Identification d'un formalisme adapté à la représentation de Règles de Décision agronomique

Contents

Le concept qui pourrait se révéler éclairant, en l'occurrence, est celui d'évènement (...)
Qu'un fait ne soit qu'une exemplification particulière d'une régularité générale ou qu'il ait le caractère d'un surgissement historique, il signifie l'occurrence d'un nouvel état de choses, le passage d'une certaine figure du monde à une autre (...)
La notion d'évènement tente de dire la facticité du fait. L'évènement est l'occurrence, dans la trame des processus qui constituent le monde et son histoire, d'un nouvel état de choses. Il est à la fois transition et surgissement, enveloppant ainsi la continuité et la discontinuité, la similitude et la différence, la reprise et la nouveauté (...)
Le véritable évènement, c'est le surgissement toujours advenant du monde, cette sorte de pulsation instauratrice en et par laquelle le cosmos ne cesse de se produire et de s'acheminer vers ce qui, de lui, est toujours encore à se constituer (...)
La notion d'évènement connote précisément un certain nombre de caractères qui affinent cette référence à la facticité. D'abord l'évènement a un aspect d'unicité : ce qui arrive, au sens fort du terme, se singularise par son occurrence même. Cette unicité joue à deux points de vue. D'une part, même s'il y a des traits qui se reproduisent, et s'il y a de l'invariance, le monde, considéré dans toutes ses déterminations, prend à chaque instant une figure singulière, et en ce sens n'est jamais totalement égal à lui-même. Et d'autre part, son devenir est en quelque sorte d'un seul tenant. Ce qui se produit en chaque moment n'est que la réfraction dans l'instant d'un processus d'ensemble qui s'étend dans la durée et qui est précisément l'incessante venue au jour du cosmos (...)
L'évènement met en œuvre un principe de liaison : ce qui se produit, à un moment donné, s'appuie toujours sur ce qui est déjà réalisé, reprend pour ainsi dire dans son acte même les processus antérieurs, qui sont, à l'égard des nouvelles opérations, des conditions de possibilité. Et en même temps, ce qui se produit annonce ce qui pourra suivre, en ce sens que de nouvelles conditions sont posées, qu'une nouvelle base est établie pour la construction d'architectures plus complexes. En somme, l'évènement effectue une transformation du champ des possibilités : en actualisant, à un moment donné, les possibilités objectives existantes à ce moment, il fait apparaître une nouvelle distribution des possibilités objectives, en laquelle s'amorcent déjà des développements ultérieurs.

— Jean Ladrière
Penser la Création (1976)

L'objectif de ce chapitre est de présenter le cadre théorique de modélisation pour mon travail. Pour cela, on analyse la nature d'un système de production agricole, soumis aux attaques d'un ou plusieurs pathogènes, et sur lequel l'agriculteur doit intervenir sous risque de perdre tout ou partie de sa production. Dans un premier temps, le problème la protection des végétaux est analysé comme un problème de contrôle. On présente ensuite le cadre théorique et les principales approches de modélisation des Systèmes à Évènements Discrets (SED). Enfin, on terminera par une présentation du langage de modélisation que j'ai utilisé : *Statechart*.

2.1 Système et contrôle pour la protection intégrée des cultures

Dans les sections qui suivent, on cherche à caractériser les objets sur lesquels porte l'étude et à présenter les notions théoriques utilisées. On les présente dans le cadre pratique de la pathologie végétale, mais il est évident que ces notions théoriques restent valables dans d'autres domaines d'applications.

2.1.1 Système continu, discret ou hybride

Une maladie se représente traditionnellement, en épidémiologie botanique, sous la forme d'un *pathosystème* (voir fig. 2.1). Il s'agit d'un *système* composé de quatre composants ou sous-systèmes, interagissant les uns avec les autres au cours du temps. En se plaçant à l'échelle de la parcelle, on observe différentes *dynamiques* de développement (des comportements) *continu* comme la croissance des plantes ou l'extension des symptômes de la maladie.

On observe également des *évènements*, une pluie, une contamination, un travail au champ ou encore le débourrement. Ces évènements entraînent parfois des évolutions *discrètes* du système (cas du mildiou nécessitant une période d'humectation pour pouvoir réaliser une nouvelle contamination). Cependant ce n'est pas toujours le cas et l'aspect *discret* ou *continu* d'un phénomène dépend en grande partie de l'échelle de *temps* avec laquelle on observe le système. Ainsi, la contamination par l'oïdium d'une culture correspond bien à un évènement : la mise en contact d'une propagule pathogène avec son hôte est un évènement discret par nature. Pourtant le développement des maladies fongiques se représente au cours de la saison comme un phénomène continu qu'on modélise classiquement par des équations différentielles (un exemple canonique est donné par Vanderplank, 1963, voir encadré 2). A l'inverse, à l'échelle de la journée, une opération culturale telle qu'un effeuillage [a] se caractérise par une durée. Néanmoins par rapport à la saison végétative, échelle à laquelle les effets de l'oïdium sont mesurables, cette durée n'a aucune importance. L'effeuillage est alors perçu comme un instant, caractérisé cette fois par un avant, un après, et les modifications du système qu'il entraîne. On peut dès lors représenter l'occurrence d'un effeuillage comme une *action* sans durée provenant d'un composant du pathosystème pour modifier *l'état* d'un ou plusieurs autres composants.

In fine, on constate qu'un pathosystème tel que celui de la vigne combine des phénomènes discrets et continus. Ce type de système est dit hybride par les automaticiens et

sa complexité motive de nombreux travaux ; on pourra se référer par exemple à (Lunze, 2000).

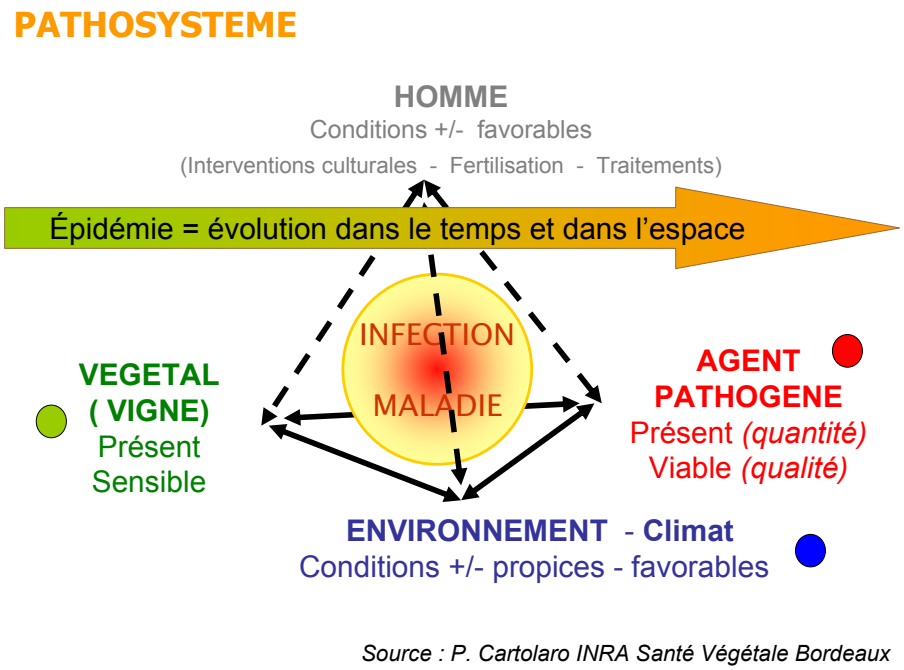


FIGURE 2.1 – Schéma des composantes d'un pathosystème

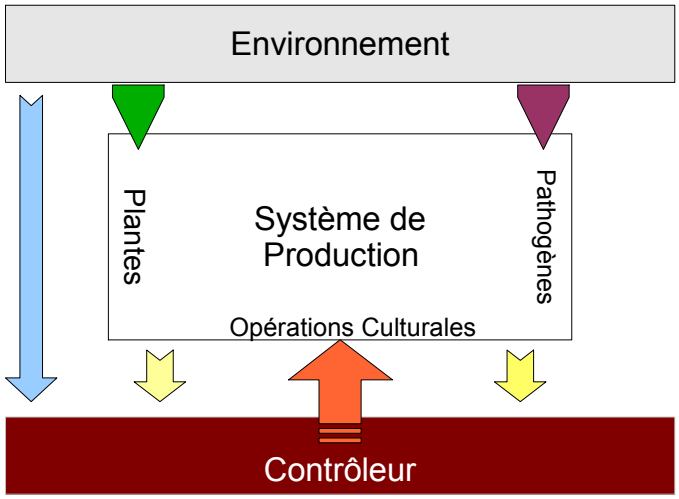
2.1.2 Contrôler une épidémie

Dans la section précédente on se plaçait du point de vue de l'épidémiologie qui modélise les épidémies de manière descriptive sous forme de pathosystèmes. Dans cette optique, le système est clos, c'est à dire que toutes les interactions ont pour origine et pour destination des éléments du système.

Prenons à présent un point de vue plus opérationnel. L'encadré 3 illustre ce point de vue. Un système de production soumis aux influences d'un environnement est *contrôlé* par un producteur. On pourrait le comparer à une usine composée de machines soumises à l'usure et aux incidents de production dont il s'agirait d'assurer les opérations de maintenance. Détaillons notre système de production agricole. Il s'agit d'une culture (la vigne) dont la production est mise à mal par des épidémies et dont la maintenance est assurée par des opérations phytosanitaires ou de prophylaxies. D'autres opérations culturales sont bien entendu nécessaires à la production. En agriculture, le système de production est soumis aux aléas du climat représentés fig. 3 par l'environnement.

La protection phytosanitaire est bien souvent, pour les maladies qui nous occupe, de nature systématique. L'agriculteur dispose dans ce cas d'un calendrier de traitements établi avant le début de la saison végétative et qui lui indique quand appliquer son premier traitement, quand renouveler l'application et à quelle cadence. Dans cette approche, il n'y a pas de prise en compte de l'épidémie en cours pour piloter la protection. L'ap-

a. En viticulture, l'effeuillage est une opération prophylactique consistant à aérer les grappes en retirant l'excès de feuillage qui les entoure, voir (Serrano, 2001).



Le système de production agricole est soumis à son environnement qui influence les plantes et les pathogènes et que contrôle un agriculteur. A partir de la surveillance du système de production et de l'environnement, l'agriculteur peut rétroagir sur le couple plante pathogène, via des opérations culturales.

Encadré 3 – Le système de production agricole

proche est purement préventive et l'objectif est de n'avoir aucune présence de maladie. *NB.* ceci n'est pas vrai pour toutes les maladies et toutes les cultures, mais s'applique fréquemment dans le cadre de la lutte contre les maladies cryptogamiques de la vigne.

L'approche systématique de la protection des cultures, diffère donc de la figure 3, en ceci qu'il n'y a pas de retour d'informations sur ce système. Par contre, dans le cadre de la protection dite raisonnée, une forme de prise en compte de l'épidémie est introduite notamment par l'utilisation des avertissements agricoles. Il est toutefois difficile de parler de contrôle, car la prise en compte de l'état du système (parcelle ou groupe de parcelles) reste très indirecte.

On ne pourra pourtant véritablement parler de l'usage d'un *contrôleur* sur le système de production que dans le cadre de la PIC (cf. section 1.2). Celle-ci peut notamment prendre la forme de la lutte biologique. Le contrôle des épidémies se alors fait en continu principalement par délégation de l'activité de contrôle à des auxiliaires biologiques (insectes, virus. . .) dont les populations doivent s'équilibrer avec celles des pathogènes (on parle alors de contrôle « feedback » puisque c'est l'état des populations de pathogènes qui détermine la dynamique des populations d'auxiliaires). Dans ce cas l'agriculteur se borne à un rôle de supervision, n'intervenant qu'en cas d'échec du contrôle biologique. L'agriculteur peut aussi, plus ponctuellement, intervenir par l'application de traitements phytosanitaires pour réduire[b] les populations de pathogènes ou par la réintroduction de populations d'auxiliaires si celles-ci sont trop faibles (Tang et al., 2005).

Pour autant, la stratégie de lutte biologique n'est pas toujours possible comme on l'a expliqué précédemment (voir chap. 1 section 1.2.2). Il convient alors de contrôler les

b. Dans le cas de maladies fongiques il s'agit plutôt d'en ralentir le développement que de réduire l'infestation.

épidémies par une action directe de l'agriculteur sur son système de production. Dans l'idéal l'information serait, en référence à la théorie économique néoclassique, pure et parfaite, et le contrôleur recevrait en permanence de l'information en provenance à la fois du système de production et de l'environnement (cf. fig. 3 les flèches provenant de l'environnement, des plantes et des pathogènes sont les données d'entrées du contrôleur). Toujours dans l'idéal, le contrôle se ferait simplement grâce à une série de seuils d'intervention permettant de positionner les interventions de manière optimale et les revenus seraient ainsi garantis pour un niveau de risque accepté.

Lorsque l'agriculteur agit directement, les décisions d'intervention et les interventions elles-mêmes sont des *évènements ponctuels* (cf. fig. 3 flèche en retour du contrôleur au système contrôlé). Pour autant, on a défini plus haut que le système contrôlé suit une dynamique continue que l'on est capable d'observer parfaitement : l'information est donc elle aussi continue. Il faut donc conclure à l'association de phénomènes continus et discrets dans le couple *système + contrôleur*, donc à un système hybride. Cependant, ce cadre idéal d'information n'est pas réaliste. En effet, l'information en entrée a un coût d'acquisition pour l'agriculteur et ne peut être obtenue que de manière ponctuelle. Entre ces instants d'observation, le contrôleur est « aveugle » quant à l'état du système contrôlé. En outre, nous ne disposons pas de modèles continus décrivant correctement le pathosystème pour ceux que nous traitons dans cette thèse. En ceci, notre étude diffère grandement des modèles de supervision hybride décrits par (Lunze, 2000).

Dans notre problématique, le contrôleur ne communique avec l'environnement et les systèmes de production que par des évènements, aussi bien pour ses entrées que pour ses sorties. Nous avons choisi le formalisme des Systèmes à Évènements Discrets pour le représenter

. Notre travail consiste à modéliser formellement le Processus Opérationnel de Décision (POD) présenté au chapitre 1 pour le contrôle de l'oïdium et du mildiou de la vigne, dans le cadre formel des SED.

2.2 Cybernétique, contrôle et modélisation d'expertise

Après avoir introduit par un exemple les notions importantes de la théorie du contrôle et des SED, il s'agit à présent de positionner mon travail dans le contexte de la commande des systèmes. Dans cette section, on présente les concepts de systémique et de théorie du contrôle permettant de raisonner sur la nature du travail de formalisation d'un POD conçu à base d'expertise.

2.2.1 Système

Le dictionnaire en ligne « le trésor de la langue française informatisé » (TLFI, 2008) *in* (Grastien, 2005, p.5) définit un système comme :

> [. . .] un ensemble dont les parties sont interconnectées et échangent [. . .] de la matière, de l'énergie ou de l'information (définition C.1).
>
> in : Grastien (2005, p.5)

Cette définition décrit bien la nature d'un système mais n'aborde pas un aspect important : l'ambiguïté de la notion de système. Cette ambiguïté est mise en exergue par Jean Ladrière dans l'article pour l'encyclopédie universalis (Ladrière, 2008, p.1029-1030) :

> La notion de système apparaît dans deux catégories de contexte fort différentes : d'une part, lorsqu'il est question de propositions (dans lesquelles sont exprimées des relations formelles ou des conceptions relatives à la réalité), d'autre part, dans des contextes où interviennent des entités d'une certaine espèce (par exemple, des corps matériels ou des organismes vivants), dont on étudie la structure et l'évolution. Dans les contextes du premier type, « système » est à peu près synonyme de « théorie » (du moins si l'on prend ce dernier terme en un sens très général). Dans ceux du second type, les considérations qui mettent en jeu la notion de système ont été généralisées dans le cadre de la théorie des systèmes [. . .] [Cette théorie] s'efforce d'établir le cadre le plus général à l'intérieur duquel on peut étudier le comportement d'une entité complexe analysable, c'est à dire son évolution au cours du temps.
>
> in : Ladrière (2008, p.1029-1030)

Cet extrait met en lumière deux points importants. (i) Les systèmes dynamiques ont un comportement analysable dans le temps ; temps qui peut être défini continu ou discret. L'existence d'un temps discret introduit la notion d'observateurs procédant à un échantillonnage. On questionnera la présence de l'observateur d'un système après avoir illustré (ii) le fait qu'il existe des systèmes réels (physique) qu'on associe à la définition de Grastien (2005) et des systèmes dits formels (systèmes logico-mathématiques) qui sont des oeuvres de l'esprit possédant une cohérence interne. Ces derniers peuvent servir de représentation intelligible des premiers. C'est dans ce contexte de modélisation d'un système réel par un système formel que se pose la question de la finalité du modèle.

Un système est généralement étudié comme une entité dans la mesure où son comportement ne peut être expliqué par l'analyse du comportement de ses composantes. C'est pour répondre à cette lacune de l'approche analytique que fut introduite l'approche cybernétique.

> En proposant, dès 1943, [. . .] de reconnaître la légitimité scientifique de l'interprétation téléologique *a priori* du comportement d'un système, Wiener autorisait une voie alternative à toute entreprise de modélisation scientifique ; au lieu de chercher d'abord les causes (mécaniques), le modélisateur était invité à s'interroger d'abord sur les finalités ou les projets du système étudié : en mettant en correspondance intelligible les comportements du système avec sa ou ses finalités, le modèle permet de le décrire effectivement, par simulation d'une boîte noire, par isomorphie fonctionnelle. L'ignorance des causes mécaniques ne constitue plus l'obstacle infranchissable à la modélisation intelligible du phénomène considéré.
>
> in : Lemoigne (2008, p.1033)

Ainsi cette conception de la modélisation place le modélisateur au centre de l'arène, avec non plus comme mission unique d'en comprendre les mécanismes internes, mais d'étudier les propriétés (par exemple stabilité, périodicité, atteignabilité) du comportement du système vu comme une boîte noire ayant une finalité. Il s'agit de définir ses

variables d'entrée et de sortie et d'en construire un modèle mathématique simulant le comportement (Klein, 2005).

Définition 4. *Un système est un objet d'étude défini par rapport à un environnement composé d'éléments (ses composants) interconnectés, discernables ou non, réels ou abstraits. Le système possède des propriétés et un comportement dynamique propre. Les interactions du système avec l'environnement (*système ouvert*) et les évolutions de ses composants (*système complexe*) conditionnent son comportement.*

2.2.2 Etat

Dans le cadre d'un système vu comme une boîte noire, c'est à dire dont on ignore la mécanique (physiques, biologique,. . .) interne du comportement, la notion d'état permet d'identifier la situation du système à un instant donné.

Précisons ce qu'est un état. On a défini que le système pouvait être en interaction avec son environnement c'est à dire recevoir ou émettre de l'information. On a également précisé que le système était constitué de composants. Un composant doit être compris comme un sous-système lui-même éventuellement composé de ses propres sous-systèmes. Bien qu'inobservables sous hypothèse de la boîte noire, les composants ont des comportements qui s'agrègent et donnent lieu à une évolution endogène de l'état (global) du système. Formellement, l'espace d'états X du système S est défini comme suit :

Définition 5. *Soit $S = \{\Sigma, O, I\}$ un système avec :*
Σ : l'ensemble des entrées,
O : l'ensemble des sorties,
I : l'ensemble des vecteurs $\overrightarrow{v}$ de dimension n des états des n composants de S où, $\overrightarrow{v}$ représente une combinaison de l'état courant de chaque composant et $n \in \mathbb{N}$

On définit alors $\eta_{nat} = (T, C)$ l'ensemble des états naturels de S où, $C : \Sigma \times O \times I$ est l'ensemble des configurations possibles et T l'ensemble des instants.

Pour faciliter la manipulation du concept d'état, on préférera X l'espace symbolique des états issus de la projection P: $T \times C \longrightarrow X$ qui à chaque couple $(t, c_i) \in \eta_{nat}$ associe un élément de $x_i \in X$ tel que $P(t, c_i) \mapsto x_i$.

Remarque : *la définition 5 est récursive puisque l'état du système dépend de celui de ses n composants si $n > 0$.* Un état x_s est donc « équivalent » à un t-uplet $(t, \sigma, o, \overrightarrow{v_t})$ avec

$t \in T$, $s \in \Sigma$, $o \in O, \overrightarrow{v_t} \in I_t$, I_t une cellule de la partition I/T. Précisons que cette définition reconnait la possibilité d'une absence d'information que l'on notera ϵ avec $\epsilon \in (\Sigma \cap O)$. Ici *information* prend un sens très large puisqu'elle désigne la nature des éléments de Σ et O. Cela peut être un évènement, la valeur d'une variable, de l'énergie ou encore une voiture entrant dans le système parking. . .

2.2.3 Modélisation

Les définitions données ci-dessus envisagent le système et ses états comme des entités « en soi ». Pour autant, lorsque le système atteint un certain degré de complexité,

il s'agit de pouvoir *l'observer* pour pouvoir le connaître. Par connaître, on entend organiser et représenter les faits de manière cohérente. Dans l'approche cybernétique, cette connaissance est complète lorsqu'on est capable de construire un modèle permettant de *simuler* le *comportement* du système.

Définition 6. *Le comportement d'un système S, noté*[c] BEH_S*, est l'ensemble des suites d'états* θ*, que le système* peut *effectivement visiter.*

C'est l'ensemble des branches d'un arbre partant de l'état initial $x_0 = P(c_{ini}, \min(T))$*, dont chaque noeud est un état* $x_k \in X$ *et dont les feuilles terminales sont les états* $x_{fol} = P(c_{max(T)}, \max(T))$ *avec* $\min(T)$, $\max(T)$ *respectivement le début du temps et la fin du temps (du système) ; et* c_{ini} *l'unique configuration initiale du système et* $c_{max(T)}$ *une configuration possible à* $\max(T)$*. Cette définition est librement inspirée de (Klein, 2005, p.24).*

Au début de l'étude d'un système totalement inconnu (par exemple une nouvelle maladie), on ne sait pas déterminer dans l'environnement les éléments qui sont en entrée, on ne sait également rien des interactions au sein du système qui génèrent les sorties. On peut faire l'hypothèse raisonnable qu'un certain nombre de sorties sont observés, ce qui motive l'étude du système. Pour illustrer notre propos, prenons l'exemple d'une nouvelle maladie (un nouveau pathosystème), il faut d'abord identifier les dégâts qu'elle cause (1^{er} type de sortie), puis on apprend à identifier des symptômes (2^{nd} type de sortie)... Un système aussi incomplètement défini n'est pas simulable. Les étapes préliminaires à la modélisation consistent donc à identifier les entrées et sorties pertinentes, les moyens de les évaluer ainsi que l'opportunité d'étudier les composants du système.

Admettons à présent que S soit une boîte grise c'est à dire que sa composition interne ne soit que partiellement comprise : *tous ses composants (leurs comportements, leurs interactions) ne sont pas bien connus* ; ou leurs états ne peuvent être *observés à chaque instant*. En pratique on est souvent confronté aux deux limitations en même temps. D'ailleurs la limite concernant « l'observabilité » à chaque instant peut dans bien des cas être étendue aux informations en entrée et en sortie. De telles limitations imposent une restriction sur l'espace d'états. On n'a en pratique accès qu'à X_{obs} l'espace des états observés.

Définition 7. *A partir de* $T_{obs} \subseteq T$ *les instants d'observation et* $C_{obs} \subseteq C$ *les configurations observées de S. On définit* $X_{obs} = P(T_{obs}, C_{obs})$ *l'espace des états observés avec* $X_{obs} \subseteq X$*.*

Remarque : *L'ensemble des états non-observés* $\overline{X}_{obs}$*, englobe aussi bien les états qui n'appartiennent pas au comportement* BEH_S *(définition 6), que ceux qui n'ont pas été vus faute d'avoir été observés à l'instant de leur occurrence, ou encore les états dont la configuration a été mal interprétée (par exemple si une variable n'est pas mesurée). Une description peut être incomplète par défaut de connaissance ou par impossibilité d'en faire la mesure.*

Remarque : *La définition ci-dessus des " états observés " ne doit pas être confondue avec la notion d'observabilité d'un système qui est évoquée plus loin et concerne les séquences d'état.*

c. *BEH* pour « behaviour ».

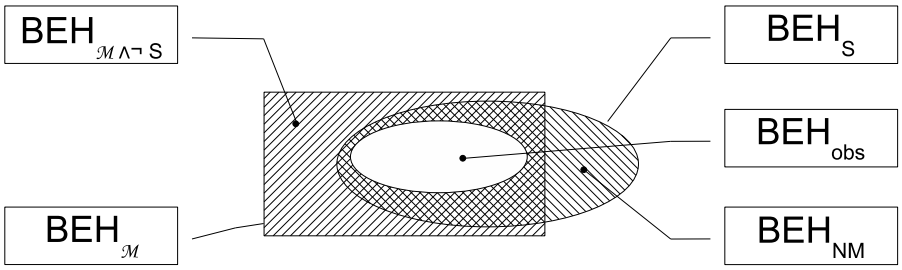
A partir des états observés, le travail du modélisateur est donc de construire un modèle, c'est à dire une carte mettant en relation cohérente les états observés afin de simuler les évolutions du système de manière aussi complète que possible.

Définition 8. *Le modèle* $\mathcal{M} = (X_{\mathcal{M}}, R)$ *avec* $R \subseteq X \times X_{\mathcal{M}}$ *une relation totale construite à partir des observations telle que pour chaque* $x \in X$ *les états suivants possibles sont donnés par R. On a :* $X_{obs} \subseteq X_{\mathcal{M}} \subseteq X$ *avec* X_{obs} *l'ensemble des états observés du système et* $X_{\mathcal{M}}$ *le sous-ensemble de X que R permet d'atteindre à partir de* X_{obs}*.*

On dit que $\mathcal{M}$ modélise le système S : noté $\mathcal{M} \models S$, si $\mathcal{M}$ simule S. Un modèle simule un système si à partir d'une même configuration initiale, le modèle et le système parcourent la même suite d'états, lorsqu'ils sont soumis aux mêmes entrées. Formellement on a :

Définition 9. *On dira que le modèle* $\mathcal{M}$ simule *le système S si le comportement* $BEH_S \subseteq BEH_{\mathcal{M}}$ *avec* BEH_S *l'ensemble des comportements de S (voir définition 6) et* $BEH_{\mathcal{M}}$ *l'ensemble des comportements du modèle.*

Rappelons que BEH_S n'est a priori pas observable. Le modélisateur ne dispose en réalité que de BEH_{obs}. La difficulté de la modélisation consiste comme le montre la figure 2.2 à simuler si possible l'intégralité de BEH_S par $BEH_{\mathcal{M}}$ avec, selon les objectifs de la modélisation, parfois la nécessité de minimiser $BEH_{\mathcal{M}\wedge\neg S}$ la part du comportement du modèle qui ne correspond pas au comportement du système.



source : Klein (2005, p.25)

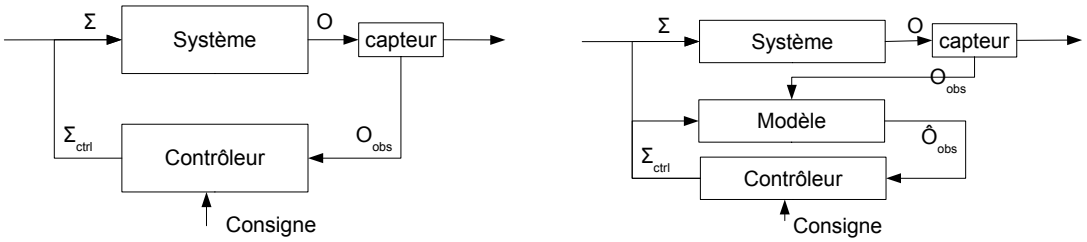
BEH_S	le comportement du système S.	BEH_{obs}	le comportement observé de S.
BEH_{NM}	le comportement de S Non Modélisé.		
$BEH_{\mathcal{M}}$	le comportement du modèle $\mathcal{M}$	$BEH_{\mathcal{M}\wedge\neg S}$	le comportement de $\mathcal{M}$ ne correspondant pas à S.

FIGURE 2.2 – Relations entre le comportement complet du Système, le comportement observé et celui du modèle.

2.2.4 Contrôle

Introduisons à présent, la théorie du contrôle [d]. Son objectif est d'influencer l'état d'un système par des commandes/contrôles de façon à ce que le système conserve des propriétés souhaitées grâce à un retour d'information sur l'état du système (on parle de *feed-back* Macki and Strauss, 1982). Il peut également s'agir d'influer sur le système

pour lui faire acquérir ces propriétés (*feed-forward*). La figure 2.3 illustre ces deux types de boucles de contrôle.



(a) un système contrôlé par « feed-back » (b) un système contrôlé par « feed-forward »

FIGURE 2.3 – Deux types de contrôleur en boucle fermée avec le système

2.2.4.1 Synthèse idéale de contrôleur

Dans le cadre de modélisation que nous avons défini plus haut, le principe de la synthèse de contrôleur est de spécifier une propriété sur le système S, de produire une partition de l'espace d'état selon que la propriété est réalisée ou non. Il s'agit ensuite de définir un système contrôleur C tel que le système couplé au contrôleur : SC, ne puisse atteindre que les états satisfaisants la propriété. On notera $\mathcal{T}$ (pour target : objectif) le sous-ensemble de X pour lequel la propriété désirée est valide et on appelle comportement spécifié (BEH_{spec}), le comportement dont toutes les suites d'états θ appartiennent à $\mathcal{T}$ **Remarque :** *Rappelons que X est un couple : (une configuration C et un instant t)*

aussi si la consigne était que le système atteigne un régime stable en 5 seconde, tous les états où le temps est inférieur à 5 $X_{<5}$ seraient des éléments de $\mathcal{T}$ c'est à dire $X_{<5} \subset \mathcal{T}$.

$$\text{On génère } \mathcal{M}_C \text{ tel que } \mathcal{M}_S \wedge \mathcal{M}_C = \mathcal{M}_{SC} \models \mathcal{M}_{spec}$$

$$\begin{array}{rrl} \text{avec} & \mathcal{M}_{spec} & \models BEH_{spec}, \\ & \mathcal{M}_S & \models BEH_S \\ \text{et} & \mathcal{M}_C & \models BEH_C, \\ \text{où} & BEH_{spec} & = \{\theta | \forall\, x \in \theta, x \in \mathcal{T} \wedge \theta \in BEH_S\} \\ \text{et} & BEH_{SC} & = \{\theta | \theta \in BEH_C \wedge BEH_S\} \end{array}$$

La difficulté du couplage se situe premièrement dans la question de savoir s'il est possible de contrôler le système et par quel biais, mais également de trouver le modèle de C qui permette de garantir que le système satisfait à la *spécification* également appelée la *consigne*.

En théorie du contrôle, la notion d'observabilité concerne la capacité à reconstruire le comportement du système sur une période donnée, à partir d'observations sur les entrées et sorties sur cette période. Voir dans le cas continue discrétisé (Borne, 1993) et pour celui des SED (Lin and Wonham, 1988) La notion duale de l'observabilité est la contrôlabilité (commandabilité). Il s'agit donc intuitivement de la capacité à mener un système dans un état souhaité, à partir d'un état initial, en agissant sur certaines entrées, ceci en un temps finis.

d. aussi appelée théorie de la commande en Français

2.2.4.2 Synthèse de contrôleur en pratique

En pratique, cette définition est souvent suffisante si l'état initial appartient à C. En boucle fermée (cf. fig. 2.3) le contrôle peut être synthétisé si le système est observable et contrôlable à partir de l'état initial. Le contrôleur C prend alors ses entrées dans O_{obs} les sorties observées et produit ses sorties appelées contrôle dans Σ_{ctrl}.

Il n'est pas forcement nécessaire de disposer d'une représentation parfaite du système pour pouvoir réaliser un contrôleur. En réalité en face d'un objectif, il faut être capable de distinguer l'intégralité des états de $\mathcal{T}$ et surtout les conditions qui conduisent à $\overline{\mathcal{T}}$ l'ensemble des états hors de l'objectif. Il est suffisant d'avoir une modélisation partielle du système permettant l'observation de la perte de la consigne (sortie de $\mathcal{T}$) et explicitant les moyens d'action permettant de restaurer cette consigne c'est à dire $x_{\overline{\mathcal{T}}} \succ_{\sigma_{ctrl}} x_{\mathcal{T}}$ pour pouvoir contrôler le système.

Dans le monde réel le contrôle est *contraint*. D'une part, les contraintes physiques du système, ne permettent pas de réaliser tous les contrôles mathématiquement possibles sans mettre en péril le système. D'autre part, les dérives (le fait de sortir de l'objectif) prennent en général du temps avant de se répercuter sur les sorties. Dans ce cas, si la consigne est impérative, il faut pouvoir anticiper, à partir des entrées que le système a subies, l'évolution future. C'est dans ce cas que l'on met en oeuvre l'approche feed-forward qui utilise un modèle d'évolution du système afin de permettre au contrôleur de corriger une dérive avant qu'elle ne se produise (cf. fig. 2.3(b)).

Il peut également être nécessaire de mettre en oeuvre un modèle d'évolution du système dans la boucle lorsque la capture des sorties est, soit impossible, soit coûteuse et que le pas de temps entre deux mesures est supérieur à celui de la dérive. Dans ce cas un contrôle plus fréquent que la mesure est nécessaire. Il faut noter que si le système est contrôlable sur une large plage d'états autour de $\mathcal{T}$ et que l'on peut autoriser et corriger ces dérives, alors l'emploi en feed-forward d'un modèle ne se justifie pas.

On a présenté ci-avant, les notions de systémique et de théorie du contrôle indispensables à la compréhension du problème de contrôle à traiter et au choix des outils efficaces pour valider le modèle de POD.

2.2.5 Une approche du contrôle à base d'expertise

On positionnera à présent ce travail de modélisation et de conception de POD dans le cadre de la théorie du contrôle et on expliquera le recours à l'expertise.

Le problème de protection du vignoble contre le mildiou et l'oïdium présente plusieurs difficultés en terme de contrôle. La dérive du système de production ne révèle ses effets que tardivement si on considère la présence de symptômes comme marqueur d'un niveau d'épidémie donné. En effet entre l'instant d'une contamination et le moment où cette contamination est perceptible, il y a incubation pendant environ une semaine. Ce décalage entre l'évènement à contrôler et le moment où cet évènement devient perceptible empêche un contrôle en feed-back au sens strict . En outre la prise de mesure (évaluation du niveau d'épidémie) ne peut être automatisée à l'heure actuelle, ce qui la rend extrêmement coûteuse.

Par ailleurs une dérive par rapport à la trajectoire idéale (absence de maladie) est difficile à corriger. Au mieux il est possible de stabiliser cette dérive (bloquer l'épidémie à un niveau donné). Cette irréversibilité des dégâts impose une intervention préventive avant la contamination, ceci d'autant plus qu'on ne sait pas, à l'heure actuelle, estimer les conséquences d'une dérive (c'est à dire faire le lien entre un niveau d'épidémie et les pertes engendrées). Dit autrement, les seuils d'interventions permettant d'éviter les états interdits ne sont pas définis.

Dans ce contexte, si les objectifs de production sont :« aucun symptôme et aucun dégât », la pratique traditionnelle consistant à traiter tout au long de la saison en prévention d'une éventuelle contamination, est relativement rationnelle. Néanmoins aujourd'hui, la consigne de ce contrôleur doit évoluer (plan Ecophyto 2018, Paillotin, 2008). L'objectif fixé par Santé végétale est de n'avoir aucune perte économique, d'autoriser quelques symptômes et ceci avec le moins d'opérations de contrôle possible .

Dans l'idéal, le contrôle des épidémies le plus économe en traitement [e] s'effectuerait au niveau de chaque plante : cela nécessiterait de disposer d'un modèle épidémiologique valide qui permettrait, en fonction des conditions climatiques et micro-climatiques du cep ainsi que du niveau actuel d'infestation, de décider d'un traitement du cep sur la base (i) de prévisions d'évolution des conditions climatiques (ii), des niveaux d'infestation qu'engendrent ces prévisions et (iii) de fonctions de dégâts permettant à chaque instant de relier un niveau d'infestation aux pertes attendues. Ce contrôle idéal prendrait schématiquement la forme présentée en fig. 2.3(b).

Pourtant, force est de constater qu'à l'heure actuelle les moyens techniques ne permettent pas de mettre en oeuvre ce type de solution. Les modèles utilisés pour le conseil sont des modèles stochastiques prenant en entrée des données climatiques (par exemple les modèles du SRPV Rouzet and Jacquin, 2003). Cependant, ils sont calibrés et calculés à une échelle « macroscopique » (« petit région ») qui ne rend pas rend pas compte nécessairement de l'état d'un système de production particulier (la parcelle).

L'expérience des phytopathologistes de Bordeaux est particulièrement édifiante à ce sujet. Les parcelles sur lesquelles les expérimentations ont été menées de 2001 à 2006 se situent toutes sur des sites de l'INRA dotés de stations météorologiques permettant de calculer les sorties pour chacun des modèles évoqués. Il est pourtant arrivé à plusieurs reprises que les modèles soient alarmistes alors qu'aucune épidémie n'était constatée (le cas inverse est probablement possible bien que cela ne nous ait pas été rapporté)

Les modèles de prévisions doivent donc être considérés comme trop pessimistes par rapport à une consigne, qui vise des objectifs de production exigeants en terme de qualité et de quantité, mais ne doit pas effectuer d'opérations de traitement inutiles.

Pour cela, les phytopathologistes ont mobilisé les concepts de dégâts, de dommages et (dans une moindre mesure) de pertes (présentés au chapitre 1, section 1.2.1). Ils se

e. Nous sommes bien conscient que la mise au point de plantes résistantes (hybride ou OGM) ou la découverte d'agents (insectes, microbes...) permettant un contrôle biologique efficace, seraient des solutions ayant un impact plus direct sur le problème des pesticides que nos propositions, au moins en terme de pollution. Pour autant, aucune de ces possibilités n'est d'actualité.

sont inspirés des principes de la PIC : « il n'est pas nécessaire de chercher à éradiquer un pathogène (c'est d'ailleurs souvent illusoire), mieux vaut le contenir en dessous d'un seuil où celui-ci ne cause pas de perte de revenu ». Une approche à base d'expertise a donc été adoptée.

Pour la conception, ils utilisent leurs expertises pratiques et scientifiques comme s'il s'agissait d'un modèle *feed forward* du comportement du système et produisent à partir d'une situation donnée, une projection raisonnable de dommages à la récolte. Le résultat de ces projections sert *ex ante* à construire une partition qualitative de l'espace d'état du pathosystème en fonction des risques de perte estimés. A partir de cette partition, ils ont construit les bases du système de décision que nous avons modélisé. Le modèle encapsule *in fine* (i) l'expertise permettant de faire les projections et (ii) les actions à effectuer pour ne par atteindre les états dangereux. La figure 2.4 représente schématiquement la manière dont le POD Mildium permet de contrôler le pathosystème viticole.

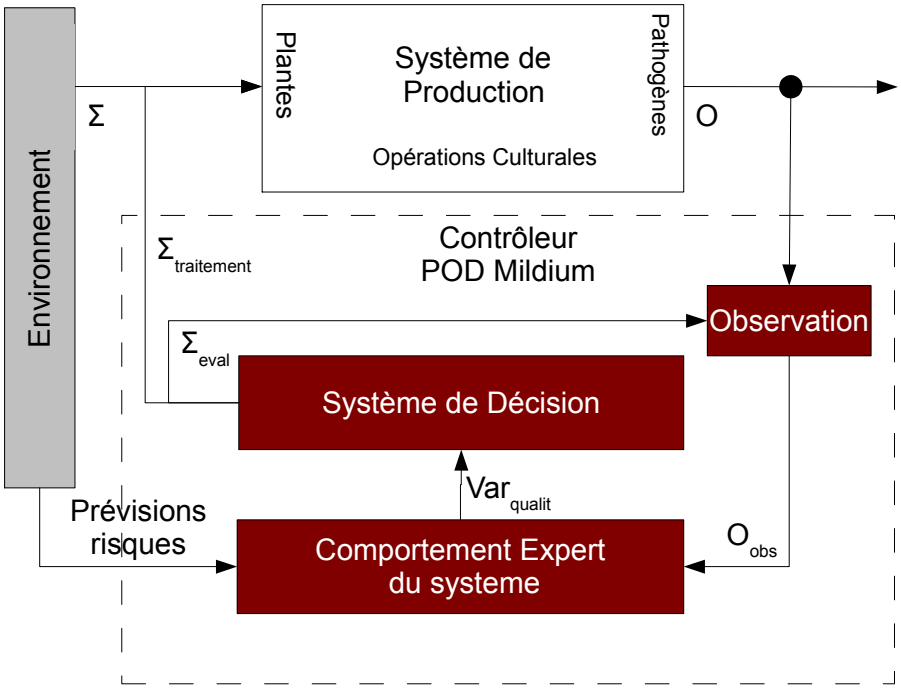


FIGURE 2.4 – Le problème de contrôle du mildiou et de l'oïdium tel que résolu par le POD Mildium.

Précisons le but des expérimentations de plein champ. On l'a vu, l'expertise est utilisée pour construire une partition des situations épidémiques en fonction du risque de dommage à la récolte. Cette partition est opérationnelle au sens où elle vise à prendre des décisions de contrôle sur le système. Il s'agit donc d'une partition qui (en simplifiant le raisonnement) établit *in fine* des seuils d'intervention, sur la base d'une fonction de dommage experte. Ces seuils restent à affiner et à valider par l'expérimentation.

En outre le positionnement de ces mesures est déclenché par le système de contrôle (voir fig. 2.4) sur la base d'évaluation de la phénologie (nous y reviendrons au chapitre 5) et du découpage en séquence du POD. On a donc une estimation des risques de dommages qui est fortement liée au contrôleur lui-même.

Notre modélisation doit donc représenter le système de décision expert. On verra au

chapitre 6 que notre travail de validation du modèle a consisté à vérifier que $BEH_{Experts} \subseteq BEH_{Mildium}$ de sorte que $\mathcal{MODÈLE}_{Mildium} \models Experts_{Mildium}$, c'est à dire à valider que le modèle formel du POD Mildium permet bien de reproduire les décisions du « POD Mildium expert » mises en œuvre par les experts au cours des expérimentations. Dans la suite et jusqu'au chapitre 6, on ne fera pas la distinction entre le « modèle du POD » et le POD expert..

2.2.6 Conclusions

Avant de poursuivre, revenons sur la distinction système réel/système formel. On a montré plus haut que notre travail se pose comme un problème de contrôle discret sur un système hybride. On a montré pourquoi l'approche par synthèse optimale du contrôleur était impraticable, justifiant l'utilisation de l'expertise, pour aboutir à un « système contrôleur » opérationnel sur le système de production viticole. Ainsi, le contrôleur est le système que nous étudions.

On précisait au début de cette section qu'un système peut être un objet logico-mathématique possédant une cohérence interne que l'on pourrait appeler un modèle. L'un des objectifs principaux de notre travail est de construire ce système par l'explicitation et la mise en cohérence des connaissances expertes.

A partir d'un protocole de décision informel et de règles de décision spécifiées de manière incomplète, il s'agit de réaliser un modèle capable de traiter l'ensemble des cas de figure pouvant se réaliser dans l'espace d'états du système de production exprimé par l'expertise. Pour réaliser ce « système contrôleur » (on parlera par la suite simplement du système), on utilise le formalisme des SED qui est bien adapté au type de commande binaire qu'est la décision d'une application phytosanitaire.

2.3 Fondements théoriques des systèmes à évènements discrets

> [Selon Ramadge and Wonham (1989)], un Système à Évènements Discrets (SED) est un système dynamique qui évolue conformément à la brusque occurrence d'évènements physiques, selon des intervalles irréguliers potentiellement inconnus.
>
> in : Ramadge and Wonham (1989)

Prenant un point de vue de modélisateur, ces auteurs précisent :

> Un SED [est] doté d'un espace d'états discrets, dont les trajectoires d'états sont constantes par morceaux ; les instants auxquels les transitions d'un état à un autre ont lieu ainsi que les transitions elles-mêmes sont en général imprévisibles. [. . .] Les transitions d'états d'un SED sont appelées évènements. [. . .]
>
> Une hypothèse simplificatrice communément faite est d'ignorer les instants d'occurrence des évènements et de considérer seulement l'ordre dans lequel ils ont lieu. Cela donne lieu à ce que l'on appelle un *modèle SED logique*. [Par la suite on dira SED pour SED logique]
>
> in : Ramadge and Wonham (1989)

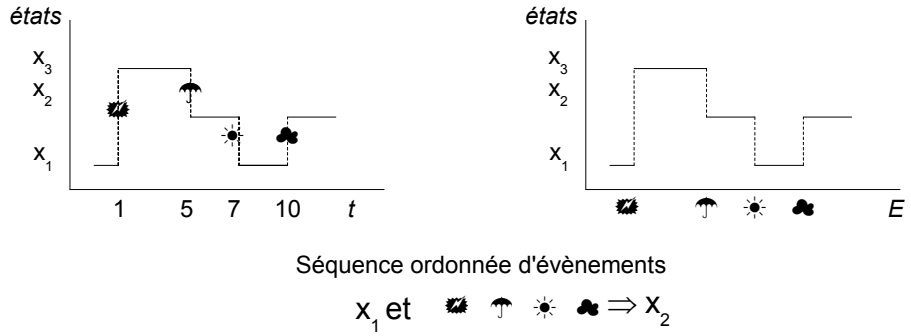


FIGURE 2.5 – L'évolution de l'état d'un SED dans le temps se résume par une séquence d'évènements à partir d'un état initial.

Ainsi, le comportement d'un SED est fondamentalement régi par des évènements (fig. 2.5) c'est à dire que tant qu'aucun évènement n'a eu lieu, son état reste inchangé, à la différence des systèmes continus (pour un SED, « le temps ne change rien à l'affaire »).

On considère ici qu'un évènement est un message, une information, qui peut éventuellement entraîner une évolution de l'état du système. Un évènement n'a pas de durée. Dans le cadre de la théorie linguistique des automates on fait l'hypothèse qu'il toujours possible d'ordonner deux évènements[f]. Tout se passe (se calcule) *comme si* il y avait toujours un intervalle de temps $\delta t > 0$, même infime, entre deux évènements $\lightning_1$ à l'instant t_1 et $\lightning_2$ à l'instant t_2, tels que $t_2 - t_1 \geq \delta t$. Cette hypothèse permet de construire des séquences d'évènements en dehors de toute référence au temps. Faire abstraction du temps (comme quantification de la durée) est pratique d'un point de vue théorique, mais ne permet pas de représenter tous les problèmes, aussi le temps peut il être réintroduit dans le cadre des SED par des formalismes dit temporisés (automates temporisés (Alur and Dill, 1994) par exemple). Précisons encore, que toutes les occurrences d'un même type d'évènement ne causent pas nécessairement de transition d'un état vers un autre, mais qu'un même type d'évènement peut être la cause de diverses transitions.

Finalement, on doit introduire la distinction entre les *évènements exogènes*, provenant de l'environnement du système et les *évènements endogènes* générés par le système lui-même. On note E l'ensemble de évènements tel que $\forall \lightning, \ \lightning \in E$. On définit E_{in} le sous-ensemble des évènements exogènes, $E_{in} \subseteq \Sigma$ (Σ est l'ensemble défini à la def. 5) et on définit E_{out} où $E_{out} \subseteq O$ (def. 5) l'ensemble des évènements endogènes émis par le système. Si le système est un pur SED, on a $E \supseteq \Sigma \cup O$, c'est à dire que l'état du système n'est déterminé que par des évènements en entrée ou en sortie.

Cette représentation d'un système dont les états ne seraient régis que par les évènements en entrée et en sortie, n'est valide vis à vis de la définition du système : $S = (\Sigma, I, O)$, que si les évènements ne sont pas dirigés d'une origine vers une destination déterminée. S'ils peuvent agir à la fois sur les composants du système tout en étant perçus de l'extérieur du système (on parle de *broadcast*, diffusion en Français) alors $S = (\Sigma, I, O)$ est cohérent avec $E \supseteq \Sigma \cup O$ dans le cadre des SED. Dans de nombreux cadres d'analyse, les évènements ont une origine et une destination, ce qui est intéressant pour étudier le

f. Toutefois d'autres cadres sémantiques SED modélisent les événements synchrones.

comportement des canaux de communication ou formaliser les systèmes concurrents et asynchrones. On ne présente pas ces cas de figure puisque, pour notre travail, on utilise des évènements diffusés à travers tout le système.

La notion d'évènement permet donc d'identifier les instants d'évolution de l'état du système et peut aussi être associé à la notion d'entrée et de sortie d'un SED. En fait, ce qui importe pour mon travail, c'est que les évènements sortant soient aussi perçus par les composants. Ce qui est le cas on le verra pour les Statecharts (voir section 2.6).

2.4 Automates

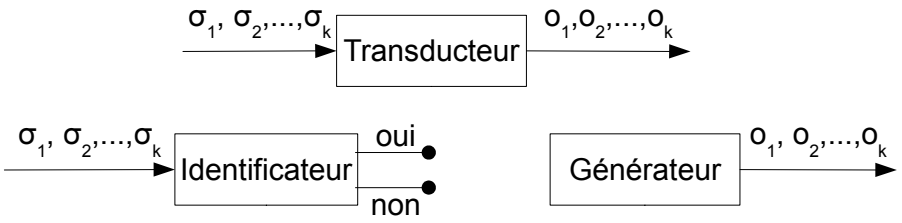
On présente dans cette section quelques éléments théoriques sur les automates qui permettent d'appréhender les fondements de l'analyse formelle des Statecharts. Il nous semble utile de préciser que ces notions ne sont pas indispensables à la lecture intuitive du Statechart du POD Mildium (présenté au chapitre 5).

Trois types de SED sont à distinguer, chacun donnant lieu à une structure d'automate différente (fig. 2.6).

Les *identificateurs* (voir section 2.4.1) résolvent un problème de décision en permettant d'identifier si une chaîne de caractère appartient au langage défini par l'accepteur.

Les *générateurs* sont des modèles de SED ne recevant aucune entrée et qui produisent en sortie les mots des langages qu'ils définissent. Ils sont notamment utilisés dans la théorie du contrôle des SED (théorie RW *in* Ramadge and Wonham, 1989; Wonham, 2008), pour modéliser le comportement du système à contrôler. Pour le contrôle on adjoint au générateur un contrôleur, qu'il s'agit de définir, dont l'objectif est d'interdire au générateur d'atteindre les états dangereux. Le contrôleur y parvient en empêchant l'occurrence des évènements (parmi les évènements contrôlables) qui permettraient au générateur d'atteindre les états interdits. Cela signifie par exemple que le contrôleur devra générer un évènement qui mette la machine dans l'état maintenance avant que l'évènement « casse » ne la mette dans l'état « en panne ».

Les *transducteurs* réagissent à un flux d'entrée qui entraînent le changement de leurs états en conséquence de quoi ils émettent des symboles en sortie. Il existe deux types d'automates pour représenter les transducteurs respectivement appelés machine de *Moore* et machine de *Mealy* (voir section 2.4.2).



source : Klein (2005)

FIGURE 2.6 – Trois types de SED selon Booth

2.4.1 Identificateur canonique (*canonical recognizer*)

Les définitions qui suivent sont tirées principalement de Wonham (2008).

Un SED dans le contexte de la boîte noire peut être vu comme une suite d'évènements en entrée et une suite d'évènements en sortie. La théorie des langages est souvent utilisée pour modéliser les SED. Pour cela, il suffit d'associer à chaque type d'évènement $\lightning_i$, un symbole σ_i de l'alphabet $\mathscr{A}$.

On définit donc la fonction $Lab_{ev} : E \longmapsto \mathscr{A}$ tel que $\lightning_i \overset{Lab_{ev}}{\longmapsto} \sigma$ avec $\lightning_i \in E$, E l'ensemble fini des types d'évènements auxquels est soumis le SED S et $\mathscr{A}$ l'ensemble fini des symboles $\sigma, \tau, \ldots$ $\mathscr{A}$ est l'*alphabet* de S.

On définit W^- l'ensemble des *mots* (chaînes de caractères ou chaîne) composés d'une suite finie s de caractères $\sigma_1\sigma_2\cdots\sigma_i\cdots\sigma_k$ avec $k \geq 1$ et $\sigma_i \in \mathscr{A}$; ici ϵ représente le mot vide, $\epsilon \notin \mathscr{A}$. On définit alors

$$W := \{\epsilon\} \cup W^-$$

L'identificateur canonique peut être défini de la manière suivante :

Définition 10. *Un identificateur canonique est un 5-uple*

$$\mathbf{R} = (X, \mathscr{A}, \hat{\xi}, x_0, X_m)$$

où, X est l'ensemble des états de **R**,
$\mathscr{A}$ l'alphabet pour lequel est défini **R**,
$\hat{\xi}$: $X \times W \longrightarrow X$ la fonction de transition qui à chaque état définit ses successeurs,
x_0 un état initial,
X_m l'ensemble des états acceptants ou états finaux.

Cette structure identifie les états successif à partir des symboles « qu'on lui soumet ». On est alors capable de suivre le comportement qu'induit une chaîne s en entrée sur le graphe de l'espace d'état que construit la structure **R**.

Définition 11. *Un Langage L sur l'alphabet $\mathscr{A}$ est un sous ensemble de W. L est un élément de l'ensemble $\emptyset \cup W$ où $\emptyset$ est le langage sans mot.*

Pour les identificateurs canoniques, on a donc $BEH_{\mathbf{R}} = W$ et $BEH_{\mathbf{R_m}}$ le sous-ensemble des comportements de **R** dont les sommets terminaux $x \in x_m$, sont composés de θ_m, où $\theta_m \equiv s$ avec $s \in L$. On dit que **R** définit une *grammaire* de L.

Définition 12. *Soient $\mathbf{R_1} = (X_1, \mathscr{A}_1, \hat{\xi}_1, x_{0_1}, X_{1_m})$ et $\mathbf{R_2} = (X, \mathscr{A}_2, \hat{\xi}_2, x_{0_2}, X_{2_m})$ deux identificateurs. Le produit $\mathbf{R} = \mathbf{R_1} \| \mathbf{R_2}$, en imposant $X_1 \cap X_2 = \emptyset$, est l'identificateur $(X, \mathscr{A}, \hat{\xi}, x_0, X_m)$ avec :*

X *les états résultant du produit cartésien de* X_1 *et* X_2 *c'est à dire* $X = X_1 \times X_2$
$\mathscr{A}$ *l'alphabet est l'union des alphabets* $\mathscr{A}_1$ *et* $\mathscr{A}_2$ *c'est à dire* $\mathscr{A} = \mathscr{A}_1 \cup \mathscr{A}_2$.
$\hat{\xi}$ *l'ensemble des transitions de* **R** *ainsi, avec* $\hat{\xi}_{1_i} = (x_1, s_1, x'_1) \in \hat{\xi}_1$ *et* $\hat{\xi}_{2_i} = (x_2, s_2, x'_2) \in \hat{\xi}_2$ *:*
si $s_1 = s_2$, *alors* $((x_1, x_2), s_1, (x'_1, x'_2)) \in \hat{\xi}$,
si $s_1 \neq s_2$, *alors* $((x_1, x_2), s_1, (x'_1, x_2)) \in \hat{\xi}$ *ainsi que* $((x_1, x_2), s_2, (x_1, x'_2)) \in \hat{\xi}$.
x_0 *l'état initial, est le produit cartésien des deux états initiaux* X_{1_m} *et* X_{2_m} *c'est à dire* $x_0 = (x_{1_0}, x_{2_0})$
X_m *l'ensemble des états acceptants résultant du produit cartésien de* X_{1_m} *et* X_{2_m} *c'est à dire* $X_m = X_{1_m} \times X_{2_m}$

L'identificateur global de deux langages est le produit des identificateurs de chacun des langages le composant (voir def. 12). Cette définition décrit le principe pour la structure la plus simple. Ce principe reste le même pour des structures plus complexes.

2.4.2 Transducteurs

Dans la section ci-dessus, on définissait **R** à partir d'un langage L. Ici, on considère à l'inverse, qu'une grammaire définit un langage. En d'autres termes, c'est le programme qui détermine les séquences d'actions possibles pour la machine. On présente des automates permettant de décrire une grammaire formelle, à la fois en entrée et en sortie.

2.4.2.1 Machines de Moore

Définition 13. *Un automate de Moore* **M** *est un 6-uplet :*

$$\mathbf{M} = (X, \Sigma, \xi, I, F, \mathscr{O}, \lambda)$$

avec :
X *: l'ensemble des états ;*
Σ *: l'alphabet des entrées ;*
ξ *:* $X \times \Sigma \longrightarrow X$*, la fonction de transition ;*
I *: l'ensemble des états initiaux le plus souvent* $I = \{x_0\}$ *;*
F *: l'ensemble des états finaux ou acceptants ou marqués ;*
$\mathscr{O}$ *: l'alphabet des sorties ;*
λ *:* $X \longrightarrow \mathscr{O}$*, la fonction qui associe une sortie à chaque état.*

une machine de *Moore* associe à chaque état, un symbole $o \in \mathscr{O}$ où $\mathscr{O}$ est un alphabet. N.B. $\mathscr{O}$ n'est pas nécessairement différent de Σ. Les symboles $o_1, o_2 \ldots o_n$ associés à chaque état sont déterminés par la fonction λ.

Remarque : *L'identificateur canonique défini par def.10 est un cas particulier de l'automate de Moore avec* $\mathscr{O} = \{0, 1\}$ *et* $\forall x \in X,\ \lambda(x) = 1$ *ssi* $x \in X_m$

Un automate se représente par *un diagramme d'états transition* : un graphe orienté où les sommets sont les états et les arcs les transitions. Prenons l'exemple de **Mo** défini de la manière suivante :

Le *diagramme d'états transitions* (ou simplement diagramme d'états) de la figure 2.7, est équivalent à la définition mathématique donnée à gauche. La flèche pointant sur l'état 1 indique que cet état est l'état initial. Les arcs entre les états 1 et 2 sont les transitions.

Mo = $(X, \Sigma, t, X_0, F, \mathscr{O}, l)$ où

X={1,2}
Σ={σ, κ}
t={[(1,σ),2],[(2,κ),1]}
X_0={1}
F={2}
$\mathscr{O}$={o_1, o_2}
l={(1,o_1),(2,o_2)}

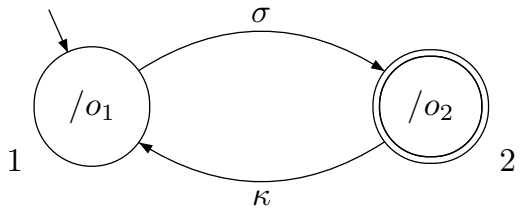


FIGURE 2.7 – Deux représentations de **Mo** un automate de Moore

A la transition est associé un *label* qui indique la condition nécessaire au franchissement de cette transition. Les *labels* respectivement σ et κ permettent d'identifier l'évènement déclenchant la transition. On parle généralement de *trigger* pour cet évènement déclenchant. Cette condition n'est pas suffisante puisqu'il faut que l'état à l'origine de la transition soit actif. L'état 2 est un état marqué (acceptant) qu'on identifie par un double cercle : $2 \in X_m$. Le langage reconnu par **Mo** en entrée est donc σ puis la suite infiniment dénombrable de κ suivie de σ noté $L_\Sigma : \sigma(\kappa\sigma)^*$. Il s'agit bien entendu du langage en entrée sur l'alphabet Σ.

Les automates de Moore sont des transducteurs, ils produisent des symboles (ou des évènements) en fonction de l'état dans lequel ils se trouvent. **Mo** produit donc de manière infiniment dénombrable les mots du langage langage $L_{\mathscr{O}}$ noté $L_{\mathscr{O}} : (o_1 o_2)^*$. Les symboles produits par **Mo** sont inscrits au centre du cercle représentant chacun des états. Ces symboles sont produits à l'instant où l'état devient actif. On aura noté qu'il n'y a pas d'équivalence entre le langage en entrée L_Σ et le langage en sortie $L_{\mathbf{Mo}}$. Bien entendu les alphabets étant disjoints, il ne peut y avoir d'égalité entre les deux mais plus profondément, il n'est pas possible de construire une projection de l'un dans l'autre. Les chaînes de caractères produites sur L_Σ permettent de reconstruire les transitions d'un état actif à l'autre, alors que les chaînes de $L_{\mathbf{Mo}}$, indiquent la succession des états atteints.

2.4.2.2 Machines de Mealy

Pour construire une machine qui informe (c'est-à-dire qui produise une sortie) non pas des états qu'elle atteint, mais de la séquence d'évènements qu'elle subit, il est nécessaire d'introduire à présent la machine de *Mealy*.

Définition 14. *Un automate de Mealy* **Y** *est un 6-uplet :*

$$\mathbf{Y} = (X, \Sigma, \xi, I, F, \mathscr{O}, \tau)$$

avec :

X : *l'ensemble des états ;*
Σ : *l'alphabet des entrées ;*
ξ : $X \times \Sigma \longrightarrow X$, *la fonction de transition ;*
I : *l'ensemble des états initiaux le plus souvent* $I = \{x_0\}$
F : *l'ensemble des états finaux ou acceptants ou marqués*
$\mathcal{O}$: *l'alphabet des sorties*
τ : $X \times \Sigma \longrightarrow \mathcal{O}$, *la fonction qui associe une sortie à chaque couple : état, symbole d'entrée.*

Les automates de Mealy se représentent sous la forme de diagramme d'états, de manière similaire aux automates de Moore. Prenons **My** (voir fig. 2.8), un automate de Mealy sur le même langage d'entrée L_Σ que **Mo**. Ici, l'automate génère en sortie le langage $L_{\mathcal{O}}$: $o_1(o_2o_1)^*$ qui représente fidèlement le flux d'entrée auquel le système est soumis. N.B cette équivalence est fortuite il n'y a pas nécessairement équivalence entre les langages, mais l'automate de Mealy possède cette possibilité de « rendre compte » de la séquence d'évènements à laquelle le système est soumis alors que cette possibilité n'est qu'indirectement accessible à l'automate de Moore.

My = $(X, \Sigma, t, X_0, F, \mathcal{O}, \tau)$ où
X={1,2}
Σ={σ, κ}
t={[(1,σ),2],[(2,κ),1]}
X_0={1}
F={2}
$\mathcal{O}$={o_1, o_2}
τ={[(1,σ),o_1],[(2,κ),o_2]}

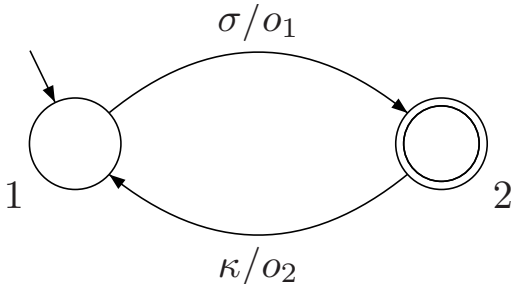


FIGURE 2.8 – Deux représentations de **My** (un automate de Mealy)

Graphiquement, l'expression des sorties se traduit figure 2.8 par l'apposition sur les labels de transition du symbole d'entrée et du symbole de sortie séparé du signe '/'. Cela produit des *labels* de transition de la forme 'σ/o' où $\sigma \in \Sigma$ et $o \in \mathcal{O}$ avec le symbole vide $\epsilon \in \Sigma \cap \mathcal{O}$. On parle de *triggers* pour les symboles en entrée et *actions* pour les symboles générés en sortie.

Il est possible d'écrire des transitions sans sortie (c'est à dire o=ϵ), dans ce cas on simplifie le label en précisant simplement le symbole en entrée. Il est également possible qu'une transition porte ϵ comme *trigger*, cela s'écrit sur le diagramme : '$/o$'. (voir fig. 2.9)

L'absence de *trigger* et d'*action* permet de faire l'économie d'un label pour la transition, mais nécessite une précision sémantique qui n'est pas intuitive. La question est de savoir quel sens précis donner à une transition ne portant pas de label ou n'ayant pas de *trigger*. En effet faut-il comprendre que, dès lors qu'aucun évènement n'a lieu, la transition doit se faire dans l'instant ^{g}, ou doit-on interpréter la transition comme signifiant que, le système pourra passer dans l'état suivant à n'importe quel moment ? Ce dernier type de sémantique de transition introduit de l'indéterminisme et ne permet plus de

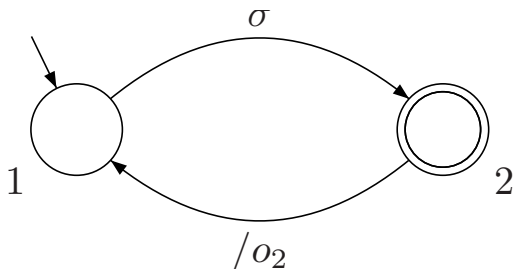


FIGURE 2.9 – Automate de Mealy avec absence de *trigger* ou d'*action*.

simuler fidèlement un système. Elle peut néanmoins permettre modéliser le constat d'une transition dont on ignore la cause.

Après ces éléments théoriques utiles à l'analyse formelle des Statecharts, on présente ci-après les réseaux de Petri.

2.5 Réseau de Petri

Une grande place a été dévolue jusqu'ici à l'introduction des automates comme formalisme SED. Cette section introduit un formalisme alternatif aux automates pour la modélisation de Systèmes à Évènements Discrets : les réseaux de Petri. Il existe par ailleurs d'autres formalismes alternatifs : tel par exemple Grafcet (David and Alla, 1992), Esterel (Berry and Gonthier, 1992) ou encore les algèbres de processus (Bergstra et al., 2001), qui ne seront pas présentés ici.

Les réseaux de Petri ont été créés par Carl Adam Petri en 1962. Il existe plusieurs variantes des Réseau de Petri (RdP) dont les sémantiques spécialisées sont plus ou moins riches.

Définition 15. *d'après Diaz (2001), un Réseau de Petri places-transitions* **R** *se définit par le tuple :* $(P, T, Pré, Post)$ *avec* :

$$
\begin{array}{rl}
P = & p_1, \ldots, p_i, \ldots, p_n \text{ un ensemble de places;} \\
T = & t_1, \ldots, t_i, \ldots t_n \text{ un ensemble de transitions, avec} P \cap T = \emptyset \\
Pré : & P \times T \mapsto \mathbb{N} \text{ une application d'incidence avant;} \\
Post : & P \times T \mapsto \mathbb{N} \text{ une application d'incidence arrière correspondant au arcs :} \\
& Pré(p_i, t) \text{ contient la valeur entière n associée à l'arc allant de } p_i \text{ à } t; \\
& Post(p_i, t) \text{ contient la valeur entière associée à l'arc allant de } t \text{ à } p_i.
\end{array}
$$

Les Réseaux de Petri (RdP) se représentent par des graphes (fig. 2.10), il existe deux types de sommets : (i) les places représentées par des cercles et (ii) les transitions par des carrés (cette notation est tirée de van der Aalst and van Hee (2002), plus traditionnellement, les transitions sont représentées par une barre.)

g. Le temps est défini dans $\mathbb{R}$. Il y a toujours une infinité d'instants entre deux instants distincts.

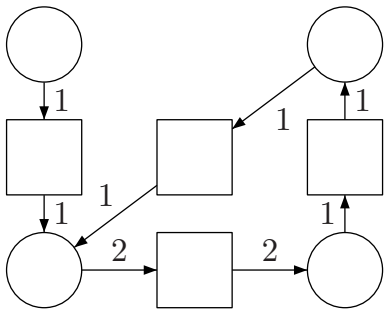


FIGURE 2.10 – Réseau de Petri (RdP) places transitions sans marquage

Le comportement du RdP est donné par l'évolution de jetons de places en places à travers les transitions selon les applications *Pré* et *Post*. La position des jetons dans le RdP est appelé le marquage du réseau.

Définition 16. *Un Réseau de Petri places-transitions* marqué *se définit par un couple* $(\mathbf{R}, m)$ *dans lequel* $\mathbf{R}$ *est un Réseau de Petri places-transitions et* $m : P \mapsto \mathbb{N}$ *une application appelée marquage.*

Les définitions ci-dessus impliquent que le comportement de $\mathbf{R}$ est connu à partir d'un unique m_0 le marquage initial. La figure 2.11 présente la séquence des différents marquages du RdP vue fig. 2.10 depuis son marquage initial m_0 .

Remarque : *le marquage d'un RdP correspond à la notion d'état global pour un système considéré comme la composition de processus concurrents, voir : états concurrents des Statecharts.*

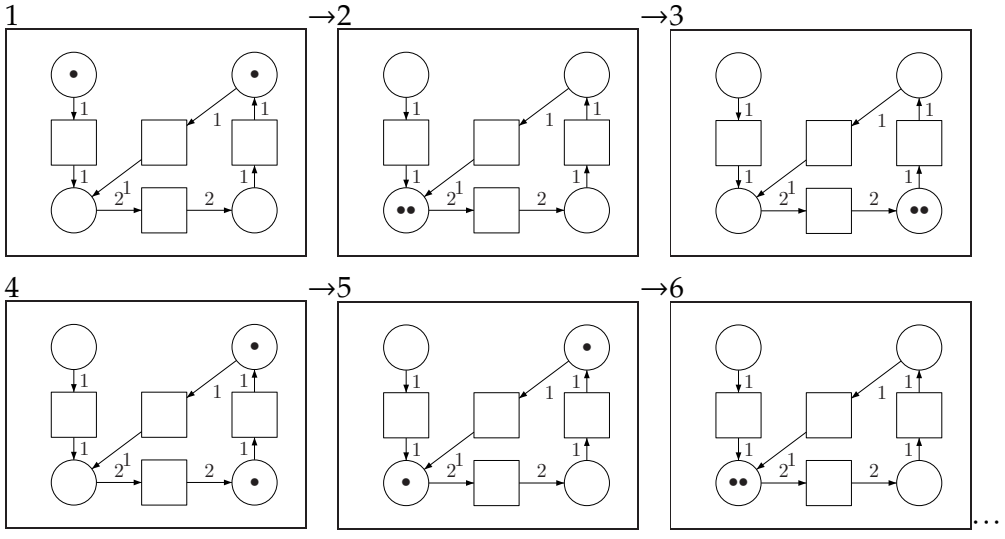
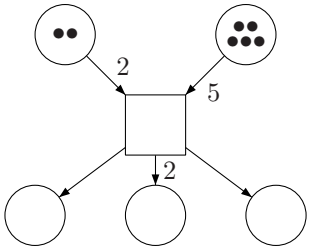


FIGURE 2.11 – L'évolution du marquage d'un RdP représente les changements états.

Lorsqu'il n'y a qu'un arc entre chaque transition et chaque place, la structure d'un RdP est analogue à celle d'un automate. En revanche, il est possible de créer des relations plus complexes entre places et transitions, avec plusieurs arcs entrants, une transition et

plusieurs arcs sortants d'une transition vers plusieurs places (par exemple fig. 2.12). La sémantique de ces transitions, permet de représenter les synchronisations d'automates concurrents ou les protocoles de communication entre processus communicants.



source : Diaz (2001)

FIGURE 2.12 – RdP synchronisant plusieurs processus : 2 et 5 jetons doivent être disponibles pour que la transition puisse être tirée ; à l'issue de la transition, les places de destination seront marqués de la manière suivant : 1,2,1.

Depuis la version initiale proposée en 1962, de nombreuses évolutions ont été proposées, notamment les Réseau de Petri temporisés (Merlin, 1974), pour lesquels une durée est associée au tir d'une transition ; les Réseau de Petri stochastiques, où les transitions possèdent une probabilité d'être tirée et les Réseau de Petri colorés pour lesquels les jetons sont dotés d'attributs (couleurs) dont les valeurs influent le tir des transitions. La figure 2.13 illustre l'organisation de ces évolutions.

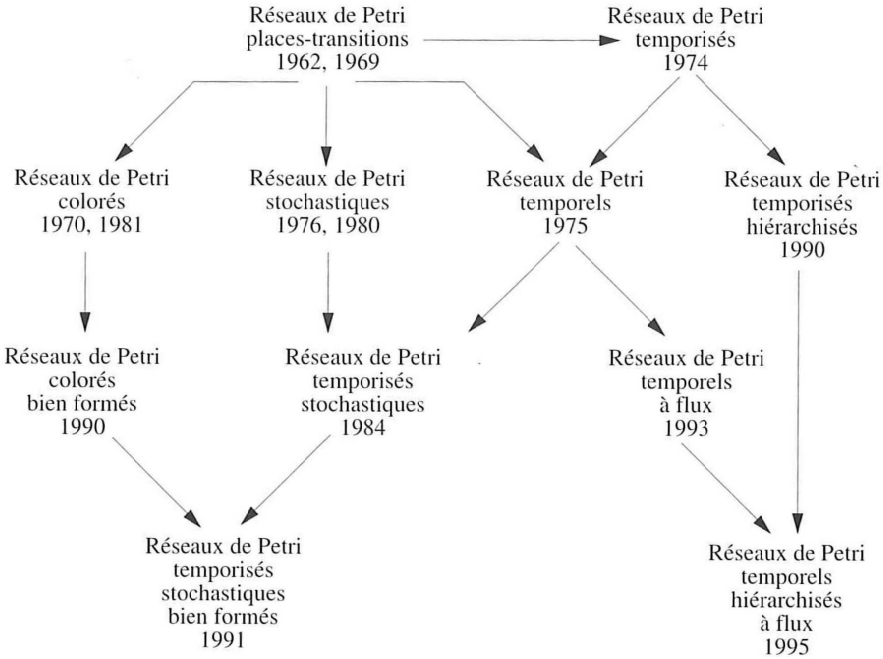


FIGURE 2.13 – Différentes familles de Réseaux de Petri

source : Diaz (2001)

En soi un RdP n'a pas de fonctionnalité. Pour modéliser un système par un RdP, il

faut donner un sens aux transitions et aux jetons. Ces derniers peuvent aussi bien représenter des ressources dans un système physique, que des messages dans un protocole de communication ou la réalisation de conditions logiques séquentielles (Hélias, 2003).

En effet, les RdP sont plus intuitifs que les automates pour représenter les communications de processus concurrents, et ils sont un formalisme bien adapté à la question de la gestion des ressources.

Cependant, au cours de mon travail, la question principale a été de modéliser la prise de décision séquentielle. On souhaitait que le modèle soit aisément compréhensible par des pathologistes, peu familier des formalismes informatiques. C'est pour cette raison que j'ai opté pour le langage des Statecharts, qui offre l'expressivité des automates à états finis pour les choix conditionnels, et une gestion suffisante des processus concurrents pour nous permettre de représenter un système de décision complexe sans avoir une explosion de la taille du graphe d'états.

Les RdP seront probablement à utiliser plus tard pour modéliser le passage à l'échelle de l'exploitation, lorsque le problème sera de gérer les flux d'informations et de ressources.

2.6 Statecharts

La section qui suit, présente le langage de modélisation graphique Statecharts qui a été utilisé dans cette thèse pour formaliser l'expertise et programmer un simulateur. On présentera en premier lieu les concepts propres aux Statecharts, puis on donnera la syntaxe du langage et enfin, on précisera les points importants de la sémantique formelle correspondant aux Statecharts objets tels qu'implémentés par Rhapsody® d'IBM [h].

2.6.1 Le concept de Statechart

Harel (1987) propose les Statecharts comme une solution au problème de l'explosion combinatoire qui handicape l'usage du formalisme Automate à États Finis (AEF) pour la modélisation de systèmes réactifs (Harel and Pnueli, 1985).

Définition 17. *Selon Harel and Pnueli (1985), un système réactif se caractérise par le fait d'être piloté par l'évènement, ayant à réagir à des stimulis internes comme externes en continu.*

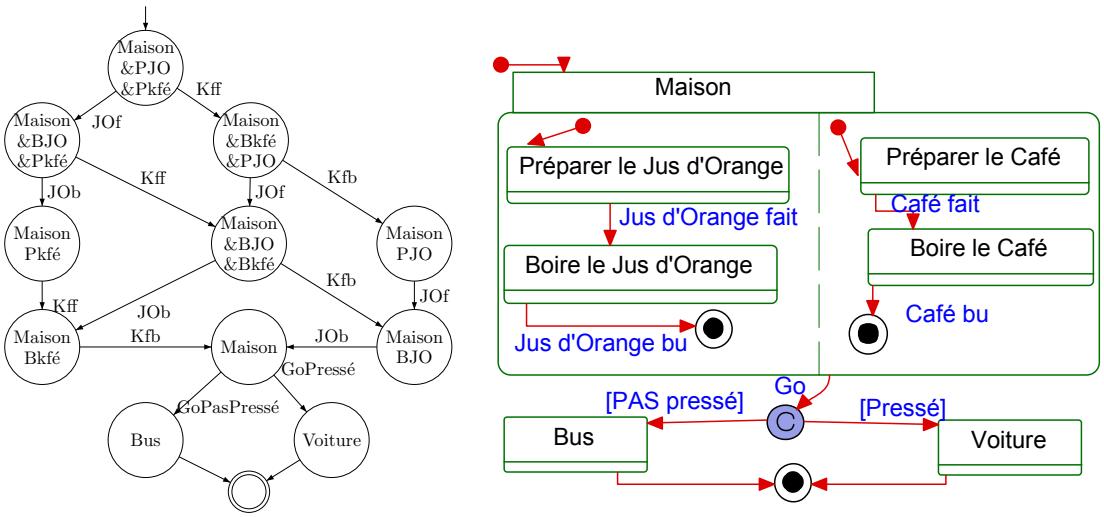
Kam et al. (2003) précise :

> [. . .] *Les systèmes réactifs sont ceux dont la complexité ne découle pas nécessairement de leur complexité algorithmique, mais de réactions complexes dans le temps. Ce sont le plus souvent des systèmes concurrents et fortement temporisés. Ils présentent des comportements hybrides principalement discrets, mais ayant aussi des aspects continus La structure d'un système réactif est constitué de nombreux composants en interactions, dans lequel le contrôle du comportement est fortement distribué parmi les composants*[. . .]
>
> *in : Kam et al. (2003)*

h. Au cours de ma thèse, le logiciel a changé trois fois de marque.

Il s'agit donc de systèmes dont la principale caractéristique est d'être en interaction permanente avec le monde (par exemple un système de feux de signalisation, une calculatrice électronique).

L'usage des AEF et leur représentation sous forme de diagramme d'états-transitions (on fera l'amalgame AEF diagramme d'états-transitions), pose un problème de combinatoire, dès lors que le système à modéliser est constitué de plusieurs sous-systèmes éventuellement concurrents (voir produit d'automates page 57). La figure 2.14 présente un détail d'une journée de travail, la préparation et consommation du petit-déjeuner (boisson froide et boisson chaude, puis départ de la maison). L'AEF est le résultat du produit de trois automates distincts : préparer un jus d'orange, préparer un café et quitter le domicile selon deux modes de déplacement possibles (les états inaccessibles, –être dans le bus et préparer un café– ont été supprimés). Les états de la fig. 2.14(a) décrivent l'état d'avancement de tous les processus de manière fusionnée, alors que le Statechart fig. 2.14(b) met plus clairement en évidence chacun des trois processus et permet de souligner les relations entre eux. Il y a synchronisation à la fin du petit-déjeuner des processus café et jus d'orange avant de pouvoir partir.



(a) diagramme d'état transition (b) Statechart

nom	label	type	nom	label	type
Préparer le café	Pkfé	état	café fait	Kff	évènement
Préparer le Jus d'Orange	PJO	état	café bu	Kfb	évènement
Boire le café	Bkfé	état	Jus d'orange fait	JOf	évènement
Boire le Jus d'Orange	BJO	état	Jus d'orange bu	JOb	évènement
Prendre la voiture	Voiture	état	partir pressé	GoPressé	évènement
Prendre le bus	Bus	état	partir à l'heure	GoPasPressé	évènement
Etre à la maison	Maison	état			

FIGURE 2.14 – détails de modèles d'une journée : la préparation du petit déjeuner.

La construction graphique des statecharts se fonde sur le concept de Higraph Harel (1988). Les Higraphs sont une organisation topologique de l'espace graphique permettant de représenter sous forme de graphe des notions ensemblistes telles que l'inclusion, la composition, l'intersection et le produit cartésien. Les Higraphs s'inspirent de deux formalismes graphiques : les diagrammes de Venn et les Hypergraphs (Berge, 1973, *in* Harel 1988). La figure 2.15 illustre l'utilisation d'un higraph pour représenter un organigramme spatialisé.

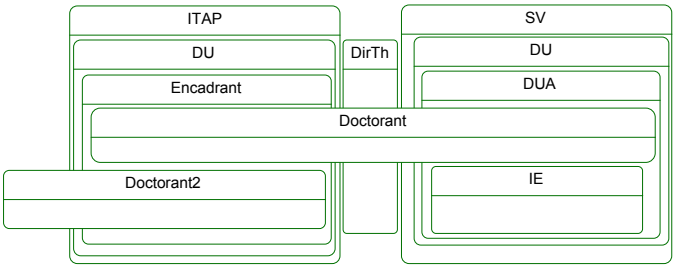


FIGURE 2.15 – Organigramme sous forme Higraph. Il présente la hiérarchie et les affectations

L'hypothèse à l'origine des statecharts est que, alors que le diagramme d'état n'utilise que la dimension géométrique de connexité du graphe pour représenter les états du système, par l'utilisation des dimensions topologiques, le Statechart rend possible une décomposition hiérarchique de la notion d'état. La modélisation de processus concurrents permet de réduire encore le nombre d'états visuellement décrits dans le modèle (à ne pas confondre avec le nombre d'états que peut prendre ce système, qui résulte des combinaisons de ces processus). Cruz-Lemus et al. (2005) a montré que ces traits rendent les Statecharts plus compréhensibles que les AEF (dès lors que l'utilisateur a un minimum d'expérience avec les Statecharts).

Harel (1987) résume les caractéristiques des Statecharts de la manière suivante :

diagramme d'état + hiérarchie + concurrence + communication diffusée (broadcast)

2.6.2 Syntaxe des Statecharts

Cette section présente la syntaxe des Statechart Unified Modelling Language (UML 2.0) tels que présentés dans Harel and Kugler (2004) et Damm et al. (2003).

NB. Les termes en petites majuscules font référence à des concepts présentés plus loin dans le texte.

2.6.2.1 Etats

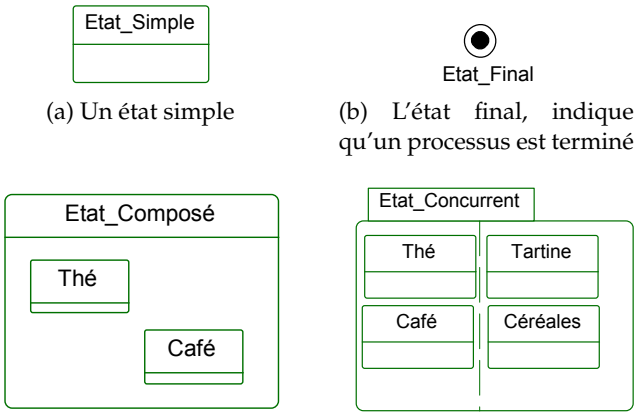
Harel introduit quatre types d'états pour les Statechart (SC) (voir fig. 2.16). *Un état simple* est un état atomique, il correspond à la même notion que l'état de l'AEF (voir fig. 2.16(a)). *Un état final* correspond à un état marqué d'un automate, il indique la fin du processus d'un SC ou d'un sous-Statechart.

La hiérarchie des états est rendue possible à travers deux types d'états composés.

Les « *AND-State* » ou *états concurrents* définissent deux ou plus sous-Statechart s'exécutant en parallèle (cf. fig. 2.14b). Les processus concurrents d'un AND-State sont séparés les uns des autres par ces lignes pointillées [i]. Lorsqu'un état concurrent est actif, chacun des sous-Statechart le composant possède un état actif.

Le second type d'état composé est : Le « *OR-State* » ou *état séquentiel*. Ces états sont composés d'un ensemble d'états enfants mutuellement exclusifs et définissant un sous-Statechart. Lorsqu'un *état composé* est *actif*, un seul de ses enfants peut être actif. Chaque état Or-State, possèdent *un état par défaut* parmi ses enfants

.



(a) Un état simple

(b) L'état final, indique qu'un processus est terminé

(c) Un état composé « OR-state » contient des états mutuellement exclusifs

(d) Un état composé « AND-state » contient des processus exécutés en parallèle

FIGURE 2.16 – Les quatre types d'états du Statechart

Remarque : *Le formalisme Statechart est une approche « modulaire » de la AEF, mais les définitions canoniques des Statecharts (Harel, 1987; Harel et al., 1990; Harel and Kugler, 2004; Damm et al., 2003, par exemple) ne garantissent pas l'indépendance des composants (ce qui est pourtant une propriété recherchée en informatique). Les travaux sur les μ-charts (Scholz, 1998) adaptent les Statecharts pour garantir la composition des états hiérarchiques. L'auteur montre que la hiérarchie d'états d'un μ-chart peut être « mis à plat » mécaniquement pour former un ensemble d'automates concurrents.*

2.6.2.2 Transitions

Les transitions relient un ensemble d'états sources à un ensemble d'états destinations. [j] Le label de la transition est composé d'un « TRIGGER » , d'une CONDITION et d'une ACTION. Le syntaxe d'un label de transition prend la forme suivante :

$$unTrigger[uneCondition]/uneAction$$

Une transition ne peut être franchie que si l'ensemble de ses états sources sont actifs, on dit alors qu'elle est « *potentialisée* ». Elle peut être *déclenchée* (triggered) par la réception

i. Dans l'implémentation de Rhapsody et la sémantique de Damm et al. (2003), à chacun des processus concurrents composant un AND-State, est associé un OR-State dans lequel le sous-Statechart est décrit.

j. La possibilité d'avoir plusieurs états sources ou plusieurs destinations permet de modéliser des comportements complexes de synchronisation entre des processus concurrents.

d'un TRIGGER, mais ne sera *franchie* que si la CONDITION est « vraie ». Comme pour les machines de Mealy, une ACTION s'exécute lors du franchissement de la transition. En théorie, les transitions sont instantanées ; dans la réalité, l'exécution de l'ACTION peut prendre plus que 0 temps.

Chacun des éléments constitutifs du label est optionnel. Une transition dépourvue de « trigger » est appelée une « transition nulle », elle est franchie des que ces états sources sont actifs [k] et que la CONDITION est vérifiée.

2.6.2.3 Triggers

Les triggers peuvent être de deux types.

Les *évènements* sont des messages pouvant émaner du Statechart ou d'une source externe. Dans les Statechart objets tels que spécifiés par UML et implémentés dans Rhapsody, les évènements ne sont pas instantanés. Ils sont d'abord stockés dans une pile « [» premier entré premier sorti] avant d'être traité dans l'ordre, lorsque le système se concentre sur l'objet (focus).

L'appel d'une « *triggered operation* » entraîne le franchissement de la transition (si la condition le permet). En Statechart UML, les « triggered opérations » sont une forme de communication synchrone entre deux objets. L'appel par un objet d'une de ses propres « triggered opérations » est interdit.

2.6.2.4 Conditions

La condition est *une formule booléenne* qui contrôle si une transition potentialisée peut effectivement être franchie. Elle est la composition de tests sur les attributs de l'objet, sur les valeurs d'une variable globale du système, ou d'une fonction booléenne.

2.6.2.5 Actions

Les actions sont des éléments algorithmiques qui modifient les valeurs internes d'un système. Il peut s'agir de la génération d'un événement, de l'affectation d'une variable, ou encore de l'appel d'une méthode de l'objet propriétaire du Statechart ou d'un objet associé.

Les Statechart sont des machines de Mealy, les sorties sont générées au cours des transitions. La figure 2.17 illustre les différentes possibilités syntaxiques permettant d'ajouter des actions à un Statechart.

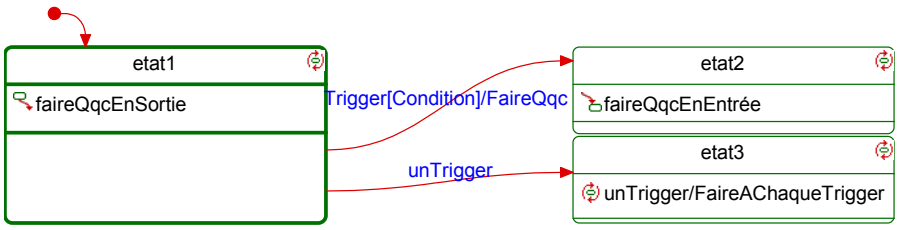


FIGURE 2.17 – Les actions peuvent être ordonnées à l'entrée, à la sortie ou pendant un état.

k. Cette sémantique correspond à Rhapsody et requiert en outre que l'objet propriétaire du Statechart détient le « focus »

En plus de pouvoir exécuter des actions pendant les transitions, le formalisme permet de spécifier un comportement générique en entrée et en sortie de chaque état. Ainsi dans l'exemple (fig. 2.17), l'`état1` possède une action en sortie. Si la transition vers l'état est franchi, une séquence d'action sera générée : `faireQqcEnSortie + faireQqch + faireQqcEnEntrée`; si l'autre transition est franchie, la séquence limitera à : `faireQqcEnSortie`. L'`état3` ne possède pas d'action en entrée ni en sortie, mais est composé d'une transition interne (static reaction) qui exécutera l'action `FaireAchaqueTrigger`, à chaque occurrence de « unTrigger » tant que l'état est actif. Ce trait est la mise en œuvre formelle de la notion d'activité (le *do*) des Statechart UML 2.0. La spécification d'UML ne permet par de garantir un état stable à chaque pas du système aussi n'a t-elle pas été reprise dans Rhapsody.

2.6.2.6 Pseudostates

Les pseudostates sont des symboles graphiques qui n'ont pas de transcription dans la sémantique formelle, mais permettent d'améliorer la lisibilité de diagramme. La figure 2.18 présente les pseudostates, ce sont :

l'état initial également appelé transition par défaut, permet d'indiquer, lorsqu'un état séquentiel parent devient actif, lequel des états enfants doit être activé par défaut.

le noeud conditionnel permet d'exprimer les choix en fonction d'une variable ou d'un état concurrent. Le noeud conditionnel évite d'écrire deux transitions chaque fois qu'une condition entraîne une alternative.

la fourche et la jointure sont utilisées pour « regrouper » les différentes sources d'une transition ou de ses destinations.

la jonction regroupe des transitions hétérogènes vers la même destination, cela économise l'espace du diagramme.

le connecteur de diagramme évite que de trop nombreuses transitions ne se croisent. Il s'agit d'un raccourci.

l'historique est le seul pseudostate modifiant le comportement du Statechart. On distingue l'historique superficiel pour mémoriser le sous-état actif le plus récent et l'historique profond qui mémorise la configuration la plus récente de la hiérarchie complète de tous les enfants actifs de l'état portant l'historique, jusqu'aux états atomiques.

Au cours de cette section, les principaux éléments syntaxiques des Statecharts ont été présentés. Ils permettent une lecture rapide des diagrammes.

2.6.3 Syntaxe formelle du Statechart

Les définitions formelles qui suivent, résument la syntaxe formelle des Statecharts donnée dans Damm et al. (2003).

Définition 18. *un StatechartSC est un 7-uplet,* $SC = ((S \cup pS), T, E, G, A, ce_0)$, *avec*
S l'ensemble des états ;
pS l'ensemble des pseudostates ;
$T \subseteq S^c ard(S) \times E* \times G \times A$ *avec* $E* = E \cup NONE$, *NONE est le symbole indiquant un transition nulle ;*

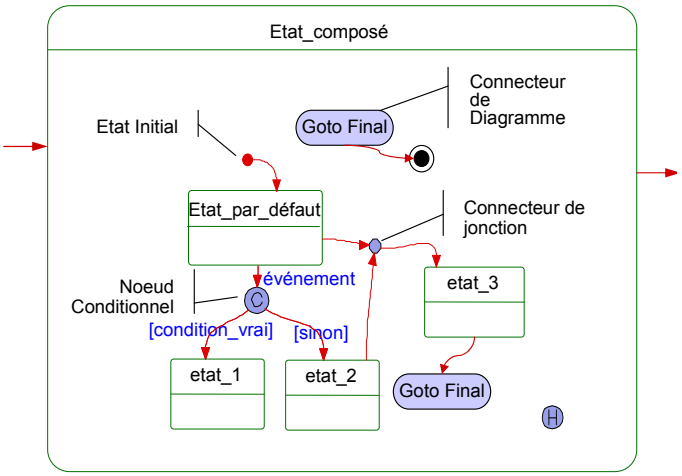


FIGURE 2.18 – Les Pseudostates (état initial, noeud conditionnel, jonction et le connecteur de diagramme) n'ont pas d'existence mathématique mais sont utiles graphiquement. L'historique permet de retrouver le dernier état actif lorsque le parent est réactivé

E l'ensemble des évènements et « triggered operation » ;
G l'ensemble des conditions contrôlant une transition ;
A l'ensemble des actions que peut produire SC ;
ce_0 la configuration par défaut du système.

La fonction $mod(s) \rightarrow SIMPLE,FINAL,OR,AND$ avec $s \in S$, détermine le type d'un état.

Les états d'un Statechart *SC* sont organisés comme une arborescence hiérarchique, avec au sommet un état nommé *top*, tel que $mod(top) = OR$.

On définit les relations entre états *child* et $father$ tel que $S' = child(s) \Rightarrow mod(s) = OR,AND$ et $s = father(S')$. précisons que $S' \subset S$

Le niveau d'un etat dans la hiérarchie se définit grâce à la fonction $depth : S \rightarrow \mathbb{N}^+$ telle que $depth(top) = 0$ et $depth(s) = depth(father(s) + 1)$. Il est alors possible de construire un ordre partiel sur SC.

Le *statut* d'un objet entre deux pas d'exécution, correspond à la *configuration d'états* active du Statechart décrivant son comportement.

Définition 19. *L'état global d'un Statechart est la configuration courante des états actifs. Une configuration d'états* ce *est définie de la manière suivante :*
$top \in ce$
si $s \in ce$ *et* $mod(s) = AND$ *alors l'ensemble* $child(s) \subset ce$
si $s \in ce$ *et* $mod(s) = OR$ *alors il existe exactement un* $s' \in$ *avec* $s' \in ce$

Définition 20. *La configuration d'états par défaut ce_0 du SC est définie par la fonction états par défaut.*

Précisons d'abord :
Soit $defaut(s)$ *la fonction qui pour un état s définit l'état par défaut s' tel que* $s' \in child(s)$ *avec* $mod(s) = OR$ *sinon* $default(s) =$*Nul.*
Soit un ensemble d'états $H \subseteq S$, *H est dit consistant, ce qui est noté* $\downarrow (H)$ *ssi chaque couple* s,s' *de S est lié par la relation* $child^*$ *(c-à-d* $depth(s) \leq depth(s')$ *ou* $depth(s) \geq depth(s')$*) ou s et s' sont*

orthogonaux (c-à-d appartiennent à différents processus enfants d'un AND-state). Un ensemble d'états consistant est un ensemble d'états susceptible d'être actifs en même temps.

Soit un ensemble consistant $H \subseteq S$, la fonction états par défaut noté $etatParDefaut(H)$ est le plus petit ensemble S' tel que :
$H \subseteq S'$,
si $s \in S'$ et $s \neq top$, alors $father(s) \in S'$,
si $s \in S'$, $mod(s) = OR$ et $child^{OR}(s) \cap S = \emptyset$, alors $defaut(s) \in S'$ où les éléments de $child^{OR}(s)$ sont de type OR,
si $s \in S'$ et $mod(s) = AND$, alors $child(s) \subseteq S'$
La configuration initiale ou configuration par défaut de SC est donnée par $etatParDefaut(top)$.

Ces précisions formelles permettent de mieux comprendre la nature du Statechart et les règles sémantiques qui déterminent son comportement.

2.6.4 Sémantique des Statecharts UML

Il existe plusieurs variantes des statecharts (von der Beeck, 1994, une revue un peu ancienne en énumère les principales) et de nombreuses sémantiques formelles. On pourra citer Harel et al. (1990) en ce qui concerne l'outil STATEMATE et Hamon and Rushby (2007) pour l'extension Stateflow du module Simulink Mathsworks (2008). Ces deux outils se distinguent des Statechart orientés objet d'UML et de Rhapsody à propos de deux aspects sémantiques importants qui seront évoqués ci-après : l'hypothèse de synchronisme parfait des transitions et les règles de priorités. La sémantique qui est présentée correspond à la sémantique formelle présentée dans Damm et al. (2003); Harel and Kugler (2004). Ces deux documents sont consistants, mais il existe des différences subtiles entre les Statechart Rhapsody et la norme UML 2.0 ; Crane and Dingel (2007) étudie ces différences.

On précise ici la sémantique des transitions. Il s'agit dans un premier temps de préciser la priorité de sélection des transitions. La figure 2.19 illustre le problème du choix de la transition. Lorsque l'évènement *e* advient, l'état final n'est pas intuitivement connu, la règle pourrait être de donner la priorité à la transition émanant de l'état le plus abstrait (ici le transition $S1 \rightarrow S2$) ou plutôt de privilégier le comportement le plus spécifique (ici $(S11, S3) \rightarrow final$). Dans Rhapsody, c'est la seconde solution qui a été choisie.

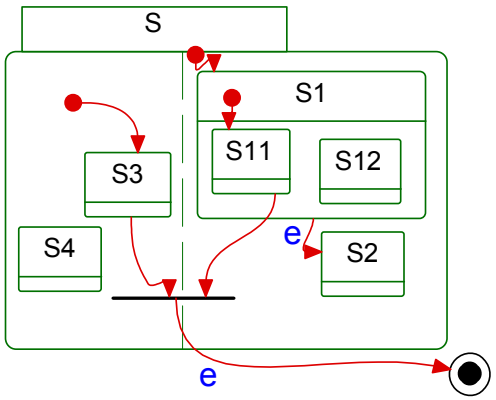


FIGURE 2.19 – quelle priorité pour les transitions ?

Ce choix est en rupture avec la sémantique proposée dans STATEMATE. Plus profond encore est l'abandon de l'hypothèse de synchronisme parfait des transitions ; dans Rhapsody, le modèle d'exécution correspond à ce qu'on nomme « run to completion » [l].

Ces deux paradigmes découlent de l'association des Statecharts dans un modèle objet. Dans Rhapsody le comportement du système est une succession de statuts observables à la fin de *pas* d'exécution. Le système se concentre sur un objet (focus), soit lorsque la pile d'appels transmet à l'objet un événement qui lui est destiné, soit lorsqu'un autre objet fait un appel direct à une triggered operation (TO). S'il existe au moins une transition dans l'ensemble des transitions potentialisées qui puissent être déclenchée par l'évènement ou la TO et dont la condition est vraie, alors le pas est déclenché. Sinon, l'événement est consommé et le statut de l'objet reste inchangé.

Chaque action (en entrée, label, en sortie) déclenche ensuite des micro-pas. Les actions pouvant prendre du temps, l'hypothèse de synchronie des transitions ne peut donc être conservé dans Rhapsody. Les micro-pas sont exécutés séquentiellement parmi tous les OR-states tant que les conditions permettent le franchissement des transitions nulles [m]. Lorsqu'aucune transition ne peut plus être franchie, le pas est complété, un nouveau statut stable est atteint.

Remarque : *La diffusion (broadcast) des événements n'est pas le mode de communication entre les objets. Par contre, au sein du Statechart d'un objet, les évènements sont perçus par l'ensemble de processus concurrents.*

Ceci clôt la présentation des Statechart.

2.7 Conclusion du chapitre

On a montré que la protection des cultures est un problème de contrôle. Le pathosystème oïdium, mildiou, vigne est un système hybride pour lequel l'impossibilité de « guérir » les organes végétaux contaminés exclut l'usage d'une stratégie de contrôle en feedback. A cela s'ajoute l'absence de modèle biotechnique qui empêche l'application d'une boucle de contrôle feed forward.

Dans ces conditions, le recours à l'expertise est pertinent pour concevoir une méthode de contrôle du mildiou et de l'oïdium dont la consigne est de produire du raisin de qualité en étant économe en commande.

On a montré que c'était le contrôleur expert qu'il s'agissait ici de modéliser. La nature du problème de modélisation qu'il me fallait résoudre a été analysée et fournit les bases théoriques à la validation présentée au chapitre 6.

Les outils formels utilisés pour modéliser le POD Mildium, ont été présentés ainsi que les bases théoriques qui les sous-tendent.

l. exécuter jusqu'à son terme

m. potentiellement il peut y avoir un infinité de micro-pas dans une boucle de transition nulle. Rhapsody gère ce problème en limitant le nombre de transition nulle successive

Ce chapitre théorique visait à analyser le problème de manière abstraite puis de présenter les outils sous l'angle technique. Le chapitre 3 apporte un regard plus méthodologique. Il présente diverses manières de modéliser les processus de décision en entreprise et notamment dans l'entreprise agricole.