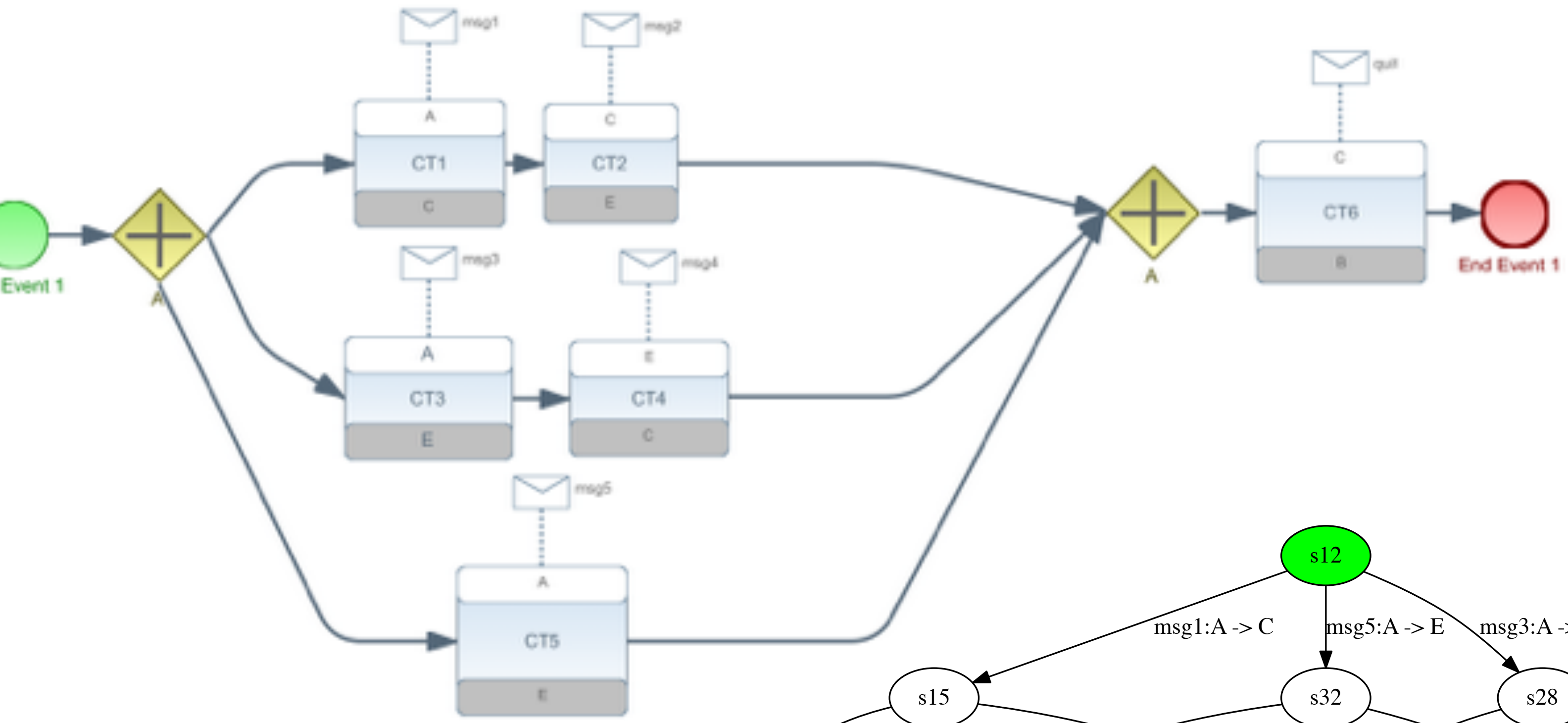
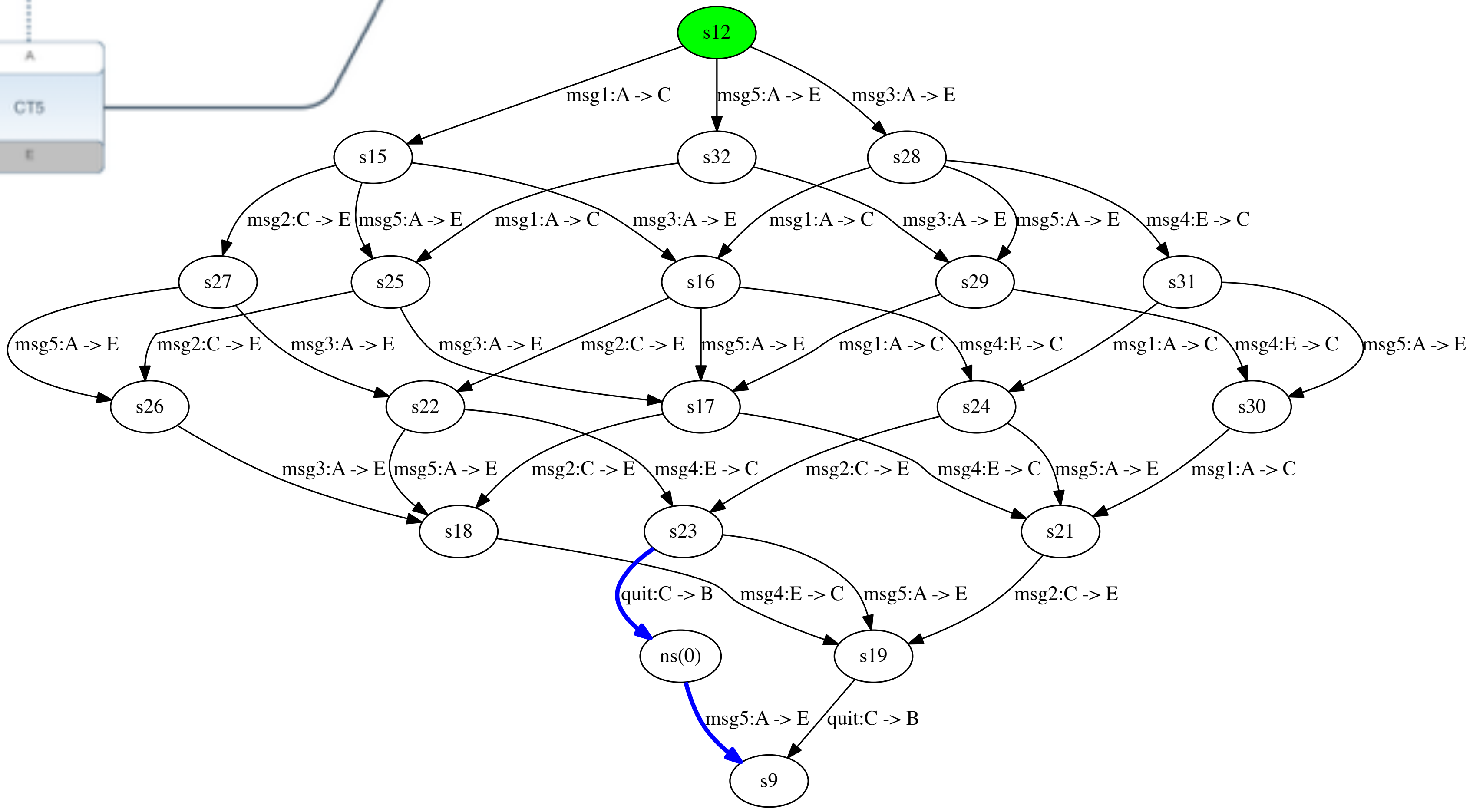

Event 1
A
msg1
A
CT1
C
msg2
C
CT2
E
msg3
A
CT3
E
msg4
E
CT4
C
msg5
A
CT5
E
A
quit
C
CT6
B
End Event 1

ADD REPAIR


s12
msg1:A -> C
msg5:A -> E
msg3:A -> E
s15
s32
s28
msg2:C -> E
msg5:A -> E
msg1:A -> C
msg3:A -> E
msg1:A -> C
msg3:A -> E
msg5:A -> E
msg4:E -> C
s27
s25
s16
s29
s31
msg5:A -> E
msg2:C -> E
msg3:A -> E
msg3:A -> E
msg2:C -> E
msg5:A -> E
msg1:A -> C
msg4:E -> C
msg1:A -> C
msg4:E -> C
msg5:A -> E
s26
s22
s17
s24
s30
msg3:A -> E
msg5:A -> E
msg2:C -> E
msg4:E -> C
msg2:C -> E
msg4:E -> C
msg5:A -> E
msg1:A -> C
s18
s23
s21
quit:C -> B
msg4:E -> C
msg5:A -> E
msg2:C -> E
ns(0)
s19
msg5:A -> E
quit:C -> B
s9

I am classifying repairs according to problem between:

- messages on the same sequential branch;
- messages on the same sequential branch in a parallel execution with other branches;
- messages between;

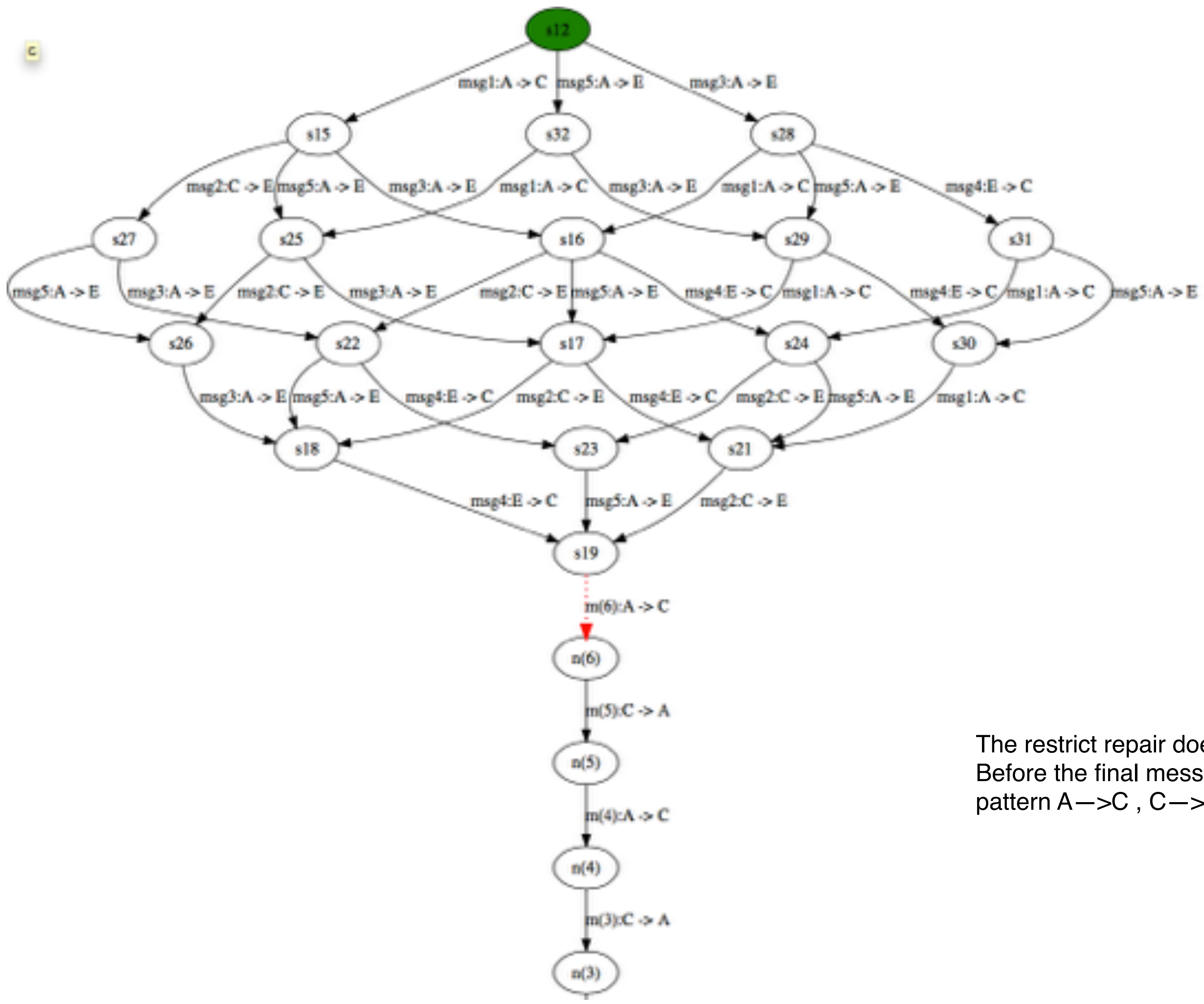
Despite the automaton of this example is a bit mazy, we are interested only in the final part. Considering the initial BPMN specification, if we execute *msg5* after all the others, we can't guarantee that *msg5* will be executed before *quit*, because all the involved peers are independent. The repair is correct, adding the possibility to execute *msg5* after *quit.*
Anyway this is a just theoretical repair, since I am modifying just one of all the possible parallel interleaving: I can't replicate this repair on the initial specification. So the practical solutions should one of these:

1. **Enfold the parallelism**
   Transform the parallel in a sequence of nested choices. This replicates the exact system's automaton and the exact repair. However I lose the specification readability and maintenance. I would discard this option.

2. **Put *msg5* after the parallel execution**
   I could put *msg5* before *quit*, but after the parallelism fragment. This restricts some good actions, and doesn't repair the choreography. Extra repair actions are needed between *msg5* and *msg2*, *msg4*. However, the extra repair needed, should be fine.

3. **Put *msg5* before the parallel execution**
   I could put *msg5* at the beginning. Since the sender of *msg5* is the same of the others branches (otherwise I should have already repaired that), it MAY repair the choreography without extra actions. However, it is not a general solution. It's only the case when there is just one message exchange in the branch.

4. **Force the *restrict* repair**



The restrict repair doesn't converge.
Before the final message *quit*, it adds the pattern A—>C , C—>A forever