UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

isp

Runtime Monitoring with Union-Find Structures

N. Decker J. Harder T. Scheffel M. Schmitz D. Thoma

{decker,harder,scheffel,schmitz,thoma}@isp.uni-luebeck.de

Institute for Software Engineering and Programming Languages,
University of Lübeck, Germany

TACAS 2016, Eindhoven

March 7, 2016

Runtime Verification

- (On-line) verification of a single run
- "Word problem" $run \stackrel{?}{\in} SPEC$

Tasks

- Specification
- Evaluation
  - Monitor construction
  - Monitor execution

Runtime Verification

- (On-line) verification of a single run
- "Word problem" $run \stackrel{?}{\in} SPEC$

Tasks

- Specification
- Evaluation
  - Monitor construction
  - Monitor execution

Goals

- Convenience and expressiveness
- Efficiency (on-line: overhead minimisation)

Object-oriented Systems

Observations

- Behaviour, interaction of individual objects

Object-oriented Systems

Observations

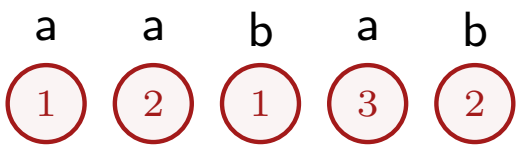
- Behaviour, interaction of individual objects
- Sequence of events

a a b a b

Object-oriented Systems

Observations

- Behaviour, interaction of individual objects
- Sequence of events
- Object IDs (a.k.a. event parameter, data value)

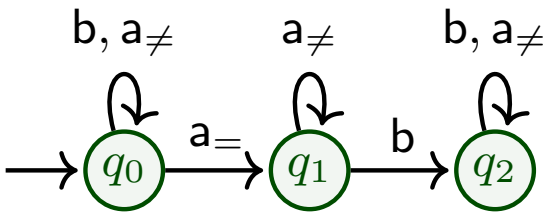


Object-oriented Systems

Monitor

- Operational model: projection automata
- “Local” perspective, “global” information
  - Execute one automaton instance per object
  - Dispatch and qualify observations individually

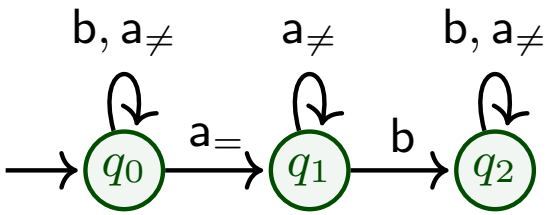
Example



$a_=, a_\neq$	local event
b	global event

Object	State
1	q_1
2	q_0
3	q_2
4	q_2
5	q_0

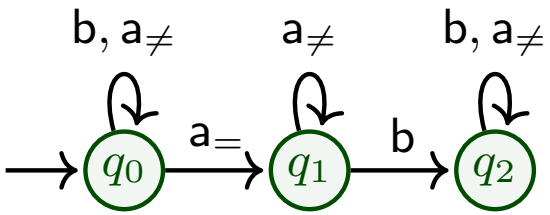
Observation $(\mathsf{a}, 5)$



$\mathsf{a}_=, \mathsf{a}_\neq$	local event
b	global event

Object	State	
1	q_1	
2	q_0	
3	q_2	
4	q_2	
5	~~q_0~~	q_1

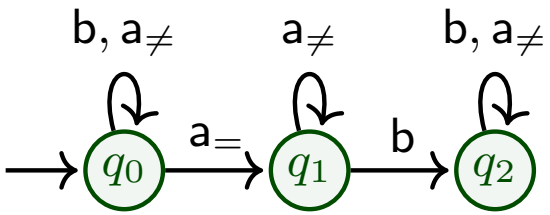
Observation (b, ◯)



$a_=, a_\neq$	local event
b	global event

Object	State	
1	~~q_1~~	q_2
2	q_0	
3	q_2	
4	q_2	
5	q_1	

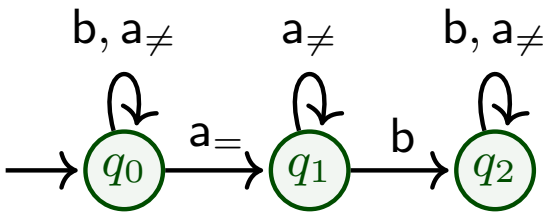
Observation (b, ◯)



$a_=, a_\neq$	local event
b	global event

Object	State	
1	~~q_1~~	q_2
2	~~q_0~~	q_0
3	q_2	
4	q_2	
5	q_1	

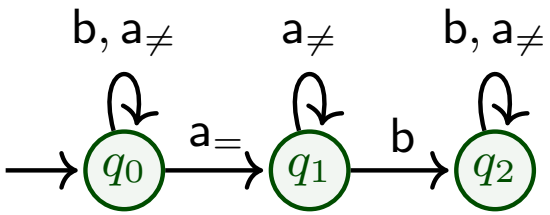
Observation (b, ◯)



$a_=, a_\neq$	local event
b	global event

Object	State	
1	~~q_1~~	q_2
2	~~q_0~~	q_0
3	~~q_2~~	q_2
4	q_2	
5	q_1	

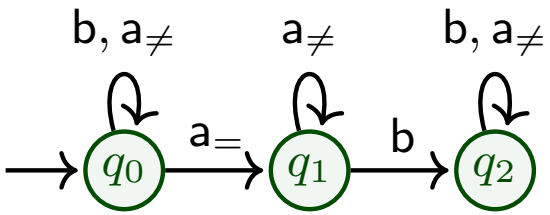
Observation (b, ◯)



$a_=, a_\neq$	local event
b	global event

Object	State	
1	~~q_1~~	q_2
2	~~q_0~~	q_0
3	~~q_2~~	q_2
4	~~q_2~~	q_2
5	q_1	

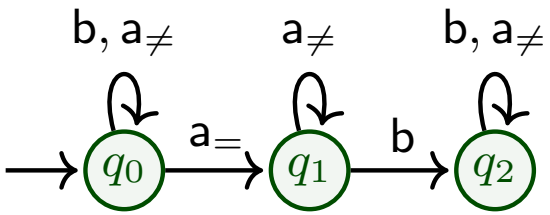
Observation (b, ◯)



$a_=, a_{\neq}$	local event
b	global event

Object	State	
1	~~q_1~~	q_2
2	~~q_0~~	q_0
3	~~q_2~~	q_2
4	~~q_2~~	q_2
5	~~q_1~~	q_2

Observation (b, ◯)



$a_=$, $a_\neq$	local event
b	global event

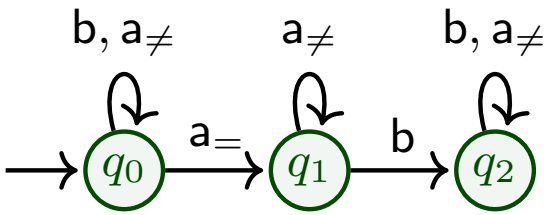
Object	State	
1	~~q_1~~	q_2
2	~~q_0~~	q_0
3	~~q_2~~	q_2
4	~~q_2~~	q_2
5	~~q_1~~	q_2

Suitable data structure?

- Hash tables

JAVAMOP [Chen and Rosu], MARQ [Reger et al.]

Observation $(b, \bigcirc)$



$a_=, a_{\neq}$	local event
b	global event

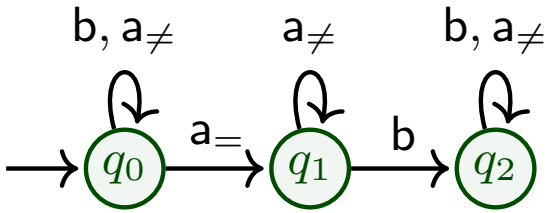
Object	State	
1	~~q_1~~	q_2
2	~~q_0~~	q_0
3	~~q_2~~	q_2
4	~~q_2~~	q_2
5	~~q_1~~	q_2

Suitable data structure?

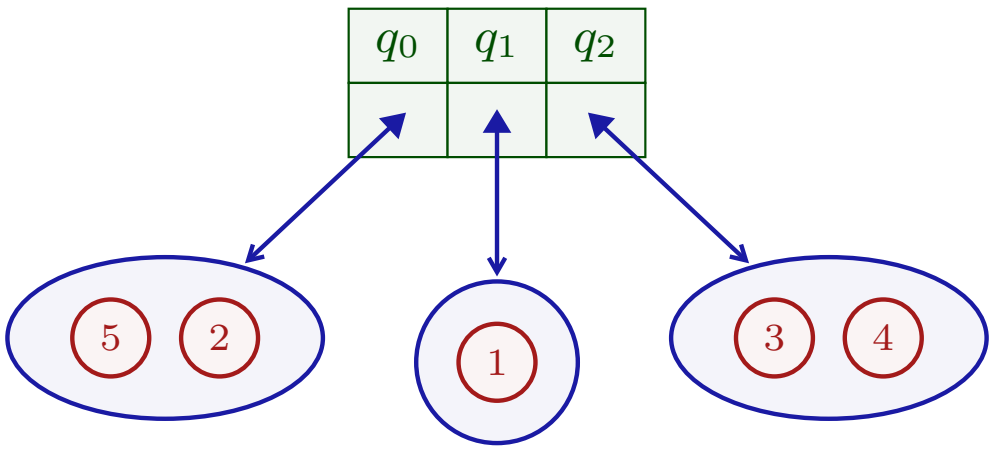
- Hash tables
- Union-Find structures

JavaMOP [Chen and Rosu], MarQ [Reger et al.]

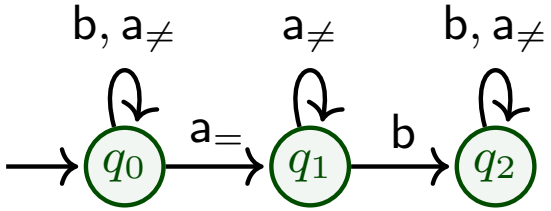
Union-Find

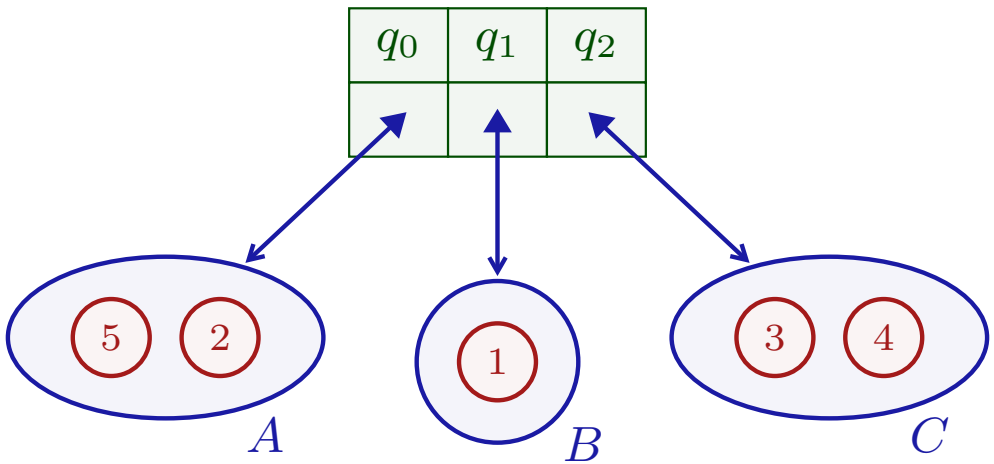


Object	State
1	q_1
2	q_0
3	q_2
4	q_2
5	q_0



Union-Find: Dispatch $\mathsf{a}_=$ to (5)





find (5) yields A

$A \longleftrightarrow q_0$

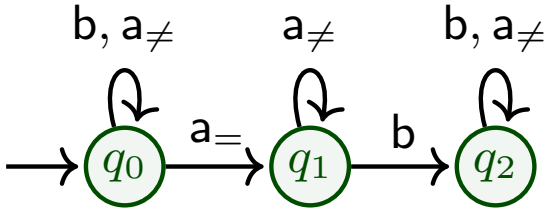
$\delta(q_0, \mathsf{a}_=) = q_1$

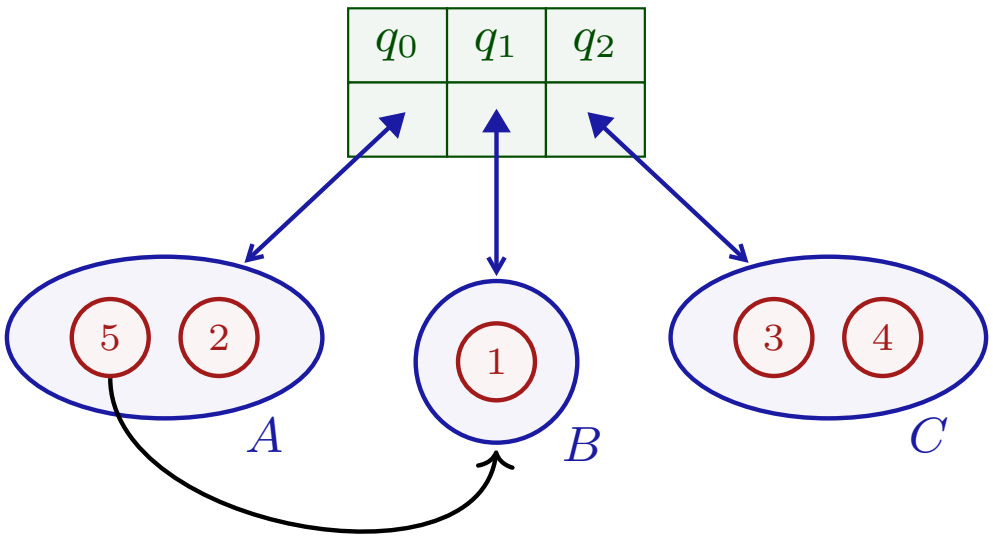
$q_1 \longleftrightarrow B$

delete (5), A

union $\{(5)\}$, B

Union-Find: Dispatch $\mathsf{a}_=$ to (5)





find (5) yields A

$A \longleftrightarrow q_0$

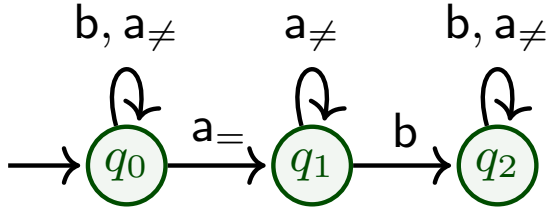
$\delta(q_0, \mathsf{a}_=) = q_1$

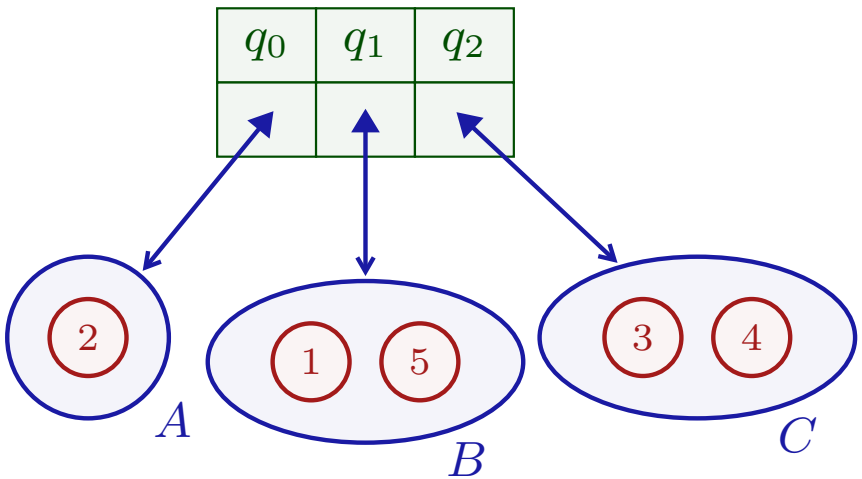
$q_1 \longleftrightarrow B$

delete (5), A

union $\{(5)\}$, B

Union-Find: Dispatch $\mathsf{a}_=$ to (5)





find (5) yields A

$A \longleftrightarrow q_0$

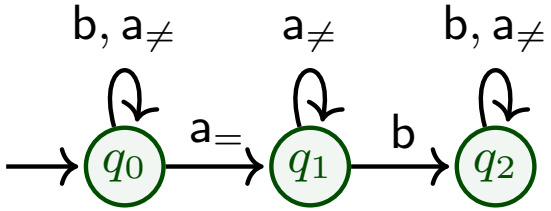
$\delta(q_0, \mathsf{a}_=) = q_1$

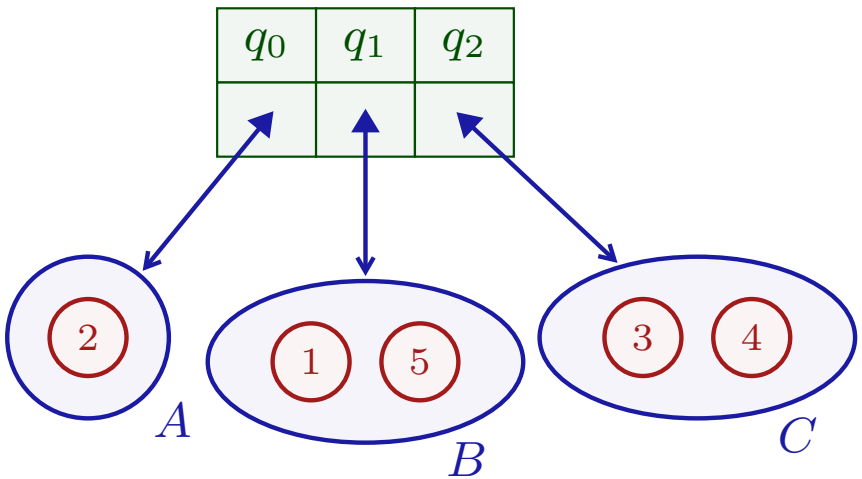
$q_1 \longleftrightarrow B$

delete (5), A

union $\{(5)\}$, B

Union-Find: Dispatch b to All





$\delta(q_0, \mathsf{b}) = q_0$

$\delta(q_1, \mathsf{b}) = q_2$

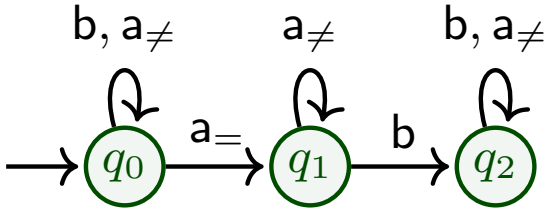
$\delta(q_2, \mathsf{b}) = q_2$

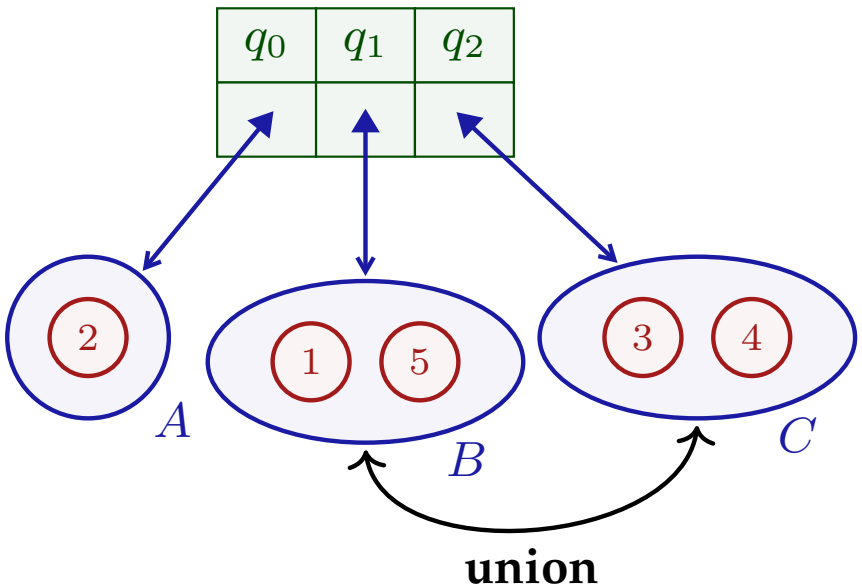
$q_1 \longleftrightarrow B$

$q_2 \longleftrightarrow C$

union B, C

Union-Find: Dispatch b to All





$\delta(q_0, \mathsf{b}) = q_0$

$\delta(q_1, \mathsf{b}) = q_2$

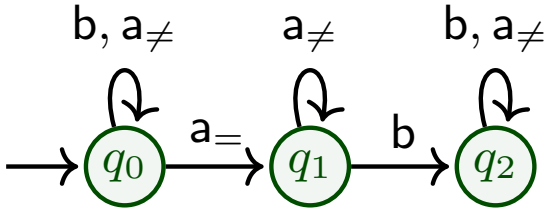
$\delta(q_2, \mathsf{b}) = q_2$

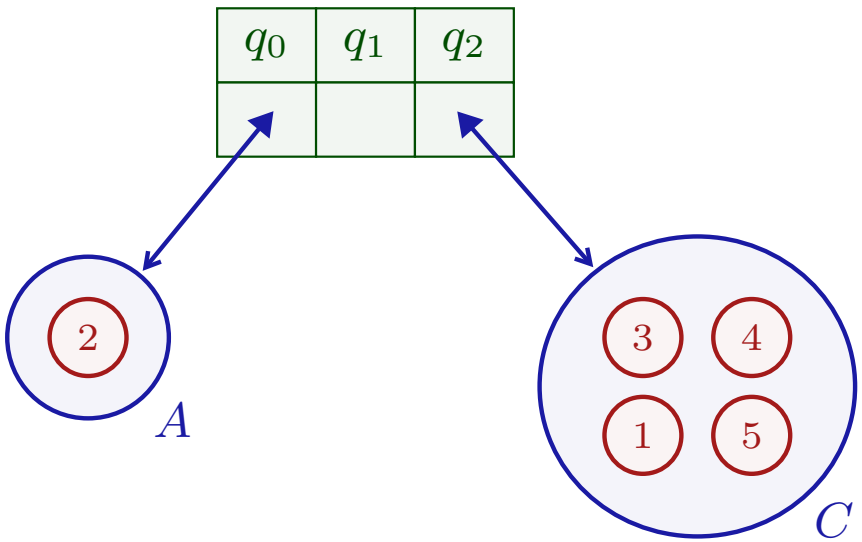
$q_1 \longleftrightarrow B$

$q_2 \longleftrightarrow C$

union B, C

Union-Find: Dispatch b to All





$\delta(q_0, \mathsf{b}) = q_0$

$\delta(q_1, \mathsf{b}) = q_2$

$\delta(q_2, \mathsf{b}) = q_2$

$q_1 \longleftrightarrow B$

$q_2 \longleftrightarrow C$

union B, C

Relation Between Objects

Union-Find structures allow for updating

- individual objects ($\mathsf{a}_{=}$)
- all objects (b)

Relation Between Objects

Union-Find structures allow for updating

- individual objects ($\mathsf{a}_{=}$)
- all objects (b)
- all but one object ($\mathsf{a}_{\neq}$)

Relation Between Objects

Union-Find structures allow for updating

- individual objects ($\mathsf{a}_{=}$)
- all objects (b)
- all but one object ($\mathsf{a}_{\neq}$)
- hierarchically structured subsets of objects

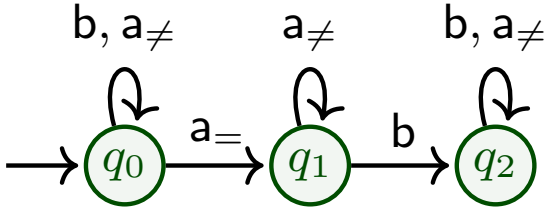
Relation Between Objects

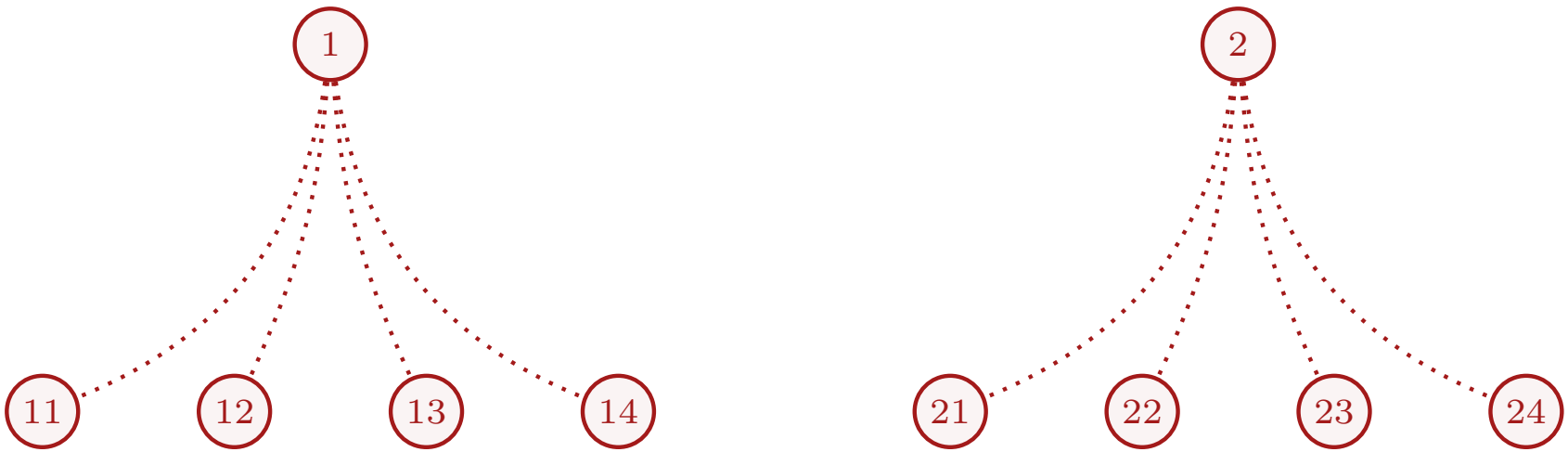
Union-Find structures allow for updating

- individual objects ($\mathsf{a}_{=}$)
- all objects (b)
- all but one object ($\mathsf{a}_{\neq}$)
- hierarchically structured subsets of objects
  - $\texttt{resource} < \texttt{lock}$
  - $\texttt{collection} < \texttt{iterator1}, \texttt{iterator2}$
  - $\texttt{immList} < \texttt{head} < \texttt{tail}$

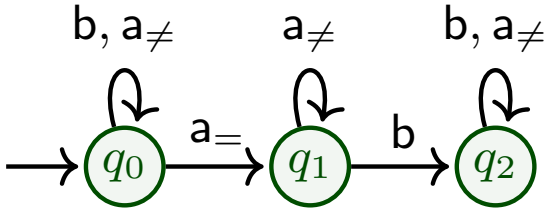
($\Rightarrow$ Tree structure)

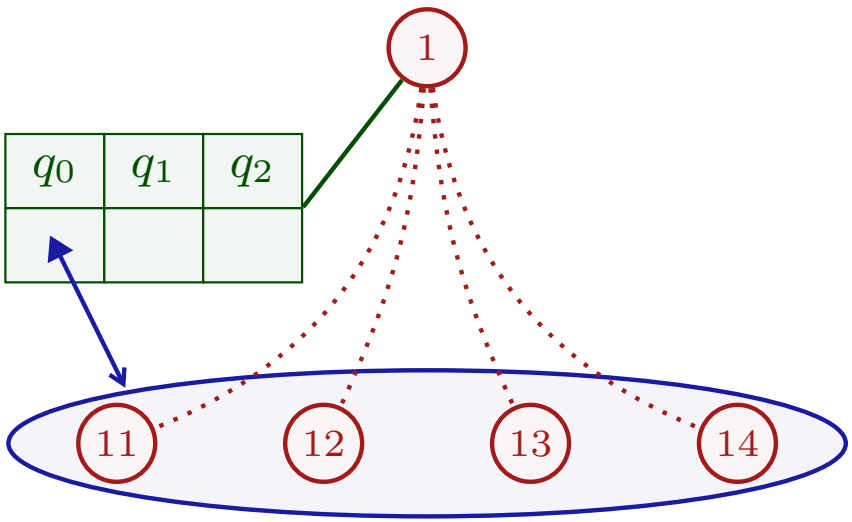
Trees

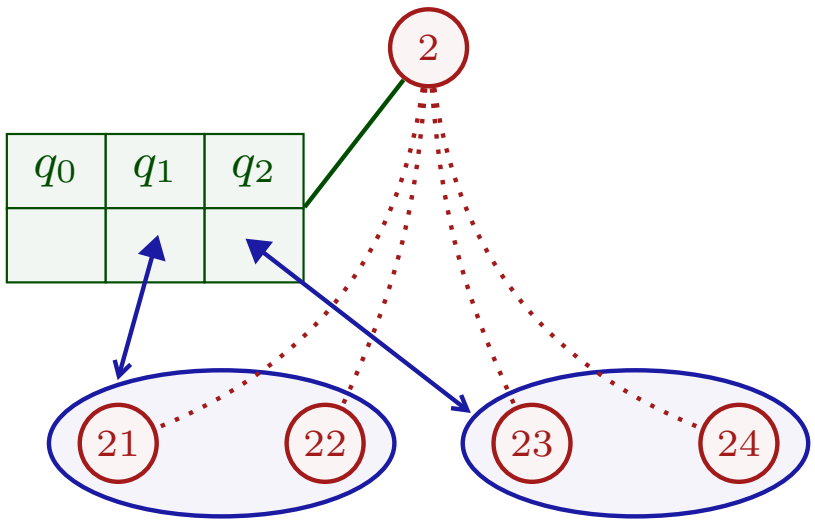




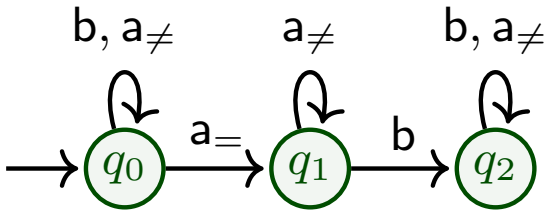
Trees

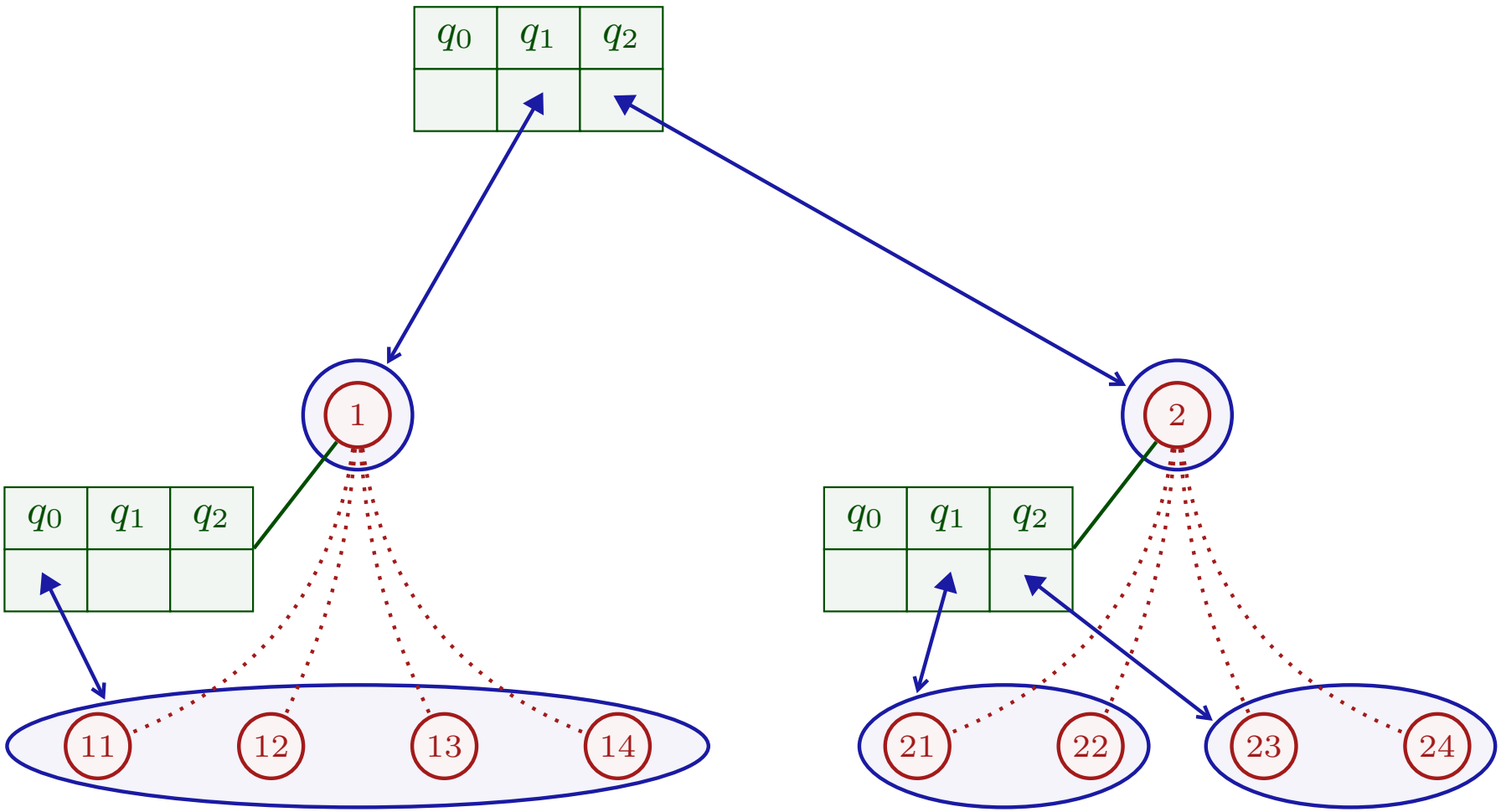




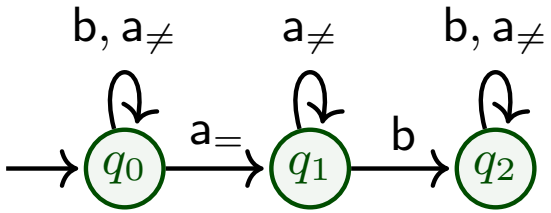


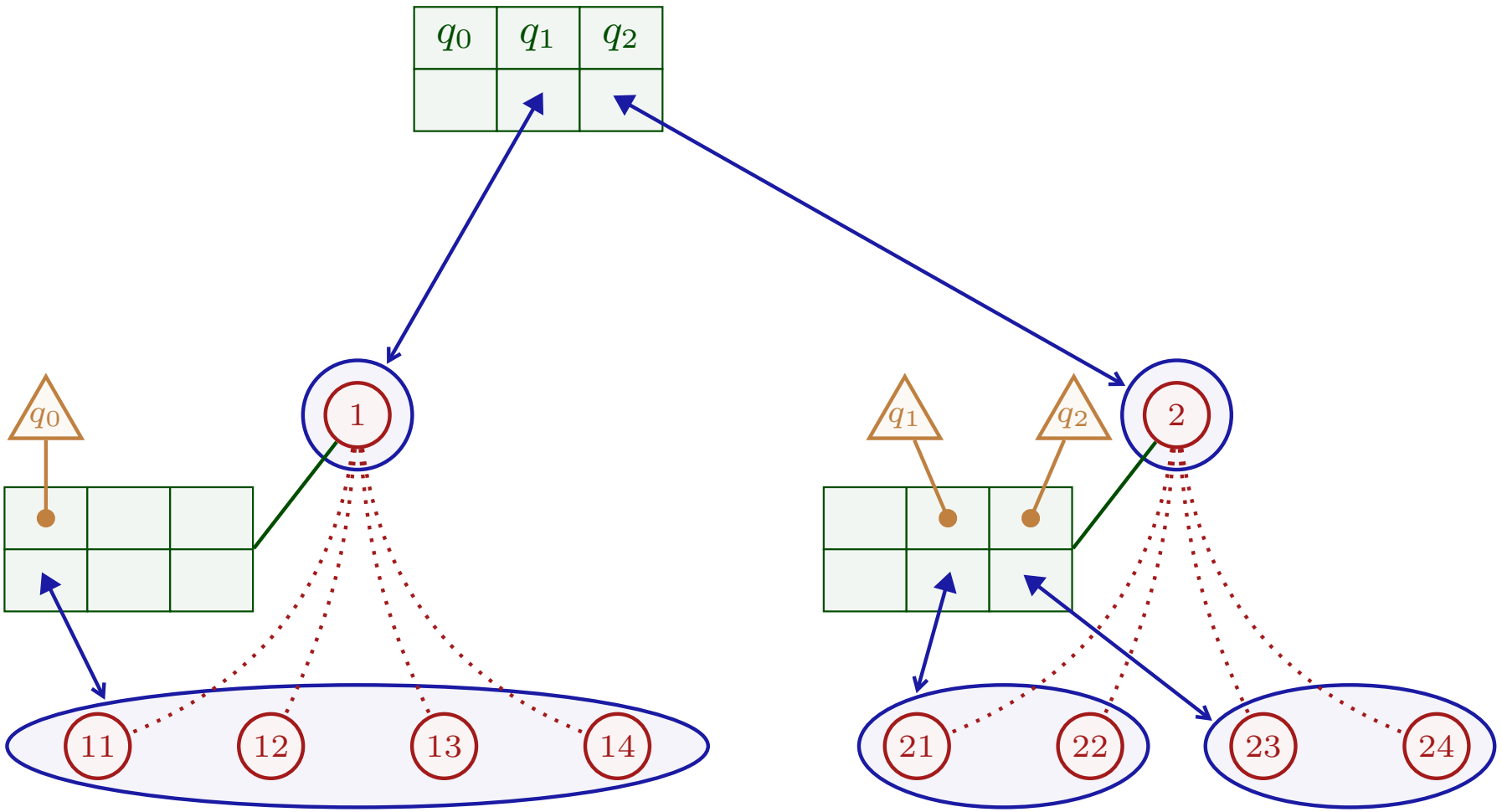
Trees



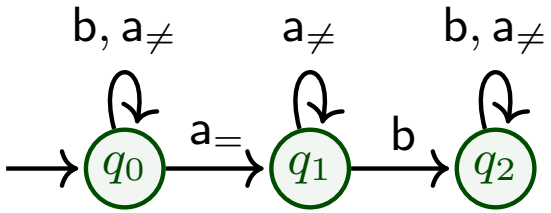


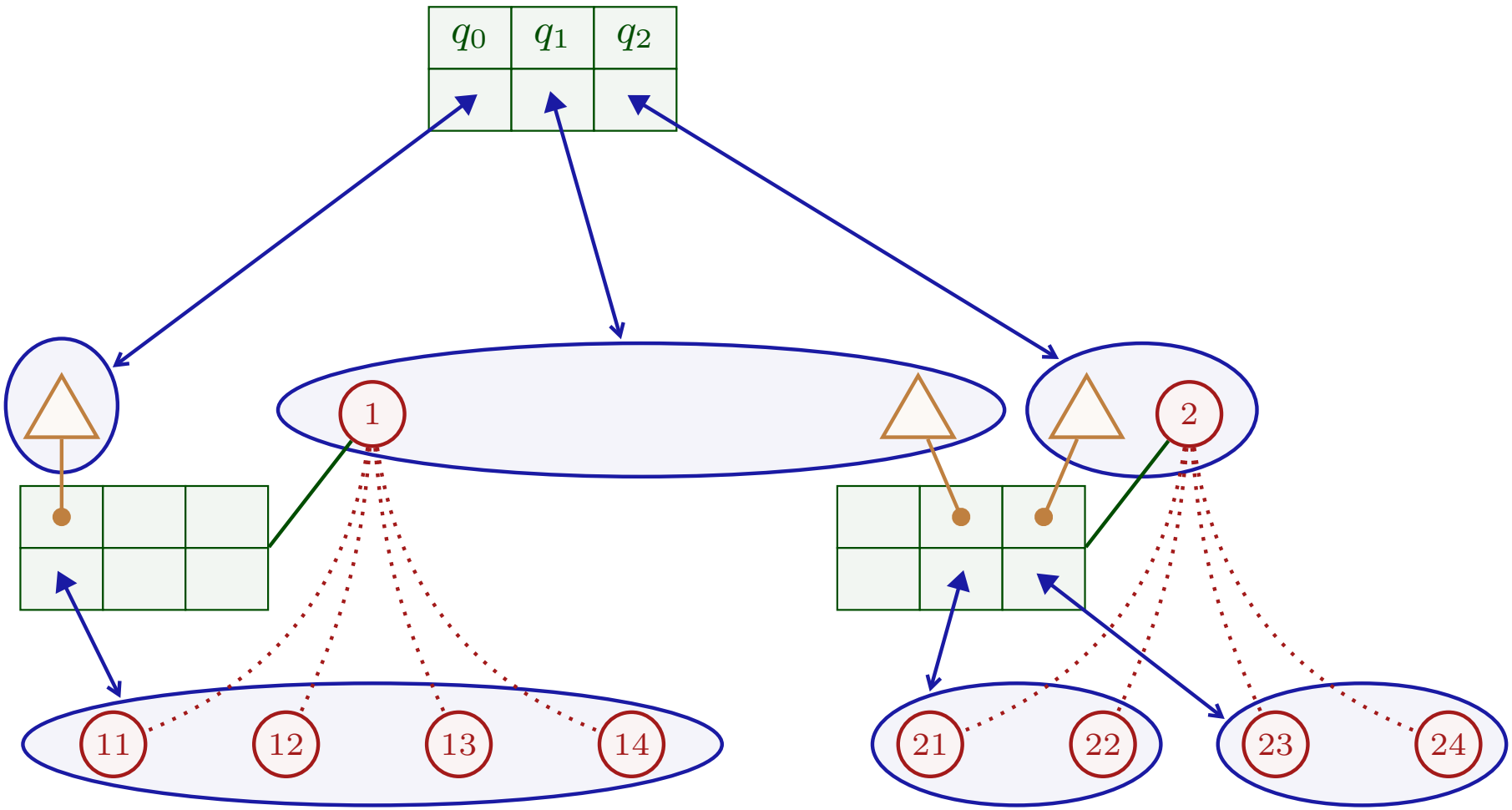
Trees



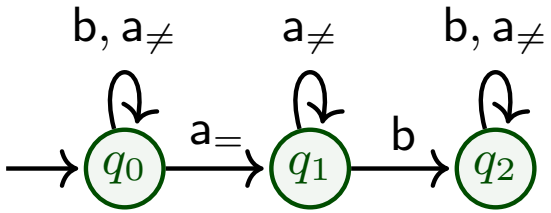


Trees





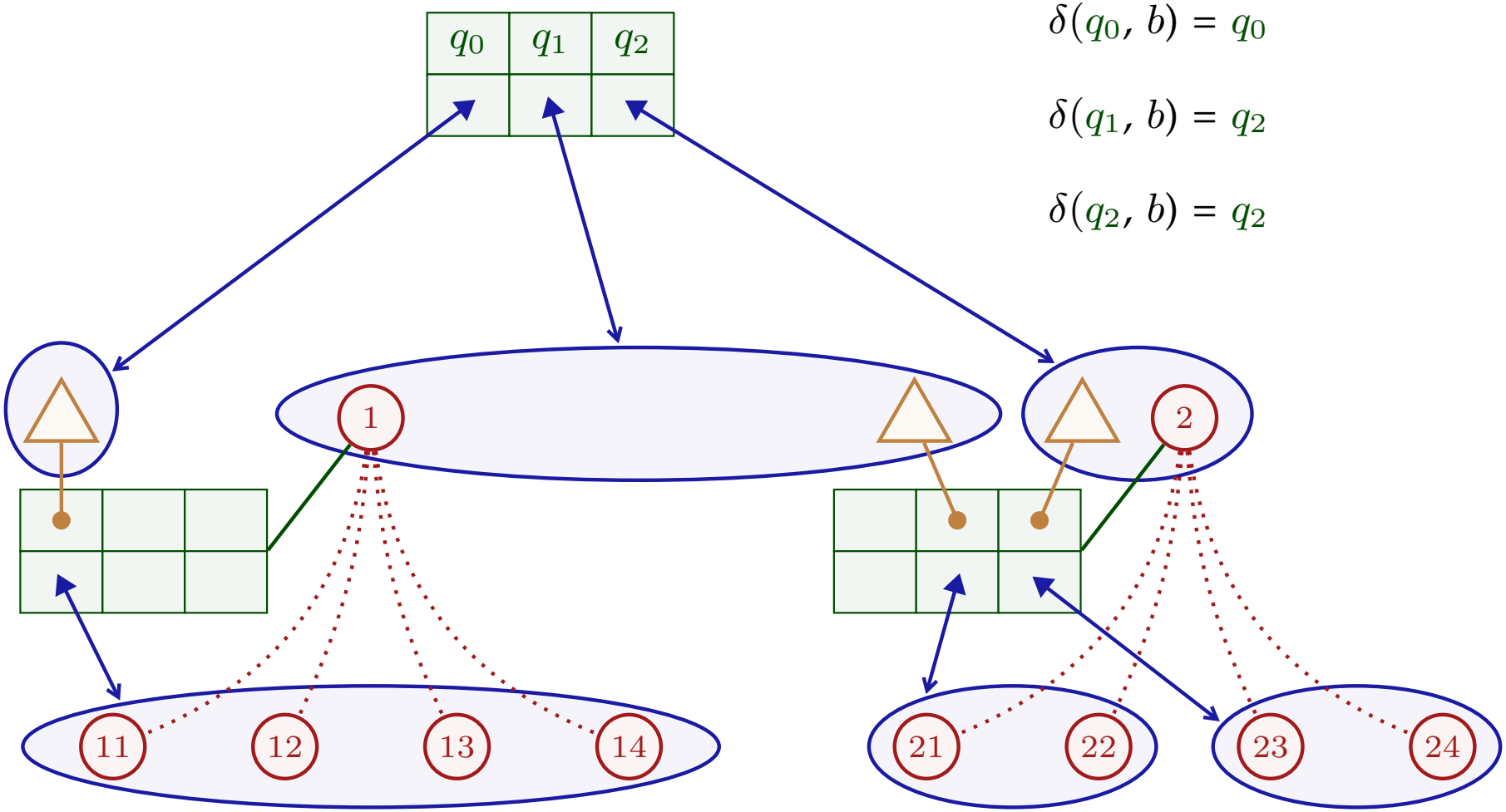
Trees: Dispatch b to All Objects



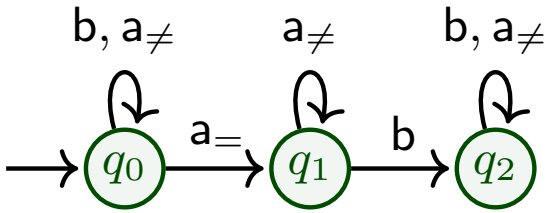
$\delta(q_0, b) = q_0$

$\delta(q_1, b) = q_2$

$\delta(q_2, b) = q_2$



Trees: Dispatch b to All Objects

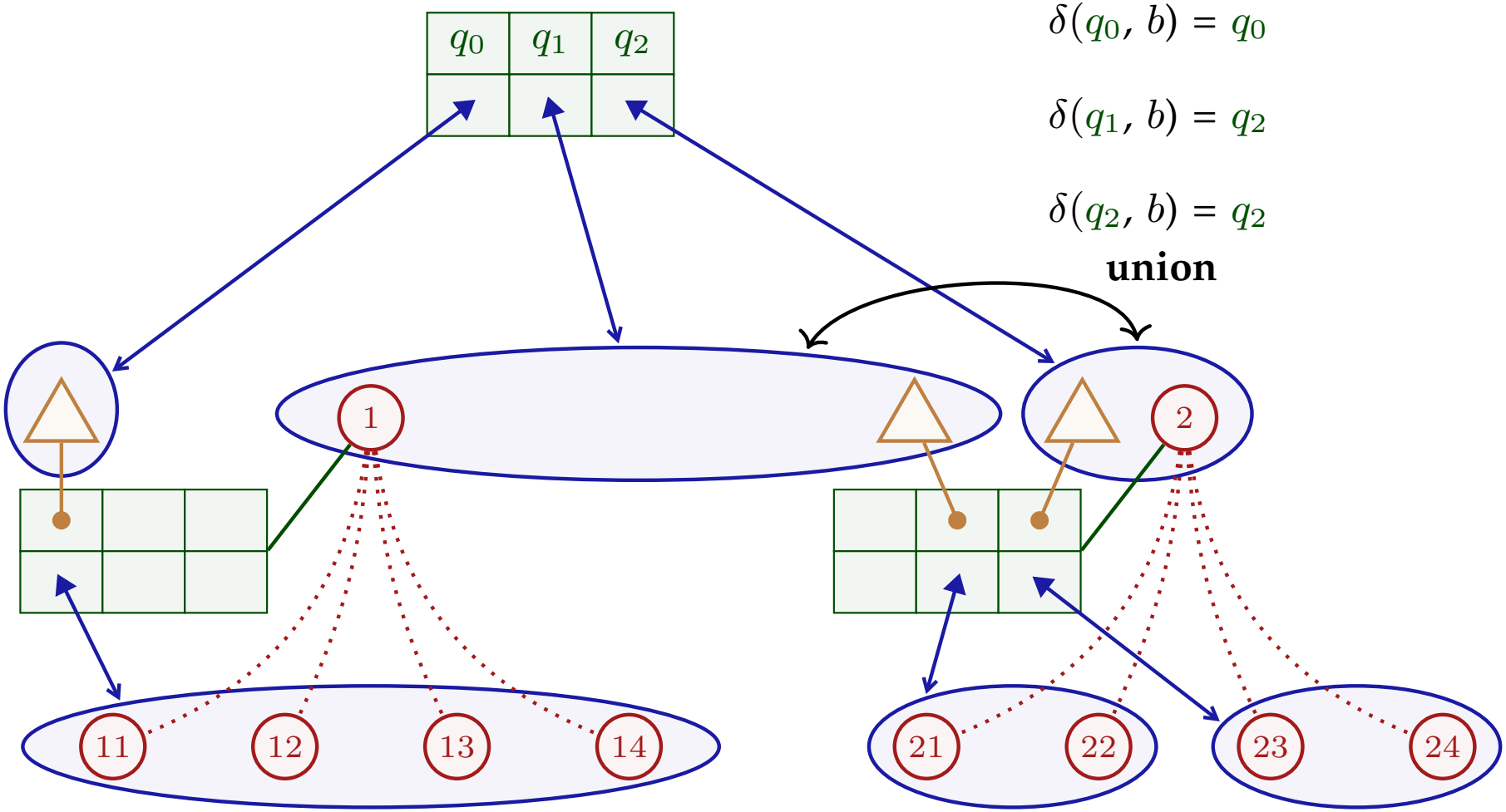


$\delta(q_0, b) = q_0$

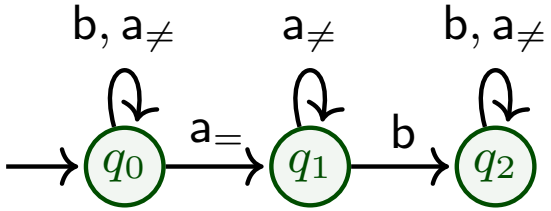
$\delta(q_1, b) = q_2$

$\delta(q_2, b) = q_2$

union



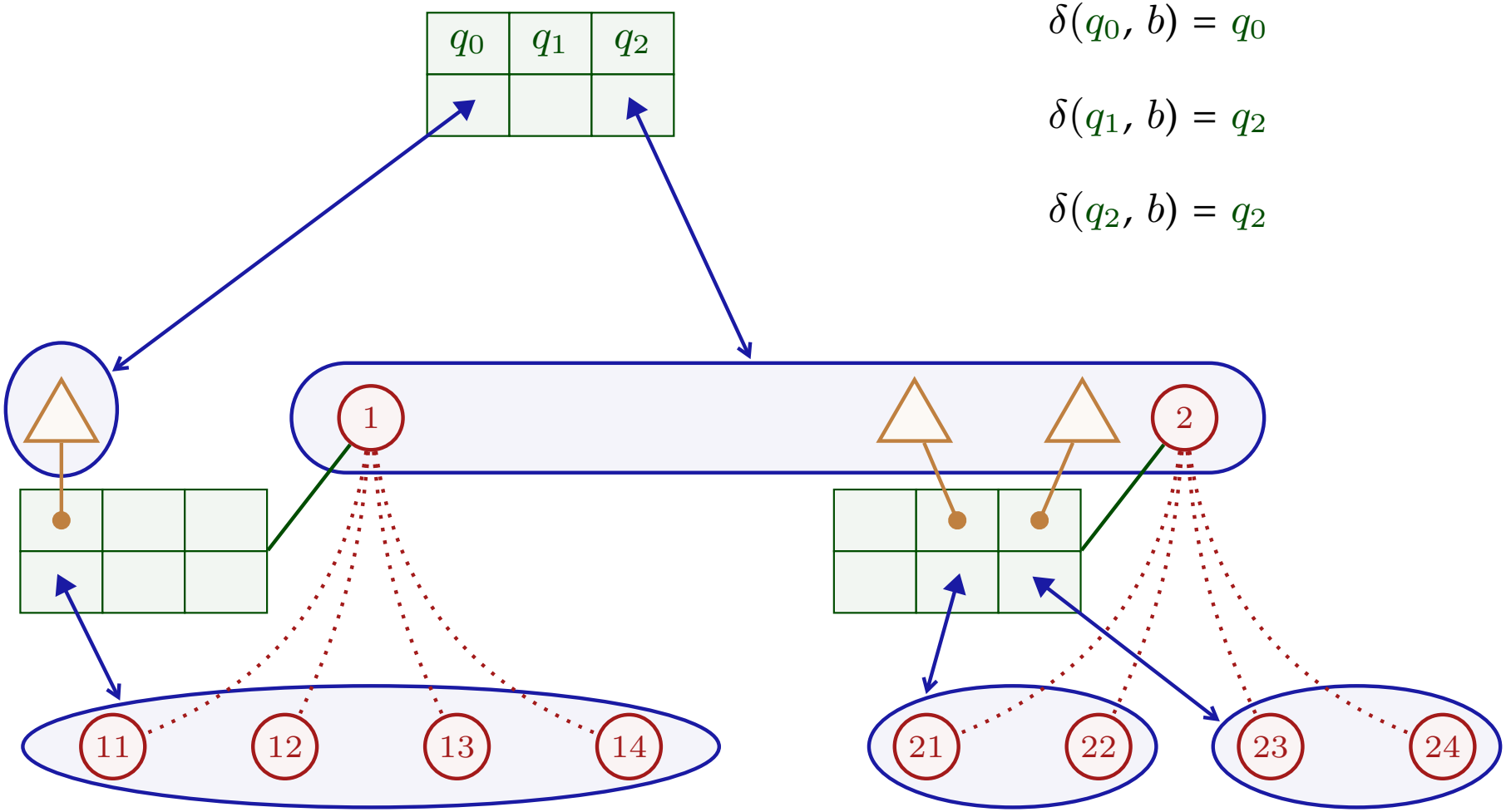
Trees: Dispatch b to All Objects



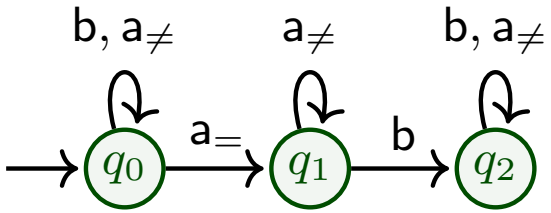
$\delta(q_0, b) = q_0$

$\delta(q_1, b) = q_2$

$\delta(q_2, b) = q_2$



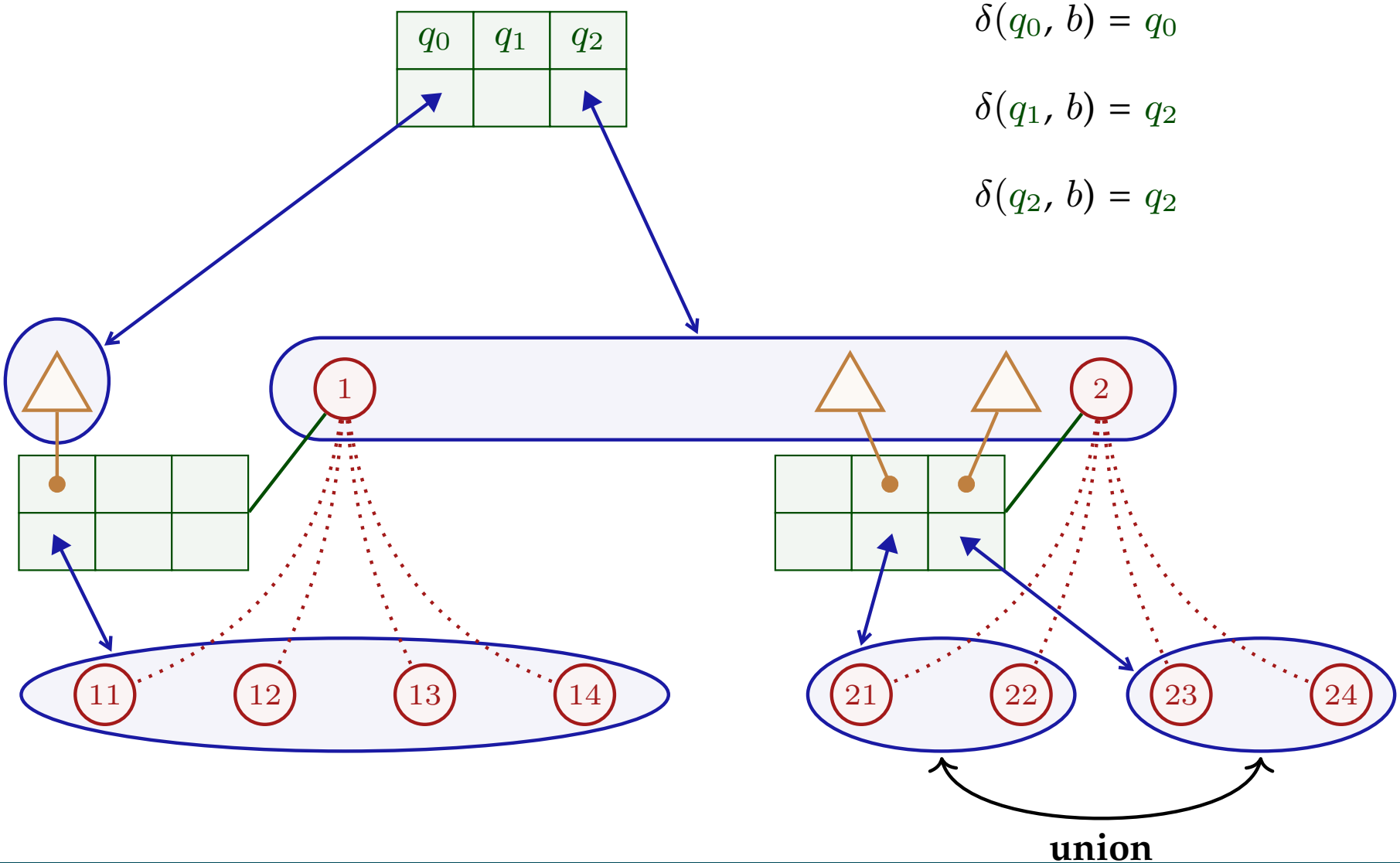
Trees: Pull Down Changes



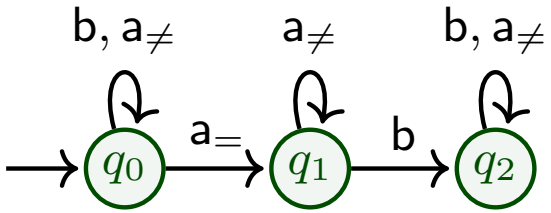
$\delta(q_0, b) = q_0$

$\delta(q_1, b) = q_2$

$\delta(q_2, b) = q_2$



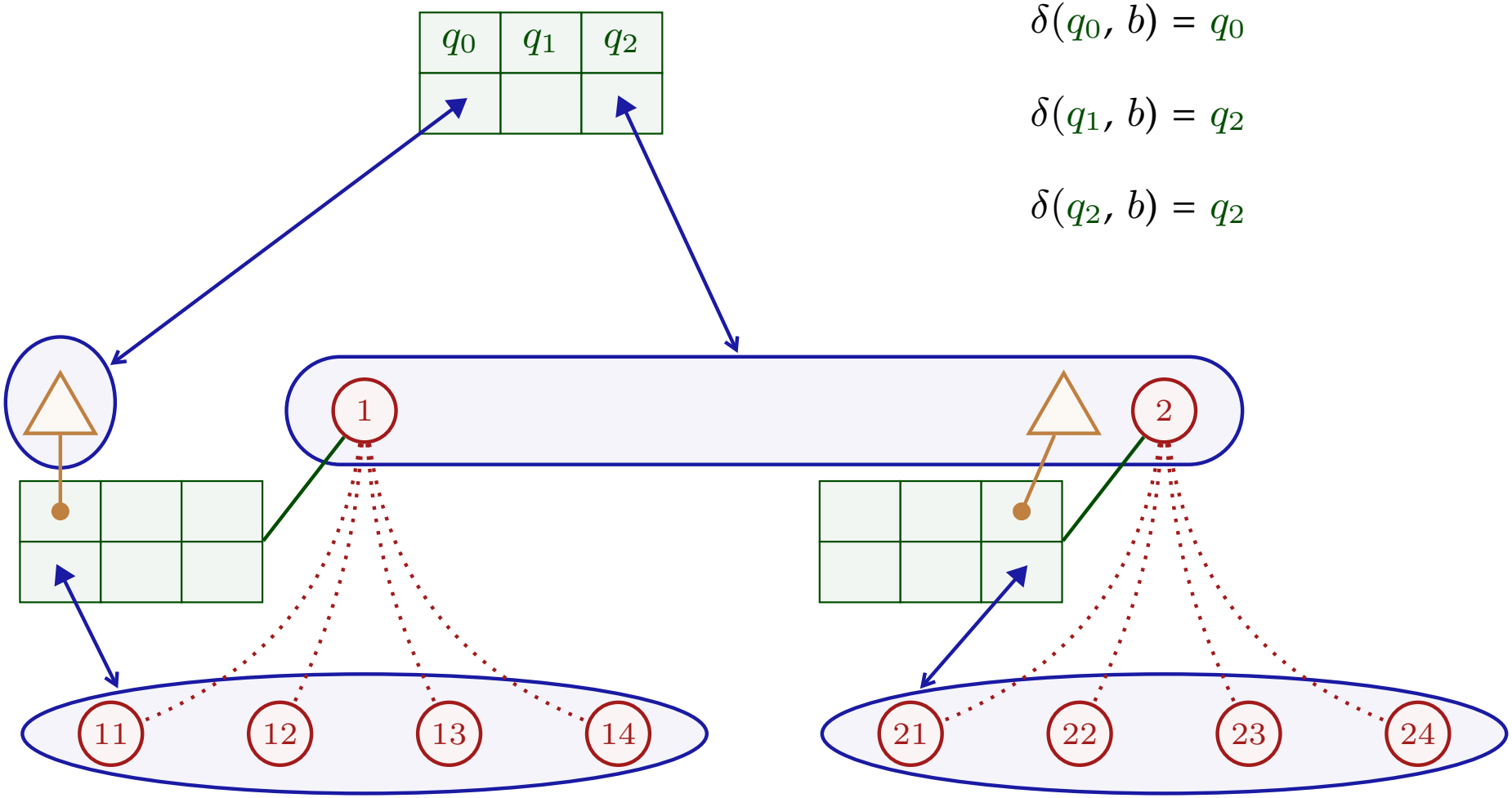
Trees: Pull Down Changes



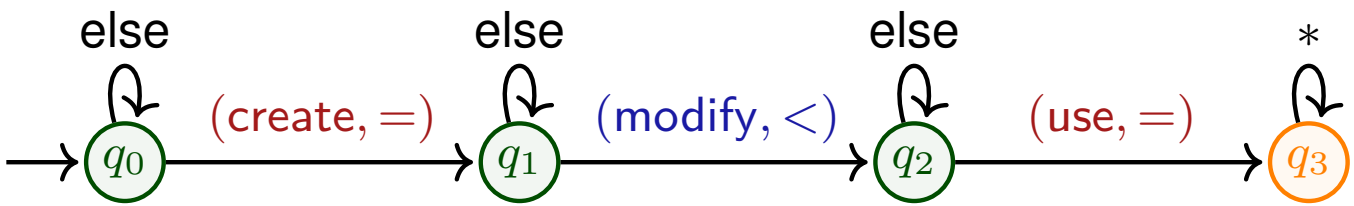
$\delta(q_0, b) = q_0$

$\delta(q_1, b) = q_2$

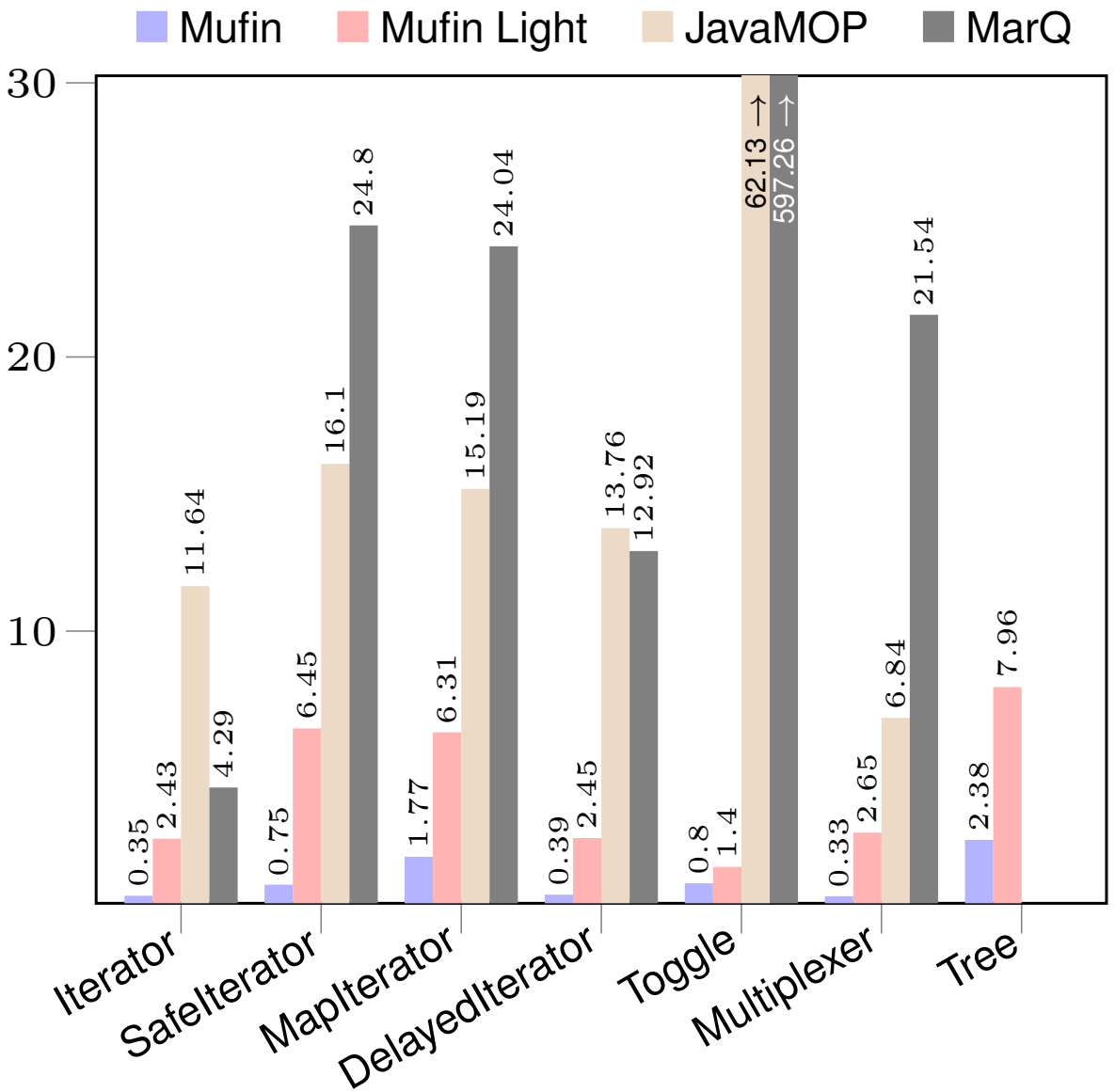
$\delta(q_2, b) = q_2$



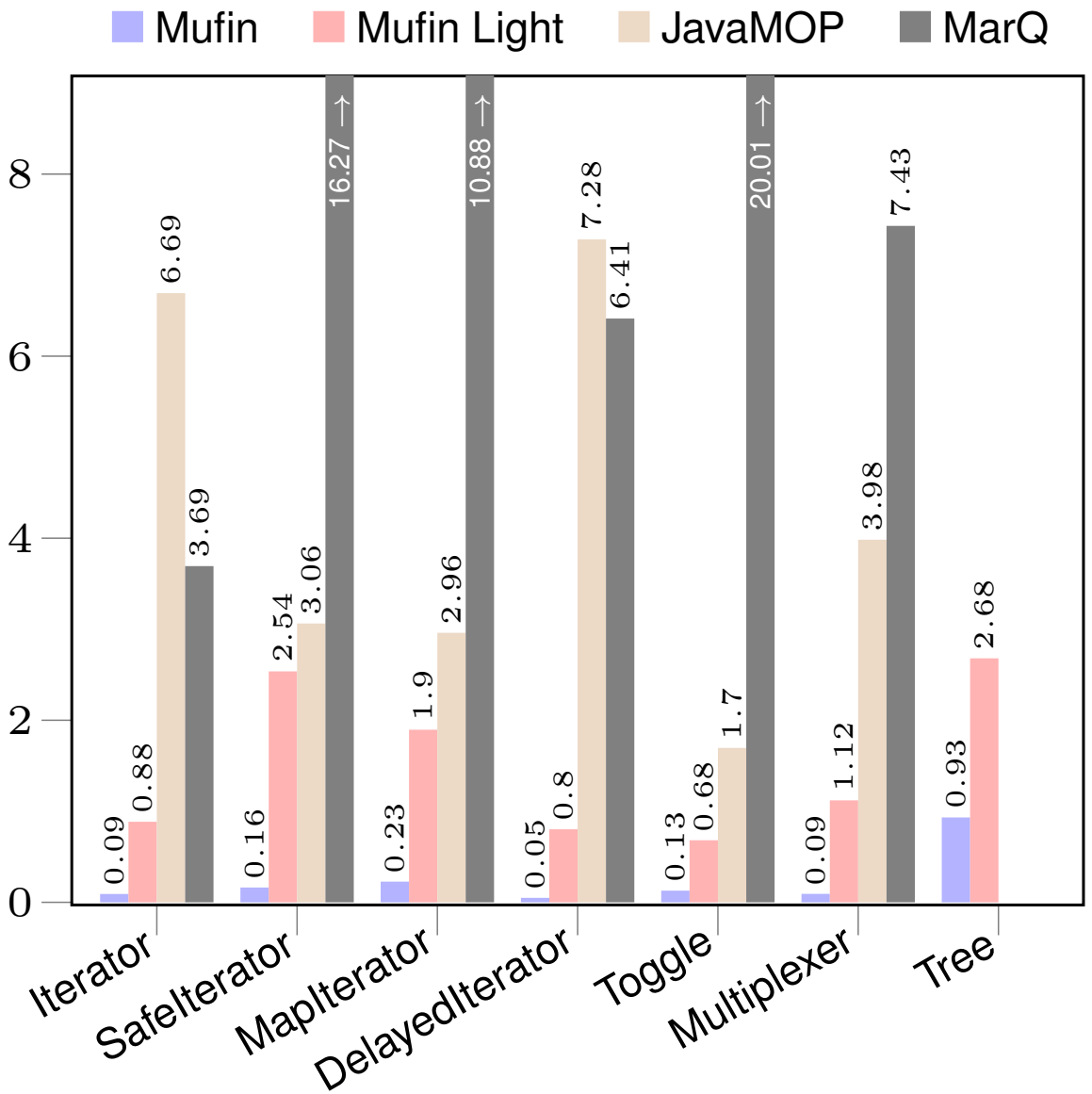
Example



Benchmarks: Relative Time Overhead



Benchmarks: Relative Memory Overhead



Logical Characterisation

$$\mathrm{G}(\text{create} \rightarrow \mathrm{G}(\text{modify} \rightarrow \neg\, \mathrm{F}\, \text{use}))$$

Iterator perspective

- If you create me and then
- modify my collection then
- don't use me any more.

Logical Characterisation

$$\mathrm{G}(\mathsf{create} \rightarrow \mathrm{G}(\mathsf{modify} \rightarrow \neg\, \mathrm{F}\, \mathsf{use}))$$

Iterator perspective

- If you create me and then
- modify my collection then
- don't use me any more.

Fragment of first-order LTL

$$\forall me.\, \mathrm{G}(\mathsf{create} \wedge \mathsf{id} = me \rightarrow \mathrm{G}(\mathsf{modify} \wedge \mathsf{id} < me \rightarrow \neg\, \mathrm{F}\, \mathsf{use} \wedge \mathsf{id} = me))$$

(Models: Sequences of FO structures))

Conclusion

- Monitoring of object-oriented systems
- Individual behaviour of objects, hierarchical dependencies
- Formal model and logical characterization

Conclusion

- Union-find as alternative to hash tables
- Execution time of one monitoring step is
  - guaranteed: logarithmic
  - amortised: almost constant

  in the number of observed objects
- Benchmarks show that Mufin[1] outperforms JavaMOP[2] and MarQ[3]

[1] `http://www.isp.uni-luebeck.de/mufin`

[2] P. O. Meredith, D. Jin, D. Griffith, F. Chen, and G. Rosu.
An overview of the MOP runtime verification framework. (STTT '12)

[3] G. Reger, H. C. Cruz, and D. E. Rydeheard.
MarQ: Monitoring at runtime with QEA. (TACAS '15)