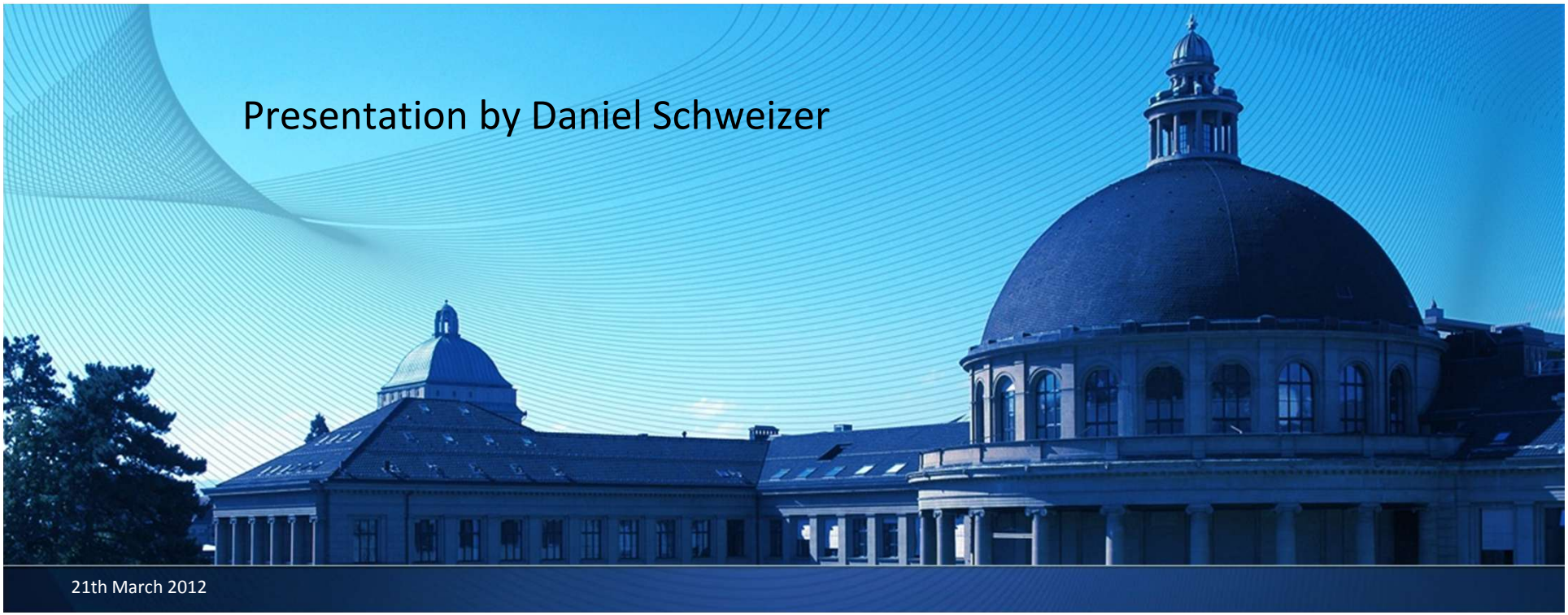
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Reachability Testing of Semaphore-based Programs

Yu Lei, Univ. of Texas at Arlington
Richard Carver, George Mason Univ. Fairfax

International Computer Software and Applications Conference, 2004



An example program

S1.count = 2; S2.count = 1		
T1	T2	T3
S1.down -- T1 in critical section print(“1“) S1.up	S1.down S2.down -- T2 in critical section print(“2“) S2.up S1.up	S1.down -- T3 in critical section print(“3“) S1.up S2.down -- T3 in critical section print(“3“) S2.up

possible outputs:

1233, 1332, 2331, 3231, ...

An example program

- possible outputs (determined statically):

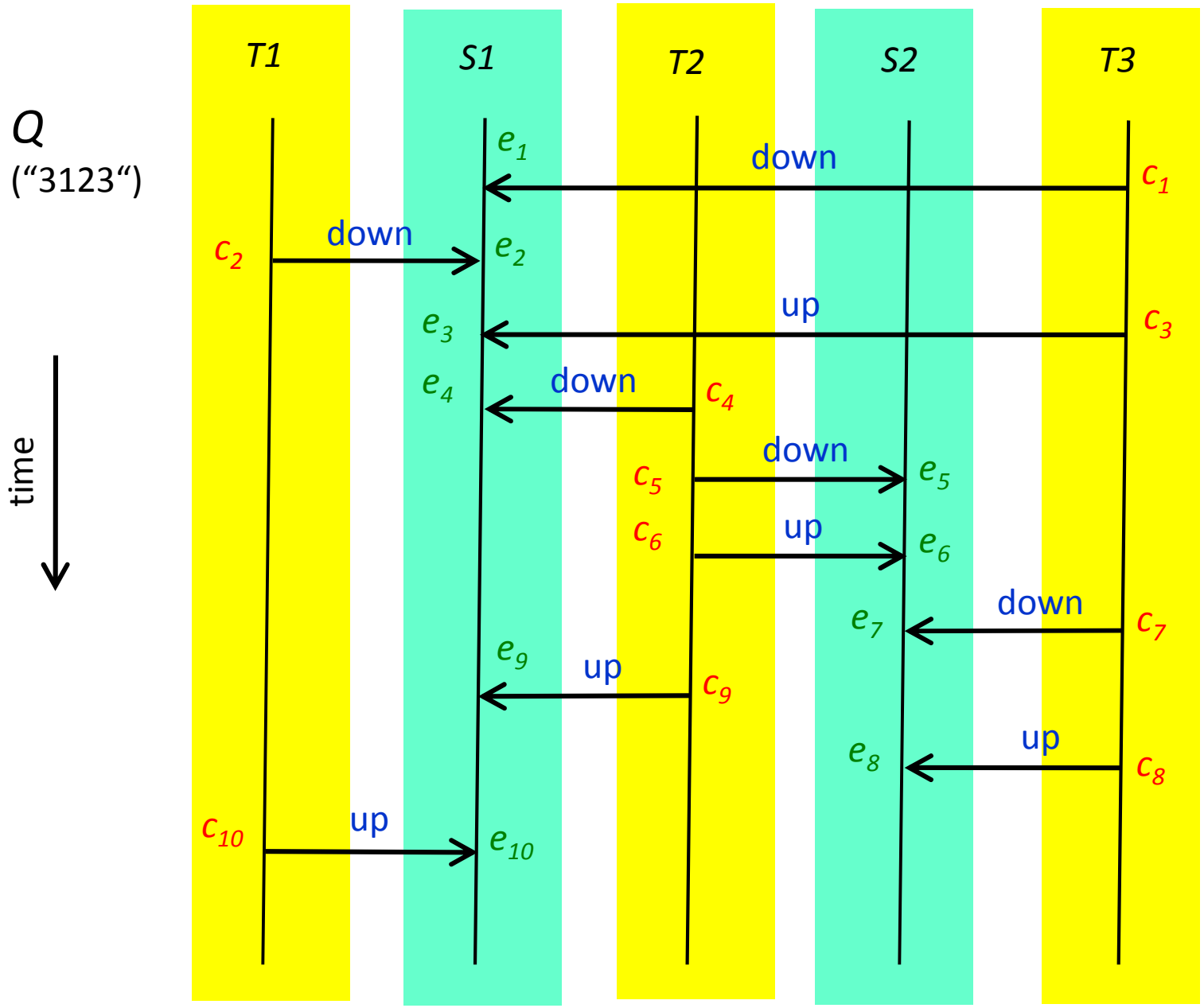
  1233, 1332, 2331, 3231, ...

- actual outputs (20 test runs):
  - 1332 3 x
  - 1233 17 x
  - but: 2331 0 x

- Problem: We did not observe all feasible executions when testing!

Definitions

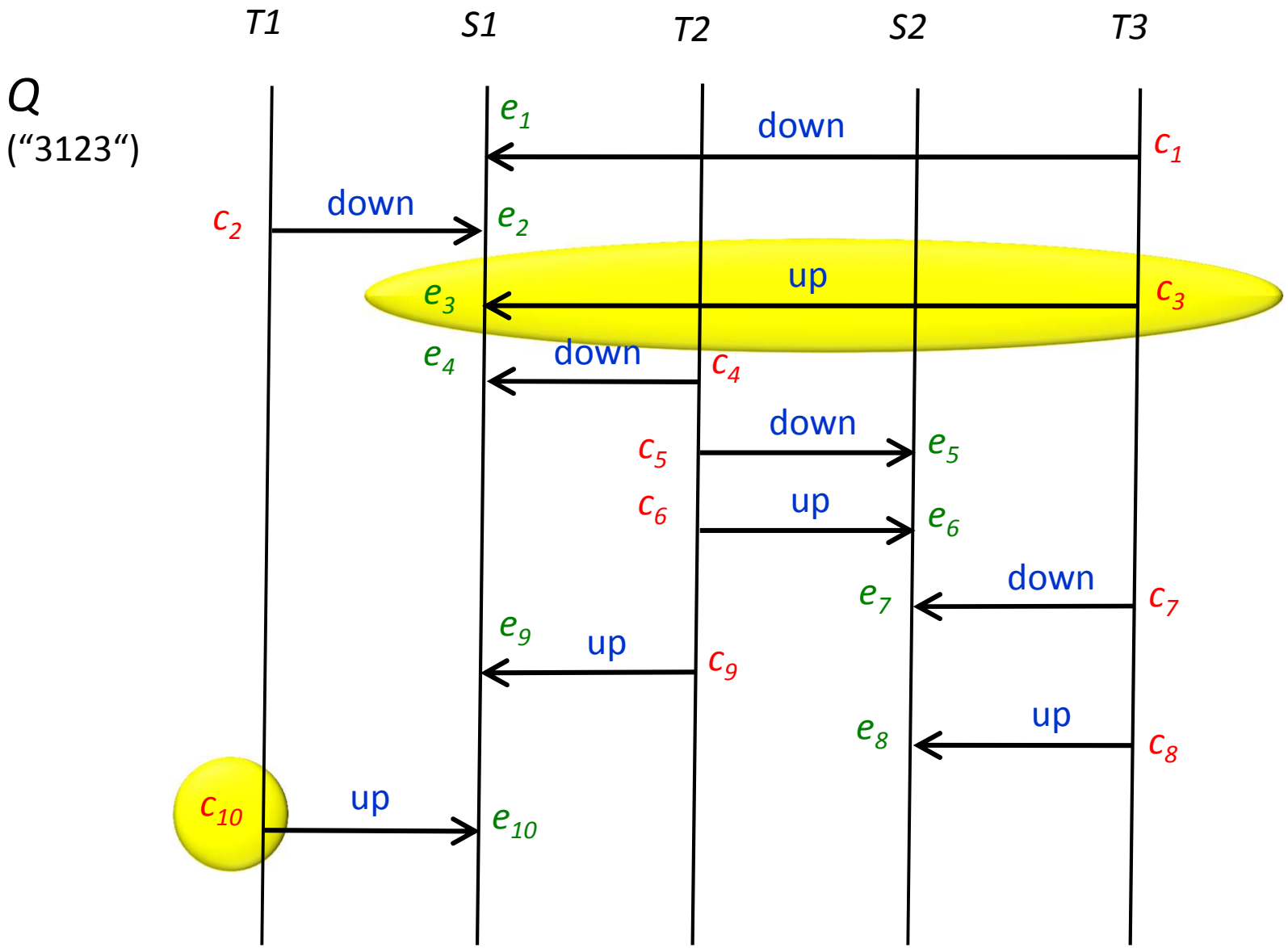
- when a thread *T* calls down or up on a semaphore *S*, a ***call event*** is performed by *T*
- when a down or up operation on a semaphore *S* is completed, a ***completion event*** occurs on *S*
- an execution of a semaphore-based program is characterized by the sequence of *call* and *completion events* it exercises, called the ***CC-sequence*** of the execution
- if the operation of a call event *c* is completed by a completion event *e*, then *c* and *e* form a ***completion pair*** <*c, e*>

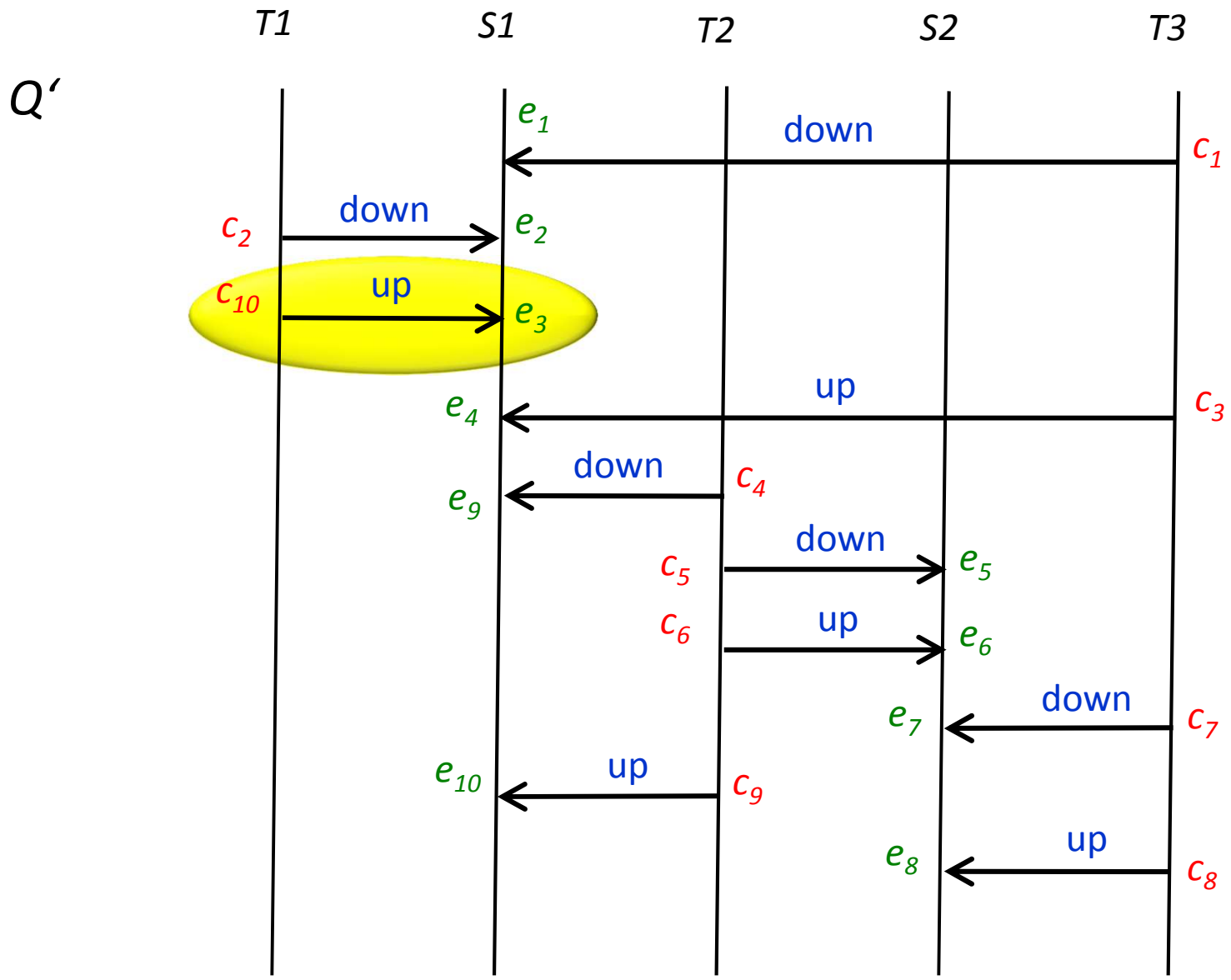

Q
("3123")
time
T1
S1
T2
S2
T3
e_1
down
c_1
c_2
down
e_2
e_3
up
c_3
e_4
down
c_4
down
c_5
e_5
c_6
up
e_6
down
e_7
c_7
e_9
up
c_9
up
e_8
c_8
c_{10}
up
e_{10}

Race

- *Q*: a CC-sequence exercised by an execution of a semaphore-based program CP
- *c, c'*: call events in *Q* ($c \neq c'$)
- *e*: completion event in *Q*
- <*c, e*> is a completion pair

- there is a ***race*** between *c'* and <*c, e*> in *Q* if *c'* and *e* can form a completion pair in another execution *Q'* of *CP*, provided that all the events that happen before *c'* or *e* in *Q* are replayed in *Q'*

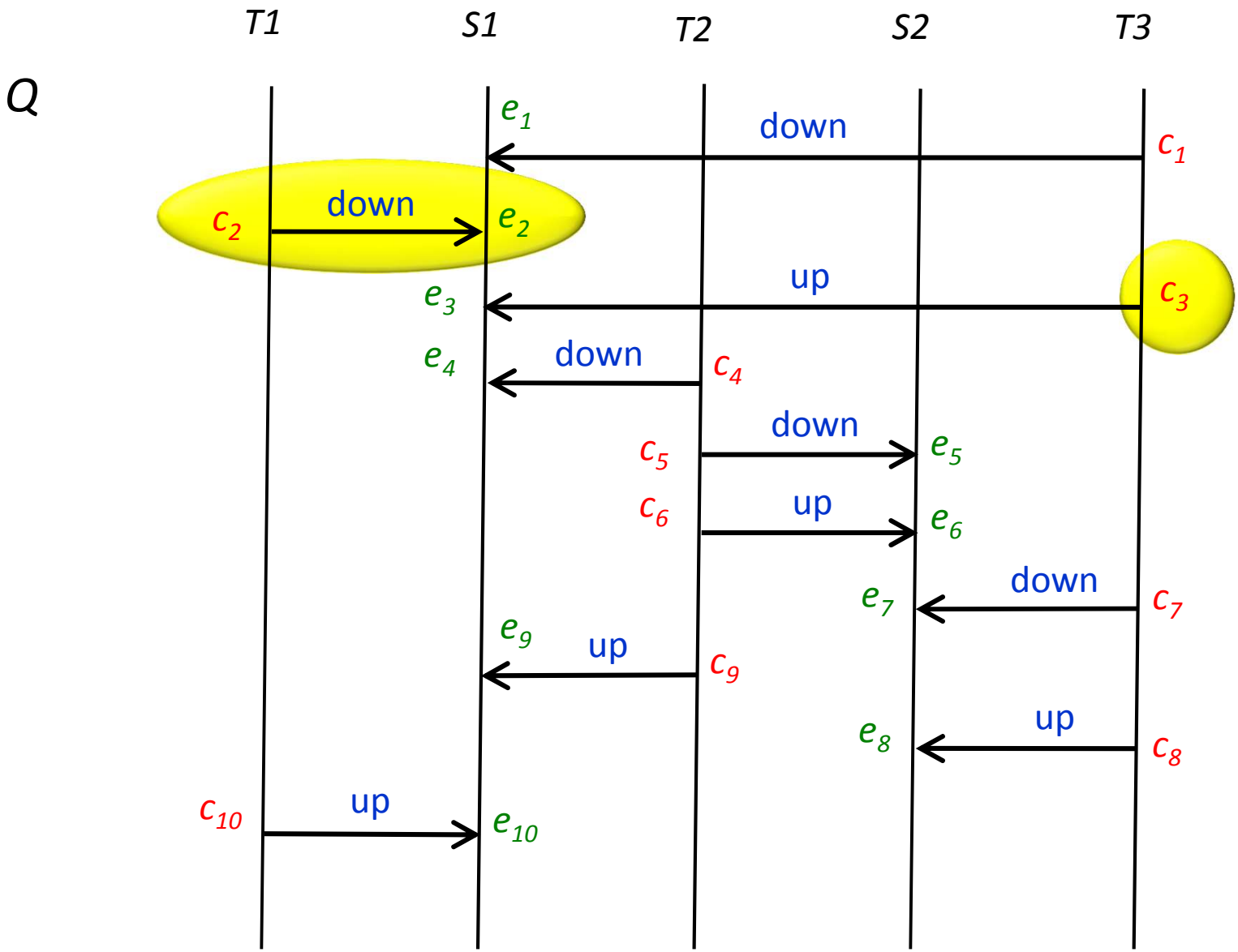

Q
("3123")
T1
S1
T2
S2
T3
e1
down
c1
c2
down
e2
e3
up
c3
e4
down
c4
down
c5
e5
c6
up
e6
down
e7
c7
e9
up
c9
e8
up
c8
c10
up
e10

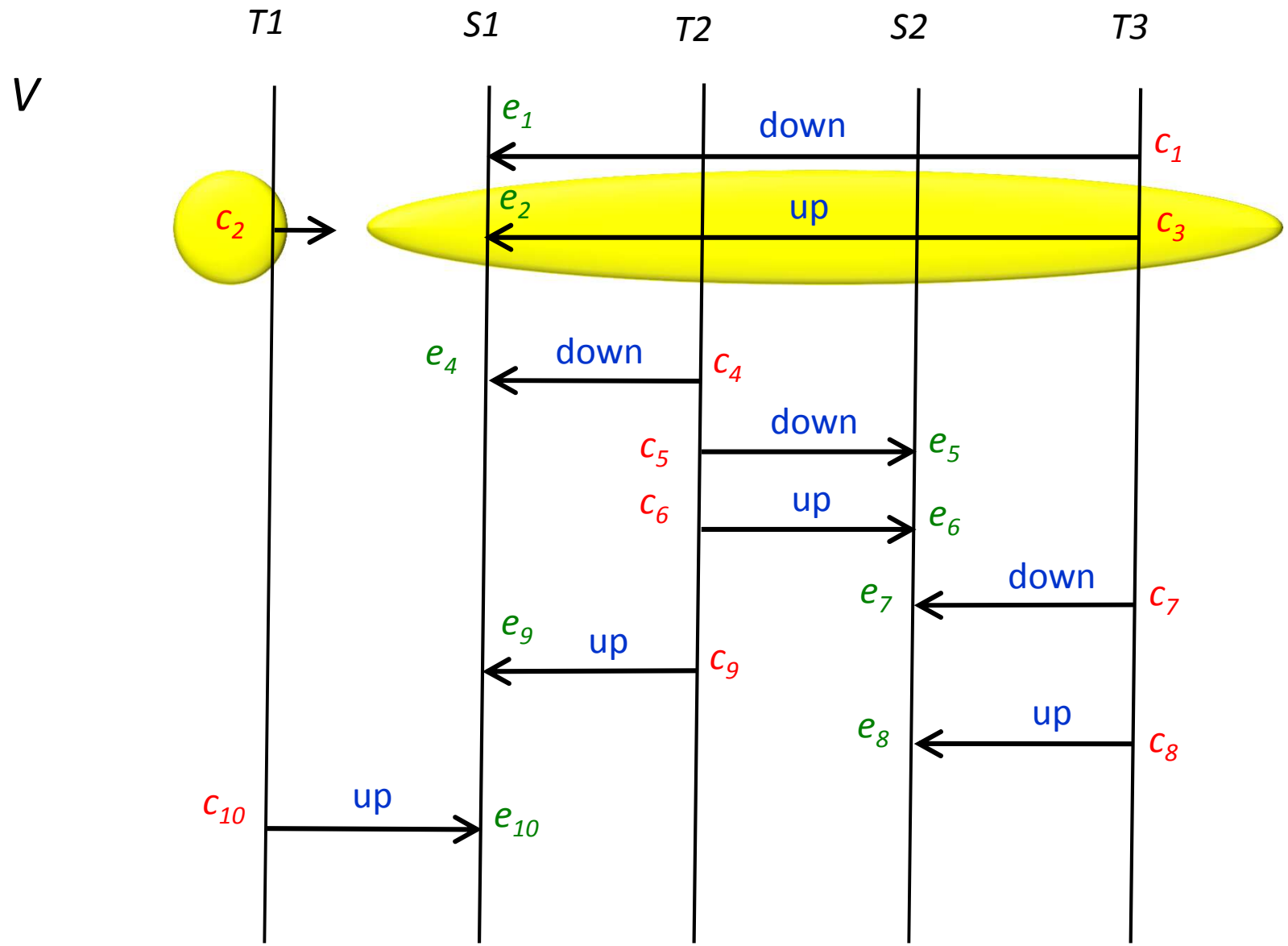

Q'
T1
S1
T2
S2
T3
e1
down
c1
c2
down
e2
c10
up
e3
e4
up
c3
down
c4
e9
down
c5
e5
c6
up
e6
down
e7
c7
e10
up
c9
e8
up
c8

Race variant

- *Q*: a CC-sequence

- a ***race variant*** *V* of *Q* is a CC-sequence that is derived by changing the call partner of one or more completion events in *Q*, with the following constraints:
- if we change the call partner of a completion event *e*,
    1) there must be a race between the new call partner of *e* and the completion pair <*c*, *e*> in *Q*
    2) we must remove all events that happen after *e*


Q
T1
S1
T2
S2
T3
e_1
down
c_1
c_2
down
e_2
up
c_3
e_3
e_4
down
c_4
down
c_5
e_5
c_6
up
e_6
down
e_7
c_7
e_9
up
c_9
up
e_8
c_8
c_{10}
up
e_{10}


V
T1
S1
T2
S2
T3
e1
down
c1
c2
e2
up
c3
e4
down
c4
down
c5
e5
c6
up
e6
down
e7
c7
e9
up
c9
up
e8
c8
c10
up
e10

Reachability-Test

Reachability-Test (*CP*: a semaphore-based program)

do

variants = ∅

collect a CC-sequence Q_0 by executing *CP* non-deterministically

derive variants(Q_0) -- the race variants of Q_0

variants = variants ∪ variants(Q_0)

while variants not empty **loop**

withdraw a variant *V* from variants

collect a CC-sequence *Q* using prefix-based testing with *V*

derive variants(*Q*) -- the race variants of *Q*

variants = variants ∪ variants(*Q*)

end

end

Results (1)

" **Theorem:** Let CP be a semaphore-based program. Assume that every execution of CP with input I terminates. Then, algorithm *Reachability-Test* terminates, and executes all feasible CC-sequences of CP with input I. "

- no proof

Results (2)

Program	Configuration	# of Seqs
BB	3P + 3C + 2S	324
BB	2P + 2C + 2S	12
RW	2R + 2W	608
RW	2R + 3W	12816
RW	3R + 2W	21744
DP		30
		0
		300
		32

Open questions:

- comparison with other methods?
- performance?
- what about larger programs?

RW: readers/ rs buted mutual exclusion algorithm

Discussion