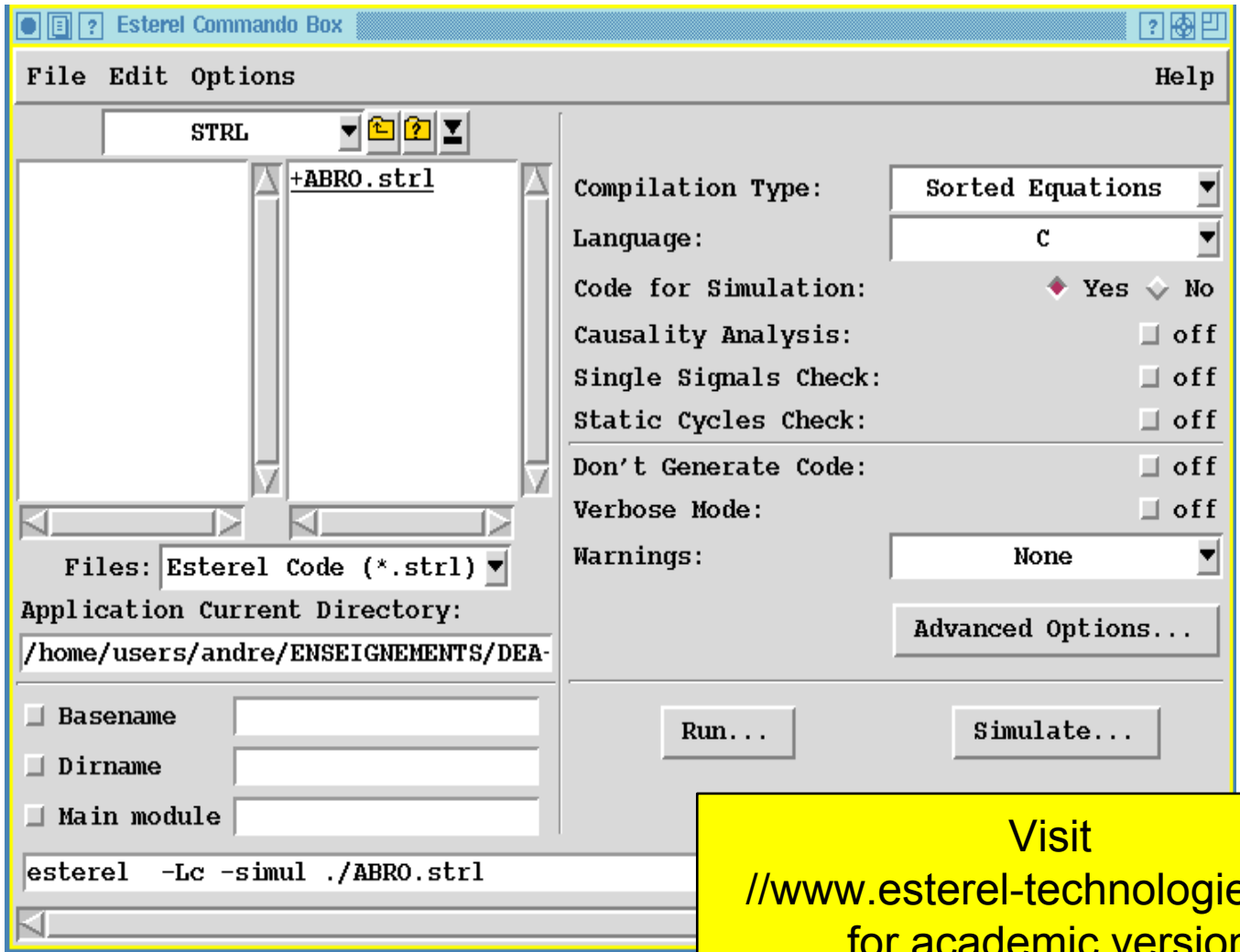
Esterel

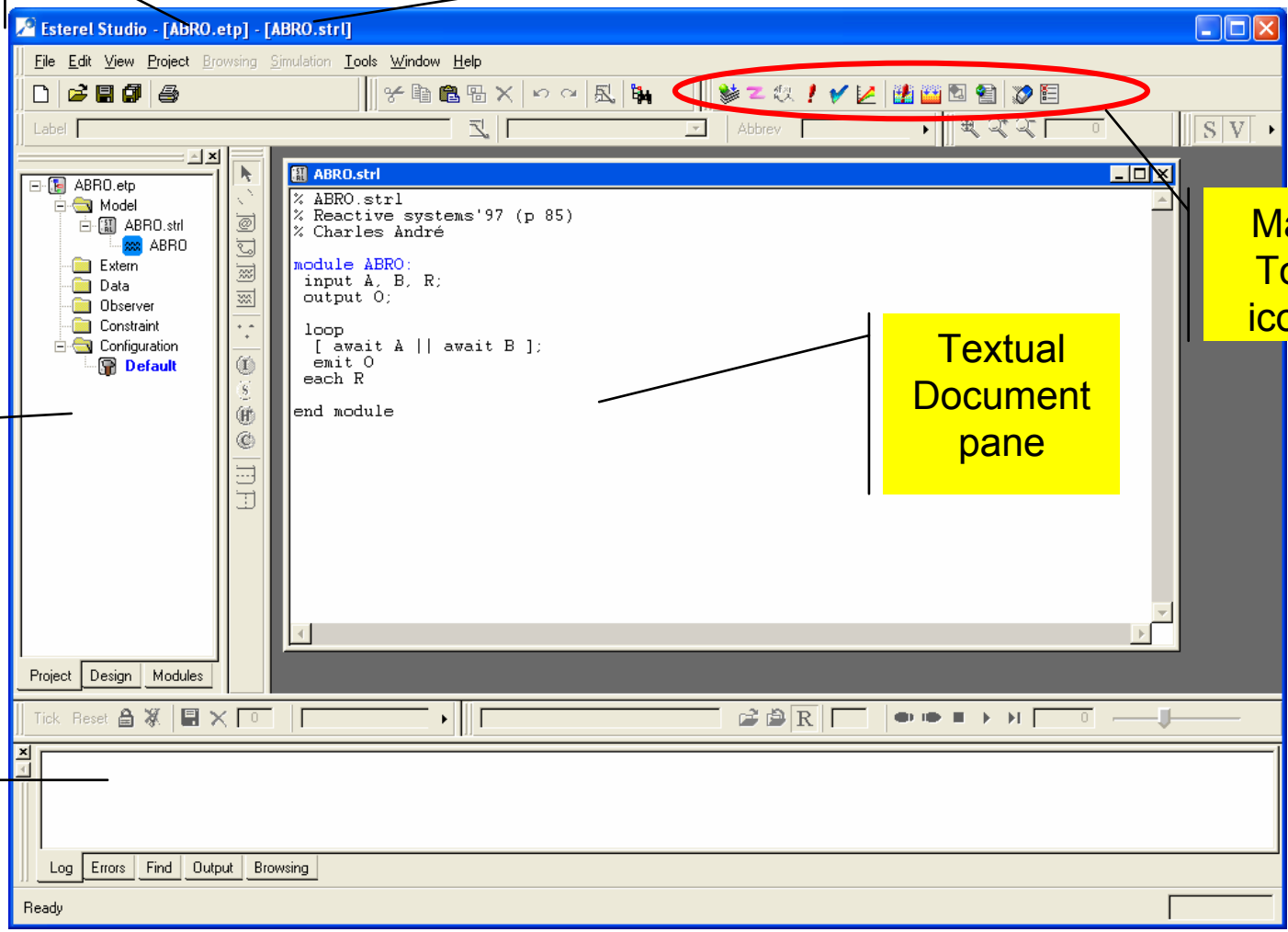
Software
Environment

xesterel GUI(v5_9x)



Visit
//www.esterel-technologies.com
for academic version

Esterel Studio



Project name

Strl file name

Main Tool icons

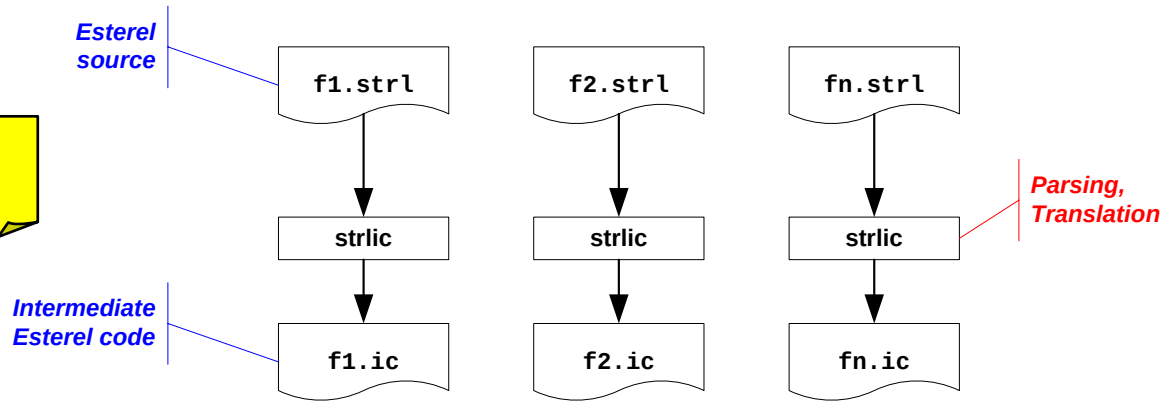
Tree pane

Textual Document pane

console

Compilation chain (1)

for Esterel v5_9x

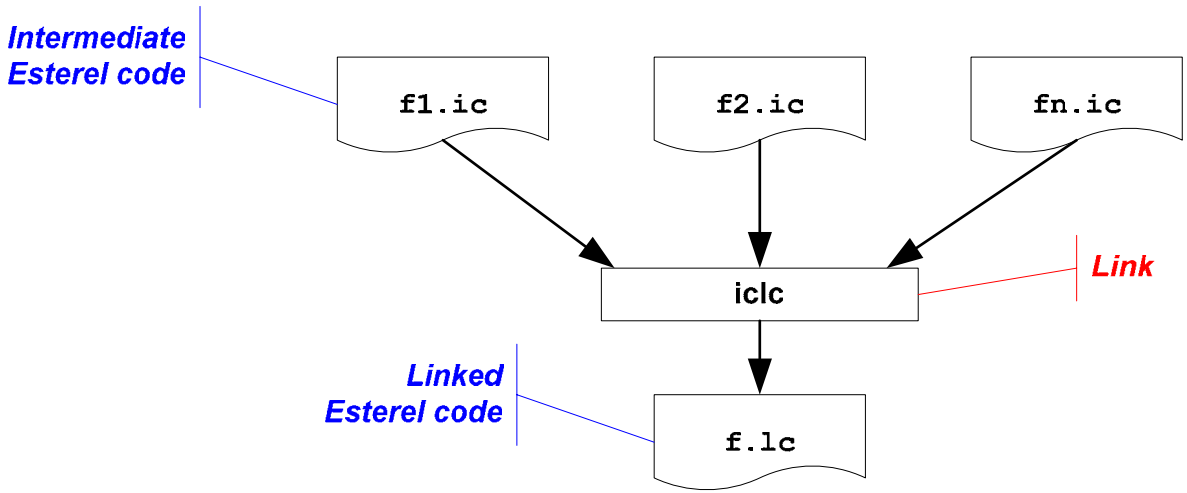


- Intermediate Esterel code (ic)

  Each source module produces an intermediate code module.

  An ic module contains tables and a set of statements written in an imperative parallel kernel code (almost kernel Esterel)

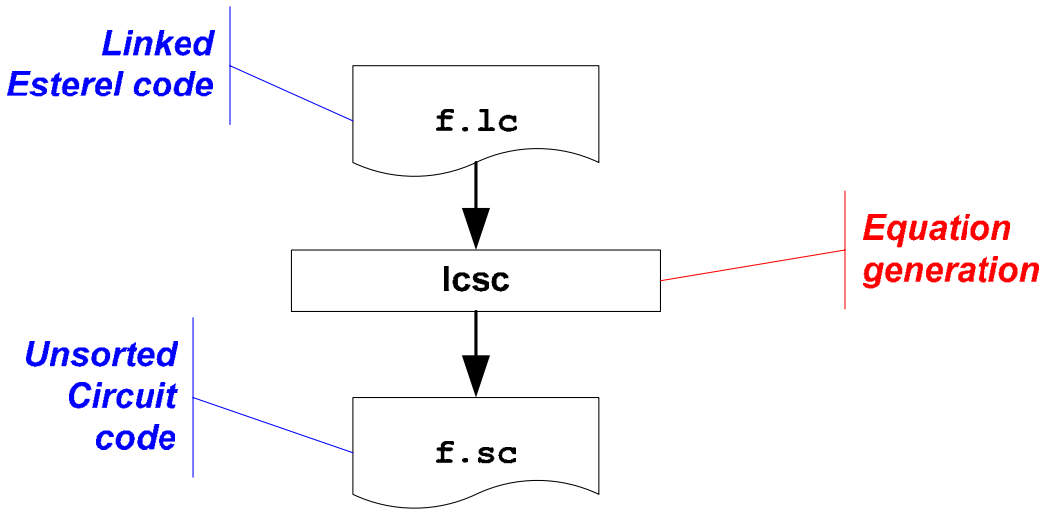
Compilation chain (2)



- Linked code (lc)

  Links the ic modules and resolves open calls.

Compilation chain (3)

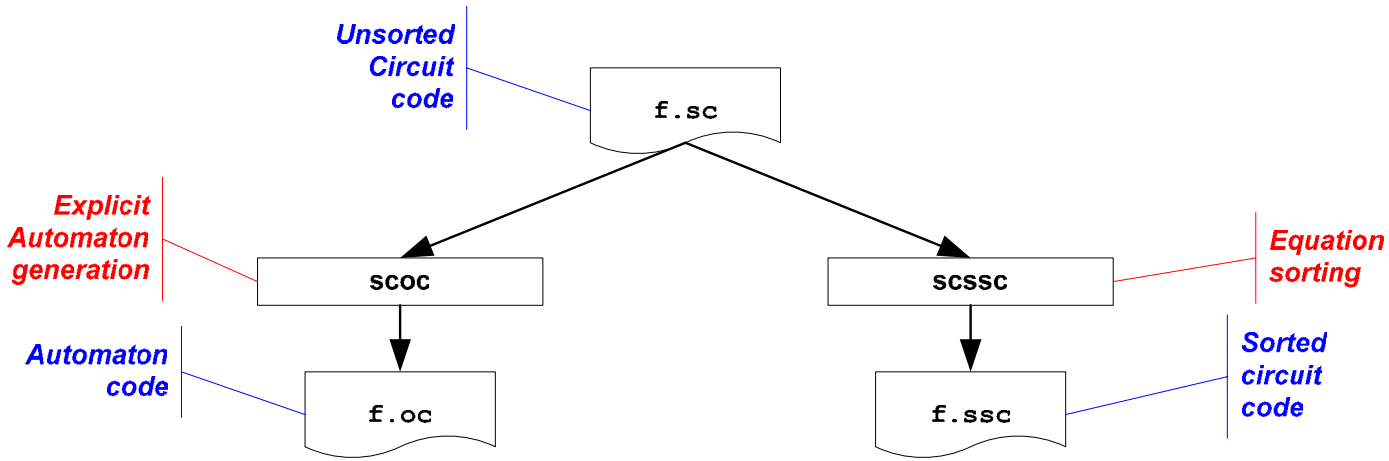


- Unsorted circuit code (sc)

  Contains the same tables as ic codes + kernel statements are translated into equations of a Boolean circuit.

  Equations are unsorted.

Compilation chain (4)



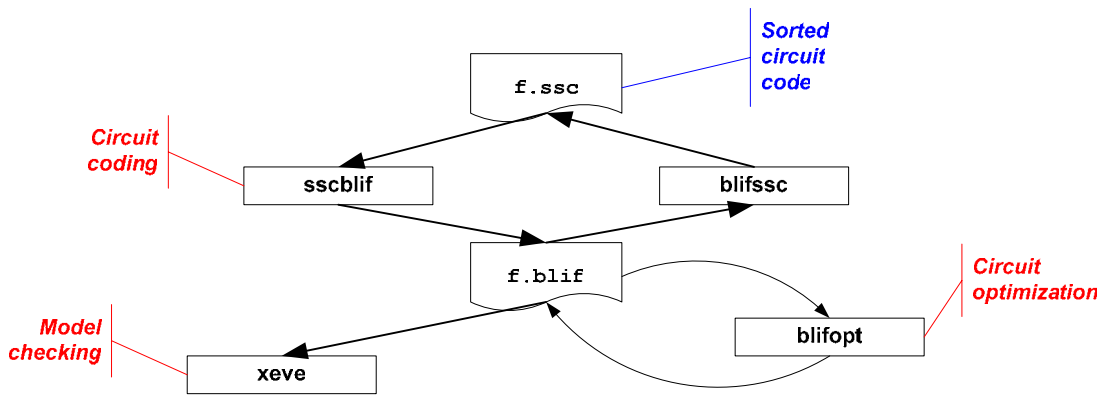
- Automaton code (oc)

  Contains the same tables as ic codes + state / transition tables

- Sorted circuit code (ssc)

  Contains the same tables as ic codes + equations sorted and printed in topological order

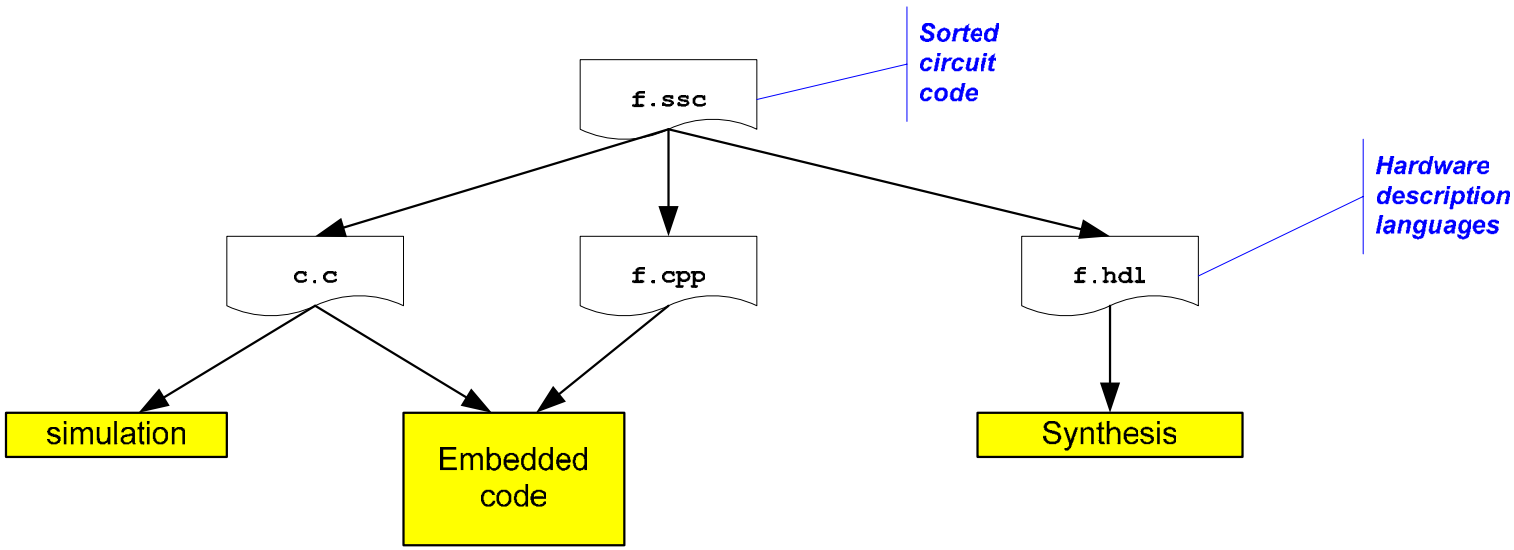
Compilation chain (5)



- Blif circuit description (blif)

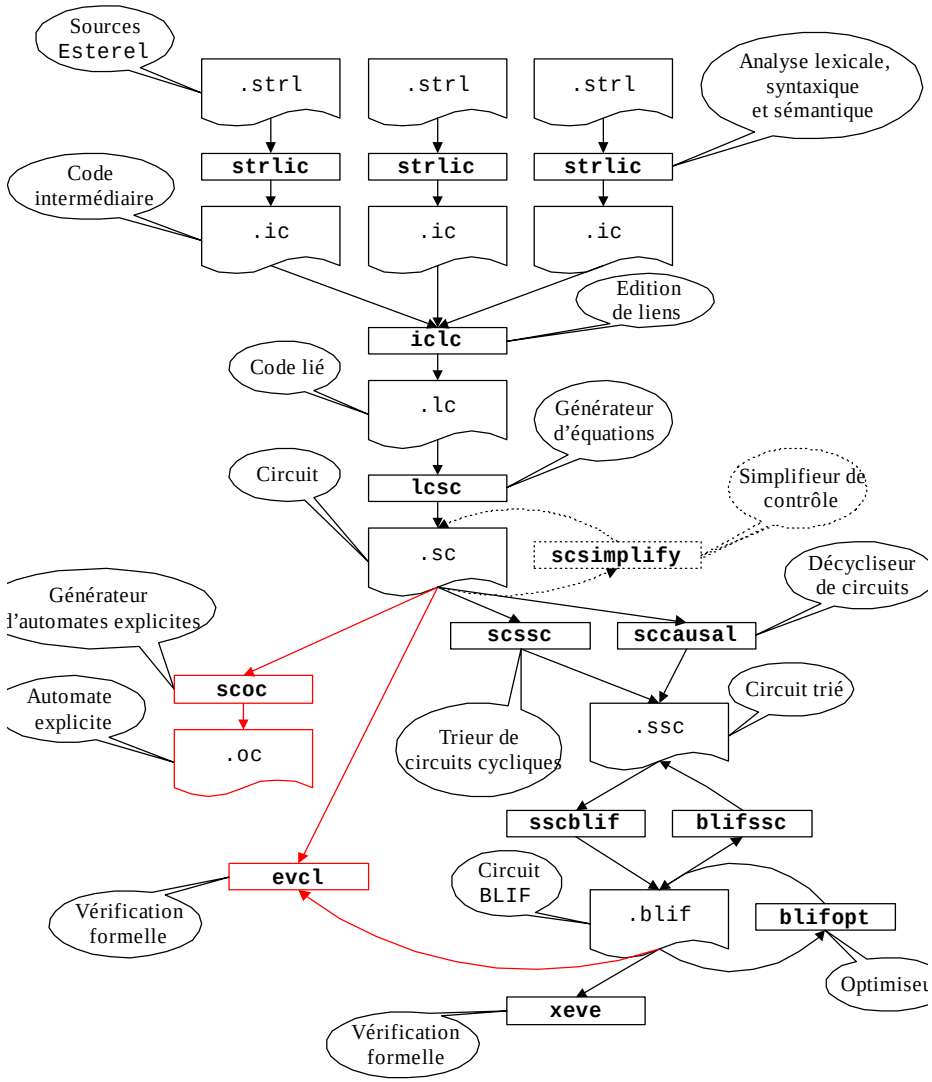
  Blif = Berkeley Logical Interchange Format
- Formal verification
- Optimization

Compilation chain (6)



Compilation chain (7)

overview

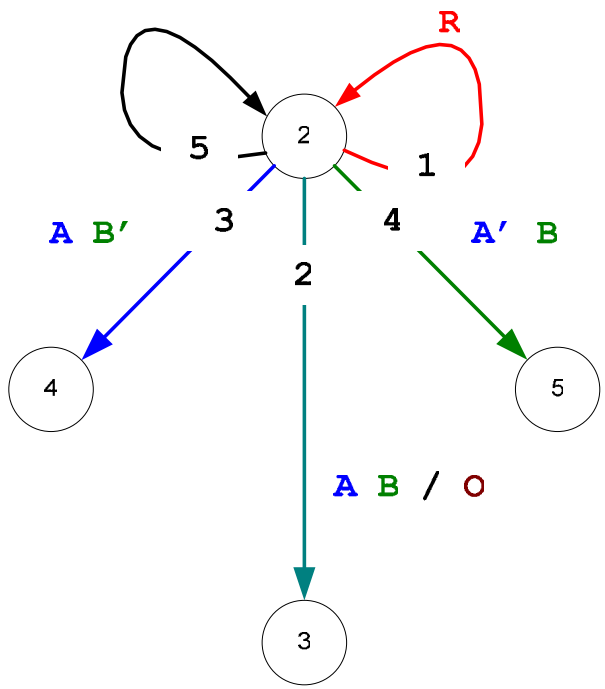


oc Format

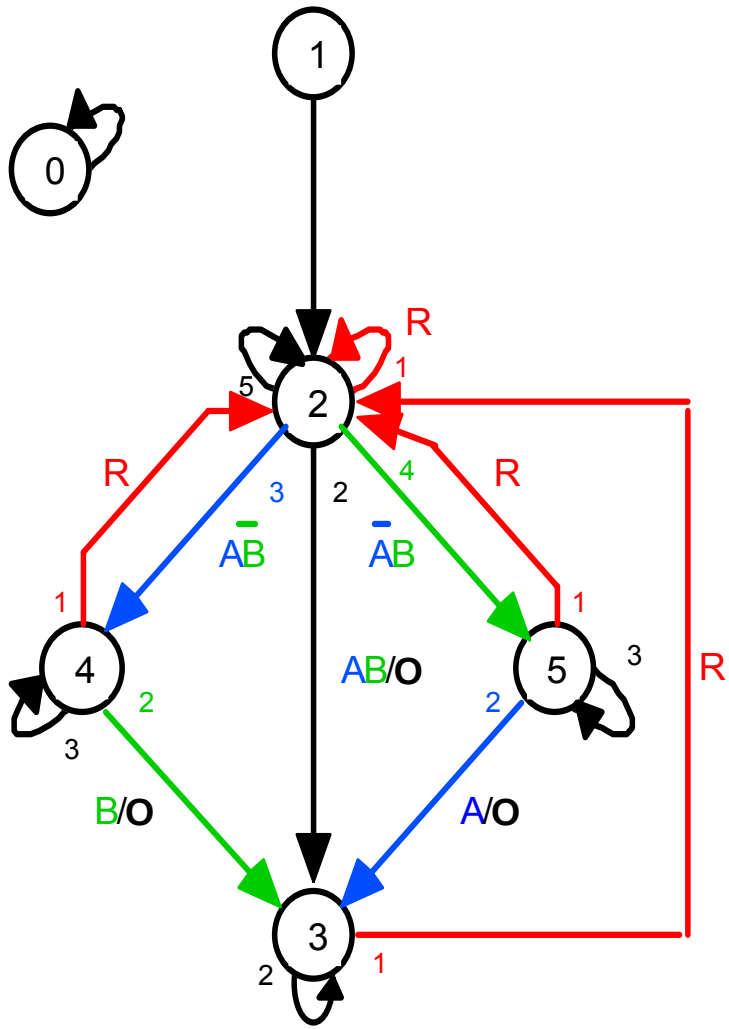
Excerpts from ABRO.oc (debug format)

```
2.2 Present signal tests
      A4 : V0 (signal A)
      A5 : V1 (signal B)
      A6 : V2 (signal R)
2.3 Output actions
      A7 : O

State 2
      if A6 then
            goto state 2
      end;
      if A4 then
            if A5 then
                  A7
                  goto state 3
            end;
            goto state 4
      end;
      if A5 then
            goto state 5
      end;
      goto state 2
```



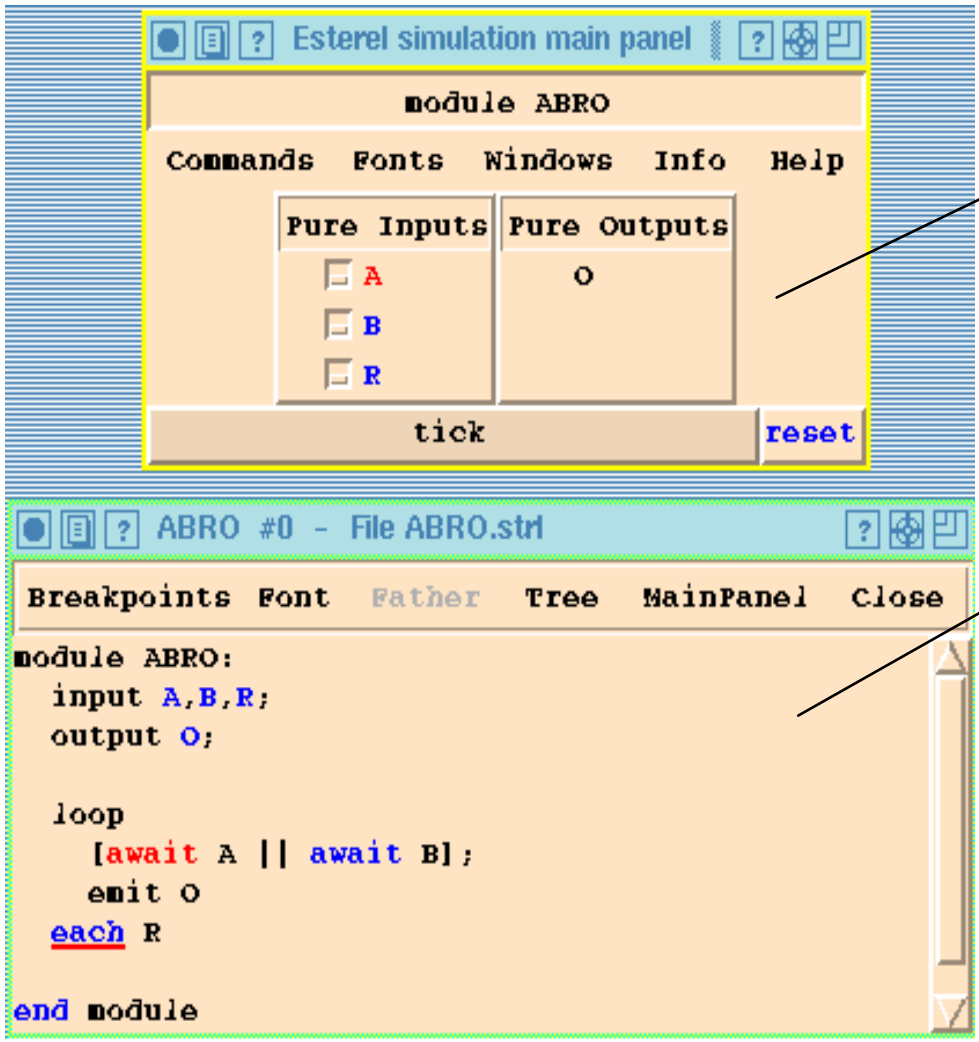
ABRO: Automaton



Reactive API (1)

- Interactive simulation (C language)
  - Textual
  - GUI (Xes in v5, ES now)
- Co-simulation (use of a user's defined simulation environment)
- Direct execution (Embedded code)

XES Interactive Simulation

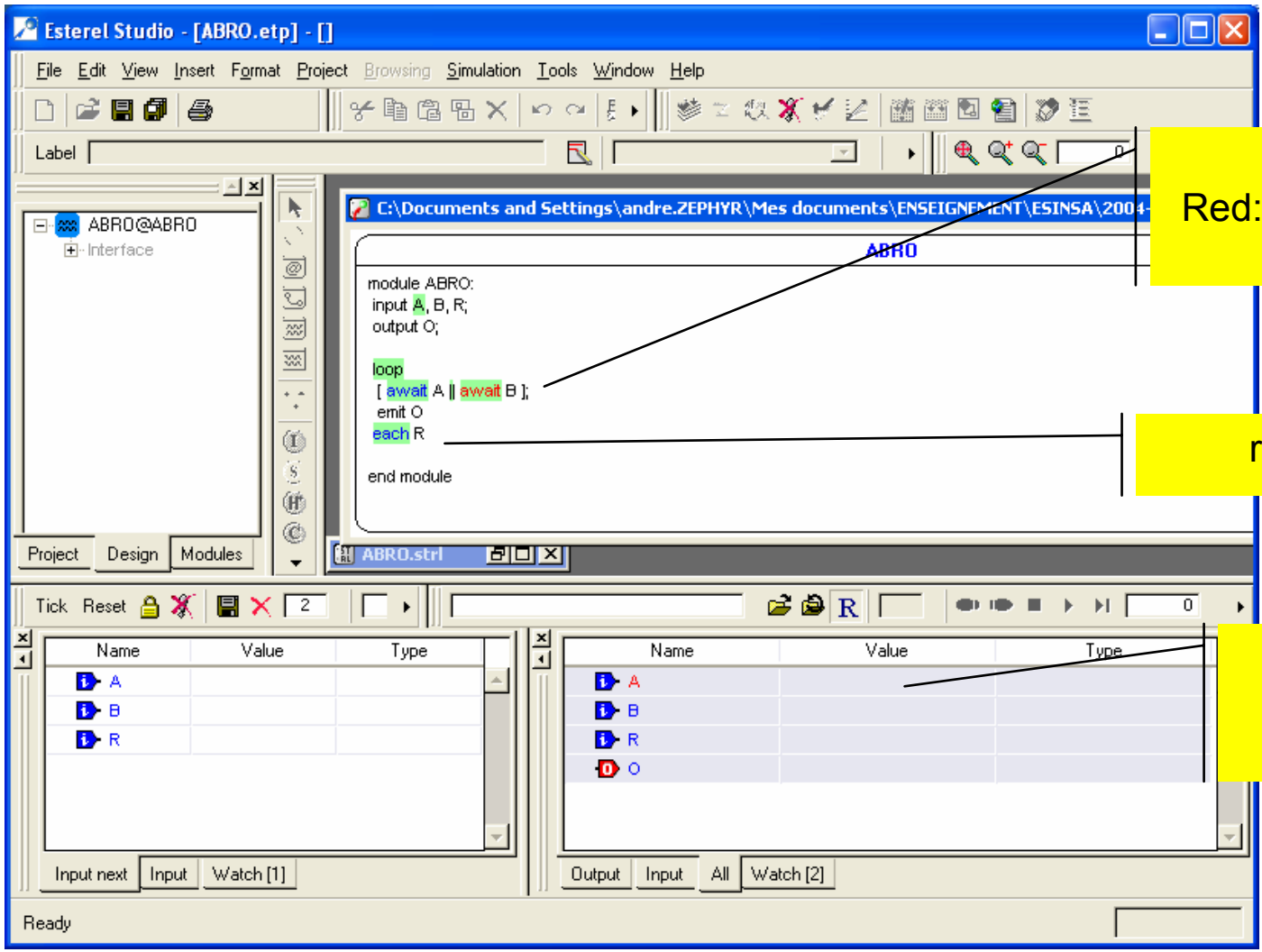


Signal Control panel

Source code animation

Tcl-Tk-based

E.S Interactive simulation



control:
Red: active control point

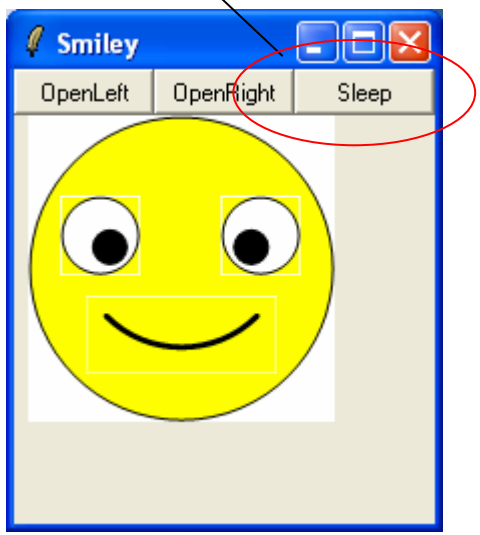
receptive

Signals:
Present = red
Absent = blue

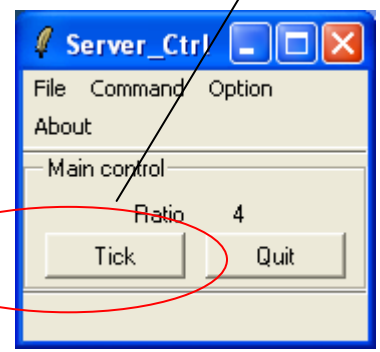
My Interactive Animation

Click on button "Sleep"

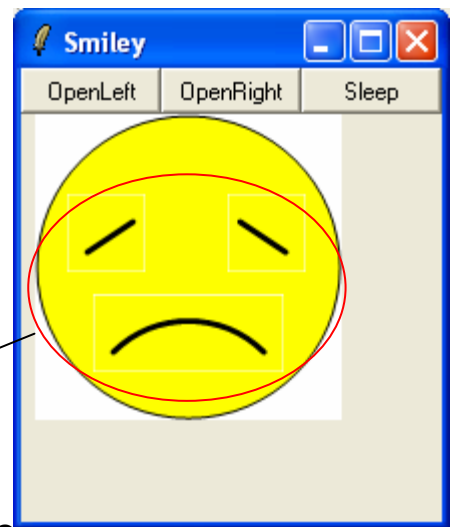
11



12



Make a "Tick"



Reaction

Tcl-Tk-based

Reactive API (2)

- Some C or C++ functions are automatically generated
- Others are to be provided by the user.

Reactive API (3)

- Reaction function – provided **`foo();`**
- Input functions – provided
  `foo_I_<sig-id>(<type>);`
- Output functions – to be provided
  `foo_O_<sig-id>(<type>);`
- Sensor functions, - to be provided
  `<type> foo_S_<sig-id>();`

Reactive API (4)

- Example: for ABRO
- **ABRO();** provided
- **ABRO_I_A();** provided
- **ABRO_I_B();** provided
- **ABRO_I_R();** provided
- **ABRO_O_O() {**

```
        printf("\a");/* bell */
    }
```

Reactive API (5)

- Non predefined data type processing: to be provided
- Declarations in **`foo.h`**

  types
  constants
  functions …
- Definitions in **`foo_data.c`**

Reactive API (6)

- An example of user's defined type
- In Esterel code: **`type TIME;`**
- In C code (in foo.h) :

```
typedef struct {
        int mn;
        int s;
} TIME;
```

Reactive API (7)

- Access functions
- Assignment - to be provided

```
_<type>(<type>*,<type>);
```

```
#define _<type>(x,y) (*(x)=y)
```

- Equality test – to be provided*

```
_eq_<type>(<type>,<type>);
```

- Inequality test – to be provided*

```
_ne_<type>(<type>,<type>);
```

Reactive API (8)

- Example: TIME
- ```
  int _eq_TIME (TIME a, TIME b)
          {
              return ( (a.mn == b.mn)
                       && (a.s ==b.s));
          }
  ```
- ```
  int _ne_TIME (TIME a, TIME b)
          {   return !_eq_TIME(a,b); }
  ```

Reactive API (9)

- Conversions <type> ↔ string
- <type> → string :

```
char * _<type>_to_text(<type>)
```

- string → <type> :

```
void _text_to_<type>(<type>*,char*);
```

- syntatic checking:

```
int _check_<type>(char *);
```

Reactive API (10)

```
int _check_TIME (char *str) {
    int m,s;
    return ((sscanf(str,"%d:%d",&m,&s)
                        ==2?1,0));  }
void _text_to_TIME (TIME*pt,char *s) {
    if (_check_TIME(s))
          sscanf(s,"%d:%d",&(pt->mn),
          &(pt->s));
    else {
          fprintf(stderr,"%s:illegal TIME\n",s);
          pt->mn = pt->s = 0;
          }
    }
char * _TIME_to_text (TIME t) {
    static char Buffer[9]="";
    sprintf(Buffer,"%02d:%02d",t.mn,t.s);
    return Buffer;    }
```

Reactive API (11)

- Constants
- In Esterel code:

```
constant n:integer, Duration:TIME;
```

- In C code :

```
/* in foo.h */

#define n 12
```

or

```
/* in foo_data.c */
int n = 12;

TIME Duration = {5,30};
```

Reactive API (12)

- Functions
- In Esterel code:

```
function TIME_ADD(TIME,TIME):TIME;
```

- In C code:

```
TIME TIME_ADD(TIME a, TIME b) {
     TIME t = a;
     if ((t.s += b.s) > 59) {
          t.s -=60;
          t.mn++;
     }
     if ((t.mn += b.mn)>59) t.mn-=60;
     return t;
}
```

Reactive API (13)

- Procedure
- In Esterel code:

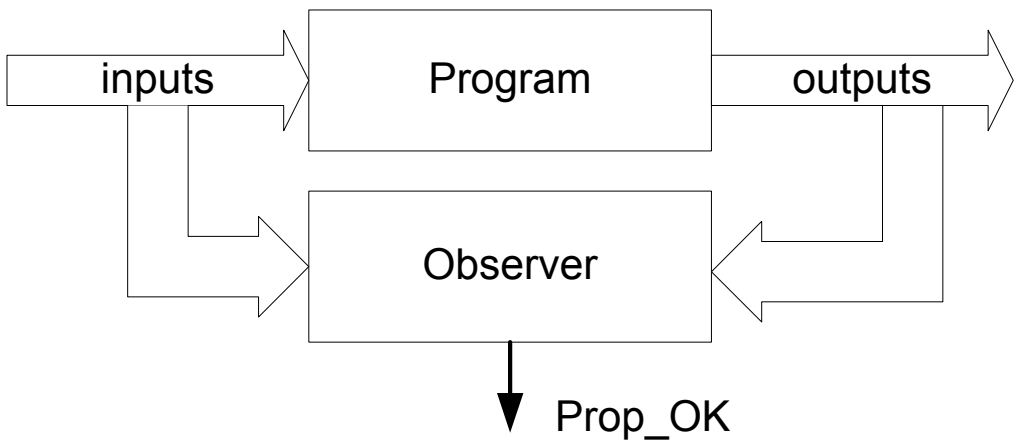
```
procedure Incr_TIME(TIME)(integer);
```

- In C code:

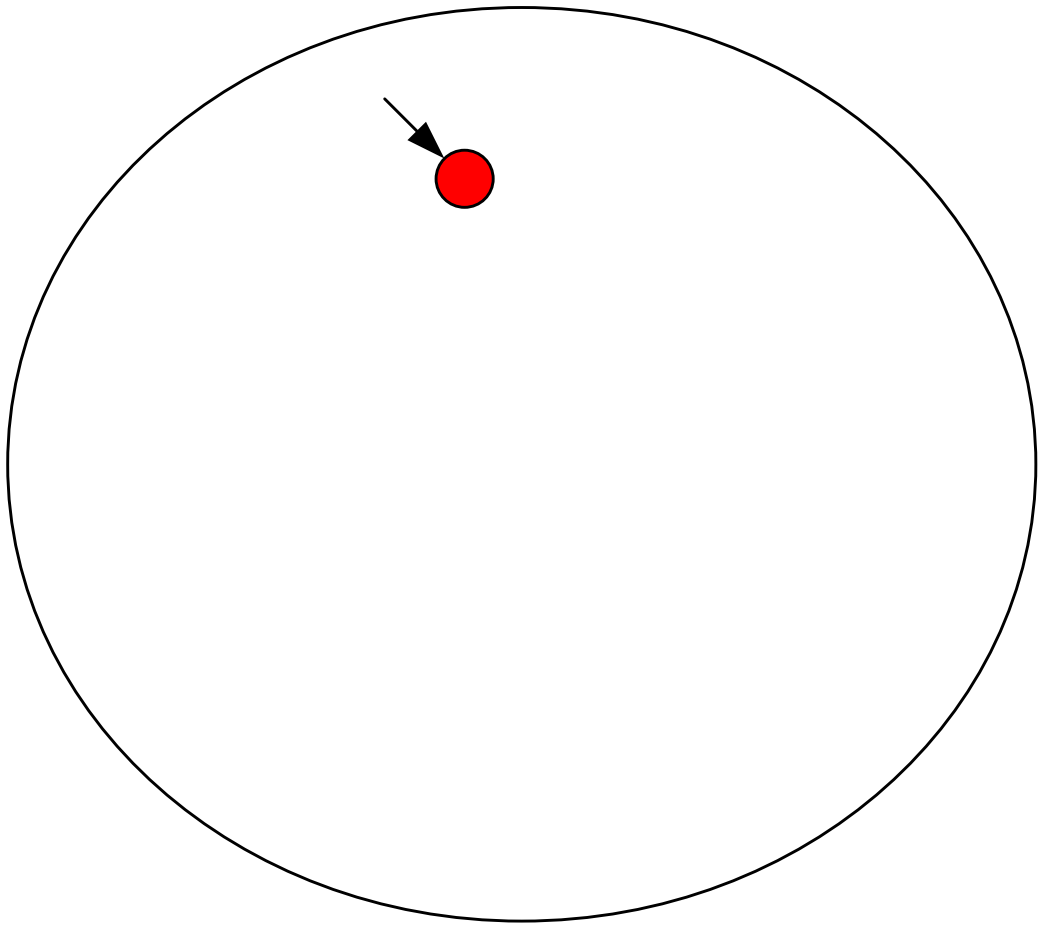
```
void Incr_TIME(TIME *pt, int n) {
     int s = pt -> s;
     int mn = pt -> mn;
     s += n;
     pt -> s = (s%60);
     mn += (s/60);
     pt -> mn = (mn%60);
}
```

Validation

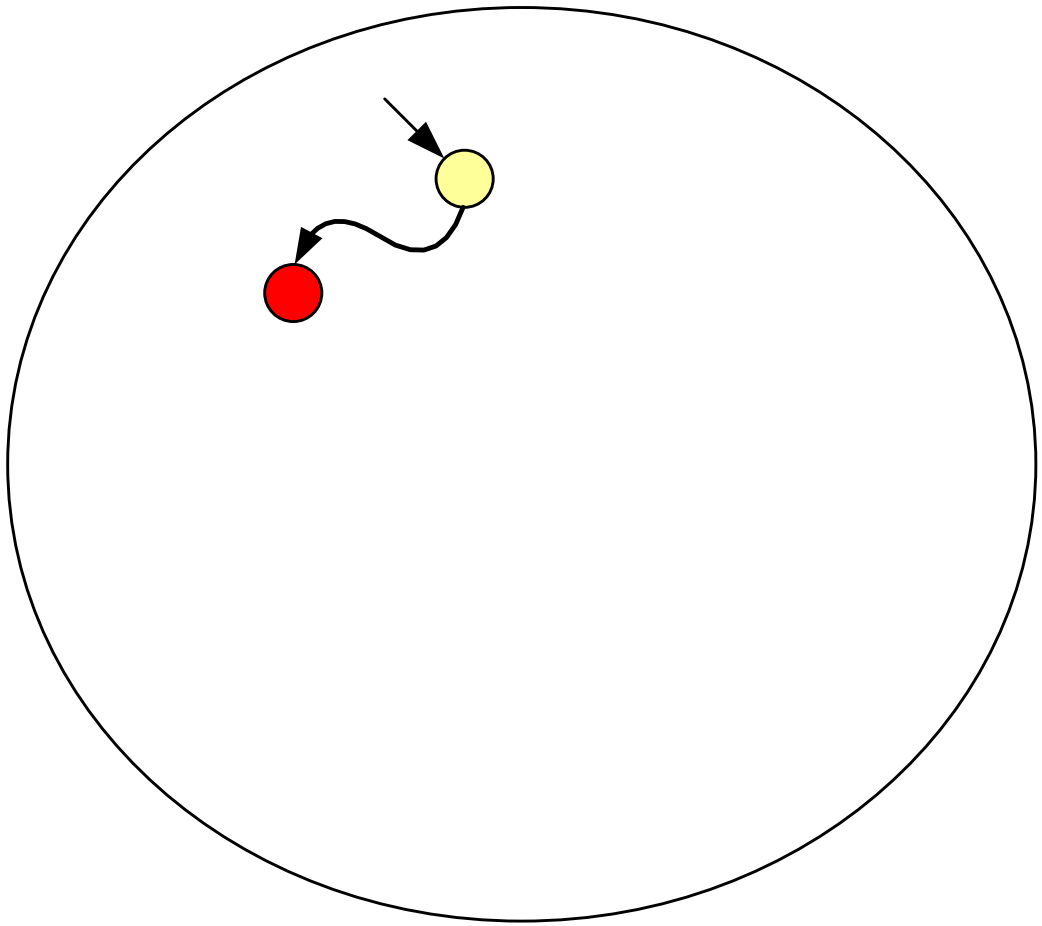
- Principle: compose your controller with an observer of the property. (parallel composition)
- In practice: It is easier to express the counter-positive of the property P and check that ¬P is never satisfied. If P is violated then generate a counter-example.



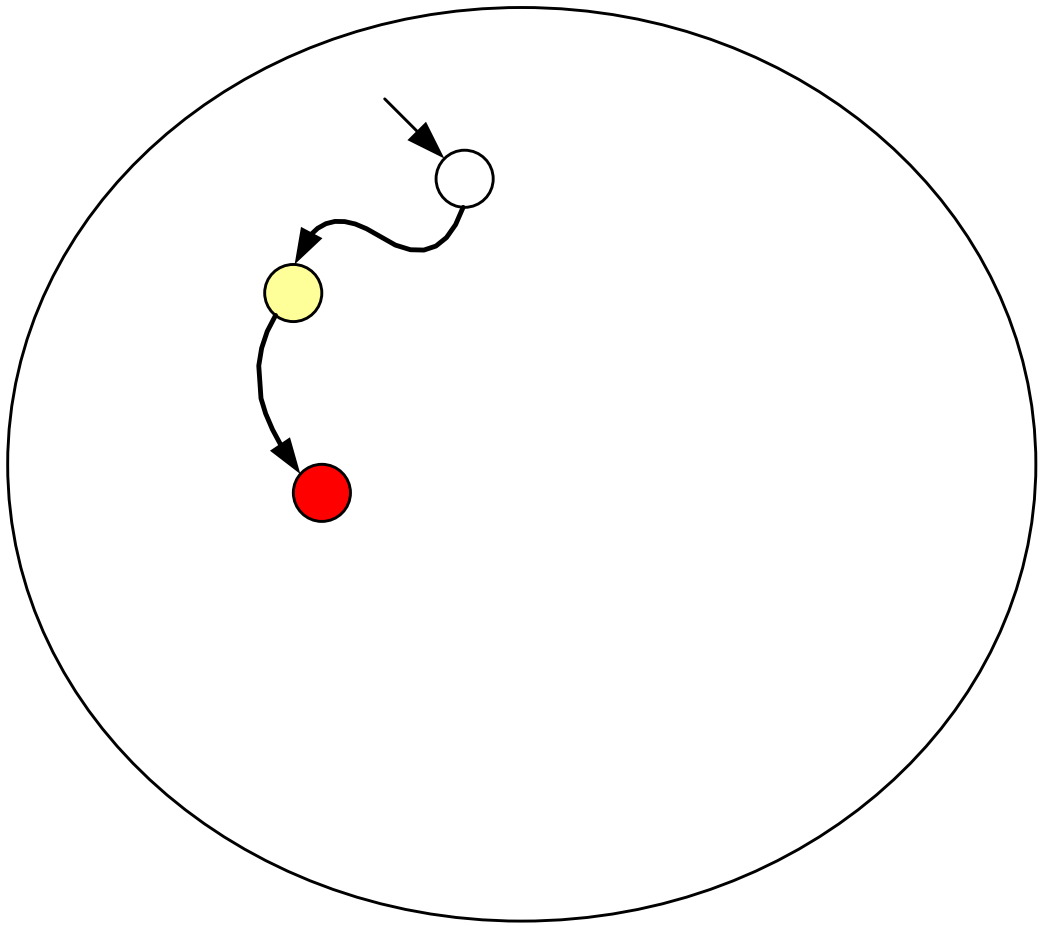
Reachable states (1)



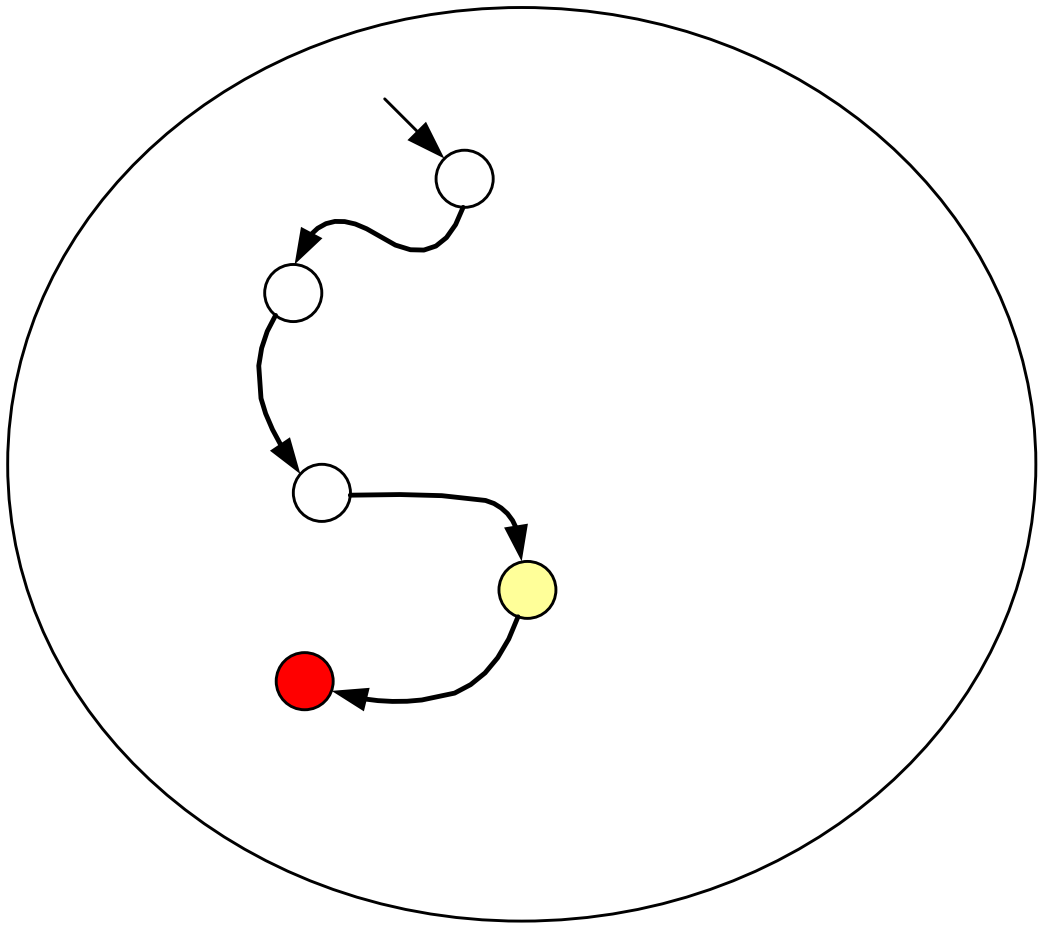
Reachable states (1)



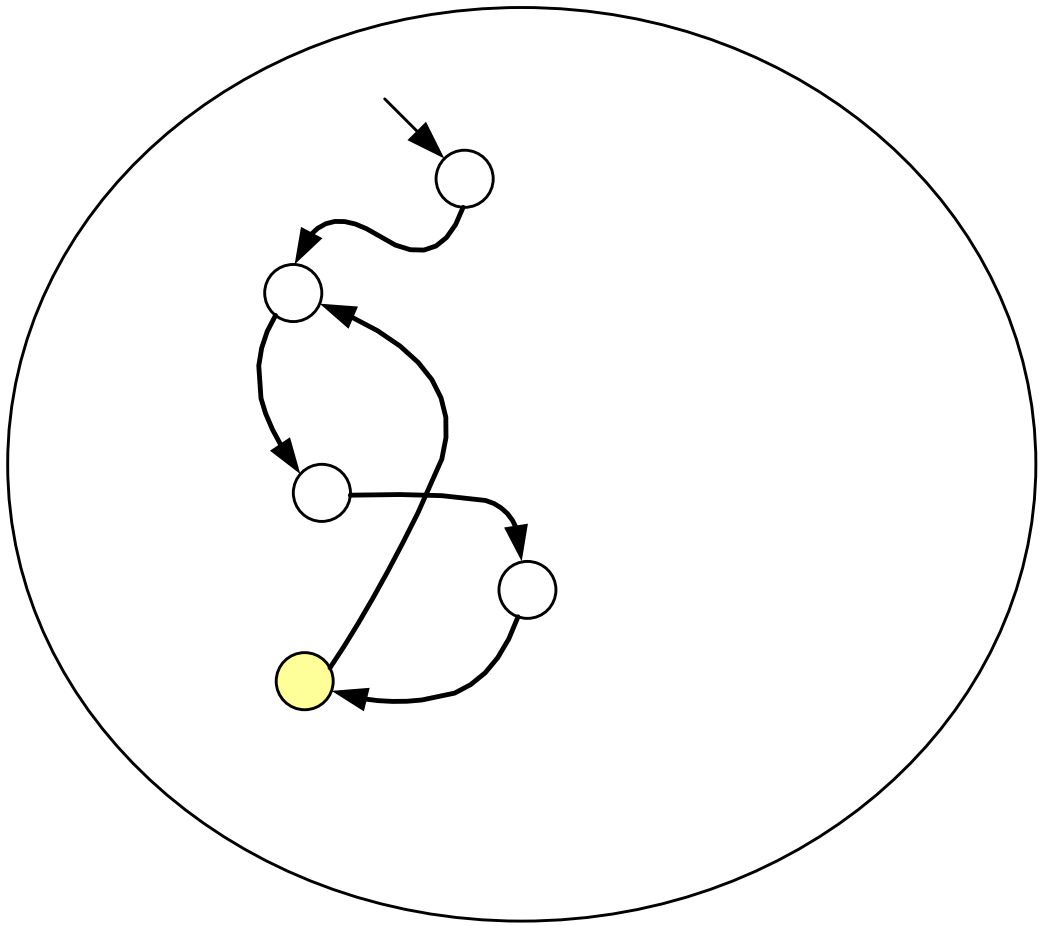
Reachable states (1)



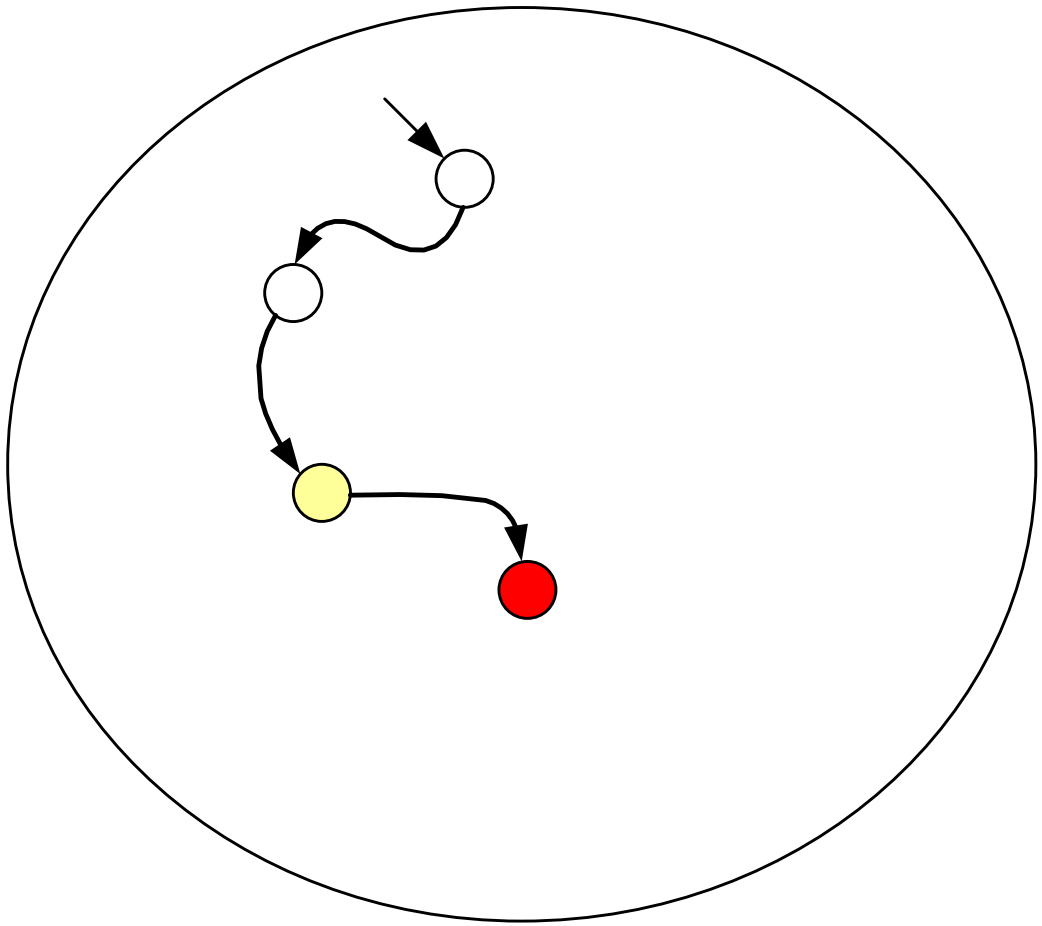
Reachable states (1)



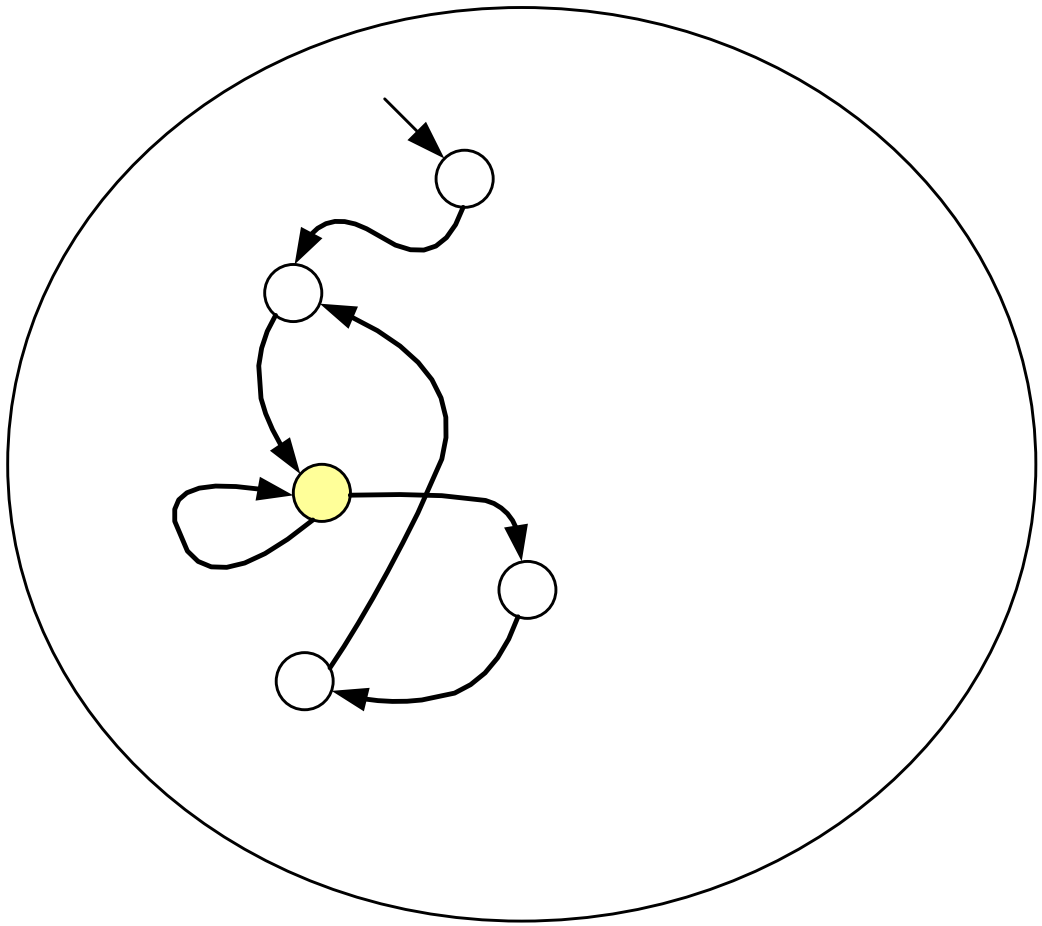
Reachable states (1)



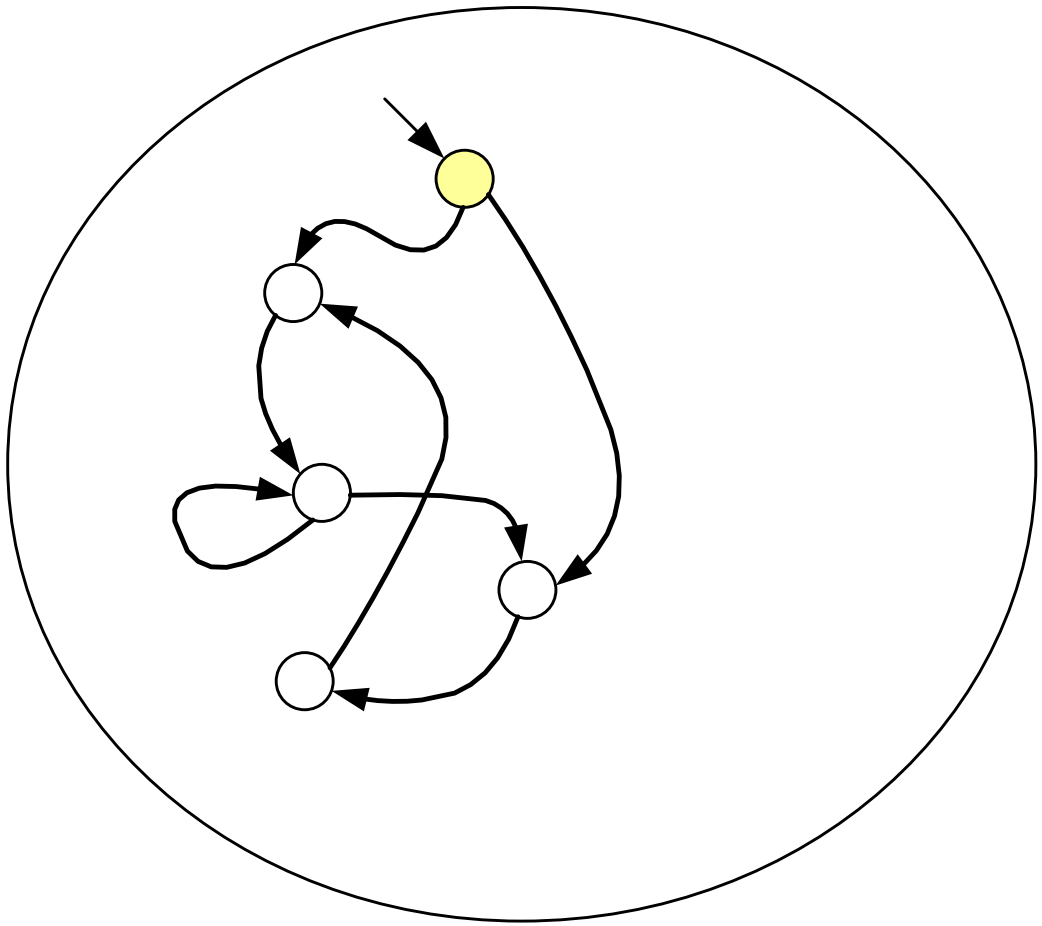
Reachable states (1)



Reachable states (1)

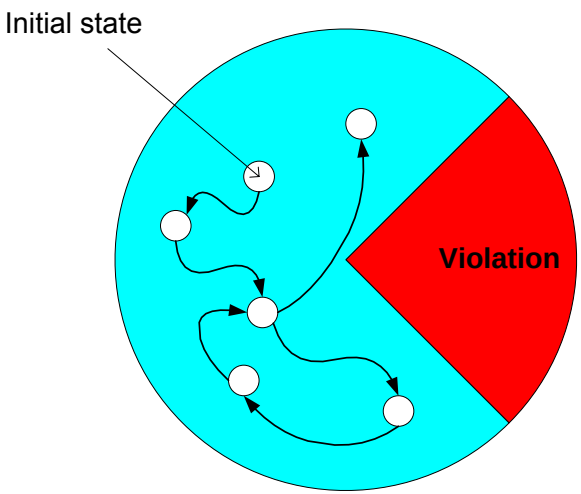


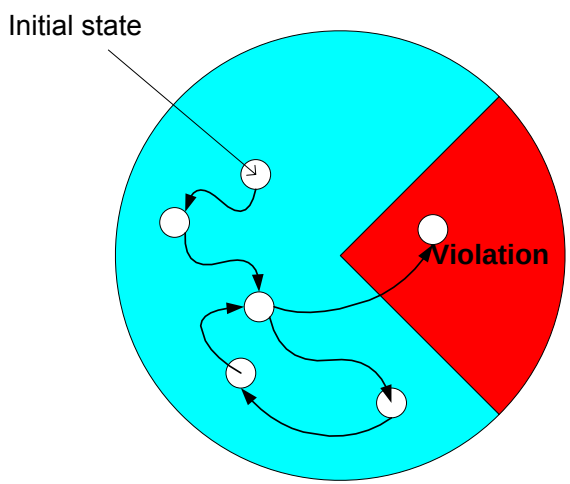
Reachable states (1)



Model-checking (1)

State enumeration

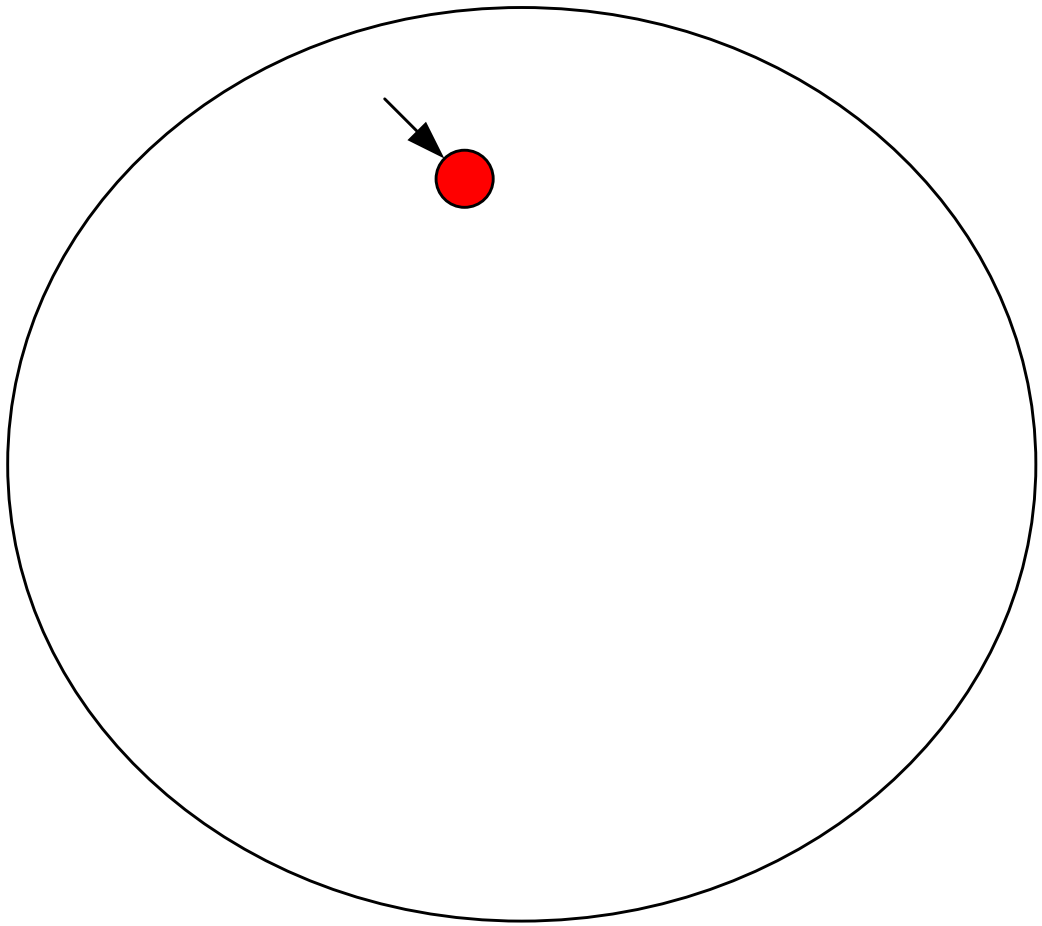




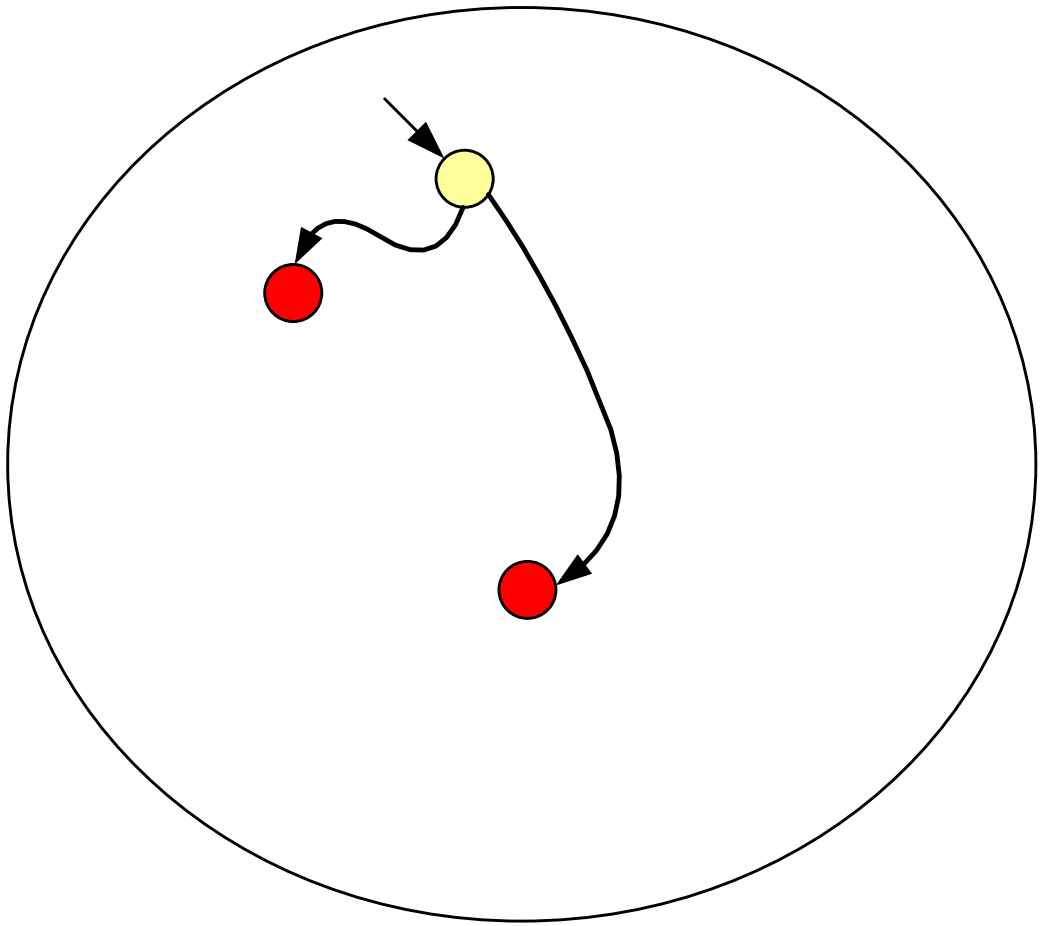
Proof succeeds

Proof fails!

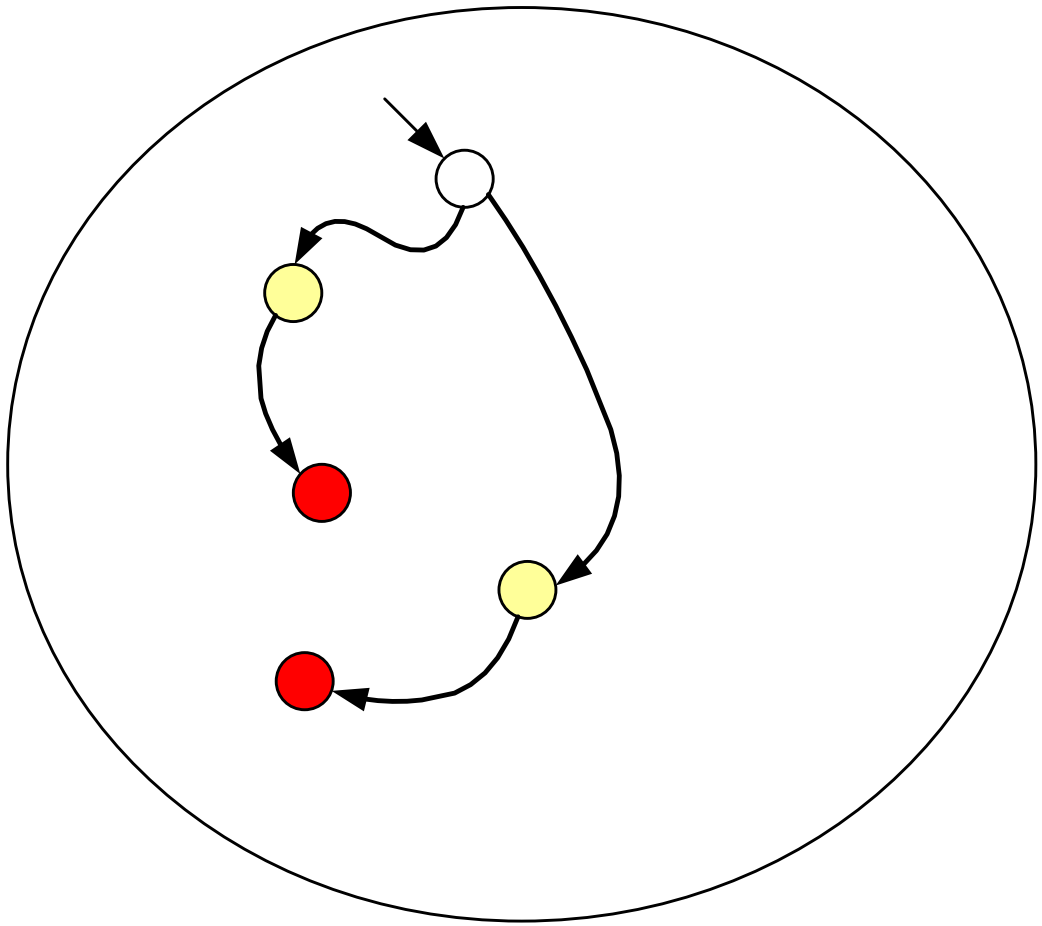
Reachable states (2)



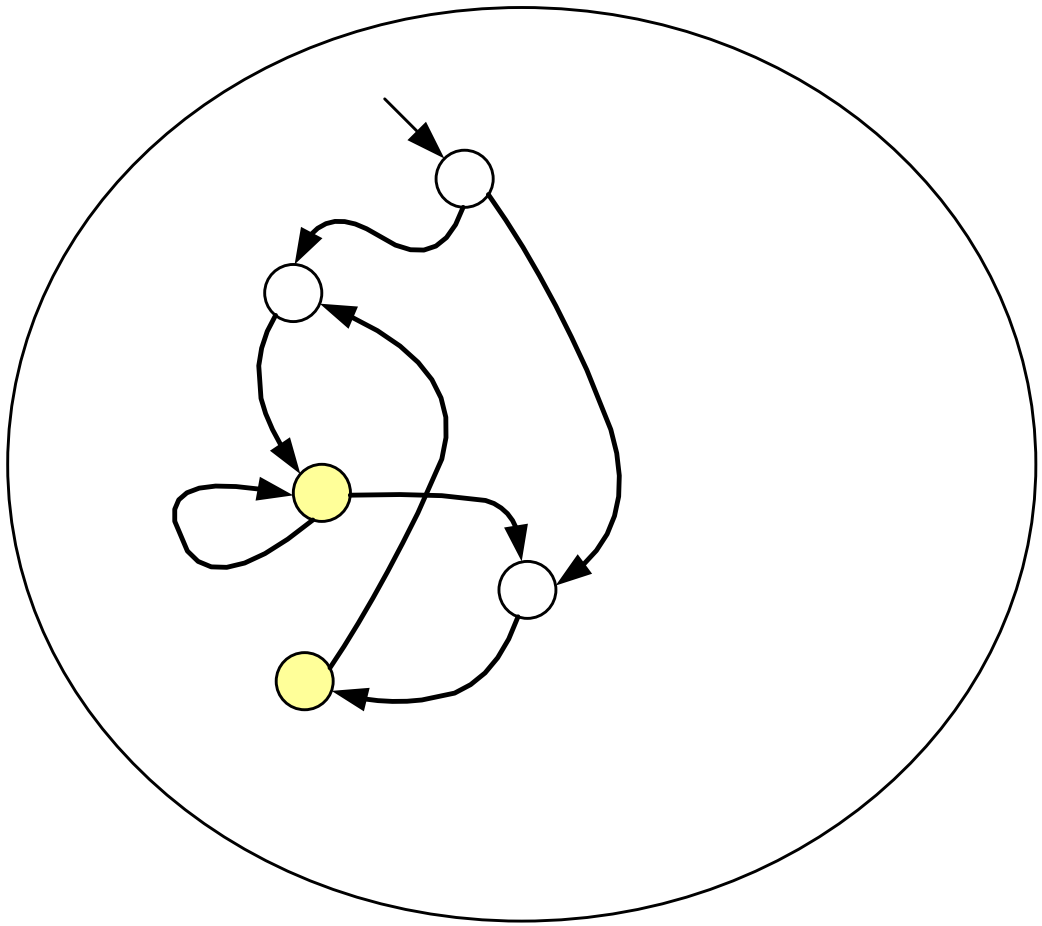
Reachable states (2)



Reachable states (2)



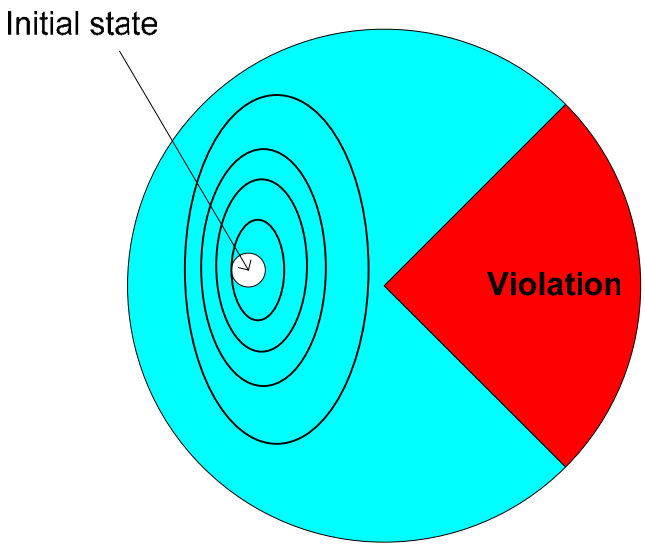
Reachable states (2)

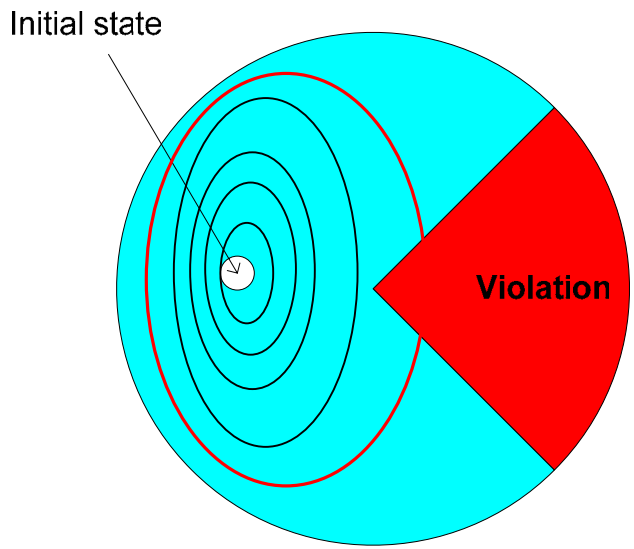


Model-checking (2)

Symbolic exploration: forward

`Post*`





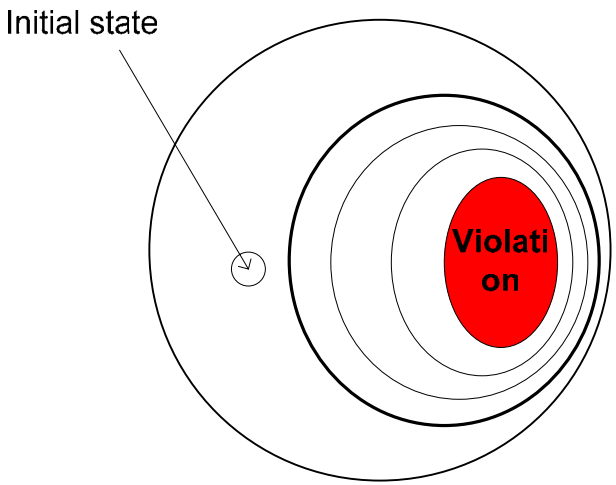
Proof succeeds

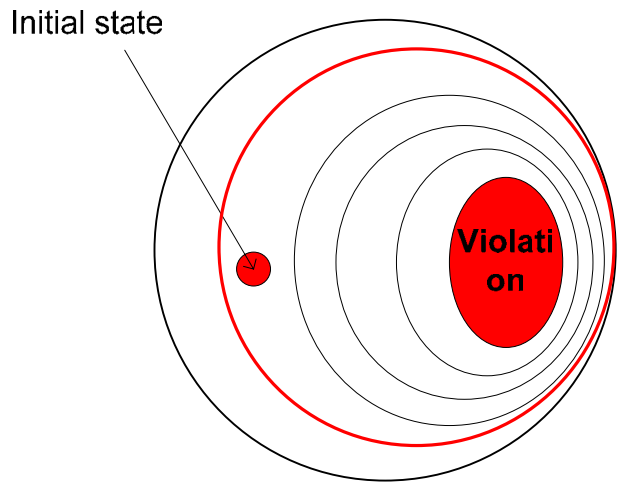
Proof fails!

Model-checking (3)

Symbolic exploration: backward

Pre*





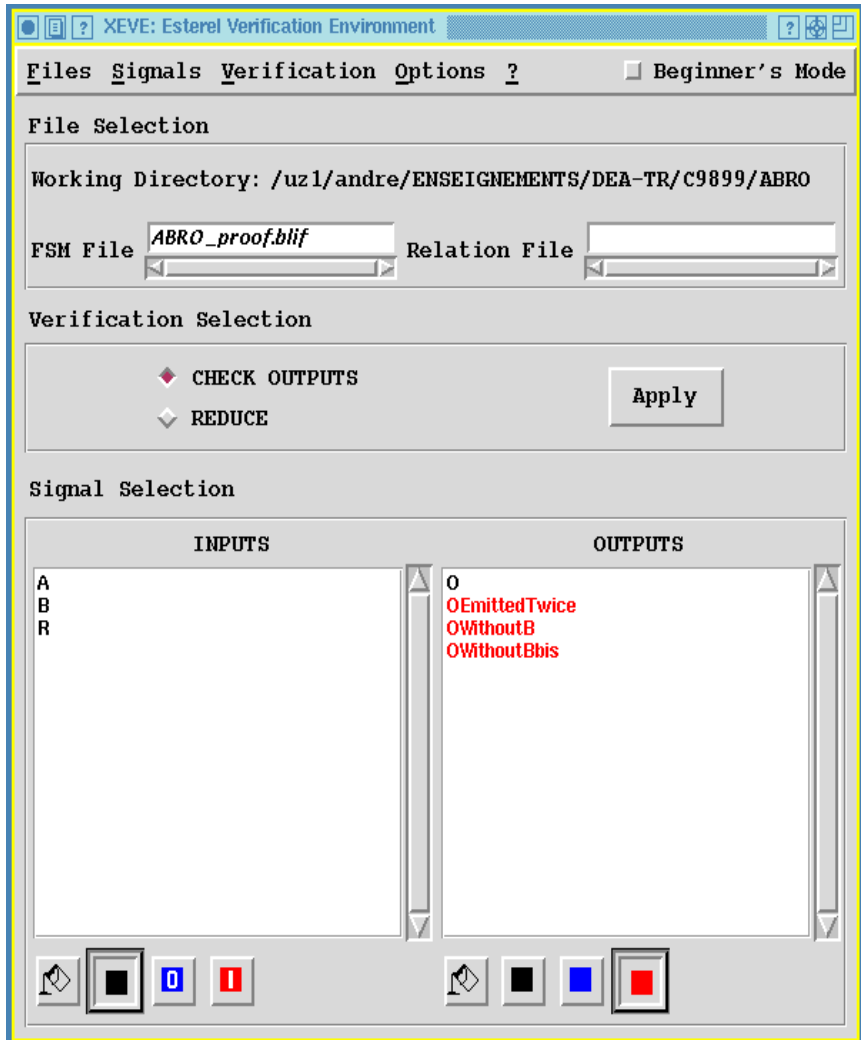
Proof succeeds

Proof fails!

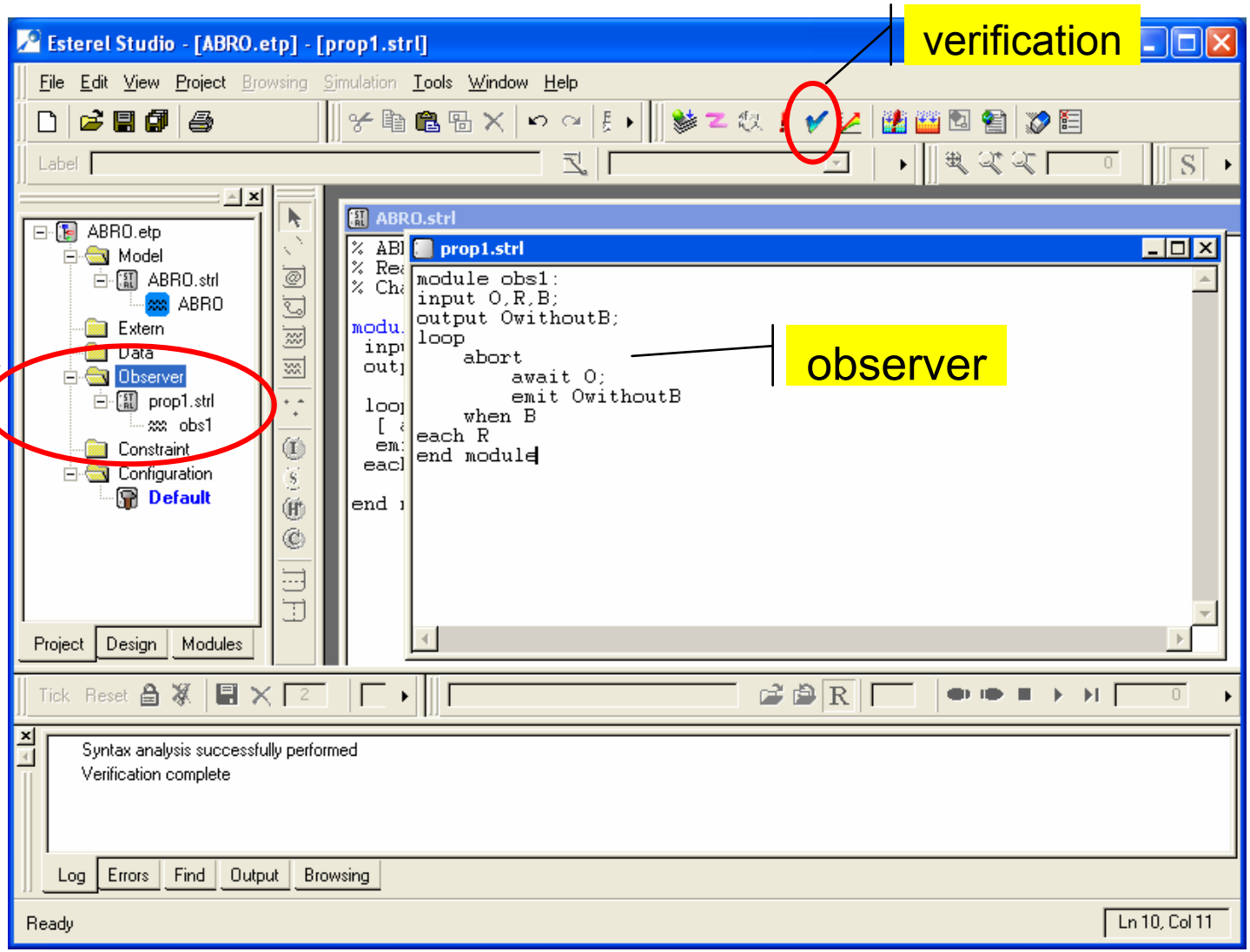
Example: Property 1

- O is never emitted without a past or simultaneous occurrence of B since the last occurrence of R or since the initial instant if R has not occurred yet.
-
```
module obs1:
    input O,R,B;
    output OwithoutB;
      loop
        abort
          await O; emit OwithoutB
        when B
      each R
    end module
```

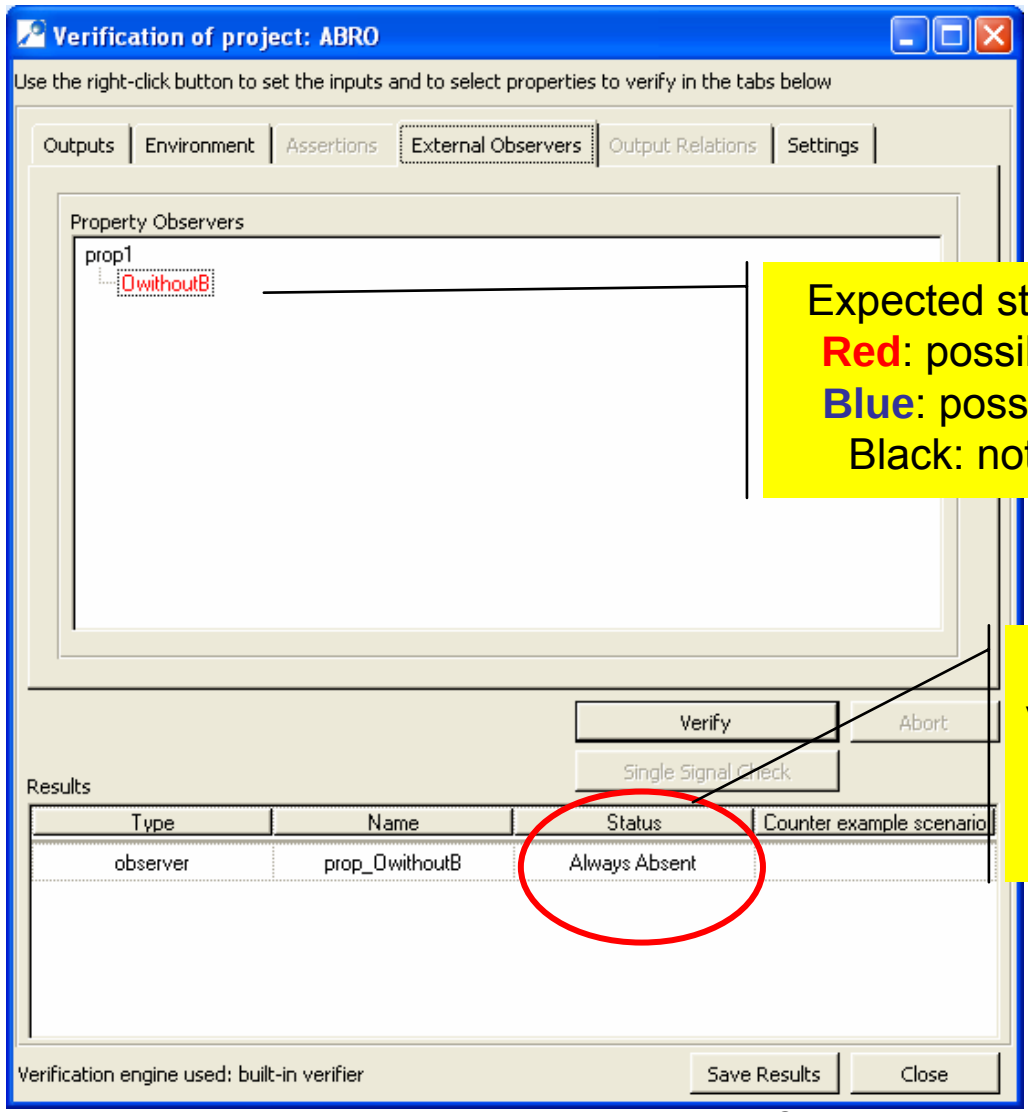
XEVE


XEVE: Esterel Verification Environment
Files Signals Verification Options ?
Beginner's Mode
File Selection
Working Directory: /uz1/andre/ENSEIGNEMENTS/DEA-TR/C9899/ABRO
FSM File
ABRO_proof.blif
Relation File
Verification Selection
CHECK OUTPUTS
REDUCE
Apply
Signal Selection
INPUTS
A
B
R
OUTPUTS
O
OEmittedTwice
OWithoutB
OWithoutBbis

E.S Verification (1)



E.S Verification



Expected status status:
Red: possibly present
Blue: possibly absent
Black: not checked

Diagnostic:
Violation signal never emitted;
Property is true