Séquence 10

Analyser, décrire et prévoir le comportement d'un système à évènements discrets

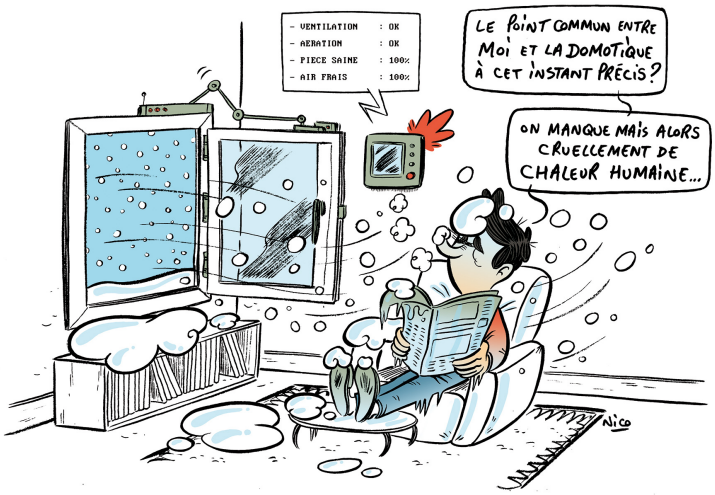
Objectif

Les systèmes peuvent être décomposés en deux familles : ceux dont les grandeurs peuvent prendre une infinité de valeurs possibles (qu'on appelle systèmes continus) et ceux dont les grandeurs sont **discrètes** et ne prennent donc qu'un nombre fini de valeurs différentes. Ils sont appelés **systèmes à événements discrets** (ou **SED**). Dans ce cours, nous étudierons comment décrire, analyser et prévoir leur comportement.

En particulier, nous étudierons d'abord comment coder convenablement des grandeurs discrètes et comment les manipuler. Ensuite, nous découvrirons un outil un outil adapté pour décrire le comportement de systèmes à évènements discrets complexes.

Table des matières

Page



1 Systèmes logiques combinatoires

1.1 Définition

À savoir

Une grandeur **discrète** ne prend des valeurs que dans un ensemble dénombrable (entiers par exemple).

Une grandeur **logique** prend des valeurs booléennes (*vrai* ou *faux*, 0 ou 1)

On dit d'une grandeur qu'elle est **logique** si elle peut être représentée par une variable **binaire**. Il s'agit donc du cas le plus simple de variable discrète. Ce type de variable est également appelé **« tout ou rien »**. Les systèmes dont les grandeurs d'entrée et de sortie sont logiques sont appelés **systèmes logiques**.

On appelle système logique **combinatoire** un système logique dont la réponse à un instant donné ne dépend que de l'état des entrées à cet instant.

Une lumière pilotée par un interrupteur bistable[1] *est un système logique simple : la lumière sera allumée si l'interrupteur est fermé et éteinte s'il est ouvert. La grandeur d'entrée (position de l'interrupteur) peut être modélisée par une variable binaire inter et il en est de même pour celle de sortie (lumière allumée) lampe.*

Un système logique combinatoire à n entrées et m sorties peut être représenté par une équation logique de la forme

$$\forall i \in [\![1;m]\!],\ s_i(t) = f_i(e_1(t), e_2(t), \dots e_n(t))$$

Le système {interrupteur+lampe} peut être représenté par $lampe(t) = inter(t)$*. Si, la lampe est montée sur un circuit équipé d'un disjoncteur, dont la position sera notée* $disj(t)$*, alors il faudra que les deux interrupteurs soient simultanément fermés pour que la lampe s'allume et donc* $lampe(t) = inter(t)\ ET\ disj(t)$.

Pour décrire une fonction logique, il est possible d'utiliser une représentation graphique sous forme de tableau, appelé **table de vérité**. La valeur des sorties est décrite pour chaque combinaison des entrées.

Décrivons le comportement d'un système de va-et-vient[2] *alimentant une lampe.*

entrées		sorties
inter1	inter2	lampe
0	0	
0	1	
1	0	
1	1	

1.2 Algèbre de Boole et équations logiques

À partir de la table de vérité d'un système, on pourra écrire l'équation logique traduisant le comportement du système. Commençons par donner les bases de l'**algèbre de Boole** ou **calcul booléen**.

En algèbre de Boole, on note la fonction logique ET comme une multiplication $(\cdot)$ et la fonction logique OU comme une somme $(+)$. Par ailleurs, on note par une barre « $\overline{\ }$ » la négation logique d'une variable.

fonction ET		
a	b	$a \cdot b$
0	0	
0	1	
1	0	
1	1	

fonction OU		
a	b	$a + b$
0	0	
0	1	
1	0	
1	1	

fonction NON	
a	$\overline{a}$
0	
1	

On définit la fonction logique décrite par une table de vérité en l'exprimant sous forme canonique comme une somme de produits de grandeurs :

$$\begin{aligned} lampe = \quad & inter1 \cdot inter2 \cdot lampe(inter1, inter2) \\ & + inter1 \cdot \overline{inter2} \cdot lampe(inter1, \overline{inter2}) \\ & + \overline{inter1} \cdot inter2 \cdot lampe(\overline{inter1}, inter2) \\ & + \overline{inter1} \cdot \overline{inter2} \cdot lampe(\overline{inter1}, \overline{inter2}) \end{aligned}$$

1. On distingue les interrupteurs monostables, de type poussoir, qui retrouvent leur position d'origine une fois que l'action de l'utilisateur cesse, et les interrupteurs bistables, de type va-et-vient, qui conservent la position donnée par l'utilisateur lorsque l'action cesse.

2. Un système de « va-et-vient » est une commande de lampe permettant l'allumage et l'extinction depuis deux (ou plus) points distincts équipés d'interrupteurs bistables.

d'où

$$\begin{aligned} lampe = \quad & inter1 \cdot inter2 \cdot 1 \\ & + inter1 \cdot \overline{inter2} \cdot 0 \\ & + \overline{inter1} \cdot inter2 \cdot 0 \\ & + \overline{inter1} \cdot \overline{inter2} \cdot 1 \end{aligned}$$

et finalement

$$lampe = inter1 \cdot inter2 + \overline{inter1} \cdot \overline{inter2}$$

✍ *Écrivez la table de vérité et la fonction logique associée à un système de « va-et-vient » à trois interrupteurs (hall avec trois portes d'accès par exemple). Le montage électrique associé à un tel système est relativement complexe. Il peut s'avérer plus simple de centraliser les informations en un point qui les traite, que ce soit électroniquement ou informatiquement.*

1.3 Propriétés et théorèmes en algèbre de Boole

Les propriétés classiques de la multiplication (fonction ET) et de l'addition (fonction OU) sont toujours vérifiées (commutativité, associativité et distributivité). Notons par ailleurs quelques résultats remarquables :

- $\overline{\overline{a}} = a$
- $a + \overline{a} = 1$
- $a + 0 = a$
- $a + 1 = 1$
- $a \cdot 1 = a$
- $a \cdot 0 = 0$
- $a + a = a$
- $a \cdot a = a$

De plus, les **théorèmes de De Morgan** permettent de transformer des sommes de termes en produits et vice-versa.

- $\overline{a \cdot b} = \overline{a} + \overline{b}$
- $\overline{a + b} = \overline{a} \cdot \overline{b}$

✍ Vérifiez les théorèmes de De Morgan en écrivant les tables de vérité de chaque membre de l'égalité.

1.4 Codage des grandeurs discrètes

De nombreux systèmes comptent parmi leurs entrées ou sorties non seulement des grandeurs logiques mais aussi des grandeurs discrètes non binaires.

En domotique, il est parfois utile de connaître la direction du vent, qui est souvent mesurée de façon discrète : les 360° sont divisés en secteurs angulaires (16, par exemple, pour une rose des vents comportant N, NNE, NE, ENE, E etc.).

Dans ce cas, les grandeurs discrètes sont écrites sur une base binaire afin de transformer une grandeur numérique en plusieurs grandeurs binaires.

1.4.1 Binaire naturel

La façon la plus simple d'obtenir des grandeurs binaires à partir d'une grandeur numérique est de l'écrire en binaire naturel, c'est-à-dire en base 2.

En associant à chaque secteur angulaire un nombre N entre 0 et 15, on obtient les grandeurs binaires e_i définies par $N = (e_3 e_2 e_1 e_0)_{(2)}$, soit $N = e_3 \cdot 2^3 + e_2 \cdot 2^2 + e_1 \cdot 2 + e_0$.

nombre de 0 à 15	e_3	e_2	e_1	e_0
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

1.4.2 Binaire réfléchi ou code Gray

À savoir

Le codage en binaire naturel présente un défaut important pour les applications de mesure : un défaut de lecture sur l'une des pistes peut entraîner la lecture d'une valeur très différente de la valeur réelle (*cf.* figure 1). Ainsi, il est possible d'utiliser un code binaire particulier, dit **binaire réfléchi** ou **code Gray** qui a la caractéristique de n'avoir qu'un seul chiffre binaire qui évolue à chaque changement de la valeur à coder. Un défaut de lecture sur une piste aura donc des conséquences limitées.

L'un des inconvénients du code Gray est l'impossibilité de coder un nombre de valeurs qui ne soit pas une puissance de 2. Un autre est la difficulté supplémentaire pour transformer la valeur lue en binaire naturel pour l'exploitation par un système d'information.

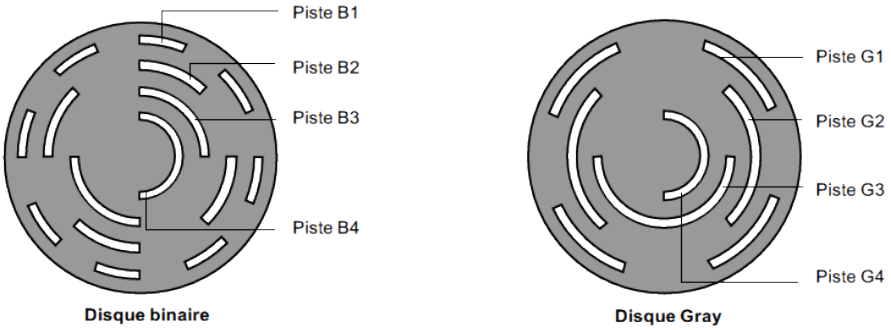


FIGURE *1 – Disques codeurs : en binaire naturel à gauche et en binaire réfléchi (ou code Gray) à droite.*

nombre de 0 à 15	e_3^G	e_2^G	e_1^G	e_0^G
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

1.4.3 Représentation hexadécimale

Les valeurs binaires sont stockées dans la plupart des systèmes d'information dans des « mots » de 8 chiffres. Il est souvent plus simple de donner la représentation hexadécimale de ces « mots » (représentation en base 16) qui ne se compose que de deux chiffres.

Pour déterminer l'écriture hexadécimale d'une expression binaire, on la décompose en paquets de 4 chiffres et chaque paquet est ensuite traduit en un chiffre hexadécimal (allant de 0 à 9 puis de A à F).

base 10	e_3	e_2	e_1	e_0	base 16
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
5	0	1	0	1	5
6	0	1	1	0	6
7	0	1	1	1	7
8	1	0	0	0	8
9	1	0	0	1	9
10	1	0	1	0	A
11	1	0	1	1	B
12	1	1	0	0	C
13	1	1	0	1	D
14	1	1	1	0	E
15	1	1	1	1	F

Pour aller plus loin

Les adresses MAC (ou adresses physiques) des cartes Ethernet sont souvent données en écriture hexadécimale, par exemple : 00 :1B :44 :11 :3A :B7. Les 6 octets sont séparés par des « : ».

✍ Traduisez en binaire naturel l'adresse MAC ci-dessus.

✍ Traduisez en hexadécimal l'adresse MAC suivante écrite en binaire naturel :

11000100 :10000101 :00001000 :11110001 :11001010 :01110100.

2 Systèmes séquentiels

À savoir

Un système **sequentiel** est un système discret dont les sorties sont fonction des entrées à l'instant donné et de **leur histoire**.

Les systèmes de « va-et-vient » à plus de deux interrupteurs sont habituellement remplacés par des systèmes à boutons poussoirs et une mémoire (télérupteurs ou systèmes d'information complexes). Dans ce cas, l'appui sur un bouton poussoir change l'état de la lampe (allumée ou éteinte). L'état de la lampe ne dépend pas uniquement de l'état des entrées qui est toujours identique (tous les interrupteurs ouverts) sauf lors d'un appui.

2.1 Chronogramme

Une façon de représenter le comportement d'un système séquentiel dans un contexte donné consiste à tracer un **chronogramme**, graphique représentant l'état de chacune des grandeurs au cours du temps pour des entrées évoluant de façon arbitraire.

Attention !

Les changements d'un système séquentiel se produisent à des instants donnés par des **événements**, faits instantanés dont l'origine peut être le changement de valeur d'une variable booléenne.

Ainsi, la lampe ne s'allumera pas lorsqu'un bouton poussoir est fermé mais lorsqu'il passe de sa position ouverte à sa position fermée (ou peut-être de sa position fermée à sa position ouverte).

La représentation du comportement des systèmes à événements discrets par des chronogrammes ne permet pas de connaître le comportement du système face à des entrées quelconques mais est un premier outil de description et d'analyse des systèmes séquentiels, pour lesquels **les tables de vérité n'ont pas de sens**.

2.2 Diagramme d'états

À savoir

Un outil plus adapté pour la représentation complète du comportement d'un système séquentiel est le **diagramme d'états**. Il est composé de

états (représentés par des rectangles) qui traduisent les états de fonctionnement (ou d'attente), éventuellement associés à une (ou des) activité(s) ;

transitions (représentés par des flèches entre les états) associées à un **événement** et éventuellement une **condition de garde**, elles représentent la possibilité du système d'évoluer d'un état à un autre et la condition de cette évolution.

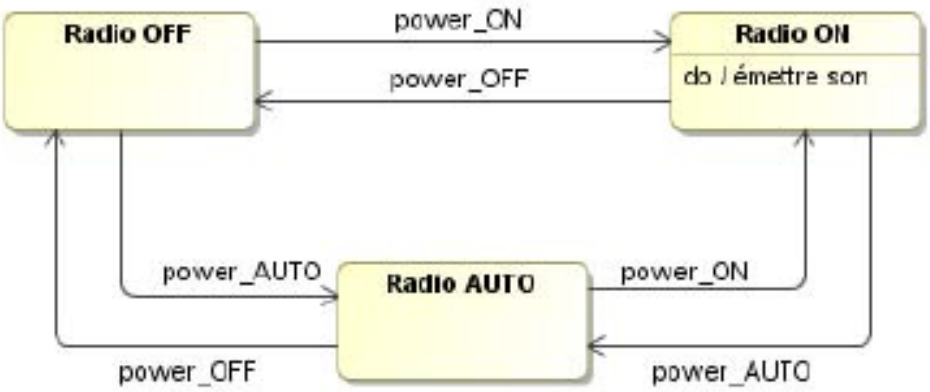


FIGURE 2 – *Diagramme d'états simple d'un radio-réveil*

2.2.1 Transitions

Une transition traduit la possibilité du système de passer d'un état à un autre. Elle est notée par une flèche et ses caractéristiques sont notées de la façon suivante :

événement[garde]/effet

Événements

À savoir

Un **événement** est un **changement de valeur** supposé **instantané** d'une variable observée par le système. Un événement ne peut pas être mémorisé. S'il n'est pas exploité lorsqu'il a lieu, il est perdu.

Quelques précisions supplémentaires

À chaque transition est associé un événement qui permet de la déclencher. Lorsque celui-ci est absent, on considère que l'événement est la fin de de l'activité associée à l'état.

Il existe des événements particuliers donnés par l'horloge du système :

- **when** (*temps courant*=temps programmé) : l'événement se produit lorsque le temps courant est égal à l'instant programmé
- **after** (*durée*) : l'événement se produit après la durée *durée* depuis l'activation de l'état associé à la transition.

Garde

À savoir

Une **garde** est une **condition de franchissement** d'une transition. Si la condition de garde n'est pas vérifiée alors, lorsque l'événement se produit, la transition ne peut pas se déclencher et il n'y aura pas d'évolution d'état.

Une garde s'exprime comme une expression logique qui est évaluée lorsque l'événement associé à la transition se produit. Dans le cas le plus simple, il s'agit simplement d'une variable logique.

Il est possible d'avoir plusieurs transitions émanant d'un même état et associées à un même événement à condition que les conditions de garde soient disjointes (un seul état peut être actif à un instant donné). Il est possible d'écrire la condition de garde else, qui est la négation de la somme de toutes les autres gardes.

Effet

À savoir

Le franchissement d'une transition peut entraîner un **effet**, c'est-à-dire une **action** ou une suite d'actions. Une action peut correspondre, par exemple, au changement de valeur d'une variable. Au cours de la réalisation d'une action, aucun autre événement ne peut être traité.

2.2.2 États

À savoir

Seul un état peut être actif à un instant donné.

Activités

À savoir

Un état peut entraîner une **activité**, qui peut avoir une durée longue et peut être interrompue. On note alors le mot-clé « do/ » suivi de l'activité à l'intérieur du rectangle matérialisant l'état.

Effets à l'activation et à la désactivation

Pour aller plus loin

Si un effet est associé à chaque transition arrivant dans un état (respectivement à toutes les transitions sortantes d'un état), il est possible de simplifier la notation en associant directement l'effet à l'entrée (resp. sortie) de l'état et non aux transitions. On note alors « entry/ » (resp. « exit/ ») suivi de l'effet à l'intérieur du rectangle matérialisant l'état.

Par souci de clarté, on écrit d'abord les effets d'entrée, puis les activités, puis les effets de sortie.

2.2.3 Pseudo-états initial et final(s)

État initial

À savoir

Lors de l'activation du système, il faut savoir dans quel état se trouve le système. Ainsi, il existe un pseudo-état, dit **état initial**, auquel n'est associée aucune activité et qui est le premier état activé lors de la mise en service du système (ou du sous-système). Il n'a **aucune transition entrante**, sa présence est **obligatoire** et il est **unique**.

État(s) final(s)

À savoir

Il traduit la fin de l'activation du système. Il n'a **aucune transition sortante** et est **optionnel**.

2.2.4 *Quelques syntaxes particulières

Il existe des syntaxes cherchant à simplifier l'écriture de diagrammes complexes.

États composites Il est possible de regrouper dans un même état un ensemble de sous-états. Dans ce cas, on dessine le sous-diagramme d'états à l'intérieur de l'état englobant, dit **composite** ou on indique que l'état est un état composite par un symbole représentant deux états et une transition.

Pour aller plus loin

Lorsqu'un état composite est activé, en principe, le pseudo-état initial contenu dans l'état composite est activé. Il est cependant possible de forcer le système à activer le dernier état actif avant la désactivation de l'état composite. Pour ce faire, la transition arrivant dans l'état composite doit avoir pour cible un pseudo-état dit **historique** noté avec un « H ». Si en plus, parmi les sous-états du l'état composite il y a d'autres états composite, on peut préciser que l'on souhaite activer le dernier sous-état actif de la dernière *échelle de profondeur* en utilisant un **pseudo-état historique profond** (*deep history*) noté par « H* ».

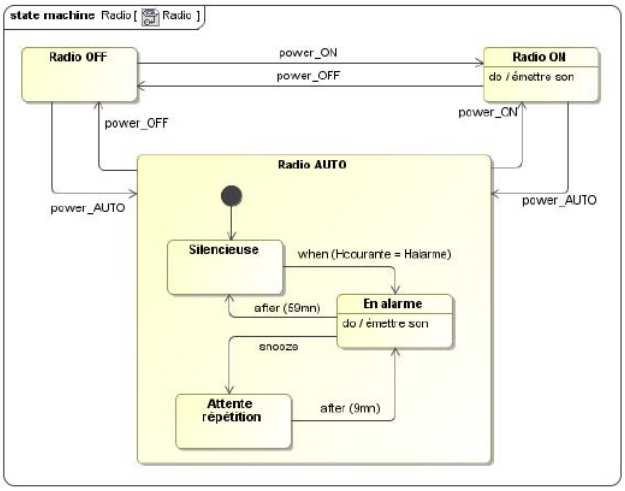


FIGURE 3 – *Diagramme d'états d'un radio-réveil (état composite).*

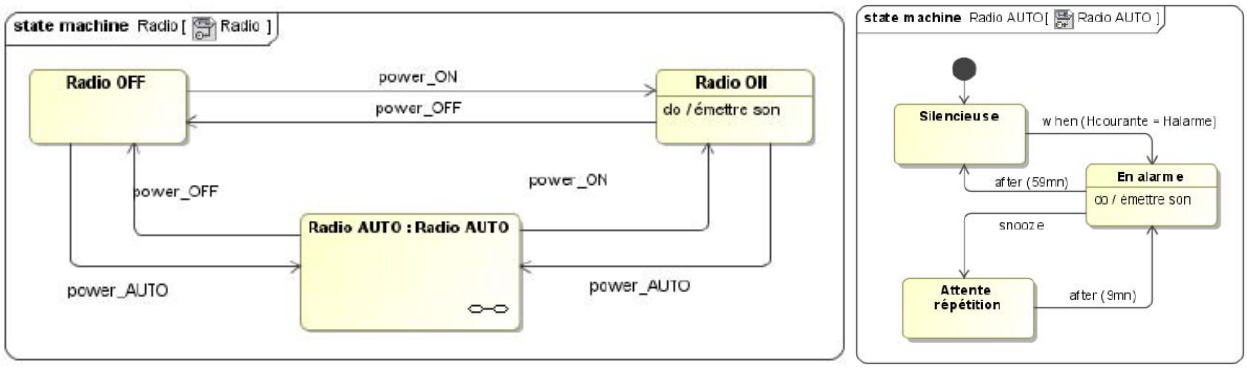


FIGURE 4 – *Diagramme d'états d'un radio-réveil (écriture alternative d'un état composite).*

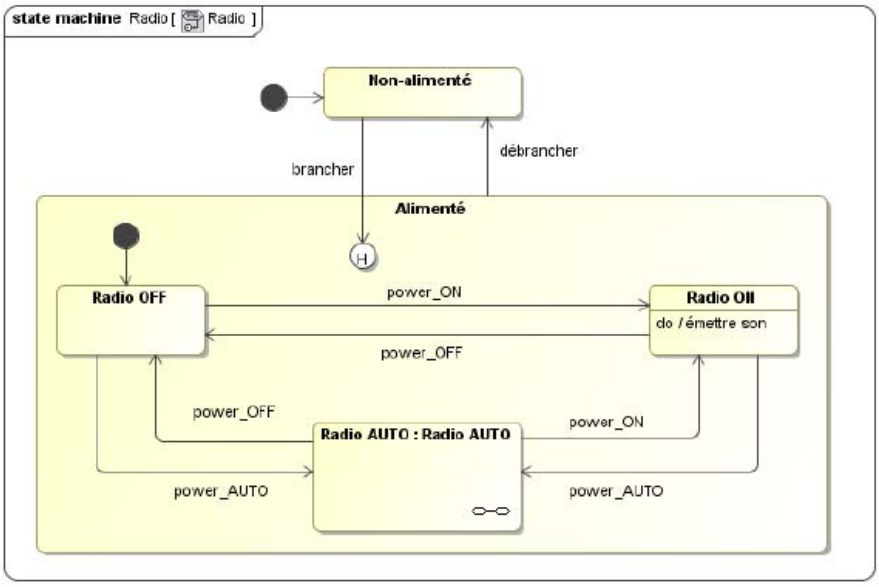


FIGURE 5 – *Diagramme d'états d'un radio-réveil (état composite historique).*

2.2.5 États orthogonaux

Un état composite peut appeler deux diagrammes qui évoluent en parallèle. On dit alors qu'il s'agit d'états **orthogonaux**. Les états orthogonaux sont séparés par un trait discontinu.

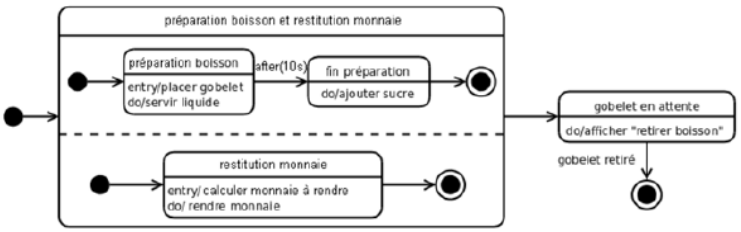


FIGURE 6 – *Diagramme d'états d'un distributeur de boissons (états orthogonaux).*

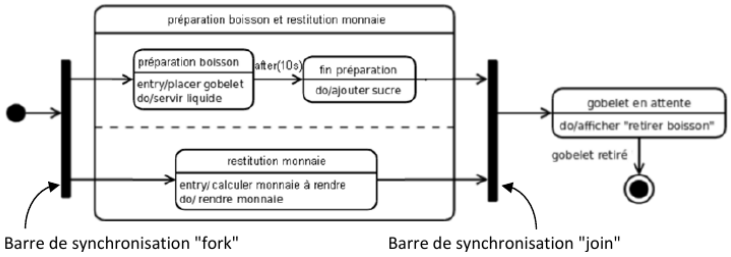


FIGURE 7 – *Diagramme d'états d'un distributeur de boissons (écriture alternative d'états orthogonaux).*

2.3 Diagramme de séquence

À savoir

Le **diagramme de séquence** est un outil de modélisation comportementale permettant de représenter les interactions entre différents composants d'un système ou entre un système et son environnement. Un diagramme de séquence est établi pour **un cas d'utilisation** donné.

Contrairement au diagramme d'états, qui met l'accent sur le comportement d'un système donné dans son ensemble, le diagramme de séquence a pour objectif principal la représentation des interactions en prenant en compte la chronologie des événements.

2.3.1 Lignes de vie et messages

Un diagramme de séquence est principalement composé de

- une **ligne de vie** par acteur : il s'agit d'une ligne verticale représentant l'activité d'un acteur. Elle traduit la chronologie des événements et se lit du haut vers le bas.
- des **messages** reliant les différentes lignes de vie et représentant les interactions entre les acteurs. Un message est **unidirectionnel** (il part d'une ligne de vie et arrive sur une autre). Il en existe trois types :

  synchrone : l'émetteur attend une réponse et son activité est bloquée jusqu'à l'arrivée de celle-ci. Il est représenté par une flèche pleine et un trait plein.

  réponse à un message synchrone : représentée par une flèche évidée et un trait discontinu.

  asynchrone : l'émetteur n'attend pas de réponse et son activité n'est donc pas bloquée. Il est représenté par une flèche évidée et un trait plein.

  réflexif : l'émetteur et le récepteur sont le même acteur. Il est représenté par une flèche pleine et un trait plein qui boucle sur la ligne de vie de l'acteur concerné.

L'activité d'un acteur se traduit par une bande verticale sur la ligne de vie. Sa représentation est optionnelle mais permet de mieux visualiser les acteurs en activité à un instant donné.

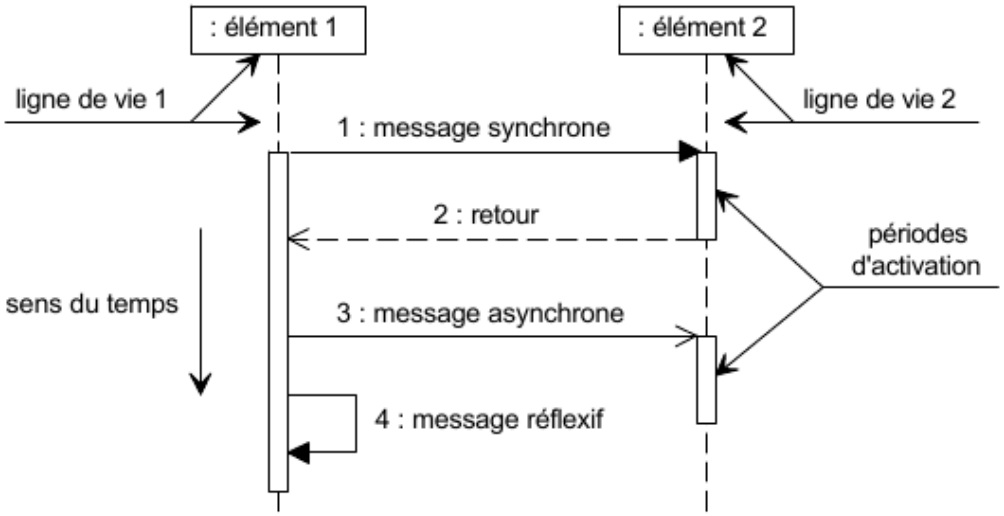


FIGURE *8 – Principe d'un diagramme de séquence.*

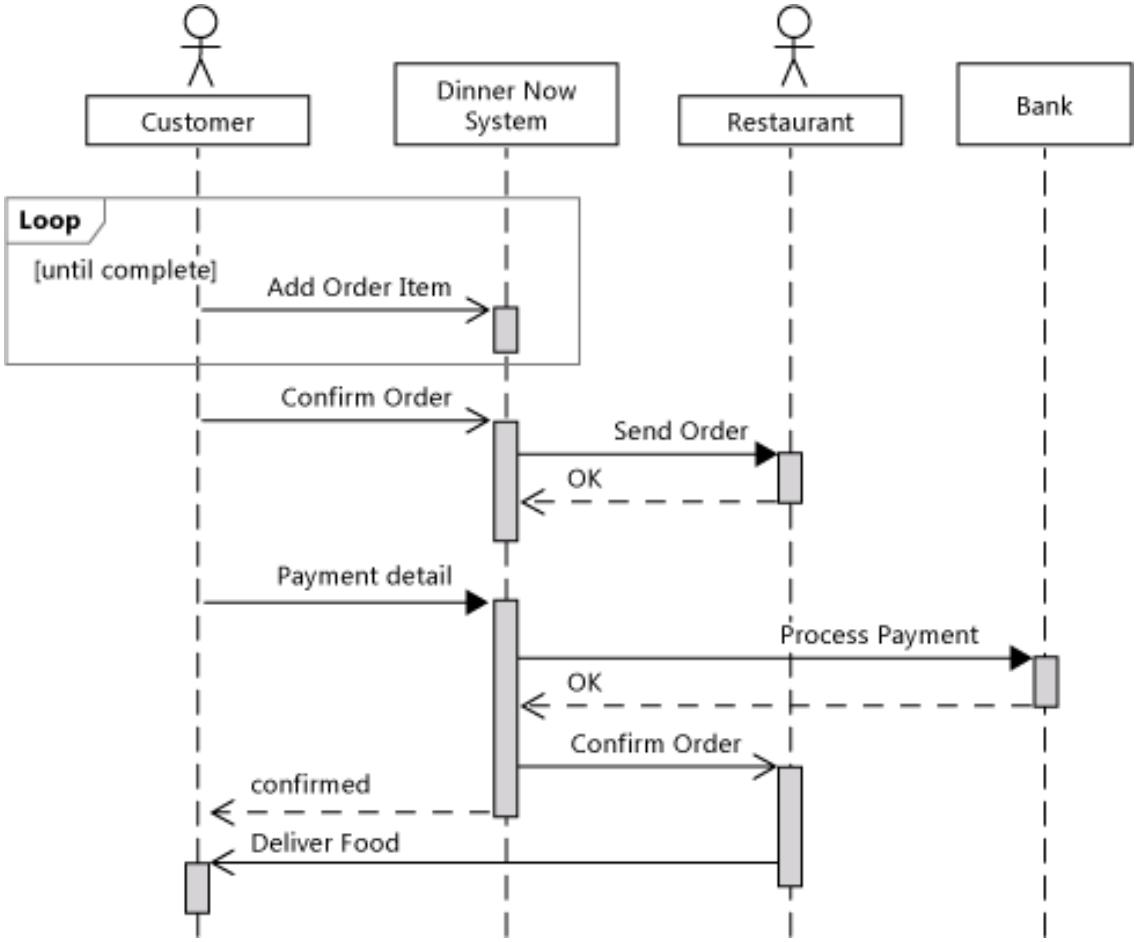


FIGURE *9 – Diagramme de séquence du système de commande de repas en ligne « Dinner Now ».*

2.3.2 Structures algorithmiques de base

Il est possible de représenter des comportements plus complexes qu'une séquence simple d'échange de messages entre les acteurs via les structures algorithmiques de base déjà étudiées en informatique. Un rectangle identifie la partie du diagramme de séquence concerné par la structure algorithmique et un mot clé spécifie sa nature.

loop (boucle) : le fragment est répété le nombre de fois indiqué ou tant que la condition donnée est vérifiée

opt (optionnel) : le fragment n'est exécuté que si la condition est vraie

alt (alternative) : plusieurs fragments sont proposés. Seul celui dont la condition est vraie est exécuté.
par (parallèle) : plusieurs fragments s'exécutent en parallèle.
break (interruption) : l'exécution du fragment s'interrompt lorsque la condition est vraie.

2.3.3 Exemples de diagrammes de séquence d'un radio-réveil

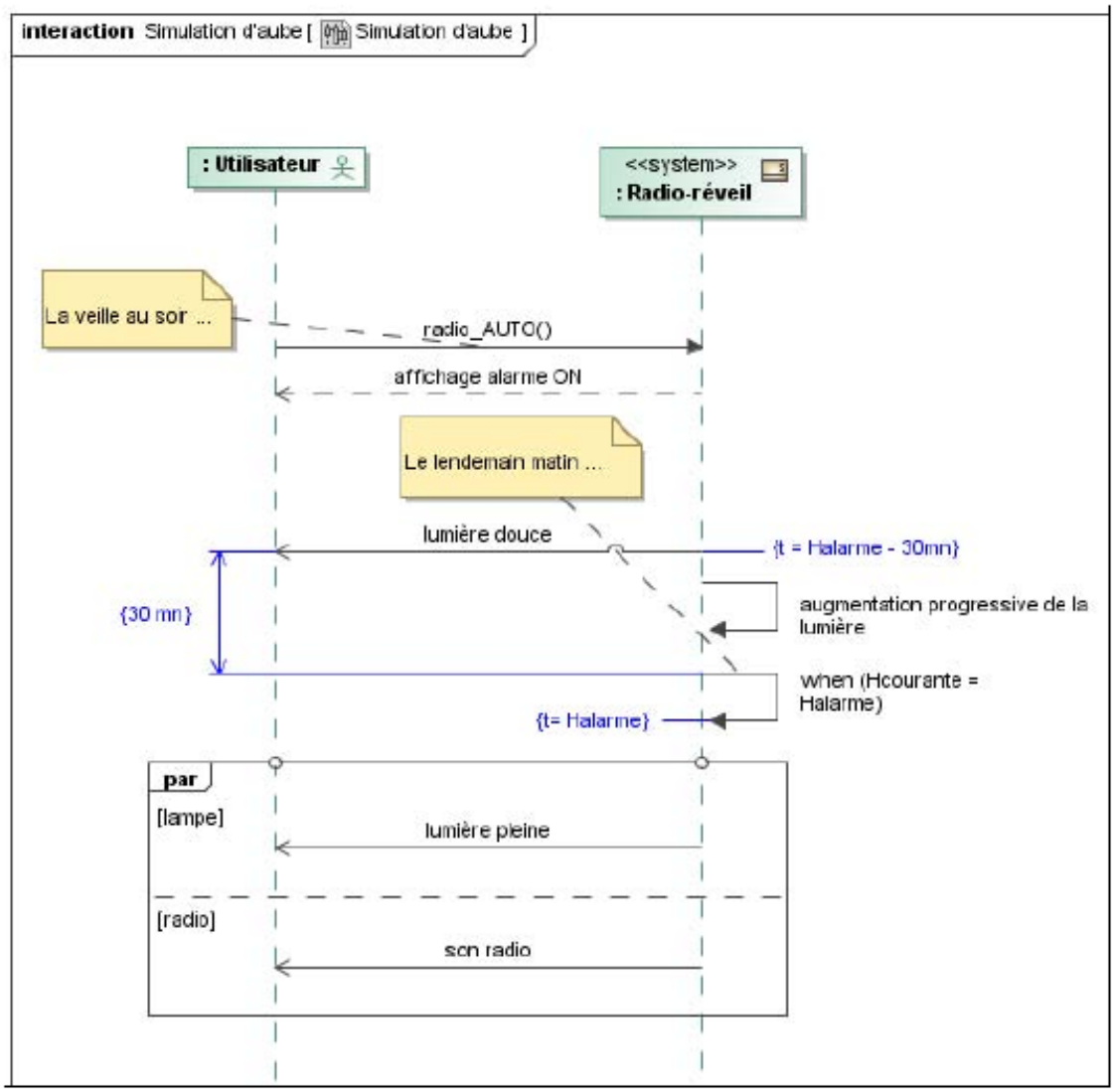


FIGURE *10 – Diagramme de séquence d'un radio-réveil avec des contraintes temporelles.*

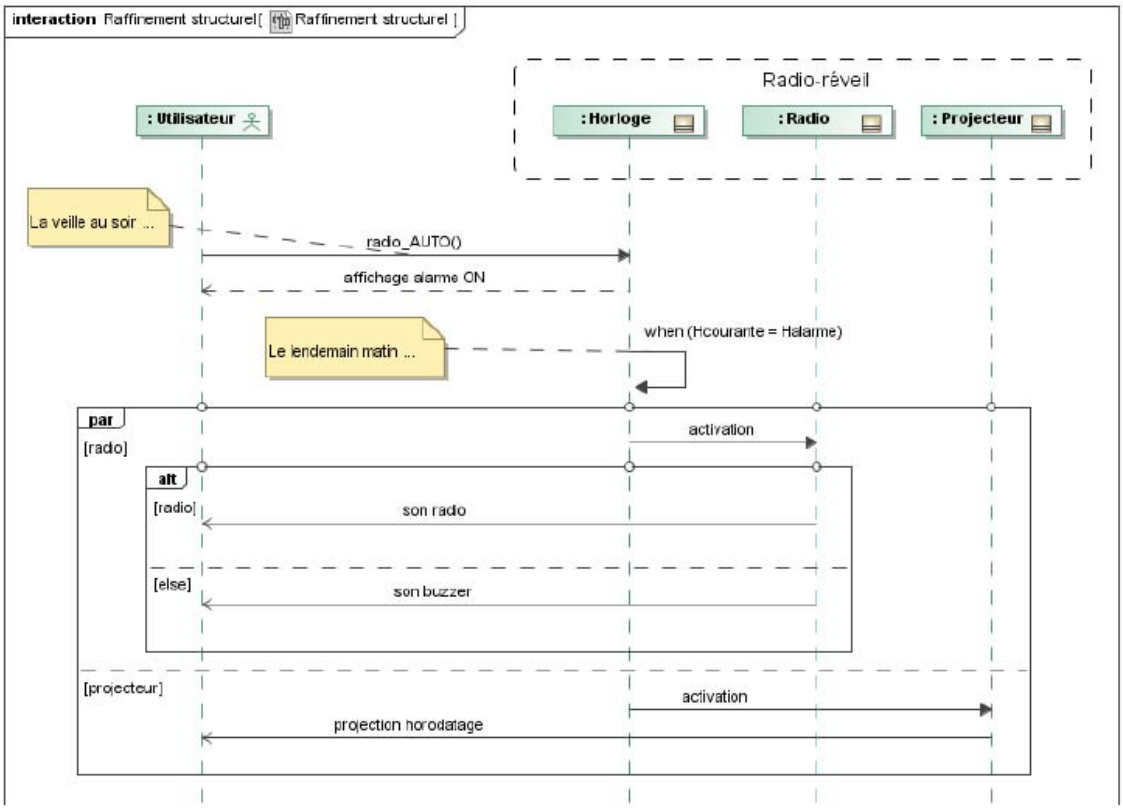


FIGURE *11 – Diagramme de séquence d'un radio-réveil avec une ligne de vie par composant principal du système.*