WITNESS AND COUNTEREXAMPLE AUTOMATA

(Robert Meolic, Alessandro Fantechi, 2004, 2013)

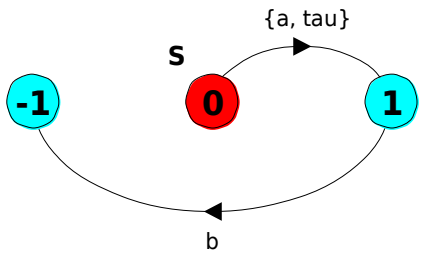
Witness and counterexample automata (WCA) recognize the set of finite linear witnesses and counterexamples, respectively. Project WCS (2004) resulted in a program that takes the model (subset of standard CCS) and the set of ACTL/ACTLW formulae on the input and produces a textual representation of a corresponding WCA. Technical report about this project can be obtained here:

http://puma.isti.cnr.it/dfdownloadnew.php?ident=/cnr.isti/2011-TR-027&langver=en&scelta=NewMetadata

Files **tau.dat**, **tau.actl**, and **tau.tcl** are simple test of ACTL model checking and WCA generation.

Here is specification of LTS and its visualisation with LTSA V3.0:

```
SORT S a,b
PROCESS S
SORT S
INITIAL STATE s0
TRANSITIONS
s0 = a?.s1 + TAU.s1
s1 = b?.s2
```



Here are ACTL formulae:

```
#### TRUE FOR ALL  ###
NOT EX {false} EX {b?} true;
EX {true} EX {b?} true;
EX {tau} EX {b?} true;
EX {a?} EX {b?} true;
NOT EX {NOT a?} EX {b?} true;
NOT AX {true} AX {b?} true;
NOT AX {tau} AX {b?} true;
NOT AX {a?} AX {b?} true;
E[true {false} U EX {b?} true];
E[true {true} U EX {b?} true];
E[true {a?} U EX {b?} true];
E[true {NOT a?} U EX {b?} true];
NOT A[true {false} U AX {b?} true];
A[true {true} U AX {b?} true];
A[true {a?} U AX {b?} true];
NOT A[true {NOT a?} U AX {b?} true];

#### FALSE FOR ALL ###
EX {false} EX {b?} true;
NOT EX {true} EX {b?} true;
NOT EX {tau} EX {b?} true;
NOT EX {a?} EX {b?} true;
EX {NOT a?} EX {b?} true;
```

```
AX {true} AX {b?} true;
AX {tau} AX {b?} true;
AX {a?} AX {b?} true;
NOT E[true {false} U EX {b?} true];
NOT E[true {true} U EX {b?} true];
NOT E[true {a?} U EX {b?} true];
NOT E[true {NOT a?} U EX {b?} true];
A[true {false} U AX {b?} true];
NOT A[true {true} U AX {b?} true];
NOT A[true {a?} U AX {b?} true];
A[true {NOT a?} U AX {b?} true];
```

Here is the log from EST:

```
Efficient Symbolic Tools, 2nd Edition, Copyright (C) 2003-2013 UM-FERI
This is free software, and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute EST; type "license" for details.

Running on i686 (Linux, 3.2.0-38-generic-pae) with tcl 8.5.11 and tk 8.5.11.

Initialization of GUI package... OK
Initialization of BDD package... OK
Initialization of Process_Algebra package... OK
Initialization of Versis package... OK
Initialization of Model checking package... OK
Initialization of Strucval package... OK
Initialization of CCS package... OK
Ready.

>cd "/home/meolic/est/est-2ed/data/automata"; source "tau.tcl"; cd "/home/meolic/est/est-2ed/data"
Reading file: tau.dat
  Sort S ... OK
  Process S ... OK

ACTL/ACTLW model checking on process S

### TRUE FOR ALL  ###
NOT EX{false} EX{b?} true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
EX{true} EX{b?} true ==> TRUE
Witness: (a?)(b?)
Witness automaton WC1S has been constructed.
EX{TAU} EX{b?} true ==> TRUE
Witness: (TAU)(b?)
Witness automaton WC2S has been constructed.
EX{a?} EX{b?} true ==> TRUE
Witness: (a?)(b?)
Witness automaton WC3S has been constructed.
NOT EX{NOT a?} EX{b?} true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT AX{true} AX{b?} true ==> TRUE
Witness: (TAU)
Witness automaton WC4S has been constructed.
NOT AX{TAU} AX{b?} true ==> TRUE
Witness: (a?)
Witness automaton WC5S has been constructed.
NOT AX{a?} AX{b?} true ==> TRUE
Witness: (TAU)
Witness automaton WC6S has been constructed.
E[true {false}U EX{b?} true] ==> TRUE
Witness: (TAU)(b?)
Witness automaton WC7S has been constructed.
E[true {true}U EX{b?} true] ==> TRUE
Witness: (TAU)(b?)
Witness automaton WC8S has been constructed.
E[true {a?}U EX{b?} true] ==> TRUE
```

```
Witness: (TAU)(b?)
Witness automaton WC9S has been constructed.
E[true {NOT a?}U EX{b?} true] ==> TRUE
Witness: (TAU)(b?)
Witness automaton WC10S has been constructed.
NOT A[true {false}U AX{b?} true] ==> TRUE
Witness: (a?)
Witness automaton cannot be constructed.
A[true {true}U AX{b?} true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
A[true {a?}U AX{b?} true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT A[true {NOT a?}U AX{b?} true] ==> TRUE
Witness: (a?)
Witness automaton cannot be constructed.
### FALSE FOR ALL ###
EX{false} EX{b?} true ==> FALSE
Counterexample not generated.
Counterexample automaton cannot be constructed.
NOT EX{true} EX{b?} true ==> FALSE
Counterexample: (a?)(b?)
Counterexample automaton WC11S has been constructed.
NOT EX{TAU} EX{b?} true ==> FALSE
Counterexample: (TAU)(b?)
Counterexample automaton WC12S has been constructed.
NOT EX{a?} EX{b?} true ==> FALSE
Counterexample: (a?)(b?)
Counterexample automaton WC13S has been constructed.
EX{NOT a?} EX{b?} true ==> FALSE
Counterexample not generated.
Counterexample automaton cannot be constructed.
AX{true} AX{b?} true ==> FALSE
Counterexample: (TAU)
Counterexample automaton WC14S has been constructed.
AX{TAU} AX{b?} true ==> FALSE
Counterexample: (a?)
Counterexample automaton WC15S has been constructed.
AX{a?} AX{b?} true ==> FALSE
Counterexample: (TAU)
Counterexample automaton WC16S has been constructed.
NOT E[true {false}U EX{b?} true] ==> FALSE
Counterexample: (TAU)(b?)
Counterexample automaton WC17S has been constructed.
NOT E[true {true}U EX{b?} true] ==> FALSE
Counterexample: (TAU)(b?)
Counterexample automaton WC18S has been constructed.
NOT E[true {a?}U EX{b?} true] ==> FALSE
Counterexample: (TAU)(b?)
Counterexample automaton WC19S has been constructed.
NOT E[true {NOT a?}U EX{b?} true] ==> FALSE
Counterexample: (TAU)(b?)
Counterexample automaton WC20S has been constructed.
A[true {false}U AX{b?} true] ==> FALSE
Counterexample: (a?)
Counterexample automaton cannot be constructed.
NOT A[true {true}U AX{b?} true] ==> FALSE
Counterexample not generated.
Counterexample automaton cannot be constructed.
NOT A[true {a?}U AX{b?} true] ==> FALSE
Counterexample not generated.
Counterexample automaton cannot be constructed.
A[true {NOT a?}U AX{b?} true] ==> FALSE
Counterexample: (a?)
Counterexample automaton cannot be constructed.

PROCESS S
INITIAL STATE s0
s0 = a? . s1 + TAU . s1
s1 = b? . s2

PROCESS WC1S
INITIAL STATE wc1
FINAL STATES  wc3
```

```
wc1 = a? . wc2
wc2 = b? . wc3

PROCESS WC2S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2
wc2 = b? . wc3

PROCESS WC3S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = a? . wc2
wc2 = b? . wc3

PROCESS WC4S
INITIAL STATE wc1
FINAL STATES  wc2
wc1 = TAU . wc2

PROCESS WC5S
INITIAL STATE wc1
FINAL STATES  wc2
wc1 = a? . wc2

PROCESS WC6S
INITIAL STATE wc1
FINAL STATES  wc2
wc1 = TAU . wc2

PROCESS WC7S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2
wc2 = b? . wc3

PROCESS WC8S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2 + a? . wc2
wc2 = b? . wc3

PROCESS WC9S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2 + a? . wc2
wc2 = b? . wc3

PROCESS WC10S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2
wc2 = b? . wc3

PROCESS WC11S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = a? . wc2
wc2 = b? . wc3

PROCESS WC12S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2
wc2 = b? . wc3

PROCESS WC13S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = a? . wc2
wc2 = b? . wc3

PROCESS WC14S
INITIAL STATE wc1
FINAL STATES  wc2
```

```
wc1 = TAU . wc2

PROCESS WC15S
INITIAL STATE wc1
FINAL STATES  wc2
wc1 = a? . wc2

PROCESS WC16S
INITIAL STATE wc1
FINAL STATES  wc2
wc1 = TAU . wc2

PROCESS WC17S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2
wc2 = b? . wc3

PROCESS WC18S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2 + a? . wc2
wc2 = b? . wc3

PROCESS WC19S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2 + a? . wc2
wc2 = b? . wc3

PROCESS WC20S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2
wc2 = b? . wc3
```

EST can explain generation of WC automata.

```
>mc_check_actl_file 0 S /home/meolic/est/est-2ed/data/automata/tau.actl [expr  $mc_automaton |
$mc_explain]
ACTL/ACTLW model checking on process S

### TRUE FOR ALL  ###
NOT EX{false} EX{b?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT EX{false} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `EX{false} EX{b?} true' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@ Witness automaton cannot be constructed.
@@ End Of Diagnostic

EX{true} EX{b?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EX{true} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   WAbuild: chosen transition s0-a?->s1 from delta2
@@   conbuild: created state wc2, created transition wc1-a?->wc2
@@   generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     WAbuild: chosen transition s1-b?->s2 from delta2
@@     conbuild: created state wc3, created transition wc2-b?->wc3
@@     generate: starting formula `true' for (s=s2, t=wc3)
@@       generate: added pair (wc3,s2) to R for the current formula
@@       generate: state wc3 marked as final
@@ Witness automaton WC21S has been constructed.
PROCESS WC21S
INITIAL STATE wc1
Number of states: 3
```

```
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = a? . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

EX{TAU} EX{b?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EX{TAU} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   WAbuild: chosen transition s0-TAU?->s1 from delta2
@@   conbuild: created state wc2, created transition wc1-TAU?->wc2
@@   generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     WAbuild: chosen transition s1-b?->s2 from delta2
@@     conbuild: created state wc3, created transition wc2-b?->wc3
@@     generate: starting formula `true' for (s=s2, t=wc3)
@@       generate: added pair (wc3,s2) to R for the current formula
@@       generate: state wc3 marked as final
@@ Witness automaton WC22S has been constructed.
PROCESS WC22S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = TAU . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

EX{a?} EX{b?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EX{a?} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   WAbuild: chosen transition s0-a?->s1 from delta2
@@   conbuild: created state wc2, created transition wc1-a?->wc2
@@   generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     WAbuild: chosen transition s1-b?->s2 from delta2
@@     conbuild: created state wc3, created transition wc2-b?->wc3
@@     generate: starting formula `true' for (s=s2, t=wc3)
@@       generate: added pair (wc3,s2) to R for the current formula
@@       generate: state wc3 marked as final
@@ Witness automaton WC23S has been constructed.
PROCESS WC23S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = a? . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

NOT EX{NOT a?} EX{b?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT EX{NOT a?} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `EX{NOT a?} EX{b?} true' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@ Witness automaton cannot be constructed.
@@ End Of Diagnostic

NOT AX{true} AX{b?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT AX{true} AX{b?} true' for (s=s0, t=wc1)
```

```
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `AX{true} AX{b?} true' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     CAbuild: chosen transition s0-TAU?->s1 from delta2
@@     conbuild: created state wc2, created transition wc1-TAU?->wc2
@@     generate: state wc2 marked as final
@@ Witness automaton WC24S has been constructed.
PROCESS WC24S
INITIAL STATE wc1
Number of states: 2
Number of final states: 1
FINAL STATES  wc2
Number of transitions: 1
wc1 = TAU . wc2
@@ End Of Diagnostic

NOT AX{TAU} AX{b?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT AX{TAU} AX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `AX{TAU} AX{b?} true' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     CAbuild: chosen transition s0-a?->s1 from delta2
@@     conbuild: created state wc2, created transition wc1-a?->wc2
@@     generate: state wc2 marked as final
@@ Witness automaton WC25S has been constructed.
PROCESS WC25S
INITIAL STATE wc1
Number of states: 2
Number of final states: 1
FINAL STATES  wc2
Number of transitions: 1
wc1 = a? . wc2
@@ End Of Diagnostic

NOT AX{a?} AX{b?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT AX{a?} AX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `AX{a?} AX{b?} true' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     CAbuild: chosen transition s0-TAU?->s1 from delta2
@@     conbuild: created state wc2, created transition wc1-TAU?->wc2
@@     generate: state wc2 marked as final
@@ Witness automaton WC26S has been constructed.
PROCESS WC26S
INITIAL STATE wc1
Number of states: 2
Number of final states: 1
FINAL STATES  wc2
Number of transitions: 1
wc1 = TAU . wc2
@@ End Of Diagnostic

E[true {false}U EX{b?} true] ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `E[true {false}U EX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   WAbuild: chosen transition s0-TAU?->s1 from delta1
@@   conbuild: created state wc2, created transition wc1-TAU?->wc2
@@   generate: starting formula `E[true {false}U EX{b?} true]' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       WAbuild: chosen transition s1-b?->s2 from delta2
@@       conbuild: created state wc3, created transition wc2-b?->wc3
@@       generate: starting formula `true' for (s=s2, t=wc3)
@@         generate: added pair (wc3,s2) to R for the current formula
@@         generate: state wc3 marked as final
```

```
@@   WAbuild: chosen transition s0-TAU?->s1 from delta2
@@   conbuild: transition wc1-TAU?->wc2 exists
@@ Witness automaton WC27S has been constructed.
PROCESS WC27S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = TAU . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

E[true {true}U EX{b?} true] ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `E[true {true}U EX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   WAbuild: chosen transition s0-TAU?->s1 from delta1
@@   conbuild: created state wc2, created transition wc1-TAU?->wc2
@@   generate: starting formula `E[true {true}U EX{b?} true]' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       WAbuild: chosen transition s1-b?->s2 from delta2
@@       conbuild: created state wc3, created transition wc2-b?->wc3
@@       generate: starting formula `true' for (s=s2, t=wc3)
@@         generate: added pair (wc3,s2) to R for the current formula
@@         generate: state wc3 marked as final
@@   WAbuild: chosen transition s0-a?->s1 from delta1
@@   conbuild: created transition wc1-a?->wc2
@@   WAbuild: chosen transition s0-TAU?->s1 from delta2
@@   conbuild: transition wc1-TAU?->wc2 exists
@@   WAbuild: chosen transition s0-a?->s1 from delta2
@@   conbuild: transition wc1-a?->wc2 exists
@@ Witness automaton WC28S has been constructed.
PROCESS WC28S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 3
wc1 = TAU . wc2 + a? . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

E[true {a?}U EX{b?} true] ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `E[true {a?}U EX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   WAbuild: chosen transition s0-TAU?->s1 from delta1
@@   conbuild: created state wc2, created transition wc1-TAU?->wc2
@@   generate: starting formula `E[true {a?}U EX{b?} true]' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       WAbuild: chosen transition s1-b?->s2 from delta2
@@       conbuild: created state wc3, created transition wc2-b?->wc3
@@       generate: starting formula `true' for (s=s2, t=wc3)
@@         generate: added pair (wc3,s2) to R for the current formula
@@         generate: state wc3 marked as final
@@   WAbuild: chosen transition s0-a?->s1 from delta1
@@   conbuild: created transition wc1-a?->wc2
@@   WAbuild: chosen transition s0-TAU?->s1 from delta2
@@   conbuild: transition wc1-TAU?->wc2 exists
@@   WAbuild: chosen transition s0-a?->s1 from delta2
@@   conbuild: transition wc1-a?->wc2 exists
@@ Witness automaton WC29S has been constructed.
PROCESS WC29S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
```

```
FINAL STATES  wc3
Number of transitions: 3
wc1 = TAU . wc2 + a? . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

E[true {NOT a?}U EX{b?} true] ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `E[true {NOT a?}U EX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   WAbuild: chosen transition s0-TAU?->s1 from delta1
@@   conbuild: created state wc2, created transition wc1-TAU?->wc2
@@   generate: starting formula `E[true {NOT a?}U EX{b?} true]' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       WAbuild: chosen transition s1-b?->s2 from delta2
@@       conbuild: created state wc3, created transition wc2-b?->wc3
@@       generate: starting formula `true' for (s=s2, t=wc3)
@@         generate: added pair (wc3,s2) to R for the current formula
@@         generate: state wc3 marked as final
@@   WAbuild: chosen transition s0-TAU?->s1 from delta2
@@   conbuild: transition wc1-TAU?->wc2 exists
@@ Witness automaton WC30S has been constructed.
PROCESS WC30S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = TAU . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

NOT A[true {false}U AX{b?} true] ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT A[true {false}U AX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `A[true {false}U AX{b?} true]' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@ Witness automaton cannot be constructed.
@@ End Of Diagnostic

A[true {true}U AX{b?} true] ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `A[true {true}U AX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@ Witness automaton cannot be constructed.
@@ End Of Diagnostic

A[true {a?}U AX{b?} true] ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `A[true {a?}U AX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@ Witness automaton cannot be constructed.
@@ End Of Diagnostic

NOT A[true {NOT a?}U AX{b?} true] ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT A[true {NOT a?}U AX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `A[true {NOT a?}U AX{b?} true]' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@ Witness automaton cannot be constructed.
@@ End Of Diagnostic
```

```
### FALSE FOR ALL ###
EX{false} EX{b?} true ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EX{false} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@ Counterexample automaton cannot be constructed.
@@ End Of Diagnostic

NOT EX{true} EX{b?} true ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT EX{true} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `EX{true} EX{b?} true' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     WAbuild: chosen transition s0-a?->s1 from delta2
@@     conbuild: created state wc2, created transition wc1-a?->wc2
@@     generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       WAbuild: chosen transition s1-b?->s2 from delta2
@@       conbuild: created state wc3, created transition wc2-b?->wc3
@@       generate: starting formula `true' for (s=s2, t=wc3)
@@         generate: added pair (wc3,s2) to R for the current formula
@@         generate: state wc3 marked as final
@@ Counterexample automaton WC31S has been constructed.
PROCESS WC31S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = a? . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

NOT EX{TAU} EX{b?} true ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT EX{TAU} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `EX{TAU} EX{b?} true' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     WAbuild: chosen transition s0-TAU?->s1 from delta2
@@     conbuild: created state wc2, created transition wc1-TAU?->wc2
@@     generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       WAbuild: chosen transition s1-b?->s2 from delta2
@@       conbuild: created state wc3, created transition wc2-b?->wc3
@@       generate: starting formula `true' for (s=s2, t=wc3)
@@         generate: added pair (wc3,s2) to R for the current formula
@@         generate: state wc3 marked as final
@@ Counterexample automaton WC32S has been constructed.
PROCESS WC32S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = TAU . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

NOT EX{a?} EX{b?} true ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT EX{a?} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `EX{a?} EX{b?} true' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
```

```
@@     WAbuild: chosen transition s0-a?->s1 from delta2
@@     conbuild: created state wc2, created transition wc1-a?->wc2
@@     generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       WAbuild: chosen transition s1-b?->s2 from delta2
@@       conbuild: created state wc3, created transition wc2-b?->wc3
@@       generate: starting formula `true' for (s=s2, t=wc3)
@@         generate: added pair (wc3,s2) to R for the current formula
@@         generate: state wc3 marked as final
@@ Counterexample automaton WC33S has been constructed.
PROCESS WC33S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = a? . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

EX{NOT a?} EX{b?} true ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EX{NOT a?} EX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@ Counterexample automaton cannot be constructed.
@@ End Of Diagnostic

AX{true} AX{b?} true ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `AX{true} AX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   CAbuild: chosen transition s0-TAU?->s1 from delta2
@@   conbuild: created state wc2, created transition wc1-TAU?->wc2
@@   generate: state wc2 marked as final
@@ Counterexample automaton WC34S has been constructed.
PROCESS WC34S
INITIAL STATE wc1
Number of states: 2
Number of final states: 1
FINAL STATES  wc2
Number of transitions: 1
wc1 = TAU . wc2
@@ End Of Diagnostic

AX{TAU} AX{b?} true ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `AX{TAU} AX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   CAbuild: chosen transition s0-a?->s1 from delta2
@@   conbuild: created state wc2, created transition wc1-a?->wc2
@@   generate: state wc2 marked as final
@@ Counterexample automaton WC35S has been constructed.
PROCESS WC35S
INITIAL STATE wc1
Number of states: 2
Number of final states: 1
FINAL STATES  wc2
Number of transitions: 1
wc1 = a? . wc2
@@ End Of Diagnostic

AX{a?} AX{b?} true ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `AX{a?} AX{b?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   CAbuild: chosen transition s0-TAU?->s1 from delta2
@@   conbuild: created state wc2, created transition wc1-TAU?->wc2
```

```
@@   generate: state wc2 marked as final
@@ Counterexample automaton WC36S has been constructed.
PROCESS WC36S
INITIAL STATE wc1
Number of states: 2
Number of final states: 1
FINAL STATES  wc2
Number of transitions: 1
wc1 = TAU . wc2
@@ End Of Diagnostic

NOT E[true {false}U EX{b?} true] ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT E[true {false}U EX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `E[true {false}U EX{b?} true]' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     WAbuild: chosen transition s0-TAU?->s1 from delta1
@@     conbuild: created state wc2, created transition wc1-TAU?->wc2
@@     generate: starting formula `E[true {false}U EX{b?} true]' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@         generate: added pair (wc2,s1) to R for the current formula
@@         WAbuild: chosen transition s1-b?->s2 from delta2
@@         conbuild: created state wc3, created transition wc2-b?->wc3
@@         generate: starting formula `true' for (s=s2, t=wc3)
@@           generate: added pair (wc3,s2) to R for the current formula
@@           generate: state wc3 marked as final
@@     WAbuild: chosen transition s0-TAU?->s1 from delta2
@@     conbuild: transition wc1-TAU?->wc2 exists
@@ Counterexample automaton WC37S has been constructed.
PROCESS WC37S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = TAU . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

NOT E[true {true}U EX{b?} true] ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT E[true {true}U EX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `E[true {true}U EX{b?} true]' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     WAbuild: chosen transition s0-TAU?->s1 from delta1
@@     conbuild: created state wc2, created transition wc1-TAU?->wc2
@@     generate: starting formula `E[true {true}U EX{b?} true]' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@         generate: added pair (wc2,s1) to R for the current formula
@@         WAbuild: chosen transition s1-b?->s2 from delta2
@@         conbuild: created state wc3, created transition wc2-b?->wc3
@@         generate: starting formula `true' for (s=s2, t=wc3)
@@           generate: added pair (wc3,s2) to R for the current formula
@@           generate: state wc3 marked as final
@@     WAbuild: chosen transition s0-a?->s1 from delta1
@@     conbuild: created transition wc1-a?->wc2
@@     WAbuild: chosen transition s0-TAU?->s1 from delta2
@@     conbuild: transition wc1-TAU?->wc2 exists
@@     WAbuild: chosen transition s0-a?->s1 from delta2
@@     conbuild: transition wc1-a?->wc2 exists
@@ Counterexample automaton WC38S has been constructed.
PROCESS WC38S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 3
```

```
wc1 = TAU . wc2 + a? . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

NOT E[true {a?}U EX{b?} true] ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT E[true {a?}U EX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `E[true {a?}U EX{b?} true]' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     WAbuild: chosen transition s0-TAU?->s1 from delta1
@@     conbuild: created state wc2, created transition wc1-TAU?->wc2
@@     generate: starting formula `E[true {a?}U EX{b?} true]' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@         generate: added pair (wc2,s1) to R for the current formula
@@         WAbuild: chosen transition s1-b?->s2 from delta2
@@         conbuild: created state wc3, created transition wc2-b?->wc3
@@         generate: starting formula `true' for (s=s2, t=wc3)
@@           generate: added pair (wc3,s2) to R for the current formula
@@           generate: state wc3 marked as final
@@     WAbuild: chosen transition s0-a?->s1 from delta1
@@     conbuild: created transition wc1-a?->wc2
@@     WAbuild: chosen transition s0-TAU?->s1 from delta2
@@     conbuild: transition wc1-TAU?->wc2 exists
@@     WAbuild: chosen transition s0-a?->s1 from delta2
@@     conbuild: transition wc1-a?->wc2 exists
@@ Counterexample automaton WC39S has been constructed.
PROCESS WC39S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 3
wc1 = TAU . wc2 + a? . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

NOT E[true {NOT a?}U EX{b?} true] ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT E[true {NOT a?}U EX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `E[true {NOT a?}U EX{b?} true]' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     WAbuild: chosen transition s0-TAU?->s1 from delta1
@@     conbuild: created state wc2, created transition wc1-TAU?->wc2
@@     generate: starting formula `E[true {NOT a?}U EX{b?} true]' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       generate: starting formula `EX{b?} true' for (s=s1, t=wc2)
@@         generate: added pair (wc2,s1) to R for the current formula
@@         WAbuild: chosen transition s1-b?->s2 from delta2
@@         conbuild: created state wc3, created transition wc2-b?->wc3
@@         generate: starting formula `true' for (s=s2, t=wc3)
@@           generate: added pair (wc3,s2) to R for the current formula
@@           generate: state wc3 marked as final
@@     WAbuild: chosen transition s0-TAU?->s1 from delta2
@@     conbuild: transition wc1-TAU?->wc2 exists
@@ Counterexample automaton WC40S has been constructed.
PROCESS WC40S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = TAU . wc2
wc2 = b? . wc3
@@ End Of Diagnostic

A[true {false}U AX{b?} true] ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
```

```
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `A[true {false}U AX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@ Counterexample automaton cannot be constructed.
@@ End Of Diagnostic

NOT A[true {true}U AX{b?} true] ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT A[true {true}U AX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `A[true {true}U AX{b?} true]' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@ Counterexample automaton cannot be constructed.
@@ End Of Diagnostic

NOT A[true {a?}U AX{b?} true] ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `NOT A[true {a?}U AX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `A[true {a?}U AX{b?} true]' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@ Counterexample automaton cannot be constructed.
@@ End Of Diagnostic

A[true {NOT a?}U AX{b?} true] ==> FALSE
@@ Creating counterexample automaton
@@ WCAgenerator: created empty counterexample automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `A[true {NOT a?}U AX{b?} true]' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@ Counterexample automaton cannot be constructed.
@@ End Of Diagnostic
```

Files **deadlock.dat**, **deadlock.actl**, and **deadlock.tcl** are another simple test of ACTL/ACTLW model checking and WCA generation.

Here is specification of LTS (empty process!)

```
SORT S a,b
PROCESS S
SORT S
INITIAL STATE s0
TRANSITIONS
```

Here are ACTL/ACTLW formulae:

```
/* ### TRUE FOR ALL  ### */

NOT EEX {a?} true;
NOT EEF {a?} true;
EEG {a?} true;
NOT EEG false {a?};
EEG true {a?};
NOT EE[{a?} true U {b?} true];
EE[{a?} true W {b?} true];
NOT EE[true {a?} U {b?} true];
EE[true {a?} W {b?} true];
```

```
AAX {a?} true;
NOT AAF {a?} true;
AAG {a?} true;
NOT AAG false {a?};
AAG true {a?};
NOT AA[{a?} true U {b?} true];
AA[{a?} true W {b?} true];
NOT AA[true {a?} U {b?} true];
AA[true {a?} W {b?} true];

NOT EX {false} false;
NOT EX {true} true;
NOT EX {tau} false;
NOT EX {tau} true;
NOT EF false;
EF true;
NOT EG false;
EG true;
E[false {a?}U  true];
NOT E[false {a?} U {b?} true];
E[true {a?} U true];
NOT E[true {a?} U {b?} true];

NOT AX {false} false;
NOT AX {true} true;
NOT AX {tau} false;
NOT AX {tau} true;
NOT AF false;
AF true;
NOT AG false;
AG true;
A[false {a?} U true];
NOT A[false {a?} U {b?} true];
A[true {a?} U true];
NOT A[true {a?} U {b?} true];

NOT <a?> false;
NOT <a?> true;
[a?] false;
[a?] true;
```

Here is the log from EST:

```
Efficient Symbolic Tools, 2nd Edition, Copyright (C) 2003-2013 UM-FERI
This is free software, and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute EST; type "license" for details.

Running on i686 (Linux, 3.2.0-38-generic-pae) with tcl 8.5.11 and tk 8.5.11.

Initialization of GUI package... OK
Initialization of BDD package... OK
Initialization of Process_Algebra package... OK
Initialization of Versis package... OK
```

```
Initialization of Model checking package... OK
Initialization of Strucval package... OK
Initialization of CCS package... OK
Ready.

>cd "/home/meolic/est/est-2ed/data/automata"; source "deadlock.tcl"; cd "/home/meolic/est/est-
2ed/data"
Reading file: deadlock.dat
  Sort S ... OK
  Process S ... OK

ACTL/ACTLW model checking on process S

NOT EEX {a?} true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT EEF {a?} true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
EEG {a?} true ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT EEG false {a?} ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
EEG true {a?} ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT EE[{a?} true U {b?} true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
EE[{a?} true W {b?} true] ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT EE[true {a?} U {b?} true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
EE[true {a?} W {b?} true] ==> TRUE
Witness not generated.
Witness automaton not constructed.
AAX {a?} true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT AAF {a?} true ==> TRUE
Witness not generated.
Witness automaton not constructed.
AAG {a?} true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT AAG false {a?} ==> TRUE
Witness not generated.
Witness automaton not constructed.
AAG true {a?} ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT AA[{a?} true U {b?} true] ==> TRUE
Witness not generated.
Witness automaton not constructed.
AA[{a?} true W {b?} true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT AA[true {a?} U {b?} true] ==> TRUE
Witness not generated.
Witness automaton not constructed.
AA[true {a?} W {b?} true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT EX{false} false ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT EX{true} true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT EX{TAU} false ==> TRUE
Witness not generated.
```

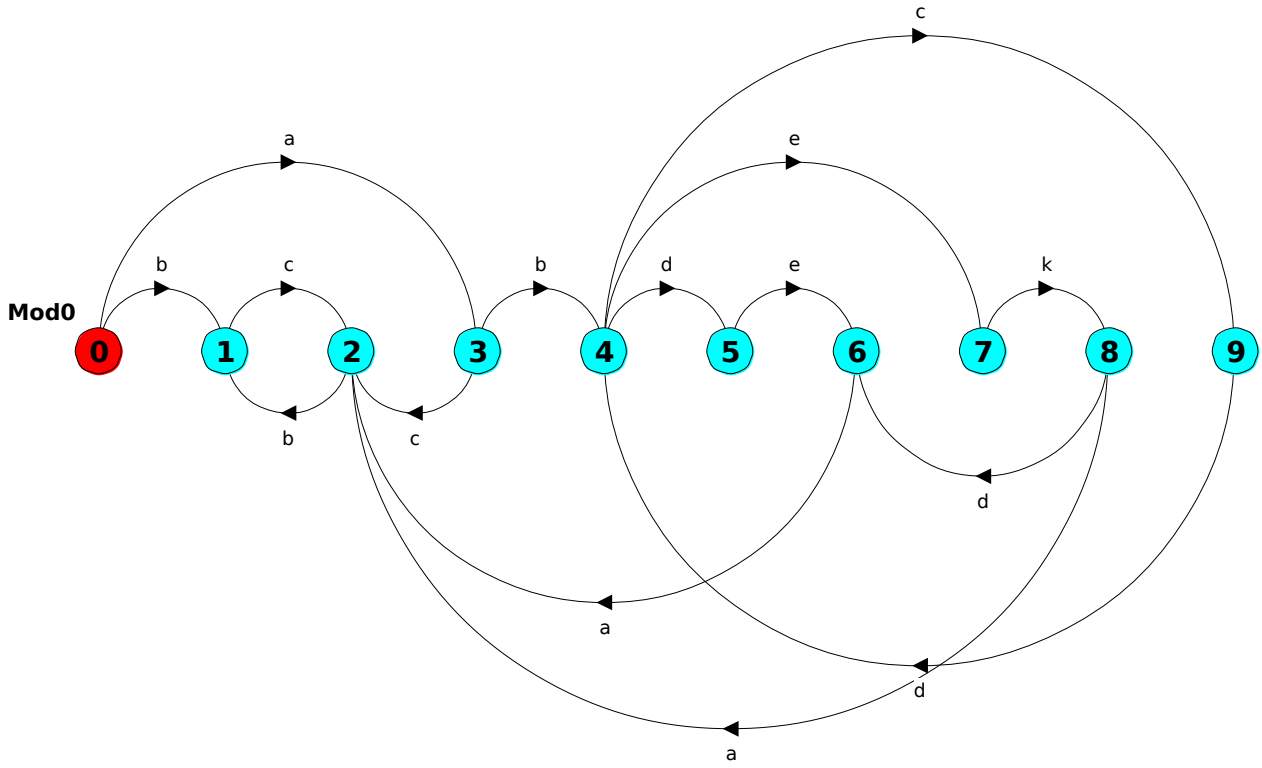
```
Witness automaton cannot be constructed.
NOT EX{TAU} true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT EF false ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
EF true ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT EG false ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
EG true ==> TRUE
Witness not generated.
Witness automaton not constructed.
E[false {a?}U true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT E[false {a?}U{b?} true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
E[true {a?}U true] ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT E[true {a?}U{b?} true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT AX{false} false ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT AX{true} true ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT AX{TAU} false ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT AX{TAU} true ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT AF false ==> TRUE
Witness not generated.
Witness automaton not constructed.
AF true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT AG false ==> TRUE
Witness not generated.
Witness automaton not constructed.
AG true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
A[false {a?}U true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT A[false {a?}U{b?} true] ==> TRUE
Witness not generated.
Witness automaton not constructed.
A[true {a?}U true] ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT A[true {a?}U{b?} true] ==> TRUE
Witness not generated.
Witness automaton not constructed.
NOT <a?> false ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
NOT <a?> true ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
[a?] false ==> TRUE
Witness not generated.
Witness automaton cannot be constructed.
[a?] true ==> TRUE
Witness not generated.
```

```
Witness automaton cannot be constructed.
```

Files **alessandro.dat**, **alessandro.actl**, and **alessandro.tcl** are test of WCA generation proposed by Alessandro Fantechi.

Here is specification of LTS and its visualisation with LTSA V3.0:

```
SORT sortAutomata a,b,c,d,e,k
PROCESS Mod0
SORT sortAutomata
INITIAL STATE s0
TRANSITIONS
s0 = a?.s1
s0 = b?.s2
s1 = b?.s4
s1 = c?.s3
s2 = c?.s3
s3 = b?.s2
s4 = e?.s5
s4 = d?.s6
s4 = c?.s7
s5 = k!.s8
s6 = e?.s9
s7 = d?.s4
s8 = a?.s3
s8 = d?.s9
s9 = a?.s3
```



The following (almost equivalent) ACTL and ACTLW formula are used:

```
EF EX {k!} true;
EEF {true} EEX {k!} true;
```

Here is the log from EST:

```
Efficient Symbolic Tools, 2nd Edition, Copyright (C) 2003-2013 UM-FERI
This is free software, and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute EST; type "license" for details.

Running on i686 (Linux, 3.2.0-38-generic-pae) with tcl 8.5.11 and tk 8.5.11.

Initialization of GUI package... OK
Initialization of BDD package... OK
Initialization of Process_Algebra package... OK
Initialization of Versis package... OK
Initialization of Model checking package... OK
Initialization of Strucval package... OK
Initialization of CCS package... OK
Ready.

>cd "/home/meolic/est/est-2ed/data/automata"; source "alessandro.tcl"; cd "/home/meolic/est/est-
2ed/data"
Reading file: alessandro.dat
  Sort sortAutomata ... OK
  Process Mod0 ... OK

ACTL/ACTLW model checking on process Mod0

EF EX{k!} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EF EX{k!} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   WAbuild: chosen transition s0-a?->s1 from delta1
@@   conbuild: created state wc2, created transition wc1-a?->wc2
@@   generate: starting formula `EF EX{k!} true' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     WAbuild: chosen transition s1-b?->s4 from delta1
@@     conbuild: created state wc3, created transition wc2-b?->wc3
@@     generate: starting formula `EF EX{k!} true' for (s=s4, t=wc3)
@@       generate: added pair (wc3,s4) to R for the current formula
@@       WAbuild: chosen transition s4-c?->s7 from delta1
@@       conbuild: created state wc4, created transition wc3-c?->wc4
@@       generate: starting formula `EF EX{k!} true' for (s=s7, t=wc4)
@@         generate: added pair (wc4,s7) to R for the current formula
@@         WAbuild: chosen transition s7-d?->s4 from delta1
@@         conbuild: created transition wc4-d?->wc3
@@       WAbuild: chosen transition s4-e?->s5 from delta1
@@       conbuild: created state wc5, created transition wc3-e?->wc5
@@       generate: starting formula `EF EX{k!} true' for (s=s5, t=wc5)
@@         generate: added pair (wc5,s5) to R for the current formula
@@         generate: starting formula `EX{k!} true' for (s=s5, t=wc5)
@@           generate: added pair (wc5,s5) to R for the current formula
@@           WAbuild: chosen transition s5-k!->s8 from delta2
@@           conbuild: created state wc6, created transition wc5-k!->wc6
@@           generate: starting formula `true' for (s=s8, t=wc6)
@@             generate: added pair (wc6,s8) to R for the current formula
@@             generate: state wc6 marked as final
@@       WAbuild: chosen transition s4-e?->s5 from delta2
@@       conbuild: transition wc3-e?->wc5 exists
@@ Witness automaton WC1Mod0 has been constructed.
PROCESS WC1Mod0
INITIAL STATE wc1
Number of states: 6
Number of final states: 1
FINAL STATES  wc6
Number of transitions: 6
wc1 = a? . wc2
```

```
wc2 = b? . wc3
wc3 = c? . wc4 + e? . wc5
wc4 = d? . wc3
wc5 = k! . wc6
@@ End Of Diagnostic

EEF {true} EEX {k!} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EEF {true} EEX {k!} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   WAbuild: chosen transition s0-a?->s1 from delta1
@@   conbuild: created state wc2, created transition wc1-a?->wc2
@@   generate: starting formula `EEF {true} EEX {k!} true' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     WAbuild: chosen transition s1-b?->s4 from delta1
@@     conbuild: created state wc3, created transition wc2-b?->wc3
@@     generate: starting formula `EEF {true} EEX {k!} true' for (s=s4, t=wc3)
@@       generate: added pair (wc3,s4) to R for the current formula
@@       WAbuild: chosen transition s4-c?->s7 from delta1
@@       conbuild: created state wc4, created transition wc3-c?->wc4
@@       generate: starting formula `EEF {true} EEX {k!} true' for (s=s7, t=wc4)
@@         generate: added pair (wc4,s7) to R for the current formula
@@         WAbuild: chosen transition s7-d?->s4 from delta1
@@         conbuild: created transition wc4-d?->wc3
@@       WAbuild: chosen transition s4-e?->s5 from delta2
@@       conbuild: created state wc5, created transition wc3-e?->wc5
@@       generate: starting formula `EEX {k!} true' for (s=s5, t=wc5)
@@         generate: added pair (wc5,s5) to R for the current formula
@@         WAbuild: chosen transition s5-k!->s8 from delta2
@@         conbuild: created state wc6, created transition wc5-k!->wc6
@@         generate: starting formula `true' for (s=s8, t=wc6)
@@           generate: added pair (wc6,s8) to R for the current formula
@@           generate: state wc6 marked as final
@@ Witness automaton WC2Mod0 has been constructed.
PROCESS WC2Mod0
INITIAL STATE wc1
Number of states: 6
Number of final states: 1
FINAL STATES  wc6
Number of transitions: 6
wc1 = a? . wc2
wc2 = b? . wc3
wc3 = c? . wc4 + e? . wc5
wc4 = d? . wc3
wc5 = k! . wc6
@@ End Of Diagnostic
```

ACTL/ACTLW formula can be also given as a string. Here is an example:

```
>mc_check_actl_string 0 Mod0 "EF EX{a?} true" [expr  $mc_automaton | $mc_explain]
ACTL/ACTLW model checking on process Mod0

EF EX{a?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EF EX{a?} true' for (s=s0, t=wc1)
@@   generate: added pair (wc1,s0) to R for the current formula
@@   generate: starting formula `EX{a?} true' for (s=s0, t=wc1)
@@     generate: added pair (wc1,s0) to R for the current formula
@@     WAbuild: chosen transition s0-a?->s1 from delta2
@@     conbuild: created state wc2, created transition wc1-a?->wc2
@@     generate: starting formula `true' for (s=s1, t=wc2)
@@       generate: added pair (wc2,s1) to R for the current formula
@@       generate: state wc2 marked as final
@@   WAbuild: chosen transition s0-a?->s1 from delta1
@@   conbuild: created state wc3, created transition wc1-a?->wc3
@@   generate: starting formula `EF EX{a?} true' for (s=s1, t=wc3)
@@     generate: added pair (wc3,s1) to R for the current formula
@@     WAbuild: chosen transition s1-b?->s4 from delta1
@@     conbuild: created state wc4, created transition wc3-b?->wc4
```

```
@@     generate: starting formula `EF EX{a?} true' for (s=s4, t=wc4)
@@       generate: added pair (wc4,s4) to R for the current formula
@@       WAbuild: chosen transition s4-c?->s7 from delta1
@@       conbuild: created state wc5, created transition wc4-c?->wc5
@@       generate: starting formula `EF EX{a?} true' for (s=s7, t=wc5)
@@         generate: added pair (wc5,s7) to R for the current formula
@@         WAbuild: chosen transition s7-d?->s4 from delta1
@@         conbuild: created transition wc5-d?->wc4
@@       WAbuild: chosen transition s4-e?->s5 from delta1
@@       conbuild: created state wc6, created transition wc4-e?->wc6
@@       generate: starting formula `EF EX{a?} true' for (s=s5, t=wc6)
@@         generate: added pair (wc6,s5) to R for the current formula
@@         WAbuild: chosen transition s5-k!->s8 from delta1
@@         conbuild: created state wc7, created transition wc6-k!->wc7
@@         generate: starting formula `EF EX{a?} true' for (s=s8, t=wc7)
@@           generate: added pair (wc7,s8) to R for the current formula
@@           generate: starting formula `EX{a?} true' for (s=s8, t=wc7)
@@             generate: added pair (wc7,s8) to R for the current formula
@@             WAbuild: chosen transition s8-a?->s3 from delta2
@@             conbuild: created state wc8, created transition wc7-a?->wc8
@@             generate: starting formula `true' for (s=s3, t=wc8)
@@               generate: added pair (wc8,s3) to R for the current formula
@@               generate: state wc8 marked as final
@@           WAbuild: chosen transition s8-d?->s9 from delta1
@@           conbuild: created state wc9, created transition wc7-d?->wc9
@@           generate: starting formula `EF EX{a?} true' for (s=s9, t=wc9)
@@             generate: added pair (wc9,s9) to R for the current formula
@@             generate: starting formula `EX{a?} true' for (s=s9, t=wc9)
@@               generate: added pair (wc9,s9) to R for the current formula
@@               WAbuild: chosen transition s9-a?->s3 from delta2
@@               conbuild: created transition wc9-a?->wc8
@@           WAbuild: chosen transition s8-d?->s9 from delta2
@@           conbuild: transition wc7-d?->wc9 exists
@@         WAbuild: chosen transition s5-k!->s8 from delta2
@@         conbuild: transition wc6-k!->wc7 exists
@@       WAbuild: chosen transition s4-d?->s6 from delta1
@@       conbuild: created state wc10, created transition wc4-d?->wc10
@@       generate: starting formula `EF EX{a?} true' for (s=s6, t=wc10)
@@         generate: added pair (wc10,s6) to R for the current formula
@@         WAbuild: chosen transition s6-e?->s9 from delta1
@@         conbuild: created transition wc10-e?->wc9
@@         WAbuild: chosen transition s6-e?->s9 from delta2
@@         conbuild: transition wc10-e?->wc9 exists
@@ Witness automaton WC3Mod0 has been constructed.
PROCESS WC3Mod0
INITIAL STATE wc1
Number of states: 10
Number of final states: 2
FINAL STATES  wc2 wc8
Number of transitions: 12
wc1 = a? . wc2 + a? . wc3
wc3 = b? . wc4
wc4 = c? . wc5 + e? . wc6 + d? . wc10
wc5 = d? . wc4
wc6 = k! . wc7
wc10 = e? . wc9
wc7 = a? . wc8 + d? . wc9
wc9 = a? . wc8
@@ End Of Diagnostic
```

Files **example1.dat**, **example1.ccs**, **example1.actl**, and **example1.tcl** are first example of WCA generation given in paper "Witness and Counterexample Automata for ACTL" (FORTE 2004). Format .**dat** is preffered over .**ccs** because CCS parser renames states and thus explanations are not directly usable.

Here is the log from EST:

```
Efficient Symbolic Tools, 2nd Edition, Copyright (C) 2003-2013 UM-FERI
This is free software, and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute EST; type "license" for details.
```

```
Running on i686 (Linux, 3.2.0-38-generic-pae) with tcl 8.5.11 and tk 8.5.11.

Initialization of GUI package... OK
Initialization of BDD package... OK
Initialization of Process_Algebra package... OK
Initialization of Versis package... OK
Initialization of Model checking package... OK
Initialization of Strucval package... OK
Initialization of CCS package... OK
Ready.

>cd "/home/meolic/est/est-2ed/data/automata"; source "example1.tcl"; cd "/home/meolic/est/est-
2ed/data"
Reading file: example1.dat
  Sort S ... OK
  Process S ... OK

ACTL/ACTLW model checking on process S

EX{a?} EX{a?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EX{a?} EX{a?} true' for (s=s1, t=wc1)
@@   generate: added pair (wc1,s1) to R for the current formula
@@   WAbuild: chosen transition s1-a?->s1 from delta2
@@   conbuild: created state wc2, created transition wc1-a?->wc2
@@   generate: starting formula `EX{a?} true' for (s=s1, t=wc2)
@@     generate: added pair (wc2,s1) to R for the current formula
@@     WAbuild: chosen transition s1-a?->s1 from delta2
@@     conbuild: created state wc3, created transition wc2-a?->wc3
@@     generate: starting formula `true' for (s=s1, t=wc3)
@@       generate: added pair (wc3,s1) to R for the current formula
@@       generate: state wc3 marked as final
@@ Witness automaton WC1S has been constructed.
PROCESS WC1S
INITIAL STATE wc1
Number of states: 3
Number of final states: 1
FINAL STATES  wc3
Number of transitions: 2
wc1 = a? . wc2
wc2 = a? . wc3
@@ End Of Diagnostic

EF EX{b?} true ==> TRUE
@@ Creating witness automaton
@@ WCAgenerator: created empty witness automaton
@@ WCAgenerator: created initial state wc1
@@ generate: starting formula `EF EX{b?} true' for (s=s1, t=wc1)
@@   generate: added pair (wc1,s1) to R for the current formula
@@   generate: starting formula `EX{b?} true' for (s=s1, t=wc1)
@@     generate: added pair (wc1,s1) to R for the current formula
@@     WAbuild: chosen transition s1-b?->s2 from delta2
@@     conbuild: created state wc2, created transition wc1-b?->wc2
@@     generate: starting formula `true' for (s=s2, t=wc2)
@@       generate: added pair (wc2,s2) to R for the current formula
@@       generate: state wc2 marked as final
@@   WAbuild: chosen transition s1-a?->s1 from delta1
@@   conbuild: created transition wc1-a?->wc1
@@   WAbuild: chosen transition s1-b?->s2 from delta1
@@   conbuild: created state wc3, created transition wc1-b?->wc3
@@   generate: starting formula `EF EX{b?} true' for (s=s2, t=wc3)
@@     generate: added pair (wc3,s2) to R for the current formula
@@     generate: starting formula `EX{b?} true' for (s=s2, t=wc3)
@@       generate: added pair (wc3,s2) to R for the current formula
@@       WAbuild: chosen transition s2-b?->s3 from delta2
@@       conbuild: created state wc4, created transition wc3-b?->wc4
@@       generate: starting formula `true' for (s=s3, t=wc4)
@@         generate: added pair (wc4,s3) to R for the current formula
@@         generate: state wc4 marked as final
@@   WAbuild: chosen transition s1-a?->s1 from delta2
@@   conbuild: transition wc1-a?->wc1 exists
@@   WAbuild: chosen transition s1-b?->s2 from delta2
@@   conbuild: transition wc1-b?->wc3 exists
```

```
@@ Witness automaton WC2S has been constructed.
PROCESS WC2S
INITIAL STATE wc1
Number of states: 4
Number of final states: 2
FINAL STATES  wc2 wc4
Number of transitions: 4
wc1 = b? . wc2 + a? . wc1 + b? . wc3
wc3 = b? . wc4
@@ End Of Diagnostic
```

Files **example2.ccs**, **example2.actl**, and **example2.tcl** are second example of WCA generation given in paper "Witness and Counterexample Automata for ACTL" (FORTE 2004).

Here is the log from EST:

```
Efficient Symbolic Tools, 2nd Edition, Copyright (C) 2003-2013 UM-FERI
This is free software, and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute EST; type "license" for details.

Running on i686 (Linux, 3.2.0-38-generic-pae) with tcl 8.5.11 and tk 8.5.11.

Initialization of GUI package... OK
Initialization of BDD package... OK
Initialization of Process_Algebra package... OK
Initialization of Versis package... OK
Initialization of Model checking package... OK
Initialization of Strucval package... OK
Initialization of CCS package... OK
Ready.

>cd "/home/meolic/est/est-2ed/data/automata"; source "example2.tcl"; cd "/home/meolic/est/est-
2ed/data"
Reading file: example2.ccs
  Process S ... OK

ACTL/ACTLW model checking on process S

EX{a?} true ==> TRUE
Witness automaton WC1S has been constructed.
Minimization of automaton WC1S... OK
PROCESS MIN_WC1S
Number of states: 2
Number of final states: 1
Number of transitions: 1

EX{a?} EX{a?} true ==> TRUE
Witness automaton WC2S has been constructed.
Minimization of automaton WC2S... OK
PROCESS MIN_WC2S
Number of states: 3
Number of final states: 1
Number of transitions: 2

EF EX{a?} true ==> TRUE
Witness automaton WC3S has been constructed.
Minimization of automaton WC3S... OK
PROCESS MIN_WC3S
Number of states: 3
Number of final states: 1
Number of transitions: 5

E[true {a?}U{b?} true] ==> TRUE
Witness automaton WC4S has been constructed.
Minimization of automaton WC4S... OK
PROCESS MIN_WC4S
Number of states: 4
Number of final states: 1
Number of transitions: 5

AX{a?} true ==> FALSE
```

```
Counterexample automaton WC5S has been constructed.
Minimization of automaton WC5S... OK
PROCESS MIN_WC5S
Number of states: 2
Number of final states: 1
Number of transitions: 2

AX{TAU} AX{a?} true ==> FALSE
Counterexample automaton WC6S has been constructed.
Minimization of automaton WC6S... OK
PROCESS MIN_WC6S
Number of states: 2
Number of final states: 1
Number of transitions: 2

AG AX{a?} true ==> FALSE
Counterexample automaton WC7S has been constructed.
Minimization of automaton WC7S... OK
PROCESS MIN_WC7S
Number of states: 4
Number of final states: 1
Number of transitions: 7

A[true {a?}U{b?} true] ==> FALSE
Counterexample automaton WC8S has been constructed.
Minimization of automaton WC8S... OK
PROCESS MIN_WC8S
Number of states: 3
Number of final states: 1
Number of transitions: 4

PROCESS S
INITIAL STATE s9
s9 = a? . s9 + TAU . s3 + a? . s6 + c? . NIL
s3 = a? . s9
s6 = b? . NIL

PROCESS WC1S
INITIAL STATE wc1
FINAL STATES  wc2 wc3
wc1 = a? . wc2 + a? . wc3

PROCESS MIN_WC1S
INITIAL STATE s1
FINAL STATES  s2
s1 = a? . s2

PROCESS WC2S
INITIAL STATE wc1
FINAL STATES  wc3 wc4
wc1 = a? . wc2
wc2 = a? . wc3 + a? . wc4

PROCESS MIN_WC2S
INITIAL STATE s1
FINAL STATES  s3
s1 = a? . s4
s4 = a? . s3

PROCESS WC3S
INITIAL STATE wc1
FINAL STATES  wc2 wc3
wc1 = a? . wc2 + a? . wc3 + TAU . wc4 + a? . wc1
wc4 = a? . wc2 + a? . wc1

PROCESS MIN_WC3S
INITIAL STATE s1
FINAL STATES  s3
s1 = TAU . s4 + a? . s1 + a? . s3
s4 = a? . s1 + a? . s3

PROCESS WC4S
INITIAL STATE wc1
FINAL STATES  wc4
wc1 = TAU . wc2 + a? . wc1 + a? . wc3
```

```
wc2 = a? . wc1
wc3 = b? . wc4

PROCESS MIN_WC4S
INITIAL STATE s1
FINAL STATES  s5
s1 = TAU . s4 + a? . s1 + a? . s3
s4 = a? . s1
s3 = b? . s5

PROCESS WC5S
INITIAL STATE wc1
FINAL STATES  wc2 wc3
wc1 = TAU . wc2 + c? . wc3

PROCESS MIN_WC5S
INITIAL STATE s1
FINAL STATES  s2
s1 = TAU . s2 + c? . s2

PROCESS WC6S
INITIAL STATE wc1
FINAL STATES  wc2 wc3 wc4
wc1 = c? . wc2 + a? . wc3 + a? . wc4

PROCESS MIN_WC6S
INITIAL STATE s1
FINAL STATES  s3
s1 = a? . s3 + c? . s3

PROCESS WC7S
INITIAL STATE wc1
FINAL STATES  wc2 wc3 wc5 wc7
wc1 = TAU . wc2 + c? . wc3 + TAU . wc4 + c? . wc5 + a? . wc1 + a? . wc6
wc4 = a? . wc1
wc6 = b? . wc7

PROCESS MIN_WC7S
INITIAL STATE s1
FINAL STATES  s2
s1 = TAU . s2 + TAU . s5 + a? . s1 + a? . s4 + c? . s2
s5 = a? . s1
s4 = b? . s2

PROCESS WC8S
INITIAL STATE wc1
FINAL STATES  wc3
wc1 = TAU . wc2 + a? . wc1 + c? . wc3
wc2 = a? . wc1

PROCESS MIN_WC8S
INITIAL STATE s1
FINAL STATES  s2
s1 = TAU . s3 + a? . s1 + c? . s2
s3 = a? . s1
```

Files **complex.dat**, **complex.actl**, and **complex.tcl** are a complex example of ACTL model checking. For the given formula, witness and counterexample automaton cannot be created in any of the given processes.

```
SORT sortAutomata a,b,c,d,e,f,g,h,k

PROCESS Mod1
SORT sortAutomata
INITIAL STATE s0
TRANSITIONS
s0 = h?.s1
s0 = g?.s2
```

```
s1 = a!.s3
s1 = b!.s4
s2 = a!.s4
s2 = c!.s6
s3 = b!.s6
s4 = b!.s5
s5 = c!.s7
s5 = f?.s4
s6 = d!.s8
s6 = f?.s3
s7 = d!.s9
s9 = e!.s8

/*
add s8 = c!.s8
*/

PROCESS Mod2
SORT sortAutomata
INITIAL STATE s0
TRANSITIONS
s0 = h?.s1
s0 = g?.s2
s1 = a!.s3
s1 = b!.s4
s2 = a!.s4
s2 = c!.s6
s3 = b!.s6
s4 = b!.s5
s5 = c!.s7
s5 = f?.s4
s6 = d!.s8
s6 = f?.s3
s7 = d!.s9
s9 = e!.s8
s8 = c!.s8

/*
add s8 = c!.s8
remove s5 = f?.s4
*/

PROCESS Mod3
SORT sortAutomata
INITIAL STATE s0
TRANSITIONS
s0 = h?.s1
s0 = g?.s2
s1 = a!.s3
s1 = b!.s4
```

```
s2 = a!.s4
s2 = c!.s6
s3 = b!.s6
s4 = b!.s5
s5 = c!.s7
s6 = d!.s8
s6 = f?.s3
s7 = d!.s9
s9 = e!.s8
s8 = c!.s8

/*
add s8 = c!.s8
remove s5 = f?.s4
remove s6 = f?.s3
*/

PROCESS Mod4
SORT sortAutomata
INITIAL STATE s0
TRANSITIONS
s0 = h?.s1
s0 = g?.s2
s1 = a!.s3
s1 = b!.s4
s2 = a!.s4
s2 = c!.s6
s3 = b!.s6
s4 = b!.s5
s5 = c!.s7
s6 = d!.s8
s7 = d!.s9
s9 = e!.s8
s8 = c!.s8
```

Here is the log from EST:

```
Efficient Symbolic Tools, 2nd Edition, Copyright (C) 2003-2013 UM-FERI
This is free software, and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute EST; type "license" for details.

Running on i686 (Linux, 3.2.0-38-generic-pae) with tcl 8.5.11 and tk 8.5.11.

Initialization of GUI package... OK
Initialization of BDD package... OK
Initialization of Process_Algebra package... OK
Initialization of Versis package... OK
Initialization of Model checking package... OK
Initialization of Strucval package... OK
Initialization of CCS package... OK
Ready.

>cd "/home/meolic/est/est-2ed/data/automata"; source "complex.tcl"; cd "/home/meolic/est/est-
2ed/data"
Reading file: complex.dat
  Sort sortAutomata ... OK
```

```
  Process Mod1 ... OK
  Process Mod2 ... OK
  Process Mod3 ... OK
  Process Mod4 ... OK

ACTL/ACTLW model checking on process Mod1

E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{a!} E[true {(NOT a!) AND (NOT
b!) AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{b!} ((E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND
(NOT d!) AND (NOT e!)}U{e!} true]) OR (EG ((EX{c!} true) IMPL (EX{c!} E[true {(NOT a!) AND (NOT b!)
AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{d!} E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND (NOT d!)
AND (NOT e!)}U{e! OR c!} true]]))))]] ==> TRUE
Witness: (g?)(a!)(b!)(f?)(b!)
Witness automaton cannot be constructed.

ACTL/ACTLW model checking on process Mod2

E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{a!} E[true {(NOT a!) AND (NOT
b!) AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{b!} ((E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND
(NOT d!) AND (NOT e!)}U{e!} true]) OR (EG ((EX{c!} true) IMPL (EX{c!} E[true {(NOT a!) AND (NOT b!)
AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{d!} E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND (NOT d!)
AND (NOT e!)}U{e! OR c!} true]]))))]] ==> TRUE
Witness: (g?)(a!)(b!)(f?)(b!)
Witness automaton cannot be constructed.

ACTL/ACTLW model checking on process Mod3

E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{a!} E[true {(NOT a!) AND (NOT
b!) AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{b!} ((E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND
(NOT d!) AND (NOT e!)}U{e!} true]) OR (EG ((EX{c!} true) IMPL (EX{c!} E[true {(NOT a!) AND (NOT b!)
AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{d!} E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND (NOT d!)
AND (NOT e!)}U{e! OR c!} true]]))))]] ==> TRUE
Witness: (h?)(a!)(b!)(f?)(b!)
Witness automaton cannot be constructed.

ACTL/ACTLW model checking on process Mod4

E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{a!} E[true {(NOT a!) AND (NOT
b!) AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{b!} ((E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND
(NOT d!) AND (NOT e!)}U{e!} true]) OR (EG ((EX{c!} true) IMPL (EX{c!} E[true {(NOT a!) AND (NOT b!)
AND (NOT c!) AND (NOT d!) AND (NOT e!)}U{d!} E[true {(NOT a!) AND (NOT b!) AND (NOT c!) AND (NOT d!)
AND (NOT e!)}U{e! OR c!} true]]))))]] ==> FALSE
Counterexample not generated.
Counterexample automaton cannot be constructed.
```