DINING PHILOSOPHERS

(Robert Meolic, 1999, 2004, 2013)

From Wikipedia: "Five silent philosophers sit at a table around a bowl of spaghetti. A fork is placed between each pair of adjacent philosophers. Each philosopher must alternately think and eat. However, a philosopher can only eat spaghetti when he has both left and right forks. Each fork can be held by only one philosopher and so a philosopher can use the fork only if it's not being used by another philosopher. After he finishes eating, he needs to put down both forks so they become available to others. A philosopher can grab the fork on his right or the one on his left as they become available, but can't start eating before getting both of them."

Files **philo5.dat**, **philo5dat-v1.ccs**, and **philo5dat-v2.ccs** are three different specifications of the system. Their equivalence can be checked by executing **philo-test.tcl**:

```
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/filozofi"; source "philo-test.tcl"; cd "/home/meolic/est/est-
2ed/data"

============================
SPECIFICATION IN RAW FORMAT
============================

Reading file: philo5.dat
  Sort sortFilozofi ... OK
  Process FLZF1 ... OK
  Process VLC1 ... OK
  Process FLZF2 ... OK
  Process VLC2 ... OK
  Process FLZF3 ... OK
  Process VLC3 ... OK
  Process FLZF4 ... OK
  Process VLC4 ... OK
  Process FLZF5 ... OK
  Process VLC5 ... OK

Parallel composition (1): S...

============================
  SPECIFICATION IN CCS
============================

Reading file: philo5dat-v1.ccs
  Process FILOZOF ... OK
  Process VILICA ... OK
  Net S1
    Composition ... OK
    Creating process S1 ... OK

Reading file: philo5dat-v2.ccs
  Process PHILO ... OK
  Process FORK ... OK
  Net S2
```

```
    Composition ... OK
    Creating process S2 ... OK

============================
    CHECKING EQUIVALENCE
============================

Strong observational equivalence checking between S1 and S... OK
Strong observational equivalence checking between S2 and S... OK
Strong observational equivalence checking between S1 and S2... OK
```

Here are some data about timing on Intel Atom D525 with 2GB RAM:

```
[CPU]versis_compose 1 S {FLZF1 VLC1 FLZF2 VLC2 FLZF3 VLC3 FLZF4 VLC4 FLZF5 VLC5}
{think1 think2 think3 think4 think5 eat1 eat2 eat3 eat4 eat5}[CPU: 160ms]
[CPU]ccs_read philo5dat-v1.ccs[CPU: 1191ms]
[CPU]ccs_read philo5dat-v2.ccs[CPU: 1208ms]
[CPU]versis_strong_equivalence 0 S1 1 S[CPU: 73634ms]
[CPU]versis_strong_equivalence 0 S2 1 S[CPU: 80076ms]
[CPU]versis_strong_equivalence 0 S1 0 S2[CPU: 79259ms]
```

Files **philo2.ccs**, **philo3.ccs**, ..., **philo8.ccs** use the style from **philo5dat-v2.ccs**, however, action "hungry" was added to make the verification more elegant. Here is the description of system with 5 philosophers from philo5.ccs:

```
PHILO = !think;!hungry;!takeleft;!takeright;!eat;!dropleft;!dropright;PHILO
FORK = ?take;?drop;FORK

net TABLE5 = // (PHILO [take5/takeleft]
                      [take1/takeright]
                      [eat1/eat] [think1/think] [hungry1/hungry]
                      [drop5/dropleft]
                      [drop1/dropright],

                FORK  [take1/take]
                      [drop1/drop],

                PHILO [take1/takeleft]
                      [take2/takeright]
                      [eat2/eat] [think2/think] [hungry2/hungry]
                      [drop1/dropleft]
                      [drop2/dropright],

                FORK  [take2/take]
                      [drop2/drop],

                PHILO [take2/takeleft]
                      [take3/takeright]
                      [eat3/eat] [think3/think] [hungry3/hungry]
                      [drop2/dropleft]
                      [drop3/dropright],

                FORK  [take3/take]
                      [drop3/drop],

                PHILO [take3/takeleft]
                      [take4/takeright]
                      [eat4/eat] [think4/think] [hungry4/hungry]
                      [drop3/dropleft]
                      [drop4/dropright],
```

```
            FORK  [take4/take]
                  [drop4/drop],

            PHILO [take4/takeleft]
                  [take5/takeright]
                  [eat5/eat] [think5/think] [hungry5/hungry]
                  [drop4/dropleft]
                  [drop5/dropright],

            FORK  [take5/take]
                  [drop5/drop]

           ) \take1\drop1
             \take2\drop2
             \take3\drop3
             \take4\drop4
             \take5\drop5
```

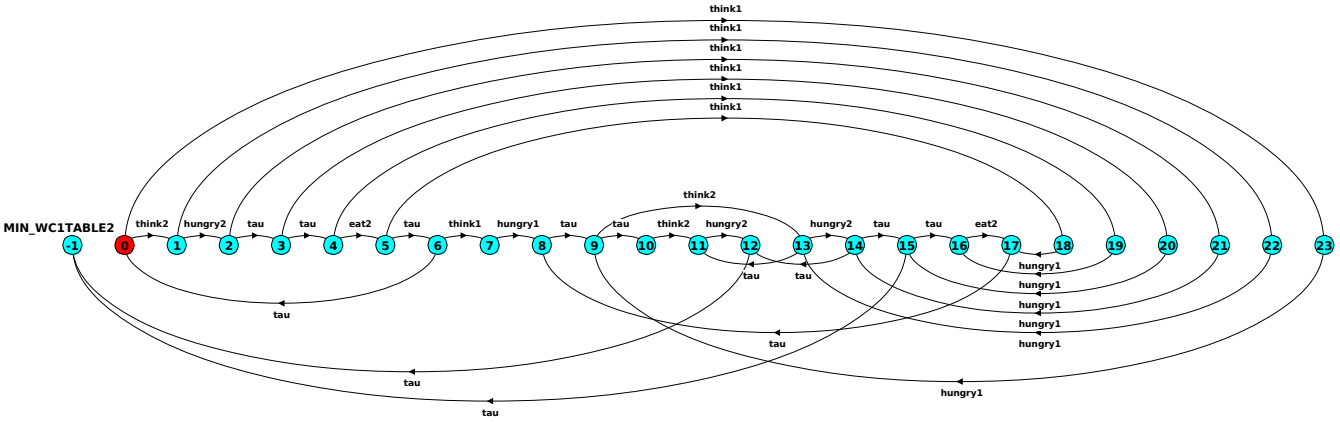
Here is log from executing **philo.tcl**:

```
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/filozofi"; source "philo.tcl"; cd "/home/meolic/est/est-2ed/data"

===========================
    SPECIFICATION IN CCS
===========================

Reading file: philo2.ccs
  Process PHILO ... OK
  Process FORK ... OK
  Net TABLE2
    Composition ... OK
    Creating process TABLE2 ... OK

Reading file: philo3.ccs
  Process PHILO ... OK
  Process FORK ... OK
  Net TABLE3
    Composition ... OK
    Creating process TABLE3 ... OK

Reading file: philo4.ccs
  Process PHILO ... OK
  Process FORK ... OK
  Net TABLE4
    Composition ... OK
    Creating process TABLE4 ... OK

Reading file: philo5.ccs
  Process PHILO ... OK
  Process FORK ... OK
  Net TABLE5
    Composition ... OK
    Creating process TABLE5 ... OK
```

```
PROCESS TABLE2
Number of states: 34
Number of transitions: 60

PROCESS TABLE3
Number of states: 199
Number of transitions: 522

PROCESS TABLE4
Number of states: 1174
Number of transitions: 4116

PROCESS TABLE5
Number of states: 6874
Number of transitions: 30120

============================
      MODEL CHECKING
============================

ACTL/ACTLW model checking on process TABLE2 using file philo.actl

AAX {false} OR EEF AAX {false} ==> TRUE

(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})) ==> FALSE

AAG [hungry1!] AAF {eat1!} true ==> FALSE

AAG [hungry2!] AAF {eat2!} true ==> FALSE

ACTL/ACTLW model checking on process TABLE3 using file philo.actl

AAX {false} OR EEF AAX {false} ==> TRUE

(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})) ==> FALSE

AAG [hungry1!] AAF {eat1!} true ==> FALSE

AAG [hungry2!] AAF {eat2!} true ==> FALSE

ACTL/ACTLW model checking on process TABLE4 using file philo.actl

AAX {false} OR EEF AAX {false} ==> TRUE

(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})) ==> FALSE

AAG [hungry1!] AAF {eat1!} true ==> FALSE

AAG [hungry2!] AAF {eat2!} true ==> FALSE

ACTL/ACTLW model checking on process TABLE5 using file philo.actl

AAX {false} OR EEF AAX {false} ==> TRUE

(EEX {true} AND EEG {TAU} EEX {true}) OR (EEF (EEX {true} AND EEG {TAU} EEX {true})) ==> FALSE

AAG [hungry1!] AAF {eat1!} true ==> FALSE

AAG [hungry2!] AAF {eat2!} true ==> FALSE

==================================
WITNESS AND COUNTEREXAMPLE AUTOMATA
==================================

ACTL/ACTLW model checking on process TABLE2
AAG [hungry1!] AAF {eat1!} true ==> FALSE
@@ Diagnostics
@@ #0:AAG [hungry1!] AAF {eat1!} true:F:(TABLE22)
@@ There exist a path not satisfying formula #0: (TABLE22)--think1!->(TABLE229)
@@ #1:[hungry1!] AAF {eat1!} true:F:(TABLE229)
@@ There exist a transition not satisfying formula #1: (TABLE229)--hungry1!->(TABLE212)
@@ #2:AAF {eat1!} true:F:(TABLE212)
@@ There exist a path not satisfying formula #2: (TABLE212)--think2!->(TABLE219)--hungry2!-
>(TABLE213)--TAU->(TABLE214)--TAU->(TABLE216)--eat2!->(TABLE215)--TAU->(TABLE211)--TAU->(TABLE212)
```

```
@@ Counterexample: (think1!)(hungry1!)(think2!)(hungry2!)(TAU)(TAU)(eat2!)(TAU)(TAU)
@@ End Of Diagnostic

ACTL/ACTLW model checking on process TABLE2
AAG [hungry1!] AAF {eat1!} true ==> FALSE
Counterexample automaton WC1TABLE2 has been constructed.
Minimization of automaton WC1TABLE2... OK
PROCESS MIN_WC1TABLE2
Number of states: 25
Number of final states: 1
Number of transitions: 35
```

Here is automaton MIN_WC1TABLE2 visualised with LTSA V3.0:



Because all final states in the automaton are deadlocked states (in fact, there is only one such state), a weak observational equivalent process is also meaningful (the first version is minimised with LTSA, the second / smaller one is minimised by EST, they are weak observational equivalent but not strong observational equivalent!).