Automates

Système et environnement de programmation

Université Grenoble Alpes

Plan

Automate pour modéliser un système

Définition : **formalisme** (ou outil mathématique) qui permet de **modéliser** le comportement de certaines classes de 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
- processeurs
- ...

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 (mécanique, energétique, thermique, ...) mais aussi en électronique et en informatique !

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, . . .)
- 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, . . .)
- 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 ?
- 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, . . .)
- 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 ?
- sorties qu'il peut alors émettre ?

Cette forme de mémoire est codée avec des états : l'état courant correspond à toutes ou une partie des entrées reçues par le système (son histoire) depuis son initialisation.

Plan

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

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

Plan

Distributeur de boissons

Modélisons un distributeur de boissons très simple

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

Les états

Le choix de boisson doit arriver **après** que le distributeur a 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 (suite)

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 et les sorties

Notre distributeur doit pouvoir répondre à trois types d'entrées

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

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

Les entrées et les sorties

Notre distributeur doit pouvoir répondre à trois types d'entrées

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

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

Les sorties émises sont

- servir un café ou un thé
- informer l'utilisateur sur son crédit
- rendre les pièces excédentaires

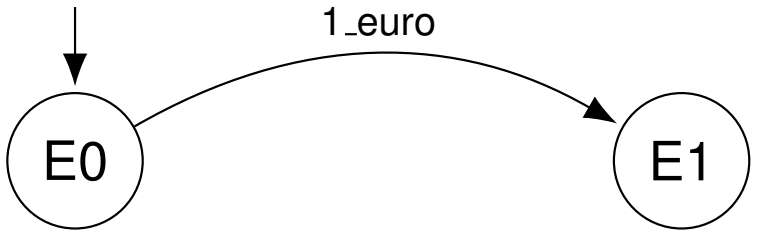
$$S = \{servir\_the, servir\_cafe, rendre\_1\_euro, credit\_nul, credit\_1\_euro\}$$

Fonction de transition

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

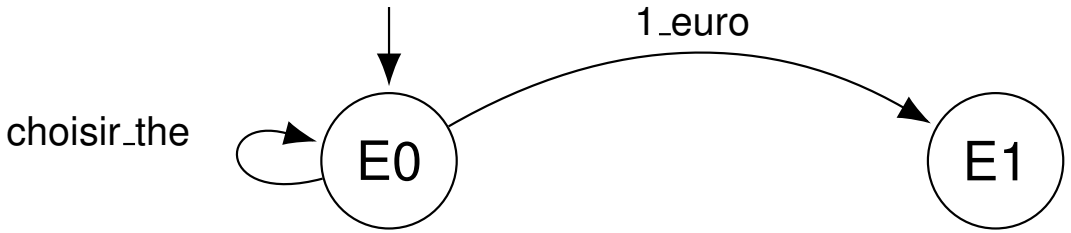
Fonction de transition

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



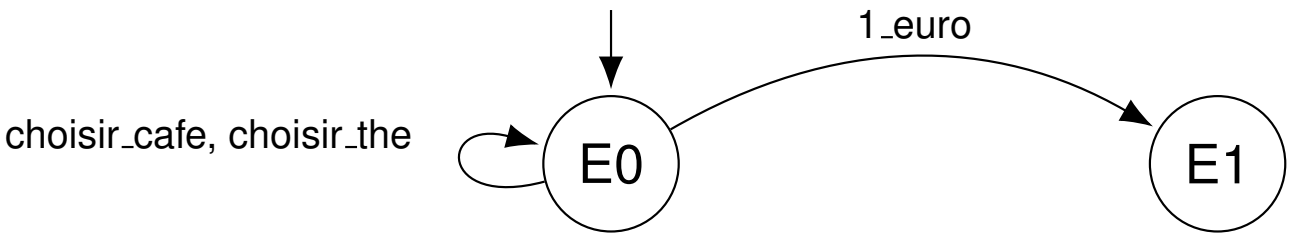
Fonction de transition

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



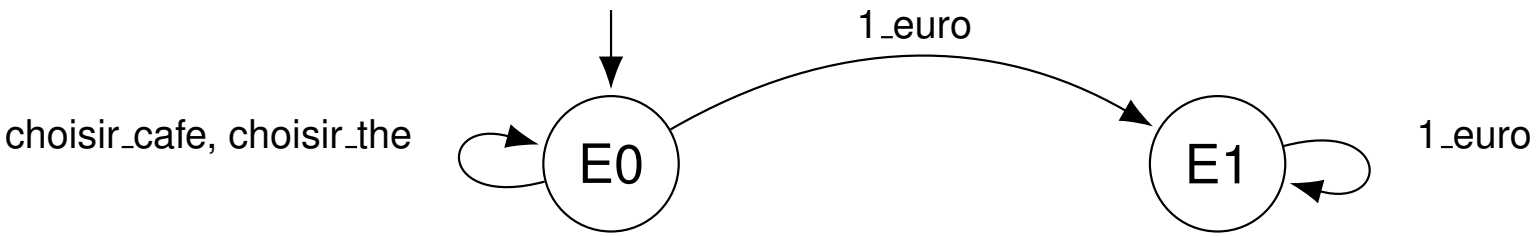
Fonction de transition

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



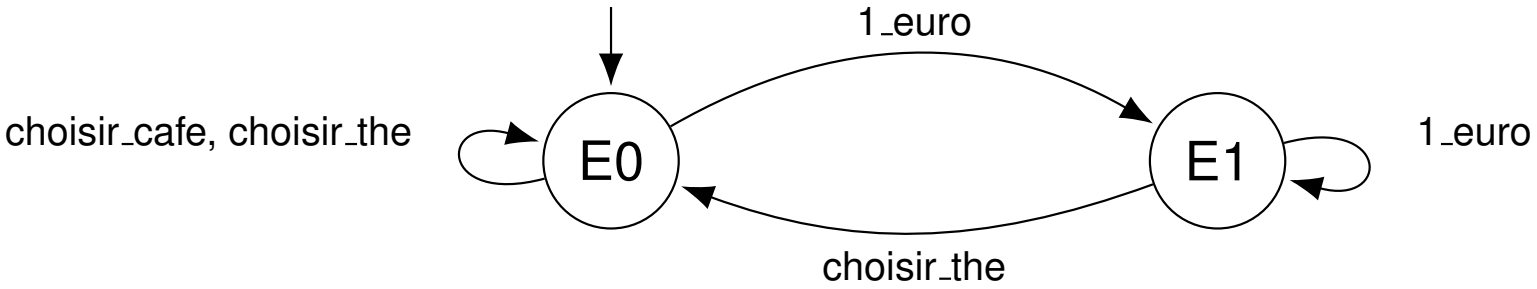
Fonction de transition

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



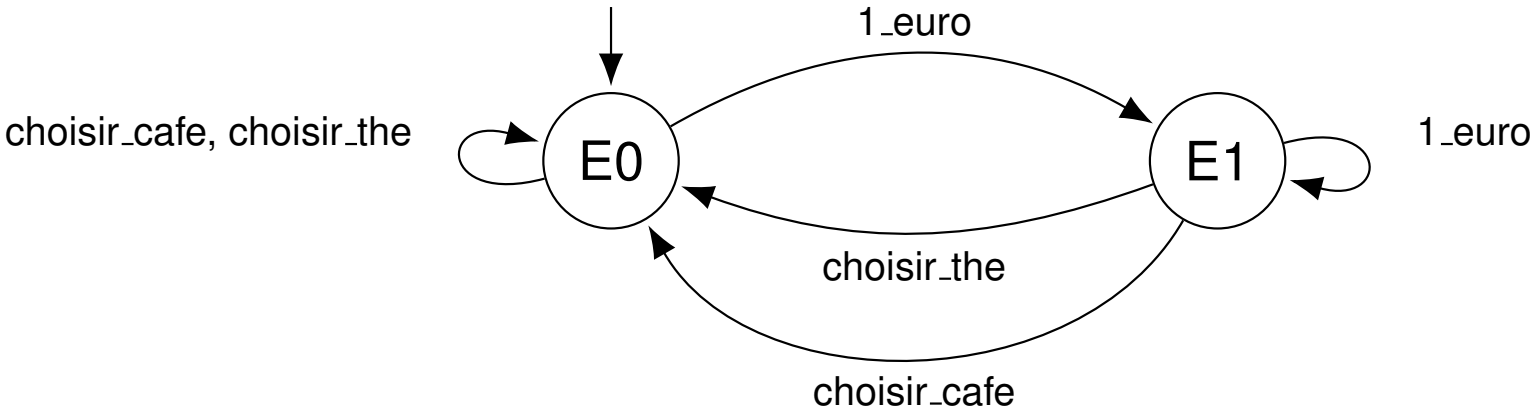
Fonction de transition

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



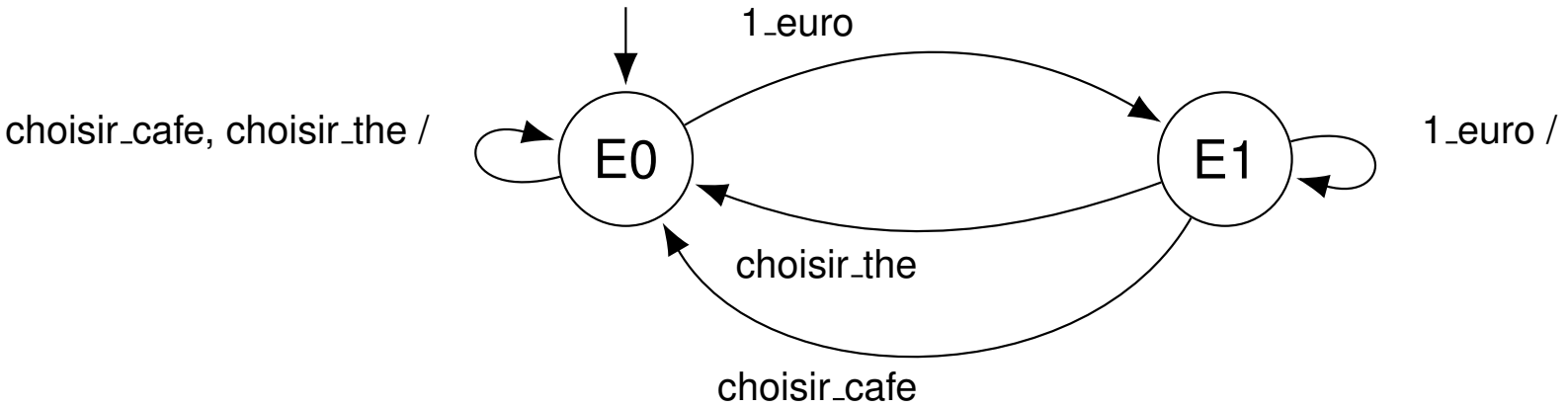
Fonction de transition

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



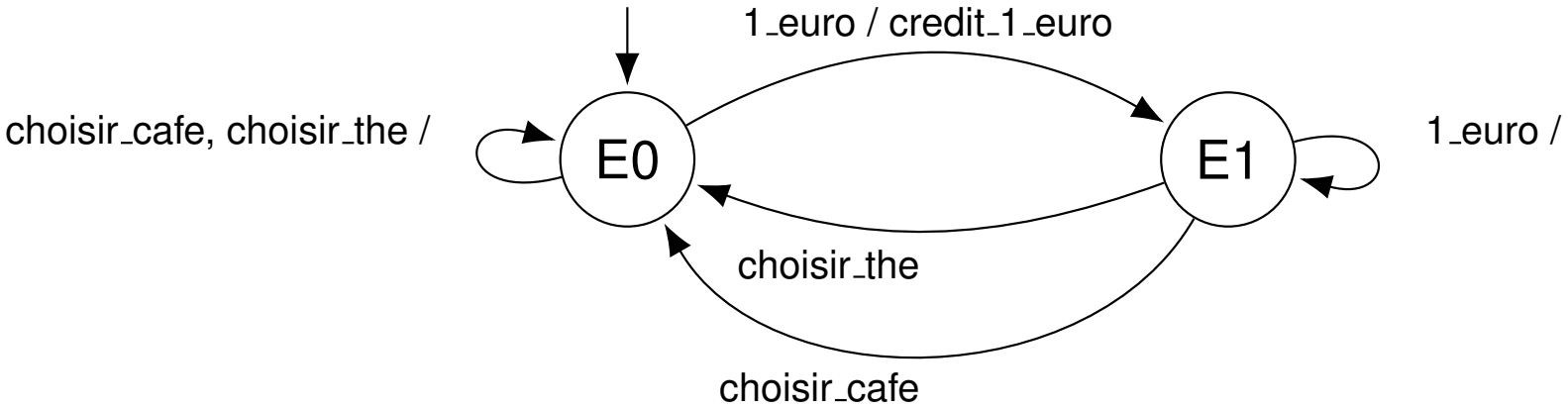
Fonction de sortie

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



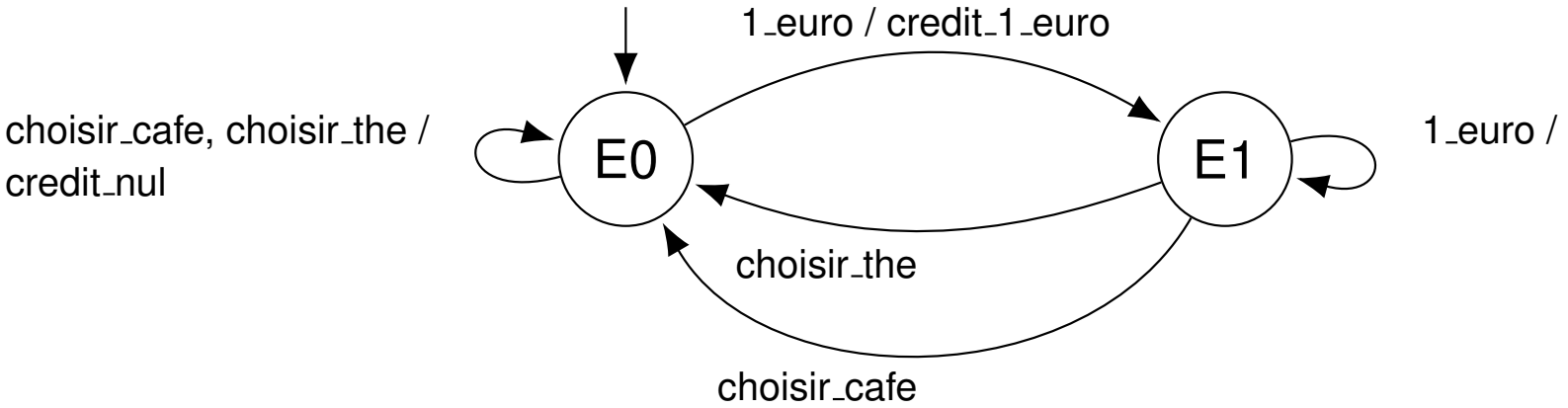
Fonction de sortie

Etat / Entrée	1_*euro*	*choisir_the*	*choisir_cafe*
*E*0	*credit_1_euro*		
*E*1			



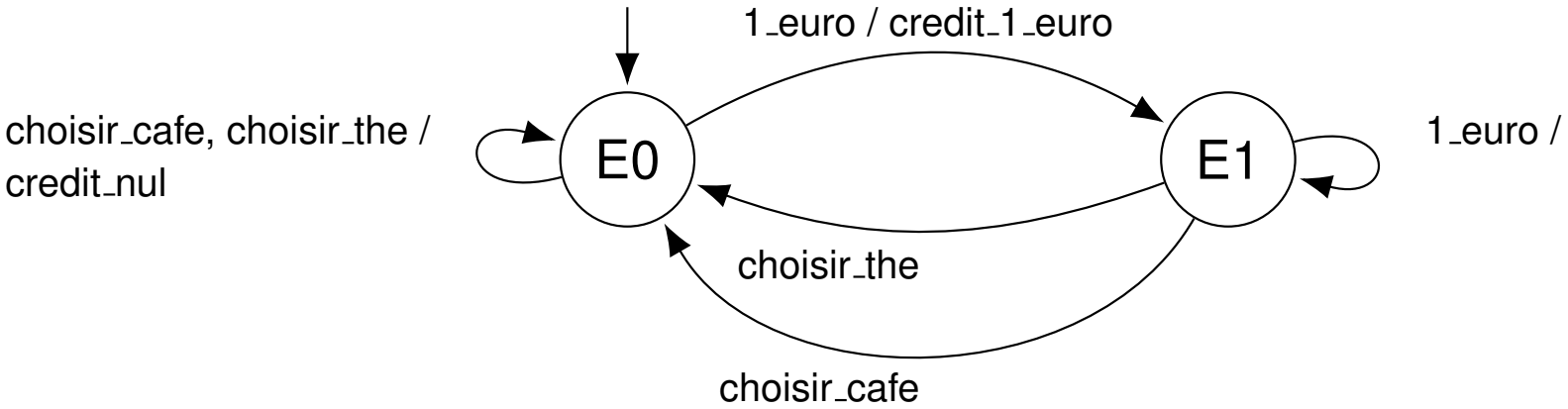
Fonction de sortie

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



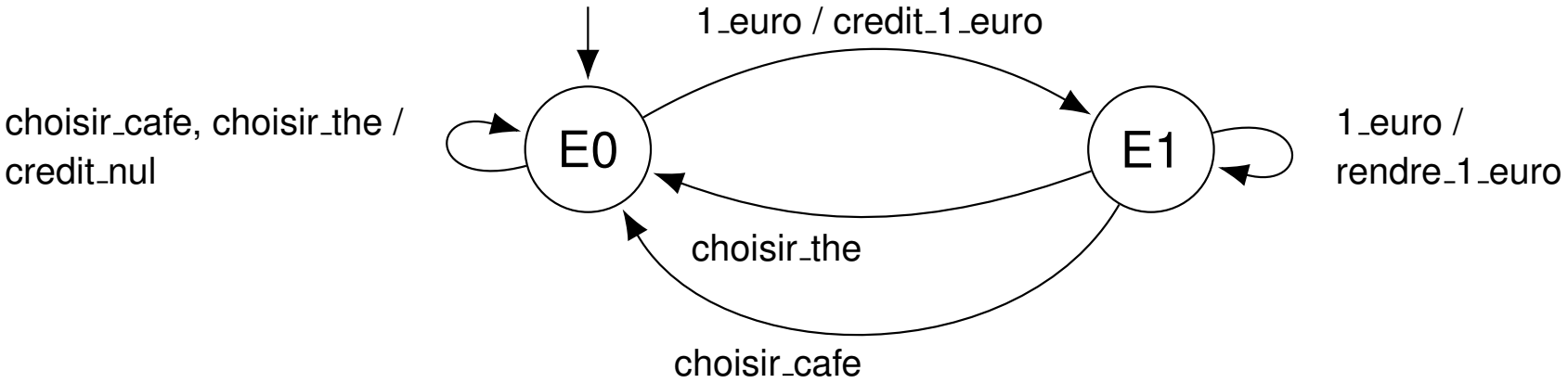
Fonction de sortie

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



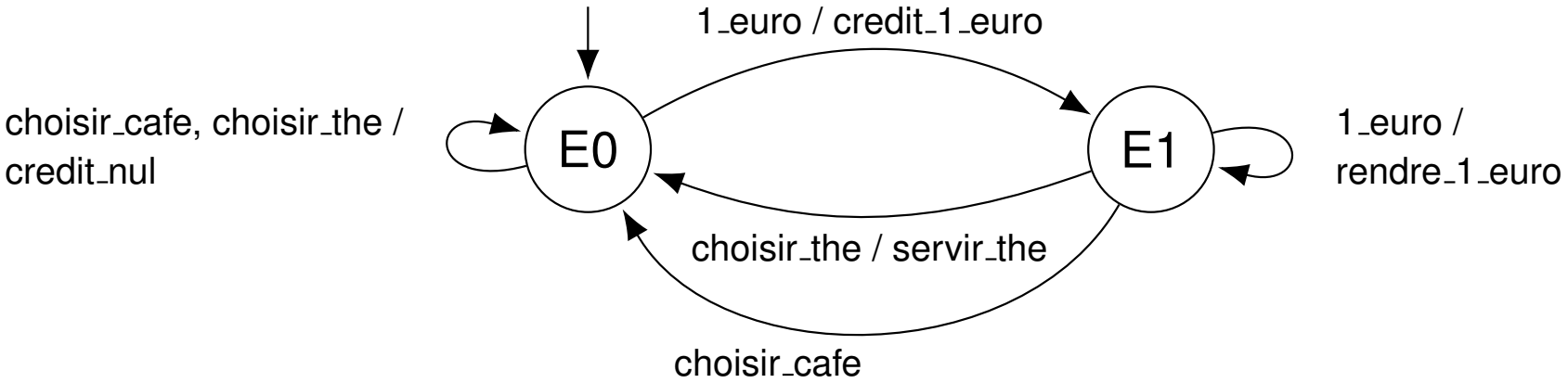
Fonction de sortie

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



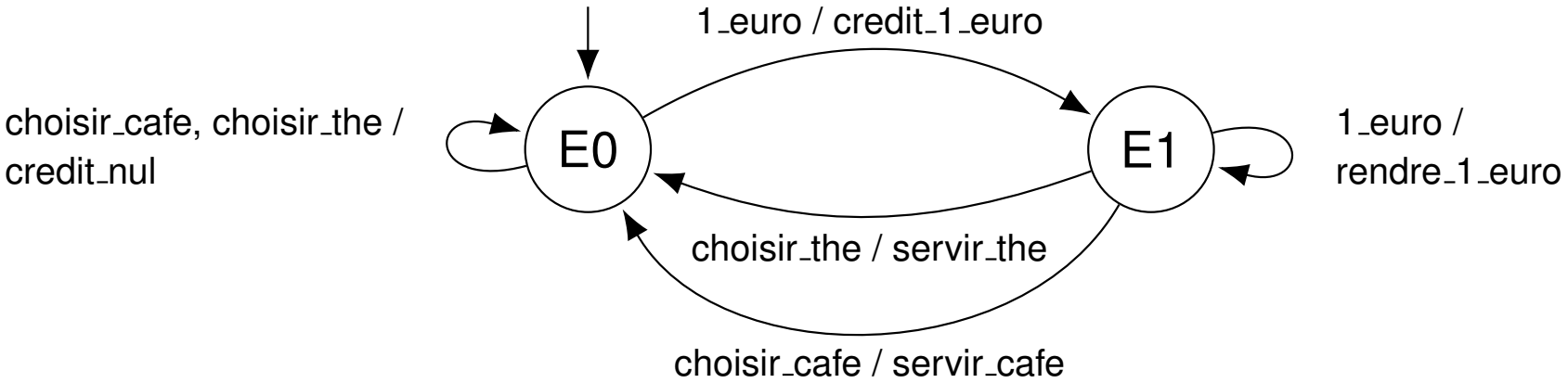
Fonction de sortie

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

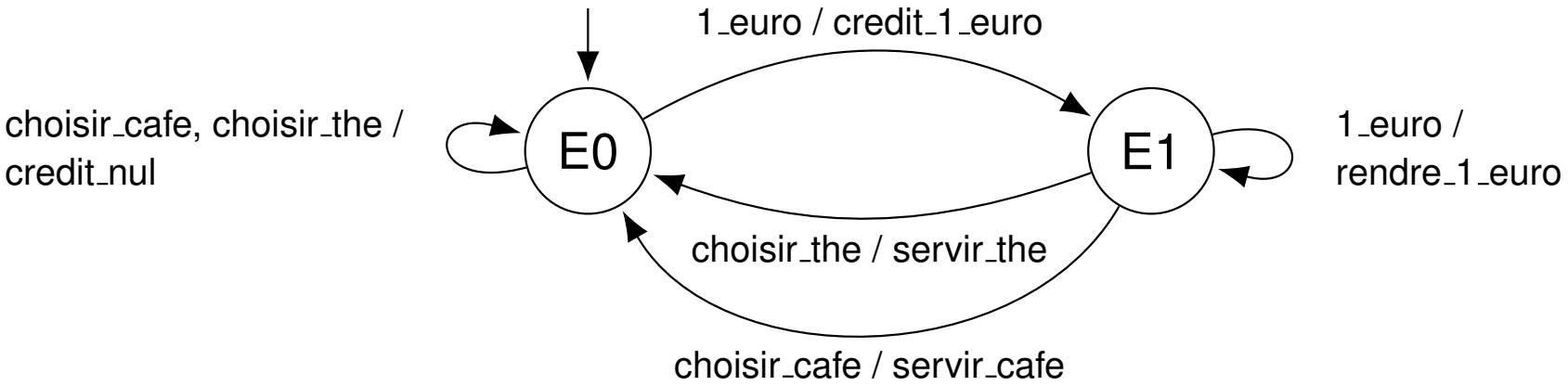


Fonction de sortie

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



Remarques

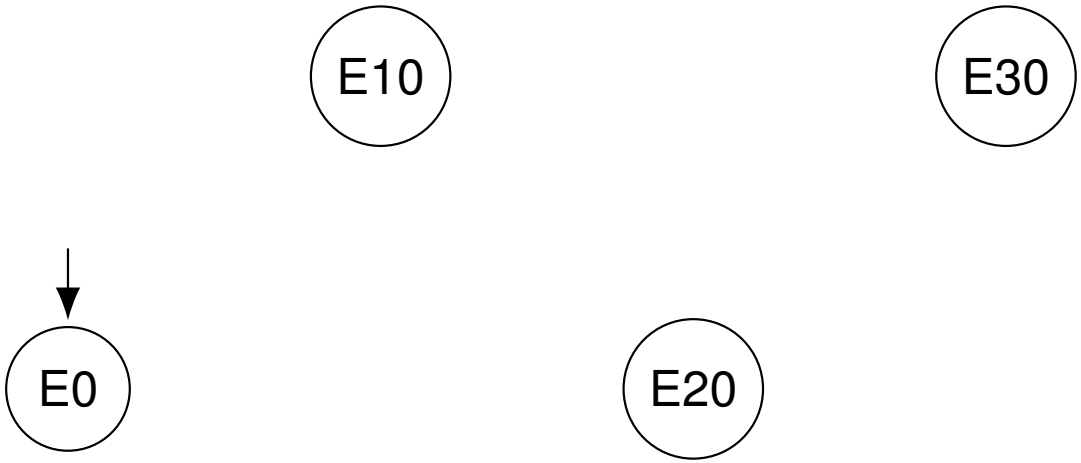
- deux transitions ayant les mêmes états de départs, états d'arrivée et sorties sont combinées avec les deux entrées séparées par une virgule
- dans cet exemple, il n'y a pas d'état final. Un état final sera représenté par un **double cercle**.

Exercice

Variante : le café coût 20 cts et le thé 30 cts.
On peut insérer des pièces de 10 ou 20 cts.

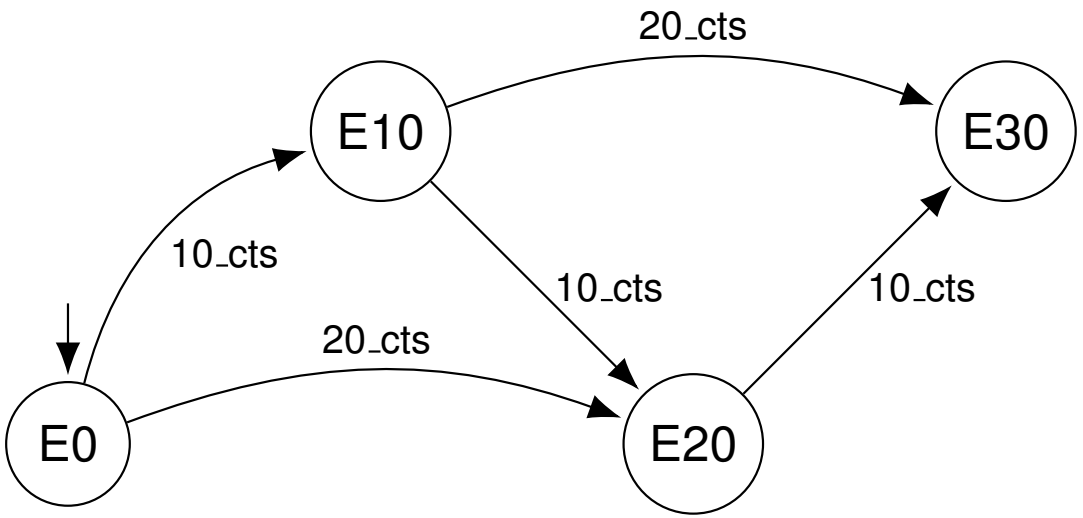
Exercice

Variante : le café coût 20 cts et le thé 30 cts.
On peut insérer des pièces de 10 ou 20 cts.



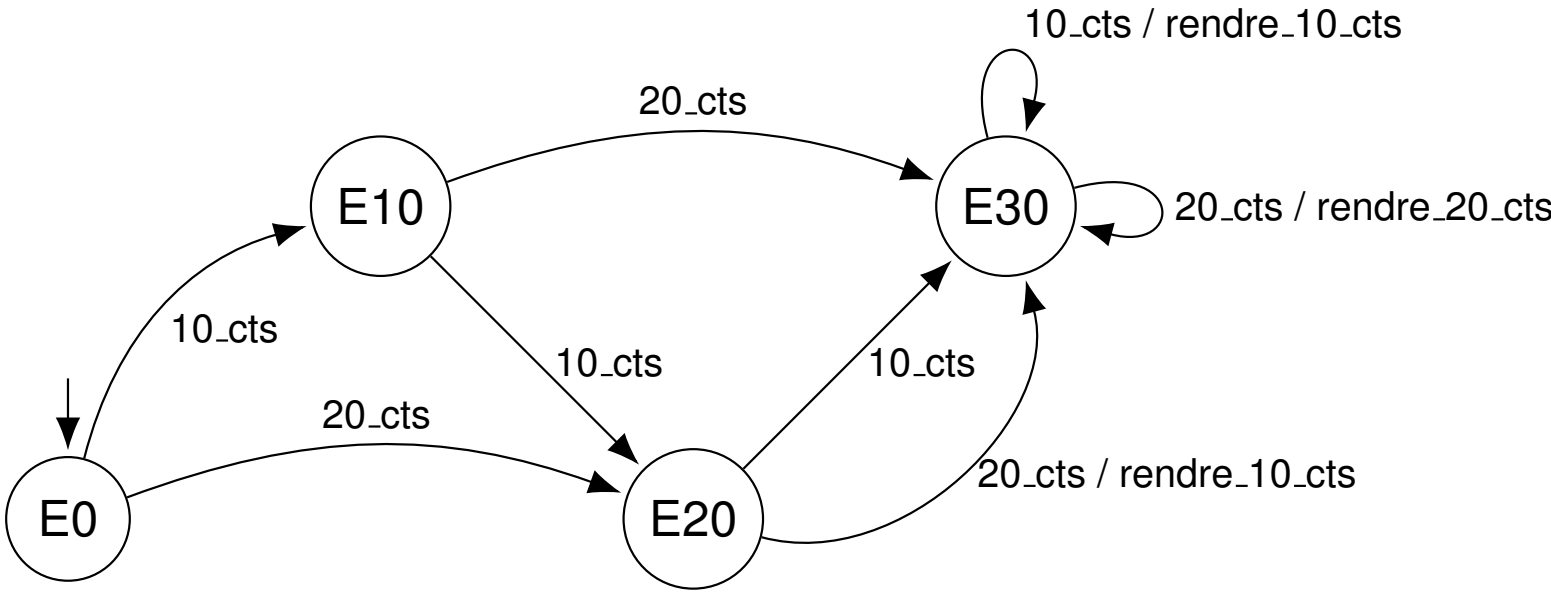
Exercice

Variante : le café coût 20 cts et le thé 30 cts.
On peut insérer des pièces de 10 ou 20 cts.



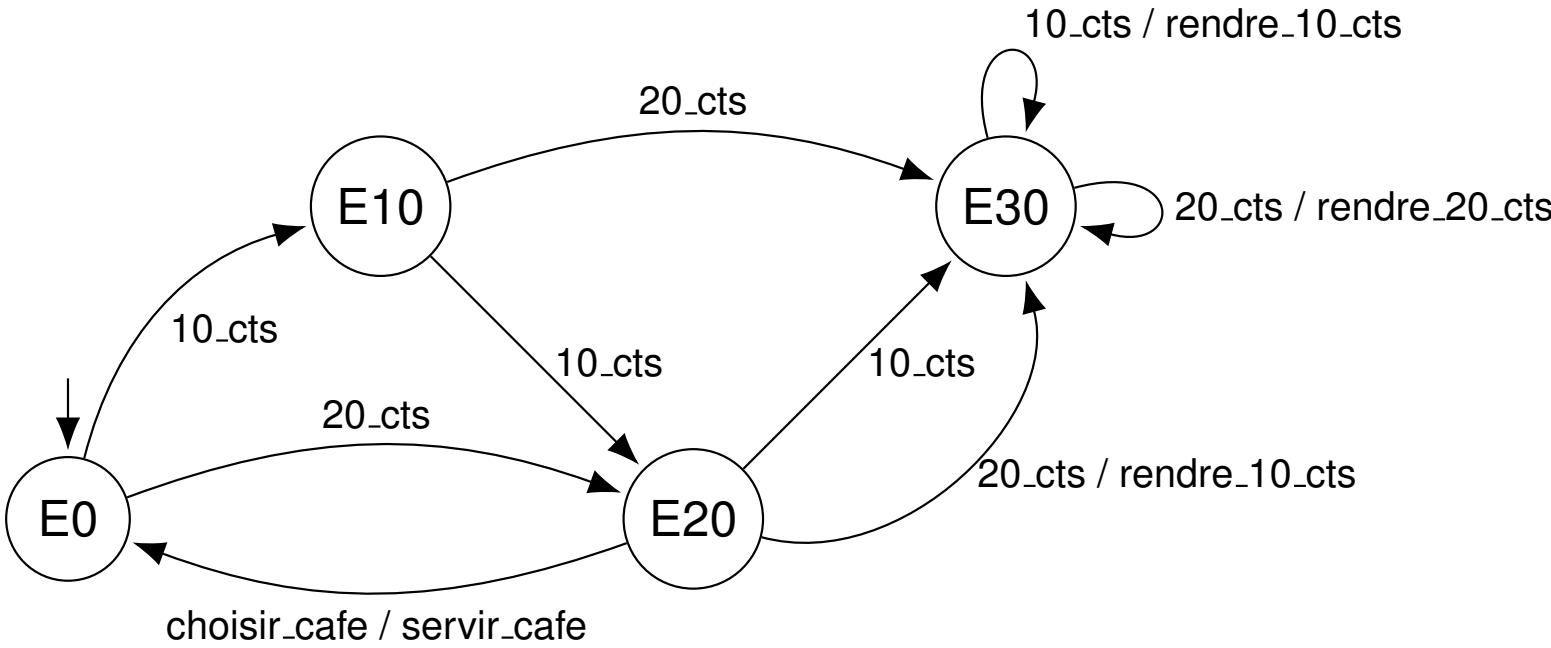
Exercice

Variante : le café coût 20 cts et le thé 30 cts.
On peut insérer des pièces de 10 ou 20 cts.



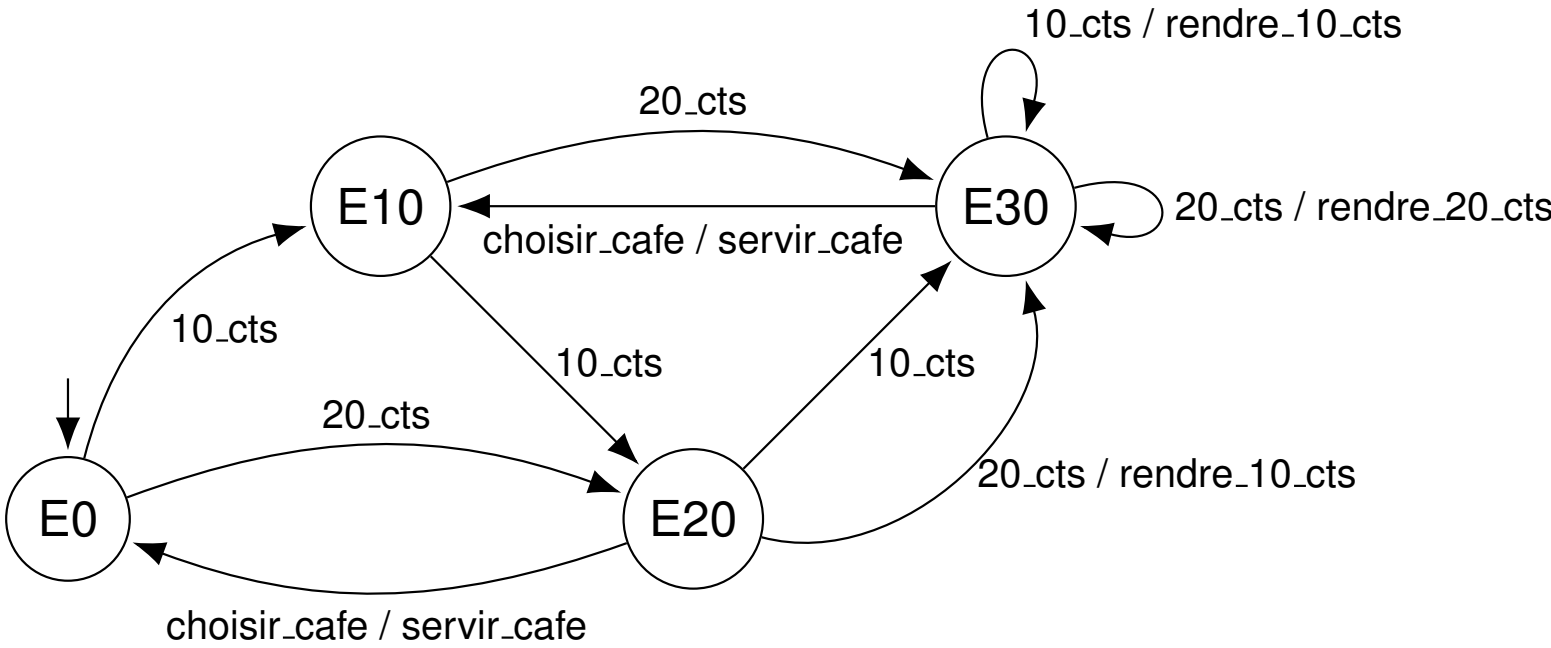
Exercice

Variante : le café coût 20 cts et le thé 30 cts.
On peut insérer des pièces de 10 ou 20 cts.



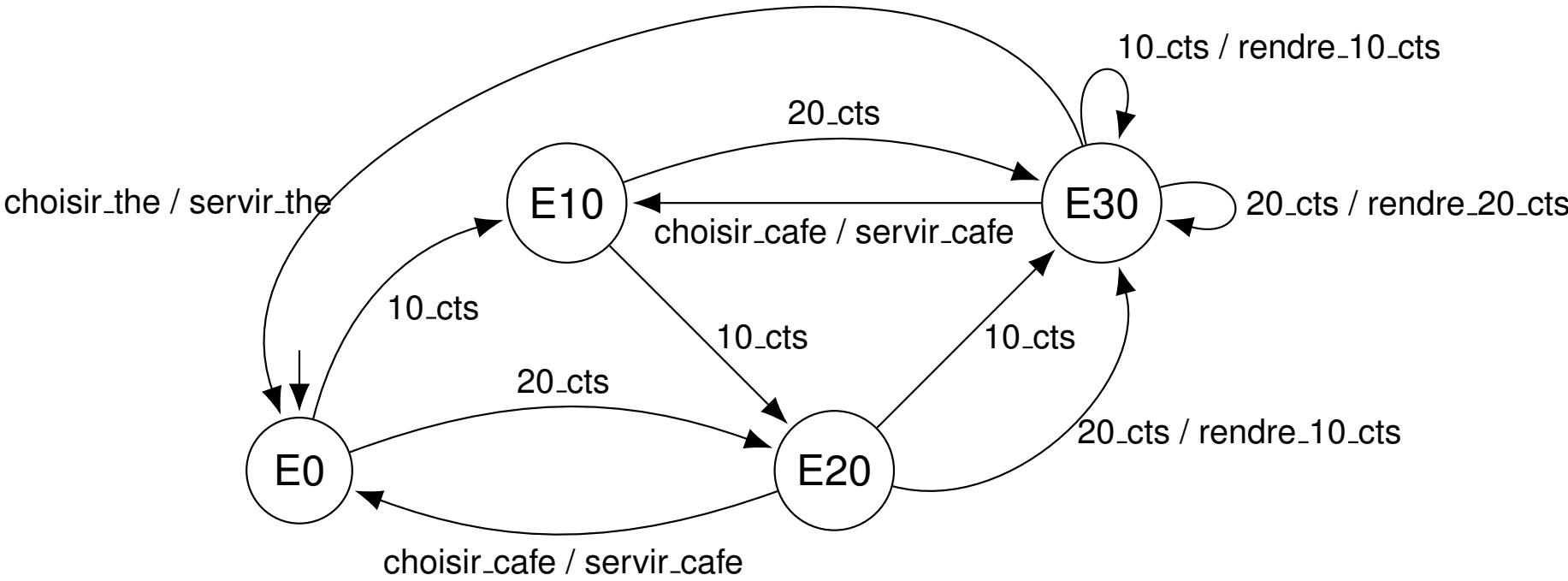
Exercice

Variante : le café coût 20 cts et le thé 30 cts.
On peut insérer des pièces de 10 ou 20 cts.



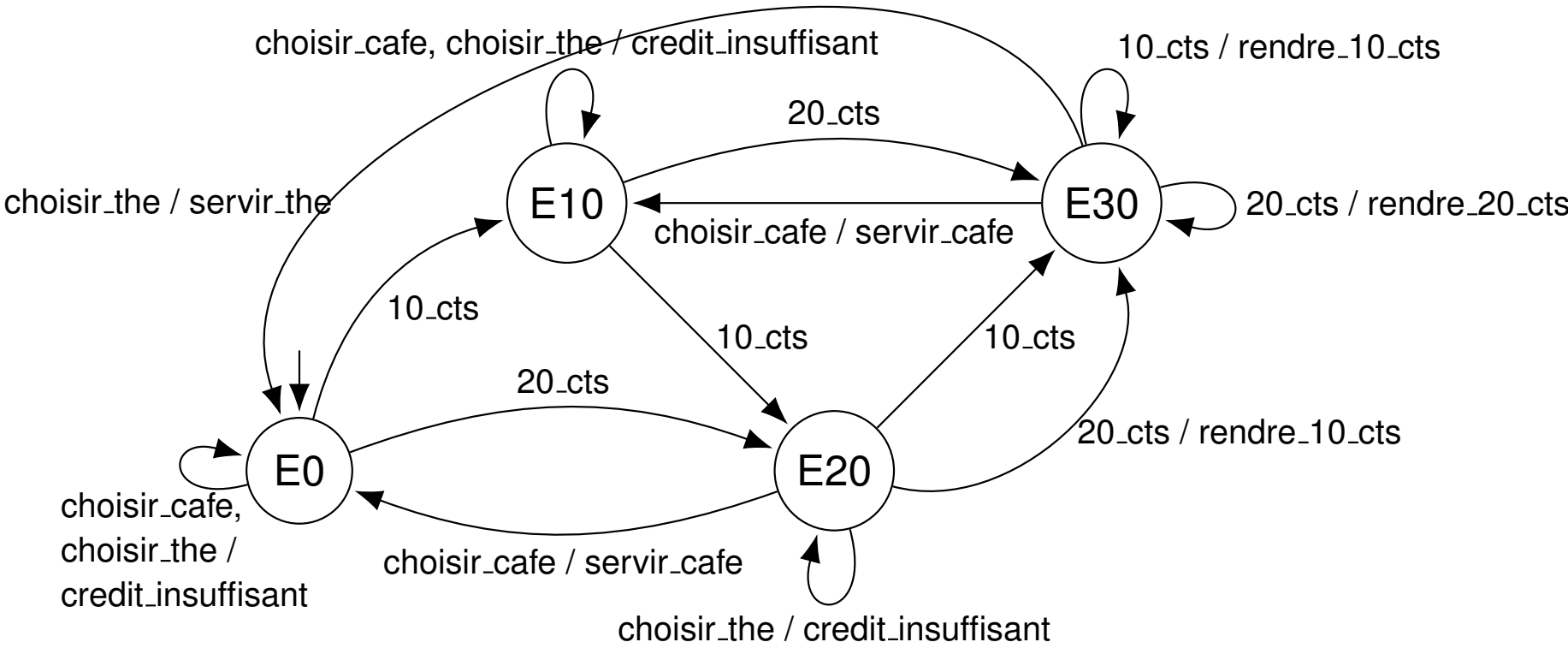
Exercice

Variante : le café coût 20 cts et le thé 30 cts.
On peut insérer des pièces de 10 ou 20 cts.



Exercice

Variante : le café coût 20 cts et le thé 30 cts.
On peut insérer des pièces de 10 ou 20 cts.



Plan

Exemple : pile ou face

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 font trois tirages, à la fin des trois tirages celui qui a le plus de points gagne

Les entrées sont donc

Exemple : pile ou face

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 font trois tirages, à la fin des trois tirages celui qui a le plus de points gagne

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

Il y aura trois sorties possibles

Exemple : pile ou face

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

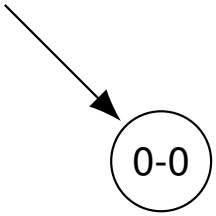
- **Anémone** marque un point quand la pièce tombe sur **pile**, **Barnabé** marque un point quand la pièce tombe sur **face**
- Ils font trois tirages, à la fin des trois tirages celui qui a le plus de points gagne

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

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

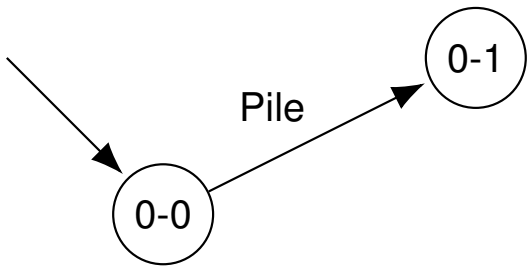
- Les deux premières sorties sont associées aux transitions qui permettent d'atteindre un des deux états finaux,
- la sortie $\ll$ Rien $\gg$ est associée aux autres transitions.

Automate du pile ou face



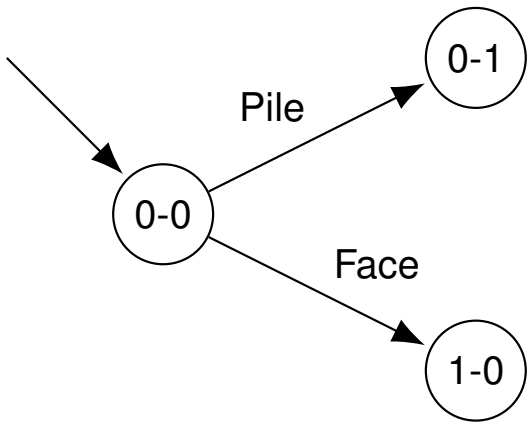
Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



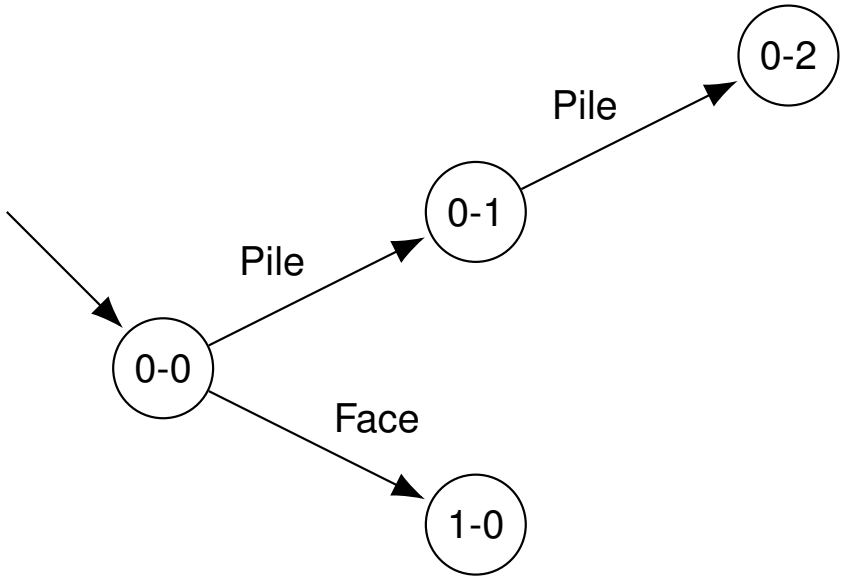
Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



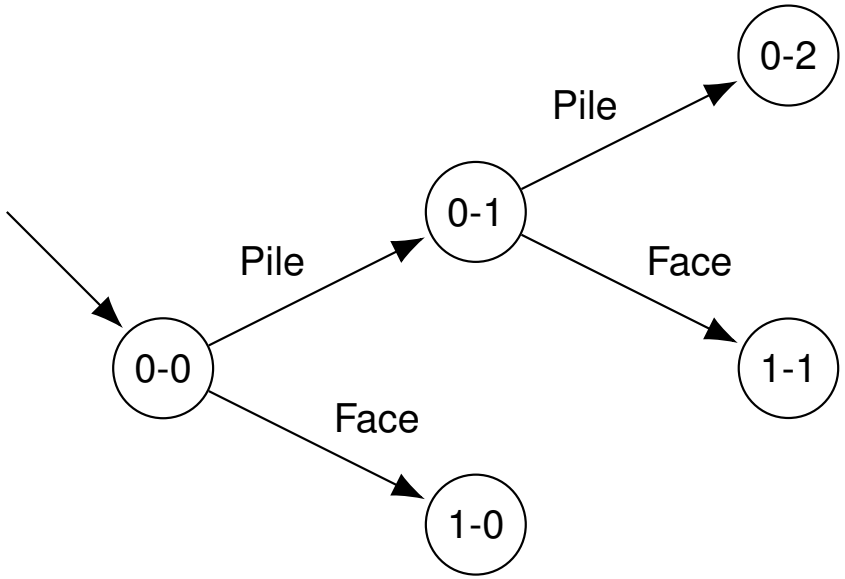
Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



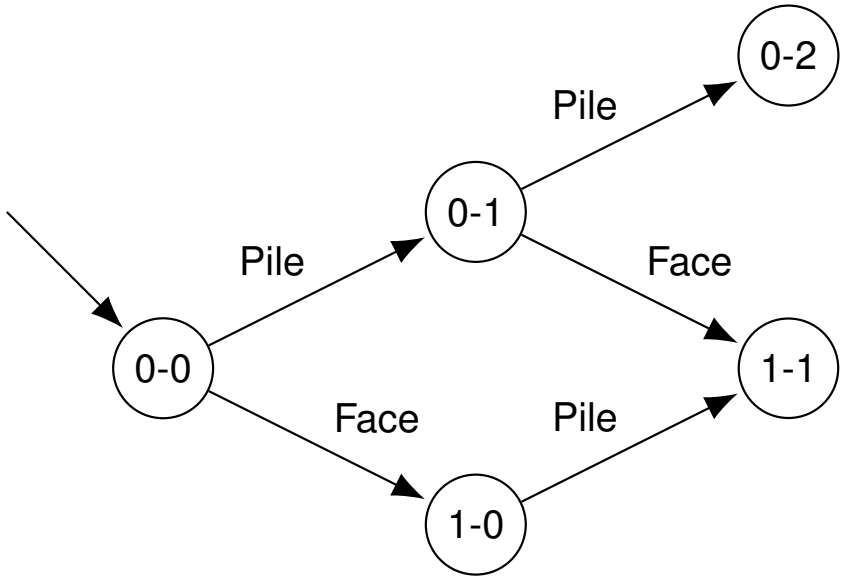
Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



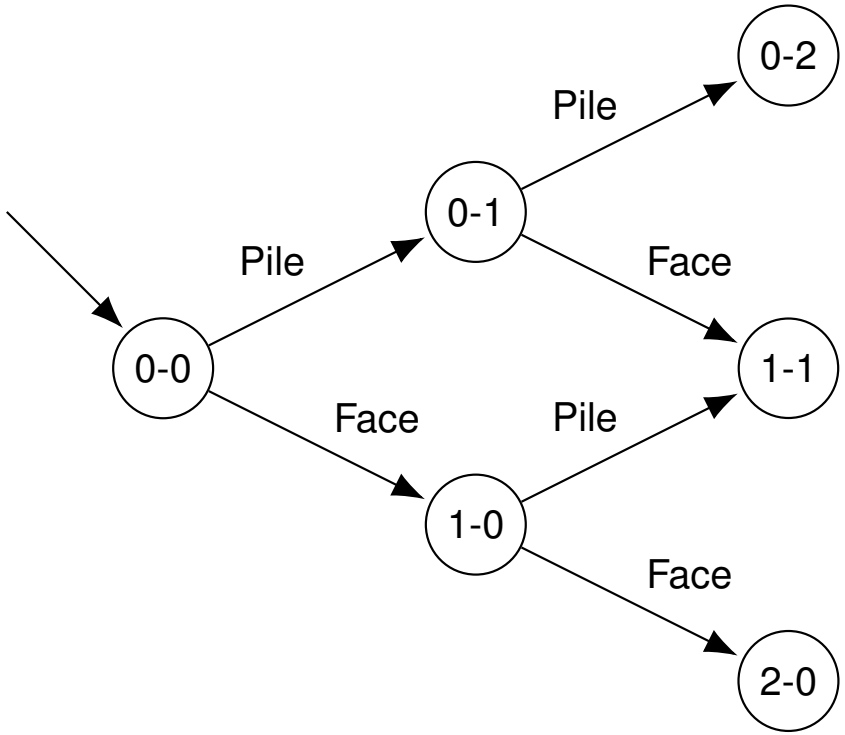
Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



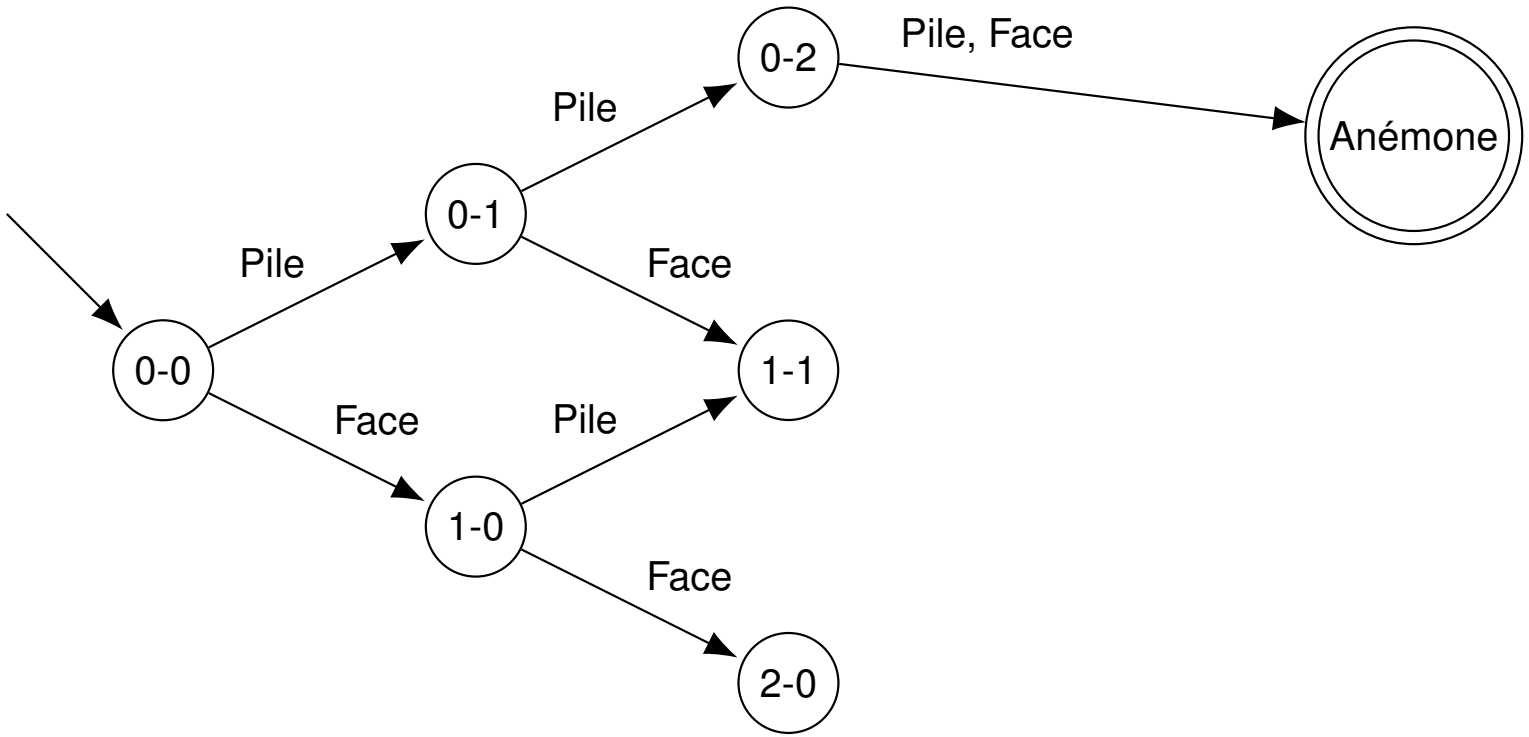
Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



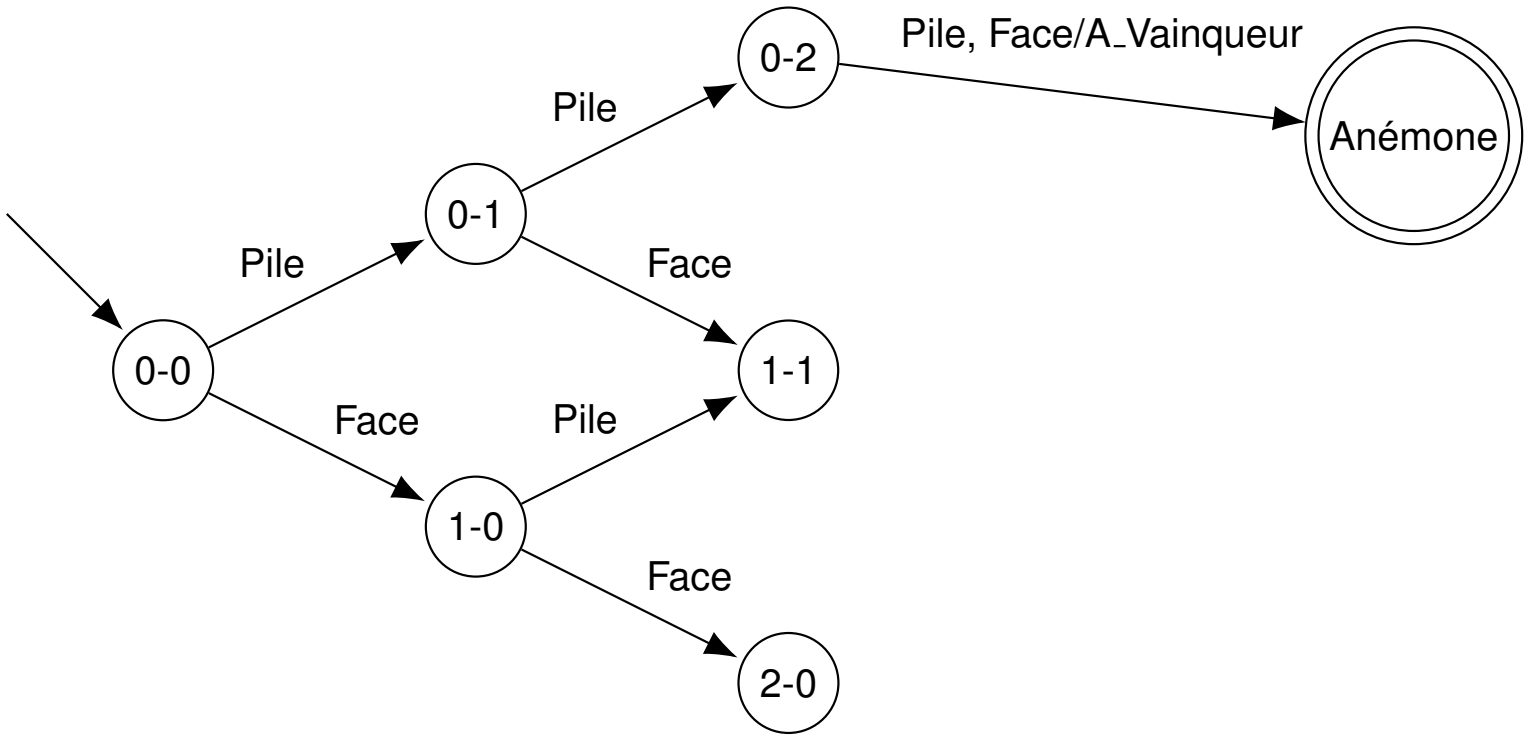
Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



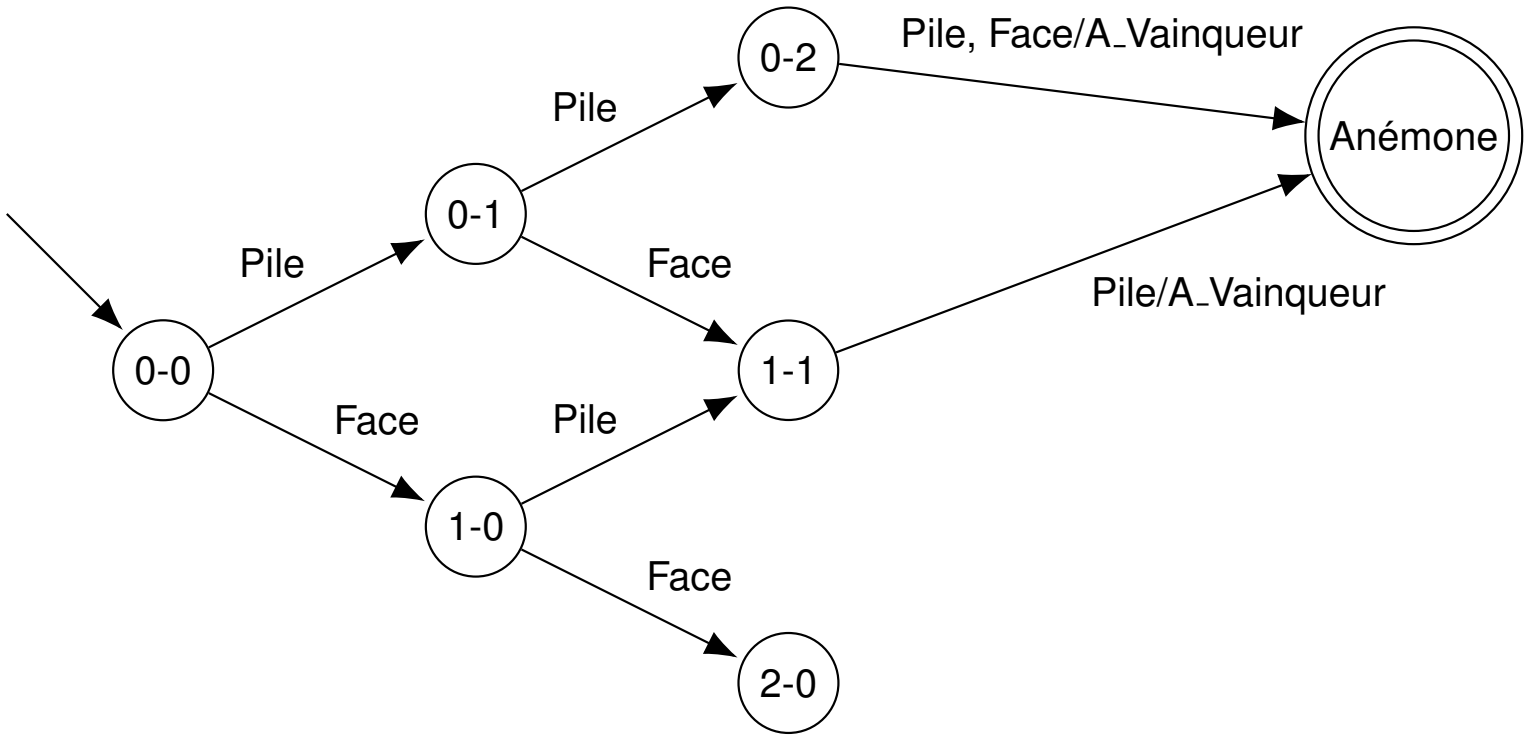
Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



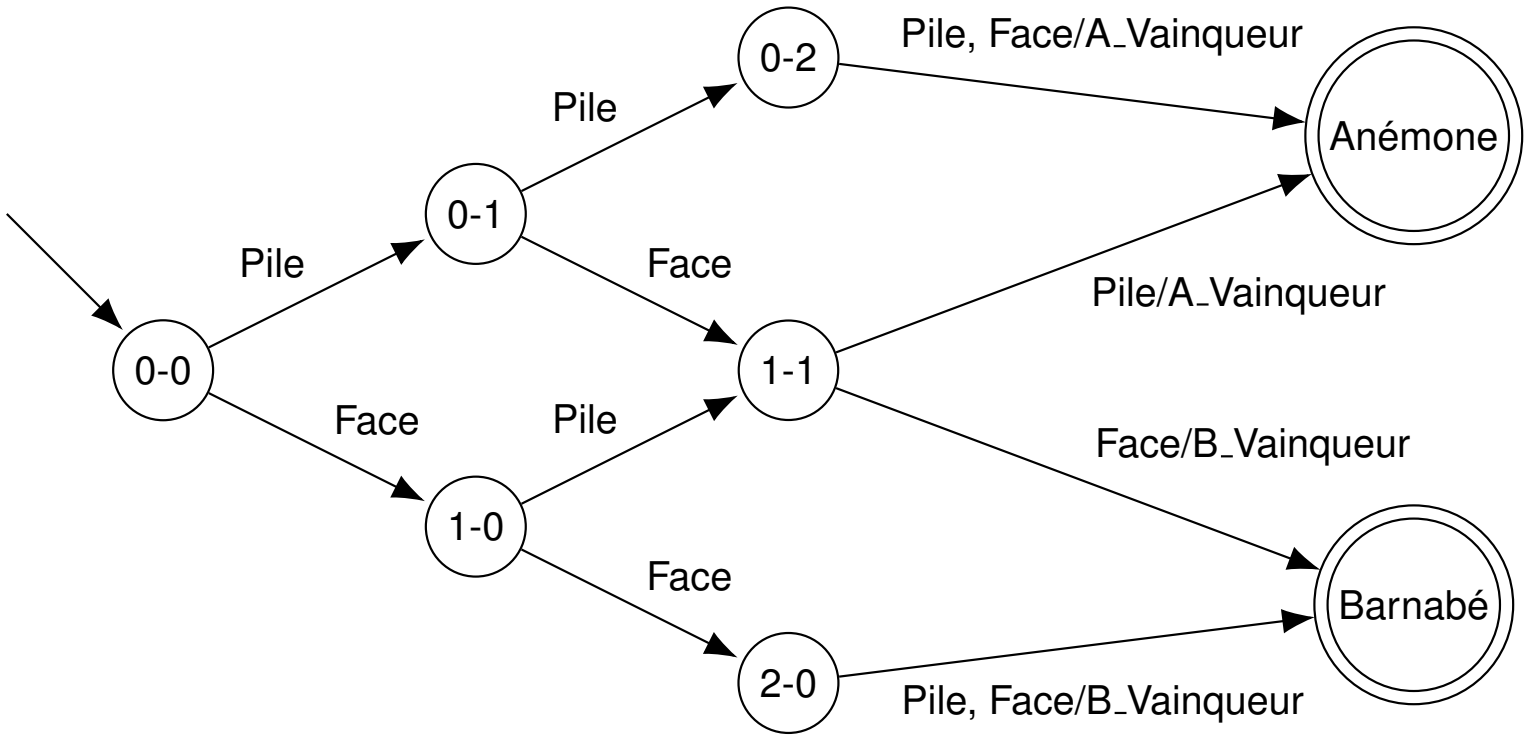
Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



Remarque : on omet les sorties vides (lisibilité)

Automate du pile ou face



Remarque : on omet les sorties vides (lisibilité)

Plan

Algorithme de simulation

```
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;
    }
```

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

Fonctions de transition et de sortie

Plusieurs options

- Forme fonctionnelle

```
int transition(int etat_courant, char entree) {
    switch (etat_courant) {
      case 0:
        switch (entree) {
          case 'c': return 0;
          case '1': return 1;
          ...
```

- Forme tabulée

  On remplit un tableau nommé `transition`, puis

  `etat_suivant = transition[etat_courant][entree];`

  Il faut que `etat_courant` et `entree` soient des entiers