Cours Qt (6° partie)
Automate fini ou Machine à états

Thierry Vaira

IUT Arles

tvaira@free.fr ☮ v1

Présentation

- Un **automate fini** ou **machine à états finis** (*finite state machine*) est un modèle mathématique de calcul utilisé dans de nombreux domaines (conception de programmes informatiques, protocoles de communication, contrôle des processus, analyse linguistique, ...).
- Lire : `fr.wikipedia.org/wiki/Automate_fini`
- Un automate fini est susceptible d'être :
  - dans un nombre fini d'états,
  - dans un seul état à la fois.
- L'état où il se trouve est appelé l'« **état courant** ». Le passage d'un état à un autre est dirigé par un **évènement** (ou une condition) appelé une « **transition** ».
- Un automate sera défini par la liste de ses états et par les conditions des transitions.

Exemples

- On rencontre couramment des automates finis dans de nombreux appareils qui réalisent des actions déterminées en fonction des évènements qui se présentent.
- Exemples :
  - un distributeur automatique de boissons qui délivre l'article souhaité quand le montant introduit est approprié,
  - les ascenseurs qui savent combiner les appels successifs pour s'arrêter aux étages intermédiaires,
  - les feux de circulation capables de s'adapter aux voitures en attente,
  - des digicodes qui analysent la bonne suite de chiffres,
  - la gestion de menu des IHM qui s'affichent en fonction des choix de l'utilisateur ...

Terminologie

- Un **état** est la description de la configuration d'un système en attente d'exécuter une transition.
- Une **transition** est un ensemble d'actions à exécuter lorsqu'une condition est remplie ou lorsqu'un évènement est reçu.
- Exemple : une chaîne audio peut être dans l'état « CD » et recevoir l'évènement « suivant ». Elle passe alors à la piste suivante du CD.
- Dans certaines représentations de machines finies, il est possible d'associer des actions à un état :
  - action d'entrée : réalisée lorsque l'on « entre » dans l'état,
  - action de sortie : réalisée lorsque l'on « quitte » l'état.
  - action de transition : réalisée lors d'une transition

Exemple : le portillon d'accès I

- Un exemple très simple d'un mécanisme que l'on peut modéliser par un automate fini est un **portillon d'accès**.

Exemple : le portillon d'accès II

- Un portillon, utilisé dans certains métros ou dans d'autres établissements à accès contrôlés est une barrière avec trois bras rotatifs à hauteur de la taille. Au début, les bras sont verrouillés et bloquent l'entrée, et empêchent les clients de passer. L'introduction d'une pièce de monnaie (ou d'un jeton dans une fente du portillon ou la présentation d'un ticket ou d'une carte) débloque les bras et permet le passage d'un et un seul usager à la fois. Une fois le client entré, les bras sont à nouveaux bloqués jusqu'à ce qu'un nouveau jeton est inséré.
- Un portillon peut être vu comme un automate fini à deux états : **verrouillé** (« *locked* ») et **déverrouillé** (« *unlocked* »).
- Deux "entrées" peuvent modifier l'état : la première si l'on insère un jeton dans la fente (entrée **jeton**) et la deuxième si l'on pousse le bras (entrée **pousser**).

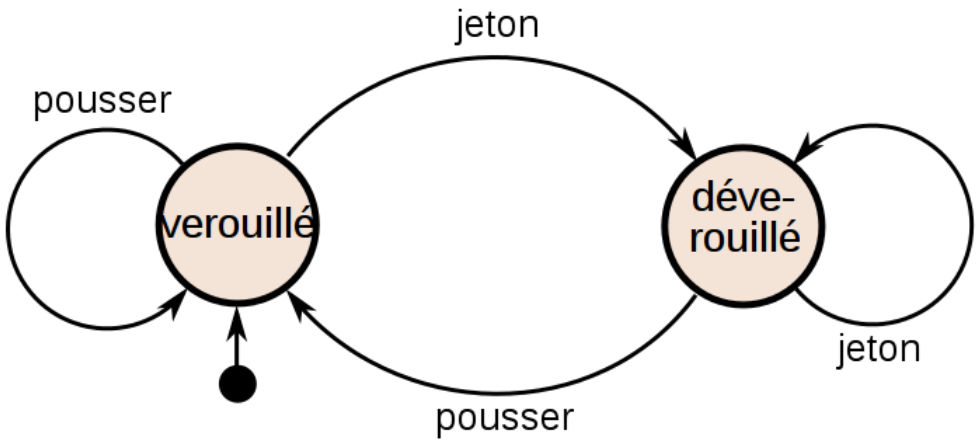
Exemple : le portillon d'accès III

- Dans l'état verrouillé, l'action de pousser n'a aucun effet : quel que soit le nombre de fois que l'on pousse, l'automate reste verrouillé. Si l'on insère un jeton, c'est-à-dire si l'on effectue une "entrée" jeton, on passe de l'état verrouillé à l'état déverrouillé. Dans l'état déverrouillé, ajouter des jetons supplémentaires n'a pas d'effet, et ne change pas l'état. Mais dès qu'un usager tourne le bras du portillon, donc fournit un pousser, la machine retourne à l'état verrouillé.

État courant	**Entrée**	**État suivant**	**Sortie**
verrouillé	jeton	déverrouillé	Déverrouille le portillon
verrouillé	pousser	verrouillé	Rien
déverrouillé	jeton	déverrouillé	Rien
déverrouillé	pousser	verrouillé	Verrouille le portillon

Graphe orienté

- On peut représenter l'automate par un **graphe orienté**.



- Chaque état est représenté par un **sommet** (visualisé par un cercle). Les **arcs** (représentés par des flèches) montrent les transitions d'un état à un autre. Chaque **flèche** porte une entrée qui déclenche la transition. Un **point noir** sert à indiquer que c'est état est l'état initial. Un **point noir dans un cercle** indiquera l'état final. En cas de non changement d'état, on représente un arc circulaire (boucle) qui tourne autour de l'état.

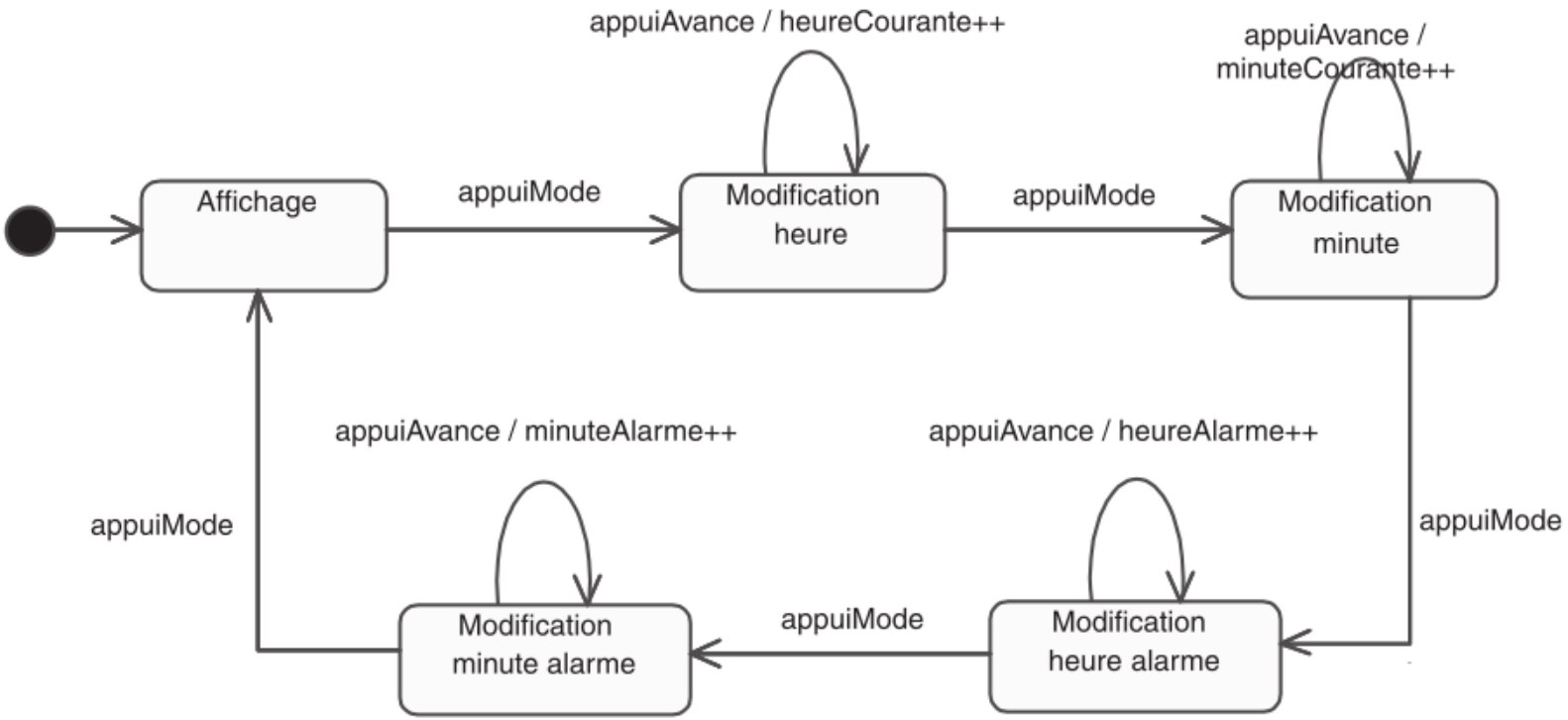
Table états-transitions

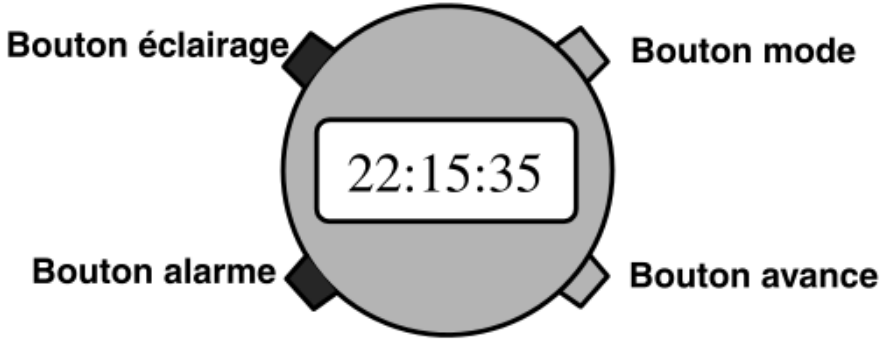
- Plusieurs types de tables de transition d'état sont utilisées. Ces diagrammes sont très populaires en **UML** notamment (cf. diagramme états-transitions).
- La représentation la plus courante est illustrée ci-dessous :

État / **Entrée**	jeton	pousser	bouton init	bouton arrêt
initial	déverrouillé	verrouillé	initial	final
verrouillé	déverrouillé	verrouillé	initial	final
déverrouillé	déverrouillé	verrouillé	initial	final
final	final	final	final	final

- Explication : La combinaison de l'état courant (par exemple « verrouillé ») et d'une entrée (par exemple « jeton ») montre l'état suivant (dans l'exemple « déverrouillé »).

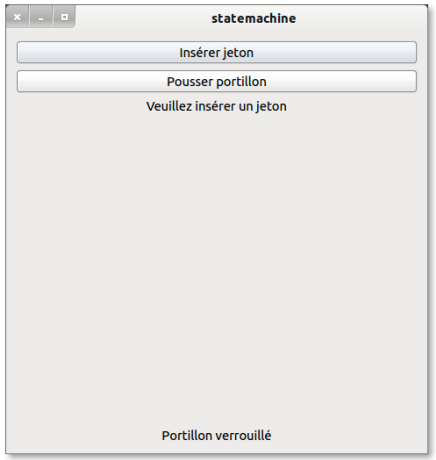
Diagramme d'états UML





Codage du portillon d'accès sous Qt

- Sous Qt, la classe `QStateMachine` fournit une machine à états finis.
- Une machine à états gère un ensemble d'**états** (classes qui héritent de `QAbstractState`) et des **transitions** (descendants de `QAbstractTransition`) entre ces états. Une fois qu'un graphe orienté a été construit, la machine à états pourra l'exécuter. L'algorithme d'exécution de `QStateMachine` est basé sur l'algorithme *State Chart XML* (SCXML).
- On va réaliser une GUI pour tester cet exemple :



Squelette de l'application

mydialog.h

```
class MaFenetre : public QWidget
{
    Q_OBJECT
public:
    MaFenetre(QWidget *p=0);

private:
    QStateMachine  *machine;
    QPushButton    *btInsererJeton;
    QPushButton    *btPousser;
    QLabel         *labelUtilisateur;
    QLabel         *labelPortillon;
};
```

mydialog.cpp

```
MaFenetre::MaFenetre(QWidget *p)
      : QWidget(p)
{
    // créer la GUI
    // créer la machine à état
    // définir les états
    // ajouter les transitions
    // associer des actions
    // démarrer la machine
}
```

Étape n°0 : créer la GUI

```
btInsererJeton = new QPushButton(QString::fromUtf8("Insérer jeton"),
    this);
btPousser = new QPushButton(QString::fromUtf8("Pousser portillon"), this
    );
labelUtilisateur = new QLabel(this);
labelUtilisateur->setAlignment(Qt::AlignCenter);
labelPortillon = new QLabel(this);
labelPortillon->setAlignment(Qt::AlignCenter);
QVBoxLayout *layout = new QVBoxLayout;
layout->addWidget(btInsererJeton);
layout->addWidget(btPousser);
layout->addWidget(labelUtilisateur);
layout->addStretch();
layout->addWidget(labelPortillon);
setLayout(layout);
```

Étape n°1 : créer la machine à états

```
machine = new QStateMachine(this);
```

Étape n°2 : définir les états

On peut maintenant définir les **états**. On utilise la méthode `addState()` pour ajouter un état à la machine à états. Les états sont supprimés avec `removeState()` (la suppression des états pendant la mise en marche de la machine est déconseillée).

```
QState *etatVerrouille = new QState();
etatVerrouille->assignProperty(labelPortillon, "text", QString::fromUtf8
    ("Portillon verrouillé"));
QState *etatDeverrouille = new QState();
etatDeverrouille->assignProperty(labelPortillon, "text", QString::
    fromUtf8("Portillon déverrouillé"));

machine->addState(etatVerrouille);
machine->addState(etatDeverrouille);
```

Remarque : Les états seront simplement visualisés par un `text` dans un `QLabel`.

Étape n°3 : ajouter les transitions

```
etatVerrouille->addTransition(btInsererJeton, SIGNAL(clicked()),
    etatDeverrouille);

etatDeverrouille->addTransition(btPousser, SIGNAL(clicked()),
    etatVerrouille);
```

Étape n°4 : associer des actions

Il est possible d'associer des **actions** à un état :

- action d'entrée : réalisée lorsque l'on « entre » dans l'état (signal `entered()`)
- action de sortie : réalisée lorsque l'on « quitte » l'état (signal `exited()`)

```
connect(etatVerrouille, SIGNAL(entered()), this, SLOT(
    afficherInviteInsererJeton()));

connect(etatVerrouille, SIGNAL(exited()), this, SLOT(
    afficherInvitePassage()));
```

Étape n°4 : démarrer la machine à états

Avant de démarrer la machine, l'état initial doit être réglé avec `setInitialState()`. L'état initial est l'état d'entrée de la machine au démarrage. On peut ensuite démarrer la machine à états avec `start()`. Le signal `started()` est émis lorsque on entre dans l'état initial.

```
machine->setInitialState(etatVerrouille);
machine->start();
```

Remarque : La machine à états émet le signal `finished()` lorsqu'on entre dans l'état final. On peut également arrêter la machine à états avec `stop()` et elle émet alorts le signal `stopped()` dans ce cas.