UPPAAL

Ciprian.TEODOROV@ENSTA-Bretagne.fr

Plan

- Introduction
- Automates temporisés
- UPPAAL
  - Spécification des modèles
  - Spécification de propriétés

Le besoin

- HW et SW dans les applications critiques
- Vérification et Validation pour garantir la qualité

- Simulation et test :
  - + utilisé à grande échelle
  - - pas d'exhaustivité -> risque de rater des erreurs
- Méthodes formelles - vérification exhaustive
  - Vérification déductive : chère, lente, difficile à automatiser
  - Model checking : très bons résultats pour des systèmes finis, automatique

Model checking est une technique de vérification formelle très accessible pour les ingénieurs.

Nombreux cas d'utilisation

- Vérification de circuit : **Intel**
- Vérification de driveurs : **Microsoft**
- Vérification de systèmes distribués : **Amazon**

Vérification formelle: Model-Checking

Système

Abstraction

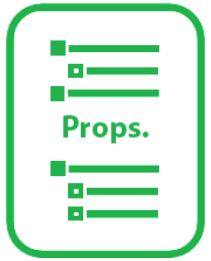
Modèle

satisfait

Spécification

Abstraction

Propriétés



satisfait

Model-checking

ENSTA Bretagne

Model-checking

Idée : **Recherche exhaustive** de contre-exemple

{Modèle, Hypothèses, Lois du domaine} |= Exigences

Modèle du Système étudié (mSE)

Environnement

Propriétés

{ Modèle Système || Environnement} |= Propriétés

Modèle à vérifier

ENSTA Bretagne

Model-checking

Avantages

- intuitive
- générique
- automatisé
- Contre-exemple

Désavantages

- Explosion de l'espace d'états
- Réduction manuelle du modèle
- Utilisation des logique temporelles

Solutions:

Réduction d'ordres partielles,
réduction de symétries,
abstraction et raffinement guidé par les contre-exemples

Plan

- Introduction
- **Automates temporisés**
- UPPAAL
  - Spécification des modèles
  - Spécification de propriétés

Automate temporisé : Syntaxe

$\mathcal{A}$= (Q, ∑, X, G, L, δ, I)

Q – un ensemble fini d'états

∑ – un alphabet

X – un ensemble fini d'horloges

G – un ensemble de contraintes d'horloge

L : Q $\rightarrow \mathcal{P}(G)$ – une fonction d'étiquetage qui associe a chaque état un ensemble de contraintes d'horloge

$\delta : Q \times \Sigma \times \mathcal{P}(G) \times \mathcal{P}(X) \rightarrow Q$ – une fonction de transition

I ⊆Q – un ensemble d'état initiaux

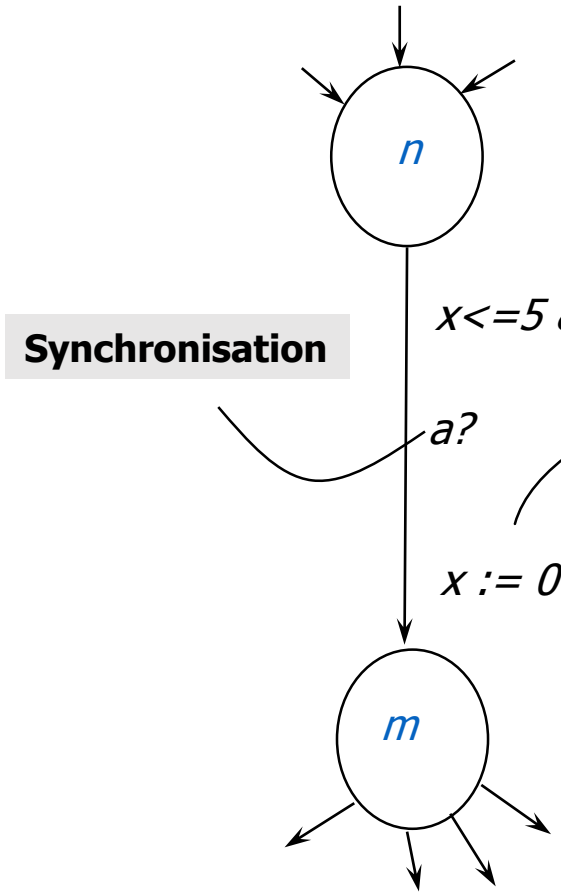
ENSTA Bretagne

Automates temporisé : Sémantique

- Pour un automate temporisé A sa sémantique est définie à travers un système de transitions infini S(A) comme suit :
  - $Q_{S(A)} = \{\langle s, v\rangle \mid s \in QA, v \in \mathbb{R}, v \vdash L(s)\}$
  - $I_{S(A)} \subseteq Q_{S(A)} = \{\langle s, 0\rangle \mid s \in IA, 0 \vdash L(s)\}$
  - $\delta(\langle s, v\rangle) = \begin{cases} \langle s, v + d\rangle, & v \vdash L(s) \ \wedge v + d \vdash L(s) \\ \langle s', v'\rangle, & \delta(s, -, g, r) = s' \wedge v \vdash g \wedge \ v' = v[r/0] \end{cases}$

ENSTA Bretagne

Automates temporisé



Clocks: *x, y*

Garde

Synchronisation

Reset
La seule action possible sur les horloges

State

(*location* , *x*=v , *y*=u) where v,u are in $\mathbb{R}$

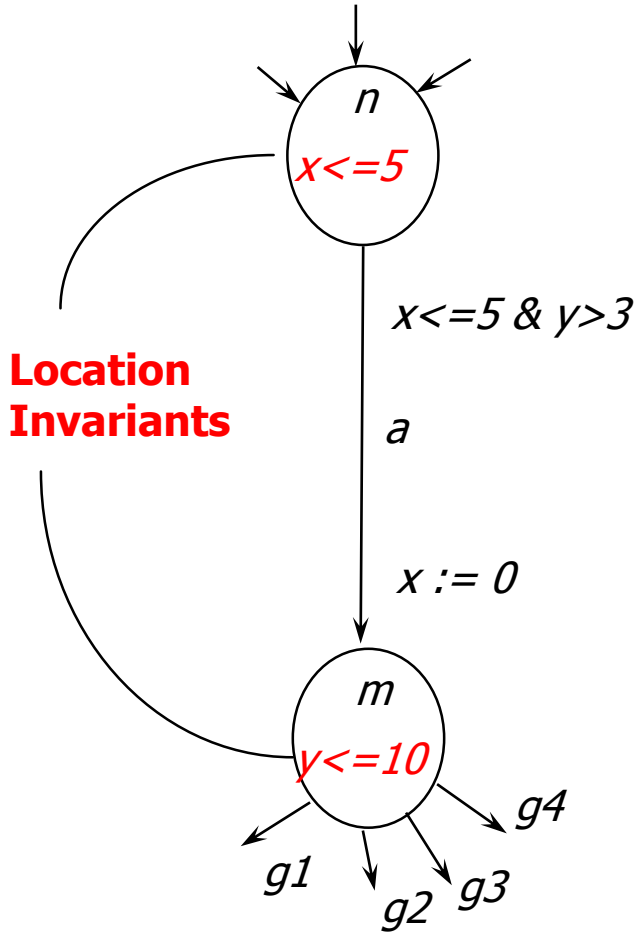
Transitions

(***n*** , *x*=2.4 , *y*=3.1415) $\xrightarrow{a}$ (***m*** , *x*=0 , *y*=3.1415)

(*n* , ***x*=2.4 , *y*=3.1415**) $\xrightarrow{\textbf{wait(1.1)}}$ (*n* , ***x*=3.5 , *y*=4.2415**)

ENSTA Bretagne

Automates temporisé



Invariants

Clocks: *x, y*

Transitions

(*n* , *x*=2.4 , *y*=3.1415) —*wait(3.2)*→ ✗

(*n* , ***x*=2.4 , *y*=3.1415**) —**wait(1.1)**→ (*n* , ***x*=3.5 , *y*=4.2415**)

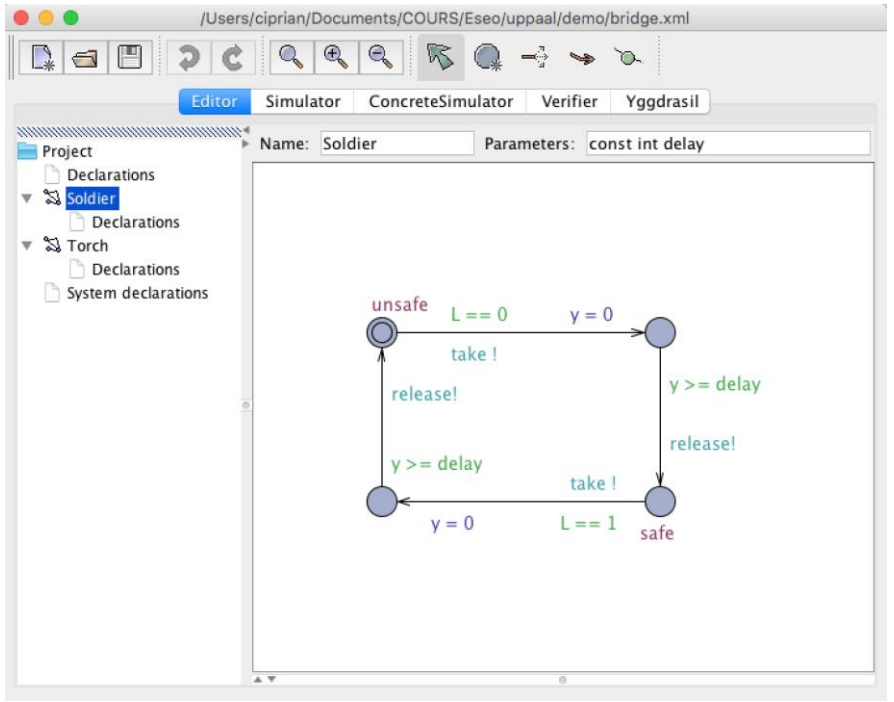
Les invariants forcent le changement d'état !!

Plan

- Introduction
- Automates temporisés
- **UPPAAL**
  - **Spécification des modèles**
  - Spécification de propriétés

UPPAAL

- Outil pour la **validation** et la **vérification** des *systèmes « temporisés »*
- Développé en partenariat entre
  - Université **Upp**sala (Suède)
  - Université **Aal**borg (Danemark)



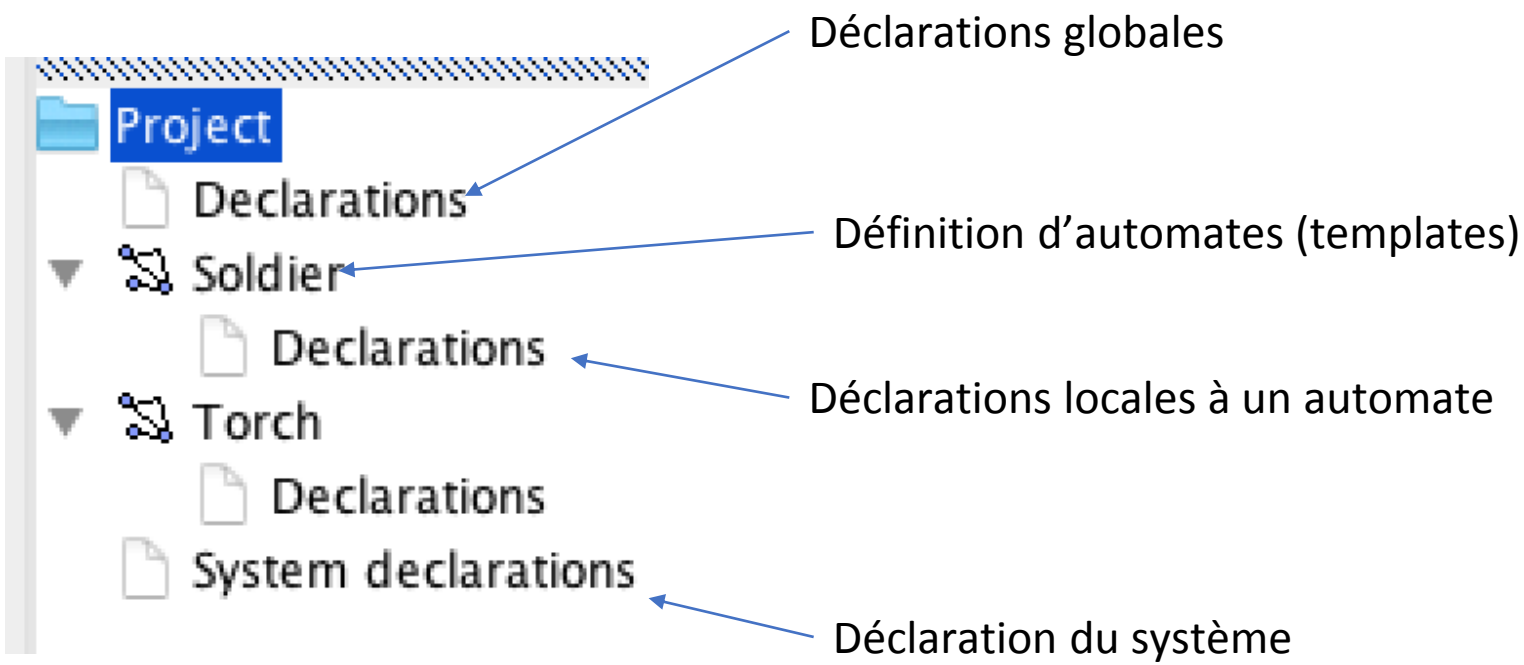
UI ⟷ Serveur

{ Modèle Système || Environnement} **|=** **Propriétés**

Automates temporisés → Uppaal ← Propriétés TCTL

ENSTA Bretagne

Structure d’un projet UPPAAL



Déclarations

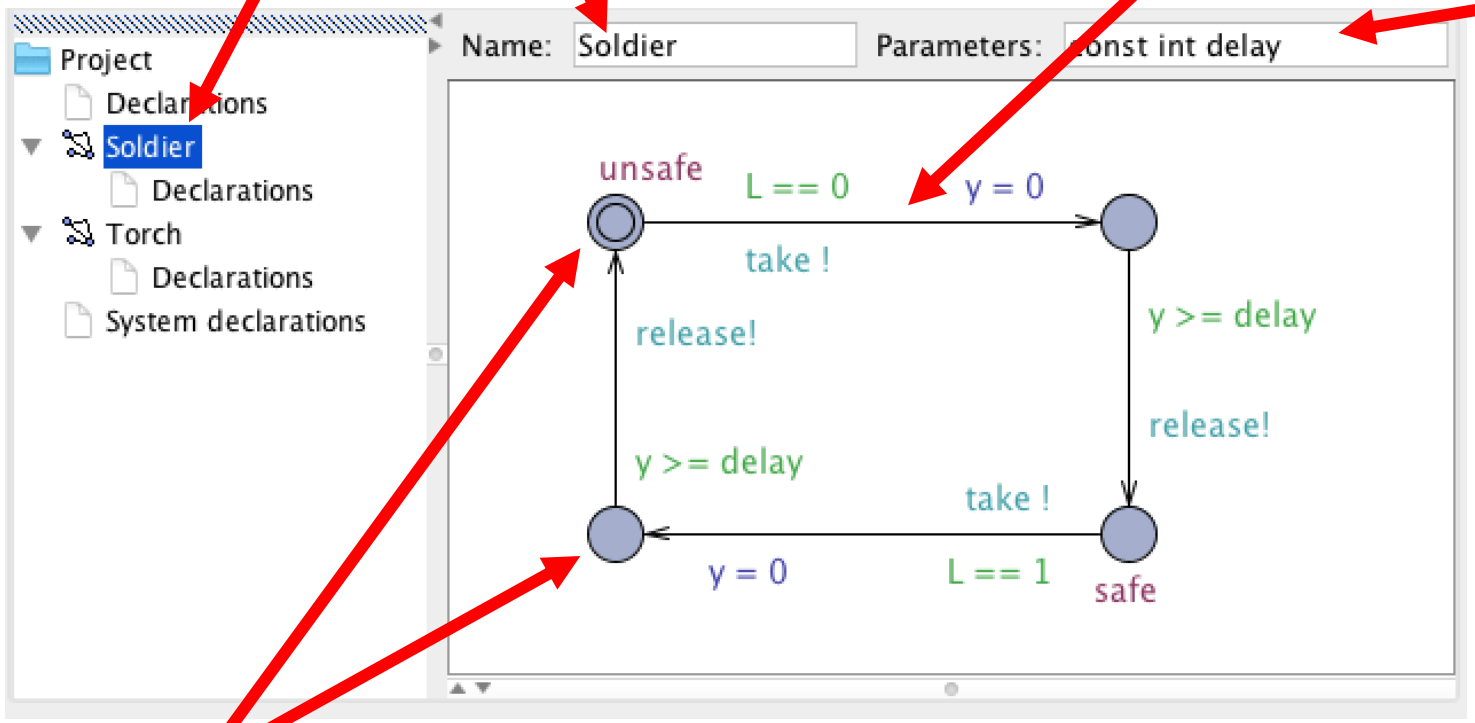
```
int x;                          // déclaration d'une variable

clock time;                     //déclaration d'une horloge

chan take, release;             // déclaration de canal synchrone

broadcast chan annonce;         //déclaration de canal de broadcast
```

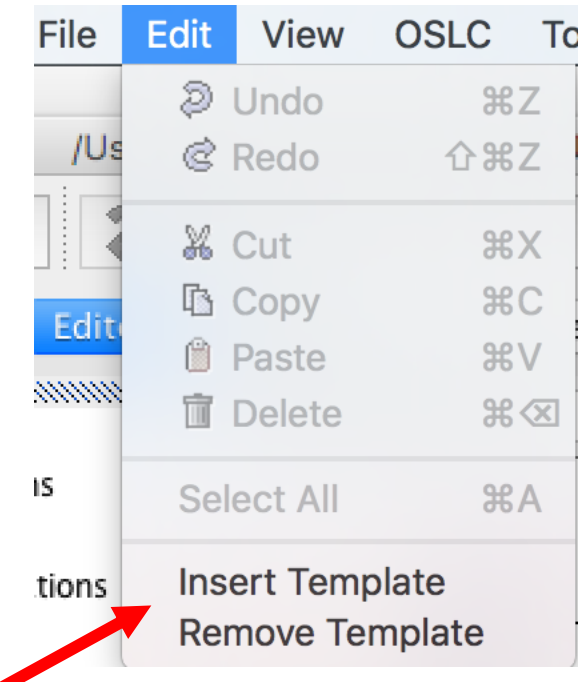
Template UPPAAL – Automate temporisé

Nom de l'automate

« Edge »

Paramètres



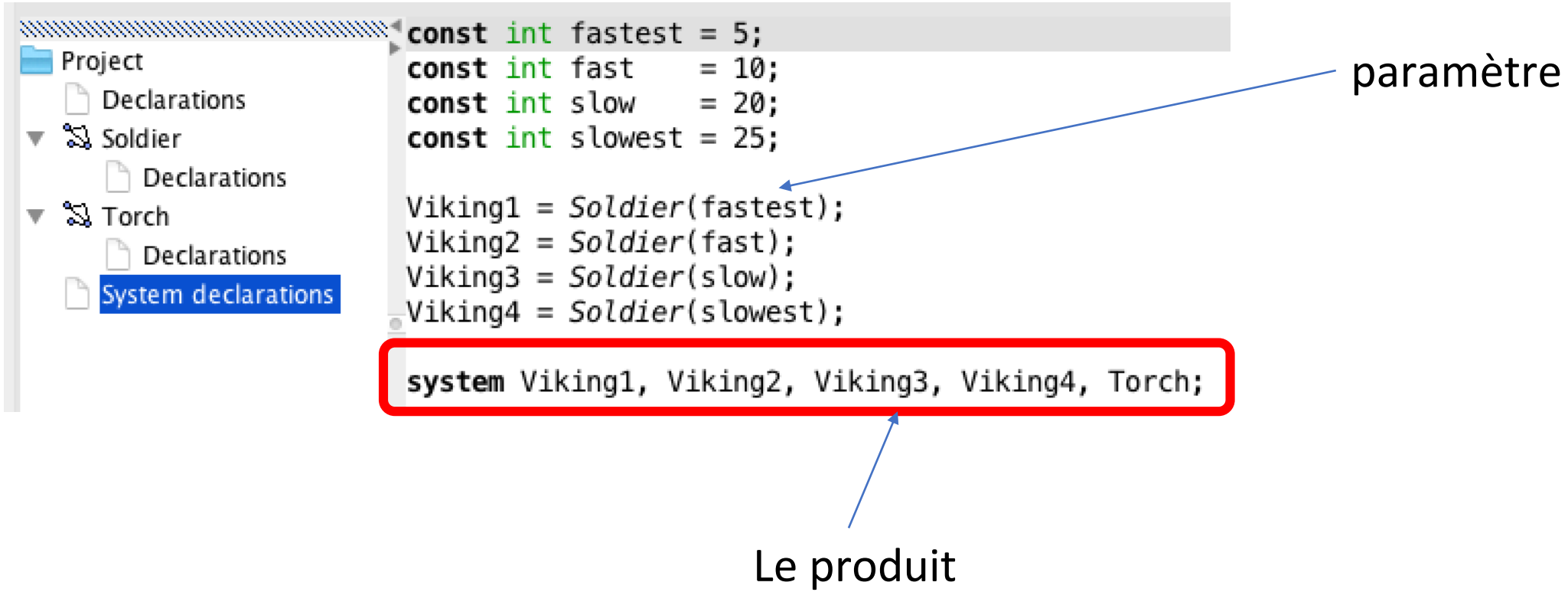


« Location »

Création d'un nouveau automate

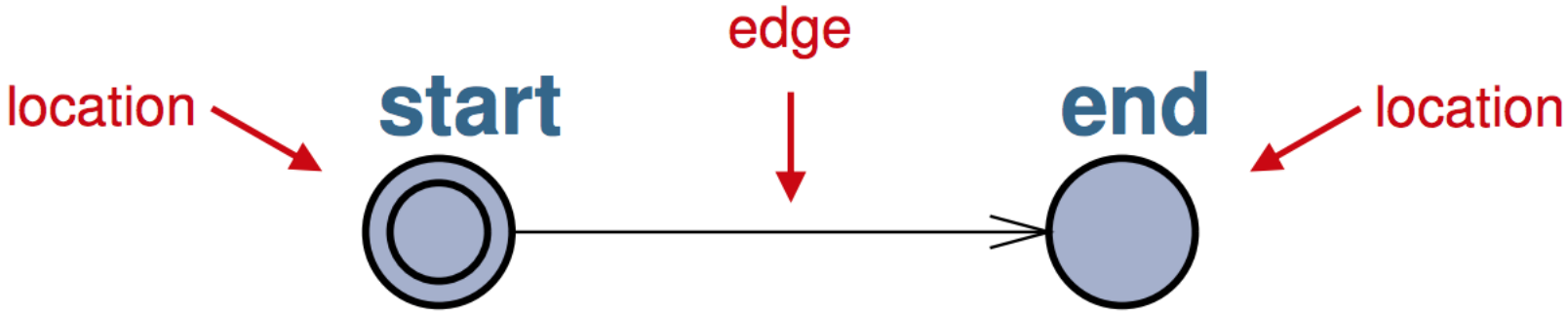
ENSTA Bretagne

Le système est un produit d'automates



ENSTA Bretagne

Les transitions d'un automate UPPAAL



Les transitions d'un automate peuvent être annotées avec :

1. Gardes : conditions booléennes **[« Guard »]**
2. Bloc d'actions : modifications de variables **[« Update »]**
3. Synchronisations : canaux (symboles) de synchronisation **[« Sync »]**
4. Non-déterminisme : sucre syntaxique pour le non-détérminisme **[« Select »]**

Gardes et Actions

Garde :

- Une expression booléenne construite avec les variables et les horloges.
- Une garde vraie indique que la transition est tirable.

Actions :

- Un changement d'une variable du système
- Une remise à zéro d'un horloge

Synchronisation : les canaux synchrones

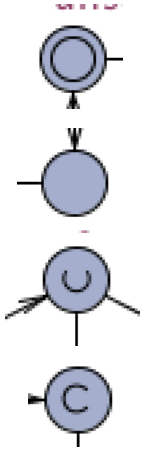
Les canaux synchrones permettent la coordination entre plusieurs automates.
La composition des automates avec des canaux synchrones = le **produit synchronisé**.
En UPPAAL:

- Déclaration d'un canal synchrone :**chan** *connexion*;
- Demande de synchronisation : *connexion*!
- Attente d'une demande de synchronisation : *connexion*?
- Trois types différents :
    - **Synchronisation normale** : la synchronisation peu prendre du temps
    - Synchronisation avec un **canal urgent** : instantanée (le temps n'avance pas)
    - **Annonce globale (broadcast)** : pas d'obligation d'avoir un partenaire en attente

ENSTA Bretagne

Etats en UPPAAL

- Chaque état peut avoir un nom.
- 4 types d'états:
  - **« Initial »** : l'état initial de l'automate
  - **« Normal »** : un état typique d'un automate temporisé
  - **« Urgent »** : un état dans lequel le temps ne peu pas avancer (l'invariant d = 0)
  - **« Committed »** : le temps n'avance pas, et le système doit sortir de l'état au plus vite possible (aucun autre automate ne bouge).



Plan

- Introduction
- Automates temporisés
- **UPPAAL**
  - Spécification des modèles
  - **Spécification de propriétés**

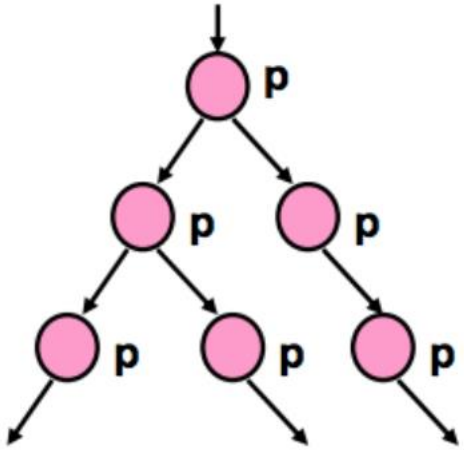
Uppaal : Propositions Atomiques

- Alice.W
- p1.cs and p2.cs
- deadlock
- p1.v < 4

Spécification: sous-ensemble de TCTL

- CTL*
  - E – il existe un chemin
  - A – pour tous les chemins
  - [] – tous les états d'un chemin
  - <> – un état d'un chemin
- UPPAAL:
  - **A[]** p – toujours p
  - **A<>** p – p est inévitable
  - **E[]** p – potentiellement toujours p
  - **E<>** p – p est atteignable
  - p **-->** q – toujours (p implique que q est inévitable)
  - A[] deadlock

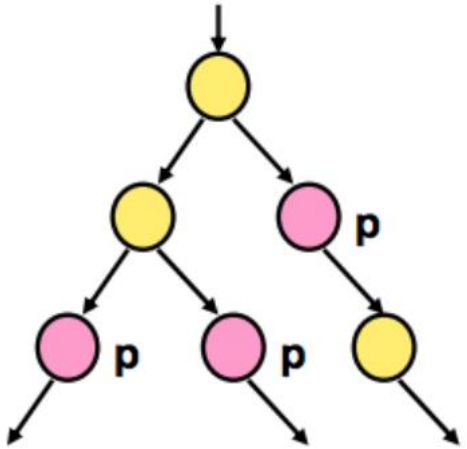
A[] p – toujours p



Sur tous les chemins p *est vrai* dans **tous les états**

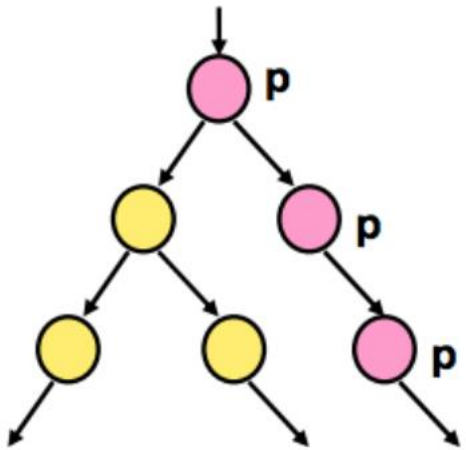
ENSTA Bretagne

A<> p – p est inévitable



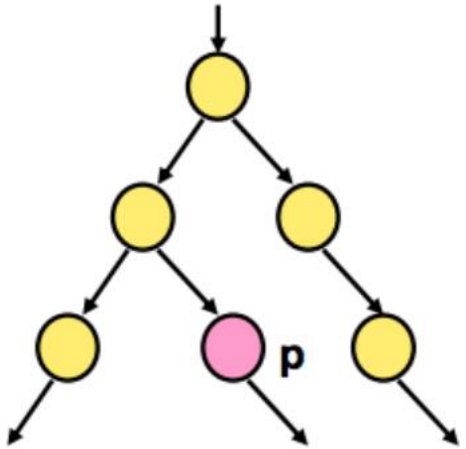
Sur tous les chemins *p est vrai* dans **au moins un état**

E[] p – potentiellement toujours p



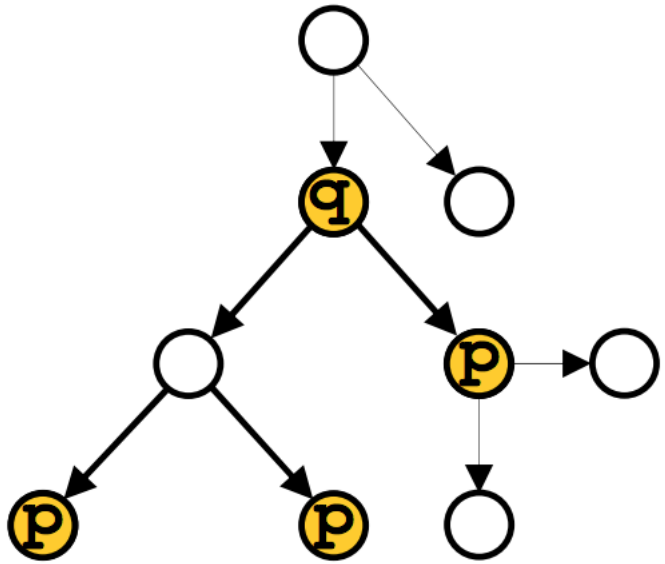
Il existe un chemin sur lequel *p est vrai* dans **tous les états**

E<> p – p est atteignable



Il existe un chemin sur lequel *p est vrai* dans **au moins un état**

q --> p – p réponds à q



Sur tous les chemins *q est vrai* **implique** que p est **inévitable**

ENSTA Bretagne

A vous de jouer

ENSTA Bretagne