TD7 : Automates

Stéphane Devismes

Université Grenoble Alpes

24 mars 2016

Plan

Plan

Automate pour modéliser un système

Automate : **formalisme** (ou outil mathématique) qui permet de **modéliser** le comportement de certaines classes de systèmes physiques, en particulier les systèmes qui **interagissent avec leur environnement**.

Automate pour modéliser un système

Automate : **formalisme** (ou outil mathématique) qui permet de **modéliser** le comportement de certaines classes de systèmes physiques, en particulier les systèmes qui **interagissent avec leur environnement**.

Exemples :

- distributeurs de boissons
- robots autonomes
- pilote automatique d'avion (en fait tout système de contrôle-commande)
- digicodes
- programmes en cours d'exécution
- ...

Modélisation

Pour concevoir/comprendre/simuler un système trop complexe on en construit **une représentation simplifiée (un modèle)**, qui ne fait intervenir que ses aspects essentiels.

(approche très classique en physique, en mécanique, *etc.*)

Interactions d'un automate avec l'environnement

Interactions d'un automate avec l'environnement

- Le système reçoit des entrées de la part de l'environnement (**ex :** pièce de monnaie, capteur d'altitude, température, . . .)

Interactions d'un automate avec l'environnement

- Le système reçoit des entrées de la part de l'environnement (**ex :** pièce de monnaie, capteur d'altitude, température, ...)
- Il émet des sorties vers son environnement (**ex :** servir une boisson, action sur un moteur, ouverture d'une vanne, résultat d'un calcul)

Interactions d'un automate avec l'environnement

- Le système reçoit des entrées de la part de l'environnement (**ex :** pièce de monnaie, capteur d'altitude, température, ...)
- Il émet des sorties vers son environnement (**ex :** servir une boisson, action sur un moteur, ouverture d'une vanne, résultat d'un calcul)

Comportement du système :

- quelles sont les entrées que le système **attend/accepte** a un certain moment ?
- quelles sont les sorties qu'il peut alors émettre ?

Interactions d'un automate avec l'environnement

- Le système reçoit des entrées de la part de l'environnement (**ex :** pièce de monnaie, capteur d'altitude, température, ...)
- Il émet des sorties vers son environnement (**ex :** servir une boisson, action sur un moteur, ouverture d'une vanne, résultat d'un calcul)

Comportement du système :

- quelles sont les entrées que le système **attend/accepte** a un certain moment ?
- quelles sont les sorties qu'il peut alors émettre ?

Cette notion de moments est codée avec des états : l'état courant encode toute ou partie des entrées reçues par le système (son histoire) depuis son initialisation.

Plan

Définition (formelle) d'un automate (de Mealy)

Définition (formelle) d'un automate (de Mealy)

- un **ensemble fini d'états**, Q, avec un **état initial** q_0

Définition (formelle) d'un automate (de Mealy)

- un **ensemble fini d'états**, Q, avec un **état initial** q_0
- un **ensemble** (éventuellement vide) **d'états finaux** F

Définition (formelle) d'un automate (de Mealy)

- un **ensemble fini d'états**, Q, avec un **état initial** q_0
- un **ensemble** (éventuellement vide) **d'états finaux** F
- un **ensemble** (ou vocabulaire) **d'entrées** E

Définition (formelle) d'un automate (de Mealy)

- un **ensemble fini d'états**, Q, avec un **état initial** q_0
- un **ensemble** (éventuellement vide) **d'états finaux** F
- un **ensemble** (ou vocabulaire) **d'entrées** E
- un **ensemble** (ou vocabulaire) **de sortie** S

Définition (formelle) d'un automate (de Mealy)

- un **ensemble fini d'états**, Q, avec un **état initial** q_0
- un **ensemble** (éventuellement vide) **d'états finaux** F
- un **ensemble** (ou vocabulaire) **d'entrées** E
- un **ensemble** (ou vocabulaire) **de sortie** S
- une **fonction de transition** $trans : Q \times E \rightarrow Q$

Définition (formelle) d'un automate (de Mealy)

- un **ensemble fini d'états**, Q, avec un **état initial** q_0
- un **ensemble** (éventuellement vide) **d'états finaux** F
- un **ensemble** (ou vocabulaire) **d'entrées** E
- un **ensemble** (ou vocabulaire) **de sortie** S
- une **fonction de transition** $trans : Q \times E \rightarrow Q$
- une **fonction de sortie** $sort : Q \times E \rightarrow S$

Définition (formelle) d'un automate (de Mealy)

- un **ensemble fini d'états**, Q, avec un **état initial** q_0
- un **ensemble** (éventuellement vide) **d'états finaux** F
- un **ensemble** (ou vocabulaire) **d'entrées** E
- un **ensemble** (ou vocabulaire) **de sortie** S
- une **fonction de transition** $trans : Q \times E \rightarrow Q$
- une **fonction de sortie** $sort : Q \times E \rightarrow S$

Remarque : les fonctions de transition et de sortie sont partielles, c'est-à-dire qu'il est possible que certains éléments de l'ensemble de départ n'aient pas d'image définie par la fonction.

Plan

Distributeur de boissons

Modélisons un distributeur de boissons très simple :

Distributeur de boissons

Modélisons un distributeur de boissons très simple :

- Il attend de recevoir une pièce d'un euro.

Distributeur de boissons

Modélisons un distributeur de boissons très simple :

- Il attend de recevoir une pièce d'un euro.
- Puis, un choix : thé ou café.

Distributeur de boissons

Modélisons un distributeur de boissons très simple :

- Il attend de recevoir une pièce d'un euro.
- Puis, un choix : thé ou café.
- Enfin, il sert la boisson choisie.

Les états

Le choix doit arriver après que le distributeur ait reçu un euro, il faut donc **deux états pour coder ≪ l'histoire du système ≫** :

- $E0$, le distributeur n'a pas reçu l'euro attendu (c'est son **état initial)**,
- $E1$, le distributeur a été crédité d'un euro.

Les états

Le choix doit arriver après que le distributeur ait reçu un euro, il faut donc **deux états pour coder ≪ l'histoire du système ≫** :

- $E0$, le distributeur n'a pas reçu l'euro attendu (c'est son **état initial)**,
- $E1$, le distributeur a été crédité d'un euro.

Ainsi, dans notre exemple, $Q = \{E0, E1\}$ et $q_0 = E0$.

Les états

Le choix doit arriver après que le distributeur ait reçu un euro, il faut donc **deux états pour coder ≪ l'histoire du système ≫** :

- $E0$, le distributeur n'a pas reçu l'euro attendu (c'est son **état initial)**,
- $E1$, le distributeur a été crédité d'un euro.

Ainsi, dans notre exemple, $Q = \{E0, E1\}$ et $q_0 = E0$.

Notre distributeur doit être capable d'avoir un **fonctionnement continu** : il se mettra en attente d'un nouvel euro après avoir servi un thé ou un café.

Donc, dans notre exemple **il n'y a pas d'état final** : $F = \emptyset$.

Les entrées

Notre distributeur doit pouvoir répondre à trois type d'entrées :

Les entrées

Notre distributeur doit pouvoir répondre à trois type d'entrées :

- la **créditation d'un euro** et

Les entrées

Notre distributeur doit pouvoir répondre à trois type d'entrées :

- la **créditation d'un euro** et
- un choix : **thé** ou **café**.

Les entrées

Notre distributeur doit pouvoir répondre à trois type d'entrées :

- la **créditation d'un euro** et
- un choix : **thé** ou **café**.

$$E = \{1\_euro, choisir\_the, choisir\_cafe\}$$

Les sorties

Les sorties émises sont soit servir un café, soit servir un thé :

$$S = \{servir\_the, servir\_cafe\}$$

Fonctions de transition et de sortie

Fonction de transition :

Etat / Entrée	1_*euro*	*choisir_the*	*choisir_cafe*
*E*0	*E*1		
*E*1		*E*0	*E*0

Fonction de sortie :

Etat / Entrée	1_*euro*	*choisir_the*	*choisir_cafe*
*E*0			
*E*1		*servir_the*	*servir_cafe*

Fonctions de transition et de sortie

Fonction de transition :

Etat / Entrée	1_*euro*	*choisir_the*	*choisir_cafe*
*E*0	*E*1		
*E*1		*E*0	*E*0

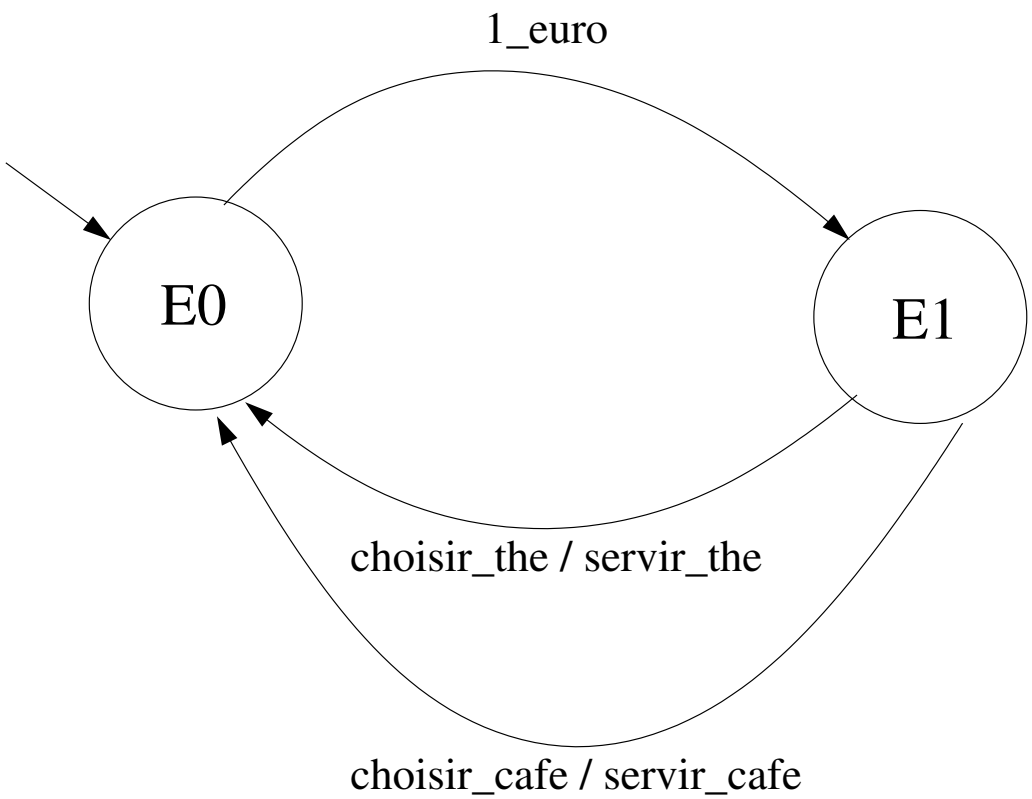
Fonction de sortie :

Etat / Entrée	1_*euro*	*choisir_the*	*choisir_cafe*
*E*0			
*E*1		*servir_the*	*servir_cafe*

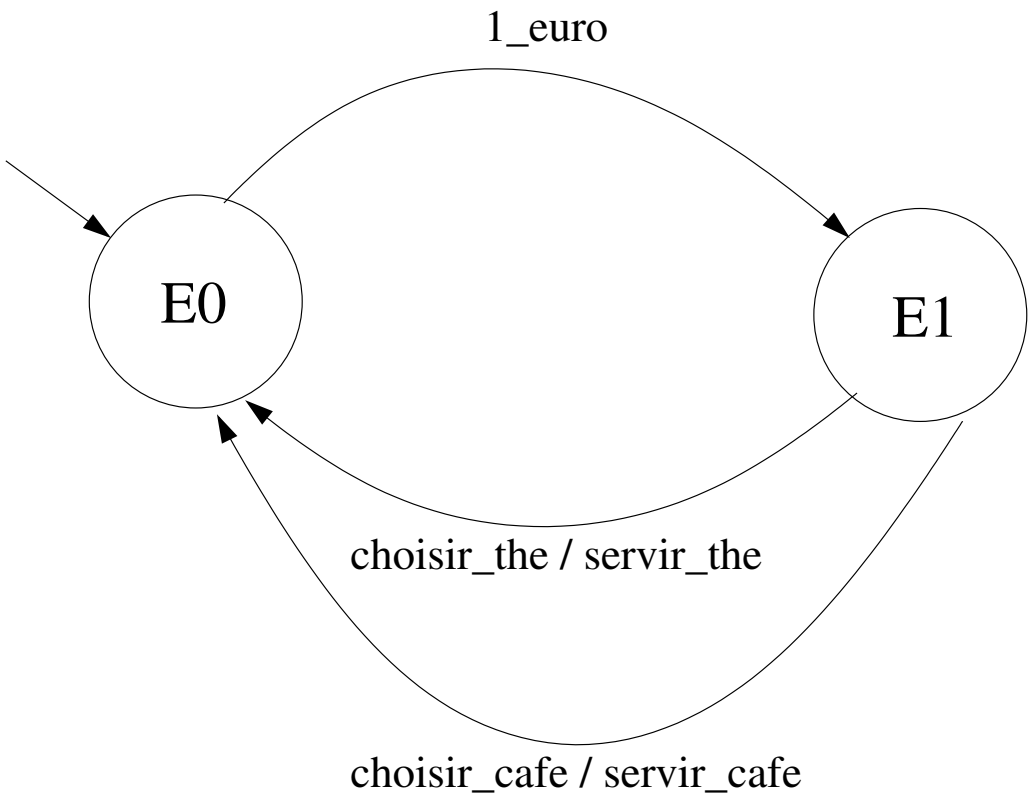
Remarque : les fonctions de transition et de sortie ne sont pas totales : certaines cases ont vides !

Plan

L'exemple du distributeur



L'exemple du distributeur



Remarque : dans cet exemple, il n'y a pas d'état final. Un état final sera représenté par un **double cercle**.

Plan

Fonction totale

Fonction de transition totale : le comportement du système doit toujours être défini quelle que soit l'entrée, c'est-à-dire quelle que soit l'action de l'utilisateur.

Fonction de sortie totale : on peut ajouter une sortie ≪ Rien ≫ éventuellement pour la compléter.

Version complète de l'automate distributeur (1/2)

Version complète de l'automate distributeur (1/2)

- $Q = \{E0, E1\}$ et $q_0 = E0$

Version complète de l'automate distributeur (1/2)

- $Q = \{E0, E1\}$ et $q_0 = E0$
- $E = \{1\_euro, choisir\_the, choisir\_cafe\}$

Version complète de l'automate distributeur (1/2)

- $Q = \{E0, E1\}$ et $q_0 = E0$
- $E = \{1\_euro, choisir\_the, choisir\_cafe\}$
- $S = \{servir\_the, servir\_cafe, rendre\_1\_euro, credit\_nul, credit\_1\_euro\}$

On a ajouté trois sorties :

- *rendre_1_euro* symbolise le fait que la machine rend un euro si on crédite deux fois.
- *credit_nul* symbolise le fait que la machine affiche un message d'erreur si on choisit une boisson sans avoir payé avant.
- *credit_1_euro* symbolise le fait que la machine affiche un message informant que le crédit est d'un euro.

Version complète de l'automate distributeur (2/2)

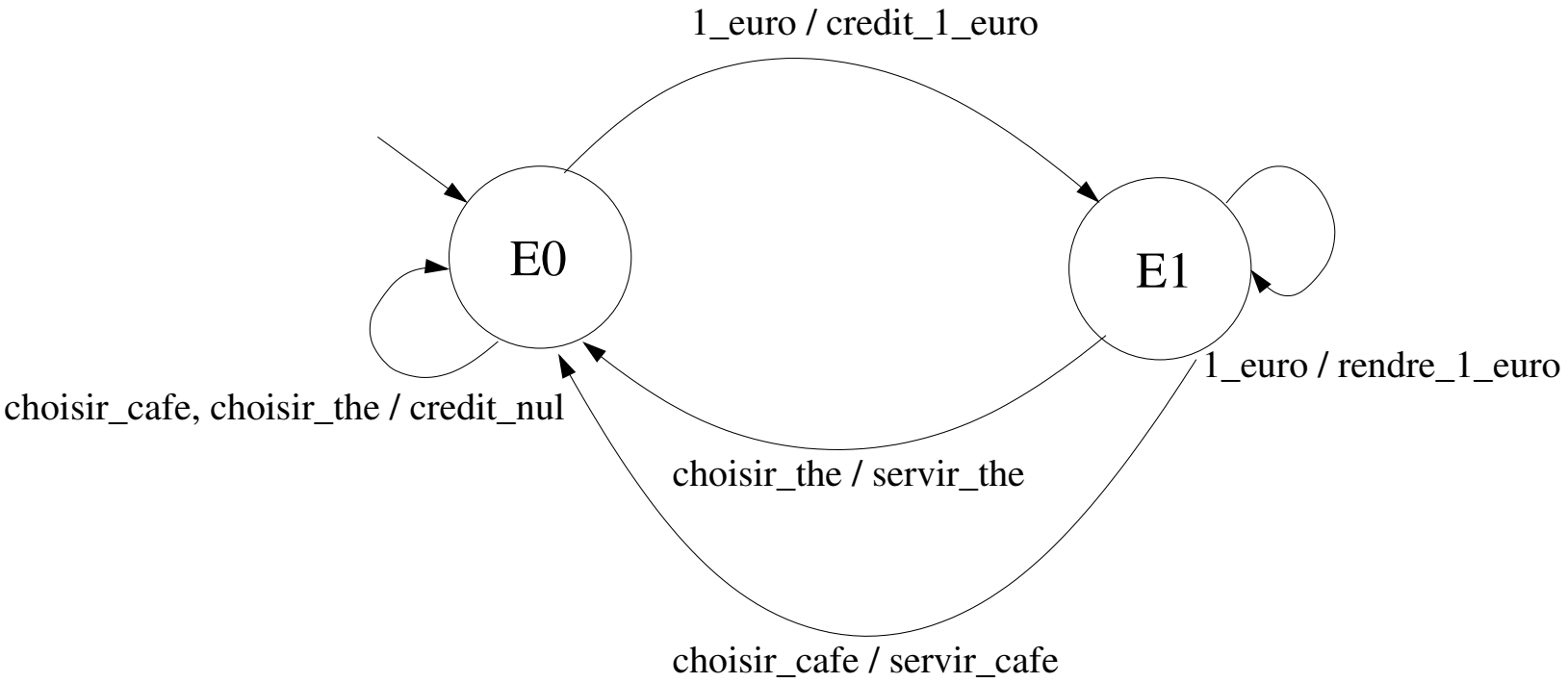
Fonction de transition

Etat / Entrée	1_*euro*	*choisir_the*	*choisir_cafe*
*E*0	*E*1	*E*0	*E*0
*E*1	*E*1	*E*0	*E*0

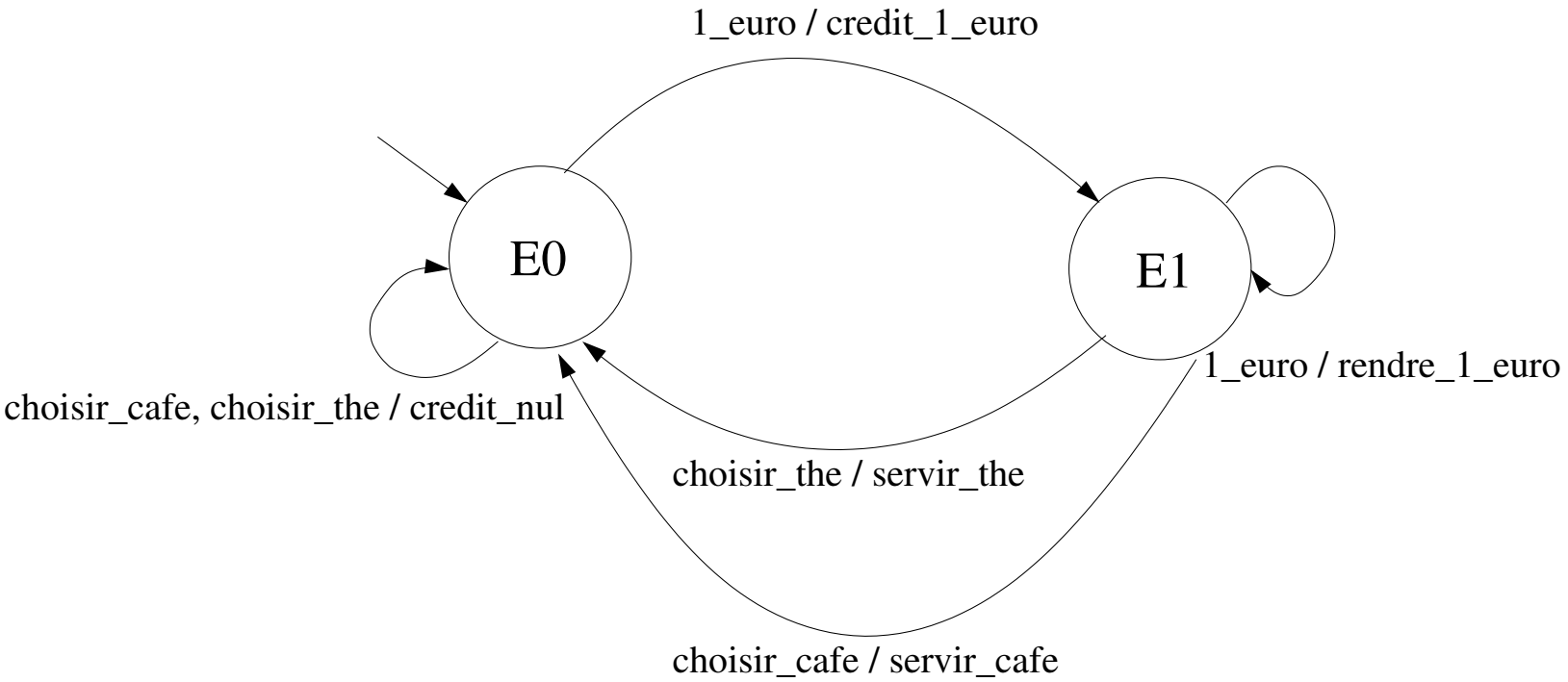
Fonction de sortie

Etat / Entrée	1_*euro*	*choisir_the*	*choisir_cafe*
*E*0	*credit_1_euro*	*credit_nul*	*credit_nul*
*E*1	*rendre_1_euro*	*servir_the*	*servir_cafe*

Forme graphique

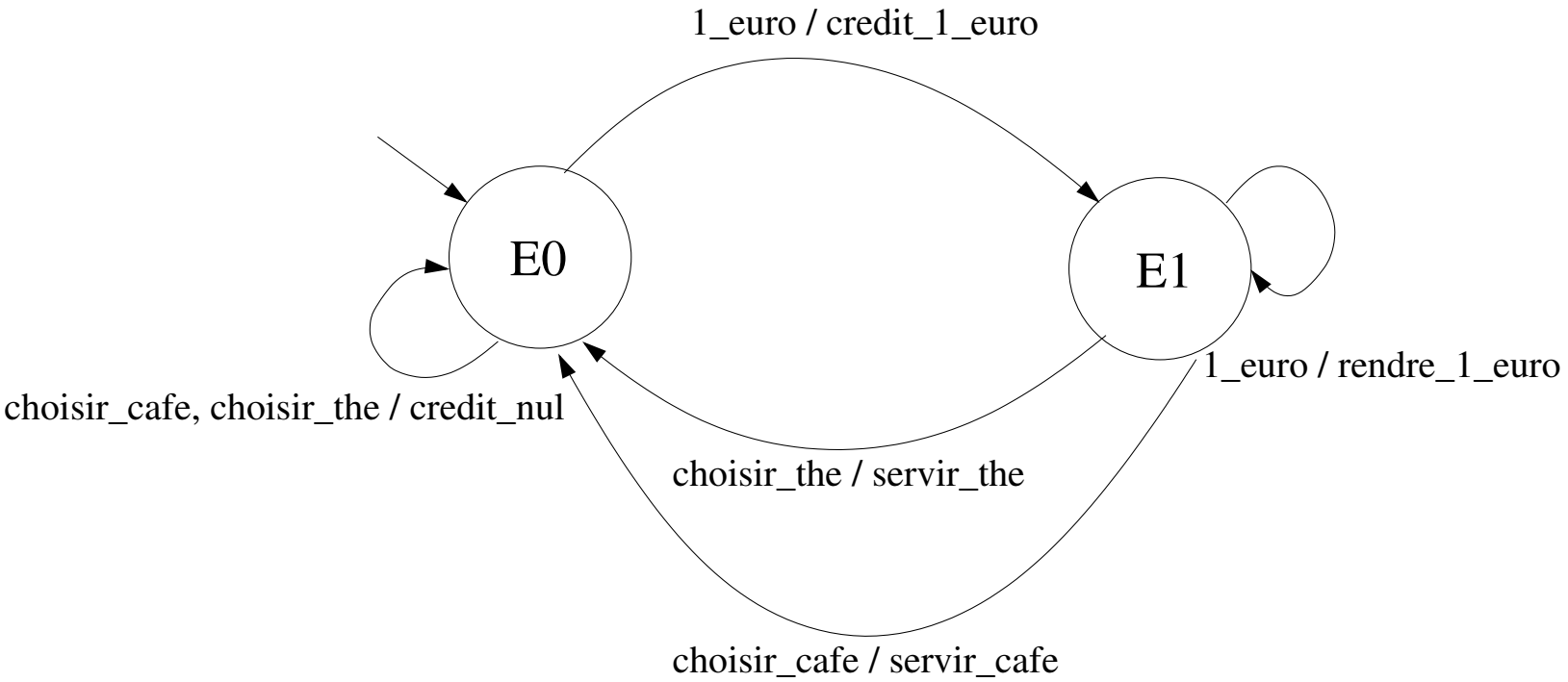


Forme graphique



Remarque 1 : lorsqu'on a deux transitions ayant les mêmes états de départs et d'arrivée et la même sortie : on a une seule flèche étiquetée avec les deux entrées séparées par une virgule suivies du slash et de la sortie.

Forme graphique



Remarque 1 : lorsqu'on a deux transitions ayant les mêmes états de départs et d'arrivée et la même sortie : on a une seule flèche étiquetée avec les deux entrées séparées par une virgule suivies du slash et de la sortie.

Remarque 2 : On peut aussi mettre des conditions sur les entrées.

Plan

Exemple : pile ou face (1/2)

Anémone et **Barnabé** jouent à pile ou face.

- **Anémone** marque un point quand la pièce tombe sur **pile**, **Barnabé** marque un point quand la pièce tombe sur **face**.
- Ils jouent tant que l'un des joueurs n'a pas 2 points de plus que l'autre. Dès que l'un des joueurs a 2 points de plus que l'autre, il gagne la partie.

Les entrées sont donc :

Exemple : pile ou face (1/2)

Anémone et **Barnabé** jouent à pile ou face.

- **Anémone** marque un point quand la pièce tombe sur **pile**, **Barnabé** marque un point quand la pièce tombe sur **face**.
- Ils jouent tant que l'un des joueurs n'a pas 2 points de plus que l'autre. Dès que l'un des joueurs a 2 points de plus que l'autre, il gagne la partie.

Les entrées sont donc : $\{Pile, Face\}$

Exemple : pile ou face (1/2)

Anémone et **Barnabé** jouent à pile ou face.

- **Anémone** marque un point quand la pièce tombe sur **pile**, **Barnabé** marque un point quand la pièce tombe sur **face**.
- Ils jouent tant que l'un des joueurs n'a pas 2 points de plus que l'autre. Dès que l'un des joueurs a 2 points de plus que l'autre, il gagne la partie.

Les entrées sont donc : $\{Pile, Face\}$

Il y aura trois sorties possible $\{A\_Vainqueur, B\_Vainqueur, Rien\}$:

- Les deux premières sorties sonts associées aux transitions qui permettent d'atteindre un des deux états finaux,
- la sortie « Rien » est associée aux autres transitions.

Exemple : pile ou face (1/2)

Anémone et **Barnabé** jouent à pile ou face.

- **Anémone** marque un point quand la pièce tombe sur **pile**, **Barnabé** marque un point quand la pièce tombe sur **face**.
- Ils jouent tant que l'un des joueurs n'a pas 2 points de plus que l'autre. Dès que l'un des joueurs a 2 points de plus que l'autre, il gagne la partie.

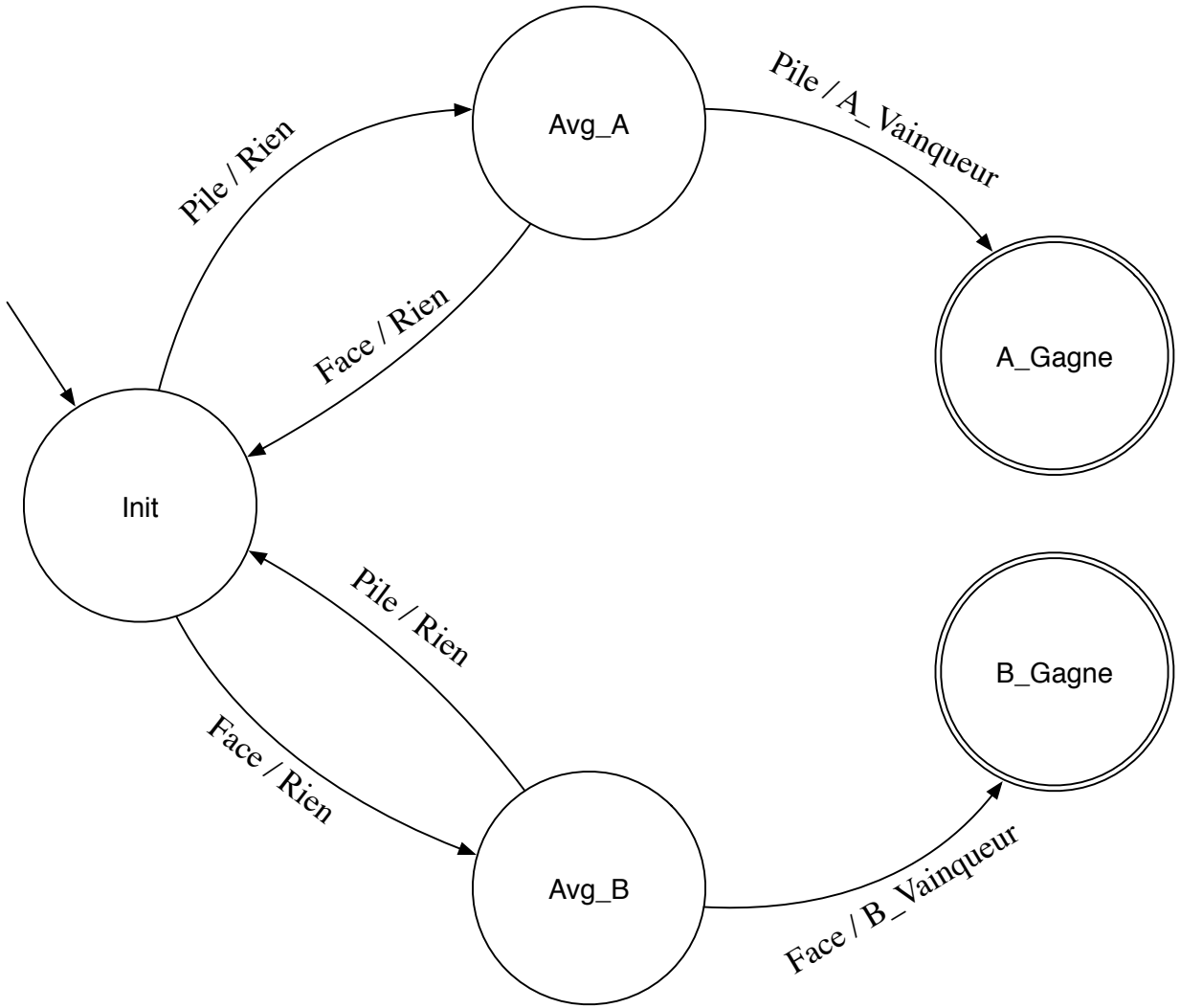
Les entrées sont donc : $\{Pile, Face\}$

Il y aura trois sorties possible $\{A\_Vainqueur, B\_Vainqueur, Rien\}$:

- Les deux premières sorties sonts associées aux transitions qui permettent d'atteindre un des deux états finaux,
- la sortie « Rien » est associée aux autres transitions.

Dessinez l'automate.

Exemple : pile ou face (2/2)



Algorithme

```
etat_courant = Init ;

while (! FINI ) {
     entree = lire_entree() ;
     sortie = sortie(etat_courant, entree) ;
     etat_suivant = transition(etat_courant, entree) ;
     traiter_sortie(sortie) ;
     etat_courant = etat_suivant ;
     mise a jour de FINI
    }
```

Algorithme

```
etat_courant = Init ;

while (! FINI ) {
     entree = lire_entree() ;
     sortie = sortie(etat_courant, entree) ;
     etat_suivant = transition(etat_courant, entree) ;
     traiter_sortie(sortie) ;
     etat_courant = etat_suivant ;
     mise a jour de FINI
    }
```

Remarque : `FINI` peut valoir toujours faux (simulation sans fin), ou être vrai si on atteint tel ou tel état (par exemple un état final).

En langage C

```
#include <stdio.h>

// entrees
#define Pile 0
#define Face 1

// sorties
#define A_Vainqueur 0
#define B_Vainqueur 1
#define Rien 2

// Etats
#define Init 0
#define Avg_A 1
#define Avg_B 2
#define A_Gagne 3
#define B_Gagne 4
```

En langage C

```
int f_transition[5][2] = {
        /* de Init */          {Avg_A, Avg_B},
        /* de Avg_A */         {A_Gagne, Init},
        /* de Avg_B  */        {Init, B_Gagne},
        /* de A_Gagne */       {A_Gagne, A_Gagne},
        /* de B_Gagne  */      {B_Gagne, B_Gagne}
                             } ;

int f_sortie[5][2] = {
        /* de Init */          {Rien, Rien},
        /* de Avg_A */         {A_Vainqueur, Rien},
        /* de Avg_B */         {Rien, B_Vainqueur},
        /* de A_Gagne */       {Rien, Rien},
        /* de B_Gagne */       {Rien, Rien}
                           } ;
```

En langage C

```
int lire_entree() {
char cc ;

do
      scanf("%c", &cc) ;
while ((cc != 'P') && (cc != 'F')) ;

if (cc == 'P') return Pile ;
else return Face ;
}
```

En langage C

```
void traiter_sortie (int message) {
switch (message) {
     case A_Vainqueur : printf("Victoire de Anemone (Pile)\n") ; break ;
     case B_Vainqueur : printf("Victoire de Barnabe (Face)\n") ; break ;
     default : break ;
    }
}
```

En langage C

```
int Est_final(int etat) {
    return ((etat == A_Gagne) || (etat == B_Gagne)) ;
}
```

En langage C

```
void simul_automate() {
int etat_courant, etat_suivant ;
int entree, sortie ;

        etat_courant = Init ;
        while (! Est_final(etat_courant)) {
                entree = lire_entree() ;
                sortie = f_sortie[etat_courant][entree] ;
                etat_suivant = f_transition[etat_courant][entree] ;
                traiter_sortie(sortie) ;
                etat_courant = etat_suivant ;
               }
}

int main() {
        simul_automate() ;
        return 0 ;
}
```