Les séquenceurs : une restriction des systèmes formels (SF)

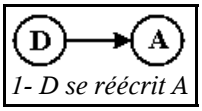
Les séquenceurs (linéaires)

Les séquenceurs : des SF très simples, caractérisés par deux contraintes

Pour illustrer cette partie du cours d'IA sur les systèmes formels, nous commençons par les brider un peu, en leur ajoutant deux contraintes, ainsi nous obtenons les séquenceurs, qui nous permettent de débuter avec des outils très simples.

Contrainte 1 :

D'abord, les séquenceurs n'utilisent que des règles de la forme :
SymboleA→ SymboleB,
Dans cette définition, la règle se lit comme le remplacement d'un *SymboleA* par un *SymboleB*, ou encore la réécriture du *SymboleA* en un *SymboleB*.



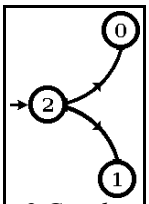
1- D se réécrit A

Convention de notation :

Disons que le *SymboleA*, partie gauche de la règle est la tête de règle ; et que le *SymboleB*, partie droite en est la queue.

Contrainte 2 :

Ensuite, dans un ensemble de telles règles de réécriture, les parties gauches doivent toutes être différentes : il ne peut y avoir plusieurs règles possédant la même tête de règle.



2 Graphe en patte d'oie

Exemple 1

Donc cette spécification interdit d'avoir deux règles comme celles-ci :
TêteDeRègle→Symbole1,
TêteDeRègle→SymboleA,

Exemple 2

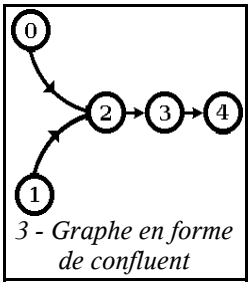
Dans un séquenceur qui comprend au moins les trois symboles *0, 1,* et *2*, il est interdit d'utiliser conjointement les deux règles suivantes :
2→0, // En effet, les deux règles utilisent la même tête de règle : *2.*
2→1, // Au sein d'un séquenceur, si une de ces deux règles est utilisé, elle exclut forcément l'autre.

Conséquence de ces deux contraintes

Quand le système est dans un état donné, au maximum, il existe une seule règle qui puisse s'appliquer.

Donc ces spécifications interdisent que le graphe du séquenceur présente une patte d'oie (i.e. une structure en delta). En fait, ainsi, on ne peut obtenir qu'une séquence dite linéaire, qu'un parcours en ligne droite (dans le sens où le graphe des transitions se déroule le long d'une ligne).

Le séquenceur confluent[1]



3 - Graphe en forme de confluent

Par contre, selon la définition initiale, un SF peut posséder plusieurs états de départ, et comme rien n'interdit que plusieurs règles concluent sur le même état, dans le graphe d'un séquenceur, nous pouvons rencontrer la structure de confluent.

Par exemple, on peut rencontrer un séquenceur structuré comme celui de gauche. Il possède deux règles concluant sur le même état *2* : *0→2*, et *1→2*. À l'image du confluent de deux rivières, c'est un séquenceur avec deux points de départ qui confluent conjointement vers une même séquence finale.

Remarque : Avec un tel séquenceur, quelque soit le point de départ choisi (*0* ou *1*), le parcours obtenu est bien séquentiel, linéaire. En partant de *0* on obtient : *0→2→3→4*. En partant de *1* on obtient : *1→2→3→4*.

1 À l'image du confluent de deux rivières : un séquenceur avec deux points de départ qui confluent conjointement vers une même séquence finale.

Finalement nous obtenons un séquenceur

Un SF très simple, caractérisé par un unique état

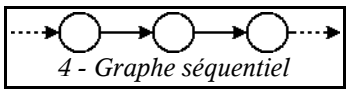
D'abord, dans ce SF très simple, l'état du séquenceur obtenu est caractérisé uniquement par la donnée d'un simple symbole.

Un SF muni d'un procès très simple

Le SF obtenu est muni d'un procès très simple. Son état est caractérisé par un unique symbole. Son procès se réduit à l'application d'une nouvelle règle, qui réécrit ce symbole en un nouveau symbole (qui, en général, est différent).

Finalement, l'exécution de plusieurs procès fournissent un graphe séquentiel

Finalement, après avoir appliqué ces contraintes 1 et 2, nous obtenons un SF présentant cette caractéristique sur laquelle nous focaliserons ultérieurement : de façon générale nous obtenons un graphe séquentiel dont la forme peut ressembler au dessin ci-contre.



4 - Graphe séquentiel

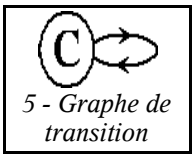
Dans la suite de ce polycopié, nous détaillerons plus précisément certaines formes remarquables de ces séquenceurs.

Étude d'exemples remarquables de séquenceurs

1) Un séquenceur simplissime : La constante permanente[2]

Il est composé d'une règle unique :

$C \rightarrow C$ // Le symbole *C* se réécrit lui-même
L'état de départ est *C*.
L'état d'arrivée n'est pas fournit.
Il n'y a pas de condition d'arrêt.
Ainsi on obtient la séquence infinie : $C \rightarrow C \rightarrow C \rightarrow C \rightarrow C \rightarrow C \rightarrow C \rightarrow C \rightarrow C...$



5 - Graphe de transition

Analyse et commentaires

Puisque le système ne comporte qu'une unique règle, il ne produit qu'un procès, i.e. un seul changement de son état, il remplace *C* par *C*.

Interprétation de ce séquenceur

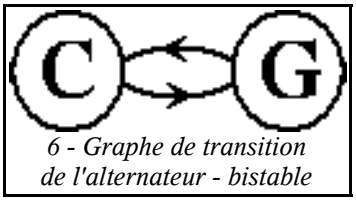
Les SF sont des outils mathématiques abstraits (ab-trahere : qui sont tirés de la réalité, vers le haut). Donc, par essence, ils sont exempts de toute connotation concrète, mais, dans une démarche pragmatique, on peut les plonger dans différents contextes afin de leur fournir des interprétations.

Si on plonge ce séquenceur dans l'univers de la vie[3] et si on interprète son état comme l'existence d'une cellule *C*, le déroulement infini de la séquence $C \rightarrow C \rightarrow C \rightarrow C$ ... peut être interprétée comme la présence éternelle d'une cellule qui se reproduit toujours identique à elle-même.

2) Un séquenceur très simple : le bistable[4], l'alternateur (oscillateur – clignotant)

Il est composé de deux règles :

$C \rightarrow G$ // Le symbole *C* se réécrit *G*
$C \rightarrow C$ // Le symbole *G* se réécrit *C*
L'état de départ est *C*.
L'état d'arrivée n'est pas précisé.
Il n'y a pas de condition d'arrêt.
Ainsi on obtient la séquence infinie : $C \rightarrow G \rightarrow C \rightarrow G \rightarrow C \rightarrow G...$



6 - Graphe de transition de l'alternateur - bistable

Analyse et commentaires

Puisqu'il comporte deux règles, le système produit deux procès, i.e. deux changements de son état, il remplace *C* par *G*, et *G* par *C*.

Interprétation de ce séquenceur

Les SF sont par nature exempts de toute connotation concrète, mais on peut les plonger dans différents contextes et leur donner différentes interprétations.

Si on plonge ce séquenceur dans l'univers de la vie et si on interprète son état comme l'existence d'une cellule *C*, la séquence infinie $C \rightarrow G \rightarrow C \rightarrow G...$ peut être interprétée comme une cellule qui produit une graine, et une graine qui produit une cellule. Et ce cycle continue éternellement !

Si on plonge ce séquenceur dans l'univers de l'électronique et si on interprète son état comme un niveau électrique, la séquence infinie $C \rightarrow G \rightarrow C \rightarrow G...$ peut être interprétée comme un multi-vibrateur astable (instable) qui oscille entre niveau haut et niveau bas. À haute fréquence on obtient un oscillateur numérique ; et à basse fréquence, le clignotant d'un véhicule automobile.

---

2 Permanent : de per-manere – rester au-travers : ce qui demeure, reste au travers du temps et des siècles.

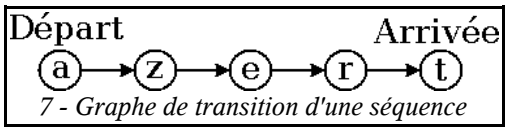
3 La suite du cours expliquera comment l'introduction de la Vie Artificielle dans un système transforme les procès en actions.

4 Selon des critères d'électronique industrielle ce montage est *a-stable (instable)*. Si on le décrit au moyen d'un séquenceur, il apparaît plutôt comme *bi-stable*.

3) Le séquenceur : un séquenceur qui déroule une séquence

Il est composé de quatre règles :

a → z // Le symbole *a* se réécrit *z*
z → e // Le symbole *z* se réécrit *e*
e → r // Le symbole *e* se réécrit *r*
r → t // Le symbole *r* se réécrit *t*
L'état de départ est *a*.
L'état d'arrivée est *t*.
Ainsi on obtient la séquence finie : *a → z → e → r → t*.



7 - Graphe de transition d'une séquence

Analyse et commentaires

Puisqu'il comporte quatre règles, le système produit quatre procès, quatre changements de son état, qui se déroulent séquentiellement.

Interprétation de ce séquenceur

Si on plonge ce séquenceur dans l'espace et si on interprète ses états comme des lieux, le déroulement de la séquence *a → z → e → r → t* peut être interprété comme le trajet d'un voyage qu'on effectue par étapes, jusqu'à un lieu *t*.

Si on plonge ce séquenceur dans le temps et si on interprète ses états comme des dates, le déroulement de la séquence *a → z → e → r → t* peut être interprété comme une séquence temporelle qu'on déroule jusqu'à une date *t*.

4) La boucle : un séquenceur qui déroule un cycle

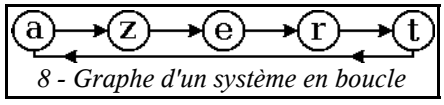
Elle est composée de quatre+1 règles

Aux quatre règles du système précédent on rajoute celle-ci :
t → a // Le symbole *t* se réécrit *a*
L'état de départ est *a*.
L'état d'arrivée n'est pas précisé.
Il n'y a pas de condition d'arrêt.

Analyse et commentaires

Ainsi, avec ce total de cinq règles, on obtient un système qui boucle indéfiniment sur la séquence :

a → z → e → r → t → a...



8 - Graphe d'un système en boucle

Interprétation de ce séquenceur

Si on plonge ce séquenceur dans l'espace et si on interprète ses états comme des lieux, le déroulement de la séquence *a → z → e → r → t → a* peut être interprété comme une migration périodique le long du même parcours.

Si on plonge ce séquenceur dans le temps et si on interprète ses états comme des dates, le déroulement de la séquence *a → z → e → r → t → a* peut être interprété comme le comptage du temps effectué par un compteur, un séquenceur qui égrène les intervalles de temps.

5) Une séquence numérique décroissante linéairement :

Elle est composée de 4 règles

4 → 3 // Le symbole *4* se réécrit *3*
3 → 2 // Le symbole *3* se réécrit *2*
2 → 1 // Le symbole *2* se réécrit *1*
1 → 0 // Le symbole *1* se réécrit *0*
L'état de départ est *4*.
L'état d'arrivée est *0*.
Ainsi on obtient la séquence finie : *4 → 3 → 2 → 1 → 0*. // Le symbole *4* se réécrit *0*

Analyse et commentaires

Ainsi, avec ce total de quatre règles, on obtient une séquence décroissante vers le but 0, qui est atteint :
4→3→2→1→0 ...

Interprétation de ce séquenceur

Si on plonge ce séquenceur dans le domaine de la gestion et si on interprète ses états comme une quantification d'un stock, le déroulement de la séquence *4→3→2→1→0* peut être interprété comme une décroissance régulière du stock au fil du temps.

6) Une séquence numérique croissante exponentiellement :

Elle est composée de 4 règles

4→6 // Le symbole *4* se réécrit *6*
6→9 // Le symbole *6* se réécrit *9*
9→13 // Le symbole *9* se réécrit *13*
13→19 // Le symbole *13* se réécrit *19*
L'état de départ est *4*.
L'état d'arrivée est *19*.
Ainsi on obtient la séquence finie : *4→6→9→13→19*. // Le symbole *4* se réécrit *19*

Analyse et commentaires

Ainsi, avec ce total de quatre règles, on obtient une séquence croissante exponentiellement, qui est atteint :
4→6→9→13→19 …

La formule de récurrence est : $f_{n+1} = 1,5 * f_n$. Attention, le calcul est effectué sur des entiers. Par exemple 9/2 = 4.

Interprétation de ce séquenceur

Si on plonge ce séquenceur dans le domaine de la gestion d'une population, et si on interprète ses états comme une quantification, le déroulement de la séquence *4→6→9→13→19* peut être interprété comme la croissance exponentielle du groupe au fil du temps.

Le problème de l'implémentation électronique d'un séquenceur

Note à l'intention des étudiants

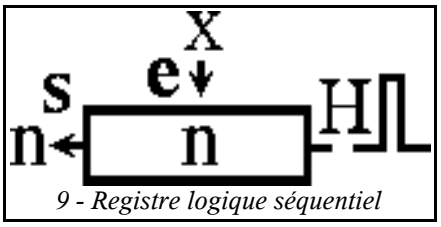
Ce chapitre constitue un jalon que nous posons afin de faciliter la suite du cours. En effet, il explique comment implémenter un séquenceur au moyen d'une table et d'un registre mémoire. Dans la suite du cours, nous retrouverons cette structure et nous lui rajouterons un ruban afin de fabriquer un automate à états finis, qui constitue une étape centrale de l'apprentissage de l'IA.

Implémenter un séquenceur au moyen d'un registre et d'une table

Dans un premier temps nous avons travaillé avec des SF, qui sont des êtres mathématiques. Maintenant, nous changeons de sujet et nous posons la question de l'implémentation électronique d'un séquenceur. Pour cela nous allons utiliser un registre et une table, i.e. un tableau à une entrée.

Étude de la partie 1 de l'implémentation logique d'un séquenceur : le registre mémoire

Commençons par en décrire la pièce maîtresse : le registre mémoire. Déjà, nous pouvons dire que c'est un montage logique séquentiel, qui, de façon générale, mémorise une donnée binaire (ici l'état du système) aussi longtemps que voulu. Cette mémorisation se fait tout simplement en enregistrant le symbole correspondant dans un registre mémoire.



9 - Registre logique séquentiel

Pour la clarté de l'exposé, supposons que ce symbole soit la lettre *n*. Dans le dessin ci-contre, au milieu du rectangle représentant le registre, on observe facilement la lettre *n*. Avant le chargement suivant, le registre contient ce symbole, qui est publié sur la sortie *s* du registre. D'autre part, la donnée *x* présente sur l'entrée *e* du registre ne peut pas déjà entrer (car elle n'y a pas été autorisée).

Par contre, au moment du top d'horloge *H*, cette donnée *x* entre dans le registre, écrase la précédente information (la lettre *n*), et se retrouve publiée en sortie.

Conclusion

Nous avons donc décrit comment est implémentée la partie mémoire d'un séquenceur, maintenant il nous reste à expliciter comment les données sont fournies à son entrée.

Étude de la partie 2 de l'implémentation logique d'un séquenceur : la table

Maintenant nous allons décrire comment implémenter un séquenceur électronique, en ajoutant au registre un dictionnaire. Il est implémenté au moyen d'une mémoire (RAM ou ROM) de N mots, d'une table, d'un tableau des N cases et à une entrée utilisé pour traduire *n* clés en *n* valeurs[5].

Poser un jalon pour le futur

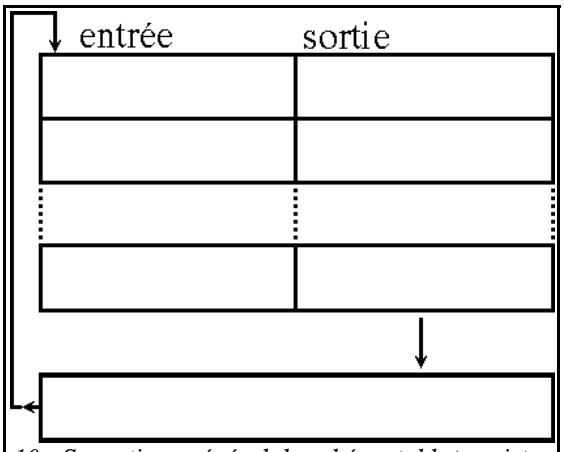
Nous décrivons cette implémentation dans le but de poser un jalon pour plus tard, car la structure obtenue nous servira dans la suite du cours. D'abord pour construire une sous-classe de séquenceurs, et ensuite, parce que, un peu prolongée, elle nous amènera au schéma de l'automate fini.

5 Dans certains cas n << N, le nombre de clés est très petit devant le nombre d'adresses, donc la table est creuse : une grande partie de ses adresses n'est pas utilisée. Ainsi tout le contenu correspondant, qui n'est pas adressé, est inemployé. Voir plus loin le cas typique de la *croissance exponentielle*.

Description du fonctionnement du schéma de logique séquentielle : table+registre

Dans le schéma ci-contre, le registre est dessiné en bas, et la table en haut. Ses entrées sont situées à gauche et ses sorties à droite. Notez que la sortie du registre remonte tout en haut pour adresser l'entrée de la table. Finalement, remarquez que la sortie de la table est présentée en entrée du registre. Ainsi la boucle est bouclée !

Chaque intervalle élémentaire de temps fournit une impulsion électrique de chargement au registre. Ce dernier mémorise en son sein la donnée présentée sur son entrée. Elle écrase l'information précédente, et se retrouve publiée en sortie. Elle remonte pour adresser la table, qui fournit sur sa sortie l'étape suivante. Cette information, ce symbole est présenté en entrée du registre, mais attend la prochaine impulsion de chargement pour y entrer.

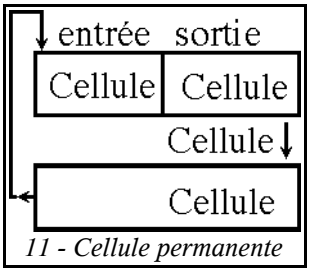


10 - Synoptique général du schéma table+registre

Transposer les 6 modèles de séquenceurs précédents vers le schéma : table+registre

Au moyen des SF, nous avons décrits six phénomènes simples : la constante permanente, le bistable, le séquenceur, la boucle, la décroissance linéaire et la croissance exponentielle. Maintenant étudions comment représenter autrement ces systèmes, comment câbler un schéma de logique séquentielle au moyen d'un registre mémoire et d'une table à une entrée(i.e. d'un dictionnaire à une clé).

On notera que le nombre de clé du dictionnaire est donnée par le nombre de règles du SF à transposer.



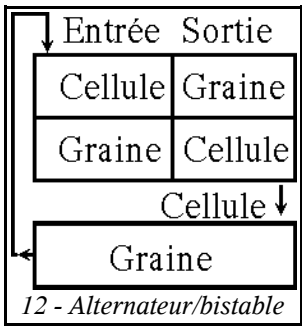
11 - Cellule permanente

Transposer la cellule permanente vers le schéma : table+registre

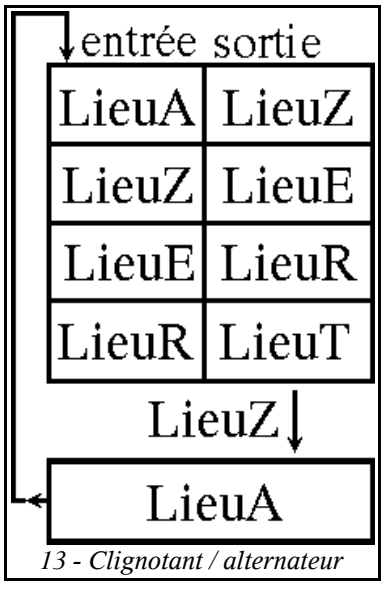
Utilisons cette méthode du schéma logique combinatoire pour transposer la cellule permanente. On utilise une règle unique, qui dit : *Cellule→Cellule* ; ou encore *C→C* ; i.e. que le symbole *C* (pour Cellule) se réécrit lui-même. Elle peut être traduite dans table à une entrée, i.e. un dictionnaire de une clé *C* et dont le contenu, la valeur est aussi *C* (figure ci-contre à gauche).

Transposer l'alternateur vers le schéma : table+registre

Utilisons cette méthode du schéma logique combinatoire pour transposer l'alternateur. Le jeu de deux règles, qui dit *Cellule→Graine,* et *Graine→Cellule*, peut être traduit dans une table de deux adresses : Un C(pour Cellule) est traduit par G (pour Graine), et réciproquement G par C (figure ci-contre à droite).



12 - Alternateur/bistable

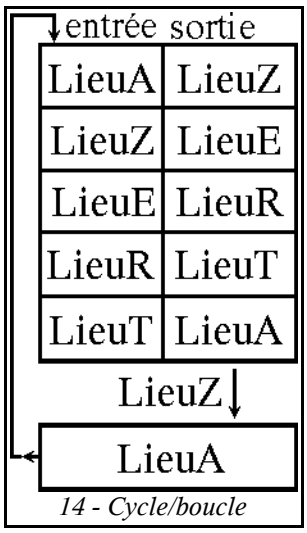


13 - Clignotant / alternateur

Transposer le séquenceur vers le schéma : table+registre

Au moyen de cette méthode, le jeu de quatre règles :
LieuA→LieuZ,
LieuZ→LieuE,
LieuE→LieuR,
LieuR→LieuT,
peut être traduit dans une table de quatre adresses :
LieuA est traduit par *LieuZ, LieuZ* par *LieuE, LieuE* par *LieuR, LieuR* par *LieuT.*

Transposer le *cycle* vers le schéma : table+registre



14 - Cycle/boucle

Le jeu de cinq règles :
LieuA→*LieuZ,*
LieuZ→*LieuE,*
LieuE→*LieuR,*
LieuR→*LieuT*,
LieuT→*LieuA*,
peut être traduit dans une table de cinq adresses :
LieuA est traduit par *LieuZ, LieuZ* par *LieuE, LieuE* par *LieuR, LieuR* par *LieuT, et LieuT* ramène à *LieuA.*

Transposer la *décroissance linéaire* vers le schéma : table+registre

Le jeu de quatre règles :
4→3 // Le symbole *4* se réécrit *3*
3→2 // Le symbole *3* se réécrit *2*
2→1 // Le symbole *2* se réécrit *1*
1→0 // Le symbole *1* se réécrit *0*
peut être traduit dans une table de quatre adresses :
4 *est traduit par* 3, 3 *par* 2, 2 *par* 1, et 1 *par* 0.

Transposer la *croissance exponentielle* vers le schéma : table+registre

Le jeu de quatre règles :
4→6 // Le symbole *4* se réécrit *6*
6→9 // Le symbole *6* se réécrit *9*
9→13 // Le symbole *9* se réécrit *13*
13→*19* // Le symbole *13* se réécrit *19*
peut être traduit dans une table de quatre adresses :
4 *est traduit par* 6, 6 *par* 9, 9 *par* 13, et 13 *par* 19.

Une transition depuis les SF vers un procès embarqué

Un SF n'est pas incarné dans la réalité quotidienne : il est abstrait comme les mathématiques et permet seulement un processus de traitement de l'information immatériel : à chaque étape, l'application d'une règle de réécriture fait transiter l'état du système.

Conclusion :

Ce faisant, tout doucement, nous amorçons un passage de l'abstrait vers le concret ; nous transitons des mathématiques (les SF), vers un procès embarqué du traitement de l'information, et à terme vers la simulation du raisonnement humain.

Un registre et une fonction analytique pour implémenter un séquenceur

Introduction

Dans ce polycopié, nous avons travaillé avec des SF, et nous avons déjà évoqué la question de leur implémentation électronique.

D'un côté, nous avons montré comment utiliser un registre séquentiel et une table pour les câbler, et nous avons pointé le fait que cette circuiterie annonçait la structure d'automate fini.

D'un autre côté, nous allons maintenant essayer de remplacer cette table par une fonction analytique fournie par un circuit logique combinatoire.

D'un côté, quand c'est possible

Ainsi, quand c'est possible, partis des séquenceurs implémentés au moyen des SF, nous débouchons sur les séquenceurs électroniques, implémentés au moyen de fonctions logiques (combinatoires et séquentielles).

De plus il faut noter que cette démarche de *traitement embarqué de l'information* pose un jalon que nous réutiliserons vers la fin du cours, pour câbler des fonctions logiques au sein des agents. Cette fonctionnalité leur permettra de traiter leurs perceptions, afin qu'ils puissent réfléchir et prendre les décisions nécessaires pour agir conséquemment, causalement.

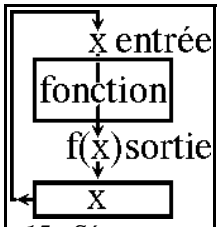
De l'autre côté, quand c'est impossible

Quand ce n'est pas possible, nous restons avec des séquenceurs implémentés au moyen d'une table.

Définir en compréhension des règles de transition d'un séquenceur régulier et abstrait

Dans le paragraphe *Utiliser une table pour implémenter un séquenceur électronique*, les circuiteries que nous avons décrites, passaient à l'état suivant au moyen d'une table (définie en extension). Mais maintenant, pour générer ces séries régulières, dans le schéma de câblage du séquenceur précédent, nous remplaçons la table/mémoire[6] par une fonction logique combinatoire, qui est définie en compréhension, i. e. par une formule analytique. Ainsi, des variables apparaissent sur nos illustrations.

Dans le dessin ci-contre, la fonction est représentée par le rectangle du haut ; le registre par celui du bas. L'entrée de ce dernier se situe sur son sommet, et sa sortie sur le côté gauche. Notez que la sortie *x* du registre remonte tout en haut pour adresser l'entrée de la fonction. Finalement, remarquez que la sortie *f(x)* de cette dernière est présentée en entrée du registre. Ainsi la boucle est bouclée !



15 - Séquenceur à fonction câblée

Chaque intervalle élémentaire de temps fournit au registre une impulsion électrique de chargement. À cet instant, la donnée présente sur son entrée est mémorisée en son sein. Elle écrase la donnée précédente, et se retrouve publiée en sortie. Elle remonte pour adresser la fonction, qui fournit sur sa sortie l'étape suivante. Cette information *f(x)*, est présentée en entrée du registre, mais attend la prochaine impulsion de chargement pour y entrer.

Essayer de transposer les 6 modèles précédents vers le schéma : fonction+registre

Au moyen des SF, nous avons décrits six phénomènes simples. Maintenant étudions comment représenter autrement ces systèmes, comment câbler un schéma de logique séquentielle au moyen d'un registre mémoire et d'une fonction analytique calculée au moyen de circuits logiques combinatoires.

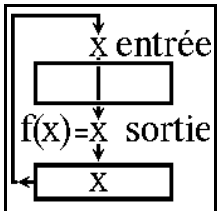
Nous remplaçons la table à une entrée par une fonction numérique plus mathématique et plus abstraite. D'un côté ceci nous permet de diminuer la complexité du câblage, mais de l'autre, nous introduisons en retour plus d'abstraction. C'est pourquoi, nous sommes amenés à utiliser une représentation encore plus abstraite : des nombres binaires.

Dans la vie quotidienne, notre travail consistera à utiliser les SF pour traiter les problèmes de nos client. En conséquence, quand nous leurs restituerons le résultat de notre traitement, nous devrons le rendre concret en l'habillant, le typant et le plongeant dans le contexte adéquat.

---

6 Remplacer une table par une fonction câblée induit une simplification. Dans la suite du cours nous verrons que quand la taille du vecteur d'entrée de la mémoire augmente, sa combinatoire, i.e. son nombre de cases d'adresse, devient explosive. Dans ce cas la seule solution possible est de la remplacer par une fonction câblée ou par un programme informatique.

Transposer la cellule permanente vers le schéma : fonction+registre



16 - Cellule permanente

Utilisons cette méthode du schéma logique combinatoire pour transposer la cellule permanente.

Dans un premier temps, le symbole C (pour cellule) peut se traduire, s'abstraire par exemple par le niveau logique 1. Ainsi la règle unique, qui disait $C \rightarrow C$; i.e. que le symbole *C* se réécrit lui-même, maintenant s'exprime ainsi en logique binaire : $1 \rightarrow 1$.

En termes de fonction mathématique cela correspond à la fonction identité : f(x) = x;

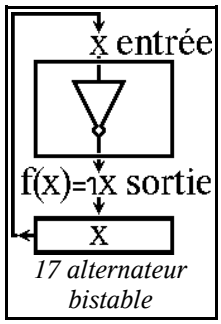
Dans un second temps la règle unique $1 \rightarrow 1$ peut être traduite dans un schéma électrique par une fonction identité, i. e. par un court-circuit (figure ci-contre à gauche).

Transposer l'alternateur vers le schéma : fonction+registre

Utilisons cette méthode du schéma logique combinatoire pour transposer l'alternateur.

Dans un premier temps, le symbole C peut se traduire, s'abstraire par exemple par le niveau logique 1 ; et le symbole G par le niveau logique 0. Ainsi le jeu de deux règles, qui disait $C \rightarrow G$, et $G \rightarrow C$, maintenant s'exprime ainsi en logique binaire : $0 \rightarrow 1$ et $1 \rightarrow 0$.

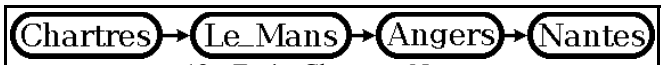
Dans un second temps le jeu de deux règles, qui dit : $0 \rightarrow 1$ et $1 \rightarrow 0$, peut être traduit dans un schéma électrique par une fonction inverseuse, i. e. par une négation f(x) = ┐x (figure ci-contre à droite).



17 alternateur bistable

Essayer de transposer des <u>trajets</u> vers le schéma : fonction+registre

Le trajet *Chartres→Nantes* ne possède pas un domaine suffisamment régulier pour être défini en compréhension[7]. Il ne peut donc pas se transposer vers le schéma : *fonction+registre*, et se représente seulement en extension[8] sous la forme *table+registre*.



18 - Trajet Chartes→Nantes

Il en va de même pour le trajet *LieuA→LieuZ→LieuE→LieuR→LieuT.*

Transposer le séquenceur linéaire vers le schéma : fonction+registre

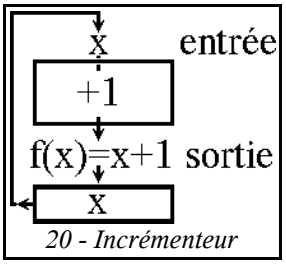
Pour illustrer ce cours par des exemples, nous devons prendre des cas plus réguliers, dont le domaine est construit par une fonction numérique. Dans le cas présent, on peut encore considérer que cette séquence est régulière si on regarde son code ASCII.

19 - Le séquenceur
A→B→C→D→E

Utilisons cette méthode du schéma logique combinatoire pour transposer le séquenceur.

Dans un premier temps, les symboles *A, B, C, D, E*, peuvent se traduire, s'abstraire par exemple par les nombres binaires 0, 1, 10, 11 et 100. Ainsi, le jeu de quatre règles :

LieuA→LieuB,
LieuB→LieuC,
LieuC→LieuD,
LieuD→LieuE,
maintenant s'exprime ainsi en logique binaire :
0→1, 1→10, 10→11, 11→100.



20 - Incrémenteur

Dans un second temps le jeu de quatre règles, qui dit : $0 \rightarrow 1$, $1 \rightarrow 10$, $10 \rightarrow 11$, $11 \rightarrow 100$, peut être traduit dans un schéma électrique par une fonction d'incrémentation, i. e. par une fonction $f_{(x)}$=x+1(figure ci-contre à gauche).

7 Définition d'un ensemble en compréhension : c'est un ensemble régulier, au point qu'on peut énoncer une règle pour construire systématiquement la liste de ses éléments.

8 Définition d'un ensemble en extension : c'est un ensemble irrégulier, au point qu'on doive énoncer un à un, la liste de ses éléments.

Ce faisant, nous retombons sur les séquenceurs de la logique combinatoire. Comme leur nom l'indique, ils constituent la forme la plus significative de la famille. Ils servent à compter et à séquencer.

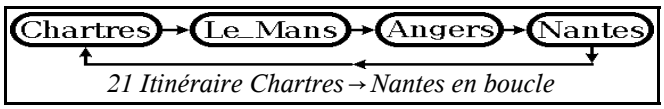
Dans l'exemple ci-dessus à droite, le séquenceur est initialisé à 0. En fait, il peut être initialisé à différentes valeurs, et compter avec un pas différent de 1. De même, on trouve encore des décompteurs qui comptent à rebours, i.e. en arrière, au lieu de compter en avant.

Il faut aussi bien voir que le séquenceur s'arrête dans l'état 5 ; ce qui ne se produit pas dans d'autres synoptique : c'est ce qui nous amène à la boucle (ou cycle).

Le domaine du séquenceur linéaire peut être construit récursivement au moyen d'une fonction numérique : l'incrémentation (figure ci-dessus à gauche).

Essayer de transposer des <u>itinéraires en boucle</u> vers le schéma : fonction+registre

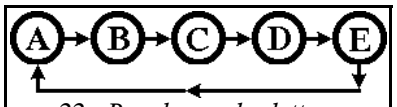
Le trajet *Chartres→Nantes* ne possède pas un domaine suffisamment régulier pour être défini en compréhension. Il ne peut pas se transposer vers le schéma : *fonction+registre*, et se représente seulement en extension sous la forme *table+registre*.



21 Itinéraire Chartres→Nantes en boucle

Il en va de même pour le trajet *LieuA→LieuZ→LieuE→LieuR→LieuT.*

Transposer le *cycle* vers le schéma : fonction+registre



22 - Boucle sur des lettres

Utilisons cette méthode du schéma logique combinatoire pour transposer le *cycle*.

Dans un premier temps, les symboles *A, B, C, D, E*, peuvent se traduire, s'abstraire par exemple par les nombres binaires 0, 1, 10, 11 et 100. Ainsi, le jeu de cinq règles :

LettreA→LettreB,
LettreB→LettreC,
LettreC→LettreD,
LettreD→LettreE,
LettreE→LettreA,

maintenant s'exprime ainsi en logique binaire :

0→1, 1→10, 10→11, 11→100, 100→0.

Dans un second temps le jeu de cinq règles, qui dit : *0→1, 1→10, 10→11, 11→100, 100→0*, peut être traduit dans un schéma électrique par une fonction conditionnelle, i. e. par la fonction $f_{(x)}$ suivante :

Si x < 100
alors $f_{(x)}$=x+1;
sinon $f_{(x)}$=0;

Transposer la *décroissance linéaire* vers le schéma : fonction+registre

Le jeu initial de quatre règles :

4→3 // Le symbole *4* se réécrit *3*
3→2 // Le symbole *3* se réécrit *2*
2→1 // Le symbole *2* se réécrit *1*
1→0 // Le symbole *1* se réécrit *0*

peut être traduit dans une table de quatre adresses : 4 *est traduit par* 3, 3 *par* 2, 2 *par* 1, et 1 *par* 0.

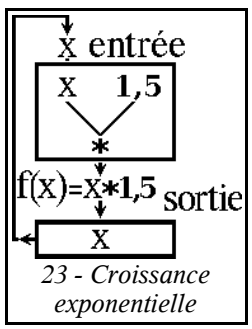
Dans un second temps ce jeu de quatre règles, qui dit : *4→3, 3→2, 2→1, 1→0*, peut être traduit dans un schéma électrique par une fonction numérique décroissante i. e. par la fonction suivante :

Si x > 0
alors $f_{(x)}$=x-1;
sinon $f_{(x)}$=0;

Transposer la *croissance exponentielle* vers le schéma : table+registre

Le jeu de quatre règles :

4→6 // Le symbole *4* se réécrit *6*
6→9 // Le symbole *6* se réécrit *9*
9→13 // Le symbole *9* se réécrit *13*
13→19 // Le symbole *13* se réécrit *19*



23 - Croissance exponentielle

peut être traduit dans une table de quatre adresses :
4 *est traduit par* 6, 6 *par* 9, 9 *par* 13, et 13 *par* 19.

Dans un second temps ce jeu de quatre règles, qui dit : *4→6, 6→9, 9→13, 13→19*, peut être traduit dans un schéma électrique par une fonction numérique décroissante i. e. par la fonction récursive suivante : $f_{(x)} = 1{,}5 * f_{(x)}$.

Attention, le calcul est effectué sur des entiers. Par exemple 9/2 = 4.

Séquenceurs sur des domaines numériques, définis par une fonction récursive

Les séquenceurs que nous avons pris en exemple appartiennent à une classe spéciale : ils prennent des valeurs dans les entiers naturels, et leur domaine peut être défini récursivement au moyen d'une fonction analytique.

Nous retrouverons cette classe des séquenceurs réguliers (SR) bientôt dans notre cours.