Detecting STRONGLY CONNECTED COMPONENTS in Large *Distributed* State Spaces

Simona Orzan Jaco van de Pol

CWI, Amsterdam and *TUE, Eindhoven*

Overview

SCC detection

- Motivation, context
- Results
- Algorithms impression
- Conclusions and (still) future work

Security

- An unusual application of the CADP model checker

Motivation and context

Distributed enumerative verification

- state space generation, reduction tools
- distributed message passing, LAM/MPI
- no. processors $<<$ size of the LTS
- algorithms exploit state space characteristics: small depth, small diameter

SCC of a directed graph = maximal subgraph in which every vertex is reachable from every other vertex

Distributed SCC detection

- interesting as preprocessing step for branching bisimulation minimization
- very efficient sequential solution exists, based on DFS

Results: some numbers

state space	size of the state space size(S) (10^6)	size(T) (10^6)	τs (10^6)	atomic SCCs (% of S)	size of largest SCC	Runtimes (s) seq.	CE1	CE2	Runtimes BB (s) after CE	original
screen.706	1.2	2.7	2.4	6.6	38	4.4	7.7	12.4	9.6	106.8
lift5	2.1	8.7	3.8	98.9	165	17.8	8.6	14.7	79	86
vasy_4338	4.3	15.6	3.1	100.0	1	31.74	6	6	82	82
vasy_8082	8.0	42.9	2.5	100.0	1	103.9	11	11	27	27
screen.801	20.7	49.7	44.9	4.3	50	145.7	172	112.5	43.6	2819.2
swp_piggy	9.6	53.4	30.9	10.5	45	197.3	125	237	122	341
cache	7.8	59.1	22.8	99.5	248	153.6	45	47	1331	1394
screen.1	29.9	72.3	65.9	1.2	50	-	210	121	36.7	180 000
lift6	33.9	165.3	74.1	99.8	486	-	160	305	930	1039

Figure 1: Some case studies: size, structure, reduction times

- CE1 optimized for speed, CE2 is constant in memory usage
- CE is not too time-expensive and might give important gains to BB reduction

Algorithms

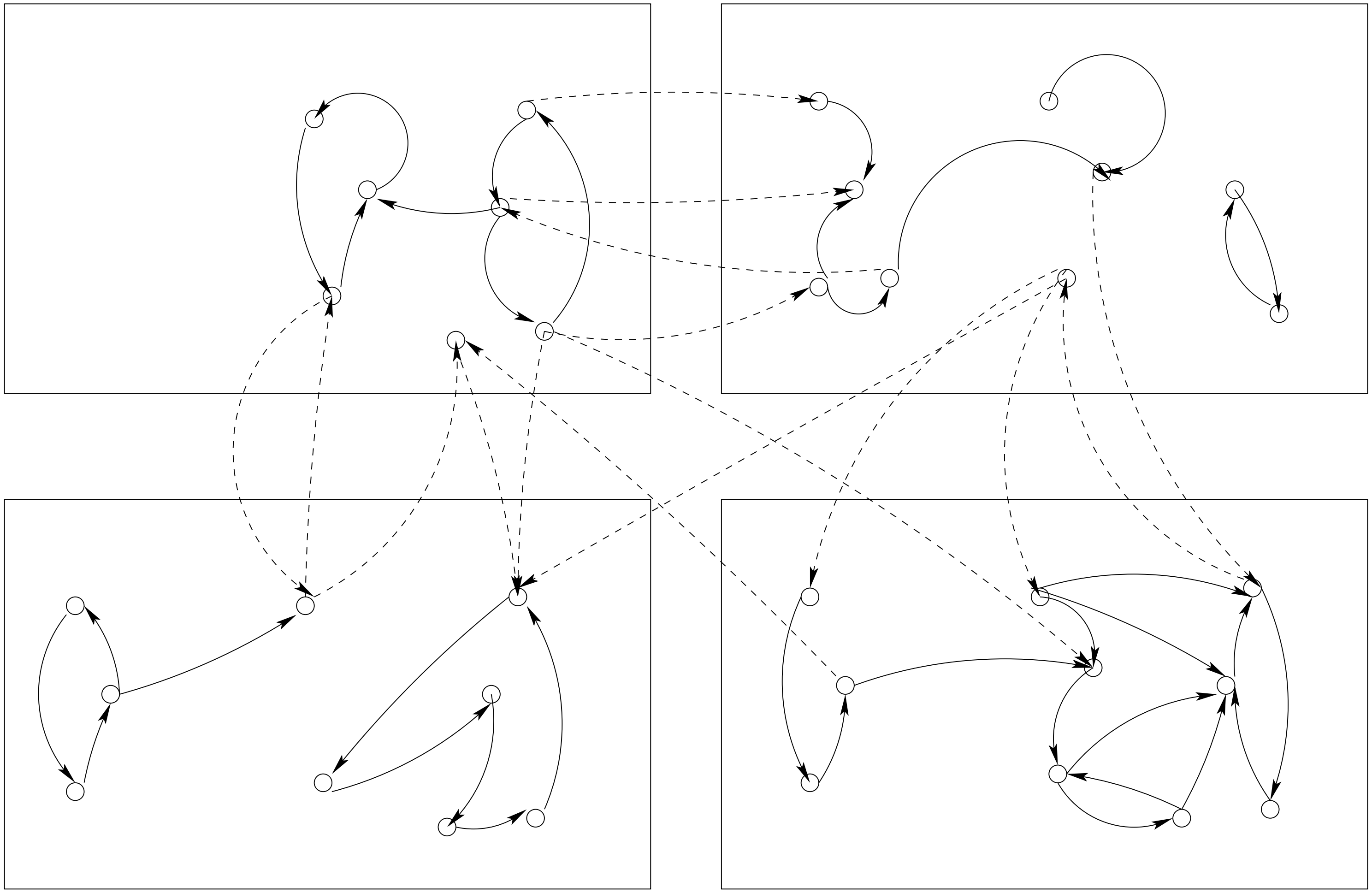
Basic observations

- goal: independent tasks, reduce the problem to smaller instances
- large computation rounds alternated with large communication steps
- BFS - no. of rounds bounded by the graph's depth

Main idea: graph transformation steps that ensure a decrease in size

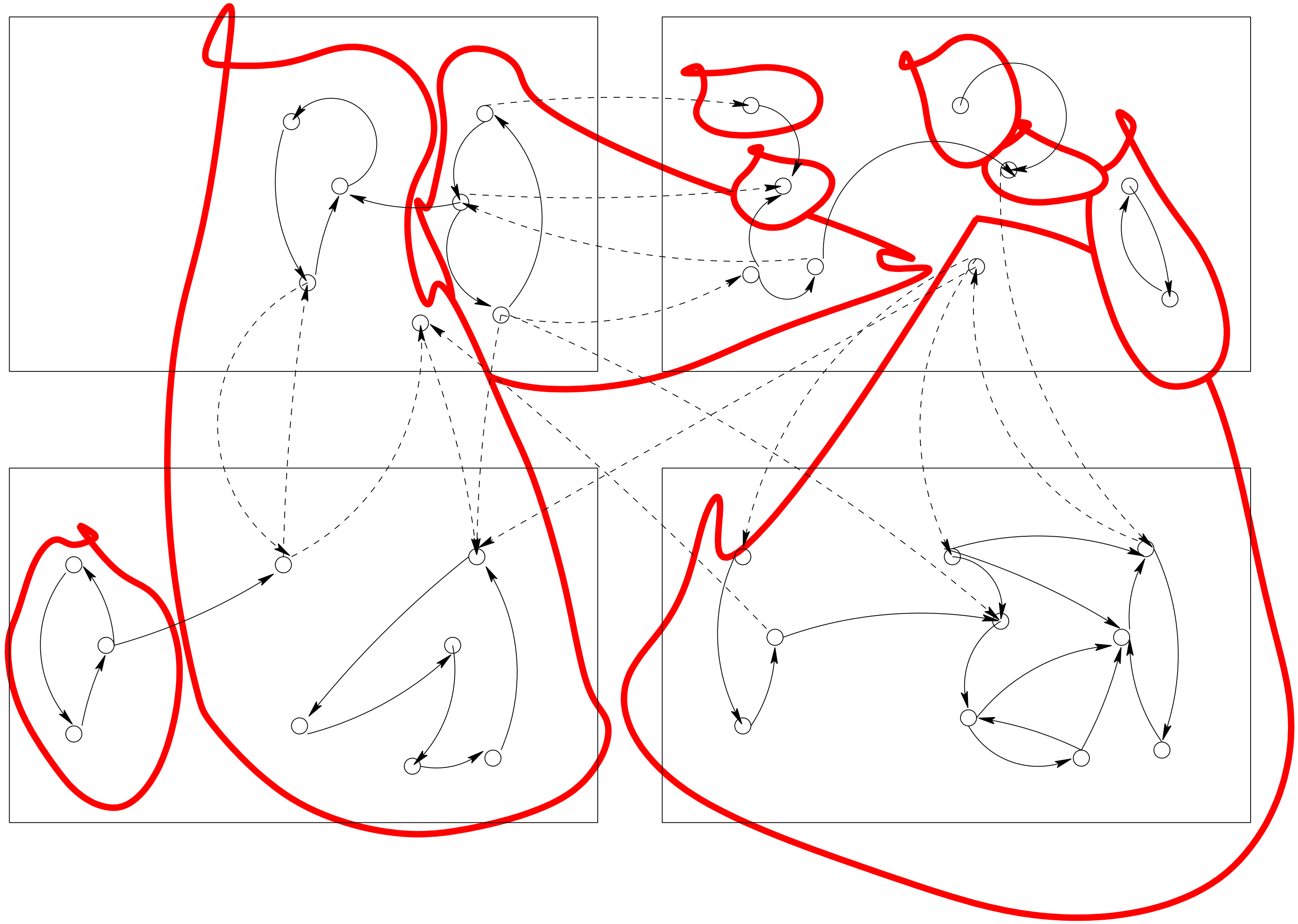
- marking atomic SCCs
- collapsing nodes that are definitely in the same SCC
- marking arcs that are definitely between two SCCs

Graph transformations



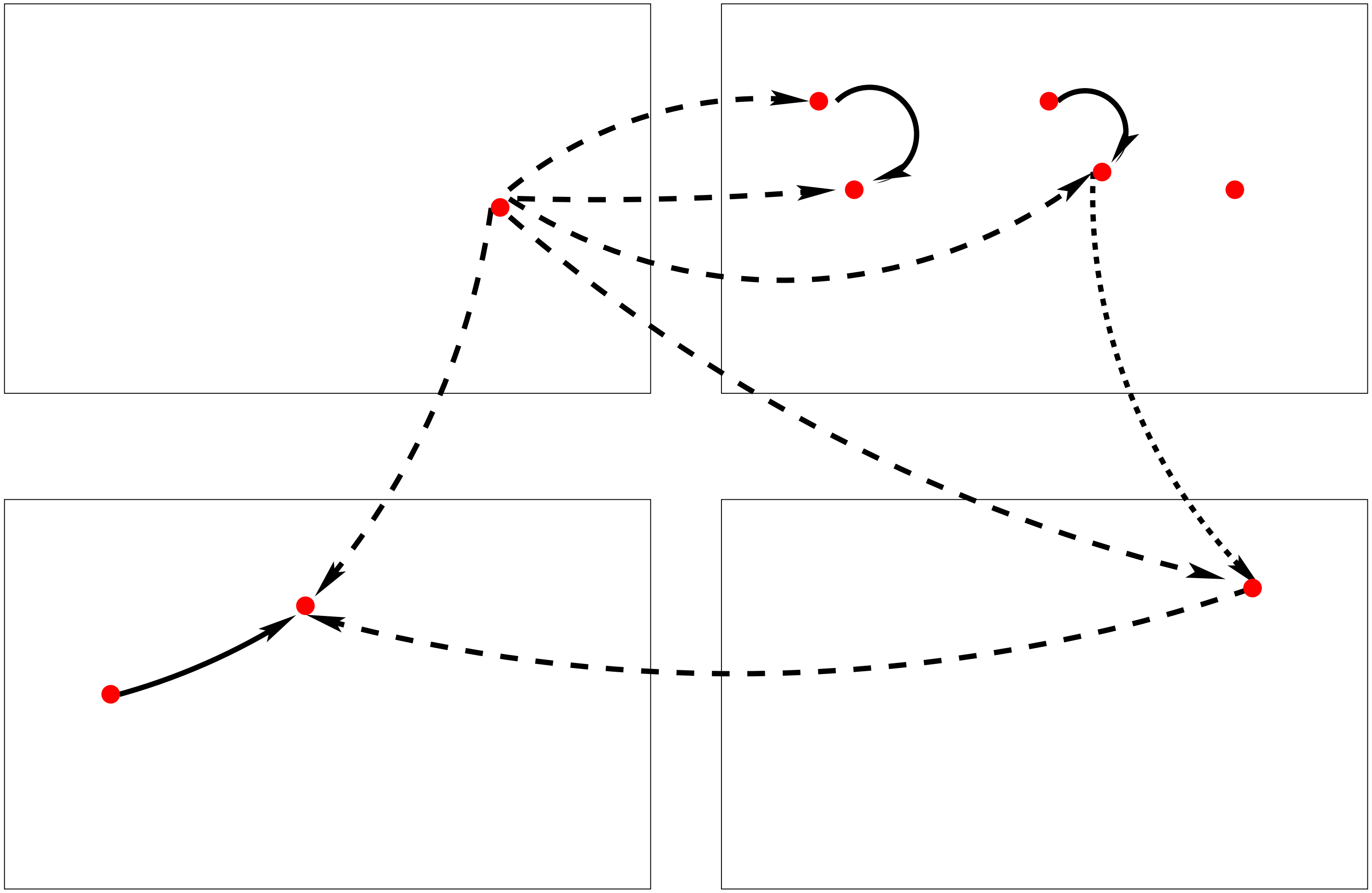
a large distributed graph

Graph transformations



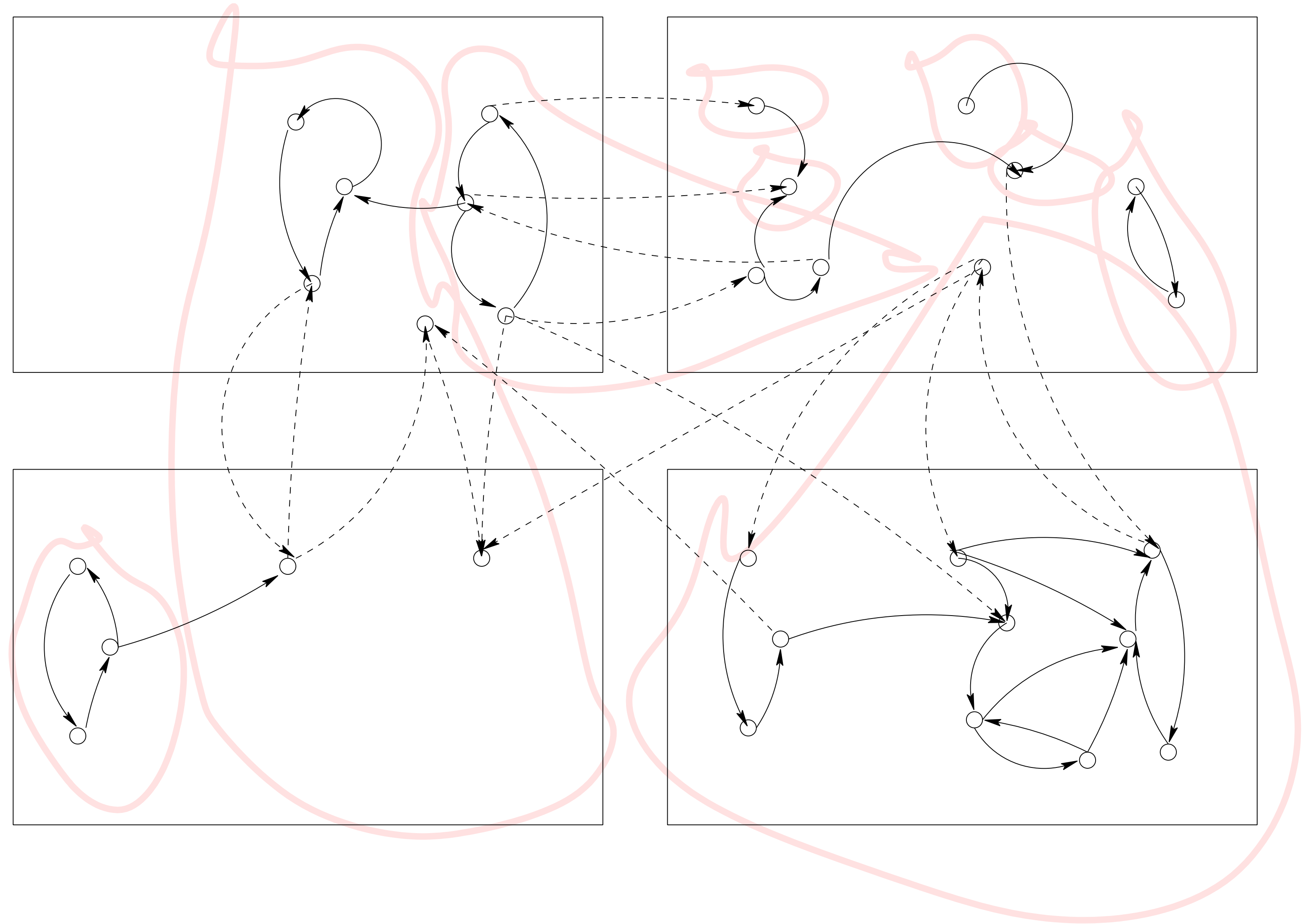
its strongly connected components

Graph transformations

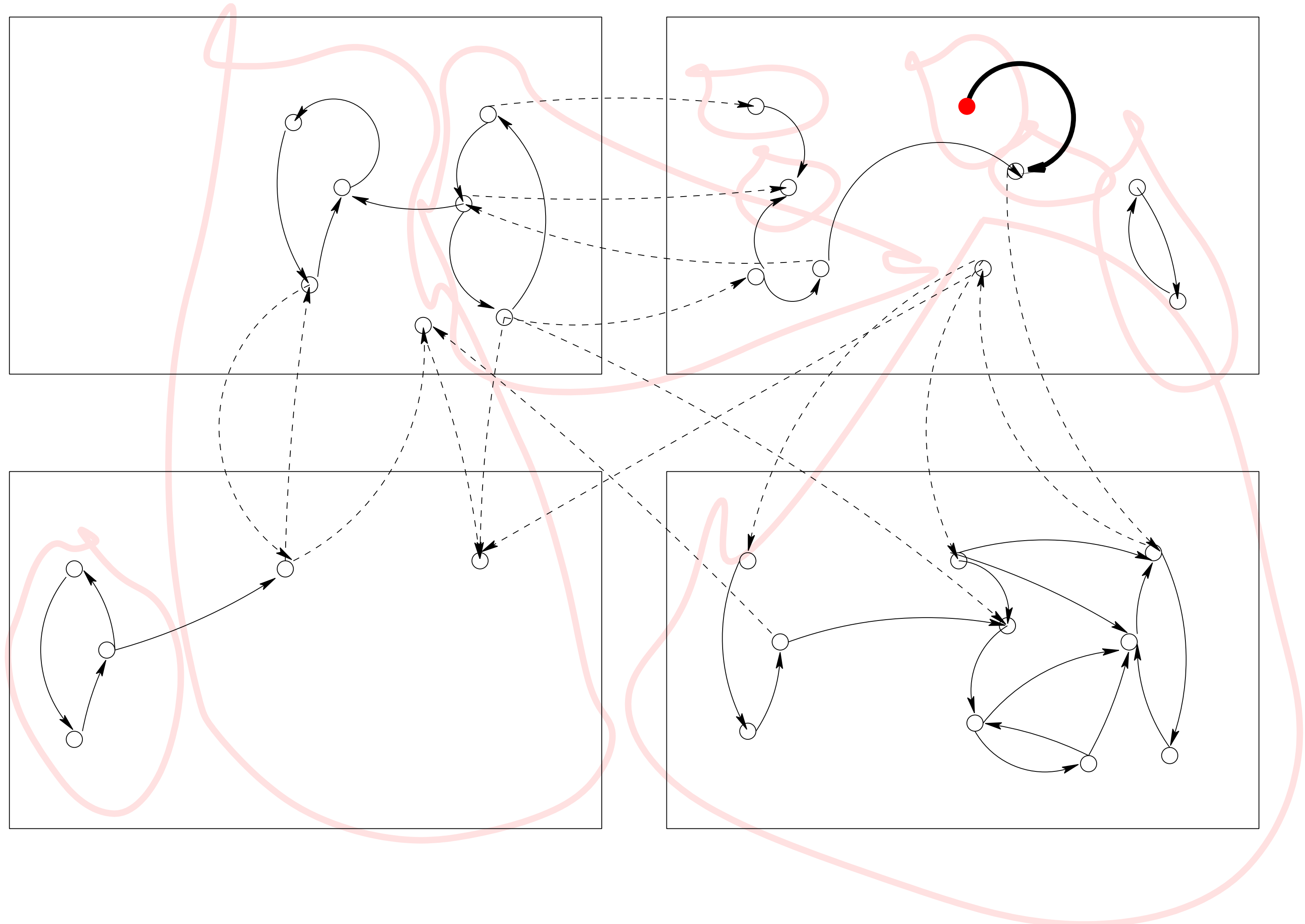


collapsed SCCs

Graph transformations

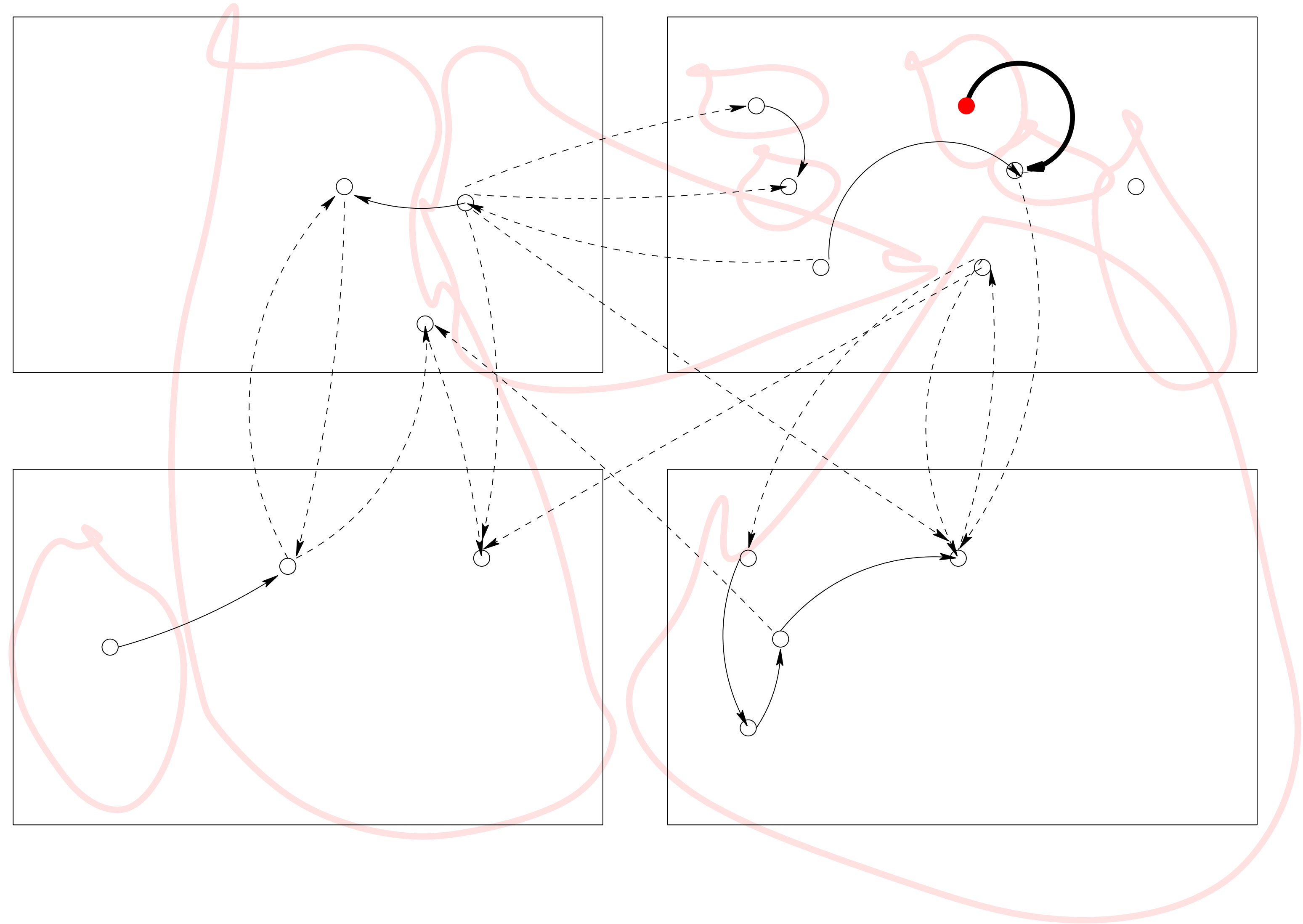


Graph transformations



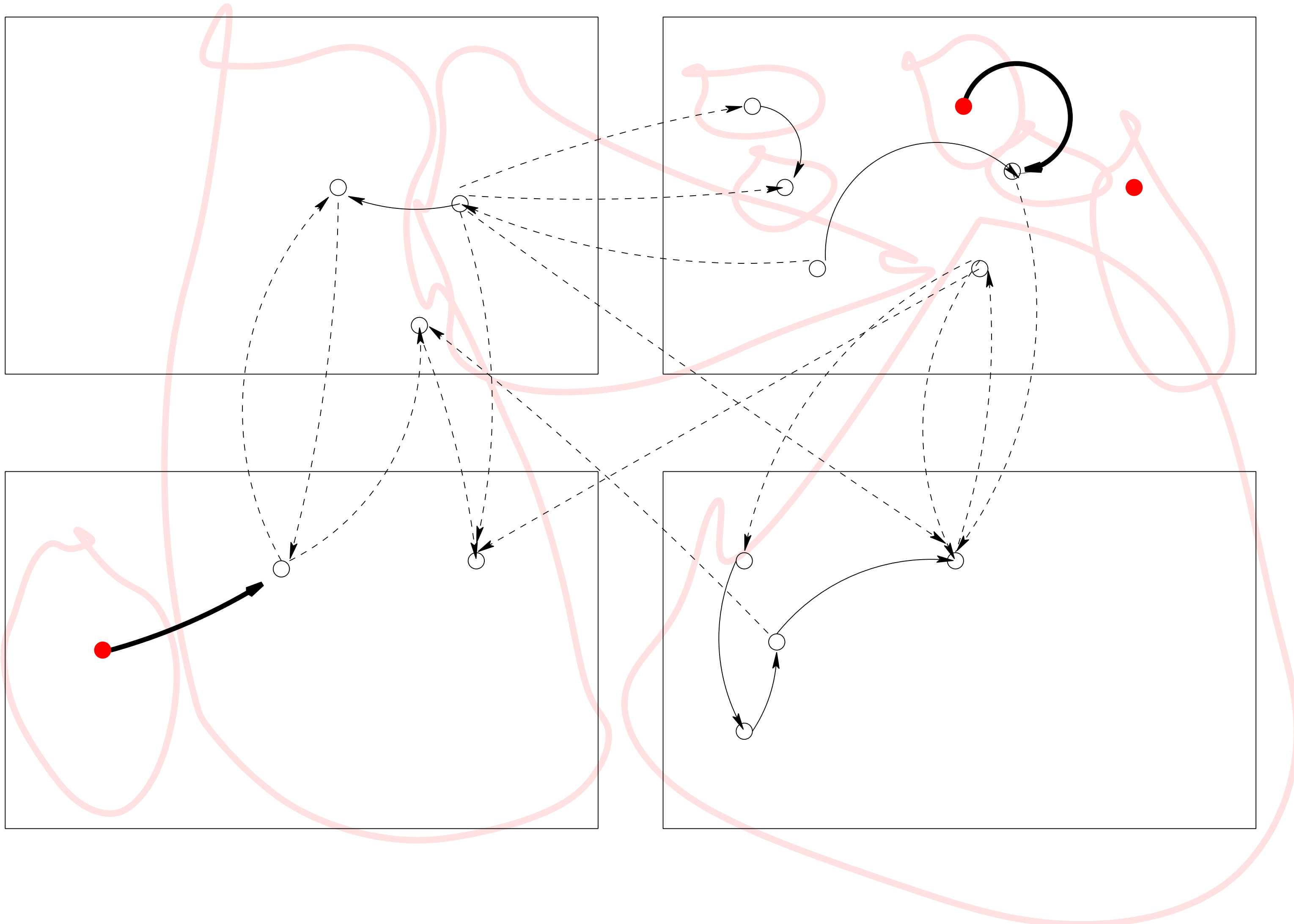
detecting atomic components

Graph transformations



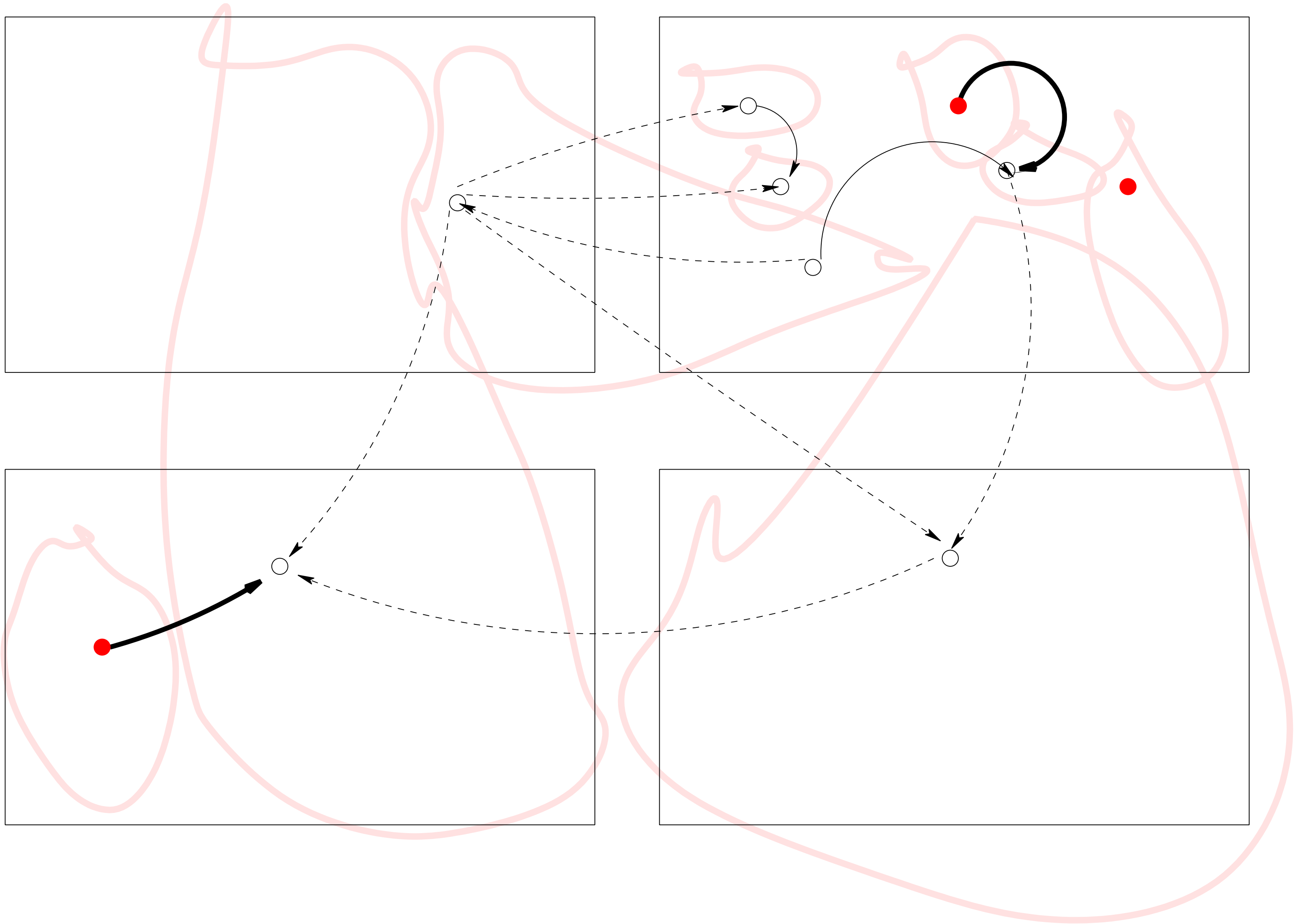
collapsing local components

Graph transformations

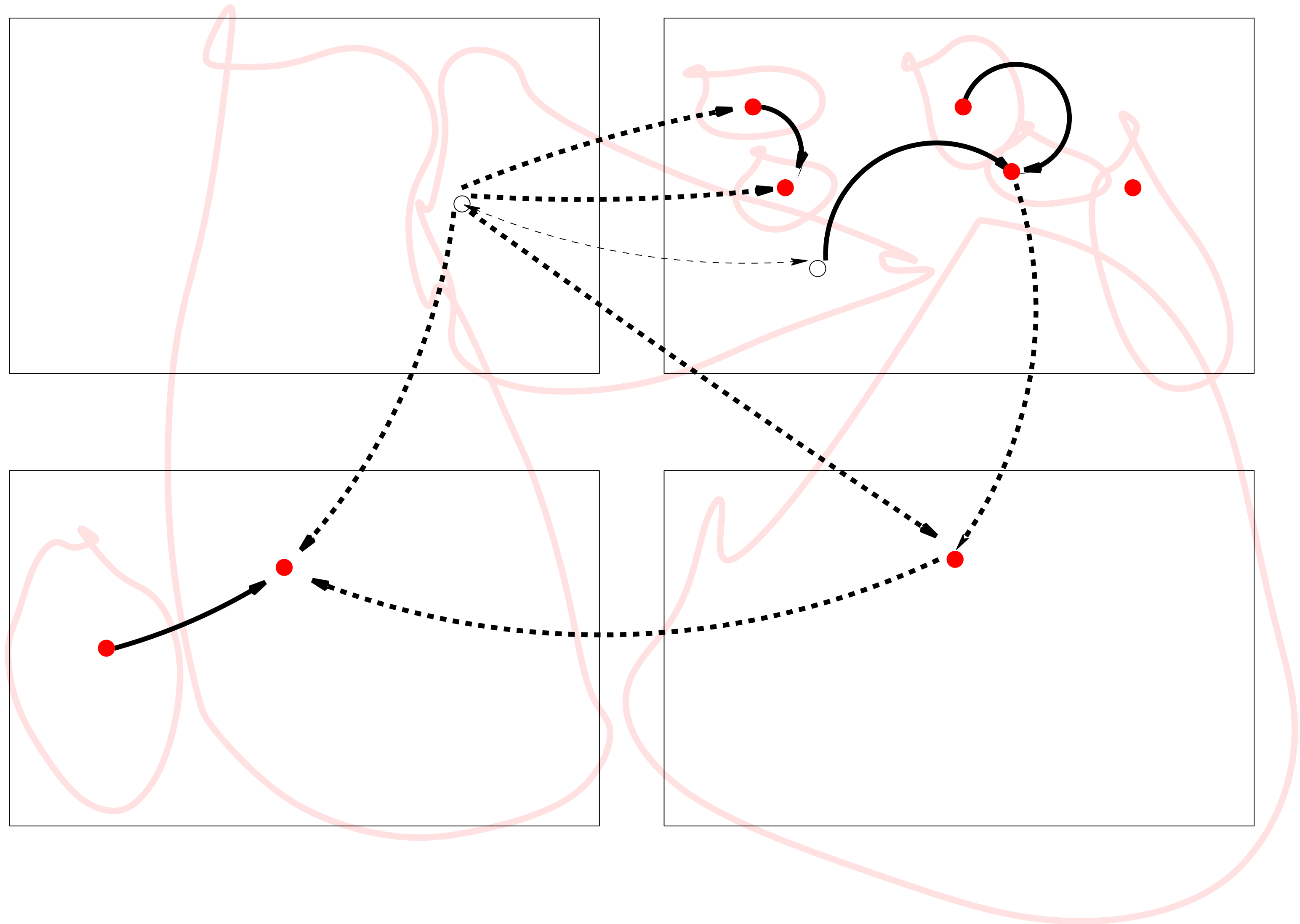


detecting atomic components

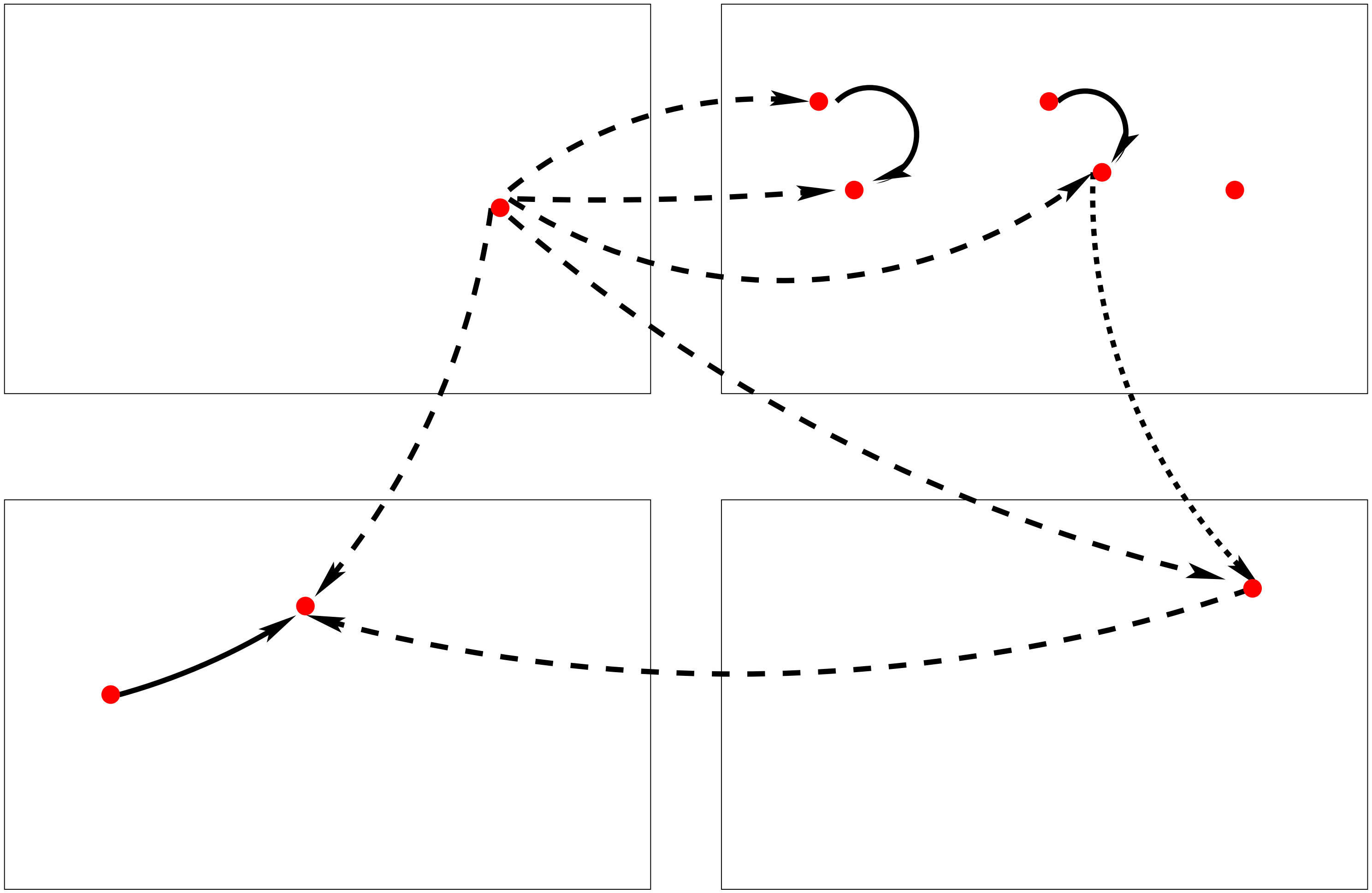
Graph transformations



Graph transformations



Graph transformations



The two algorithms

CE1

- make groups of workers: groups of size 1,2,4,8 ..
- every time collapse, in parallel, the respective subgraphs
- works well for small graphs and for dense ones

The two algorithms

CE2

- label every node with its unique id
- transfer the label of every node to its predecessors until the labeling is stable
- reverse the arrows and repeat the labeling
- if `label1(`x`) = label2(`x`) =` r then x is in r's SCC
- with both labels, arrows between nodes with different labeling are final
- repeat the labelings, collapsing of components and marking of final arrows until nothing left

Conclusions and future work

- a distributed tool for detection of SCCs
- relies on BFS traversals
- applications?

- optimize the branching bisimulation reduction algorithm
- combine the groups method with the labeling method
- use better partition functions (now: random)

An unusual application of the CADP model checker

Epistemic models: possible worlds, real worlds, indistinguishability relations

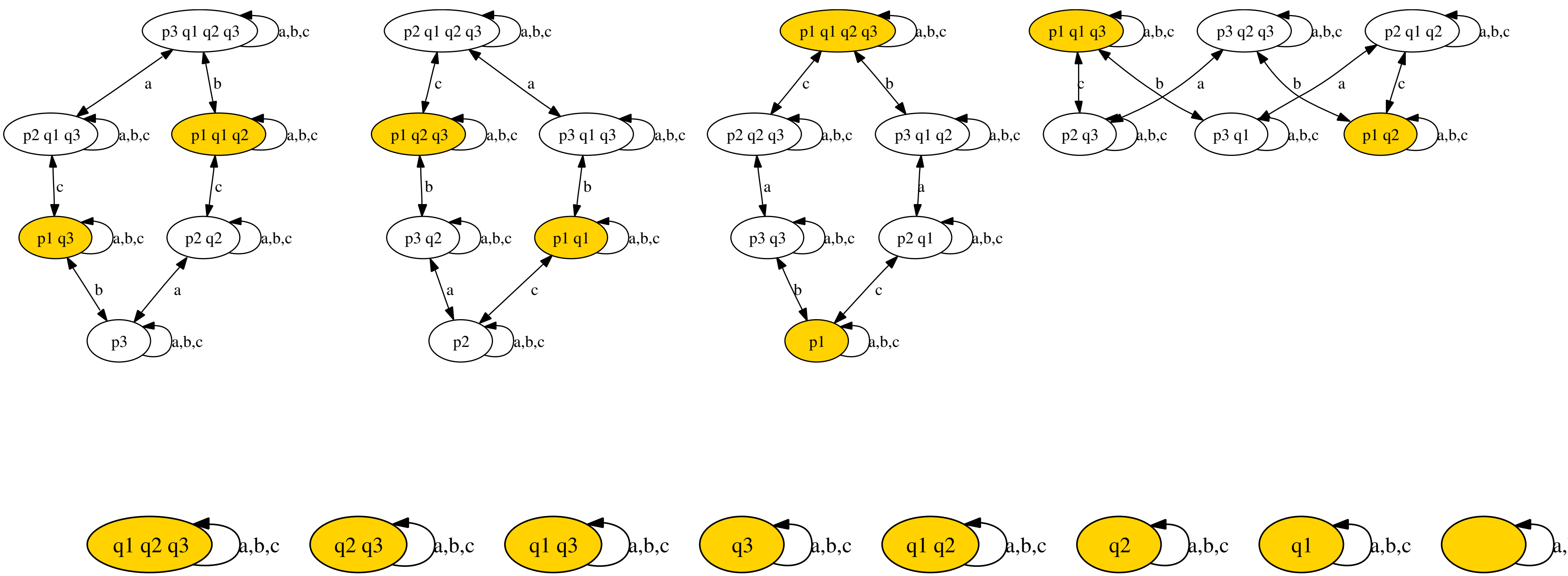
Epistemic formulas: a knows p, a knows that b knows p, a and b do not know p separately but they can deduce it together

With CADP

knowledge	$K_a\phi$	$[a]\phi$
common knowledge	$C_{a,b,c}\phi$	$\nu X.(\phi \wedge X)$
distributed knowledge	$D_{a,b,c}\phi$	`mgroup` + regular expressions on labels

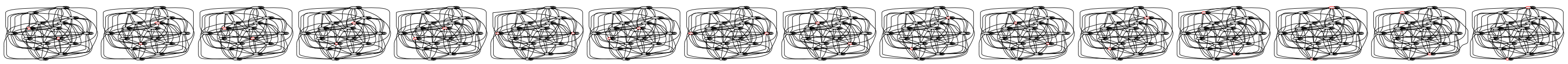
An unusual application of the CADP model checker

An epistemic state

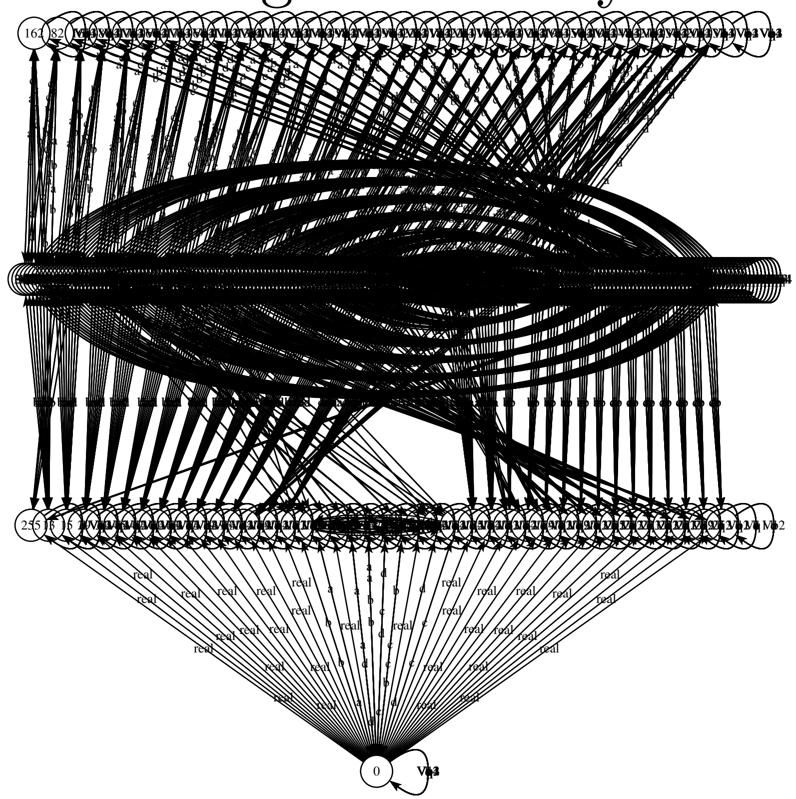


An unusual application of the CADP model checker

Another epistemic state



The same epistemic state as generated by CADP



An unusual application of the CADP model checker

<table>
<tr><th>Property</th><th>Corresponding AFMC formula</th><th>Answer</th></tr>
<tr><td>in DC3,
$C_{a,b,c}\ p_1 \vee p_2 \vee p_3$</td><td>["real"](
[’a|b|c’*]
(<"Vp1">true or <"Vp2">true or
<"Vp3">true)
or [’a|b|c’*]
(not(<"Vp1">true) and not(<"Vp2">true)
and not(<"Vp3">true)))</td><td>TRUE</td></tr>
<tr><td>in DC3,
$p_1 \vee p_2 \vee p_3 \Rightarrow \neg K_b p_1 \wedge \neg K_c p_1$</td><td>["real"] (<’’Vp1’’ | ‘‘Vp2’’
| ‘‘Vp3’’>true implies
(not([b]’’Vp1’’true) and not([c] <
‘‘Vp1’’> true))).</td><td>FALSE</td></tr>
<tr><td>in DC3,
$C_{a,b,c}(\neg K_b p_1 \wedge \neg K_c p_1)$</td><td>["real"] (nu X. ["a" | "b" | "c"]
((<"b"> ["Vp1"] false) and (<"c">
["Vp1"] false) and X))</td><td>TRUE</td></tr>
<tr><td>in DC3,
$p_1 \Rightarrow K_a\, q_2 \vee K_a\, \neg q_2$</td><td>["real"] (["b"] <"Vp2"> true or
["b"](not (<"Vp2">true)))</td><td>TRUE</td></tr>
<tr><td>in DC4,
$p_1 \Rightarrow C_{b,c}\, p_1$</td><td>["real"] ([’b|c’*]<"Vp1">true)</td><td>TRUE</td></tr>
</table>