Model-Based Testing

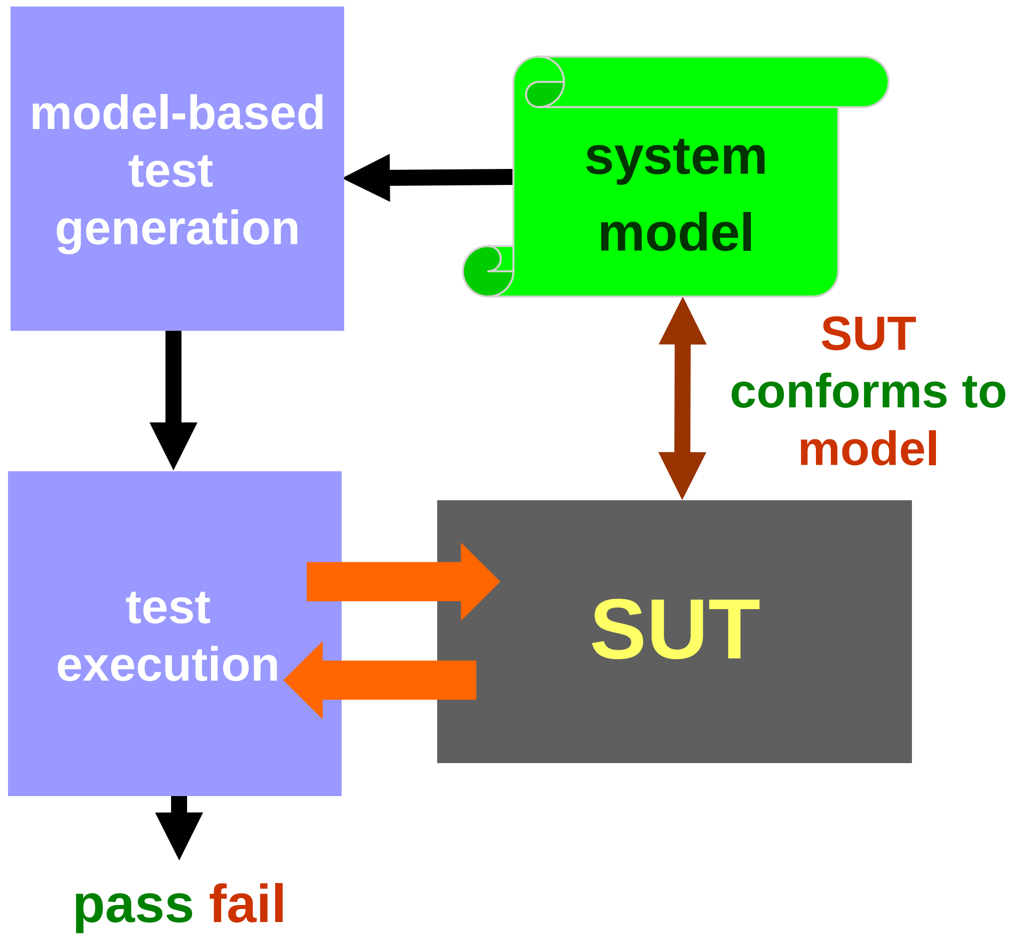
with Labelled Transition Systems

MBT : Model-Based Testing

SUT conforms to model

sound ⇓ ⇑ *exhaustive*

SUT passes tests

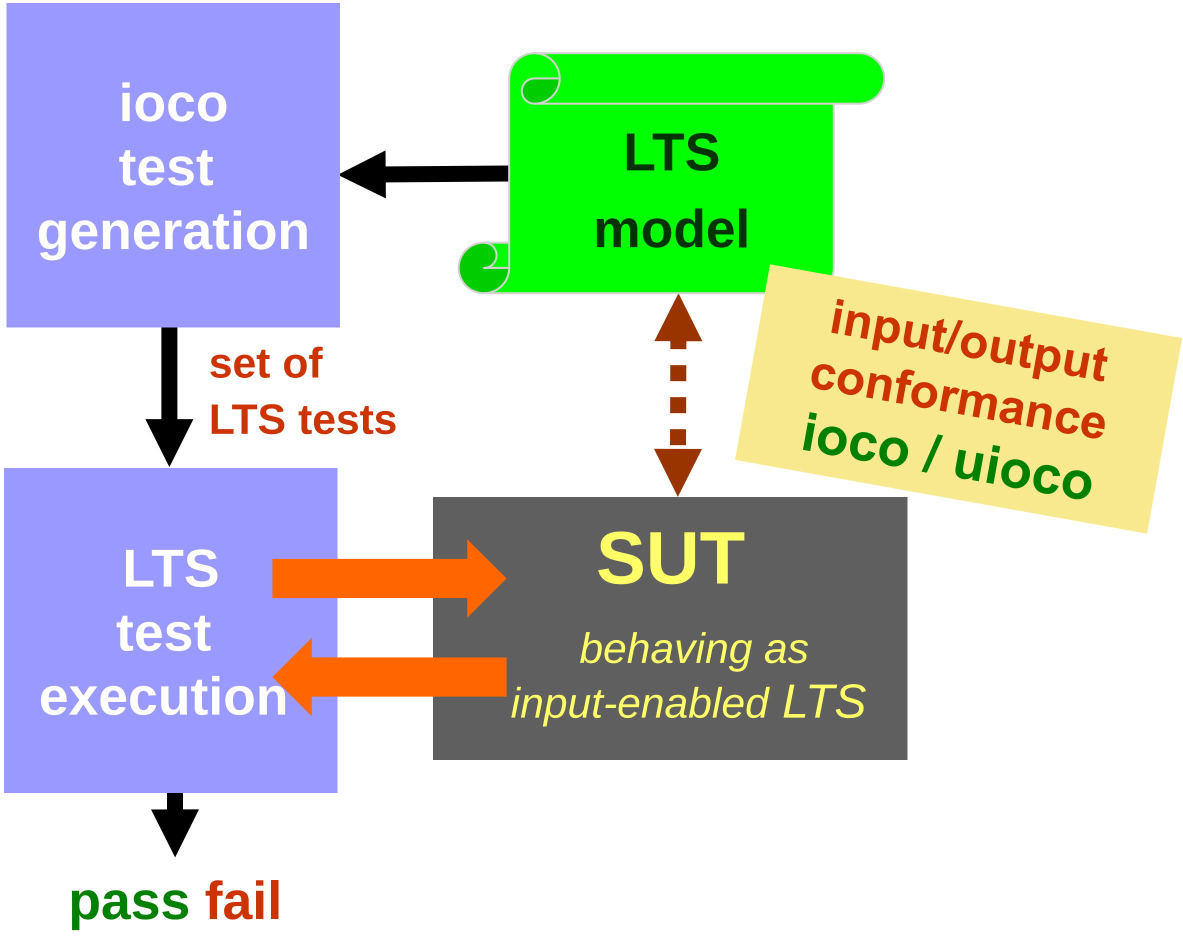


MBT : Labelled Transitions Systems

SUT ioco model

sound ⇓ ⇑ *exhaustive*

SUT passes tests



Models

of Systems

Modelling Methods and Formalisms

- Labelled Transition Systems
- Automata
- Formal Languages
- Petri Nets
- Finite-State Machines
  (Mealy - , Moore Machines)
- (First Order) Properties
- Abstract Data Types
- Streams
- Data Flow Models
- . . . . .

- Functions over Time
- Linear Differential Equations
- PDE
- Simulink Models
- Baysian Networks
- Queueing Networks
- Fault Trees
- Programming Language Models
- Drawings
- Clay Models
- . . . .

Modelling Formalism $\neq$ Modelling Languages

Modelling Systems

- **Traditional systems theory**: (piecewise-) continuous functions of time; analysis and control with ordinary and partial differential equations
- Nowadays, **Digital systems:** discrete, event-driven

SYSTEMS

|

STATIC <----> DYNAMIC

|

TIME-VARYING <---> TIME-INVARIANT

|

LINEAR <---> NONLINEAR

|

CONTINUOUS-STATE <---> DISCRETE-STATE

|

TIME-DRIVEN <---> EVENT-DRIVEN

|

DETERMINISTIC <---> STOCHASTIC

|

DISCRETE-TIME <---> CONTINUOUS-TIME

Cassandras, Lafortune:
Introduction to Discrete Event Systems

Modelling Systems

- **Traditional systems theory**: (piecewise-) continuous functions of time; analysis and control with ordinary and partial differential equations
- Nowadays, **Digital systems:** discrete, event-driven
- Model : “any representation of a system not being the system itself” *(Edward Lee)*

SYSTEMS
|
STATIC <----> DYNAMIC
|
TIME-VARYING <---> TIME-INVARIANT
|
LINEAR <---> NONLINEAR
|
CONTINUOUS-STATE <---> DISCRETE-STATE
|
TIME-DRIVEN <---> EVENT-DRIVEN
|
DETERMINISTIC <---> STOCHASTIC
|
DISCRETE-TIME <---> CONTINUOUS-TIME

Cassandras, Lafortune:
Introduction to Discrete Event Systems

Model $\neq$ System

Modelling Systems

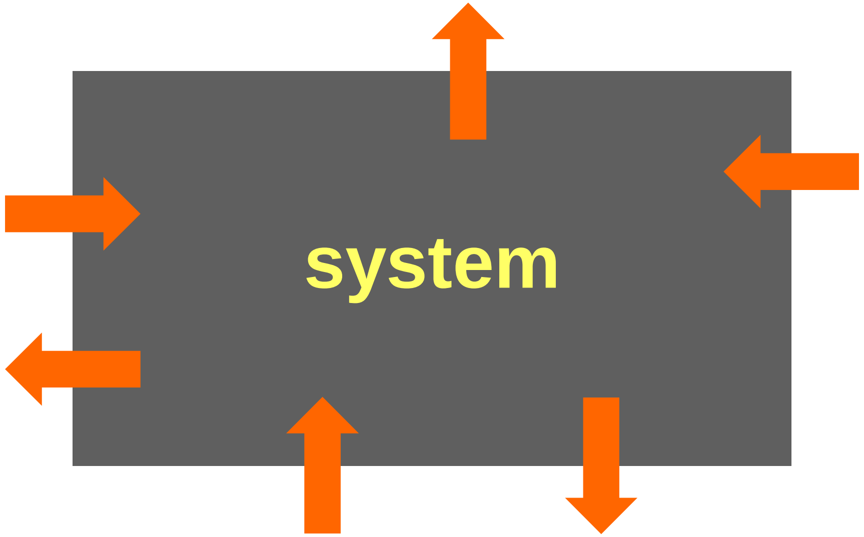
Model $\neq$ System

- **System** : “something real”
- **Model : ** an “abstraction”
- Model : “any representation of a system not being the system itself” *(Edward Lee)*
- By choosing a model, or a class of models, or a modeling formalism,ou give a view on the system (and restrict the properties under consideration)
- *A system is not continuous or discrete, a model is*
- One system has many models:
  - quality characteristics
  - abstraction levels
  - prescriptive ←→ descriptive
  - black-box / functional ←→ white-box / structural
  - . . . . .

MBT : System Modelling for MBT

Our systems are digital systems:

- *discrete, event-driven,*
- *reactive, dynamic,*
- *data-intensive,*
- *black-box*



Typical modelling formalisms:

- automata
- formal languages, grammar
- labelled transition system
- symbolic transition systems
- (extended) finite-state machine
- petri nets
- . . . . .

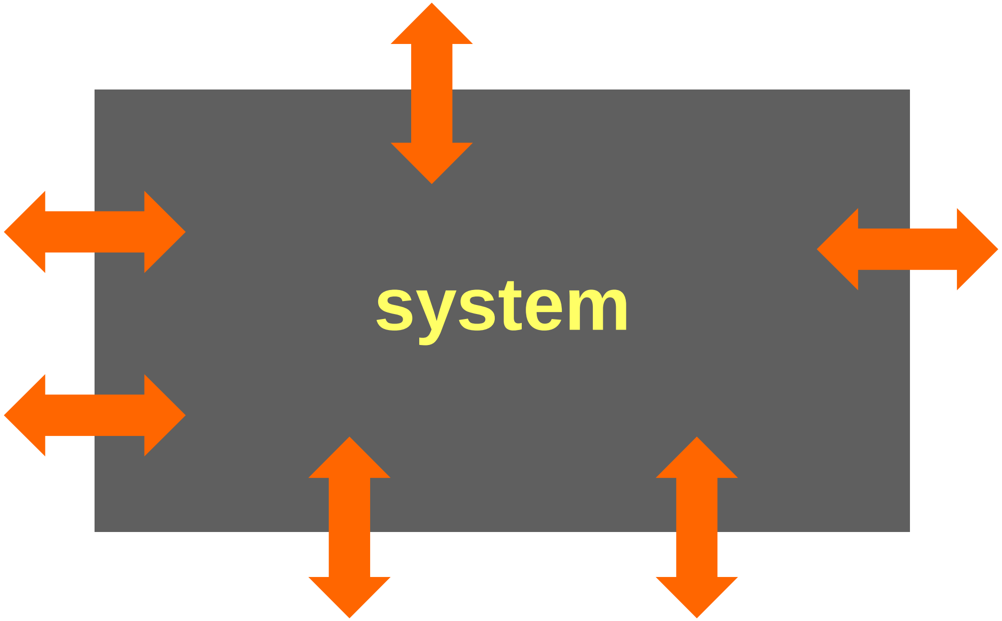
View on a system:

black box, with discrete,

atomic events on interfaces:

- inputs, initiated by environment
- outputs, initiated by the system

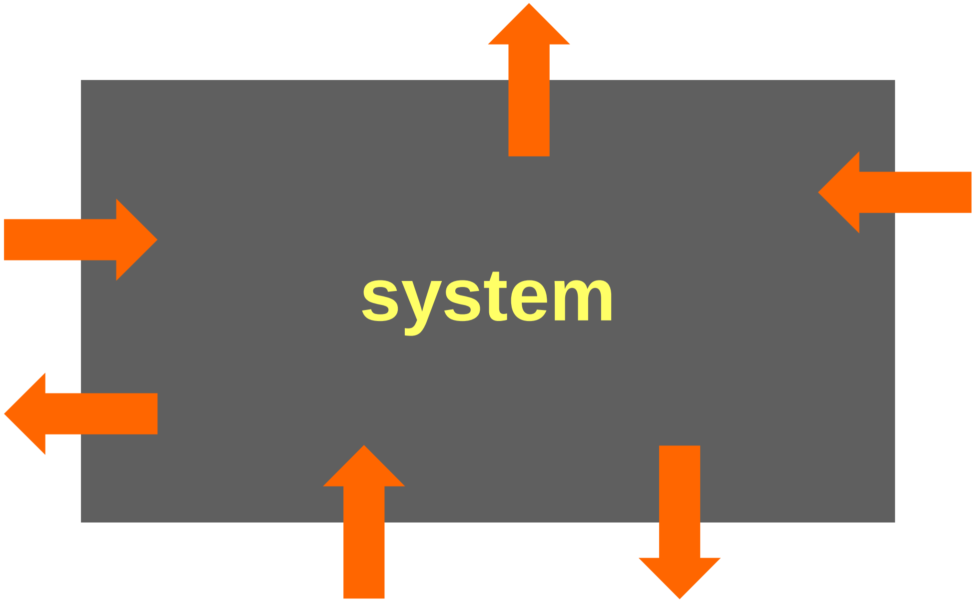
MBT : Abstract System Modelling



System

- ☞ black box
- ☞ abstract interactions on interfaces

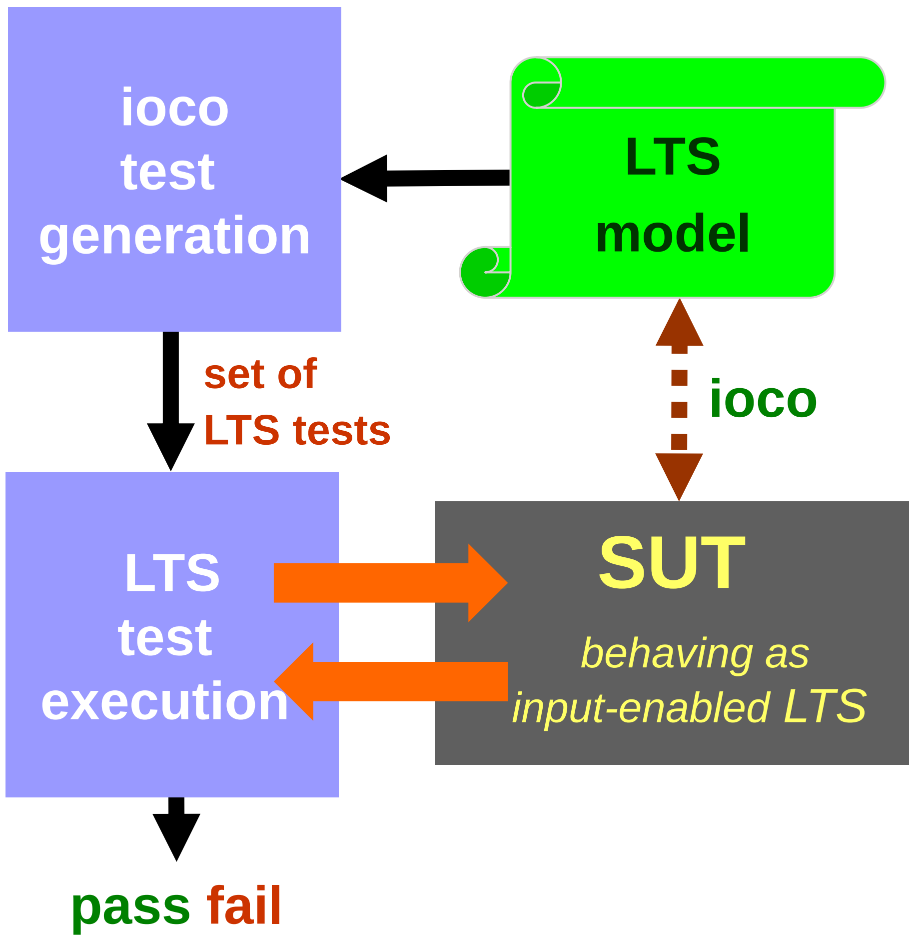
MBT : System Modelling



System

- ☞ black box
- ☞ inputs
  on interfaces
- ☞ outputs
  on interfaces

MBT : Labelled Transitions Systems



MBT with LTS topics:

- ☞ specification model
- ☞ implementation (SUT)
- ☞ SUT model
- ☞ correctness **ioco**
- ☞ test cases
- ☞ test generation
- ☞ test execution
- ☞ test result analysis

Model-Based Testing :

Labelled Transition Systems

Models: Labelled Transition Systems

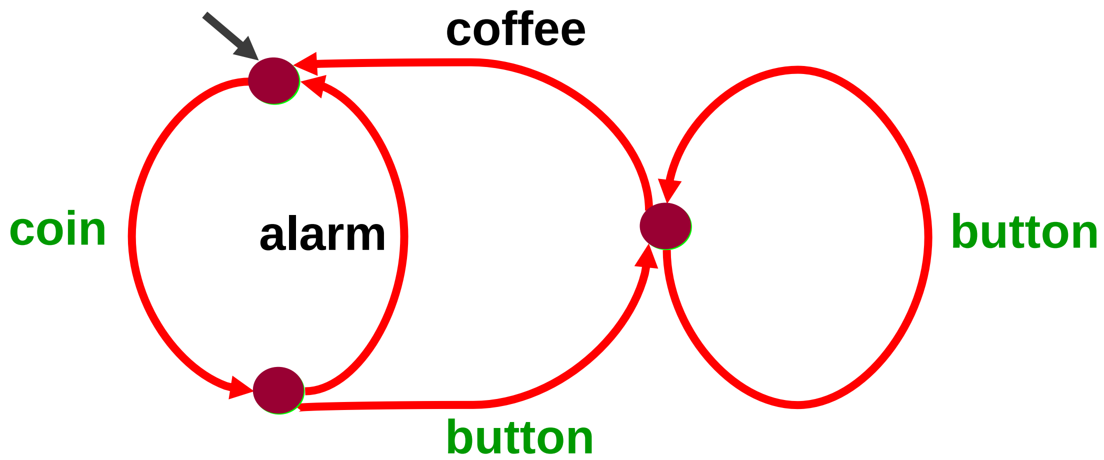
Labelled Transition System: $\langle$ **S**, **L**, **T**, $\mathbf{s_0}$ $\rangle$

set of states

set of labels

transitions $T \subseteq S \times (L \cup \{\tau\}) \times S$

initial state $s_0 \in S$



Models: Labelled Transition Systems

Labelled Transition System: $\langle\ S,\ L_I, L_U, T,\ s_0\ \rangle$

states

input labels

output labels

transitions
$T \subseteq S \times (L \cup \{\tau\}) \times S$

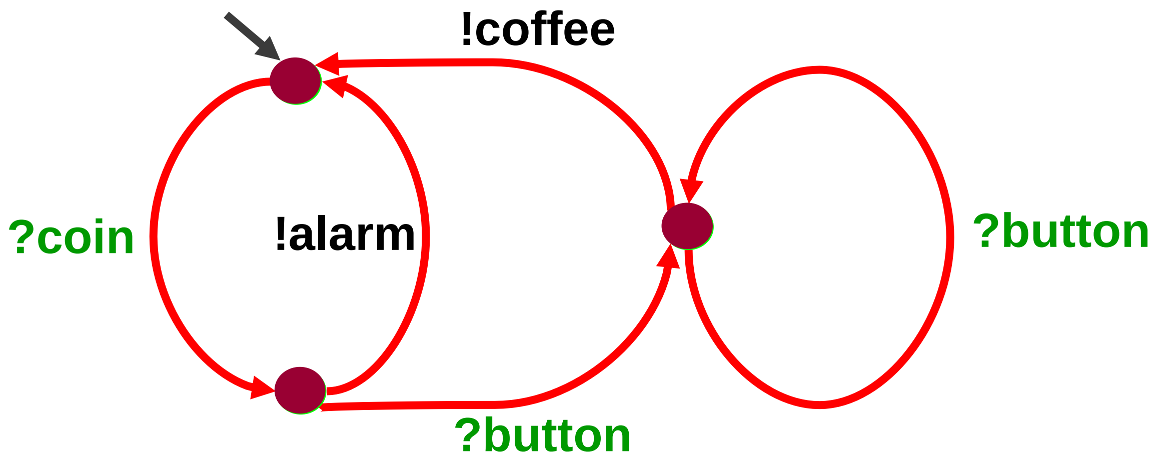
initial state
$s_0 \in S$

? = input

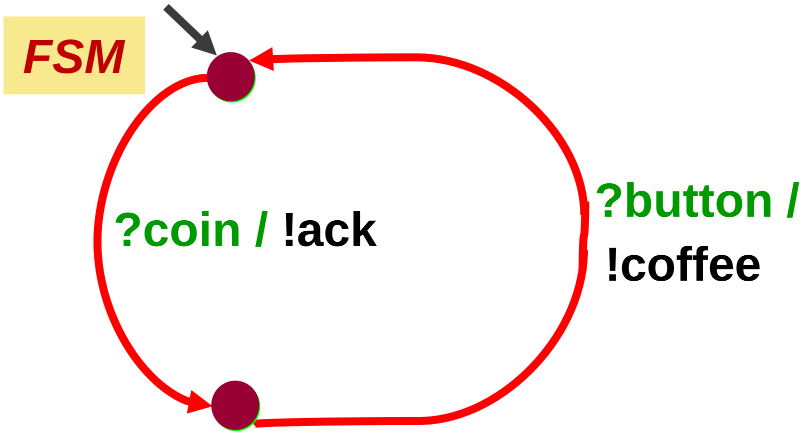
! = output

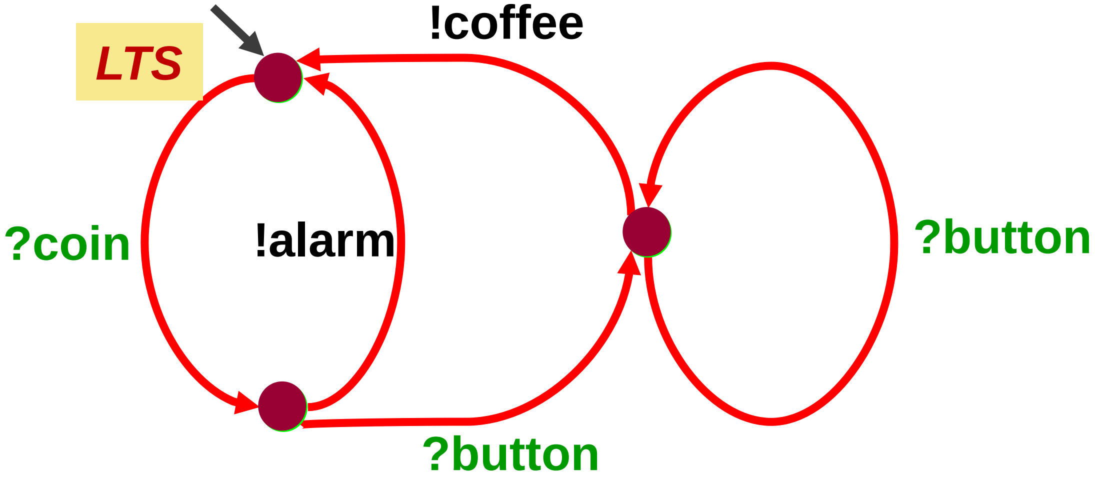
$L = L_I \cup L_U$

$L_I \cap L_U = \emptyset$

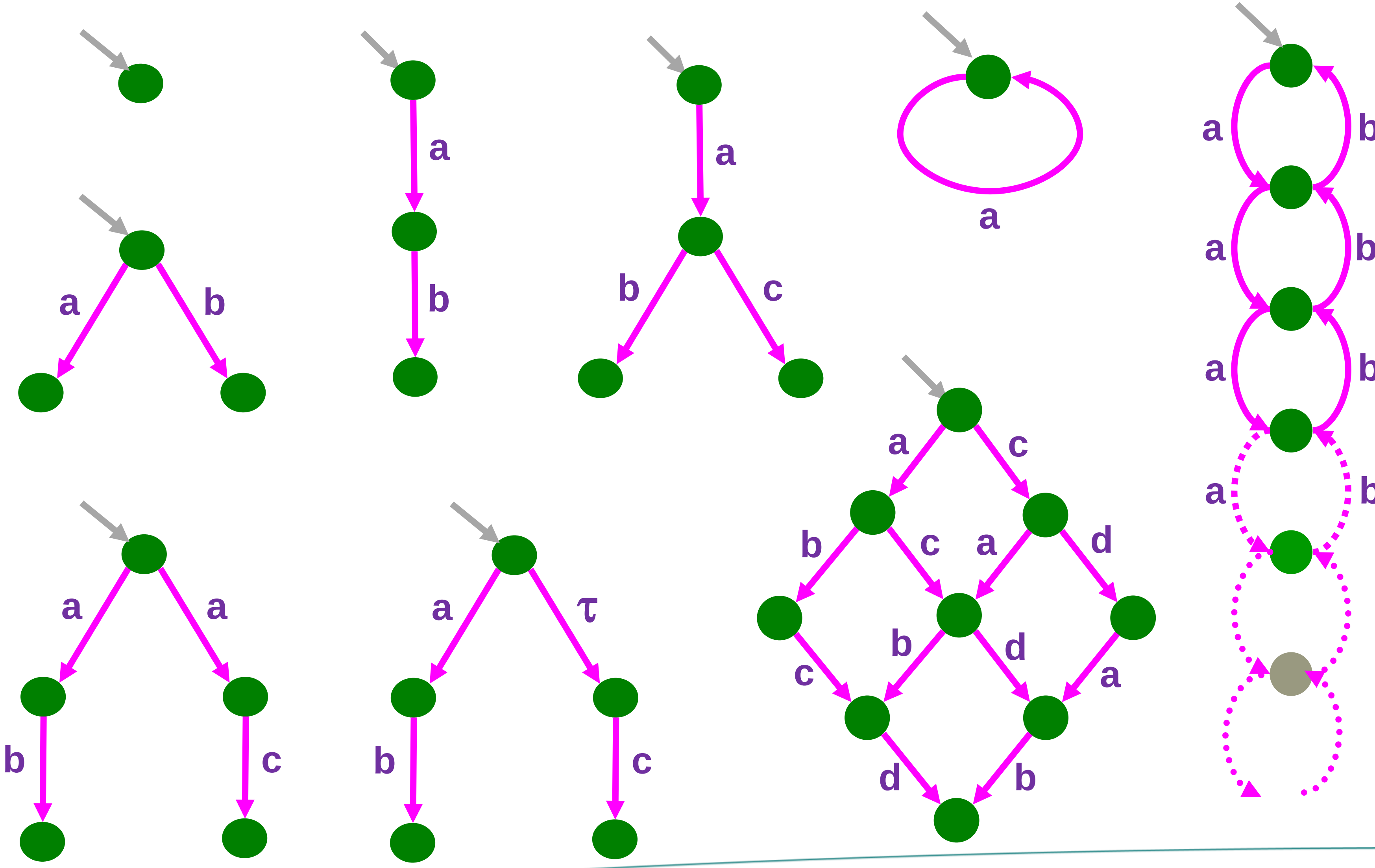


Models: FSM or LTS ?

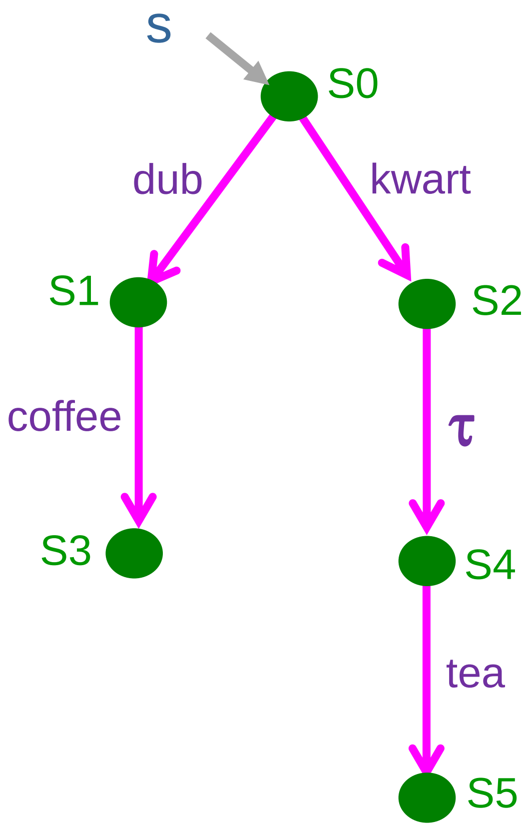




LTS : Examples



LTS : Reasoning



$S0 \xrightarrow{\text{dub}} S1$ transition

$S0 \xRightarrow{\text{dub coffee}} S3$ transition composition

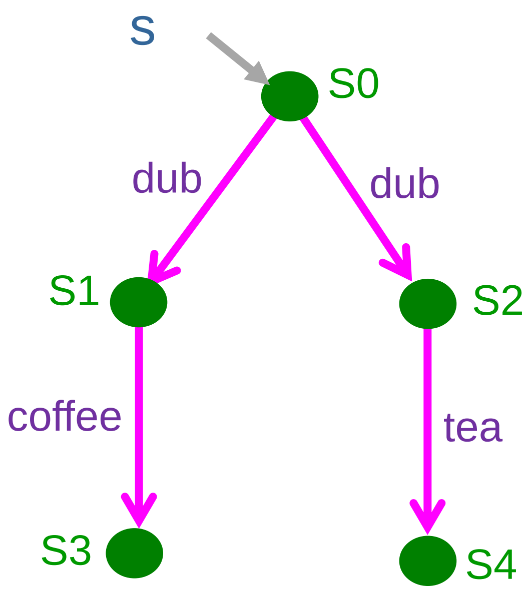
$S0 \xRightarrow{\text{kwart tea}}$ executable sequence

$S0 \not\xRightarrow{\text{kwart soup}}$ non-executable sequence

L = { dub, kwart,
coffee, tea, soup}

LTS(L) all transition systems over L

LTS : Reasoning



L = { dub, kwart,
coffee, tea, soup}

Sequences of observable actions:

traces (s) = $\{ \ \sigma \in L^* \mid s \overset{\sigma}{\Longrightarrow} \ \}$

traces(s) = { ε, dub, dub coffee, dub tea }

Reachable states:

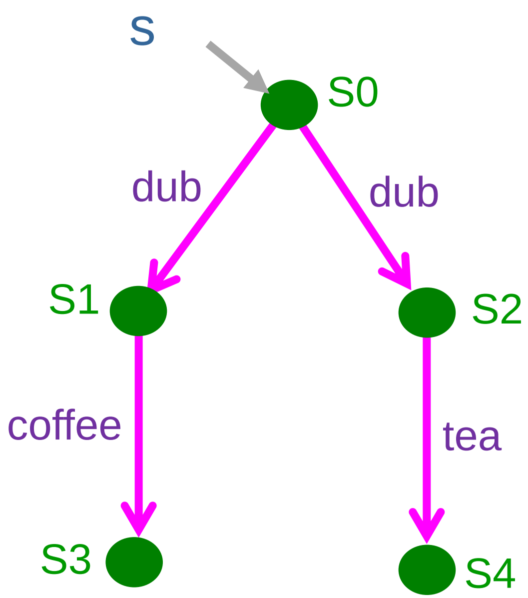
s **after** σ = $\{ \ s' \mid s \overset{\sigma}{\Longrightarrow} s' \ \}$

s **after** dub = { S1, S2 }

s **after** dub tea = { S4 }

s **after** dub soup = $\emptyset$

LTS : Reasoning



L = { dub, kwart,
coffee, tea, soup}

Refusals sets:

s after σ refuses ***A*** $\Leftrightarrow$

$\exists\, s' : s \xRightarrow{\sigma} s'$ and $\forall\, \mu \in A \cup \{\tau\} : s' \overset{\mu}{\not\longrightarrow}$

S0 **after** ε **refuses** {coffee, tea}

S0 **after** dub **refuses** {tea}

S0 **after** dub tea **refuses** L

S0 **after** dub **refuses** Ø

not S0 **after** dub **refuses** {coffee, tea}

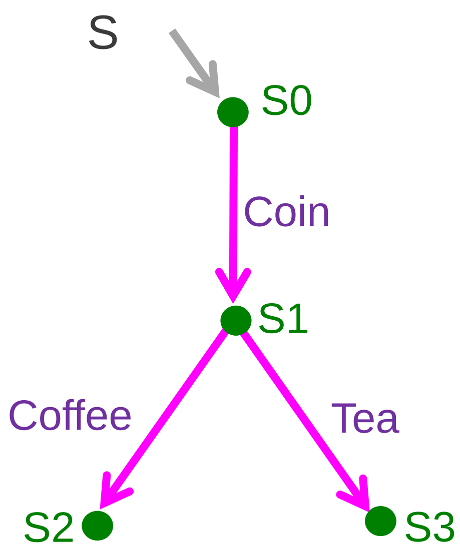
not S0 **after** tea **refuses** Ø

Model-Based Testing :

Labelled Transition Systems

Representations

LTS : Representation



- Explicit : ⟨ { S0, S1, S2, S3 } ,

  { Coin, Coffee, Tea } ,

  { (S0, Coin, S1),

  (S1, Coffee, S2),

  (S1, Tea, S3) } ,

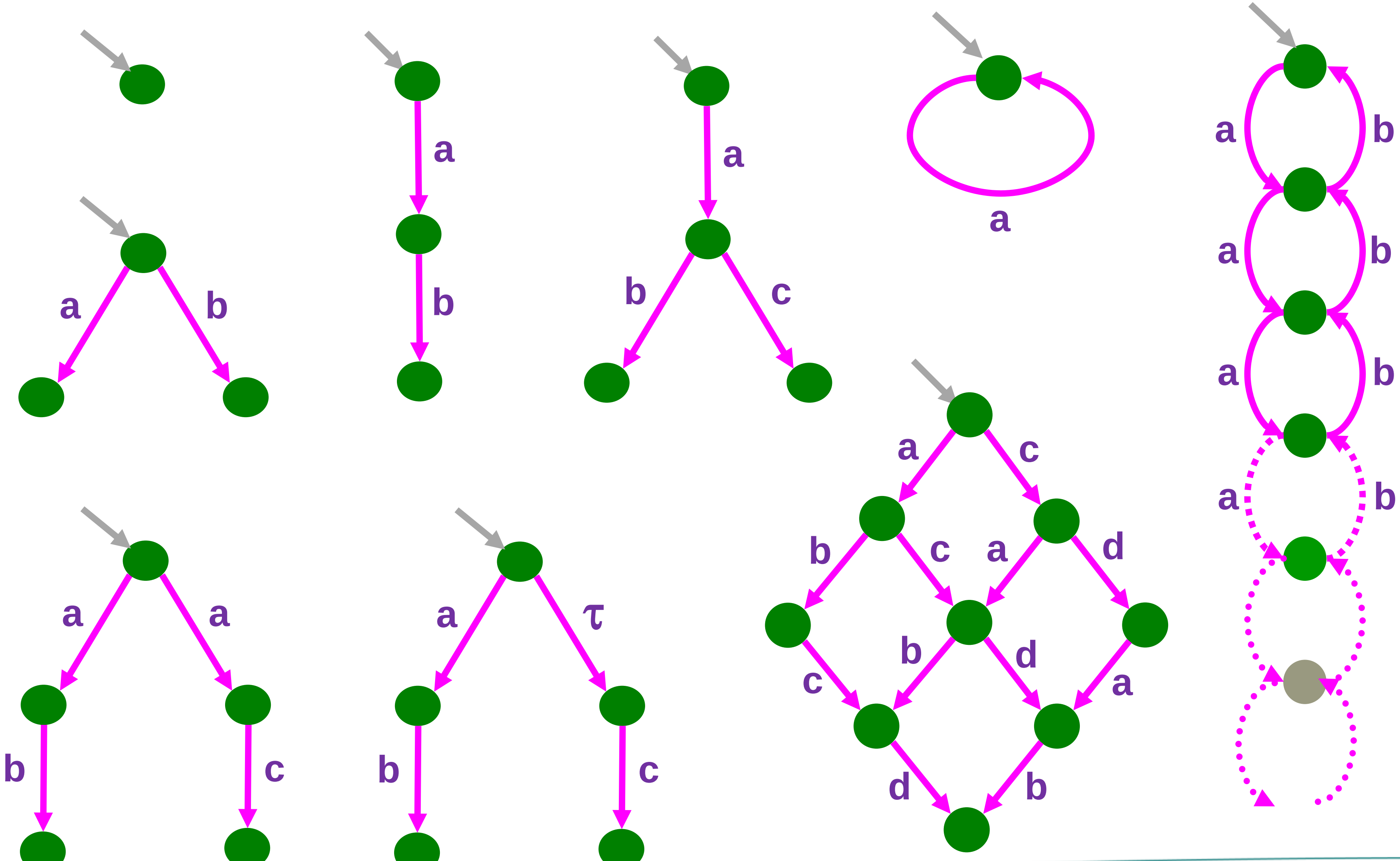
  S0 ⟩

- Transition tree / graph

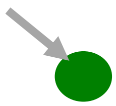
- Language :

S ::= Coin >-> (Coffee ## Tea)

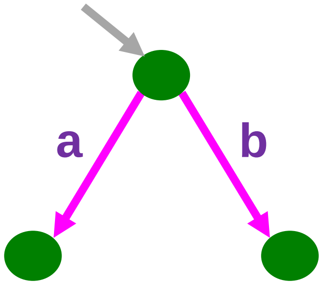
LTS : Examples



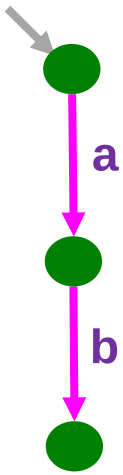
LTS : Representation



STOP

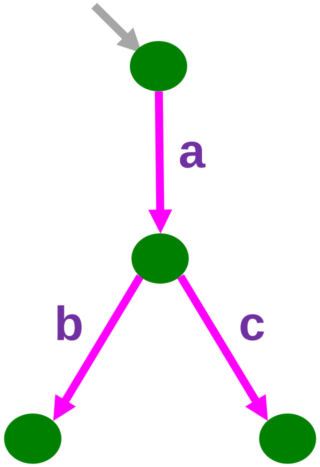


a ## b

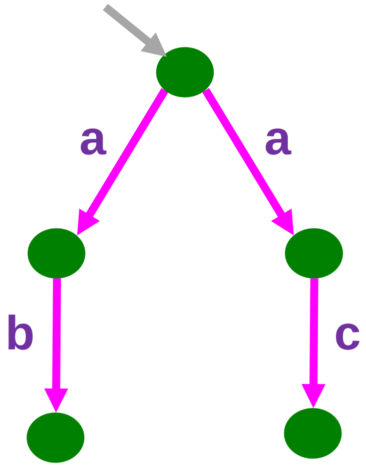


a >-> b

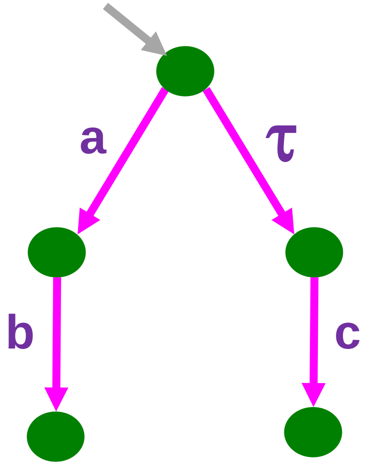
LTS : Representation



a >-> (b ## c)



a >-> b ## a >-> c

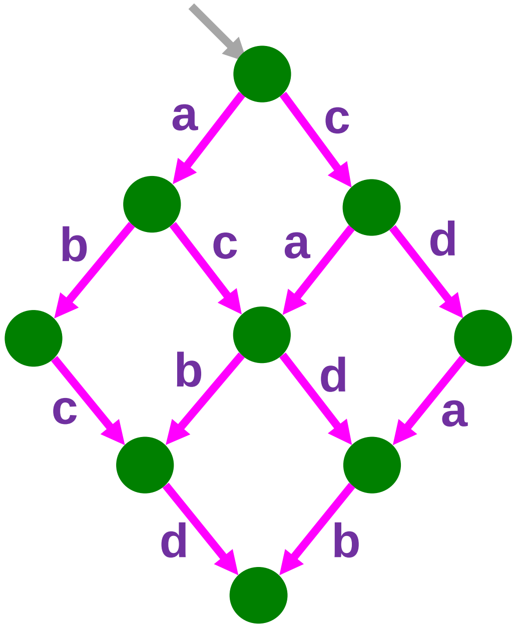


a >-> b ## ISTEP >-> c

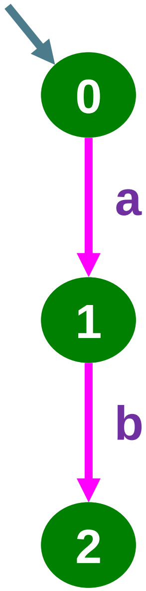
LTS : Representation

a >-> (b >-> c >-> d ## c >-> (b >-> d ## d >-> b))
##
c >-> (d >-> a >-> b ## a >-> (b >-> d ## d >-> b))

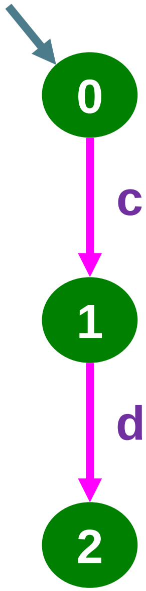
a >-> b ||| c >-> d



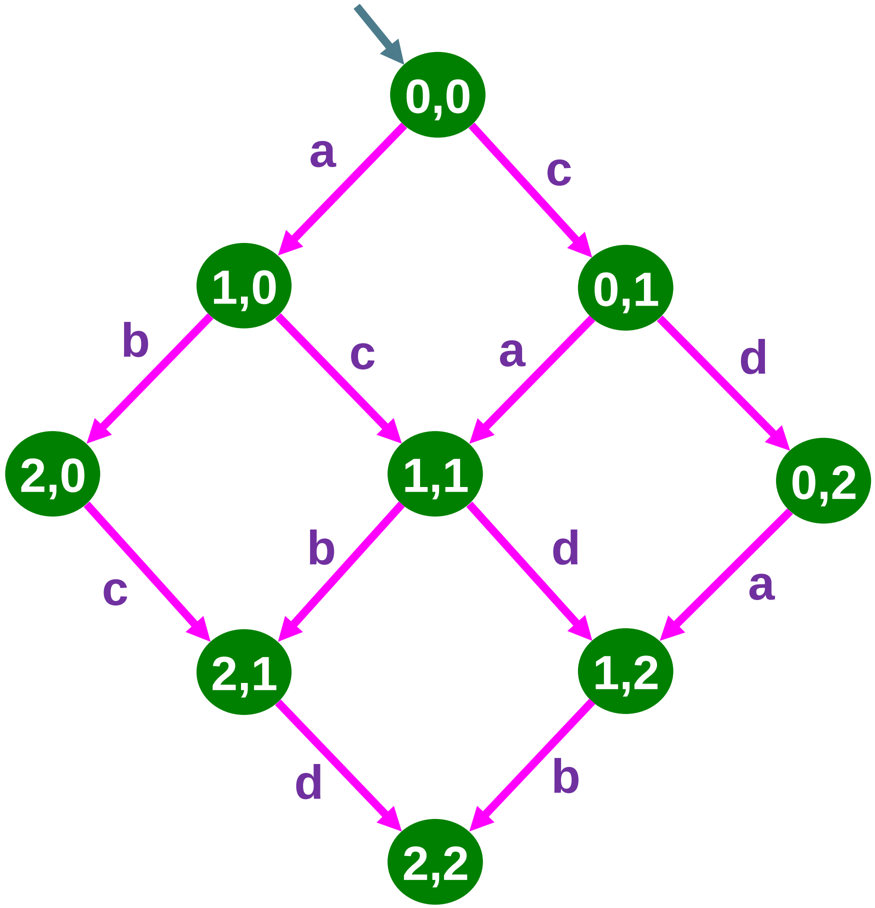
LTS : Representation



a >-> b

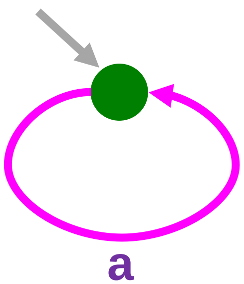


c >-> d



a >-> b ||| c >-> d

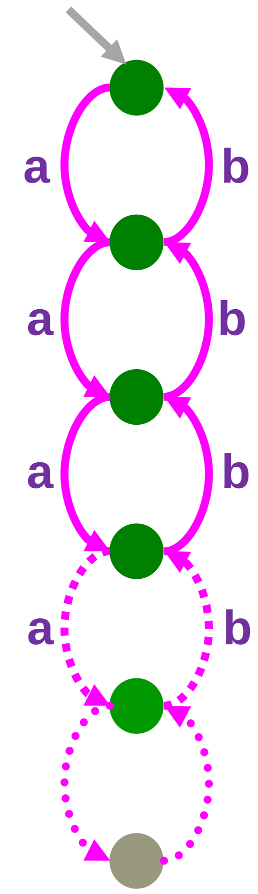
LTS : Representation



Q

where

Q ::= a >-> (b ||| Q)



P

where

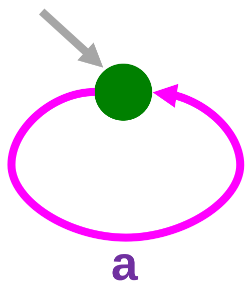
P ::= a >-> P

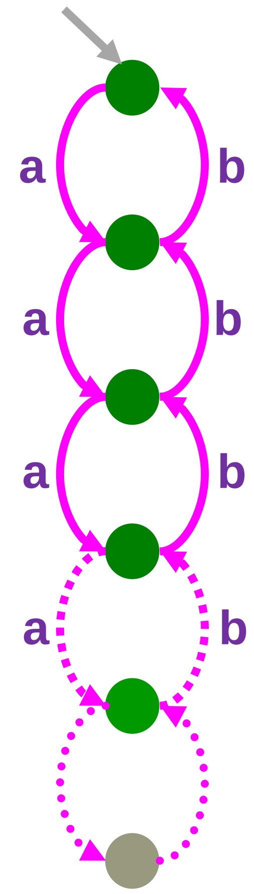
LTS : Representation

```
Q

where

    Q  ::=  a >-> ( b ||| Q )
```

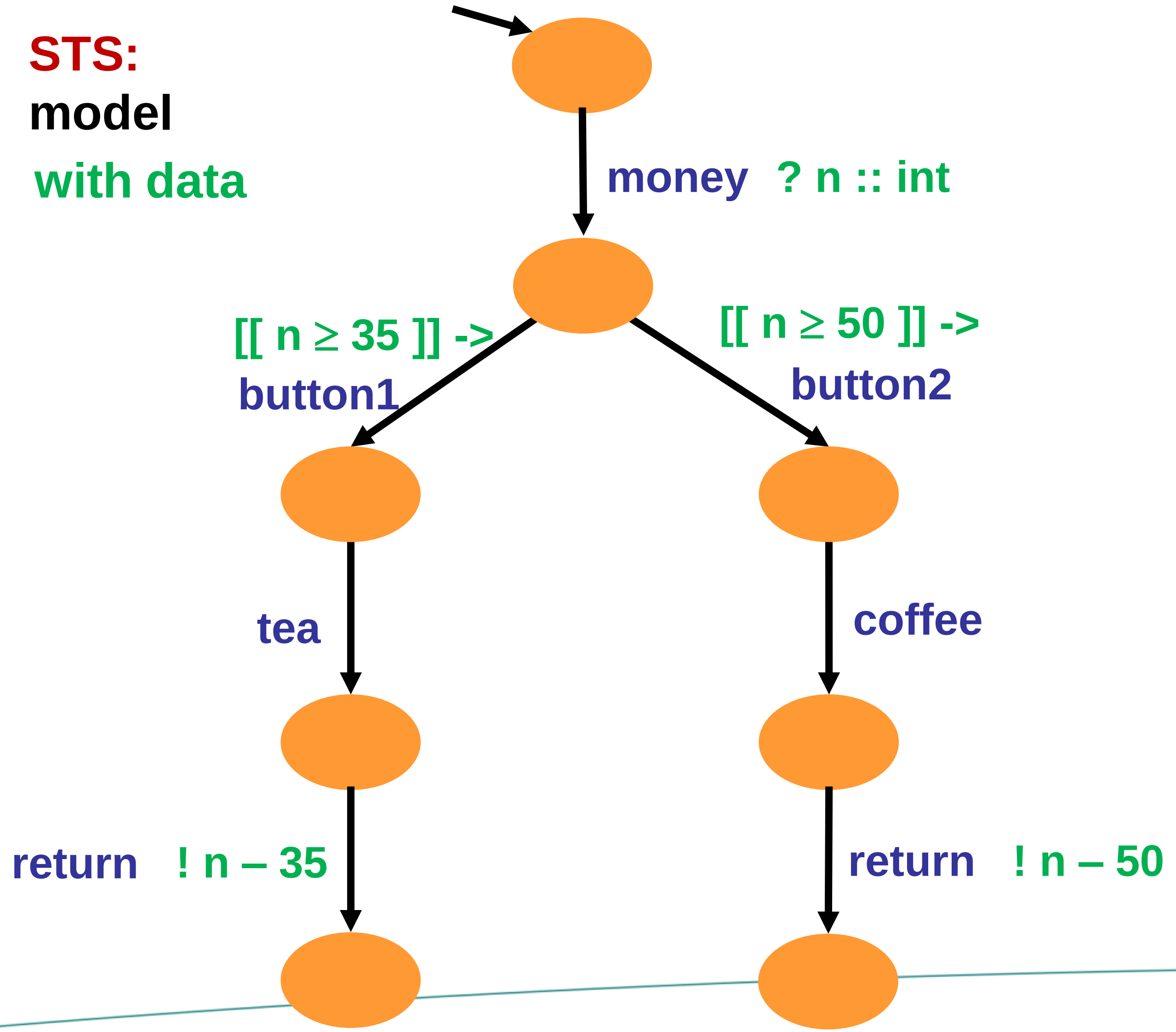




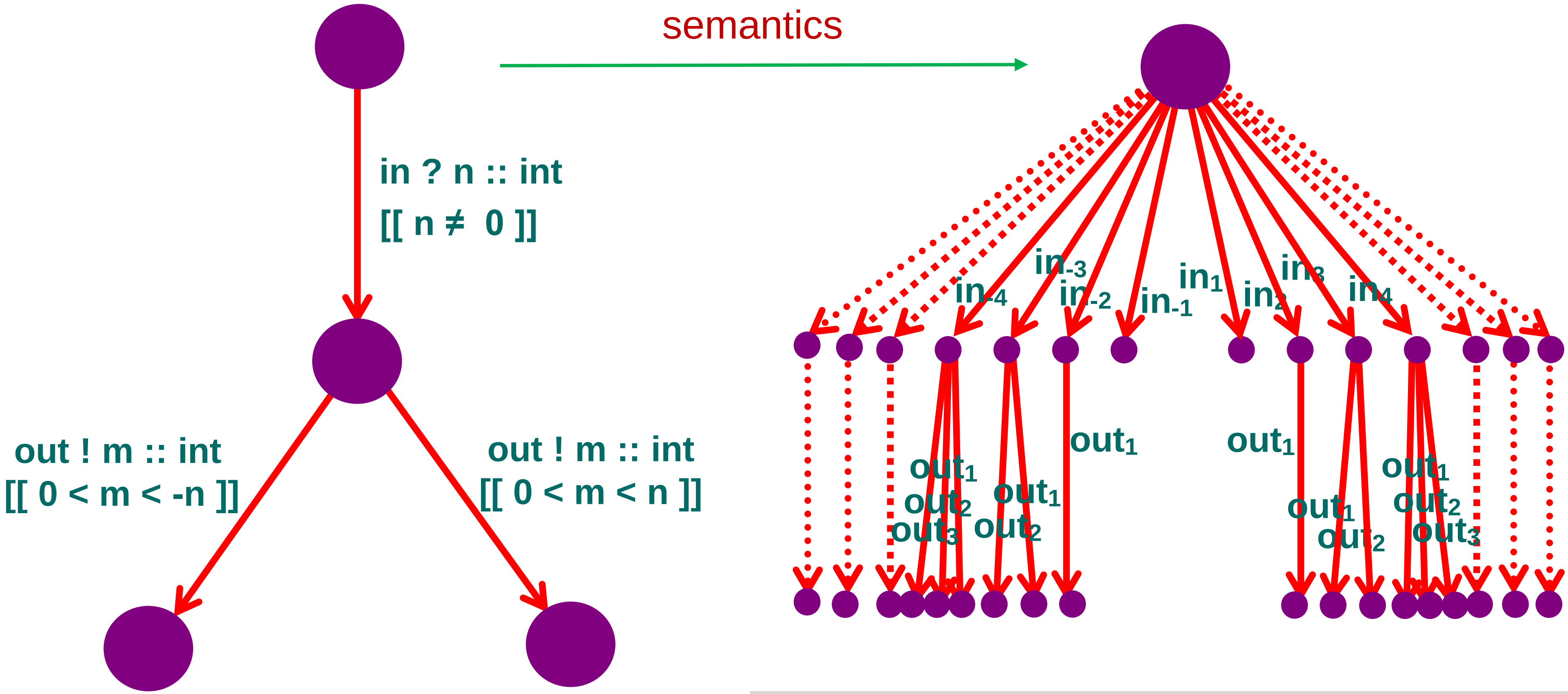

```
STAUTDEF P [..] (..)
 ::=
   STATE  p
   INIT   p
   TRANS  p -> a -> p
ENDDEF
```

LTS with Data : Symbolic Transition Systems

STS:
model
with data



STS : Symbolic Transition Systems



Disadvantages unfolded representation:

- infinity
- loss of information
  (e.g. for test selection)

STS : Symbolic Transition Systems