Modélisation et analyse des systèmes hétérogènes (DIS-MASTIC-05)

Bases

J. Christian Attiogbé

Février 2023 - Collège doctoral Pays de la Loire

Nantes Université

Plan de la suite (adaptation selon contraintes)

1. Introduction
2. Abstractions - modélisations
3. Heterogeneous Model Composition: trends

Présentation du cours

- Hétérogénéité des systèmes : la question, des éléments de caractérisation
- Abstractions nécessaires et leur formalisation avec des objets manipulables
- Ouverture sur l'interdisciplinarité, pendant les études/constructions/travaux

Chapitre 3 : **Composition de modèles hétérogènes**

☞ Composer les divers modèles des composants du système

Analysis of systems

Systems are analysed with respect to required properties.

$$M \vDash P$$

The system is rigorously described with a model.

$$M$$

These properties should be rigorously(formally) described:

$$P$$

Modeling of components or systems ✓
What about the description of properties?

General properties required for systems

We often need to prove **Safety** and **Liveness** properties for systems.
(+ reachability, invariant properties, fairness...)
Rigorous proof, requires a semantics of the system model !

Safety (informally)

The system behaves well (it has only good behaviors); a safety property is a property which specifies that some bad behavior will never occur (these are bad behaviors, to avoid).

Example

- a program always terminates;
- no errors in a program/software/system.

General properties required for systems

Proof of Satefy properties

Formally $M \vDash P$

- With a trace semantics and model-checker: all states of all traces satisfy P, or by counter-example.
- With a theorem prover: P is a theorem under assumptions in M

General properties required for systems (continued)

Liveness (informal)

The system will have a good behavior (it will do something good);
a liveness property specifies that a good property will eventually occur.

Example

- a system will respond to a request
- a system will terminate after all execution (trace)

General properties required for systems

$M \vdash P$

Proof of Liveness properties

P expressed with Modal logics (LTL, CTL, ...)
always globally (a property always holds on a path) : $\Box P$
Finaly (eventualy a property holds): $\Diamond P$
next P,
P **until** Q,
...

$$\Box(badge \Rightarrow \Diamond enter)$$

LTL: Linear Temporal Logic or linear-time temporal logic - Amir Pnueli
CTL: Computation tree logic (braching time Logic) - Edmund Clark
CTL*: CTL with quantifiers; LTL and CTL can be described in CTL*

Pioneers

E. Clarke, A. Emerson, J. Sifakis



Model checking as highly effective verification technology
Turing Award 2007

Specific properties of systems

Apart from **safety and liveness** properties,
a given system may have specific properties like:

- reachability (some thing is possible),
- invariant properties (some properties are always true),
- fairness (may happen if it may happen infinitly often);

Some other properties are qualified as **non-functional**.

A glimpse of LTL Semantics

$$M \vDash \phi$$

LTL formulas are interpreted on (maximal) traces t

$$t = s_0.s_1.s_2 \cdots s_i. \cdots$$

The set of atomic properties holds in all state s_i

$$s \vDash \phi \quad \Leftrightarrow \quad s(i) \vDash \phi$$

- A property holds (is true) for a trace t if it is true at the beginning.
- A property holds for a system if it is true for all its (maximal) trace

☞ Trace semantics (later in the chapter)

Specific properties of systems

Example of non-functional properties

- Quality of services
- Time-constraints properties
- The manufacturing system produces 3 parts per day
- ...

How one can establish that a given task in a heterogeneous system is performed without errors;
knowing that the task may depend on several components.

$$C(M_1, M_2, \cdots, M_n) \vdash \Phi \quad ???$$

SafeP: One cannot enter a protected room without a valid acces card

Heterogeneity and Semantics

Given the **diversity of aspects** often present in a heterogeneous system, its components are varied, as they cover different concerns and are even/often **developed with different formalisms+semantics** and tools.

The semantics behind models of components are varied; this **semantics heterogeneity** hampers analysis of heterogeneous systems.

Examples:

natural systems, **industrial** control systems, **CPS** systems, and more and more **IoT-based systems** that inforce the interconnexion of numerous and various components and systems.

Features of Heterogeneity of components

- Semantics with continuous time
- Semantics with discrete time
- Hybrid (discrete and continuous)
- Trace semantics of behavioral semantics
- ...

Features of Semantics

Some Means to define semantics (of programs/systems):

Operational semantics (SOS) (with semantics rules)
SOS leads to working on states and their relations (=traces) !

Denotational semantics (with semantics functions)
It focuses on mathematical objets (=denotations) that represent programs/systems.
Does not consider the states. Abstract. Compositional.

Axiomatic Semantics (Hoare-method like)

Trace semantics

Trace semantics for labeled transition systems. Execution trace.

Consider (programs/systems), through their states.
Their running/execution is denoted by **transitions from state to state** (depending on actions/statement/instruction).

A **finite trace** is a finite sequence of states $s_1, \cdots s_n$ observed during an execution ; often noted $\langle s_0, \cdots, s_n \rangle$

A **trace can be infinite** as $\langle s_0, \cdots \rangle$

Given a TS or an LTS, a finite trace is denoted S^*, while an infinite trace is denoted s^ω

Trace semantics

Operations defined on traces :

- concatenation,
- comparison on prefix (between finite traces)
- ... between inifinite traces,
- ... between finite and infinite traces,

For a system system defined by a transition system $\langle S, \rightarrow \rangle$ (where L hidden), the trace semantics of Sys written $[\![Sys]\!]^*$ is
the set of finite traces of Sys in S^*

reasoning with infinite traces, infinite+finite traces ?!

Denotational semantics

Communication-based semantics /communication traces

Communications may be synchronous or asynchronous

Synchronous: handshake between emitter and receiver ;
channel of size 1 ;

Asynchronous: channels are queues,
emission is always possible but receptions depend on channels state (empty queue or not)

Partial order semantics

Compositional Semantics

A semantics $[\![.]\!]$ is compositional if the semantics of a system S made of components C_i is obtained "in terms" of the semantics of its components $[\![C_i]\!]$

$$\frac{[\![C_1]\!], \cdots, [\![C_2]\!], \cdots, [\![C_n]\!] \quad S = f(C_1, \cdots, C_n)}{[\![S]\!] = g([\![C_1]\!], \cdots, [\![C_n]\!])}$$

Semantics and Formal Analysis

Give the semantics of components, the analysis concern can be dealt with

$$\frac{C_1 \vDash \phi_1 \quad \cdots \quad C_i \vDash \phi_i \quad \cdots \quad C_n \vDash \phi_n \qquad S = C(C_1, \cdots, C_n)}{S \vDash h(\phi_1 \ \cdots \phi_n)}$$

Heterogeneous models composition

- Synchronous product of LTS
- GALS Globally Asynchronous Locally Synchronous
- Composing models of computation (Edward Lee)
- (Formal) System Engineering approach (Event-B like)
- Multifacets composition and analysis (C. Attiogbé)
- ...

Few concepts : environment - perception - interaction

- Components: what and how to sense?

  events or signals from environment
- Vehicle of signals / events?

  channels, registers
- Interactions?

  send signal - send message
  wait for a signal - wait for a message
  write/read signal-message
  protocol, ...
- Tags, clocks?

  hierarchy, synchronisation, chroreography, ...

Former trends: embedding techniques

Shallow Embedding

Structural translation of components into other formalisms

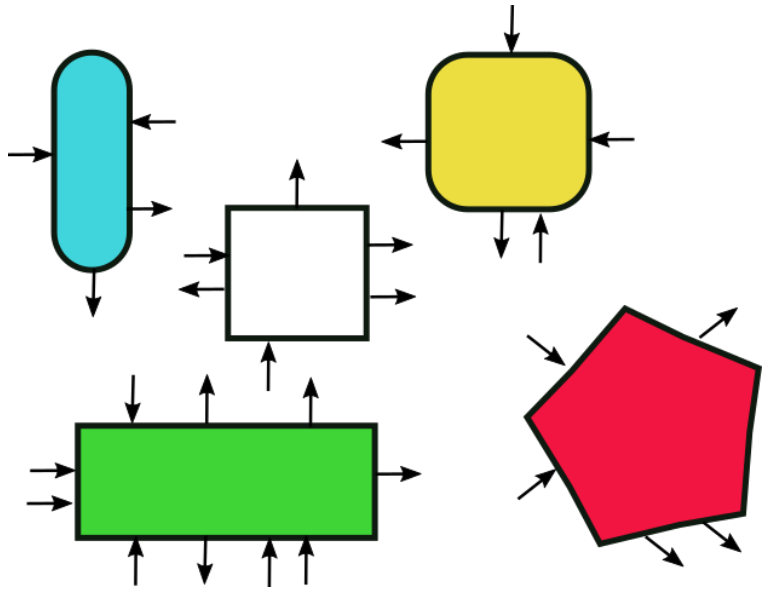
Semantic/Deep Embedding

Translation of the semantics of components into a common semantic domains

Not satisfactory!

New trends

A system as a (weak) composition of different components



But, interaction means should be normalised/standardized

Toward a Heterogeneous Statecharts?

- Toward a Heterogeneous Statecharts?: well-researched and equipped with tools
- **Reuse composition** aspects
- **hide basic behaviour** of components
- promote **only interaction mechanisms**

Statecharts : introduction

- Introduit par Prof. David Harel, 1987
- Apporte des solutions aux limitations des machines à états (FSM classiques) :
  - complexité élevée lorsque le nombre d'états est très grand
  - manque de support pour la concurrence (évolutions simultanées de plusieurs FSM)

Statecharts : automates hiérarchiques

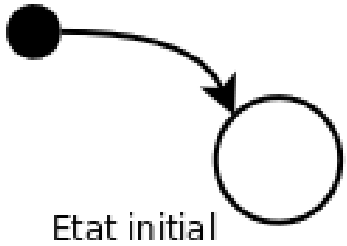
Statechart : extension des automates de Mealy

- les transitions :

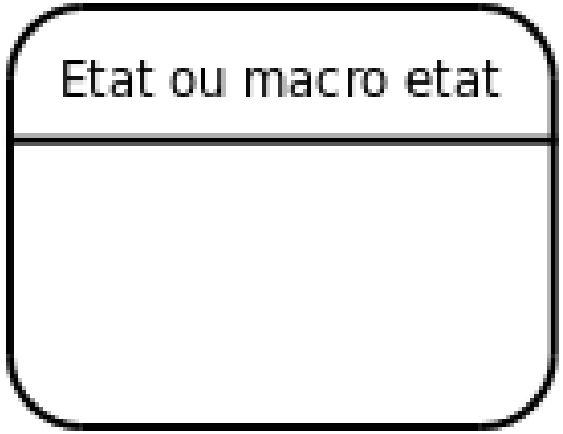
- Macro-états (AND-State et OR-State)
- Concurrence + synchronisation
- Hiérarchisation

Notation des statecharts

- Etat initial et état final :

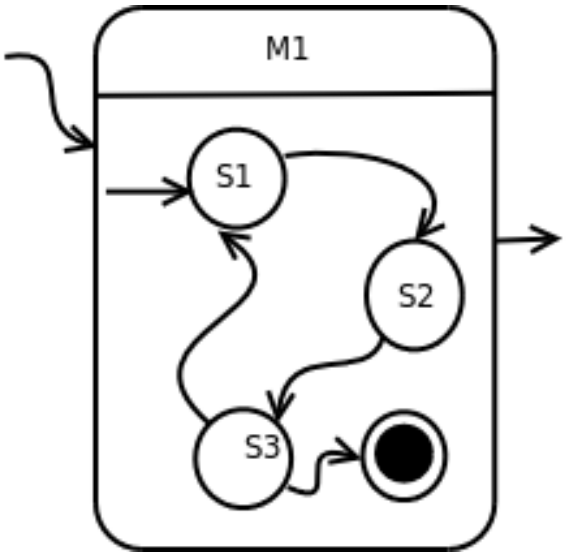


- Macro-état (contient un automate) : c'est la hiérarchie



Notation des statecharts

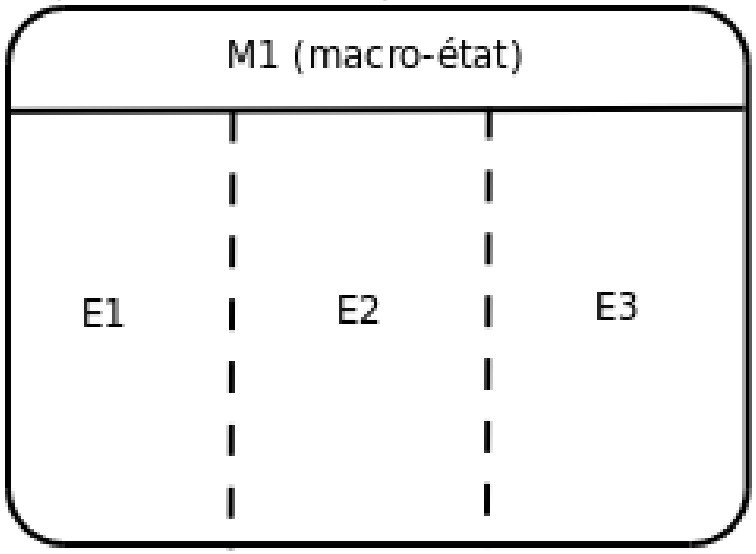
- Etats (OR-States):



S1, S2, S3 sont des états-OU
M1 est exclusivement dans un état des états S1, S2, S3

Notation des statecharts

- Etats concurrents (AND-States):



E1, E2, E3 sont des macro-états concurrents (des automates qui se déroulent simultanément)
M1 est le produit des E1, E2, E3

Exemple de statechart

Modélisation de l'évolution d'un employé dans une entreprise

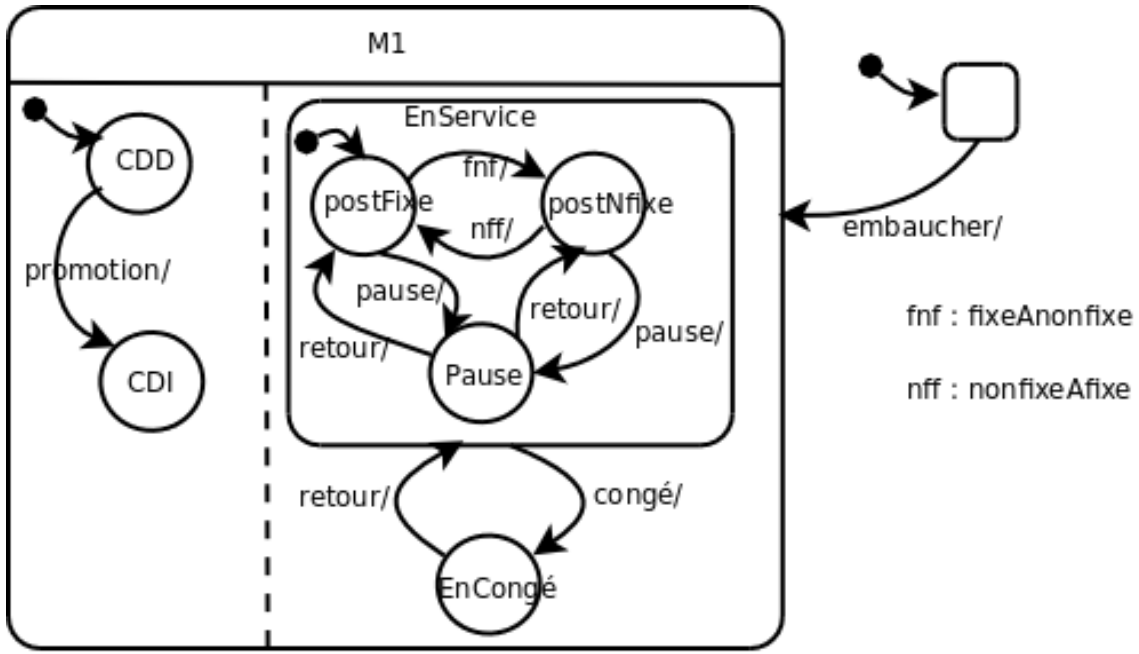


Figure: Evolution d'un employé

Sémantique

Transitions

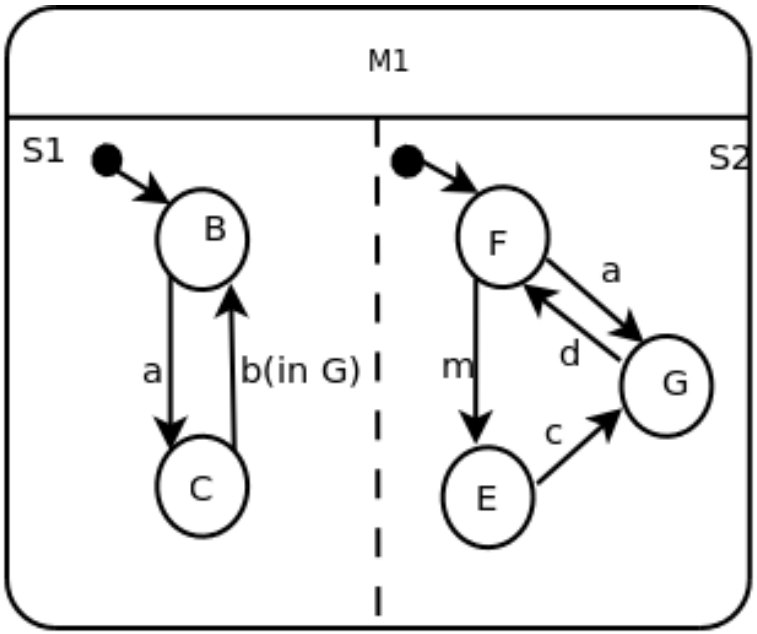
- D'un état vers un macro-état : flot de déroulement de l'état (initial) vers un macro-état.
  Si le macro-état est composé de sous-états concurrents (AND-State) alors le déroulement se poursuit à partir des états initiaux de chacun des sous-états concurrents.
- A l'intérieur d'un macro-état : déroulement en fonction des étiquettes des transitions (jusqu'à l'état final).
  On peut aussi sortir d'un état interne au macro-état vers un état externe.
- D'un état composite vers un état : déroulement immédiat à partir de n'importe quel sous-état.

Sémantique (suite)

Transitions

- Lorsque tous les sous-états concurrents sont arrivés au final : transition immédiate vers l'état indiqué (transition prévue).
- Les transitions avec le même événement (étiquette) dans un macro-état concurrent, se déroulent simultanément

Exemple et sémantique



S1 et S2 sont parallèles/concurrents

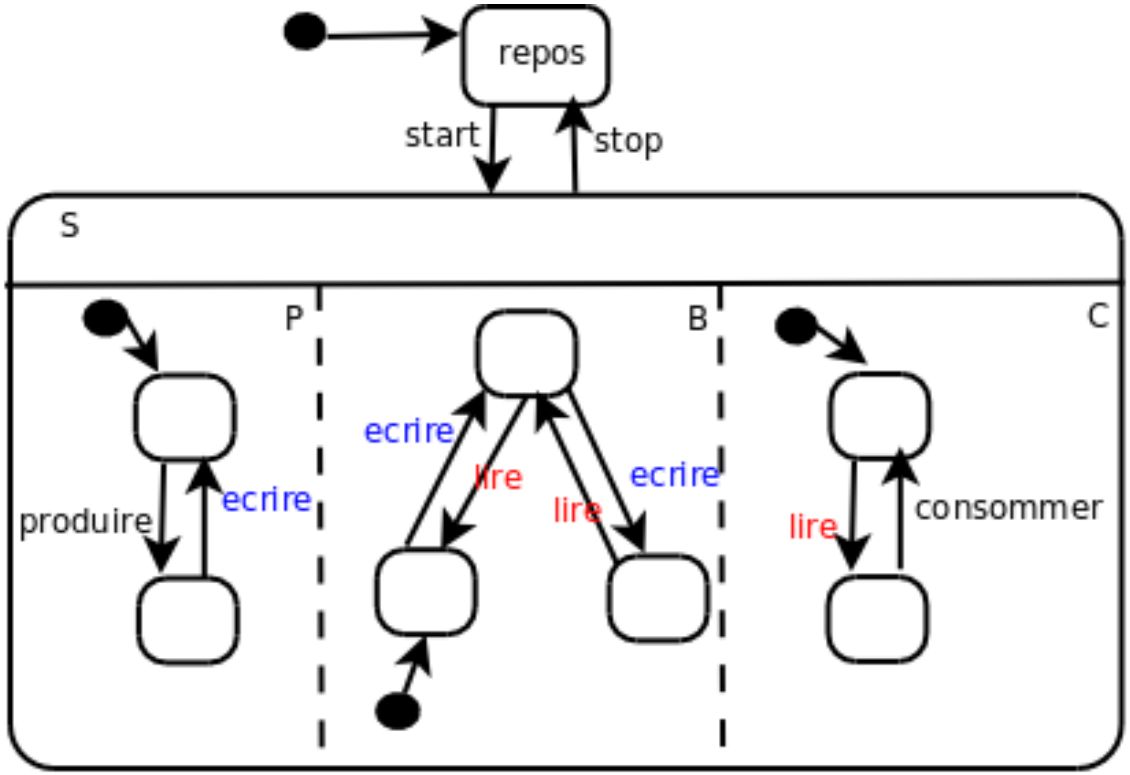
Le système représenté par le macro-état peut être dans chacun des états de S1 et S2 (selon le déroulement à partir des états initiaux, puis des transitions).

Avec a, les systèmes passent de (B, F) vers (C,G) ;

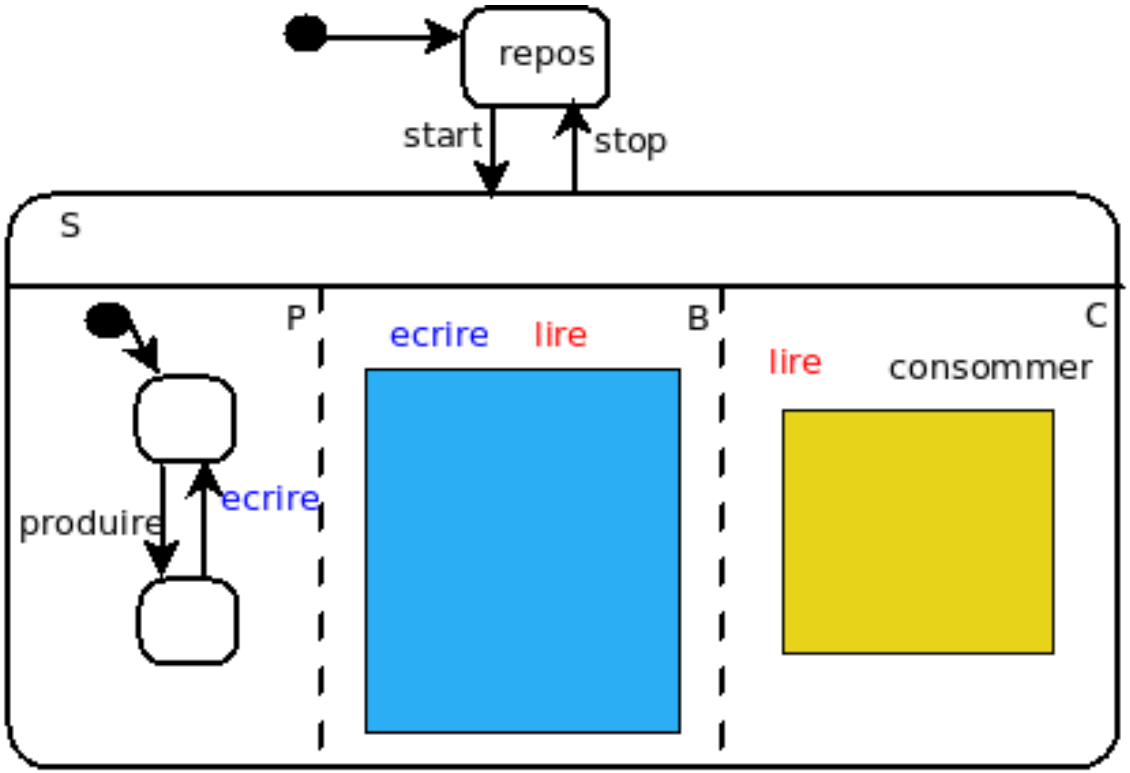
Exemple et sémantique

Exercice : Soit un processus (application) qui produit un message puis l'écrit dans un Buffer.
Un processus Consommateur lit un message en le retirant du Buffer puis le consomme.
Le buffer a une capacité de 2 places.
On veut modéliser ce système composé des Producteur/Buffer/Consommateur.
Initialement le système est au repos ;
puis on le démarre (start) et on l'arrête (stop).

Exemple et sémantique



Extension proposée (Heterogeneous Statecharts)



Les composants P, B, C ont différents modèles sémantiques, mais interagissent par échanges de messages.

Exemple et sémantique

Exercice : spécification du fonctionnement d'une lampe à intensité progressive.
La lampe dispose d'un bouton poussoir.
Lorsqu'on appuie une fois sur le bouton,
la lampe s'allume avec une intensité de 60W.
Un deuxième appui fait passer l'intensité à 120W.
Un troisième appui éteint la lampe (0W).

- Modélisez le fonctionnement de cette lampe à l'aide d'un statechart de Harel.
- Décomposez le système en deux parties : une lampe et un système qui gère son allumage.

Exemple et sémantique

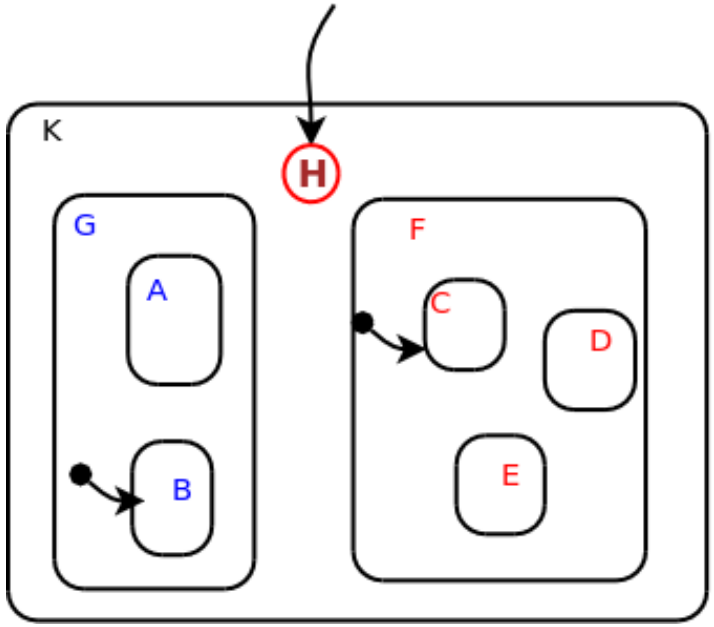
Exercice : Soit un dispositif de contrôle d'éclairage avec deux boutons G et D et deux lampes.

- Initialement les deux lampes sont éteintes.
- Lorsqu'on appuie sur le bouton G, une lampe verte s'allume.
- Lorsqu'on appuie sur le bouton D, une lampe rouge s'allume. 90ms après l'allumage d'une lampe, il y a un blocage du système.
- Lorsqu'une des lampes est allumée,
  - si on appuie sur le bouton de la même couleur de lampe (avant les 90 ms), le système se bloque.
  - si on appuie sur le bouton de la lampe de l'autre couleur (avant les 90 ms), la lampe appropriée s'allume.

Autres caractéristiques : Etat à Historique

On indique l'historique par une transition étiquetée et allant vers un cercle autour d'un H.
Sémantique : selon l'étiquette de la transition, on reprend le déroulement d'un macro-état dans l'état d'où on y est sorti la dernière fois.

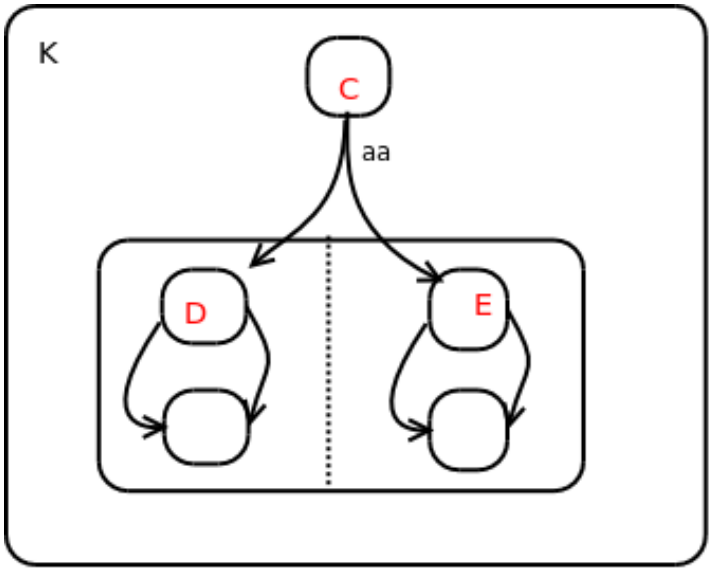
Autres caractéristiques : Etat à Historique



En rentrant dans K, on choisit entre G et F ;
Par défaut (état initial), le système rentre dans A s'il était en A ou B (à la dernière sortie);
Le système rentre en C s'il était en C, D, E la dernière fois.
On peut préciser de rentrer dans l'état d'avant la dernière sortie avec H*

Autres caractéristiques : Hyper-arc

On indique l'hyper-arc par un arc orienté vers plusieurs états
Sémantique : on continue le déroulement dans chacun des états indiqués par l'hyper arc.
Par exemple en entrant dans un macro-état, mais pas dans les sous-états (initiaux) par défaut mais d'autres qui sont indiqués par l'hyper-arc.



On peut aussi avoir un hyper-arc venant de plusieurs états et allant vers un état (par exemple en sortant d'un macro-état).

Eléments de conclusion

- Modélisation des systèmes hétérogènes
- Modélisation hétérogène : activité à priori multiformalisme
- Activité multidisciplinaire
- Complexité pour l'analyse des systèmes
- Décomposition, interaction, normalisation

MERCI de votre attention !