Hands-on session

Model checking logical regulatory networks

Pedro T. Monteiro

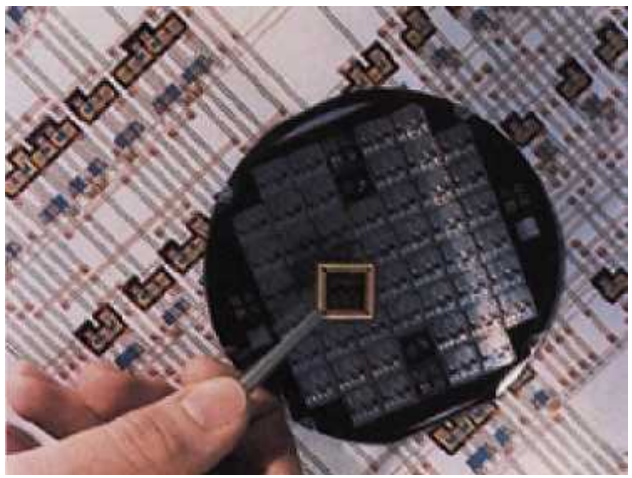
INESC-ID Lisbon, PT

[BC]^2 - Tutorial T04
Logical modelling of regulatory networks
June 9, 2015

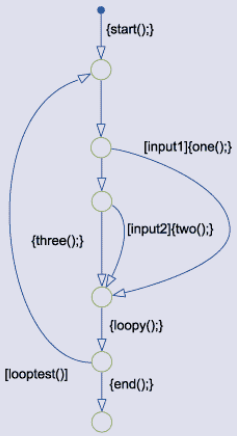
Formal verification

Formal verification based on **temporal logic** and **model checking** provides a powerful technology to query models of discrete systems.

It has been developed, since the 70s, for hardware and software verification



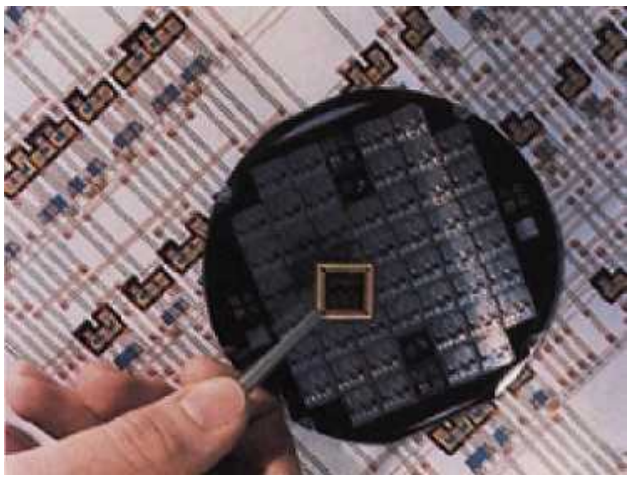
```
#include "structuredflow.h"
#include "Chart.h"
void Chart(void);
void Chart(void)
{
  start();
  do {
    if(input1) {
      one();
    } else if(input2) {
      two();
    } else {
      three();
    }
    loopy();
  } while(looptest());
  end();
}
```



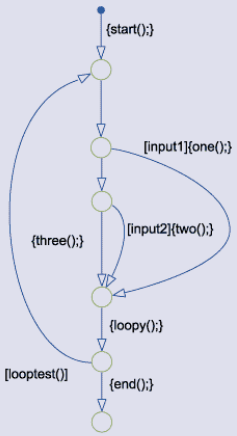
Formal verification

Formal verification based on **temporal logic** and **model checking** provides a powerful technology to query models of discrete systems.

It has been developed, since the 70s, for hardware and software verification



```
#include "structuredflow.h"
#include "Chart.h"
void Chart(void);
void Chart(void)
{
  start();
  do {
    if(input1) {
      one();
    } else if(input2) {
      two();
    } else {
      three();
    }
    loopy();
  } while(looptest());
  end();
}
```



In the last decade model checking has been successfully employed to tackle the analysis of **large biological regulatory models**.

(Chabrier et al., *CMSB* 2003) (Monteiro et al., *Bioinformatics* 2008) (Batt et al., *Bioinformatics* 2010)

Formal verification: Model checking

Model checking

Automated exhaustive exploration of the state space of the model.

- Specify the **model** transition function for state space generation
  Remember: LRG (model) → STG (dynamics)
- Specify dynamical **properties** as logical statements that are interpreted on STG

(Emerson and Clarke, *ICALP* 1980) (Queille and Sifakis, *Intl. Symp. Program.* 1982)

Formal verification: Model checking

Model checking

Automated exhaustive exploration of the state space of the model.

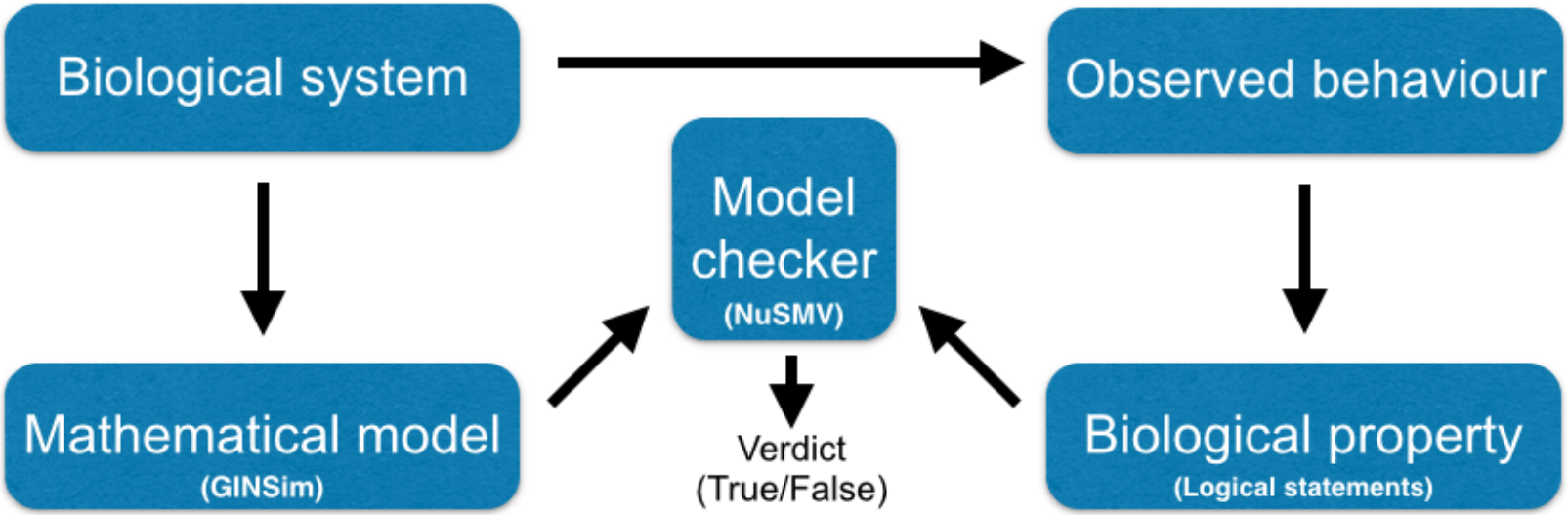
- Specify the **model** transition function for state space generation
  Remember: LRG (model) → STG (dynamics)
- Specify dynamical **properties** as logical statements that are interpreted on STG

(Emerson and Clarke, *ICALP* 1980) (Queille and Sifakis, *Intl. Symp. Program.* 1982)

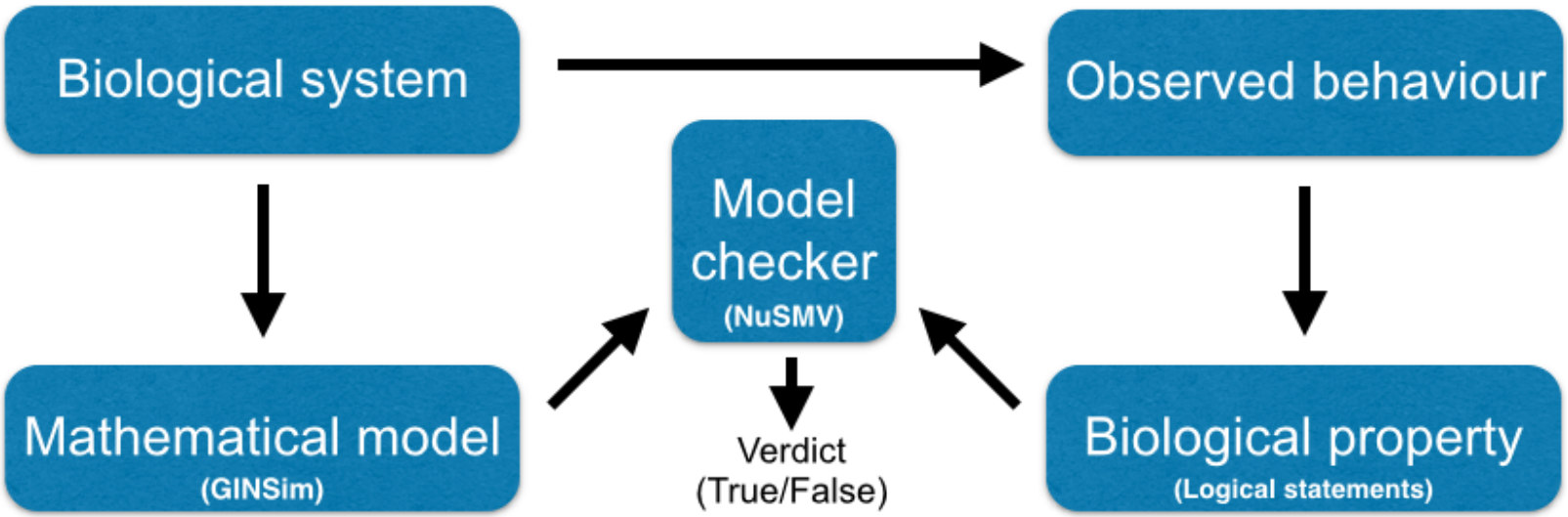
In this tutorial, we will:

- use the symbolic model checker **NuSMV** v2.4.3 `http://nusmv.fbk.eu`
- **export** the logical **model** defined in GINsim into a NuSMV specification
- verify **reachability properties** between T-helper **cell types**

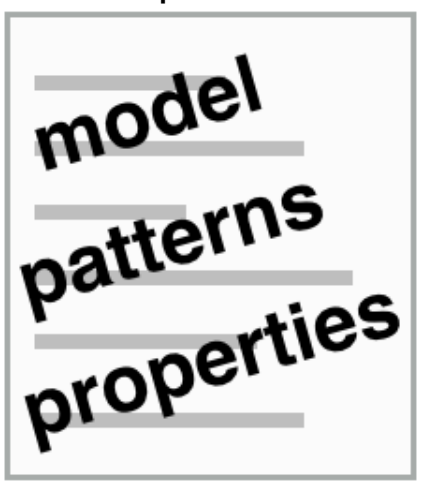
Global approach



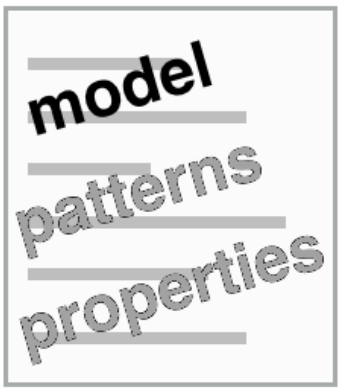
Global approach



NuSMV specification file

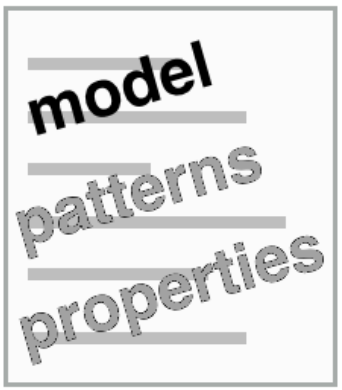


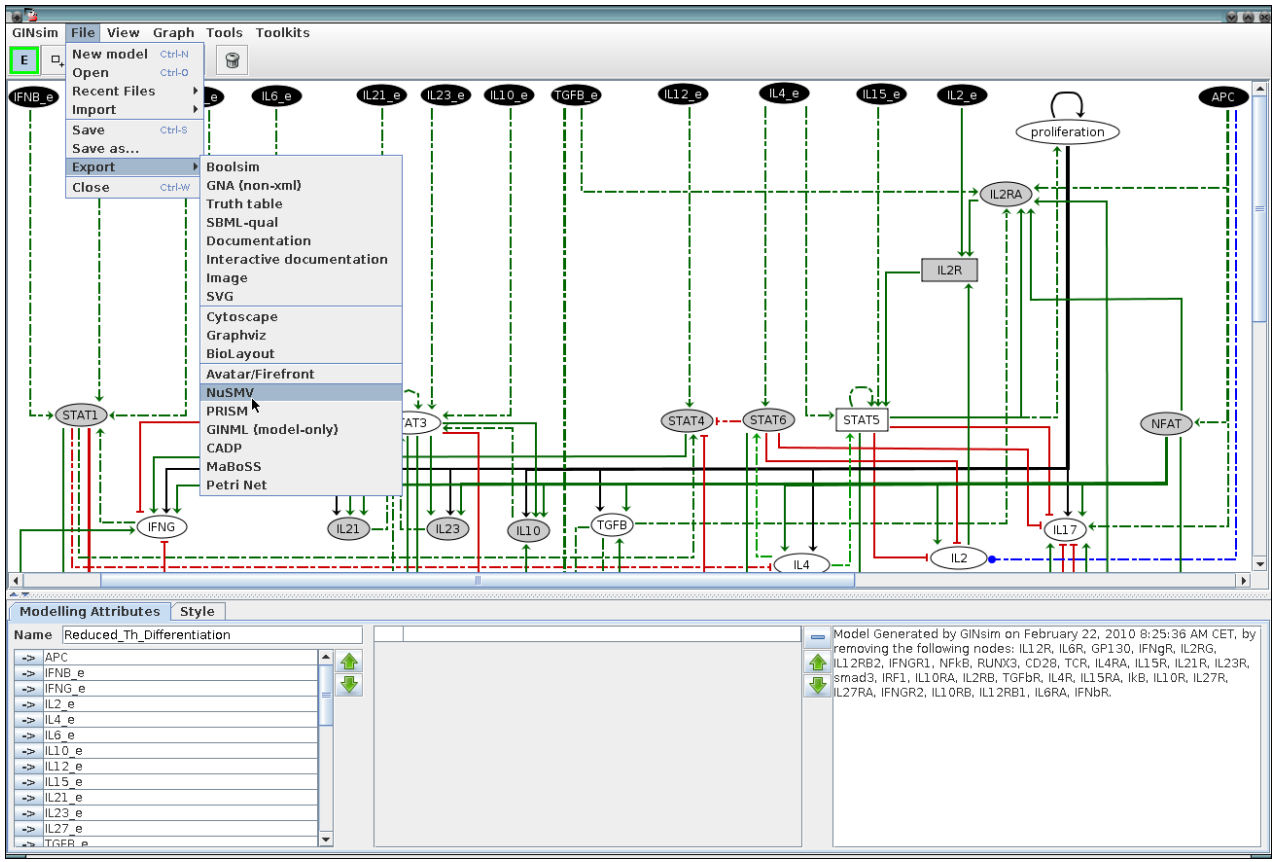
NuSMV specification file



File → Export → NuSMV

NuSMV specification file

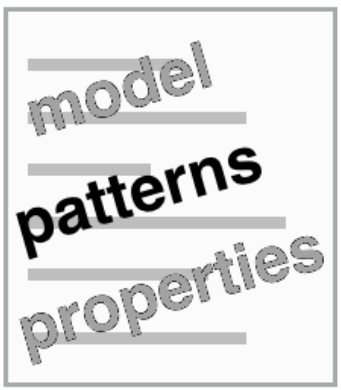




(Naldi *et al.*, *PLoS Comp Biol* 2010)

```
VirtualBox:~$ cd Desktop/Tutorial/ModelChecking
VirtualBox:~/Desktop/Tutorial/ModelChecking$ ls
th-reduced-model.smv th-reduced-input-patterns.smv
th-reduced-state-patterns.smv ...
```

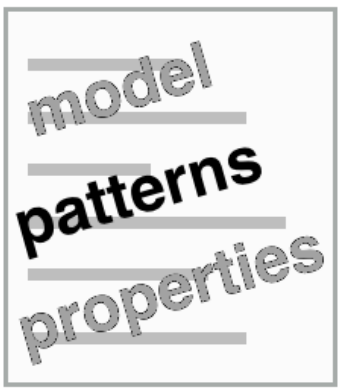
NuSMV specification file



T-helper cell environmental conditions

Input patterns

NuSMV specification file

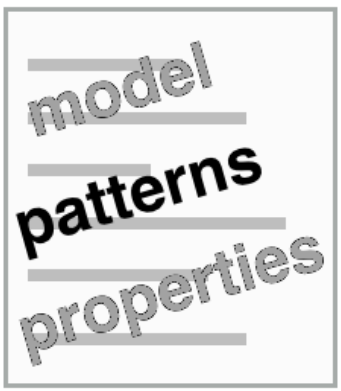


Tile code	description	Inputs: APC	IL2_e	IL4_e	IL6_e	IL10_e	IL12_e	TGFB_e	IFNG_e
□	no stimulation								
■	APC only	■							
■	pro-Th1 (i)	■	■						■
■	pro-Th1 (ii)	■					■		
■	pro-Th2	■		■	■				
■	pro-Th17	■			■			■	
■	pro-Treg (i)	■	■					■	
■	pro-Treg (ii)	■				■			

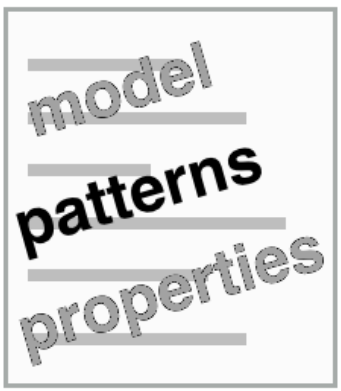
(Naldi *et al.*, *PLoS Comp Biol* 2010)

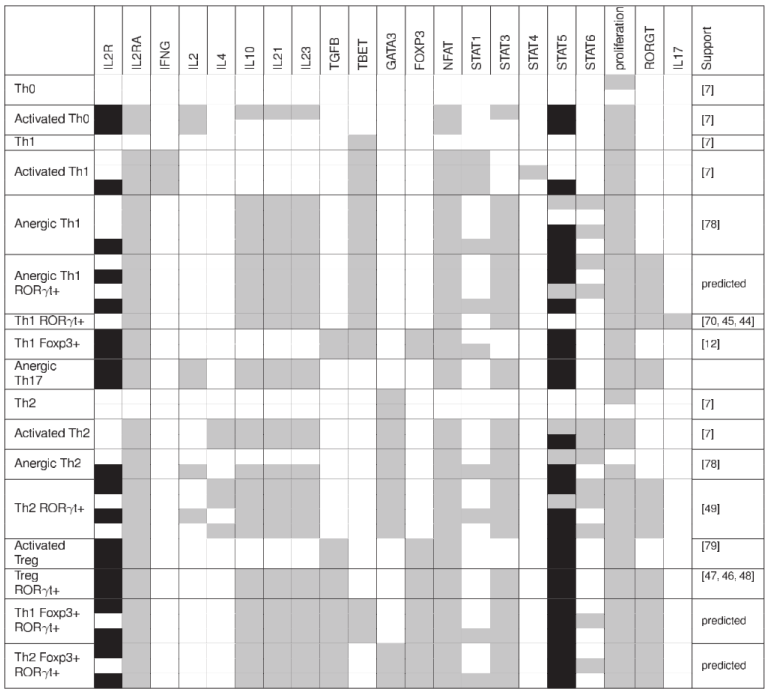
```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-input-patterns.smv
noStml := APC=0 & IL2_e=0 & IL4_e=0 & IL6_e=0 & IL10_e=0 & IL12_e=0 & TGFB_e=0 & IFNG_e=0;
APConly := APC=1 & IL2_e=0 & IL4_e=0 & IL6_e=0 & IL10_e=0 & IL12_e=0 & TGFB_e=0 & IFNG_e=0;
proTh1a := APC=1 & IL2_e=1 & IL4_e=0 & IL6_e=0 & IL10_e=0 & IL12_e=0 & TGFB_e=0 & IFNG_e=1;
...
```

NuSMV specification file



NuSMV specification file

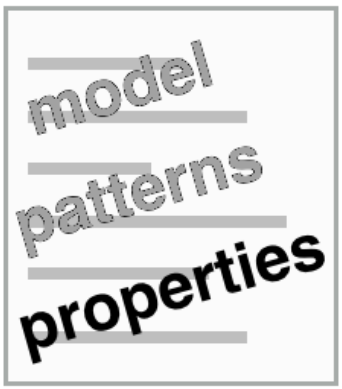




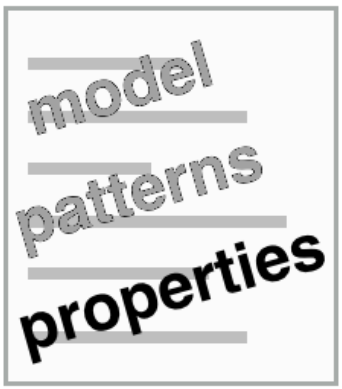
(Naldi *et al.*, *PLoS Comp Biol* 2010)

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-state-patterns.smv
...
Th1 := IL2R=0 & IL2RA=0 & IFNG=0 & IL2=0 & IL4=0 & IL10=0 & IL21=0 & IL23=0
     & TGFB=0 & TBET=1 & GATA3=0 & FOXP3=0 & NFAT=0 & STAT1=0 & STAT3=0
     & STAT4=0 & STAT5=0 & STAT6=0 & proliferation=1 & RORGT=0 & IL17=0;
...
```

NuSMV specification file



NuSMV specification file

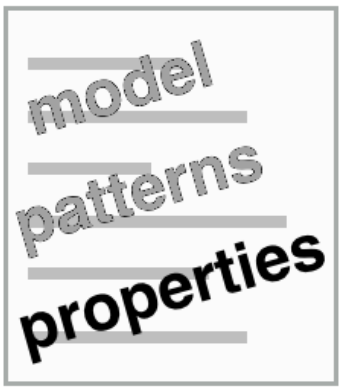


Temporal logic is a formalism for describing sequences of transitions between states (the biological system observations)

Different kinds of Temporal Logics

- CTL*:
  - CTL - Computation Tree Logic (branching-time)
  - LTL - Linear Temporal Logic (linear-time)
- CTRL - Computation Tree Regular Logic

NuSMV specification file

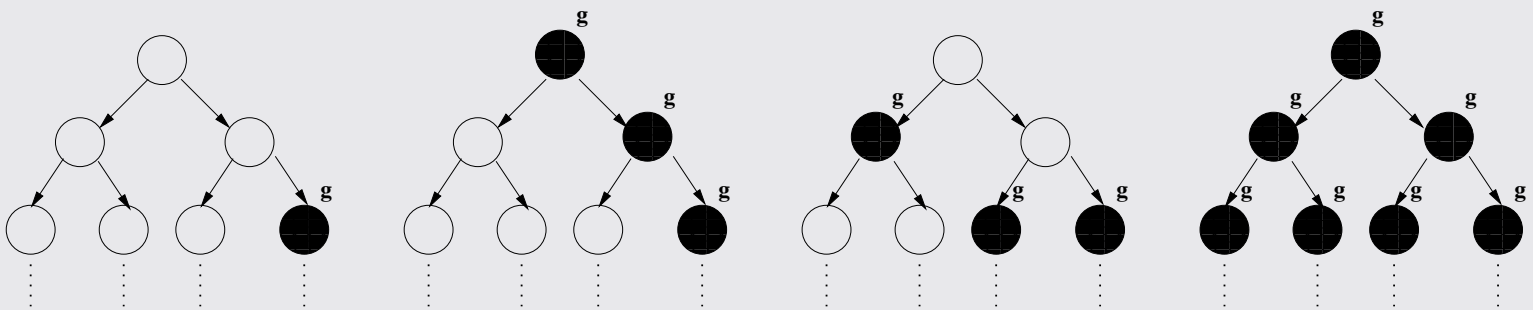


Temporal logic is a formalism for describing sequences of transitions between states (the biological system observations)

Different kinds of Temporal Logics

- CTL*:
  - CTL - Computation Tree Logic (branching-time)
  - LTL - Linear Temporal Logic (linear-time)
- CTRL - Computation Tree Regular Logic

Main CTL operators



Verification of complex properties performed through composition of operators

The model should differentiate from *Th*0 into *Th*1 under *proTh*1*a* environmental conditions. This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-1-Th0-proTh1a-Th1Act.smv
INIT Th0
INIT proTh1a
SPEC EF ( Th1Act & AG ( Th1Act ))
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-1-Th0-proTh1a-Th1Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (Th1Act & AG Th1Act) is true
```

The model should differentiate from *Th*0 into *Th*1 under *proTh*1*a* environmental conditions. This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-1-Th0-proTh1a-Th1Act.smv
INIT Th0
INIT proTh1a
SPEC EF ( Th1Act & AG ( Th1Act ))
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-1-Th0-proTh1a-Th1Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (Th1Act & AG Th1Act) is true
```

The model should differentiate from *Th*0 into *Th*1 under *proTh*1*a* environmental conditions. This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-1-Th0-proTh1a-Th1Act.smv
INIT Th0
INIT proTh1a
SPEC EF ( Th1Act & AG ( Th1Act ))
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-1-Th0-proTh1a-Th1Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (Th1Act & AG Th1Act) is true
```

The model should differentiate from *Th*0 into *Th*2 under *proTh*2 environmental conditions. This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-2-Th0-proTh2-Th2Act.smv
INIT Th0
INIT proTh2
SPEC EF ( Th2Act & AG ( Th2Act ))
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-2-Th0-proTh2-Th2Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (Th2Act & AG Th2Act) is true
```

The model should differentiate from *Th*0 into *Th*2 under *proTh*2 environmental conditions. This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-2-Th0-proTh2-Th2Act.smv
INIT Th0
INIT proTh2
SPEC EF ( Th2Act & AG ( Th2Act ))
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-2-Th0-proTh2-Th2Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (Th2Act & AG Th2Act) is true
```

The model should differentiate from *Th*0 into *Th*2 under *proTh*2 environmental conditions. This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-2-Th0-proTh2-Th2Act.smv
INIT Th0
INIT proTh2
SPEC EF ( Th2Act & AG ( Th2Act ))
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-2-Th0-proTh2-Th2Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (Th2Act & AG Th2Act) is true
```

The model should **not** differentiate from *Th*1 into *Th*2 under *proTh*2 environmental conditions. This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-3-Th1-proTh2-Th2Act.smv
INIT Th1
INIT proTh2
SPEC EF ( Th2Act & AG ( Th2Act ))
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-3-Th1-proTh2-Th2Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (Th2Act & AG Th2Act) is false
```

The model should **not** differentiate from *Th*1 into *Th*2 under *proTh*2 environmental conditions. This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-3-Th1-proTh2-Th2Act.smv
INIT Th1
INIT proTh2
SPEC EF ( Th2Act & AG ( Th2Act ))
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-3-Th1-proTh2-Th2Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (Th2Act & AG Th2Act) is false
```

The model should **not** differentiate from *Th*1 into *Th*2 under *proTh*2 environmental conditions. This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-3-Th1-proTh2-Th2Act.smv
INIT Th1
INIT proTh2
SPEC EF ( Th2Act & AG ( Th2Act ))
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-3-Th1-proTh2-Th2Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (Th2Act & AG Th2Act) is false
```

Could the model differentiate from *Th*1 into *Th*2 under **any** environmental condition? This can be stated as a reachability property to be interpreted by the model checker.

1. Create a script that calls NuSMV for all combinations of environmental conditions.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-4-Th1-x-Th2Act.sh
...
for input in $INPUTS; do
  echo "INIT␣Th1␣under␣$input"
  cat $MODEL $STATEPATTERNS $INPUTPATTERNS init-input-$input.tmp \
  init-state-Th1.tmp spec-state-Th2Act.tmp | $NuSMV -dcx 2> /dev/null\
  | grep specification
done
...
```

2. Verify the properties running the script.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ sh prop-4-Th1-x-Th2Act.sh
INIT Th1 under i00000000
-- specification EF (Th2Act & AG Th2Act) is false
INIT Th1 under i00000001
-- specification EF (Th2Act & AG Th2Act) is false
INIT Th1 under i00000010
-- specification EF (Th2Act & AG Th2Act) is false
...
```

Could the model differentiate from *Th*1 into *Th*2 under **any** environmental condition? This can be stated as a reachability property to be interpreted by the model checker.

1. Create a script that calls NuSMV for all combinations of environmental conditions.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-4-Th1-x-Th2Act.sh
...
for input in $INPUTS; do
  echo "INIT␣Th1␣under␣$input"
  cat $MODEL $STATEPATTERNS $INPUTPATTERNS init-input-$input.tmp \
  init-state-Th1.tmp spec-state-Th2Act.tmp | $NuSMV -dcx 2> /dev/null\
  | grep specification
done
...
```

2. Verify the properties running the script.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ sh prop-4-Th1-x-Th2Act.sh
INIT Th1 under i00000000
-- specification EF (Th2Act & AG Th2Act) is false
INIT Th1 under i00000001
-- specification EF (Th2Act & AG Th2Act) is false
INIT Th1 under i00000010
-- specification EF (Th2Act & AG Th2Act) is false
...
```

Could the model differentiate from *Th*1 into *Th*2 under **any** environmental condition? This can be stated as a reachability property to be interpreted by the model checker.

1. Create a script that calls NuSMV for all combinations of environmental conditions.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-4-Th1-x-Th2Act.sh
...
for input in $INPUTS; do
  echo "INIT␣Th1␣under␣$input"
  cat $MODEL $STATEPATTERNS $INPUTPATTERNS init-input-$input.tmp \
  init-state-Th1.tmp spec-state-Th2Act.tmp | $NuSMV -dcx 2> /dev/null\
  | grep specification
done
...
```

2. Verify the properties running the script.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ sh prop-4-Th1-x-Th2Act.sh
INIT Th1 under i00000000
-- specification EF (Th2Act & AG Th2Act) is false
INIT Th1 under i00000001
-- specification EF (Th2Act & AG Th2Act) is false
INIT Th1 under i00000010
-- specification EF (Th2Act & AG Th2Act) is false
...
```

We should be able to obtain the reachability between all T-helper cell types under all possible environmental conditions (Figure 7 of Naldi *et al.* 2010).

1. Create a script that calls NuSMV for all combinations of environmental conditions, between all T-helper cell types.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-5-all-x-all.sh
...
for srcstate in $STATES; do
  for input in $INPUTS; do
    echo "INIT␣$srcstate␣under␣$input"
    for target in $STATES; do
      cat $MODEL $STATEPATTERNS $INPUTPATTERNS init-input-$input.tmp \
      init-state-$srcstate.tmp spec-state-$target.tmp | $NuSMV -dcx 2> \
      /dev/null | grep true
    done
  done
done
...
```

2. Verify the properties running the script.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ sh prop-5-all-x-all.sh
INIT Th0 under noStml
-- specification EF (Th0 & AG Th0) is true
INIT Th0 under APConly
-- specification EF (Th0Act & AG Th0Act) is true
INIT Th0 under proTh1a
-- specification EF (Th1Act & AG Th1Act) is true
...
```

We should be able to obtain the reachability between all T-helper cell types under all possible environmental conditions (Figure 7 of Naldi *et al.* 2010).

1. Create a script that calls NuSMV for all combinations of environmental conditions, between all T-helper cell types.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-5-all-x-all.sh
...
for srcstate in $STATES; do
  for input in $INPUTS; do
    echo "INIT␣$srcstate␣under␣$input"
    for target in $STATES; do
      cat $MODEL $STATEPATTERNS $INPUTPATTERNS init-input-$input.tmp \
      init-state-$srcstate.tmp spec-state-$target.tmp | $NuSMV -dcx 2> \
      /dev/null | grep true
    done
  done
done
...
```

2. Verify the properties running the script.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ sh prop-5-all-x-all.sh
INIT Th0 under noStml
-- specification EF (Th0 & AG Th0) is true
INIT Th0 under APConly
-- specification EF (Th0Act & AG Th0Act) is true
INIT Th0 under proTh1a
-- specification EF (Th1Act & AG Th1Act) is true
...
```

We should be able to obtain the reachability between all T-helper cell types under all possible environmental conditions (Figure 7 of Naldi *et al.* 2010).

1. Create a script that calls NuSMV for all combinations of environmental conditions, between all T-helper cell types.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-5-all-x-all.sh
...
for srcstate in $STATES; do
  for input in $INPUTS; do
    echo "INIT␣$srcstate␣under␣$input"
    for target in $STATES; do
      cat $MODEL $STATEPATTERNS $INPUTPATTERNS init-input-$input.tmp \
      init-state-$srcstate.tmp spec-state-$target.tmp | $NuSMV -dcx 2> \
      /dev/null | grep true
    done
  done
done
...
```

2. Verify the properties running the script.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ sh prop-5-all-x-all.sh
INIT Th0 under noStml
-- specification EF (Th0 & AG Th0) is true
INIT Th0 under APConly
-- specification EF (Th0Act & AG Th0Act) is true
INIT Th0 under proTh1a
-- specification EF (Th1Act & AG Th1Act) is true
...
```

The model should differentiate from *Th*0 into *Th*1 under *proTh*1*a* environmental conditions, with an activation of IL2 ($IL2 = 1$). This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-6-Th0-IL2-Th1Act.smv
INIT Th0
INIT proTh1a

SPEC EF (IL2=1 & EF (Th1Act & AG (Th1Act)))
SPEC E [ IL2=1 U Th1Act & AG (Th1Act)]
SPEC EF (Th1Act) & ! E [ IL2=0 U Th1Act ]
SPEC E [ IL2=0 U Th1Act & AG (Th1Act)]
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-6-Th0-IL2-Th1Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (IL2 = 1 & EF (Th1Act & AG Th1Act)) is true
-- specification E [ IL2 = 1 U (Th1Act & AG Th1Act) ] is false
-- specification (EF Th1Act & !E [ IL2 = 0 U Th1Act ] ) is false
-- specification E [ IL2 = 0 U (Th1Act & AG Th1Act) ] is true
```

The model should differentiate from *Th*0 into *Th*1 under *proTh*1*a* environmental conditions, with an activation of IL2 ($IL2 = 1$). This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-6-Th0-IL2-Th1Act.smv
INIT Th0
INIT proTh1a

SPEC EF (IL2=1 & EF (Th1Act & AG (Th1Act)))
SPEC E [ IL2=1 U Th1Act & AG (Th1Act)]
SPEC EF (Th1Act) & ! E [ IL2=0 U Th1Act ]
SPEC E [ IL2=0 U Th1Act & AG (Th1Act)]
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-6-Th0-IL2-Th1Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (IL2 = 1 & EF (Th1Act & AG Th1Act)) is true
-- specification E [ IL2 = 1 U (Th1Act & AG Th1Act) ] is false
-- specification (EF Th1Act & !E [ IL2 = 0 U Th1Act ] ) is false
-- specification E [ IL2 = 0 U (Th1Act & AG Th1Act) ] is true
```

The model should differentiate from *Th*0 into *Th*1 under *proTh*1*a* environmental conditions, with an activation of IL2 ($IL2 = 1$). This can be stated as a reachability property to be interpreted by the model checker.

1. Create a NuSMV specification file that verifies this behaviour.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat prop-6-Th0-IL2-Th1Act.smv
INIT Th0
INIT proTh1a

SPEC EF (IL2=1 & EF (Th1Act & AG (Th1Act)))
SPEC E [ IL2=1 U Th1Act & AG (Th1Act)]
SPEC EF (Th1Act) & ! E [ IL2=0 U Th1Act ]
SPEC E [ IL2=0 U Th1Act & AG (Th1Act)]
```

2. Verify the property using NuSMV.

```
VirtualBox:~/Desktop/Tutorial/ModelChecking$ cat th-reduced-model.smv \
th-reduced-state-patterns.smv th-reduced-input-patterns.smv \
prop-6-Th0-IL2-Th1Act.smv | NuSMV-2.4.3 -ctlei -dcx

-- specification EF (IL2 = 1 & EF (Th1Act & AG Th1Act)) is true
-- specification E [ IL2 = 1 U (Th1Act & AG Th1Act) ] is false
-- specification (EF Th1Act & !E [ IL2 = 0 U Th1Act ] ) is false
-- specification E [ IL2 = 0 U (Th1Act & AG Th1Act) ] is true
```