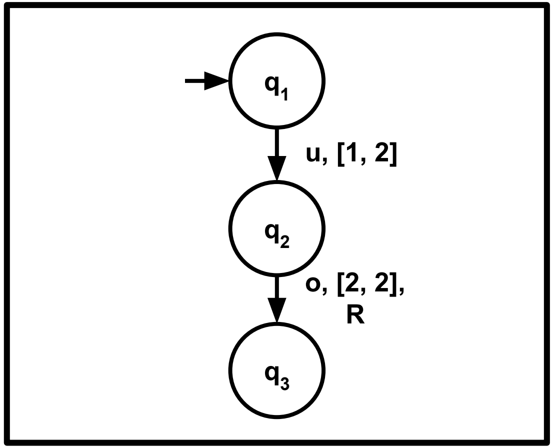
Estimator



(Offline) State Estimation

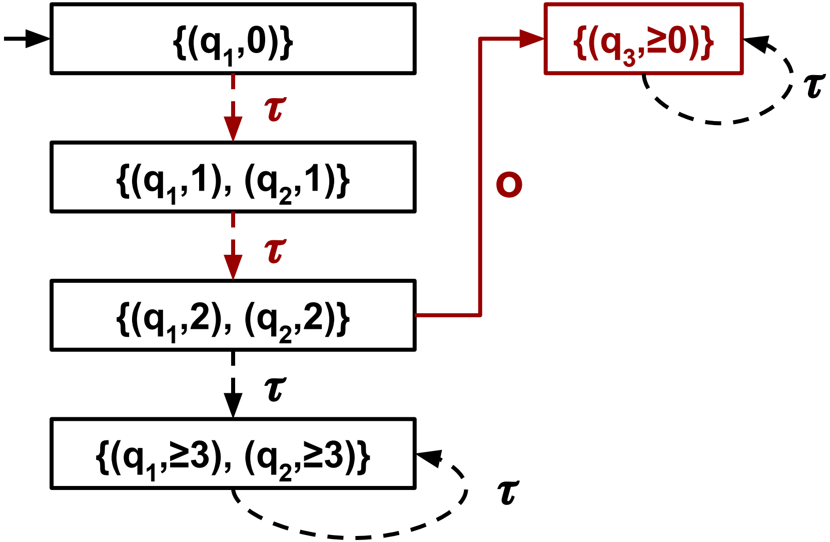
- Assume **u** is **u**nobservable and **o** is **o**bservable

System



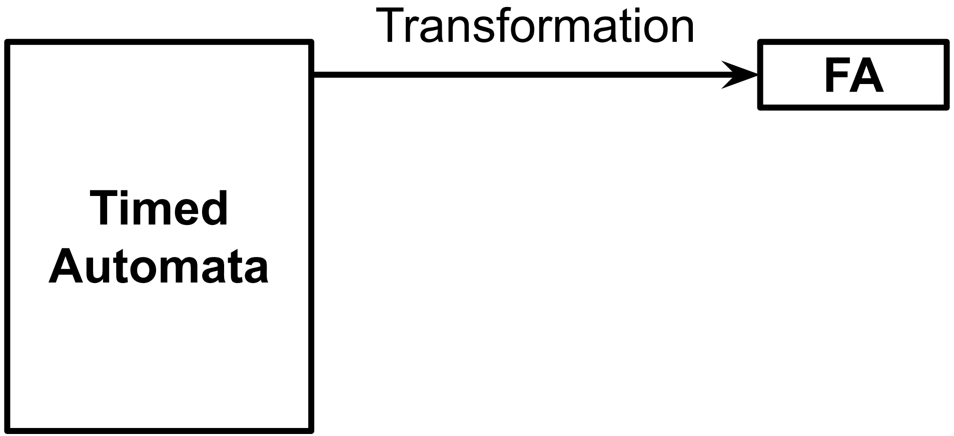
$\tau\tau$o

Estimator



FA-Based State Estimation

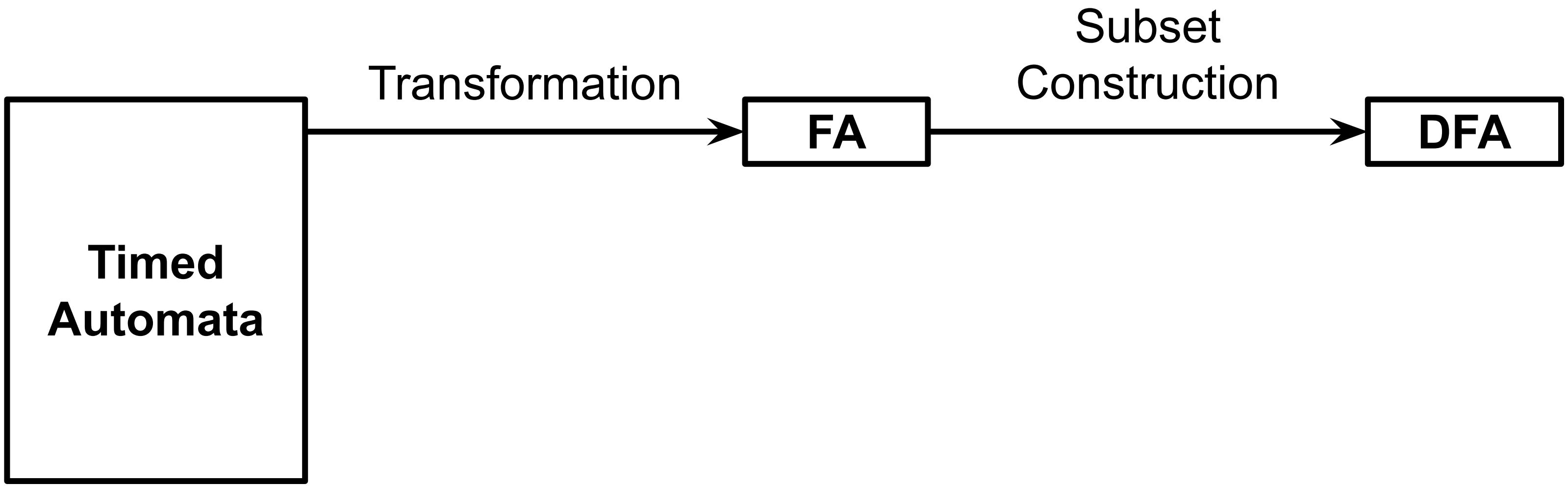
- … allows a **transformation** from TA to FA (**Gruber et al. 2005, Klein et al. 2024**)



FA-Based State Estimation

- … allows a **transformation** from TA to FA (**Gruber et al. 2005, Klein et al. 2024**)

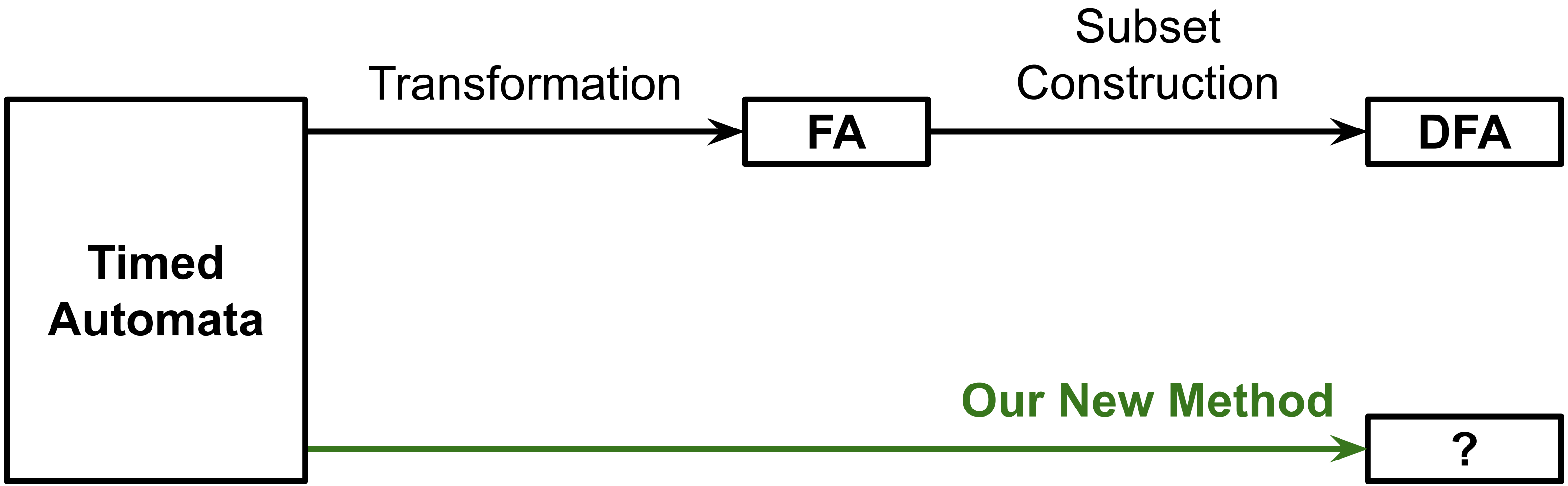
➔ enables standard state estimation methods like subset construction (**Noord 2000**)



FA-Based State Estimation

- … allows a **transformation** from TA to FA (**Gruber et al. 2005, Klein et al. 2024**)

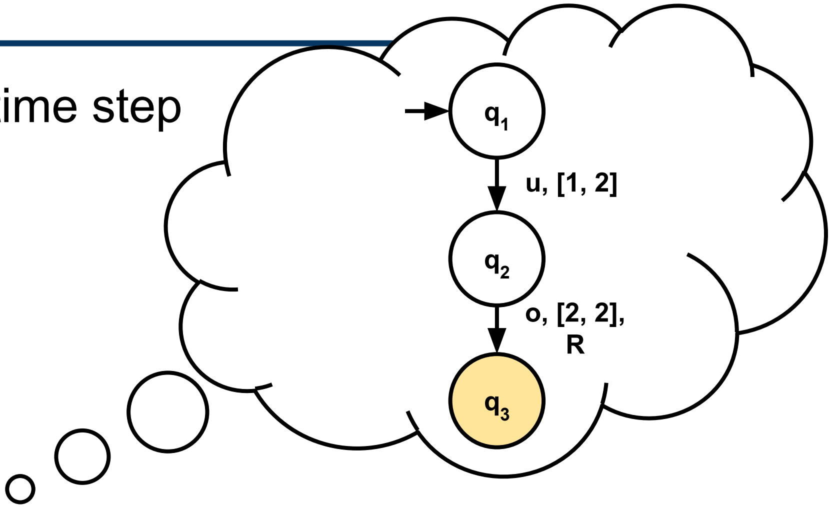
➔ enables standard state estimation methods like subset construction (**Noord 2000**)

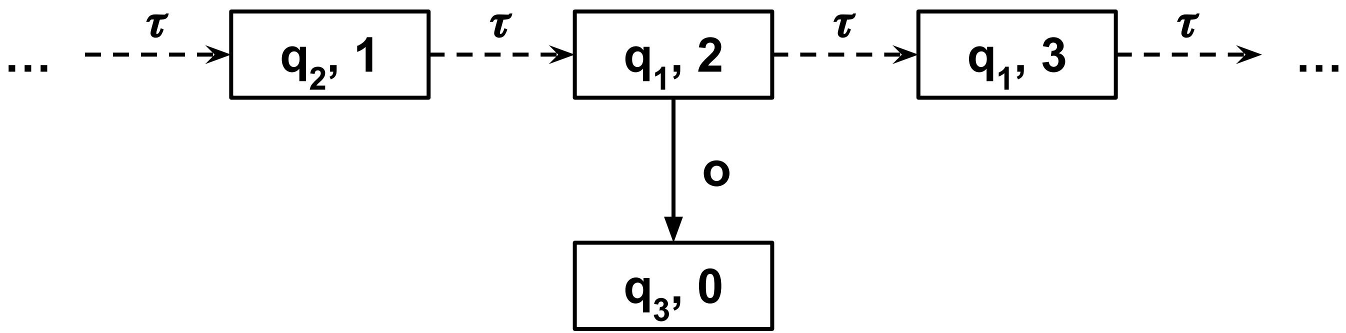


Our New Method

Stepwise State Estimation

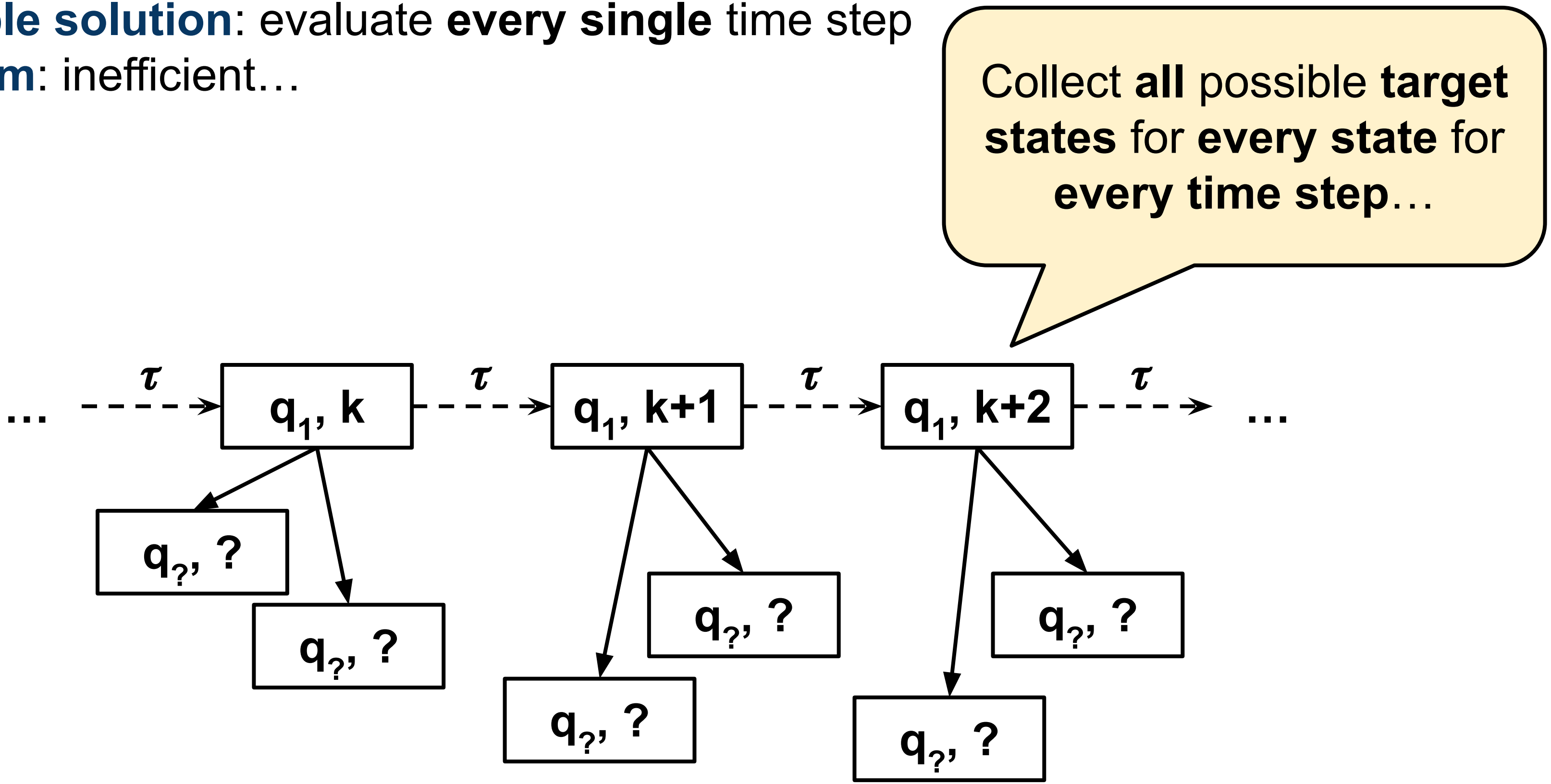
- **Possible solution**: evaluate **every single** time step
- **Problem**: inefficient…





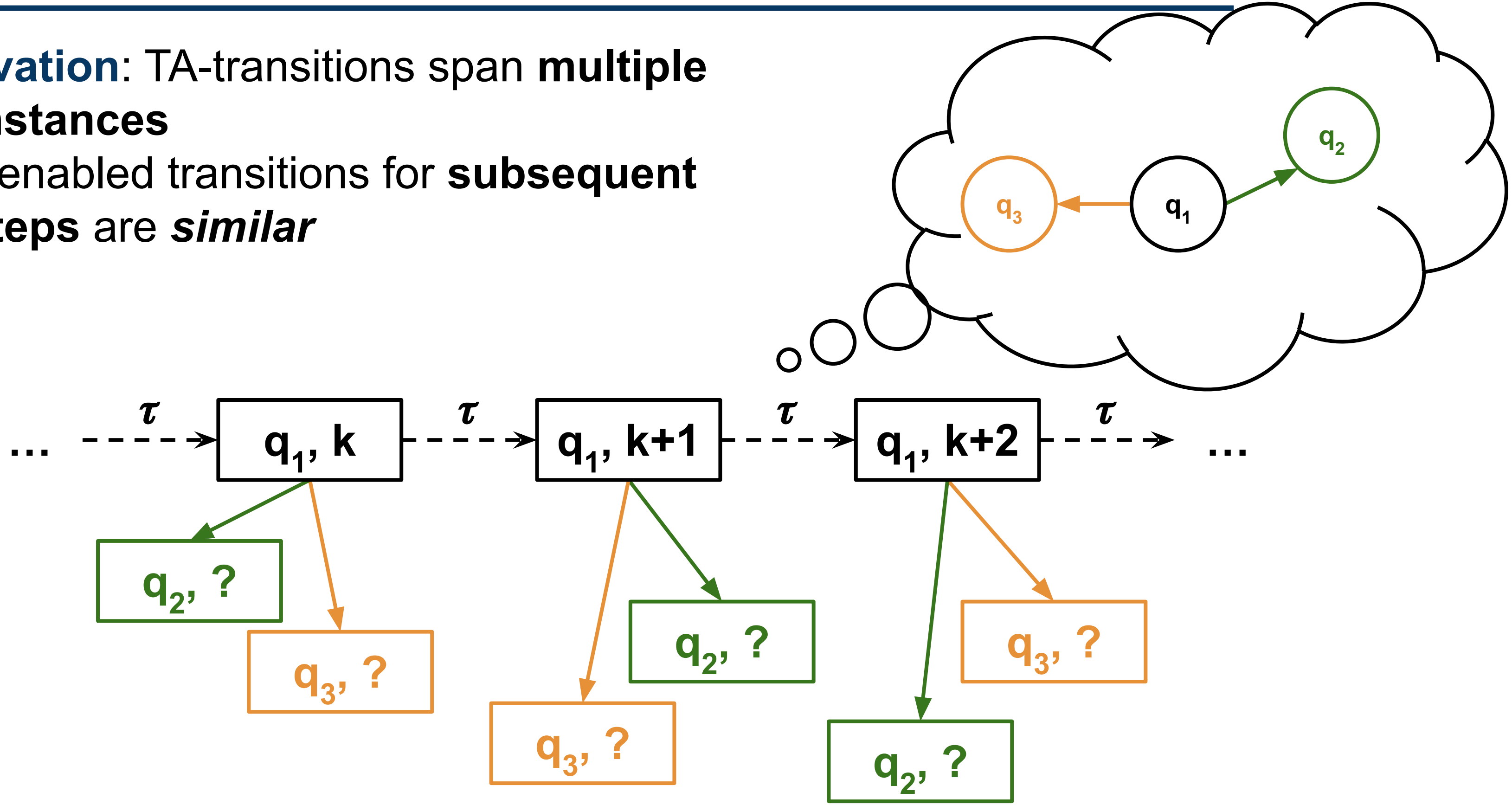
Stepwise State Estimation

- **Possible solution**: evaluate **every single time step**
- **Problem**: inefficient…



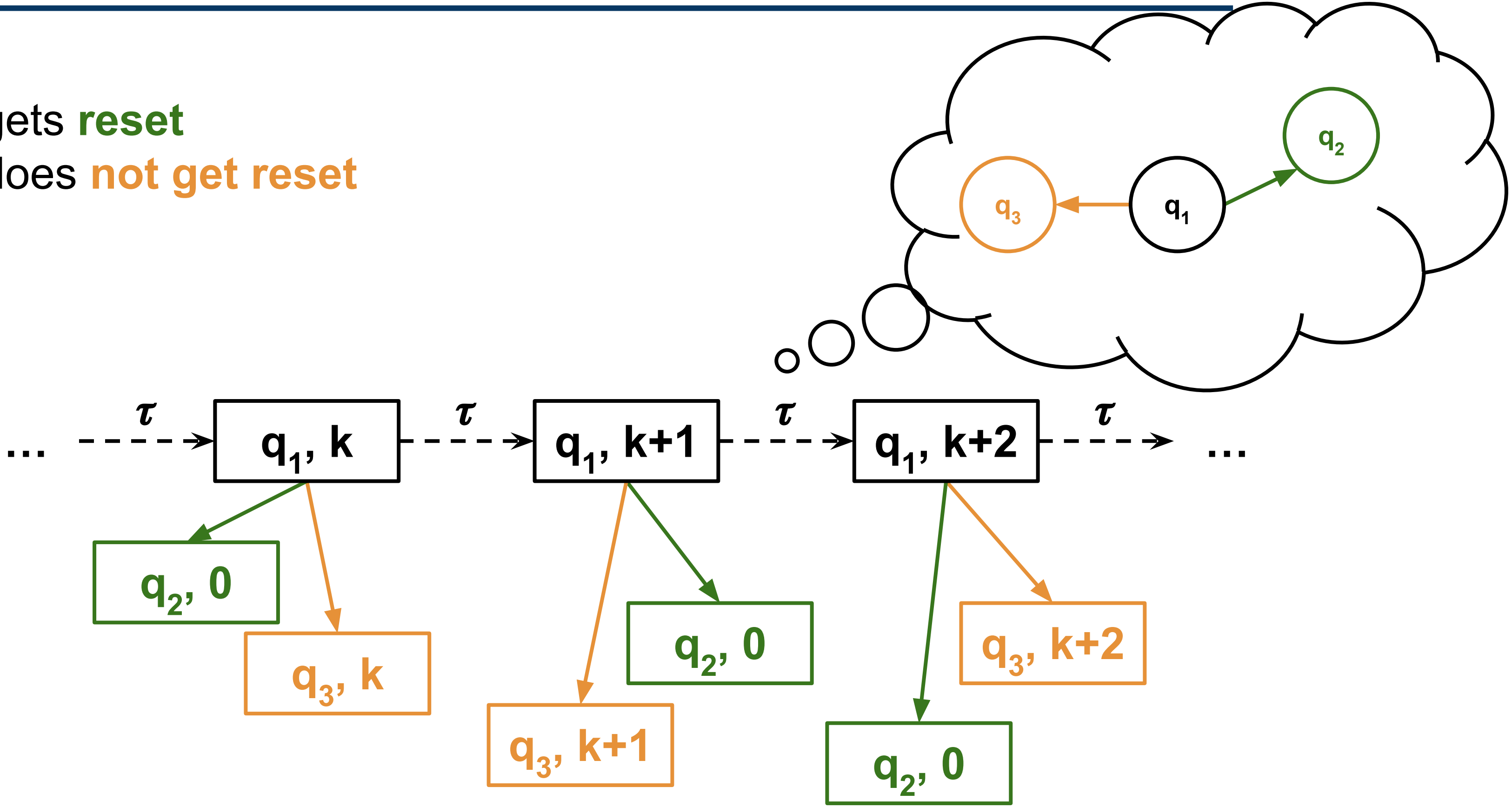
Threshold Estimation

- **Observation**: TA-transitions span **multiple time instances**
- **Hope**: enabled transitions for **subsequent time steps** are ***similar***



Threshold Estimation

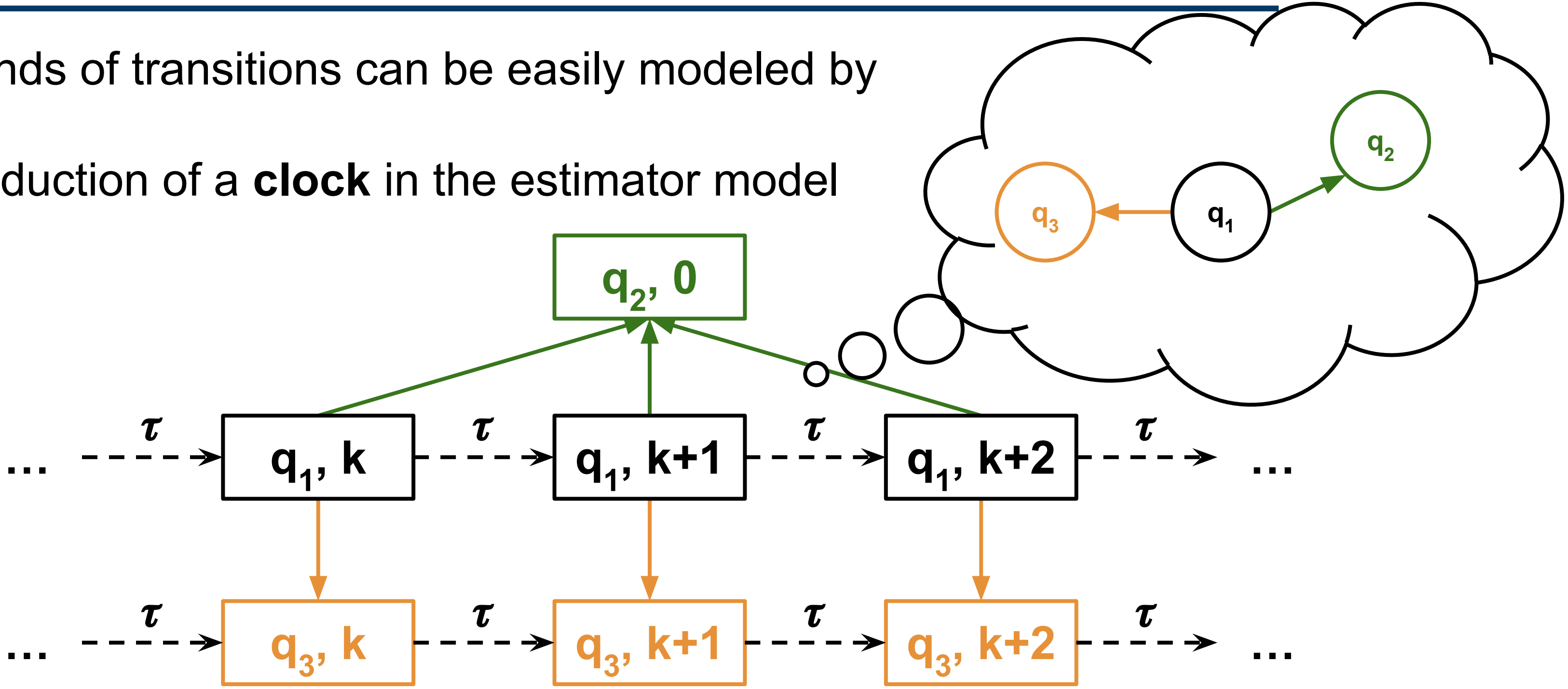
Two cases:
- Clock gets **reset**
- Clock does **not get reset**



Threshold Estimation

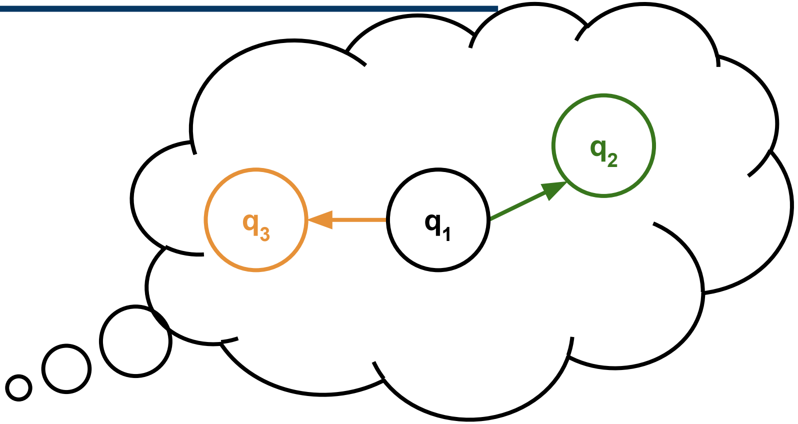
- Both kinds of transitions can be easily modeled by a clock

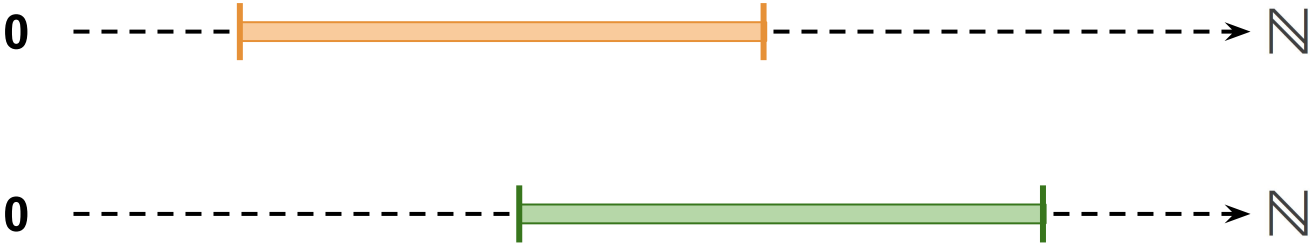
➔ Reintroduction of a **clock** in the estimator model



Threshold Values

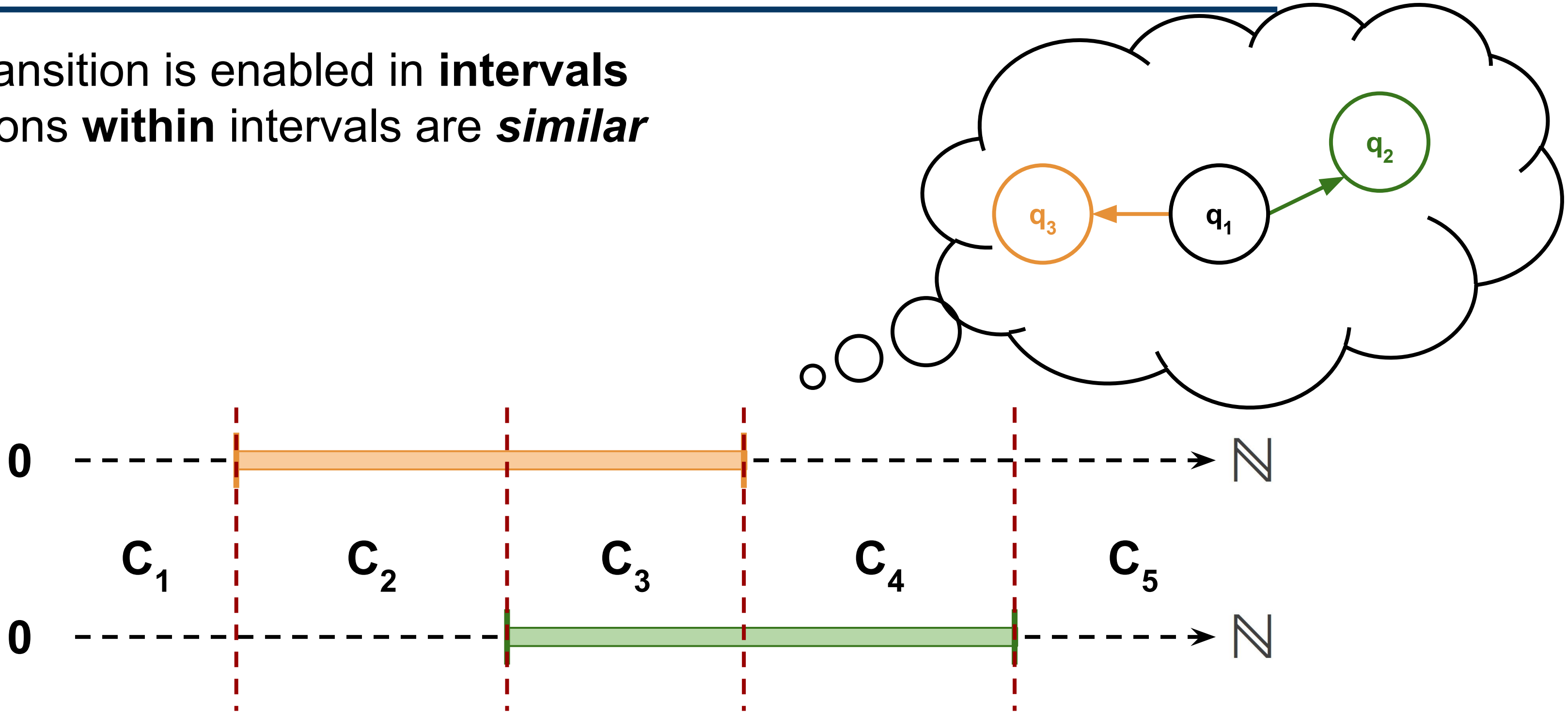
- Each transition is enabled in **intervals**
- Transitions **within** intervals are ***similar***





Threshold Values

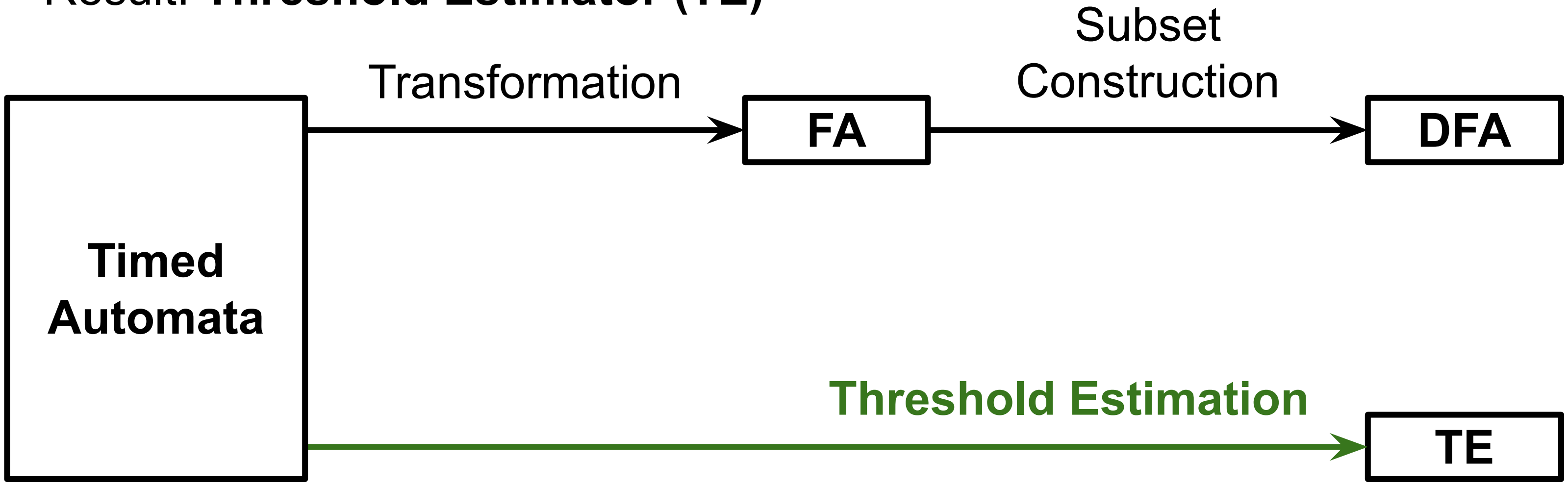
- Each transition is enabled in **intervals**
- Transitions **within** intervals are ***similar***



Threshold Estimators

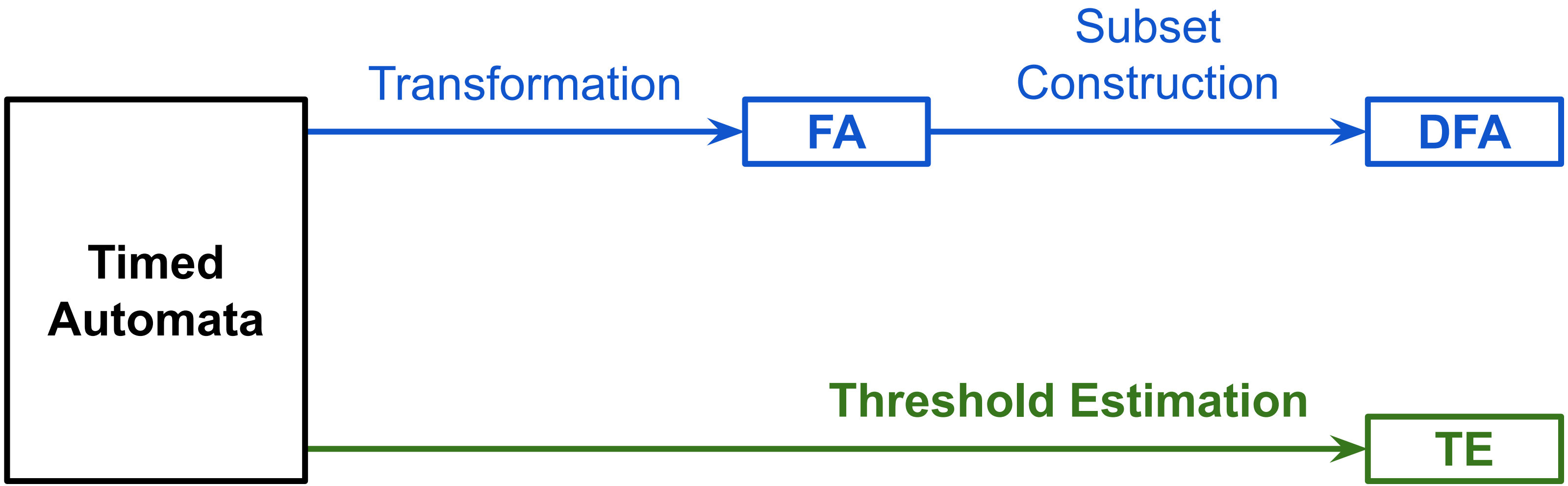
- Compute **threshold values** for each location
- Use threshold values in an **adapted subset construction**

➔ Result: **Threshold Estimator (TE)**



Evaluation

Evaluation: Test Setup



Evaluation: Case Studies

- Prototype implementation publicly available[1]
- Evaluation on 11 case studies from the literature

System	AKM	TCP	SCTP	PC	CAS	SCHED	OVEN	HVAC	WSN	WSNET	MED
Number of locations	4	11	41	8	8	23	89	11	63	25	8
Number of transitions	18	19	155	24	17	28	179	41	185	50	9
Largest constant	2500	240	1000	10	27000	15	5000	2000	$3 \cdot 10^5$	30	10
Source	Vaandrager et al. 2023	Postel et al. 1981	Stewart 2007	Aichernig et al. 2020	Aichernig et al 2013	An et al. 2021	Kogel et al. 2023	Taylor et al. 2021	Kogel et al. 2023	Klein et al. 2024	Klein et al. 2024

[1] https://gitlab.com/julianklein/threshold-estimation

Evaluation: Computation Time Results

- **Threshold estimation** outperforms **FA-based estimation** on most systems

System	Threshold Estimation	FA-based Estimation	Improvement
AKM	42.57ms	43.83ms	**2.86%**
TCP	16.25ms	16.36ms	**0.65%**
SCTP	0.66s	1.76ms	**62.11%**
PC	0.22ms	3.27ms	**92.99%**
CAS	4.50s	6.43s	**29.96%**
SCHED	0.34ms	4.29ms	**91.98%**
OVEN	1.77ms	1.51s	**99.88%**
HVAC	6.25s	36.92s	**83.06%**
WSN	11.94s	-	-
WSNET	7.81ms	10.57ms	**26.08%**
MED	0.25ms	0.32ms	**22.55%**

Conclusion

Summary

- **Efficient** state estimation method for discrete-timed automata
- Key idea: only evaluate threshold values
- Significant **reduction in computation** time compared to traditional FA-based methods

Future Work

- Adapt **state-based opacity** to our discrete-time setting

➔ **Unified opacity verification method** based on TE

References (1/2)

- **[1]** Lai, A., Lahaye, S., Giua, A.: Verification of detectability for unambiguous weighted automata. IEEE Transactions on Automatic Control (2020). https://doi.org/10.1109/TAC.2020.2995173
- **[2]** Li, J., Lefebvre, D., Hadjicostis, C.N., Li, Z.: Observers for a class of timed automata based on elapsed time graphs. IEEE Transactions on Automatic Control (2021). https://doi.org/10.1109/TAC.2021.3064542
- **[3]** Zhang, K.: Detectability of labeled weighted automata over monoids. Discrete Event Dynamic Systems (2022). https://doi.org/10.1007/s10626-022-00362-8
- **[4]** Lai, A., Lahaye, S., Giua, A.: State estimation of max-plus automata with unobservable events. Automatica (2019). https://doi.org/10.1016/j.automatica.2019.03.003
- **[5]** Gruber, H., Holzer, M., Kiehn, A., König, B.: On timed automata with discrete time - structural and language theoretical characterization. In: International Conference On Developments In Language Theory. Springer (2005). https://doi.org/10.1007/11505877_24
- **[6]** Klein, J., Kogel, P., Glesner, S.: Verifying opacity of discrete-timed automata. In: International Conference on Formal Methods in Software Engineering. Association for Computing Machinery (2024). https://doi.org/10.1145/3644033.3644376
- **[7]** Noord, G.v.: Treatment of epsilon moves in subset construction. Computational Linguistics (2000). https://doi.org/10.1162/089120100561638
- **[8]** Vaandrager, F., Ebrahimi, M., Bloem, R.: Learning mealy machines with one timer. Information and Computation (2023). https://doi.org/10.1016/j.ic.2023.105013

References (2/2)

- **[9]** Postel, J.: Transmission Control Protocol. RFC 793 (Sep 1981). https://doi.org/10.17487/RFC0793, https://www.rfc-editor.org/info/rfc793
- **[10]** Stewart, R.: Stream control transmission protocol. Tech. rep. (2007)
- **[11]** Aichernig, B.K., Pferscher, A., Tappler, M.: From passive to active: Learning timed automata efficiently. In: NASA Formal Methods. Springer (2020). https://doi.org/10.1007/978-3-030-55754-6_1
- **[12]** Aichernig, B.K., Lorber, F., Ničković, D.: Time for mutants — model-based mutation testing with timed automata. In: Veanes, M., Viganò, L. (eds.) Tests and Proofs. pp. 20–38. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
- **[13]** An, J., Zhan, B., Zhan, N., Zhang, M.: Learning nondeterministic real-time automata. ACM Trans. Embed. Comput. Syst. (2021). https://doi.org/10.1145/3477030
- **[14]** Kogel, P., Klös, V., Glesner, S.: Learning mealy machines with local timers. In: International Conference on Formal Engineering Methods. Springer (2023). https://doi.org/10.1007/978-981-99-7584-6_23
- **[15]** Taylor, J.T., Taylor, W.T.: Patterns in the Machine. Springer (2021)
- **[16]** Baier, C., Bertrand, N., Bouyer, P., & Brihaye, T. When are timed automata determinizable?. In Automata, Languages and Programming. Springer (2009). https://doi.org/10.1007/978-3-642-02930-1_4

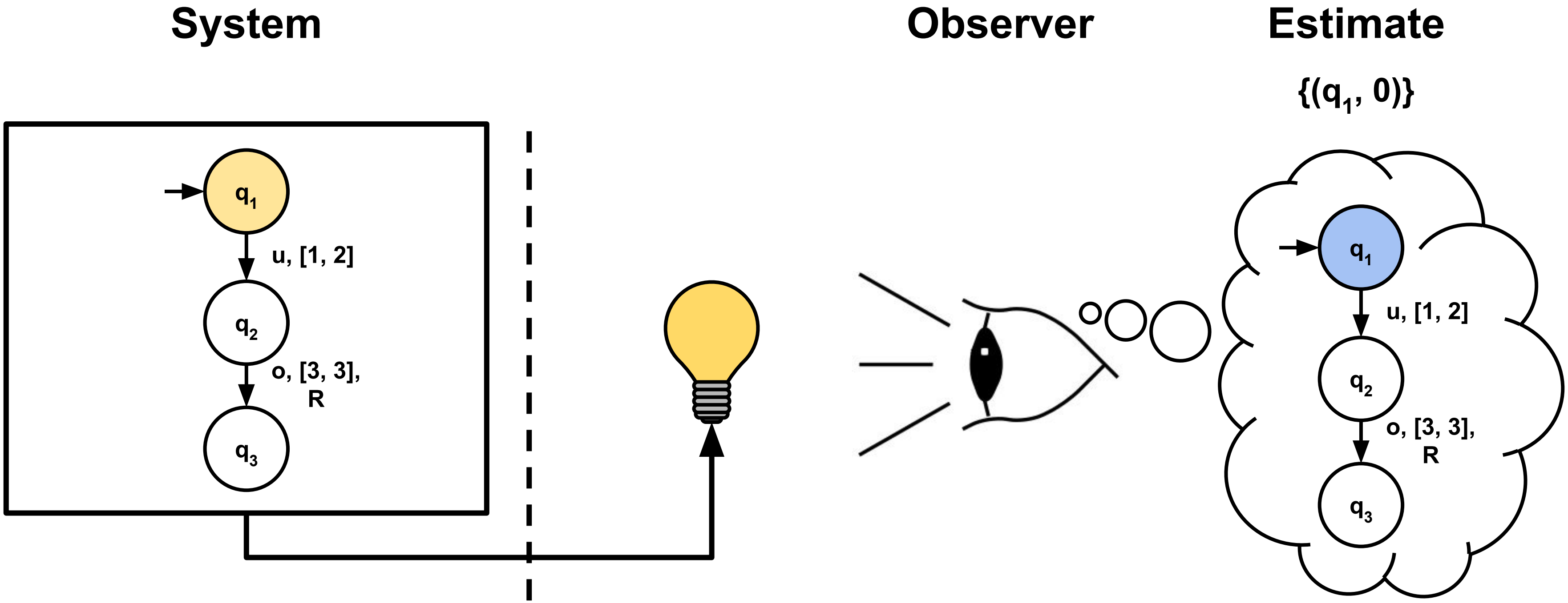
TITLE

Region Abstraction

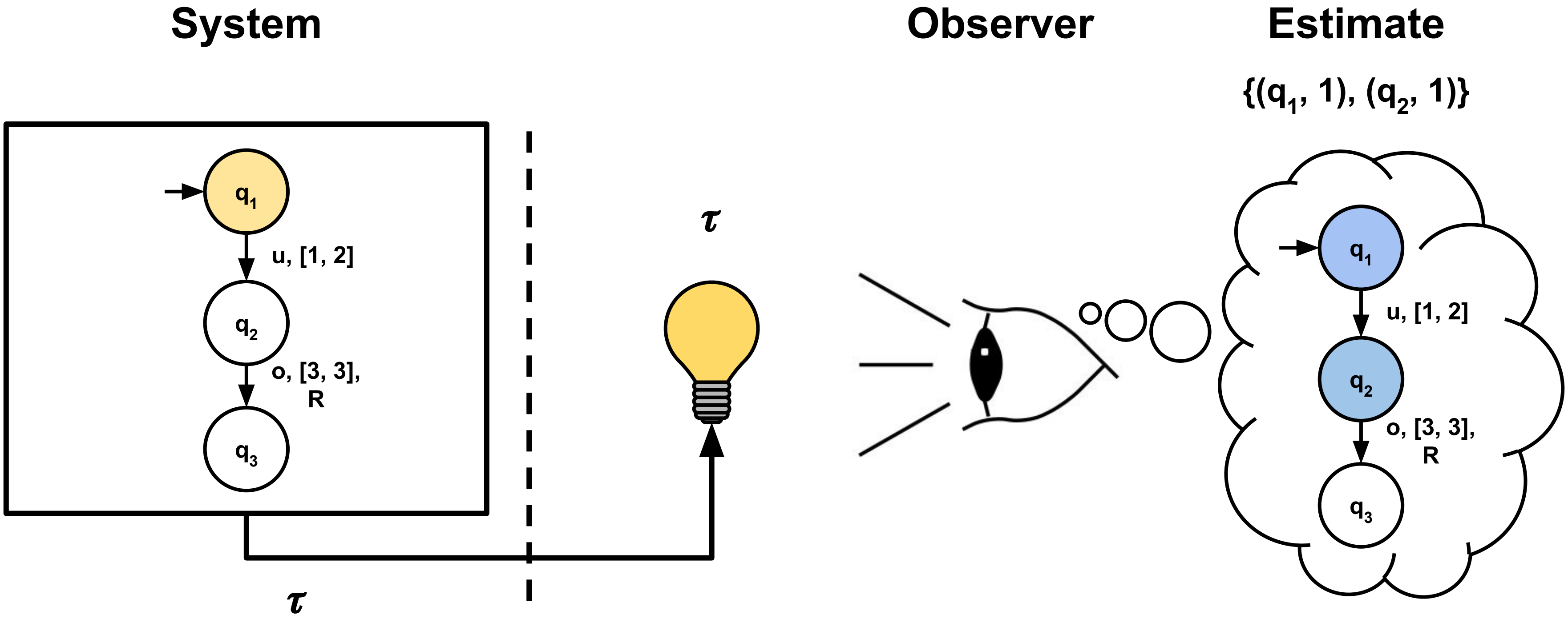
- Threshold estimation outperforms FA-based estimation on most systems

System	Threshold Estimation	FA-based Estimation	Improvement
`AKM`	18.51s	32.53s	**43.11%**
`TCP`	0.18s	0.29s	**38.13%**
`SCTP`	21.62s	-	-
`PC`	0.21ms	3.43ms	**93.73%**
`CAS`	-	-	-
`SCHED`	0.30ms	4.12ms	**92.71%**
`OVEN`	1.09ms	11.51s	**99.99%**
`HVAC`	16.70s	-	-
`WSN`	-	-	-
`WSNET`	13.05ms	28.16ms	**53.65%**
`MED`	0.85ms	3.68ms	**76.81%**

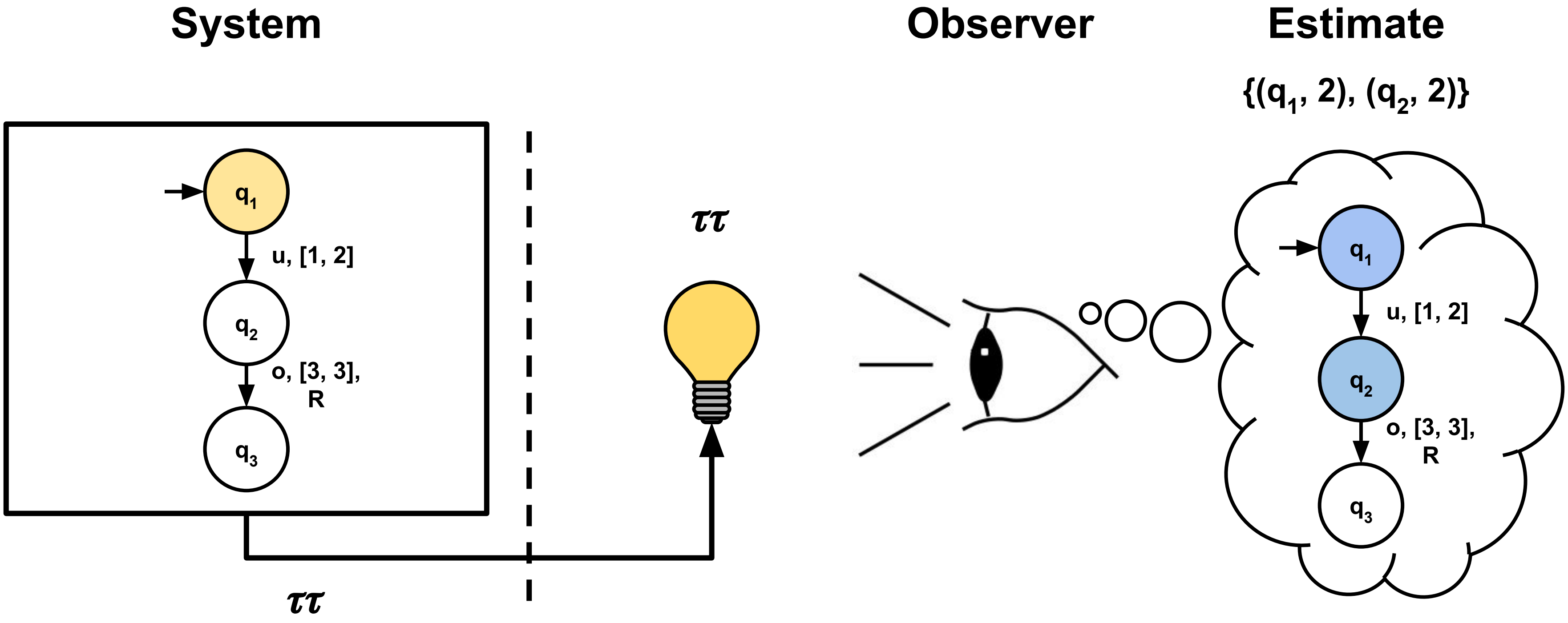
State Estimation



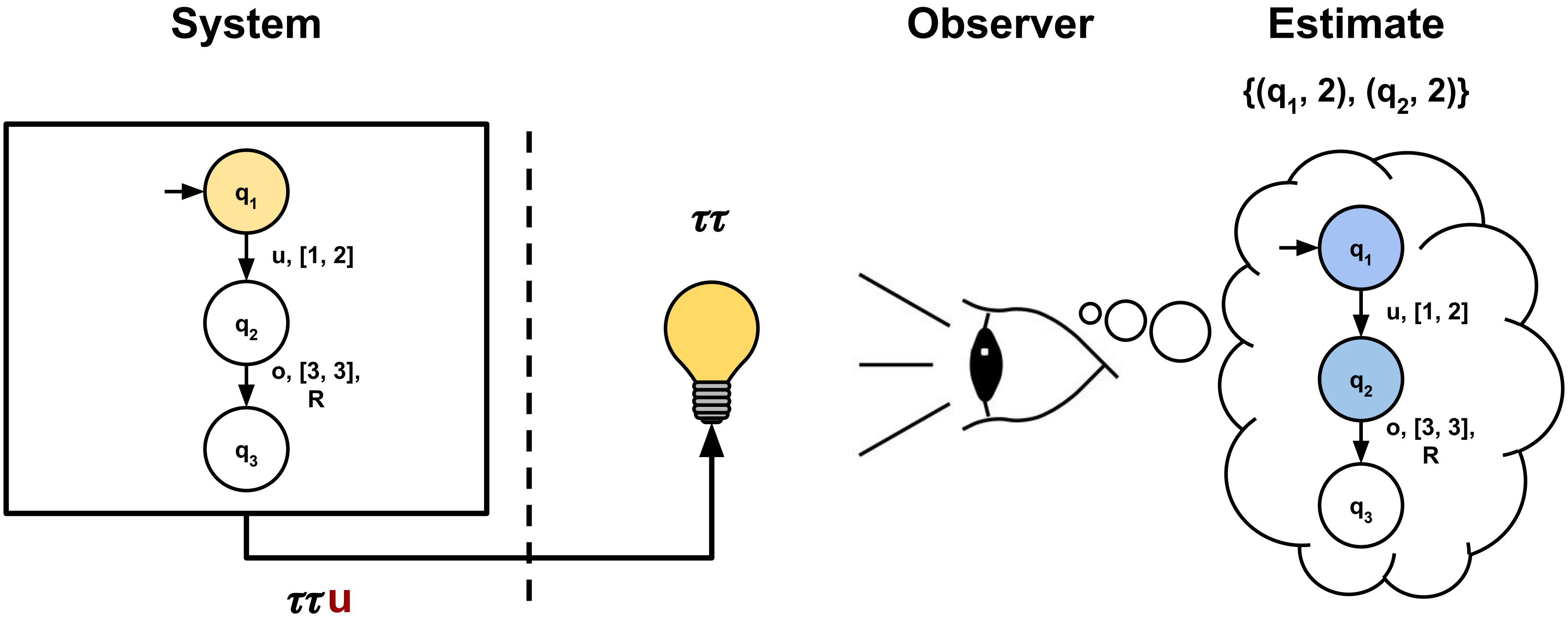
State Estimation



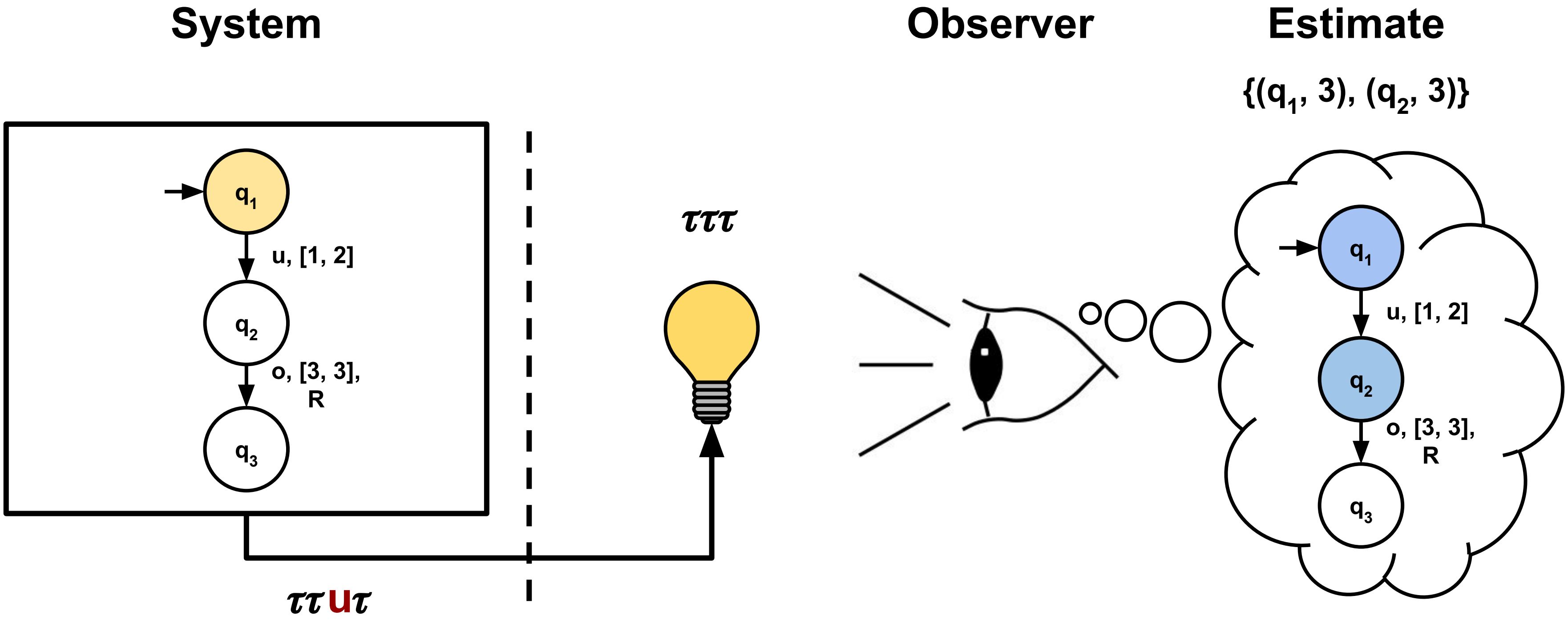
State Estimation



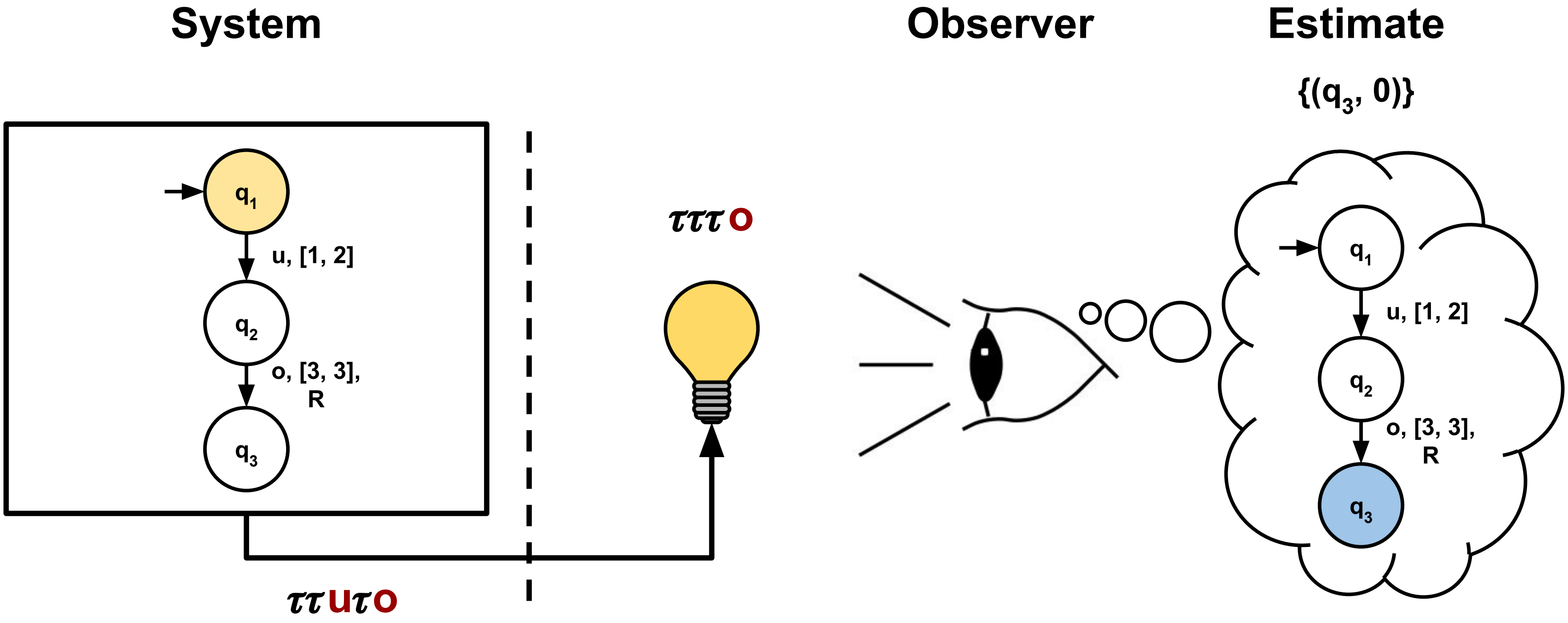
State Estimation



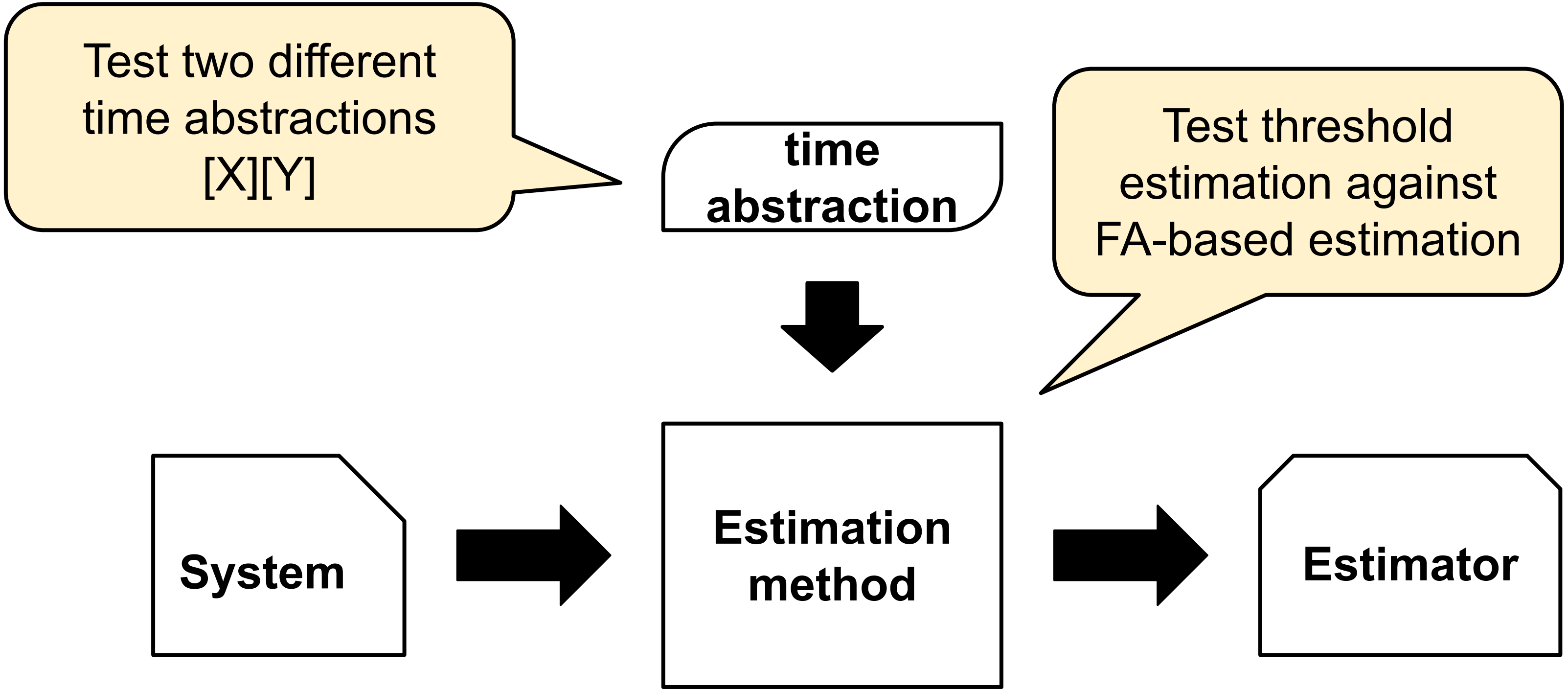
State Estimation



State Estimation



Test Setup


Test two different
time abstractions
[X][Y]
time
abstraction
Test threshold
estimation against
FA-based estimation
System
Estimation
method
Estimator

0-10	0-10
11-20	11-20
21-30	21-30
31-40	31-40
41-50	41-50
51-60	51-60
61-70	61-70
71-80	71-80
81-90	81-90
91-100	91-100