Embedded Systems Innovation BY TNO

Model-Based Testing

Using TorXakis

TorXakis : Installation

1. Installation: https://github.com/TorXakis/TorXakis/
2. Windows : Get and install TorXakis.msi

   Also Linux and Mac-OS
3. Windows installation : C:\Program Files (x86)\TNO TorXakis\TorXakis
4. Download examples https://www.cs.ru.nl/~tretmans/exampsWS.zip
5. Optional: install notepad++ plug-in for keyword high-lighting
6. Optional: install eclipse plug-in for syntax directed editing (readme.txt)
7. For some SUTs, install JDK* – Java Development Kit

* *http://www.oracle.com/technetwork/java/javase/downloads/*

TorXakis

View on Systems and Models

TorXakis : A Black-Box View on Systems

Channels with messages

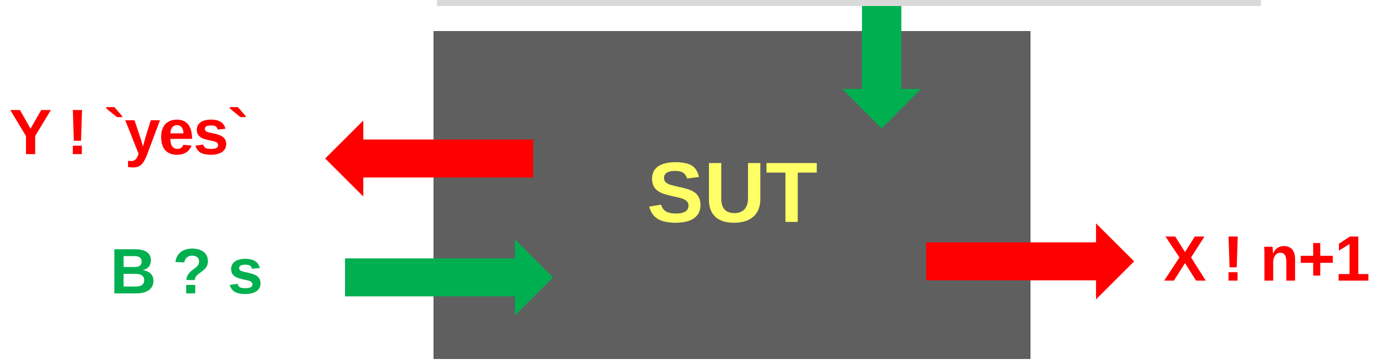
- **Inputs Channels:**
  - A :: Int; B :: Struct
- **Output Channels:**
  - X :: Int; Y :: String

SUT

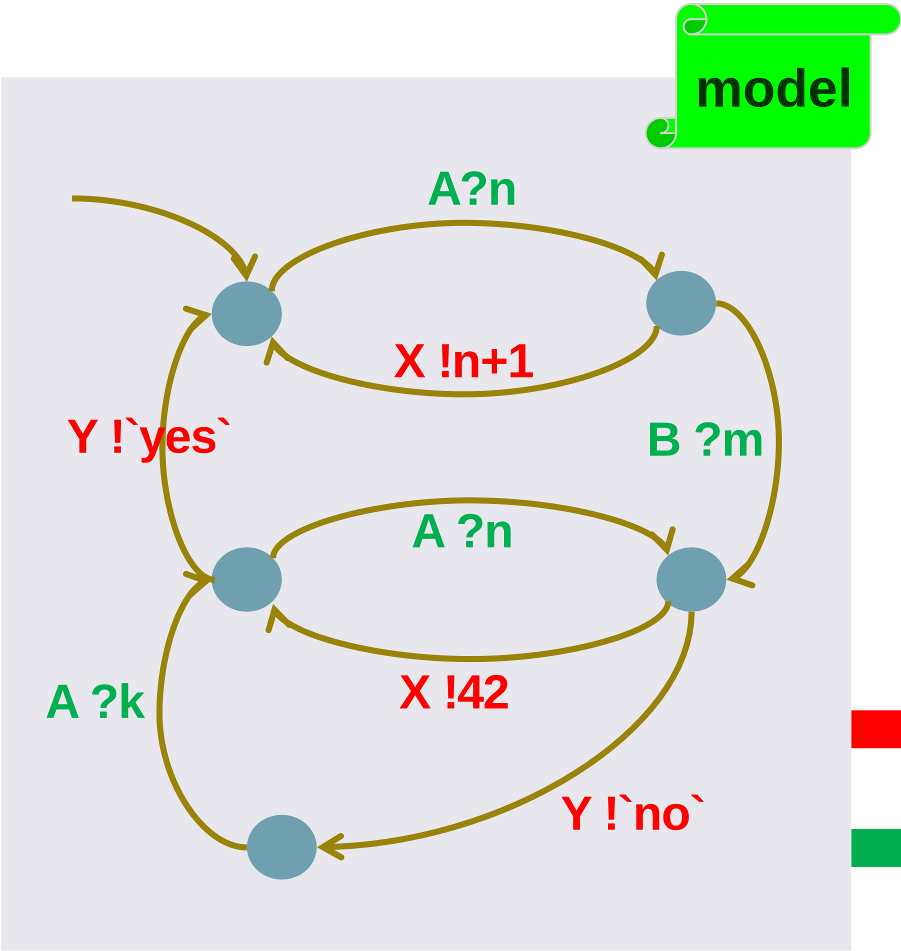
real, black-box system

communicating with its environment

via messages on input- and output channels



TorXakis : A View on Models



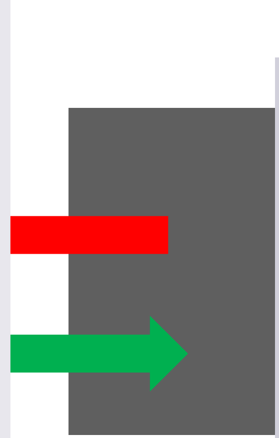
MODEL

labelled transition system

with parameterized actions on

input- and output channels

A ? n



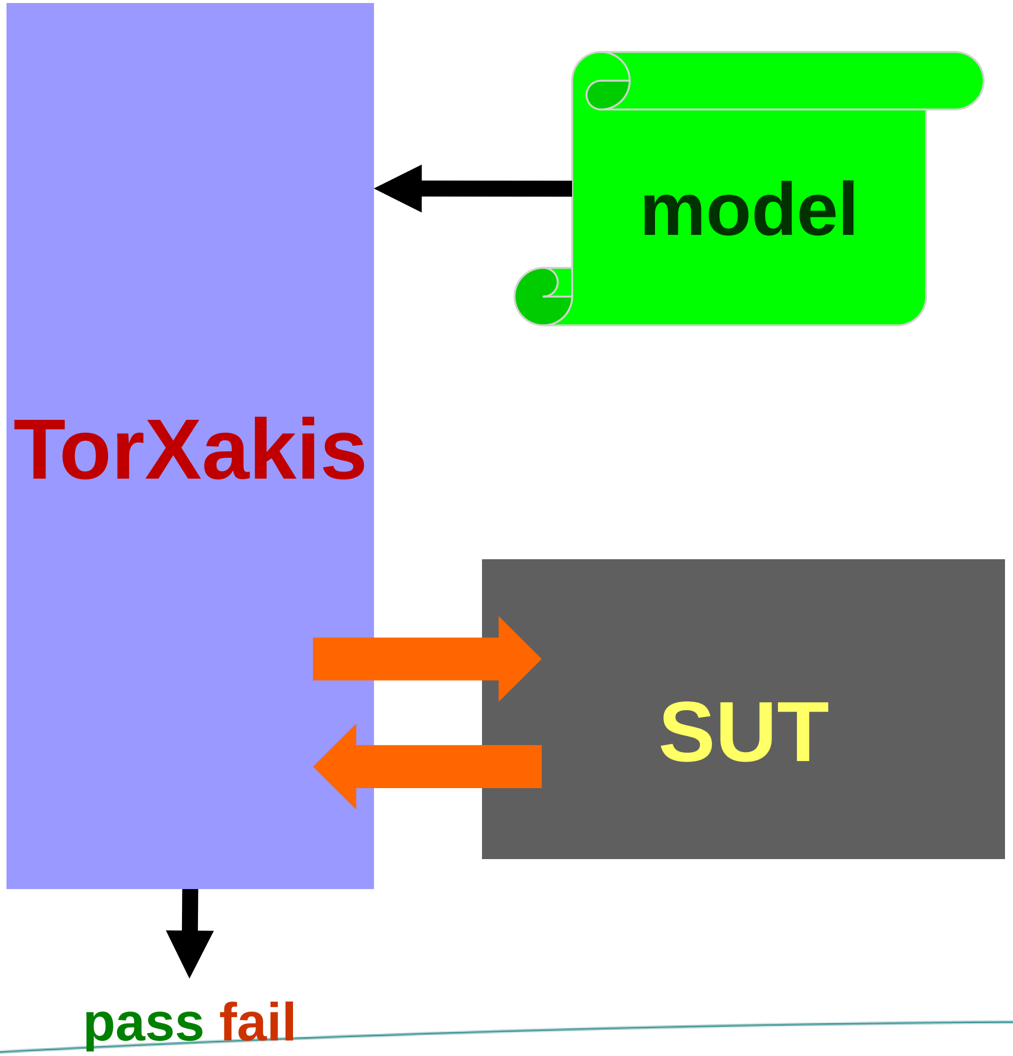
Not (yet) in TorXakis:

- **real-time**
- **probabilities**
- **derivatives (hybrid)**

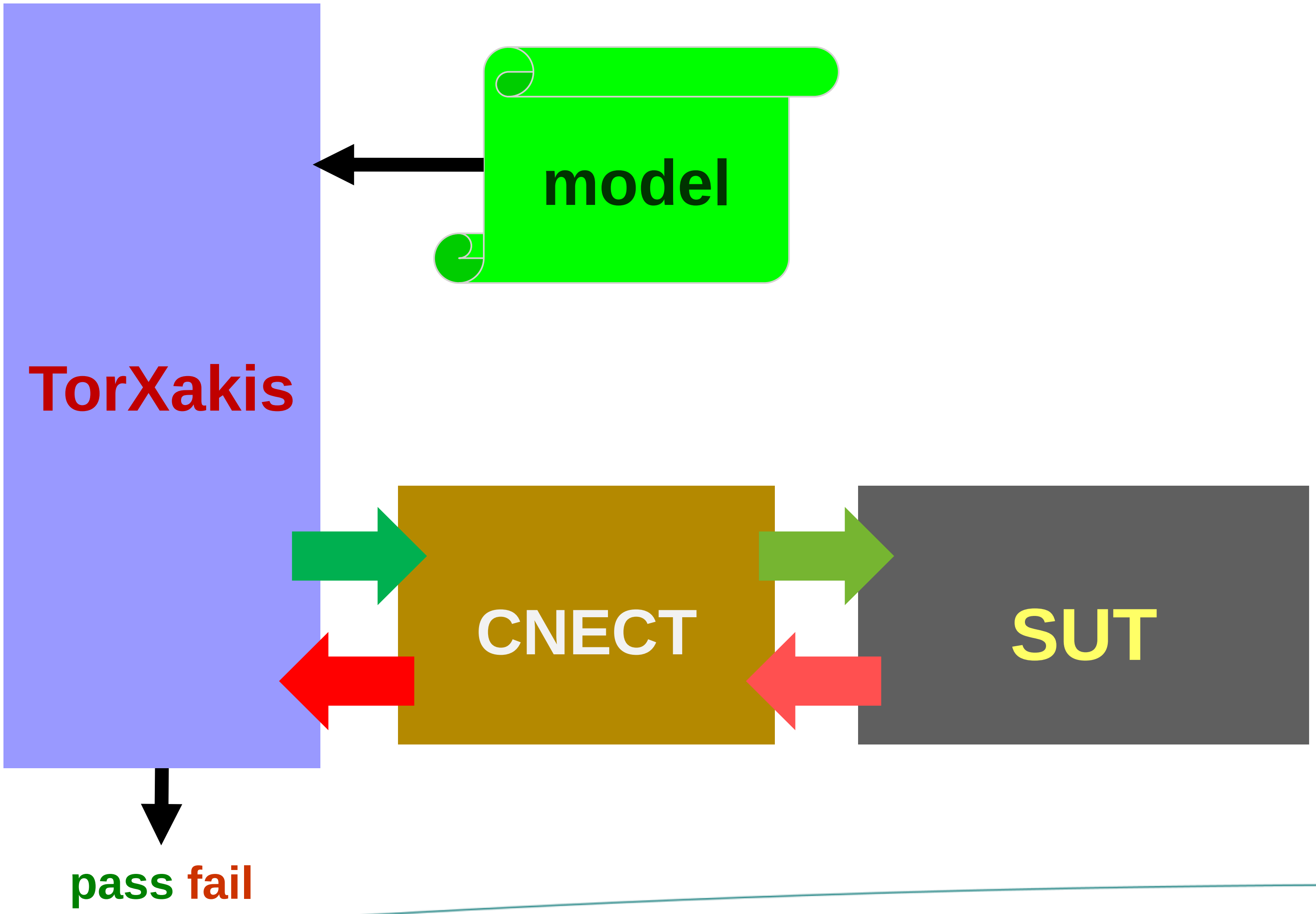
Model-Based Testing

TorXakis - The Tool

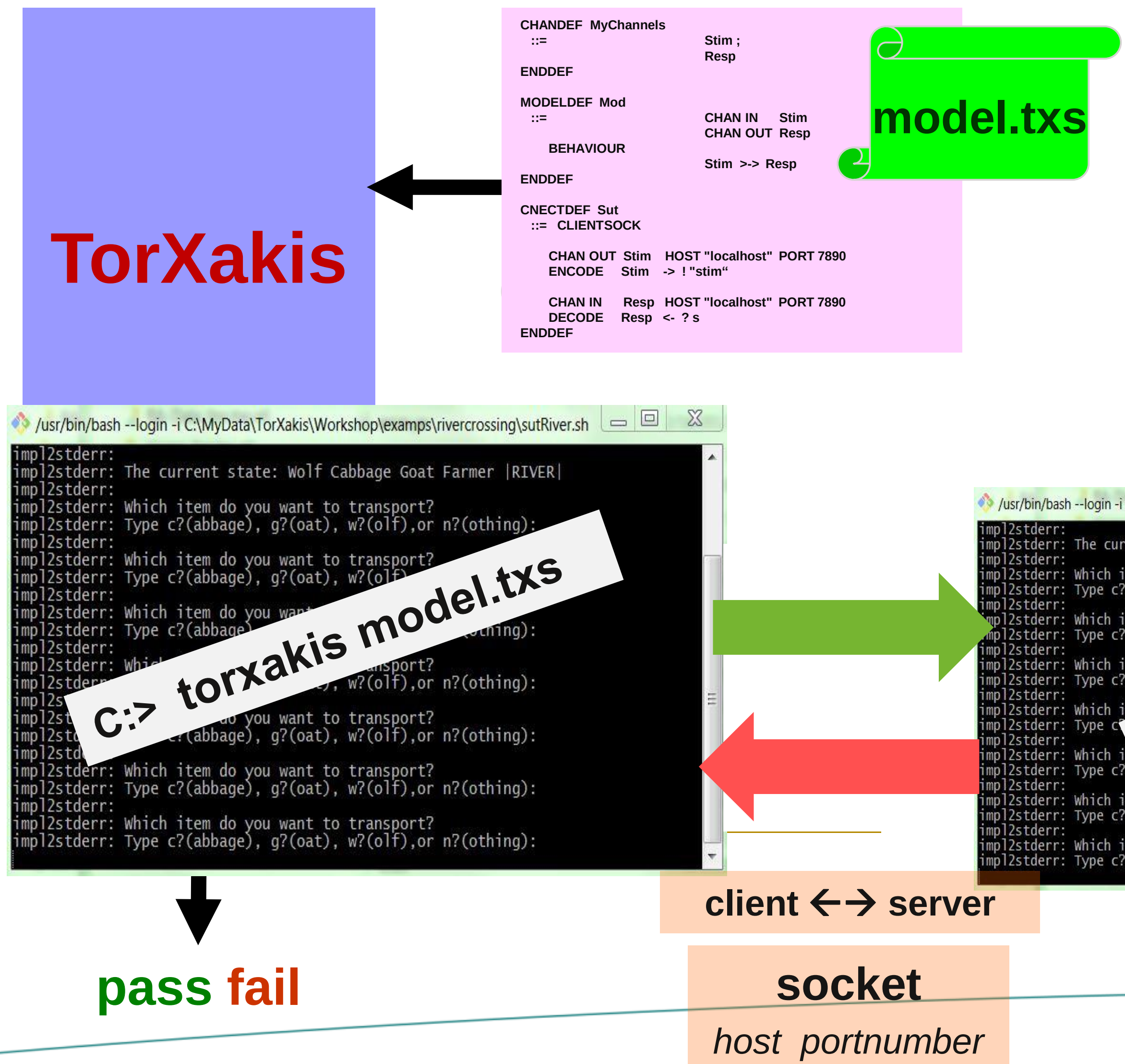
TorXakis : An On-Line MBT Tool



TorXakis and SUT



TorXakis and SUT



TorXakis : Installation

1. Installation: https://github.com/TorXakis/TorXakis/
2. Windows : Get and install TorXakis.msi

   Also Linux and Mac-OS
3. Windows installation : C:\Program Files (x86)\TNO TorXakis\TorXakis
4. Download examples https://www.cs.ru.nl/~tretmans/exampsWS.zip
5. Optional: install notepad++ plug-in for keyword high-lighting
6. Optional: install eclipse plug-in for syntax directed editing (readme.txt)
7. For some SUTs, install JDK* – Java Development Kit

* *http://www.oracle.com/technetwork/java/javase/downloads/*

TorXakis

1. My First TorXakis Test Run

2. My First TorXakis Model
   - Channels
   - SUT
   - Model

3. More TorXakis Models and Runs

Embedded Systems Innovation BY TNO

TorXakis

1. My First TorXakis Test Run

2. My First TorXakis Model
   - Channels
   - SUT
   - Model

……/examps/StimulusResponse/ /model/SRfinite.txs

1. More TorXakis Models

TorXakis : Running a Test (Windows)

1. Start two Command Prompt windows, one for TorXakis, one for the SUT
2. There are three versions of SUT:
   - pre-compiled (Windows) executable \winexe
   - Java source * \java
   - simulated TorXakis model \model
3. In SUT window, start SUT: SRfinite.exe 7890
4. In TorXakis window, go to ...\examps\StimulusResponse\model
5. Start TorXakis with the model file C:> torxakis SRfinite.txs
6. In TorXakis start the tester TXS << tester Mod Sut
7. In TorXakis run 4 test steps TXS << test 4
8. Try some other TorXakis command TXS << help

* http://www.oracle.com/technetwork/java/javase/downloads/

Embedded Systems Innovation BY TNO

Running TorXakis and SUT

```
$ torxakis SRfinite.txs

TXS >>  TorXakis :: Model-Based Testing

TXS >>  txsserver starting: "PC-14411.tsn.tno.nl" : 9876
TXS >>  input files parsed: SRfinite.txs
TXS >>  smt solver initialized: Z3 [4.4.2 - build hashcode e4b7ac37f38f]
TXS >>  txsserver initialized
TXS <<  help
```

Embedded Systems Innovation BY TNO

Running TorXakis and SUT

```
TXS >> tester Mod Sut
TXS >> test 4
TXS >>  ....1: IN: Act { { ( Stim, [] ) } }
TXS >>  ....2: OUT: Act { { ( Resp, [] ) } }
TXS >>  ....3: OUT: No Output (Quiescence)
TXS >>  ....4: OUT: No Output (Quiescence)
TXS >>  PASS
TXS <<
```

Embedded Systems Innovation BY TNO

TorXakis

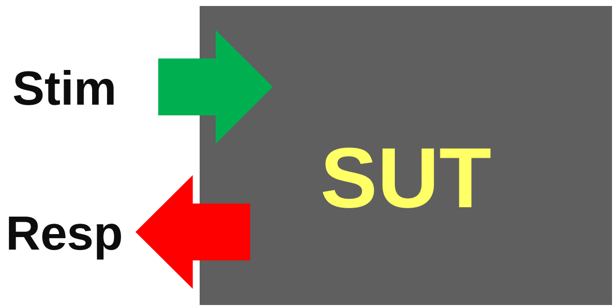
....../examps/StimulusResponse/model/SRfinite.txs

1. My First TorXakis Test Run

2. My First TorXakis Model
   - Channels
   - SUT
   - Model

3. More TorXakis Models

TorXakis : Definition of Channels



MODEL

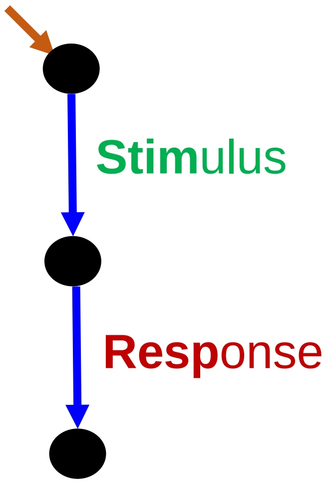
labelled transition system

with parameterized actions on

input- and output channels

```
CHANDEF  MyChannels
    ::=
                    Stim ;
                    Resp
ENDDEF
```

TorXakis: My first model

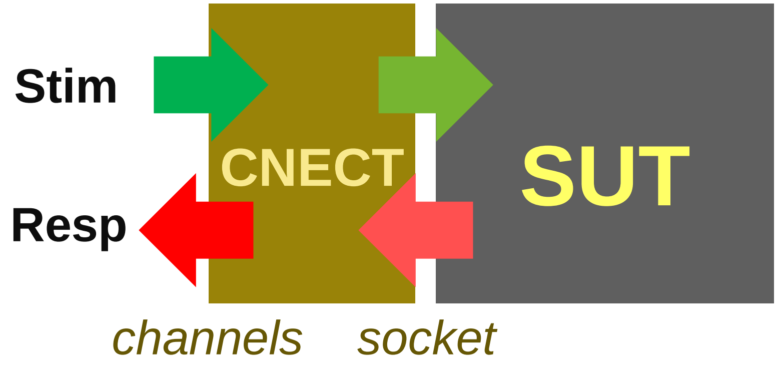


MODEL

labelled transition system with actions on input- and output channels

```
MODELDEF  MyModel
  ::=
          CHAN IN        Stim
          CHAN OUT       Resp

          BEHAVIOUR      Stim  >->  Resp

ENDDEF
```

TorXakis : Definition of SUT



SUT

real, black-box system

communicating with its environment

via messages on input- and

output channels

```
CNECTDEF  Sut
    ::=
          CLIENTSOCK

          CHAN OUT  Stim             HOST "localhost"  PORT 7890
          ENCODE    Stim       ->    ! "stim“

          CHAN IN   Resp             HOST "localhost"  PORT 7890
          DECODE    Resp       <-    ? s
ENDDEF
```

TorXakis

...../examps/StimulusResponse/model/SRfinite.txs

1. My First TorXakis Test Run

2. My First TorXakis Model
   - Channels
   - SUT
   - Model

3. More TorXakis Models

Embedded Systems Innovation BY TNO

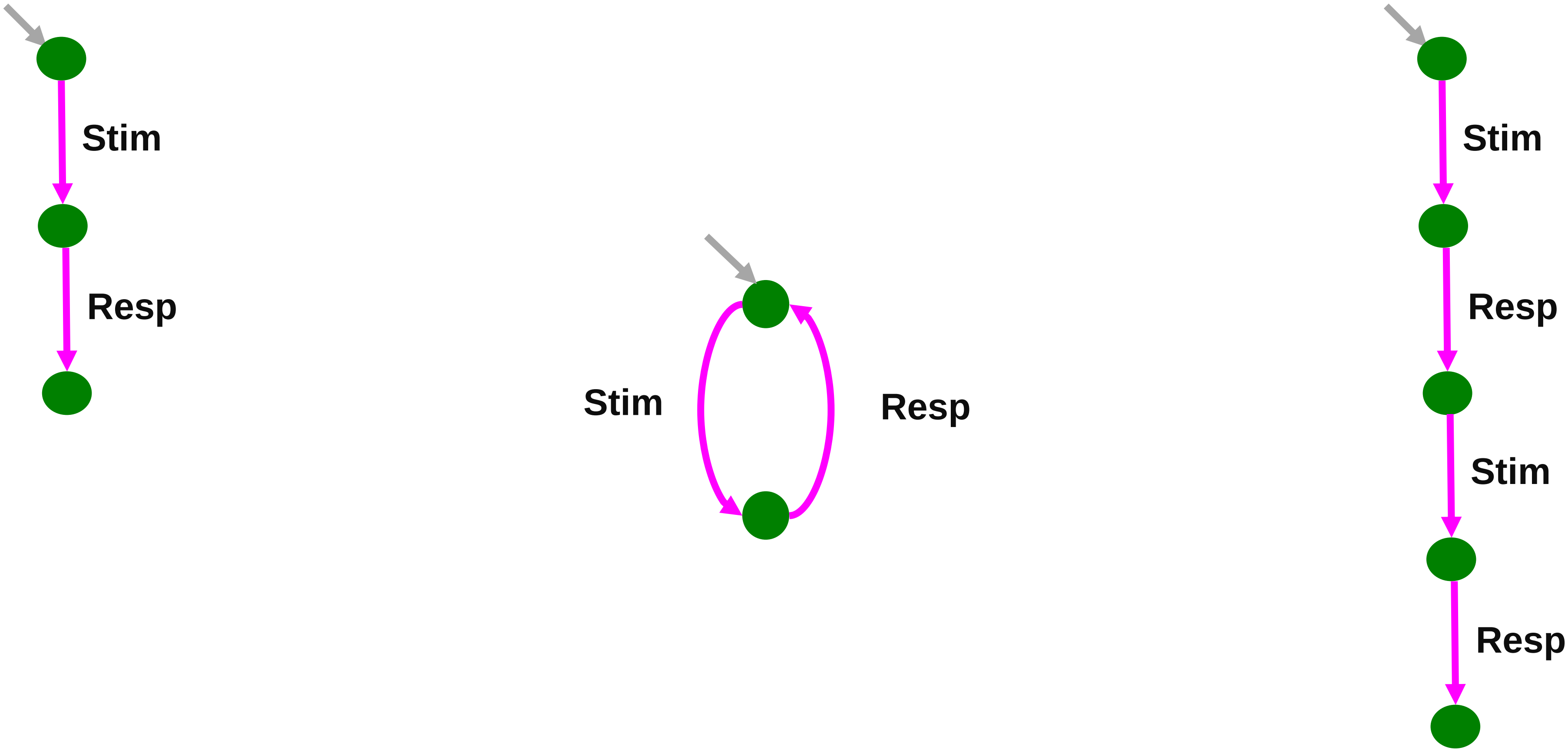
TorXakis

1. My First TorXakis Model
   - SUT
   - Model
   - Adapter

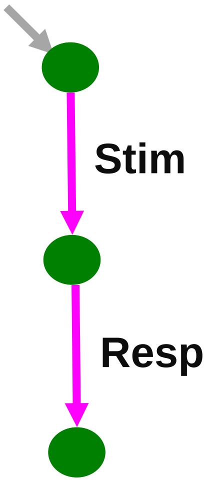
2. My First TorXakis Test Run

3. More TorXakis Models

TorXakis: Process Definition

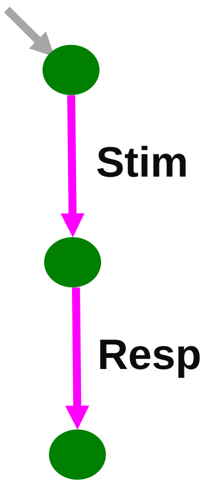


TorXakis: Process Definition




```
MODELDEF  Spec
   ::=
          CHAN IN     Stim
          CHAN OUT    Resp

          BEHAVIOUR

                         Stim   >-> Resp

ENDDEF
```

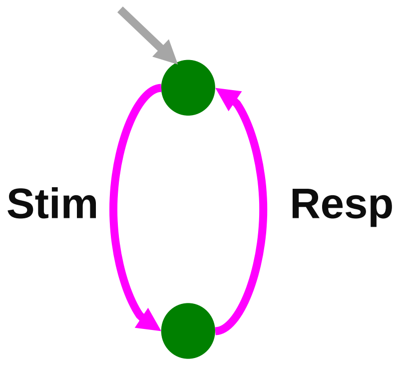
TorXakis: Process Definition




```
PROCDEF  stimResp  [ Stm, Rsp ]  ( )
   ::=
            Stm  >-> Rsp
ENDDEF
```

```
MODELDEF  Mod
   ::=
            CHAN IN          Stim
            CHAN OUT         Resp

            BEHAVIOUR

                   stimResp [ Stim, Resp ]  ( )

ENDDEF
```

TorXakis: Process Definition




```
PROCDEF stimResp [ Stm, Rsp ] ( )
  ::=
                    Stm
            >-> Rsp
            >-> stimResp [ Stm, Rsp ] ( )
ENDDEF


MODELDEF Mod
  ::=       CHAN IN      Stim
            CHAN OUT   Resp

            BEHAVIOUR
                        stimResp [ Stim, Resp ] ( )
ENDDEF
```

TorXakis: Exercise

TorXakis identifiers:

Type, Constructor, Model, Cnect: ***start with capital***

Function, Constant, Variable: ***start with small letter***

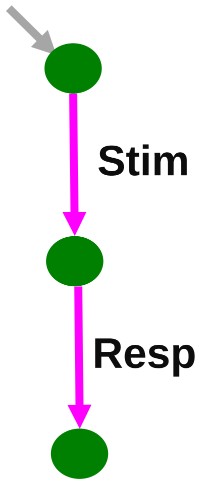
Make a model for the looping StimulusResponse system (or look at SRloop.txs)

Test SUT = SRloop.java / .exe against your looping StimulusResponsemodel (SRloop.txs)

Test the finite system SUT = SRfinite.java / .exe against your looping StimulusResponsemodel.

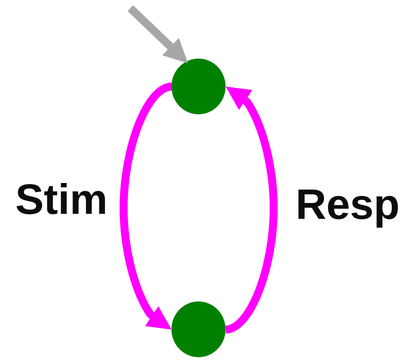
Repeat for the looping SUT = SRloop.java/.exe against the old model = SRfinite.txs. Explain the results.

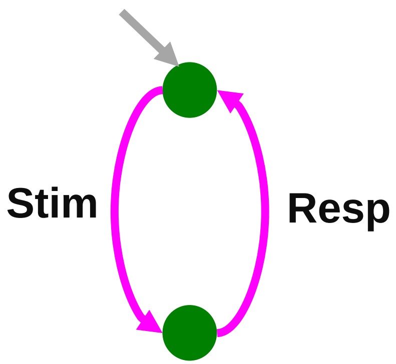
TorXakis: Exercise Result



~~implements~~

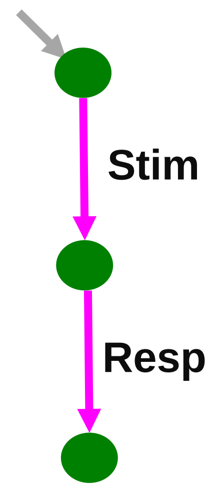
~~ioco~~





implements

ioco



TorXakis: Exercise

Experiment, using testing, simulation, or stepping, with

SRfinite.txs, SRlnone.txs, SRloop,txs, SRnone

and corresponding SUTs.

If you have JDK installed you can make *mutants*,

i.e., small modifications/errors in the Java SUTs,

and see whether you can detect the errors.

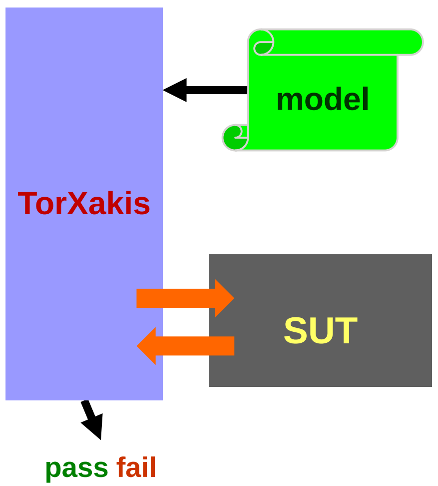
Embedded Systems Innovation BY TNO

TorXakis

More Models

Composition and Representation

TorXakis : Definitions



TorXakis input =

list of definitions

data

- **type** — **TYPEDEF**
- **function** — **FUNCDEF**
- **constant** — **CONSTDEF**

test architecture

- **channels** — **CHANDEF**
- **model** — **MODELDEF**
- **connection** — **CNECTDEF**

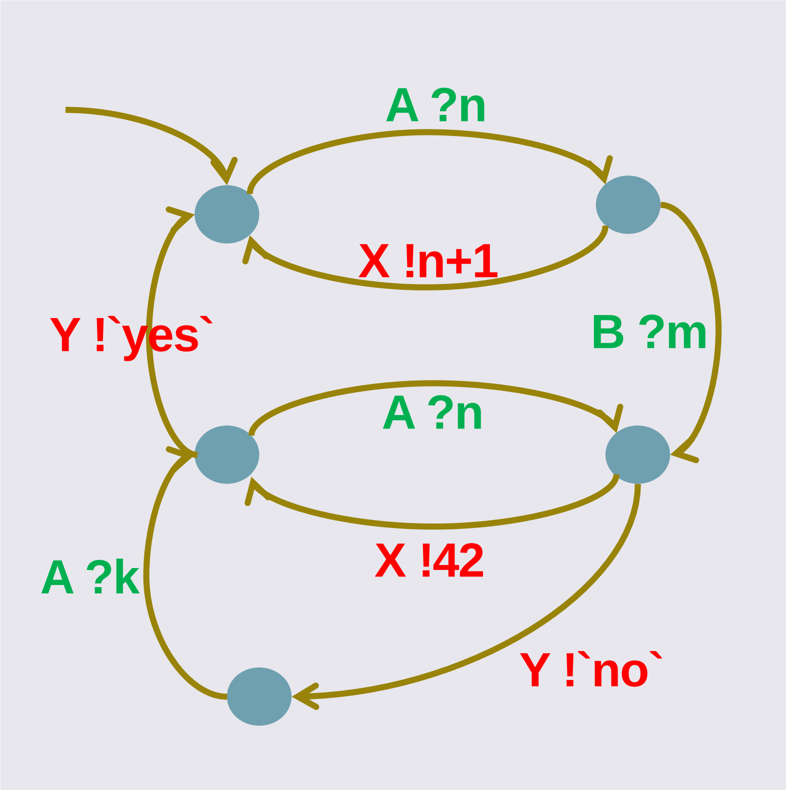
behaviour /

labelled transition system

- **process** — **PROCDEF**
- **state automaton** — **STAUTDEF**

TorXakis : Defining Behaviour - STS

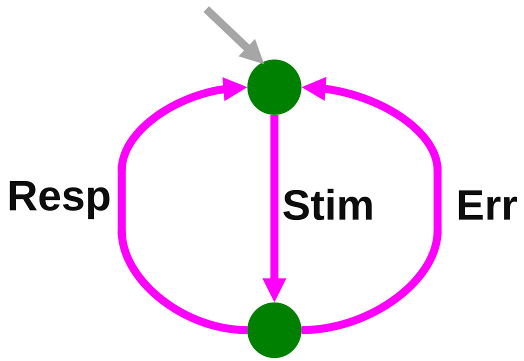
basic behaviour
= transition system



complex behaviour
= combining transition systems

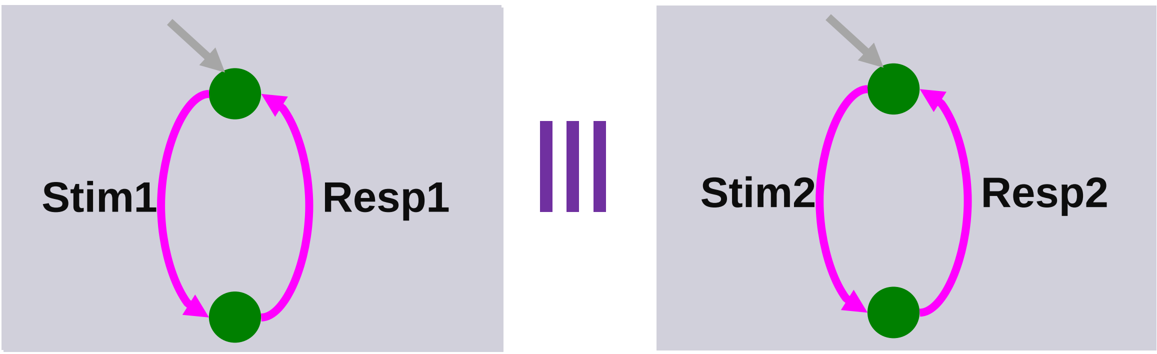
- **named behaviour definition**
- **named behaviour use**
- **sequence**
- **choice**
- **parallel**
- **communication**
- **exception**
- **interrupt**
- **hiding**

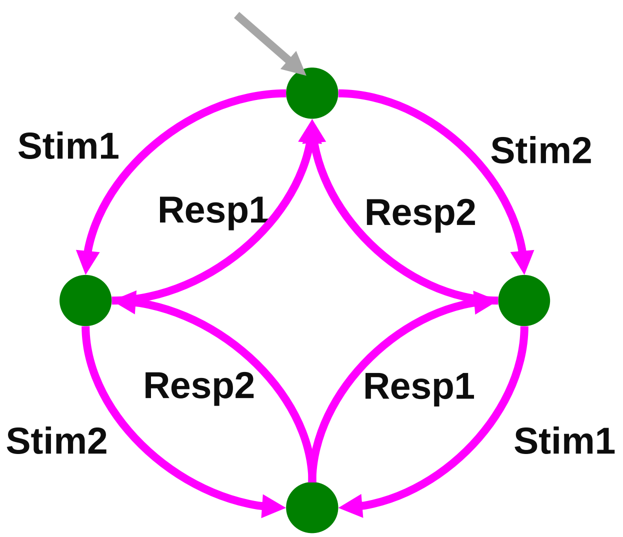
TorXakis: Choice




```
--  Stimulus-Response with Error

PROCDEF  errSR  [ Stim, Resp, Err ]  ()
  ::=
       Stim   >->
            (    Resp  >->    errSR [Stim,Resp,Err] ()
              ##
                Err      >->    errSR [Stim,Resp,Err] ()
            )
ENDDEF
```

TorXakis: Parallel Interleaving

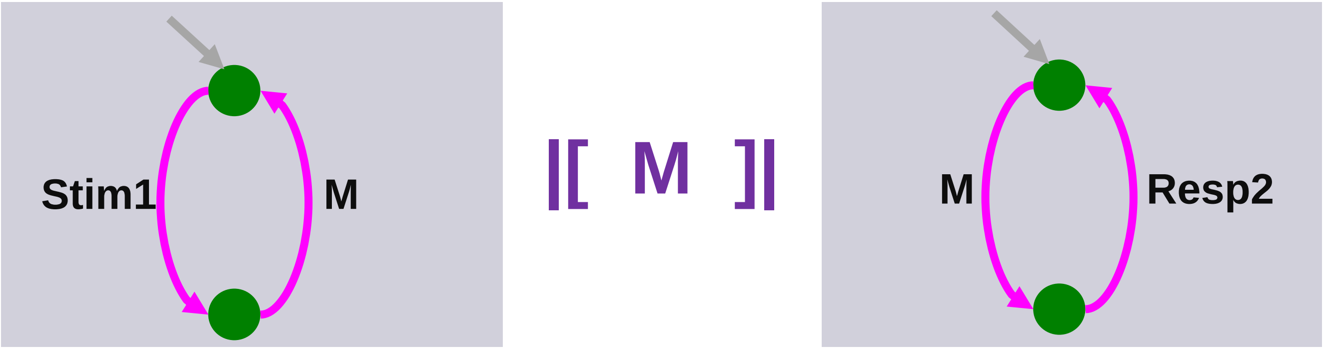


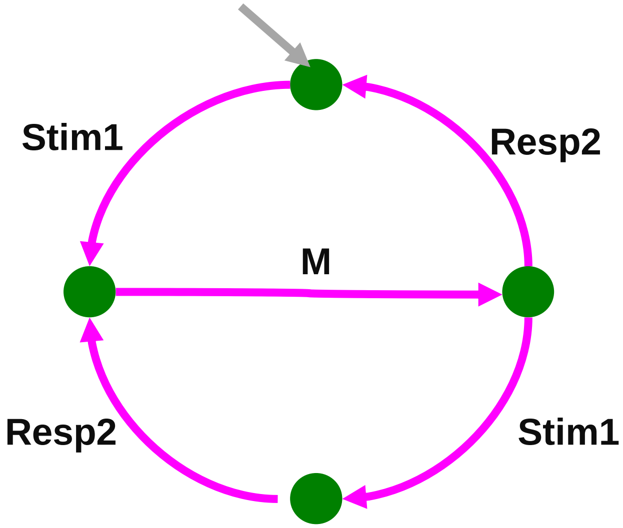


Parallelism with interleaving:

stimResp1 ||| stimResp2

TorXakis: Parallel Communication

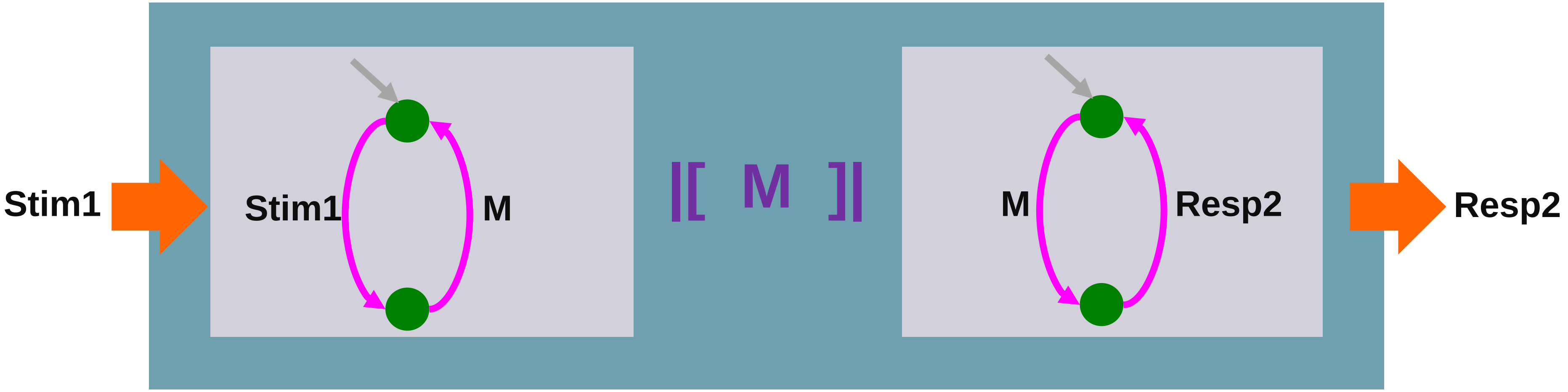


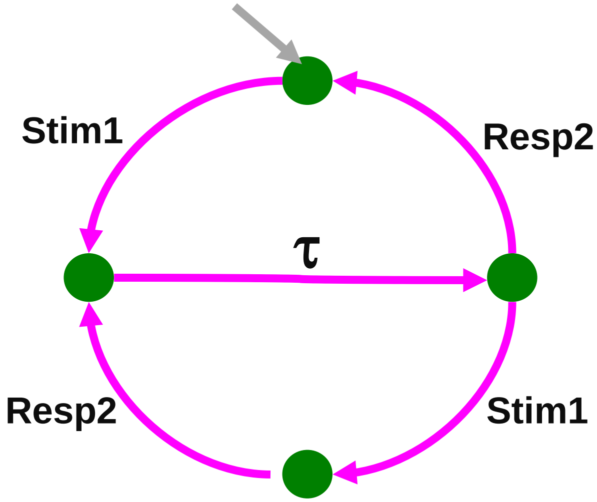


Parallelism with communication:

stimResp1 |[M]| stimResp2

TorXakis: Communication + Hiding (Abstraction)





Communication + Hiding:

HIDE [M]
IN
stimResp1 |[M]| stimResp2
NI

TorXakis: Behaviour Compositions

Enable

```
        proc1  [ A, X ]  ( )
>>>
        proc2  [ B, Y ]  ( )
```

when proc1 *finishes,* proc2 *continues*

Disable

```
        proc1  [ A, X ]  ( )
[>>
        proc2  [ B, Y ]  ( )
```

the first action of proc2 *disables* proc1

Interrupt

```
        proc1  [ A, X ]  ( )
[><
        proc2  [ B, Y ]  ( )
```

the first action of proc2 *disables* proc1*;*
when proc2 *finishes,* proc1 *continues where it stopped*

TorXakis: Exercise

Experiment, using testing, simulation, or stepping,

with the different models in SRparallel.txs ,

and various SUTs.

If you have JDK installed you can make *mutants*,

i.e., small modifications/errors in the Java SUTs,

and see whether you can detect the errors.

TorXakis

Data Definitions and Functions

TorXakis: Data Types

- Standard types: Int, Bool, String, *Regular Expression*
- Algebraic data types

```
TYPEDEF  Colour  ::=  Red | Yellow | Blue  ENDDEF

TYPEDEF  IntList  ::=    Nil
                       | Cons  {  hd :: Int
                               ,  tl :: IntList
                               }
ENDDEF
```

Embedded Systems Innovation BY TNO

TorXakis: Func

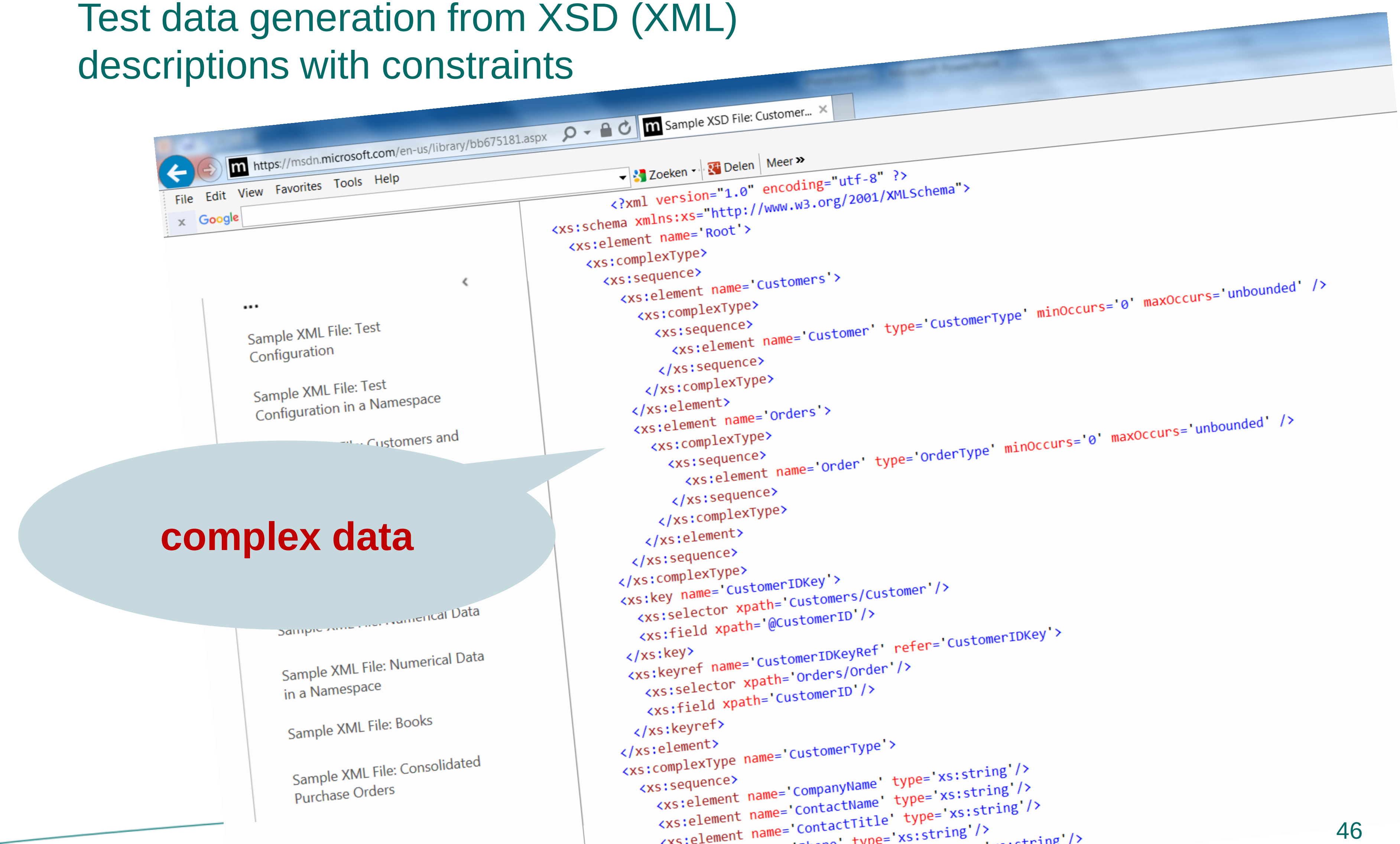
```
TYPEDEF   IntList   ::=     Nil
                          | Cons  {   hd  ::  Int
                                  ,   tl  ::  IntList
                                  }
ENDDEF
```

- Functions: name, pa
- Overloading
- Standard functions for: Int, Bool, String, *Regular Expression*

```
FUNCDEF   ++  ( s :: IntList;  x :: Int )  ::  IntList
    ::=
          IF    isNil ( s )
          THEN   Cons ( x, Nil )
          ELSE   Cons ( hd ( s ),  tl ( s ) ++ x )
          FI
ENDDEF
```

More Complex Data

Test data generation from XSD (XML) descriptions with constraints



TorXakis

1. My First TorXakis Model
   - SUT
   - Model
   - Adapter

2. My First TorXakis Test Run

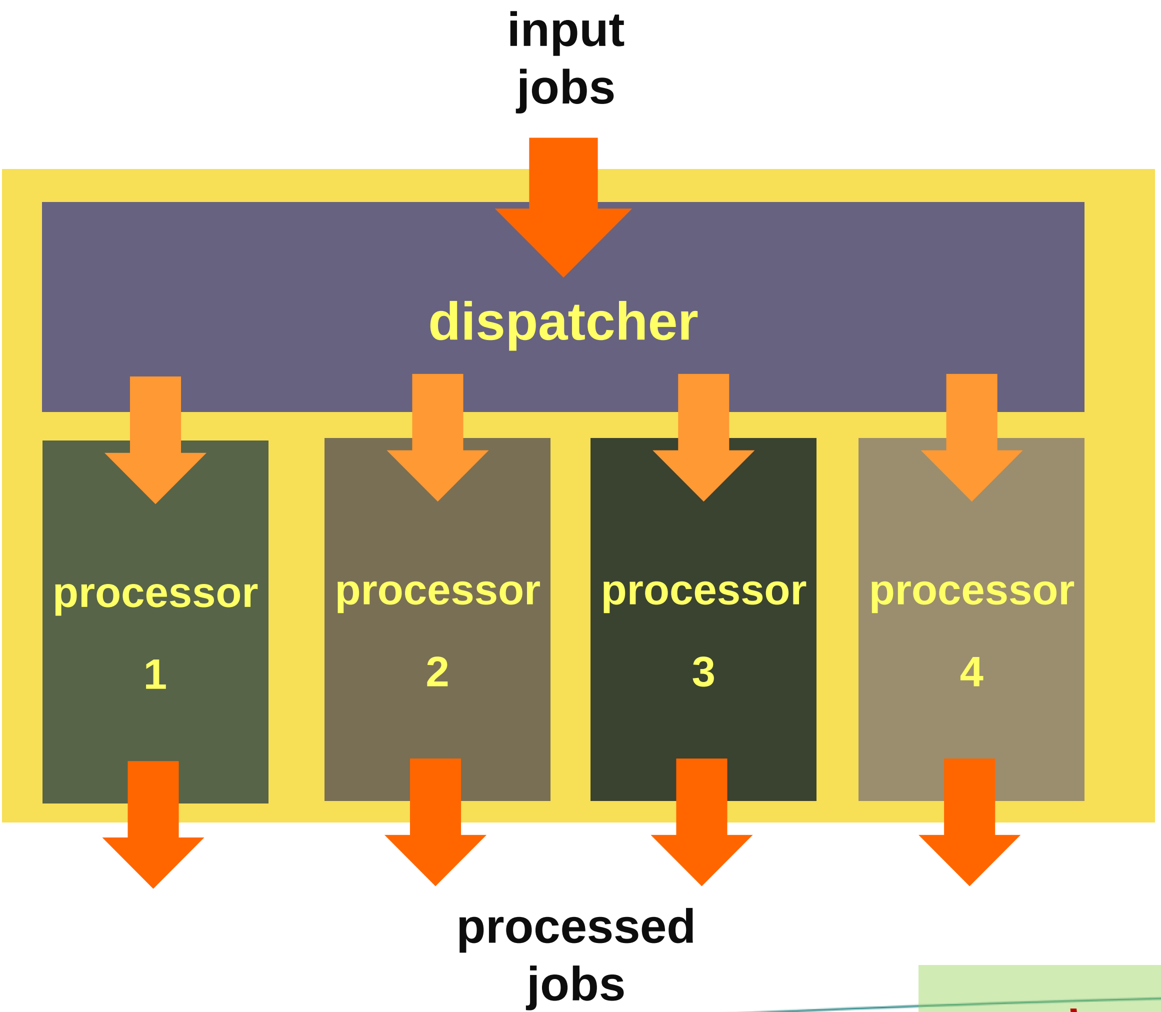
3. More TorXakis Models

4. **Even More TorXakis Models**

TorXakis Example :

Dispatcher-Processing System

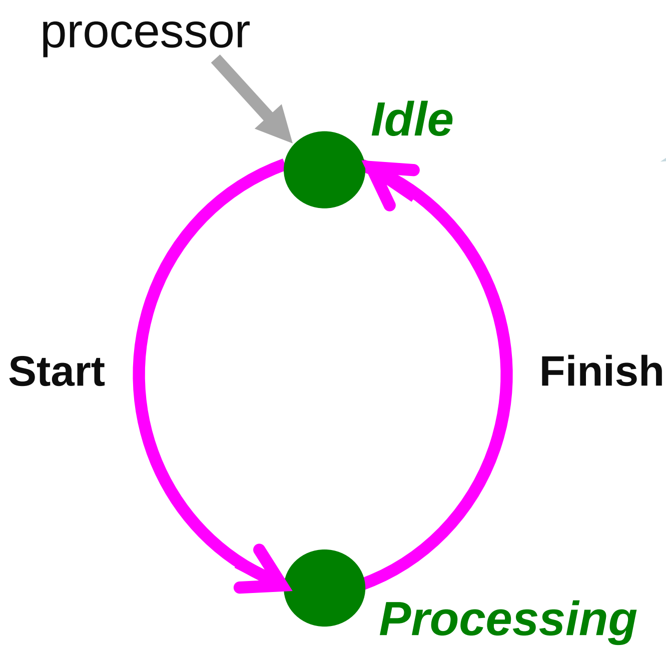
Example: Dispatcher-Processing System



…..\examps\DispatchProcs

Embedded Systems Innovation BY TNO

Example: Dispatcher-Processing System

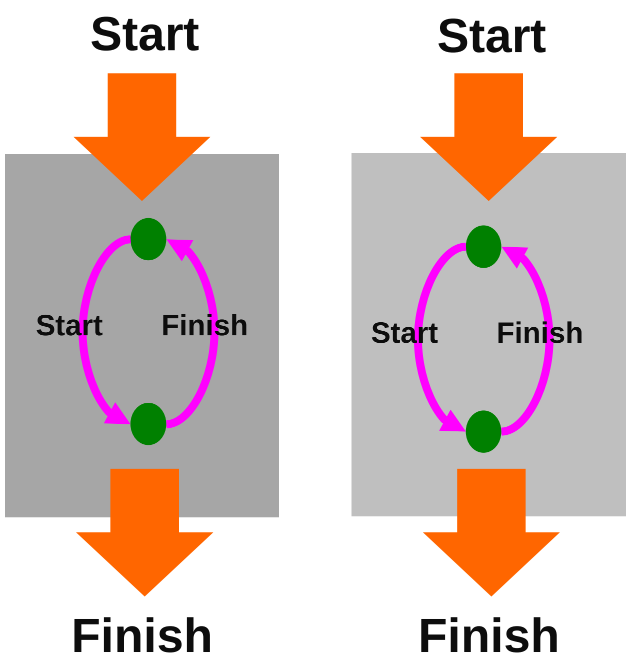


labelled transition system

PROCDEF processor
::=
Start >-> Finish >-> processor
ENDDEF

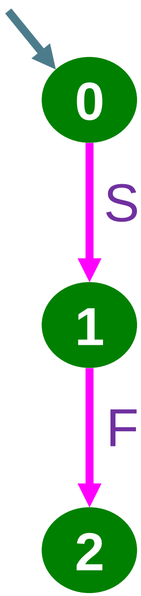
DisPro01-processor.txs

Example: Two Parallel Processors

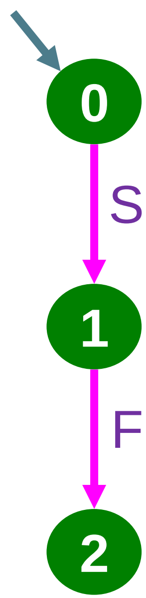


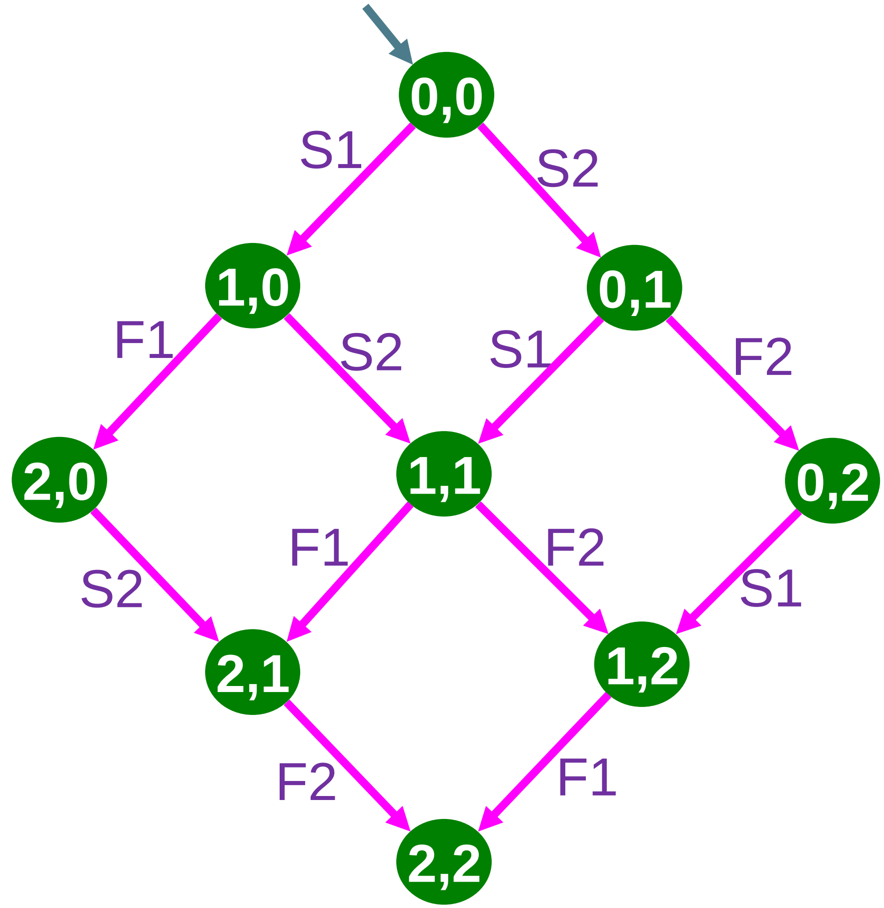
Example: Two Parallel Processors

processor 1



processor 2



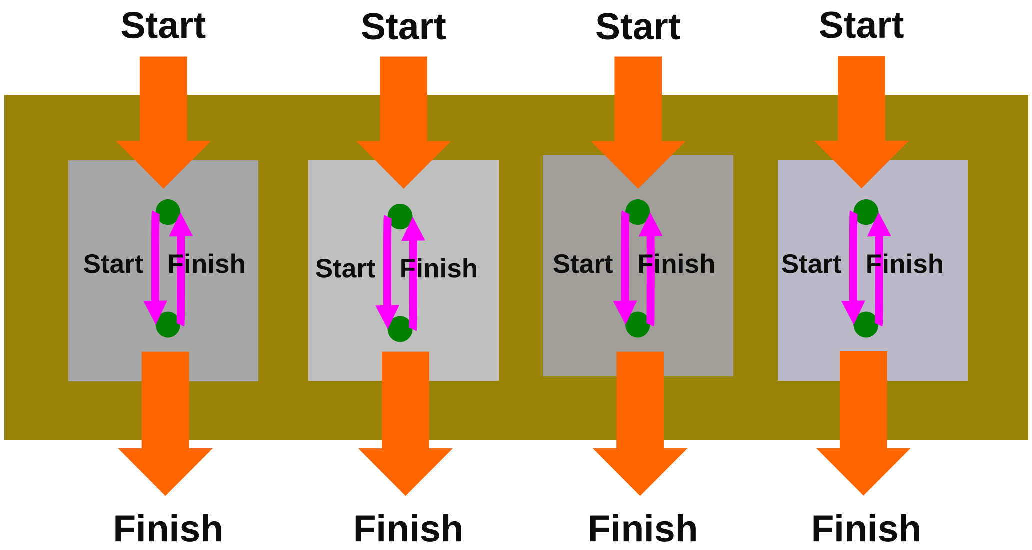


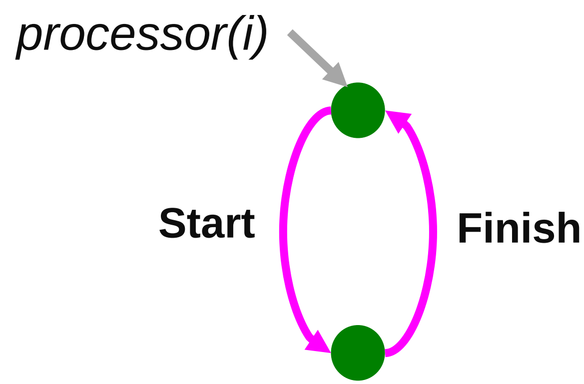
parallelism

processor 1 ||| processor 2

Embedded Systems Innovation BY TNO

Example: Dispatcher-Processing System

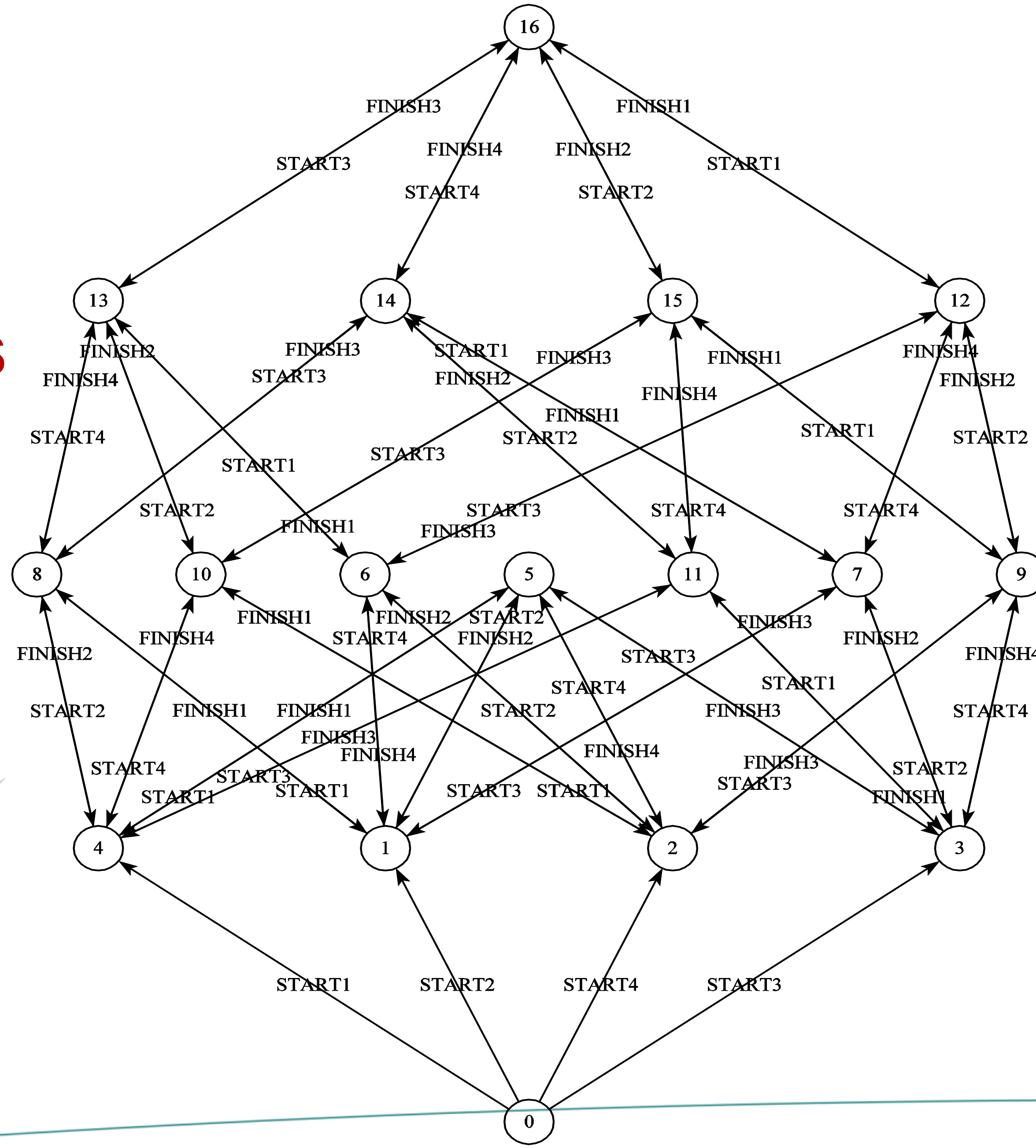




parallel composition

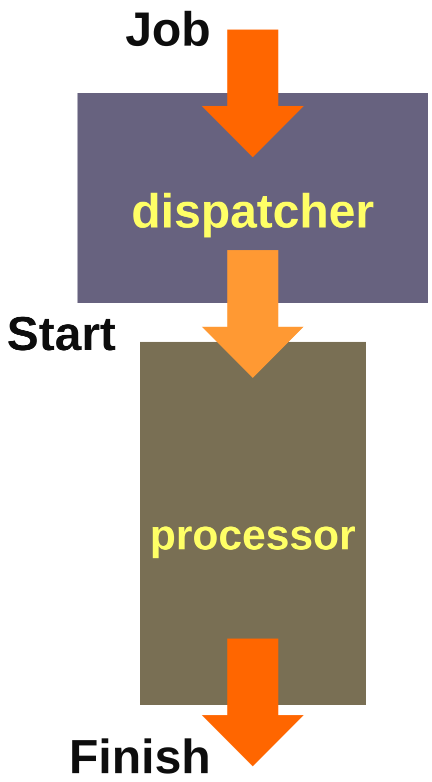
processors
::=
processor(1) ||| processor(2) ||| processor(3) ||| processor(4)

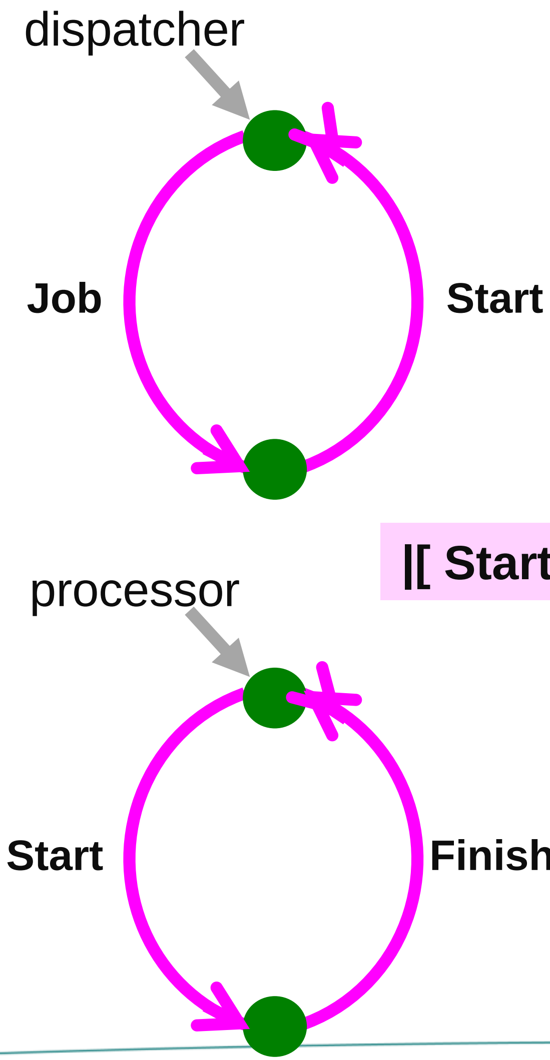
Example: Four Parallel Processors



parallelism

Example: Dispatcher-Processing System

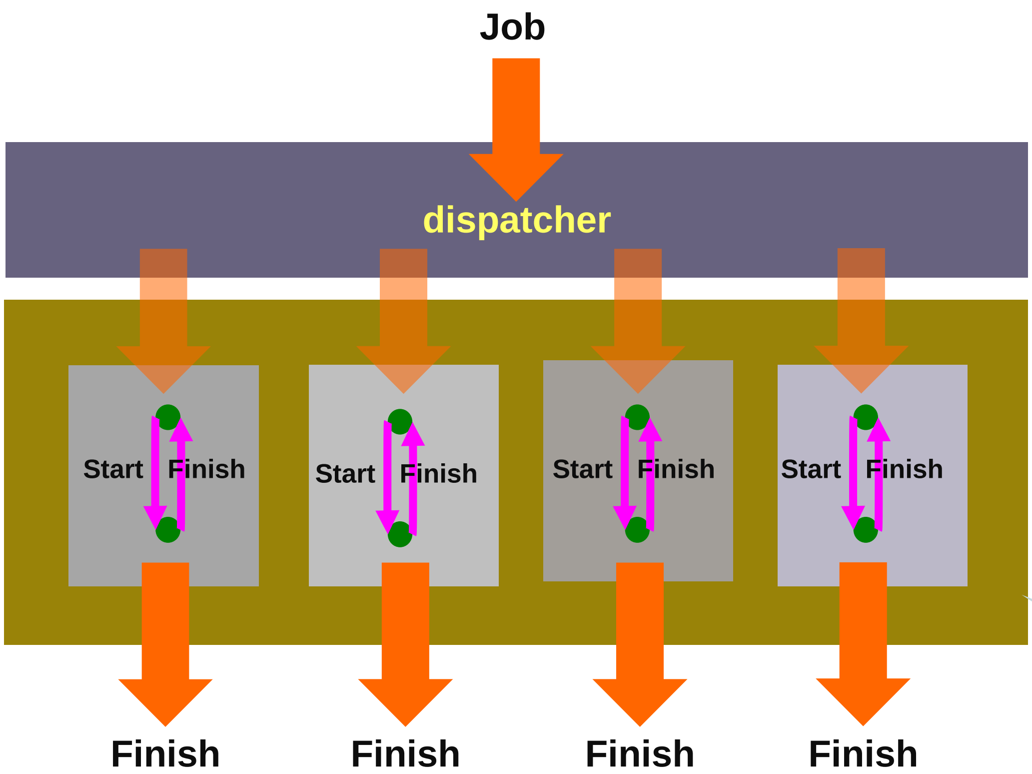


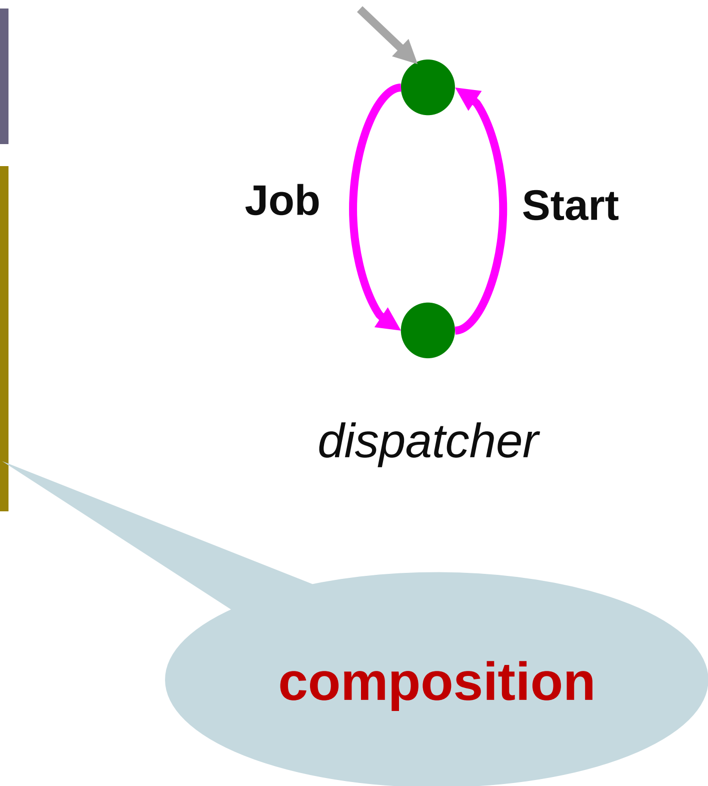


|[Start]|

DisPro02-dispatch.txs

Example: Dispatcher-Processing System

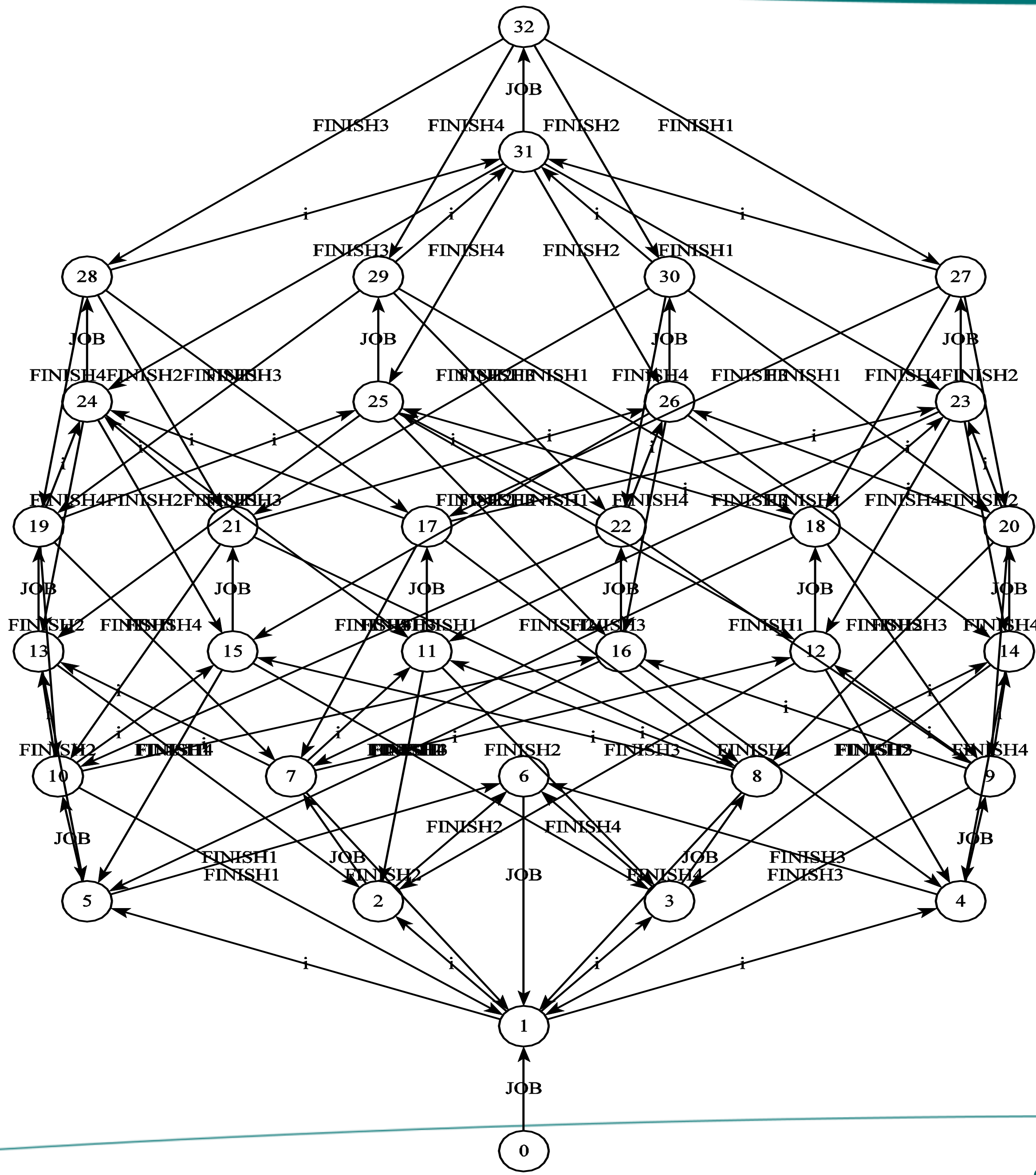




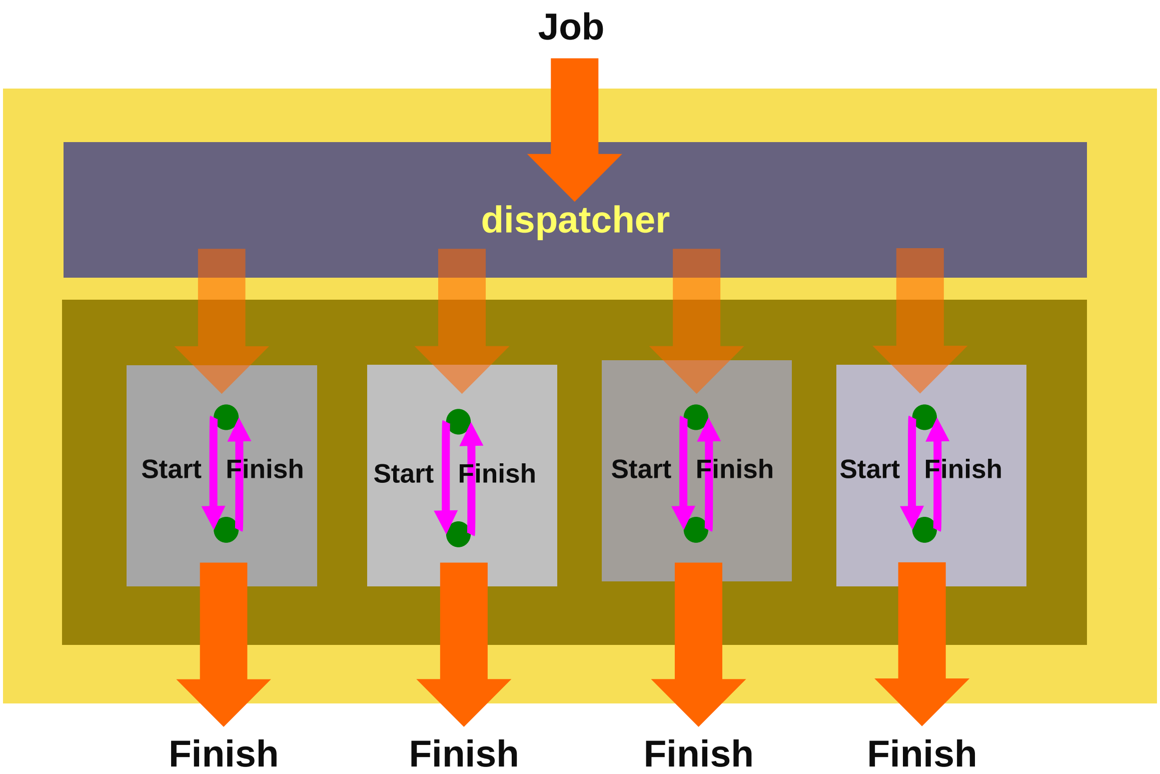
dispatch_procs
::=
dispatcher |[Start]| processors

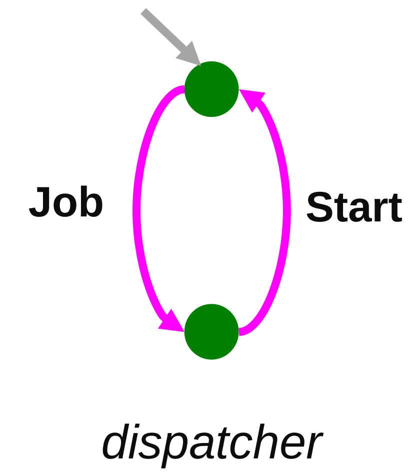
DisPro03-processors.txs

Example: Dispatcher Processing System



Example: Dispatcher-Processing System



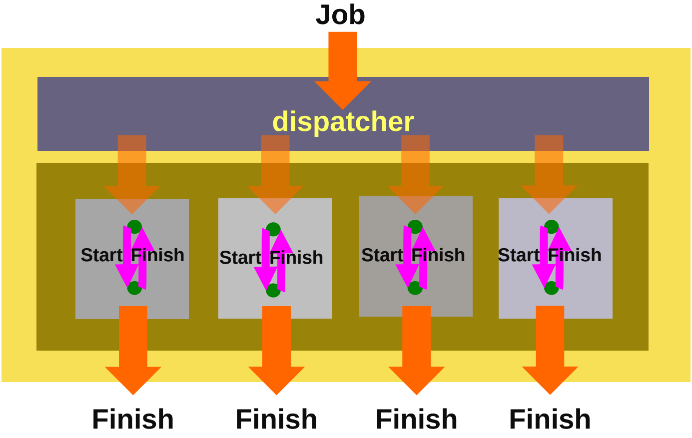


abstraction

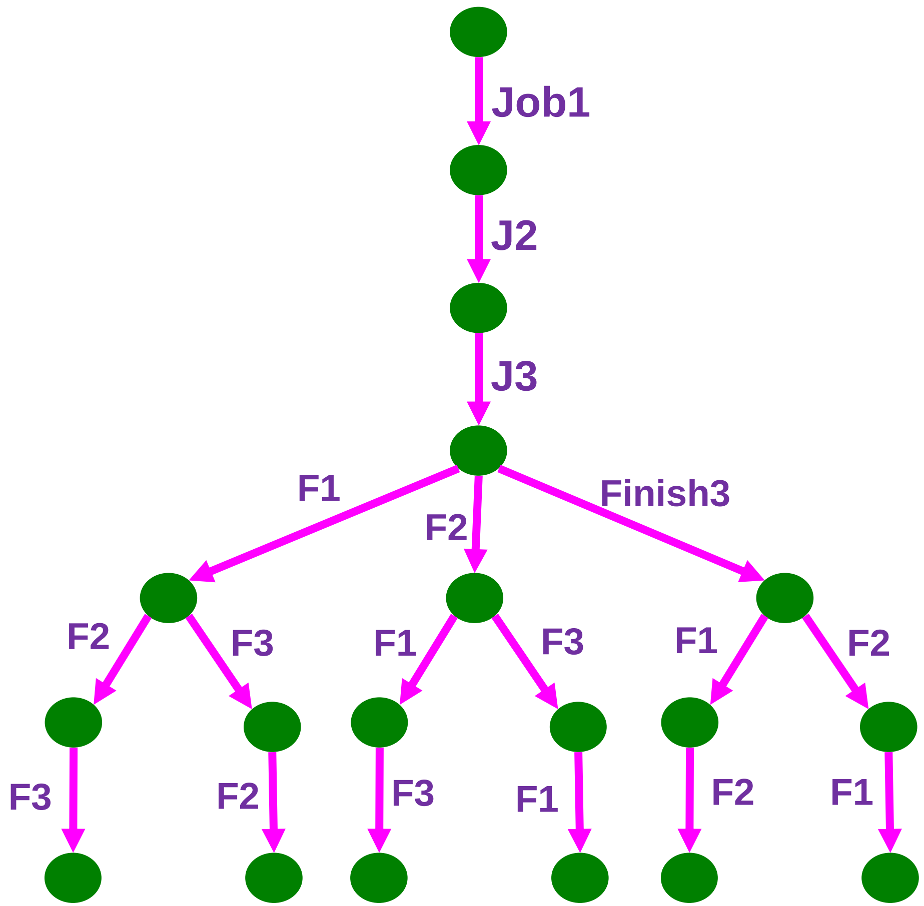
dispatch_procs
::= HIDE [Start]
IN
dispatcher |[Start]| processors
NI

DisPro04-hide.txs

Example: Dispatcher-Processing System

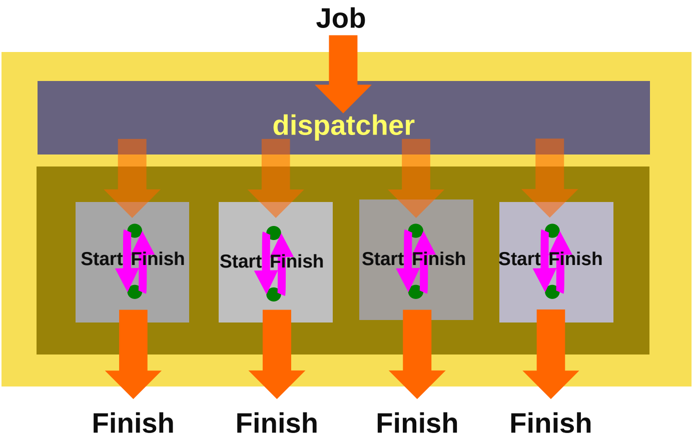


Inputs: Job1, Job2, Job3:

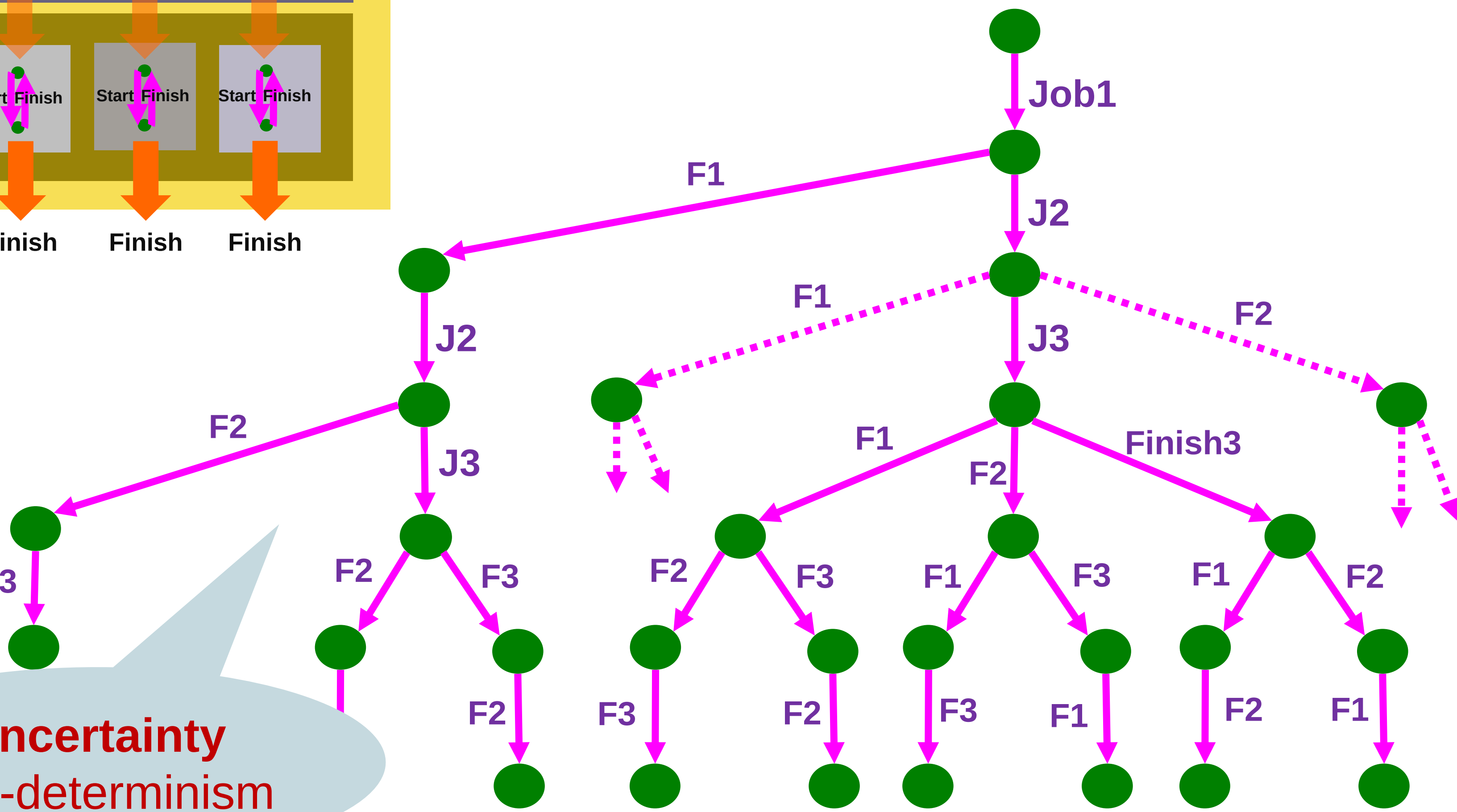


uncertainty
no unique expected result

Example: Dispatcher-Processing System



Inputs: Job1, Job2, Job3:



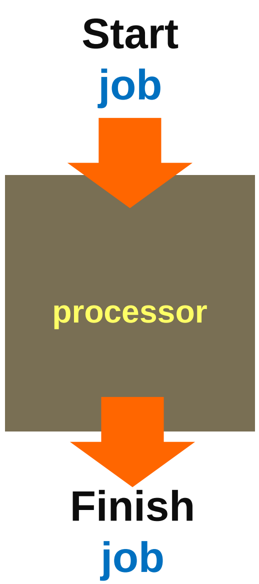
uncertainty
non-determinism

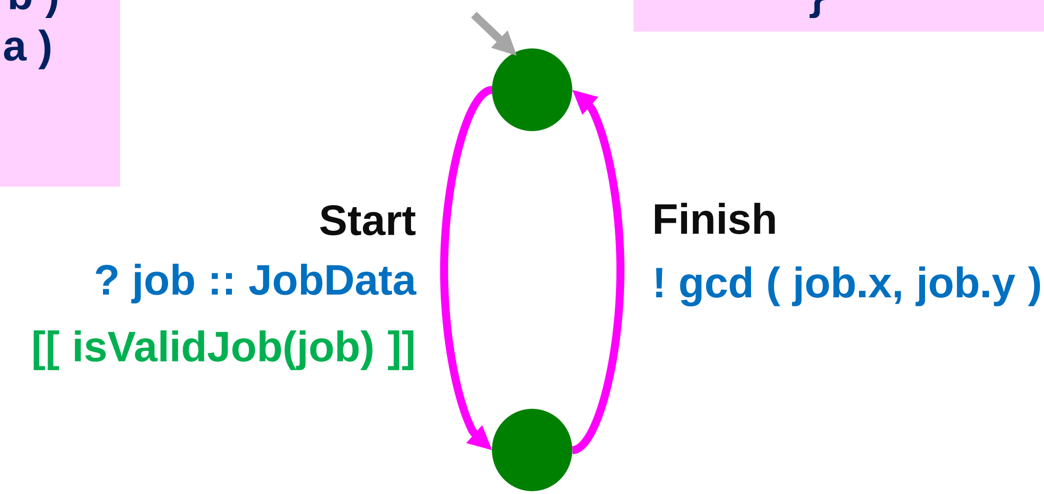
Example: Dispatcher-Processing System

```
FUNCDEF  gcd ( a, b :: Int ) :: Int
 ::=
       IF a == b
       THEN  a
       ELSE  IF a > b
                 THEN  gcd ( a - b, b )
                 ELSE  gcd ( a, b - a )
                 FI
       FI
```

state + data

```
TYPEDEF  JobData
  ::=  JobData
          {  jobId       :: Int
          ;  jobDescr :: String
          ;  x, y          :: Int
          }
```






```
FUNCDEF  isValidJob ( jobdata :: JobData )  :: Bool
  ::=
          jobdata.jobId > 0
     /\  strinre ( jobdata.jobDescr, REGEX('[A-Z][0-9]{2}[a-z]+') )
     /\  . . . . .
```

Embedded Systems Innovation BY TNO

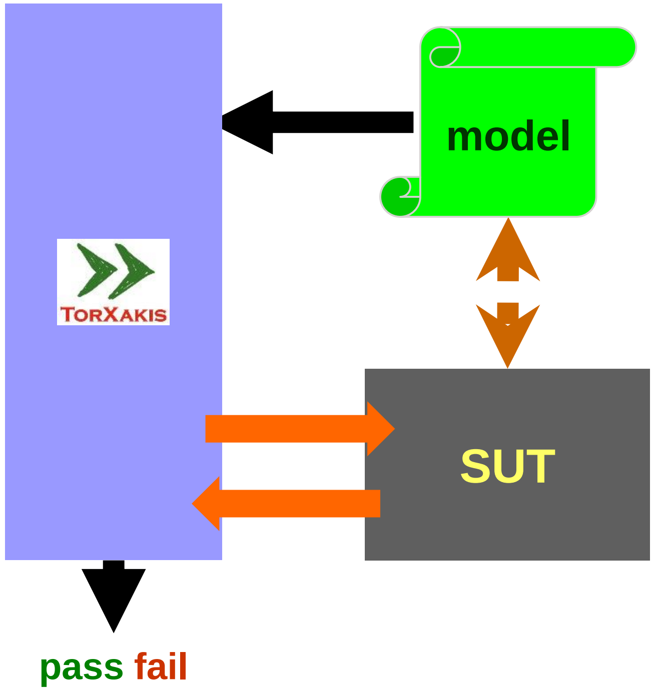
TorXakis Model

```
FUNCDEF  gcd ( a, b :: Int ) :: Int
 ::=
      IF a == b THEN  a
                ELSE  IF a > b THEN  gcd ( a - b, b )
                               ELSE  gcd ( a, b - a )
                               FI
      FI
ENDDEF


--  ----------------------------------------------------------------

PROCDEF  processor [ Start :: JobData; Finish :: JobOut ] ( procnum :: Int )
 ::=
         Start ? job :: JobData
      >->
         Finish ! JobOut ( jobId(job)
                         , procnum
                         , gcd ( x(job) , y(job) )
                         )
      >->
         processor [ Start, Finish ] ( procnum )
ENDDEF


--  ----------------------------------------------------------------

PROCDEF  processors [ Start :: JobData; Finish :: JobOut ] ( procnum :: Int )
 ::=
         processor [ Start, Finish ] ( procnum )
      |||
         [[ procnum > 1 ]]  =>>  processors [ Start, Finish ] ( procnum-1 )
ENDDEF


--  ----------------------------------------------------------------

PROCDEF  dispatcher [ Job, Dispatch :: JobData ] ( )
 ::=
         Job ? job :: JobData  [[ isValidJob(job) ]]
      >->
         Dispatch ! job
      >->
```

Demo: Dispatcher-Processing System

1.	See	...\examps\dispatchprocess\....
2.	Model	...\model\DisPro10-data.txs
3.	Correct SUT:	...\sut\Sut.java
4.	Erroneous SUT:	...\sutWithError\SutWithError.java

TorXakis



The next step in

Model-Based Testing