Chapitre 1.

Systèmes dynamiques à événements discrets

1.1 Introduction

Dans un certain nombre de systèmes conçus par l'homme, tels que
- les réseaux de communication et d'ordinateurs,
- les systèmes informatiques,
- les unités centrales d'ordinateurs elles-mêmes,

l'essentiel de l'enchaînement dynamique des tâches provient de phénomènes
- de synchronisation, et
- d'exclusion mutuelle ou de compétition dans l'utilisation de ressources communes, ce qui nécessite une politique d'arbitrage ou de priorité, questions généralement désignées sous le terme générique d'***ordonnancement***.

De tels systèmes, dans lesquels les transformations sont déclenchées par des événements ponctuels, typiquement l'arrivée d'un signal ou l'achèvement d'une tâche, sont appelés depuis le début des années 80 « Systèmes à Evénements Discrets (SED) » .

Le mot « discret » ne signifie ni « temps discret », ni « état discret », mais réfère au fait que la dynamique est composée d'***événements***, qui peuvent être des débuts et fins de tranches d'évolution continue mais qui seuls nous préoccupent dans la mesure où les fins conditionnent de nouveaux débuts.

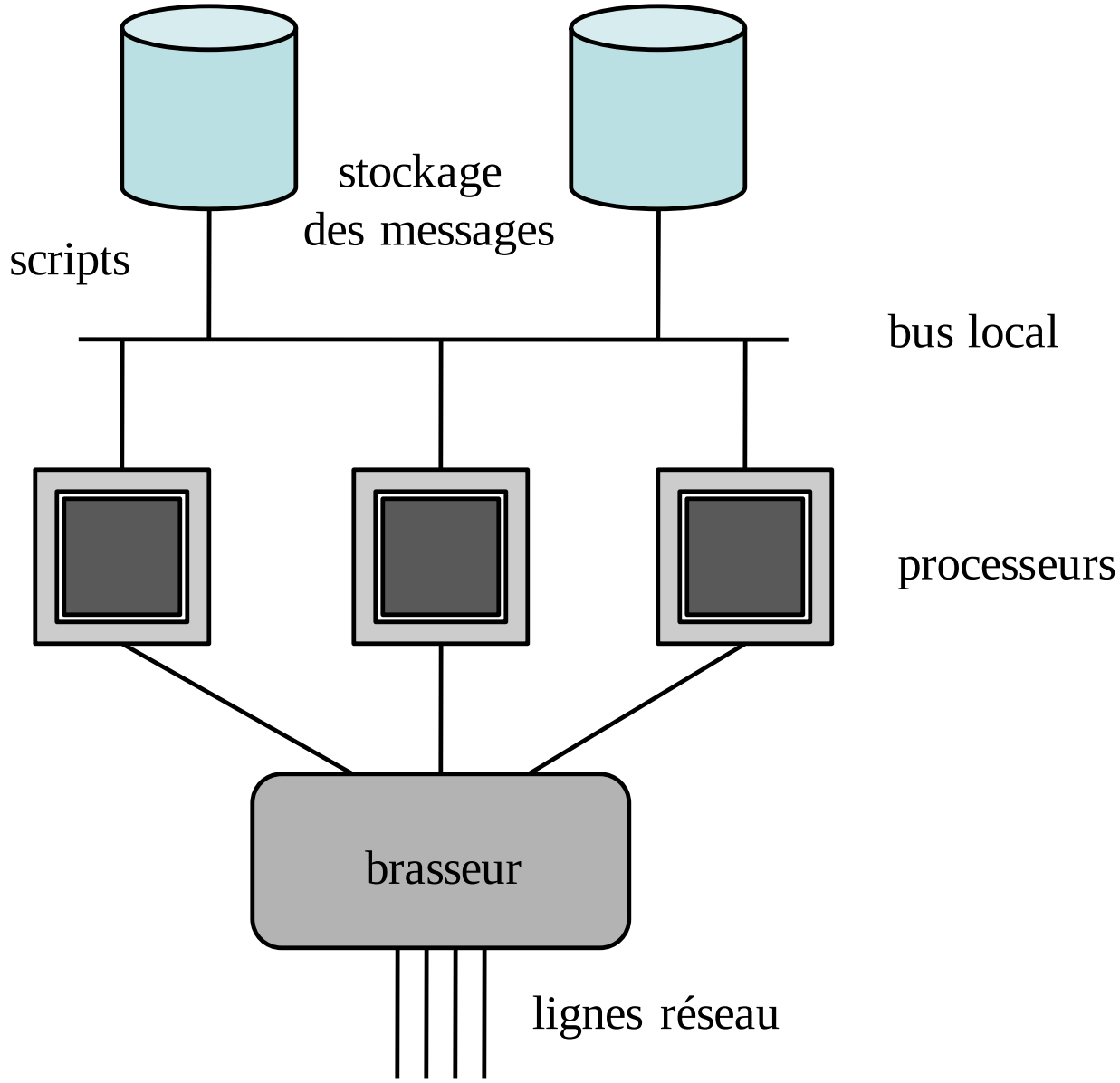
Nous allons étudier les méthodologies qui ont été développées depuis une trentaine d'années afin de modéliser, concevoir, analyser les performances et contrôler de tels systèmes.

Un exemple de système temps réel : un serveur de messagerie vocale

Considérons une machine, utilisée par les services de messagerie vocale (un ***serveur vocal***), répondeurs, boîtes aux lettres, jeux, annonces, etc.

Le dialogue usager-service est vocal ou utilise les touches du combiné.
Fonctions du serveur : stocker et gérer les messages, les restituer sur demande.

La figure qui suit donne un exemple d'architecture d'une telle machine.



Architecture d'un serveur vocal

Chaque communication se conforme approximativement au schéma :
- sur détection d'un appel du réseau, affectation d'un processeur;
- dialogue avec l'usager (envoi d'un message, écoute de la réponse, action suivante, etc.);
- le « service » désigne un processeur particulier chargé du pilotage du dialogue; c'est lui qui contient le service effectif; on parle de « dérouleur de script »;
- consultation de la BD où sont stockés les messages vocaux de l'usager, pour les délivrer;
- etc.

Chaque communication est décrite par un « script », dans lequel chaque action correspond à un travail d'une des entités du serveur. Le script contient de nombreux branchements, correspondant aux décisions de l'usager ou aux fausses manœuvres.

On pourrait expliciter encore ce schéma, en y faisant apparaître les actions élémentaires des processeurs, les lectures et écritures en mémoire, par exemple.

On voit combien le déroulement d'une communication est complexe, et ressemble en fait au traitement des programmes d'un système informatique multiprogrammé.

Les questions que cette description amène concernent en premier lieu les possibilités et les performances de la machine – combien de communications peut-elle accepter ? quel est son temps de réponse ? etc.
Elles concernent aussi les choix de conception et de dimensionnement – nombre de processeurs, taille de mémoires, organisation du traitement des requêtes, etc.

1.2 Le concept d'événement

Un événement se produit instantanément et cause la transition de l'état d'une valeur (discrète) à une autre valeur.

Il peut être identifié avec une action spécifique qui a été prise (quelqu'un qui a pressé un bouton ou appuyé sur une touche du clavier, …) .

Il peut être vu comme une occurrence spontanée dictée par la nature (un ordinateur se bloque pour une raison trop compliquée à déterminer).
Ce peut aussi être la conséquence de plusieurs conditions soudainement toutes satisfaites.

On représentera un événement par le symbole ***e***.
Lorsqu'un système est affecté par différents types d'événements, on supposera qu'il est possible de définir un ensemble d'événements ***E*** dont les éléments sont tous ces événements.

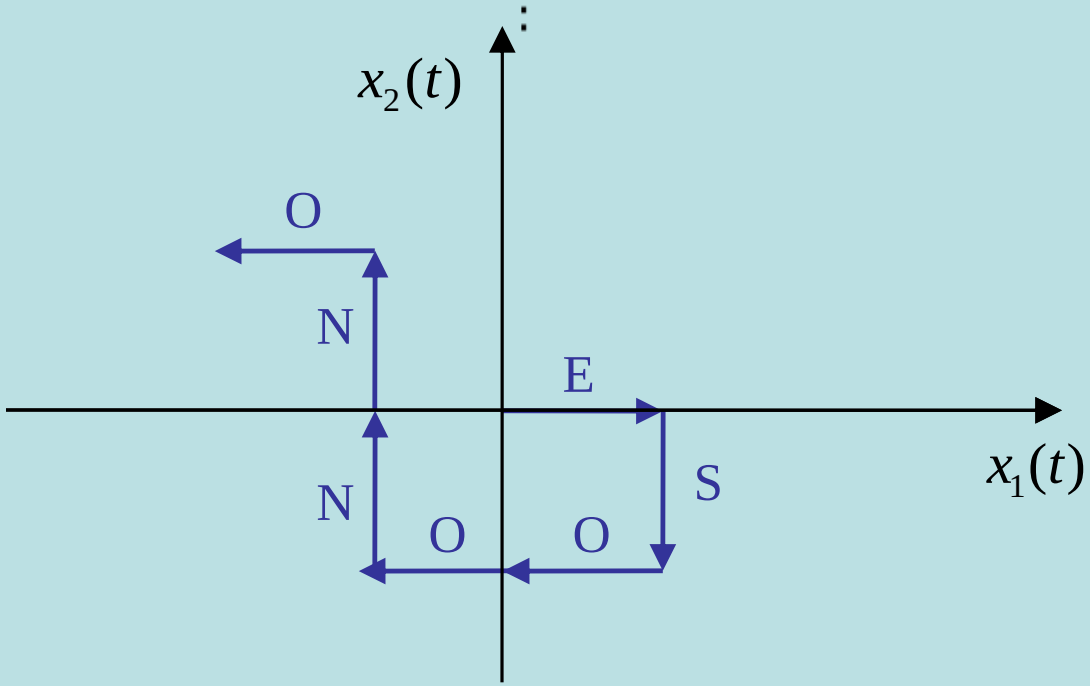
Exemple 1. Marche aléatoire

Une marche aléatoire est un modèle utile pour plusieurs processus intéressants, en particulier pour quelques jeux de hasard.

Quand la marche aléatoire s'effectue dans deux dimensions, elle peut être visualisée comme une particule qui peut être déplacée d'une unité de distance (« un pas ») à la fois dans l'une des quatre directions : nord, sud, ouest ou est.

On suppose que la direction est choisie de manière aléatoire et indépendante de la position courante.

L'état de ce système est la position de la particule, (x_1, x_2) mesurée dans un plan, avec x_1 et x_2 entiers, c.-à-d. l'espace d'état est l'ensemble discret $\boldsymbol{X} = \{(i,j) : i,j = \ldots, -1, 0, 1, \ldots\}$.



Marche aléatoire dans un plan

Chaque événement (N, S, E ou O) provoque une transition d'état dans l'espace d'état (x_1, x_2). Si aucun événement ne se produit, l'état ne peut pas changer.

Dans ce cas un ensemble naturel d'événements est

$$\boldsymbol{E} = \{\text{N, S, E, O}\}$$

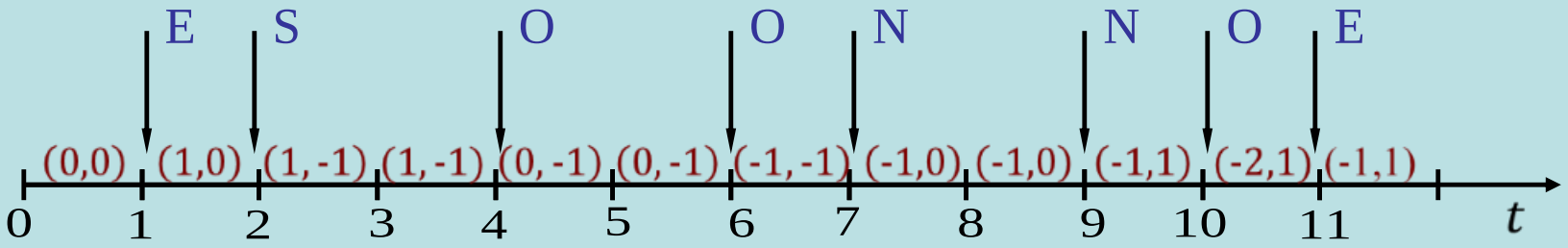
correspondant aux quatre événements « un pas vers le nord », « un pas vers le sud », « un pas vers l'ouest » et « un pas vers l'est ».

La figure montre le chemin de progression (trajectoire) dans l'espace (x_1, x_2) résultant de l'état initial (0, 0) et de la séquence d'événements (E, S, O, O, N, N, O).

Voyons maintenant comment les événements se situent dans le temps.
Supposons qu'il y a quatre « joueurs » distincts, chacun responsable du mouvement de la particule dans une seule direction (N, S, E et O).
Chaque joueur agit en émettant occasionnellement un signal pour que la particule se déplace dans sa direction.
Le système est donc actionné par les événements définis par ces quatre joueurs agissant de manière asynchrone.

A titre d'exemple, supposons que le joueur N émette des signaux aux instants discrets (7, 9), S le fasse à l'instant 2, O à (4, 6, 10) et E à (1, 11).

La figure ci-dessous représente la trajectoire dans l'espace d'états sous la forme d'un chronogramme où les événements causent les transitions d'état. On suppose que l'état initial est (0, 0).



Dans cet exemple on a supposé que deux événements ne se produisent jamais au même instant. Si tel était le cas, alors la transition d'état résultante devrait refléter l'occurrence des deux événements.

Supposons, par exemple, qu'à l'instant 1 à la fois E et S émettent un signal. On peut supposer sans causer de problème que l'un ou l'autre des deux événements se produit en premier (mais au même instant), puisque l'état résultant est le même, (1, -1), indépendamment de l'ordre.

Bien sûr, ce n'est pas toujours le cas. En général, l'ordre exact d'occurrence des événements importe pour déterminer l'état résultant.

Par exemple si l'état est le solde d'un compte en banque, égal à 0 à l'heure actuelle, et si D représente l'événement « dépôt de 100 € sur le compte » et C l'événement « encaissement d'un chèque de 100 € tiré sur le compte », les événements peuvent se produire au même instant mais nécessairement l'un d'eux affecte le solde en premier :
- si D survient le premier, l'effet global est un solde résultant égal à 0,
- par contre, si C est le premier, le compte est pénalisé avant le dépôt et l'effet global est un solde négatif, égal aux frais de service pour l'émission d'un chèque sans provision.

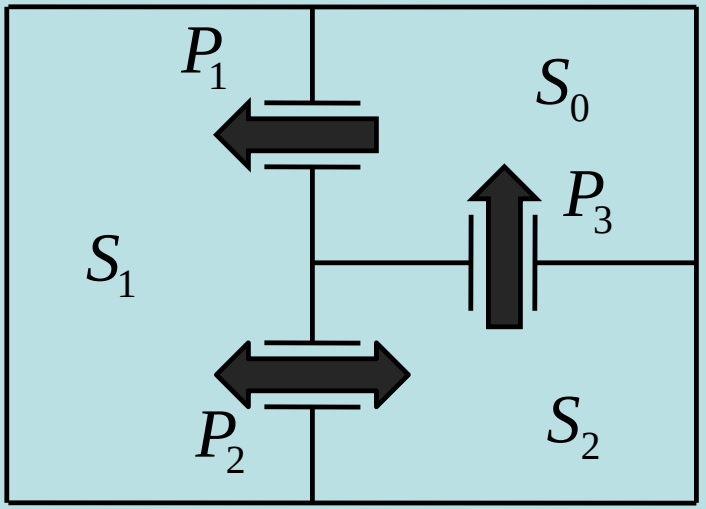
Dans certains cas, il faut modéliser explicitement l'effet de deux événements qui se produisent simultanément comme totalement distinct de l'occurrence de ces événements dans un ordre ou dans l'autre.

Exemple 2. Souris dans un labyrinthe

Une souris se déplace de manière spontanée (elle agit sans intervention extérieure) à l'intérieur d'un labyrinthe.

Les salles S_i communiquent par des portes unidirectionnelles P_1 et P_3 et bidirectionnelle P_2.

On note par p_i l'événement « la souris passe par la porte P_i ».

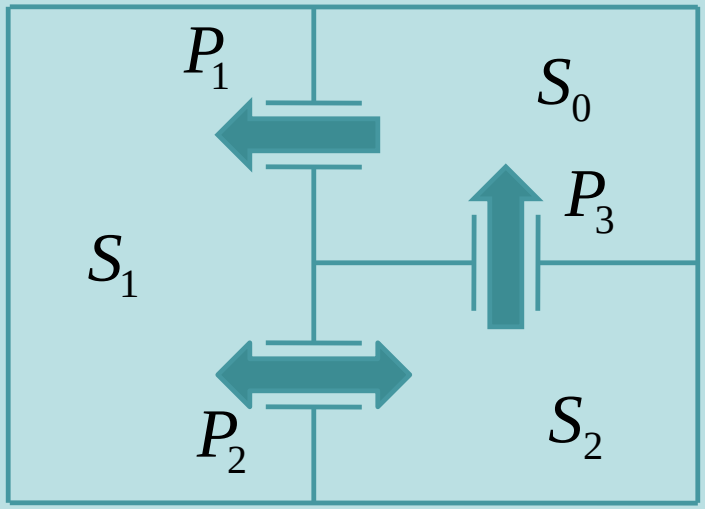


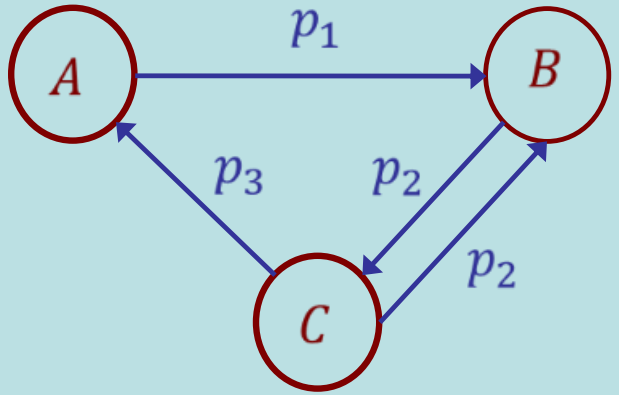
Les situations sont :

- la souris est dans la salle S_1
- la souris est dans la salle S_2
- la souris est dans la salle S_3

ce qui définit trois états différents (A, B et C), relatifs aux possibilités d'occupation des salles. L'espace d'états s'écrit $\boldsymbol{X} = \{A, B, C\}$.

Soit E l'ensemble des événements : $E = \{p_1, p_2, p_3\}$.
Le passage de l'état « souris en S_0 » (état A) à « souris en S_1 » (état B) a lieu sur l'occurrence de l'événement p_1, etc.





Nous nous intéressons à des systèmes pour lesquels chaque événement $e \in E$ définit un processus distinct par lequel les instants auxquels e se produit sont déterminés.
Les transitions d'état sont le résultat de la combinaison de processus ***asynchrones*** et ***concurrents***.
En outre ces processus ne sont pas nécessairement indépendants entre eux.

Ce sont les événements qui font progresser le système.
De tels systèmes sont plus difficiles à modéliser ou à analyser que des systèmes qui effectuent leurs transitions d'état de manière synchrone, sur les tops d'une horloge qui est à l'origine de chaque transition d'état.

Pour comprendre les systèmes qui nous intéressent il y a par contre plusieurs mécanismes asynchrones de déclenchement d'événements à spécifier.

Ce type de cause de transition d'état correspond à la notion familière d'interruption (externe) dans les ordinateurs.

Alors que beaucoup des fonctions d'un ordinateur sont synchronisées par une horloge, les systèmes d'exploitation sont conçus pour répondre aussi aux appels asynchrones qui peuvent se produire à tout instant.
Par exemple, une requête par un utilisateur externe ou un message d'arrivée à échéance (*timeout*) peuvent être la conséquence d'événements spécifiques, mais être complètement indépendants de l'horloge de l'ordinateur.

1.3 Propriétés caractéristiques des systèmes à événements discrets

Les systèmes à événements discrets (SED) satisfont les deux propriétés suivantes :

- l'état est décrit par des grandeurs discrètes, telles que le nombre de processeurs en activité, le nombre de messages en attente, etc.
- l'évolution est conditionnée par l'occurrence d'événements à « certains instants », tels que la fin d'exécution d'une tâche, le franchissement d'un seuil devant déclencher une action, l'arrivée d'un datagramme, la défaillance d'un périphérique, etc. Ainsi, les transitions d'état apparaissent seulement suite à des événements, qui se produisent en des instants discrets du temps.

On utilisera par la suite l'acronyme SED.

Dans la discussion qui suit, une première définition des SED, basée sur ces propriétés, est présentée de façon informelle.
Des définitions formelles seront données dans le chapitre 2, dans lequel des procédures de modélisation détaillées seront développées.

Définition : Un système à événements discrets (SED) est un système à espace d'état discret dont les transitions entre états sont associées à l'occurrence d'événements discrets asynchrones.

« L'espace d'état est discret » signifie que les états sont dénombrables (en particulier ils peuvent être en nombre fini, mais pas toujours).
« Les événements sont discrets » implique que le temps n'est pas pensé comme continu.

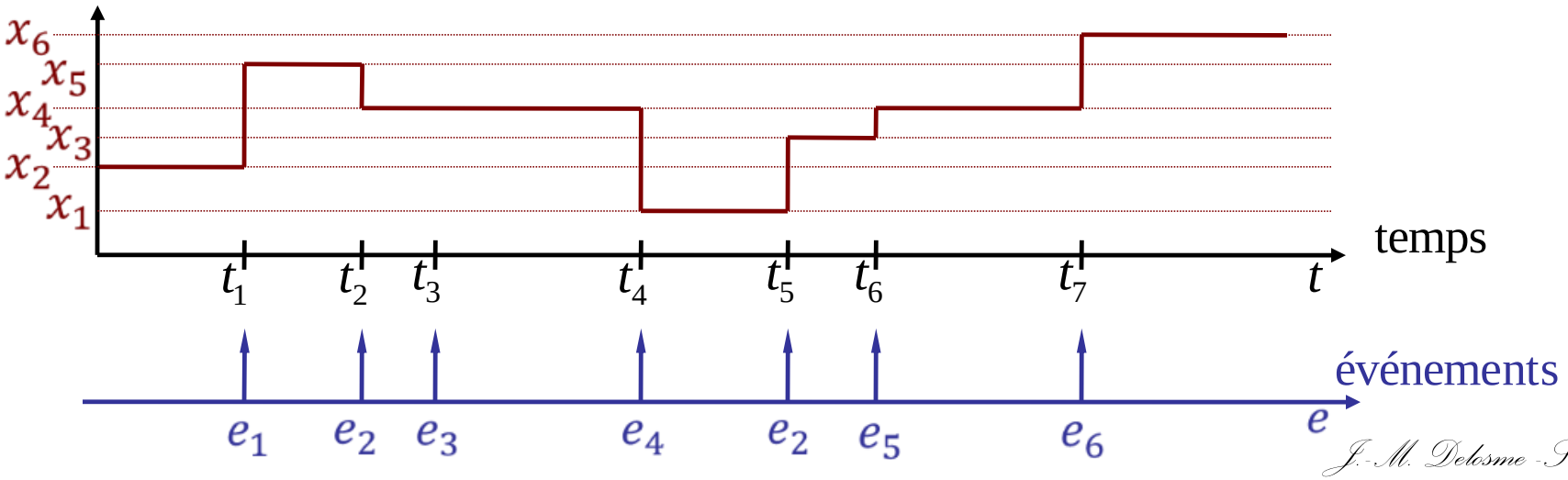
Beaucoup de systèmes, en particulier des systèmes conçus par l'homme, sont des systèmes à états discrets. Même si ce n'est pas le cas, pour beaucoup d'applications, une vue discrète d'un système complexe peut être nécessaire.

Voici quelques exemples de systèmes à états discrets :
- l'état d'une machine peut être sélectionné dans un ensemble tel que {on, off} (*marche/arrêt*) ou {busy, idle, down} (*occupé, inactif, en panne*),
- un processus peut être dans 3 états {en exécution, prêt, en attente}. On peut vouloir décomposer les états de manière plus fine, par exemple incorporer dans l'état la valeur du compteur ordinal.

- la plupart des jeux peuvent être modélisés comme ayant un espace d'état discret, par exemple, pour le jeu d'échecs chaque configuration de l'échiquier définit un état. L'espace résultant est discret (et seulement grand, pas infini).

Du point de vue de la modélisation, le fait que les transitions entre états soient associées à l'occurrence d'événements discrets asynchrones implique que, si l'on peut identifier un ensemble d'événements pouvant chacun provoquer une transition d'état, alors le temps n'est pas vraiment la variable à considérer comme causant les transitions du système.

La trajectoire ci-dessous met en évidence les propriétés caractéristiques des SED.



La trajectoire (constante par morceaux) saute d'un état à l'autre avec l'occurrence d'un événement.

Remarques :
- un même événement (e_2) peut conduire à des états différents (x_4 et x_3)
- des événements différents (e_2 et e_5) peuvent conduire à un même état (x_4)
- des événements (e_3) peuvent se produire et être inactifs pour ce système
- la séquence des états et le temps de maintien associé à chaque état caractérisent la trajectoire.

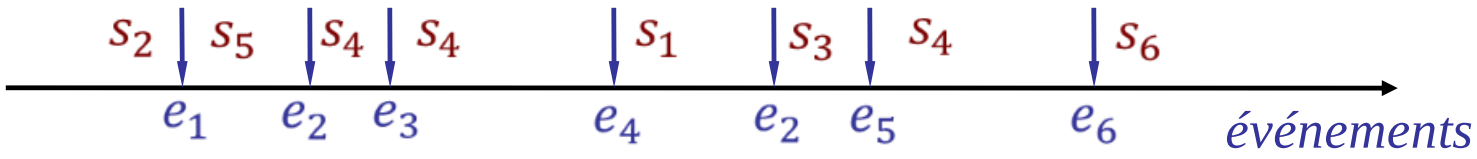
Nous n'avons pas, à ce point, d'analogue à

$$\dot{x}(t) = f(x(t), u(t), t)$$

pour les systèmes continus.

Lorsque l'entrée du SED est spécifiée de manière déterministe comme une séquence d'événements ($e_1, e_2, \ldots$) sans information sur le temps auquel les occurrences de ces événements se produisent partant d'un état initial, on décrit la trajectoire en termes de séquence d'états résultants : on a un ***modèle non temporisé*** du système.

Reprenant l'exemple précédent, on spécifie les états mais on ne peut pas spécifier les instants où s'effectuent les transitions. La trajectoire devient



Un ***modèle non-temporisé*** décrit le comportement logique du système et permet en particulier de répondre à la question « un état particulier peut-il être atteint ? »

Lorsque l'entrée du SED est spécifiée de manière déterministe comme une séquence d'événements auxquels sont associées les dates d'occurrence t_i, on peut construire la trajectoire complète du ***modèle temporisé*** et répondre notamment aux questions :
« Quand un état particulier sera-t-il atteint ? »
« Combien de temps le système passe-t-il dans un état particulier ? »
« Combien de fois un état particulier peut-il être atteint sur un intervalle de temps donné ? »

L'entrée est spécifiée sous la forme de séquence d'événements temporisés $((e_1, t_1), (e_2, t_2), (e_3, t_3), \dots)$. On verra cependant que le fait que différents processus événements sont concurrents et souvent interdépendants de façon complexe rend délicates la modélisation et l'analyse de ces systèmes.

1.4 Exemples de systèmes à événements discrets

Nous allons voir deux classes d'exemples : systèmes informatiques et réseaux de communication.
Pour faciliter leur présentation, un bloc de base qui va être fort utile pour les représenter ainsi que beaucoup d'autres SED est présenté en premier.

La station de service de base

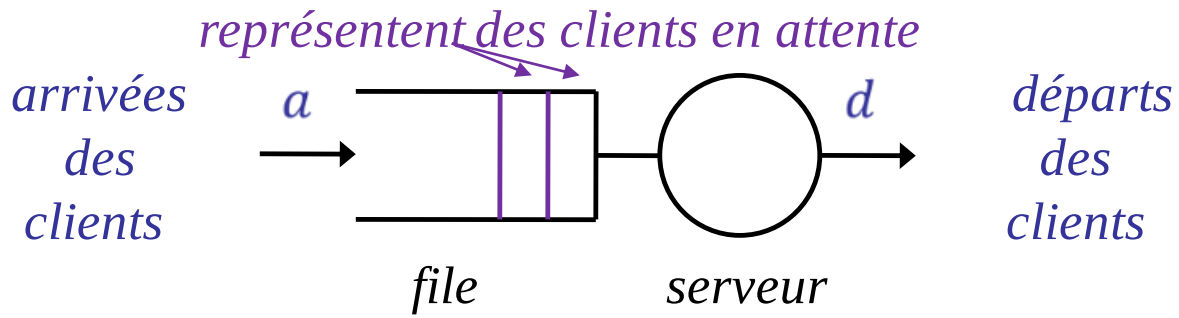
La théorie des files d'attente considère des clients, qui se déplacent dans un réseau de serveurs. Lorsque plusieurs clients tentent simultanément d'obtenir un service, certains doivent patienter et attendre dans des ***files d'attente***, ou ***tampons.***

Le vocabulaire est fixé et conventionnel.

- Les clients peuvent être des humains faisant la queue à un guichet pour obtenir un billet de train. Le serveur est l'employé qui délivre les billets.

- Ils peuvent représenter les paquets qui sont aiguillés vers une ligne de transmission et qui attendent la disponibilité de la ligne (deux paquets peuvent être en contention). Le serveur est alors l'automate d'émission, et le temps de service le temps de l'émission.

La station de service est composée d'une file d'attente, de capacité finie ou non, vidée par un (ou des) serveur(s), et alimentée par un flux de clients.



La station de service de base

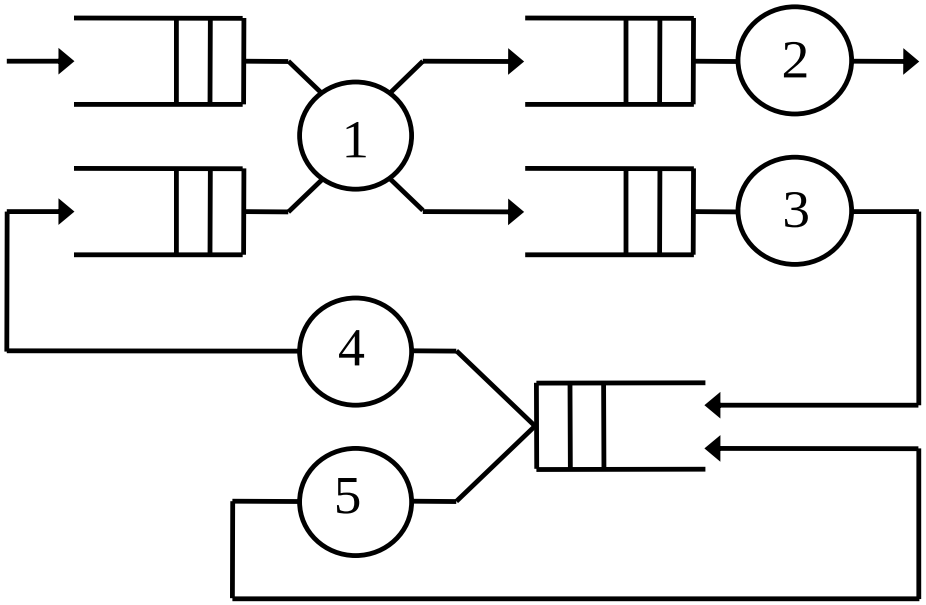
Le processus de service des clients est supposé prendre un temps strictement positif (sinon il n'y aurait pas d'attente).
Ainsi, un serveur peut être vu comme un « bloc retard » (*delay block*) qui retient un client pendant le temps nécessaire au service.

Pour décrire ce système de façon précise, il faut pouvoir spécifier
- la ***capacité de la file d'attente,*** qui est le nombre maximal de clients pouvant attendre dans l'espace de la file. Dans beaucoup de modèles cette capacité est supposée infinie, dans le sens où l'espace de la file est suffisamment grand pour que la probabilité qu'il soit plein est négligeable.
- la ***discipline de service,*** qui est la règle selon laquelle quand le (un) serveur se libère, le nouveau client à être servi est choisi dans la file. La règle la plus simple est FIFO (*first-in first-out = premier entré premier servi*) : les clients sont servis dans l'ordre précis dans lequel ils arrivent.

Vu comme un SED, la station de service de base a pour ensemble d'événements $\boldsymbol{E} = \{a, d\}$, où a représente un événement « arrivée » et d représente un événement « départ ». Une variable d'état naturelle est le nombre de clients dans la file, c.-à-d. la longueur de la file d'attente.

Par convention, on s'autorise à inclure dans la longueur à l'instant t un client en train d'être servi à cet instant. Ainsi l'espace d'état est l'ensemble des entiers non négatifs, $\boldsymbol{X} = \{0, 1, 2, \ldots\}$.

Une file d'attente peut être reliée à plusieurs serveurs, et plusieurs files à un serveur, ou les deux. Clairement, des connections séries et parallèles arbitraires de ce bloc de base sont possibles, comme



Un exemple de réseau de files d'attente

Les clients sont perçus comme un flux à travers les composantes (files d'attente et serveurs) du système selon des règles spécifiques.

Un tel réseau de files d'attente peut être utilisé pour modéliser l'exemple introductif du serveur de messagerie; chaque serveur y représente une ressource active (processeur) devant laquelle s'accumulent les clients, chaque communication est représentée par un client qui chemine entre les ressources au gré des différentes étapes du script.

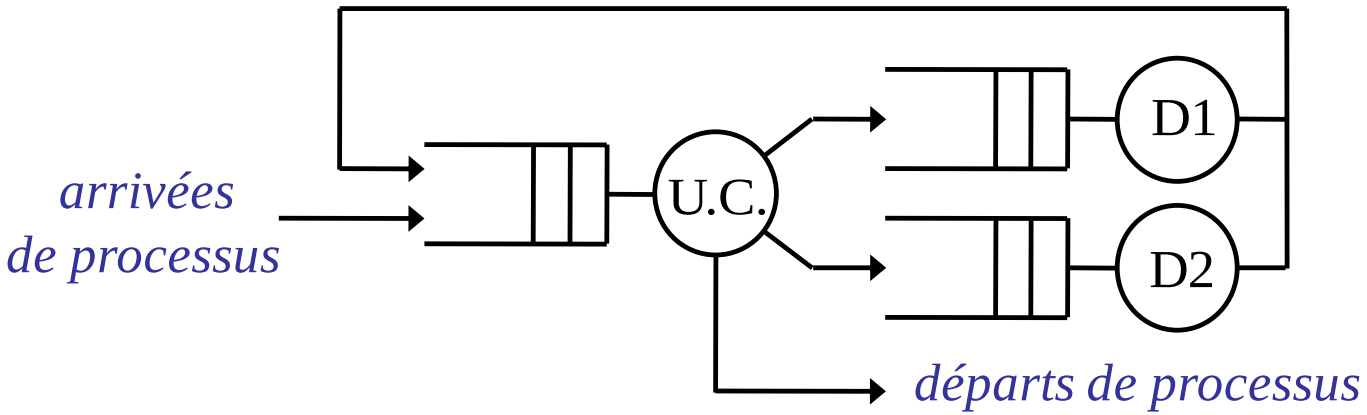
Systèmes Informatiques

Dans un S.I. typique, les travaux, tâches ou transactions sont les clients en compétition pour l'attention des serveurs, qui sont des processeurs (U.C., etc.), ou des périphériques (imprimantes, disques durs, etc.).

Quand un serveur est occupé au moment d'une demande d'un processus (ou tâche), ce processus est placé dans une file d'attente, qui est une partie essentielle du S.I.

Il est souvent pratique de représenter un tel système avec un modèle de réseau de files d'attentes utilisant le bloc de base décrit plus haut.
Un exemple simplifié de configuration de S.I. est présenté ci-dessous.

Dans ce système les processus arrivent dans la file d'attente de l'U.C. (processus prêts). Une fois servis par l'U.C., soit ils quittent le système (terminés), soit ils demandent l'accès à l'un des deux disques durs (mise en attente, E/S, suspension) puis ils retournent ensuite dans la file d'attente pour l'U.C.



Un exemple de configuration de système informatique

L'ensemble des événements est constitué des arrivées et des départs des serveurs. Soit ici

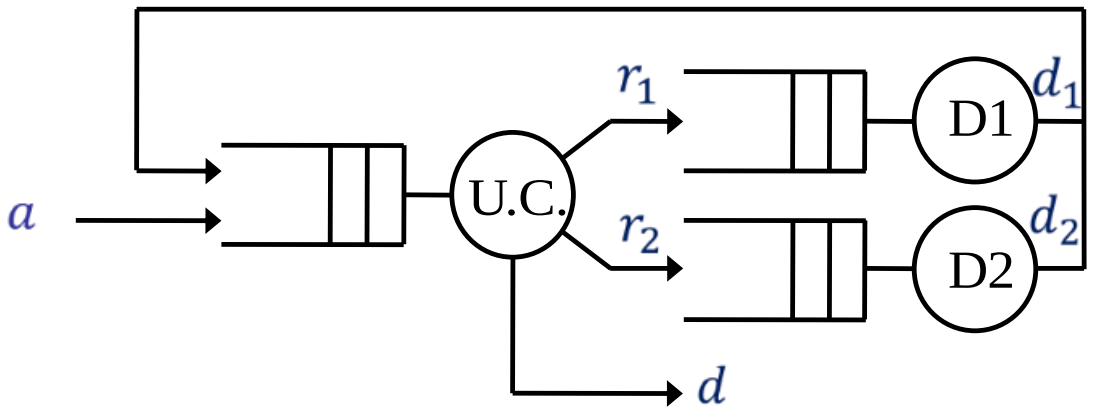
$$\boldsymbol{E} = \{a, d, r_1, r_2, d_1, d_2\}$$

où - a est une arrivée de l'extérieur dans le système,

- d est un départ de l'U.C. (état « en exécution ») vers l'extérieur (état « terminé »),

- r_1, r_2 sont des départs de l'U.C. vers les disques 1 et 2, respectivement,

- d_1, d_2 sont des départs des disques 1 et 2, respectivement, qui retournent toujours vers la file d'attente de l'U.C.



Modèle d'un système informatique simplifié

Une représentation possible de l'état de ce système est le vecteur

$$\boldsymbol{X} = [x_{UC}, x_{D1}, x_{D2}]$$

des longueurs des files d'attente de l'U.C. et des disques 1 et 2.
Dans ce cas l'espace d'état est

$$\boldsymbol{X} = \{[x_{UC}, x_{D1}, x_{D2}]: x_{UC}, x_{D1}, x_{D2} \geq 0\}$$

Note : espace discret, infini si l'on n'impose pas de contrainte sur les capacités.

Systèmes de communication

Deux stations, A et B, sont reliées au moyen d'un canal.
Le canal permet à tout instant de transmettre une seule trame, en provenance d'une station. Ainsi si B envoyait sur le support de transmission une trame en même temps que A, les deux signaux seraient brouillés et aucun message ne serait transmis en final. C'est ce qu'on appelle une collision.

Il s'ensuit que le ***canal*** peut être dans l'un de trois états, I pour inactif ou disponible (*idle*), T pour transmettant un message, et C pour transmettant deux messages (ou plus), résultant en une collision.

De manière similaire, chaque ***station*** peut être dans l'un des trois états, I pour inactif, T pour transmettant (ou émettant), W (*wait*) pour en attente pour transmettre un message disponible.

Deux problèmes doivent être résolus. Chacune des stations ne connaît pas l'état de l'autre, et, de plus, les stations peuvent ne pas connaître l'état du canal (ou n'être capables de reconnaître qu'un état). On voit donc que la possibilité de collision est présente à moins que les utilisateurs opèrent en appliquant des règles (superviseur) garantissant que ce processus de communication réussisse.

De manière générale, dans les systèmes de communication, on a besoin de mécanismes de contrôle pour garantir que
- l'accès aux serveurs (le canal est un serveur, de capacité 1) soit accordé de manière équitable et efficace (QS, qualité de service), et que
- les objectifs du processus de communication soient atteints (c.-à-d. chaque message est bien transmis à sa destination).

Ces mécanismes de contrôle, ce sont les protocoles, qui peuvent être complexes.

La conception des protocoles et la vérification qu'ils fonctionnent comme spécifié posent des problèmes difficiles.

Dans notre exemple, pour un réseau local, Ethernet ou, plus précisément, la norme IEEE 802.3 (CSMA/CD : Carrier Sense Multiple Access with Collision Detection) est un protocole classique pour gérer les stations émettrices-réceptrices (pas nécessairement réduites à deux ni avec un câblage linéaire, mais pouvant être nombreuses avec un réseau sous-jacent arborescent par exemple).

Revenant à notre exemple jouet, les événements déclencheurs pour le système sont les arrivées de messages à transmettre en A et B, les émissions par A et B de messages sur le médium, et l'achèvement d'une transmission par le canal.

Ainsi on peut définir l'espace d'état (avec x_C l'état du canal)

$$\boldsymbol{X} = \{[x_A, x_B, x_C]: x_A \in \{I, T, W\}, x_B \in \{I, T, W\}, x_C \in \{I, T, C\}\}$$

et l'ensemble d'événements

$$\boldsymbol{E} = \{a_A, a_B, \tau_A, \tau_B, \tau_C\}$$

où - a_A, a_B sont les arrivées des messages à transmettre par A et B, respectivement,

- τ_A, τ_B sont les événements d'envoi de message sur le canal par A et B, respectivement,
- τ_C est l'événement d'achèvement d'une transmission sur le canal (avec un message ou plus présents).