République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de
la Recherche Scientifique
Université Ibn Khaldoun – Tiaret –

Faculté des Sciences et de la Technologie et Sciences de la Matière
Département Informatique

MEMOIRE EN VUE DE L'OBTENTION DU DIPLÔME DE MAGISTER

SPECIALITE : Informatique

OPTION : Systèmes d'**I**nformation et de **C**onnaissances (SIC)

Présenté par

MOSTEFAOUI Kadda

SUJET DU MEMOIRE :

Apport de la simulation à événements discrets DEVS Dans la navigation réactive

SOUTENU LE ……2012 Devant Le Jury Composé de :

Mr. Amar BALLA	Professeur (ESI) - Alger -	Président
Mr. Omar NOUALI	Directeur de recherche CERIST-Alger-	Examinateur
Mr. Rachid CHALAL	Maître de conférences A (ESI) - Alger -	Examinateur
Mr. Youcef DAHMANI	Maître de conférences A (UIK) -TIARET-	Directeur du mémoire

ANNEE UNIVERSITAIRE 2011/2012

Remerciements

En premier lieu, je remercie Dieu de m'avoir donné la force, le courage et la volonté pour achever ce modeste travail.

Je tiens à exprimer ma sincère gratitude à Monsieur DAHMANI Youcef, Maître de conférence 'A' à l'université de Tiaret, qui ma fait l'honneur de diriger et de veiller au bon déroulement de ce travail. Ses connaissances et son expérience ont été une source constante de savoir. Qu'il trouve en ces quelques lignes ma profonde sympathie.

Mes vifs remerciements vont à monsieur BALLA Amar, Professeur à Ecole nationale Supérieur d'Informatique (ESI) de m'avoir fait l'honneur d'accepter, de présider et d'honorer de sa présence le jury de soutenance du présent mémoire.

Je remercie également Monsieur NOUALI Omar, Directeur de recherche au CERIST et Monsieur Rachid CHALAL Maître de conférence 'A' à Ecole nationale Supérieur d'Informatique (ESI), pour l'intérêt qu'ils ont porté au travail effectué en acceptant de participer à ma soutenance en tant qu'examinateurs.

Je souhaite également remercier ma famille, et plus particulièrement mes parents, mon épouse et mes enfants Ibrahim Mohamed et Ahmed Yasser pour leurs encouragements constants.

Résumé

Ce travail est concentré sur la modélisation et la simulation de la navigation réactive d'un robot mobile en utilisant les caractéristiques des systèmes à événement discret (DEVS), qui est une jeune approche dans la modélisation et la simulation. La mobilité et l'autonomie des robots posent des problèmes complexes en matière de génération de trajectoire dans des espaces fortement contraints, et de prise de décision à partir d'informations capteurs imprécises ou incomplètes. Dans ce travail nous proposons une architecture décisionnelle reposant sur le formalisme DEVS (Discrete EVent system Specification) et la théorie de la logique floue pour la modélisation et la simulation d'un système intelligent observant, décidant, agissant sur un environnement dynamique et incertain dans le contexte du formalisme DEVS. Cette problématique se situe à la frontière de deux domaines de recherche : l'intelligence artificielle(application robotique mobile) et la modélisation et la simulation.

Mots clés : Modélisation & Simulation, DEVS, Logique Floue, Soft computing, Navigation réactive, Robotique Mobile, Contrôleur flou.

ملخص :

في هذا العمل ينصب اهتمامنا حول نمذجة ومحاكاة ملاحة الروبوت الآلي عبر استخدام خصائص النظم ذات الأحداث المنفصلة (DEVS) التي هي عبارة عن منهاج حديث في النمذجة والمحاكاة .

إن التجوال الآلي و استقلالية الروبوت تطرح مشاكل معقدة من حيث تحديد المسار في مساحات مقيدة للغاية، و اتخاذ القرار بناء على معلومات غير دقيقة أو غير مكتملة التي يجمعها عن طريق ملتقطاته.

إن الغرض من هذا العمل هو اقتراح نموذج يرتكز على النظم ذات الأحداث المنفصلة ((DEVSونظرية المنطق الغامض من أجل نمذجة ومحاكاة نظام روبوتي ذكي يلاحظ، يقرر و يتفاعل مع وسط ديناميكي ومجهول في سياق النظم ذات الأحداث المنفصلة (DEVS) .

إن هذه الإشكالية تقع على الحدود بين مجالين من مجالات البحث و هما :الذكاء الاصطناعي والنمذجة والمحاكاة.

كلمات مفتاحية :النمذجة والمحاكاة، النظم ذات الأحداث المنفصلة (DEVS) ، المنطق الغامض، والحوسبة ، الملاحة التفاعلية، الروبوتات الجوالة، وحدة تحكم المنطق الغامض.

Abstract

This work presents the modeling and simulation of reactive navigation of a mobile robot using the characteristics of discrete event systems (DEVS), which is a young approach in modeling and simulation. The mobility and the autonomy of robots pose complex problems, as regards generation of trajectory in strongly constrained and unstructured spaces, and decision-making based on inaccurate or incomplete information sensors. In this work we propose an architecture based on the DEVS (Discrete Event System Specification) and the theory of fuzzy logic for modeling and simulation of an intelligent agent observing, deciding, acting on a dynamic and uncertain environment in context of the DEVS formalism. This problem is at the border of two areas of research: artificial intelligence (Mobile Robotics case) and modeling and simulation.

Keywords: Modeling & Simulation, DEVS, Fuzzy Logic, Soft Computing, Reactive navigation, Mobile Robots, Fuzzy Logic Controller.

Liste des figures

Introduction générale

Introduction générale

La mobilité et l'autonomie des robots posent des problèmes complexes en matière de génération de trajectoire dans des espaces fortement contraints, et de prise de décision à partir d'informations capteurs imprécises ou incomplètes.

La modélisation et la simulation sont des outils incontournables pour analyser le comportement des systèmes dynamiques. Plusieurs méthodes ont été proposées pour améliorer le processus d'analyse de comportement de ces systèmes. Ces propositions tentent d'atteindre des modèles plus réalistes, assez simples et fortement flexibles. Notre intérêt porte sur DEVS (Discrete EVent system Specification) qui est un formalisme de modélisation des systèmes à événements discrets.

Depuis les années 1970, des travaux formels ont été menés pour développer les fondements théoriques de la modélisation et de la simulation des systèmes dynamiques à évènements discrets.

Le formalisme DEVS, a été introduit par le professeur B.P. Zeigler comme un formalisme abstrait pour la modélisation à évènements discrets.

Le formalisme DEVS est une approche de modélisation basée sur la théorie générale des systèmes. Plus précisément, c'est un formalisme modulaire et hiérarchique pour la modélisation, centré sur la notion d'état.

La Logique Floue permet la formalisation des imperfections dues aux connaissances inexactes d'un système et à la description du comportement du système par des mots. C'est un moyen efficace de prendre en compte l'imprécision dans des connaissances. Elle permet alors de modéliser pour les systèmes informatiques des modes de raisonnement proches des modèles humains.

Le formalisme DEVS peut être vu comme un environnement de multi-modélisation regroupant de manière cohérente d'autres formalismes de modélisation basés eux aussi sur la théorie générale des systèmes et centrés sur les états. Sa capacité d'ouverture, au sens informatique, en fait un formalisme adapté à un grand nombre de domaines d'application.

Notre problématique de recherche relève de l'intelligence artificielle, et plus particulièrement la question de la modélisation et de la simulation de la navigation réactive d'un robot mobile dans un environnement dynamique et incertain dans le contexte DEVS.

Nous présentons dans ce mémoire, une approche pour la prise en compte de l'aspect flou dans formalisme DEVS pour décrire et simuler un système complexe de la navigation réactive d'un robot mobile.

Ce mémoire est divisé en quatre chapitres. Dans un premier temps, nous abordons quelques notions de base nécessaires à la compréhension du domaine de la robotique mobile. Ensuite nous présentons les différents types de robots mobiles, puis nous étudions les outils permettant aux robots de percevoir leur environnement et de s'y repérer, étape primordiale nécessaire à l'autonomie totale des robots mobiles. Ensuite nous exposons les approches existantes dans la navigation d'un robot mobile autonome. Enfin nous présentons les architectures de contrôle des robots.

Le deuxième chapitre, est consacré à la logique floue, nous commençons par énoncer les fondements de la logique floue. Nous verrons comment elle permet d'exprimer selon un formalisme unique des informations très diverses (données incertaines ou imprécises, connaissances exprimées sous forme linguistique, . .). Ensuite, nous présentons en détail les systèmes d'inférence floue.

Le troisième chapitre présente un bref état de l'art sur les principaux concepts de Modélisation et de Simulation, puis nous rappellerons, les concepts de bases des formalismes DEVS pour la spécification de modèles à événements discrets.

Le dernier chapitre présente une application de l'environnement JDEVS permettant de modéliser et simuler un système complexe de la navigation réactive d'un robot mobile dans un environnement incertain.

Enfin, ce manuscrit se termine par une conclusion générale qui récapitule les travaux réalisés et propose des visions pour d'éventuels travaux futurs.

Chapitre 1

La Robotique Mobile

1. La robotique mobile

1.1 Introduction

Le rêve de créer des machines qui pourraient accomplir des tâches spécifiques de manière autonome, et l'intérêt porté à la robotique plus généralement remonte à très longtemps. Des premiers automates aux plus récents robots.

L'objet de la robotique est l'automatisation des systèmes mécaniques. En dotant le système de capacités de perception, d'action et de décision pour lui permettre d'interagir avec son environnement d'une façon autonome.

La robotique est un domaine de recherche qui se situe à la frontière de l'intelligence artificielle, l'automatique, l'informatique et de la perception par ordinateur ; cette interdisciplinarité est à l'origine d'une certaine complexité.

La robotique mobile autonome vise plus spécifiquement à concevoir des systèmes capables de se déplacer de façon autonome. Ses applications existent dans plusieurs domaines notamment dans l'automobile, de l'exploration planétaire ou de la robotique de service ou plus récemment le secteur médical qui démontrent aujourd'hui l'intérêt économique et social de ces recherches. De nombreuses applications restent à découvrir, qui ne découlent pas directement des avancées de la robotique mais qui utilisent ses méthodes et ses développements [OLIVIER, 2006].

1.2 Définition d'un robot mobile

En général, un robot mobile est définit [DAVID, 2006] comme étant une machine équipée de capacités de perception, de décision et d'action qui lui permettent d'agir de manière autonome dans son environnement en fonction de la perception qu'il en a.

En particulier, un robot mobile autonome est un système mécanique, électronique et informatique complexe mettant en œuvre :

- un ensemble de capteurs. Ils peuvent être de deux types différents :
  - Extéroceptifs (télémètres, caméras, etc.).
  - Intéroceptifs (odomètres par exemple).

Les capteurs extéroceptifs ont pour objectif d'acquérir des informations sur l'environnement. Les capteurs intéroceptifs fournissent des données sur l'état interne du robot (telles que sa vitesse ou sa position).

- un ensemble d'effecteurs.

L'objectif du robot est d'atteindre un objectif dans son environnement en évitant les obstacles. Le problème que l'on doit résoudre est de déterminer en fonction des données capteurs quelles commandes doivent être envoyées à chaque instant au robot pour atteindre cet objectif. Ces effecteurs permettent donc au robot d'évoluer dans son environnement.

1.3 Les robots mobiles à roues

Les roues sont le moyen de locomotion le plus répandu en matière de robotique mobile [DAVID, 2006]. En fait, les robots mobiles à roues sont faciles à réaliser et présentent de grandes possibilités de déplacement et de manœuvrabilité avec une vitesse et une accélération importantes. Bien évidemment, pour un ensemble donné de roues, toute disposition ne conduit pas à une solution viable. Un mauvais choix peut limiter la mobilité du robot ou occasionner d'éventuels blocages. Par exemple, un robot équipé de deux roues fixes non parallèle ne pourrait pas aller en ligne droite. Pour qu'une position de roues soit viable et n'entraîne pas de glissement des roues sur le sol, il faut qu'il existe pour toutes ces roues un unique point de vitesse nulle autour duquel tourne le robot de façon instantané. Ce point lorsqu'il existe, est appelé centre instantané de rotation (CIR).

1.4. Type de robots mobiles à roues

1.4.1. Les robots mobiles à roues différentielles

Les robots à roues différentielles possèdent deux roues motrices conventionnelles non orientables et une ou deux roues folles pour assurer la stabilité du robot ; les roues motrices ont des angles de rotation indépendants et des axes de rotation confondus. Leur degré de mobilité est de deux. Ce type de robot est très répandu en raison de sa simplicité de construction et de ses propriétés cinématiques intéressantes [BENCHELOUI, 2011].

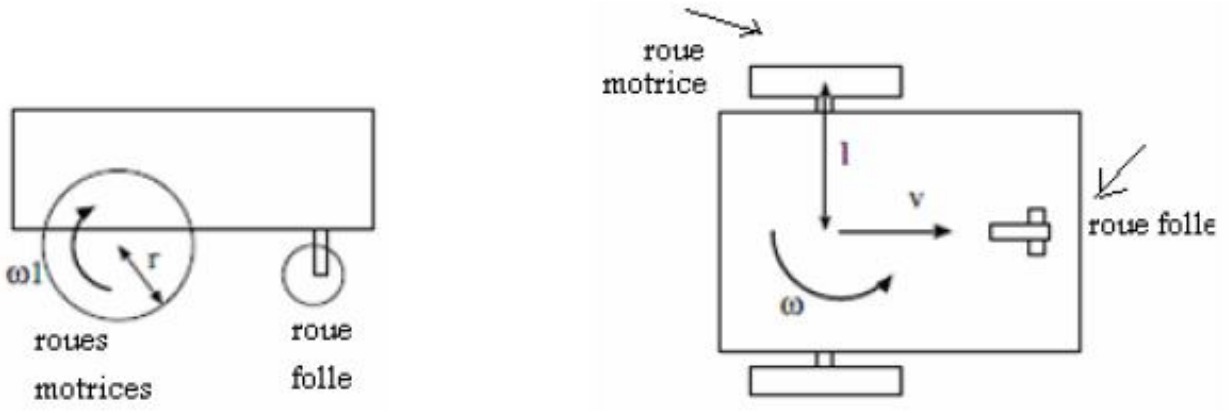


Figure 1.1 : Robot à roues différentielles

1.4.2. Les robots mobiles de type « tricycle »

Sont équipés d'un essieu arrière fixe muni de deux roues non orientables et d'une roue avant centrée orientable. Ils n'ont qu'un degré de mobilité et un degré de directionnalité. Pour la commande et la localisation de ce type de robot, un capteur d'orientation est associé à la roue orientable, et deux encodeurs sont associés aux roues motrices [BENCHELOUI, 2011].

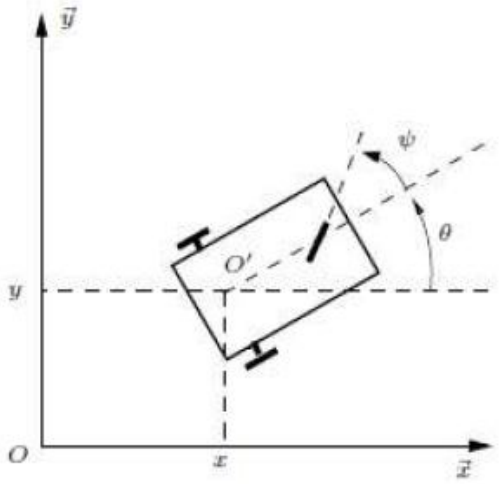


Figure 1.2 : Robot de type tricycle

1.4.3. Les robots mobiles de type "voiture"

Possèdent, de manière classique, à l'arrière un essieu non orientable muni de deux roues non orientables et libres en rotation et, à l'avant deux roues centrées orientables (équivalentes à une simple roue centrée orientable). Ils possèdent un degré de mobilité et un degré de directionnalité [BENCHELOUI, 2011].

.

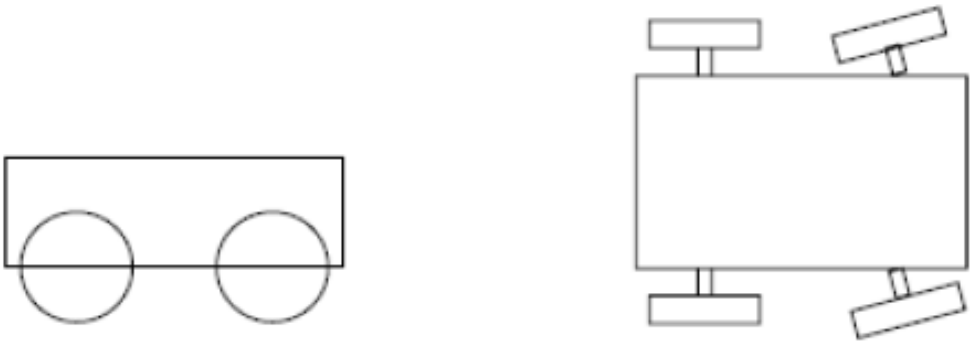


Figure 1.3: Robot de type voiture

1.5 Les systèmes de perception

La perception en robotique mobile est la capacité du robot à recueillir, traiter et mettre en forme des informations qui lui sont utiles pour agir et réagir dans l'environnement qui l'entoure. Elle est donc la faculté de détecter et /ou appréhender l'environnement proche ou éloigné du robot.

La perception est nécessaire pour la sécurité du robot, la modélisation de l'environnement et l'évitement et le contournement d'obstacles.

Les moyens utilisés pour la perception de l'environnement sont nombreux et variés, parmi ceux-ci nous pouvons citer :

- Les télémètres laser et ultrasonores
- Les capteurs optiques et infrarouges
- Les capteurs tactiles.
- Les systèmes de vision (Caméras)

La fonction perception consiste globalement à saisir un certain nombre d'informations sensorielles dans le but d'acquérir une connaissance et une compréhension du milieu d'évolution [DROCOURT, 2002], elle est le préalable indispensable aux étapes suivantes qui sont généralement pour un robot mobile les étapes de localisation et de mise à jour de carte de l'environnement (Figure 1.4).

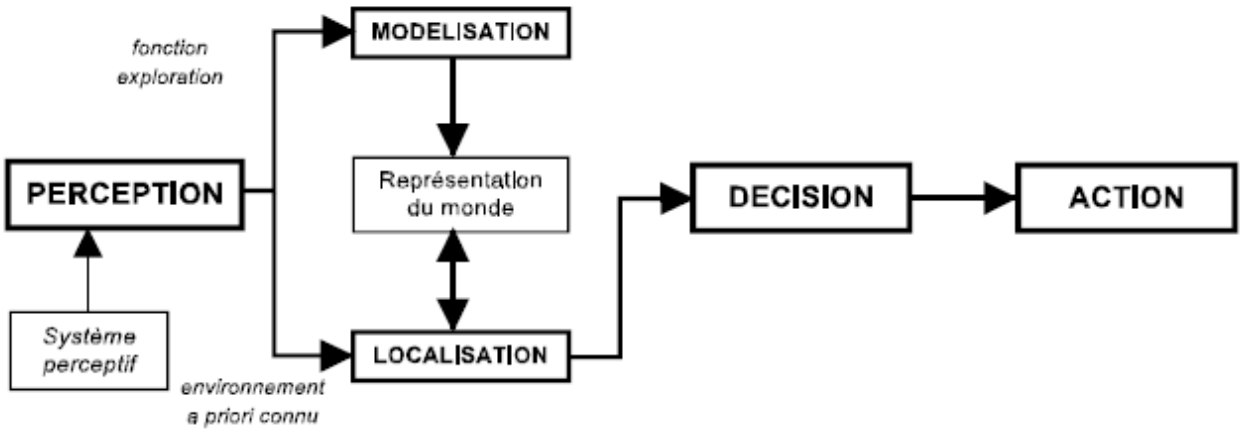


Figure 1.4 : Chaîne fonctionnelle d'un système de navigation

La classification des capteurs est généralement faite par rapport à deux familles :

- les capteurs proprioceptifs qui fournissent des informations propres au comportement interne du robot, c'est-à-dire sur son état à un instant donné,
- les capteurs extéroceptifs qui fournissent des informations sur le monde extérieur au robot.

1.5.1 Les capteurs proprioceptifs

Les capteurs proprioceptifs fournissent, par intégration, des informations élémentaires sur les paramètres cinématiques du robot. Les informations sensorielles gérées dans ce cadre sont généralement des vitesses, des accélérations, des angles de giration, des angles d'attitude.

Les capteurs proprioceptifs sont regroupés en deux familles [FRAPPIER, 1990]:

- ✓ les capteurs de déplacement qui comprennent les odomètres, les accéléromètres, les radars Doppler, les mesureurs optiques. Cette catégorie permet de mesurer des

déplacements élémentaires, des variations de vitesse ou d'accélération sur des trajectoires rectilignes ou curvilignes.

- ✓ les capteurs d'attitude, qui mesurent deux types de données : les angles de cap et les angles de roulis et de tangage. Ils sont principalement constitués par les gyroscopes et les gyromètres, les capteurs inertiels composites, les inclinomètres, les magnétomètres.

1.5.2 Les capteurs extéroceptifs

Les capteurs extéroceptifs permettent de percevoir le milieu d'évolution du robot. Ils sont généralement le complément indispensable aux capteurs présentés précédemment. Des méthodes de fusion de données seront alors utilisées pour conditionner et traiter les informations sensorielles de natures différentes. Deux familles de capteurs extéroceptifs embarqués peuvent être identifiées : les capteurs télémétriques et les systèmes de vision.

1.5.2.1 Les capteurs télémétriques

Il existe différents types de télémètres, qui permettent de mesurer la distance à l'environnement, utilisant divers principes physiques.

a)- Télémètres à ultrasons

Ils utilisent la mesure du temps de retour d'une onde sonore réfléchie par les obstacles pour estimer la distance. Ces télémètres sont très simple et peu cher.

b)- Télémètres à infrarouges

Ils utilisent une lumière infrarouge au lieu d'une onde sonore pour la détection. Il est possible de mesurer simplement le retour ou le non-retour d'une impulsion codée, ce qui permet de détecter la présence ou l'absence d'un obstacle dans l'espace.

c)- Télémètres laser

Ces télémètres sont les plus utilisés à l'heure actuelle pour des applications de cartographie et de localisation. Ils utilisent un faisceau laser mis en rotation afin de balayer un plan horizontal, et qui permet de mesurer la distance des objets qui coupent ce plan.

1.5.2.2 Les systèmes de vision (les caméras)

Ils utilisent une caméra pour percevoir l'environnement parce qu'elle semble proche des méthodes utilisées par les humains. Le traitement des données volumineuses et complexes fournies par ces capteurs reste cependant difficile à l'heure actuelle, même si cela reste une voie de recherche très explorée.

1.6 Localisation

La localisation est définie comme : "fonction consistant à déterminer, dans un repère de travail donné, certains paramètres de position et/ou d'attitude du robot qui sont nécessaires à l'accomplissement de sa mission"[SLIMANE, 2005].

La localisation d'un robot mobile s'effectue par des capteurs proprioceptifs et/ou extéroceptifs. Cette localisation est d'autant plus nécessaire que le lieu d'évolution est encombré et complexe. Le comportement de l'être vivant illustre bien ces propos, en effet il doit toujours connaître sa situation pour se déplacer d'un point à un autre, soit en identifiant des repères artificiels, on parle de localisation absolue, soit tout simplement en mesurant les distances parcourues et les directions empruntées depuis sa position initiale. Un robot mobile doit connaître ses coordonnées de position pour être autonome vis -à- vis de l'espace et de l'intervention humaine.

Plusieurs techniques et méthodes ont été développées pour assurer la connaissance exacte et de façon autonome de la position d'un robot mobile dans son environnement. A ce jour, ces techniques peuvent être regroupées en deux catégories principales : Les méthodes de localisation relative et les méthodes de localisation absolue [BENCHELOUI, 2011].

1.6.1 Localisation relative ou à l'estime

La position du robot est calculée en incrémentant sa position précédente de la variation mesurée de ses déplacements grâce à des capteurs proprioceptifs.

Les erreurs dues à cette méthode peuvent être importantes. On distingue deux méthodes principales de localisation relative : La méthode odométrique et la méthode inertielle.

1.6.1.1 La méthode odométrique

La technique d'odomètrie est très utilisée pour localiser les robots mobiles, elle présente l'avantage d'être simple d'emploi et d'un coût faible. L'idée fondamentale de cette méthode est l'intégration de l'incrément de la position, calculé grâce à des encodeurs montés sur les roues, par rapport au temps.

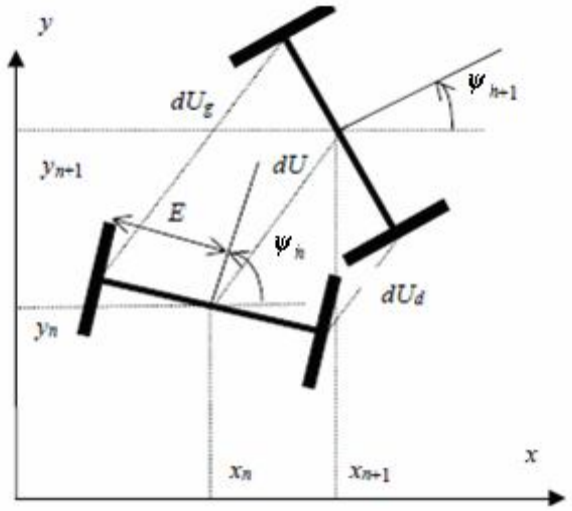


Figure 1.5 : Calcul de la position grâce à l'odométrie

1.6.1.2 Localisation inertielle

Cette technique utilise des accéléromètres pour calculer l'accélération subie par la base mobile et des gyroscopes pour calculer la variation de sa rotation [BARSHAN, 1995]. L'intégration de cette mesure permet de calculer la variation de la position. Les capteurs utilisés dans ce type de localisation présentent l'avantage d'être « auto-suffisants » puisqu'ils ne nécessitent aucune référence externe.

1.6.2 Localisation absolue

La localisation absolue est une technique qui permet à un robot de se repérer directement dans son milieu d'évolution, que ce soit en environnement extérieur (mer, espace, terre), ou en environnement intérieur (ateliers, immeubles, centrales nucléaires...). Ces méthodes de localisation sont basées sur l'utilisation de capteurs extéroceptifs [DROCOURT, 2002].

Pour répondre à la problématique qu'est la localisation d'un robot dans son environnement, deux types de stratégies sont utilisables : la première consiste à utiliser des points de repère naturels et la deuxième à utiliser des points de repère artificiels.

1.6.2.1 Repères artificiels

Les repères artificiels sont des balises caractéristiques qui sont ajoutées au milieu d'évolution du robot et dont les positions sont connues. L'inconvénient de ce type de techniques réside essentiellement dans son manque de souplesse et dans sa lourdeur d'utilisation [DROCOURT, 2002]. En effet un domaine d'évolution vaste nécessitera un investissement lourd en équipement. En outre tout changement de configuration de l'environnement impliquera une remise en cause du réseau de balises. En revanche cette technique a le gros avantage d'être précise, robuste et surtout de satisfaire la contrainte temps réelle.

1.6.2.2 Repères naturels

Cette technique consiste à utiliser les éléments caractéristiques de l'environnement pour estimer la position du robot. L'intérêt de ces méthodes est donc sa souplesse d'utilisation puisqu'elles ne nécessitent pas d'aménager le milieu d'évolution du robot [DROCOURT, 2002].

1.7 Navigation

1.7.1 Introduction

Il y a deux grandes familles de méthodes permettent à un robot d'atteindre une position souhaitée. Les méthodes sans trajectoire explicite (champs de potentiels, réseaux de neurones et logique floue) cherchent plutôt à contrôler le mouvement global du robot de manière à le guider vers son but. L'autre famille, celle des méthodes de suivi de trajectoire, permet au robot de suivre « du mieux possible » une trajectoire de référence donnée, connaissant les contraintes cinématiques [MORETTE, 2009].

1.7.2 Définition

Un navigateur désigne généralement un module chargé de faire mouvoir le robot mobile dans son milieu d'évolution [MORETTE, 2009].

1.7.3 Méthodes sans trajectoire

Le premier type d'approche pour résoudre le problème de navigation d'un robot mobile autonome est de rendre la cible du robot attractive pour celui-ci, de manière à ce qu'il cherche à s'en rapprocher. Et inversement rendre les obstacles répulsifs, pour qu'il s'en écarte et reste à bonne distance. Dans cette approche, on ne cherche pas à anticiper la trajectoire du robot, on le laisse librement décider à chaque instant dans quelle direction il doit se rendre, à partir des informations recueillies concernant son environnement local [MORETTE, 2009].

Il existe trois méthodes qui reposent sur ce principe : les champs de potentiels artificiels, les réseaux de neurones et la logique floue.

1.7.3.1 Champs de potentiels artificiels

Les méthodes de type APF (Artificial Potential Field) ont été développées dans les années 80 [KHATIB , 1986]. Le principe des méthodes APF est de construire un champ de potentiels sur l'environnement de navigation du robot. La valeur de ce champ est minimale sur le point que le robot doit atteindre, et s'augmente au fur et à mesure que l'on s'écarte de ce point. Les obstacles génèrent un champ de potentiel répulsif pour le robot, de valeur supérieure à n'importe quel autre point ne correspondant pas à un obstacle du champ de potentiel. Et le champ de répulsion s'étend autour des obstacles avec une intensité inversement proportionnelle à la distance par rapport à l'obstacle. L'idée est alors de demander au robot de se déplacer dans la direction du gradient de potentiel négatif le plus fort, sur le champ de potentiel global obtenu [MORETTE, 2009] (figure 1.6).

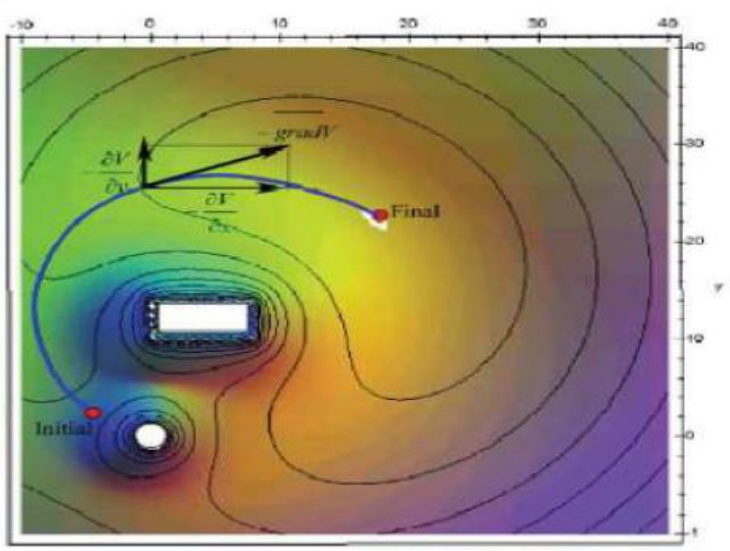


Figure 1.6 : Carte des champs de potentiels autour d'un obstacle [MORETTE, 2009]

L'idée de forces imaginaires agissant sur le robot a été proposée pour la première fois par Khatib, ainsi que Andrews et Hogan [ANDREWS, 1983]. Ces méthodes présentent l'avantage d'être simples à mettre en œuvre, et ont été implantées physiquement sur de vrais robots en 1985 par Brooks [BROOKS, 1986] et en 1989 par Atkin. Malgré le matériel informatique encore limité à cette époque, les calculs de trajectoires/commande étant très rapides avec ce type d'approche, cela permit de premières expérimentations sur des robots relativement lents.

1.7.3.2 Réseaux de neurones

Le neurone est l'entité de base du système de réaction animal, son fonctionnement chez la grenouille a été étudié pour la première fois par les neurologues Mc Culloch et Pitts [LETTVIN, 1959].

Les réseaux de neurones désignent à la fois l'étude de ces systèmes biologiques et leur modélisation informatique, plus ou moins simplifiée, à différentes applications, comme la reconnaissance de caractères, et la navigation des robots mobiles autonomes. Dans ce cadre a été définie la notion de réseaux de neurones artificiels.

Dans un réseau de neurones artificiel (figure 1.7), chaque neurone possède plusieurs entrées et une sortie. Chacune de ses entrées affecte un poids (dit poids synaptique) différent, et si la somme ainsi pondérée des signaux des différentes entrées dépasse un seuil, la sortie prend une valeur positive (le neurone se déclenche) [MORETTE, 2009].

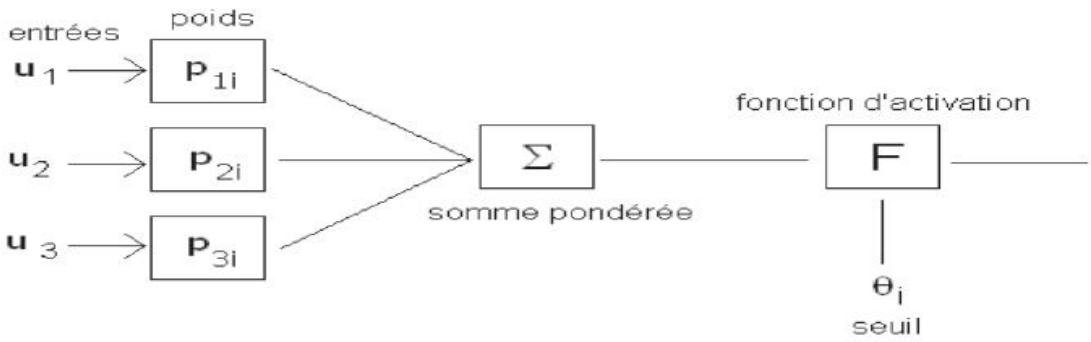


Figure 1.7 : Modélisation d'un réseau de neurones artificiel

Un réseau complet est constitué de différentes couches de neurones. Les premières couches correspondent à des couches de détection, dans le cas de la robotique mobile les informations délivrées par les capteurs du robot correspondent à ces premières couches. Les couches supérieures correspondent plutôt à des couches d'interprétation, et pour un robot elles correspondent aux effecteurs. De plus il y a généralement un certain nombre de couches intermédiaires entre les couches d'entrée et de sortie.

Un apprentissage du réseau est réalisable en adaptant les poids des entrées synaptiques. Pour cela, le principe est de donner au réseau un grand nombre d'exemples, et de rétropropager

l'erreur obtenue entre la sortie du réseau et la sortie attendue. L'algorithme d'apprentissage aura alors pour tâche de régler les différents poids synaptiques pour minimiser cette erreur [MORETTE, 2009].

Plusieurs contrôleurs pour les robots mobiles ont été développés sur ces principes [LEBEDEV, 2005], [BELKER, 2002]. Les dernières générations utilisent un codage en fréquence pour la transition de l'information, comme le font les neurones biologiques.

Wang dans [WANG, 2008] a développé un contrôleur basé sur ce principe, pour un robot unicycle équipé de capteurs ultrasonores.

1.7.3.3 Logique floue

La logique binaire présente l'avantage de la simplicité, mais n'est pas conforme au raisonnement humain. Si on prend l'exemple de la qualification de la proximité d'un obstacle, la logique floue permet de faire intervenir des notions telles que « assez près » ou « très loin », au lieu d'utiliser une définition binaire « obstacle ou pas obstacle ». Celle ci a été formalisée par Zadeh en 1965.

Le principe d'un contrôleur basé sur la logique floue se décline en trois phases : la première phase est appelée la fuzzification, qui transforme les variables d'entrée en variables floues ; dans la deuxième phase on fait appel à une table des règles logiques de comportement du type « si (condition 1) et/ou (condition 2) alors (action sur les sorties) » ; et dans la dernière phase dite défuzzification, qui permet de traduire l'action déterminée par les règles de comportement en commande à envoyer aux actionneurs.

L'étape de fuzzification fait appel à des intervalles flous, qui délimitent l'espace des variables d'entrée en un certain nombre de sous ensembles flous (par exemple pour une proximité, on pourra avoir très proche (contact), assez proche, distance moyenne, assez loin et très loin) ainsi les fonctions d'appartenance pour définir le degré d'appartenance de la variable floue en fonction de la grandeur d'entrée. Ces fonctions peuvent être de type triangle, gaussienne etc. Ainsi pour une mesure de distance donnée, la règle d'appartenance nous informe que « il y a 95% de chances que l'obstacle soit assez proche, 5% de chance que l'on soit en contact ». Cette notion est basée sur le fait qu'il y a toujours des incertitudes concernant les mesures des capteurs et les informations dont on dispose en général.

La seconde étape consiste en l'élaboration de règles de comportement pour le robot, suivant la combinaison des variables floues en entrée. La table des règles de comportement est construite manuellement, par un expert qui va régler le contrôleur. Par exemple pour un robot mobile, une règle a la forme suivante : «s'il y a un obstacle assez proche sur la droite, il faut tourner à gauche et diminuer la vitesse du robot » [MORETTE, 2009].

La dernière étape, appelée défuzzification, consiste à transformer le comportement obtenu par la table des règles, en commande pour le robot. On utilise l'une des méthodes de la défuzzification pour faire cela qui est le centre de gravité, qui va consister à faire la moyenne pondérée des commandes à appliquer.

1.7.4 Méthodes de suivi de trajectoire

La deuxième approche consiste à contrôler le robot pour lui faire suivre une trajectoire précise. Ces méthodes (figure 1.8) nécessitent généralement de déterminer un modèle inverse du robot, c'est-à-dire un modèle qui permet de calculer les commandes à envoyer au robot (espace articulaire) connaissant la trajectoire que l'on souhaite lui faire suivre (dans l'espace cartésien) [MORETTE, 2009].

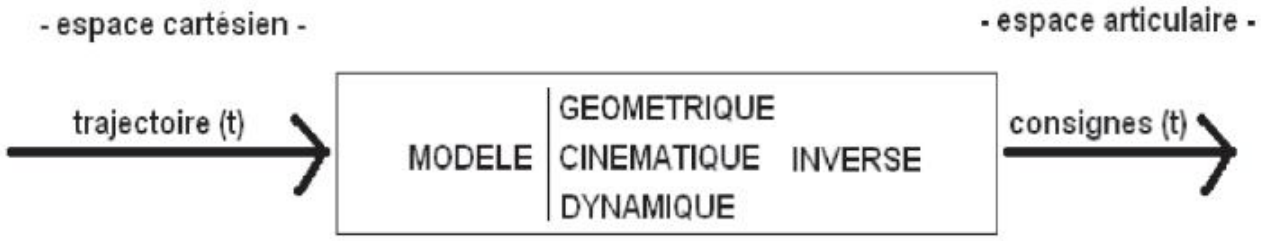


Figure 1.8 : Approche par modèle inverse pour effectuer un suivi de trajectoire

La détermination d'un tel modèle est souvent le point central de ce type de méthode, et le but est d'obtenir la meilleure convergence possible du robot vers la trajectoire de référence.

1.8 Les architectures de contrôle de robots

Un robot est un système complexe qui doit satisfaire à des exigences variées et parfois contradictoires. Un exemple typique pour un robot mobile est l'arbitrage qui doit être fait entre l'exécution la plus précise possible d'un plan préétabli pour atteindre un but et la prise en compte d'éléments imprévus, tels que les obstacles mobiles. Ces arbitrages, que ce soit au niveau de l'utilisation des capteurs, des effecteurs ou des ressources de calcul, sont réglés par un ensemble logiciel appelé architecture de contrôle du robot. Cette architecture permet donc d'organiser les relations entre les trois grandes fonctions que sont la perception, la décision et l'action.

Ces différentes architectures de contrôles peuvent être classées en trois grandes catégories: les contrôleurs hiérarchiques, les contrôleurs réactifs et les contrôleurs hybrides.

1.8.1 Les contrôleurs hiérarchiques

Ces approches s'appuient sur les techniques de l'intelligence artificielle classique. Elles visent à reproduire le mode de raisonnement humain ou tout au moins une certaine vision du mode de raisonnement humain. Le traitement est décomposé en une série d'opérations successives décrites par la figure 1.9.

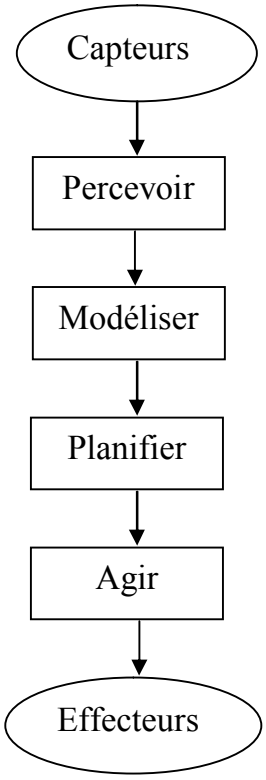


Figure 1.9 : contrôleur hiérarchique

- La première opération consiste à traiter les données sensorielles qui fournissent au robot des informations sur son environnement.
- L'étape suivante consiste à construire ou à mettre à jour, à partir des données capteurs, un modèle du monde dans lequel évolue le robot. Ce modèle peut être par exemple une identification du type et de la position des obstacles que le robot doit éviter.
- Ce modèle est utilisé pour planifier une suite d'états conduisant le robot à effectuer la tâche recherchée.
- La dernière opération a pour but de calculer puis d'exécuter les actions permettant au robot de suivre le plan généré par l'étape précédente.

La partie la plus importante, et la plus gourmande en calcul, est celle qui concerne la modélisation et la planification qui peuvent prendre un temps assez long, et ceci notamment en environnement peu structuré. Le temps disponible pour l'action est par conséquent beaucoup plus court. L'intervalle de temps entre les phases de perception et d'action peut de plus être relativement long.

Ces architectures ont rapidement montré leurs limites et leur incapacité à fonctionner dans un environnement qui ne soit pas statique et simplifié à l'extrême [YAHYAOUI, 2006].

Les points forts sont

- ✓ L'avantage essentiel de ces architectures réside dans la possibilité d'intégrer des raisonnements de haut niveau (niveau mission, planification) qui s'appuient sur des modèles assez complets (cartes par exemple) de l'environnement dans lequel évolue le robot. Cet aspect est absent dans les architectures purement réactives.

Les points faibles sont

- ✓ Leur principal inconvénient provient de leur faible vitesse de réaction. Elles sont en effet totalement incapables de prendre en compte des obstacles dynamiques ou non détectés lors de la modélisation. Ceci est dû à la lenteur des phases de modélisation et de planification qui ne peuvent généralement pas être exécutées en temps réel, même lorsqu'elles sont implantées sur de gros calculateurs extérieurs au robot.
- ✓ La construction du modèle sur lequel se base la planification n'est pas une tache aisée. Il est en pratique souvent difficile de relier les données sensorielles aux objets du monde réel. Ceci est dû en grande partie à l'imprécision de l'information délivrée par les capteurs utilisés en robotique.
- ✓ Ces systèmes ne sont pas adaptatifs dans la mesure où il est difficile de réagir à des situations légèrement différentes de celles qu'ils connaissent. Cela exprime une lacune dans la possibilité de généraliser des situations possibles.
- ✓ Ces systèmes sont en général peu robustes de par leur modèle centralisé. Une défaillance d'un module peut provoquer le blocage de toute l'architecture.

L'intelligence artificielle traditionnelle s'est appuyée sur la volonté de reproduire des systèmes vivants, à travers des concepts plus ou moins valides issus d'observations macroscopiques empiriques, ne correspondant pas aux mécanismes réels régissant les êtres vivants.

L'utilisation de ces approches semble donc limitée aux robots évoluant dans des environnements statiques et dont la structure est fortement contrainte et connue a priori. Les limitations de cette approche ont conduit certains chercheurs à développer des systèmes réactifs.

1.8.2 Les contrôleurs réactifs

Les architectures réactives ont pour but de concevoir et de réaliser un robot, capable d'apporter une réponse immédiate à toute nouvelle modification de l'environnement le concernant. Ces approches sont initialement fondées sur l'étude de l'interaction de l'animal avec son environnement naturel (l'éthologie) de manière à essayer de comprendre les mécanismes mis

en œuvre et de les reproduire. Elles sont fondées sur le fait que lorsqu'un être humain évolue dans un environnement, il ne passe pas son temps à modéliser son environnement, et à planifier ses actions, mais qu'au contraire il réagit à de simples stimuli qui lui permettent de se déplacer vers son but tout en faisant face aux obstacles.

Ainsi, contrairement aux systèmes hiérarchiques, le comportement global du robot apparaît comme le résultant de plusieurs comportements actifs simultanément. Dans ce type d'architectures il n'y a pas de phase de planification ou de modélisation de l'environnement

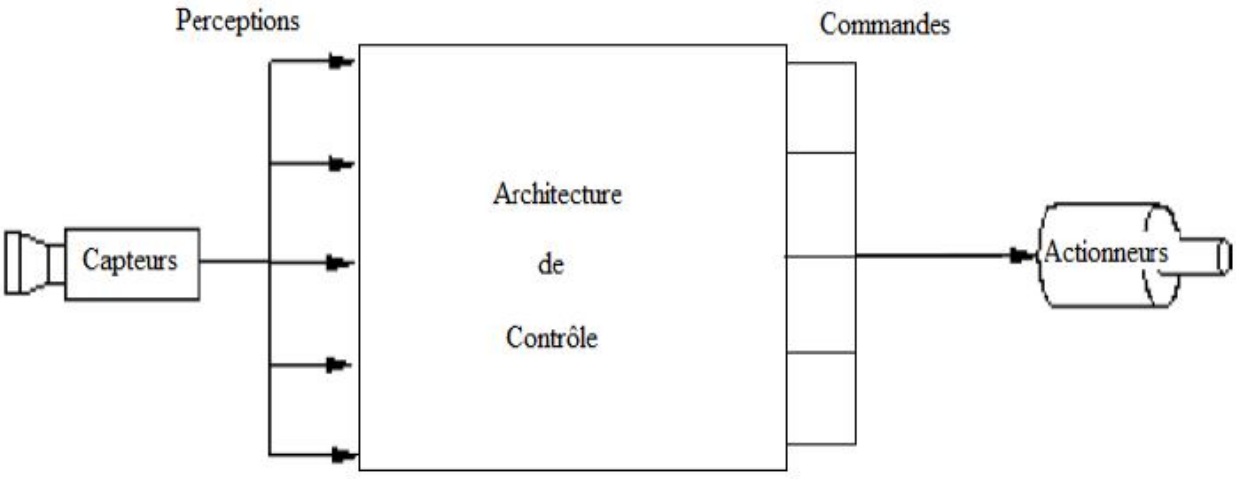


Figure 1.10 : Architecture de contrôle réactive *[AYCARD 1998]*.

Points forts des architectures de contrôle réactives : [ADOUANE, 2005]

- ✓ La robustesse, du fait de la relative indépendance de chacun des comportements les uns par rapport aux autres, si l'un d'eux vient à avoir une défaillance, le robot est toujours capable de réaliser une mission, ce qui est très important lorsque l'on a à faire à un robot autonome;
- ✓ La rapidité de réponse, du fait que les comportements sont directement placés entre les capteurs et les effecteurs. De plus, comme en général, les comportements sont simples à câbler, on peut les réaliser en hardware pour une vitesse encore accrue ;
- ✓ Le fonctionnement en parallèle et la gestion très simple du choix du comportement actif procure une haute réactivité et une certaine tolérance aux pannes ;
- ✓ Faible coût : un robot avec une telle architecture est d'un coût moindre, de part la conception et le test incrémental des niveaux, que celui d'un robot doté d'une architecture cognitive réalisant la même tâche.

Points faibles des architectures de contrôle réactives : [ADOUANE, 2005]

- ✓ Absence de capacité de modélisation et de raisonnement ;
- ✓ Quand de nombreux comportements sont utilisés, il devient très difficile de prévoir le comportement global du système et ainsi de savoir s'il va effectivement réaliser toutes les tâches qui lui sont confiées ;
- ✓ Du fait de leur priorité plus élevée, les comportements de bas niveau – action réflexes de sécurité - prennent le pas sur les comportements en charge de la mission du robot.

1.8.3 Les contrôleurs hybrides

Les approches hiérarchiques et réactives sont diamétralement opposées. Cependant, chacune présente des caractéristiques intéressantes. Pour cela, des chercheurs ont essayé de les combiner en mettant au point des architectures hybrides permettant notamment d'allier des capacités de raisonnement et de décision de haut niveau, s'appuyant sur des représentations abstraites des connaissances, avec des comportements réactifs garantissant robustesse et flexibilité.

La plupart des contrôleurs actuellement utilisés choisissent une solution intermédiaire entre ces deux approches sous la forme d'une architecture hybride. Cette notion est à distinguer de celle de système hybride en intelligence artificielle (ou système hybride neurosymbolique) qui désigne un système alliant des composantes utilisant des techniques issues de l'intelligence artificielle symbolique classique et des réseaux de neurones artificiels.

L'architecture hybride se compose de deux niveaux. Le premier est chargé des tâches de navigation de haut niveau, telles que la localisation, la cartographie et la planification. Pour cela, il s'appuie sur un second niveau réactif qui est chargé d'exécuter les commandes avec le plus de précision possible et de gérer les éléments non modélisés de l'environnement tels que les obstacles inconnus ou dynamiques. L'action conjointe de ces deux niveaux permet de réagir rapidement face aux variations imprévues de l'environnement, tout en permettant la réalisation d'actions planifiées à plus long terme.

Le bas niveau de ces architectures peut être réalisé sous forme de comportements, tels que ceux utilisés dans les architectures réactives. Ces comportements sont des boucles sensorimotrices qui relient les actions aux perceptions avec une phase de décision très courte, qui assurent la réactivité. Dans le même temps, les informations sensorielles sont utilisées par le haut niveau dans une boucle sensorimotrice à une échelle de temps beaucoup plus longue. C'est la mise en parallèles de ces deux échelles de temps qui fait la force de ces architectures.

1.9 Conclusion

Dans ce chapitre nous avons résumé toutes les notions de base nécessaires à la compréhension du domaine de la robotique mobile. Après une présentation des différents types de robots mobiles, nous avons étudié les outils permettant aux robots de percevoir leur environnement et de s'y repérer, étape primordiale nécessaire à l'autonomie totale des robots mobiles. Ensuite nous avons exposé les approches existantes dans la navigation d'un robot mobile autonome. Enfin nous avons présenté les architectures de contrôle des robots. Dans le chapitre qui suit nous allons présenter un aperçu sur les notions de base de la logique floue.

Chapitre 2

La Logique Floue

2.1 Introduction

Dans le fonctionnement de l'esprit humain, les incertitudes sont particulièrement remarquables. La capacité d'établir des classes d'éléments de la nature ayant des propriétés analogues est très naturelle chez l'homme. Il sait déterminer l'âge approximatif d'un individu en l'observant. Il sait aussi rendre compte de données vagues «Large taille», imprécises «vitesse est environ de 50 km/h », dont la validité n'est pas absolue «dans 85% des cas», soumises à une incertitude «très probable». Il est tout aussi naturel à l'homme de traiter des données affectées d'incertitude, inhérentes à l'univers ou dues à sa méconnaissance de certains facteurs, que d'utiliser des critères subjectifs, donc imprécis [BOUCHON, 1995].

Le souci d'automatiser ou d'assister de façon automatique les actions humaines, qui sont de nature empiriques et empreintes d'imprécisions, dans par exemple le domaine d'aide à la décision ou du contrôle, attire l'intérêt des scientifiques pour l'approche floue et justifie son intense développement au cours de ces dernières années. C'est pour les possibilités qu'elle offre de gérer l'incertitude et l'imprécision.

Les bases théoriques de la logique floue ont été formulées en 1965 par le professeur Lotfi A. Zadeh, de l'Université de Berkeley en Californie [ZADEH, 1965]. Il a introduit la notion de sous-ensemble flou pour répondre aux problèmes auxquels sont confrontés de nombreux systèmes complexes, qui doivent traiter des informations qui sont de nature imparfaite, son concept de base est de graduer l'appartenance à un ensemble, c'est un moyen efficace pour prendre en compte l'imprécision dans la connaissance et de formaliser le processus de raisonnement humain. De nombreuses applications basées sur la logique floue ont été développées dans divers domaines, dans lesquelles aucun modèle déterministe n'existe ou n'est pas implémentable. Ainsi que dans des situations pour lesquelles l'imprécision sur les données rend le contrôle par des méthodes classiques impossible.

2.2 Logique classique et logique floue

Dans le cadre de la logique classique, une proposition est soit vraie, soit fausse (1 ou 0). Par exemple, la logique classique peut facilement partitionner la température d'une pièce en deux sous-ensembles, «moins de 15 degrés» et «15 degrés ou plus». La figure 2.1a montre le résultat de cette partition. Toutes les températures de moins de 15 degrés sont alors considérées comme appartenant à l'ensemble «moins de 15 degrés». On leur affecte une valeur de 1. Toutes les températures atteignant 15 degrés ou plus ne sont pas considérées comme appartenant à l'ensemble «moins de 15 degrés». On leur attribue une valeur de 0. Cependant, le raisonnement humain s'appuie fréquemment sur des connaissances ou des données inexactes, incertaines ou imprécises. Une personne placée dans une pièce dont la température est soit de 14.95 degrés soit

de 15.05 degrés, ne fera certainement pas de distinction entre ces deux valeurs. Cette personne sera pourtant capable de dire si la pièce est «froide» ou «chaude», sans pour cela utiliser de température limite ni de mesure précise. La logique floue permet de définir des sous-ensembles, comme «froid» ou «chaud», en introduisant la possibilité pour une valeur d'appartenir plus ou moins à chacun de ces sous-ensembles. La logique floue permet de faire intervenir les notions d'imprécision et d'incertitude dans un système. Cela permet par exemple de faire intervenir une température «d'environ 15 degrés» dans un contrôleur flou. L'incertitude et l'imprécision peuvent également être prises en compte dans le cadre de la logique floue quand on utilise une connaissance issue d'un expert humain.

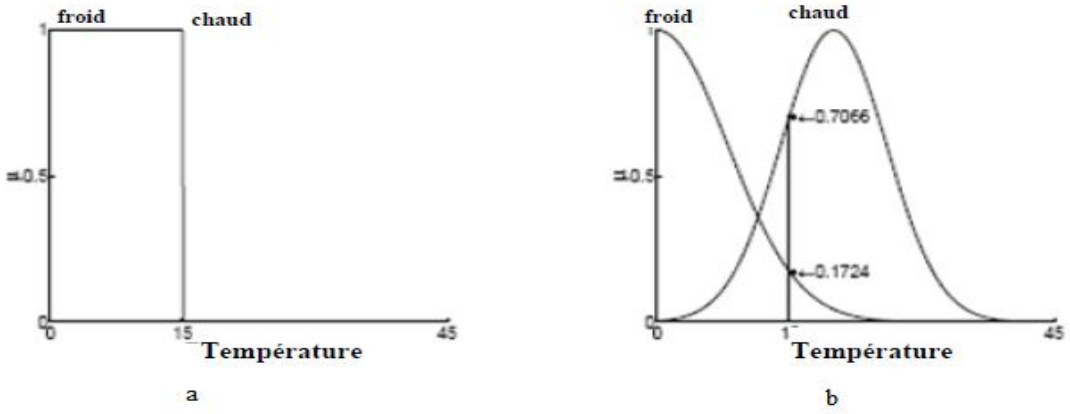


Figure 2.1 : Classification des températures d'une pièce en deux sous ensembles

2.2.1 La logique floue

Actuellement la logique floue a un intérêt majeur de la part de tous ceux qui éprouvent le besoin de formaliser des méthodes empiriques, de généraliser des modes de raisonnement naturels, d'automatiser la prise de décision dans leur domaine, de construire des systèmes artificiels effectuant les tâches habituellement prises en charge par les humains.

Les connaissances qui nous disposons sur une situation quelconque sont généralement imparfaites, soit parce que nous avons un doute sur leur validité (incertaines), soit parce que nous éprouvons une difficulté à les exprimer clairement (imprécises). [BOUCHON, 1993].

En ce qui concerne l'incertain, il a été abordé par la notion de probabilité dès le XVIIe siècle par Pascal et Fermat. Cependant, La notion de probabilité n'est pas adaptée à la représentation d'incertitudes liées à la fiabilité d'un informateur. F. Ramsey en 1931 [RAMSEY, 1931] a introduit les notions de probabilité subjective qui permet de manipuler les degrés de confiance qu'un observateur attribue à la validité de certains faits. Cependant, celle-ci ne permet pas de traiter des croyances subjectives, ni de résoudre le problème posé par les connaissances imprécises.

L'imprécision n'a été prise en considération qu'à partir de 1965 [ZADEH, 1965], lorsque L.A. ZADEH , professeur à l'Université de Californie à Berkeley, a introduit la notion d'ensemble flou «Fuzzy set», à partir de l'idée d'appartenance partielle à une classe, de catégorie aux limites mal définies, de gradualité dans le passage d'une situation à une autre, dans une généralisation de la théorie classique des ensembles, admettant des situations intermédiaires entre le tout et le rien. Les développements de cette notion fournissent des moyens de représenter et de manipuler des connaissances imparfaitement décrites, vagues ou imprécises et ils établissent une interface entre des données décrites symboliquement (avec des mots) et numériquement (avec des chiffres).

La logique floue conduit à raisonner sur de telles connaissances. La théorie des possibilités qui a été introduite en 1978, également par L.A. ZADEH, constitue un cadre permettant de traiter des concepts d'incertitude de nature non probabiliste. Lorsqu'elle est considérée à partir de la notion d'ensemble flou, la théorie des possibilités constitue un cadre permettant d'exploiter, dans un même formalisme, imprécisions et incertitudes[BOUCHON, 1993].

Un nombre important de scientifiques se sont intéressés très tôt à cette nouvelle théorie. La logique floue et la théorie des possibilités se sont développées depuis la fin des années 60, aussi bien en Europe qu'aux États-Unis, en Chine et au Japon. Les premières réalisations de commandes floues de processus industriels sont ainsi apparues en Europe au début des années 70 et la méthode développée a été transformée par les Japonais au début des années 80 en succès industriels.

La logique floue impose une standardisation de la signification des descriptions du système étudié exprimées linguistiquement. La logique floue présente néanmoins l'avantage de permettre le passage de la description d'un expert ou d'un observateur à un autre et la mise au point d'une standardisation consensuelle en cas de divergence de signification entre deux individus.

2.3 Concepts et définitions

2.3.1 Sous ensembles flous

La description imprécise d'une certaine situation, d'un phénomène ou d'une grandeur physique ne peut se faire que par des expressions relatives ou floues à savoir :

-**Q**uelque **Q**, **B**eaucoup **B**, Souvent **S**,

-**C**haud **C**, **F**roid **F**, **R**apide **R**, **L**ent **L**,

-**G**rand **G**, **P**etit **P**, etc.

Ces différentes classes d'expressions floues dites ensembles flous forment ce qu'on appelle des variables linguistiques. Afin de pouvoir traiter numériquement ces variables linguistiques (normalisées généralement sur un intervalle bien déterminé appelé univers de discours), il faut les soumettre à une définition mathématique à base de fonctions d'appartenance qui montre le

degré de vérification de ces variables linguistiques relativement aux différents sous ensembles flous de la même classe [NERZIOU, 2006].

Dans le cas d'un ensemble de référence E, un sous ensemble flou de ce référentiel E est caractérisé par une fonction d'appartenance μ, et de E dans l'intervalle des nombres réels [BUHLER, 1994] qui indique avec quel degré un élément appartient à cette classe.

Définition : Un sous-ensemble flou A dans un univers du discours X est caractérisé par sa fonction d'appartenance $\mu_A(x)$ qui associe à chaque élément x de X une valeur dans l'intervalle des nombres réels [0, 1].

$$\mu_A : X \rightarrow [0, 1]$$

Un sous ensemble flou est caractérisé par un noyau, un support et une hauteur. Dans la figure 2.2 nous indiquons un exemple de sous ensemble normalisé ainsi que son noyau, son support et sa hauteur

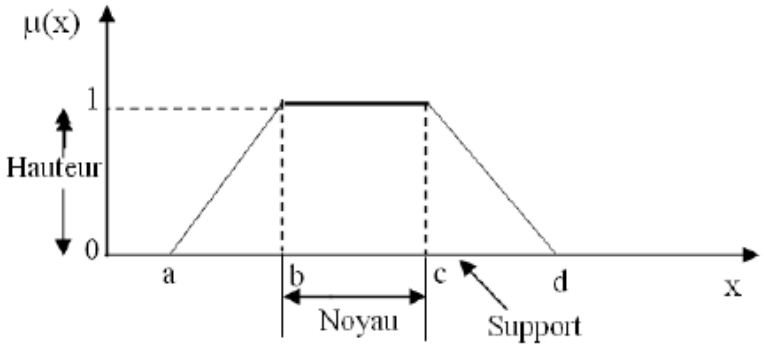


Figure 2.2 : format d'un ensemble flou normalisé

- **Noyau**

  C'est l'ensemble des éléments qui sont vraiment dans *E* :

  $$\text{Noy}(E)=\{x/\ \mu_E(x)=1\}$$

- **Support**

  C'est l'ensemble des éléments qui sont dans E à des degrés divers :

  $$\text{Sup}(E)=\{x/\ \mu_E(x)>0\}$$

- **Hauteur**

  C'est la borne supérieure de la fonction d'appartenance :

  $$h(E)=\text{Max}_{(x\in E)}\ \mu_E(x)$$

- **Ensemble normalisé**

  Un ensemble est dit normaliser s'il est de hauteur 1.

On attribue à chaque valeur de la variable linguistique une fonction d'appartenance μ dont la valeur varie entre 0 et 1, en tenant compte de la classification en un certain nombre d'ensemble

flou. La fonction d'appartenance est désignée par, μ_E (x).L'argument x se rapporte à la variable linguistique, tandis que l'indice E indique l'ensemble concerné. Une valeur précise pour la fonction d'appartenance, liée à une valeur déterminée de la variable x, est désignée par : "facteur d'appartenance".

2.3.2. Les fonctions d'appartenance

On peut utiliser pour les fonctions d'appartenance des formes différentes (fig.2.3), le plus souvent, des formes triangulaires ou trapézoïdales, il s'agit des formes les plus simples.

Composées par morceaux de droites [MILOUDI, 2006]. La forme rectangulaire est utilisée pour représenter la logique classique. Dans la plupart des cas, en particulier pour le réglage par logique floue, ces deux formes sont suffisantes pour déterminer des ensembles flous

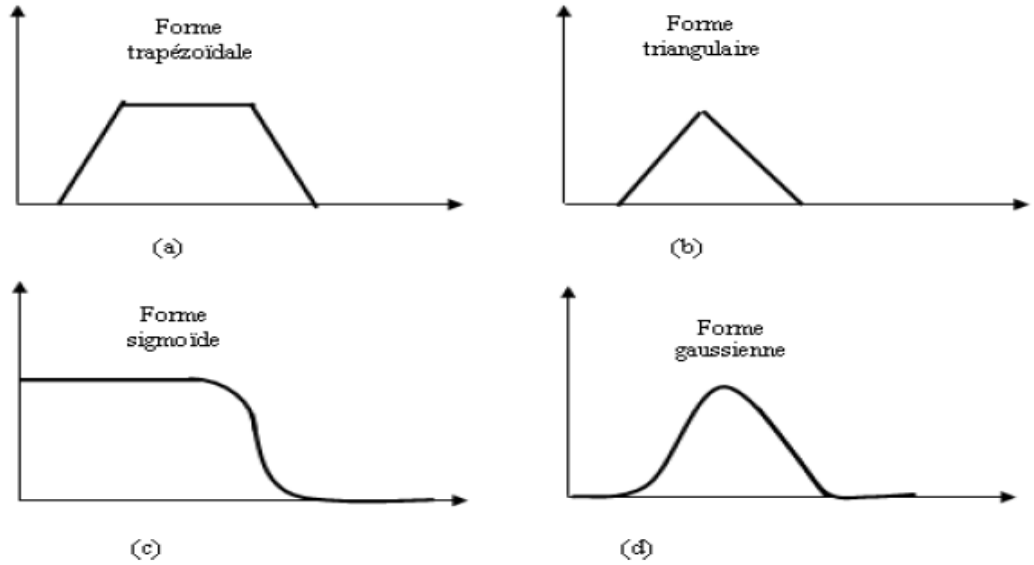


Figure 2.3 : Formes usuelles des fonctions d'appartenance

2.3.3. Les opérateurs de la logique floue

Comme dans la théorie des ensembles classiques [KLIR, 1995], on définit l'intersection, l'union des ensembles flous ainsi que le complémentaire d'un ensemble flou. Ces relations sont traduites par les opérateurs « Et », « Ou » et « Non ». Soit, μ_A (x), μ_B(x) les degrés d'appartenance de l'élément x dans L'univers de discours X (dénommé par $x \in X$)

Opérateur Non

Selon la théorie des ensembles, l'ensemble complémentaire : $C = \overline{A}$ =Non(A)

est défini par les éléments de x qui n'appartiennent pas à l'ensemble A. Dans le cas de la logique floue, cette définition peut être exprimée par les fonctions d'appartenances de la manière suivante :

$$\mu_C(z)=1-\mu_A(x)$$

Opérateur Et

L'opérateur Et correspond à l'intersection de deux ensembles A et B on écrit :

C = A ∩B = A Et B Dans la logique floue, l'opérateur Et est réalisé dans la plupart des cas par la formulation du minimum, appliqué aux fonctions d'appartenance $\mu_A(x)$ et $\mu_B(x)$ des deux ensembles A et B, à savoir : $\mu_C(x) = \min(\mu_A(x),\mu_B(x))$. On parle alors de l'opérateur minimum.

Opérateur Ou

L'opérateur Ou correspond à l'union de deux ensembles A et B. on a donc: C = A ∪ B = A Ou B

La réalisation de l'opérateur ou au niveau de la logique floue se fait en général par la formulation du maximum, appliquée aux fonctions d'appartenance $\mu_A(x)$ et $\mu_B(x)$ des deux ensembles A et B.

2.3.4. Opérations sur les sous ensembles flous

Supposons que A et B sont deux sous-ensembles flous définis dans un univers du discours X par les fonctions d'appartenance μ_A et μ_B . On peut définir des opérations ensemblistes telles que l'égalité, l'inclusion, l'intersection, l'union et le complément grâce à des opérations sur les fonctions d'appartenance.

Egalité : A et B sont dits égaux, propriété que l'on note A = B, si leurs fonctions d'appartenance prennent la même valeur en tout point de X : $\forall x \in X\ \mu_A(x) = \mu_B(x)$

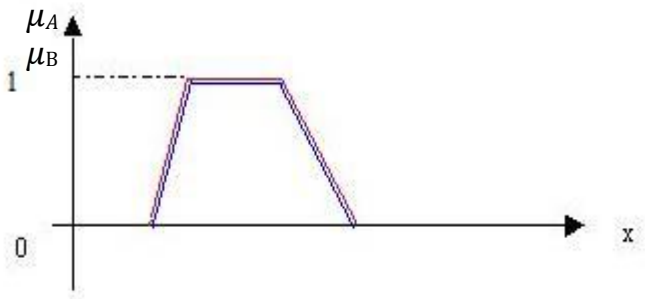


Figure 2.4 : L'égalité de deux ensembles flous

Inclusion : A est dit inclus dans B, propriété que l'on note A⊂ B, si tout élément x de A appartient à B : $\forall x \in X\ \mu_A(x) \leq \mu_B(x)$

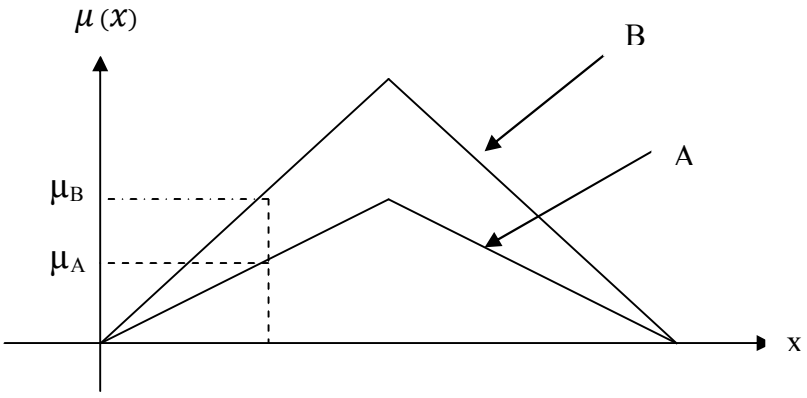


Figure 2.5 : L'inclusion de A dans B

Intersection : L'intersection de A et B, que l'on note A∩B, est le sous-ensemble flou constitué des éléments de X affectés du plus petit des deux degrés d'appartenance μ_A et μ_B .

$\forall x \in X \ \mu_{A \cap B}(x) = min(\mu_A(x), \mu_B(x)) = \mu_A(x) \wedge \mu_B(x)$

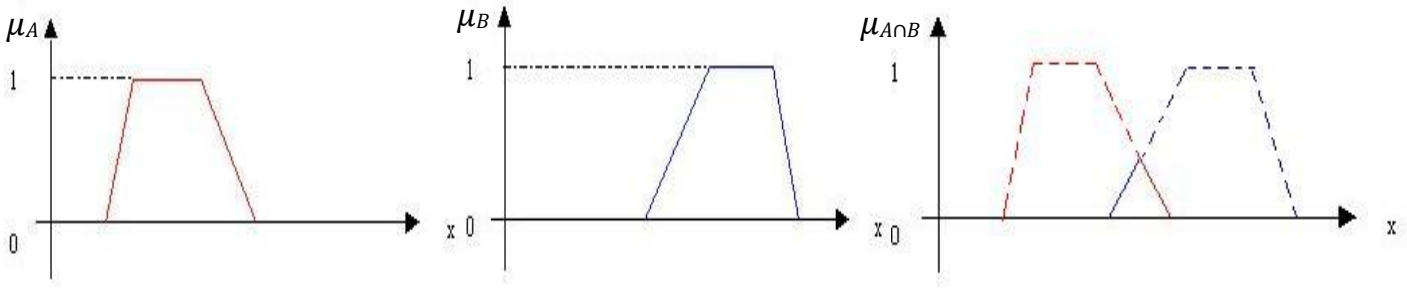


Figure 2.6 : L'intersection de deux ensembles flous

Union : L'union de A et B, que l'on note A∪ B, est le sous-ensemble flou constitué des éléments de X affectés du plus grand des deux degrés d'appartenance μ_A et μ_B :

$\forall x \in X \ \ \mu_{A \cup B}(x) = max\ (\mu_A(x), \mu_B(x)) = \mu_A(x) \vee \mu_B(x)$

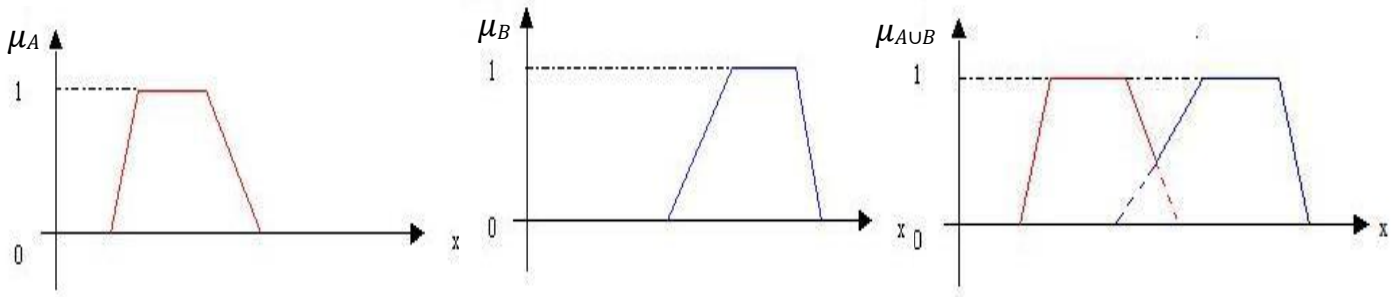


Figure 2.7 : L'union de deux ensembles flous

Complément : Le complément de A, que l'on note A^C, est le sous-ensemble flou de X constitué des éléments x qui appartiennent à A^C et qui n'appartiennent pas à A :

$$\forall x \in X \ \mu_{A^C}(x) = 1 - \mu_A(x)$$

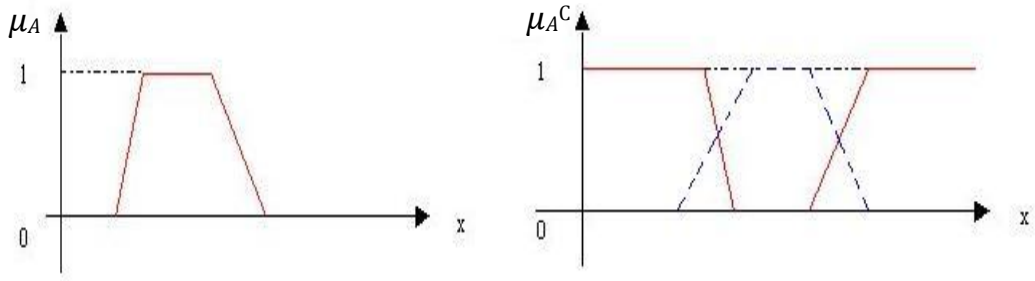


Figure 2.8 : Complément d'un ensemble flou

2.3.5. Relations floues

Une relation floue est un concept qui permet de définir un sous-ensemble flou sur un univers du discours qui est un produit cartésien, tout en tenant compte des relations qui relient les univers du discours initiaux. La fonction d'appartenance définissant ce sous-ensemble flou ne se résume pas forcément à une simple combinaison de fonctions d'appartenance définissant des sous-ensembles flous dans les univers du discours initiaux [DJAABOUB, 2009].

Définition : Une relation floue R sur les univers de références X_1, X_2,..., X_n est définie comme un ensemble flou du produit cartésien $X_1 \times X_2 \times ... \times X_n$ ayant la fonction d'appartenance μ_R.

$$R_X = \{((x_1, x_2, \ldots, x_n), \mu_R(x_1, x_2, \ldots, x_n)) | (x_1, x_2, \ldots, x_n) \in X\}.$$

Où $\mu_R(x_1, x_2, \ldots, x_n)$ est une fonction de mappage :

$$\mu_R(x_1, x_2, \ldots, x_n) : X_1 \times X_2 \times \ldots \times X_n \rightarrow [0, 1]$$

Exemple : La relation floue R: "approximativement égal à" peut être définie sur $\mathcal{R} \times \mathcal{R}$ par la fonction : $\mu_{\mathcal{R}}(x, y) = \frac{1}{1+(x-y)^2}$

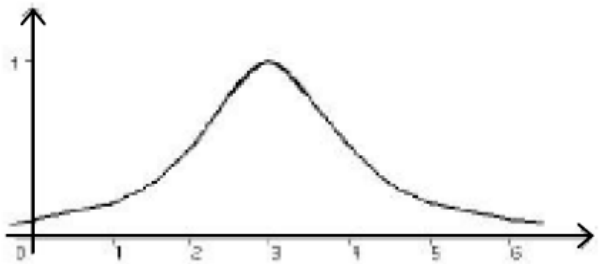


Figure 2.9 : Représentation de la relation x approximativement égal à 3

La différence entre une relation floue et une relation classique (exacte) est que pour la première, toute valeur d'appartenance dans l'intervalle [0,1] est permise alors que pour la seconde, seules les valeurs 0 et 1 sont permises.

2.3.6. Propositions floues

Une proposition floue est définie à partir d'un ensemble de variables linguistiques afin de représenter une connaissance. Par exemple, «la température de la pièce est froide».

Propositions floues élémentaires

Une forme élémentaire de proposition floue est définie à partir d'une seule variable linguistique (V, T(V), X) et exprimée simplement par la phrase [DJAABOUB, 2009] :

$$p : V \text{ est } A$$

Où V est une variable qui prend sa valeur dans l'univers du discours X, et A est l'un des termes linguistiques de T(V). Une valeur particulière V = v appartient à A avec le degré d'appartenance $\mu_A(v)$. Cela permet de définir la valeur de vérité V (p) de la proposition lorsque V vaut v :

$$V(p) = \mu_A(v).$$

Exemple : Supposons que l'on veuille définir la température V d'une pièce. La proposition floue correspondante est exprimée par la phrase :

p : V est tiède.

A partir de la fonction d'appartenance définissant «tiède», nous trouvons que le degré d'appartenance d'une température de 15 degrés au sous-ensemble flou «tiède» est de 0.7, comme on le voit sur la figure 2.10. La valeur de vérité V (p) de la proposition p lorsque la température vaut 15 degrés est donc de 0.7.

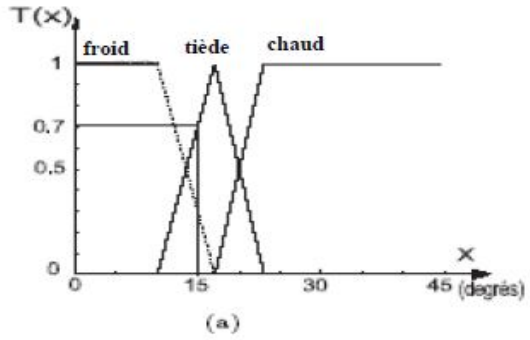


Figure 2.10 : Valeur de vérité de la proposition floue p : la température est tiède

Propositions floues générales

Une proposition floue générale est obtenue par la composition de propositions élémentaires «x est A», «y est B»,... pour des variables x, y,….. [DJAABOUB, 2009]. Habituellement les propositions floues générales sont classées en quatre types :

- **La conjonction de propositions floues élémentaires**

p : (x_1 est A_1) et · · · et (x_n est A_n)

Dans ce cas, la conjonction est associée au produit cartésien $A_1 \times A_2 \times \cdots \times A_n$ caractérisant la variable conjointe $(x_1, x_2, \cdots, x_n)$ sur les univers de discours $X_1 \times X_2 \times \cdots \times X_n$. Sa valeur de vérité est alors définie par :

$$V(p) = \min \{ \mu_{A1}(x_1), \ldots, \mu_{An}(x_n) \}$$

- **La disjonction de propositions floues élémentaires**

p : (x_1 est A_1) ou . . . ou (x_n est A_n)

La valeur de vérité de la disjonction sur les univers du discours $X_1 \times X_2 \times \cdots \times X_n$ est définie par :

$$V(p) = \max \{ \mu_{A1}(x_1), \ldots, \mu_{An}(x_n) \}$$

- **Les implications entre propositions floues**

Une implication floue associe à toute règle floue de la forme si V est A alors W est B, construite à partir des variables linguistiques (V, T(V), X) et (W, T(W),Y) une relation floue R entre X et

Y, de fonction d'appartenance : La fonction d'appartenance μ_R de cette relation dépend des fonctions d'appartenance

μ_A et μ_B de A et B. $\forall x \in X, \forall y \in Y, \mu_R(x, y) = F(\mu_A(x), \mu_B(y))$

Où F est une fonction particulier appelée fonction d'implication floue, de $[0, 1] \times [0, 1] \rightarrow [0, 1]$. En logique floue, de nombreuses implications floues ont été développées, correspondant chacune à une interprétation différente des règles floues.

Les plus utilisées sont les implications de MAMDANI (minimum) et LARSEN (produit).

- Mamdani

$$\mu_R(x,y) = min(\mu_A(x), \mu_B(x))$$

- Larsen

$$\mu_R(x,y) = \mu_A(x) . \mu_B(x)$$

• **Les combinaisons** de conjonction, disjonction et implication de propositions floues élémentaires. Par exemple, «si (X_1 est A_{11}) et (X_2 est A_{12}) alors (Y est B_1)», etc.

2.3.7. Inférence floue

Un des apports principaux de la logique standard a été la formalisation des méthodes de déduction, qui sont en quelque sorte un outil de raisonnement. Les méthodes de déduction utilisées en logique standard permettent de définir une nouvelle certitude à partir d'autres connaissances certaines. Dans le cadre de la logique floue, il est possible de généraliser les méthodes de raisonnement lorsqu'on dispose de connaissances incertaines ou imprécises. Les méthodes d' inférence, utilisées habituellement en logique standard, peuvent être généralisées dans le cadre de la logique floue pour permettre de raisonner lorsque les règles ou les faits sont connus de façon imparfaite. La méthode d'inférence la plus connue est le modus ponens, qui permet de déduire une nouvelle connaissance en se basant sur la connaissance d'un seul fait et d'une seule règle.

2.3.8 Méthodes d'inférence floue

Nous distinguons une variété importante de méthodes d'inférence floue, mais nous nous contentons d'en présenter les deux méthodes les plus utilisés. L'inférence de Mamdani (max-min) et l'inférence par la méthode additive. Les deux inférences diffèrent par la manière de déterminer les sorties**.** Supposons que la base de connaissances contient n règles d'inférence contenant chacune m prémisses et une même conclusion y est B_i.

Règle 1 : Si (X_1 est A_{11}) et · · · et (X_m est A_{1m}) ; alors (Y est B_1)

Règle 2 : Si (X_1 est A_{21}) et · · · et (X_m est A_{2m}) ; alors (Y est B_2)

. . .

Règle n : Si (X_1 est A_{n1}) et · · · et (X_m est A_{nm}) ; alors (Y est B_n)

La méthode max-min : se décompose en deux étapes : pour chaque règle appliquée, le minimum de degré d'appartenance est retenu dans le résultat. Par contre, si plusieurs règles donnent un même résultat, le maximum de ces résultats est retenu [Cox, 1994]. Ces opérations sont expliquées par :

$$\mu_{Ri}(y) = min\ (\mu_{A\,i1}(x_1), \mu_{Ai2}(x_2), \ldots, \mu_{Aim}(x_m)).$$

$$\mu_{Résultatfinal}(y) = \mu_{Bi}(y) = \max\ (\mu_{R1}(x), \mu_{R2}(x), \ldots, \mu_{Rn}(x)).$$

La méthode additive : Contrairement à la méthode max-min, dans la méthode additive le résultat final d'inférence est le minimum entre un «1», et l'addition de toutes les appartenances individuelles. Cette méthode est représentée par :

$$\mu_{Ri}(y) = min\ (\mu_{A\,i1}(x_1), \mu_{Ai2}(x_2), \ldots, \mu_{Aim}(x_m)).$$

$$\mu_{Résultatfinal}(y) = \mu_{Bi}(y) = min\ (1, \mu_{R1}(x) + \mu_{R2}(x) + \cdots + \mu_{Rn}(x)).$$

2.3.9. La Commande floue

La Commande floue est l'application la plus utilisée de la logique floue. Elle consiste à remplacer les algorithmes de réglage conventionnels par des règles linguistiques. La figure 2.11 illustre les différentes étapes du développement du processus.

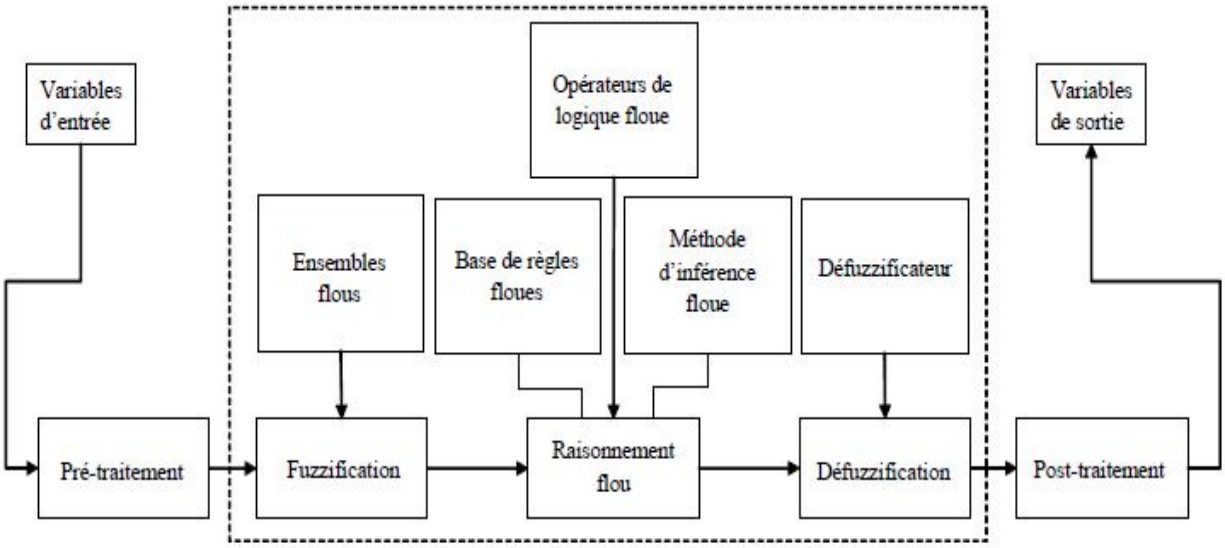


Figure 2.11 : Schéma d'une commande floue

On procède tout d'abord à la partition en sous-ensembles flous des différents univers de discours que le système impose. Ensuite on détermine la base de règles qui va caractériser le fonctionnement désiré du système. Puis on transforme les variables réelles, c'est à dire celles qui ont une réalité physique, en variables floues. On appelle cette étape la fuzzification.

On utilise alors ces variables floues dans un mécanisme d'inférence qui crée et détermine les variables floues de sortie en utilisant les opérations sur les fonctions d'appartenance.

Enfin, on opère à la défuzzification qui consiste à extraire une valeur réelle de sortie à partir de la fonction d'appartenance du sous-ensemble flou de sortie établi par le mécanisme d'inférence..

2.3.10. Système d'inférence floue (SIF)

Nous avons vu que les imprécisions et les incertitudes pouvaient être prises en compte avec la théorie des sous ensembles flous, ces deux imperfections sur les connaissances sont souvent liées.

Pour les systèmes dont le comportement n'est pas précisément connu (incomplet) des outils comme les systèmes d'inférence floue permettent à la fois de décrire son comportement mais aussi d'en exploiter les résultats [BISGAMBIGLIA, 2008].

Les systèmes d'inférence floue (SIF) sont composés d'une collection de règles floues qui ont la forme générale : "Si p alors q".

La conception des SIF s'appuie en général sur des connaissances expertes pour la définition des termes linguistiques correspondant à chaque variable (ensemble de fonctions d'appartenance) d'une part, et sur des algorithmes d'apprentissage pour la génération des règles d'autre part. Par exemple, les termes linguistiques associés à la vitesse du Robot (Petite, Moyenne, Grande) correspondent à des vitesses définies dans la navigation réactive d'un robot mobile.

Les SIF sont à utiliser quand : (1) il existe une expertise humaine que l'on veut exploiter et introduire dans un système automatique, (2) on veut extraire des connaissance à partir de données numériques, en les exprimant dans un langage proche du langage naturel, (3) on veut réaliser une interface homme-machine, donner des explications ou établir des diagnostics immédiatement interprétables.

2.3.10.1 Structure interne d'un système flou

De manière classique, le fonctionnement interne d'un système flou repose sur la structure présentée par la figure 2.12 qui inclut quatre blocs:

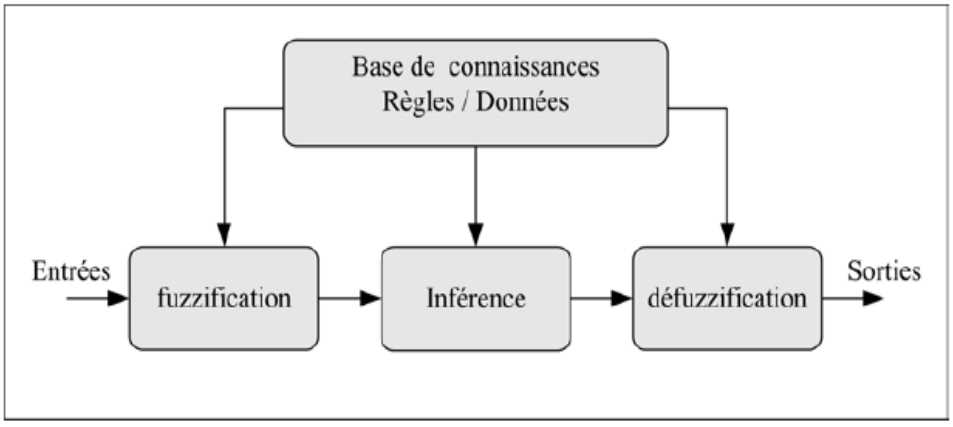


Figure 2.12 : Structure interne d'un système flou

La réalisation d'un système d'inférence passe par plusieurs étapes, la fuzzification et la défuzzification des variables d'entrées et de sorties, et la réalisation d'un moteur d'inférence.

La fuzzification

La fuzzification consiste à définir les fonctions d'appartenance pour les différentes variables. Il s'agit de la conversion numérique / linguistique, ainsi que le traitement des grandeurs d'entrées et de leur transformation en variables linguistiques **[**DRAINCOV, 1996**]**.

a\ Opérateur de fuzzification

Cette étape s'occupe de la transformation des valeurs numériques aux entrées en valeurs floues (linguistiques) en utilisant les bases de données. Comme le traitement des données dans un SIF est basé sur la théorie des ensembles flous, la fuzzification doit être faite a priori. Pour la fuzzification proprement dite, il faut choisir la stratégie de fuzzification et effectuer l'opération de fuzzification qui a pour forme symbolique :

x = fuzzification (x_0)

Où : x_0 est la valeur numérique de l'entrée.

x : est un ensemble flou.

Cet opérateur calcule le degré d'appartenance à un ensemble flou une entrée déterministe donnée.

b\ Définition de fonctions d'appartenance

La stratégie de fuzzification comprend le choix des fonctions d'appartenance. Suivant les plages de valeurs, il existe deux méthodes de définition ; Numérique pour les valeurs discrètes et fonctionnelle pour les valeurs continues.

Les formes des fonctions d'appartenance courantes sont : trapézoïde, triangulaire, gaussienne et sigmoïde **[**DRAINCOV, 1996**]**.

Bases de connaissance

Elle comporte une connaissance dans le domaine d'application et le résultat de commande prévu. Il consiste en « base de données » et en « base de règles linguistiques (floues) de commande ».

- La base de données effectue des définitions qui sont nécessaires pour établir les règles de commandes linguistiques et manipuler les données floues dans un SIF La base de règles est l'ensemble d'expressions linguistiques basé sur la connaissance d'un expert ou bien la logique de fonctionnement du processus. Cette connaissance est formulée sous forme de règles « si- alors », chacune de ces règles est composées d'une ou plusieurs prémisses reliées entres elles généralement par l'opérateur flou « Et » ; et une conclusion précédée de l'opérateur « Alors ». L'ensemble des règles floues sont reliées entre elles généralement par l'opérateur flou « Ou » **[**BAGHLI, 1999**]**.

Inférence

La stratégie de réglage dépend essentiellement des inférences adaptées qui lient les grandeurs mesurées qui sont les variables d'entrée (transformé en variable linguistique à l'aide de la fuzzification) à la variable de sortie, cette dernière est exprimée comme variable d'une table de règles d'un SIF.

$$\begin{cases} R_1 : Si\ A_{11}(X_1)\ et\ A_{12}(X_2)\ et\ .....A_{1n}(X_n)\ alors\ B_1(Y) \\ \qquad\qquad Ou \\ R_2 : Si\ A_{21}(X_1)\ et\ A_{22}(X_2)\ et\ .....A_{2n}(X_n)\ alors\ B_2(Y) \\ \qquad\qquad . \\ \qquad\qquad . \\ \qquad\qquad . \\ \qquad\qquad Ou \\ R_n : Si\ A_{n1}(X_1)\ et\ A_{n2}(X_2)\ et\ .....A_{nn}(X_n)\ alors\ B_n(Y) \end{cases}$$

B(Y) : conséquence.

On remarque que cette structure linguistique est caractérisée par les données d'entrée, les règles, les connecteurs (Et) et (Ou) et pour une entrée donne plusieurs actions de contrôle différentes mais l'action générée doit être unique.

Pour le contrôle par la logique floue, on utilise en général l'une des méthodes d'inférence suivantes **[**GODJEVAC, 1999**]** :

- Méthode d'inférence max- min (Mamdani).
- Méthode d'inférence max- prod.
- Méthode d'inférence somme- prod.

Chacune de ces trois méthodes utilise un traitement numérique propre des opérateurs de la logique floue :

- La Méthode d'inférence max-min, réalise l'opérateur Et par la fonction min, la conclusion Alors de chaque règle est réalisé par la fonction min et la liaison entre toutes les règles (opérateur Ou) par la fonction max. La dénomination de cette méthode, dite max-min ou " implication de Mamdani ", est due à la façon de réaliser les opérateurs Alors et Ou de l'inférence.
- La méthode d'inférence max-prod réalise en général, au niveau de la condition, l'opérateur " Ou " par la formation du maximum et l'opérateur " Et " par la formation du minimum. Par contre, la conclusion dans chaque règle, introduite par " Alors ", qui lie le facteur d'appartenance de la condition avec la fonction d'appartenance de la variable de Sortie par l'opérateur "Et", est réalisée cette fois-ci par la formation du produit. L'opérateur "Ou" qui lie les différentes règles est réalisé de nouveau par la formation du

maximum. Dans cette méthode l'opérateur " Ou ", liant les différentes règles est réalisé par la formation du maximum et " Alors " est réalisé par la formation du produit, d'où la désignation de cette méthode d'inférence par max-prod.

- Par opposition aux méthodes d'inférence précédentes, la méthode d'inférence somme-prod réalise, au niveau de la condition, l'opérateur " Ou " par la formation de la somme, plus précisément par la valeur moyenne, tandis que l'opérateur "Et" est réalisé par la formation du produit. La conclusion de chaque règle, précédée par "Alors ", liant le facteur d'appartenance de la condition avec la fonction d'appartenance de la variable de sortie par l'opérateur " Et", est réalisée par la formation du produit. L'opérateur "Ou" qui lie les différentes règles est réalisé par la formation de la somme, donc de la valeur moyenne. Dans ce cas, l'opérateur "Ou" liant les règles est réalisé par la formation de la somme et " Alors " est réalisé par la formation du produit : ainsi s'explique la désignation par prod-somme de cette méthode d'inférence.

Défuzzification

Les méthodes d'inférence fournissent une fonction d'appartenance μ(Y) pour la variable de sortie Y, il s'agit donc d'une information floue. Etant donné que l'organe de commande nécessite un signal de commande précis à son entrée ; il prévoit une transformation de cette information déterminée. Il existe plusieurs méthodes de défuzzification, mais les plus utilisées sont **[**DRAINCOV, 1996**]** :

a- Défuzzification par le centre de gravité

C'est la méthode de défuzzification la plus utilisée, dans cette méthode, la valeur déterministe (non floue) de la variable de sortie se calcule à partir de la surface totale de toutes les fonctions d'appartenance de la sortie selon l'expression :

$$Y^* = \frac{\int Y \ \mu(Y) dY}{\int \mu(Y) dY}$$

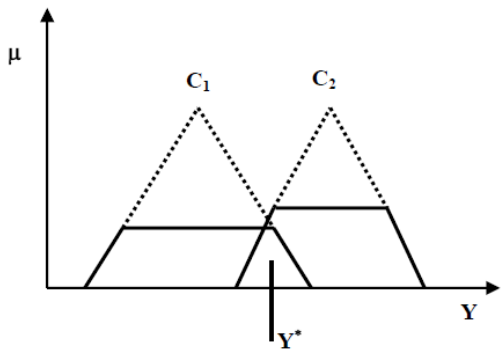


Figure 2.13 : Défuzzification par la méthode de centre de gravité

b- Déffuzzification par la méthode de maximum

Cette méthode consiste à prendre la commande locale ayant le degré d'appartenance maximal

$$Y^* = \max(\mu_{Ri}(Y))$$

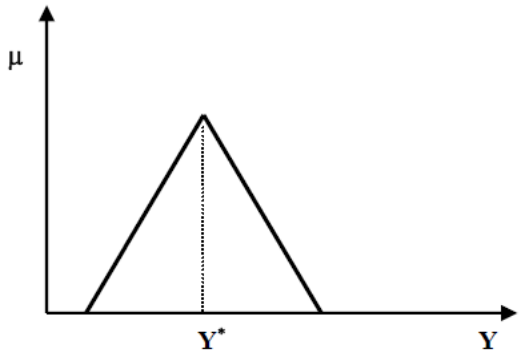


Figure 2.14 : Défuzzification par la méthode de maximum

c- Défuzzification par la méthode de moyenne des maximums

Cette méthode correspond à un simple calcul de la moyenne arithmétique des valeurs ayant le plus grand degré d'appartenance.

$$Y^* = \frac{Y_1 + Y_2}{2}$$

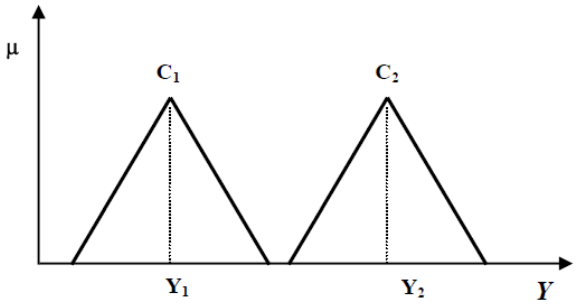


Figure 2.15 : Défuzzification par la méthode de la moyenne des maximums

2.3.11. Exemple d'application

Dans cet exemple nous appliquons la théorie de la logique floue. Soient les deux univers de discours suivants :

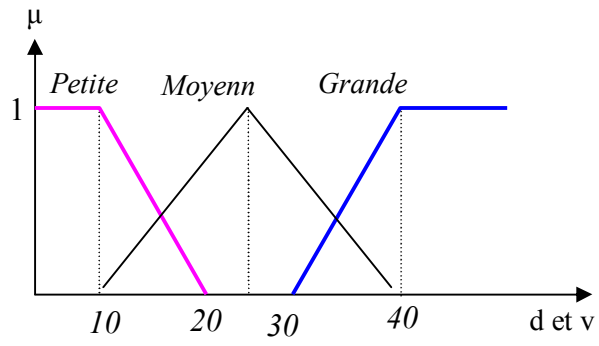


Figure 2.16 : Univers de discours pour d et v

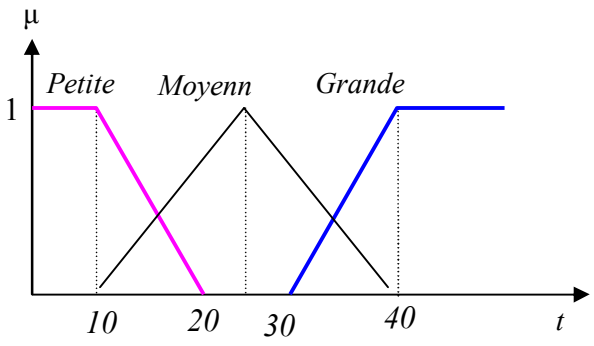


Figure 2.17 : Univers de discours pour t

Nous avons la table des règles suivantes :

t d	P	M	G
P	P	M	M
M	M	M	M
G	P	M	G

Donc les règles sont définies comme suit :

R1 : Si t est P Et d est P Alors v est P

R2 : Si t est P Et d est M Alors v est M

R3 : Si t est P Et d est G Alors v est M

R4 : Si t est M Et d est P Alors v est M

R5 : Si t est M Et d est M Alors v est M

R6 : Si t est M Et d est G Alors v est M

R7 : Si t est G Et d est P Alors v est P

R8 : Si t est G Et d est M Alors v est M

R9 : Si t est G Et d est G Alors v est G

Soit le tableau suivant qui donne certaines valeurs pour la fonction d'appartenance :

t/d $\mu(x)$	0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
P	1	1	1	0.8	0.4	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0.14	0.4	0.67	0.94	0.8	0.52	0.26	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	0.2	0.6	1	1	1	1	1	1

Nous allons calculer la valeur de la commande pour les entrées t=16, et d=12 .

D'après la règle 1 :

R1 : **Si** t est P **Et** d est P **Alors** v est P

$\mu_{R1} \equiv (\mu_p \text{ Et } \mu_p) \rightarrow \mu_p$

$\equiv (\mu_p(16) \text{ Et } \mu_p(12)) \rightarrow \mu_p(P)$

L'opérateur Et représente l'opérateur min et l'implication est définie par l'opérateur min

$\mu_{R1} \equiv \min(0.4,0.8,(1,1,1,0.8,0.4,0,0,0,0,0,0,0,0,0,0,0))$

$\equiv (0.4,0.4,0.4,0.4,0.4,0,0,0,0,0,0,0,0,0,0,0)$

de même pour les autre règles linguistiques, nous trouvons :

μ_{R2} ≡min(0.4,0.14,(0,0,0,0.14,0.4,0.67,0.94,0.8,0.52,0.26,0,0,0,0,0,0))

≡(0, 0 ,0 ,0.14,0.14,0.14,0.14,0.14,0.14,0.14,0,0,0,0,0,0)

μ_{R3} ≡min(0.4,0,(0,0,0,0,0,0,0,0,0.2,0.6,1,1,1,1,1,1))

≡(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

μ_{R4} ≡min(0.4,0.8,(0,0,0,0.14,0.4,0.67,0.94,0.8,0.52,0.26,0,0,0,0,0,0))

≡(0,0,0,0.4,0.4,0.4,0.4,0.4,0.26,0,0,0,0,0,0,0)

μ_{R5} ≡min(0.4,0.14,(0,0,0,0.14,0.4,0.67,0.94,0.8,0.52,0.26,0,0,0,0,0,0))

≡(0,0,0,0.14,0.14,0.14,0.14,0.14,0.14,0.14,0,0,0,0,0,0)

μ_{R6} ≡min(0.4,0,(0,0,0,0,0,0,0,0,0.2,0.6,1,1,1,1,1,1))

≡(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

μ_{R7} ≡min(0,0.8,(1,1,1,0.8,0.4,0,0,0,0,0,0,0,0,0,0,0))

≡(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

μ_{R8} ≡min(0.4,0.14,(0,0,0,0.14,0.4,0.67,0.94,0.8,0.52,0.26,0,0,0,0,0,0))

≡(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)

μ_{R9} ≡min(0,0,(0,0,0,0.14,0.4,0.67,0.94,0.8,0.52,0.26,0,0,0,0,0,0))

≡(0.4,0.4,0.4,0.4,0.4,0,0,0,0,0,0,0,0,0,0,0)

Nous avons donc neuf sous ensembles flous caractérisés par les fonctions d'appartenance :

μ_{R1} μ_{R2} , μ_{R3} , μ_{R4} , μ_{R5} , μ_{R6} , μ_{R7} , μ_{R8} , μ_{R9} respectivement.

Ces fonctions sont exploitées de la façon suivante :

R=($\mu_{R1,}$ μ_{R2} , μ_{R3} , μ_{R4} , μ_{R5} , μ_{R6} , μ_{R7} , μ_{R8} , μ_{R9})

Nous avons donc μ_R =max($\mu_{R1,}$ μ_{R2} , μ_{R3} , μ_{R4} , μ_{R5} , μ_{R6} , μ_{R7} , μ_{R8} , μ_{R9})

=(0.4,0.4,0.4,0.4,0.4,0.4,0.4,0.4,0.4,0.4,0.4,0.26,0,0,0,0,0,0)

La défuzzification est donnée par :

1. la méthode de centre gravité :

$$u=\frac{\sum_{i=0}^{16} u_R(u_i).u_i}{\sum u_i}$$ =(0.4*(0+4+8+12+16+20+24+28+32)+0.26*36+0*40+0*44+0*48+0

*52+0*56+0*60)/(0.4+0.4+0.4+0.4+0.4+0.4+0.4+0.4+0.4+0.4+0.26++0+0+0+0+0+0)

=7.21/3.86=18.65

Donc la sortie de ce système est V=18.65.

2.4 Conclusion

Dans ce chapitre après la présentation des différents concepts de la logique floue, et les principes d'inférence floue, nous avons décrit les systèmes d'inférence floue. Cette approche permet de tenir compte à la fois des connaissances d'un expert humain, de l'incertitude et de l'imprécision des données traitées. Cela va permettre de créer des systèmes intelligents de manière facile et qui possèdent des capacités de raisonnement et de prise de décision proche à celles de l'être humain.

Dans le chapitre suivant nous allons présenter les bases théoriques de la modélisation et la simulation de systèmes et plus précisément le formalisme DEVS.

Chapitre 3

Le Formalisme DEVS

3. Modélisation & Simulation

On ne résout pas un problème directement sur un système *réel*, mais toujours au travers d'un modèle que nous construisons de ce système. « Un modèle *M* d'un système *S* pour une expérimentation *E* est toute chose à laquelle on peut appliquer *E* pour répondre à des questions concernant *S* » [MINSKY, 1956]. Un modèle est une forme intelligible d'un système construit pour permettre de trouver une réponse à un problème précis. En agissant sur les modèles à partir de jeux de données et d'une base de temps on définit une simulation.

3.1 Introduction à la Modélisation et Simulation

La théorie de la modélisation et simulation a été introduite dans les années 70. Bernard P. Zeigler a proposé, à cette époque, une décomposition de la théorie de la modélisation selon deux aspects orthogonaux [ZEIGLER, 1976] :

1- Les **niveaux de spécification** sont les niveaux qui définissent le comportement du système et les mécanismes qui permettent d'exprimer sa dynamique. Ces niveaux font référence à ses propres travaux ainsi qu'à ceux de George Klir qui définit des niveaux de connaissance des systèmes [KLIR, 1985].

2- Les **formalismes de spécification** des systèmes définissent des classes de modèles. Un modèle est construit en respectant un paradigme. Un paradigme est un ensemble de concepts, de lois et de moyens visant à définir une collection de modèles.

3.2 Les niveaux de spécification d'un système

« Un système est une source potentielle de données » pour B.P. Zeigler, mais il reste à sélectionner les données pertinentes et à les observer. Pour cela, [KLIR, 1985] introduit une représentation de la spécification d'un système décomposée en niveau de connaissance.

Le **niveau Source 0** de la représentation de [KLIR, 1985] identifie la partie du monde réel à modéliser et les moyens par lesquels l'observer.

Le **niveau Donnée 1** définit une base de données de mesures et d'observations faites pour le système source.

Le **niveau Génératif 2**, donne la capacité de recréer les données précédentes utilisant une représentation plus compacte sous forme de formule. Le niveau génératif constitue la connaissance qui ne figure pas au niveau du système de données.

Enfin, le **niveau Structure 3** indique comment produire des données en termes de composantes systèmes inter connectées.

Parallèlement, B.P. Zeigler [ZEIGLER, 2000] a proposé une représentation des niveaux de spécification (Figure 3.1) assez proche de celle de G. Klir mais plus orienté sur la modélisation et simulation.

Niveau	Nom de Spécification	Correspondance avec Klir	Ce qui est connu à ce niveau
0	Cadre d'observation	Système source	Comment stimuler le système avec des entrées. Quelles variables mesurer et comment les observer au travers d'une base de temps
1	Comportement E/S	Données	Les données collectées depuis un système source indexées dans le temps, elles consistent en des couples entrées sorties
2	Fonction E/S		L'état initial. A partir d'un état initial, chaque stimulus d'entrée produit une sortie unique
3	Transition d'état	Génératif	Comment les états sont affectés par les entrées. A partir d'un état et d'une entrée quel est l'état suivant après que le stimulus d'entrée soit reçu ; quel événement de sortie est généré par l'état
4	Couplage de composants	Structure	Les composants et comment ils sont couplés entre eux. Les composants peuvent être spécifiés à un niveau inférieur ou être eux-mêmes des systèmes structurés construits hiérarchiquement

Figure 3.1 : Hiérarchie de spécification des systèmes

3.3 Les formalismes de spécification des systèmes

Il existe deux classes de méthodes de résolution de problèmes. Les méthodes analytiques et les méthodes basées sur la simulation. Les premières permettent de rechercher une solution générale mais se heurtent au problème de la complexité du modèle. Les méthodes basées sur la simulation permettent de traiter des modèles plus complexes, mais se limitent à des solutions particulières.

B.P. Zeigler [ZEIGLER, 2000] présente une structure pour la modélisation et simulation (Figure 3.2), il y définit les entités essentielles et leurs rapports primordiaux dans la définition d'une

modélisation et simulation. En effet, les termes « modèle » et « simulateur » sont souvent utilisés dans la pratique actuelle sans toujours un réel fondement ; cette structure identifie les questions de base et les problèmes rencontrés dans l'exécution des activités de modélisation et simulation, ils peuvent ainsi être mieux compris et générer des solutions plus cohérentes. Il est donc important de comprendre ce qui est inclus et exclus des définitions.

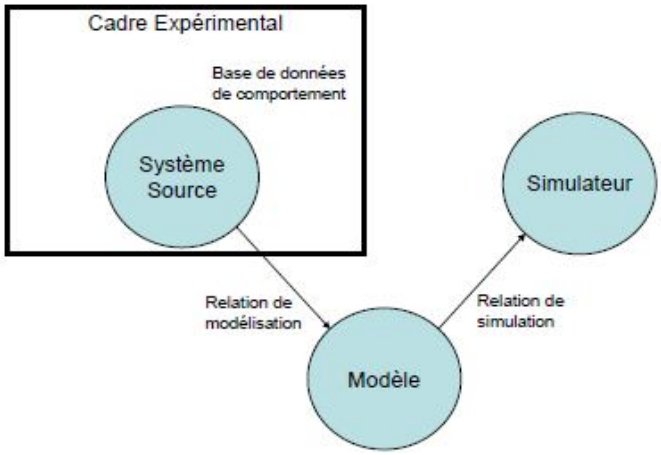


Figure 3.2 : Relations de modélisation [ZEIGLER, 2000]

Les entités de la structure sont : le *système source***,** le *cadre expérimental*, le *modèle,* et le *simulateur* ; ils sont reliés par les relations de *modélisation* et de *simulation*. Chaque entité est formellement caractérisée comme un système à un niveau approprié de spécification d'un système dynamique générique. De même chaque relation est caractérisée comme un morphisme approprié entre deux systèmes dynamiques.

3.3.1 Le Système et son cadre expérimental

Le *système source* est la partie délimitée (niveau 0 de spécification) du monde réel ou virtuel. Il est considéré comme une boite noire sur lequel l'environnement agit et qui réagit à ces stimuli. En d'autres termes, il est vu comme une *source de données observables*, sous formes de trajectoires des variables indexées par le temps. Les données qui sont recueillies de l'observation ou en expérimentant le système sont appelées *base de données de comportement de système* (niveau 1 de spécification). Il faut se rappeler que ces données sont vues ou générées lorsque le système est plongé dans un contexte particulier du monde réel, et pour prendre en considération cela, [ZEIGLER, 2000] identifie ce contexte par des cadres expérimentaux utiles pour le modélisateur.

Un *cadre expérimental* est une spécification des conditions dans lesquelles le système est observé ou dans lesquelles on l'expérimente. Un cadre expérimental est la formulation opérationnelle des conditions dans lesquelles se déroule un projet de modélisation et de simulation. Un cadre est

définit comme un système qui interagit avec le système étudié pour obtenir les données étudiées dans les conditions spécifiées. Dans certains processus de modélisation et simulation, la prise en considération du cadre expérimental est omise ou mal délimitée impliquant ainsi des omissions dans les modèles crées et donc des imprécisions dans les résultats obtenus par simulation.

3.3.2 Le Modèle

Il existe de multiples significations attribuées au mot « modèle ». Un modèle peut être conçu à partir d'une représentation physique, mathématique, ou logique afin de reproduire un système, une entité, un phénomène, ou un processus. La définition, en terme de spécification de système, détient pour avantages de reposer sur une base mathématique formelle et se base sur une sémantique clairement définie que chacun peut comprendre de façon non équivoque.

D'un point de vue général, un *modèle* est une spécification d'un système réel ou virtuel. Dans le contexte de la modélisation et simulation, la spécification de système est classiquement décrite à un haut niveau. Cette spécification peut être décrite par un jeu d'instructions, de règles, d'équations, ou de contraintes pour produire un comportement d'entrée/sortie, ce comportement a pour objectif de produire des données de valeurs proches des données observées sur le système réel.

La relation de modélisation entre le système réel et le modèle définit les parties du système à représenter et comment elles le seront. Plus en détail, construire un modèle de génération de sorties acceptant des trajectoires d'entrée et produisant en réponse des trajectoires de sortie. A ce stade, il apparaît que certaines entrées observées ne produisent pas toujours les mêmes sorties. En effet, la spécification de niveau 1 est non déterministe (ambiguë ou incomplète). Il est donc nécessaire d'introduire la notion « d'état initial » (niveau 2 de spécification) pour trouver une correspondance unique entre les entrées et sorties en fonction d'un état initial du modèle. Ce niveau vise à générer un modèle conceptuel implémentable sur un calculateur à la condition de savoir revenir dans l'état initial [GIAMBIASI, 2001]. Il faut ensuite décrire les mécanismes de transition d'état (niveau 3 de spécification) et la prise en compte de l'état courant pour la génération de sorties.

Par ailleurs, la construction de modèles suppose des hypothèses simplificatrices provenant de plusieurs facteurs : choix du modélisateur, connaissance partielle du système, limitation de l'outil de modélisation. Ces hypothèses permettent de garder à l'esprit la différence entre le système et modèle.

Au niveau du modèle il convient de distinguer le modèle conceptuel et le modèle informatique. Le modèle informatique doit être implémentable sur un calculateur et doit donc respecter certaines règles. A partir des niveaux 3 et 4 de la spécification, il est possible de définir directement des spécifications exécutables si l'on traduit la spécification dans un formalisme avec une sémantique

opérationnelle. Plus en détail, les modèles conceptuels doivent respecter les deux principes suivants pour être également considérés comme modèles informatiques [GIAMBIASI, 1995] :

Le **principe de causalité** définit que le futur ne peut influencer le passé. L'état du système à l'instant présent (t) est indépendant de tout ce qui peut se produire à toute date (t') supérieure à (t).

Le **principe de déterminisme** définit que le futur du système peut être déterminé à partir de son état présent et de son passé. A tout instant (t), il existe une valeur positive (e) telle que le comportement du système puisse être calculé jusqu'à (t + e).

Le modèle représente un point de vue de l'entité modélisée. Ce point de vue est fonction du modélisateur, cependant, la théorie des systèmes distingue classiquement deux points de vue : comportemental et structurel.

3.3.2.1 Le modèle comportemental

Ce modèle est caractérisé par les variables, le temps et les fonctions qui expriment des relations entre les variables et le temps.

L'état du modèle est définit par un sous ensemble nécessaire des variables pour connaître le prochain état. Ce sous ensemble est nommé ensemble des variables d'état.

Une variable d'états est une variable descriptive nécessaire à un instant t pour calculer la valeur future d'au moins une des variables descriptive du modèle. [GIAMBIASI, 2001]

Le comportement du modèle est défini par des fonctions qui expriment le moyen de passer d'état en état. Le modèle comportemental contient donc les règles pour faire évoluer le modèle d'un état à un autre état, en d'autres termes sa dynamique.

De tels modèles forment les composants de base de modèles plus complexes (niveau 4 de spécification) qui sont construits par couplage de modèles entre eux pour former des modèles à spécification de niveau supérieur : les modèles structurels.

3.3.2.2 Le modèle structurel

Ce modèle définit la structure du modèle, en d'autres termes, quels sont les sous-modèles interconnectés qui définissent le comportement global du modèle. Un concept important est la décomposition, il définit comment le système peut être décomposé en sous-composants systèmes. Un autre concept est la composition qui précise comment des modèles peuvent être couplés pour former des modèles plus grands. Ces concepts reposent sur la propriété nommée « fermeture sous la composition » [ZEIGLER, 1999] issue de la théorie des systèmes, cette théorie repose sur la preuve mathématique de l'équivalence d'un modèle comportemental et structurel. Les deux types de modèles peuvent être exprimés dans les mêmes termes de la théorie des systèmes. Il est ainsi possible d'obtenir un modèle ayant une construction hiérarchique composé de sous-modèles

modulaires qui possèdent des portes d'entrée et de sortie interconnectées et qui communiquent afin de reproduire un comportement global.

3.3.3 Les Classes de modèles (formalismes)

Les modèles peuvent être exprimés dans une variété de formalismes qui peuvent être compris comme des moyens pour spécifier les sous-classes de systèmes dynamiques. Ces formalismes sont choisis en fonction de l'objectif de l'étude à réaliser. Les modèles formels, reposant sur des concepts et propriétés mathématiques, possèdent deux dimensions: l'espace et le temps [OREN, 1987].

3.3.3.1 Classification temporelle

La représentation du temps peut être continue ou discrète.

Modèle à temps continu : la variable qui représente le temps prend des valeurs réelles.

Modèle à temps discret : le temps progresse par pas temporel, les valeurs prises de la variable temps sont des multiples du pas.

3.3.3.2 Classification spatiale

La deuxième dimension de classification des modèles est l'espace dont les variables descriptives peuvent prendre des valeurs dans un ensemble fini ou infini.

Modèle à variables continues : les variables descriptives prennent des valeurs réelles.

Modèle à variables discrètes : l'ensemble des valeurs des variables est fini.

3.3.3.3 Formalismes de spécification

Les différents formalismes pour représenter un système résultent de la combinaison des classifications de l'espace et du temps présentées ci-dessus. La Figure 3.3, proposée dans [GIAMBIASI, 2001], présente ces formalismes.

1- Formalisme à équations différentielles : Association d'une base de temps et de variables également continues, les équations définissent les changements d'états.

2- Formalisme à temps discret : Les variables sont ici continues, l'évolution est définie par pas temporels discrets. Ce formalisme est également nommé équation aux différences.

3- Formalisme à événements discrets : Association d'une base de temps continue et de changements d'états discrets. Les changements de valeurs des variables, appelés « **événements** », se produisent à des instants nommés « **date d'occurrence** » d'événements, ces dates sont des nombres réels non négatifs. L'évolution de l'état du modèle survient suite au traitement d'un événement à un instant quelconque.

4- Formalisme à états discrets et temps discret : Association d'une base de temps discrète et de variables d'état discrètes, également nommé machine à états finis synchrones.

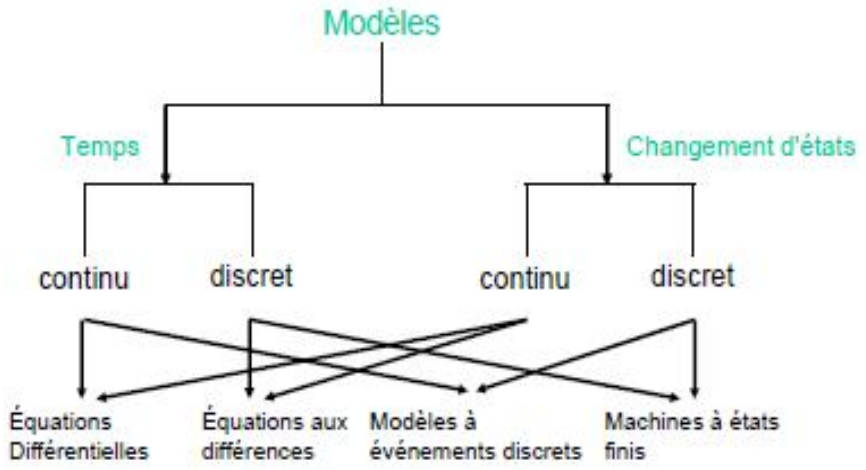


Figure 3.3 : Différents types de modèles

Il faut garder à l'esprit que la simulation sur calculateur digital implique la discrétisation du temps, de l'espace ou des deux. Certains formalismes présentés ci-dessus possèdent déjà une composante discrète, dans le cas contraire, il faudra réaliser la discrétisation du temps ou de l'espace. Pour les modèles décrits par des équations différentielles, le temps est discrétisé dans les techniques d'intégrations numériques.

3.3.4 Le Simulateur

Un *simulateur* est un système de calcul qui obéit à des instructions pour exécuter un modèle et en produire son comportement [ZEIGLER, 2000]. Ce dispositif technique peut être, par exemple, un ordinateur mono processeur, un réseau de processeurs, ou plus abstraitement un processeur logique ou un algorithme.

3.3.4.1 La relation de simulation

La relation de simulation permet de vérifier qu'en obéissant aux instructions définies dans un modèle, le simulateur reproduit un comportement spécifié. C'est-à-dire que le simulateur génère correctement des sorties en fonction d'un état initial, de données d'entrée et du temps.

3.3.4.2 La notion de temps

Il existe plusieurs notions importantes relatives au « temps » dans la simulation. Il est très important de différencier ces notions, en effet la confusion de celles-ci est une source d'erreur courante en simulation. Pour remédier à cela [FUJIMOTO, 2000] présente un ensemble de définitions relatives au temps :

Temps physique : ce temps fait référence au temps dans le système physique.

Temps simulé : est la représentation du temps physique pendant la simulation, c'est une abstraction. Le temps simulé est défini par un ensemble de valeurs complètement ordonnées ou chaque valeur représente un instant du système physique modélisé.

Temps absolu (horloge) : temps réel qui s'écoule pendant l'exécution de la simulation (fourni par une horloge matérielle), une simulation pourra lire ce temps donné par le processeur de l'ordinateur sur lequel se déroule la simulation.

Chacun des concepts de temps utilise une échelle ou un axe du temps sur lequel sont représentées les notions de précédence entre deux instants. Par exemple si (t1) et (t2) représentent deux dates simulées et si ($t_1 < t_2$), alors dire que (t_1) arrive avant (t_2) et (t_2) arrive après (t_1).

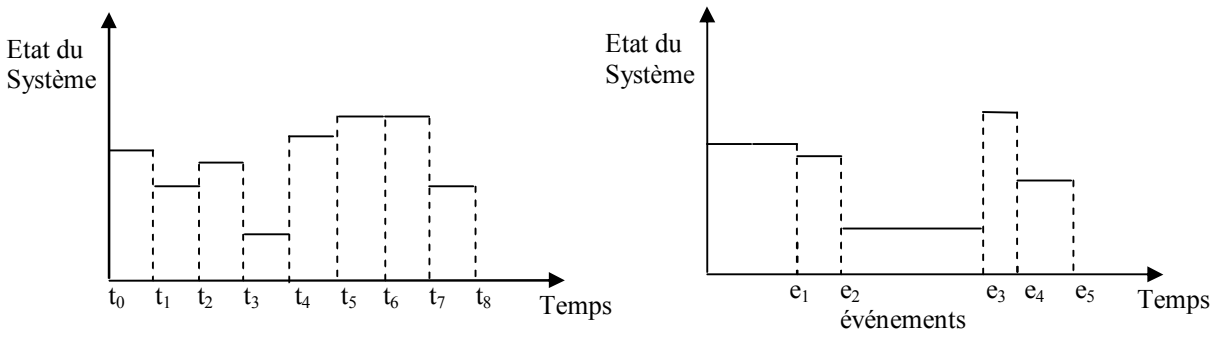
Le temps simulé peut avancer plus lentement que le temps physique si le modèle est complexe à simuler. Le temps peut, dans le cas de modèle plus simple, avancer plus rapidement que le temps physique. Dans ce dernier cas le temps peut avancer aussi rapidement que le temps physique, alors c'est la simulation « temps réel ». Il faut prendre garde dans le cas de la simulation temps réel, le temps simulé avançant de façon inégale, certaines phases de simulation peuvent être associées à des modèles simples dont l'exécution pourra être plus rapide que le temps absolu écoulé. Inversement, certaines phases de simulation peuvent être plus lentes que le temps absolu, dans ce cas la simulation temps réel n'est pas possible car le temps de simulation doit être à tout instant supérieur ou égal au temps physique.

3.3.4.3 Principe de simulation « continue »

La simulation « continue » sur des équations différentielles permet de résoudre des problèmes dont la complexité exclue une solution analytique. La limitation de ces simulations provient du taux d'échantillonnage important nécessaire pour obtenir des solutions non divergentes qui engendre un temps de calcul très important.

3.3.4.4 Principe de simulation discrète

Dans une simulation « discrète », le modèle change d'état en des points discrets sur l'axe des temps simulés. Le programme de simulation met en œuvre l'ensemble des variables d'état et des règles défini dans le modèle permettant de modifier les valeurs des variables au travers du temps de simulation. Les deux catégories de changement d'état les plus communes sont nommées « dirigées par le temps » et « dirigée par les événements ». Ces deux catégories se différencient par le mécanisme d'avancement du temps présentés Figure 3.4.



(a) Simulation dirigée par le temps (b) Simulation dirigée par événements

Figure 3.4 : Schéma évolution de la simulation

3.3.4.5 Evolution dirigée par le temps

Dans une simulation dirigée par le temps (ou synchrone), le temps de simulation est une séquence de pas de tailles égales (Figure 3.4. a). La simulation est cadencée par une horloge globale qui dirige l'évolution de pas en pas. Dans ce cas, l'algorithme traite un ensemble d'actions associées à un processeur logique pour la plage de temps comprise entre la dernière date d'occurrence et le pas. Il faut noter qu'il peut y avoir zéro ou plusieurs actions associées à la plage de temps. Les actions traitées pour une même date sont supposées indépendantes, il est en effet difficile de déterminer la causalité des actions simultanées dans ce cas. Pour ces raisons le choix de la valeur du pas de temps est primordial dans ce type de simulation, un pas trop petit peut entraîner la simulation d'un certain nombre de dates associées à aucune action. Par opposition un pas trop grand peut poser des problèmes dans la représentation de phénomènes causaux car les actions d'une même plage sont toutes simulées en supposant leurs occurrences simultanées. Une action « cause » survenue pendant un pas donné doit donc, dans ce cas, attendre le prochain pas pour provoquer une action en « conséquence ».

3.3.4.6 Evolution dirigée par les événements

Une autre solution de simulation consiste à ne pas simuler les dates qui sont associées à aucun évènement, on ne simule que lorsque « quelque chose se produit » (l'occurrence d'un événement, Figure 3.4 b). Ces simulations sont appelées « dirigées par les événements » ou « asynchrones ». Dans ce cas il n'y a pas d'horloge globale, le temps de chaque simulateur avance par saut de durée variable, d'événement en événement à traiter. Ces événements sont stockés par ordre croissant de date d'occurrence dans une liste appelée **échéancier**. Il n'existe donc plus de problème de représentation des événements puisque chaque événement est simulé à sa date exacte et non regroupé sur une date (multiple du pas) comme dans l'évolution dirigée par le temps. L'échéancier

est géré par des algorithmes de synchronisation permettant sa mise à jour en cohérence avec l'environnement. Ces algorithmes et échéanciers peuvent être de type séquentiel ou repartis sur différentes machines qui posséderaient une partie de la simulation. Les limitations de ce type de simulation peuvent être liées à la performance dans le cas d'un nombre d'événements important car les événements ne sont pas regroupés et traités « en paquet ». L'autre difficulté est la communication avec un environnement temps réel, dans ce cas il est préférable d'utiliser une évolution dirigée par le temps ou de prévoir des barrières de synchronisation, ces barrières sont des dates qui permettent de se mettre en phase avec l'horloge absolue et les événements provenant du système.

3.4 Modélisation et simulation à événements discrets

A travers ces deux dernières décennies, plusieurs travaux ont été réalisés pour développer un cadre formel pour la simulation. La théorie des systèmes en est un exemple, elle permet de construire des modèles pour représenter la dynamique des systèmes complexes et ensuite les simuler. Les modèles construits sont décrits avec des formalismes à événements discrets comme DEVS(Discret EVent system Specification) [ZEIGLER, 1976].

Le formalisme DEVS est une approche de modélisation basée sur la théorie générale des systèmes. Plus précisément, c'est un formalisme modulaire et hiérarchique pour la modélisation, centrée sur la notion d'état. Un système est représenté, pour sa forme structurelle, par deux types de modèles (atomique et couplé).

La modélisation consiste à interconnecter ces différents types de modèles afin de former un nouveau modèle décrivant le comportement du système étudié, c'est l'aspect fonctionnel.

Les modèles atomiques sont les composants de base du formalisme, ils décrivent le comportement du système. Leur fonctionnement est proche de celui des "states machine" (machines d'états).

Pour décrire un système plus complexe nous interconnectons plusieurs modèles atomiques pour former un modèle couplé. Ce nouveau modèle peut être utilisé comme modèle de base dans une description de plus haut niveau, c'est l'aspect hiérarchique du formalisme.

Au niveau de la structure du système, cette approche peut sembler statique ; le formalisme DEVS dans sa forme basique ne tient pas compte de l'évolution potentielle de la structure du système, seul les états peuvent évoluer. Toutefois le formalisme a été étendu pour permettre ces changements de structure, et il en est, ou peut en être de même pour d'autres aspects.

Le formalisme DEVS peut être vu comme un environnement de multi-modélisation regroupant de manière cohérente d'autres formalismes de modélisation basés eux aussi sur la théorie générale des systèmes et centrés sur les états. Sa capacité d'ouverture, au sens informatique, en fait un

formalisme adapté à un grand nombre de domaines d'application [UHRMARCHER, 2001] [WAINER, 2003].

Au niveau de la simulation, il permet l'analyse de systèmes complexes à évènements discrets décrits par des fonctions de transitions d'états, et des systèmes continus décrits par des équations différentielles [ZEIGLER, 2000]. A ce niveau, le principal avantage de l'approche tient au fait que, pour un modèle décrit suivant les spécifications du formalisme, les algorithmes de simulations sont générés automatiquement. Cela permet de s'abstraire totalement de l'implémentation des simulateurs lors de la phase de modélisation, ce qui conduit à une séparation explicite entre la modélisation et la simulation.

3.5 La modélisation DEVS

Le formalisme DEVS définit deux catégories de modèles : les modèles atomiques et les modèles couplés. Les modèles atomiques sont chargés de représenter le(s) comportement(s) du système.

Les modèles couplés sont définis par un ensemble de sous-modèles (atomiques et/ou couplés) et traduisent la structure interne des sous-parties du système grâce à la définition de couplages entre les sous-modèles.

3.5.1 Le modèle atomique

Le formalisme de base DEVS dit "classique" considère un modèle atomique comme un concept mathématique basé sur le temps, un ensemble de valeur caractérisant tous les stimuli possibles en entrée et en sortie du système ainsi que deux fonctions permettant de déterminer la réponse comportementale à ces stimuli [CAPOCCHI, 2005]. Dans le formalisme DEVS dit "classique avec ports" la notion de couple (port, valeur) est introduite pour chaque port (entrée ou sortie) d'un modèle atomique. Les spécifications classiques d'un modèle atomique MA avec ports sont les suivantes :

$$MA = (X , Y , S , \delta_{int} , \delta_{ext} , \lambda , ta)$$

où,

- ✓ $X = \{(p,v) | p \in Ports\_entree, v \in X_p\}$ est la liste des ports et des valeurs d'entrées,
- ✓ $Y = \{(p,v) | p \in Ports\_entree, v \in X_p\}$ est la liste des ports et des valeurs de sorties,
- ✓ S est l'ensemble des variables d'états,
- ✓ $\delta_{int} : S \rightarrow S$ est la fonction de transition interne,
- ✓ $\delta_{ext}: Q \times X \rightarrow S$ est la fonction de transition externe où,
  - $Q = \{(s,e) | s \in S, 0 \leq e \leq ta(s)\}$ est l'ensemble des états,
  - e est le temps écoulé depuis la dernière transition.
- ✓ $\lambda : S \rightarrow Y$ est la fonction de sortie,
- ✓ $ta : S \rightarrow R+$ est le temps de vie de l'état S (réels positifs de 0 à ∞).

Les modèles réagissent à deux types d'événements : les événements externes et les événements internes. Les événements externes proviennent d'un autre modèle, déclenchent la fonction de transition externe et mettent à jour le temps de vie du composant. Les événements internes correspondent à des changements d'états du modèle, déclenchent les fonctions de transitions internes et de sorties, le modèle calcul ensuite grâce à la fonction ta la date du prochain événement interne.

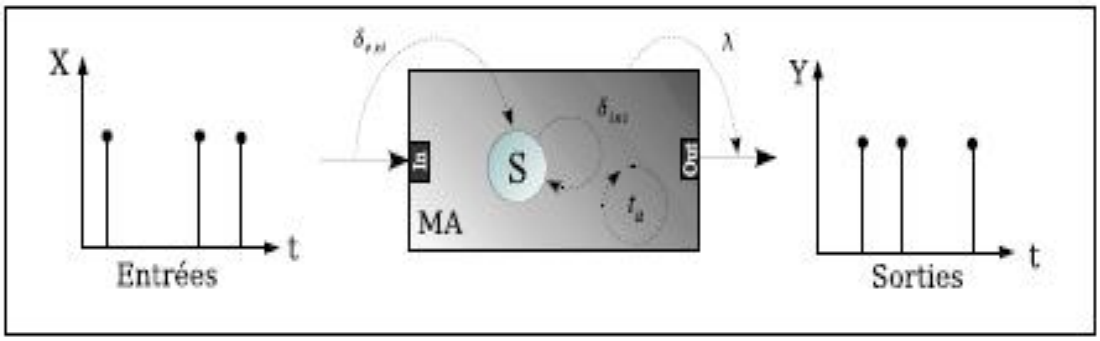


Figure 3.5 : Modèle atomique en action

L'interprétation des ces éléments est illustré sur la figure 3.5. A chaque instant le système (modèle atomique) est dans un état s. Si aucun événement externe n'intervient, le système restera dans cet état pendant un temps donné par la fonction ta(s). Lorsque le temps de vie du modèle est expiré, i.e. lorsque qu'il s'est écoulé e = ta(s) le système active sa fonction de sortie $\lambda(s)$ et change d'état grâce à l'exécution de la fonction de transition interne δ_{int} (s). Si un événement externe $x \in X$ intervient avant que le temps ne soit expiré, i.e. quand le système est dans l'état total (s,e) avec $e \leq ta(s)$, le système change d'état grâce à l'exécution de la fonction de transition externe δ_{ext} (s,e,x). Dans les deux cas, le système est alors dans un nouvel état s' avec un nouveau temps restant ta(s') et ainsi de suite.

Le temps de vie du composant peut être égal à zéro ou à l'infini. Dans le premier cas, la durée de l'état s est tellement courte qu'aucun événement externe ne peut intervenir avant l'arrivée du prochain changement d'état. Nous disons de s qu'il est dans un état transitoire. Dans le second cas, le système restera dans l'état s indéfiniment si aucun événement externe ne vient l'interrompre. Nous disons dans ce cas que s est un état passif.

L'explication précédente de l'activité d'un modèle atomique DEVS suggère, mais ne décrit pas totalement, le fonctionnement d'un simulateur qui exécuterait de tels modèles pour produire leurs comportements. Cependant, le comportement de DEVS est bien défini et peut être représenté sur la figure 3.6. Ici la trajectoire d'entrée X représente une série d'événements occurrents au temps t1 et t3. Entre ces événements temporels peuvent intervenir des temps comme t2 correspondant à des événements internes. La courbe de la trajectoire d'état s montre le changement d'état lors de la venue d'événements internes et externes. La trajectoire du temps écoulé e est en dent de scie et montre l'écoulement du temps par un compteur qui est remis à zéro à chaque changement d'événement. Finalement, la trajectoire de sortie Y montre les événements de sortie qui
sont produits par l'exécution de la fonction de sortie après avoir appliqué la fonction de transition interne.

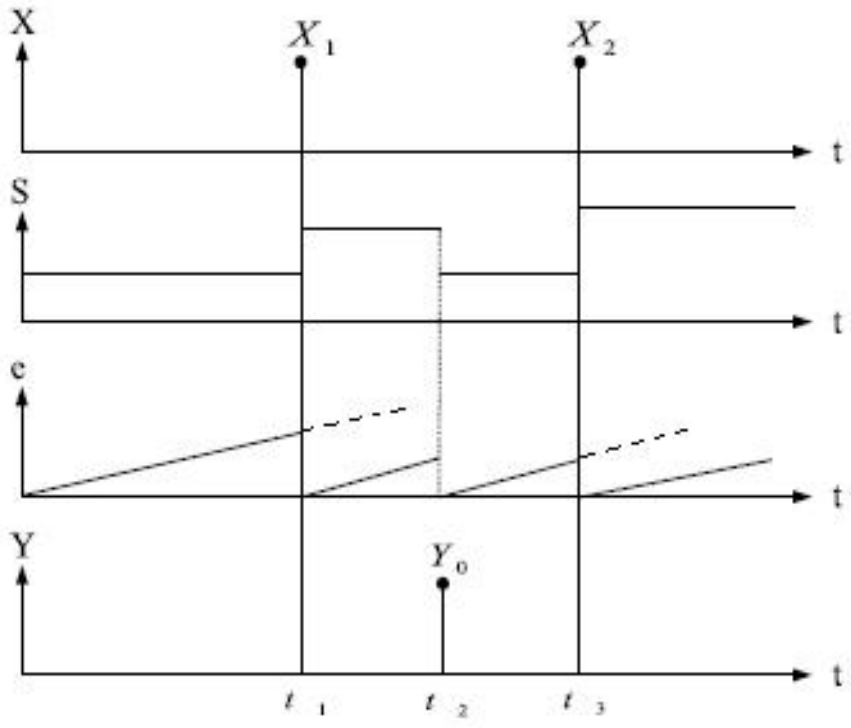


Figure 3.6 : Trajectoires d'un modèle atomique

Notons que dans DEVS classique avec ports, un port p reçoit une seule valeur v d'un événement externe. Avec le formalisme DEVS parallèle [CHOW et ZEIGLER, 2003], un modèle DEVS peut recevoir sur un port d'entrée donné plusieurs valeurs simultanées. La gestion des événements simultanés est prise en compte au sein d'une fonction de conflit δ_{con} dictant un ordre de priorité entre ces événements.

3.5.2 Le modèle couplé

Le formalisme DEVS utilise la notion de hiérarchie de description qui permet la construction de modèles dits "couplés" à partir d'un ensemble de sous-modèles et de trois relations de couplage avec ces sous-modèles [CAPOCCHI, 2005]. Un modèle couplé est constitué de sous-modèles qui peuvent être atomiques ou couplés et il possède les trois relations de couplage suivantes :

– IC (Internal Coupling): une relation de couplage interne pour le couplage entre les ports des sous-modèles qui composent le modèle couplé.

– EIC (External Input Coupling): une relation de couplage des entrées externes pour le couplage entre les ports d'entrées du modèle couplé et les ports d'entrées des sous-modèles.

– EOC (External Output Coupling): une relation de couplage des sorties externes pour le couplage entre les ports de sorties du modèle couplé et les ports de sorties des sous-modèles.

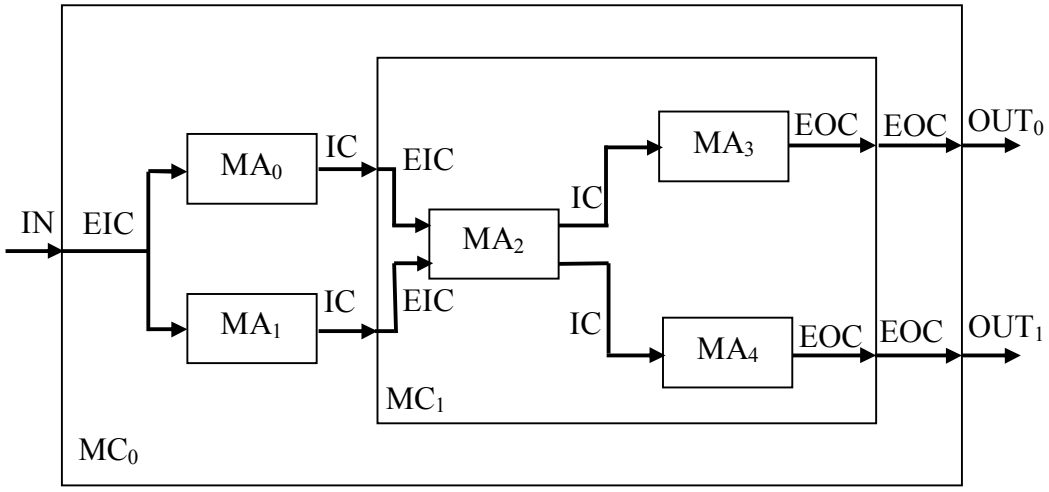


Figure 3.7 : Hiérarchie de modélisation DEVS

La figure 3.7 montre un exemple de hiérarchie entre les sous-modèles d'un système. Le modèle couplé MC_0 modélise au plus haut niveau le système étudié. Il possède deux ports de sortie OUT_0 et OUT_1, un port d'entrée IN et il contient les deux modèles atomiques MA_0 et MA_1 ainsi qu'un modèle couplé supplémentaire MC_1. Les modèles atomiques MA_0 et MA_1 sont reliés par une relation de couplage d'entrées externe au modèle couplé MC_0, et par une relation de couplage interne au modèle couplé MC_1.

Dans le cas du formalisme DEVS classique avec port les spécifications d'un modèle couplé sont les suivantes :

$$MC =< X,Y,D,\{M_i\},\{I_i\},\{Z_{i,j}\},Select >$$

où,

– $X = \{(p,v) \mid p \in ports\_entrées, v \in X_p\}$ est la liste des ports et des valeurs d'entrées.

– $Y = \{(p,v \mid p \in ports\_sorties, v \in Y_p)\}$ est la liste des ports et des valeurs de sorties.

– D est la liste des composants constituant le modèle couplé.

– $Mi =< X_i,Y_i,S_i, \delta_{ext,i}, \delta_{int,i}, \lambda_i, ta_{,i} >$ est un modèle atomique,

– Pour chaque modèle $i \in D \cup \{MC\}$, I_i est l'ensemble des modèles qui influencent i,

– $Z_{i,j}$ est la fonction de translation des sorties de i vers j telle que :

– $Z_{MC,j} : X_{MC} \rightarrow X_j$ est la fonction de couplage des entrées externes,

– $Z_{i,MC} : Y_i \rightarrow Y_{MC}$ est la fonction de couplage des sorties externes,

– $Z_{i,j} : Y_i \rightarrow X_j$ est la fonction de couplage interne.

– Select est la fonction de sélection qui donne la priorité entre les éléments deux à deux, afin d'éviter tout conflit.

La structure d'un modèle couplé doit répondre à des contraintes telles que, $\forall i \in D$:

1. $Mi =< X_i,Y_i,S_i, \delta_{ext,i}, \delta_{int,i}, \lambda_i, ta_{,i} >$ est un modèle atomique,

2. Une seule fonction $Z_{i,j}$ contient l'ensemble des informations sur les couplages du modèle couplé,

3. I_i est un sous-ensemble de $D \cup \{MC\}$ et $i \notin I_i$.

La cohérence et la conservation des comportements du système entre ces niveaux hiérarchiques est résumée par la propriété dite" fermeture sous couplage" [ZEIGLER, 1990]. Cette propriété assure qu'un modèle couplé, représenté par le couplage d'un ensemble de sous-modèles plus détaillés, est équivalent à un modèle atomique.

3.6 La simulation DEVS

L'une des propriétés importantes du formalisme DEVS est qu'il fournit automatiquement un simulateur pour chacun des modèles. DEVS établit une distinction entre la modélisation et la simulation d'un système tel que n'importe quel modèle DEVS puisse être simulé sans qu'il ne soit nécessaire d'implémenter un simulateur spécifique [CAPOCCHI, 2005]. Chaque modèle atomique est associé à un simulateur chargé de gérer le comportement du composant et chaque modèle couplé est associé à un coordinateur chargé de la synchronisation temporelle des composants sous-jacents. L'ensemble de ces modèles est géré par un coordinateur spécifique appelé "Root".

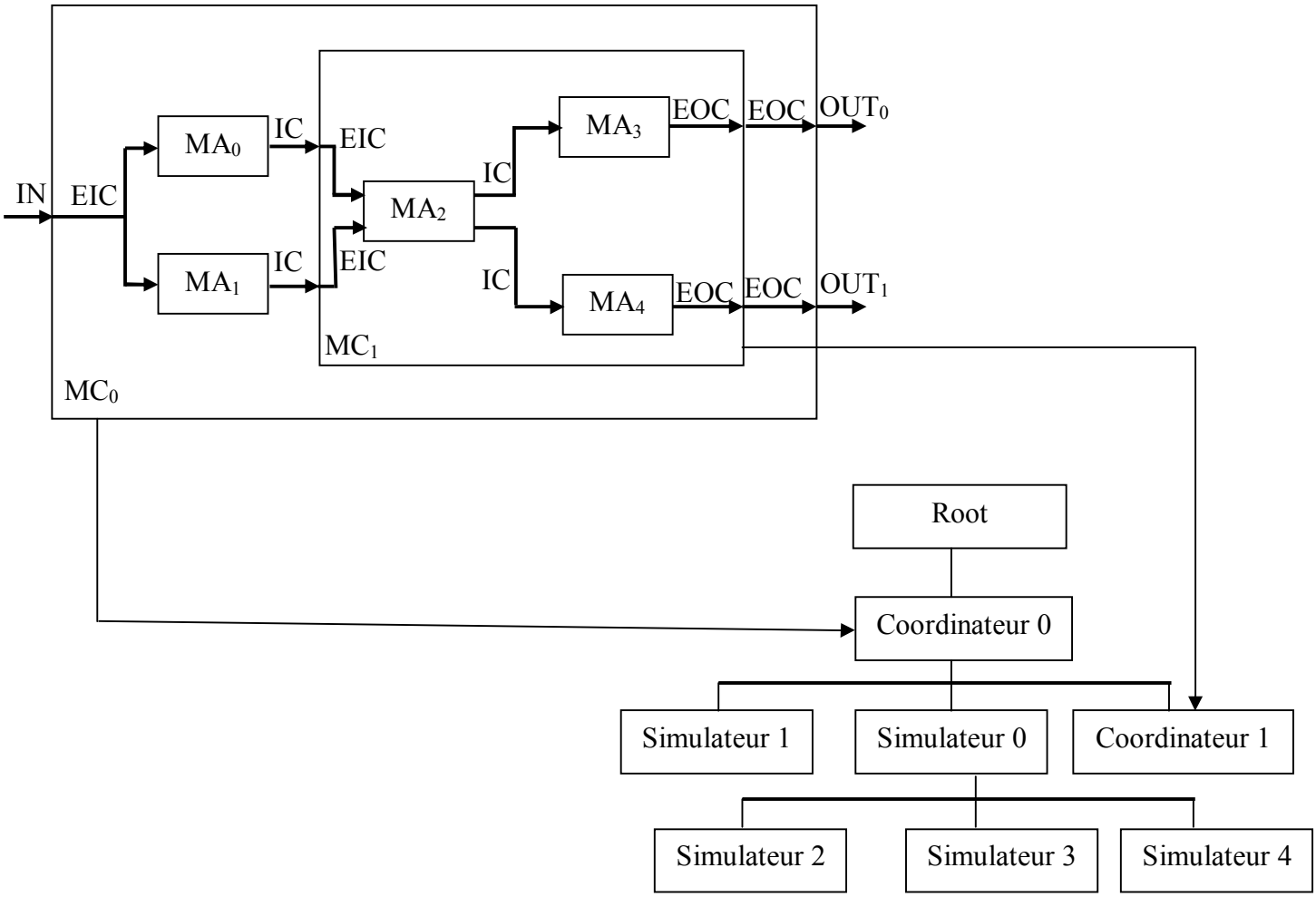


Figure 3.8 : Arbre hiérarchique de simulation DEVS

La figure 3.8 montre la structure d'un simulateur associé à un modèle DEVS quelconque. La hiérarchie du simulateur DEVS est construite sur une arborescence constituée de simulateurs et de coordinateurs. Les deux coordinateurs "Coordinateur 0" et "Coordinateur 1" sont associés aux modèles couplés MC_0 et MC_1. Le "Coordinateur 0" est chargé de gérer le "Coordinateur 1" ainsi que les simulateurs "simulateur 0" et "Simulateur 1" associés aux deux modèles atomiques MA_0 et MA_1. De la même manière, comme le modèle couplé MC_1 encapsule les trois modèles atomiques MA_2, MA_3 et MA_4, le coordinateur "Coordinateur 1" associé gère les trois simulateurs "Simulateur 2", "Simulateur 3" et "Simulateur 4". Le coordinateur "Root" est chargé de coordonner la totalité des composants de l'arbre de simulation.

L'ordre d'exécution des fonctions comportementales (δ_{int} ,δ_{ext} ,λ ,ta) d'un modèle atomique DEVS de la figure 3.5 sous-entend la présence d'un mécanisme de simulation. Ce mécanisme est géré par le composant simulateur qui, comme le coordinateur, peut recevoir ou émettre 4 types de messages :

– Le message d'initialisation (i,t) permet à tous les acteurs d'effectuer une synchronisation temporelle initiale.

– Le message de transition interne (*,t) permet la gestion d'un événement interne avec l'exécution de la fonction δ_{int} .

– Le message de sortie (y,t) permet le transport des sorties (données par λ) aux éléments parents et résulte de la réception d'un message (*,t).

– Le message de transition externe (x,t) permet la gestion d'un événement externe avec l'exécution de la fonction δ_{ext} .

3.7 Exemple d'un modèle DEVS atomique et d'un modèle DEVS couplé comprenant deux DEVS atomique

Sur la Figure 3.9, M1 est un modèle DEVS atomique. Sur ce modèle, les deux types de transitions existantes entre états sont présentés [ANDRIEN, 2007].

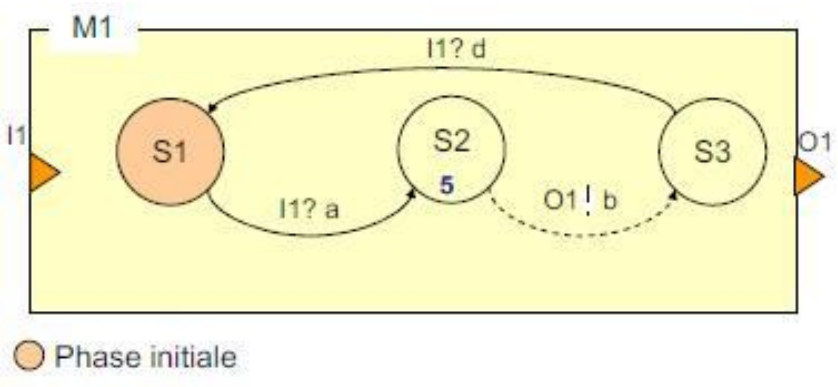


Figure 3.9 : Exemple d'un DEVS atomique

M1= <X, S, Y, δ_{int}, δ_{ext}, λ, ta>

Où :

- X = {a,d},
- S = {S1, S2, S3},
- Y = {b},
- δ_{int} : S2 -> S3
- δ_{ext} : (S1, e, a) -> S2
  (S3, e, d) -> S1
- λ : S2 -> S3,
- ta : S2 -> 5.

La dynamique du modèle peut être décrite de la manière suivante. Le modèle est dans sa phase initiale S1. Lorsque le modèle reçoit l'événement "a" sur son port "I1", il passe dans la phase "S2". Au bout de 5 unités de temps, le modèle évolue dans la phase "S3", en émettant l'événement "b". Le modèle de la Figure 3.10 est un modèle DEVS couplé.

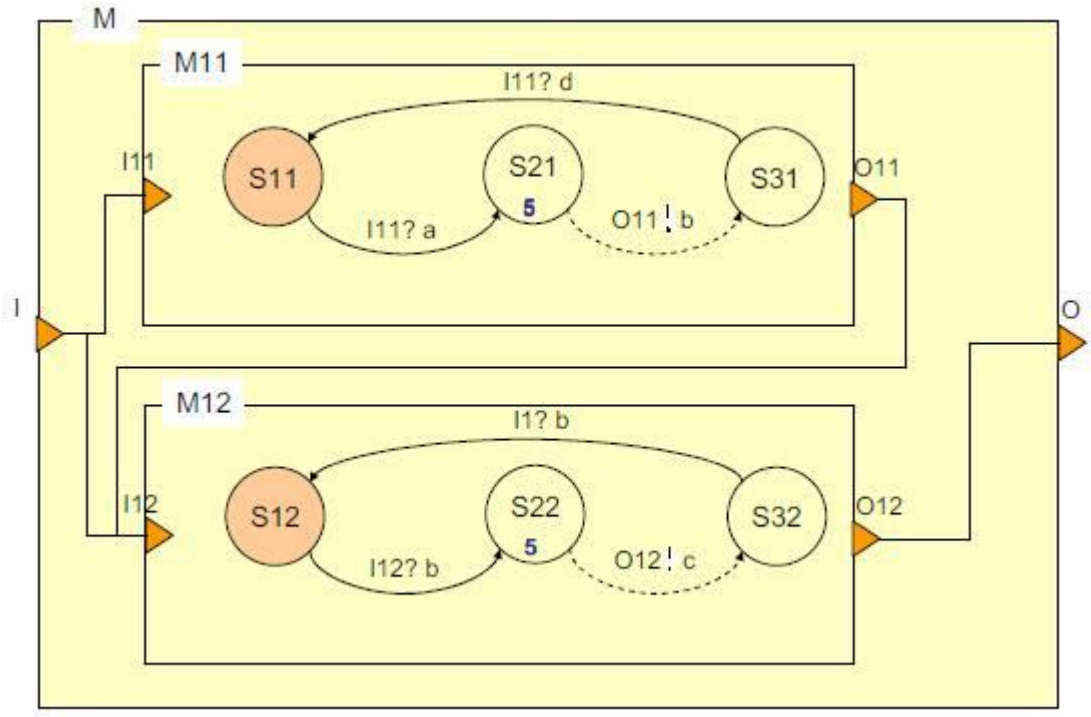


Figure 3.10 : Exemple d'un DEVS couplé

M= <X, Y, D, {M_d|d∈D}, EIC, EOC, IC, Select>

Où :

- X= {(I, a), (I, b), (I, d)},
- Y= {(O, c)},
- D = {M11, M12}, où M11 et M12 sont deux modèles DEVS atomiques,
- EIC= {((M, I), (M11, I11)), ((M, I), (M12, I12))},
- EOC= {((M12, O12), (M, O))},
- IC= {((M11, O11), (M12, O12))},
- Select : (M11, M12) -> M11.

Les deux modèles M11 et M12 sont semblables au modèle atomique décrit ci-dessus.

M11 et M12 sont initialement dans leur phase respective S11 et S12. Lorsque le modèle couplé M reçoit un événement sur son port "I", il l'envoie à M11 et à M12, sur les ports respectifs I11 et I12.

Si l'événement reçu par "I" est "a", M11 évolue dans la phase S21, et M12 ne change pas de phase. Au bout de 5 unités de temps, M11 émet l'événement "b" par son port de sortie O11, qui le transmet

à M12 sur le port I12. Si M12 est encore dans sa phase initiale S12, il évolue dans S22, et émet l'événement "c" par son port de sortie "O12", sur le port de sortie "O" du modèle M.

Si l'événement reçu par "I" est "b", M12 évolue dans la phase S22, et M11 ne change pas de phase. Au bout de 5 unités de temps, M12 émet l'événement "c" par son port de sortie O12, qui le transmet à M sur le port O.

3.8 Avantages de l'utilisation de DEVS

DEVS permet la spécification de modèles à événements discrets. Ses principales caractéristiques sont les suivantes [DUBOZ, 2004] :

- ✓ DEVS est un formalisme abstrait indépendant de l'implémentation et par conséquent des environnements de simulation ;
- ✓ Il offre une vision modulaire et hiérarchique des systèmes dynamiques ;
- ✓ La fonction de transition interne, formalise et permet le comportement autonome du modèle. Les modèles peuvent ainsi évoluer et émettre des événements lorsque aucun événement externe n'est programmé pendant une certaine durée (qui correspond à la durée de vie de l'état dans lequel se trouve le modèle);
- ✓ DEVS est fermé sous composition. Cela signifie que toute composition obtenue par couplage de composants spécifiés par le formalisme est elle-même spécifiée par le formalisme. La construction hiérarchique des modèles par l'application récursive des procédures de couplage est alors facilitée [ZEIGLER, 1984] ;
- ✓ L'interprétation, dans le monde réel, des concepts d'abstraction et de modélisation est explicite ;
- ✓ Il garantit la cohérence de la représentation construite, et par conséquent il offre un modèle implémentable pour simuler des modèles élaborés.

Ces avantages nous ont fortement influencés dans le choix du formalisme. En effet, les modèles doivent être facilement interprétables, guidés par les événements et simples. La simplicité est permise grâce à la construction hiérarchique et modulaire possible.

3.9 Conclusion

Nous avons présenté dans ce chapitre les notions théoriques de la modélisation et de la simulation de système, et en particulier nous avons détaillé le formalisme développé par B.P Zeigler. Ce formalisme présente plusieurs avantages en ce qui concerne les objectifs que nous nous sommes fixés, à savoir la modélisation de systèmes naturels complexes à paramètres imparfaits.

En effet, le formalisme DEVS peut être défini comme une méthodologie universelle et générale qui fournit des outils pour modéliser et simuler des systèmes dont le comportement repose sur la notion d'évènements. Ce formalisme est basé sur la théorie générale des systèmes, la notion de modèle, et

permet la spécification de systèmes complexes à évènements discrets sous forme modulaire et hiérarchique. Nous pouvons aussi définir le formalisme DEVS comme un outil de multi-modélisation regroupant plusieurs formalismes de modélisation de manière cohérente, de fait, lorsqu'il est replacé dans le contexte spécifique d'un domaine d'application, si aucune approche de modélisation ne permet de traiter le domaine, il doit être adapté et étendu.

Dans le chapitre suivant nous allons présenter la modélisation et la simulation du système de la navigation réactive d'un robot mobile.

Chapitre 4

Modélisation et Simulation

4.1 Introduction

La modélisation du système de la navigation réactive d'un robot mobile, basée sur l'utilisation du formalisme DEVS combiné avec des techniques floues, est une approche pour la prise en compte de l'incertitude, l'imprécision et l'incomplétude d'informations dans le formalisme DEVS. Concevoir un système flou signifie, en particulier, que le système ayant la capacité : de traiter des problèmes avec des connaissances imparfaites par inférence floue, de prendre des décisions sous incertitude.

Nous présentons dans ce chapitre une nouvelle approche pour la prise en compte de l'aspect flou dans le formalisme DEVS. Notre objectif est de proposer un modèle pour le système de la navigation réactive d'un robot mobile dans un environnement dynamique et incertain.

Le cadre de modélisation et simulation DEVS est devenu une référence pour la question du couplage de modèles de simulation hétérogènes, la modélisation de processus décisionnels en DEVS est crucial aujourd'hui dans l'étude des systèmes complexes.

Les systèmes robotiques autonomes sont profondément aléatoires du fait de leur interaction avec un environnement imprévisible. La modélisation de ces systèmes sous DEVS devient difficile d'où de leur nature aléatoire. Le formalisme DEVS classique permet de la modélisation et l'analyse de systèmes complexes à évènement discret avec une structure et un comportement connus. Notre contribution s'inscrit dans le formalisme DEVS combiné avec un système d'inférence floue.

DEVS étant lui-même une formalisation des systèmes dynamiques, il est possible d'utiliser ce formalisme pour la spécification d'un système de la navigation d'un robot mobile.

Le robot mobile peut être vu comme une entité autonome qui perçoit son environnement et agit sur lui. Le comportement et les interactions d'un robot mobile peuvent être très complexes. Les principes de modularité et de décomposition hiérarchique énoncés dans notre présentation de DEVS nous permettent d'adopter un niveau de détail suffisant pour exprimer cette complexité.

Le comportement du robot mobile peut se résumer à une boucle Perception → Choix_Action → Execution(action). Le choix de l'action est le fruit d'un raisonnement.

4.2 Architecture du robot

Le robot simulé dans notre étude est un robot qui a une plate forme circulaire, par ce qu'elle est la plus standard et la plus utilisée dans la robotique mobile. De plus elle est facile à commander, il suffit de spécifier les vitesses des roues motrices pour faire tourner le robot et au contraire aux plates formes non holonomes, la plate forme circulaire permet au robot de tourner sur place [ZAGANE, 2010]. La figure 4.1 présente la plate forme du robot mobile à simuler.

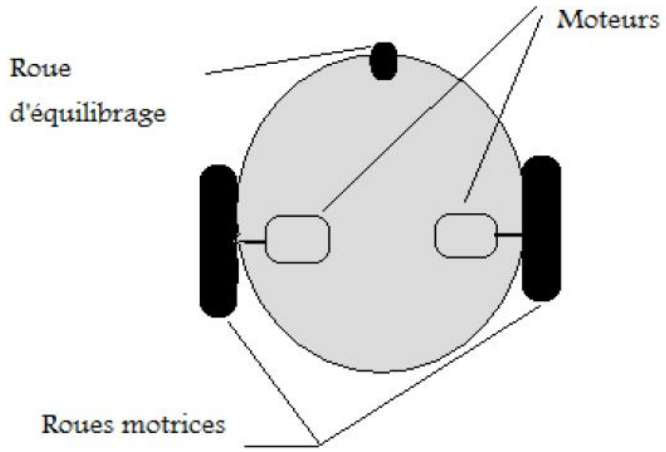


Figure 4.1 : Plate forme mobile utilisée

Les capteurs jouent un rôle vital dans le fonctionnement du robot, ils fournissent une quantité d'information très importante. Les informations perçues sont traitées et utilisées ensuite par les différents modules et comportements de l'architecture.

Nous avons limité le champ global de perception du robot par 45°. Le champ global de perception est décomposé en trois secteurs de détection. Chaque secteur comporte un capteur pour obtenir une plus grande précision relative à la position des obstacles par rapport à notre robot. La figure 4.2 présente la disposition des capteurs du robot à simuler.

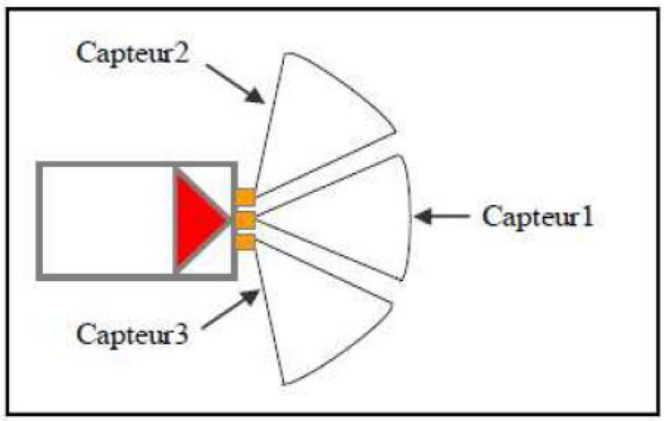


Figure 4.2 : Secteurs de détection du robot

Nous présentons dans la figure 4.3 le modèle cinématique du robot. Le robot doit se déplacer le long d'une trajectoire, d'un point de départ D à un point objectif C quelconque. Par ses trois capteurs, le robot calcule la distance par rapport à un éventuel obstacle, son orientation et l'angle par rapport à l'objectif.

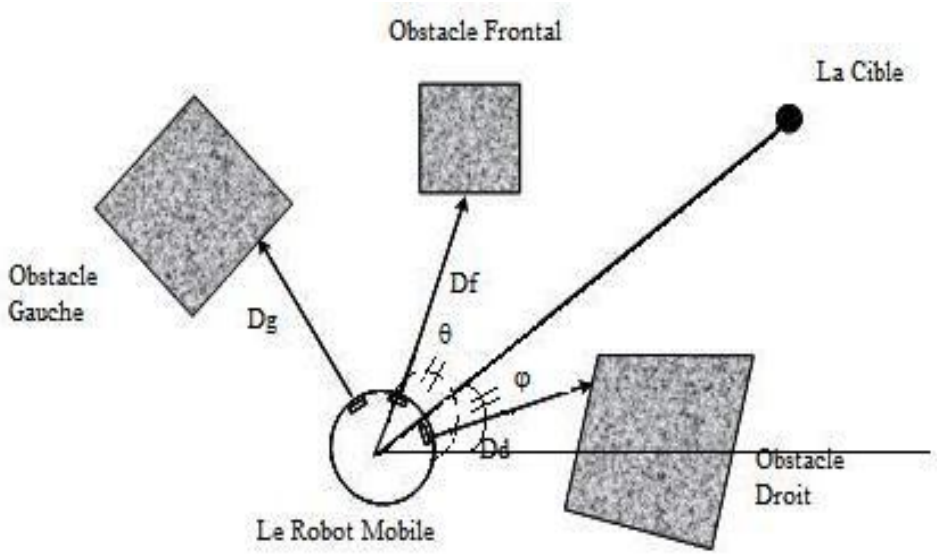


Figure 4.3 : Modèle cinématique du robot

4.2.1 Architecture structurelle du système robot mobile

Le modèle que nous proposons divise le système de la navigation d'un robot mobile en quatre sous systèmes: Localisation, Perception, Contrôleur, Actuateur.

Le sous-système localisation : permet d'estimer la position du robot et sa situation relativement à l'environnement ou à des objectifs géographiques par les capteurs extéroceptifs.

Le sous-système perception : consiste globalement à saisir un certain nombre d'informations sensorielles dans le but d'acquérir une connaissance et une compréhension du milieu d'évolution grâce à des capteurs proprioceptifs.

Le sous-système contrôleur : permet la prise de décision conditionnement le plus robuste possible des données sensorielles issues du sous-système de perception.

Le sous-système actuateur : permet d'exécuter l'action choisie par le sous-système contrôleur.

La figure 4.4 présente le diagramme de la structure du système robot mobile en DEVS.

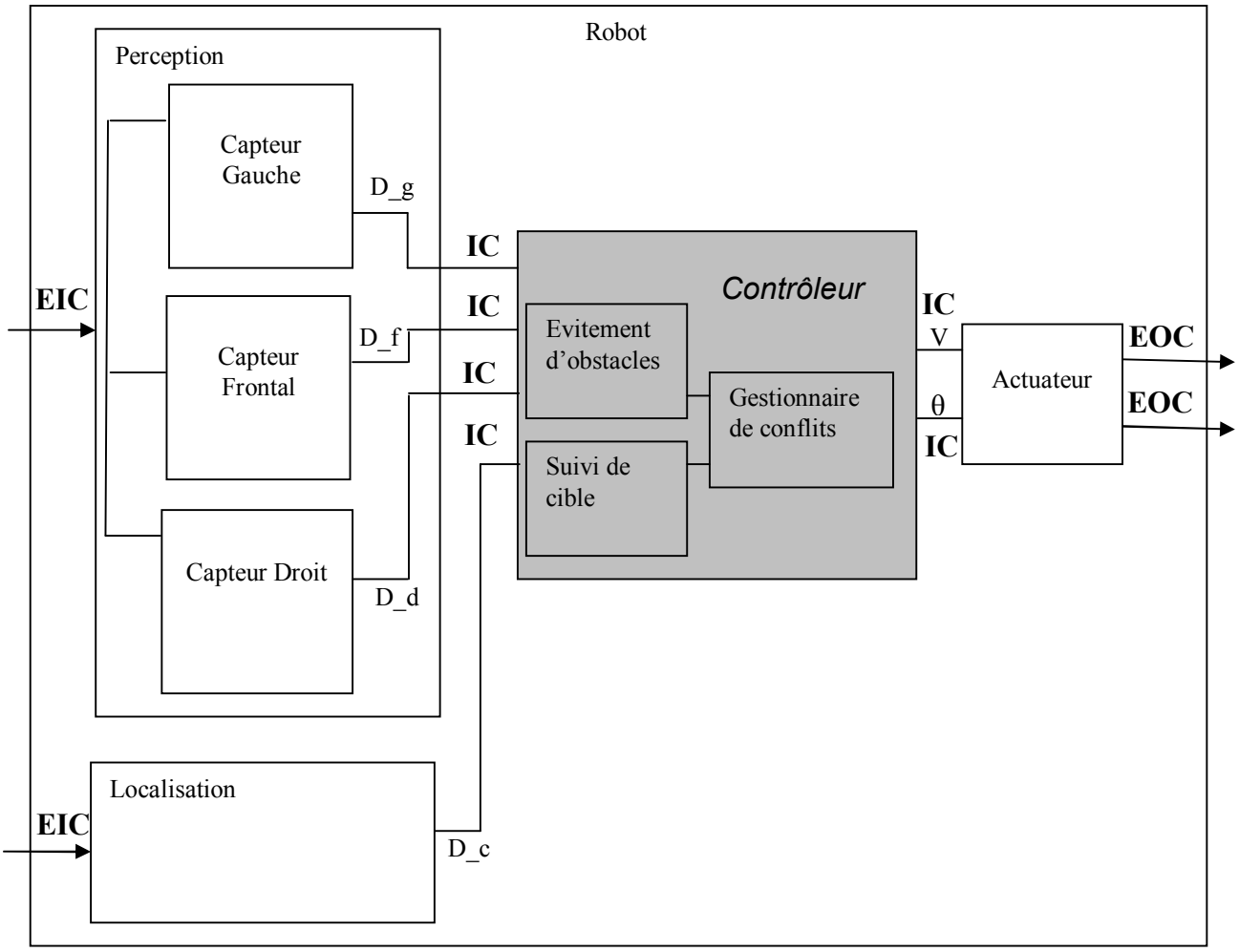


Figure 4.4 : Structure du système robot mobile en modèle DEVS couplé

4.3 Formalisation du système robot mobile en DEVS

Le robot mobile peut être vu comme une entité autonome qui perçoit son environnement et agit sur lui.

Nous pouvons considérer le robot mobile comme un modèle DEVS. En effet, un modèle DEVS a un nom et une structure particulière qui définissent son identité et ses frontières avec l'extérieur.

Les ports d'entrée et de sortie représentent respectivement les récepteurs et actuateurs du robot. Les fonctions de transitions externes gouvernent les changements d'états du robot liés à l'arrivée de stimuli, les fonctions de sorties définissent les actions du robot, et les fonctions de transitions internes correspondent aux comportements autonomes.

Néanmoins, un système de la navigation d'un robot mobile n'est pas simple. Aussi, un modèle atomique ne peut pas suffir à sa formalisation. Ainsi nous postulons qu'un robot mobile est un modèle DEVS couplé.

Le comportement et les interactions d'un robot mobile peuvent être très complexes. Les principes de modularité et de décomposition hiérarchique énoncés dans notre présentation de DEVS nous permettent d'adopter un niveau de détail suffisant pour exprimer cette complexité. Ainsi, il peut y avoir plusieurs modèles DEVS qui formalisent le système du robot mobile. Cela semble de décomposer le système en sous-système perception, localisation, contrôleur et actuateur. Chaque sous-système est formalisé par un modèle DEVS atomique.

Nous donnons à un robot mobile la structure d'un modèle DEVS couplé suivante :

$$\text{Robot} = \langle X, Y, D, \{M_d | d \in D\}, EIC, EOC, IC, SELECT \rangle$$

Où, comme pour un modèle DEVS classique :

Le modèle couplé Robot désigne le modèle DEVS couplé d'un système de la navigation réactive d'un robot mobile robot.

$X = \{(p,v) / p \in IPorts, v \in X_p\}$ est l'ensemble des récepteurs et des valeurs qu'ils peuvent prendre (les distances gauche Dg, frontale Df, droite Dd).

$Y = \{(p,v) / p \in OPorts, v \in Y_p\}$ est l'ensemble des actuateurs et des valeurs qu'ils peuvent prendre (la vitesse du déplacement V, l'angle de déviation du robot θ).

D est l'ensemble des noms des modèles atomiques ou couplés composants le robot.

D={ perception, localisation, contrôleur , actuateur}

M_D est l'ensemble des modèles atomiques ou couplés composants le robot.

$M_{perception}$ est le modèle du sous-système de perception.

$M_{localisation}$ est le modèle du sous-système localisation.

$M_{contrôleur}$ est le modèle du sous-système contrôleur.

$M_{actuateur}$ est le modèle du sous-système actuateur.

Select est la fonction qui donne la priorité entre les éléments deux à deux, afin d'éviter tout conflit.

Avec en particulier :

EIC={((Robot,$X_{g,f,d}$),(Perception,$X_{g,f,d}$)), ((Robot,$Y_{g,f,d}$),(Perception,$Y_{g,f,d}$)),

((Robot,X_c),(Localisation,X_c)) ((Robot,Y_c),(Localisation,Y_c))}

EOC={((Actuateur,X_r),((Robot, X_r)), ((Actuateur,Y_r),((Robot, Y_r))}

IC={((Perception,$D_{g,f,d}$),(Contrôleur,$D_{g,f,d}$)), ((Localisation,D_c),(Contrôleur,D_c)),

((Contrôleur,V),(Actuateur,V)), ((Contrôleur, θ),(Actuateur, θ))}

L'ensemble EIC définit les ports du modèle du robot qui sont connectés aux ports des modèles composants qui reçoivent des évènements externes.

L'ensemble EOC définit les ports du modèle du robot connectés aux ports des modèles composants qui émettront des évènements.

4.3.1 Formalisation du sous-système perception en DEVS

La perception est le processus par lequel le robot acquiert des informations sur le monde qui l'entoure. La perception est représentée en DEVS par l'arrivée d'un évènement externe dans un modèle robot, entrainant un changement d'état dans un ou plusieurs modèles composants. Nous considérons donc la perception comme un changement de l'état interne du robot dû à un stimulus externe. Dans la figure 4.5 nous présentons le modèle couplé du sous-système perception.

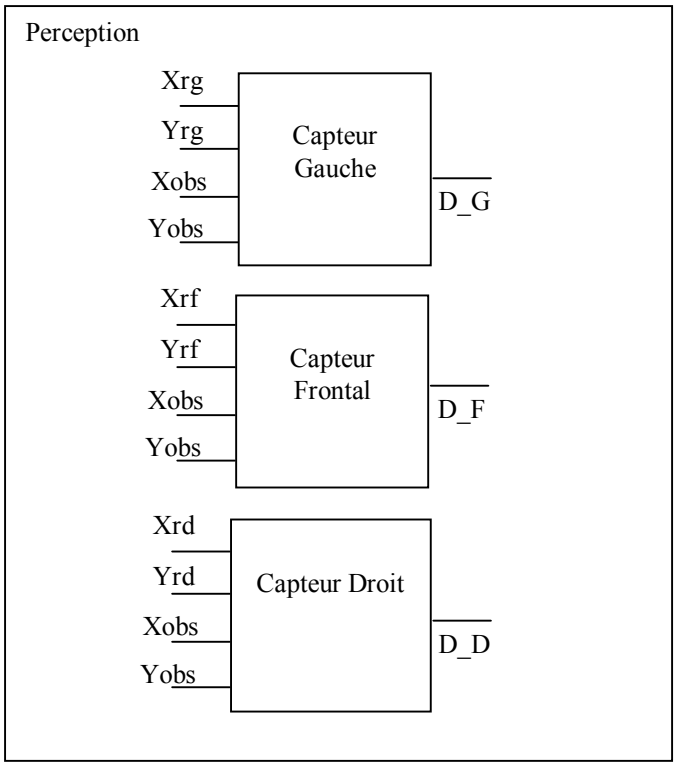


Figure 4.5 : Sous-système perception en DEVS

Nous formalisons le système capteur par la structure d'un modèle DEVS atomique suivante:

$$Capteur = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$$

$X=\{ (X_{rob},Y_{rob}),(X_{obs},Y_{obs})\}$: les coordonnées cartésiennes du robot et l'obstacle.

$S=\{Distance\}$: Distance $\in \Re$.

$Y=\{ D\}$: qui correspond à la distance entre le robot et l'obstacle.

$\delta_{ext}(S,X,e)=\sqrt{(X_{obstacle} - X_{robot})^2 + (Y_{obstacle} - Y_{robot})^2}$: Permet de calculer la distance entre le robot et l'obstacle.

$\delta_{int}=\varnothing$ .la fonction de transition interne.

$\lambda(S) =\varnothing$, la fonction de sortie.

ta = ∞, fonction d'avancement du temps.

4.3.2 Formalisation du sous-système localisation en DEVS

A travers ce sous-système, le robot peut se localiser dans l'environnement en calculant son orientation et ses coordonnées cartésiennes à chaque instant. Il peut aussi localiser la cible à atteindre. La localisation de la cible consiste à calculer la distance qui la sépare du robot et l'orientation que fait ce dernier avec elle. La figure 4.6 présente la structure de sous-système localisation en modèle DEVS atomique.

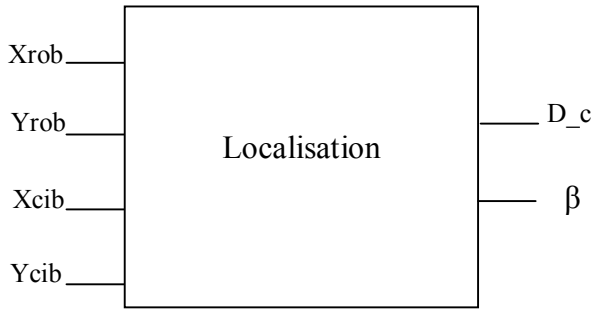


Figure 4. 6 : Sous-Système localisation en DEVS

Nous formalisons le sous-système localisation par la structure d'un modèle DEVS atomique suivante:

$$\text{Localisation} = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$$

$X=\{ (X_{rob},Y_{rob}),(X_{cib},Y_{cib})\}$: les coordonnées cartésiennes du robot et la cible.

$S=\{\text{Distance}\}$: Distance $\in \Re$.

$Y=\{ D, \beta\}$ qui correspondent respectivement à la distance entre le robot et la cible et L'angle que fait le robot avec la cible.

$\delta_{ext}(S,X,e)= \sqrt{(X_{robot} - X_{cible})^2 + (Y_{robot} - Y_{cible})^2}$: permet de calculer la distance entre le robot et la cible.

$\delta_{ext}(S,X,e)$= ArcTang((Ycible - Yrobot) / (Xcible - Xrobot)) : permet de calculer l'angle de déviation du robot.

$\delta_{int}=\varnothing$ .la fonction de transition interne.

$\lambda(S) =\varnothing$, la fonction de sortie.

ta = ∞, fonction d'avancement du temps.

4.3.3 Formalisation du sous-système actuateur en DEVS

A travers ce sous-système, le robot répond aux perceptions, par les actions suivantes:

- Evitement des objets obstacles.
- Rapprochement de l'objet cible.

Le robot se déplace par l'utilisation la règle générale de la cinématique donnée par la formule mathématique suivante :

- **X (n) = X (n-1) + V. Δ T. Cos (θ)** ;
- **Y (n) = Y (n-1) + V. Δ T .Sin (θ)** ;

Où V est la vitesse de déplacement , T est une période de temps et θ est l'angle de déplacement. Dans la figure 4.7 nous présentons le modèle couplé du sous-système actuateur.

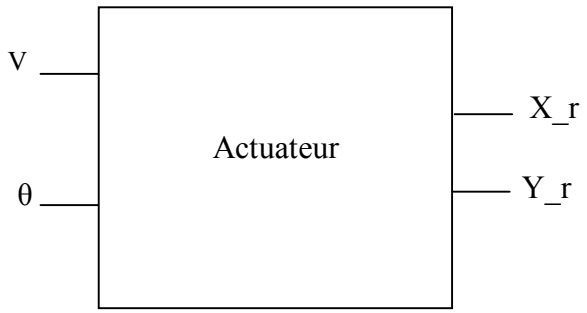


Figure 4.7 : Sous-Système actuateur en DEVS

Nous formalisons le sous-système actuateur par la structure d'un modèle DEVS atomique suivante:

$$\text{Actuateur} = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$$

X={ V, θ} : qui correspondent respectivement à la vitesse du déplacement et l'angle de déviation du robot.

S= {(Xr,Yr)} : (les coordonnées cartésiennes du robot).

Y={ X_r,Y_r} qui correspondent aux nouvelles coordonnées cartésiennes du robot.

$\delta_{ext}(S,X,e)= X(n-1) + V. \Delta T. Cos(\theta)$;

$\delta_{ext}(S,X,e)= Y(n-1) + V. \Delta T .Sin(\theta)$; Calcule les nouvelles coordonnées cartésiennes du robot .

$\delta_{int}=\varnothing$ .la fonction de transition interne.

$\lambda(S) =\varnothing$, la fonction de sortie.

ta = ∞, fonction d'avancement du temps.

4.3.4 Formalisation du sous-système contrôleur en DEVS

La navigation autonome en environnement dynamique représente un défi important pour la recherche en robotique. La détection des obstacles mobiles, la prédiction de l'état futur du robot présentent des difficultés majeures dans la navigation réactive d'un robot mobile dans un environnement dynamique inconnu. Le formalisme DEVS classique ne permet pas de prendre en compte imprécision et incertitude sur les événements et sur les états. Le multi-formalisme DEVS permet l'intégration de nombreux autres formalismes ou méthodes de modélisation.

Dans ce travail nous proposons un contrôleur flou classique ou traditionnel combiné avec le formalisme DEVS, Une fois que les entrées sont connues (entrées capteurs), les sorties sont déterminées de manière unique. Les contrôleurs de ce type utilisant l'approche classique dite Mamdani dans laquelle chaque règle est représentée par une conjonction (minimum) et l'agrégation des règles par une disjonction(maximum). La figure 4.8 donne le principe de conception de notre contrôleur en DEVS.

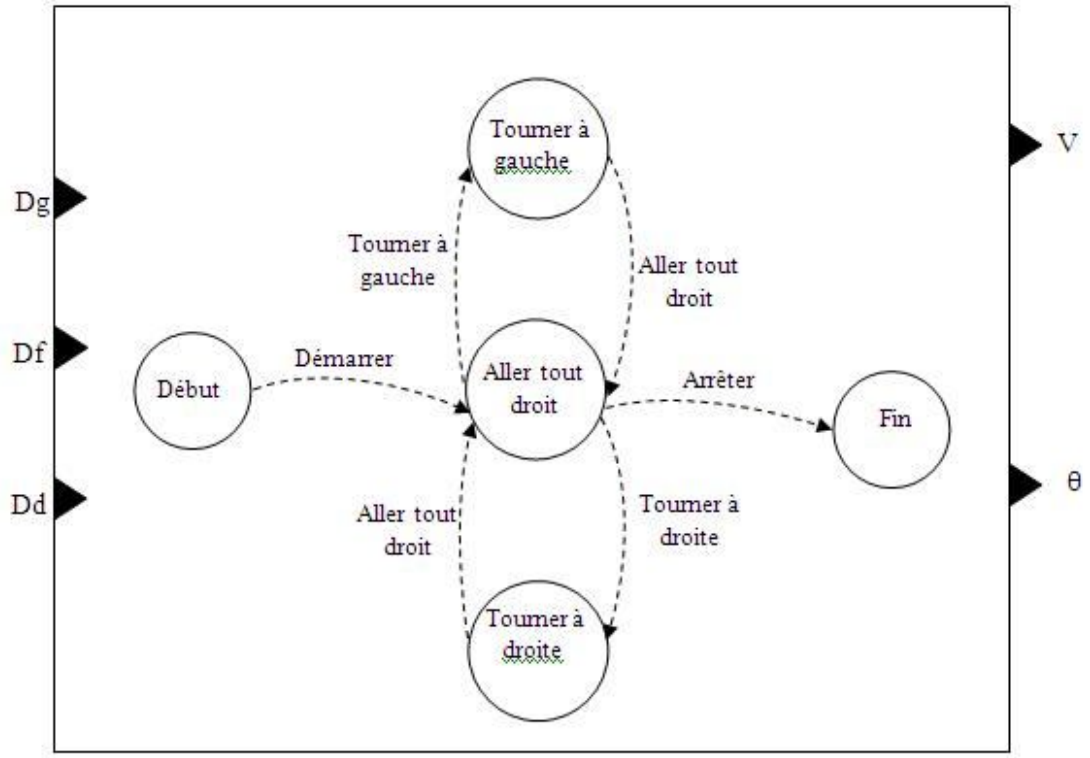


Figure 4.8 : Un contrôleur flou en DEVS

La spécification formelle du contrôleur flou sous DEVS est comme suit :

$$\text{Contrôleur} = (X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta)$$

X= {Dg, Df, Dd} : distances données par le sous-système perception.

S={Début, Tourner à gauche , Tourner à droite, Aller tout droit, Fin}.

Y={V, θ} : qui correspondent respectivement à la vitesse du déplacement et l'angle de déviation du robot.

$\delta_{ext}(S,X,e)$= SIF(Dg, Df, Dd) : est système d'inférence floue, ce dernier reçoit les entrées(D) à partir du module de perception.

$\delta_{int}=\varnothing$,la fonction de transition interne.

$\lambda(S) =\varnothing$, permet d'envoyer les résultats du SIF vers les ports de sortie.

ta = ∞ : fonction d'avancement du temps.

4.3.4.1 La logique floue en robotique

La logique floue est le meilleur moyen de convertir les stratégies de contrôle du langage naturel qui sont utilisées par l'homme à une forme utilisable par les machines. Des expériences ont montré qu'un contrôleur classique donne parfois même de meilleurs résultats que l'operateur humain [FATMI, 2005]. Le principal avantage de la logique floue est qu'elle permet aux experts humains d'exprimer des concepts. Ce qui entraine un gain de temps et d'espace pour une recherche dans les règles pour l'exécution d'une situation donnée.

Un autre avantage de la logique floue est le fait que le calcul de la charge des systèmes d'inférence floue est considérablement léger. Le succès de la gestion de l'incertitude et/ou l'incomplétude des données est un attribut de la logique floue fondée sur des moteurs d'inférence.

La commande floue a montré depuis plusieurs années sa fiabilité et sa capacité d'adaptation aux problèmes industriels concrets. Les exemples d'utilisation concluant de la logique floue sont très nombreux.

4.3.4.2 Le Contrôleur flou classique

La stratégie de navigation de base est une approche réactive, combinant deux comportements élémentaires :

L'évitement d'obstacles et le rapprochement du but. Le contrôleur flou reçoit comme entrées les distances mesurées par le capteur gauche " D_g", le capteur frontal " D_f" et droit " D_d " ainsi que l'angle r-c correspondant à l'angle existant entre l'orientation de la cible et l'orientation θ du robot et renvoie en sortie les deux commandes θ et V, qui correspondent respectivement au changement de direction du robot et à la variation de sa vitesse.

Les trois distances sont évaluées par rapport aux trois sous-ensembles flous « Petite » (P) et « Moyenne » (M), et « Grande » (G).

4.3.4.3 La fuzzification

La fuzzification est un processus qui permet d'associer les éléments du vecteur d'entrée aux ensembles flous selon un degré d'appartenance.

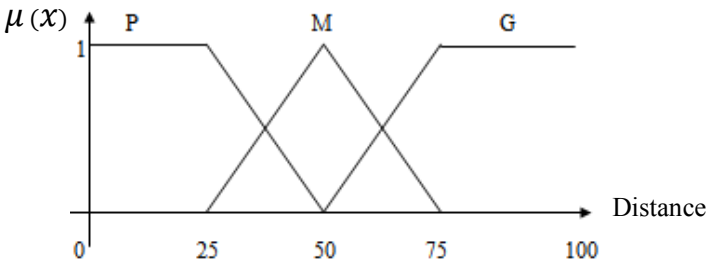


Figure 4.9 : Fonction d'appartenance de la distance(Dg, Df, Dd)

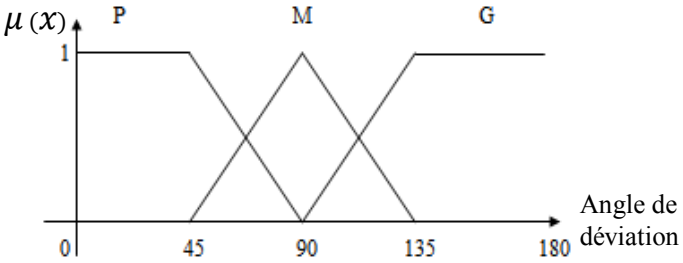


Figure 4.10 : Fonction d'appartenance de la sortie θ

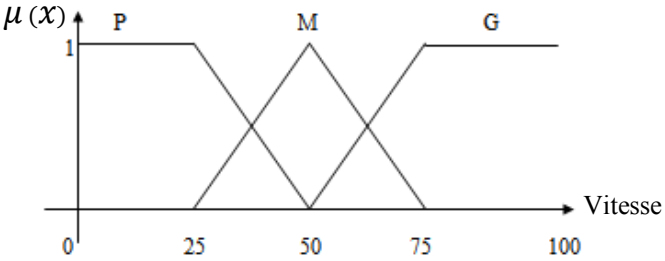


Figure 4.11 : Fonction d'appartenance de la vitesse V

4.3.4.4 L'inférence

Un contrôleur flou prend généralement la forme d'une série de règles «Si-alors». L'inférence est basée sur des opérations min et max afin d'effectuer l'inférence des règles et l'opérateur max pour l'agrégation des règles.

Les règles utilisées dans notre système sont des règles universelles dans la navigation des robots mobiles. Le tableau suivant montre les règles utilisées :

Règle	1	2	3
Distance	G	M	P
Angle	P	M	G
Vitesse	G	M	P

Les règles floues exprimées sur le tableau ci-dessus ont la syntaxe suivante :

- Si Distance est **Grande** Alors Vitesse est **Grande**
- Si Distance est **Moyenne** Alors Vitesse est **Moyenne**
- Si Distance est **Petite** Alors Vitesse est **Petite**
- Si Distance est **Grande** Alors Angle est **Petite**
- Si Distance est **Moyenne** Alors Angle est **Moyenne**
- Si Distance est **Petite** Alors Angle est **Grande**

4.3.4.5 La défuzzification

Il s'agit, pour chaque variable de sortie, de passer des degrés de validité calculés pour chaque conclusion à des grandeurs numériques.

La méthode la plus utilisée est celle du centre de gravité. On calcule le centre de gravité formé par l'union des degrés de validité.

4.4 L'environnement de modélisation et de simulation JDEVS

Les outils de simulation permettent aux utilisateurs d'observer la structure et le comportement d'un système réel. Actuellement il y a plusieurs environnements de modélisation et simulation basés sur le formalisme DEVS permettant la représentation de différents modèles DEVS atomiques ou couplés(DEVSJAVA, DEVS-Suite, CD++, simjava, Ptolemy II…etc).

JDEVS est le résultat du travail de thèse de Jean Baptiste Filippi [FILIPPI, 2003]. Il met en application le paradigme de DEVS de Zeigler adapté à la modélisation et la simulation de systèmes complexes. JDEVS a été développé pour servir de cadre expérimental aux techniques de modélisation de systèmes naturels. Il permet de développer et d'animer des simulations à événements discrets, orientés objets, basées sur des composants interconnectés et collaborant.

Pour spécifier, concevoir et développer notre système, nous avons choisi l'environnement JDEVS comme un outil de simulation pour notre proposition. Parce qu'il offre les avantages suivants :

- ✓ JDEVS est une implémentation du Formalisme DEVS en Java.
- ✓ Permet au modélisateur de spécifier directement les modèles atomiques.
- ✓ Cet outil permet également la définition de modèles DEVS couplés.
- ✓ JDEVS fournit des classes et des objets JAVA.
- ✓ Fournit des outils graphiques qui facilitent la conception et la simulation des modèles DEVS complexes.
- ✓ Un outil robuste et évolutif de simulation.
- ✓ Permet l'intégration et le couplage de méthodologies diverses d'analyses de systèmes par la simulation.

4.4.1 Représentation du modèle proposé à l'aide de l'outil de modélisation JDEVS

La Figure 4.12 est une vue en JDEVS de la figure 4.4 du modèle proposé de la navigation réactive d'un robot mobile. Ce modèle est composé de deux modèles couplés et deux modèles atomiques tels décrits sur la figure 4.4.

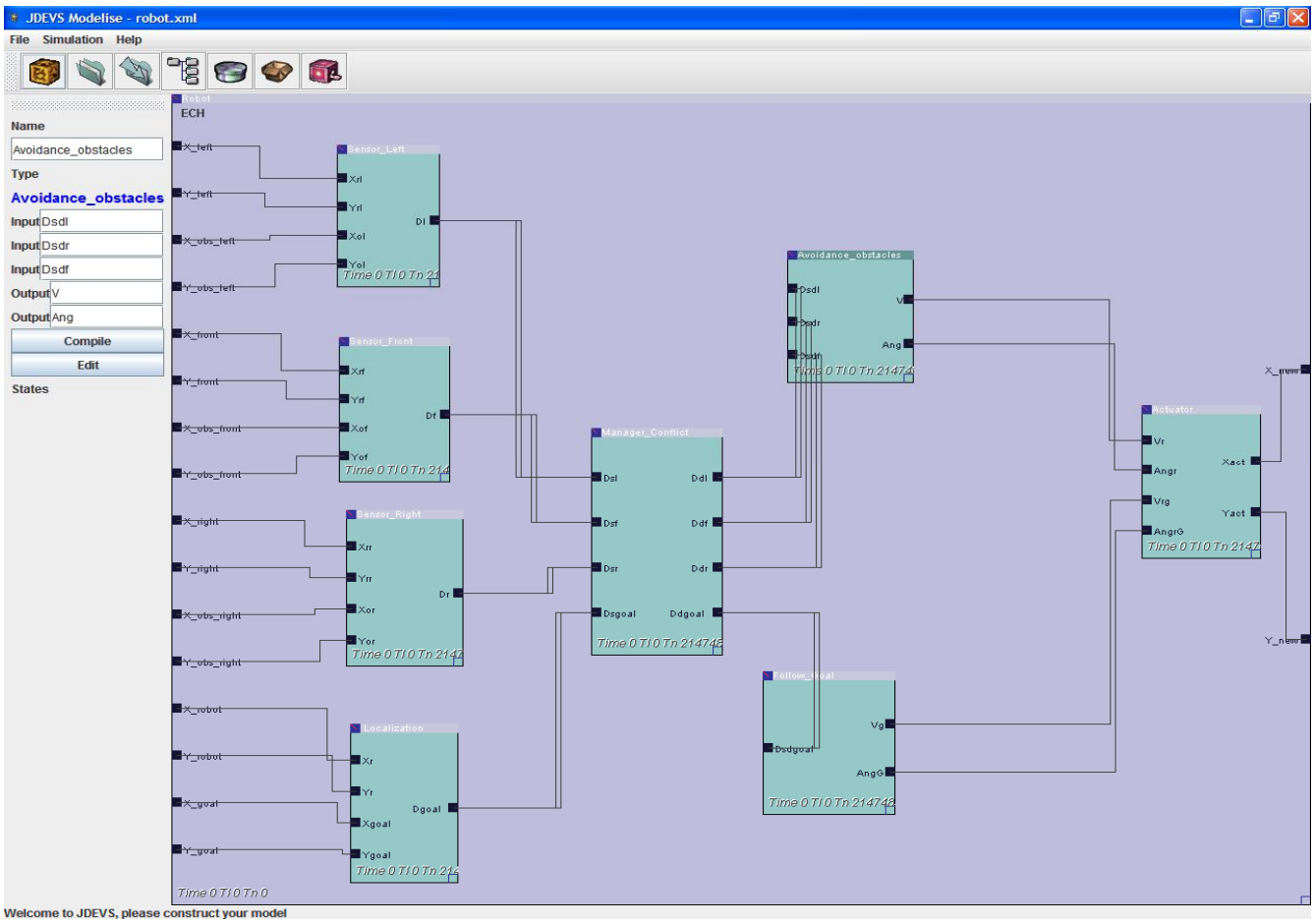


Figure 4.12 : Représentation du modèle sous JDEVS

4.5 Résultats et discussions

Pour valider notre approche couplant le Formalisme DEVS et la logique floue sur le système de la navigation réactive d'un robot mobile dans un environnement incertain, nous avons réalisé à l'aide de notre simulateur, plusieurs expérimentations de navigation dans des environnements plus au moins complexes, et très différents en changeant à chaque fois les positions et le nombre des obstacles et la position de la cible.

4.5.1 Navigation dans un environnement simple

Il s'agit d'un système composé d'un seul robot qui évolue dans un environnement qui ne comporte aucun obstacle.

A cause de l'absence des obstacles dans l'environnement, le seul comportement utilisé par le robot dans ce cas pour accomplir la tâche (atteindre la cible) est le comportement de rapprochement de la cible.

Le chemin suivi par le robot dans sa navigation vers la cible est un chemin optimal (une droite) (Figure 4.13).

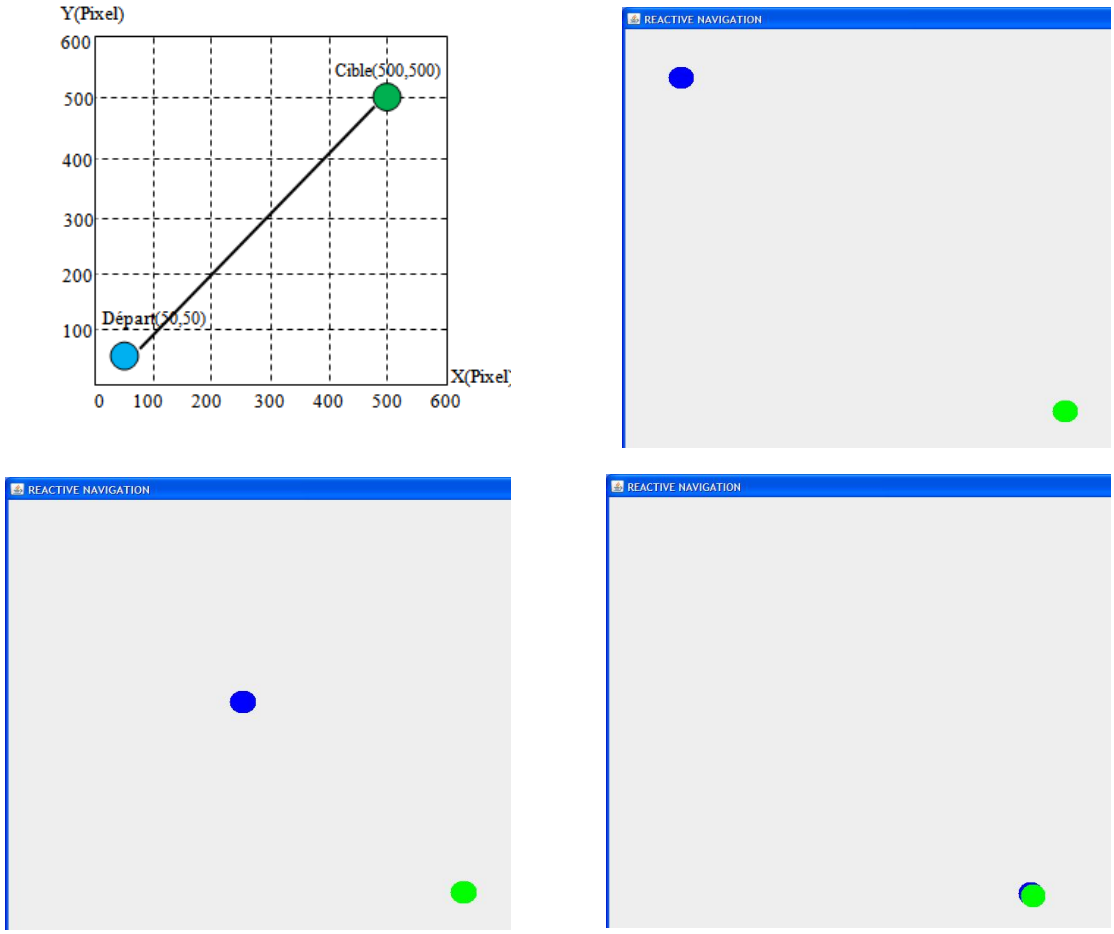


Figure 4.13 : La trajectoire dans un environnement simple

4.5.2 Navigation dans un environnement complexe

La présence des obstacles dans l'environnement de navigation augmente la complexité du système.

Pendant sa navigation vers la cible et pour éviter les obstacles présents dans l'environnement le robot utilise, en plus du comportement de rapprochement de la cible, le comportement d'évitement d'obstacles (Figure 4.14).

Le chemin suivi par le robot dans les environnements complexes est, dans la majorité des cas, un chemin indirect plus au moins optimal.

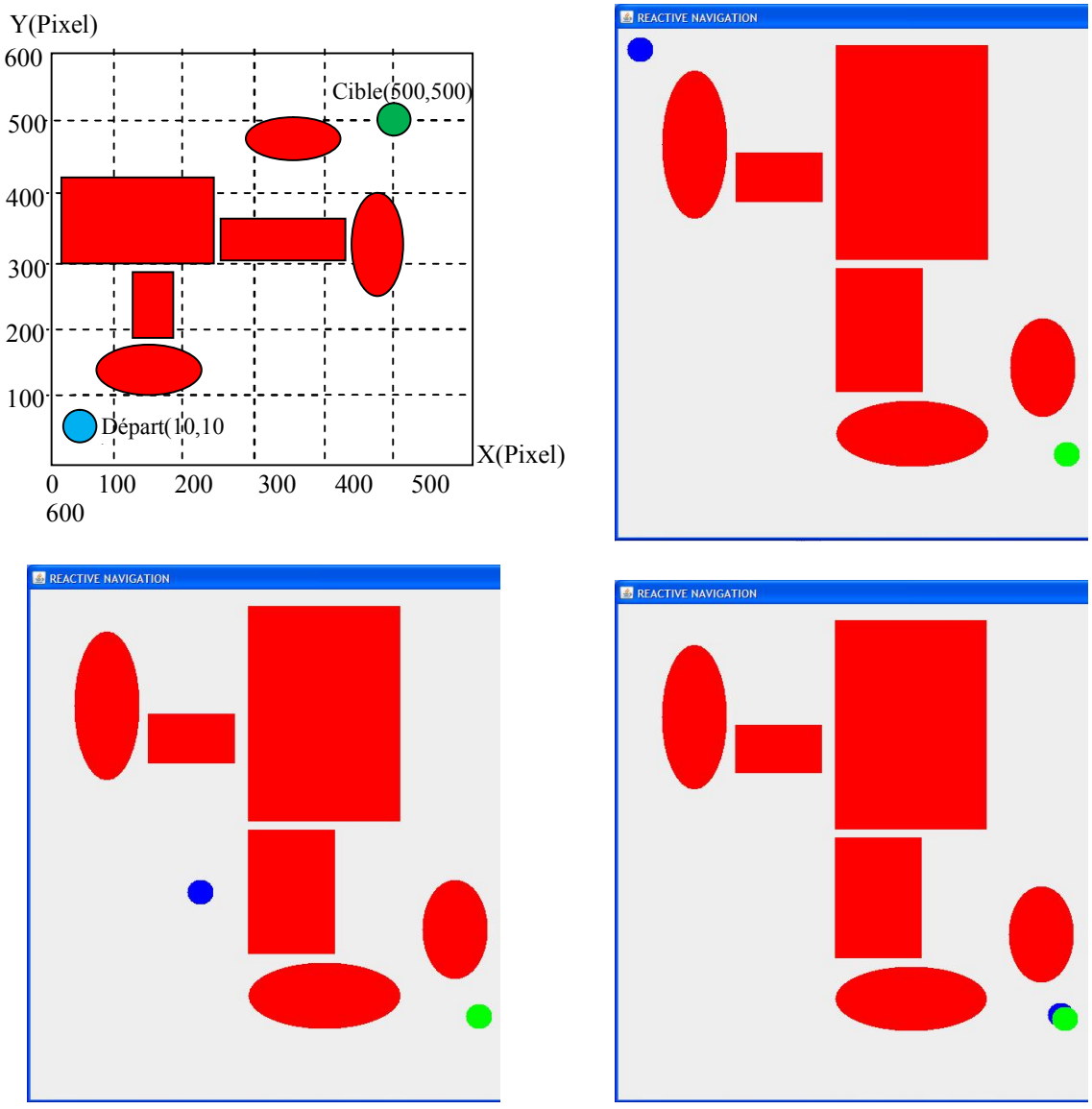


Figure 4.14 : La trajectoire dans un environnement complexe

4.6 Conclusion

Ce chapitre se focalise sur la modélisation et simulation de la navigation réactive d'un robot mobile dans un environnement dynamique et incertain.

Tout d'abord, nous avons décrit la structure globale et l'architecture détaillée de notre système. Ensuite nous avons formalisé en détail le système sous le formalisme DEVS. Enfin nous avons présenté les résultats de simulation de notre système en utilisant l'environnement de modélisation et simulation JDEVS.

Les résultats obtenus en simulant le fonctionnement de notre système dans des environnements simples aussi dans des environnements relativement complexes sont probants. Ceci dit, cette approche nous encourage à la tester dans des environnements plus complexes et dans une dynamique plus contraignante.

Conclusion générale

Conclusion générale et perspectives

L'objectif de ce mémoire est la modélisation et simulation de la navigation réactive d'un robot mobile dans un environnement dynamique et incertain. Notre travail s'inscrit dans le contexte du formalisme DEVS et la théorie de la logique floue.

Le formalisme DEVS, de l'anglais Discrete EVent system Specification, a été introduit par le professeur B.P. Zeigler comme un formalisme abstrait pour la modélisation à évènements discrets.

Le formalisme DEVS est une approche de modélisation basée sur la théorie générale des systèmes. Plus précisément, c'est un formalisme modulaire et hiérarchique pour la modélisation, centré sur la notion d'état. Sa capacité d'ouverture, au sens informatique, en fait un formalisme adapté à un grand nombre de domaines d'application.

La Logique Floue permet la formalisation des imperfections dues aux connaissances inexactes d'un système et à la description du comportement du système par des mots. C'est un moyen efficace pour traiter le problème de la navigation autonome d'un robot mobile.

Dans ce travail, nous avons proposé le couplage entre le formalisme DEVS et la logique floue. Ce formalisme a été choisi à la fois pour sa capacité d'intégration de modèles hétérogènes et pour ses notions de couplage et de décomposition hiérarchique.

Nous avons proposé un modèle qui décrit le système étudié. Basé sur ce modèle, nous avons développé une application à l'aide de l'outil de modélisation et simulation JDEVS pour simuler le comportement du robot mobile autonome dans un environnement inconnu.

L'approche proposée a donné de bons résultats dans des environnements simples aussi dans des environnements relativement complexes.

La suite de nos travaux portera sur plusieurs points. Tout d'abord, nous envisageons l'application de cette proposition sur des systèmes plus complexes. De plus, nous souhaitons combiner le formalisme DEVS avec les autres outils du Soft computing (Réseaux de neuronnes, Algorithmes génétiques etc…). Afin d'ajouter un aspect rationnel dans l'approche que avons adoptée. De plus, au niveau architecture une éventuelle amélioration est souhaitable afin de prendre en compte les différentes facettes de la navigation réactive.

Bibliographie

Bibliographie

[ADACHI, 2003] Adachi, Y.; Saito, H.; Matsumoto, Y.; Ogasawara, T., *"Memory-based navigation using data sequence of laser range finder*", Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International symposium on Volume 1, Issue , 16-20 July 2003 Page(s): 479 -484 vol.1

[ADOUANE, 2005] L. ADOUANE. *Architecture de contrôle comportementale et réactive pour la coopération d'un groupe de robots mobiles.* Thèse de doctorat, Laboratoire d'automatique de Besançon UMR (CNRS 6589), Avril 2005.

[ANDREWS, 1983] Andrews, J. R. and Hogan, N., "*Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator.*" Control of Manufacturing Processes and Robotic Systems, Eds. Hardt, D. E. and Book, W., ASME, Boston, 1983, pp. 243-251.

[ANDRIEN, 2007] ANDRIEN Karine, *"Méthode, formalisme et outil pour le recueil, la modélisation et la simulation de scénarios de comportement"*. Application au domaine des CGF , Thèse de doctorat, 2007, UNIVERSITE PAUL CEZANNE AIX-MARSEILLE III.

[ARNOULD, 1993] P. Arnould – *"Etude de la localisation d'un robot mobile par fusion de données"* – Thèse de doctorat de l'Institut National Polytechnique de Lorraine, Nancy, septembre 1993.

[AYCARD, 1998] O. AYCARD. *"Architecture de contrôle pour robot mobile en environnement intérieur structuré"*. Thèse de doctorat, université Henri Poincaré-Nancy 1, LORIA, Nancy, Juin 1998.

[BAGHLI, 1999**]** L. Baghli, *"Contribution à la commande de la machine asynchrone, utilisation de la logique floue, des réseaux de neurones et des algorithmes génétiques"*, thèse de doctorat de l'université Henri Poincaré, Nancy-I, janvier 1999.

[BARSHAN, 1995] B. Barshan and H.F. Durrant-White, " Inertial navigation system for a mobile robot ", IEEE Trans on Robotics and Automation, Vol 12, N° 5, pp. 328-342, 1995.

[BELKER, 2002] Belker, T., Schulz, D. *"Local Action Planning for Mobile Robot Collision Avoidance"*, Intelligent Robots and System, 2002. IEEE/RSJ International Conference on Volume 1, 30 Sept.-5 Oct. 2002 Page(s):601 - 606 vol.1

[BENCHELOUI, 2011] BENCHELOUI Mohamed, *"Reconstruction Incrémentale de Cartes d'Environnement en Robotique Mobile"*, Mémoire Magister 2011, Université de Batna.

[BISGAMBIGLIA, 2008] BISGAMBIGLIA M. Paul-Antoine, *"Approche de modélisation approximative pour des systèmes à événements discrets : Application à l'étude de propagation de feux de forêt"*, Thèse de doctorat, 2008, UNIVERSITÉ DE CORSE – PASQUALE PAOLI.

[BOREINSTEIN, 1996]. J. Boreinstein, L. Feng – *"Gyrodometry : a new method for combining data from gyros and odometry in mobile robots"* – Proc. of the IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota - April 1996.

[BOREINSTEIN, 1997] J. Borestein, H.R. Everett, L. Feng et D. Wehe – *"Mobile robot positionning - Sensors and techniques"* – Journal of Robotic Systems, vol. 14, n. 4, p. 231-249, 1997.

[BOUCHON, 1993] B. Bouchon-Meunier. *"La logique floue"*, Edition Que sais-je?, Presses universitaire de France.

[BOUCHON, 1995] B. Bouchon-Meunier. *"La logique floue et ses applications"*. Addison-Wesley France, 1995.

[BRAITENBERG, 1984] V. BRAITENBERG. *" Vehicles: Experiments in synthetic psychology"*, MA : MIT Press, Cambridge, 1984.

[BROOKS, 1986] BROOKS, R. *"A robust Layered Control System for a Mobile Robot"*, IEEE, Journal of Robotics and Automation, RA-2, pp: 14-23, April, 1986.

[BUHLER, 1994] H.Buhler *"Réglage par logique floue"* presses polytechniques et universitaires romandes.1994.

[CAPOCCHI, 2005] CAPOCCHI Laurent, *" Simulation concurrente de fautes comportementales pour des systèmes à événements discrets :Application aux circuits digitaux"*, Thèse de doctorat, 2005, UNIVERSITÉ DE CORSE – PASQUALE PAOLI.

[COURBON, 2008] Courbon, J., Mezouar, Y., Lequievre, L., Eck, L., *"Navigation of urban vehicle: An efficient visual memory management for large scale environments"*, IROS 2008: 1817-1822.

[COURBON, 2009] J. Courbon, Y. Mezouar, L. Eck, P. Martinet, "*A generic framework for topological navigation of urban vehicle*", ICRA09 Workshop on Safe navigation in open and dynamic environments Application to autonomous vehicles, ICRA09, Kobe, Japan, May 12th, 2009

[CHATTERJEE, 1996] Chatterjee, R., Matsuno, F., *"Use of single side reflex for autonomous navigation of mobile robots in unknown environments"* ; Robotics and Autonomous Systems, Volume 35, Issue 2, 31 May 2001, Pages 77-96.

[CHOW et ZEIGLER, 2003] Chow, A. C. et Zeigler, B. P. (2003). *"Revised DEVS : A Parallel Hierarchical Modular Modeling Formalism"*. ACIMS Laboratory, University of Tucson, Arizona.

[DJAABOUB, 2009] DJAABOUB Salim, *"Logique floue et SMA : Aide à la décision floue dans les systèmes multi-agents",* Mémoire de magister, 2009 , Universitaire Mentouri de Constantine.

[DAVID, 2006] David FILLIAT *Cours C10 – 2 " Robotique Mobile "* ENSTA 6 octobre 2006 .

[DRAINCOV, 1996**]** D. Draincov, H. Hellendoorn, M.R. Frank. *"An introduction to fuzzy control"*, springer, verlag, 1996

[DROCOURT, 2002] DROCOURT Cyril, *"Localisation et modélisation de l'environnement d'un robot mobile par coopération de deux capteurs omnidirectionnels"*, Thèse de doctorat 2002 , Université de Technologie de Compiègne.

[DUBOZ, 2004] Raphaël Duboz, *"Intégration de modèles hétérogènes pour la modélisation et la simulation de systèmes complexes"*, Thèse de doctorat, Laboratoire d'Informatique du Littoral 2004.

[FATMI, 2005] A. Fatmi, A. Al Yahmadi, L. Khriji, and N. Masmoudi, *"A Fuzzy Logic Based Navigation of a Mobile Robot"* International Journal of Applied Mathematics and Computer Sciences 1;2 , Spring 2005.

[FILLATREAU, 1994] P. Fillatreau – *"Localisation et modélisation tridimensionnelle pour un robot mobile autonome tout terrain"* – Thèse de doctorat de l'Université Paul Sabatier de Toulouse, octobre 1994.

[FILLIAT, 2005] D. FILLIAT. *"Robotique Mobile"*. École Nationale Supérieure de Techniques Avancées, France, 2005.

[FILIPPI, 2003] Jean Baptiste FILIPPI. *"Une architecture logicielle pour la multi-modélisation et la simulation à évènements discrets de systèmes naturels complexes"*. Thèse de doctorat 2003. UNIVERSITÉ DE CORSE PASQUALE PAOLI.

[FRAPPIER, 1990] G. Frappier – *"Système inertiels de navigation pour robots mobiles"* – Séminaire "Les robots mobiles", EC2, Paris, 1990

[FUJIMOTO, 2000] Fujimoto, R.M. *"Parallel discrete event simulation"*. Wiley Interscience, New York, NY, 2000.

[GIAMBIASI, 1995] Giambiasi, N., Frydman, C. and Escudé, B. *"Hierarchical/Multi-View Modelling and Simulation"*. in 7th European Simulation Symposium (Erlangen Nuremberg - Germany., 1995).

[GIAMBIASI, 2001] Giambiasi, N. *"Cours DEA MCAO",* Université Aix Marseille III, 2001.

[GODJEVAC, 1999**]** J.Godjevac. *"Idées nettes sur la logique floue"*, presse polytechnique et universitaire Romandes, Lausanne, 1999.

[HOLLINGUM, 1991] J. Hollingum – *"Caterpillar make the earth move : automatically"* – Industrial Robot, Vol. 18, N°2, pp 15-18, 1991.

[HOPPENOT, 1997] P. Hoppenot, *"Contribution de la robotique mobile à l'assistance aux personnes handicapée "*, Thèse de doctorat, Université d'Evry Val d'Essonne (EVE), 27 nov.1997.

[KEMMOTSU, 1993] K. Kemmotsu, T. Kanade – *"Uncertainty in object pose determination with three light-stripe range measurements"* – Proc. IEEE International Conference on Robotics and Automation, Atlanta (USA), pp. 128-134, May 1993.

[KHATIB , 1986] Khatib, O. 1986. *" Real-time obstacle avoidance for manipulators and mobile robots "*International Journal of Robotic Research, vol. 5, no.1, pp. 90-98

[KLIR, 1995**]** G.J. Klir and B. Yuan, *"Fuzzy Sets and Fuzzy Logic Theory and Applications"*, Prentice Hall PTR, Upper Saddle River, New Jersey, United States of America, 1995.

[KOREN, 1991] Koren, Y.; Borenstein, J. **, "***Potential field methods and their inherent limitations for mobile robot navigation*"; Robotics and Automation, Sacramento, California, April 7-12, 1991, pp. 1398-1404 vol.2

[KORTENKAMP, 1994] D. Kortenkamp – *"Perception for mobile robot navigation: a survey of the state of the art"* – Proc. NASA Dual-use Space Technology Transfer Conf., 1994.

[KLIR, 1985] Klir, G.J.*"Architecture of Systems Problem Solving",* Plenum Press, New York, 1985.

[LEBEDEV, 2005] Lebedev, D.V., Steil, J.J. and Ritter, H.J. *"The dynamic wave expansion neural network model for robot motion planning in time-varying environments"* ; Neural Networks, Volume 18, Issue 3, April 2005, Pages 267-285

[LETTVIN, 1959] Lettvin, J.Y., Maturana, H.R., McCulloch, W.S., & Pitts, W.H., *"What the Frog's Eye Tells the Frog's Brain",* Proceedings of the IRE, Vol. 47, No. 11, pp. 1940-51, 1959

[MORETTE, 2009] MORETTE Nicolas, *"Contribution à la Navigation de robots mobiles : approche par modèle direct et commande prédictive"*, Thèse de doctorat, 2009 , UNIVERSITÉ D'ORLÉANS.

[MILOUDI, 2006] A. Miloudi, *" Etudes et Conception de Régulateurs Robustes dans Différentes Stratégies de Commande d'un Moteur Asynchrone"*, Thèse de doctorat es-sciences, Université des sciences et de la technologie Mohamed Boudiaf d'Oran USTO, Juin 2006.

[MINSKY, 1956] Minsky, M. *"Some Universal Elements For Finite Automata in Automata Studies".* Princeton Uni-versity Press, 1956.

[NERZIOU, 2006] M. Nerziou, *"Modélisation et Commande des Processus Multivariables à base Logique Floue: Application à la Régulation de Vitesse d'un Moteur Asynchrone",* Mémoire de Magister, Ecole Normale Supérieure de l'Enseignement Technologique d'Oran, E.N.S.E.T Oran, Septembre 2006.

[OLIVIER, 2006] Olivier Lefebvre , *"Navigation autonome sans collision pour robots mobiles nonholonomes".* Thèse de doctorat, LAAS-CNRS , 2006.

[OREN, 1987] TI Orën, *"Taxonomy of simulation Model processing in encyclopedia of systems and control"* (eds. M. Singh), Pergamon Press, 1987.

[OUADAH , 2005] Ouadah, N., Azouaoui, O., Hamerlain, M., *"Implémentation d'un contrôleur flou pour la navigation d'un robot mobile de type voiture"*, Troisième Congrès francophone, Majecstic 2005, 16-18 Novembre 2005, Rennes (France).

[PRADALIER, 2005] Pradalier, C., Hermosillo, J., Koike, C., Braillon, D., Bessière, P., Laugier, C., *"The CyCab: a car-like robot navigating autonomously and safely among "*. Robotics and Autonomous Systems (RAS) 50(1):51-67 (2005)

[RAMSEY, 1931] F.P. Ramsey. Truth and probability. *"The Foundations of Mathematics and other Logical Essays"*, 1931.

[SLIMANE, 2005] SLIMANE Noureddine *" Système de localisation pour robots mobiles". Thèse de doctorat 2005,* Université de BATNA

[UHRMARCHER, 2001] A. Uhrmarcher. Dynamic Structures in Modeling and Simulation : *"A Reflective Approach"*. ACM Transactions on Modeling and Computer Simulation vol. 11 2001, pages 206–232, 2001.

[VAGANAY, 1993] J. Vaganay – *"Conception d'un système multisensoriel de localisation dynamique 3D pour robot mobile"* . Thèse de doctorat, LIRMM, Montpellier, juillet 1993.

[WAINER, 2003] Alejandro Troccoli and Gabriel Wainer. *"Implementing parallel cell-DEVS"*. In IEEE, editor, *Proceedings of the 36th Annual Simulation Symposium*, 2003.

[WANG, 2008] Wang, X.; Hou, Z.; Zou, A.; Tan, M.; Cheng, L. , *"A behavior controller based on spiking neural networks for mobile robots"* ,Neurocomputing Volume 71 , Issue 4-6 (January 2008) Pages 655-666.

[YAHYAOUI, 2006] K. YAHYAOUI. *"Conception d'un système Multi-Agents pour une navigation réactive d'une plate-forme robotique Mobile"*. Thèse de magister, université de mascara, Algérie, 2006.

[ZADEH, 1965] L.A. Zadeh. *"Fuzzy sets"*. Information and Control, 8 :338–353, 1965.

[ZADEH, 1968] L.A. Zadeh. *"Fuzzy algorithm ",* Information and Control, 12 :94–102, 1968.

[ZAGANE, 2010] M. ZAGANE. *"Une architecture décisionnelle de contrôle pour un groupe de robots mobiles coopératifs dans un environnement dynamique et non structuré"*. Thèse de magister, ESI, Algérie, 2010.

[ZEIGLER, 1976] Zeigler, B.P. *"Theory of Modelling and Simulation"*. Wiley & Sons, New York, NY, 1976.

[ZEIGLER, 1984] Zeigler, B.P. *"Multifacetted modelling and discrete event simulation"*. Academic Press Profes-sional, Inc., San Diego, CA, 1984.

[ZEIGLER, 1990] Zeigler, B. P. (1990). *"Object-Oriented Simulation with Hierarchical, Modular Models"*.

[ZEIGLER, 1999] Zeigler, B.P., Ball, G., Cho, H.J. and Lee, J.S., *"Implementation of the DEVS formalism over the HLA/RTI: Problems and solutions"*. In Simulation Interoperation Workshop (SIW), (Orlando, FL, 1999).

[ZEIGLER, 2000] Zeigler, B.P., Praehofer, H. and Kim, T.G. *"Theory of Modeling and Simulation"*, New York, NY, 2000.

Table des matières

Chapitre 3 :
Le Formalisme DEVS

Chapitre 4 :
Modélisation et Simulation