Apprentissage multisource par programmation logique inductive : application à la caractérisation d'arythmies cardiaques

Elisa Fromont

N° d'ordre: 3291

THÈSE

Présentée devant

devant l'Université de Rennes 1

pour obtenir

le grade de : Docteur de l'Université de Rennes 1
Mention Informatique

par

Élisa Fromont

Équipe d'accueil : Dream - IRISA
École Doctorale : Matisse
Composante universitaire : IFSIC

Titre de la thèse :

Apprentissage multisource par programmation logique inductive : application à la caractérisation d'arythmies cardiaques

soutenue le 7 décembre 2005 devant la commission d'examen

Président	Mme Mireille Ducassé	Pr. INSA de Rennes
Rapporteurs :	Mme Catherine Garbay	DR CNRS
	Mme Céline Rouveirol	MdC, HDR, Université Paris Sud
Examinateur	M. Hendrik Blockeel	Pr. Katholieke Universiteit, Leuven
Co-encadrant	M. Guy Carrault	Pr. Université de Rennes 1
Directrice de thèse	Mme Marie-Odile Cordier	Pr. Université de Rennes 1
Co-encadrant	M. René Quiniou	CR INRIA

Remerciements

Tout d'abord je tiens à exprimer ma gratitude aux personnes qui ont accepté de juger ce travail, en particulier Mireille Ducassé, professeur à l'INSA de Rennes, qui m'a fait l'honneur de présider mon jury et de relire ma thèse comme elle seule sait le faire ; Catherine Garbay, directrice de recherche au CNRS et Céline Rouveirol, maître de conférences habilitée à diriger des recherches à l'Université de Paris Sud, qui ont accepté la charge de rapporteur et m'ont toutes deux fourni des commentaires très précieux pour améliorer ce document. Un grand merci également à Hendrik Blockeel qui m'a fait l'immense plaisir de venir de Belgique pour cette soutenance.

Le travail présenté dans ce mémoire de thèse a été financé grâce à un contrat RNTS sur un projet en collaboration avec le LTSI, l'équipe DREAM de l'IRISA, le département de cardiologie du CHU de Rennes et la société ELA Medical. Je tiens donc à remercier tous les acteurs de ce projet en particulier Philippe Mabo, professeur et cardiologue au CHU de Rennes, et Marcel Limousin de la société ELA Medical pour l'intérêt qu'ils ont montré envers mes travaux lors de nos différentes rencontres. Je remercie également Guy Carrault et Alfredo Hernández repectivement professeur et chercheur Inserm au Laboratoire du Traitement du signal et de l'Image de l'Université de Rennes 1. Guy pour m'avoir encadré, m'avoir gardée connecté aux réalités cliniques et soutenu dans les moments les plus décourageants que l'on peut traverser durant une thèse ; Alfredo, pour son enthousiasme constant et communicatif, son impressionnante réactivité et l'aide précieuse qu'il m'a apportée au début de ma thèse. À ce sujet, je remercie profondément Mr Pierre Scordia, ancien cardiologue au CHU de Rennes pour ses "cours particuliers" en cardiologie sans lesquels j'aurais eu bien du mal à m'approprier mon sujet de thèse.

Je souhaiterais, encore une fois, exprimer mes remerciements à Mireille Ducassé, pour m'avoir donné le goût de chercher, pour m'avoir poussée et encouragée tout au long de ces dernières années et notamment pour m'avoir recommandée à l'équipe DREAM à la fin de mon stage de DEA.

Quand j'exprime ma reconnaissance quant à avoir été recommandée à l'équipe DREAM, je ne mets pas suffisamment en valeur l'environnement de travail incomparable dans lequel j'ai pu évoluer. Je remercie donc du fond du coeur tous les membres de l'équipe DREAM pour leur bonne humeur, leur soutien, leur aide et leur solidarité. Parmi ces

membres j'adresse mes plus vifs remerciements à mes encadrants Marie-Odile Cordier et René Quiniou pour leur disponibilité et leur patience tout au long de ces trois années. Je décerne une mention spéciale à Philippe qui, non content d'être un incomparable compagnon de jeux, n'a pas d'égal pour faire comprendre, avec patience, les principes les plus alambiqués de la logique. La palme revient à mes compères Alban et François (et ceux qui sont arrivés ensuite, notamment Alex et Ronan) pour avoir supporté avec beaucoup de tolérance mes lacunes en orthographe et mes problèmes techniques récurrents (et légèrement surnaturels) et pour leur humour (toujours très fin :-p) sans lequel ces trois années n'auraient jamais pu être aussi agréables : MERCI !

Je remercie affectueusement mes parents et mon frère pour la confiance qu'ils m'ont toujours témoignée et pour leur encouragements à poursuivre cette longue route. Un dernier et immense MERCI à Olivier et mes amis pour m'avoir soutenue constamment tout en maintenant autour de moi une bulle d'air frais qui m'a permis de m'éloigner à chaque fois que je le souhaitais des préoccupations, parfois étouffantes, de cette thèse. Une pensée pour Merwann qui devra finalement m'appeler docteur...

Elisa

Table des matières

Introduction

Les maladies cardio-vasculaires représentent aujourd'hui la principale cause de mortalité chez les adultes (29.3% des décès enregistrés) [World Health Organization, 2004] dans tous les pays membres de l'OMS, la mortalité la plus élevée (12.6% des décès en 2002) étant attribuée aux *cardiopathies ischémiques*. Les cardiopathies ischémiques sont des troubles de la fonction cardiaque provoqués par un flux sanguin insuffisant vers les tissus musculaires du cœur. L'interruption prolongée de l'apport de sang aux tissus du myocarde peut avoir comme conséquence la nécrose du muscle cardiaque (ou *infarctus du myocarde*). L'infarctus du myocarde peut provoquer de graves troubles du rythme cardiaque, certains comme la *fibrillation ventriculaire*, pouvant provoquer une mort subite.

Les appareils utilisés pour suivre l'évolution du cœur sont appelés systèmes de *monitoring cardiaque*. Ces systèmes sont composés à la fois des capteurs permettant de recueillir des données sur l'activité du cœur et des unités centrales permettant de stocker ou de visualiser ces données. L'évolution de ces systèmes entre les années 60 et 90, la multiplication du nombre de données disponibles et la relative autonomie de ces systèmes quand ils sont, par exemple, sous la forme de pacemaker cardiaque, ont conduit à l'émergence des *systèmes de monitoring intelligents*. Les unités centrales doivent donc maintenant, en plus des tâches de stockage et de visualisation définies précédemment, interpréter les données pour prévenir et assister le personnel médical et éventuellement, agir lorsque l'état du cœur évolue dangereusement.

L'importance de ces tâches rend la qualité, la sensibilité et la robustesse de ces systèmes primordiales. Pour augmenter la fiabilité de la détection des problèmes cardiaques, notamment en présence des *bruits* liés au mouvements des patients, à la perte d'un signal, à la présence d'artefacts sur le signal etc., les systèmes de monitoring intelligents peuvent utiliser plusieurs sources de données.

Pour diagnostiquer des troubles du rythme cardiaque, les médecins ont pour habitude d'utiliser principalement l'électrocardiogramme (ECG). Lorsque l'ECG est bruité, le cerveau humain est capable, dans une certaine mesure, de retrouver les informations exactes parmi les informations bruitées ou de chercher dans d'autres sources de données (par exemple sur la courbe de la pression artérielle) la confirmation de ces informations. Les rares instants où l'ECG est inutilisable, le médecin se sert des autres sources de données en se reposant sur son bon sens. L'automatisation de cette recherche d'informations n'est pas triviale, notamment parce qu'il est difficile d'extraire du signal ECG toutes les informations dont se sert le médecin pour diagnostiquer des arythmies, mais

aussi parce que le système de monitoring doit être capable, si l'ECG est trop bruité pour pouvoir être utilisé tel quel, de trouver l'information nécessaire dans les autres sources de données, même si elles ne sont *a priori* pas suffisantes pour permettre un diagnostic fiable. En outre, fournir à un système suffisamment de connaissances expertes pour automatiser le diagnostic est une tâche extrêmement coûteuse et fastidieuse.

Avec l'arrivée de nouveaux capteurs, le besoin d'acquérir de la connaissance sur de nouvelles données automatiquement et de développer des techniques permettant de combiner plusieurs sources de données pour augmenter la fiabilité du diagnostic est d'autant plus présent. Ceci est vrai d'une part dans les unités de soins intensifs spécialisées dans les maladies du cœur, mais également dans le cadre des prothèses cardiaques de nouvelle génération.

Les travaux présentés ici sont financés par le projet RNTS Cepica dont l'objectif à long terme est de développer des prothèses cardiaques élaborées, capables de gérer simultanément et en temps réel des informations provenant de capteurs reflétant l'activité hémodynamique (liés à la circulation sanguine) et rythmique des patients souffrant de cardiopathie et d'insuffisance cardiaque. Ce projet fait directement suite à un autre projet (Pise) dont les résultats principaux ont été publiés dans la thèse de Feng Wang [Wang, 2002]. Ce dernier projet a débouché sur la création d'un prototype de système de monitoring cardiaque intelligent, Calicot, permettant de diagnostiquer en ligne des arythmies cardiaques à partir de données provenant d'un ECG.

Le but de cette thèse est, d'une part de proposer des techniques permettant d'acquérir de la connaissance sur des données provenant de nouveaux capteurs, et d'autre part de développer des méthodes pour pouvoir utiliser conjointement plusieurs sources de données : on cherche notamment à acquérir de la connaissance multisource ie., de la connaissance qui met en relation les informations disponibles sur chacune des sources de données.

Suite aux résultats très encourageants obtenus avec le prototype Calicot, la méthode d'acquisition automatique de la connaissance choisie est la *programmation logique inductive* (PLI). La PLI est une technique d'apprentissage relationnel. L'apprentissage relationnel, comme son nom l'indique, permet d'exhiber des relations entre les évènements caractéristiques se produisant sur les différentes sources. La logique du premier ordre utilisée dans ce paradigme permet une formulation des règles produites et des briques nécessaires à l'apprentissage par PLI dans un langage assez facilement interprétable par les médecins.

Cependant, l'expressivité du langage utilisé en PLI est également le principal inconvénient de cette technique. Le nombre de solutions envisageables pour trouver une description compacte et discriminante d'un concept (dans notre cas, d'une arythmie) est en général trop important pour effectuer une simple énumération. Le défi des différents systèmes de PLI existant dans la littérature est donc de trouver un moyen efficace de parcourir l'espace des solutions afin de trouver la meilleure solution possible.

Pour pouvoir apprendre des règles mettant en relation les évènements se produisant sur les différentes sources, une méthode naïve consiste à agréger toutes les informations disponibles puis à effectuer un apprentissage sur ces données multisources. Cependant, chaque source apporte un langage propre permettant de décrire les données, et le vo-

lume de ces données augmente proportionnellement au nombre de sources. La richesse du langage influe sur la taille de l'espace de recherche des solutions et le volume des données sur les temps de calcul. Comme l'a fait remarquer Fürnkranz [Fürnkranz, 1997], la recherche de techniques novatrices pour réduire la dimensionalité des problèmes de PLI est encore d'actualité. Quinlan propose dans [Quinlan, 1983] une technique de *fenêtrage* améliorée par la suite par Fürnkranz [Fürnkranz, 1998] pour réduire l'espace des exemples (le nombre d'exemples à tester pour vérifier le bien fondé d'une hypothèse). Pour réduire l'espace de recherche, de nombreuses méthodes ont été proposées, l'une d'elle étant l'utilisation de biais déclaratifs [Nédellec *et al.*, 1996]. Sans un tel biais, le système de PLI peut produire des solutions aberrantes dans le cas général, ne produire aucune solution dans le pire des cas ou nécessiter des temps de calcul trop élevés dans le meilleur. Cependant, l'écriture d'un tel biais, pour qu'il soit suffisamment restrictif sans pour autant perdre la ou les solutions au problèmes, demande des connaissances très fines sur le domaine d'application et dans le cas multisource, des connaissances liées aux relations entre les éléments des différentes sources. Pour résoudre ce problème d'apprentissage multisource, nous avons proposé une méthode décrite dans [Fromont *et al.*, 2005a], qui tire parti d'apprentissages effectués source par source pour biaiser automatiquement l'espace de recherche lors de l'apprentissage sur l'ensemble des données multisources.

Les résultats sur une base d'apprentissage non bruitée montrent, d'une part, que les apprentissages monosources donnent de très bons résultats (précision en apprentissage et en test supérieure à 90%) pour la majorité des arythmies, ce qui confirme le choix de l'utilisation de l'apprentissage par PLI. De plus, la méthode multisource biaisée offre toujours des résultats aussi bons, voire meilleurs dans le cas où les sources sont complémentaires, que les apprentissages monosources, ce qui confirme le bien fondé de l'utilisation de plusieurs sources pour améliorer la précision du diagnostic. Enfin ces résultats multisources biaisés sont souvent meilleurs et obtenus de manière plus efficace que des résultats multisources obtenus en construisant le biais de langage manuellement.

Les résultats obtenus sur une base de données bruitées, ce qui est le contexte réaliste des milieux hospitaliers, montrent que l'utilisation de plusieurs sources de données en permanence par un système de monitoring médical, ne donne pas de très bons résultats. En effet, les règles multisources sont plus sensibles aux bruits que les règles monosources puisqu'elles nécessitent, pour fonctionner correctement, que toutes les sources soient non bruitées pendant une période donnée. L'utilisation de plusieurs sources de données est en revanche très intéressante couplée à un module de pilotage automatique qui permet de sélectionner en temps réel les sources utilisables et les règles dont le langage est adapté au bruit détecté sur ces sources.

Pour une meilleure compréhension du contexte applicatif de cette étude, nous présentons dans le chapitre 1 une définition précise du monitoring médical et nous donnons quelques bases d'électrocardiographie clinique centrées sur l'utilisation de deux mesures : le recueil de l'ECG et la mesure de pression artérielle. Nous donnons également une brève description des arythmies cardiaques les plus courantes et nous mettons l'accent sur les arythmies utilisées dans cette étude. Une description complémentaire des

signaux utilisés se trouve dans l'annexe A. Nous donnons ensuite un état de l'art des systèmes de monitoring médicaux intelligents et en particulier ceux qui s'intéressent, comme nous, à l'acquisition automatique des connaissances. Nous présentons ensuite les techniques envisagées en Intelligence Artificielle appliquées au domaine médical pour prendre en compte plusieurs sources de données simultanément. Ces techniques sont essentiellement empruntées au domaine très vaste de la fusion de données mais nous nous limiterons volontairement à celles ayant une application dans le domaine de la cardiologie.

Le chapitre 2 donne une description détaillée de deux systèmes de monitoring cardiaque : CALICOT, le prototype créé pendant le projet PISE, et KARDIO, un système permettant l'interprétation des rythmes cardiaques avec lequel est souvent comparé CALICOT. Nous donnons dans ce chapitre la première contribution de nos travaux de thèse consistant en une méthodologie de comparaison permettant de différencier ces deux systèmes [Fromont *et al.*, 2003].

Le chapitre 3 explique en détail le paradigme de la PLI et la différence entre les différents systèmes existant dans la littérature. Nous en avons testé deux : ICL [De Raedt et Van Laer, 1995] et ALEPH [Srinivasan, 2003] et nous avons cherché à acquérir des règles permettant de caractériser certaines arythmies cardiaques à partir de données provenant d'un ECG et de mesures de pression artérielle. La sélection des attributs significatifs pour décrire chaque source, la construction des biais et les règles apprises forment la seconde contribution de cette thèse. Différents niveaux de langage d'apprentissage ont été expérimentés, les règles apprises et leurs performances respectives acquises par une technique de validation croisée sont données en annexe C. Ces résultats, très largement en faveur d'ICL sont utilisés dans le chapitre 4 sur l'apprentissage multisource.

Le chapitre 4 donne une formalisation de l'apprentissage multisource par PLI et propose un algorithme pour biaiser efficacement l'espace de recherche pour ce type d'apprentissage. Des résultats obtenus en comparant l'apprentissage biaisé à un apprentissage plus naïf avec un biais conçu manuellement sont présentés et comparés aux résultats monosources. Des expériences ont également été menées dans le cadre du prototype CALICOT en milieu bruité et offre des résultats encourageants.

Nous concluons par un bilan des principales contributions de nos travaux et nous proposons quelques pistes de recherches ouvertes à l'issue de ces derniers.

Chapitre 1

Le diagnostic automatique des arythmies cardiaques à partir de données multisources

Nous nous intéressons à la détection automatique et à la caractérisation des arythmies cardiaques à partir de données multisources.

Ce chapitre débute par un bref aperçu des possibilités actuelles, dans le milieu médical, de surveillance continue des patients. Cette surveillance peut permettre par l'intermédiaire d'appareils de *monitoring*, de détecter de manière précoce une dégradation de l'état général du patient et, dans le meilleur des cas, de proposer un diagnostic automatiquement.

Après une introduction générale au *monitoring médical*, nous définirons précisément le terme *d'arythmie cardiaque* en présentant brièvement le fonctionnement du cœur et des signaux utiles pour analyser son état.

La seconde partie de ce chapitre donne un état de l'art des systèmes actuels dit "intelligents" élaborés dans un cadre de monitoring médical permettant de fournir un diagnostic automatique. Cette partie s'éloignera volontairement du cadre restrictif des arythmies cardiaques pour montrer l'étendue des techniques utilisées actuellement.

La troisième partie aborde le problème de l'utilisation de plusieurs sources de données dans un but de diagnostic fiable.

1.1 Contexte applicatif

Cette section présente le contexte applicatif de notre étude. Nous commençons par une introduction générale au monitoring médical puis nous nous intéressons plus particulièrement au monitoring dans le cadre des unités de soins intensifs pour coronariens c'est à dire dans les unités hospitalières spécialisées dans la surveillance des maladies liées au coeur. Pour comprendre précisément les termes qui seront utilisés dans la suite de ce document, nous faisons une brève introduction à la cardiologie centrée sur le domaine particulier de la rythmologie.

1.1.1 Le monitoring

Le terme *monitoring* est un abus de langage emprunté aux médecins pour parler de la surveillance continue de patient. En français, il faudrait parler de *monitorage*. L'activité de *monitorage* s'applique à de nombreux domaines industriels autres que la médecine tels que les télécommunications, la surveillance de centrales nucléaires, l'industrie du pétrole etc. Le groupe de recherche ALARM [Cauvin *et al.*, 1998], composé de plusieurs chercheurs travaillant sur ce domaine particulier, ont proposé la définition suivante au terme de *monitorage* : "l'activité de monitorage d'un système dynamique peut être vue comme le suivi continu d'un système par un module de haut niveau qui analyse toutes les situations rencontrées, communique avec un opérateur humain et suggère des décisions à prendre en cas de dysfonctionnement du système." L'activité de *monitorage* repose sur l'utilisation d'alarmes qui elles mêmes reposent sur l'utilisation de capteurs. Le défi pour l'opérateur humain est d'interpréter le nombre très important (dans un système réaliste) de ces alarmes.

Dans le contexte médical le terme *monitoring* désigne à la fois des fonctions de surveillance, de diagnostic et de test, il s'applique au patient et aux systèmes qui les supportent [Pierson, 1998]. Nous garderons donc ce terme dans la suite de cette étude pour parler de *monitorage médical*.

On peut distinguer deux types de monitoring dans le contexte médical : le *monitoring en temps réel* et le monitoring *hors ligne*.

Le monitoring en temps réel est habituellement associé aux unités de soins intensifs (USI). Les USI sont des unités cliniques qui accueillent les patients dont les conditions cliniques peuvent changer rapidement et pour lesquels les risques de décès sont élevés. Les soins pratiqués dans ces unités peuvent eux mêmes provoquer des changements rapides et importants de l'état physiologique du patient. Dans ces conditions, le monitoring des patients est vital. Le monitoring en USI est plus précisément défini comme l'observation et la mesure, en temps réel, de manière répétée ou continue, du patient, de ses fonctions physiologiques et des instruments qui lui permettent de se maintenir dans un état stable, pour faciliter les prises de décisions du personnel médical. Ces décisions peuvent aller de l'intervention thérapeutique proprement dite à la simple estimation de l'effet de ces interventions. Une liste détaillée des buts recherchés par les systèmes de monitoring en USI est donnée dans [Pierson, 1998]. Les appareils de monitoring se servent d'informations fournies par des capteurs. Ces appareils et surtout ces capteurs peuvent être invasifs i.e. implantés dans le corps du patient (c'est le cas des cathéters artériels qui mesurent la pression sanguine artérielle, de la ventilation mécanique, etc.), semi-invasifs (comme les sondes œsophagiennes) ou non invasifs (l'électrocardiogramme, la température etc.), selon l'état du patient. Les données provenant d'un capteur invasif sont beaucoup plus précises et permettent un suivi continu de longue durée mais ces capteurs sont beaucoup moins bien tolérés par les patients et leur utilisation est plus coûteuse. Certains appareils de monitoring, comme le pacemaker cardiaque (utilisé hors USI) nécessitent un passage au bloc opératoire, ce qui augmente les risques post-opératoires pour le patient. L'intensité du monitoring, i.e. le nombre de capteurs utilisés et l'*invasivité* de ces capteurs, est fonction de la gravité de l'état du patient et

des interventions qui ont été précédemment faites sur ce même patient.

Le second type de monitoring concerne les bandes enregistrées par exemple, par les moniteurs électrocardiographiques non invasifs Holter ou les Mesures Ambulatoires de Pression Artérielle (MAPA). Un Holter permet d'enregistrer l'activité électrique cardiaque de manière continue pendant 24 heures. Le Holter est un appareil de monitoring qui enregistre l'activité cardiaque du patient par le biais d'électrodes placées sur son corps. Le but d'un tel système est de s'assurer de la tolérance d'un patient à des activités de tous les jours telles que la marche, la conduite, l'ingestion de nourriture ou le sommeil. Pour permettre une interprétation *a posteriori* correcte des signaux enregistrés, le patient est tenu de répertorier lui même tous les changements d'activités effectués pendant ces 24 heures.

Les unités de soins intensifs spécialisées dans le soin des maladies cardiaques sont appelées unités de soins intensifs pour coronariens (USIC). Dans la suite de ce mémoire, nous nous focalisons sur le monitoring cardiaque pratiqué dans les USIC.

1.1.2 Le monitoring cardiaque

FIG. 1.1 – Un appareil de monitoring cardiaque

FIG. 1.2 – Exemple de signaux enregistrés par des systèmes de monitoring cardiaques

La Figure 1.1 donne un exemple de moniteur cardiaque et la Figure 1.2 un exemple de signaux apparaissant sur un tel moniteur. La plupart des unités de soins intensifs sont équipées d'un grand nombre d'appareils permettant de surveiller l'état d'un patient et de maintenir cet état stable. Parmi les plus représentés, on trouve des appareils de mesure numérique de la fréquence cardiaque, des appareils d'oxygénation (ventilateur), des mesures de saturation en oxygènes (SpO2), des mesures hémodynamiques (pression

sanguine) invasives et non invasives, des recueils de l'électrocardiogramme (ECG) avec des moniteurs centraux permettant de stocker une partie des données, des appareils d'échographie, des phono-cardiogrammes etc.

Deux types de données provenant de ces appareils sont particulièrement utilisés pour le monitoring des arythmies cardiaques : l'ECG (cf. les deux premiers signaux de la Figure 1.2) et le signal hémodynamique ou signal de pression sanguine. Pour comprendre l'intérêt de ces mesures pour détecter et soigner les troubles du rythme cardiaque, nous proposons de donner une brève introduction à la cardiologie centrée sur l'utilisation de ces deux appareils.

1.1.3 Introduction à la cardiologie

Cette introduction à la cardiologie est volontairement limitée aux notions utiles à la compréhension des chapitres suivants. Le lecteur intéressé pourra cependant trouver des informations complémentaires détaillées dans [Blondeau et Hiltgen, 1980] et de manière plus intuitive dans [Wang, 2002] et [Hernández, 2000].

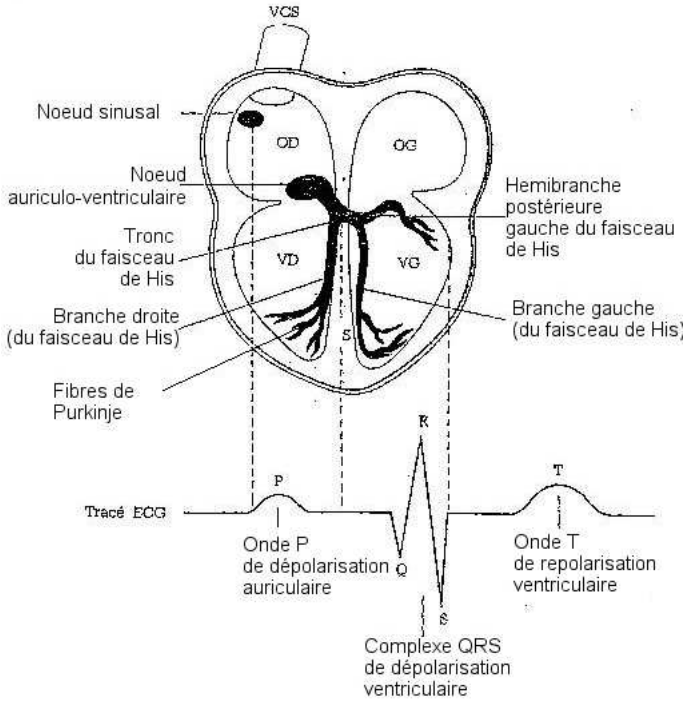


FIG. 1.3 – Éléments fondamentaux de la conduction électrique cardiaque et tracé ECG correspondant. OD, OG, VD et VG signifient respectivement oreillette gauche et droite et ventricule gauche et droit.

1.1.3.1 La conduction électrique

La contraction du muscle cardiaque (ou myocarde) a pour origine la propagation d'une onde électrique qui excite les cellules musculaires dans un ordre bien établi afin que la contraction soit la plus efficace possible. Le système de conduction électrique comprend (cf. Figure 1.3) : le nœud sinusal, les voies spécialisées internodales, le nœud auriculo-ventriculaire, le faisceau de His, les branches gauche et droite et les fibres de Purkinje qui terminent les deux branches. Dans le cas normal, le stimulus physiologique à l'origine de l'onde électrique (et donc du battement cardiaque) provient du nœud sinusal. Ce stimulus est fourni rythmiquement à une fréquence comprise entre 60 et 100/minute. L'impulsion cardiaque initiée dans le nœud sinusal est ensuite transmise aux deux oreillettes au moyen des voies internodales qui relient le nœud sinusal au nœud auriculo-ventriculaire. L'onde se propage ensuite à travers le faisceau de His puis vers les branches gauche et droite jusqu'aux fibres de Purkinje. La propagation de l'onde se fait par une succession de dépolarisation des tissus du myocarde. Après le passage de l'onde, la membrane des tissus se repolarise. Cette succession de dépolarisations et de repolarisations des tissus cardiaques est visible sous plusieurs angles sur un électrocardiogramme (ECG).

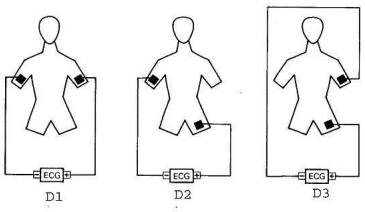


FIG. 1.4 – Dérivations bipolaires (voies I II et III)

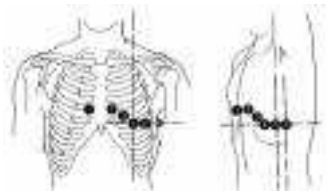


FIG. 1.5 – Dérivations précordiales (voies V1 V2...V6)

L'électrocardiogramme L'ECG reflète l'activité électrique cardiaque. L'électrocardiographie moderne s'appuie sur l'étude de douze dérivations (ou dérivations standards). Ces dérivations sont obtenues en plaçant des électrodes sur le thorax près du cœur, les précordiales (6 voies de V1 à V6), ou sur les bras et les jambes (dérivations bipolaires D1 à D3 et uni-polaires aVR, aVL et aVF) comme indiquées sur les Figures 1.4 et 1.5. Ces différentes dérivations permettent d'avoir plusieurs *vues* de la propagation électrique. Lorsqu'il n'y a aucun dysfonctionnement, ces vues sont redondantes, on se sert alors principalement des voies bipolaires I et II. L'ensemble des vues est cependant nécessaires en cas de dysfonctionnement pour localiser précisément la source du problème. Lors de la propagation de l'onde électrique dans le cœur, les électrodes permettent d'enregistrer un motif électrique composé d'une succession d'ondes caractéristiques désignées par les lettres de l'alphabet à partir de P (motif ECG Figure 1.3). L'onde P initiale représente la dépolarisation des oreillettes. L'ensemble $Q-R-S$ (appelé aussi complexe QRS) traduit la dépolarisation des ventricules tandis que l'onde

T correspond à leur repolarisation (la repolarisation des oreillettes est masquée par celle des ventricules). L'onde U, présente sur la Figure 1.6 est inconstante et de faible amplitude, elle a une signification encore discutée.

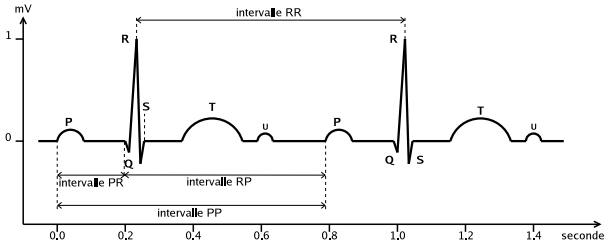


FIG. 1.6 – Battements normaux et intervalles d'intérêt

1.1.3.2 L'activité mécanique cardiaque

Pour caractériser le rythme cardiaque, les cardiologues se fient à la forme générale de l'ECG. Ils portent une attention particulière à la forme des ondes citées précédemment et aux intervalles de temps qui les séparent. La Figure 1.6 répertorie les intervalles qui seront utilisés dans la suite de cette étude. Les valeurs numériques utilisées proviennent de la littérature médicale et notamment de [Stein, 1999].

- l'intervalle RR (ou RR1) sépare les sommets de deux ondes R consécutives et définit le rythme ventriculaire. Un intervalle RR normal est généralement mesuré entre 600 et 1000 ms,
- l'intervalle PP (ou PP1) sépare les sommets de deux ondes P consécutives et définit le rythme auriculaire,
- l'intervalle PR (ou PR1) s'étend du début de l'onde P au début du complexe QRS. Un intervalle PR normal mesure entre 120 et 200 ms,
- l'intervalle RP (ou RP1) s'étend du début du complexe QRS jusqu'à l'onde P suivante,
- les intervalles RR2, PP2 séparent respectivement les sommets de deux QRS (resp. onde P) séparés par un QRS (resp. une onde P).

Les instants des ondes Q, S et T sont encore très difficiles à détecter automatiquement et ne seront donc pas utilisés dans cette étude. La régularité du rythme est mesurée en terme de variation de durée d'un intervalle RR à l'intervalle suivant.

La propagation des ondes électriques citées précédemment a pour conséquence la contraction et la décontraction du muscle cardiaque qui peut alors jouer son rôle fondamental, celui de pompe permettant d'assurer un flux de sang continu aux organes et aux tissus cellulaires du corps. Les veines cave et pulmonaire (cachée derrière la crosse aortique sur la Figure 1.7) communiquent respectivement avec les oreillettes droite et gauche tandis que l'aorte et l'artère pulmonaire (cf. tronc pulmonaire sur la Figure 1.7) communiquent respectivement avec les ventricules gauche et droit par l'intermédiaire des valves aortique et pulmonaire (cf. Figure 1.7). En outre, les oreillettes commu-

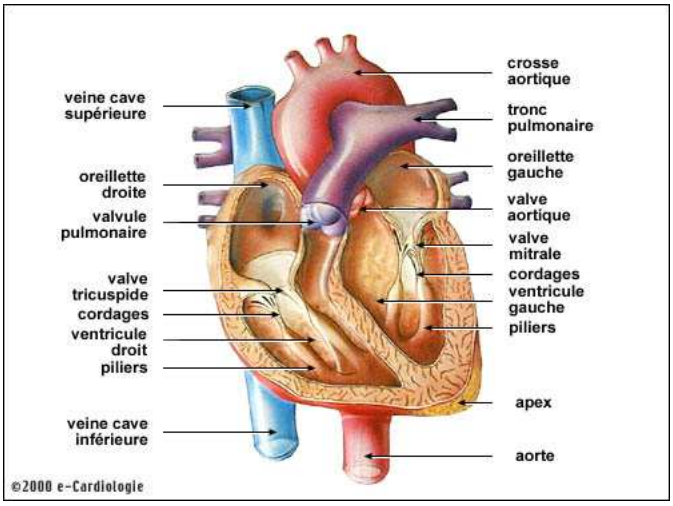


FIG. 1.7 – Anatomie du cœur

niquent avec les ventricules par l'intermédiaire des valves auriculo-ventriculaires (AV) mitrales et tricuspides.

Les événements mécaniques qui caractérisent la fonction de pompe du coeur peuvent être divisés en deux périodes : la *diastole* et la *systole* (cf. Figure 1.8). L'étape e correspond à la fin de la *diastole* : les valves AV sont ouvertes et les valves aortique et pulmonaire sont fermées. Le sang entre dans le cœur pendant toute la diastole remplissant les oreillettes et les ventricules.

L'étape représentée aux points a et f correspond à la *systole auriculaire*. Lorsque l'onde de dépolarisation se propage entre le nœud sinusal et le nœud auriculo-ventriculaire (correspondant à l'onde P sur l'ECG), les oreillettes se contractent et un flux de sang supplémentaire passe des oreillettes vers les ventricules (les ventricules sont déjà remplis à 70%). Le muscle auriculaire (autour des oreillettes), en se contractant, entoure les orifices entre les oreillettes et les veines cave et pulmonaire empêchant ainsi une éjection de sang vers les veines.

L'étape b correspond au début de la *systole ventriculaire*. Les ventricules se contractent suite au passage de l'onde de dépolarisation (correspondant au complexe QRS sur l'ECG) et la pression à l'intérieur des ventricules augmente entraînant la fermeture des valves AV qui empêchent la remontée du sang des ventricules vers les oreillettes. Pendant environ 50 ms, la pression n'est pas suffisante pour ouvrir les valves aortique et pulmonaire, cette période est appelée *contraction isovolumique ventriculaire* puisque la pression de chaque coté des valves est identique. Pendant cette brève période, les valves AV sont distendues vers les oreillettes et la pression dans les oreillettes augmente légèrement. Quand la pression à l'intérieur des ventricules est maximale (pic connu comme la *pression artérielle systolique*), les valves aortique et pulmonaire s'ouvrent et le sang s'échappe vers les artères (cf. étape c). À la fin de la systole ventriculaire, il reste environ 50 ml de sang dans les ventricules.

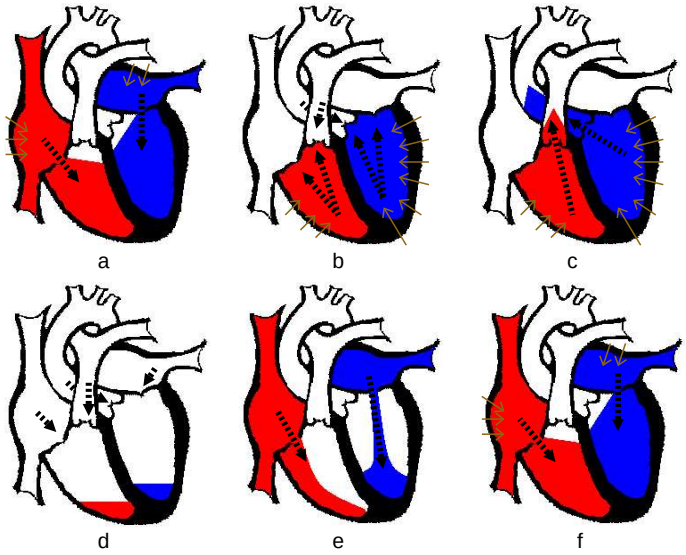


FIG. 1.8 – Activité mécanique cardiaque

L'étape d correspond au début de la diastole, les ventricules commencent à se relâcher, la pression ventriculaire chute et provoque la fermeture des valves aortique et pulmonaire. Après la fermeture des valves, la pression continue de diminuer avec le relâchement des ventricules jusqu'à descendre plus bas que la pression dans les oreillettes. La différence de pression provoque une nouvelle ouverture des valves AV permettant le remplissage des ventricules.

Les mesures hémodynamiques et leurs liens avec l'ECG Les différentes phases du rythme cardiaque provoquent des changements dans les variables hémodynamiques du système cardio-vasculaire qui peuvent être mesurées de manière invasive ou non invasive pour caractériser quantitativement l'activité mécanique du coeur. La figure 1.9 montre l'évolution de ces valeurs pour des mesures de pression aortique, auriculaire ou ventriculaire. Dans la suite de cette étude, nous utiliserons principalement des mesures invasives de pressions artérielles dont les courbes standards sont proches de celles des courbes de pression ventriculaire.

Dans un souci de modélisation qualitative, dans la suite de ce document, nous appellerons *diastole* le point de pression ventriculaire le plus bas et *systole*, le pic de pression artérielle systolique.

Les attributs choisis pour décrire la courbe de pression artérielle sont représentés Figure 1.10. Ce sont :

- l'intervalle de temps dd (ou $dd1$) qui sépare deux diastoles consécutives,
- l'intervalle de temps ss (ou $ss1$) qui sépare deux systoles consécutives,
- les intervalles de temps $dd2$ et $ss2$ qui séparent deux diastoles (resp. deux systoles) séparées par une diastole (resp. une systole),

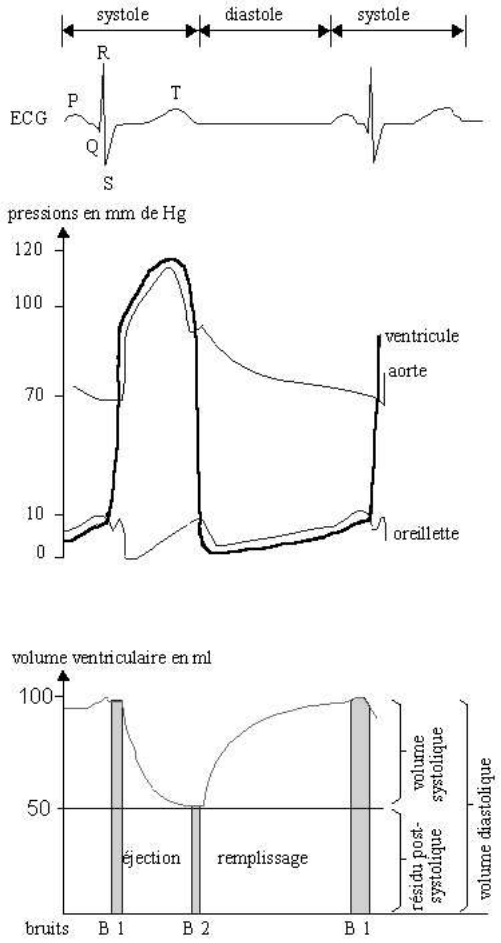


FIG. 1.9 – ECG et mesures hémodynamiques

- l'intervalle de temps ds (ou $ds1$) qui sépare une diastole et une systole consécutives,
- l'intervalle de temps sd (ou $sd1$) qui sépare une systole et une diastole consécutives,
- la différence d'amplitude amp_sd entre une systole et la diastole qui la précède directement,
- la différence d'amplitude amp_ds entre une diastole et la systole qui la précède directement,
- la différence d'amplitude amp_ss entre deux systoles consécutives,
- la différence d'amplitude amp_dd entre deux diastoles consécutives.

En présence de l'ECG, les cardiologues ne se servent des mesures hémodynamiques que pour confirmer leur diagnostic ou pour évaluer la gravité d'une pathologie. La Figure 1.9 montre une corrélation forte entre les instants d'apparition des ondes P et QRS et

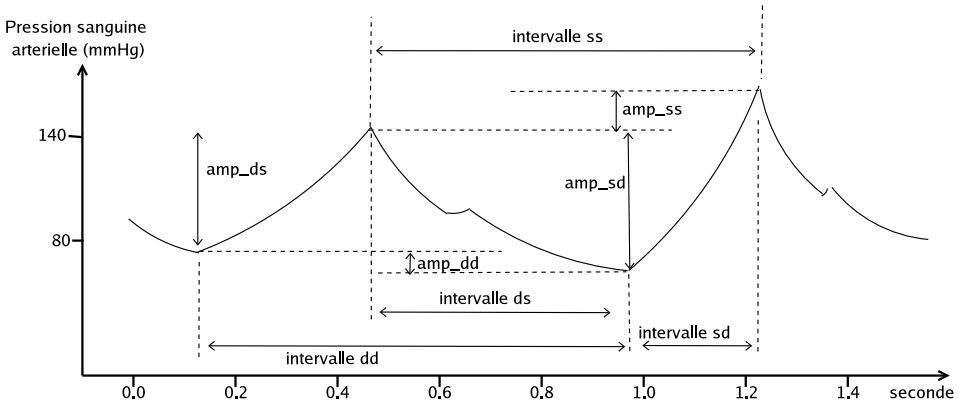


FIG. 1.10 – Deux cycles représentant l'évolution de la pression artérielle

les instants d'apparition de la systole et de la diastole. La principale corrélation mesurée effectivement par les médecins est le *délai électromécanique*, il correspond à la différence de temps entre l'instant d'apparition du complexe QRS et l'instant d'apparition du pic systolique. Ce délai est mesuré entre 200 et 300 ms lorsque le rythme cardiaque est normal.

1.1.3.3 Les arythmies cardiaques

Un ECG *normal* est une succession de battements normaux comme ceux de la Figure 1.6.

Comme décrit dans la partie précédente, une des particularités du cœur est de générer lui-même son rythme. Un rythme normal est un rythme sinusal (noté *rs*) régulier dont la fréquence est comprise, chez l'adulte normal, entre 60 et 100 cycles par minute et dont la régularité des intervalles PP (mesurée en terme de variation de durée d'un intervalle PP à l'intervalle suivant) ne doit pas excéder 0,16 s. Dans le cas normal, l'onde électrique se propage à travers les chemins de conductions décrits dans le paragraphe précédent. Lorsqu'un dysfonctionnement survient au moment de l'impulsion électrique dans le nœud sinusal ou lors de la conduction électrique, on peut diagnostiquer une *arythmie*.

Il existe deux types principaux d'arythmies : les troubles de la conduction intra-cardiaque et ceux du rythme. Il n'y a pas de délimitation précise entre ces deux types d'arythmies, non seulement parce que les troubles conductifs majeurs s'accompagnent fréquemment d'anomalies rythmiques, mais aussi parce que le mécanisme des troubles du rythme comportent souvent, à l'origine, des désordres localisés de la conduction. Les paragraphes suivants introduisent succinctement les arythmies liées à ces deux dysfonctionnements pour donner les connaissances essentielles à la compréhension du document.

Dans cette étude, nous nous intéressons principalement à neuf arythmies : dans la section 2.4 nous analyserons les performances de systèmes de monitoring sur les aryth-

mies *mobitz de type II*, *bloc de branche gauche* et sur le rythme normal ou *rythme sinusal*. Puis, dans la suite du document, nous chercherons à caractériser une *extrasystole ventriculaire*, un *doublet ventriculaire*, un *bigéminisme*, une *tachycardie ventriculaire*, une *tachycardie supra-ventriculaire* et une *fibrillation auriculaire* à partir de données provenant de différentes voies d'un électrocardiogramme et de mesures de pression artérielle. Les descriptions systématiques des aspects électrocardiographiques et hémodynamiques de arythmies sont présentées dans l'annexe A de ce document.

Les troubles de la conduction intracardiaque La conduction intracardiaque provient d'une propriété commune à toutes les cellules du myocarde qui leur permet de propager de proche en proche l'onde électrique en provenance du nœud sinusal. Les troubles de conduction résultent soit d'obstacles organiques ou fonctionnels capables d'arrêter ou de ralentir la progression de l'excitation cardiaque dans un secteur quelconque du myocarde, soit de cheminements anormaux de la conduction auriculo-ventriculaires. Dans ce cas, l'onde électrique emprunte des voies accessoires conduisant à des phénomènes de pré-excitation ventriculaire. Les arythmies sont provoquées par des troubles localisés de la conduction :

- entre le nœud sinusal et le myocarde auriculaire (muscle cardiaque auriculaire) commun, le trouble de la conduction est nommée *bloc sino-auriculaire*;
- dans la paroi des oreillettes, nommé *bloc intra-auriculaire* ou *inter-auriculaire*;
- à la jonction oreillettes/ventricules, c'est à dire dans le noeud de Tawara ou le faisceau de His, nommé *bloc auriculo-ventriculaire*. Le *mobitz* est une des arythmies causée par un bloc auriculo-ventriculaire
- dans l'une ou l'autre des branches de division du faisceau de His, ou dans une subdivision de celles-ci, nommé *blocs de branche*. Les arythmies causées par ce type de bloc sont nommées *bloc de branche gauche* (noté dans la suite *lbbb* pour *left bundle branch block*) quand le trouble de conduction est situé dans la branche gauche et *bloc de branche droit* (noté dans la suite *rbbb*) quand le trouble est situé dans la branche droite;
- dans la paroi des ventricules, nommé *bloc intra-ventriculaire pariétal*.

Les arythmies causées par les blocs de branches sont les plus fréquentes et ont les plus riches expressions électrocardiographiques. Les arythmies causées par les blocs auriculo-ventriculaires peuvent être distinguées plus précisément en 3 degrés différents par les rapports entre les ondes P et les complexes QRS. Le bloc responsable du *mobitz de type II* présente la caractéristique de bloquer la propagation du complexe QRS un battement sur deux (cf. Annexe A pour la description précise des signatures électriques).

Les troubles du rythme cardiaque Les troubles du rythme cardiaque peuvent résulter soit d'un dérèglement sinusal, soit de l'intervention d'autres centres de commande (d'où partent les impulsions électriques) ou de processus pathologiques divers entravant la propagation des ondes électriques. En général, l'apparition d'un trouble rythmique provient de désordres électrophysiologiques cellulaires entraînant un dérèglement plus ou moins localisé des grandes propriétés du muscle cardiaque : l'automaticité, l'excitabilité (la capacité à émettre les impulsions électrique) et la conductivité. Les mécanismes

intervenant, isolément ou en association (dans le cas d'*arythmies multiples*), dans la genèse des arythmies sont :

- L'anomalie de la commande sinusale : l'activité cardiaque demeure sous la commande du nœud sinusal mais celle-ci s'écarte des normes de fréquence et de régularité, pouvant aller jusqu'à un arrêt occasionnel ou prolongé de l'activité sinusale.
- Les commandes non sinusales ou ectopiques : les centres de commandes ectopiques peuvent siéger dans le myocarde auriculaire, dans le myocarde ventriculaire, ou à la jonction auriculo-ventriculaire. Lorsque le centre ectopique active le coeur pour un seul battement, on désigne le battement d'origine ectopique par le terme d'extrasystole s'il est prématuré par rapport au rythme de base, d'échappement s'il survient au contraire tardivement à la suite d'un ralentissement ou d'une défaillance du rythme de base. Au contraire, lorsque le centre ectopique prend la commande pour une durée plus ou moins longue, on parle de rythme ectopique. Parmi les rythmes ectopiques, on oppose les rythmes passifs, lents, instaurés par un mécanisme d'échappement et les rythmes actifs, rapides, prenant la commande grâce à une accélération importante de la fréquence naturelle. On donne le nom de tachycardies aux rythmes ectopiques actifs et rapides. Suivant la localisation (auriculaire, ventriculaire, jonctionnelle) du nouveau centre de commande, on trouvera ainsi des *tachycardies ventriculaires*, des *tachycardies supra-ventriculaires* (auriculaires ou jonctionnelles) ou des *extrasystoles ventriculaires*. Lorsque deux extrasystoles ventriculaires se produisent consécutivement, on parle de *doublet ventriculaire*. Lorsque le rythme ectopique est régulier, on peut diagnostiquer des arythmies comme le *bigéminisme*.
- La compétition multisite : un centre de commande ectopique peut partager la totalité de l'activation cardiaque avec le nœud sinusal ou avec un second centre ectopique. L'activité électrique dans les oreillettes peut ainsi devenir complètement indépendante de celle des ventricules.
- La fibrillation et le flutter : le flutter et la fibrillation sont des situations où les oreillettes ou les ventricules sont soumis à des impulsions ectopiques de fréquence très élevée soit régulières (flutter) soit irrégulières et anarchiques (fibrillation), déterminant une activité électrique ondulante et permanente sans période de repos global. Lorsque le foyer ectopique se situe dans une des oreillettes, on parle de *fibrillation auriculaire*.

1.1.4 Le diagnostic des arythmies cardiaques en USIC

Comme explicitée dans la définition générale du *monitoring* donné par le groupe ALARM présentée au début de ce chapitre, les appareils de monitoring sont équipés d'un certain nombre d'alarmes permettant de prévenir le personnel médical en cas de changement important de l'état d'un patient monitoré. Les alarmes sont en général déclenchées lorsqu'un des paramètres monitorés (le rythme cardiaque, la pression artérielle etc.) atteint un seuil jugé critique par le personnel médical. Compte tenu de la gravité de l'état des patients admis en USI (ou en USIC), les seuils d'activation de ces alarmes sont très bas.

Le personnel médical doit ainsi faire face à plusieurs centaines d'alarmes journalières produites par les appareils de monitoring. Seules un très petit nombre de ces alarmes (10% en USI) nécessitent une réelle modification de la thérapie appliquée au patient. Les alarmes déclenchées alors qu'elles n'ont pas de réelle signification clinique sont appelées *fausses alarmes*. Une étude [Tsien et Fackler, 1997] a ainsi montré que sur 10 semaines d'enregistrement en USI, 86% des 2942 alarmes déclenchées étaient des fausses alarmes. Ces fausses alarmes sonores ont des conséquences très néfastes sur le personnel médical et les patients puisqu'elles peuvent entraîner de graves troubles du sommeil et laissent les occupants des USI dans un état de stress permanent. Les seuils de déclenchement des alarmes utilisés dans des appareils de monitoring instaurés dans les USI permettent cependant de détecter la totalité des situations véritablement dangereuses, rendant l'utilisation de ces appareils indispensables.

De nombreuses études ont été conduites pour tenter de diminuer le nombre de fausses alarmes en USI. Dans [Chambrin, 2001], M.-C. Chambrin propose diverses méthodes dont : la classification des alarmes suivant leur niveau de gravité (conjointement à une spécification médicale standardisée des seuils associés à ces niveaux de gravité); l'utilisation de techniques plus élaborées que le simple passage d'un seuil critique pour le déclenchement d'une alarme; l'utilisation de plusieurs sources de données pour affiner ou confirmer le diagnostic.

En USIC, les niveaux d'alarmes dépendant de la gravité des arythmies ont d'ores et déjà été hiérarchisés. Les *asystolies* (arrêt de l'activité ventriculaire pendant plus de 4 secondes), les tachycardies et fibrillations ventriculaires ainsi que les bradycardies extrêmes (réduction importante du rythme sinusale) sont considérées comme des *alarmes rouges* qui nécessitent une réponse immédiate du personnel médical. Les *doublets ventriculaires*, les extrasystoles ventriculaires prématurées (notées *PCV* pour *Premature Ventricular Contraction*), les *bigéminismes*, les *tachycardies supra-ventriculaires* sont des *alarmes oranges* indiquant une situation dangereuse demandant une réponse rapide du personnel médical. Les *fibrillations auriculaires* en revanche, peuvent être bien tolérées par les patients et déclenchent donc des alarmes de plus basse priorité. Il est cependant difficile pour les moniteurs actuels de différencier une fibrillation supra-ventriculaire d'une fibrillation ventriculaire lorsque l'onde P n'est pas bien détectée, ce type d'ambiguïté sur les arythmies déclenche donc beaucoup de fausses alarmes. Les techniques de traitement du signal évoluent très rapidement pour détecter et diagnostiquer ces arythmies sur des bases plus précises que de simples changements de seuil dans les paramètres monitorés. Des travaux de recherche importants restent cependant à effectuer pour améliorer la spécificité des détections d'arythmies en USIC et ainsi diminuer le nombre de fausses alarmes, notamment en combinant plusieurs sources de données.

1.2 Les systèmes de monitoring médical intelligents

Après la percée de technologies à base de systèmes experts à la fin des années 70 (MYCIN [Shortliffe, 1976], COMPAS [Sittig *et al.*, 1989]) et par la suite des systèmes

à base de connaissances, l'intelligence artificielle a pris une grande place dans la médecine en général et dans les systèmes de monitoring en particulier. En effet, dans un contexte d'USI par exemple, la quantité de données reflétant l'état d'un patient dépasse de beaucoup la capacité de synthèse et d'assimilation du personnel soignant. Pour éviter au personnel médical d'être submergé par les données au risque de manquer certains événements importants, les unités de soins intensifs sont équipées de *systèmes de monitoring intelligents*. En théorie, ces systèmes complexes ont pour but de collecter les données en temps réel, de les traiter, de les interpréter, de faire un premier diagnostic de la situation, de prévoir l'évolution du système surveillé, de proposer des actions voire même de les exécuter (cas des systèmes en boucle fermée). En pratique, très peu de systèmes utilisés en milieu clinique disposent de toutes ces facultés. En cardiologie par exemple, le diagnostic est limité à certaines arythmies dangereuses et les systèmes n'effectuent aucune action spontanément.

Coiera présente dans [Coiera, 1993] les étapes de la construction d'un système de monitoring intelligent. La première étape, au niveau du traitement du signal, consiste en le développement de nouveaux capteurs ou en la sélection des capteurs permettant de donner l'information la plus pertinente. La seconde étape consiste en la validation du signal fourni par les capteurs. L'auteur propose en effet d'inclure un système de gestion des alarmes intelligent qui validerait le signal avant de générer une alarme et ainsi éviterait le problème de l'émission des fausses alarmes en cas de signal trop bruité. La troisième étape est la reconnaissance de motifs caractéristiques dans le signal suivie éventuellement d'une phase de transformation des données en données qualitatives. Dans le milieu médical, cette phase est encore appelée phase d'*abstraction temporelle* car beaucoup de données manipulées font référence au temps et les symptômes d'un patient évoluent eux-mêmes au cours du temps. La dernière phase est une phase d'inférence conduisant au but que se donne un système de monitoring intelligent. Nous verrons dans la suite de cet état de l'art que la plupart des systèmes conçus actuellement ne prennent pas en compte toutes ces phases de construction et ne sont donc, pour la plupart, pas viables en milieu clinique.

Nous différencions deux types de systèmes de monitoring intelligent suivant qu'ils prennent en compte une partie acquisition automatique de la connaissance ou non.

1.2.1 Des systèmes de monitoring médical

Dans [Uckun, 1993], Uckun présente un état de l'art sur les systèmes de monitoring médical intelligents existant en 1992. Ces systèmes ont pour but de remédier aux défauts des systèmes de monitoring classiques sur diverses tâches telles que la gestion en temps réel d'une quantité importante de données (par exemple, par des méthodes de perception sélective consistant à se focaliser sur les données les plus critiques), la manipulation de données bruitées (en utilisant, par exemple, plusieurs sources de données), la prise en compte de l'expérience médicale sous forme de connaissances pour aider le diagnostic et la diminution des fausses alarmes [Chambrin, 2001]. Aucun de ces systèmes n'a fait l'objet d'une réelle évaluation ni d'une utilisation clinique poussée. Parmi les systèmes référencés par cet état de l'art, on peut citer GUARDIAN [Larsson et Hayes-Roth, 1998],

GANESH et son successeur NÉOGANESH [Dojat *et al.*, 1997] ou le système VIE-VENT [Miksch *et al.*, 1996] de Miksch et al; ces deux derniers sont de rares exemples de systèmes de monitoring ayant été testés en milieu clinique.

GUARDIAN est un système de monitoring à base de connaissances dont le but est d'assurer les tâches de monitoring et de diagnostic pour des patients venant de subir une intervention chirurgicale cardiaque. Ce système a été développé sur de nombreuses années (de 87 à 98) et inclut un simulateur de scénarios, plusieurs algorithmes de diagnostic et un algorithme de planification de tâches décrits dans un langage permettant d'étendre les fonctionnalités décrite précédemment. La base de connaissances expertes contient un très grand nombre de règles concernant jusqu'à 61 pathologies différentes couvrant les cas les plus communs de complications postopératoires liées aux interventions chirurgicales cardiaques. Il n'a pas été testé en milieu clinique mais une évaluation du système a été conduite en comparant les réactions de GUARDIAN à des scénarios cliniques critiques à celles de personnels hospitaliers sur les mêmes scénarios dans des exercices de simulation. Les résultats prouvent que des systèmes tels que GUARDIAN peuvent s'avérer plus efficaces et naturellement moins sensibles au stress ou à la surcharge d'événements pour évaluer l'état du patient et proposer des soins lors de situations critiques.

VIE-VENT est un système de monitoring à base de connaissances fonctionnant en boucle ouverte et dédié à la surveillance et à la planification d'actions pour la respiration artificielle des nouveaux-nés en USI. Le système est, en particulier, équipé d'un module d'abstraction temporelle perfectionné qui permet d'unifier les représentations de données quantitatives (par exemple les paramètres du ventilateur) ou qualitatives (sexe du patient, connaissances expertes, etc.) estimées de manière continue ou discontinue (cas des mesures faites à la demande). Ce module générique est réutilisable dans de nouvelles applications. VIE-VENT donne de bons résultats en milieu clinique lorsque les données sont fiables (peu bruitées). Il est adapté aux domaines dans lesquels les connaissances sont incomplètes car il permet, grâce aux possibilités du module d'abstraction temporelle, d'acquérir de nouvelles connaissances à la volée. Il n'a, en revanche, pas été conçu pour la tâche spécifique du diagnostic.

NÉOGANESH est également un système de monitoring à base de connaissances qui contrôle, cette fois en boucle fermée, l'assistance respiratoire mécanique de patients hospitalisés en USI. Ce système fonctionne avec une architecture distribuée de type multi-agent qui contient un module d'acquisition de données, un module de classification de l'état respiratoire du patient, un module d'abstraction temporelle et un module de planification d'actions. Le module d'abstraction temporelle dont les mécanismes principaux sont l'agrégation d'événements et l'oubli de certains événements non critiques, est décrit en détail dans [Dojat et Chittaro, 1997]. Le diagnostic et la planification des actions sont basés sur plusieurs bases de connaissances orientées-objet qui contiennent la description des objets qui composent l'environnement et des règles permettant d'effectuer des opérations sur ces objets. Pour des raisons de sécurité et d'éthique, l'utilisation du système de monitoring en boucle fermée est limitée. Il n'est, en particulier, dédié qu'à un mode respiratoire particulier.

Plus récemment, le système DÉJÀ VU de Dojat et al. [Dojat *et al.*, 1998] per-

met de reconnaître des scénarios médicaux critiques à partir d'une base de scénarios décrits de manière symbolique par des experts. Proches des chroniques de Dousson [Dousson, 1996] (cf. section 2.2), ces scénarios sont des réseaux de contraintes qui permettent de garder une part d'incertitude sur l'instant d'apparition des événements et ainsi permettre non seulement une reconnaissance des scénarios en ligne mais également une reconnaissance *a posteriori*.

D'autres systèmes, en plus des traitements décrits en introduction, ont proposé des solutions au problème de l'acquisition automatique des connaissances. En effet, les connaissances médicales nécessaires au bon fonctionnement des systèmes de monitoring intelligents sont très coûteuses à obtenir en terme de *temps expert* et leur acquisition est bien souvent un frein au développement de tels systèmes.

1.2.2 Acquisition des connaissances dans les systèmes de monitoring

Cette section décrit différents systèmes de monitoring médical intelligents prenant en compte un module d'acquisition automatique des connaissances. Deux de ces systèmes, KARDIO et CALICOT, seront détaillés dans le chapitre 2.

Parmi les systèmes de monitoring médical intelligents à base de connaissances, on peut différencier ceux qui se servent de connaissances profondes, par exemple la connaissance physiologique du cœur ou celle des mécanismes de propagation des ondes électriques, comme KARDIO, CARMEN [Hernández *et al.*, 2000] ou son projet précurseur : CARDIOLAB [Siregar *et al.*, 1995]; et ceux qui se servent de connaissances superficielles (qui relient directement les observations du patient aux conclusions) comme CALICOT, le système de Morik et al. [Morik *et al.*, 2000] ou celui de Duchêne *et al.* [Duchêne, 2004]. KARDIO et CARDIOLAB ne sont pas à proprement parler des systèmes de monitoring, mais peuvent servir de module d'acquisition de connaissances pour un système de monitoring intelligent.

La version originale de CARDIOLAB [Le Moulec, 1991] est un simulateur d'ECG basé sur un modèle qualitatif du système de conduction électrique cardiaque, implémenté comme un ensemble de règles du premier ordre. Différents rythmes peuvent être générés à partir de listes prédéfinies de paramètres physiologiques statiques, en liaison avec les structures internes du cœur. Le but du modèle à base de connaissances profondes CARDIOLAB était de permettre au clinicien d'affiner son diagnostic et de déterminer les thérapies adaptées en cas de situations nouvelles ou de situations dans lesquelles la solution au problème n'est pas évidente. CARDIOLAB a ensuite été amélioré [Siregar *et al.*, 1995] pour fournir un niveau de représentation cellulaire plus détaillé donnant des informations quantitatives précises. Cependant, l'interface entre CARDIOLAB et les signaux ECG observés, c'est-à-dire l'intégration des différentes améliorations de CARDIOLAB dans un système à base de modèle pour l'interprétation du rythme cardiaque, n'a jamais été développée, en partie à cause de la représentation statique de l'activité cellulaire et du nombre important de paramètres à adapter. Ce pas a été franchi avec le modèle cardiaque CARMEN.

CARMEN est également un modèle cardiaque de niveau macroscopique semi-quantitatif qui permet de synthétiser des signaux ECG mais aussi de générer

des interprétations directes de ces ECG par le biais de diagrammes de Lewis [Hernández *et al.*, 2002] (employés habituellement par les médecins dans l'approche analytique d'interprétation des arythmies cardiaques). Différents troubles du rythme cardiaque peuvent ainsi être simulés en sélectionnant manuellement un ensemble approprié de paramètres du modèle ou en faisant automatiquement varier les paramètres du modèle pour que l'ECG simulé concorde avec un signal ECG réel. En utilisant cette deuxième fonctionnalité [Hernández et Carrault, 2004], CARMEN peut donc être utilisé pour le monitoring des arythmies cardiaques. Notons que CARMEN peut aussi générer des représentations symboliques des ondes ECG synthétisées en décrivant par exemple leurs instants d'occurrence, leurs morphologies ou leurs relations avec les ondes précédentes. Un exemple d'utilisation concrète de cette fonctionnalité est donné dans le chapitre 2.

Dans [Morik *et al.*, 2000], les auteurs proposent un système de monitoring en ligne du signal hémodynamique proposant des recommandations d'actions au personnel clinique des USI. Pour pallier le coût prohibitif en temps de l'acquisition des bases de connaissance, les auteurs proposent de combiner des méthodes statistiques à base de Séparateurs à Vastes Marges (SVM ou *Support Vector Machine*) [Joachims, 1998] pour déterminer les actions à appliquer (il s'agit d'augmenter ou de diminuer l'injection de médicaments). Puis, des méthodes d'apprentissage inductif basées sur le système MOBAL [Morik *et al.*, 1993] sont utilisées pour apprendre des règles de recommandation à partir des actions possibles et de la connaissance médicale fournie par les experts. Le système proposé comporte également une partie traitement du signal permettant de transformer les données de pression provenant des capteurs en tendances symboliques du signal (par exemple, de t0 à t1, la pression augmente).

Dans un autre contexte, Duchêne présente dans [Duchêne, 2004] ses travaux sur la télésurveillance médicale à domicile. Le but est de détecter et de prévenir l'occurrence de situations critiques d'une personne à domicile impliquant la transmission de messages d'alarmes au personnel médical distant prêt à intervenir en cas de nécessité. Pour effectuer ces détections, l'auteur recherche à acquérir automatiquement des motifs représentatifs de comportements humains (les activités de la vie quotidienne d'une personne à domicile) à partir de données hétérogènes (qualitatives ou quantitatives) enregistrées par un ensemble de capteurs. L'auteur propose tout d'abord une phase d'abstraction des données qui comprend une phase de pré-traitement des données (filtrage) et une phase d'extraction de caractères permettant de représenter la séquence temporelle obtenue après pré-traitement par une succession d'informations pertinentes au regard de la décision (un résumé des situations stationnaires observées), sur une ou plusieurs dimensions. Cette phase d'abstraction est suivie d'une phase de fouille de données qui comprend elle-même une phase de fouille de caractères permettant d'isoler des sous séquences de données représentatives et une phase de classification permettant de regrouper ces sous séquences en ensemble de motifs multidimensionnels représentatifs.

1.3 L'approche multisource

Le domaine de recherche qui propose des techniques permettant d'utiliser plusieurs sources de données est celui de la *fusion de données*. La fusion de données est définie par Bloch et al. [Bloch *et al.*, 2001] comme consistant à *réunir ou amalgamer des informations provenant de différentes sources de données dans le but d'exploiter cette union ou fusion d'informations pour différentes tâches telles que: répondre à des questions, prendre des décisions, faire des estimations numériques, etc.*. L'intérêt de fusionner plusieurs sources de données est d'obtenir des informations plus précises (en bénéficiant de la *redondance* des sources), qui concernent des attributs impossibles à percevoir ou dont la fiabilité est incertaine avec un seul capteur (en bénéficiant de la *complémentarité* des sources), en des temps et des coûts réduits (en parallélisant les calculs).

Après une présentation des objectifs de la fusion de données, nous donnerons les principales caractéristiques des techniques de fusion présentées dans la littérature et nous les illustrons, quand cela est possible, par des exemples empruntés au domaine médical en général et à la cardiologie en particulier.

1.3.1 Objectifs de la fusion de données

Les objectifs de la fusion de données sont divers. Trois tâches principales sont évoquées dans [Bloch *et al.*, 2001] :

1. Raffiner ou accroître nos connaissances, informations ou croyances sur le monde réel;
2. Construire un point de vue global sur un modèle du monde;
3. Mettre à jour, réviser ou rafraîchir l'information sur le monde réel ou sur un modèle du monde.

Le monde réel est généralement un objet ou une collection d'objets décrits par un ensemble d'attributs de nature et complexité variable (fréquences, positions, vélocités, taux, classes, scénarios). Les objets peuvent être dynamiques, i.e. inclure une dimension temporelle et sont, dans ce cas, appelés *événements*.

Le premier objectif présenté ci-dessus regroupe les applications dont le but est de reconnaître ou de classer des *objets* du monde réel en se servant de plusieurs sources de données. Les informations sont dites *complémentaires* lorsque chaque capteur ne peut fournir qu'une partie des attributs décrivant le monde. Ces attributs sont en général indépendants des attributs perçus par les autres capteurs. La fusion doit alors tirer parti de la complémentarité des sources et en particulier de leur pouvoir discriminant relatif. Par exemple, chaque source de données peut mettre en valeur différentes caractéristiques des objets et une seule source n'est peut être pas suffisante pour fournir une classification précise et fiable d'objets presque similaires. Dans ce cas, lorsque l'on combine les sources, des informations incomplètes sur une source peuvent être complétées par les autres sources.

Ce premier objectif est celui de notre étude. Nous cherchons à reconnaître des arythmies cardiaques, c'est-à-dire des enchaînements d'objets temporels (des QRS, des

systoles, etc.) ayant des caractéristiques propres, à partir de plusieurs sources de données. Cet objectif est également celui de Duchêne *et al.* qui cherchent à reconnaître des séquences d'objets représentés par des vecteurs multidimensionnels caractéristiques d'un comportement humain. Ces vecteurs contiennent des informations sur l'activité du patient, la moyenne de sa fréquence cardiaque, sa posture (assis, couché, debout), etc.

Le second objectif présenté au début de cette section, consiste à rechercher un point de vue global, en prenant en compte différents points de vue sous forme de préférences individuelles, de règles consensuelles, d'obligation etc. Le problème n'est pas, ici, d'aboutir à une description plus fine ou plus précise du monde mais plutôt d'évaluer les alternatives possibles. Des informations *redondantes* peuvent ainsi être fournies par un capteur ou un groupe de capteurs lorsque chacun de ces capteurs perçoit, avec une fiabilité différente (cf. [Cholvy, 2003] pour des critères d'évaluation des sources de données), les mêmes attributs d'un environnement. La redondance permet non seulement d'améliorer la précision de l'information communiquée au système mais également sa fiabilité dans le cas d'erreurs ou de dysfonctionnements au niveau des capteurs.

Pour le troisième objectif, nous supposons disposer d'un modèle du monde réel à un instant donné (obtenu éventuellement par combinaison de sources), décrit en terme d'attributs instanciés. La problématique est d'ajouter de nouvelles informations disponibles sur le monde au modèle. Cette problématique est celle des systèmes de monitoring intelligents (Guardian, Vie-Vent, Neogannesh, Déjà Vu, le système de Morik *et al.*) décrits précédemment. Ces systèmes utilisent des données provenant de différents capteurs tels que la fréquence respiratoire, le volume sanguin, la fréquence cardiaque, des mesures de concentration de certains médicaments dans le sang, etc. pour définir un état global du patient. Les différents états possibles sont décrits par des attributs provenant des différentes sources. L'arrivée de nouvelles informations (le changement de la valeur ou de la tendance [Calvelo *et al.*, 2000] d'un paramètre) produit un changement d'état du système. Ce changement permet de fournir un diagnostic ou éventuellement, une liste des actions à appliquer pour faire à nouveau évoluer l'état courant. Notons que pour ce type de fonctionnement, il est nécessaire que la valeur des attributs permettant de décrire l'état du patient soit accessible à chaque instant. Ce n'est pas possible sur un signal analogique tel que l'ECG. Dans la littérature on trouve le terme *multi-paramétriques* pour designer plusieurs sources de données représentant la variation d'un paramètre précis au cours du temps (par exemple, la valeur du volume sanguin dans les ventricules).

1.3.2 Techniques de fusion de données

Les articles de Luo et Kay [Luo et Kay, 1989] et T. Garvey [Garvey, 1990] offrent une revue de différentes méthodes de fusion multicapteur (ou multisource) dans des systèmes intelligents mais également une revue de différentes méthodes d'intégration de ces capteurs ainsi que les problématiques liées à ces méthodes. Un système permettant de fusionner des données intègre au moins l'une ou la combinaison des trois fonctions suivantes : i) la sélection des capteurs permettant de choisir à un instant donné les

capteurs les plus intéressants pour arriver à un but donné, ii) une représentation du monde (i.e. de la situation actuelle) que l'on va chercher à mettre à jour en fusionnant des informations, et iii) des fonctions de transformation des données pour les faire coïncider avec la représentation du monde choisie.

Dans la chaîne de traitement de l'information liée au système de monitoring, les données passent d'un état paramétrique ou analogique (le signal), à un état d'information (les données numériques ou symboliques) et enfin à un état de connaissance (après apprentissage, par exemple). La fusion de données peut intervenir à chacune de ces différentes étapes : les signaux peuvent directement être fusionnés à la sortie des capteurs, on parle alors de *fusion de capteurs ou fusion de données brutes*, les données symboliques ou numériques peuvent elles mêmes être fusionnées avant traitement (on parle alors du problème de *fusion de données symboliques*), et l'on peut également fusionner des données quand elles se trouvent sous forme de connaissances (par exemple des règles de classification, de prédiction etc.).

Les données provenant des différentes sources peuvent être *homogènes* ou *hétérogènes*. Des données homogènes peuvent généralement être fusionnées directement au niveau des capteurs alors que les données hétérogènes le sont souvent après transformation en données symboliques ou fournies à différentes parties du système sans être amalgamées (cf. [Benferhat et Lang, 2001]). Dans les deux cas, la phase de fusion doit être précédée d'une phase *d'association des données* [Luo et Kay, 1989] permettant de s'assurer que les informations provenant des différents capteurs se réfèrent bien à la même situation. En particulier, [Cholvy et Hunter, 1997] présentent des méthodes de raisonnement sur des données inconsistantes et sur la désambiguïsation de ces données.

1.3.2.1 Fusion de capteurs

La fusion des informations *a priori*, c'est-à-dire directement au niveau des capteurs consiste à se ramener à un capteur *virtuel* dont les données seront fonction des mesures fournies par les différents capteurs. Le passage des données brutes vers la représentation symbolique puis vers la connaissance à proprement parler se fait de la même manière qu'avec un seul capteur. La fusion à ce niveau peut entraîner une perte d'information et une structuration indésirable des données.

A titre d'exemple, la technique de Gritzali [Gritzali, 1988] pour la détection du complexe QRS à partir de plusieurs voies d'un électrocardiogramme s'appuie sur ce type de fusion. L'énergie des ondes présentes sur les différentes voies est *sommée* pour former un nouveau signal contenant des complexes QRS de plus forte amplitude, plus faciles à traiter par des algorithmes de détection à base de seuils. Le nouveau signal créé n'est, par contre, pas utilisable pour des tâches comme la classification des QRS puisque la forme des nouveaux QRS n'a aucune signification clinique et on ne peut en particulier pas décider si le complexe est normal ou anormal au contraire des complexes isolés sur chacune des voies de l'ECG. En outre, la transformation effectuée n'est pas bijective (ce qui permettrait de reconstruire le signal initial) puisque des fusions de signaux différents peuvent donner le même résultat.

Dans [Duchêne, 2004], Duchêne présente également une technique de fusion de capteurs pour construire, à partir de plusieurs signaux, des vecteurs multisources décrivant des segments temporels de longueur variable dans lesquels l'état des différents paramètres (sur toutes les sources) est stationnaire au regard du niveau de décision. L'abstraction des données en de tels segments est faite grâce à une mesure de similarité entre les valeurs des différents paramètres et celles d'un vecteur de paramètres moyen. Cette abstraction entraîne une perte de précision des données mais est suffisante dans le cas exposé pour effectuer un raisonnement de haut niveau sur les données tel que la recherche de motifs caractéristiques d'un comportement humain.

.

1.3.2.2 Fusion de données symboliques

Cette section est consacrée à la fusion de données symboliques avant l'étape d'extraction de connaissances à partir de ces données. Ce type de fusion fait l'objet des travaux présentés dans ce mémoire.

L'approche multisource que nous proposons consiste à agréger les données symboliques issues d'une phase de traitement du signal sur différentes sources sans qu'il y ait de perte d'information (cette approche diffère donc de celle de Duchêne *et al.*). Ces données agrégées sont utilisées pour exhiber des relations inter-sources explicites grâce une méthode d'apprentissage symbolique. La méthode proposée est décrite en détail dans le chapitre 4. Elle suppose une représentation symbolique unifiée pour chacune des sources pour pouvoir *agréger* les données, et une détection des événements adaptée sur chacune des sources pour fournir cette représentation symbolique. L'ensemble des relations inter-sources possibles étant potentiellement très vaste, la tâche de l'algorithme d'apprentissage est difficile. Les calculs effectués peuvent néanmoins être simplifiés lorsque les sources sont corrélées (une partie de l'information est redondante) pour peu que l'on dispose de connaissances suffisantes pour réduire les données initiales.

Dans la suite de cette section, nous comparons notre approche aux autres approches de fusion symbolique à partir de données cardiaques.

Dans [Thoraval *et al.*, 1997], le but est de surveiller le rythme ventriculaire et en particulier de surmonter les problèmes liés à la détection, par les algorithmes de traitement du signal, des complexes QRS en USI (en présence de bruit) en utilisant le signal hémodynamique en plus de l'ECG. Les auteurs considèrent les signaux comme des processus stochastiques, successions de complexes QRS sur l'ECG et de pics systoliques sur la pression. Ils considèrent également le délai électromécanique entre ces deux ondes (cf. Section 1.1.3). L'idée est de détecter ces événements caractéristiques sur chacune des sources séparément en utilisant pour l'ECG des algorithmes de détection de QRS appropriés ([Pan et Tompkins, 1985, Gritzali, 1988]) et pour le signal de pression, un simple filtre dérivatif (on cherche à connaître un maximum local). Cette phase de détection permet d'obtenir des événements symboliques datés pour les deux sources. Lors d'une phase de mise en correspondance, les auteurs cherchent à trouver, pour chaque événement détecté sur une source, son pendant sur la seconde en utilisant

le délai électromécanique. Cette phase permet de produire pour chacune des sources une série de quadruplets formés des deux types d'événements quand le pendant a été trouvé et un symbole vide à la place de l'événement manquant sinon, et de l'instant d'occurrence de ces événements. Ces informations symboliques sont ensuite fusionnées par des opérateurs permettant de régler les conflits pouvant apparaître entre les sources pour obtenir les événements manquants sur chacune des sources. La fusion permet ainsi de diminuer le nombre de fausses alarmes (faux positifs) liés aux mauvaises détections de l'activité ventriculaire.

Au contraire des buts visés par notre approche, ce type de fusion ne permet pas de mettre en relation des événements se produisant sur les différentes sources. Ici, l'intérêt de la fusion n'est pas de produire de nouvelles connaissances mais de conforter les détections en tirant partie de la redondance des sources.

Les travaux d'A. Hernandez et al. [Hernández *et al.*, 1999] pour la détection de l'activité auriculaire et ventriculaire dans le but de diagnostiquer les arythmies cardiaques illustre également ce propos. Plusieurs types de capteurs sont utilisés : plusieurs voies d'un signal ECG combinées à un signal hémodynamique pour détecter l'activité ventriculaire, d'une part, et un EECG (électrocardiogramme mesuré au niveau de l'œsophage) et un ECG pour détecter l'activité auriculaire, d'autre part. L'électrode fournissant l'EECG est particulièrement adaptée pour détecter l'activité électrique auriculaire : l'onde A visualisée a une plus grande amplitude que l'onde P sur l'ECG et est facilement dissociable du complexe QRS étant donnée la très faible intensité de celui-ci. Pour détecter l'activité auriculaire par exemple, on utilise deux pré-processeurs basés l'un sur l'ECG et l'autre sur l'EECG. Dans chaque pré-processeur, on détecte grâce à des méthodes de traitement du signal élaborées l'onde P dans le cas du pré-processeur basé sur l'ECG et l'onde A dans le cas de celui basé sur l'EEG. La sortie des pré-processeurs est un chiffre de valeur -1 si aucune onde P ou A n'a été détectée et 1 sinon. La fusion des pré-processeurs consiste en la somme (pondérée si une source est plus fiable qu'une autre) des valeurs en sortie de ceux-ci.

Sharshar et al. [Sharshar *et al.*, 2005] proposent une chaîne de traitement de plusieurs signaux numériques acquis en USI pour apprendre des motifs caractéristiques d'événements cliniques. La méthode s'appuie sur deux niveaux de représentation symbolique. Le premier, au niveau d'une seule source de données, détecte les tendances du signal de type "augmente", "diminue", "reste stable" ou "est transitoire", en utilisant des techniques d'analyse de données sur plusieurs échelles d'échantillonage du signal. Le second, au niveau multisource, produit des mots décrivant un état global du système à un instant donné en utilisant les tendances provenant des signaux qui contribuent le plus au comportement global du système. Ces mots prenant en compte plusieurs sources de données sont ensuite analysés par un système d'apprentissage automatique pour trouver les motifs caractéristiques [Vilhelm *et al.*, 2000]. Cette approche de la fusion est proche des objectifs de notre étude puisque les mots introduits dans le système d'apprentissage sont une représentation du système dans sa globalité. Les résultats de l'apprentissage par le système THINK! n'ont cependant pas encore été publiés. Notons

que, comme pour les systèmes GUARDIAN, VIE-VENT, NEOGANNESH, DÉJÀ VU et le système de Morik *et al.*, les signaux utilisés sont *multi-paramétriques* et la phase de prétraitement des données effectuée source par source sur une échelle de temps différente pour chacune des sources ne permet pas d'obtenir des relations précises (temporelles ou spatiales) entre les événements se produisant sur chacune des sources.

1.3.2.3 Fusion de connaissances

Cette section traite de la fusion de données sous forme de connaissance. Il s'agit de combiner les résultats obtenus par les phases précédentes de traitement des données, par exemple la phase d'apprentissage sur des données symboliques, pour répondre à un des objectifs de la fusion définis en section 1.3.1. Les connaissances peuvent par exemple être décrites sous forme de règles de classification ou de règles de diagnostic. Des méthodes telles que la combinaison de classifieurs [Schapire, 2001, Wemmert et Gançarski, 2001] ou des méthodes de *co-training* [Blum et Mitchell, 1998] sont utilisées pour tirer parti de la présence de plusieurs sources afin d'affiner les connaissances acquises. Des méthodes de vote [Dubois *et al.*, 2001] permettant de pondérer les connaissances apprises en fonction du degré de fiabilité et de pertinence qu'on leur attache. Des règles de "bonne fusion" issues des connaissances sur le domaine [Bloch *et al.*, 2001] peuvent également être utilisées pour fusionner les données disponibles. Ce type de fusion semble plus adapté si les données fournies par les sources sont indépendantes. Il permet en outre de disposer de l'information extraite des autres sources en cas de dysfonctionnement de l'une d'elles.

Dans la suite nous développons la méthode de *co-training* qui utilise des résultats d'apprentissage sur différentes sources représentant des vues d'un même phénomène pour augmenter les performances des apprentissages sur chacune des sources de manière itérative. Le *co-training* est une méthode d'apprentissage semi-supervisée introduite par Blum et Mitchel [Blum et Mitchell, 1998]. Le terme *semi-supervisé* est employé lorsqu'une partie seulement des données d'apprentissage sont étiquetées, i.e. préalablement classées avec une étiquette correspondant au nom de la classe auxquelles elles appartiennent. La technique de *co-training*, proche de l'algorithme de *bootstrapping* de Yarowski [Yarowsky, 1995], permet d'utiliser un gros volume de données non étiquetées pour accroître les performances d'un algorithme d'apprentissage sur un petit volume de données étiquetées. L'idée est de se servir de la possibilité de représenter un concept sous plusieurs vues différentes pour augmenter les performances de classifieurs appris sur chacune des vues. L'exemple de l'article de Blum et Mitchel consiste à apprendre un classifieur de pages de "cours en ligne" à partir des mots contenus dans ces pages et des mots contenus dans les liens qui pointent sur ces pages. Les résultats montrent que la méthode de *co-training* permet de diminuer de manière significative le taux d'erreurs de classification commises par les classifieurs appris sur chacune des sources.

Les conditions d'utilisation du *co-training* sont :

– Chaque exemple doit être décomposable en deux vues différentes ($e = e_1 \times e_2$), chacune d'elles devant être suffisante pour obtenir un classifieur correcte ;

- La fonction cible (le concept que l'on cherche à discriminer) $f = (f_1, f_2)$ doit être compatible avec chaque vue : les exemples doivent être étiquetés de la même façon par la fonction cible correspondant à chacune des vues;
- Chaque attribut d'une vue doit être conditionnellement indépendant des attributs des autres vues : il ne doit pas y avoir de fonction déterministe permettant de passer de e_1 à e_2.

Si les conditions citées précédemment sont réunies, l'algorithme à appliquer est le suivant :

Algorithme 1 (co-training)
Soit L un ensemble d'exemples d'apprentissage étiquetés et U un ensemble d'exemples non étiquetés :

1. *Créer un nouvel ensemble U' en choisissant aléatoirement u exemples de U.*
2. *Répéter k fois :*
   - *Utiliser l'ensemble L pour apprendre un classifieur h_1 en ne se servant que de la partie e_1 des exemples e.*
   - *Utiliser l'ensemble L pour apprendre un classifieur h_2 en ne se servant que de la partie e_2 des exemples e.*
   - *Étiqueter p exemples positifs et n exemples négatifs de U' avec le classifieur h_1.*
   - *Étiqueter p exemples positifs et n exemples négatifs de U' avec le classifieur h_2.*
   - *Ajouter ces nouveaux exemples étiquetés à L.*
   - *Choisir aléatoirement $2p + 2n$ exemples de U pour compléter U'.*
3. *Choisir entre entre h_1 et h_2, le classifieur offrant les meilleures performances .*

L'intérêt du *co-training* repose sur la propriété 1. La notion de PAC-apprenabilité est définie dans [Valiant, 1984].

Propriété 1 *Si les données utilisées sont conditionnellement indépendantes et si le concept que l'on cherche à classifier est PAC-apprenable à partir d'un ensemble aléatoire d'exemples contenant éventuellement du bruit alors les performances d'un classifieur faible peuvent être améliorées jusqu'à obtenir une précision choisie arbitrairement élevée en utilisant des exemples non étiquetés par la méthode de* co-training.

Lorsque le classifieur faible (appris avec un petit nombre d'exemples étiquetés) étiquette les données, il introduit du bruit de manière aléatoire dans la nouvelle base d'exemples. Si les deux vues ne sont pas indépendantes l'ensemble de données étiquetées par un classifieur h_1 ajoutée à l'ensemble de données préexistant n'est pas considéré comme un ensemble aléatoire pour la vue h_2 et la propriété 1 ne peut pas être utilisée.

Si la théorie du *co-training* parait particulièrement intéressante pour améliorer les performances d'un apprentissage à partir de données provenant de plusieurs sources, elle est en pratique difficilement utilisable puisque la condition d'indépendance est rarement atteinte dans les données réelles. L'efficacité de la méthode a pourtant été avérée empiriquement dans de nombreux cas [Kiritchenko et Matwin, 2001, Denis *et al.*, 2003] notamment en combinant [Nigam et Ghani, 2000] cette méthode à

d'autres méthodes d'apprentissage semi-supervisées comme *Expectation Maximisation* [Dempster *et al.*, 1977].

Dans notre problématique multisource, un exemple d'arythmie peut être décrit sous différentes vues correspondant aux différentes sources de données, par exemple, l'ECG et la mesure de pression. L'apprentissage multisource par co-training n'est cependant pas adapté à notre problématique multisource puisque les règles apprises par *co-training* ne permettent pas d'exhiber des relations réellement multisources. Au point 3 de l'algorithme 1 de *co-training*, l'utilisateur choisit le classifieur qui offre les meilleures performances sur l'une des deux vues mais l'algorithme ne fournit pas de classifieurs qui utilisent les deux vues simultanément pour classer les nouveaux exemples.

1.4 Conclusion

Ce chapitre a présenté les bases nécessaires à la compréhension du contexte applicatif de notre étude ainsi qu'un bref état de l'art des systèmes de monitoring médicaux intelligents, et en particulier de ceux qui incluent un module d'acquisition automatique des connaissances. La dernière section traite de l'utilisation de plusieurs sources de données dans le but d'améliorer les performances des systèmes de monitoring intelligents.

On peut remarquer que la plupart des systèmes de monitoring multisource existants utilisent des sources de données reflétant l'évolution d'un paramètre précis au cours du temps. La chaîne de traitement des données multisources de ce type diffère d'une chaîne de traitement utilisant des données comme un électrocardiogramme qui ne reflète pas une mesure exploitable directement. L'objectif de l'utilisation des différentes sources n'est également pas le même dans les systèmes présentés et dans l'approche visée par notre étude. Nous cherchons à apprendre de nouvelles relations inter-sources pour bénéficier, quand cela est possible, de la complémentarité des sources. Les systèmes présentés dans ce chapitre utilisent les sources de données, soit pour modifier l'état d'un système, soit globalement, pour apprendre des séquences multidimensionnelles. Les méthodes proposées ne permettent donc pas de répondre à nos attentes. La méthode proposée pour résoudre ce problème est détaillée dans les chapitres 3 et 4.

La section suivante présente de manière détaillée deux systèmes de monitoring cardiaque existants : Calicot et Kardio, ainsi que la première contribution de ce travail : une méthodologie de comparaison de systèmes de monitoring cardiaque.

Chapitre 2

Calicot : un système de monitoring des arythmies cardiaques

Les travaux présentés dans cette thèse ont pour but d'améliorer le système de monitoring cardiaque CALICOT afin de le rendre plus précis et plus robuste au bruit en milieu hospitalier. Ce chapitre présente de manière détaillée le système CALICOT dans son état à la fin du projet PISE [Carrault *et al.*, 2003]. Après une brève description de ce projet et de son successeur CEPICA, qui a permis le financement de ces travaux, nous fournirons une comparaison de CALICOT avec KARDIO [Bratko *et al.*, 1989], un autre système de renom permettant de diagnostiquer des arythmies cardiaques. KARDIO est en effet l'un des premiers systèmes permettant l'interprétation de rythmes cardiaques à partir de règles obtenues par simulation d'un modèle cardiaque et apprentissage.

2.1 Les Projets RNTS

A travers le projet Européen KISS (*Knowledge-based Interactive Signal Monitoring System*) [Siregar *et al.*, 1989], le Laboratoire de Traitement du Signal et de l'Image (LTSI) a développé au début des années 90, une grande expérience dans le monitoring intelligent des arythmies cardiaques en USIC. Dans KISS apparaissaient les notions importantes de modèle de surface/modèle profond (cf. Section 1.2.2) et de leur ajustement mutuel. Le projet a également mis en évidence le besoin et l'importance d'acquérir des données multisources, de prendre en compte l'état et le contexte d'entrée du patient en USIC et enfin, la nécessité d'avoir un système capable de réagir et de se reconfigurer face aux multiples changements pouvant se produire en USIC. Tous ces objectifs très ambitieux n'ont pu être atteints dans KISS, mais ils constituent les racines de la collaboration entre l'équipe DREAM de l'IRISA et le LTSI. En effet, profitant des compétences de l'équipe DREAM dans le domaine de l'acquisition automatique de connaissances, les deux équipes ont lancé, en collaboration avec des médecins et des industriels, deux projets RNTS (*Réseau National des Technologies et de la Santé*) successifs financés par le

ministère de la recherche : Pise et Cepica.

2.1.1 Pise

L'action concertée incitative (ACI) Pise, initiée en 1999 a mis en relation i) un concepteur fabriquant de prothèses cardiaques ELA Medical (ELA), ii) une équipe clinique : le département de Cardiologie et Maladies vasculaires du Centre Hospitalier Universitaire de Rennes, et deux équipes de chercheurs universitaires : iii) le LTSI) et iv) l'équipe Dream de l'Irisa. Ce projet s'intégrait dans une stratégie à long terme visant à développer des prothèses cardiaques multisites multifonctions capables de gérer simultanément et en temps réel les problèmes hémodynamiques et rythmiques des patients souffrant de cardiomyopathie ou d'insuffisance cardiaque. Il s'articulait autour de deux sous-projets : 1) la conception de deux sondes pour la stimulation du ventricule gauche et 2) le développement de méthodes robustes pour la reconnaissance et la classification d'arythmies. 57 patients ont ainsi été implantés avec les nouvelles sondes avec des résultats très bons en terme de sécurité et de performances électriques. Le second objectif du projet a fait l'objet de la thèse de Feng Wang [Wang, 2002], collaboration entre le LTSI et l'équipe Dream. Ces travaux ont abouti au prototype de système de monitoring intelligent Calicot (voir section 2.2 pour plus de détails) qui permet de diagnostiquer en ligne des arythmies cardiaques.

Parallèlement à ces travaux, la société ELA a développé de nouveaux capteurs pour la mesure de la fréquence de stimulation cardiaque : un capteur d'évaluation de la ventilation minute par mesure d'impédance intra-thoracique, reflet direct de l'activité respiratoire du patient et de l'intensité de l'effort fourni par ce dernier; un capteur de mesure de l'accélération (accéléromètre) placé dans la prothèse cardiaque; un accéléromètre placé au bout d'une sonde ventriculaire appelé PEA (*Peak Endocardial Accelerometer*). Ces dernières mesures permettent, d'une part, la détection des périodes d'activité du patient et, d'autre part, la mesure de l'intensité de l'activité exercée.

Pour poursuivre les travaux amorcés dans le projet Pise et évaluer l'intérêt de ces nouveaux capteurs, ces mêmes partenaires ont initié le projet Cepica.

2.1.2 Cepica

Le projet Cepica (*Conception et Évaluation d'une Prothèse Implantable CArdiaque*) est un projet RNTS pré-compétitif qui a débuté en 2002 et se poursuivra jusqu'en juillet 2006. Le contexte médical de Cepica est le même que celui du projet Pise puisqu'il s'agit d'exploiter les fonctionnalités futures des prothèses cardiaques, et en particulier les mesures provenant des capteurs reflétant l'état hémodynamique et fonctionnel du patient tels que la ventilation minute ou l'accélération, pour optimiser les paramètres de stimulation des prothèses, évaluer la tolérance d'un patient face à un trouble du rythme et permettre un diagnostic plus précis afin de diminuer les risques des insuffisances cardiaques, en particulier les risques de mort subite. Le projet s'articule autour de quatre tâches : 1) la mise au point d'un prototype externe permettant le recueil de l'impédance intra-cardiaque et des mesures d'accélération et l'étude des cor-

rélations entre ces mesures et les paramètres physiologiques et hémodynamiques; 2) le traitement du signal provenant de ces capteurs, l'étude des possibilités d'acquisition automatique de connaissances à partir des données provenant des capteurs et l'intégration de ces nouvelles connaissances pour le diagnostic des arythmies cardiaques à partir de données multisources; 3) le développement d'une prothèse implantable (miniaturisation, augmentation de la longévité, couplage des capteurs d'impédance et d'accéléromètrie à des fonctions de stimulation cardiaque performante, etc.); 4) une pré-évaluation de prototypes externes de telles prothèses.

Le travail décrit dans ce document de thèse s'inscrit dans le point 2) cité précédemment. Il s'est focalisé sur les tâches de monitoring et de diagnostic. Cependant, l'acquisition de données multisources provenant de nouveaux capteurs n'est pas aisée puisqu'elle nécessite de faire subir au patient une anesthésie, une intervention chirurgicale pour la pose du pacemaker puis d'observer le patient sur de longues périodes afin d'être sûr de recueillir des enregistrements d'arythmies. Trop peu de données provenant du projet CEPICA sont pour le moment disponibles, notre étude a donc cherché à démontrer la faisabilité de l'apprentissage de séquences arythmiques multisources à partir d'une base de données reconnue. Les données multisources utilisées proviennent de la base MIMIC (*Multi-parameter Intelligent Monitoring for Intensive Care*) [Moody et Mark, 1996]. Ces données concernent des signaux, recueillis dans des conditions d'USIC, provenant de différentes voies d'un ECG (nous nous servirons en particulier de la voie I) et de signaux hémodynamiques concernant la mesure de la pression artérielle (ABP pour *Arterial Blood Pressure*). La réduction du nombre de fausses alarmes sur ces données est déjà un défi important mais nous espérons également pouvoir généraliser les techniques proposées pour caractériser les arythmies cardiaques sur ces signaux, aux données provenant de nouveaux capteurs tels que les mesures de ventilation ou d'accélération. Dans la suite de ce mémoire, nous nous consacrons donc à l'étude de la combinaison des signaux électriques et hémodynamiques pour améliorer la précision et la robustesse du diagnostic des arythmies cardiaques.

2.2 CALICOT

CALICOT est un système de monitoring intelligent avec acquisition automatique des connaissances ayant pour but la reconnaissance en ligne d'arythmies cardiaques à partir de données provenant d'un ECG.

La Figure 2.1 donne une vision globale de l'architecture de CALICOT à la fin du projet PISE (cf. Section 2.1). Cette architecture se compose de deux parties distinctes, l'une utilisée en ligne qui comprend un module d'abstraction temporelle et un module de reconnaissance de chroniques, et l'autre utilisée hors ligne qui comprend un module d'acquisition par apprentissage supervisé de motifs discriminants caractéristiques des arythmies.

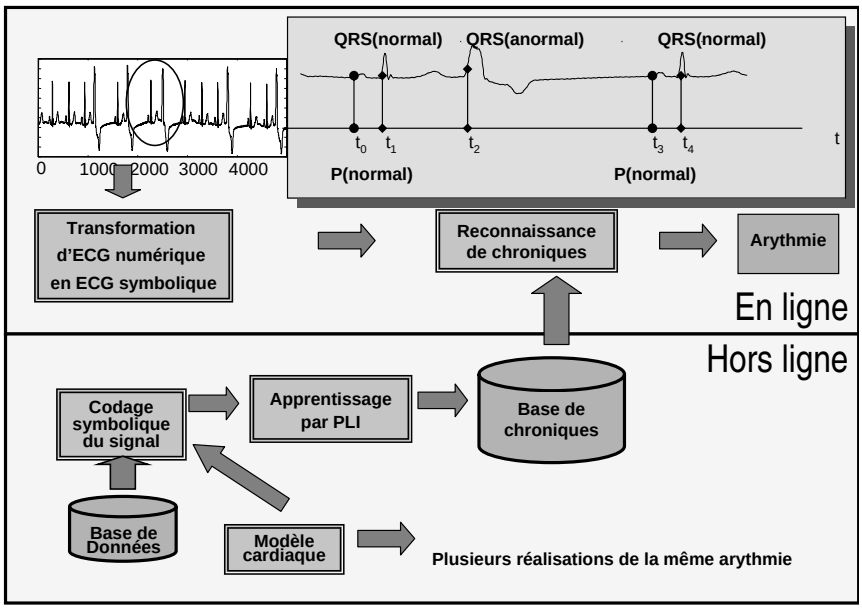


FIG. 2.1 – Architecture de CALICOT

2.2.1 Le diagnostic en ligne

Deux modules principaux entrent dans le processus de diagnostic en ligne des arythmies cardiaques dans le système CALICOT : le module d'abstraction temporelle et le module de reconnaissance de chroniques.

Abstraction temporelle Le module d'abstraction temporelle (cf. Figure 2.2) a été développé pendant la thèse de Feng Wang [Wang, 2002]. Il a pour but la détection et la classification (*normales* ou *anormales*) des ondes caractéristiques de l'ECG : l'onde P et le complexe QRS. La détection du QRS est principalement basée sur l'algorithme de Gritzali [Gritzali, 1988]. La classification des QRS se fonde sur la transformée en ondelettes des signaux suivie par une classification des motifs extraits par réseaux de neurones. La détection de l'onde P s'inspire des travaux d'Hernandez *et al.* [Hernández, 2000]. Elle utilise une technique d'annulation du complexe QRS et exploite les extrema de la transformée en ondelettes pour faciliter le repérage et la détection des ondes P [Senhadji *et al.*, 2002]. La détection se fait ensuite via un réseau de neurones qui permet de différencier les ondes réelles des artefacts du signal. En effet, la difficulté majeure liée à ces détections et classifications est la présence de bruits et d'artefacts sur le signal à analyser. Ces bruits peuvent être causés par les mouvements du patient, les défaillances du matériel médical ou des facteurs physiologiques tels que la respiration. Les résultats présentés sur des données réelles de la base MIT-BIH permettent d'obtenir des détecteurs/classifieurs tout à fait acceptables pour transformer ces ondes en données symboliques de type *évènements caractérisés et datés* (par

exemple, “un QRS anormal est détecté au temps 2567”).

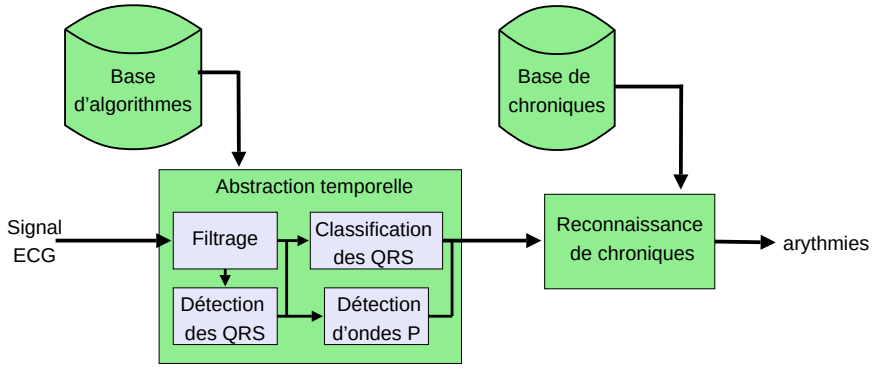


FIG. 2.2 – Abstraction temporelle dans CALICOT

Raisonnement temporel Pour reconnaître en ligne des arythmies cardiaques à partir des évènements symboliques datés fournis par le module d'abstraction temporelle, CALICOT s'appuie sur un système de reconnaissance de chroniques. Les modèles de chronique [Dousson *et al.*, 1993] sont des ensembles d'évènements temporellement contraints représentant l'évolution normale ou anormale d'un système observé. Les contraintes associées aux évènements permettent de limiter les délais entre chaque occurrence d'un évènement. Dans CALICOT, les modèles de chronique représentent les *signatures* des arythmies que l'on cherche à reconnaître, c'est-à-dire les séquences d'évènements minimales discriminant une arythmie par rapport à une autre. Ces séquences comprennent également les contraintes temporelles entre les évènements. Le reconnaisseur de chroniques analyse le flot d'évènements produit par le module d'abstraction temporelle pour détecter ces signatures. Si les évènements observés et leurs instants d'apparition coïncident avec les contraintes spécifiées dans un des modèles de chronique, une chronique (précisément, une instance du modèle de chronique) est *reconnue*. Un modèle de chronique peut également spécifier des évènements à générer ou des actions à déclencher lorsqu'une chronique est reconnue. Cette possibilité n'est pas utilisée dans CALICOT. Le système de reconnaissance de chroniques a également une fonction prédictive puisqu'il peut anticiper l'apparition d'évènements à venir en maintenant un journal des chroniques en cours de reconnaissance. En effet, lorsque le premier évènement d'une chronique se produit et tant que la durée de la chronique n'est pas dépassée, la chronique est stockée dans le journal comme étant *en cours de reconnaissance* et les évènements qui la composent qui ne sont pas encore intervenus sont *attendus*. Le reconnaisseur de chroniques utilisé dans CALICOT est CRS (*Chronicle Recognition System*) [Dousson, 1996]. Ce système de raisonnement temporel a été conçu pour reconnaître efficacement (avec une complexité polynomiale) les instances de chroniques en ligne. Il a été appliqué et validé dans des domaines industriels tels que la supervision de réseaux de télécommunications.

2.2.2 Acquisition automatique de chroniques (hors ligne)

Dans CALICOT, les chroniques sont construites à partir de règles de classification décrites sous forme de clauses PROLOG. Ces clauses sont apprises par une technique d'*apprentissage relationnel* nommée *Programmation Logique Inductive* (PLI) à partir d'un ensemble d'exemples décrits de manière symbolique pour chaque type d'arythmie que l'on cherche à caractériser. Un exemple de règle PROLOG apprise pour l'arythmie *bloc de branche gauche* (*lbbb*) est présenté ci-dessous.

Exemple 1

```
rule(lbbb):-
  qrs(R0, anormal),
  p_wav(P1, normal), suc(P1,R0),
  qrs(R1, anormal), suc(R1,P1),
  pr1(P1, R1, normal).
```

Cette règle signifie qu'une arythmie de type *lbbb* est reconnue si l'on repère sur le signal ECG un complexe QRS nommé $R0$ de forme anormale, suivi (relation *suc*) d'une onde P ($P1$) de forme normale, elle-même suivie d'un autre QRS ($R1$) de forme anormale. En plus des relations de succession, la règle précise que l'intervalle temporel (*pr1*) entre $P1$ et $R1$ doit être normal (dans notre cas, la normalité de cette intervalle se situe entre 120 et 320 ms cf. Section 1.1.3). Une description détaillée de la méthode d'apprentissage par PLI utilisée pour apprendre ce type de règle est donnée dans le chapitre suivant.

La règle PROLOG précédente est traduite en un modèle de chronique dans le formalisme du système de reconnaissance de chroniques utilisé (CRS) . Le modèle de chronique correspondant à l'exemple 1 est donné Exemple 2.

Exemple 2

```
%initialisation des intervalles et déclaration des évènements
interval  nb_cycles1 = [0, 2000]  //durée maximale de la chronique
interval  normalpr1 = [120, 320]
....
domain wave_dom = {normal, abnormal}
message p_wave[?x] { ?x in wave_dom}
message qrs[?x] {?x in wave_dom}
...

%début de la chronique
chronicle lbbb[]() {
occurs(0,0,p_wave[*],(start+1,R0-1))
occurs(0,0,qrs[*], (start+1,R0-1))
event(qrs[?w0], R0)  //(qrs( R0 ,anormal, _ ),
?w0 in {anormal}
```

```
occurs(0,0,p_wave[*],(R0+1,P1-1))
occurs(0,0,qrs[*], (R0+1,P1-1))
event(p_wave[?w1], P1)  //p_wav( P1 ,normal, R0 ),
?w1 in {normal}
R0 < P1

occurs(0,0,p_wave[*],(P1+1, R1-1))
occurs(0,0,qrs[*], (P1+1, R1-1))
event(qrs[?w2], R1)  //qrs( R1,anormal, P1),
?w2 in {anormal}

P1 < R1
R1 - P1 in  normalpr1 //pr1( P1 , R1 ,normal)
 end - start in nb_cycles1}
```

La syntaxe des chroniques CRS s'appuie sur quatre types d'énoncés :

1. `occurs(n1, n2, E, (t1, t2))` signifie que le modèle de chronique attend entre `n1` et `n2` évènements de type `E` entre les instants `t1` et `t2` (`t1` inclus et `t2` exclus). Lorsque les valeurs `n1`=`n2`=0, il spécifie qu'aucun évènement de type `E` ne doit se produire entre `t1` et `t2`. Lorsque `n1=n2=1` et `t2`=`t1`+1, le modèle de chronique attend seulement un évènement de type `E` à l'instant `t1`. Ceci peut être exprimé plus simplement par l'énoncé `event(E,t1)`;
2. Le changement instantané de la valeur d'un attribut s'exprime par l'expression `event(P:(b,c),t)` signifiant que `P` passe de la valeur `b` à `c` à l'instant `t`;
3. L'assertion `hold(P:a,(t1,t2))` représente la persistance de la valeur `a` de l'attribut `P` dans l'intervalle `[t1,t2]`.
4. `t2 - t1 in I` signifie que les instants `t1` et `t2` sont reliés par une contrainte temporelle (définie ici par l'intervalle `I`).

Dans l'exemple 2, le mot clef `start` symbolise l'instant débutant l'intervalle temporel couvert par la chronique et le mot clef `end`, la fin de cet intervalle. `occurs(0,0,p_wave[*],(start+1,R0-1))` signifie qu'il n'y a pas d'onde P dans l'intervalle `[start+1,R0-1]`. `[*]` signifie que toutes les valeurs de l'attribut spécifiant la forme de l'onde P sont acceptées. `event(qrs[?w0], R0)` signifie qu'un évènement de type QRS de forme `?w0` (`?x` est la syntaxe des variables) se produit au temps `R0`. `?w0 in anormal` signifie que la variable `?w0` prend ses valeurs dans l'ensemble {anormal} (ici réduit à un singleton).

La syntaxe des chroniques, bien que limitée à la résolution des problèmes de satisfaction de contraintes temporelles simples dans lesquels toutes les contraintes sont définies par un unique intervalle temporel [Dechter *et al.*, 1991], est suffisante pour représenter toutes les arythmies apprises dans le cadre du projet PISE.

Dans [Ghallab, 1996], Ghallab propose d'apprendre des modèles de chroniques (des ensembles de chroniques représentant les évolutions possibles d'un système observé) sous forme d'automates [Oncina et García, 1991]. Les exemples d'apprentissages sont

des instances de chroniques, c'est à dire des séquences d'évènements étiquetés par le nom d'une chronique. La méthode proposée considère les successions d'évènements comme des chaînes de mots sur un alphabet et utilise des techniques d'inférence grammaticale pour générer des automates représentant une succession valide d'évènements. À ces automates sont ensuite ajoutées des contraintes temporelles pour former des chroniques (cf. [Ghallab, 1996] pour une description précise du processus).

Les règles de classification que nous engendrons bénéficient de l'expressivité de la logique du premier ordre et sont plus facilement interprétables par les médecins que les automates proposés par Ghallab. De plus, une technique d'apprentissage supervisé telle que la PLI permet de découvrir des relations existant entres les évènements caractéristiques de l'ECG autres que la simple succession.

2.2.3 Conclusion sur Calicot

Deux types de signaux ont été utilisés pour valider le système Calicot à la fin du projet Pise : des signaux non bruités de la base d'électrocardiogrammes MIT-BIH [Moody, 1997b], et des signaux générés par le modèle cardiaque Carmen (cf. section 1.2.2). Les résultats obtenus sur ces signaux parfaits sont très prometteurs, puisqu'ils donnent des mesures de précision de l'ordre de 98% en apprentissage, et de très bons résultats en reconnaissance en ligne de signaux réels sur les quatre types d'arythmie étudiés dans [Carrault *et al.*, 2003] : le *bloc de branche gauche*, l'*extrasystole ventriculaire*, *le bigéminisme* et *le mobitz de type II*. Calicot est à notre connaissance le seul système de reconnaissance d'arythmies cardiaques pouvant fonctionner en ligne avec une chaîne de traitement des données complètes permettant de transformer le signal analogique en représentations symboliques et proposant une méthode permettant l'acquisition automatique de connaissances.

2.3 Le système Kardio

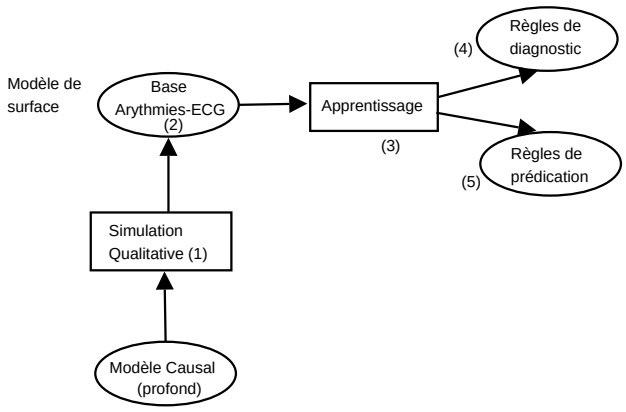


Fig. 2.3 – Architecture de Kardio

KARDIO (cf. Figure 2.3) est un système fonctionnant à partir de connaissances profondes (cf. Section 1.2.2) dont le but est d'apprendre des règles permettant le diagnostic de toutes les arythmies cardiaques possibles (simples ou combinées) correspondant à une description particulière de l'ECG.

L'apprentissage de ces règles repose sur un modèle qualitatif du cœur (cf. Figure 2.4) qui permet de simuler l'activité électrique cardiaque. Des descriptions d'ECG sont générées à partir du modèle cardiaque. La génération repose sur l'utilisation de quatre composants principaux :

1. les nœuds du réseau électrique;
2. un dictionnaire d'arythmies simples en relation avec des dysfonctionnements cardiaques;
3. des contraintes sur les états possibles du cœur;
4. un ensemble de règles locales et globales.

Des états sont associés aux chemins de conduction et aux générateurs d'impulsions (un chemin de conduction pourra ainsi être dans un état *anormal*). Les valeurs des attributs décrivant les états des nœuds du modèle et les représentations symboliques associées aux impulsions électriques permettent de modéliser tous les dysfonctionnement simples du cœur. Les arythmies simples sont décrites dans le dictionnaire à partir des états possibles du cœur : chaque arythmie simple correspond au dysfonctionnement d'un des composants du cœur. Les contraintes sur les états permettent d'éviter les états physiologiquement impossibles à atteindre ou simplement inintéressants d'un point de vue médical. Les règles locales spécifient les comportements des composants du cœur en présence d'état anormaux (par exemple, "la conduction doit être bloquée", "le rythme doit être compris entre 60 et 100", etc.). À un même état peut correspondre plusieurs comportements possibles (par exemple un rythme variable).

Les règles globales (cf. Exemple 3) décrivent en logique du premier ordre le schéma électrique du cœur dont la figure 2.4 donne une représentation simplifiée.

Exemple 3

```
[permanent(atria:form(_,Rythm0, Rate0)),heart(av_conduct:State)]⇒
permanent (av_conduct : form(State,Rythm1,Rate1))] &
av_conduct(State,Rythm0,Rythm1,Rate0,Rate1).
```

Cette règle signifie :
Si *il y a une impulsion permanente dans le pacemaker auriculaire au rythme* `Rythm0` *et à la fréquence* `Rate0` *et que l'état de la conduction auriculo-ventriculaire est* `State`*,* **Alors** *il y a des impulsions de type* `State`*, de rythme* `Rythm1` *et de fréquence* `Rate1` *à la sortie de la conduction AV;*
Et `State, Rythm0, Rate0, Rythm1` *et* `Rate1` *doivent satisfaire la relation* `av_conduct` *(spécifiée dans les règles locales).*

Le modèle cardiaque de KARDIO permet de générer 140000 descriptions d'ECG par simulation qualitative (cf point 1 Figure 2.3) à partir de toutes les combinaisons d'arythmies simples physiologiquement possibles (environ 2400). Chaque arythmie combinée a

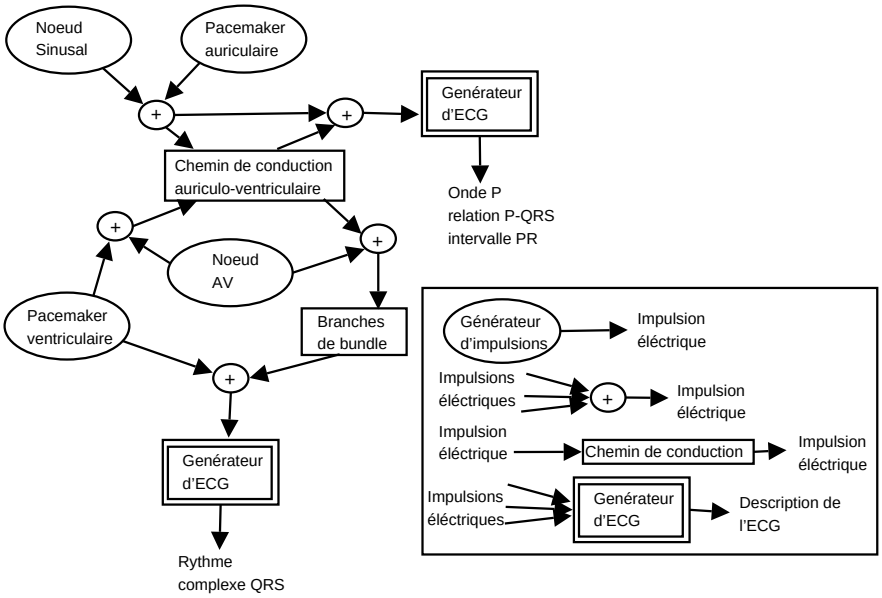


FIG. 2.4 – Modèle qualitatif du cœur de KARDIO (modèle profond)

donc en moyenne 60 descriptions d'ECG correspondantes. L'initialisation du système se fait en instanciant le modèle avec l'une des arythmies simples du dictionnaire. Le processus de simulation des descriptions de l'ECG à partir des quatre composants principaux du modèle détaillés ci-dessus est donné dans [Bratko *et al.*, 1989], il est proche du fonctionnement d'un interpeteur PROLOG.

Cette méthode produit des règles qui permettent de répondre à des questions du type "étant donnée une arythmie, quels sont les ECG possibles qui lui correspondent" mais elles ne permettent pas de répondre aux questions usuelles de diagnostic du type : "étant donné un ECG, quelles sont les arythmies qui peuvent en être la cause?". Pour obtenir ce type de règles de diagnostic, une méthode naïve aurait été d'utiliser le modèle en se servant des règles globales en chaînage inverse i.e., partir des descriptions d'ECG pour aboutir aux états du cœur. Cette méthode s'est avérée trop complexe (beaucoup d'indéterminismes en prenant les règles à l'envers) pour être utilisable. La solution proposée a donc été de "compiler" le modèle cardiaque pour générer à partir de la connaissance profonde un *modèle de surface* (cf. point 2 Figure 2.3) permettant de représenter les relations arythmies-ECG par des règles associatives de type `(arythmie multiple, descriptions d'ECG)`. 8314 règles associatives ont ainsi été générées. Elles se lisent $AM \Rightarrow E$ où AM est une arythmie combinée (ou multiple) et E est la disjonction de toutes les descriptions d'ECG possibles causées par AM. À une même description de l'ECG peut correspondre plusieurs arythmies multiples.

Pour effectuer un diagnostic usuel à partir de ces règles associatives, il suffit d'utiliser leur contraposée. "Si un ECG donné ne correspond pas à E alors l'arythmie correspondante AM n'est pas un diagnostic possible". Toutes les arythmies qui ne sont pas

éliminées sont des diagnostics possibles.

Le manque de lisibilité de la base composée des 8314 règles associatives a poussé les auteurs de KARDIO à s'intéresser à des règles compressées. Des techniques d'apprentissage inductif (cf. point 3 Figure 2.3) ont été utilisées pour généraliser les règles associatives décrites précédemment. Pour former les exemples d'apprentissage, une sous-partie des règles précédentes, choisie de telle manière qu'il y ait le moins de perte de connaissances possible, a été transformée en deux ensembles d'exemples sous forme de paires du type :`(une arythmie simple, des descriptions d'ECG)` et `(une caractéristique de l'ECG, une arythmie multiple)`. Ces deux ensembles d'exemples ont permis d'apprendre respectivement des *règles de prédiction compressées* et des *règles de diagnostic compressées* (cf. Exemple 4).

Exemple 4 (Règles compressées)
Une règle de prédiction pour le mobitz *de type II:*

```
[av_conduct = mob2] ⇒
[relation_P_QRS = after_P_some_QRS_miss] &
[dominant_PR = normal].
```

Une règle de diagnostic:

```
[dominant_QRS = normal] ⇒
[av_conduct = avb1 ∨ wen ∨ mob2 ∨  avb3 ∨ lgl ∨ normal] &
[bundle_branches = normal] &
[reg_vent_focus = quiet].
```

La règle de prédiction signifie "si la conduction auriculo-ventriculaire est dans l'état *mobitz 2* alors il manque certaines ondes P entre les complexes QRS et, l'intervalle PR provenant du foyer d'excitation principal est normal". La règle de diagnostic signifie "si le complexe QRS provenant du foyer d'excitation principal (dominant) est *normal* alors la conduction auriculo-ventriculaire peut être dans l'état *avb1* ou *wen*, etc. , les branches de bundles sont dans un état normal et l'intervalle PR dominant est normal".

2.4 Méthodologie de comparaison de deux systèmes de monitoring des arythmies cardiaques

KARDIO, en sa qualité de système précurseur, a souvent été comparé à CALICOT, sans que cette comparaison ne s'appuie sur une étude rigoureuse. Une première contribution de notre travail [Fromont *et al.*, 2003] a été de fournir une méthodologie de comparaison détaillée de ces systèmes, dont l'objectif commun est l'interprétation de rythmes cardiaques à partir de données qualitatives.

La Figure 2.4 donne les architectures comparées de ces deux systèmes. Nous débutons cette section par une analyse de la différence de sémantique des règles de diagnostic proposées dans les deux systèmes puis de la différence des langages de représentation des attributs de l'ECG permettant aux deux systèmes de diagnostiquer une arythmie. Pour pouvoir comparer de manière équitable les performances des systèmes, il a fallu

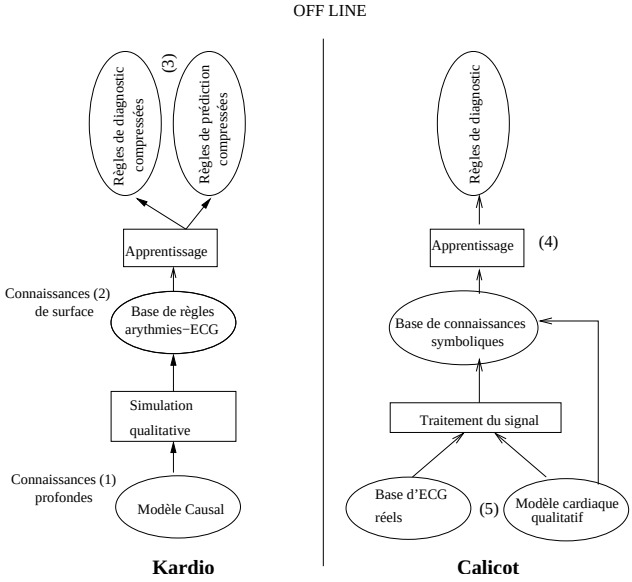


FIG. 2.5 – Architectures comparées

trouver une représentation commune des règles de diagnostic, puis un langage commun pour la description de l'ECG.

2.4.1 Règles de diagnostic

L'exemple 1 de la section 2.2 montre une règle de classification de CALICOT. Elle s'exprime sous la forme $A \Leftarrow a_1 \wedge a_2 \ldots a_n$ qui signifie que l'arythmie A peut être diagnostiquée si les caractéristiques (attributs) $a_1 \wedge a_2 \ldots a_n$ sont visibles sur l'ECG. L'exemple 4 de la section 2.3 montre une règle de diagnostic et une règle de prédiction de KARDIO. La règle de diagnostic est du type $a_1 \Rightarrow A_1 \wedge A_2 \wedge A_n$ et signifie que la présence d'un attribut particulier dans la description de l'ECG implique la présence d'une arythmie multiple décrite par une conjonction d'arythmies simples A_i. La règle de prédiction est une règle causale du type $A \Rightarrow (S_1 \vee S_2 \ldots S_n)$ où $S_i = (a_1 \wedge a_2 \ldots a_n)$ et signifie que la présence d'une arythmie simple A implique plusieurs descriptions possibles de l'ECG S_i. Les règles de *diagnostic* de KARDIO n'ont pas la même sémantique que celles de CALICOT : si l'on dispose d'une description de l'ECG, il faut utiliser plusieurs de ces règles, en vérifiant les possibles incompatibilités entre les attributs et en supposant un état par défaut pour chaque attribut non défini pour connaître l'ensemble des arythmies simples qui ont pu causer une telle description. Ces règles ne sont pas faites pour connaître la caractérisation d'une arythmie particulière.

Les règles de *prédiction* de KARDIO sont, en revanche, beaucoup plus proches des règles de classification de CALICOT. En effet, ces règles ont été déduites du modèle de surface de KARDIO à partir de paires de type `(une arythmie simple, des descriptions d'ECG)` qui contiennent les mêmes éléments que les règles de classifica-

tion de CALICOT. Il est possible d'obtenir un diagnostic partiel en utilisant les règles de prédiction de KARDIO (de même que pour les règles obtenues directement avec le modèle de surface, cf. Section 2.3) en contraposant la règle de prédiction pour obtenir une règle du type $\neg A \Leftarrow \neg(a_1 \wedge a_2 \ldots a_n)$. Dans la suite, nous nous servirons des règles de prédiction de KARDIO pour obtenir un diagnostic dont la sémantique est proche de celle du système CALICOT.

2.4.2 Description qualitative de l'ECG

Comme décrit dans la section 2.2, le but final de CALICOT est d'analyser le signal en ligne pour diagnostiquer des arythmies en faisant correspondre des représentations symboliques du signal à des motifs pré-appris. Les représentations symboliques du signal sont fournies par le module d'abstraction temporelle via des méthodes d'analyse du signal. Le nombre de caractéristiques des ondes extraites du signal est donc limité par les technologies de traitement du signal. Le langage utilisé pour apprendre ces motifs doit donc être adapté à ce que les algorithmes de traitement du signal peuvent effectivement produire. Par exemple, il est difficile de fournir deux descriptions qualitatives différentes pour un bloc de branche gauche (*lbbb*) et un bloc de branche droite (*rbbb*) à partir d'un signal réel (cf section 1.1.3.3 et Annexe A).

Dans KARDIO en revanche, la description qualitative des ondes est fournie par le modèle cardiaque (il n'y a pas d'étape de traitement du signal). Le langage utilisé pour la description du signal ECG et des arythmies dans KARDIO est choisi et généré grâce au modèle de tel manière qu'il soit suffisamment riche pour décrire et distinguer les arythmies sans se préoccuper de savoir si la détection de tels objets et de leur attributs est à la portée des algorithmes de traitement du signal. Dans le cas des blocs de branche, KARDIO utilise des valeurs d'attributs telles que *wide-lbbb* ou *wide-rbbb* pour décrire le signal même si ce niveau de raffinement est très difficile à atteindre par un algorithme de traitement du signal. On peut donc s'attendre à ce que le pouvoir de discrimination de KARDIO soit meilleur que celui de CALICOT puisque, à cause de son langage limité, CALICOT ne peut pas différencier certaines arythmies. L'utilisation en ligne des règles produites dans KARDIO pour la détection en ligne des arythmies impliquerait cependant l'utilisation d'algorithmes de traitement du signal capables de produire une description du signal adaptée aux besoins de KARDIO. Même avec l'évolution actuelle très rapide de ces algorithmes, une description si précise n'est pas encore envisageable dans les années à venir (plus particulièrement dans les conditions bruitées associées aux signaux des USI ou aux signaux enregistrés par des appareils de type Holter).

En outre, les règles PROLOG apprises hors ligne dans CALICOT sont facilement transposables en chroniques [Dousson *et al.*, 1993] et le résultat du diagnostic en ligne peut être aisément comparé, grâce au reconnaisseur de chroniques (CRS) utilisé, aux annotations fournies par les médecins. Pour KARDIO en revanche, le seul moyen de valider les résultats est de requérir l'avis des experts, ce qui reste très coûteux en temps. De plus, l'expert humain aura du mal à juger des règles dont la signification n'est pas exactement conforme à la connaissance que l'on peut trouver dans la littérature médicale.

2.4.3 Méthodologie de comparaison

Pour effectuer cette comparaison, nous avons sélectionné des règles représentant des arythmies reconnaissables à la fois par KARDIO et par CALICOT : le *rythme sinusal* (*rs*), le *bloc de branche gauche* (*lbbb*) et *le mobitz de type II*. Le but est de transformer chacune de ces règles en chroniques CRS pour tester les performances des deux systèmes avec le reconnaisseur de chroniques. Dans le cas de CALICOT, la transformation est automatique (cf Section 2.2.2). Dans le cas de KARDIO, il est tout d'abord nécessaire de transformer les règles de prédiction causales en règles de classification transposables en chroniques.

Soit $A \Rightarrow D$ une règle de prédiction de KARDIO où A est une arythmie et $D = a_1 \wedge, \ldots \wedge a_n$ une description de l'ECG. Soient $A_1 \Rightarrow D, A_2 \Rightarrow D, \ldots, A_n \Rightarrow D$ toutes les autres règles de prédiction dans lesquelles toute la description de l'ECG D est impliquée par une arythmie A_i.
On a donc $A \vee A_1 \vee A_2 \vee \ldots \vee A_n \Rightarrow D$.
Si on considère que la théorie de KARDIO est complète, on peut utiliser l'hypothèse du monde clos [Genesereth et Nilsson, 1987] et compléter la règle précédente : $D \Leftrightarrow A \vee A_1 \vee A_2 \vee \ldots \vee A_n$.

Le fait de sélectionner, pour la comparaison des deux systèmes, seulement trois arythmies dans la base de connaissance de KARDIO exclut la possibilité d'utiliser l'hypothèse du monde clos pour l'interprétation des règles de KARDIO. En particulier, lorsque l'on ignore volontairement les connaissances concernant $A_1, A_2, \ldots, A_n$, on ne peut pas réduire la règle $D \Rightarrow A \vee A_1 \vee A_2 \vee \ldots \vee A_n$ à $D \Rightarrow A$. Par conséquent, nous utilisons un nouveau symbole NC (pour "Non Connu") représentant la connaissance possiblement perdue et notons la règle précédente $D \Rightarrow A \vee NC$. Cette règle est interprétée comme suit : "si D est une description de l'ECG alors A est un trouble possible". Si une chronique construite à partir d'une règle de classification de KARDIO est reconnue, cela ne signifie donc pas que l'arythmie associée est l'unique diagnostic possible correspondant à une description d'ECG donnée. On peut en revanche conclure que, si la chronique n'est pas reconnue, l'arythmie associée n'est pas un diagnostic possible.

Exemple 5 (Règle de prédiction de Kardio pour l'arythmie lbbb)

```
 [bundle_branches = lbbb] =>
 [dominant_QRS = wide_LBBB] &
 [rate_of_QRS = under_60..between_100_250]&
 [ectopic_QRS(1) = wide_LBBB \/ wide_non_specific \/ absent]
\/
[dominant_QRS = wide_LBBB]&
[ectopic_PR(1) = meaningless \/ after_QRS_is_P]&
[ectopic_QRS(1) = wide_RBBB ]
\/
[dominant_QRS = delta_LBBB \/ delta_RBBB]&
```

```
[ectopic_QRS(1) = wide_LBBB ]
\/
[rhythm_QRS = regular]&
[relation_P_QRS = meaningless \/ after_P_always_QRS \/ independent_P_QRS]&
[dominant_PR = meaningless \/ after_QRS_is_P]&
[dominant_QRS = wide_RBBB \/ wide_non_specific] &
[ectopic_QRS(1) = wide_LBBB ]
```

Pour la règle de prédiction de l'exemple 5, un grand nombre de chroniques vont être construites. En particulier, pour la première partie de la règle, l'attribut `under_60..between_100_250` du langage de KARDIO qui signifie que l'intervalle RR est soit inférieur à 60 ms soit compris entre 100 et 250 ms ne peut pas être représenté par une seule règle de classification. De plus, dans cette première partie, la règle ne donne aucune information sur le comportement de l'onde P. Nous avons donc considéré que cette onde n'était pas significative. Du point de vue de la reconnaissance de chroniques, cela revient à dire qu'on autorise toutes les ondes P possibles entre les complexes QRS. L'exemple 6 donne un ensemble de règles *Prolog* correspondant à la première partie de la règle de prédiction de l'exemple 5 pour un rythme inférieur à 60 ms. Le prédicat *non_representative(p)* signifie que l'onde P peut intervenir à n'importe quel moment et sous n'importe quelle forme durant la séquence décrite par le règle. *wide_LBBB* et *wide_other* sont des caractéristiques du complexe QRS.

Exemple 6

```
%RR1 short = under 60
rule(lbbb) :-
        non_representative(p),
        qrs(R0, wide_LBBB),
        qrs_ectopic(R01,X), suc(R01,R0),
        in(X,[wide_LBBB,wide_other]),
        qrs(R1,wide_LBBB),
        R0 \== R1,
        rr1(R0,R1,short).

rule(lbbb) :-
        non_representative(p),
        qrs(R0, wide_LBBB), %qrs_ectopic absent
        qrs(R1,wide_LBBB),
       R0 \== R1,
        rr1(R0,R1,short).

%RR1 normal = between 100-250
....
```

Nous avons utilisé le modèle cardiaque CARMEN (cf. Section 1.2.1) pour la première partie de nos expérimentations afin de simuler des électrocardiogrammes. CARMEN

fournit également la représentation des signaux sous forme symbolique comme le fait le module d'abstraction temporelle de CALICOT. La synthèse des représentations symboliques se fait en trois étapes. Avant de commencer la simulation, le modèle cardiaque CARMEN est initialisé. Un ensemble de paramètres identifiés comme déclencheur de pathologies cardiaques (comme *lbbb* ou *mobitz*) est chargé dans CARMEN. Puis, des scénarios associés aux pathologies voulues sont générés aléatoirement par le modèle en modifiant les paramètres physiologiques du modèle toutes les quatre secondes. Enfin, à la fin de chaque simulation, la représentation symbolique interne de chaque onde générée par le modèle est transformée en évènements décrits à la fois dans un langage propre à CALICOT mais aussi dans un langage propre à KARDIO. La figure 2.6 montre un exemple de telles représentations symboliques dans le cas de KARDIO et de CALICOT. Ces évènements sont les entrées du reconnaisseur de chroniques.

CALICOT	KARDIO
4377 qrs[abnormal]	4377 qrs[wide_LBBB]
5208 qrs[abnormal]	5208 qrs_ectopic[wide_LBBB]
5239 p_wave[normal]	5239 p_wave[normal]
6323 p_wave[abnormal]	6323 p_wave[abnormal]
6525 qrs[abnormal]	6525 qrs[wide_LBBB]
7408 p_wave[normal]	7408 p_wave[normal]
7618 qrs[abnormal]	7618 qrs[wide_LBBB]
8111 qrs[abnormal]	8111 qrs_ectopic[wide_LBBB]
8493 p_wave[abnormal]	8493 p_wave[abnormal]
.....	.....

FIG. 2.6 – Exemple de descriptions d'ECG correspondant à des séquences (Instant d'apparition Évènement correspondant) pour KARDIO et CALICOT pour une arythmie de type *mobitz*.

Les résultats de la reconnaissance de chroniques sont ensuite utilisés pour évaluer les performances des deux systèmes. Pour compenser la différence de sémantique des règles de classification provenant des deux systèmes, nous avons décidé de compter comme bonne reconnaissance (vrai positif : VP) une arythmie qui est dans l'ensemble des diagnostics possibles et qui devrait être reconnue, comme faux négatif (FN) une arythmie qui n'est pas dans l'ensemble des arythmies reconnues alors qu'elle devrait s'y trouver, comme faux positif (fausse alarme), une arythmie qui est dans l'ensemble des arythmies reconnues alors qu'elle ne devrait pas y être. Dans le cas de faux positifs, lorsque l'arythmie annotée n'est pas l'une des trois arythmies étudiées (les fichiers d'annotations médicales n'ont bien sûr pas été modifiés et contiennent d'autres arythmies), elle est exceptionnellement comptée comme VP si l'arythmie détectée à la place est un *rythme sinusal*. En effet, dans les règles du système KARDIO, le rythme sinusal peut très souvent être combiné à d'autres troubles du rythme et dans ce cas, il est possible que la bonne arythmie ait été reconnue avec la base de connaissances complète. En pratique, la détection des vrais négatifs (VN) est calculée par la formule $VN = Tot - VP + FP + FN$ ou Tot est le nombre total de reconnaissances.

Les critères de comparaison utilisés sont la *sensibilité* qui donne la probabilité de bonne classification d'un rythme observé et la *spécificité* qui reflète la capacité du système à ne pas proposer un rythme particulier si le rythme observé n'est pas le rythme annoté. Ces mesures sont respectivement calculées par les formules :
$SENS = \frac{VP}{VP+FN}$, $SPEC = \frac{VN}{FP+VN}$.

2.4.4 Résultats

Deux expérimentations ont été conduites et les résultats sont donnés dans les matrices de confusion présentées dans les tableaux 2.1, 2.2, 2.3 et 2.4. La première expérimentation compare directement KARDIO et CALICOT et la seconde compare CALICOT avec le système KARDIO dont le langage a été affaibli. Dans chacune des expérimentations, les lignes de la matrice représentent les détections des deux systèmes. Les colonnes de la matrice représentent les annotations fournies par CARMEN pour la première expérimentation et par la base de données du MIT pour la seconde. Le mot NC est utilisé pour les arythmies *inconnues*, ie les arythmies qui ne sont pas considérées dans cette étude (par exemple *rbbb* ou *extrasystole*). Le mot NR est utilisé pour les arythmies *non reconnues*.

2.4.4.1 Kardio vs Calicot

Dans cette première expérimentation, nous avons décidé de ne pas changer le langage de KARDIO et de comparer directement les performances des règles de classification avec celles de CALICOT. Pour cela, nous avons utilisé le même ECG décrit avec deux représentations symboliques différentes adaptées aux langages des deux systèmes. Nous ne possédons pas d'ECG réel décrit dans le langage de KARDIO, nous avons donc utilisé Carmen (cf. section 2.4.3) pour produire les deux représentations symboliques d'un même ECG (cf . Fig. 2.6).

Un exemple de chroniques CRS utilisé pour tester le système KARDIO est donné Figure 2.7 (la chronique correspondante pour CALICOT a été donnée en section 2.2.2, Figure 2). Un commentaire est donné après chaque évènement pour expliciter la signification du prédicat utilisé ou pour donner la règle PROLOG à partir de laquelle la chronique a été construite. On peut noter qu'une seule chronique suffit à CALICOT pour reconnaître une arythmie de type *lbbb* alors qu'avec KARDIO, dont le langage est beaucoup plus précis, treize chroniques sont nécessaires pour décrire un *lbbb*. La première de ces treize chroniques est exposée Figure 2.7. Dans cette chronique, le QRS dominant (celui sur lequel est calculé le rythme) est spécifié comme étant du type *wide_lbbb* et le QRS ectopique comme étant soit du type *wide_lbbb* soit du type *wide_other*. Dans la règle issue de CALICOT (cf. Figure 2), il n'y a pas de différence entre un QRS dominant ou ectopique, seule la forme du QRS est spécifiée comme étant *anormale*.

Les résultats des expérimentations sont donnés dans le tableau 2.1 pour KARDIO et dans le tableau 2.2 pour CALICOT. Il y a énormément de *mobitz* non reconnu (FN) pour les deux systèmes. Cela provient de l'annotation des arythmies fournie par CARMEN. En effet, le modèle CARMEN génère des évènements de manière aléatoire et peut

```
chronicle lbbb[]() {
occurs(0, 0,qrs[*], (start+1, R0-1)) //pas de qrs entre [START, R0-1]
event(qrs[?w0], R0) //qrs(R0,_,wide_lbbb),
?w0 in {wide_lbbb}

occurs(0, 0,qrs[*], (R0+1, R01-1)) //pas de qrs  entre[R0+1, R01-1]
event(qrs_ectopic[X], R01) //qrs_ectopic(R01,R0,X),
R0 < R01

X in {wide_lbbb,wide_other}
occurs(0, 0,qrs[*], (R01+1, R1-1)) //pas de qrs entre [R01+1, R1-1]
event(qrs[?w3], R1) //qrs(R1,_,wide_lbbb),
?w3 in {wide_lbbb}

R0 < R1
R1 - R0 in shortrr1 //rr1(R0, R1, short)

end - start in nb_cycles1}
```

FIG. 2.7 – Une des chroniques CRS de KARDIO pour l'arythmie de type lbbb

donc générer certains motifs d'évènements très rares d'un point de vue médical mais correspondant toujours à la pathologie *mobitz* (par exemple quatre ondes P consécutives). Cependant, ces motifs n'étant pas courants dans un contexte clinique, les règles correspondant à ces cas particuliers n'ont pas été apprises par les deux systèmes et les motifs ne sont donc pas reconnus. De plus, les règles correspondant au *mobitz* sont plus précises pour CALICOT que pour KARDIO puisque la première règle spécifie que la forme de l'onde P doit être normale alors que pour KARDIO, aucune information sur la forme de l'onde n'est spécifiée. La règle de KARDIO permet donc de reconnaître plus de types de *mobitz* et a donc une meilleure sensibilité. En outre, nous pouvons remarquer que les résultats pour l'arythmie de type *lbbb* et en particulier les FP sont meilleurs pour KARDIO (0) que pour CALICOT (705). En effet, le langage de CALICOT ne permet pas de faire la différence entre une arythmie de type *rbbb* et une arythmie de type *lbbb* comme expliqué dans la section 2.4.2, ces deux arythmies sont donc obligatoirement confondues, ce qui génère un FP pour chaque *rbbb* (NC pour notre expérience). Enfin, il y a beaucoup plus de VP pour le rythme sinusal (rythme normal) dans le cas de KARDIO (2785) que dans celui de CALICOT (1816) à cause du choix fait pour la détection des arythmies non reconnues (cf. section 2.4.3). En effet, lorsque KARDIO détecte un rythme normal à la place d'une arythmie inconnue (par exemple *rbbb*), nous avons considéré cette détection comme une détection correcte pour le rythme normal puisque KARDIO a éliminé *mobitz* et *lbbb* des arythmies encore possibles.

KARDIO obtient donc de meilleurs performances que CALICOT mais les bases de la comparaison pour cette expérience sont très favorables à KARDIO.

	mobitz	lbbb	normal	NC	Total
mobitz	114	0	0	32	146
lbbb	0	20	0	0	20
normal	14	6	2765	0	2785
NR	909	3	0	0	912
Total	1037	29	2765	32	3863
Sensibilité	0.11	0.69	1	0	
Spécificité	0.99	1	0.98	1	

TAB. 2.1 – Matrice de confusion pour les règles de KARDIO avec des signaux issus de Carmen

	mobitz	lbbb	normal	NC	Total
mobitz	30	0	0	20	50
lbbb	0	22	0	705	727
normal	1	0	1816	0	1817
NR	1328	7	0	0	1335
Total	1358	29	1816	725	3928
Sensibilité	0.02	0.76	1	0	
Spécificité	0.99	0.66	1	1	

TAB. 2.2 – Matrice de confusion pour les règles de Calicot avec des signaux issus de Carmen

2.4.4.2 Kardio affaibli vs Calicot

Dans une seconde expérience, nous avons affaibli le pouvoir d'expression de KARDIO pour qu'il corresponde aux possibilités des algorithmes de traitement du signal actuels. Dans cette expérience, toutes les formes d'ondes qui n'étaient pas explicitées comme *normales* dans KARDIO ont été spécifiées comme étant *anormales* et chaque QRS spécifié comme *ectopique* est maintenant considéré comme un QRS dominant. Il est en effet très difficile avec les technologies de traitement du signal actuelles de différencier, à partir d'un signal ECG réel, la provenance de l'onde QRS ou de donner la forme de l'onde de manière précise.

Pour cette expérimentation, le signal provient d'un ECG réel de la base MIT-BIH et la description symbolique est la même pour les deux systèmes. Un exemple de chronique CRS pour le langage KARDIO affaibli est donné Figure 2.8. Cette chronique correspond à la même règle utilisée pour créer la chronique de la Figure 2.7.

Les résultats sont donnés dans les tableaux 2.4 et 2.3. Ces résultats sont bons et presque équivalents pour les deux systèmes, excepté le fait que KARDIO donne deux fois plus de détections de rythme normaux que CALICOT. En effet, en cas de multiples détections par le système KARDIO, un VP supplémentaire est compté pour le rythme normal si la bonne solution est dans l'ensemble des arythmies détectées. En particulier, pour chaque *rythme normal*, KARDIO affaibli a détecté à la fois le rythme *normal* et

```
chronicle lbbb[]() {
occurs(0, 0,qrs[?],(start+1, R0-1)) //pas de qrs entre [START, R0-1]
event(qrs[?w0], R0) //qrs(R0,_,anormal),
?w0 in {anormal}

occurs(0, 0,qrs[?],(R0+1, R1-1))  //pas de qrs entre [R0+1, R1-1]
event(qrs[?w1], R1) //qrs(R1,_,anormal),
?w1 in {anormal}

occurs(0, 0,qrs[?],(R1+1, R2-1)) //pas de qrs entre [R1+1, R2-1]
event(qrs[?w2], R2) //qrs(R2,_,anormal),
?w2 in {anormal}

R0 < R2
R2 - R0 in shortrr1 //rr1(R0,R2,short)

end - start in nb_cycles2 }
```

FIG. 2.8 – Une des chroniques CRS de KARDIO affaibli pour l'arythmie de type lbbb

	mobitz	lbbb	normal	NC	Total
mobitz	427	0	0	0	427
lbbb	0	2006	0	1432	3438
normal	0	0	7406	0	7406
NR	0	0	0	0	0
Total	427	2006	7406	1432	11271
Sensibilité	1	1	1	0	
Spécificité	1	0.85	1	1	

TAB. 2.3 – Matrice de confusion pour l'évaluation des règles de KARDIO affaibli

	mobitz	lbbb	normal	NC	Total
mobitz	427	0	0	0	427
lbbb	0	2006	8	1106	3120
normal	0	0	2292	0	2292
NR	0	0	291	0	291
Total	427	2006	2591	1106	6130
Sensibilité	1	1	0.88	0	
Spécificité	1	0.73	1	1	

TAB. 2.4 – Matrice de confusion pour l'évaluation des règles de Calicot

plusieurs autres arythmies (*lbbb* ou *mobitz*). On peut également remarquer qu'il y a beaucoup de FP pour l'arythmie de type *lbbb* pour les deux systèmes puisque KARDIO ne peut plus faire la différence entre une arythmie de type *rbbb* et une arythmie *lbbb*

avec un langage affaibli.

2.4.5 Conclusion sur la comparaison

Nous avons présenté la comparaison de deux classifieurs d'arythmies cardiaques très différents. Ce genre de comparaison est loin d'être triviale, compte tenu des différentes sémantiques attribuées aux règles de classification provenant des deux systèmes. Des adaptations ont dû être effectuées sur les deux systèmes pour la rendre possible. Nous avons fourni une évaluation des règles de KARDIO qui n'avait jamais été envisagée pour ce système. La comparaison des deux systèmes a montré que KARDIO, avec sa richesse de langage, est plus précis que CALICOT, mais que ce dernier est adapté aux algorithmes de traitement du signal actuel et peut donc être utilisé en ligne.

Avec l'évolution des méthodes de traitement du signal, nous envisageons de développer la base de connaissances du module d'apprentissage de CALICOT pour apprendre des règles avec un langage de description plus précis et ainsi pouvoir développer cette comparaison.

2.5 Conclusion

Dans ce chapitre, nous avons présenté CALICOT, le prototype de système de monitoring cardiaque sur lequel ont été appliqués les travaux de cette thèse. Nous avons également présenté en détails KARDIO, un autre système permettant de générer des règles de diagnostic et de prédications d'un très grand nombre d'arythmies cardiaques avec lequel CALICOT est très souvent comparé. Nous avons présenté une méthodologie permettant de comparer effectivement ces deux systèmes et nous avons montré que CALICOT, plus proche des possibilités actuelles de traitement du signal, permet d'obtenir des résultats comparables à ceux de KARDIO sur des données réalistes.

Les premiers résultats obtenus avec CALICOT pour le diagnostic en ligne d'arythmies cardiaques en utilisant des signaux d'électrocardiogrammes provenant de bases de données médicales existantes non bruitées sont excellents. Cependant, l'utilisation d'une seule source de données tel qu'un signal ECG dans CALICOT rend le système peu robuste aux bruits présents en milieu clinique. Pour répondre à ce problème ainsi qu'à la problématique plus globale présentée dans le projet CEPICA, nous présentons dans la suite de ce document une étude pour caractériser les arythmies cardiaques à partir d'une nouvelle source de données : la pression artérielle. Dans le dernier chapitre nous exposerons les possibilités d'utilisation conjointe de ces différentes sources pour améliorer les performances de CALICOT en milieu hospitalier.

Le chapitre suivant présente la méthode d'apprentissage que nous avons choisie d'utiliser pour apprendre des règles de classification permettant de diagnostiquer en ligne des arythmies cardiaques. Cette méthode est dans un premier temps appliquée à plusieurs sources prises séparément.

Chapitre 3

Apprentissage par programmation logique inductive

Comme expliqué en section 2.2, le paradigme d'apprentissage des règles permettant de caractériser les arythmies cardiaques pour construire la base de chroniques du système CALICOT est la Programmation Logique Inductive (PLI). Ces règles sont décrites sous forme de clauses de Horn puis transformées en chroniques CRS. Des exemples de telles règles et leurs chroniques correspondantes ont été donnés dans la section 2.4 portant sur la comparaison des systèmes KARDIO et CALICOT.

Nous allons, dans ce chapitre, expliquer en détail la méthode d'apprentissage par PLI à travers deux systèmes ayant deux sémantiques différentes : ALEPH [Srinivasan, 2003] et ICL [De Raedt et Van Laer, 1995]. Les bases de logique nécessaires à la compréhension de ce chapitre sont données dans l'annexe B. Nous ne détaillerons pas les notions d'induction et d'apprentissage artificiel, mais le lecteur intéressé peut se référer à [Mitchell, 1997] et [Cornuéjols et Miclet, 2003].

La dernière partie du chapitre présente les résultats obtenus avec ALEPH et ICL sur l'apprentissage de règles caractérisant des arythmies cardiaques à partir de données provenant d'un électrocardiogramme d'une part et de mesures de pression artérielle d'autre part.

3.1 Généralités sur la PLI

Cette introduction est en partie basée sur l'article fondateur de De Raedt et Muggleton [Muggleton et De Raedt, 1994]. Les différences pouvant exister entre les systèmes de PLI sont explicitées et nous donnons, pour chacune de ces spécificités, le ou les systèmes fondateurs ayant introduit cette notion.

3.1.1 Motivations

Le but de la PLI est de créer des outils et des techniques permettant d'induire des hypothèses à partir d'observations et de synthétiser des nouvelles connaissances à partir

d'expériences. Cette technique permet de dépasser les deux principales limitations des techniques classiques d'apprentissage automatique que sont :

- l'utilisation d'un formalisme de représentation des connaissances limité (essentiellement de la logique propositionnelle),
- les difficultés rencontrées pour utiliser des connaissances préalables au moment du processus d'apprentissage.

La PLI a déjà prouvé son intérêt dans de nombreuses applications nécessitant l'acquisition automatique de connaissances [Lavrač et Džeroski, 1993]. Dans le domaine de la physique, Dolsak *et al.* [Dolsak et Muggleton, 1992] déterminent par PLI la résolution appropriée d'un maillage à éléments finis permettant de modéliser numériquement une structure physique afin de pouvoir analyser les efforts infligés à cette structure. Dans le domaine de la biologie, King *et al.* [King *et al.*, 2004] utilisent des techniques de PLI pour automatiser un processus scientifique d'expérimentations visant à établir la fonction de gènes. Srinivasan *et al.* [Srinivasan *et al.*, 1996] apprennent grâce à la PLI des structures chimiques responsables d'activités mutagènes. Dans le domaine de l'écologie, Blockeel *et al.* [Blockeel *et al.*, 2004] cherchent à prédire la biodégradabilité de composants chimiques dans l'eau à partir de leur description structurelle. Des travaux sont en cours [Cordier, 2005] pour élaborer un système d'aide à la décision permettant d'améliorer la qualité des eaux de rivières : le but est alors d'apprendre par PLI les relations entre les pratiques agricoles et la qualité des eaux. De nombreuses applications ont également vu le jour en traitement automatique des langues ([Cussens et Džeroski, 2000, Zelle et Mooney, 1993]).

3.1.2 Sémantiques

L'article [Muggleton et De Raedt, 1994] expose deux sémantiques différentes : la sémantique *normale* qui comprend la sémantique *définie* (ensemble des théories T dont le plus petit modèle de Herbrand $\mathcal{M}^+(T)$ est unique) et la sémantique *non monotone*.

Exemple 7 `B :`

```
grandpere(X, Y) :- pere(X, Z), parent(Z, Y)
pere(henri, jeanne) :-
mere(jeanne, jean) :-
mere(jeanne, alice) :-

E+ :
grandpere(henri, jean) :-
grandpere(henri, alice) :-

E- :
:- grandpere(jean, henri)
:- grandpere(alice, jean)

Une hypothèse (H) pouvant être apprise par PLI :
parent(X,Y) :- mere(X,Y).
```

3.1.2.1 Les sémantiques normale et définie

Les systèmes de PLI utilisant une sémantique *normale* disposent d'un ensemble de connaissances *a priori* T et d'un ensemble d'exemples E décrits dans un langage L_E composé d'exemples positifs E^+ et d'exemples négatifs E^- (cf. exemple 7). Les exemples sont des clauses de Horn (éventuellement réduites à des faits) et T est une théorie clausale (i.e. un ensemble de clauses de Horn). Dans la sémantique proposée par [Muggleton et De Raedt, 1994], le but est d'apprendre une théorie clausale H décrite dans un langage L_H telle que les conditions suivantes soient satisfaites :

Définition 3.1 (Sémantique normale)
- *Satisfaisabilité a priori :* $T \wedge E^- \not\models \square$ *(faux)*
- *Satisfaisabilité a posteriori (correction) :* $T \wedge H \wedge E^- \not\models \square$
- *Nécessité a priori :* $T \not\models E^+$
- *Suffisance a posteriori (complétude) :* $T \wedge H \models E^+$

La définition de la sémantique la plus communément acceptée par les utilisateurs de la PLI est celle proposée, par exemple, par Džeroski et Lavrač [Džeroski et Lavrač, 2001] :

Définition 3.2
1. *(Correction)* $\forall e^- \in E^-, T \wedge H \not\models e^-$
2. *(Complétude)* $\forall e^+ \in E^+, T \wedge H \models e^+$

La *relation de couverture* entre les hypothèses de L_H et les exemples de L_E est la conséquence logique. Les apprentissages par PLI qui vérifient la sémantique normale sont nommés *apprentissage par conséquence logique* (traduit de *learning from entailment*). La condition de *correction* se lit "H ne couvre aucun e^- relativement à T" et la condition de *complétude* se lit "H couvre tous les e^+ relativement à T".

Intermezzo Un lecteur non averti qui chercherait à comprendre la sémantique standard de la PLI à travers ces définitions pourrait penser qu'il y a une contradiction entre la règle de *satisfaisabilité a posteriori* de la définition 3.1 qui stipule que l'union de T, H et E^- n'est pas contradictoire et le premier point de la définition 3.2 qui stipule qu'aucun exemple négatif n'est conséquence logique de la connaissance T et de l'hypothèse trouvée. La différence essentielle entre les deux définitions provient de la représentation des exemples (qui n'est pas explicitée lorsque les auteurs proposent leur définition). Dans la définition 3.1, les exemples sont représentés comme indiqué dans l'exemple 7. Dans le langage logique utilisé ici, les exemples positifs f_i^+ sont des faits PROLOG et les exemples négatifs f_i^- sont des clauses instanciées ayant une tête vide (correspondant à la négation d'un fait). Le symbole :− utilisé est une implication logique ($\Leftarrow$) donc $b{:-}a \Leftrightarrow b \vee \neg a$. En général dans la littérature, les exemples positifs et négatifs sont tous deux représentés par des faits. En posant $e_i^- = \neg f_i^-$ on prouve que les deux définitions sont équivalentes (cf. Preuve 1).

Preuve 1 *Soit* e_1^- *et* e_2^- *des exemples négatifs représentés par des faits.*
$(\forall e^- \in E^-, T \wedge H \not\models e^-) \Leftrightarrow$
$((T \wedge H \not\models e_1^-) \wedge (T \wedge H \not\models e_2^-)) \Leftrightarrow$
$(\neg(T \wedge H \models e_1^-) \wedge \neg(T \wedge H \models e_2^-)) \Leftrightarrow$
$(\neg((T \wedge H \models e_1^-) \vee (T \wedge H \models e_2^-))) \Leftrightarrow$

$(\neg(T \wedge H \models e_1^- \vee e_2^-)) \Leftrightarrow$
1- $(T \wedge H \not\models e_1^- \vee e_2^-)$.
Soit $f_1^- et f_2^-$ *des exemples négatifs représentés par des négations de faits.*
$(T \wedge H \wedge E^- \not\models \square) \Leftrightarrow$
$(T \wedge H \wedge (f_1^- \wedge f_2^-) \not\models \square) \Leftrightarrow$
2- $(T \wedge H \not\models \neg f_1^- \vee \neg f_2^-$.
En posant $e_i^- = \neg f_i^-$, *on a bien* $1 \Leftrightarrow 2$.
Le cas de la complétude est trivial.

Dans le cas de la définition 3.2, tous les exemples sont représentés par des faits "positifs" et la définition semble plus intuitive.

Dans la plupart des systèmes de PLI existants, les hypothèses recherchées et les clauses présentes dans la théorie du domaine T sont des clauses *définies* (NB : qui ne contiennent qu'un seul littéral positif). La définition 3.1 peut alors se simplifier en utilisant la propriété d'unicité du plus petit modèle de Herbrand ($\mathcal{M}^+$) des théories utilisant des clauses définies.

Définition 3.3 (Sémantique définie)
- *Satisfaisabilité a priori : tous les* $e \in E^-$ *sont faux dans* $\mathcal{M}^+(T)$
- *Satisfaisabilité a posteriori (correction) : tous les* $e \in E^-$ *sont faux dans* $\mathcal{M}^+(T \wedge H)$
- *Nécessité a priori : certains* $e \in E^+$ *sont faux dans* $\mathcal{M}^+(T)$
- *Suffisance a posteriori (complétude) : tous les* $e \in E^+$ *sont vrais dans* $\mathcal{M}^+(T \wedge H)$

Ce qui devait être vrai dans tous les modèles de $T \wedge H$ pour la sémantique normale, doit maintenant être vrai dans le plus petit modèle de Herbrand pour la sémantique définie. La sémantique définie est donc un cas particulier de la sémantique normale.

Notons que les sémantiques précédentes définissent l'apprentissage d'une théorie clausale couvrant une classe d'exemple : les exemples positifs. En inversant le rôle des exemples positifs et négatifs, on peut également apprendre une théorie clausale couvrant la classe des exemples négatifs (pour la théorie clausale originale).

3.1.2.2 Sémantiques non monotones

A l'origine [Helft, 1989], le cadre non monotone a été conçu pour les problèmes d'induction descriptive dans lesquels l'hypothèse H apprise n'est plus supposée expliquer les exemples positifs ($\forall e^+ \in E^+, H \models e^+$) comme dans le cadre de la sémantique normale mais doit au contraire être une caractéristique des exemples positifs ($\forall e^+ \in E^+, e^+ \models H$).

En PLI, le premier système à utiliser une telle sémantique, CLAUDIEN [De Raedt et Bruynooghe, 1993], possède une théorie du domaine composée de clauses définies mais d'un ensemble d'exemples vide. En effet, les exemples positifs font partie de la théorie du domaine T et les exemples négatifs sont dérivés implicitement en faisant une hypothèse de monde clos. Les hypothèses H sont décrites dans le même langage que la théorie du domaine. Suivant ce modèle, la sémantique non monotone est définie dans [Muggleton et De Raedt, 1994] comme suit :

Définition 3.4 (Sémantique non monotone)

- *Validité : tous les* $h \in H$ *sont vraies dans* $\mathcal{M}^+(T)$
- *Complétude: si une clause* g *est vraie dans* $\mathcal{M}^+(T)$ *alors* $H \models g$
- *Minimalité : il n'y a pas de sous ensemble* G *de* H*, valide et complet*

Dans cette définition, T et H sont toujours des théories clausales. La condition de *validité* signifie que toutes les hypothèses h de H sont cohérentes avec la théorie T (rappelons que T code aussi les exemples positifs) : elles sont des propriétés des données de T. La *complétude* signifie que toutes les informations valides de T doivent être déductibles de l'hypothèse H. La condition de *minimalité* empêche de dériver des hypothèses redondantes. Aucune condition de *correction* n'est nécessaire puisque les exemples négatifs ne sont pas pris en compte.

Avec le développement de nouveaux systèmes de PLI, les exemples négatifs ont été à nouveau introduits dans la définition de la sémantique non monotone sous forme d'interprétations non modèles de la théorie cible (cf. Annexe B). Par opposition, les exemples positifs sont des interprétations modèles de la théorie cible. Syntaxiquement un exemple e est une théorie clausale dont l'interprétation est calculée en prenant le modèle de Herbrand minimal (unique si les clauses sont définies) $\mathcal{M}^+$ de T et e. Le système cherche donc à apprendre des théories clausales H pour lesquelles les exemples positifs sont modèles de la théorie et les exemples négatifs contredisent cette théorie.

La sémantique de l'apprentissage par interprétations présentée ici (cf. Définition 3.5) est réécrite à partir de celle présentée dans l'article de De Raedt et Van Laer [De Raedt et Van Laer, 1995] en considérant une interprétation I comme un ensemble de faits clos et en constatant dans ce cas que si I est modèle d'une théorie clausale H alors $I \models H$.

Définition 3.5 (Apprentissage à partir d'interprétation)

- *(Correction)* $\forall e^- \in E^-$, $\mathcal{M}^+(T \cup e^-) \not\models H$
- *(Complétude)* $\forall e^+ \in E^+$, $\mathcal{M}^+(T \cup e^+) \models H$

Comme dans le cas de la sémantique normale, la condition de *correction* se lit "H ne couvre pas e^- relativement à T" et la condition de *complétude* se lit "H couvre e^+ relativement à T". Notons que dans la plupart des cas, les exemples e sont des ensembles de faits clos. Par abus de langage, ils peuvent donc être vus comme des interprétations partielles que l'on complète en prenant le modèle minimal de Herbrand $\mathcal{M}^+$ de T et e.

De Raedt montre dans [De Raedt, 1997] qu'une réécriture du problème par interprétations assure d'obtenir des solutions qui vérifient également la sémantique *normale* de la définition 3.2. Le contraire n'est cependant pas possible puisqu'on ne peut pas réécrire des théories clausales sous forme d'interprétations, toute la connaissance disponible n'étant pas forcément explicitée dans les clauses. L'apprentissage par interprétations est donc un cas particulier de l'apprentissage par conséquence logique. Cette spécificité de la sémantique par interprétations permet cependant de diminuer la complexité de l'apprentissage. De Raedt et Džeroski montrent ainsi dans [De Raedt et Džeroski, 1994] que l'apprentissage par interprétations est PAC-apprenable [Valiant, 1984] contrairement à

l'apprentissage par conséquence logique. Blockeel donne dans [Blockeel, 1998](*p84-89*) une revue des avantages et des inconvénients de l'apprentissage par interprétations par rapport à l'apprentissage par conséquence logique. En particulier Blockeel voit l'apprentissage par interprétations comme un intermédiaire entre l'apprentissage attribut-valeur, extension du langage propositionnel, (voir [Michalski, 1993] pour une description détaillée des langages d'hypothèses attribut-valeur) et l'apprentissage par conséquence logique.

3.1.3 Structuration de l'espace de recherche

La problématique de la PLI comme celle des autres méthodes d'apprentissage de concept peut se ramener à une recherche dans un espace de clauses (les hypothèses de H vues précédemment) satisfaisant un certain nombre de propriétés nommé *espace de recherche* [Mitchell, 1982]. La recherche des hypothèses dans cet espace est guidée par une notion de généralité (notée $\succeq$) existant entre les clauses. Intuitivement, une hypothèse h_1 est plus générale qu'une hypothèse h_2 ($h1 \succeq h2$) si l'ensemble des exemples couverts par h_1 inclut l'ensemble des exemples couverts par h_2. La couverture des exemples dépend de la sémantique utilisée. Rappelons qu'en sémantique normale la *relation de couverture* entre les hypothèses et exemples est la conséquence logique ($\models$).

La relation de conséquence logique semble la plus intuitive pour hiérarchiser les clauses dans l'espace de recherche. Cependant, des résultats d'indécidabilité [Nienhuys-Cheng et de Wolf, 1996] en empêche l'utilisation dans le cas général même si l'on se restreint aux clauses de Horn (comme c'est souvent le cas en PLI). Plotkin [Plotkin, 1970] a donc introduit le quasi ordre nommé : θ-subsomption.

Définition 3.6 *Une clause c_1 θ-subsume une clause c_2 ($c_1 \succeq_\theta c_2$) si et seulement si il existe une substitution θ telle que $c_1\theta \subseteq c_2$. c_1 est une généralisation de c_2 (et c_2 une spécialisation de c_1) par θ-subsomption.*

Par exemple, la clause `p(a,b) <- r(a,b)` est θ-subsumée par la clause C = `p(Y1,Y2) <- r(Y2,Y1)`. On a bien $\{p(Y1,Y2), \neg r(Y2,Y1)\}\theta 1 \subseteq \{p(a,b), \neg r(a,b)\}$ avec $\theta 1 = Y1/a, Y2/b$. La clause `p(X1,X2) <- r(Y2,Y1), q(X1)` est également θ-subsumée par la clause C avec, par exemple, l'utilisation de la substitution $\theta 2 = Y1/X1, Y2/X2$.

Plusieurs clauses peuvent être équivalentes en termes de θ-subsomption. Plotkin a donc introduit la notion de *clause réduite*.

Définition 3.7 *La clause réduite r de c est la clause possédant le sous-ensemble minimal de littéraux de c telle que r soit toujours équivalente à c.*

Par exemple, les clauses `parents(X,Y) <- mere(X,Y)` et `parents(X,Y) <- mere(X,Y),mere(X,Z)` sont équivalentes en terme de θ-subsomption. La clause `parents(X,Y) <- mere(X,Y)` est une clause réduite.

La θ-subsomption induit un ordre plus faible que la conséquence logique ($C_1 \succeq_\theta C_2 \Rightarrow C_1 \models C_2$ et l'inverse n'est pas toujours vrai [Plotkin, 1971]) mais décidable et donc plus facilement manipulable bien qu'il soit encore NP-difficile

[Kapur et Narendran, 1986]. Il est cependant montré que ces deux ordres sont très proches et même équivalents si on exclut l'utilisation de clauses récursives.

L'ensemble des clauses réduites forme un *treillis*, i.e. deux clauses ont une unique plus petite généralisation (appelée *lgg* pour *least general generalisation* ou *moindre généralisé*) et une unique spécialisation la plus générale (*mgs* pour *most general specialization* ou *spécialisation maximale*).

3.1.4 La gestion du bruit

Dans des applications réelles d'apprentissage, les conditions expérimentales sont rarement aussi parfaites que ce qui est attendu en théorie. L'une des causes les plus évidentes de ces conditions d'apprentissage dégradées est l'imperfection des données d'apprentissage [Džeroski et Bratko, 1992]. Ces imperfections peuvent être dues à un manque de données ou à des erreurs dans les données : on parle alors de *bruit*.

En apprentissage supervisé, le bruit dans les données est principalement de trois natures différentes :

- il peut s'agir d'erreurs dans la classe attribuée à un exemple : c'est une erreur de supervision due à l'expert,
- il peut s'agir d'une erreur dans la description de l'exemple (assigner par exemple, une mauvaise valeur à l'un des arguments d'un prédicat servant à décrire les exemples),
- il peut s'agir d'une description erronée des connaissances dans la théorie du domaine T.

Les techniques d'apprentissage doivent donc s'adapter à ces conditions dégradées pour préserver au mieux leurs caractéristiques théoriques. Les conditions 1 et 2 de la définition 3.2 sont ainsi atténuées en utilisant un critère de qualité f_e telle que l'ensemble d'hypothèses H recherché soit optimal par rapport à f_e. f_e va ainsi permettre qu'une partie des exemples négatifs soit couverte et inversement qu'une partie des exemples positifs ne soit pas couverte.

3.1.5 Les biais d'apprentissage

En PLI, au contraire de la majorité des méthodes d'apprentissage, l'expressivité de la logique du premier ordre interdit de définir l'espace de recherche des hypothèses de manière exhaustive. L'espace de recherche est créé au fur et à mesure de l'induction de nouvelles hypothèses et le défi d'un *bon* système de PLI est de parcourir le moins de nœuds possible dans l'espace de recherche pour trouver une hypothèse satisfaisante.

La façon la plus naturelle d'*élaguer* l'espace de recherche est d'exploiter la relation de généralité entre les hypothèses dans cet espace. Par exemple, si l'on parcourt l'espace de recherche des hypothèses, des hypothèses les plus générales vers les plus spécifiques (voir Section 3.1.6 sur les stratégies de recherche) en cherchant à répondre au critère de complétude de la définition 3.2, il ne sert à rien d'évaluer les hypothèses plus spécifiques qu'une hypothèse qui ne couvre pas un exemple positif. Ces élagages ne sont cependant pas toujours suffisants pour réduire l'espace des hypothèses, d'où l'introduction de *biais*

d'apprentissage.

Le biais correspond à tout ce qui peut influencer le système d'apprentissage par PLI. Il y a trois principaux types de biais : *le biais de langage*, de type *déclaratif* [Nédellec *et al.*, 1996], qui définit le langage des hypothèses recherchées et donc l'espace de recherche que doit considérer le système (le *quoi*); *le biais de recherche* qui oriente le parcours de l'espace de recherche en terme d'élagage, de création de nouvelles hypothèses, etc. (le *comment*) : il comprend les biais de préférence (les hypothèses réellement intéressantes) et les biais de restriction (les hypothèses interdites); *le biais de validation* qui englobe les critères d'arrêt de la recherche. L'arrêt de la recherche peut intervenir lorsque les hypothèses apprises sont complètes et correctes vis-à-vis de l'ensemble des exemples ou lorsqu'un seuil a été franchi relativement aux critères du biais de recherche (on peut, par exemple, autoriser la couverture d'un certain nombre d'exemples négatifs etc.).

L'adéquation des biais d'un système de PLI conditionne le succès de l'apprentissage. Si les contraintes imposées par les biais sont trop fortes ou mal ciblées, la (ou les) hypothèse(s) décrivant le concept cible peuvent être exclues de l'espace de recherche ou tout chemin vers cette hypothèse interdit. Si le biais est trop faible, le système de PLI peut *errer* dans l'espace de recherche sans converger vers la solution attendue, même si elle appartient à l'espace de recherche. L'expression des biais adaptés à un problème spécifique d'apprentissage est donc une étape cruciale (et non triviale) lors de l'application d'un système d'apprentissage et particulièrement en PLI, où la taille de l'espace de recherche potentiel peut aboutir à des conséquences désastreuses pour l'apprentissage si le biais est trop faible.

Dans la suite, nous présentons les différentes techniques permettant de biaiser l'apprentissage par PLI.

3.1.5.1 Biais de préférence

Une fois l'espace structuré, on peut le parcourir de manière non informée (e.g. profondeur d'abord ou largeur d'abord) ou heuristique. Les heuristiques évaluent la qualité d'une clause en fonction du nombre d'exemples positifs et/ou du nombre d'exemples négatifs couverts par cette clause. Elles peuvent donc traiter les données imparfaites.

Quatre types d'estimateurs probabilistes sont employés afin d'évaluer la qualité d'une clause [Lavrač et Džeroski, 1993, Mitchell, 1997]. Soit $n(C)$ le nombre d'exemples couverts par la clause C, $n^{\oplus}(C)$ le nombre d'exemples positifs, $p_a(\oplus)$ la probabilité *a priori* de la classe positive estimée par sa fréquence relative $\frac{n^{\oplus}}{n}$ et m un paramètre choisi en fonction de la quantité des exemples bruités dans la base d'apprentissage (plus le bruit estimé est important, plus m est grand). Les estimateurs sont :

- la fréquence relative : $p(\oplus \mid C) = \frac{n^{\oplus}(C)}{n(C)}$
- l'estimateur de Laplace : $p(\oplus \mid C) = \frac{n^{\oplus}(C)+1}{n(C)+2}$
- le m-estimateur : $p(\oplus \mid C) = \frac{n^{\oplus}(C)+m*p_a(\oplus)}{n(C)+m}$
- l'entropie : soit S l'ensemble d'exemples couverts par la règle H, c le nombre de classes d'exemples (pour l'instant $c = 2$: la classe positive et la classe négative) et

p_i la proportion d'exemples de S appartenant à la $i^{\text{ème}}$ classe, $(-Entropie(S)) = \sum_{i=1}^{c} p_i log_2 p_i$.

Les trois premiers estimateurs donnent pratiquement le même résultat en présence d'un grand nombre d'exemples dans la base d'apprentissage. Plus la valeur de l'estimation se rapproche de 1, meilleure est la clause. L'estimateur de Laplace, basé sur une loi de succession, est plus fiable que la fréquence relative lorsque le nombre d'exemples est faible. Cependant, l'estimation de Laplace ne s'applique qu'à un problème à deux classes (apprentissage booléen) et suppose que les deux classes ont une distribution de probabilité a priori uniforme. Malheureusement, cette supposition est rarement satisfaite en pratique. Le m-estimateur évite le problème des deux autres heuristiques en tenant compte de la probabilité a priori des classes : c'est une généralisation des deux premières heuristiques. Quand $m = 0$, il est équivalent à la fréquence relative. Lorsque $p_a(\oplus) = 0.5$ et $m = 2$, c'est exactement l'estimateur de Laplace. Notons que m peut augmenter indéfiniment, sa valeur de départ est subjective.

L'entropie mesure l'homogénéité d'un ensemble d'exemples vis à vis d'un ensemble de classes que l'on cherche à caractériser. Si tous les exemples appartiennent à une seule classe, l'entropie est égale à 0, si tous les exemples sont uniformément répartis pour chacune des classes, l'entropie est égale à $log_2(c)$. Ce nombre correspond au nombre de bits nécessaires pour coder le nombre de classes (chaque exemple a une probabilité équivalente d'appartenir à chacune des classes). La négation de l'entropie est choisie ici pour que les meilleures hypothèses aient la meilleure entropie. Le système FOIL [Quinlan et Cameron-Jones, 1993] utilise une adaptation de la mesure d'entropie pour mesurer un gain fourni par l'addition d'un nouveau littéral dans une hypothèse.

3.1.5.2 Biais déclaratifs

Il existe deux types de biais déclaratifs, le biais *syntaxique* et le biais *sémantique*.

- Le biais syntaxique est constitué d'un ensemble de contraintes sur la forme des hypothèses : leur taille, le nombre maximal de variables dans chaque hypothèse, le nombre maximal de littéraux, les prédicats autorisés dans les hypothèses permettant de rendre l'espace de recherche fini. Certains systèmes de PLI utilisent des langages dédiés pour la description des biais comme DLAB pour les systèmes CLAUDIEN ou ICL (détaillé en section 3.1.8.4) ou les schémas d'hypothèses de MOBAL [Kietz et Wrobel, 1992]. Ces langages permettent de définir en intention l'ensemble des clauses syntaxiquement valides de l'espace de recherche.
- Les biais sémantiques permettent de spécifier le type des arguments et leur mode (entrée/sortie) (cf. ALEPH [Srinivasan, 2003]), de donner des contraintes sur la structuration des littéraux dans les clauses (des arguments en mode *sortie* dans la tête des clauses doivent apparaître avec le même mode dans le corps de la clause), de spécifier des clauses interdites etc.

Ces différents biais sont combinés à des stratégies de recherche élaborées pour parcourir efficacement l'espace de recherche des hypothèses. Nous présentons dans la suite, différentes stratégies de recherche utilisées en PLI.

3.1.6 Algorithme générique et stratégies de recherche en PLI

Les systèmes de PLI existants utilisent différents algorithmes pour parcourir l'espace de recherche. L'algorithme 2 présente un cadre générique.

Algorithme 2 $H :=$ Initialiser
Répéter *tant que* critère_arrêt$(H) \neq$ *satisfait*
Retirer h *de* H
Choisir *des règles d'inférence* $r_1, \ldots, r_k \in R$ *à appliquer à* h
Appliquer les règles $r_1, \ldots, r_k \in R$ *à* h *pour obtenir* $h_1, \ldots, h_n$
Ajouter $h_1, \ldots, h_n$ *à* H
Élaguer H
Fin Répéter

- Initialiser *permet de définir le choix de la* graine *de l'apprentissage. Il s'agit de la première hypothèse à considérer dans l'espace de recherche. Ce choix dépend de la stratégie de recherche employée.*
- R *est l'ensemble des règles d'inférence applicables selon la stratégie employée.*
- Retirer *influence la stratégie de recherche qui peut être effectuée en largeur d'abord, en profondeur d'abord ou selon une autre stratégie.*
- critère_arrêt *donne la condition d'arrêt de l'algorithme.*

L'espace de recherche est parcouru grâce à des opérations dites de *raffinement* [Shapiro, 1983] (de généralisation et/ou de spécialisation) utilisant les relations de subsomption définies précédemment.

Définition 3.8 (Règle d'inférence déductive) *Une règle d'inférence déductive* $r \in R$ *fait correspondre une conjonction de clauses* G *à une conjonction de clauses* S *telle que* $G \models S$*;* r *est appelée* règle de spécialisation.

Définition 3.9 (Règle d'inférence inductive) *Une règle d'inférence inductive* $r \in R$ *fait correspondre une conjonction de clauses* S *à une conjonction de clauses* G *telle que* $G \models S$*;* r *est appelée* règle de généralisation.

Les algorithmes étudiés peuvent permettre de considérer les hypothèses des plus générales aux plus spécifiques, on parle de recherche *descendante* (ou *top-down*), des plus spécifiques aux plus générales, on parle de recherche *ascendante* (ou *bottom-up*) ou un mélange des deux, on parle alors de recherche *mixte*. Les stratégies de recherche descendante utilisent des règles d'inférence déductive, les stratégies de recherche ascendante utilisent des règles d'inférence inductive.

On peut aussi distinguer les algorithmes selon qu'ils sont de type *générer et tester* (*generate and test*) ou guidés par les exemples (*example-driven*). Les algorithmes sont de type *générer et tester* si l'espace de recherche est limité à toutes les hypothèses syntaxiquement légales. Les hypothèses générées peuvent ainsi couvrir plusieurs exemples. Le système $\mathcal{HAIKU}$ [Nédellec et Rouveirol, 1994] représente un cadre unifié pour ce type d'algorithme puisqu'il permet de s'identifier à tous les systèmes existants suivant

le type de paramétrage choisi. Les algorithmes sont au contraire dits guidés par les exemples si les exemples contraignent tour à tour individuellement la génération des hypothèses comme c'est le cas des algorithmes de la famille AQ [Michalski *et al.*, 1986] ou de CIGOL [Muggleton et Buntine, 1988a] et ses successeurs.

Dans la suite de cette section, nous présentons les différentes stratégies de recherche et en particulier les opérateurs de raffinements utilisés.

3.1.6.1 Recherche descendante

Dans le cas des algorithmes de recherche descendante, l'hypothèse de départ est la clause la plus générale (la clause *true*). Les hypothèses sont spécialisées en appliquant des règles d'inférence déductive pour diminuer le nombre d'exemples négatifs couverts tout en conservant une partie de la couverture des exemples positifs.

Les algorithmes les plus utilisés pour parcourir l'espace de recherche de la clause la plus générale vers des clauses plus spécifiques sont des adaptations à la logique du premier ordre de l'algorithme de *couverture séquentielle* utilisé pour induire des hypothèses dans le cas propositionnel [Mitchell, 1997]. L'algorithme recherche une hypothèse à la fois et retire de E^+ les exemples positifs couverts par cette hypothèse. Une autre hypothèse est alors cherchée pour couvrir le reste des exemples positifs. Lorsque l'espace de recherche est hiérarchisé par la relation de θ-subsomption, les règles de spécialisation utilisées sont :

- ajouter des littéraux à la clause de départ,
- appliquer une substitution et ainsi transformer des variables de la clause de départ en constantes ou, unifier plusieurs variables.
- introduire des symboles fonctionnels (par exemple X devient `[Y|Z]`)

La majeure partie des systèmes de PLI utilisent cet algorithme sous la forme *générer et tester* (cf. FOIL [Quinlan, 1990, Quinlan et Cameron-Jones, 1993]). En effet, lorsque les données sont bruitées, le système peut poursuivre la couverture séquentielle des hypothèses jusqu'à ce qu'un compromis soit trouvé entre la précision, la couverture et la complexité des hypothèses.

Notons que l'opérateur d'ajout de littéraux permet d'explorer un espace de recherche infini puisque le nombre de littéraux ajoutables est potentiellement infini. Pour contrôler la recherche descendante, il est donc nécessaire d'ajouter un biais syntaxique sur la longueur maximale des clauses ou de borner l'espace en utilisant un exemple positif et rendre ainsi l'espace fini.

Claudien [De Raedt et Bruynooghe, 1993] est le premier système efficace de PLI fonctionnant avec une sémantique non monotone permettant de dériver des théories clausales complètes à partir de bases de données. Le même type d'algorithme avec une sémantique non monotone est également utilisé dans Tilde [Blockeel et De Raedt, 1998] qui adapte un algorithme d'apprentissage d'arbres de décision à l'induction en logique du premier ordre et ICL [De Raedt et Van Laer, 1995] qui adapte cette fois un algorithme attribut-valeur. Le système MOBAL-BLIP [Kietz et Wrobel, 1992] utilise en sémantique normale une variante de la θ-subsomption compatible avec ses modèles de hypothèses (voir section 3.1.5) pour générer les hypo-

thèses des plus générales aux plus spécifiques.

3.1.6.2 Recherche Ascendante

Dans le cas des algorithmes de recherche ascendante, le point de départ de la recherche est un ensemble d'hypothèses sous ensemble des exemples positifs (souvent des paires d'exemples) que l'on cherche à généraliser en une clause en appliquant des règles d'inférence inductive.

À la base de la plupart des algorithmes ascendants se trouvent les moindres généralisés correspondant à la relation de θ-subsomption (cf. *lgg* Section 3.1.3 et [Plotkin, 1970] pour la construction de la *lgg* de deux clauses). En pratique en PLI, la notion de *lgg* est adaptée pour prendre en compte la base T de connaissance *a priori*. Cette adaptation s'appelle *rlgg* pour *relative least general generalisation* [Plotkin, 1971]. La *rlgg* de deux clauses c_1 et c_2 est définie définit comme suit : $rlgg(c_1, c_2) = lgg(c_1 \Leftarrow \mathcal{M}(T), c_2 \Leftarrow \mathcal{M}(T))$ où $\mathcal{M}(T)$ est un modèle de T. Intuitivement, la construction de la *rlgg* est basée sur la notion de saturation d'une clause c par une théorie T. Saturer une clause consiste à ajouter à cette clause tous les littéraux impliqués par une théorie donnée (dans notre cas il s'agit de la théorie du domaine). Par exemple :

Soit la clause c : $p(a,b){:-}q(a,b), r(a), w(b).$,
soit la théorie $T = \{m(X,Y){:-}q(X,Y), r(Y)., n(X,Y){:-}r(X), m(X,Y).\}$

Notons que relativement à T, on peut déduire $m(a,b)$ de $q(a,b), r(a)$ et également $n(a,b)$ de $m(a,b), r(a)$. On peut donc saturer la clause c comme suit : c : $p(a,b){:-}q(a,b), r(a), w(b), m(a,b), n(a,b)$.

Cette approche est utilisée dans un logiciel pionnier de la PLI, GOLEM [Muggleton et Feng, 1990] et dans ses successeurs. L'utilisation des *rlgg* pose cependant le problème de la construction des modèles de T (et plus particulièrement du plus petit modèle de Herbrand) qui peuvent êtres infinis. La *rlgg* de deux clauses est dans ce cas également infinie. De plus, même dans le cas où un sous-ensemble fini d'un modèle T est utilisé, la *rlgg* peut être de taille très importante et croître exponentiellement avec le nombre d'exemples. En effet, l'algorithme calcule d'abord toutes les *rlgg* des différentes paires possibles d'exemples positifs puis conserve la meilleure *rlgg* R en terme de couverture des exemples et recalcule récursivement la *rlgg* de R et d'un certain nombre d'autres exemples.

Une autre approche utilisant des règles d'inférence inductive pour explorer l'espace de recherche des clauses plus spécifiques vers les plus générales repose sur l'idée qu'il est possible d'inverser le principe de résolution de Robinson. L'ajout de nouveaux prédicats dans les hypothèses se fait par *inversion de la SLD-résolution* utilisée dans les interpréteurs PROLOG (cf. Annexe B). La règle d'intra_construction par exemple (cf. Figure 3.1) est une des règles de résolution inversée (cf. [Muggleton et Buntine, 1988b] pour la totalité des règles) permettant de générer de nouveaux prédicats. CIGOL

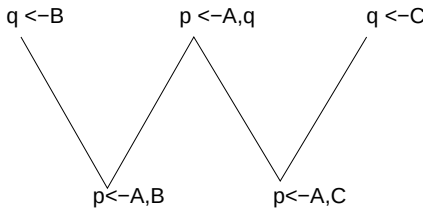


FIG. 3.1 – Opérateur W : intra_construction

[Muggleton et Buntine, 1988a] est l'un des premiers systèmes de PLI à avoir utilisé cette méthode. Des successeurs se sont attachés à restaurer la complétude de la résolution inverse (l'absorption par exemple donne seulement un résultat possible) [Rouveirol et Puget, 1990], à renverser en un pas plusieurs étapes de résolution ou à perfectionner la gestion des clauses contenant des fonctions.

3.1.6.3 Recherche mixte

Des stratégies hybrides combinant des approches *bottom up* et *top down* peuvent êtres utilisées pour trouver plus efficacement les hypothèses intéressantes dans l'espace de recherche.

Par exemple, l'approche retenue dans PROGOL [Muggleton, 1995] combine successivement et dans cet ordre, une approche ascendante et une approche descendante. Dans l'approche ascendante, l'inférence inductive ne s'appuie pas sur la résolution inverse présentée précédemment mais sur le principe d'*inversion de l'implication* (ou inversion de la conséquence logique). Le principe de l'inversion de l'implication repose sur la constatation que la condition $T \wedge h \models e$ de [Džeroski et Lavrač, 2001] est équivalente à $T \wedge \overline{e} \models \overline{h}$ en notant $\overline{x}$ la négation de x. Puisque h et e sont des clauses, leur négation sont des clauses unitaires skolémisées (cf. Annexe B), c'est-à-dire des programmes logiques clos. Soit $\overline{\bot}$ la conjonction (potentiellement infinie) de littéraux sans variables, vraie pour tous les modèles de $T \wedge \overline{e}$. Puisque $\overline{h}$ doit être vraie dans tous les modèles de $T \wedge \overline{e}$, $\overline{\bot}$ doit contenir $\overline{h}$. Ainsi, $T \wedge \overline{e} \models \overline{\bot} \models \overline{h}$. Soit, pour tout h, on a $h \models \bot$. La borne $\bot$ est en général construite à partir d'un exemple positif et sert de *base de littéraux* ajoutables : en recherche descendante, un littéral ajouté afin de spécialiser une hypothèse non satisfaisante sert nécessairement à généraliser cette borne inférieure. Le nombre d'hypothèses plus générales que $\bot$ étant potentiellement infini, PROGOL et ses successeurs utilisent un ensemble de biais syntaxiques (limitation de la taille des hypothèses) et surtout des biais sémantiques (déclaration de modes) pour définir le langage des hypothèses. L'espace de recherche est ensuite parcouru de manière descendante par un algorithme de type $A*$.

Les stratégies descendantes ont la faveur des développeurs des systèmes de PLI les plus récents sans doute parce que les clauses les plus courtes et donc les plus simples sont plus vite atteintes par une stratégie descendante que par une stratégie ascendante.

Dans la suite, nous détaillons l'algorithme d'un des systèmes successeurs de PRO-

GOL : ALEPH [Srinivasan, 2003].

3.1.7 ALEPH : un système de PLI multiforme

ALEPH (*A Learning Engine for Proposing Hypotheses*) [Srinivasan, 2003] est un système de PLI à sémantique normale (cf. Définition 3.1.2). Ce système, successeur de PROGOL, a à la base été conçu dans un but universitaire pour appréhender la notion d'*inversion de l'implication* [Muggleton, 1995]. Il a ensuite été amélioré de manière à pouvoir s'identifier à un grand nombre de systèmes existants ayant des stratégies et des algorithmes de recherche variés : PROGOL, FOIL, FORS [Karalic et Bratko, 1997], MIDOS [Wrobel, 1997], TILDE [Blockeel et De Raedt, 1998], et WARMR [Dehaspe, 1999].

3.1.7.1 Algorithme d'ALEPH

À l'algorithme de base (cf. Algorithme 3) pouvant être défini en seulement 4 étapes s'ajoutent un très grand nombre de paramètres permettant de modifier le choix de la meilleure clause, la construction de la clause la plus spécifique de l'espace de recherche (la *bottom-clause* : $\perp$), les stratégies de recherche, les fonctions d'évaluation, les opérateurs de raffinement etc.

Algorithme 3 (Algorithme de base d'Aleph)
Répéter *tant que* $E^+ \neq \emptyset$

1. *Choisir aléatoirement* e^+ *dans* E^+*;*
2. **Étape de saturation** *: construire la clause* $\perp$ *la plus spécifique de l'espace de recherche telle que* $T \wedge e+ \models \perp$*. Cette clause est généralement une clause définie contenant un grand nombre de littéraux.*
3. **Étape de réduction** *: parcourir l'espace de recherche pour trouver une hypothèse correcte plus générale que* $\perp$*.*
4. *La clause ayant obtenu le meilleur score est ajoutée à la théorie courante et tous les exemples déjà couverts par cette théorie sont retirés de l'ensemble des exemples.*

Fin répéter

Par défaut, l'algorithme d'ALEPH s'apparente à celui de PROGOL, c'est un algorithme de type *générer et tester* mixte. La recherche bornée par la *bottom clause* $\perp$ s'effectue par couverture séquentielle de la clause la plus générale vers les plus spécifiques. Notons que dans l'étape 3, réduire l'espace de recherche aux sous ensembles de la *bottom clause* ne permet pas de prendre en compte toutes les clauses plus générales. En pratique, ALEPH utilise un algorithme *branch-and-bound* qui permet une énumération *intelligente* des clauses candidates.

3.1.7.2 Biais d'apprentissage

Des déclarations de mode et de type permettent de définir des biais sémantiques. Les premiers permettent de préciser le fonctionnement attendu des prédicats utilisés en termes de variables d'entrée et de sortie. Il est nécessaire de spécifier le prédicat à employer en tête de clause (en utilisant la directive *modeh*) et les prédicats qu'il est possible d'utiliser dans le corps des clauses (en utilisant la directive *modeb*). Cela permet, par exemple, de se restreindre aux clauses dont les prédicats utilisent en variables d'entrée les variables de sortie des prédicats précédents. Les déclarations de type indiquent de quel type sont les variables apparaissant dans les prédicats des hypothèses recherchées. Ainsi, si dans une hypothèse, un prédicat utilise une variable d'un certain type t, les autres prédicats ne peuvent utiliser cette même variable que s'ils sont déclarés comme pouvant fonctionner avec des variables de type t.

La construction de la bottom clause dans l'étape de *saturation* de l'algorithme 3 dépend des déclarations de mode données pour chaque prédicat autorisé dans les hypothèses. Le détail de la construction est donné en Annexe D.1 de [Muggleton, 1995].

3.1.8 ICL : apprentissage par interprétations

L'article *Inductive Constraint Logic* [De Raedt et Van Laer, 1995] décrit la sémantique du système ICL ainsi que l'algorithme utilisé pour induire une théorie sous forme normale conjonctive (CNF) ou sous forme normale disjonctive (DNF) à partir d'exemples décrits comme des interprétations partielles (cf. définition 3.5 section 3.1.2.2).

3.1.8.1 Algorithmes

L'objectif d'ICL est de concilier l'efficacité de l'apprentissage attribut-valeur avec l'expressivité de la logique du premier ordre. Les systèmes d'apprentissage attribut-valeur apprennent des théories sous forme normale disjonctive (DNF) alors que les systèmes d'apprentissage par PLI cherchent classiquement à apprendre des programmes logiques sous forme normale conjonctive (CNF). Les auteurs ont tout d'abord cherché à créer un nouveau système de PLI apprenant des CNF en adaptant un algorithme d'apprentissage attribut-valeur nommé CN2 [Clark et Niblett, 1989, Clark et Boswell, 1991].

CN2, un algorithme propositionnel Dans CN2, les exemples sont décrits par un nombre d'attributs fixe ayant chacun une valeur donnée. Chaque exemple possède un attribut particulier qui détermine sa classe. Les hypothèses apprises sont de la forme `classe = Class if Condition` où `Condition` est une disjonction de conjonctions de paires attribut-valeur (forme DNF). Dans ce formalisme, une règle *couvre* un exemple si la condition de la règle vérifie l'exemple. Pour une classe `Class` donnée, les exemples positifs sont ceux pour lesquels la valeur `Class` est associée à l'attribut `classe` et les exemples négatifs sont tous les autres exemples. Dans la condition de la règle, chaque

conjonction de paires attribut-valeur couvre tous les exemples positifs et aucun exemple négatif.

L'algorithme de couverture de CN2 est l'algorithme de couverture séquentielle sur les exemples positifs présenté en section 3.1.6.1. L'apprentissage séquentiel des règles pour chaque classe c se fait par un parcours de l'espace de recherche en *faisceau* (*beam*) (cf. Algorithme 4).

Algorithme 4 (CN2 : Algorithme de recherche en faisceau d'une règle)

```
Trouver-une-règle{Pos, Neg,c};
  1. rpg := classe=c si vrai;
  2. Beam := {rpg};
  3. MeilleureRègle := {}
  4. tant que Beam est non vide faire
       (a) NewBeam := {}
       (b) Pour chaque règle r dans Beam faire
           Pour tout raffinement Ref de r faire
          i.  Si Ref est meilleur que MeilleureRègle
              et Ref est statistiquement significative alors
                  - MeilleureRegle := Ref;
          ii. Si Ref ne doit pas être élagué alors
                  - ajouter Ref à NewBeam;
                  - si la taille de NewBeam > MaxBeamSize alors
                      enlever la plus mauvaise règle de NewBeam;
       (c) Beam := NewBeam;
  5. retourner MeilleureRègle;
```

Dans l'algorithme 4, la recherche commence avec la règle la plus générale (*rpg*) de l'espace de recherche (`classe=c si vrai`) qui couvre tous les exemples. Durant la recherche, CN2 conserve un faisceau des règles candidates et la meilleure règle trouvée jusqu'à présent (*MeilleureRègle*). À chaque étape dans la recherche par faisceau, tous les raffinements Ref des règles dans $Beam$ sont considérés et évalués. Puis, selon le résultat de l'évaluation, la règle issue de Ref devient la $MeilleureRegle$ et/ou Ref est ajoutée à $NewBeam$ (seules les *MaxBeamSize* meilleures candidates restent dans le faisceau).

CN2 utilise une fonction heuristique d'évaluation de la qualité de la meilleure règle basée sur un estimateur de Laplace (cf. Section 3.1.5 sur les biais de préférence). Cette mesure est utilisée pour diriger la recherche (comme seules les *MaxBeamSize* règles sont présentes dans le faisceau) et pour déterminer la meilleure des règles trouvées dans le faisceau. L'espace de recherche peut être également réduit par élagage. En effet, la règle r peut être élaguée s'il n'y a aucun raffinement de r qui puisse donner un meilleur résultat que la meilleure règle du moment (cf. l'introduction de la Section 3.1.5). On peut arrêter le raffinement d'une règle lorsqu'il n'est plus possible qu'elle soit statistiquement significative, c'est-à-dire, lorsqu'elle ne couvre plus assez d'exemples positifs. S'il existe

encore des exemples positifs possiblement couverts par les raffinements d'une règle non statistiquement significative, il sont considérés, pour cette règle, comme du bruit.

L'algorithme d'ICL L'algorithme d'ICL est dérivé de celui de CN2 de manière à être adapté à la logique du premier ordre [Van Laer, 2002]. Le pouvoir d'expression de la logique du premier ordre étant beaucoup plus important que celui du langage attribut-valeur utilisé dans CN2, Van Laer [Van Laer et De Raedt, 2001] a inclus un biais déclaratif permettant de restreindre l'espace de recherche. Le langage de biais utilisé est le même que celui de CLAUDIEN [De Raedt et Bruynooghe, 1993]. La stratégie de recherche est descendante en faisceau. L'opérateur de raffinement est celui défini en Section 3.1.6.1. Les exemples sont des ensembles de faits clos (vu comme des interprétations de Herbrand de la théorie cible puisque les faits représentent tous les atomes qui sont vrais pour un exemple donné, les faits non exprimés dans l'exemple sont considérés faux) et non pas des faits ou clauses comme pour la sémantique normale.

ICL permet d'apprendre des théories sous forme CNF ou sous forme DNF. L'algorithme DNF est l'adaptation directe de l'algorithme de CN2 en prenant en compte la théorie du domaine T. Pour obtenir un algorithme qui apprend directement une théorie sous forme CNF, il faut inverser le rôle des exemples positifs et des exemples négatifs dans l'algorithme de CN2 notamment lors de l'algorithme de couverture séquentielle permettant de trouver une règle discriminante (cf. Algorithme 5) : la couverture se fait donc sur les exemples négatifs. Van Laer montre dans [Van Laer, 2002] que les règles obtenues en CNF pour une classe donnée sont les mêmes (sous certaines conditions cf. Section 2.4.4 de [Van Laer, 2002]) que celles obtenues en prenant la négation du résultat obtenu avec un algorithme construisant des DNF sur la classe complémentaire ($\neg\exists(A \vee B) \Leftrightarrow \forall(\neg A \wedge \neg B)$). Dans l'algorithme CNF présenté ci dessous, chaque clause appartenant à la théorie cible doit couvrir tous les exemples positifs et exclure certains exemples négatifs. L'ensemble des clauses apprises permet d'exclure la totalité des exemples négatifs.

Algorithme 5 (Algorithme CNF de couverture d'ICL)

```
Inductive-Contraint-Logic(Pos,Neg,T,c);
  1. Initialiser H := {}
  2. répéter tantque (la meilleure clause h n'est pas trouvée)
                     ou (Neg non vide)
       (a) h := Trouver-une-clause(Pos,Neg,T,c)
       (b) si la meilleure clause h est trouvée alors
             i. ajouter h à H
             ii. enlever de Neg toutes les interprétations qui
                 sont fausses pour h
  3. retourner H
```

L'algorithme de recherche séquentielle des clauses `Trouver-une-clause(Pos,Neg,T,c)` est adapté de l'algorithme de recherche `Trouver-une-règle(Pos,Neg,c)` de CN2 de manière à prendre en compte les

connaissances du domaine T. La recherche commence à partir de la clause la plus générale du treillis de raffinement ($false \leftarrow true$) qui ne couvre aucun exemple (la relation de couverture est définie en section 3.5). ICL utilise deux fonctions heuristiques d'évaluation : à l'estimateur de Laplace est ajouté le *m_estimateur* (cf. Section 3.1.5 sur les biais de préférences).

Notons qu'ICL, comme CN2 n'apprend que des concepts binaires, des classifieurs. Un exemple est soit accepté par une règle soit rejeté. Ce résultat est moins général que celui des systèmes de PLI classiques, qui apprennent des théories clausales, mais l'apprentissage est plus efficace.

3.1.8.2 Apprentissage multiclasse avec ICL

Dans le cas des classes multiples (le nombre de classes m est supérieur à 2), ICL apprend m théories, une pour chaque classe avec l'algorithme présenté précédemment. Dans le cas contraire, on ne pourrait différencier une classe (celle reconnue par la théorie) de toutes les autres classes. Pour chaque classe, les exemples positifs sont les modèles de la théorie associée à la classe et les exemples négatifs sont toutes les autres interprétations. Le système apprend d'abord une théorie $H_i = C_{i,1} \vee \ldots \vee C_{i,ni}$ sous forme DNF pour chaque classe c_i, puis fusionne ces m théories séparées pour construire la théorie $H_{multi} = C_{1,1} \vee \ldots \vee C_{1,n1} \vee \ldots \vee C_{m,1} \vee \ldots \vee C_{m,nm}$.

Si l'apprentissage multiclasse ne pose pas de problèmes particuliers, la difficulté est tout autre lorsqu'il s'agit de classer un nouvel exemple. Lorsque que le problème est réduit à deux classes, il est possible de classer un nouvel exemple si l'exemple vérifie la définition $e \models H_c$. Dans le cas CNF cela revient à dire que toutes les clauses de H_c couvrent l'exemple et dans le cas DNF, une des clauses doit couvrir l'exemple. ICL utilise une méthode similaire à celle de CN2 [Clark et Niblett, 1989] pour traiter le problème des classes multiples [Van Laer *et al.*, 1996, Van Laer *et al.*, 1997]. À chaque règle $C_{i,j}, 1 \leq i \leq m$ est associé un vecteur $V_{i,j}$ de dimension m dont le $k^{\text{ème}}$ élément représente le nombre d'exemples d'apprentissage appartenant à la classe c_k couverts par la règle $C_{i,j}$. $V_{i,j}$ décrit la distribution des exemples de la base d'apprentissage dans chacune des classes.

Un nouvel exemple e peut être classé par H_{multi} en suivant les étapes suivantes :

1. on initialise un vecteur V tel que $\forall k \in [1, \ldots, m], v_k = 0$;
2. pour chaque règle $C_{i,j}$ telle que $C_{i,j}$ couvre e, $V = V + V_{i,j}$;
3. si $\forall k \in [1, \ldots, m], v_k = 0$, i.e. aucune règle ne couvre l'exemple e, on retourne la classe par défaut définie comme étant la classe couvrant le plus grand nombre d'exemples dans la base d'apprentissage. Sinon, on retourne la classe k telle que v_k est l'élément maximal du vecteur V.

3.1.8.3 Sémantique opérationnelle

Dans ce mémoire, nous utilisons une nouvelle définition (cf. Définition 3.10) de l'apprentissage par interprétation plus adaptée au problème multiclasse d'ICL. Elle s'inspire de la formulation de Blockeel et al. [Blockeel, 1998] pour un apprentissage

multiclasse à partir d'interprétations. H_c est un ensemble d'hypothèses décrites sur le langage L_H et représentées sous forme de théorie clausale décrivant une classe $c \in C$ à partir d'un ensemble d'exemples E. Les exemples sont des ensembles de faits PROLOG clos étiquetés par la classe qu'ils représentent et définis sur le langage L_E. L'ensemble des faits et son étiquette constitue une interprétation. L'apprentissage peut également utiliser l'ensemble T (théorie du domaine) sous forme de clauses PROLOG définies sur le langage $L = L_E \cup L_H$.

Définition 3.10 (Apprentissage multiclasse)
Un problème de PLI multiclasse défini par un tuple $< L, E, T, C >$ *est tel que* $\forall (e,c) \in E, \forall c' \in C - \{c\}$*, on cherche une théorie clausale* H_c *qui satisfait les conditions suivantes :*

- $H_c \wedge e \wedge T \models c$ *(H_c couvre (e,c))*
- $H_c \wedge e \wedge T \not\models c'$ *(H_c est discriminante)*

3.1.8.4 DLAB : un langage de biais déclaratif

DLAB [Dehaspe et De Raedt, 1996, De Raedt et Dehaspe, 1997] est un langage de biais syntaxique initialement créé pour le système CLAUDIEN [De Raedt et Bruynooghe, 1993]. Ce langage offre un moyen efficace de définir l'espace des hypothèses. Il utilise une grammaire formelle constituée d'un ensemble de modèles (*templates*), permettant de décrire des espaces d'hypothèses de manière implicite. Chaque modèle est de la forme $HeadTemplate \leftarrow BodyTemplate$ où la tête $HeadTemplate$ et le corps $BodyTemplate$ sont des termes DLAB. Un terme DLAB spécifie les littéraux et leurs arguments utilisables dans les clauses générées. Un terme est : soit un atome; soit une formule sous la forme $Min - Max : L$, où L est une liste de termes DLAB et où Min et Max sont deux entiers tels que : $0 \leq Min \leq Max \leq taille(L)$. Le déploiement d'une grammaire DLAB consiste à sélectionner de façon récursive tous les sous-ensembles de L dont la taille est comprise entre Min et Max.

L'exemple 8 tiré de [Nédellec *et al.*, 1996] illustre la manière de se servir des termes DLAB et donne une première idée du pouvoir d'expression de ce formalisme relativement simple.

Exemple 8 *Soit un terme* DLAB $Min - Max : L$*. Différentes valeurs des variables* Min *et* Max *devant une grammaire* $DGRAM_i$ *permettent de spécifier différents ensembles de clauses* $\mathcal{L}_i$ *:*

- *Tous les sous-ensembles :* $Min = 0, Max = taille(L)$
  (spécification : 0-len)
  $DGRAM_1$ = `{0-len:[humain(X)] <- 0-len:[femme(X),homme(X)]}`

$$\mathcal{L}_1 = \begin{cases} \leftarrow \\ humain(X) \leftarrow \\ \leftarrow femme(X) \\ \leftarrow homme(X) \\ \leftarrow femme(X) \wedge homme(X) \\ humain(X) \leftarrow femme(X) \\ humain(X) \leftarrow homme(X) \\ humain(X) \leftarrow femme(X) \wedge homme(X) \end{cases}$$

– *Tous les sous-ensembles non vides :* $Min = 1, Max = taille(L)$ *(spécification : 1-len)*
$DGRAM_2 =$ `{0-len: [humain(X)] <- 1-len:[femme(X), homme(X)]}`

$$\mathcal{L}_2 = \begin{cases} \leftarrow femme(X) \\ \leftarrow homme(X) \\ \leftarrow femme(X) \wedge homme(X) \\ humain(X) \leftarrow femme(X) \\ humain(X) \leftarrow homme(X) \\ humain(X) \leftarrow femme(X) \wedge homme(X) \end{cases}$$

– *Le Ou exclusif :* $Min = Max = 1$ *(spécification : 1-1)*
$DGRAM_3 =$ `{0-1: [humain(X)] <- 1-1 : [femme(X), homme(X)]}`

$$\mathcal{L}_3 = \begin{cases} \leftarrow femme(X) \\ \leftarrow homme(X) \\ humain(X) \leftarrow femme(X) \\ humain(X) \leftarrow homme(X) \end{cases}$$

– *La combinaison des différentes spécifications :* $Min = Max = taille(L)$ *(spécification : len-len)*
$DGRAM_4 =$

```
{1-1: [humain(X)] <-
              0-2 : [len-len: [femme(X), est_fille(X)],
                     len-len: [homme(X), est_fils(X)]]}
```

$$\mathcal{L}_4 = \begin{cases} humain(X) \leftarrow \\ humain(X) \leftarrow femme(X) \wedge est\_fille(X) \\ humain(X) \leftarrow homme(X) \wedge est\_fils(X) \\ humain(X) \leftarrow femme(X) \wedge est\_fille(X) \wedge homme(X) \wedge est\_fils(X) \end{cases}$$

Dans la suite de ce chapitre, nous présentons les résultats obtenus avec les logiciels ICL et Aleph pour la caractérisation d'arythmies cardiaques à partir de données provenant d'électrocardiogrammes d'une part, et de mesure de pression artérielle d'autre part.

3.2 Acquisition de règles monosources caractérisant des arythmies cardiaques : deux approches

Cette partie présente l'acquisition de règles caractérisant des arythmies cardiaques à partir de données provenant de deux dérivations d'un électrocardiogramme (la voie I et une voie précordiale V) et d'une mesure de pression artérielle invasive (cf. Chapitre 1 pour le détail de ces signaux) provenant de la base MIMIC [Moody et Mark, 1996].

Des règles caractérisant des arythmies basées sur les 12 dérivations de l'ECG (cf. Section 1.1.3) pourraient être obtenues directement auprès des cardiologues. Cependant, les techniques d'apprentissage se montrent d'un grand intérêt, lorsqu'il s'agit :

- de faire gagner du temps aux experts qui peuvent avoir des difficultés à exprimer certaines règles de décision dans un formalisme adapté (les chroniques);
- de formaliser des règles adaptées à chaque patient;
- de paramétrer la complexité des règles devant être apprises selon que l'on cherche à obtenir des chroniques facilement lisibles par le spécialiste, des chroniques minimales pour des raisons d'efficacité lors de la reconnaissance ou encore des chroniques robustes privilégiant les éléments les plus faciles à détecter (par exemple pour l'ECG, les complexes QRS). Cette façon de penser est nouvelle en cardiologie et sort du schéma classique de représentation des arythmies à partir de règles proposées par un expert;
- d'obtenir des règles discriminantes (ie. qui permettent de différencier une arythmie d'une autre) et non pas des règles simplement descriptives.

La caractérisation des arythmies cardiaques à partir de données provenant du signal de pression artérielle est en revanche une nouveauté pour les experts et rend donc l'approche par apprentissage automatique d'autant plus intéressante.

Les résultats présentés dans cette partie font suite aux travaux de Quiniou *et al.* [Quiniou *et al.*, 2001]. Ces résultats ont permis la caractérisation d'arythmies cardiaques avec ICL à partir de données provenant de signaux ECG de la base MIT-BIH et pour des arythmies différentes de celles présentées dans ce chapitre. Les premiers résultats d'apprentissage à partir de données de pression artérielle proviennent des travaux de DEA de Céline Fildier [Fildier, 2001] avec le système ALEPH. La majeure partie des données MIMIC [Moody et Mark, 1996] utilisées dans cette étude a été annotée par Isabelle Maguy [Maguy, 2002] lors de son stage de DESS. De nouveaux exemples notamment pour la fibrillation auriculaire (*fa*) et la tachycardie supra-ventriculaire (*tsv*) ont été collectés et annotés durant cette thèse amenant à 50 le nombre d'exemples d'apprentissage pour les sept classes d'arythmie (soit, en moyenne, 7 exemples par classe). Un exemple de tracé pour chacune des arythmies étudiées est donné en Annexe A.

Les évènements caractéristiques de l'ECG et de la pression qui seront codés dans les données d'apprentissage des deux systèmes ont été décrits en section 1.1.3. La totalité des règles apprises est donnée en Annexe C.

3.2.1 Protocole expérimental : la validation croisée

Les différents types d'apprentissage sont évalués par une technique de validation croisée de type *leave-one-out*. La validation croisée consiste à partager l'ensemble d'apprentissage composé de n exemples en N parties égales. On procède alors à N expériences d'apprentissage à partir de $\frac{(N-1)n}{N}$ exemples. Les $\frac{n}{N}$ exemples restants servent au test des règles apprises. Si n est le nombre d'exemples total, la technique *leave-one-out* consiste à effectuer $N = n$ expériences d'apprentissage en enlevant un seul exemple à chaque fois (on garde donc $n-1$ exemples pour l'apprentissage). La validation croisée de type *leave-one out* est particulièrement adaptée à un petit nombre d'exemples, comme c'est le cas dans notre problème, puisqu'elle permet de garder le plus grand nombre possible d'exemples pour l'apprentissage tout en conservant une mesure intéressante des performances de reconnaissance des règles apprises.

Si VP (vrai positif) dénote le nombre d'exemples positifs correctement classés, FN (faux négatif), le nombre d'exemples négatifs mal classés, FP (faux positif), le nombre d'exemples positifs mal classés et VN (vrai négatif), le nombre d'exemples négatifs correctement classés, la précision de l'apprentissage est calculée par la formule $\frac{VP+VN}{VP+FN+FP+VN}$. Dans la suite, *PrecAp* désigne la précision moyenne des règles calculée lors des apprentissages pendant la validation croisée de type *leave-one out*. *PrecT* désigne la précision moyenne calculée lors des tests de la validation croisée. Le nombre de cycles cardiaques (*Nbcycles*) donne une indication sur la taille de la fenêtre temporelle concernée par les règles produites. Comme indiqué dans la section 1.1.3, un cycle cardiaque correspond à la succession d'une onde P et d'un complexe QRS sur l'ECG et à la succession d'une *diastole* et d'une *systole* sur la voie ABP. Plus la fenêtre temporelle concernée par la règle apprise est petite, plus l'arythmie pourra être détectée tôt et moins la reconnaissance risque d'être affectée par la présence de bruit sur la (les) source(s). Si plusieurs règles ont été apprises pour une même classe, la colonne *Nbcycles* donne le nombre de cycles correspondant à chacune des règles. Le nombre de nœuds (*Nds*) correspond au nombre de nœuds visités dans l'espace de recherche et les temps de calcul (*Tps*) correspondent à des mesures en secondes de temps CPU effectuées sur un ordinateur SUN Ultra-Sparc 5.

3.2.2 Acquisition de règles monosources avec ICL

Le système ICL a été présenté en section 3.1.8. Les composants de base nécessaires au bon fonctionnement du système sont :

- la base d'exemples composée des interprétations. Chaque interprétation est étiquetée par l'une des arythmie (classe) étudiée (fichier *.kb),
- le biais syntaxique (fichier *.l),
- l'environnement (fichier *.s) qui contient les biais sémantiques et la configuration d'ICL,
- la théorie du domaine (fichier*.bg)

```
begin(model(db1_ECGI)).
doublet.
...
p_ECGI(p3db1_ECGI,1826,normal,r2db1_ECGI,p2db1_ECGI,808,p1db1_ECGI,1594,r2db1_ECGI,610).
qrs_ECGI(r3db1_ECGI,2012,normal,p3db1_ECGI,r2db1_ECGI,796,r1db1_ECGI,1606,p3db1_ECGI,186).
p_ECGI(p4db1_ECGI,2636,normal,r3db1_ECGI,p3db1_ECGI,810,p2db1_ECGI,1618,r3db1_ECGI,624).
qrs_ECGI(r4db1_ECGI,2824,normal,p4db1_ECGI,r3db1_ECGI,812,r2db1_ECGI,1608,p4db1_ECGI,188).
qrs_ECGI(r5db1_ECGI,3376,anormal,r4db1_ECGI,r4db1_ECGI,552,r3db1_ECGI,1364,p4db1_ECGI,740).
qrs_ECGI(r6db1_ECGI,3880,anormal,r5db1_ECGI,r5db1_ECGI,504,r4db1_ECGI,1056,p4db1_ECGI,1244).
p_ECGI(p5db1_ECGI,4302,normal,r6db1_ECGI,p4db1_ECGI,1666,p3db1_ECGI,2476,r6db1_ECGI,422).
qrs_ECGI(r7db1_ECGI,4462,normal,p5db1_ECGI,r6db1_ECGI,582,r5db1_ECGI,1086,p5db1_ECGI,160).
...
end(model(db1_I)).
```

Quelques faits appartenant à une interprétation correspondant à un doublet ventriculaire décrit sur la voie I de l'ECG.

```
begin(model(db1_ABP)).
doublet.
...
diastole(pd3db1_ABP,2194,-663,ps2db1_ABP,742,650,pd2db1_ABP,808,pd1db1_ABP,1618,-8).
systole(ps3db1_ABP,2328,37,pd3db1_ABP,700,134,ps2db1_ABP,784,ps1db1_ABP,1604,-41).
diastole(pd4db1_ABP,3004,-675,ps3db1_ABP,713,676,pd3db1_ABP,810,pd2db1_ABP,1618,-12).
systole(ps4db1_ABP,3138,-18,pd4db1_ABP,657,134,ps3db1_ABP,810,ps2db1_ABP,1594,-55).
diastole(pd5db1_ABP,4070,-730,ps4db1_ABP,712,932,pd4db1_ABP,1066,pd3db1_ABP,1876,-55).
systole(ps5db1_ABP,4192,-473,pd5db1_ABP,257,122,ps4db1_ABP,1054,ps3db1_ABP,1864,-455).
diastole(pd6db1_ABP,4622,-734,ps5db1_ABP,261,430,pd5db1_ABP,552,pd4db1_ABP,1618,-4).
systole(ps6db1_ABP,4768,4,pd6db1_ABP,738,146,ps5db1_ABP,576,ps4db1_ABP,1630,476).
...
end(model(db1_ABP)).
```

Quelques faits appartenant à une interprétation correspondant à un doublet ventriculaire décrit sur la voie de pression.

FIG. 3.2 – Exemple de représentation des données

3.2.2.1 Codage de l'ECG et de la pression

Les exemples sont des interprétations construites à partir de faits Prolog et d'une étiquette de classe comme montré Figure 3.2. Les faits PROLOG sont une succession de prédicats p_I/10 (p_I est le nom du prédicat et 10 le nombre d'arguments) et qrs_I/10 pour la voie I, p_V/10 et qrs_V/10 pour la voie V et diastole/11 et systole/11 pour la voie de pression. La description précise de ces prédicats est donnée en Annexe C. Sur ce grand nombre d'arguments, très peu sont réellement nécessaires à la description de chaque onde : le nom de l'onde, son instant d'apparition, son amplitude pour les ondes de pression, la forme de l'onde pour l'ECG et l'onde précédente seraient des arguments suffisants pour coder toutes les informations disponibles. Le choix de coder plus d'informations (notamment la durée des intervalles entre l'onde courante et les ondes précédentes ou les différences d'amplitude) a été fait afin de diminuer la durée des apprentissages et permettre la discrétisation automatique de ces attributs par ICL

(cf. Section suivante). Les temps d'apprentissage sont réduits car l'interpréteur PROLOG n'a pas de calcul numérique à faire lors des tests de couverture des exemples.

3.2.2.2 Acquisition des seuils de pression

rs		
	seuil 1	seuil 2
dd1	748	942
dd2	1496	1874
ds1	154	152
sd1	624	842
ss1	748	942
ss2	1496	1874
amp_ds	1079	700
amp_sd	1087	1366
amp_dd	-31	20
amp_ss	-36	53

rs			
	min	max	moyen
dd1	734	942	817
dd2	1470	1874	1635
ds1	76	170	124
sd1	582	842	692
ss1	718	972	817
ss2	1454	1874	1635
amp_ds	588	1366	913
amp_sd	578	1382	909
amp_dd	-38	52	2
amp_ss	-161	189	4

TAB. 3.1 – Résultats de la discrétisation pour la classe *rs* (à gauche) et statistiques pour la même classe (à droite) pour tous les arguments numériques.

Pour accélérer les temps d'apprentissage et accroître la lisibilité des règles produites, les valeurs numériques présentes dans les faits Prolog (cf. Figure 3.2) ne sont pas utilisées directement. Nous leur préférons des intervalles symboliques de type *grand*, *normal* ou *court*. En ce qui concerne l'ECG, les bornes de ces intervalles sont directement données par les valeurs associées au rythme *normal* indiquées dans la littérature médicale.

Pour la pression, en revanche, les valeurs des bornes correspondant aux différences d'amplitude ou aux durées entre une diastole et une systole sont propres à chaque patient. Pour un apprentissage efficace, ces bornes (appelées *seuils* dans la suite) doivent donc être calculées automatiquement.

Dans ICL, ces seuils peuvent être calculés par un algorithme de discrétisation [Van Laer *et al.*, 1997] basé sur celui de Fayyad et Irani [Fayyad et Irani, 1993] pour le cas propositionnel. Cet algorithme global et supervisé [Dougherty *et al.*, 1995] permet de discrétiser les valeurs numériques prises par un des arguments d'un prédicat fixé. Dans notre cas, l'ensemble des valeurs numériques correspondant à un intervalle temporel ou à une différence d'amplitude a été discrétisé de manière à fournir, pour chaque classe, les deux seuils les plus significatifs (cf. seuils 1 et 2 Table 3.1, le seuil 1 étant le plus significatif) qui serviront, quand cela est pertinent, à créer les trois intervalles symboliques (*court,normal,long*).

Le calcul des valeurs discrétisées est basé sur la mesure d'entropie présentée en section 3.1.5. Pour chaque classe, les valeurs d'un argument donné correspondant aux exemples représentatifs de la classe forment un premier ensemble et les valeurs du même argument pour les exemples décrivant toutes les autres classes forment un second

ensemble. Plus l'entropie entre ces deux ensembles est petite, plus le seuil choisi est significatif. Les seuils de discrétisation $seuil1$ et $seuil2$ sont les deux meilleures valeurs minimisant l'entropie des deux ensembles.

Pour caractériser les arythmies, il nous importe de savoir comment se distingue un rythme pathologique d'un rythme normal (représenté par la classe *rs*). Nous avons donc, dans un premier temps, utilisé l'algorithme de discrétisation d'ICL pour calculer les seuils de normalité sur les valeurs de tous les attributs pour la classe *rs*. Tout ce qui n'est pas à l'intérieur de l'intervalle borné par ces seuils est soit *court* soit *long*. Les valeurs des seuils trouvés automatiquement par l'algorithme basé sur la mesure d'entropie sont présentées dans le tableau de gauche Table 3.1. On peut comparer ces seuils aux minima, maxima et moyenne trouvées pour chacune de ces valeurs dans le tableau de droite. En effet, pour chaque attribut (par exemple, *amp_sd*), les valeurs de cet attribut représentées sur l'axe des réels (**R**) Figure 3.3, sont réparties sur un intervalle inclus dans l'intervalle des valeurs prises par le même attribut pour les autres classes. On peut donc penser, pour chaque attribut, que les valeurs d'entropie minimale (les seuils recherchés) pour la classe *rs* (calculées contre les autres classes) vont se trouver proches des minima et maxima des valeurs prises par cet attribut pour cette même classe. Cela n'est vrai qu'en l'absence de valeurs aberrantes et si les valeurs de cet attribut pour les autres classes sont suffisamment distinctes de celles de la classe *rs*.

On remarque Table 3.1 que les seuils de discrétisation fournis par ICL pour chaque attribut pour la classe *rs* sont, en effet, proches des valeurs minimales et maximales prises par ces attributs, notamment pour les intervalles de temps (à l'exception des valeurs concernant l'attribut *ds1*). Pour cet attribut, les deux seuils fournis par l'algorithme de discrétisation sont très proches (154 et 152) ce qui laisse penser qu'il n'y avait pas de deuxième seuil donnant une mesure d'entropie satisfaisante pour la classe *rs*. Les seuils pour cet attribut ont donc été calculés à partir des seuils de l'intervalle $dd1$ et de l'intervalle $ds1$ ($dd1 = sd1 + ds1$), les valeurs choisies sont donc $ds1(normal) \in [100, 124]$.

Les seuils de discrétisation et les valeurs minimales et maximales sont par contre très éloignés pour les attributs dénotant des différences d'amplitude (en particulier *amp_sd*, *amp_ds* et *amp_ss*). Le schéma présenté Figure 3.3 donne la répartition des valeurs numériques de l'attribut *amp_sd* pour le rythme sinusal (ligne avec les croix); pour l'ensemble des autre arythmies (ligne avec les "plus"); ainsi que les seuils de discrétisation trouvés pour cet attribut (ligne avec les carrés vides); les valeurs minimales et maximales des valeurs prises pour ces attributs pour un rythme sinusal (ligne avec les étoiles) et les seuils finalement choisis (ligne avec les carrés pleins). Les seuils de discrétisation donnés par ICL sont très mauvais puisque l'intervalle borné par ces seuils correspondant à une valeur *normale* de cet attribut ne prend pas en compte la moitié des valeurs numériques de la classe *rs* (cf. le nombre de croix comprises entre les deux carrés vides qui bornent l'intervalle *normal*). Cela vient du fait que ces valeurs varient beaucoup d'un exemple à l'autre. Par exemple, pour le rythme sinusal, les valeurs de l'attribut *amp_sd* s'étalent entre 600 et 800 pour certains exemples, entre 850 et 950 ou entre 1300 et 1400 pour d'autres. L'importance de ces variations d'un exemple à l'autre rend impossible l'utilisation de la fonction de discrétisation d'ICL. Pour déter-

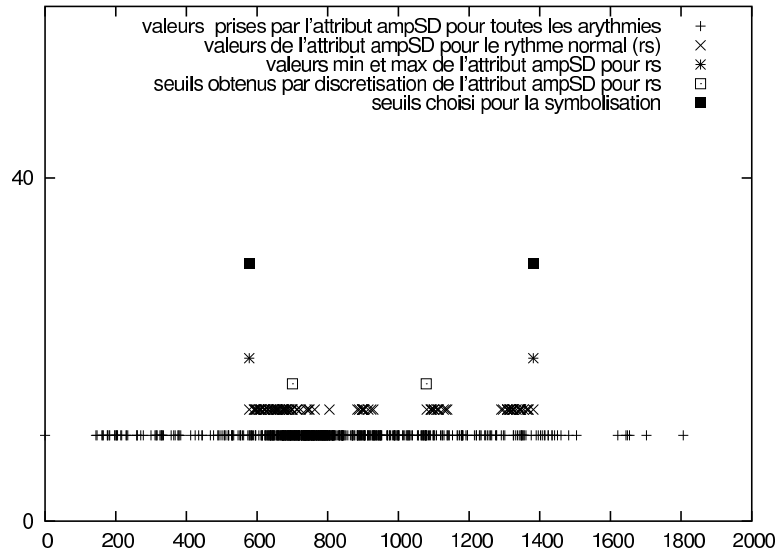


FIG. 3.3 – Répartitions des valeurs numériques et des différents seuils trouvés pour l'attribut amp_sd

miner les bornes de l'intervalle symbolique *normal* pour cet attribut, nous avons donc décidé d'utiliser les valeurs minimales et maximales de l'attribut pour la classe *rs* (cf. Table 3.1).

La figure 3.4 donne une représentation des attributs et des seuils choisis pour la pression sous forme de *boites à moustaches*. Un quart des données se situent entre la ligne centrale de la boite (la médiane des données) et l'extrémité supérieure (resp. inférieur) de la boite. Les lignes à l'extérieur des boites (les *moustaches*) bornent 90% des données. Les points à l'extérieur des boites correspondent aux valeurs aberrantes (*outliers*). Toutes les données ont été centrées et réduites. Les seuils choisis pour délimiter l'intervalle symbolique *normal* correspondant à chacun de ces attributs (cf. les étoiles noires sur chacune des boites) sont placés sur chacune des 4 boites.

On peut remarquer que les boites représentées ici sont effectivement petites relativement à la loi centrée réduite : les attributs choisis sont donc intéressants car ils offrent peu de variabilité entre les exemples. Les seuils choisis pour borner l'intervalle symbolique *normal*) semblent donc pertinents puisque cet intervalle englobe la majorité des valeurs de chaque attribut pour le rythme normal.

3.2.2.3 Connaissances du domaine

la théorie du domaine code de manière explicite et condensée les ondes et les valeurs des intervalles d'intérêt (cf. Section 1.1.3). Il permet de définir explicitement le langage

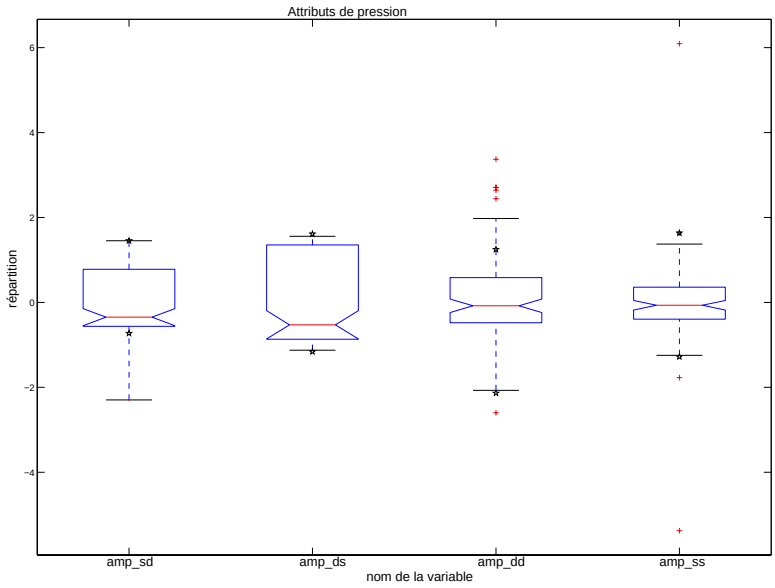


FIG. 3.4 – Mesure de stabilité des attributs de pression et des seuils choisis pour le rythme sinusal

des hypothèses.

Les exemples 9 et 10 donnent une partie des prédicats de la théorie du domaine correspondant respectivement à l'ECG et à la mesure de pression.

Voie I de l'ECG

Exemple 9

```
p_wav(P,D):- p_I(P,_,D,_,_,_,_,_,_,_).
qrs(Q,D):- qrs_I(P,_,D,_,_,_,_,_,_,_).
...
cycle_I(P,FormeP,R,FormeR):-
  p_wav(P,FormeP),
  qrs(R,FormeR).
...
rr1(R0,R1,D):-
  qrs_I(R1,_,_,_,R0,D1,_,_,_,_),
  kk1(D1,DD).
...
rythme(R0,R1,R2,regulier):-
  rr1(R0,R1,D1),
  rr1(R1,R2,D2),
```

```
  D1 == D2.
...
rythme(R0,R1,R2,irregulier):-
  rr1(R0,R1,D1),
  rr1(R1,R2,D2),
  D1 \== D2.
...
suc(A,B):- p_I(A,_,_,B,_,_,_,_,_,_).
suc(A,B):- qrs_I(A,_,_,B,_,_,_,_,_,_).
...
kk1(D1,short):-integer(D1),D1<600,!.
kk1(D1,long):-integer(D1),D1>1000,!.
kk1(D1,normal):-integer(D1),D1>599,D1<1001!.
....
```

Le prédicat *cycle_I* a été introduit pour améliorer la lisibilité des hypothèses produites.

Le prédicat $kk1$ permet de tester l'appartenance des valeurs numériques des intervalles d'intérêt entre les ondes à des intervalles symboliques. Un intervalle $rr1$ sera donc considéré *court* s'il est inférieur à 600 ms (cf. Section 1.1.3 pour la valeurs des intervalles).

Le prédicat $suc(A, B)$ permet de coder la succession immédiate de deux ondes.

Le prédicat *rythme* a été introduit pour évaluer la régularité du rythme en fonction de l'intervalle $rr1$. Deux intervalles $rr1$ de même valeur symbolique dénotent un rythme régulier.

Voie de pression

Exemple 10

```
%codage des ondes

diastole(Dias,S):-
  diastole(Dias,_,_,S1,_,_,_,_,_,_,_,_),amp(S1,S).

systole(Sys,S):-
  systole(Sys,_,_,S1,_,_,_,_,_,_,_,_),amp(S1,S).
...
cycle_ABP(Dias,AmpSD,Sys,AmpDS):-
  diastole(Dias,AmpSD),
  systole(Sys,AmpDS).
...

%codages des intervalles
dd1(Dias0,Dias1,D):-
  diasole(Dias1,_,_,_,_,_,_,Dias0,D1,_,_,_),
```

```
  kk1(D1,DD).
...
amp_dd(Dias1, Dias2, pos, long) :-
diastole(Dias2, _,_,_,_,_, Dias1, _,_,_,D1),D1 >=20.

amp_dd(Dias1, Dias2, neg, long) :-
diastole(Dias2, _,_,_,_,_, Dias1, _,_,_,D1),D1 < -31.

amp_dd(Dias1, Dias2, nul, normal) :-
diastole(Dias2, _,_,_,_,_, Dias1, _,_,_,D1),
D1 >= -31,D1 < 20.
...
%relation de succession
suc(A,B):- p_I(A,_,_,B,_,_,_,_,_,_).
suc(A,B):- qrs_I(A,_,_,B,_,_,_,_,_,_).
...

%codage des valeurs symboliques des intervalles
amp(S,short):-integer(S),S<578,!.
amp(S,high):-integer(S),S>1382,!.
amp(S,normal):-integer(S).
....
kk1(D1,short):-integer(D1),D1<600,!.
kk1(D1,long):-integer(D1),D1>1000,!.
kk1(D1,normal):-integer(D1),D1>600,D1<1000.
```

Le deuxième argument des prédicats *diastole* et *systole* dans l'exemple 10 représente la différence d'amplitude entre la systole précédente et la diastole (`amp_sd`) dans le premier cas et entre la diastole précédente et la systole dans le second cas (`amp_ds`). Le prédicat `amp_dd(Dias1, Dias2, pos, long)` signifie que la différence d'amplitude entre deux diastoles `Dias1` et `Dias2` consécutives est importante (*long*) et que la seconde diastole a une amplitude supérieure à la première (l'introduction des valeurs `pos` et `neg` permet de visualiser de manière plus intuitive le tracé de courbe de pression).

3.2.2.4 Construction de l'espace de recherche

Pour biaiser l'espace de recherche avec ICL, deux méthodes doivent être utilisées conjointement : le biais syntaxique DLAB et le biais sémantique décrit dans le fichier d'environnement.

Exemple 11 (Fichier d'environnement d'ICL)

```
classes([rs,bige,esv,doublet,tv, tsv,fa]).
language(dnf).
maxbody(30).
types(off).
```

```
modes(off).
heuristic(m_estimate).
significance_level(0.0).
min_coverage(2).
min_accuracy(0.1).
beam_size(15).
%cross validation leave-one-out
cv_sets(50).
```

Le fichier d'environnement (cf. Exemple 11) permet de changer les paramètres d'ICL. Nous avons choisi d'utiliser ICL dans son mode DNF permettant d'apprendre des classifieurs. Ce mode a été choisi empiriquement en raison de la compacité des règles apprises (contrairement au mode CNF) qui sont alors beaucoup plus aisées à transformer en chroniques CRS. Nous n'avons fait aucune déclaration de types ou de modes, nous avons choisi de limiter le nombre de littéraux à l'intérieur des hypothèses à 30. Le score de chacune des clauses est évalué par *m_estimateur* compte tenu du faible nombre d'exemples. On impose que les règles couvrent au moins deux exemples positifs (pour éviter un apprentissage par cœur consistant à apprendre une règle par exemple) et nous imposons une précision minimale de 0.1 aux règles sélectionnées dans le faisceau durant la recherche. La taille du faisceau de recherche est fixée arbitrairement à 15. Le nombre de validations croisées est fixé au nombre total d'exemples pour permettre une validation croisée de type *leave-one-out*.

L'espace de recherche doit être adapté au domaine d'application. Le signal ECG présente un caractère cyclique, nous avons donc construit le biais DLAB sur le modèle des cycles cardiaques.

```
1     1-1:[
2           len-len:[cycle_I(P0,w_feature ,R0, w_feature),suc(R0,P0),
3                           0-1:[pr1(P0,R0, r_feature)]],
4           len-len:[p_wav(P0, w_feature, _), equal(P0,R0)],
5           qrs(R0, w_feature)
6           ],
7  ...
8  dlab_variable(w_feature, 1-1, [normal, anormal]).
```

FIG. 3.5 – Spécification syntaxique d'un cycle cardiaque en DLAB pour l'ECG

Sur l'ECG, un cycle cardiaque est caractérisé par l'une des trois possibilités suivantes (cf. Figure 3.5) :

- une onde P nommée `P0` suivie par un complexe QRS nommé `R0` (ligne 2 dans le prédicat `cycle_I`) suivi par un prédicat optionnel (contrainte `0-1`) `pr1` (ligne 3);
- une onde P seule (ligne 4) suivi d'un prédicat `equal` permettant de maintenir la chaîne temporelle entre les prédicats même en l'absence du QRS;
- un QRS (`R0`) seul (ligne 5).

L'utilisation des variables `dlab_variable` permet de condenser la représentation du biais : la variable sera substituée, pendant l'apprentissage, par le terme DLAB décrit

en deuxième argument. Dans l'exemple précédent le terme `w_feature` est considéré comme une variable qui peut prendre les valeurs `normal,anormal`.

Les biais construits pour apprendre des règles à partir des données de la voie I et de la pression artérielle sont donnés ci-dessous. Ils sont composés d'enchaînements de cycles comme celui de la figure 3.5. Cette fois, au lieu d'un prédicat optionnel tel que `pr1`, il y aura plusieurs prédicats optionnels précédés du terme `0-len:[]` dont la signification est explicitée dans l'exemple 8 de la section 3.1.8.4.

```
dlab_template('
 false       <--
 len-len:[
          1-1:[len-len:[
          1-1:[
          len-len:[cycle_I(P0,w_feature ,R0, w_feature),suc(R0,P0),
                          0-1:[pr1(P0,R0, r_feature)]],
          len-len:[p_wav(P0, w_feature, _), equal(P0,R0)],
          qrs(R0, w_feature)
          ],
          0-1:[len-len:[
          1-1:[
               len-len:[cycle_I(P1,w_feature, R1, w_feature), suc(P1,R0), suc(R1,P1),
                0-len:[rr1(R0, R1, r_feature), pr1(P1, R1, r_feature)]],
               len-len:[p_wav(P1, w_feature), suc(P1,R0),
                0-1:[pp1(P0, P1, r_feature)], equal(P1, R1)],
               len-len:[qrs(R1, w_feature), suc(R1,R0),
                0-1:[rr1(R0, R1, r_feature)]]
          ],
          0-1:[len-len:[
          1-1:[
               len-len:[cycle_I(P2, w_feature, R2, w_feature), suc(P2,R1), suc(R2,P2),
                0-len:[rythm(R0,R1,R2,rythm_feature),
                rr1(1-1:[R1,R0], R2, r_feature),
                rr2(R0, R2, r_feature),
                pr1(P2, R2, r_feature)]],
               len-len:[p_wav(P2, w_feature), suc(P2,R1),
                0-1:[pp1(1-1:[P1, P0], P2, r_feature)], equal(P2, R2)],
               len-len:[qrs(R2, w_feature), suc(R2,R1),
                                0-len:[
                rythm(R0,R1,R2,rythm_feature),
                rr2(R0, R2, r_feature),
                rr1(1-1:[R1,R0], R2, r_feature)]]
                                ],
%          .... (quatre autres cycles cardiaques)
                        ]]
                        ]]
                        ]]
                        ]]
         ]
').

dlab_variable(r_feature, 1-1, [short, normal, long]).
dlab_variable(w_feature, 1-1, [normal, anormal]).
dlab_variable(rythm_feature, 1-1, [regular,irregular]).
```

On peut remarquer que les cycles sont imbriqués de manière à être utilisés dans l'apprentissage exactement dans l'ordre dans lequel ils ont été spécifiés. Cela permet d'éviter l'apprentissage de clauses non connectées, i.e., des règles possédant des cycles non rattachés (par le prédicat `suc`) à un autre cycle. De telles règles seraient très difficiles à interpréter. La spécification `0-1:[len-len:[` en début de chaque cycle permet en outre de spécifier les cycles optionnels ou obligatoires. Par exemple, pour les deux biais présentés dans cette section, le premier cycle cardiaque est obligatoire et les autres cycles sont optionnels. On peut ainsi générer des espaces d'hypothèses imposant un nombre variable de cycles cardiaques.

```
dlab_template(' false <--
len-len:[
           1-1:[len-len:[
           cycle_ABP(Dias0,_,Sys0,w_feature), suc(Sys0,Dias0),
           0-0:[ds1(Dias0,Sys0, r_feature)],

          0-1:[len-len:[
          cycle_ABP(Dias1, w_feature, Sys1,w_feature), suc(Dias1,Sys0) ,suc(Sys1,Dias1),
          0-1:[amp_ss(Sys0, Sys1,  amp_featureS,  amp_feature)],
          0-1:[amp_dd(Dias0, Dias1,  amp_featureS,  amp_feature)],
          0-1:[ss1(Sys0, Sys1, r_feature)],
          0-1:[dd1(Dias0, Dias1, r_feature)],
          0-1:[ds1(Dias1,Sys1, r_feature)],
          0-1:[sd1(Sys0, Dias1, r_feature)],

          0-1:[len-len:[
          cycle_ABP(Dias2, w_feature, Sys2,w_feature), suc(Dias2,Sys1) ,suc(Sys2,Dias2),
          0-1:[amp_ss(Sys1, Sys2, amp_featureS,  amp_feature)],
          0-1:[amp_dd(Dias1, Dias2,  amp_featureS,  amp_feature)],
          0-1:[ss1(Sys1, Sys2, r_feature)],
          0-1:[dd1(Dias1, Dias2, r_feature)],
          0-1:[ds1(Dias2, Sys2, r_feature)],
          0-1:[sd1(Sys1, Dias2, r_feature)],
          0-1:[ss2(Sys0, Sys2, r_feature)],
          0-1:[dd2(Dias0, Dias2, r_feature)],

           ....(trois autres cycles cardiaques)
          ]]
          ]]
          ]]
     ]
  ').
dlab_variable(r_feature, 1-1, [short, normal, long]).
dlab_variable(w_feature, 1-1, [short, normal, high]).
dlab_variable(amp_feature, 1-1, [normal, long]).
dlab_variable(amp_featureS, 1-1, [pos, neg,nul]).
```

Pour la pression le découpage des cycle est plus simple que pour l'ECG puisqu'une systole est toujours précédée d'une diastole et inversement.

3.2.2.5 Résultats de l'apprentissage

Les règles obtenues avec ICL sont présentées sous forme de clauses du type :

```
class(<nomclass>,
<répartition des exemples couverts par la règle pour chaque classe>,
<répartition des exemples non couverts par la règle pour chaque classe>):-
< suite de prédicats caractérisant la classe>).
```

Par exemple :

```
class(rs, [8, 3, 10, 6, 6, 0, 4], [0, 4, 0, 0, 1, 5,3]) :-
qrs(R0, normal, _),
p_wav(P1, normal, R0), qrs(R1, normal, P1).
```

Les deuxième et troisième arguments du prédicat *class* donnent la répartition des exemples positifs couverts (et non couverts) par les différentes théories comme expliqué en section 3.1.8.2. L'ordre des classes est celui imposé dans le fichier d'environnement, i.e. : *rs,bige,esv,doublet,tv,tsv,fa*. `[8, 3, 10, 6, 6, 0, 4]` signifie donc que la règle couvre sept exemples de la classe *rs*, trois exemples de la classe *bige*, dix exemples de la classe *esv*, six exemples de la classe *doublet*, six exemples de la classe *tv*, aucun exemple de la classe *tsv* et quatre exemples de *fa*. Le troisième argument `[0, 4, 0, 0, 1, 5, 3]` montre que tous les exemples des classes *rs,esv* et *doublet* ont été couverts par cette règle, quatre exemples de la classe *bige* n'ont pas été couverts, etc.

monosource ECG_I					
	PrecAp	PrecT	Nbcycl	Nds	Tps
rs	0.62	0.60	7	4634	3660
esv	0.981	0.94	5	1654	189
bige	1	0.98	4	708	160
doublet	1	1	4	790	125
tv	1	1	3	428	109
tsv	1	0.98	3	232	105
fa	1	1	2	64	4

TAB. 3.2 – Résultats de la validation croisée pour l'apprentissage sur la voie I.

Voie I Les résultats de l'apprentissage sur la voie I de l'ECG sont donnés dans le tableau 3.2. Les règles correspondantes sont présentées ci-dessous. Ces règles donnent des résultats de précision en test et en apprentissage parfaits pour le *doublet ventriculaire*, la *tachycardie ventriculaire* et la *fibrillation auriculaire*. Ils sont en revanche beaucoup moins bons pour le rythme sinusal. Ce rythme est en effet difficile à discriminer d'une extrasystole (mais aussi d'un doublet ventriculaire ou d'un accès de tachycardie ventriculaire) puisqu'un exemple d'extrasystole est en tout point identique à un rythme

normal excepté la présence d'un seul QRS anormal dans la succession de battements (deux QRS anormaux pour le doublet et trois pour un accès de tachycardie ventriculaire). Pour pouvoir discriminer ces arythmies, il faudrait utiliser un très grand nombre de cycles cardiaques couvrant pratiquement la totalité de la durée des exemples. Cependant, le nombre de cycles cardiaques autorisés est limité à 7. Cette limitation est imposée par l'utilisation réaliste de ces règles en milieu bruité. Plus les règles sont longues, plus la durée pendant laquelle la détection des ondes et la qualité des signaux doit être parfaite est élevée. La règle *esv* qui s'étale sur 5 cycles cardiaques a elle aussi de forts risques de ne pas fonctionner en milieu bruité.

```
class((rs,[8,0,10,5,4,0,0],[0,7,0,1,3,5,7]) :-
qrs(R0,normal),
cycle_I(P1,normal,R1,normal), suc(P1,R0), suc(R1,P1),
cycle_I(P2,normal,R2,normal), suc(P2,R1), suc(R2,P2),
rr1(R1,R2,normal),
cycle_I(P3,normal,R3,normal), suc(P3,R2), suc(R3,P3),
cycle_I(P4,normal,R4,normal), suc(P4,R3), suc(R4,P4),
cycle_I(P5,normal,R5,normal), suc(P5,R4), suc(R5,P5),
cycle_I(P6,normal,R6,normal), suc(P6,R5), suc(R6,P6).

class(bige,[0,7,0,0,0,0,0],[8,0,10,6,7,5,7]) :-
cycle_I(P0,normal,R0,normal), suc(R0,P0),
qrs(R1,anormal), suc(R1,R0),
cycle_I(P2,normal,R2,normal), suc(P2,R1), suc(R2,P2),
qrs(R3,anormal), suc(R3,R2).

class(esv,[0,0,9,0,0,0,0],[8,7,1,6,7,5,7]) :-
cycle_I(P0,normal,R0,normal), suc(R0,P0),
 cycle_I(P1,normal,R1,normal), suc(P1,R0), suc(R1,P1),
qrs(R2,anormal), suc(R2,R1),
cycle_I(P3,normal,R3,normal), suc(P3,R2), suc(R3,P3),
rr1(R2,R3,normal),
cycle_I(P4,normal,R4,normal), suc(P4,R3), suc(R4,P4).

class(doublet,[0,0,0,6,0,0,0],[8,7,10,0,7,5,7]) :-
cycle_I(P0,normal,R0,normal), suc(R0,P0),
qrs(R1,anormal), suc(R1,R0),
qrs(R2,anormal), suc(R2,R1),
cycle_I(P3,normal,R3,normal), suc(P3,R2), suc(R3,P3).

class(tv,[0,0,0,0,7,0,0],[8,7,10,6,0,5,7]) :-
qrs(R0,anormal),
qrs(R1,anormal), suc(R1,R0),
qrs(R2,anormal), suc(R2,R1).
```

```
class(tsv,[0,0,0,0,0,5,0],[8,7,10,6,7,0,7]) :-
cycle_I(P0,normal,R0,normal), suc(R0,P0),
cycle_I(P1,normal,R1,normal), suc(P1,R0), suc(R1,P1),
cycle_I(P2,normal,R2,normal), suc(P2,R1), suc(R2,P2),
rr2(R0,R2,short).

class(fa,[0,0,0,0,0,0,7],[8,7,10,6,7,5,0]) :-
qrs(R0,normal),
qrs(R1,normal), suc(R1,R0).
```

Les règles présentées ici sont toutes cohérentes du point de vue des connaissances expertes, mise à part la règle concernant la *fa* qui n'a aucune signification médicale. Il est en effet facile pour ICL de discriminer une *fa* puisque les exemples représentatifs de cette classe sont les seuls à ne pas avoir d'onde P et le QRS reste toujours normal. Pour ICL, deux QRS normaux consécutifs sont donc discriminants. Pour pouvoir obtenir une règle caractérisant la *fa* satisfaisante d'un point de vue médical, il faudrait introduire des exemples de nouvelles arythmies (notamment des arythmies pour lesquelles l'onde P n'est pas décelable et où le QRS reste normal).

Les règles pour la voie V et les apprentissages sans prendre en compte la forme du QRS, plus proches des capacités de traitement du signal, sont donnés en Annexe C. Les résultats de la validation croisée sont donnés dans les tableaux 3.3 et 3.4. Les temps de calcul et la taille des espaces de recherche sont inférieurs à ceux obtenus pour la voie I car le nombre de relations possibles entre les évènements est réduit puisqu'on ne considère que des évènements de type QRS.

monosource ECG_V					
	CorAp	CorT	Nbcycl	Nds	Tps
rs	0.48	0.46	6	1617	253.35
esv	1	0.98	6	713	21.49
bige	1	0.98	4	156	6.64
doublet	1	1	4	277	5.93
tv	1	1	3	101	4.15
tsv	0.96	0.94	4	1377	170.78
fa	0.961	0.92	3	2408	85.26

TAB. 3.3 – Résultats de la validation croisée pour l'apprentissage sur la voie V.

Les résultats de la validation croisée restent très bons pour la plupart des arythmies (excepté pour le *rs*) et les règles sont très claires pour les médecins, en particulier pour la *fa* grâce à la prise en compte de l'attribut `rythm`. Le *rs* est encore plus dur à discriminer sur la voie V (et sur la voie V sans la forme des ondes) que sur la voie I. Ces observations montrent qu'il semble tout à fait pertinent d'apprendre des règles sur une source de données contenant moins d'information que la voie I, chacun

monosource ECG_V					
	CorAp	CorT	Nbcycl	Nds	Tps
rs	0.48	0.44	6	1133	209
esv	0.52	0.46	6/6	2334	455
bige	0.98	0.90	6	690	41
doublet	0.851	0.78	5/5	2862	152
tv	0.883	0.72	6	1246	111
tsv	0.96	0.96	6	1134	169
fa	0.977	0.9	4/5	992	72

TAB. 3.4 – Résultats de la validation croisée pour l'apprentissage sur la voie V sans la forme du QRS.

des apprentissages effectués avec ICL sur l'ECG semblant apporter des informations précieuses pour caractériser les arythmies. Dans un système de monitoring cardiaque tel que CALICOT, l'information redondante apportée par les différentes voies de l'ECG peut permettre de maintenir une détection fiable en cas de bruitage ou de détérioration d'une des voies.

Voie hémodynamique Les valeurs utilisées pour la représentation symbolique des intervalles sont données en section 3.2.2.2.

monosource ABP					
	PrecAp	PrecT	Nbcycl	Nds	Tps
rs	1	0.98	5	5475	1415
esv	0.99	0.76	4/5/3	19971	2362
bige	0.96	0.80	3	7365	337
doublet	0.94	0.78	4/5	9840	596
tv	0.99	0.86	6/4	10833	691
tsv	1	1	2	1326	438
fa	0.99	0.96	3/4	6477	1923

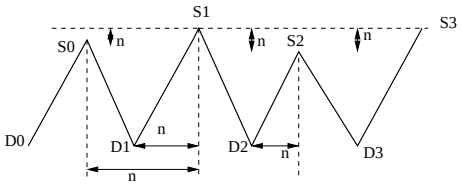
TAB. 3.5 – Résultats de la validation croisée pour l'apprentissage sur la voie pression

Les résultats de l'apprentissage sur la voie hémodynamique sont donnés dans le tableau 3.2. Les espaces de recherche sont beaucoup plus vastes que pour la voie I car le nombre d'attributs permettant de décrire le signal de pression est beaucoup plus important : en plus de toutes les relations temporelles, les différences d'amplitudes sont prises en compte. Notons que le temps de calcul permettant d'apprendre la règle pour le *rs* est réduit de moitié par rapport à celui de la voie I, même si le nombre de noeuds est supérieur. En effet, les relations entre diastoles et systoles sont beaucoup plus simples que les relations entre les complexes QRS et les ondes P puisqu'à une diastole

correspond toujours une systole, ce qui n'est pas le cas pour les ondes de la voie I. Les règles apprises ont une bonne précision (meilleure que pour la voie I pour le *rs*), mais les résultats en tests peuvent encore être améliorés. Pour l'*esv* en particulier, trois règles ont été apprises pour couvrir les dix exemples de la base. Ces règles, bien qu'ayant une signification médicale, sont encore trop proches des exemples, ce qui explique les résultats moyens en test. Pour améliorer les performances de l'apprentissage sur cette source de données, il est nécessaire d'augmenter le volume de la base d'exemples.

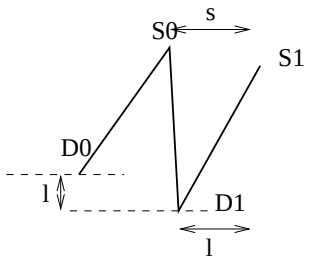
La totalité des règles correspondantes est présentée en Annexe C. Pour chacune des classes, un croquis de la première règle a été donné pour faciliter la compréhension des règles. Les doubles flèches verticales représentent des différences d'amplitude et les flèches horizontales des intervalles temporels. Les valeurs s, l et n sur les flèches représentent respectivement les intervalles symboliques *courts, longs* et *normaux* utilisés dans les règles. Di représente la diastole (*Dias*) i et Si, la systole (*Sys*) i.

```
class((rs,[8,0,0,0,0,0,0],[0,7,10,6,7,5,7]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_ss(Sys0,Sys1,nul,normal),
ss1(Sys0,Sys1,normal),
ds1(Dias1,Sys1,normal),
cycle_ABP(Dias2,normal,Sys2,normal), suc(Dias2,Sys1), suc(Sys2,Dias2),
amp_ss(Sys1,Sys2,nul,normal),
ds1(Dias2,Sys2,normal),
cycle_ABP(Dias3,normal,Sys3,normal), suc(Dias3,Sys2), suc(Sys3,Dias3),
cycle_ABP(Dias4,normal,Sys4,normal), suc(Dias4,Sys3), suc(Sys4,Dias4),
amp_ss(Sys3,Sys4,nul,normal).
```



Au contraire des règles correspondantes pour les voies I et V, la règle obtenue pour le *rs* sur la voie de pression est parfaitement discriminante. Elle se compose d'une succession de 5 cycles diastole/systole avec des différences d'amplitude et des intervalles de temps normaux.

```
class(tsv,[0,0,0,0,0,5,0],[8,7,10,6,7,0,7]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_dd(Dias0,Dias1,neg,long),
ss1(Sys0,Sys1,short), ds1(Dias1,Sys1,long).
```



La règle apprise pour la tachycardie supra-ventriculaire est également parfaitement discriminante et ne nécessite que 2 cycles cardiaques. Des intervalles de temps `ss1(Sys0,Sys1,short)` et `ds1(Dias1,Sys1,long)`, on peut déduire que l'intervalle entre `Sys0` et `Dias1` doit être très court d'où une chute brutale de la pression dans les ventricules jusqu'à un niveau inférieur au niveau précédent marquée par le prédicat `amp_dd(Dias0,Dias1,neg,long)`.

3.2.2.6 Conclusion

ICL permet d'apprendre des règles à partir des données de l'ECG ayant une très bonne précision tant au niveau de la phase d'apprentissage qu'au niveau de la phase de test pour la majorité des arythmies étudiées. Pour la pression en revanche, les résultats obtenus lors du test sont moyens (environ 0.8). Les différences de résultat pour le *rs* laissent penser que ces deux sources sont utiles pour reconnaître de manière fiable la totalité des arythmies.

3.2.3 Acquisition de règles monosources avec Aleph

Les composants de base nécessaires au bon fonctionnement du système Aleph sont :
- la théorie du domaine et les paramètres de l'apprentissage (fichier *.b)
- les exemples positifs (fichier *.f)
- les exemples négatifs (fichier *.n)

3.2.3.1 Codage de l'ECG, de la pression, et construction de l'espace de recherche

Les exemples 12 et 13 donnent une partie des prédicats de la théorie du domaine (le .b) correspondant à l'électrocardiogramme et à la source hémodynamique. Cette théorie du domaine contient à la fois les paramètres d'apprentissage, la théorie du domaine et la description des exemples.

Voie I

Exemple 12

```
%Paramètres d'apprentissage, déclaration de modes et de types
:- set(clauselength,20).
:- set(nodes,300000).
```

```
%XX doit être remplacé pour chaque classe par le nom de la classe.
% ‘‘-’’ signifie sortie, ‘‘+’’ signifie entrée et ‘‘#’’ signifie : valeur instanciée
:- modeh(*,XX(+ecg)).
:- modeb(1,time(+wave,#time)).
:- modeb(1,shape(+wave,#shape)).
:- modeb(*,p(+ecg,-wave,#shape)).
:- modeb(*,qrs(+ecg,-wave,#shape)).
:- modeb(1,suc(+wave,+wave)).
:- modeb(1,rythm(+wave,+wave,+wave,#interval)).
:- modeb(1,pp_1(+wave,+wave,#interval)).
...
:- modeb(*,has_wave(+ecg,+wave,-wave,#type,#shape,-wave,#type,#shape)).
...
:- determination(XX/1,p/3).
:- determination(XX/1,qrs/3).
:- determination(XX/1,suc/2).
:- determination(XX/1,rythm/3).
:- determination(XX/1,has_wave/8).
:- determination(XX/1,rr_1/3).

%détermination des valeurs symboliques des intervalles
rr_1(X,Y,normal) :- rr1(X,Y,Z), Z>600, Z<1000.
rr_1(X,Y,court) :- rr1(X,Y,Z),  Z<600.
rr_1(X,Y,long) :- rr1(X,Y,Z),  Z>1000.
...
rythm(X,Y,Z,regular):-
        rr_1(X,Y,T1),
        rr_1(Y,Z,T1).
rythm(X,Y,Z,irregular):-
        rr_1(X,Y,T1),
        rr_1(Y,Z,T2),
        T1 \== T2.
...
%codages des ondes
has_wave(E,N3,N1,qrs_I,S1,N2,p_I,S2) :-
        has_wave(E,N3),
        has_wave(E,N1),
        suc(N1,N3),
        qrs_I(N1),
        shape(N1,S1),
        has_wave(E,N2),
        suc(N2,N1),
        p_I(N2),
```

```
        shape(N2,S2).
....
qrs(E,Q,S):-
        has_wave(E,Q),
        qrs_I(Q),
        shape(Q,S).
p(E,Q,S):-
        has_wave(E,Q),
        p_I(Q),
        shape(Q,S).
.....
%description des exemples
ecg(rs_1_I).
...
p_I(p10_rs_1_I).
p_I(p11_rs_1_I).
...
qrs_I(r10_rs_1_I).
qrs_I(r11_rs_1_I).
....
has_wave(rs_1_I,p10_rs_1_I).
....
shape(p10_rs_1_I,normal).
shape(p11_rs_1_I,normal).
....
suc(p10_rs_1_I,r9_rs_1_I).
suc(p11_rs_1_I,r10_rs_1_I).
....
time(p10_rs_1_I,8164).
time(p11_rs_1_I,8936).
....
rr1(r10_rs_1_I,r11_rs_1_I,796).
rr1(r11_rs_1_I,r12_rs_1_I,780).
....
```

La description des exemples a été générée directement à partir des données d'ICL, les attributs utilisés et leur valeur numérique sont donc identiques. La description précise de chacun des prédicats utilisés est donnée en Annexe C. Les déclarations des types et modes (cf. section 3.1.7.2) et la déclaration des prédicats autorisés dans les hypothèses (commandes `:- determination/2`) servent à biaiser l'espace de recherche.

Le prédicat `rr_1/3` utilise le prédicat `rr1/2` donné dans la partie concernant la description des exemples. Il permet d'associer à des intervalles de temps distincts (ce sont les mêmes que pour ICL), des valeurs symboliques.

L'imbrication des cycles permettant d'assurer l'absence de clauses non connectées (cf. Section 3.2.2.4) est assurée par la déclaration des modes entrée/sortie. Le deuxième argument des prédicats `qrs(+ecg,-wave,#shape)` et `p(+ecg,-wave,#shape)` est un argument de type `wave` en mode *sortie*. Dans les hypothèses apprises, cela va correspondre à une variable *libre* (non connectée à une autre variable dans l'hypothèse apprise). Le deuxième argument du prédicat `has_wave(+ecg,+wave,-wave,#type,#shape,-wave,#type,#shape)` est un argument de type `wave` en mode *entrée*. Sémantiquement, il s'agit de l'onde qui précède les deux ondes codées dans le prédicat `has_wave/8`. Le fait de mettre cet argument en mode *entrée* oblige la variable qui sera utilisée à avoir déjà été utilisée dans un des prédicats du corps de l'hypothèse (dans notre cas soit dans un autre prédicat `has_wave/8` ou dans les prédicats `qrs/3` ou `p/3`.

Pression

Exemple 13

```
:- set(clauselength,20).
:- set(noise,5).
:- set(nodes,300000).
% XX doit être remplacé pour chaque classe par le nom de la classe.
%déclaration de types et de modes
:- modeh(*,XX(+sap)).
:- modeb(*,suc(+pression,+pression)).
:- modeb(*,has_wave(+sap,+pression,-pression,#type,-pression,#type)).
:- modeb(*,systole(+sap,-pression)).
:- modeb(*,dd_1(+pression,+pression,#intervalle)).
...
%les prédicats qu'on a le droit d'utiliser :
:- determination(XX/1,suc/2).
:- determination(XX/1,systole/2).
:- determination(XX/1,has_wave/6).
:- determination(XX/1,dd_1/3).
...
%prédicats permettant d'élaguer dynamiquement l'espace de recherche
prune((_Head:-Body)) :- violate_constraints(Body).
violate_constraints(Body) :-
        has_pieces(Body,Pieces),
        Pred =.. [systole,_X,_Y,_Z],
        member(Pred,Pieces),
        remove(Pred,Pieces,Pieces1),
        mon_member(Pred,Pieces1).

has_pieces((A,B),[A|L]) :- has_pieces(B,L), !.
has_pieces((A),[A]) :- !.
```

```
remove(E,[E|L],L):-!.
remove(E,[X|L],[X|L1]):-remove(E,L,L1).
....
%détermination des valeurs symboliques des intervalles
sd1(A,B,normal) :- sd(A,B,X), suc(B,A), X=<842, X>=582.
sd1(A,B,court) :- sd(A,B,X), suc(B,A), X<582.
sd1(A,B,long) :- sd(A,B,X), suc(B,A), X>842.
....

%codage des ondes
has_wave(S,P,A,diastole,A2,systole) :-
        has_wave(S,A),
        has_wave(S,P),
        diastole(A),
        suc(A,P),
        has_wave(S,A2),
        systole(A2),
        suc(A2,A).

systole(E,Q):-
        has_wave(E,Q),systole(Q).
...

%description des exemples
...
```

Il n'a pas été possible d'apprendre des règles satisfaisantes à partir des données de pression avec un biais calqué sur celui de la voie I. De nouveaux paramètres ont donc été rajoutés. Pour biaiser plus efficacement l'espace de recherche, on peut déclarer des clauses interdites, définies avec le prédicat `prune/2`. Le prédicat de l'exemple 13 permet ainsi d'élaguer dans l'espace de recherche toutes les clauses dans lesquelles le prédicat `systole/2` est présent deux fois. En effet, ce prédicat dont le deuxième argument est défini en mode *sortie* `systole(+sap,-pression)` ne sert qu'à initier la séquence de cycles permettant d'éviter les clauses non connectées comme expliqué dans le paragraphe précédent sur la voie I. Ce prédicat n'apporte pas d'information particulière pour caractériser l'arythmie cardiaque contrairement au prédicat `has_wave/6`.

Pour alléger les contraintes de couverture des exemples (tous les exemples positifs et aucun exemple négatif), on peut autoriser la couverture d'une partie des exemples négatifs (N exemples) via la commande `:- set(noise,N)`.

3.2.3.2 Résultats de l'apprentissage sur la voie I

Les résultats obtenus pour la voie I, la voie de pression et la voie V sont donnés en Annexe C. Les règles apprises ne sont pas toujours satisfaisantes car elles dénotent

souvent un sur-apprentissage (très peu d'exemples couverts par chaque règle et des apprentissages par cœur correspondant à une règle couvrant un seul exemple). Notons qu'aucune règle n'a pu être apprise pour le rythme sinusal sur la voie I. Un exemple de règle apprise est donné ci-dessous.

```
[Rule 1] [Pos cover = 10 Neg cover = 0]
esv(A) :-
   qrs(A,B,normal),
   has_wave(A,B,C,p_I,normal,D,qrs_I,normal),
   has_wave(A,D,E,p_I,normal,F,qrs_I,normal),
   has_wave(A,F,G,qrs_I,anormal,H,p_I,normal),
   has_wave(A,H,I,qrs_I,normal,J,p_I,normal).

Accuracy = 1.0
[Training set summary] [[10,0,0,40]]
[time taken] [1223.61]

[total clauses constructed] [48692]
```

Cette règle représente une succession $B, C, D, E, F, G, H, I, J$ d'ondes P et QRS normales avec un complexe QRS anormal (l'onde G) non précédé d'une onde P. Elle s'étend sur cinq cycles cardiaques et permet de discriminer parfaitement une *esv* par rapport aux autres arythmies. Notons qu'il n'y a pas de clauses non connectées dans cette description puisque le deuxième argument du prédicat `has_wave/8` représentant l'onde précédente est toujours connecté à un autre prédicat (par exemple la variable D en deuxième argument du deuxième prédicat `has_wave/8` se retrouve en sixième argument du premier prédicat `has_wave/8`).

```
[Rule 1] [Pos cover = 7 Neg cover = 0]
fa(A) :-
   qrs(A,B,normal),
   has_wave(A,B,C,qrs_I,normal,D,qrs_I,normal).

Accuracy = 1.0
[Training set summary] [[7,0,0,43]]
[time taken] [0.13]

[total clauses constructed] [16]
```

Comme pour la règle correspondante apprise avec ICL, la *fa* est caractérisée par 3 QRS normaux consécutifs. C'est l'absence des ondes P qui rend la règle discriminante.

3.2.3.3 Conclusion

Les résultats obtenus avec Aleph sont très bons pour la voie I (excepté pour le rythme sinusal) mais restent très difficiles à obtenir pour la pression. Les théories ap-

prises contiennent beaucoup d'apprentissage par cœur. De plus, pour obtenir des résultats satisfaisants, nous avons dû introduire des prédicats complexes tels que `has_wave/8` pour l'ECG et `has_wave/6` pour la pression qui n'ont pas forcément de réelle signification médicale (excepté pour `has_wave/6` qui dénote un cycle cardiaque complet sur la pression) mais permettent d'introduire un biais syntaxique pour l'apprentissage. La création d'un tel biais, des paramètres d'apprentissage et des prédicats permettant l'élagage dynamique de clauses pendant la recherche n'est pas intuitive et s'est fait empiriquement. De plus, ALEPH ne permet pas à notre connaissance de travailler sur des données multiclasses, il est donc difficile d'évaluer les performances globales pour la validation croisée sur plusieurs classes.

3.2.4 Comparaison des résultats entre ALEPH et ICL

Les résultats obtenus à ce jour montrent qu'ICL semble plus performant qu'ALEPH au niveau de l'apprentissage puisque tous les apprentissages ont pu être effectués (notamment ceux sur la voie de pression) alors que ceux concernant le rythme sinusal ainsi que certains apprentissages concernant les classes apprises à partir des données de pression sont insatisfaisants ou n'ont pas pu être effectués avec ALEPH. Cela provient probablement des formes de biais utilisées par les deux logiciels. Le biais d'ALEPH offre plus de liberté liée à la syntaxe PROLOG, mais cette liberté est souvent trop importante pour limiter l'espace de recherche des clauses résultats et ces recherches sont souvent très longues et dans certains cas aboutissent à des résultats insatisfaisants. On peut cependant noter que, dans le cas des apprentissages effectifs, les règles données par ALEPH sont beaucoup plus compactes que celles données par ICL même si leur lisibilité demande une certaine expertise. En effet, il est nécessaire de définir des prédicats structurés tels que `has_wave/8` pour obtenir des résultats satisfaisants. Ces prédicats offre un biais syntaxique à l'apprentissage mais sont moins intuitifs que les prédicats *cycle_I* ou simplement $p/2$ ou $qrs/2$ utilisés par ICL.

Ces raisons nous poussent à utiliser ICL comme système d'apprentissage dans la suite de nos recherches.

3.3 Conclusion

Dans ce chapitre, nous avons proposé une réalisation basée sur la PLI pour induire automatiquement les règles de classification qui permettent de caractériser des arythmies à partir de différentes sources de données : l'ECG et une mesure de pression artérielle pour deux systèmes de PLI ayant une sémantique différente : ICL et ALEPH. Le codage des exemples et des biais utilisés pour les deux systèmes nous a conduit à faire des choix sur les attributs pertinents permettant de décrire les différentes sources et à créer des biais complexes permettant d'apprendre des règles satisfaisantes.

Les résultats attestent que la méthode de PLI est appropriée pour l'apprentissage automatique de règles de classification pour différentes arythmies à partir d'exemples et de connaissance *a priori* sur le domaine. Ces résultats, en faveur d'ICL, nous ont conduit à garder ce dernier système pour la suite de nos travaux. Les règles induites

sont lisibles et compréhensibles pour des médecins. Ceci leur permet d'évaluer les règles par rapport à leurs propres connaissances.

Bien que les résultats d'apprentissage avec ICL soient presque parfaits, il faut noter que les classes d'arythmies choisies sont très distinctes et donc relativement faciles à discriminer, tout en étant représentatives de troubles du rythme. De plus, l'exactitude des résultats de l'apprentissage automatique dépend fortement de la qualité des données en entrée. Ici, les données utilisées ne sont pas liées aux résultats d'un module de traitement du signal mais proviennent des annotations de la base de MIT BIH qui sont relativement correctes. Afin d'évaluer plus complètement la PLI à l'application de reconnaissance d'arythmies, il est nécessaire d'utiliser les règles apprises sur des signaux réels bruités. Des expériences dans ce sens sont menées dans le chapitre suivant. Il faut également noter que la base d'exemples reste trop restreinte pour appendre des règles réellement fiables. Il faudrait qu'elle puisse contenir, pour chaque arythmie, des exemples de toutes les formes que peuvent prendre ces arythmies pour obtenir des règles discriminantes ayant une signification médicale vraiment pertinente. Puisque la construction d'une base d'exemples représentative à partir de signaux réels est coûteuse, on peut envisager d'utiliser un modèle du cœur tel que CARMEN (cf. Section 1.1.3) qui permet de simuler de nombreuses classes d'arythmies ainsi qu'un ensemble exhaustif des manifestations possibles d'une arythmie donnée.

D'autres apprentissages ont été également effectués sur la voie V, la voie V sans prendre en compte la forme du QRS et la voie de pression sans prendre en compte les informations sur la *diastole* (la totalité des résultats est donnée en Annexe C). Ces apprentissages sont intéressants car les règles apprises, en utilisant moins d'information sur les données, sont *a priori* plus robustes à la présence de bruit sur les sources. Les résultats de la validation croisée sont moins bons que les résultats obtenus avec la voie I ou avec la voie de pression mais restent acceptables (notamment pour la voie V). Ces nouvelles règles pourront être utilisées par le système CALICOT pour améliorer la robustesse de la reconnaissance d'arythmies en présence de bruit. Pour augmenter la fiabilité des apprentissages sur des données ainsi réduites, il peut être utile de bénéficier de la complémentarité des sources pour apprendre des règles utilisant des caractéristique de chacune des sources. Ceci fait l'objet du chapitre suivant.

Chapitre 4

Apprentissage de règles caractérisant des arythmies cardiaques à partir de données multisources

Dans de nombreux domaines tels que la météorologie, la bourse ou la médecine, l'utilisation de plusieurs points de vue reflétant un même phénomène est souvent nécessaire pour caractériser ce phénomène de façon précise. Lorsque les données proviennent de signaux clairs et lorsque les sources de données sont redondantes, le problème est de sélectionner la source la plus porteuse d'information. Lorsque la redondance entre les données est difficile à établir *a priori* ou lorsque les données provenant des différentes sources sont reconnues comme complémentaires, l'utilisation conjointe de plusieurs sources peut améliorer la robustesse et la précision de cette caractérisation.

Dans le but de travailler sur des données inconnues (cf projet Cepica Section 2.1.2), nous cherchons une méthode permettant d'une part, d'établir la redondance ou la complémentarité des sources de données et d'autre part, dans le cas où les données sont complémentaires, de fournir des règles permettant de tirer parti de manière simultanée des informations provenant des différentes sources.

La première section de ce chapitre nous permet de formaliser le concept d'apprentissage multisource par PLI puis, nous proposons une méthode d'apprentissage multisource efficace qui tire parti d'apprentissages monosources pour restreindre l'espace de recherche multisource. Cette méthode a été présentée dans l'article [Fromont *et al.*, 2004] et est détaillée dans [Fromont *et al.*, 2005a]. La deuxième section présente les résultats obtenus avec cette méthode sur les apprentissages monosources présentés en section 3.2.2.5 et dans l'annexe C. Ces résultats sont présentés dans [Fromont *et al.*, 2005b]. Dans la troisième section, nous testons nos règles sur de nouveaux signaux cliniques bruités et montrons que l'apprentissage multisource couplé à des méthodes de pilotage permet d'améliorer la précision et la sensibilité des détections des arythmies dans Calicot. Ces résultats sont présentés dans l'article

[Fromont et Portet, 2005].

4.1 L'apprentissage multisource en PLI

Nous posons tout d'abord le problème de l'apprentissage multisource en PLI. Puis, nous exposons une méthode permettant de construire un biais syntaxique automatiquement pour réduire l'espace de recherche de l'apprentissage multisource. Nous présentons les étapes de construction d'un tel biais et nous donnons les propriétés de l'espace ainsi réduit.

4.1.1 Définition

La définition 4.1 s'inspire de la formulation de Blockeel donné en section 3.1.8.3 pour un apprentissage multiclasse à partir d'interprétation. Nous rappelons que H_c est un ensemble d'hypothèses décrites sur le langage L_H et représentées sous forme de théories clausales décrivant une classe $c \in C$. Les hypothèses h_c de H_c sont apprises à partir d'un ensemble d'exemples E. Les exemples sont représentés par des ensembles de faits PROLOG clos étiquetés par la classe qu'ils représentent et définis sur le langage L_E. L'apprentissage peut utiliser un ensemble T de règles générales sous forme de clauses PROLOG définies sur le langage $L = L_E \cup L_H$ appelées théorie du domaine.

Dans la suite, nous appellerons *prédicat évènementiel* tout prédicat qui, comme *qrs(R0,normal)*, décrit un évènement se produisant sur une des sources (on supposera qu'il y a un et un seul prédicat évènementiel par évènement) et *prédicat relationnel global* tout prédicat qui, comme *suc(R0,R1)*, dénote une relation particulière (ici l'ordonnancement chronologique) commune à chacune des sources. Par opposition, un *prédicat relationnel local* comme *rr1(R0,R1,normal)* fait partie du langage spécifique à une source.

Définition 4.1 (Apprentissage multisource)
Soit $i \in [1, s]$ *le nombre de sources.*
Soient les problèmes de PLI $< L_i, E_i, T_i, C >$ *tels que :* $E_i = \{(e_{i,k}, c) | k \in [1, p]; c \in C\}$ *où* p *est le nombre d'exemples sur chaque source.* T_i *est la théorie du domaine correspondant à chaque source* i *et décrite sur le langage* L_i.
Un problème multisource en PLI est défini par un tuple $< L, E, T, C >$ *tel que :*

- $E = F_{agg}(E_1, E_2, \ldots, E_s)$ *où* F_{agg} *est une fonction d'agrégation dépendant du problème.*
- L *est le langage multisource tel que :*
  $L = L_E \cup L_H$ *avec* $L_E = F_{agg}(L_{E_1}, L_{E_2}, \ldots, L_{E_s})$ *et* $L_H \subseteq \prod_{i=1}^{s} L_{H_i}$,
- T *est un ensemble de règles exprimées dans le langage* L.

Le but est de trouver un ensemble de règles $H = \{H_c | c \in C\}$ *telles que :*
$\forall (e, c) \in E, \forall c' \in C - \{c\}$:

- $H_c \wedge e \wedge T \vDash c$ *(*H_c *couvre* (e, c) *noté* $couvre(H_c, (e, c))$)
- $H_c \wedge e \wedge T \nvDash c'$ *(*H_c *est discriminante)*

Nous nous intéressons à l'induction de connaissances à partir de l'observation du comportement de systèmes dynamiques. Par conséquent, un exemple, i.e. une situation sur une source, est une collection d'évènements datés. Nous supposons que les situations sont décrites au moyen d'une horloge commune. Ce n'est pas souvent le cas pour des données brutes, nous supposons donc que les données ont été pré-traitées pour assurer cette propriété.

4.1.2 Agrégation des exemples

Les exemples $e_{i,k}$ provenant des différentes sources sont bidimensionnels. La première dimension, $i \in [1, s]$, fait référence à une source, la seconde, k, fait référence à une situation. Les exemples indexés par la même situation correspondent à des vues contemporaines d'un phénomène unique. L'agrégation est l'opération consistant à fusionner les vues contemporaines d'un phénomène. La fonction d'agrégation F_{agg} dépend du type des données d'apprentissage et peut différer d'un problème d'apprentissage à l'autre. Dans notre cas, la fonction d'agrégation est simplement l'union ensembliste ($F_{agg} = \bigcup_{i=1}^{s}$). Les exemples inconsistants sont éliminés. Les exemples correspondant à une même situation sont inconsistants sur les différentes sources si $\exists i, j, k, i \neq j, (e_{i,k}, c) \wedge (e_{j,k}, c') \wedge c \neq c'$. Cette définition de l'inconsistance s'apparente à celle de Blum et Mitchell [Blum et Mitchell, 1998] pour les fonctions compatibles.

Définition 4.2 (Exemples agrégés)
Soit s le nombre de sources.
Soit $E_i = \{(e_{i,k}, c) | k \in [1, p]; c \in C\}$.
$\forall c \in C, F_{agg}(E_1, E_2, \ldots, E_s) = E = \{(e_k, c) | k \in [1, p], e_k = \bigcup_{i=1}^{s} e_{i,k}$ et e_k est consistant$\}$. E contient des faits exprimés dans le langage L_E.

Il serait possible d'enrichir le langage L_E en y incorporant des relations nouvelles entre évènements provenant de sources différentes. Dans ce cas $L_E \supseteq \bigcup_{i=1}^{s} L_{E_i}$. Nous avons décidé de ne pas changer la définition des e_k donc $L_E = \bigcup_{i=1}^{s} L_{E_i}$ et F_{agg} est simplement l'union ensembliste. Toute la connaissance d'agrégation, telle que la spécification de la redondance entre sources, la correspondance entre attributs et les contraintes temporelles sont décrites dans la théorie du domaine T.

Propriété 2 *Soit s le nombre de sources et F_{agg} l'union ensembliste. $L_{i_{ev}}$ dénote la restriction du langage L_i aux prédicats évènementiels.*

1. $couvre(H_{i_c}, (e_{i,k}, c)) \Rightarrow couvre(H_{i_c}, (e_k, c))$.
2. *Soit H_i une hypothèse décrivant la classe c. Soit $c' \in C$ telle que $c' \neq c$.*
   $((L_{i_{ev}} \cap \bigcup_{j=1, j \neq i}^{s} L_{j_{ev}} = \emptyset)$
   $\wedge (\neg couvre(H_{i_c}, (e_{i,k}, c'))))$
   $\Rightarrow \neg couvre(H_{i_c}, (e_k, c'))$.

Le premier point de la propriété 2 signifie que si l'hypothèse apprise H_{i_c} couvre l'exemple *monosource* $(e_{i,k}, c)$ alors H_{i_c} couvre l'exemple agrégé (e_k, c). Le second point

signifie que si les langages L_i n'ont pas de prédicats évènementiels en commun et si l'hypothèse apprise H_{i_c} ne couvre pas l'exemple $(e_{i,k}, c')$ alors H_{i_c} ne couvre pas l'exemple agrégé (e_k, c'). La preuve 2 fournit une démonstration de cette propriété.

Preuve 2 *1. Les exemples provenant des différentes sources sont consistants donc s'il existe* $i = 1, s$ *tel que* $H_{i_c} \wedge e_{i,k} \wedge T \vDash c$ *alors* $H_{i_c} \wedge [e_{1,k} \wedge e_{2,k}, \wedge \ldots \wedge e_{n,k}] \wedge T \vDash c$ *et donc* $H_{i_c} \wedge e_k \wedge T \vDash c$.

2. On a $(L_{i_{ev}} \cap \bigcup_{j=1, j\neq i}^{s} L_{j_{ev}} = \emptyset) \Rightarrow$
$\forall k, \forall j \neq i, \forall c' \in C - \{c\}, \neg(couvre(H_{i_c}, (e_{j,k}, c')))$.
On a $\forall k, \forall c' \in C - \{c\}, \neg(couvre(H_{i_c}, (e_{i,k}, c')))$.
Donc $\forall k, \forall l, H_{i_c} \wedge e_{l,k} \wedge T \nvDash c'$.
De plus, les exemples provenant des différentes sources sont consistants donc $H_{i_c} \wedge [e_{1,k} \wedge e_{2,k}, \wedge \ldots \wedge e_{n,k}] \wedge T \nvDash c'$.
On en conclut donc $H_{i_c} \wedge e_k \wedge T \nvDash c'$.

Apprentissage multisource naïf En l'absence de connaissances sur les sources, une approche naïve d'apprentissage multisource consiste à apprendre directement à partir des exemples agrégés et avec un biais complet qui couvre tout l'espace de recherche relatif au langage agrégé L. Dans ce cas, le langage L_H des hypothèses recherchées est le produit des langages utilisés pour décrire chacune des sources, i.e., l'union de tous les langages L_i auquel on ajoute toutes les relations pouvant exister entre les évènements se produisant sur les différentes sources. Le principal inconvénient de cette approche est la taille de l'espace de recherche résultant. Dans beaucoup de situations, le système d'apprentissage ne pourra le gérer ou nécessitera un temps de calcul trop important. Une solution consiste à spécifier un biais de langage efficace mais cela s'avère être une tâche difficile particulièrement en l'absence de connaissances expertes permettant de limiter le nombre de relations possibles entre les évènements se produisant sur chacune des sources.

Dans la suite, nous proposons une méthode permettant de créer automatiquement un biais syntaxique DLAB pour ICL (cf. section 3.1.8) permettant de résoudre ce problème.

4.1.3 Construction automatique du biais d'apprentissage multisource

Nous proposons une méthode d'apprentissage multisource appelée *apprentissage multisource biaisée* qui consiste à apprendre indépendamment à partir de chaque source, puis à s'appuyer sur les règles apprises pour obtenir des règles multisources en effectuant une nouvelle étape d'apprentissage sur les données agrégées. Dans le cas des arythmies cardiaques, la construction s'apparente à une *synchronisation* des règles monosources. La synchronisation des évènements sur les différentes sources est décrite par le prédicat `suci(X,Y)` (pour *succession immédiate*) qui signifie que l'évènement X est le successeur immédiat de l'évènement Y. L'algorithme 6 décrit la méthode pour un apprentissage à partir de deux sources, elle peut être facilement étendue à l'apprentissage à partir de n sources moyennant une complexité accrue.

Algorithme 6 *1.* **Apprendre** *avec le biais B_1 sur le problème de PLI $< L_1, E_1, T_1, C >$. Soit H_1 l'ensemble des règles apprises pour une classe c donnée.*

2. **Apprendre** *avec le biais B_2 sur le problème de PLI $< L_2, E_2, T_2, C >$. Soit H_2 l'ensemble des règles apprises pour la classe c.*

3. **Agréger** *les ensembles d'exemples E_1 et E_2 pour produire E_3.*

4. ***Construire*** *à partir de toutes paires $(h_{1j}, h_{2k}) \in H_1 \times H_2$ et d'un ensemble de contraintes fournies par l'utilisateur, un ensemble de* bottom clauses *$BT = \{bt_1, bt_2, \ldots, bt_n\}$ tel que chaque clause bt_i construite à partir de h_{1j} et h_{2k} est plus spécifique que h_{1j} et h_{2k}. En particulier bt_i contient tous les prédicats apparaissant dans h_{1j} et h_{2k} ainsi que de nouveaux prédicats relationnels (*suci*) permettant de synchroniser h_{1j} et h_{2k} tout en respectant l'ordonnancement relatif des évènements sur chacune des sources.*

5. ***Construire*** *B à partir de BT. Chaque clause de BT correspond à une partie de l'espace de recherche (que nous appellerons "bloc"). La syntaxe des littéraux dans B doit être telle que les hypothèses envisagées dans l'espace de recherche soient identiques ou plus générales que les* bottom clauses *bt_i et ces hypothèses doivent être ordonnancées dans l'espace de recherche de manière à garantir la séquence des évènements spécifiques à chaque bloc.*

6. ***Apprendre*** *un ensemble de règles multisources pour la classe c avec le biais B sur le problème de PLI $< L, E_3, T_3, C >$ où*
- *L est le langage multisource décrit dans la définition 4.1,*
- *T_3 est un ensemble de règles exprimées dans le langage L.*

4.1.3.1 Construction des *bottom clauses*

Pour construire le biais multisource, nous cherchons, dans un premier temps, à produire un ensemble de clauses particulières que nous appellerons *bottom clauses* en référence à [Muggleton, 1995] car ce sont les clauses les plus spécifiques de l'espace de recherche. La construction s'effectue au point 4 de l'algorithme 6. Pour chaque paire (h_{1j},h_{2k}), l'algorithme peut créer autant de *bottom clauses* qu'il y a d'ordonnancements maintenant l'ordre relatif des évènements sur chacune des sources (cf. l'exemple Figure 4.1 pour une paire (h_1,h_2)). Le nombre de *bottom clauses* pouvant ainsi être créées est C^n_{n+p} où n est le nombre de prédicats évènementiels apparaissant dans la règle h_{1j} et p le nombre de prédicats évènementiels apparaissant dans la règle h_{2k}. Le cardinal de BT est égal au nombre total de *bottom clauses* créées pour chaque paire possible (h_{1j},h_{2k}). Ce nombre peut paraître très élevé dans le cas général où les ensembles de règles H_1 et H_2 contiennent plus d'une clause et où le nombre d'évènements caractéristiques d'une classe donnée pour chaque source est important. En pratique, un nombre significatif de ces *bottom clauses* ne sont pas générées car certaines séquences n'ont aucun sens du point de vue de l'application considérée. Sur l'exemple de la figure 4.1, un expert du domaine interdirait toutes les séquences comprenant un évènement de la source 1 situé entre les évènements *diastole(A,B)* et *systole(C,D)* de la source 2 car elles

Soit h_1 =
```
class(x) :- %séquence P0-R0
        p(P0,normal), qrs(R0,normal),
        pr1(P0,R0,normal), suc(R0,P0).
```
la règle induite pour la classe x sur les données de la source 1.
Soit h_2 =
```
class(x) :- %séquence D0-S0
        diastole(D0,normal), systole(S0,normal),
        suc(S0,D0).
```
la règle apprise pour la même classe x sur les données de la source 2. Les *bottom clauses* générées pour la paire (h_1,h_2) sont :
bt_1 =
```
class(x) :- %séquence P0-D0-R0-S0
        p(P0,normal), diastole(D0,normal),
        suci(D0,P0),
        qrs(R0,normal), pr1(P0,R0,normal),
        suci(R0,D0), suc(R0,P0),
        systole(S0,normal), suci(S0,R0),
        suc(S0,D0).
```
bt_2 =
```
class(x) :- %séquence D0-P0-S0-R0
        diastole(D0,normal), p(P0,normal),
        suci(P0,D0),
        systole(S0,normal), suci(S0,P0),
        suc(S0,D0),
        qrs(R0,normal), pr1(P0,R0,normal),
        suci(R0,S0), suc(R0,P0).
```
...,
bt_{n-1} =
```
class(x) :- %séquence D0-S0-P0-R0
        diastole(D0,normal), systole(S0,normal),
        suc(S0,D0), p(P0,normal),
        suci(P0,S0), qrs(R0,normal),
        pr1(P0,R0,normal), suc(R0,P0).
```
bt_n =
```
class(x) :- %séquence P0-R0-D0-S0
        p(P0,normal), qrs(R0,normal),
        pr1(P0,R0,normal), suc(R0,P0),
        diastole(D0,normal), suci(D0,R0),
        systole(S0,normal), suc(S0,D0).
```

FIG. 4.1 – Exemple de génération d'un ensemble de *bottom clauses* à partir d'une paire de clauses (h_1,h_2) décrivant une même classe x.

sont physiologiquement impossibles. Cette contrainte élimine, en particulier, les *bottom clauses* bt_1 et bt_2 dans l'exemple de la figure 4.1.

Les prédicats *suci* ne sont introduits que s'il y a réellement synchronisation entre les différentes sources. Par exemple pour la clause bt_n, le prédicat `suc(R0,P0)` garde la sémantique qu'il avait en monosource, c'est-à-dire que l'onde `R0` suit directement l'onde

`P0` sur la voie I mais, on autorise la présence de n'importe quelle onde provenant d'une autre source de données entre `P0` et `R0`. La sémantique du prédicat `suci(D0,R0)` est différente puisqu'il ne doit y avoir aucun évènement entre `D0` et `R0`.

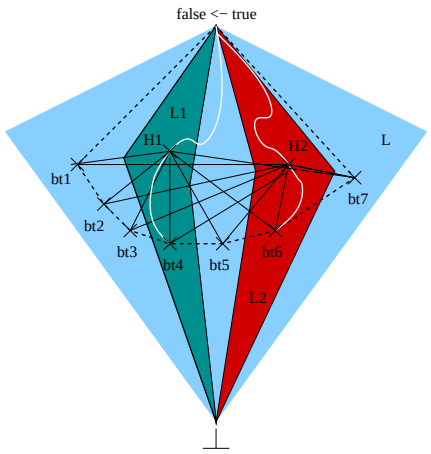


FIG. 4.2 – Construction de l'espace de recherche pour l'apprentissage multisource biaisé

4.1.3.2 Espace de recherche multisource biaisé

À l'instar des *bottom clauses* utilisées dans les systèmes de PLI tels que PROGOL ou ALEPH (cf. Section 3.1.7), chaque *bottom clause* construite précédemment nous sert à limiter un espace de recherche. L'intersection des espaces de recherche définis par l'ensemble des *bottom clauses* est supposée non vide. Pour chaque espace de recherche défini, on recherche des hypothèses plus générales que la *bottom clause* qui borne cet espace.

De la même manière que pour ALEPH, chaque *bottom clause* va servir de "réservoir" de littéraux lors de l'opération d'ajout de littéral de l'algorithme de recherche d'ICL. Pour rendre chaque espace de recherche fini, on ajoute des restrictions aux hypothèses recherchées :

1. pour un espace de recherche donné, le nombre de littéraux dans les hypothèses est limité par le nombre de littéraux dans la *bottom clause* qui borne cet espace;
2. un prédicat relationnel ne peut être ajouté à une hypothèse que si les évènements qu'il met en relation ont déjà été ajoutés à l'hypothèse par l'intermédiaire de prédicats évènementiels;
3. chaque prédicat évènementiel doit être en relation (directe ou indirecte) avec tous les autres prédicats évènementiels (pour éviter les clause non connexes) via un ou plusieurs prédicats relationnels globaux.

Chaque nœud de l'espace de recherche correspond ainsi à une clause sémantiquement acceptable : d'une part, les ordonnancements physiquement impossibles ne sont pas

générés et, d'autre part, les littéraux utilisés sont sémantiquement acceptables du point de vue de la classe considérée puisqu'ils apparaissent dans une règle apprise lors de l'apprentissage monosource.

Dans un second temps, nous construisons un biais DLAB de manière à ce que l'espace de recherche multisource engendré par ce biais corresponde exactement à l'ensemble des espaces de recherche définis par les *bottom clauses* et les contraintes définies précédemment.

La figure 4.2 illustre les différents espaces de recherche en jeu lors d'un apprentissage multisource biaisé. L représente l'espace de recherche multisource naïf. L_1 et L_2 les espaces monosources. L'espace délimité par les pointillés représente l'union des espace de recherche définis par chacune des *bottom clauses*. L'espace de recherche associé à une *bottom clause* correspond à la ligne blanche. Il contient l'ensemble des hypothèses plus générales que la *bottom clause* satisfaisant les contraintes définies précédemment.

4.1.3.3 Construction du biais DLAB

```
1-1:[
%on choisit un et un seul des bloc suivants :
 ...,
1-1:[len-len:[...]],  %(n-1)ieme bloc :  bt(n-1)

1-1:[len-len:[ % btn
  p(P0,normal),qrs(R0,normal),suc(R0,P0),
  0-1:[pr1(P0,R0,normal)]
   ],
    0-1:[len-len:[
   len-len:[
   diastole(D0,normal),suci(D0,R0)
   ],
    0-1:[len-len:[
    len-len:[
    systole(S0,normal),suc(S0,D0)
   ]
]]]]]
]
```

FIG. 4.3 – Exemple de biais créé à partir d'un ensemble de bottom clauses

Le biais est généré automatiquement à partir de l'ensemble des *bottom clauses* créées et de l'ensemble des contraintes fournies par l'utilisateur. Chaque espace de recherche borné par l'une des *bottom clause* est défini par un *bloc* du biais DLAB. La figure 4.3 explicite la partie d'un biais DLAB correspondant à la *bottom clause* bt_n de la figure 4.1. Chaque bloc permet d'induire une hypothèse identique ou plus générale que la *bottom clause* considérée. Tous les espaces de recherche doivent être parcourus pour trouver la ou les hypothèses multisources les plus précises dans l'union des espaces de

recherche. Tous les blocs doivent donc être évalués mais seul un bloc doit être choisi pour apprendre une hypothèse multisource. Cette contrainte est exprimée par l'expression `1-1:[...]` qui parenthèse le biais DLAB exposé Figure 4.3.

A l'intérieur de chaque bloc, le biais est défini de manière à satisfaire toutes les contraintes sur les hypothèses présentées dans la section précédente :

- les seuls prédicats utilisés sont ceux apparaissant dans la *bottom clause* (contrainte 1). Les prédicats relationnels locaux sont tous optionnels.
- tous les prédicats relationnels sont définis après que les prédicats évènementiels correspondants ont été définis (contrainte 2).
- Pour éviter les littéraux non connexes, un prédicat évènementiel est toujours défini conjointement à un prédicat relationnel global (excepté pour les deux premiers prédicats évènementiels qui doivent être définis en même temps pour satisfaire la contrainte 2 avec le premier prédicat relationnel global). Les prédicats évènementiels sont imbriqués de manière à satisfaire la contrainte 3.

4.1.4 Propriétés de l'espace de recherche multisource biaisé

Propriété 3 (Précision) *L'espace de recherche engendré par le biais B de l'algorithme 6 permet d'obtenir des solutions dont la précision est égale ou supérieure à celle obtenue avec H_1 et H_2.*

La mesure de précision de l'apprentissage est celle donnée en section 3.2.1 (soit $\frac{VP+VN}{VP+FN+FP+VN}$) pour évaluer les apprentissages monosources. Intuitivement la propriété 3 exprime le fait que l'espace de recherche défini par le biais B est construit de telle manière qu'il contient également les ensembles des hypothèses de H_1 et de H_2. Grâce à la propriété 2 sur la couverture des exemples multisources, les hypothèses de H_1 et de H_2 peuvent être retenues si aucune hypothèse de l'espace de recherche n'a une meilleure précision.

Propriété 4 (Optimalité) *L'espace de recherche engendré par le biais B de l'algorithme 6 ne permet pas de garantir l'obtention d'une solution optimale pour l'apprentissage multisource.*

L'espace de recherche limité par le biais construit automatiquement au point 5 de l'algorithme 6 ne contient pas forcément la solution optimale répondant au problème multisource. En effet, considérons la fenêtre temporelle englobant tous les évènements participant à une règle monosource, il n'est pas possible d'obtenir une hypothèse multisource mixte (qui contient des relations inter-sources) avec la méthode proposée s'il n'existe pas de recouvrement entre les fenêtres temporelles concernées par les règles provenant des différentes sources.

Propriété 5 (Réduction de l'espace) *L'espace de recherche engendré par le biais B de l'algorithme 6 est plus petit que l'espace de recherche multisource naïf.*

La taille de l'espace de recherche de l'apprentissage à partir du biais DLAB peut être quantifiée explicitement à partir de la définition 18 de [De Raedt et Dehaspe, 1997]. Dans notre cas, la grammaire DLAB est réduite à un seul *DLAB template* (cf. Section 3.1.8.4). La définition peut donc se simplifier comme suit :

Définition 4.3 (Taille de l'espace de recherche) *Soit DGRAM une grammaire DLAB de la forme* $\{H < -B\}$*. La taille de l'espace de recherche induit (*dlab_size(DGRAM)*) est telle que :*

1. $dlab\_size(DGRAM) = ds(H) * ds(B)$ *avec :*
2. $ds(A) = 1$*, où* $A \neq Min - Max : L$
3. $ds(Min - Max : [L_1, \ldots, L_n]) = \sum_{k=Min}^{Max} e_k(ds(L_1), \ldots, ds(L_n))$
4. $e_0(L) = 1$
5. $e_n(s_1, \ldots, s_n) = \prod_{i=1}^{n} s_i$
6. $e_k(s_1, s_2, \ldots, s_n) = e_k(s_2, \ldots, s_n) + s_1 * e_{k-1}(s_2, \ldots, s_n)$ *avec* $k < n$.

Si le terme DLAB est réduit à un atome (par exemple $H = false$), sa taille est égale à 1 (cf. point 2 def. 4.3). Si par contre, le terme DLAB est une formule du type $Min - Max : \{L_1, L_2, \ldots, L_n\}$ le but est de sélectionner k ($k \in [Min, Max]$) sous-formules L_i d'où la sommation (point 3) $\sum_{k=Min}^{Max}$. Le problème du calcul de la taille ne se résume pas à celui bien connu de chercher toutes les façons de tirer k objets (sans remise) dans une urne contenant n objets (combinaisons sans répétition : C_n^k) mais plutôt de chercher dans n urnes ($\{L_1, L_2, \ldots, L_n\}$) qui contiennent au moins 1 objet ($ds(L_i) \geq 1$) les combinaisons qui utilisent au plus un objet de chaque urne. La formule générale pour calculer la taille est donnée par $e_k(s_1, s_2, \ldots, s_n)$ où e_k est la fonction élémentaire symétrique de degré k et s_i est le nombre d'objets dans chaque urne. Les points 4 et 5 sont des cas particuliers. Le premier signifie qu'il n'y a qu'un seul moyen de choisir 0 objet. Le point 5 (on prend exactement un objet de chaque urne) indique que le nombre de combinaisons pour chaque urne étant égal à s_i, le nombre total de combinaison est le produit des s_i.

Intuitivement, plus le nombre de formules de type $Min - Max : \{L_1, L_2, \ldots, L_n\}$ imbriquées est grand et plus la différence $Max - Min$ est importante et plus la taille du biais est grand.

Pour comparer l'espace de recherche multisource naïf et l'espace de recherche multisource biaisé, il faut donc comparer l'imbrication relative de tous les prédicats (évènementiels, relationnels locaux et relationnels globaux) apparaissant dans les deux biais. Dans le cas naïf, le biais doit être construit de manière à produire toutes les séquences d'évènements possibles (beaucoup d'imbrications), alors que dans le cas biaisé les séquences sont déjà construites, il faut simplement choisir une séquence parmi celles proposées. Dans ces séquences, les prédicats évènementiels (obligatoire) utilisés sont les mêmes que dans le langage multisource naïf. Cependant, le nombre de prédicats relationnels (optionnels) est beaucoup plus important dans le cas naïf que dans le cas biaisé, puisque dans le cas naïf on prend on compte tous les prédicats relationnels de chacune des sources (plus éventuellement d'autres prédicats relationnels *multisources* si

on possède des connaissances sur le domaine) alors que dans le cas biaisé, on ne garde qu'un sous ensemble correspondant au prédicats présents dans les règles monosources apprises (plus les prédicats *suci* qui sont obligatoires et vont de pair avec les prédicats évènementiels). L'espace de recherche biaisé est donc plus petit que l'espace multisource naïf.

4.2 Résultats

Les résultats monosources présentés en section 3.2.2.5 sont utilisés pour la construction de l'espace de recherche multisource biaisé et pour la comparaison avec les résultats multisources obtenus. Diverses expériences multisources sont présentées dans cette section. Dans une première partie, nous utilisons les données de la voie I et toutes celles disponibles pour la pression. Au vu des résultats monosources, ce sont les données qui permettent d'obtenir les meilleurs précisions en apprentissage et en test. Nous comparons sur ces données les performances de l'apprentissage multisource biaisé avec les apprentissages monosources, d'une part, et de l'apprentissage multisource biaisé avec l'apprentissage multisource naïf, d'autre part. Dans une seconde partie, nous utilisons les données de la voie V sans prendre en compte la forme du QRS et des données réduites sur la pression (on ne prend en compte que la systole). Ces données, plus complémentaires que les précédentes, permettent de mettre en valeur l'apport de l'utilisation de plusieurs sources de données pour augmenter la précision des apprentissages. Une dernière expérience est réalisée à partir des données de la voie V avec des données de la voie I en prenant en compte seulement l'onde P. Les résultats sur ces données très complémentaires confirment les conclusion de l'expérience précédente. La totalité des règles multisources apprises est donnée en Annexe C.

4.2.1 Protocole expérimental multisource biaisé

Lors des validations croisées de type *leave-one-out* effectuées lors des apprentissages monosources, la base d'apprentissage et la théorie apprise varient selon l'exemple qui a été retiré pendant une étape de validation croisée donnée. Le biais d'apprentissage, la théorie du domaine et les paramètres d'apprentissage pour chacune des sources restent par contre identiques quelle que soit l'étape de validation croisée puisqu'ils ne dépendent pas de la base d'apprentissage.

Pour l'apprentissage multisource biaisé, le problème est légèrement différent puisque le biais d'apprentissage est construit à partir d'apprentissages monosources, il doit donc être différent à chaque étape de validation croisée au même titre que les théories monosources apprises durant la validation croisée. L'algorithme utilisé pour la validation croisée multisource est le suivant :

Algorithme 7 (Leave-one-out multisource biaisé)
Soit $E = \{(e_k, c) | k = 1, p; e_k = \bigcup_{i=1}^{s} e_{i,k}\}$ *l'ensemble des exemples multisources.*
Pour une classe c donnée :

1. *Effectuer pour chaque source i, p étapes de validation croisée pour apprendre p théories. À chaque étape, un exemple est retiré de la base d'exemples et gardé pour le test. Pour chaque source, enlever toujours les exemples dans le même ordre $e_{i,1}, e_{i,2}, \ldots e_{i,p}$.*
2. *Créer p biais multisources à partir des p théories apprises pour chaque source.*
3. *Créer p ensembles d'exemples multisources E_p tels que $E_p = E - (e_p, \_)$. L'exemple retiré peut être une interprétation étiquetée par la classe c ou non.*
4. *Effectuer p apprentissages multisources.*
   La précision (PrecAp) est la moyenne des précisions de chacun des apprentissages (la formule est donnée en Section 3.2.1). La précision (PrecT) est la moyenne des précisions obtenues lors des tests des p exemples multisources retirés sur les p théories multisources apprises. Si l'exemple retiré est (e_p, c), le test vaut 1 *si l'exemple est une interprétation de la théorie,* 0 *sinon; si l'exemple retiré est $(e_p, c'), c \neq c'$, le test vaut* 1 *si l'exemple n'est pas une interprétation de la théorie,* 0 *sinon.*

Les paramètres d'apprentissage et la théorie du domaine T multisource ne varient pas d'une étape de validation croisée à l'autre. Pour avoir des résultats conformes à la théorie, les paramètres d'apprentissage doivent être les mêmes lors des apprentissages monosources et multisources mise à part la taille du faisceau de recherche qui doit être au moins égal au nombre de *bottom clauses* créées (pour être sûr que tout l'espace de recherche défini par le biais sera parcouru) lors de l'apprentissage multisource biaisé. Une manière simpliste de concevoir la théorie du domaine automatiquement pour l'apprentissage multisource est de définir T comme l'union des T_i auquel on ajoute la définition du prédicat *suci*.

4.2.2 Comparaison des performances d'apprentissage pour la voie I et la voie de pression

Le tableau 4.1 donne une idée de la taille des espaces de recherche explorés lors de chaque apprentissage. Nous rappelons que le nombre de nœuds (*Nds*) correspond au nombre de nœuds visités dans l'espace de recherche et les temps de calcul (*Tps*), correspondent à des mesures en secondes de temps CPU effectuées sur un ordinateur SUN Ultra-Sparc 5. Les seconds temps de calcul donnés pour l'apprentissage biaisé prennent en compte les temps des apprentissages monosources.

On remarque que l'espace de recherche parcouru lors de l'apprentissage multisource biaisé est très inférieur à celui parcouru pour les autres apprentissages, ce qui est conforme à la propriété 5, notamment en comparaison avec l'apprentissage naïf. En outre, l'espace multisource biaisé est en moyenne pour les sept classes, 50 fois plus petit que l'espace multisource naïf.

En revanche, le temps de calcul n'est pas corrélé au nombre de nœuds puisque le test de couverture en chaque nœud de l'espace de recherche est beaucoup plus complexe pour les apprentissages multisources que pour les apprentissages monosources, notamment car les prédicats utilisés pour les apprentissages multisources sont plus complexes ou

	monosource : ECG		monosource : ABP	
	Nds	Tps *	Nds	Tps *
rythme sinusal (rs)	4634	3660	5475	1415
extra-systole ventriculaire (esv)	1654	189	19971	2362
bigéminisme (bige)	708	160	7365	337
doublet ventriculaire (doublet)	790	125	9840	596
tachycardie ventriculaire (tv)	428	109	10833	691
tachycardie supra-ventriculaire (tsv)	232	105	1326	438
fibrilation auriculaire (fa)	64	4	6477	1923

	multisource naïf		multisource biaisé		
	Nds	Tps	Nds	Tps	Tps ($\supset$ *)
rythme sinusal (rs)	12889	36984	415	471	5546
extra-systole ventriculaire (esv)	12887	22347	2020	951	3502
bigéminisme (bige)	15103	22298	77	23	520
doublet ventriculaire (doublet)	19084	7832	346	105	826
tachycardie ventriculaire (tv)	4317	7046	140	205	1005
tachycardie supra-ventriculaire (tsv)	2155	14544	75	114	657
fibrilation auriculaire (fa)	135	325	16	47	1974

TAB. 4.1 – Nombre de nœuds visités lors de l'apprentissage et temps de calcul

mettent en jeu plus de relations. Les temps de calcul de l'apprentissage multisource biaisé en comptant les temps des apprentissages monosources sont, en moyenne pour les sept arythmies, 13 fois inférieurs à ceux des apprentissages multisources naïfs. À noter cependant le cas très particulier de la *fibrillation auriculaire* (*fa*) pour lequel l'apprentissage multisource naïf est plus rapide que l'apprentissage multisource biaisé. En effet, il est très facile de différencier une *fa* des six autres arythmies choisies pour nos expériences en se basant seulement sur les informations de l'ECG puisque seule la *fa* se caractérise par une absence d'onde P et des complexes QRS normaux. L'apprentissage multisource naïf utilise uniquement des informations provenant de l'ECG alors que l'apprentissage multisource biaisé pâtit du temps de calcul beaucoup plus important nécessaire pour discriminer la *fa* par rapport aux autres arythmies sur les données de pression. Ce cas particulier est directement lié à la spécificité de nos données.

On peut remarquer que les arythmies *tv, tsv* et *fa* dont les caractéristiques sont très éloignées du rythme normal, sont plus faciles à discriminer que les autres arythmies ce qui se traduit par de meilleures performances lors des apprentissages monosources et multisources.

Le tableau 4.2 donne les résultats de l'évaluation des performances des apprentissages monosources, d'une part, et des apprentissages multisources effectués avec et sans la méthode de réduction de l'espace de recherche et de génération de biais, d'autre part. Nous rappelons que la signification des colonnes *PrecAp* et *PrecT* est donnée dans l'algorithme 7 et la colonne *Nbcycles* correspond au nombre de cycles cardiaques décrits par la ou les règles apprises sur l'ensemble de la base d'apprentissage.

Les résultats des apprentissages monosources sont très bons, notamment sur l'ECG où la mesure de précision sur la base d'apprentissage donne une précision maximale de 1

	monosource ECG			monosource ABP		
	PrecAp	PrecT	Nbcycles	PrecAp	PrecT	Nbcycles
rs	0.62	0.60	7	1	0.98	5
esv	0.981	0.94	5	0.990	0.76	4/5/3
bige	1	0.98	4	0.962	0.80	3
doublet	1	1	4	0.942	0.78	4/5
tv	1	1	3	0.996	0.86	6/4
tsv	1	0.98	3	1	1	3
fa	1	1	2	0.998	0.86	3/4
	multisource naïf			multisource biaisé		
	PrecAp	PrecT	Nbcycles	PrecAp	PrecT	Nbcycles
rs	0.98	0.96	4	1	0.98	5
esv	0.965	0.86	4	0.999	0.88	5/5
bige	0.997	0.86	3/3	1	0.98	4
doublet	0.98	0.84	5	1	1	4
tv	1	1	3	1	1	3
tsv	1	0.98	2	1	1	3
fa	1	1	2	1	1	2

TAB. 4.2 – Résultats de la validation croisée pour les apprentissages monosources et multisources sur la voie I et la voie de pression

pour toutes les classes excepté pour le rythme sinusal et l'extrasystole ventriculaire (*rs* et *esv*). Cette précision de 1 se retrouve pour l'apprentissage du rythme sinusal et de la tachycardie supra-ventriculaire sur la source ABP. Ces résultats étant presque parfaits du point de vue de l'apprentissage, il est illusoire d'espérer faire mieux avec l'apprentissage multisource et en particulier avec l'apprentissage multisource biaisé. Pour cette expérience, lors de l'apprentissage multisource biaisé, le système de PLI s'est comporté dans la majorité des cas comme un système de vote entre les différentes sources pour retrouver la règle de précision parfaite. Les règles apprises par apprentissage multisource biaisé sont en particulier, identiques aux règles apprises sur l'ECG seul pour les cas *doublet, tv, tsv* et *fa* et à celle apprise sur la source ABP pour le *rs*. La précision lors de l'apprentissage multisource biaisé est donc, dans ces cas, égale à la meilleure des précisions des deux apprentissage monosources. Pour l'*esv*, les règles apprises lors de la validation croisée sont parfois des règles ne prenant en compte que des évènements de pression, parfois seulement des évènements de l'ECG et parfois des règles mixtes (i.e. qui prennent en compte des évènements des deux sources) ce qui explique la valeur de la précision en test (PrecT) médiane par rapport aux valeurs monosources pour l'apprentissage biaisé.

On peut également noter que, pour cette expérience, la précision des apprentissages multisources biaisés est toujours supérieure ou égale à celle des apprentissages multisources naïfs.

4.2.3 Comparaison des performances sur des données complémentaires

Les sources de données sur lesquelles nous travaillons sont bien connues des cardiologues (en particulier l'ECG) et nous possédons de nombreuses informations pour constituer les biais d'apprentissage. Ces informations telles que les attributs intéressants pour décrire une source, les prédicats évènementiels, les contraintes entre les évènements provenant des différentes sources, expliquent en partie les très bons résultats obtenus pour les apprentissages monosources dans l'expérience précédente. Ces résultats sont également expliqués par le nombre réduit d'exemples de la base d'apprentissage et l'absence d'exemples "bruités". L'apport de l'utilisation conjointe des deux sources pour améliorer les performances d'apprentissage peut paraître faible dans ce contexte.

4.2.3.1 Apprentissages voie V sans forme du QRS et voie ABP sans diastole

	monosource ECG			monosource ABP			multisource biaisé		
	PrecAp	PrecT	Nbcy	PrecAp	PrecT	Nbcy	PrecAp	PrecT	Nbcy
rs	0.48	0.44	6	1	0.98	5	1	1	5
esv	0.52	0.46	6/6	0.928	0.80	5	0.942	0.76	8/6/5
bige	0.98	0.90	6	0.997	0.84	4/5	0.999	0.88	4/5
doub	0.851	0.78	5/5	0.982	0.86	4/5	0.993	0.88	4/4
tv	0.883	0.72	6	0.93	0.82	4/4	0.97	0.8	6/4/7
tsv	0.96	0.96	6	0.96	0.82	4	0.99	0.96	8
fa	0.977	0.9	4/5	0.978	0.78	5/5	0.984	0.82	5/4

TAB. 4.3 – Résultats de le validation croisée pour les apprentissages monosources et multisource biaisé sans information sur l'onde P, la forme du QRS ni sur la diastole

Nous avons décidé de nous placer dans une situation plus réaliste dans laquelle la quantité d'informations disponibles sur les sources est réduite. En particulier, nous avons décidé de ne pas prendre en compte l'onde P ni la forme du complexe QRS pour l'ECG (nous travaillons avec les données de la voie V sans la forme du QRS), et la diastole pour la pression. Le fait de ne pas prendre en compte l'onde P a un sens du point de vue du traitement du signal puisque cette onde est encore difficilement détectable automatiquement. La diastole correspond d'un point de vue médical à l'intervalle de temps entre deux systoles (le moment ou les ventricules se vident). Dans la modélisation symbolique que nous avons choisie, la diastole correspond simplement au point de pression le plus bas entre deux systoles. Ce point peut également être difficile à détecter par des algorithmes de traitement du signal.

Les résultats de cette expérience sont donnés Table 4.3. Les résultats monosources obtenus sur l'ECG restent très acceptables sur des données moins informatives excepté pour le rythme sinusal et l'extrasystole ventriculaire pour la voie V (pour les raisons énoncées dans le chapitre précédent). L'ensemble des règles apprises est fourni en An-

nexe C. La validation croisée de type *leave-one-out* n'a pas pu être menée à terme pour l'apprentissage multisource naïf à cause des temps excessifs des 50 apprentissages nécessaires. Les résultats ne sont donc pas fournis Table 4.3 pour ce type d'apprentissage. Les règles obtenues pour les apprentissages multisources biaisés sont mixtes pour cinq arythmies (*esv, doublet, tv, tsv* et *fa*), c'est-à-dire que les règles possèdent à la fois des éléments de l'ECG et de la pression. Les deux sources sont donc utiles pour caractériser les arythmies puisque, dans le cas contraire, les règles obtenues avec la méthode naïve ne contiendraient que des événements de la source la plus pertinente.. Dans les deux autres cas (*rs* et *bige*), les règles apprises sont les mêmes que celles apprises sur la source ABP seule. Les mesures de précision pour l'apprentissage sont, comme le montre la théorie, supérieures ou égales aux résultats monosources.

```
class(tsv,[0,0,0,0,0,5,2],[8,7,10,6,7,0,5]):-
qrs(R0),  qrs(R1),
suc(R1,R0), qrs(R2),
suc(R2,R1), rr1(R1,R2,short),
rythm(R0,R1,R2,regular), qrs(R3),
suc(R3,R2), rr1(R2,R3,short), qrs(R4),
suc(R4,R3), rr1(R3,R4,short).
```

FIG. 4.4 – Exemple de règle apprise pour la classe *tsv* sur la voie V sans prendre en compte la forme du QRS

```
class(tsv,[0,0,0,0,1,5,1],[8,7,10,6,6,0,6]):-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,nul,normal),
ss1(Sys0,Sys1,short),
systole(Sys2), suc(Sys2,Sys1),
amp_ss(Sys1,Sys2,neg,long),
ss2(Sys0,Sys2,short),
systole(Sys3), suc(Sys3,Sys2),
```

FIG. 4.5 – Exemple de règle apprise pour la classe *tsv* sur la pression sans diastole

```
class(tsv) :-
qrs(R0,normal),
equal(R0,Suc0), cycle_qrs_sys(R1,normal,Sys1),
equal(Sys1,Suc1), suc(R1,Suc0), rr1(R0,R1,short),
amp_ss(Sys0,Sys1,nul,normal), cycle_qrs_sys(R2,normal,Sys2),
equal(Sys2,Suc2), suci(R2,Suc1), amp_ss(Sys1,Sys2,neg,long).
cpu(97283.9), local(5,0,0,45), total(5,0,0,45)
```

FIG. 4.6 – Exemple de règle apprise pour la *tsv* par apprentissage multisource naïf

À titre d'exemple, nous donnons Figures 4.4, 4.5, 4.6 et 4.7 les règles apprises pour la classe *tsv* lors des apprentissages monosources sur la voie V et la pression, d'une

```
rule(tsv):-
        qrs(R0), qrs(R1), suc(R1, R0),
        qrs(R2), rr1(R1, R2,short), rythm(R0, R1, R2,regular), suc(R2, R1),
        qrs(R3), rr1(R2, R3,short), suc(R3, R2),
        systole(Sys0), suci(Sys0, R3),
        qrs(R4), suci(R4, Sys0),
        systole(Sys1), amp_ss(Sys0, Sys1,nul,normal), suci(Sys1, R4).
cpu(125.07), local((5,0,0,45)), total((5,0,0,45))
```

FIG. 4.7 – Exemple de règle apprise pour la *tsv* par apprentissage multisource biaisé

part, et lors des apprentissages multisources naïfs et biaisés, d'autre part. Les règles monosources couvrent toutes deux des exemples des autres classes : deux *fa* pour la règle monosource apprise sur la voie V et une *tv* et une *fa* pour la règle apprise sur les données de pression. Les règles multisources sont correctes et complètes et la règle multisource obtenue avec la méthode biaisée à été apprise en 777 fois moins de temps que la règle multisource obtenue avec la méthode naïve.

4.2.3.2 Apprentissages séparés pour l'onde P et le QRS

Pour mettre en évidence l'intérêt de l'utilisation de plusieurs sources dans un contexte non bruité, il peut être intéressant de tester l'apprentissage quand les données provenant des différentes sources sont réellement complémentaires. Nous n'avons pas connaissance de l'existence d'une telle base de données multisources. Nous avons donc choisi d'utiliser une source virtuelle fournissant des données sur l'onde P seule en plus de la voie V sans information sur la forme du QRS. Cette simulation a également un sens d'un point de vue médical car l'EECG (Électrocardiogramme mesuré au niveau de l'œsophage) est, par exemple, une source fournissant seulement des données sur l'activité auriculaire (cf.[Hernández *et al.*, 1999]). Une première étude a été conduite et les résultats sont donnés Table 4.4.

Ces résultats montrent une réelle amélioration de la précision entre les apprentissages monosources et les apprentissages multisources sur des données non bruitées, non seulement au niveau de l'apprentissage mais aussi au niveau du test. On remarque que les apprentissages multisources naïfs couvrant tout l'espace de recherche multisource donnent des résultats comparables à ceux de l'apprentissage multisource biaisé.

4.2.4 Discussion

Pour obtenir des résultats multisources selon la méthode biaisée présentée dans ce chapitre, les règles monosources apprises indépendamment doivent posséder des prédicats relationnels communs décrivant une relation d'ordre (ici la relation temporelle de succession). Dans le cas contraire, il n'y a pas d'ordonnancement possible entre les évènements se produisant sur les différentes sources, et les résultats multisources seront identiques aux résultats monosources. De même, si les données provenant des différentes sources sont connues comme étant redondantes *a priori*, la méthode biaisée effectue un apprentissage multisource inutile puisqu'un simple vote sur la source ayant la meilleure

	Onde P seule			QRS sans forme		
	PrecAp	PrecT	Nbcycles	PrecAp	PrecT	Nbcycles
rs	0.62	0.62	7	0.38	0.36	5
esv	0.76	0.74	6	0.42	0.4	5
bige	1	0.98	4	0.96	0.92	4
doublet	0.78	0.72	3/7	0.92	0.92	4
tv	0.92	0.9	2	0.901	0.84	5
tsv	1	1	2	0.96	0.94	5
fa	0	0	0	0.94	0.86	4/4
	multisource naïf			multisource biaisé		
	PrecAp	PrecT	Nbcycles	PrecAp	PrecT	Nbcycles
rs	0.6	0.6	7	0.63	0.44	11
esv	1	0.94	6	0.98	0.8	6
bige	1	0.98	4	1	0.86	4
doublet	1	1	4	1	0.86	6
tv	1	1	4	0.92	0.72	2
tsv	1	1	2	1	0.86	2
fa	0.98	0.94	7	0.94	0.96	4

TAB. 4.4 – Ondes ECG séparées

mesure de précision lors de l'apprentissage suffirait. En revanche, lorsque les sources sont complémentaires, la méthode biaisée donne de très bons résultats par rapport aux apprentissages monosources et des mesures de précision proches de celles obtenues avec la méthode naïve.

Notons que les règles multisources mixtes sont a priori plus sensibles que les règles monosources à la présence de bruit sur le signal puisqu'elles nécessitent des données claires sur les deux sources à la fois pour bénéficier d'une bonne précision en test. De plus, les règles multisources apprises mettent en jeu un nombre important de cycles cardiaques ce qui les rend d'autant plus sensibles aux erreurs de détection des algorithmes de traitement du signal qui fournissent les évènements symboliques. De plus amples expériences sont réalisées dans un contexte bruité dans la section suivante pour estimer plus précisément les performances de ces règles.

Enfin, l'algorithme actuel semble très efficace sur un petit volume de données mais des expériences complémentaires doivent être réalisées sur des bases d'apprentissage multisources plus volumineuses pour valider totalement la méthode multisource biaisée.

4.3 Adaptation de CALICOT pour l'apprentissage multisource

Cette section présente des expériences de reconnaissance des arythmies cardiaques en temps réel avec le système de monitoring cardiaque CALICOT (cf. chapitre 2) sur des données réelles bruitées artificiellement. Les règles utilisées sont les règles monosources apprises avec ICL présentées au chapitre 3 et en Annexe C et les règles multisources présentées dans la section précédente. L'utilisation de l'ensemble des règles dans le système de monitoring ne semble pas très pertinente puisque les résultats obtenus avec la voie I seule sont supérieurs à tous les autres résultats d'apprentissage. L'intérêt d'avoir à disposition cet ensemble de règles est bien évidemment d'utiliser la ou les sources les plus adaptées en fonction du bruit présent sur les sources de données. Pour caractériser ce bruit et sélectionner les chroniques intéressantes, le système de monitoring doit se munir d'un module de pilotage. La création d'un tel module fait l'objet de la thèse de François Portet [Portet, 2005] et a abouti à la création du système IP-CALICOT. Nous présentons brièvement ce système et les modifications à apporter pour utiliser de manière pertinente plusieurs sources de données afin d'augmenter la fiabilité de la reconnaissance d'arythmies cardiaques en ligne.

4.3.1 Des règles PROLOG aux chroniques CRS

Si les règles apprises pour l'ECG lors du projet PISE (cf. Chapitre 2) étaient facilement transposables en chroniques CRS, les nouvelles règles apprises pour cette étude posent des problèmes de conversion en syntaxe CRS. En effet, CRS ne permet pas de manipuler des attributs numériques (seulement des évènements datés). En particulier, il n'est pas possible de représenter en langage CRS un prédicat représentant une différence d'amplitude de pression tel que `amp_ss(Sys0, Sys1, nul, normal)` puisqu'il faudrait pour cela que CRS connaisse et mémorise l'amplitude de la systole `Sys0`, l'amplitude de la systole `Sys1` et vérifie que la différence des deux amplitudes appartient à un intervalle `normal` précédemment défini. Or, la syntaxe des évènements en entrée de CRS est limité à :
`date NomEvenement[attributs symboliques de NomEvenement]`
et les attributs des évènements ne peuvent être que symboliques pour des raisons d'efficacité lors de la reconnaissance de chroniques.

Pour faire face à ce problème, le module d'abstraction temporelle de CALICOT doit faire un pré-traitement des données pour transformer les valeurs numériques des attributs en valeurs symboliques et générer un évènement fictif pour chacun de ces attributs. Un exemple d'une séquence d'évènements contenant ces évènements fictifs utilisée en *entrée* du reconnaisseur de chroniques est donné Table 4.5. Les seuils utilisés pour la transformation en données symboliques sont ceux définis dans la section 3.2.2.2.

Un autre problème se pose pour les prédicats tels que `rythm(R0,R1,R2,regular)` dont la définition en PROLOG est :

```
rythm(R0,R1,R2,regular):-
```

...	
838	p_wave[normal]
1005	qrs[normal]
1116	diastole[normal]
1117	amp_dd[long]
1247	systole[normal]
1248	amp_ss[normal]
1583	qrs[anormal]
1705	diastole[short]
1706	amp_dd[long]
1827	systole[short]
1828	amp_ss[normal]
...	

TAB. 4.5 – Évènements multisource en entrée du reconnaisseur de chroniques

```
rr1(R0,R1,I1),
rr2(R1,R2, I2),
I1=I2.
```

Pour vérifier que le rythme est régulier dans la syntaxe CRS, il faut tester l'appartenance de la différence `R1-R0` à un intervalle `I1`, puis l'appartenance de `R2-R1` à un intervalle `I2` (cf. la syntaxe des chroniques CRS définie en section 2.2.2). Or, CRS ne donne pas la possibilité de laisser les intervalles symboliques `I1` et `I2` sous forme de variable pour pouvoir tester `I1 = I2`, donc la représentation de ce type de prédicat est actuellement impossible. Pour contourner ce problème, la même méthode que pour les amplitudes de pression peut être utilisée : un évènement de type `rythm` qui caractérise le rythme suit chaque évènement de type `qrs`. Ce type de codage oblige cependant le module d'abstraction temporelle à mémoriser l'instant d'apparition des deux évènements de type `qrs` qui précèdent l'évènement courant. Ce dernier problème, plus facile à résoudre car n'augmentant pas la combinatoire du problème de satisfaction de contraintes dans CRS, pourra faire l'objet d'une modification directe du reconnaisseur de chroniques.

4.3.2 Intégration des aspects multisources dans IP-CALICOT

La figure 4.8 présente le système IP-CALICOT étendu pour prendre en compte plusieurs sources de données.

4.3.2.1 Présentation d'IP-CALICOT

IP-CALICOT (*Integrated Piloting and Cardiac Arrhythmias Learning for Intelligent Classification of On-line Tracks*) est une amélioration du système CALICOT dans lequel un module de pilotage a été ajouté pour pouvoir reconfigurer le système de monitoring

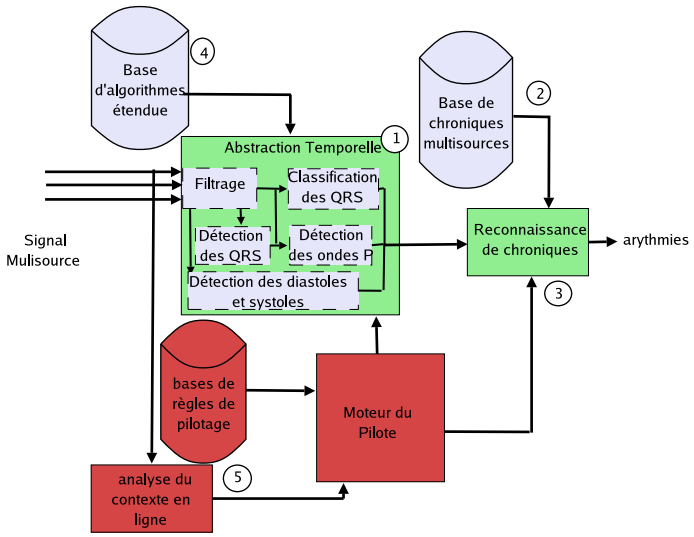


FIG. 4.8 – Architecture du système multisource IP-CALICOT

en fonction de la qualité des signaux en entrées du système [Portet *et al.*, 2005a] et des arythmies en cours de reconnaissance. Le pilote agit i) au niveau de la reconnaissance des arythmies (cf. point 3 Figure 4.8), ii) au niveau de l'abstraction temporelle (cf. point 1 Figure 4.8) et iii) au niveau des algorithmes de traitement du signal utilisés (cf. point 4 Figure 4.8).

Pilotage du reconnaisseur de chroniques Une arythmie est diagnostiquée si l'on peut reconnaître sur le signal les caractéristiques de cette arythmie. Dans CALICOT, les attributs de l'ECG sont extraits en permanence et fournis, via le module d'abstraction temporelle au reconnaisseur de chroniques (cf. Section 2.2). Cependant, dans certains contextes, un nombre réduit de ces attributs peut être suffisant pour diagnostiquer une arythmie. Par exemple, en présence d'un rythme rapide, on peut diagnostiquer une tachycardie ventriculaire ou une tachycardie supra-ventriculaire. On peut différencier ces deux arythmies en se reposant uniquement sur la description des ondes P, mais une analyse de la morphologie du complexe QRS, moins coûteuse en temps que la détection de l'onde P, peut s'avérer suffisante pour les discriminer. Dans ce contexte, le pilotage d'algorithme consiste à choisir les modèles de chroniques dont le langage est adapté au niveau d'abstraction fourni par le module d'abstraction temporelle et lui même peut être adapté en fonction du contexte arythmique du patient (par exemple si le rythme cardiaque est rapide). Pour pouvoir effectuer ce choix des modèles de chroniques, une hiérarchie de chroniques doit être apprise à partir d'exemples exprimés dans différents langage :

- un langage qui décrit la voie I de l'ECG avec les ondes P, les QRS, leur forme respective et les relations qui les lient;
- un langage qui décrit la voie V de l'ECG avec les QRS, leur forme et les relations qui les lient;
- un langage qui décrit la voie V avec les QRS et les relations qui les lient mais sans prendre en compte la classification (normale/anormale) de ces QRS.

Pilotage de l'abstraction temporelle Comme expliqué dans le paragraphe précédent, toutes les tâches associées au module d'abstraction temporelle de CALICOT (la détection et la classification des ondes P et QRS, cf. point 1 Figure 4.8) ne sont pas nécessairement utiles à un instant donné. Par exemple, lorsque le signal est trop bruité, la détection de l'onde P est vouée à l'échec et entraîne beaucoup d'erreurs. Cette tâche est donc pénalisante car elle fournit de fausses informations au reconnaisseur de chroniques. Pour fonder le diagnostic sur des informations fiables, le pilote peut ainsi activer ou désactiver certaines tâches en fonction du contexte.

Pilotage des algorithmes de traitement du signal Dans le module d'abstraction temporelle, chaque tâche est réalisée par un ou plusieurs algorithmes de traitement du signal inter-connectés. Il existe dans la littérature plusieurs algorithmes, dont les performances varient en fonction du type de bruit sur le signal, développés pour réaliser chacune de ces tâches. Le pilote peut ainsi choisir (cf. [Portet *et al.*, 2005b] pour le choix de l'algorithme dans le cas de l'ECG), les algorithmes et leur paramètres les plus adaptés aux tâches à accomplir en fonction du contexte courant.

4.3.2.2 Adaptation du système

Le module d'abstraction temporelle (cf. points 1 et 4 Figure 4.8) doit être étendu pour prendre en compte le signal de pression. Pour cela, de nouveaux algorithmes de détection des diastoles et des systoles doivent être inclus dans la base d'algorithmes. Il existe également dans la littérature, plusieurs algorithmes pour effectuer cette tâche, comme celui de Hoeksel et. al. [Hoeksel *et al.*, 1997] à base de filtres adaptés pour les valeurs de pression systolique et diastolique. Cet algorithme offre de bonnes performances mais n'a pas encore été inclus dans le système. Dans notre étude, nous avons simulé un détecteur d'évènements diastoliques et systoliques à partir des annotations du signal fournies par la base de données.

Nous avons également enrichi la base de chroniques (cf. point 2 Figure 4.8) avec l'ensemble des règles répertoriées en Annexe C et préalablement transformées en chroniques CRS. Le principe du pilotage est donné Figure 4.9. Les points 2 et 3 sont des agrandissements des mêmes points sur la Figure 4.8.

4.3.3 Résultats préliminaires

Nous présentons deux façons possibles de tirer parti de l'utilisation de plusieurs sources pour améliorer le diagnostic des arythmies. Dans le cas de sources bruitées,

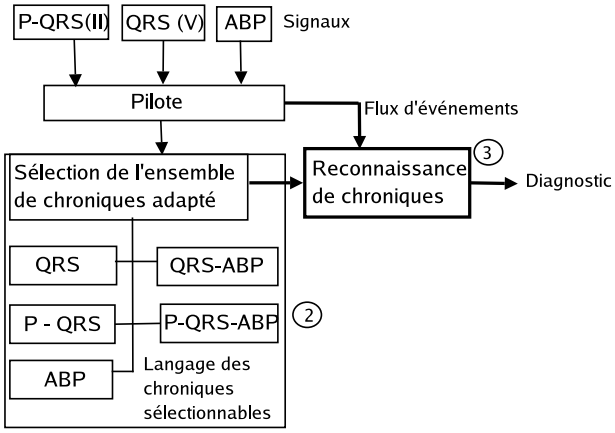


FIG. 4.9 – Principe du pilotage des sources

le pilote choisit, en fonction du bruit détecté, la ou les sources dont l'information est la plus satisfaisante. Si les sources ne sont pas bruitées, deux cas de figure sont pris en compte : si les sources sont complémentaires, il peut être très intéressant d'utiliser des chroniques qui incluent des évènements provenant des différentes sources combinées pour augmenter la fiabilité du diagnostic. Dans le cas où les sources sont redondantes, la source qui apporte le plus d'informations doit être retenue.

Les résultats des reconnaissances des arythmies sont donnés dans des matrices de confusion. Ces matrices répertorient le nombre de reconnaissances correctes et de confusions pour chaque type d'arythmies en fonction des annotations fournies par le signal test. Les lignes des matrices correspondent aux arythmies détectées et les colonnes aux arythmies annotées. Dans la suite, VP (vrai positif) dénote le nombre de bonnes reconnaissances, FN (faux négatif) le nombre d'arythmies non reconnues et FP (faux positif) le nombre de confusions. Pour chaque expérience, la sensibilité et la prédictivité positive du système de monitoring sont évaluées. Ces mesures sont données par les formules suivantes : $SENS = \frac{VP}{VP+FN}$, $Pred+ = \frac{VP}{VP+FP}$. La sensibilité donne la probabilité qu'un rythme observé soit bien diagnostiqué. La prédictivité positive donne le taux de bonne détections pour chaque arythmie parmi toutes les détections (c'est une mesure de spécificité des alarmes). Le principe de construction de ces matrices est donné dans [Carrault *et al.*, 2003] (pages 254 et 255).

4.3.3.1 Les données

Les signaux utilisés proviennent des bases de données MIMIC (Multi-parameter Intelligent Monitoring for Intensive Care [Moody et Mark, 1996]) pour l'apprentissage et de la base MGH/MF (Massachusetts General Hospital/Marquette Foundation) pour les tests. Ces données ont été annotées par des cardiologues. Pour une arythmie donnée, le médecin ne donne qu'une seule annotation par battement cardiaque (pour nous, il s'agira d'un ensemble P-QRS-Diastole-Systole quand toutes les sources sont présentes).

L'apprentissage a été fait sur les cinquante exemples utilisés dans les sections précédentes mais seules six arythmies ont été considérées pour le test : le bigéminisme, le doublet ventriculaire, l'extrasystole ventriculaire, la tachycardie ventriculaire, la fibrillation auriculaire et le rythme sinusal. Les signaux de tests ne contiennent en effet aucun exemple de tachycardie supra-ventriculaire.

4.3.3.2 Monitoring en milieu bruité

Nous avons décidé de faire plusieurs expériences en bruitant successivement les différentes sources de données pour chacun des cas suivants : systèmes de monitoring monosource et multisource non pilotés et système de monitoring multisource avec pilotage complet. Pour obtenir des ECG bruités réalistes, nous avons ajouté aux signaux de test des bruits cliniques réels fournis par une base de données[1]. Pour le signal de pression, le bruit a été modélisé par une perte totale de ligne. Notons, que ce cas extrême n'est pas rare en milieu clinique. Un exemple de lignes bruitées est donné Figure 4.10. Sur le premier signal, les artefacts rendent l'onde P très difficile à détecter sans erreur. Le second signal est propre. Le dernier signal montre une perte de ligne sur la source ABP.

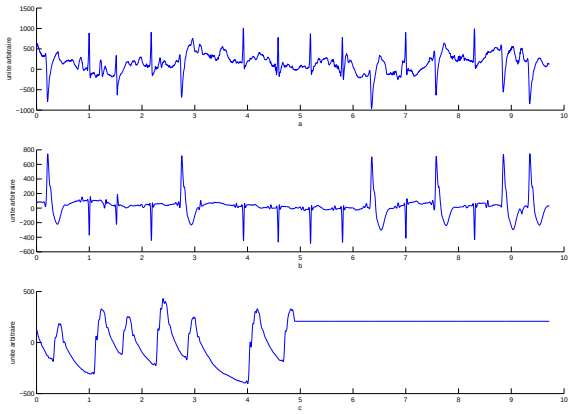


FIG. 4.10 – Exemple de bruits sur les sources de données

Reconnaissance d'arythmies sans pilotage Les résultats de l'apprentissage sur des sources uniques bruitées sans utiliser le module de pilotage sont présentés dans les tables 4.6, 4.7, 4.8. La table 4.9 présente des résultats obtenus sans pilotage en utilisant des règles multisources biaisées apprises à partir de données de la voie V (sans prendre en compte, la forme du QRS) et de pression (sans prendre en compte les informations sur la systole).

[1] http://www.physionet.org/physiobank/database/nstdb/

	rs	esv	bige	doublet	tv	fa	Total
rs	0	0	0	0	0	0	0
esv	0	0	0	0	0	0	0
bige	75	16	55	0	0	0	146
doublet	46	2	3	26	0	0	77
tv	48	5	15	26	270	25	389
fa	123	4	29	5	20	412	593
NR	564	20	87	26	13	64	774
Total	856	47	189	83	303	501	1979
Sens	0	0	0.29	0.31	0.89	0.82	
Pred+	1	1	0.38	0.34	0.69	0.69	

TAB. 4.6 – Matrice de confusion à partir de l'ECG bruité seul

La table 4.6 montre des résultats de précision correcte pour les arythmies tachycardies ventriculaires (*tv*) et fibrillation auriculaire (*fa*) sur un ECG bruité mais la prédictivité reste faible (0.69). La reconnaissance des bigéminismes (*bige*) et des doublets ventriculaires (*doublet*) est meilleure que sur les autres voies. Par contre, aucun rythme normal et aucune extrasystole ventriculaire n'a été reconnu pour la voie I de l'ECG. Ceci s'explique par le fait que les chroniques décrivant ces arythmies occupent une fenêtre temporelle beaucoup plus large que celle des chroniques décrivant les autres arythmies. Ces arythmies sont caractérisées par un seul ou aucun (dans le cas *rs*) battement anormal et sont beaucoup plus difficiles à discriminer que les arythmies caractérisées par plusieurs battements anormaux. Il y a donc très peu de chance que la chronique soit reconnue dans sa totalité (et donc que l'arythmie soit détectée) avant que le signal soit à nouveau bruité.

	rs	esv	bige	doublet	tv	fa	Total
rs	356	0	14	1	0	0	371
esv	0	24	12	3	2	0	41
bige	5	1	16	0	0	0	22
doublet	15	2	11	13	0	0	41
tv	36	4	3	11	296	442	792
fa	6	3	1	0	3	59	72
NR	435	13	129	55	2	0	634
Total	853	47	186	83	303	501	1973
Sens	0.42	0.51	0.09	0.16	0.98	0.12	
Pred+	0.96	0.59	0.73	0.32	0.37	0.82	

TAB. 4.7 – Matrice de confusion pour la voie V bruitée seule

La matrice de confusion présentée en table 4.7 montre que les règles apprises sur les

données de la voie V (cf. Section C.1.3.2) présentent des performances moyennes pour les arythmies *rs* et *esv* mais une bonne prédictivité positive pour le *rs* en particulier (0.96). Cette source est moins sensible au bruit que la voie I puisqu'elle ne prend pas en compte les mauvaises détections de l'onde P et les mauvaises classifications de la forme du QRS. Ce constat explique la meilleure sensibilité obtenue pour les arythmies ayant des fenêtres temporelles larges comme le rythme sinusal ou les extrasystoles. Les résultats sont, en revanche, beaucoup moins bons pour les arythmies *bige*, *doublet* et *fa*. Pour les arythmies *bige* et *fa*, ces résultats s'expliquent car la reconnaissance de ces arythmies repose, pour la voie V, uniquement sur les intervalles de temps entre les QRS. Nous avons constaté que ces intervalles varient beaucoup d'un patient à l'autre et ne concordent pas forcément avec les intervalles utilisés pendant l'apprentissage même si ces intervalles proviennent directement des ouvrages médicaux. Les signaux de test utilisés proviennent de patients dont le rythme de base est rapide, les intervalles RR sont donc majoritairement courts. La relation *rr1(R0,R1,long)* est donc difficile à trouver dans ces données. Pour le *doublet*, les règles dénotent clairement un sur-apprentissage puisque chaque règle ne couvre que deux exemples de *doublet*. Pour améliorer ces performances et rendre ces règles moins spécifiques, il faudrait ajouter de nouveaux exemples lors de l'apprentissage.

Les résultats sont, en revanche, très bons pour la *tv* mais la prédictivité positive a nettement diminué par rapport aux performances obtenues avec la voie I de l'ECG. Ceci s'explique par le nombre importants de confusions avec la *fa* car la première règle décrivant cette arythmie est très proche de la règle décrivant la *tv*.

	rs	esv	bige	doublet	tv	fa	Total
rs	436	0	14	5	0	4	459
esv	72	19	53	5	0	1	150
bige	0	0	0	0	0	0	0
doublet	7	0	1	6	0	18	32
tv	14	1	1	8	178	73	275
fa	2	0	5	1	0	115	123
NR	326	27	115	58	125	291	942
Total	857	47	189	83	303	502	1981
Sens	0.51	0.40	0	0.07	0.59	0.23	
Pred+	0.95	0.13	1	0.19	0.65	0.93	

TAB. 4.8 – Matrice de confusion pour la mesure de pression bruitée (avec la systole seule)

La matrice de confusion correspondant à la pression avec systole seule (cf. Table 4.8) donne les meilleurs résultats des trois sources testées seules pour le rythme sinusal. Les résultats pour *bige, doublet* et *fa* sont en revanche très faibles. Ceci est dû au grand nombre de contraintes temporelles entre les évènements caractéristiques de ces arythmies. Les seuils choisis pour passer de la représentation numérique des données à

la représentation symbolique semblent trop spécifiques aux données d'apprentissage.

	rs	esv	bige	doublet	tv	fa	Total
rs	427	0	2	1	1	6	437
esv	99	24	29	27	8	25	212
bige	22	0	38	12	0	0	72
doublet	0	0	0	0	0	6	6
tv	8	0	2	9	188	81	288
fa	8	0	0	1	0	215	224
NR	292	23	117	33	106	169	740
Total	856	47	188	83	303	502	1979
Sens	0.50	0.51	0.20	0	0.62	0.43	
Pred+	0.98	0.11	0.53	0	0.65	0.96	

TAB. 4.9 – Matrice de confusion pour la fusion de la voie V et de la pression

La table 4.9 correspond à la matrice de confusion pour la fusion de deux sources : la voie V de l'électrocardiogramme et la voie de pression. Les résultats sont globalement meilleurs que pour les deux sources prises séparément. La reconnaissance sur les sources V et ABP fusionnées donne un meilleur résultat pour la fibrillation auriculaire (0.43 pour la *fa*) que pour les sources prises séparément. De même que pour l'ECG seul, les résultats sont bons pour la tachycardie ventriculaire. Ils restent en revanche médiocres pour les autres arythmies. En effet, les fenêtres temporelles associées aux chroniques décrivant un rythme sinusal (*rs*) sont encore plus larges que celles des chroniques *rs* pour l'ECG seul. En outre, les chroniques correspondant à la fusion ECG/ABP sont les plus sensibles au bruit puisque la voie I et la voie ABP doivent être *propres* en même temps pour que les chroniques donnent leur meilleur résultat. Ce cas de figure est beaucoup plus rare que dans les cas de l'utilisation d'une source seule. Les mauvais résultats obtenus pour le *doublet* sont dûs aux deux règles multisources apprises. La première semble en effet trop spécifique aux exemples d'apprentissage avec la présence de deux QRS consécutifs sans évènements de pression entre les deux QRS. La seconde, qui n'utilise que des évènements de pression, offre de très mauvais résultats sur des données bruitées (cf. résultats pour le *doublet* Table 4.8).

Utilisation du module de Pilotage complet Dans cette expérience, le module de pilotage multisource complet est utilisé. Les résultats sont donnés Table 4.10. En plus de sélectionner la ou les sources les plus intéressantes, le module de pilotage va également sélectionner la chronique la plus adaptée au niveau de détail des évènements fournis par le module d'abstraction temporelle.

Les résultats du pilotage sont en moyenne meilleurs que sur chacune des autres sources prises séparément. La sensibilité est meilleure pour l'*esv*, le *doublet* et le *rs* en utilisant le pilotage que pour toutes les autres expériences. Cette sensibilité est moins bonne que pour la voie I seule pour *bige* et *fa* mais la prédictivité est égale ou meilleure.

	rs	esv	bige	doublet	tv	fa	Total
rs	498	0	30	2	0	0	530
esv	25	25	25	3	0	1	79
bige	24	5	23	4	0	0	56
doublet	28	1	8	33	2	5	77
tv	14	0	1	9	278	289	591
fa	26	0	23	11	3	142	205
NR	241	16	79	21	20	65	442
Total	856	47	189	83	303	502	1980
Sens	0.58	0.53	0.12	0.40	0.92	0.28	
Pred+	0.94	0.32	0.41	0.43	0.47	0.69	

TAB. 4.10 – Matrice de confusion la reconnaissance utilisant toutes les sources avec pilotage

La sensibilité pour la *fa* et le *bige* pâtit des très mauvais résultats de la pression et de la voie V seules. En effet, le pilote choisit la ou les sources les moins bruitées même si celles-ci ne sont pas les plus adaptées pour reconnaître un type d'arythmie particulier. Cependant, une liste des chroniques en cours de reconnaissances à un instant donné pourrait servir dans de futures expériences de pilotage à guider le choix de la source quand toutes les sources ne sont pas bruitées. La prédictivité pour la *tv* a également baissé par rapport à la voie I seule mais les confusions sont principalement faites avec la *fa* (289 arythmies détectées comme *tv* qui sont en faites des *fa*). Ces confusions ne sont pas les plus grave puisqu'elles ont tout de même une signification médicale (au contraire de confusion avec le rythme sinusal).

4.4 Conclusion

Nous avons formalisé dans ce chapitre l'apprentissage multisource par programmation logique inductive. Puis, nous avons présenté une méthode permettant de mettre en œuvre l'apprentissage multisource malgré les problèmes de dimensionnalité introduits. L'idée est de s'appuyer sur des apprentissages effectués préalablement sur chacune des sources indépendamment et de générer automatiquement un biais déclaratif réduisant ainsi l'espace de recherche multisource.

Cette méthode "biaisée" d'apprentissage multisource est évaluée dans le cadre de l'apprentissage de règles caractérisant des arythmies cardiaques à partir d'électrocardiogramme et de mesures de pression artérielle. Les résultats obtenus sont comparés à ceux obtenus avec un algorithme naïf d'apprentissage multisource ainsi qu'aux résultats monosources présentés dans le chapitre 3 et en Annexe C.

Nous avons montré à travers de nombreuses expériences que la méthode d'apprentissage multisource biaisée donne des résultats toujours aussi bons et souvent meilleurs que les apprentissages monosources du point de vue de l'apprentissage et de la recon-

naissance. Ces résultats, déjà vérifiés lorsque les règles monosources sont très bonnes, sont encore plus nets lorsque les données provenant des différentes sources sont complémentaires. De plus, les résultats de la méthode biaisée sont comparables à ceux obtenus par apprentissage multisource naïf du point de vue de l'apprentissage et de la reconnaissance. Cependant, la méthode biaisée permet d'obtenir des règles dans des temps en moyenne treize fois inférieurs aux temps de la méthode naïve. En outre, l'apprentissage multisource biaisé permet également d'apprendre des règles mixtes, tirant parti de la complémentarité des deux sources et ainsi répondre aux objectifs présentés en introduction de ce mémoire.

Des expériences ont également été menées dans le cadre du prototype CALICOT en milieu bruité et montre que l'utilisation de plusieurs sources de données couplée à l'ajout d'un module de pilotage dans CALICOT permet d'augmenter la fiabilité de la reconnaissance d'arythmies cardiaques en ligne.

Conclusion et perspectives

L'objectif de cette thèse était d'évaluer les possibilités d'acquisition automatique de connaissances à partir de données provenant de nouveaux capteurs et d'intégrer ces nouvelles connaissances pour le diagnostic des arythmies cardiaques à partir de données multisources.

Nous nous sommes intéressés à l'acquisition automatique de connaissances à partir de données provenant de différentes voies de l'ECG et d'une voie de pression artérielle. Cette dernière, beaucoup moins utilisée par les médecins pour la reconnaissance d'arythmies cardiaques nous a permis d'exhiber de nouvelles connaissances. Les connaissances acquises ont été intégrées dans le prototype de système de monitoring cardiaque Calicot développé dans de précédents travaux [Carrault *et al.*, 2003].

Une première contribution de ce travail [Fromont *et al.*, 2003] a été de fournir une méthodologie de comparaison des systèmes de monitoring cardiaques Kardio et Calicot souvent comparés sans qu'il y ait eu d'étude rigoureuse permettant de les différencier. Les règles de diagnostic de Kardio ont été générées à partir de la simulation d'un modèle cardiaque suivie d'une phase d'apprentissage. Ces règles causales permettent d'associer des troubles du rythme à des descriptions de l'ECG. Elles ne peuvent pas être utilisées telles quelles pour le diagnostic en ligne. Une première étape a donc été de transformer ces règles pour associer une arythmie donnée à une description de l'ECG. Les règles de Calicot ont été apprises à partir de descriptions symboliques de signaux réels. Le langage de description des arythmies utilisé dans ces règles est donc adapté aux technologies de traitement du signal actuelles qui permettent de fournir les représentations symboliques des signaux. Pour pouvoir tester les règles de Kardio dans un système de monitoring sur des signaux réels et comparer ces performances à celles des règles de Calicot, nous avons dû adapter le langage de Kardio. Nous avons ainsi montré que les règles obtenues pour les deux systèmes offrent des résultats comparables pour la reconnaissance en ligne d'arythmie cardiaques.

Nous avons utilisé la programmation logique inductive (PLI) comme technique d'acquisition automatique de connaissances sur de nouvelles sources de données. Ce choix fait suite à de précédents travaux [Quiniou *et al.*, 2001] utilisant la PLI et ayant permis d'obtenir des résultats très satisfaisants pour la caractérisation d'arythmies cardiaques à partir d'une seule voie de l'ECG et pour des arythmies différentes de celles considérées dans notre étude. Nous avons testé sur nos données deux systèmes de PLI ayant une

sémantique différente : ALEPH [Srinivasan, 2003] et ICL [De Raedt et Van Laer, 1995]. Le premier fonctionne à partir d'exemples décrits sous forme de faits PROLOG et permet d'apprendre une théorie clausale. Le second fonctionne à partir d'interprétations partielles de la théorie cible et permet d'apprendre un ensemble de règles de classification. En outre, ICL dispose d'un langage de biais puissant permettant de limiter l'espace de recherche des solutions alors qu'ALEPH construit la clause la plus spécifique de l'espace de recherche (*bottom clause*) à partir d'exemples positifs et utilise ensuite un ensemble de biais sémantiques lors de la recherche. L'acquisition des résultats pour les deux systèmes a nécessité la spécification de biais d'apprentissage élaborés et une sélection pertinente des attributs décrivant les évènements essentiels se produisant sur les voies. Les solutions proposées par ALEPH pour limiter l'espace de recherche et trouver des solutions sémantiquement intéressantes se sont avérées moins efficaces que celles d'ICL pour notre problème. Dans la suite de nos travaux, nous avons donc choisi d'utiliser ce second système.

Les résultats fournis par ICL ont confirmé le bien fondé de l'utilisation de la PLI pour apprendre des règles discriminantes, non seulement sur différentes voies de l'ECG mais sur des données moins connues comme la pression artérielle. Le système ICL offre en effet des résultats de précision en apprentissage et en test presque parfaits pour l'ECG (sur la voie I et, dans une moindre mesure, sur la voie V) et des résultats très encourageants sur la voie de pression.

On peut retirer deux types de bénéfices de l'utilisation de plusieurs sources de données dans un système de monitoring cardiaque : soit les sources sont redondantes et l'utilisation opportuniste des différentes sources permet de pallier les problèmes de pertes de signal ou de sources trop bruitées; soit les sources sont complémentaires et l'utilisation conjointe de celles-ci permet d'augmenter les performances en détection du système de monitoring. Pour le premier cas, il faut acquérir des règles monosources ayant la meilleure précision possible. Pour le second cas, il faut acquérir des règles multisources mettant en relation les évènements importants se produisant sur chacune des sources. L'acquisition de telles règles par PLI fait l'objet de la dernière partie de nos travaux.

Nous avons dans un premier temps formalisé le problème de l'apprentissage multisource par PLI. Puis, nous avons proposé une technique qualifiée de naïve, consistant à agréger les données provenant des différentes sources et à utiliser la technique de PLI avec un biais peu restrictif compte tenu du manque de connaissances sur les relations pertinentes pouvant être apprises. L'agrégation des différentes sources et le nombre important de relations pouvant exister entre les évènements se produisant sur ces sources augmentent considérablement la taille de l'espace de recherche lors de l'apprentissage, ce qui pose des problèmes sérieux aux systèmes de PLI. Nous avons donc développé une méthode décrite brièvement dans [Fromont *et al.*, 2004, Fromont *et al.*, 2005a] pour biaiser efficacement et automatiquement l'espace de recherche multisource. Cette technique s'appuie sur des règles acquises par apprentissage monosource obtenues préalablement. Elle garantit l'obtention de règles ayant une précision en apprentissage au moins aussi bonne que les règles monosources sur lesquelles elle s'appuie.

Les premiers résultats obtenus sur nos exemples montrent que la voie I de l'ECG

(notée ECG_I) et la pression artérielle (notée ABP) sont redondantes pour la caractérisation des arythmies cardiaques. En effet, aucune règle mettant en relation des évènements des différentes voies n'a été apprise et pourtant, les résultats sur des données non bruitées sont presque parfaits. Des résultats sur des sources plus complémentaires telles que les sources ECG_V et ABP mais aussi des résultats combinant la source ECG_V et une source ne fournissant que des informations sur l'onde P (comme ça pourrait être le cas avec une sonde œsophagienne) montrent que les apprentissages multisources utilisant la méthode proposée offrent effectivement des performances toujours aussi bonnes et souvent meilleures que les apprentissages monosources du point de vue de l'apprentissage et de la reconnaissance sur une base de données non bruitée. En outre, la méthode proposée offre des meilleurs résultats que des apprentissages multisources naïfs obtenus avec un biais manuel et cela, dans des temps en moyenne 13 fois inférieurs. Le chapitre 4 a également donné une description détaillée des étapes de construction automatique du biais multisource ainsi qu'une méthode permettant de valider automatiquement les apprentissages multisources biaisés. L'ensemble de ces résultats est donné dans [Fromont *et al.*, 2005b, Fromont *et al.*, 2006].

La méthode proposée concerne deux sources de données. Elle peut être théoriquement étendue à un plus grand nombre de sources mais la complexité du calcul lors de la génération des *bottom clauses* limitant l'espace de recherche multisource risque de poser des problèmes de passage à l'échelle. Pour résoudre ce problème, on peut imaginer une stratégie utilisant des apprentissages multisources partiels sur seulement deux sources de données pour apprendre des règles multisources partielles permettant de réduire à nouveau l'espace de recherche multisource. En réitérant le processus et en utilisant les sources deux à deux, on peut ainsi couvrir l'espace multisource complet. En outre, pour réduire le nombre de *bottom clauses* créées, nous fournissons à l'algorithme de génération de biais un ensemble de contraintes permettant d'éliminer les séquences d'évènements physiologiquement impossibles. L'acquisition de ces contraintes pourrait être réalisée automatiquement lors d'un pré-traitement des données consistant en une recherche des motifs fréquents de taille limitée par le nombre d'évènements pouvant se produire sur les sources.

Les règles monosources et multisources apprises ont été intégrées au prototype Calicot et leur pouvoir de détection a été testé sur des signaux réels bruités artificiellement. Les expériences montrent que l'utilisation de plusieurs sources de données couplée à l'ajout d'un module de pilotage dans Calicot permet d'augmenter la fiabilité de la reconnaissance d'arythmies cardiaques en ligne (cf. [Fromont et Portet, 2005]) sur ce type de données.

L'utilisation des règles apprises sur des données bruitées pose une problématique différente de celle exposée au début de cette conclusion puisqu'il n'est plus ici seulement question d'apprendre des règles discriminantes ayant une bonne précision lors des apprentissages et lors des tests mais également d'obtenir des règles robustes à la présence importante de bruits sur les données de tests. Les expériences menées avec Calicot ont ainsi permis de soulever un certain nombre de problèmes constituant des perspectives de ce travail.

Les performances du système de monitoring sur des sources bruitées sont moins bonnes que les mesures de précision en apprentissage et en test obtenues lors des validation croisées sur la base d'apprentissage non bruitée. Ces différences sont particulièrement marquées pour le rythme sinusal, l'extrasystole ventriculaire et le doublet ventriculaire à partir de l'ECG et pour le doublet ventriculaire et la fibrillation auriculaire à partir du signal de pression.

Concernant la source ECG, le problème provient principalement des larges fenêtres temporelles couvertes par les règles apprises qui rendent ces règles moins robustes à la présence de bruit sur le signal et aux erreurs de détection des algorithmes de traitement du signal. Les exemples d'apprentissage pour ces arythmies contiennent en effet un grand nombre de cycles cardiaques normaux ce qui rend difficile la discrimination de ces arythmies dont la manifestation est ponctuelle, face au rythme sinusal. Pour obtenir des règles compactes plus robustes aux bruits, il faudrait réduire la taille des exemples d'apprentissage de manière à ce qu'ils ne contiennent que la manifestation de l'arythmie. Dans ce cas, deux extrasystoles ventriculaires seraient tout de même détectées lors d'un doublet ventriculaire. Pour discriminer de telles arythmies, l'utilisation conjointe du signal de pression pourrait être essentielle. Une autre solution serait d'utiliser pour l'acquisition des règles, un langage de description des évènements dont les attributs seraient très robustes aux bruits comme c'est le cas, par exemple, pour la voie V lorsque l'information sur la forme du QRS n'est pas utilisée.

Le problème des performances en détection obtenues sur le signal de pression est, par contre, lié au choix des seuils choisis pour passer de la représentation numérique des données à la représentation symbolique. En effet, l'allure générale du signal de pression est spécifique à chaque patient et les attributs utilisés pour l'apprentissage présentent une trop grande variabilité inter-patient. Des recherches sont en cours pour déterminer des attributs offrant une moins grande variabilité.

D'un point de vue général, il faudrait augmenter la base d'exemples, d'une part pour valider plus amplement la méthode multisource et d'autre part, pour mettre en évidence la complémentarité de l'ECG et de la pression quand il s'agit de déterminer la gravité d'une arythmie. En effet, les cardiologues se servent en général du signal de pression pour déterminer la sévérité ou la tolérance d'une arythmie préalablement détectée sur l'ECG. Une extrasystole très prématurée est particulièrement intéressante à détecter car souvent annonciatrice de troubles plus graves telle que la tachycardie ventriculaire. Sur l'ECG, une extrasystole prématurée a la même signature morphologique qu'une extrasystole non prématurée tandis que sur le signal de pression, une extrasystole prématurée se traduit par un pic systolique très peu visible suivit d'une diastole de très faible amplitude. Pour pouvoir subdiviser les classes d'arythmies suivant leur sévérité, un plus grand nombre d'exemples est nécessaire. En l'absence de grande base d'exemples multisources étiquetées, des modèle cardiaques tels que CARMEN [Hernández *et al.*, 2000] peuvent être utilisés. Cependant, les signaux générés par de tels modèles sont parfois assez éloignés des signaux que l'on trouve réellement en milieu hospitalier et les systèmes requièrent une adaptation pour générer conjointement le signal ECG et son image sur le plan hémodynamique.

Enfin, nous avons utilisé des données déjà connues des cardiologues. Ceux-ci étaient donc capables d'expliquer quels étaient, de leur point de vue, les évènements intéressants sur ces données et notre tâche a consisté à choisir la représentation symbolique des données la plus adaptée à l'apprentissage de règles discriminantes. Une des perspectives les plus intéressantes de ce travail est d'être capable d'utiliser des données multisources provenant de nouveaux capteurs dont les signatures (arythmiques) ne sont pas nécessairement connues du spécialiste. L'approche multisource proposée ici pourra être exploitée. Cependant, dans le cas de données inconnues, des techniques d'invention de prédicats [Kramer, 1995] et de sélection d'attributs pourraient être envisagées pour exhiber les évènements intéressants et les attributs discriminants se produisant sur les différentes sources de données.

Annexe A

Signatures des arythmies

Cette annexe présente les signature électriques et hémodynamiques typiques des arythmies qui sont abordées dans cette étude : le *mobitz de type II*, le *bloc de branche gauche* (lbbb) et le rythme normal ou *rythme sinusal*. Puis, l'*extrasystole ventriculaire*, le *doublet ventriculaire*, le *bigéminisme*, la *tachycardie ventriculaire*, la *tachycardie supra-ventriculaire* et la *fibrillation auriculaire*. Pour les deux premières arythmies citées, seules les signatures électriques sont fournies puisque nous n'avons pas à notre disposition d'exemples de signaux hémodynamiques leur correspondant.

La cause de chacune des arythmies est détaillés en section 1.1.3. La description (signature) des arythmies provient de [Stein, 1999].

A.1 Tracés correspondant aux arythmies étudiées dans le chapitre 2

Ces tracés correspondent à des arythmies étudiées pendant le projet PISE à partir de la base de données MIT-BIH [Moody, 1997a] sur la voie II de l'électrocardiogramme uniquement et utilisées dans le chapitre 2 pour la comparaison entre les systèmes de monitoring cardiaques KARDIO et CALICOT.

A.1.1 Le bloc de branche gauche : lbbb

Une arythmie de type bloc de branche gauche se différencie du rythme sinusal par une forme anormale des complexes QRS. La largeur du QRS dépasse 120 millisecondes (contre 100 ms pour le rythme normal). Sur la voie II, le complexe QRS est en général entièrement positif et l'onde R est en général crochetée ou avec un plateau. La repolarisation ventriculaire est inversée avec une onde T négative profonde et asymétrique.

A.1.2 Le mobitz de type II : mobitz

Lors d'un bloc A-V (cf. section 1.1.3) de type mobitz, la durée de l'intervalle PR est plus longue que la normale. Toutes les ondes P ne sont pas suivies d'un complexe QRS mais tous les QRS sont précédés d'une onde P. Le rythme reste régulier si la

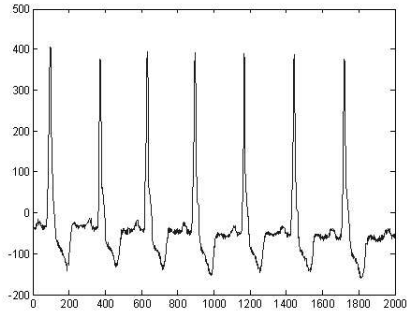


FIG. A.1 – Bloc de branche gauche : voie II

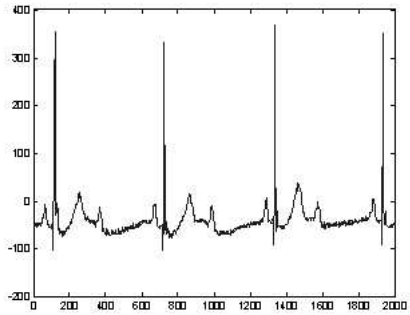


FIG. A.2 – Mobitz de type II : voie II

relation entre l'activité auriculaire et ventriculaire est stable. Dans le cas contraire, le rythme est irrégulier.

A.2 Tracés des arythmies utilisés en apprentissage multi-source

Les exemples suivants sont des tracés électrocardiographiques et hémodynamiques correspondant aux données extraites de la base MIMIC [Moody et Mark, 1996]. Les traits verticaux présents sur les tracés indiquent le début et la fin du QRS pour les signaux ECG et les instants d'apparition de la diastole et de la systole pour la pression.

A.2.1 Le rythme sinusal ou rythme normal : rs

Un exemple de rythme sinusal est donné Figure A.3.

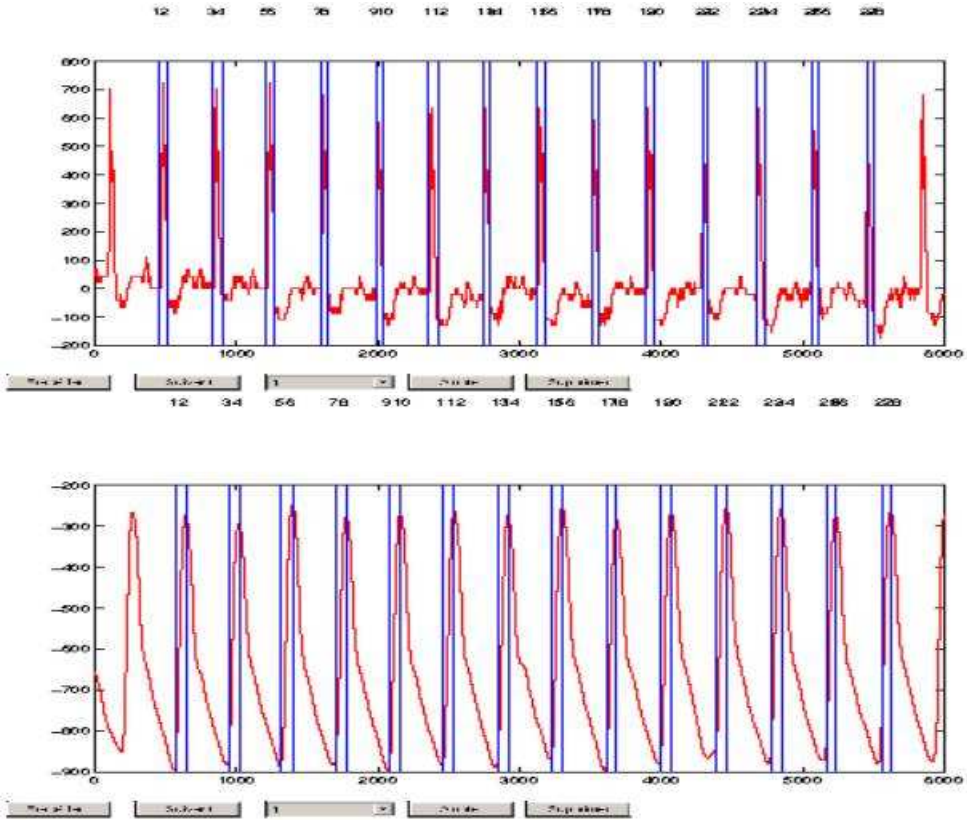


FIG. A.3 – Rythme sinusal : voie II et voie de pression

A.2.2 L'extrasystole ventriculaire : esv

Les extrasystoles sous forme de PVC (les contractions ventriculaires prématurées) ne sont pas précédées d'une onde P. La forme du QRS est souvent inhabituelle. La régularité du rythme est contrariée par la présence des PVC, celui-ci pouvant devenir totalement irrégulier en présence d'un grand nombre de PVC.

Dans nos données, seulement 3 exemples sont des PVC. Ces exemples ont une signature hémodynamique très intéressante puisque l'apparition très prématurée d'un nouveau QRS après le QRS normal ne laisse pas suffisamment de temps au ventricule pour se remplir à nouveau de sang. Dans ce cas la systole ventriculaire peut être manquante ou trop petite pour être détectée automatiquement.

Figure A.4, l'extrasystole n'est pas suffisamment prématurée pour empêcher totalement le remplissage des ventricules. On distingue donc une systole de plus faible amplitude que la normale.

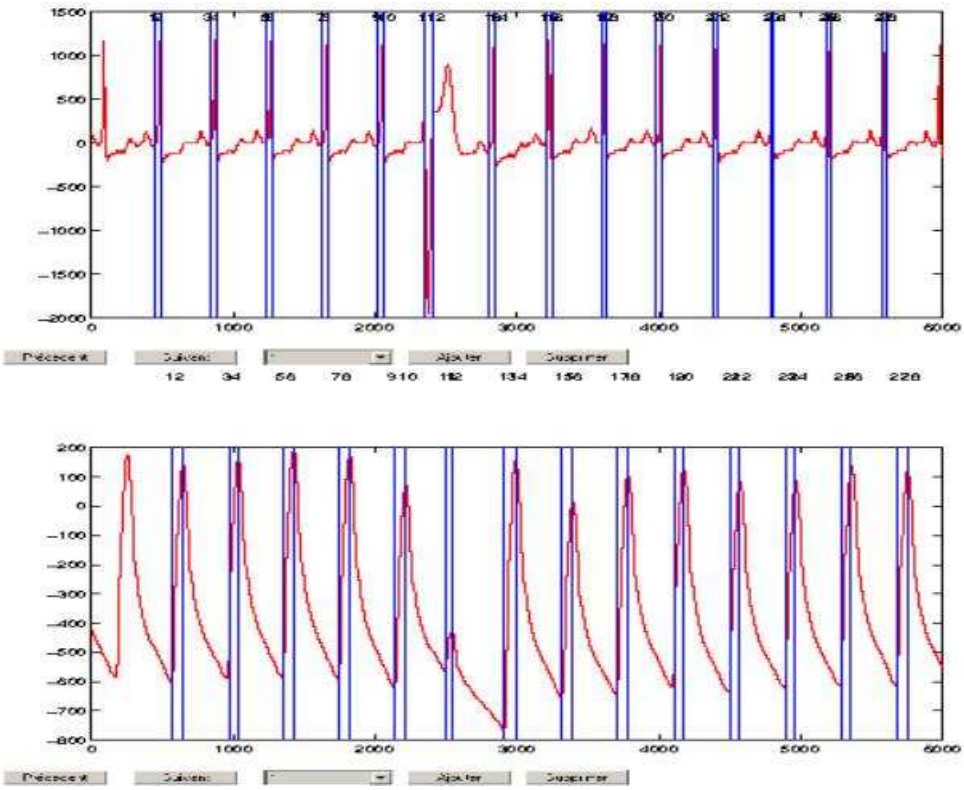


FIG. A.4 – Extra systole ventriculaire : voie II et voie de pression

A.2.3 Le doublet ventriculaire : doublet

Les doublets ventriculaires (cf. A.5) correspondent à des extrasystoles se produisant par groupe de deux, de manière assez rapprochée. Lorsque trois extrasystoles se produisent consécutivement, on parle normalement de triplets ventriculaires. Lorsque plus de trois extrasystoles se produisent consécutivement, on peut diagnostiquer une tachycardie ventriculaire.

D'un point de vue hémodynamique, le remplissage des ventricules est altéré (comme pour l'extrasystole seule) dans le cas d'un doublet ventriculaire produisant des systoles de faibles amplitudes ou manquantes.

A.2.4 Le bigéminisme : bige

Le bigéminisme correspond à un rythme cardiaque dans lequel le cœur bat par groupe de deux battements. Dans nos exemples, ces groupes sont toujours constitués d'un battement normal suivi d'une extrasystole (cf. Figure A.6).

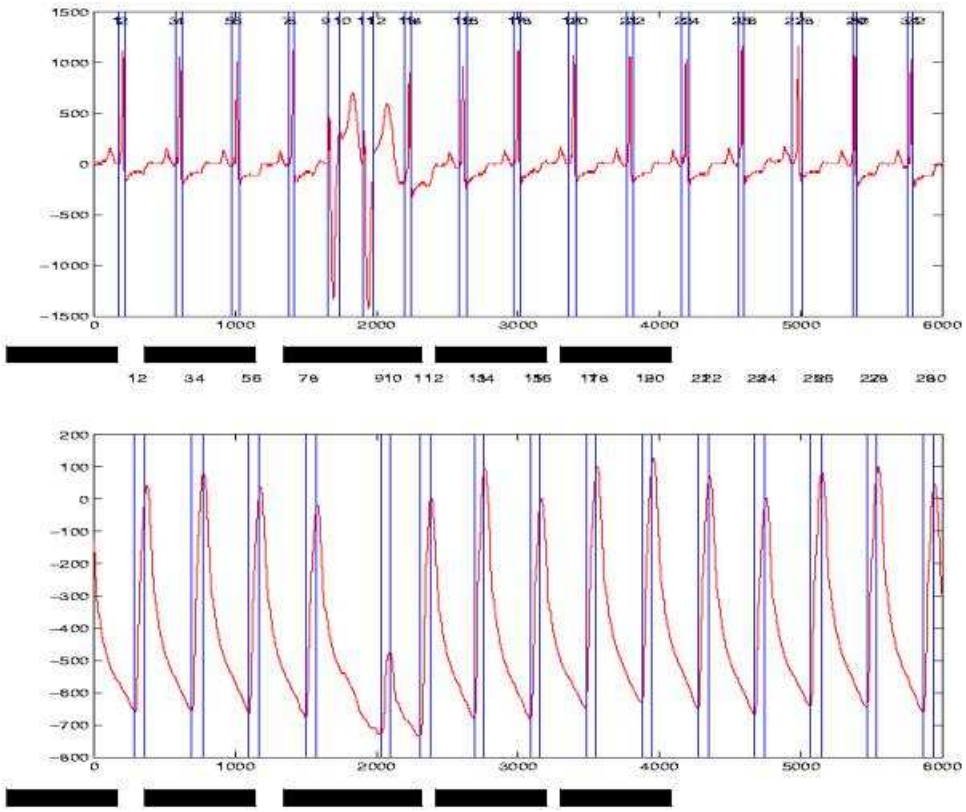


FIG. A.5 – Doublet ventriculaire : voie II et voie de pression

A.2.5 L'accès de tachycardie ventriculaire : tv

Lorsque le cœur est en état de tachycardie ventriculaire, sa fréquence de battement varie entre 150 et 250 battements par minute (comparée à 60 - 100 pour un rythme normal). L'intervalle RR prend donc des valeurs entre 240 et 400 ms (comparées à 600-1000 pour le rythme normal).

Le complexe QRS est plus large que pour le rythme normal et a une forme inhabituelle.

Le rythme dominant, même s'il reste régulier, est fourni par un foyer ectopique situé dans un des ventricules. L'onde P peut donc être impossible à distinguer même si dans certain cas, l'activité électrique auriculaire dissociée de l'activité électrique ventriculaire n'est pas affectée. Dans tous les exemples extraits de la base de données sur laquelle nous travaillons, l'onde P est invisible.

On peut diagnostiquer un accès de tachycardie ventriculaire en présence de 4 battements consécutifs (ou plus) ayant les propriétés ci-dessus. Dans nos exemples cependant, en l'absence de triplets ventriculaires, un accès de tachycardie ventriculaire sera diagnostiqué dès que 3 extrasystoles rapides se produiront successivement (cf. Figure A.7).

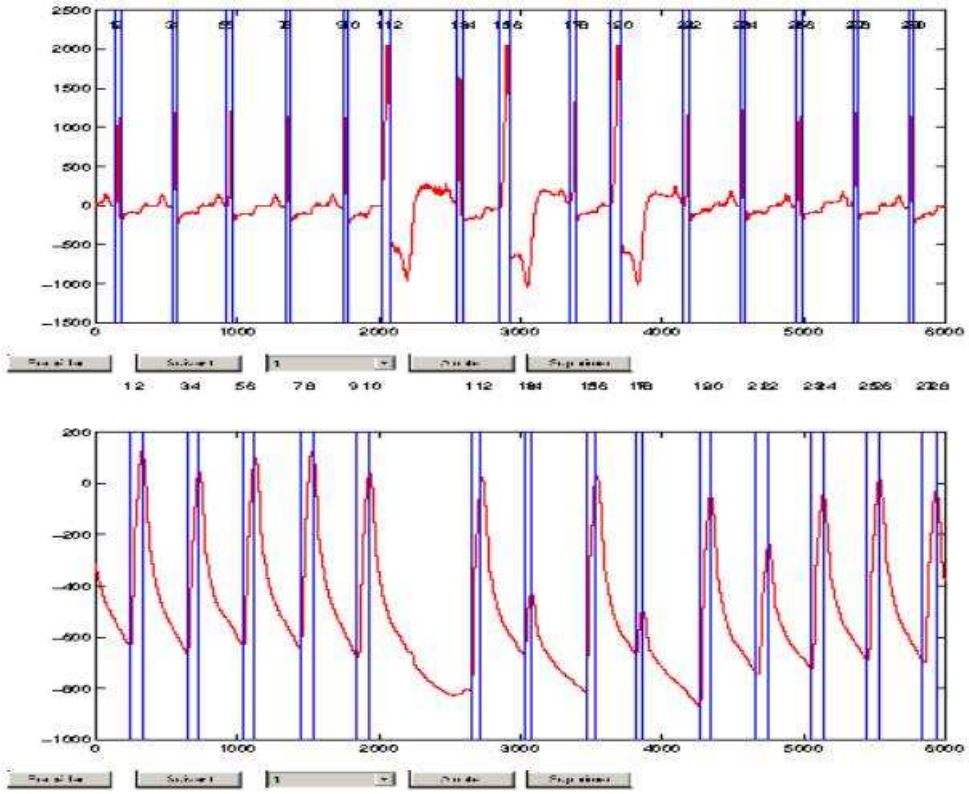


FIG. A.6 – Bigéminisme : voie II et voie de pression

A.2.6 La tachycardie supra-ventriculaire : tsv

Lors d'une tachycardie supra-ventriculaire, l'impulsion électrique provient d'un foyer situé au dessus des ventricules souvent à la jonction AV (tachycardie jonctionnelle) ou au niveau du nœud sinusal (tachycardie sinusale).

Pour une tachycardie sinusale, le rythme est conforme à un rythme normal (présence d'onde P et forme des ondes normale) mais la fréquence cardiaque est plus élevée (100 et 160 battements par minute pour une tachycardie sinusale). L'intervalle RR se situe entre 375 et 600 ms.

Pour une tachycardie jonctionnelle, l'onde P est souvent invisible car noyée dans le complexe QRS. Lorsqu'elle est visible, l'onde P est inversée sur la voie II. La fréquence cardiaque se situe entre 100 et 170. L'intervalle PR est inférieur à 120 ms. Le rythme reste régulier.

Les exemples de tachycardies supra-ventriculaires utilisés dans notre étude sont tous des tachycardies sinusales (cf Figure A.8).

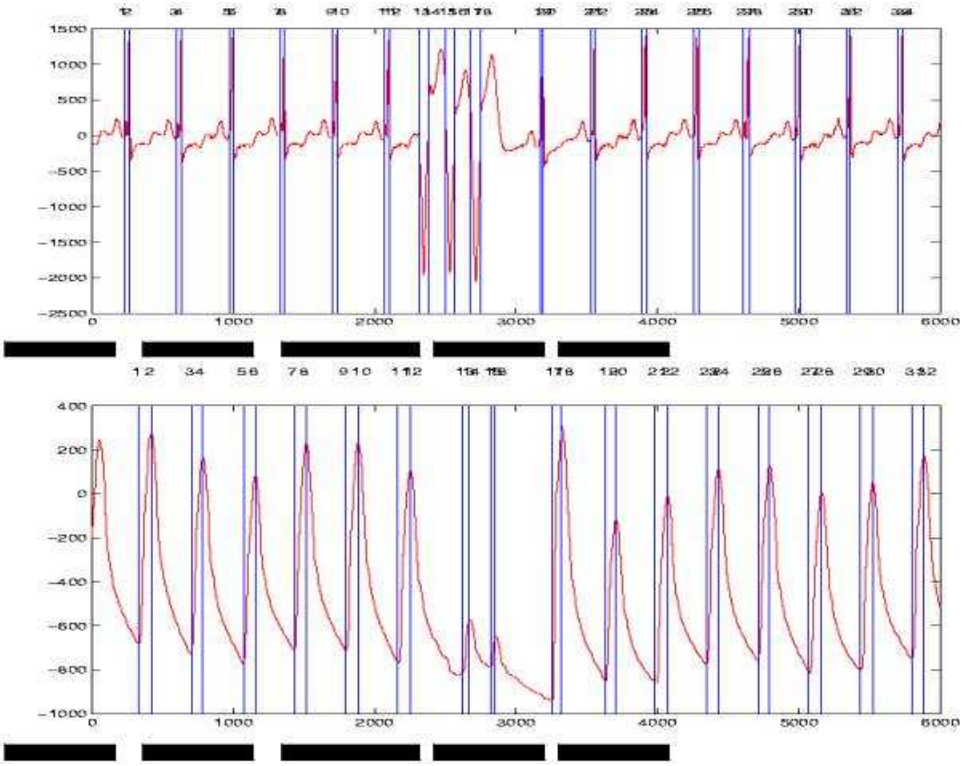


FIG. A.7 – Tachycardie ventriculaire : voie II et voie de pression

A.2.7 La fibrillation auriculaire : fa

Lors d'une fibrillation auriculaire (cf. Figure A.9), il n'y pas d'onde P identifiable sur l'ECG. On peut par contre distinguer des ondes fibrillatoires : des mouvements irréguliers de la ligne de base. Le complexe QRS a une forme normale, sa largeur est soit normale soit plus étroite que pour un rythme normal.

Le rythme est irrégulièrement irrégulier c'est à dire irrégulier sans motif spécifique. La fréquence du rythme auriculaire se situe entre 350 et 600 battements par minutes mais la plupart des impulsions électriques sont bloquées au nœud AV. La réponse ventriculaire reste en générale entre 60 et 100 battements par minutes (comme pour un rythme normal) mais des variations (plus rapides ou moins rapides) ne sont pas rares.

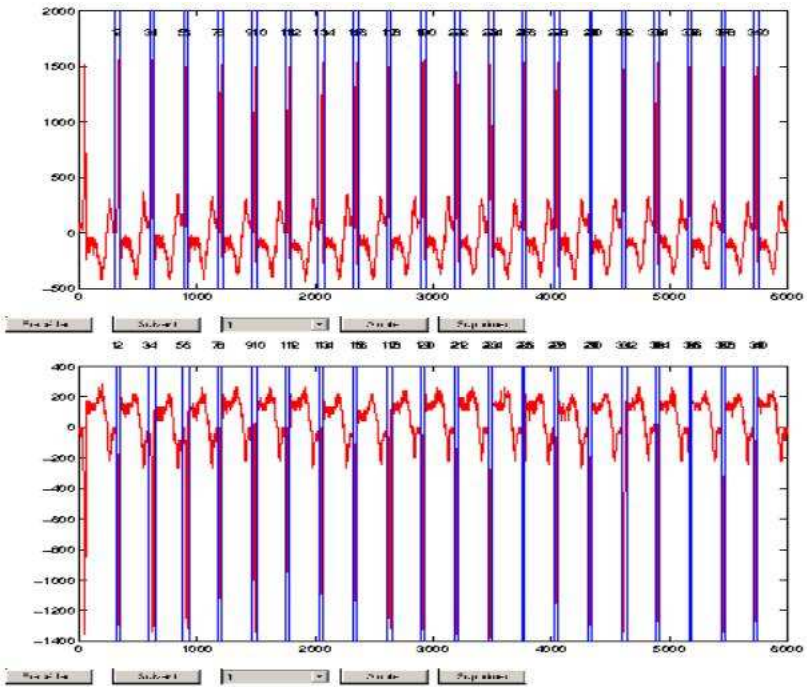


FIG. A.8 – Tachycardie supra ventriculaire : voie II et voie V

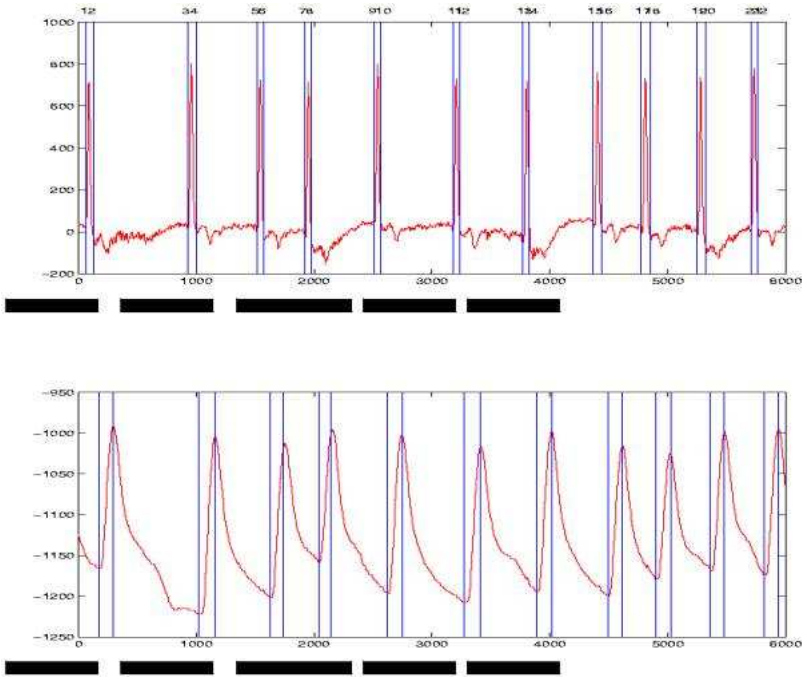


FIG. A.9 – Fibrillation auriculaire : voie II et voie de pression

Annexe B

Généralités sur la logique des prédicats

Cette introduction à la logique des prédicats et au langage PROLOG est basée sur [Lloyd, 1987] et sur les livres [Delahaye, 1988] et [Benzaken, 1991].

B.1 Définitions et notations

Cette partie présente les éléments de base de la logique des prédicats à la base de la plupart des algorithmes de PLI. Après un rappel du vocabulaire élémentaire du langage des prédicats, nous présentons un cas particulier : le langage des clauses utilisé dans PROLOG.

B.1.1 Le langage des prédicats : $\mathcal{L}$

Le vocabulaire utilisé en logique des prédicats est le suivant :
- quatre connecteurs logiques : $\Rightarrow$ (l'implication, binaire), $\neg$ (la négation, unaire), $\wedge$ (le *et* logique, binaire) et $\vee$ ((le *ou* logique, binaire) ;
- deux quantificateurs : $\forall$ (quantification universelle) et $\exists$ (quantification existentielle);
- une infinité de variables (représentées comme dans la notation de PROLOG par des identificateurs commençant par une majuscule);
- une infinité de symbole de fonction d'arité i (notée f/i). Par exemple `somme/2`, `la_soeur_de/2`, `f, g,h` etc. Notons qu'une fonction d'arité 0 est une constante.
- une infinité de symbole de prédicats (fonction à valeur dans $\{vrai, faux\}$). Notons qu'une proposition est un prédicat d'arité 0. Dans la suite, les lettres p, q, r, s désigneront un prédicat.

B.1.2 Grammaire de $\mathcal{L}$

La grammaire du langage des prédicats est la suivante :
`<formule> ::= <atome> |<formule composée>`

```
<formule composée> ::= ¬ <formule> | <formule> connecteurs <formule> |
∀ var <formule> |∃ var <formule>
<atome>  ::= proposition |pred(<terme>{,<terme>*})
<terme> ::= var |constante | fonction(<terme>{,<terme>*})
<littéral> ::=  <atome> | ¬ <atome>
```

- toute variable apparaissant dans une formule doit être quantifiée (pas de formule telle que $p(X) \vee p(a)$);
- les identificateurs de variables doivent être différents quand ils désignent des objets différents (pas de $\forall X, (p(X) \Rightarrow \forall X, r(X))$).

Par exemple $\forall X, (p(f(X)) \Rightarrow q(g(X)) \wedge r(X))$ est une formule bien formée pour le langage des prédicats.

B.1.3 Skolémisation

Définition B.1 (Skolémisation) *La skolémisation d'une formule ϕ consiste à éliminer dans ϕ les quantificateurs existentiels. Les variables quantifiées existentiellement sont alors remplacées par une fonction des variables universellement quantifiées dont le quantificateur universel est placé avant dans la formule ϕ. Si ce n'est pas possible (pas de quantificateur universel placé avant), les variables sont remplacées par de nouvelles constantes.*

Par exemple : $\forall X, \exists Y, p(X, Y)$ est remplacée par $\forall X, p(X, f(X))$; $\exists X, \forall Y, p(X, Y)$ est remplacée par $\forall Y, p(a, Y)$, etc.

La skolémisation est utilisée en PLI dans les systèmes comme Progol ou Aleph pour construire la *bottom clause* (cf. Section 3.1.7).

B.1.4 Le langage des clauses : un cas particulier du langage des prédicats

Le langages des clauses et plus particulièrement celui des clauses de Horn et des clauses définies (voir ci-dessous), permet de ne manipuler que des formules de $\mathcal{L}$ ayant une certaine forme. Les programmes logiques sont composés d'une conjonction de clauses de Horn.

Définition B.2 (Clause) *Une clause est une formule composée d'une disjonction finie de littéraux dans laquelle toutes les variables sont quantifiées universellement.*

Par souci de lisibilité, on simplifie généralement l'écriture de ces clauses en omettant de faire apparaître les quantificateurs universels.

Définition B.3 (Clause de Horn) *Une clause de Horn est une clause contenant au plus un littéral positif.*

Définition B.4 (Clause définie) *Une clause définie est une clause de Horn contenant exactement un littéral positif.*

En PROLOG*, une clause telle que* $h \vee \neg b_1 \vee \ldots \neg b_n, n \geq 0$ *est représentée par la formule syntaxique suivante :* $h{:-}b_1, \ldots, b_n$*.* h *est appelée la* tête de clause*,* $b_1, \ldots, b_n$ *le* corps de clause*. Une clause sans tête (*h*) est un* but *ou une* requête*. Une clause sans corps (*$n = 0$*) est un* fait*.*

Un *programme logique* est un ensemble fini de clauses définies. Par exemple, le programme : $p(X){:-}a(X)$.
$a(1)$.
$a(2)$.
correspond à l'ensemble de formules de $\mathcal{L}$ $\{\forall X, a(X) \Rightarrow p(X),\ a(1),\ a(2)\}$. Ce petit programme logique permet de démontrer $p(1)$ et $p(2)$ en réponse à la requête $p(X)$.

B.2 Calcul dans le langage des prédicats : les interprétations

Pour calculer si une formule est vraie ou fausse, il est nécessaire de l'*interpréter* : le calcul se fait alors à partir de l'interprétation.

Pour interpréter une formule on impose :
- de définir le domaine d'interprétation $D \neq \emptyset$;
- d'assigner un élément de D à chaque identificateur de constante;
- d'assigner un élément de $\{0, 1\}$ à chaque identificateur de proposition;
- de choisir les fonctions d'arité n telles qu'elles soient une application de $D^n \rightarrow D$
- de choisir les prédicats d'arité n tels qu'ils soient une application de $D^n \rightarrow \{0, 1\}$

Par exemple, la formule $\phi = \forall X, (r(X) \Rightarrow q(f(X), a))$ peut être interprétée en prenant :$D = \{4, 5\}$, $f : 4 \rightarrow 5$, $f : 5 \rightarrow 4$, $r : 4 \rightarrow 0$, $r : 5 \rightarrow 1$, $q : (5, 5) \rightarrow 0$, $q : (5, 4) \rightarrow 0$, $q : (4, 4) \rightarrow 1$, $q : (4, 5) \rightarrow 1$.

Pour calculer la valeur de l'interprétation :
- on prend les valeurs assignées par les fonctions/prédicats/propositions si la formule ne contient pas de variables;
- si la formule possède des variables, il y aura une solution par valeur que peuvent prendre ces variables. Une formule du type $\forall X, \phi'$ est vraie si pour toutes les valeurs de X, le calcul de l'interprétation est égale à 1. Une formule du type $\exists X, \phi'$ est vraie si pour au moins une des valeurs de X le calcul de l'interprétation est égale à 1.

Définition B.5 (Modèle) *Une interprétation* I *est modèle d'une formule* ϕ *si la valeur de* ϕ *selon* I *est vraie.*

Définition B.6 (Satisfaisabilité) *Un ensemble de formules* $\mathcal{H}$ *est satisfaisable ssi il existe une interprétation telle que pour toute formule* $\phi \in \mathcal{H}$*,* I *est modèle de* ϕ*.*

Définition B.7 (Conséquence logique) *Une formule* ϕ *est conséquence logique d'un ensemble de formule* $\mathcal{H}$ *(noté* $\mathcal{H} \models \phi$*) ssi tout modèle de* $\mathcal{H}$ *est modèle de* ϕ*.*

Vérifier la relation de conséquence logique entre deux ensembles de formules nécessite de s'assurer que tous les modèles de l'un sont modèles de l'autre. Dans le cas de la logique des prédicats, le nombre de modèles est potentiellement infini, il est donc impossible de vérifier la conséquence logique directement. Heureusement, un sous ensemble spécial d'interprétations nommé *ensemble des interprétations de Herbrand* suffit, sous certaines conditions, à vérifier la conséquence logique.

Définition B.8 (Univers de Herbrand) *L'univers de Herbrand d'un ensemble de formules (avec au moins une constante) est l'ensemble de tous les termes clos (qui ne contiennent plus de variables) pouvant être formés avec les symboles de constantes et de fonctions utilisés dans les formules. Si aucune constante n'est utilisée, on en ajoute arbitrairement une.*

Pour illustrer cette définition, considérons l'ensemble de formules $\mathcal{H}$ composé de $\forall X(p(a,X) \Rightarrow p(X,b))$ et $\forall X \forall Y \forall Z((p(X,Y) \wedge p(Y,Z)) \Rightarrow p(X,Z))$. L'univers de Herbrand de $\mathcal{H}$ est a, b.

Définition B.9 (Base de Herbrand) *La base de Herbrand d'un ensemble de formules ϕ est l'ensemble des atomes clos qui peuvent être formés en utilisant les symboles de prédicats de ϕ et en argument, les termes de l'univers de Herbrand de ϕ.*

La base de Herbrand de l'ensemble de formules $\mathcal{H}$ est $\{p(a,a), p(a,b), p(b,a), p(b,b)\}$.

Définition B.10 (Interpétation de Herbrand) *Une interprétation de Herbrand d'un ensemble de formules est n'importe quel sous-ensemble de sa base de Herbrand.*

Les interprétations de Herbrand de l'ensemble $\mathcal{H}$ précédent sont (par défaut on ne met dans l'interprétation que les prédicats évalués à 1) :

$\{\}$	$\{p(b,a), p(b,b)\}$
$\{p(a,a)\}$	$\{p(a,b), p(b,a)\}$
$\{p(a,b)\}$	$\{p(a,b), p(b,b)\}$
$\{p(b,a)\}$	$\{p(a,a), p(b,a), p(b,b)\}$
$\{p(b,b)\}$	$\{p(a,a), p(a,b), p(b,a)\}$
$\{p(a,a), p(a,b)\}$	$\{p(a,a), p(a,b), p(b,b)\}$
$\{p(a,a), p(b,a)\}$	$\{p(a,b), p(b,a), p(b,b)\}$
$\{p(a,a), p(b,b)\}$	$\{p(a,a), p(a,b), p(b,a), p(b,b)\}$

Définition B.11 (Modèle de Herbrand) *Un modèle de Herbrand d'un ensemble de formules ϕ est une interprétation de Herbrand de ϕ qui est modèle de ϕ.*

Ces modèles de Herbrand sont très utiles puisqu'ils permettent, grâce au théorème suivant, de décider plus simplement de la satisfaisabilité des formules. Dans l'exemple précédent, $\{p(b,b)\}$ est un modèle de Herbrand de $\mathcal{H}$ donc $\mathcal{H}$ est satisfaisable. Notons que $\{\}$ est également un modèle de Herbrand de $\mathcal{H}$, de même que $\{p(a,b), p(b,b)\}$ etc.

Propriété 6 *Si l'ensemble de clauses S est satisfaisable alors il possède un modèle de Herbrand. On en déduit que S est insatisfaisable ssi S ne possède pas de modèle de Herbrand.*

L'intérêt de cette propriété est qu'elle permet de se limiter à un nombre restreint de modèles lorsque l'on veut vérifier l'insatisfaisabilité d'une formule clausale.

Un cas particulier se pose lorsque que l'on travaille avec des programme définis, c'est à dire des programmes ne possédant que des clauses définies. Dans ce cas, il existe un unique *plus petit modèle de Herbrand* que l'on choisira pour prouver la satisfaisabilité des formules. Ce plus petit modèle est calculé théoriquement comme l'intersection de tous les modèles de Herbrand (dans l'exemple précédent il s'agit de {}) mais intuitivement, en programmation logique, il s'agit du plus petit ensemble de connaissances positives qui permet de satisfaire le programme.

B.3 Fonctionnement de PROLOG : l'exécution SLD

Les exécutions de la programmation logique sont souvent vues comme des arbres appelés *arbres de recherche*. Cette structure arborescente vient du fait que, pour un même but, on peut généralement utiliser plusieurs clauses. Les nœuds de l'arbre représentent donc le but courant du programme : *la résolvante*. Chaque branche qui en part symbolise l'application d'une des clauses possibles. Les feuilles de l'arbre sont soit des échecs (les choix faits mènent à une impasse), sont des réussites (il n'y a plus rien à prouver). L'exécution est décrite par le parcours de l'arbre de recherche en profondeur d'abord, en prenant toujours en premier la branche la plus à gauche, on la nomme *exécution SLD : fonction de Selection pour résolution Linéaire et pour clauses Définies* . Lorsque l'on arrive à une feuille, on revient en arrière sur le dernier nœud rencontré. C'est le *backtracking*. Un tel parcours est schématisé par les flèches de la Figure B.1.

```
(1) p(X):- q(X), r(X).
(2) p(X):- t(X).
(3) q(a).
(4) q(b).
(5) r(a).
(6) t(a).
(7) t(c).
```

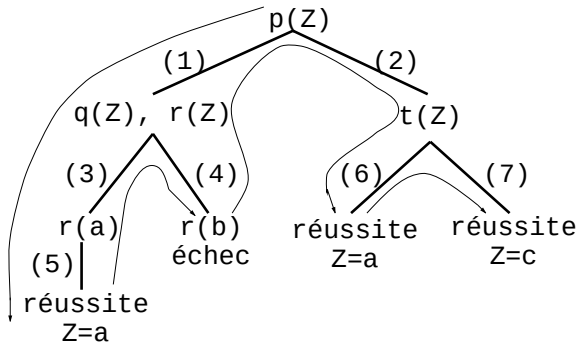


FIG. B.1 – Un petit programme *Prolog* et l'arbre de recherche associé au but *p(Z)*, avec le parcours fait par l'exécution.

Prenons l'exemple du petit programme présenté en Figure B.1. Chaque ligne de ce programme est une clause. Les deux premières clauses de ce programme peuvent se

lire ainsi : “ Pour prouver `p(X)`, il faut prouver `q(X)` *et* `r(X)` *ou bien* il faut prouver `t(X)` ”. La suite signifie que `q(a)` est vrai, de même que `q(b)`, `r(a)`, `t(a)` et `t(c)`, où `a`, `b` et `c` sont des constantes. L'exécution d'un tel programme est définie comme la *résolution* d'un but, c'est-à-dire sa démonstration. Prenons comme but à démontrer `p(Z)`. `Z` commençant par une majuscule, le système en déduit que c'est une variable à laquelle il peut donner n'importe quelle valeur. À la fin de l'exécution, le système répond : `Z = a ; Z = a ; Z = c`, ce qui se lit “ `Z` est égal à `a`, `a` *ou* `c` ”. `Z = a` est une *substitution*. Chaque réponse correspond à un arbre de preuve particulier. La première réponse trouvée (`Z = a`) vient de la première clause : `p(a)` est vrai car `q(a)` et `r(a)` sont vrais. La deuxième réponse vient de la deuxième clause et donne, une nouvelle fois, le résultat `Z = a`. La répétition signale qu'il y a deux manières de démontrer `p(a)`. La deuxième clause donne en plus un dernier résultat, `Z = c`. On remarquera qu'il n'y a pas `Z = b`. Le système peut prouver `q(b)` mais aucune clause ne lui permet de prouver `r(b)` : il y a échec.

Annexe C

Détails des apprentissages par PLI

C.1 Apprentissages monosources avec ICL

C.1.1 Codage de l'ECG et de la pression

Pour les voies I et V, les exemples (base de connaissances) sont représentés par une succession de prédicats p_wave/10 et qrs_complex/10 dont la description est :

```
p_wave(
<1- nom de l'onde >,
<2 - instant d'apparition de l'onde>,
<3 - forme de l'onde>,
<4 - nom de l'onde précédente>,
<5 - nom de l'onde précédente de même type>,
<6 - valeur de l'intervalle pp1>,
<7 - nom de l'onde deux fois précédente de même type>,
<8 - valeur de l'intervalle pp2>,
<9 - nom de l'onde précédente de type différent>,
<10 - valeur de l'intervalle rp1>)

qrs_complex(
<1- nom de l'onde >,
<2 - instant d'apparition de l'onde>,
<3 - forme de l'onde>,
<4 - nom de l'onde précédente>,
<5 - nom de l'onde précédente de même type>,
<6 - valeur de l'intervalle rr1>,
<7 - nom de l'onde deux fois précédente de même type>,
<8 - valeur de l'intervalle rr2>,
<9 - nom de l'onde précédente de type différent>,
```

```
<10 - valeur de l'intervalle pr1>,
<11 - nom de l'onde deux fois précédente de type différent>,
<12 - valeur de l'intervalle pr2>)
```

En ce qui concerne la voie hémodynamique, les exemples sont représentés par une succession de prédicats diastole/11 et systole/11 dont la description est :

```
diastole(<1- nom de l'onde >,
<2 - instant d'apparition de l'onde>,
<3 - amplitude de l'onde>,
<4 - nom de l'onde précédente>,
<5 - différence d'amplitude entre la diastole et la systole précédente>,
<6 - intervalle de temps entre la systole précédente et l'onde>,
<7 - nom de l'onde précédente de même type>,
<8 - valeur de l'intervalle dd1>,
<9 - nom de l'onde deux fois précédente de même type>,
<10 - valeur de l'intervalle dd2>,
<11 - différence d'amplitude entre la diastole et diastole précédente>)

systole(<1- nom de l'onde >,
<2 - instant d'apparition de l'onde>,
<3 - amplitude de l'onde>,
<4 - nom de l'onde précédente>,
<5 - différence d'amplitude entre la systole et la diastole précédente>,
<6 - intervalle de temps entre la diastole précédente et l'onde>,
<7 - nom de l'onde précédente de même type>,
<8 - valeur de l'intervalle ss1>,
<9 - nom de l'onde deux fois précédente de même type>,
<10 - valeur de l'intervalle ss2>,
<11 - différence d'amplitude entre la systole et systole précédente>)
```

Notons qu'une diastole suit toujours une systole et inversement, alors qu'il peut manquer une onde QRS entre deux ondes P ou au contraire une onde P entre deux onde QRS.

C.1.2 Langage de biais

C.1.2.1 Voie V (pas d'onde P)

```
dlab_template('
 false       <--
len-len:[

        1-1:[len-len:[
        qrs(R0,w_feature),
```

```
        0-1:[len-len:[
        qrs(R1,w_feature),suc(R1,R0),
        0-1:[rr1(R0, R1, r_feature)],

        0-1:[len-len:[
        qrs(R2, w_feature),suc(R2,R1),
        0-1:[rr1(R1, R2, r_feature)],
        0-1:[rr2(R0, R2, r_feature)],
        0-1:[rythm(R0,R1, R2, rythm_feature)],

%plus 3 autres cycles cardiaques sur le même modèle
        ]]
        ]]
        ]]
        ]
').
dlab_variable(rythm_feature, 1-1, [regular,irregular]).
dlab_variable(r_feature, 1-1, [short, normal, long]).
dlab_variable(w_feature, 1-1, [normal, abnormal]).
```

C.1.2.2 Voie V sans la forme du QRS

```
dlab_template('
 false       <--
len-len:[

        1-1:[len-len:[
        qrs(R0),

        0-1:[len-len:[
        qrs(R1),suc(R1,R0),
        0-1:[rr1(R0, R1, r_feature)],

        0-1:[len-len:[
        qrs(R2),suc(R2,R1),
        0-1:[rr1(R1, R2, r_feature)],
        0-1:[rr2(R0, R2, r_feature)],
        0-1:[rythm(R0,R1, R2, rythm_feature)],
%plus 3 autres cycles cardiaques sur le même modèle
        ]]
        ]]
        ]]
        ]
```

```
').
dlab_variable(rythm_feature, 1-1, [regular,irregular]).
dlab_variable(r_feature, 1-1, [short, normal, long]).
```

C.1.2.3 Pression sans diastole

```
dlab_template(' false <--
len-len:[
            1-1:[len-len:[
           systole(Sys0),

          0-1:[len-len:[
          systole(Sys1),suc(Sys1,Sys0),
          0-1:[amp_ss(Sys0, Sys1,  amp_featureS,  amp_feature)],
          0-1:[ss1(Sys0, Sys1, r_feature)],


          0-1:[len-len:[
          systole(Sys2),suc(Sys2,Sys1),
          0-1:[amp_ss(Sys1, Sys2, amp_featureS,  amp_feature)],
          0-1:[ss1(Sys1, Sys2, r_feature)],
          0-1:[ss2(Sys0, Sys2, r_feature)],
%plus 2 autres cycles cardiaques sur le même modèle
          ]]
          ]]
          ]]
      ]
  ').

dlab_variable(r_feature, 1-1, [short, normal, long]).
dlab_variable(amp_feature, 1-1, [short, normal, long]).
dlab_variable(amp_featureS, 1-1, [pos, neg,nul]).
```

C.1.3 Résultats

Cette section donne tous les résultats non présentés dans la section 3.2.2 qui seront utile dans le chapitre sur l'apprentissage multisource.

C.1.3.1 Résultats de la voie V

```
class(rs,[8,1,10,6,6,0,3],[0,6,0,0,1,5,4]) :-
qrs(R0,normal),
qrs(R1,normal),
suc(R1,R0), rr1(R0,R1,normal), qrs(R2,normal),
suc(R2,R1), qrs(R3,normal),
```

```
suc(R3,R2), rr2(R1,R3,normal), qrs(R4,normal),
suc(R4,R3), qrs(R5,normal),
suc(R5,R4), rr2(R3,R5,normal).

class(bige,[0,7,0,0,0,0,0],[8,0,10,6,7,5,7]) :-
qrs(R0,normal),
qrs(R1,abnormal),
suc(R1,R0), qrs(R2,normal),
suc(R2,R1), qrs(R3,abnormal),
suc(R3,R2).

class(esv,[0,0,10,0,0,0,0],[8,7,0,6,7,5,7]) :-
qrs(R0,normal),
qrs(R1,normal),
suc(R1,R0), qrs(R2,normal),
suc(R2,R1), qrs(R3,abnormal),
suc(R3,R2), qrs(R4,normal),
suc(R4,R3), qrs(R5,normal),
suc(R5,R4).

class(doublet,[0,0,0,6,0,0,0],[8,7,10,0,7,5,7]) :-
qrs(R0,normal),
qrs(R1,abnormal),
suc(R1,R0), qrs(R2,abnormal),
suc(R2,R1), qrs(R3,normal),
suc(R3,R2).

class(tv,[0,0,0,0,7,0,0],[8,7,10,6,0,5,7]) :-
qrs(R0,abnormal),
qrs(R1,abnormal),
suc(R1,R0), qrs(R2,abnormal),
suc(R2,R1).

class(tsv,[0,0,0,0,0,5,2],[8,7,10,6,7,0,5]) :-
qrs(R0,normal),
qrs(R1,normal),
suc(R1,R0), rr1(R0,R1,short), qrs(R2,normal),
suc(R2,R1), qrs(R3,normal),
suc(R3,R2), rr2(R1,R3,short).

class(fa,[0,0,0,1,0,0,6],[8,7,10,5,7,5,1]) :-
qrs(R0,normal),
qrs(R1,normal),
suc(R1,R0), qrs(R2,normal),
```

```
suc(R2,R1), rythm(R0,R1,R2,irregular).
```

C.1.3.2 Résultats de la voie V sans utiliser la forme du QRS

```
class(rs,[8,2,10,6,7,0,1],[0,5,0,0,0,5,6]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), qrs(R2),
suc(R2,R1), rr2(R0,R2,normal),
rythm(R0,R1,R2,regular), qrs(R3),
suc(R3,R2), rr1(R2,R3,normal), qrs(R4),
suc(R4,R3), rr1(R3,R4,normal), qrs(R5),
suc(R5,R4), rr1(R4,R5,normal).

class(bige,[0,6,0,0,0,0,0],[8,1,10,6,7,5,7]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), qrs(R2),
suc(R2,R1), rr1(R1,R2,short),
rr2(R0,R2,normal), qrs(R3),
suc(R3,R2), rr1(R2,R3,long), qrs(R4),
suc(R4,R3), qrs(R5),
suc(R5,R4).


class(esv,[0,1,2,0,0,0,0],[8,6,8,6,7,5,7]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), rr1(R0,R1,normal), qrs(R2),
suc(R2,R1), rr1(R1,R2,normal), qrs(R3),
suc(R3,R2), rythm(R1,R2,R3,regular), qrs(R4),
suc(R4,R3), rr1(R3,R4,short), qrs(R5),
suc(R5,R4), rr1(R4,R5,long).

%idem rs
class(esv,[8,2,10,6,7,0,1],[0,5,0,0,0,5,6]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), qrs(R2),
suc(R2,R1), rr2(R0,R2,normal),
rythm(R0,R1,R2,regular), qrs(R3),
suc(R3,R2), rr1(R2,R3,normal), qrs(R4),
suc(R4,R3), rr1(R3,R4,normal), qrs(R5),
suc(R5,R4), rr1(R4,R5,normal).
```

```
class(doublet,[0,0,0,2,1,0,0],[8,7,10,4,6,5,7]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), rr1(R0,R1,short), qrs(R2),
suc(R2,R1), rr2(R0,R2,short), qrs(R3),
suc(R3,R2), rr2(R1,R3,normal),
rythm(R1,R2,R3,regular), qrs(R4),
suc(R4,R3), rr1(R3,R4,normal).

class(doublet,[0,0,0,2,3,0,2],[8,7,10,4,4,5,5]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), qrs(R2),
suc(R2,R1), rr2(R0,R2,short),
rythm(R0,R1,R2,regular), qrs(R3),
suc(R3,R2), rr2(R1,R3,normal), qrs(R4),
suc(R4,R3), rr1(R3,R4,normal).

class(tv,[0,0,0,3,6,0,2],[8,7,10,3,1,5,5]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), rr1(R0,R1,short), qrs(R2),
suc(R2,R1), rr2(R0,R2,short), qrs(R3),
suc(R3,R2), rr2(R1,R3,normal), qrs(R4),
suc(R4,R3), rr1(R3,R4,normal), qrs(R5),
suc(R5,R4).

class(tsv,[0,0,0,0,0,5,2],[8,7,10,6,7,0,5]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), rr1(R0,R1,short), qrs(R2),
suc(R2,R1), rythm(R0,R1,R2,regular), qrs(R3),
suc(R3,R2), rr1(R2,R3,short), qrs(R4),
suc(R4,R3), rr1(R3,R4,short), qrs(R5),
suc(R5,R4), rr1(R4,R5,short).

class(fa,[0,0,0,0,0,0,3],[8,7,10,6,7,5,4]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), rr1(R0,R1,short), qrs(R2),
suc(R2,R1), rr2(R0,R2,short),
rythm(R0,R1,R2,irregular), qrs(R3),
suc(R3,R2), rr2(R1,R3,short).
```

monosource ECG					
	CorAp	CorT	Nbcycl	Nds	Tps
rs	0.54	0.54	6	767	87.1
esv	0.76	0.74	6	888	44.51
bige	1	0.98	4	139	7.37
doublet	0.72	0.66	3/6	1242	55.11
tv	0.92	0.9	6	1058	10.77
tsv	1	1	5	11	0.52

TAB. C.1 – Résultats de la validation croisée pour l' apprentissage sur la voie I avec l'onde P seule

```
class(fa,[0,0,0,0,0,0,3],[8,7,10,6,7,5,4]) :-
qrs(R0),
qrs(R1),
suc(R1,R0), rr1(R0,R1,long), qrs(R2),
suc(R2,R1), qrs(R3),
suc(R3,R2), rr1(R2,R3,normal), qrs(R4),
suc(R4,R3), rr1(R3,R4,long).
```

C.1.3.3 Ondes P seules

Ce type d'apprentissages appris ici à partir de la voie I en n'utilisant que les informations sur l'onde P pourrait être appris à partir de signaux provenant d'une sonde œsophagienne.

```
class(rs,[8,1,10,6,6,0,0],[0,6,0,0,1,5,7]) :-
p_wav(P0,normal), p_wav(P1,normal), suc(P1,P0),
p_wav(P2,normal), suc(P2,P1),
pp2(P0,P2,normal),
p_wav(P3,normal), suc(P3,P2),
rythm(P1,P2,P3,regular),
p_wav(P4,normal), suc(P4,P3),
 p_wav(P5,normal), suc(P5,P4),
pp1(P4,P5,normal).

class(bige,[0,7,0,0,0,0,0],[8,0,10,6,7,5,7]) :-
p_wav(P0,normal), p_wav(P1,normal), suc(P1,P0),
pp1(P0,P1,long),
p_wav(P2,normal), suc(P2,P1),
rythm(P0,P1,P2,regular),
p_wav(P3,normal), suc(P3,P2).
```

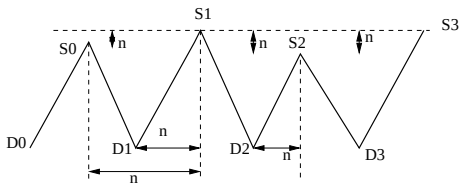
```
class(esv,[0,1,8,5,4,0,0],[8,6,2,1,3,5,7]) :-
p_wav(P0,normal), p_wav(P1,normal), suc(P1,P0),
p_wav(P2,normal), suc(P2,P1),
pp1(P1,P2,normal), rythm(P0,P1,P2,irregular),
p_wav(P3,normal), suc(P3,P2),
p_wav(P4,normal), suc(P4,P3),
pp2(P2,P4,normal),
p_wav(P5,normal), suc(P5,P4).

class(doublet,[0,0,0,3,4,0,0],[8,7,10,3,3,5,7]) :-
p_wav(P0,normal), p_wav(P1,normal), suc(P1,P0),
pp1(P0,P1,verylong),
p_wav(P2,normal), suc(P2,P1),
pp1(P1,P2,normal).

class(doublet,[0,1,8,5,4,0,0],[8,6,2,1,3,5,7]) :-
p_wav(P0,normal), p_wav(P1,normal), suc(P1,P0),
p_wav(P2,normal), suc(P2,P1),
pp1(P1,P2,normal),
rythm(P0,P1,P2,irregular),
p_wav(P3,normal), suc(P3,P2),
p_wav(P4,normal), suc(P4,P3),
pp1(P3,P4,normal),
p_wav(P5,normal), suc(P5,P4).

class(tv,[0,0,0,3,6,0,0],[8,7,10,3,1,5,7]) :-
p_wav(P0,normal), p_wav(P1,normal), suc(P1,P0),
pp1(P0,P1,verylong).

class(tsv,[0,0,0,0,0,5,0],[8,7,10,6,7,0,7]) :-
p_wav(P0,normal), p_wav(P1,normal),
suc(P1,P0), pp1(P0,P1,short).

Pas d'ondes P :
class(fa,[0,0,0,0,0,0,7],[8,7,10,6,7,5,0]) :-
\\+p_wav(_,_).
```

La règle pour la *fa* n'est pas utilisable pour la reconnaissance en ligne.

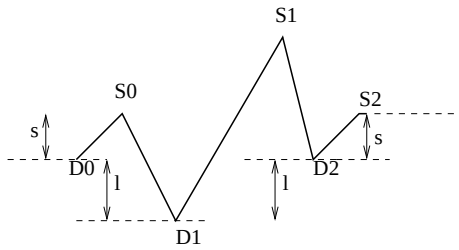
C.1.3.4 Résultats obtenus avec la pression seule

```
class(rs,[8,0,0,0,0,0,0],[0,7,10,6,7,5,7]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
```

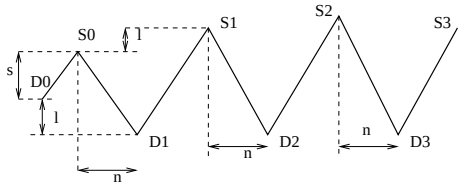
```
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_ss(Sys0,Sys1,nul,normal),
ss1(Sys0,Sys1,normal),
ds1(Dias1,Sys1,normal),
cycle_ABP(Dias2,normal,Sys2,normal), suc(Dias2,Sys1), suc(Sys2,Dias2),
amp_ss(Sys1,Sys2,nul,normal),
ds1(Dias2,Sys2,normal),
cycle_ABP(Dias3,normal,Sys3,normal), suc(Dias3,Sys2), suc(Sys3,Dias3),
cycle_ABP(Dias4,normal,Sys4,normal), suc(Dias4,Sys3), suc(Sys4,Dias4),
amp_ss(Sys3,Sys4,nul,normal).
```



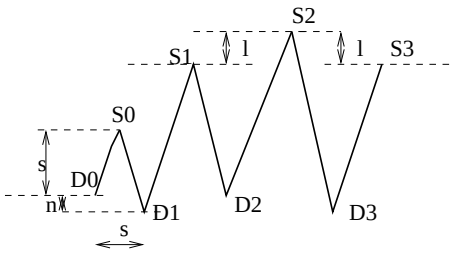

```
class(bige,[0,6,0,0,1,0,0],[8,1,10,6,6,5,7]) :-
cycle_ABP(Dias0,_,Sys0,short), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_dd(Dias0,Dias1,neg,long),
cycle_ABP(Dias2,normal,Sys2,short), suc(Dias2,Sys1), suc(Sys2,Dias2),
amp_dd(Dias1,Dias2,pos,long).
```




```
class(esv,[0,0,5,0,0,0,0],[8,7,5,6,7,5,7]) :-
cycle_ABP(Dias0,_,Sys0,short), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_ss(Sys0,Sys1,pos,long),
amp_dd(Dias0,Dias1,neg,long),
 sd1(Sys0,Dias1,normal),
cycle_ABP(Dias2,normal,Sys2,normal), suc(Dias2,Sys1), suc(Sys2,Dias2),
sd1(Sys1,Dias2,normal),
cycle_ABP(Dias3,normal,Sys3,normal), suc(Dias3,Sys2), suc(Sys3,Dias3),
sd1(Sys2,Dias3,normal).
```

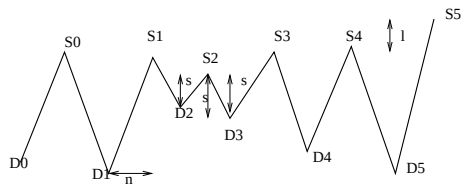



```
class(esv,[0,0,5,0,0,0,0],[8,7,5,6,7,5,7]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_dd(Dias0,Dias1,neg,long),
ds1(Dias1,Sys1,long), sd1(Sys0,Dias1,normal),
cycle_ABP(Dias2,normal,Sys2,normal), suc(Dias2,Sys1), suc(Sys2,Dias2),
ds1(Dias2,Sys2,long), sd1(Sys1,Dias2,normal),
cycle_ABP(Dias3,normal,Sys3,normal), suc(Dias3,Sys2), suc(Sys3,Dias3),
ss1(Sys2,Sys3,normal),
cycle_ABP(Dias4,normal,Sys4,normal), suc(Dias4,Sys3), suc(Sys4,Dias4),
ss1(Sys3,Sys4,normal).

class(esv,[0,0,5,0,0,0,0],[8,7,5,6,7,5,7]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_ss(Sys0,Sys1,pos,long),
amp_dd(Dias0,Dias1,nul,normal),
sd1(Sys0,Dias1,normal),
cycle_ABP(Dias2,normal,Sys2,normal), suc(Dias2,Sys1), suc(Sys2,Dias2),
amp_dd(Dias1,Dias2,pos,long).

class(doublet,[0,0,0,3,2,0,0],[8,7,10,3,5,5,7]) :-
cycle_ABP(Dias0,_,Sys0,short), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_dd(Dias0,Dias1,nul,normal),
dd1(Dias0,Dias1,short),
cycle_ABP(Dias2,normal,Sys2,normal), suc(Dias2,Sys1), suc(Sys2,Dias2),
amp_ss(Sys1,Sys2,pos,long),
cycle_ABP(Dias3,normal,Sys3,normal), suc(Dias3,Sys2), suc(Sys3,Dias3),
amp_ss(Sys2,Sys3,neg,long).
```



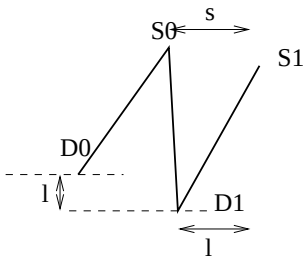
```
class(doublet,[0,0,0,3,0,0,0],[8,7,10,3,7,5,7]) :-
cycle_ABP(Dias0,_,Sys0,short), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
cycle_ABP(Dias2,normal,Sys2,normal), suc(Dias2,Sys1), suc(Sys2,Dias2),
amp_ss(Sys1,Sys2,pos,long), sd1(Sys1,Dias2,normal),
cycle_ABP(Dias3,normal,Sys3,normal), suc(Dias3,Sys2), suc(Sys3,Dias3),
cycle_ABP(Dias4,normal,Sys4,normal), suc(Dias4,Sys3), suc(Sys4,Dias4),
amp_dd(Dias3,Dias4,pos,long).
```

```
class(tv,[0,0,0,0,5,0,0],[8,7,10,6,2,5,7]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
ds1(Dias1,Sys1,normal),
cycle_ABP(Dias2,short,Sys2,short), suc(Dias2,Sys1), suc(Sys2,Dias2),
cycle_ABP(Dias3,short,Sys3,normal), suc(Dias3,Sys2), suc(Sys3,Dias3),
cycle_ABP(Dias4,normal,Sys4,normal), suc(Dias4,Sys3), suc(Sys4,Dias4),
cycle_ABP(Dias5,normal,Sys5,normal), suc(Dias5,Sys4), suc(Sys5,Dias5),
amp_ss(Sys4,Sys5,pos,long).
```

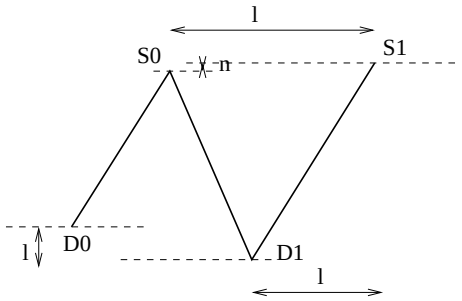



```
class(tv,[0,0,0,0,5,0,0],[8,7,10,6,2,5,7]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,short), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_dd(Dias0,Dias1,neg,long),
cycle_ABP(Dias2,normal,Sys2,normal), suc(Dias2,Sys1), suc(Sys2,Dias2),
amp_dd(Dias1,Dias2,pos,long),
cycle_ABP(Dias3,normal,Sys3,normal), suc(Dias3,Sys2), suc(Sys3,Dias3),
ds1(Dias3,Sys3,normal).
```

```
class(tsv,[0,0,0,0,0,5,0],[8,7,10,6,7,0,7]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_dd(Dias0,Dias1,neg,long),
ss1(Sys0,Sys1,short), ds1(Dias1,Sys1,long).
```




```
class(fa,[0,0,0,0,0,0,4],[8,7,10,6,7,5,3]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_ss(Sys0,Sys1,nul,normal), amp_dd(Dias0,Dias1,neg,long),
ss1(Sys0,Sys1,long), ds1(Dias1,Sys1,long).
```




```
class(fa,[0,0,0,0,0,0,4],[8,7,10,6,7,5,3]) :-
cycle_ABP(Dias0,_,Sys0,normal), suc(Sys0,Dias0),
cycle_ABP(Dias1,normal,Sys1,normal), suc(Dias1,Sys0), suc(Sys1,Dias1),
amp_ss(Sys0,Sys1,nul,normal), ss1(Sys0,Sys1,short),
ds1(Dias1,Sys1,normal),
cycle_ABP(Dias2,normal,Sys2,normal), suc(Dias2,Sys1), suc(Sys2,Dias2),
amp_dd(Dias1,Dias2,neg,long).
```

C.1.3.5 Résultats avec la pression sans information sur la diastole

```
class(rs,[7,0,0,0,0,0,0],[1,7,10,6,7,5,7]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,nul,normal),
systole(Sys2), suc(Sys2,Sys1),
amp_ss(Sys1,Sys2,nul,normal),
ss2(Sys0,Sys2,normal),
systole(Sys3), suc(Sys3,Sys2),
amp_ss(Sys2,Sys3,nul,normal),
systole(Sys4), suc(Sys4,Sys3),
amp_ss(Sys3,Sys4,nul,normal).
```

monosource ABP					
	CorAp	CorT	Nbcycl	Nds	Tps
rs	1	1	5	1511	164
esv	0.928	0.80	5	2790	210
bige	0.997	0.84	4/5	3517	246
doublet	0.982	0.86	4/5	4619	518
tv	0.93	0.82	4/4	2995	355
tsv	0.96	0.82	4	2333	316
fa	0.978	0.78	5/5	3762	549

TAB. C.2 – Résultats de la validation croisée pour l'apprentissage sur la voie de pression sans la diastole

```
[mrs_8_ABP] non couvert

class(bige,[0,6,0,0,0,0,0],[8,1,10,6,7,5,7]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,pos,long),
ss1(Sys0,Sys1,long),
systole(Sys2), suc(Sys2,Sys1),
systole(Sys3), suc(Sys3,Sys2),
ss1(Sys2,Sys3,long).

[mbige_2_ABP] non couvert

class(esv,[1,0,9,1,1,0,0],[7,7,1,5,6,5,7]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,neg,long),
ss1(Sys0,Sys1,normal),
systole(Sys2), suc(Sys2,Sys1),
amp_ss(Sys1,Sys2,nul,normal),
ss1(Sys1,Sys2,normal),
systole(Sys3), suc(Sys3,Sys2),
ss1(Sys2,Sys3,normal),
systole(Sys4), suc(Sys4,Sys3),
ss1(Sys3,Sys4,normal).

[mesv_9_ABP]  non couvert

class(doublet,[0,0,0,4,0,0,0],[8,7,10,2,7,5,7]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,pos,long),
```

```
systole(Sys2), suc(Sys2,Sys1),
ss2(Sys0,Sys2,short),
systole(Sys3), suc(Sys3,Sys2),
ss1(Sys2,Sys3,normal),
systole(Sys4), suc(Sys4,Sys3),
amp_ss(Sys3,Sys4,pos,long).

class(doublet,[0,0,0,2,0,0,0],[8,7,10,4,7,5,7]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,neg,long),
ss1(Sys0,Sys1,normal),
systole(Sys2), suc(Sys2,Sys1),
amp_ss(Sys1,Sys2,neg,long),
systole(Sys3), suc(Sys3,Sys2),
ss1(Sys2,Sys3,long),
ss2(Sys1,Sys3,normal),
systole(Sys4), suc(Sys4,Sys3),
ss2(Sys2,Sys4,normal).

class(tv,[0,0,0,0,3,0,0],[8,7,10,6,4,5,7]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,neg,long),
systole(Sys2), suc(Sys2,Sys1),
ss1(Sys1,Sys2,normal), ss2(Sys0,Sys2,long).

class(tv,[0,1,0,0,3,0,0],[8,6,10,6,4,5,7]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
ss1(Sys0,Sys1,short),
systole(Sys2), suc(Sys2,Sys1),
ss1(Sys1,Sys2,short),
ss2(Sys0,Sys2,short),
systole(Sys3), suc(Sys3,Sys2),
amp_ss(Sys2,Sys3,pos,long),
systole(Sys4), suc(Sys4,Sys3),
ss1(Sys3,Sys4,normal).

[mtv_4_ABP] non couvert

class(tsv,[0,0,0,0,1,5,1],[8,7,10,6,6,0,6]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,nul,normal),
ss1(Sys0,Sys1,short),
systole(Sys2), suc(Sys2,Sys1),
amp_ss(Sys1,Sys2,neg,long),
```

```
ss2(Sys0,Sys2,short),
systole(Sys3), suc(Sys3,Sys2),
ss1(Sys2,Sys3,short).


class(fa,[0,0,0,0,0,0,5],[8,7,10,6,7,5,2]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,nul,normal),
systole(Sys2), suc(Sys2,Sys1),
ss1(Sys1,Sys2,long),
systole(Sys3), suc(Sys3,Sys2),
systole(Sys4), suc(Sys4,Sys3),
ss2(Sys2,Sys4,normal).

class(fa,[0,0,0,0,0,4,3],[8,7,10,6,7,1,4]) :-
systole(Sys0), systole(Sys1), suc(Sys1,Sys0),
amp_ss(Sys0,Sys1,nul,normal),
ss1(Sys0,Sys1,short),
systole(Sys2), suc(Sys2,Sys1),
systole(Sys3), suc(Sys3,Sys2),
amp_ss(Sys2,Sys3,neg,long),
systole(Sys4), suc(Sys4,Sys3).
```

C.2 Apprentissages monosources avec Aleph

C.2.1 Codage de l'ECG et de la pression et connaissances du domaine

En plus des prédicats utilisés dans les règles présentées dans la section 3.2.3, la théorie du domaine est composée de prédicats générés directement à partir des données d'ICL qui contiennent les valeurs numériques des attributs, la forme et les instants d'apparition des ondes etc. Ces prédicats sont trop précis pour être utilisés directement lors de l'apprentissage mais ils sont utiles pour définir des prédicats plus complexes qui permettent d'offrir un premier biais syntaxique pour l'apprentissage. La description de ces prédicats *simples* est donnée ci-dessous :

- Les prédicats `p_I/1` ou `p_V/1`, `qrs_I/1` ou `qrs_V/1`, `diastole/1` et `systole/1` donnent le type de chaque onde.
- `has_wave/2` : relie une onde au numéro de l'exemple auquel elle appartient.
- `shape(+nom-onde,+forme)` (pour l'ECG uniquement) associe à chaque onde sa forme (*normal* ou *anormal*).
- `pression(+nom-onde,+valeur)` (pour la pression uniquement) associe à chaque onde la valeur numérique de son amplitude.
- `suc(+nom-onde1,+nom-onde2)` : l'onde `nom-onde2` suit l'onde `nom-onde1`.
- `time/2` donne pour chaque onde l'instant à laquelle elle apparaît dans l'ECG.
- Tous les intervalles caractéristiques de l'ECG

(`pp1(+nom-onde-p1,+nom-onde-p2, valeur)`, etc.) et ceux de la pression (`ampds(nom-onde-d1,nom-onde-s2,valeur)`, etc.) ont également été codés.

C.2.2 Résultats

C.2.2.1 Voie I

```
[Rule 1] [Pos cover = 10 Neg cover = 0]
esv(A) :-
   qrs(A,B,normal),
   has_wave(A,B,C,p_I,normal,D,qrs_I,normal),
   has_wave(A,D,E,p_I,normal,F,qrs_I,normal),
   has_wave(A,F,G,qrs_I,abnormal,H,p_I,normal),
   has_wave(A,H,I,qrs_I,normal,J,p_I,normal).

Accuracy = 1.0
[Training set summary] [[10,0,0,40]]
[time taken] [1223.61]
[total clauses constructed] [48692]
```

Cette règle représente une succession $B, C, D, E, F, G, H, I, J$ d'ondes P et QRS normales avec un complexe QRS anormal (l'onde G) non précédé d'une onde P.

```
[Rule 1] [Pos cover = 7 Neg cover = 0]
bige(A) :-
   qrs(A,B,normal),
   has_wave(A,B,C,qrs_I,abnormal,D,p_I,normal),
   has_wave(A,D,E,qrs_I,normal,F,qrs_I,abnormal).

Accuracy = 1.0
[Training set summary] [[7,0,0,43]]
[time taken] [2.27]
[total clauses constructed] [410]
```

Cette règle représente une succession B, C, D, E, F d'ondes alternant un QRS normal éventuellement précédé (rien n'est indiqué pour le premier QRS B) d'une onde P et d'un QRS anormal (ondes C et F) non précédé d'une onde P.

```
[Rule 1] [Pos cover = 6 Neg cover = 0]
doublet(A) :-
   qrs(A,B,normal),
   has_wave(A,B,C,qrs_I,abnormal,D,qrs_I,abnormal),
   has_wave(A,C,D,qrs_I,abnormal,E,p_I,normal).

Accuracy = 1.0
```

```
[Training set summary] [[6,0,0,44]]
[time taken] [1.407]
[total clauses constructed] [509]
```

Cette règle représente une succession d'ondes B, C, D, E : un QRS normal suivi de deux QRS anormaux consécutifs (l'onde D est rappelée deux fois) suivi d'une onde P.

```
[Rule 1] [Pos cover = 7 Neg cover = 0]
tv(A) :-
   qrs(A,B,abnormal),
   has_wave(A,B,C,qrs_I,abnormal,D,qrs_I,abnormal).

Accuracy = 1.0
[Training set summary] [[7,0,0,43]]
[time taken] [0.67]
[total clauses constructed] [30]
```

Ici la *tv* est caractérisée par 3 QRS anormaux consécutifs (B, C et D).

```
[Rule 1] [Pos cover = 5 Neg cover = 0]
tsv(A) :-
   p(A,B,normal),
   has_wave(A,B,C,qrs_I,normal,D,p_I,normal),
   pp_1(B,D,court).

Accuracy = 1.0
[Training set summary] [[5,0,0,45]]
[time taken] [2.24]
[total clauses constructed] [1435]
```

La *tsv* est caractérisée par 3 ondes B, C et D : une onde $P1$ suivi d'un QRS normal suivi d'une autre onde $P2$ avec un intervalle $P1 - P2$ *court*.

```
[Rule 1] [Pos cover = 7 Neg cover = 0]
fa(A) :-
   qrs(A,B,normal),
   has_wave(A,B,C,qrs_I,normal,D,qrs_I,normal).

Accuracy = 1.0
[Training set summary] [[7,0,0,43]]
[time taken] [0.13]
[total clauses constructed] [16]
```

C.2.2.2 Voie V

Les apprentissages sur la voie V donnent des règles apprises par coeur avec les paramètres établis pour le voie I. Pour laisser plus de *liberté* à l'algorithme d'apprentissage, nous avons introduit la possibilité de couvrir des exemples négatifs.

```
[Rule 1] [Pos cover = 8 Neg cover = 6]
rs(A) :-
   qrs(A,B,normal),
  has_wave(A,B,C,qrs_V,normal,D,qrs_V,normal),
  has_wave(A,D,E,qrs_V,normal,F,qrs_V,normal),
  has_wave(A,F,G,qrs_V,normal,H,qrs_V,normal),
  rr_2(E,G,normal),
  has_wave(A,H,I,qrs_V,normal,J,qrs_V,normal),
  has_wave(A,J,K,qrs_V,normal,L,qrs_V,normal).

Accuracy = 0.88
[Training set summary] [[8,6,0,36]]
[time taken] [1086.417]
[total clauses constructed] [57972]
---------------------------------
[Rule 1] [Pos cover = 10 Neg cover = 0]
esv(A) :-
   qrs(A,B,normal),
   has_wave(A,B,C,qrs_V,normal,D,qrs_V,normal),
   has_wave(A,D,E,qrs_V,abnormal,F,qrs_V,normal),
   has_wave(A,E,F,qrs_V,normal,G,qrs_V,normal).

Accuracy = 1.0
[Training set summary] [[10,0,0,40]]
[time taken] [2.71]
[total clauses constructed] [933]
---------------------------------
[Rule 1] [Pos cover = 7 Neg cover = 0]
bige(A) :-
   qrs(A,B,abnormal),
  has_wave(A,B,C,qrs_V,normal,D,qrs_V,abnormal),
  has_wave(A,C,D,qrs_V,abnormal,E,qrs_V,normal).

Accuracy = 1.0
[Training set summary] [[7,0,0,43]]
[time taken] [0.891]
[total clauses constructed] [140]
---------------------------------
[Rule 1] [Pos cover = 6 Neg cover = 0]
```

```
doublet(A) :-
   qrs(A,B,normal),
   has_wave(A,B,C,qrs_V,abnormal,D,qrs_V,abnormal),
   has_wave(A,C,D,qrs_V,abnormal,E,qrs_V,normal).

Accuracy = 1.0
[Training set summary] [[6,0,0,44]]
[time taken] [1.054]
[total clauses constructed] [169]
----------------------------------
[Rule 1] [Pos cover = 7 Neg cover = 0]
tv(A) :-
   qrs(A,B,abnormal),
   has_wave(A,B,C,qrs_V,abnormal,D,qrs_V,abnormal).

Accuracy = 1.0
[Training set summary] [[7,0,0,43]]
[time taken] [0.926]
[total clauses constructed] [18]
----------------------------------
[Rule 1] [Pos cover = 5 Neg cover = 3]
tsv(A) :-
   qrs(A,B,normal),
  has_wave(A,B,C,qrs_V,normal,D,qrs_V,normal),
  rr_2(B,D,court).

Accuracy = 0.94
[Training set summary] [[5,3,0,42]]
[time taken] [3.885]
[total clauses constructed] [383]
----------------------------------
[Rule 1] [Pos cover = 6 Neg cover = 0]
fa(A) :-
   qrs(A,B,normal),
   has_wave(A,B,C,qrs_V,normal,D,qrs_V,normal),
   rythm(B,C,D,irregular),
   has_wave(A,C,D,qrs_V,normal,E,qrs_V,normal),
   has_wave(A,E,F,qrs_V,normal,G,qrs_V,normal),
   has_wave(A,G,H,qrs_V,normal,I,qrs_V,normal).

[Rule 2] [Pos cover = 1 Neg cover = 0]
fa(fa_2_I).

Accuracy = 1.0
```

```
[Training set summary] [[7,0,0,43]]
[time taken] [21476.021]
[total clauses constructed] [300001]
```

C.2.2.3 Voie de pression (ABP)

```
[Rule 1] [Pos cover = 2 Neg cover = 0]
rs(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,short),
   has_wave(A,D,E,diastole,F,systole,normal),
   has_wave(A,F,G,diastole,H,systole,normal),
   has_wave(A,H,I,diastole,J,systole,normal),
   has_wave(A,J,K,diastole,L,systole,normal),
   amp_ss(F,H,normal,nul), amp_ss(B,D,normal,nul).

[Rule 2] [Pos cover = 3 Neg cover = 0]
rs(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,short),
   has_wave(A,D,E,diastole,F,systole,short),
   has_wave(A,F,G,diastole,H,systole,short),
   has_wave(A,H,I,diastole,J,systole,short),
   ss_2(D,H,normal), amp_ss(F,H,normal,nul),
    ss_1(B,D,normal).

[Rule 3] [Pos cover = 3 Neg cover = 0]
rs(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,long),
   has_wave(A,D,E,diastole,F,systole,long),
   has_wave(A,F,G,diastole,H,systole,long),
   has_wave(A,H,I,diastole,J,systole,long),
   ss_1(B,D,normal).

[Rule 4] [Pos cover = 1 Neg cover = 0]
rs(rs_8_ABP).

Accuracy = 1.0
[Training set summary] [[8,0,0,42]]
[time taken] [41756.82]
[total clauses constructed] [925422]

-------------------------------------
```

```
[Rule 1] [Pos cover = 4 Neg cover = 1]
esv(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,normal),
   has_wave(A,D,E,diastole,F,systole,normal),
   has_wave(A,F,G,diastole,H,systole,short),
   has_wave(A,H,I,diastole,J,systole,normal),
   amp_ss(D,F,long,pos), ss_1(B,D,normal).

[Rule 2] [Pos cover = 2 Neg cover = 0]
esv(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,short),
   has_wave(A,D,E,diastole,F,systole,normal),
   has_wave(A,F,G,diastole,H,systole,normal),
   has_wave(A,H,I,diastole,J,systole,normal),
   has_wave(A,J,K,diastole,L,systole,normal),
   has_wave(A,L,M,diastole,N,systole,normal),
   has_wave(A,N,O,diastole,P,systole,normal),
   amp_ss(N,P,normal,nul), ss_1(B,D,normal).

[Rule 3] [Pos cover = 2 Neg cover = 0]
esv(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,normal),
   has_wave(A,D,E,diastole,F,systole,short),
   has_wave(A,F,G,diastole,H,systole,normal),
   amp_ss(D,F,long,neg), amp_ss(B,D,long,neg).

[Rule 4] [Pos cover = 5 Neg cover = 0]
esv(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,short),
   has_wave(A,D,E,diastole,F,systole,normal),
   has_wave(A,F,G,diastole,H,systole,normal),
   has_wave(A,H,I,diastole,J,systole,short),
   has_wave(A,J,K,diastole,L,systole,normal),
   amp_ss(H,J,long,neg), ss_1(D,F,normal).

Accuracy = 0.98
[Training set summary] [[10,1,0,39]]
[time taken] [15005.059]
[total clauses constructed] [372618]
---------------------------------
```

```
Rule 1] [Pos cover = 2 Neg cover = 0]
bige(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,normal),
   has_wave(A,D,E,diastole,F,systole,short),
   has_wave(A,F,G,diastole,H,systole,normal),
   has_wave(A,H,I,diastole,J,systole,short),
   ss_1(B,D,long).

[Rule 2] [Pos cover = 2 Neg cover = 0]
bige(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,normal),
   has_wave(A,D,E,diastole,F,systole,normal),
   has_wave(A,F,G,diastole,H,systole,normal),
   has_wave(A,H,I,diastole,J,systole,short),
   ss_1(H,J,normal), ss_2(D,H,normal),
   amp_ss(D,F,long,neg),
   amp_ss(B,D,long,pos).

[Rule 3] [Pos cover = 5 Neg cover = 0]
bige(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,short),
   has_wave(A,D,E,diastole,F,systole,normal),
   ss_1(D,F,long), amp_ss(D,F,long,pos).

Accuracy = 1.0
[Training set summary] [[7,0,0,43]]
[time taken] [15119.52]
[total clauses constructed] [308854]
---------------------------------
Rule 1] [Pos cover = 3 Neg cover = 0]
doublet(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,short),
   has_wave(A,D,E,diastole,F,systole,short),
   has_wave(A,F,G,diastole,H,systole,normal),
   has_wave(A,H,I,diastole,J,systole,normal),
   amp_ss(D,F,long,neg), amp_ss(B,D,long,neg).

[Rule 2] [Pos cover = 1 Neg cover = 0]
doublet(doublet_2_ABP).
```

```
[Rule 3] [Pos cover = 2 Neg cover = 0]
doublet(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,normal),
   has_wave(A,D,E,diastole,F,systole,normal),
   has_wave(A,F,G,diastole,H,systole,normal),
   has_wave(A,H,I,diastole,J,systole,normal),
   has_wave(A,J,K,diastole,L,systole,normal),
   amp_ss(H,J,long,pos),
   amp_ss(F,H,long,pos),
   amp_ss(B,D,normal,nul).
Accuracy = 1.0
[Training set summary] [[6,0,0,44]]
[time taken] [51631.84]
[total clauses constructed] [621844]
---------------------------------
[Rule 1] [Pos cover = 3 Neg cover = 0]
tv(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,short),
   has_wave(A,D,E,diastole,F,systole,short),
   has_wave(A,F,G,diastole,H,systole,short),
   ss_2(D,H,court), ss_1(B,D,normal).

[Rule 2] [Pos cover = 2 Neg cover = 0]
tv(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,normal),
   has_wave(A,D,E,diastole,F,systole,short),
   ss_1(D,F,long), ss_2(B,F,normal).

[Rule 3] [Pos cover = 2 Neg cover = 0]
tv(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,short),
   has_wave(A,D,E,diastole,F,systole,short),
   ss_1(D,F,court), ss_2(B,F,normal),
   amp_ss(D,F,normal,nul).

[Training set summary] [[7,0,0,43]]
[time taken] [48.947]
[total clauses constructed] [9906]
---------------------------------
[Rule 1] [Pos cover = 4 Neg cover = 1]
```

```
tsv(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,normal),
   ss_1(B,D,court), amp_ss(B,D,normal,nul).

[Rule 2] [Pos cover = 1 Neg cover = 0]
tsv(tsv_7_ABP).

Accuracy = 0.98
[Training set summary] [[5,1,0,44]]
[time taken] [12122.96]
[total clauses constructed] [333625]
---------------------------------
[theory]

[Rule 1] [Pos cover = 1 Neg cover = 0]
fa(fa_1_ABP).

[Rule 2] [Pos cover = 1 Neg cover = 0]
fa(fa_2_ABP).

[Rule 3] [Pos cover = 2 Neg cover = 0]
fa(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,short),
   ss_1(B,D,long), amp_ss(B,D,normal,nul).

[Rule 4] [Pos cover = 3 Neg cover = 0]
fa(A) :-
   systole(A,B),
   has_wave(A,B,C,diastole,D,systole,long),
   has_wave(A,D,E,diastole,F,systole,short),
   has_wave(A,F,G,diastole,H,systole,long).

Accuracy = 1.0
[Training set summary] [[7,0,0,43]]
[time taken] [42668.76]
[total clauses constructed] [900292]
```

C.3 Apprentissages multisources naïfs avec ICL

C.3.1 Codage de l'ECG et de la pression ECG-ABP

```
cycle_complet_I(P, FormP,R, FormQRS, Dias, Symbd, Sys, Symbs):-
        p_wave_I(P, _, FormP, _PrecP, _, _, _, _, _, _),
        qrs_I(R, TpsR, FormQRS, P, _, _, _, _, _, _),
        suci(R,P),
        diastole(Dias, _, _, PrecS,Val1, _, _, _,_,_,_),
        suci(Dias,R),
        qual_amp(Val1,Symbd),
        systole(Sys, TpsSys, _, Dias,Val, _,PrecS,_, _,_ , _),
        qual_amp(Val,Symbs),
        suci(Sys,Dias).

cycle_simple_I(P, D1, R, D):-
        p_wave_I(P, _, D1, _Prec, _, _, _, _, _, _),
        qrs_I(R, TpsR, D, P, _, _, _, _, _, _),
        suci(R,P).


cycle_simple_ABP(Dias,Symb1,Sys,Symb2):-
        diastole(Dias, Symb1),
        systole(Sys, Symb2, Dias),
        suci(Sys,Dias).

%ondes simples
qrs(R, D):-qrs_I(R, _TpsR,D,_, _, _, _, _, _, _).

p_wav(P, D):- p_wave_I(P, _,D,_, _, _, _, _, _, _).


diastole(Dias, Symb) :-
        diastole(Dias, _, _,_,Val, _, _, _,_,_,_),
        qual_amp(Val,Symb).

systole(Sys, Symb) :-
        systole(Sys, _, _, _,Val, _, _,_ , _,_,_),
        qual_amp(Val,Symb).

%codage des successions
suc(A,B):- qrs_I(A, _, _, B, _, _, _, _, _, _),!.
suc(A,B):- p_wave_I(A, _, _, B, _, _, _, _, _, _),!.
suc(A,B):- systole(A, _, _, B,_, _, _,_ , _,_,_),!.
suc(A,B):- diastole(A, _, _, B,_, _, _, _,_,_,_),!.

wave(A,T):-qrs_I(A, T, _, _, _, _, _, _, _, _).
wave(A,T):-p_wave_I(A,T, _, _, _, _, _, _, _, _).
```

```
wave(A,T):-diastole(A, T, _, _,_, _, _, _,_,_,_).
wave(A,T):-systole(A, T, _, _,_, _, _,_ , _,_,_).


append([], List2, List2).
append([First | Rest], List2, [First | Intermediate]) :-
        append(Rest, List2, Intermediate).

%le prédicat seq est introduit dans le .kb pour
%gagner du temps lors des tests de couverture
suci(B,A):-
        seq(L),
        wave(A,TA),wave(B,TB),
        append(_X,[wave(A,TA),wave(B,TB)|_Y],L).
...
```

Le codage des intervalles et leur seuil respéctif est le même qu'en monosource.

C.3.2 Langage de biais naïf (I-ABP)

```
dlab_template('
 false   <--
 len-len:[
    1-1:[len-len:[
    1-1:[
    len-len:[
    qrs(R0, w2_feature),
    equal(R0,Suc0)
    ],
    len-len:[
    p_wav(P0, w2_feature),
    equal(P0, Suc0)
    ],
    len-len:[
    cycle_simple_I(P0, w2_feature, R0, w2_feature),
    equal(R0,Suc0),
    0-1:[pr1(P0, R0, r_feature)]
        ],
    len-len:[
    cycle_simple_ABP(Dias0, w_feature, Sys0, w_feature),
    equal(Sys0,Suc0),
    0-1:[ds1(Dias0, Sys0,  r_feature)]
    ],
    len-len:[
```

```
cycle_complet_I(P0, w2_feature,R0,w2_feature,Dias0,_,Sys0,w_feature),
equal(Sys0,Suc0),
0-1:[pr1(P0,R0, r_feature)],
0-1:[ds1(Dias0,Sys0,  r_feature)]
]
],

0-1:[len-len:[
1-1:[
len-len:[
cycle_complet_I(P1, w2_feature,R1,w2_feature,Dias1,w_feature,Sys1,w_feature),
1-1:[suci(P1,Suc0),suc(P1,Suc0)],
equal(Sys1,Suc1),
0-1:[pr1(P1,R1, r_feature)],
0-1:[rr1(R0, R1, r_feature)],
0-1:[amp_ss(Sys0, Sys1, 1-1:[neg,pos,nul],  r_feature)],
0-1:[amp_dd(Dias0, Dias1, 1-1:[neg,pos,nul],  r_feature)],
0-1:[ds1(Dias1,Sys1,  r_feature)],
0-1:[ss1(Sys0, Sys1, r_feature)],
0-1:[dd1(Dias0, Dias1, r_feature)],
0-1:[sd1(Sys0, Dias1, r_feature)]
],
len-len:[
cycle_simple_ABP(Dias1, w_feature, Sys1, w_feature),
1-1:[suci(Dias1,Suc0),suc(Dias1,Suc0)],
equal(Sys1,Suc1),
0-1:[ds1(Dias1,Sys1,  r_feature)],
0-1:[ss1(Sys0, Sys1, r_feature)],
0-1:[dd1(Dias0, Dias1, r_feature)],
0-1:[sd1(Sys0, Dias1, r_feature)]
],
len-len:[
cycle_simple_I(P1, w2_feature, R1, w2_feature),
1-1:[suci(P1,Suc0),suc(P1,Suc0)],
equal(R1,Suc1),
0-1:[pr1(P1,R1, r_feature)],
0-1:[rr1(R0, R1, r_feature)]
],
len-len:[
qrs(R1, w2_feature),
1-1:[suci(R1,Suc0),suc(R1,Suc0)],
equal(R1,Suc1),
0-1:[rr1(R0, R1, r_feature)]
],
```

```
    len-len:[
    p_wav(P1, w2_feature),
    1-1:[suci(P1,Suc0),suc(P1,Suc0)],
    equal(P1,Suc1)
    ]
    ],
%...5 cycles additionnels sur le modèle du deuxième cycle
  ]]
  ]]
]
   ').

dlab_variable(w2_feature, 1-1, [normal, abnormal]).
dlab_variable(r_feature, 1-1, [short, normal, long]).
dlab_variable(w_feature, 1-1, [short, normal, high]).
dlab_variable(amp_featureS, 1-1, [pos, neg,nul]).
```

Les biais pour les apprentissages `V-ABP sans diastole`, `V sans forme du QRS-ABP sans diastole` sont faits sur ce modèle. Les prédicats ont simplement été simplifiés. Le biais `P-QRS` est le même que celui de la voie I en monosource sans prendre en compte la forme du QRS.

C.3.3 Résultats

C.3.3.1 I-ABP

```
class(rs,[7,0,0,0,0,0,0],[1,7,10,6,7,5,7]) :-
cycle_complet_I(P0,normal,R0,normal,Dias0,_,Sys0,normal),
equal(Sys0,Suc0), pr1(P0,R0,long),
cycle_complet_I(P1,normal,R1,normal,Dias1,normal,Sys1,normal),
suci(P1,Suc0), equal(Sys1,Suc1), pr1(P1,R1,long),
amp_ss(Sys0,Sys1,nul,normal), amp_dd(Dias0,Dias1,nul,normal),
cycle_complet_I(P2,normal,R2,normal,Dias2,normal,Sys2,normal),
suci(P2,Suc1), equal(Sys2,Suc2), p_wav(P3,normal), suci(P3,Suc2),
equal(P3,Suc3).

class(bige,[0,5,0,0,0,0,0],[8,2,10,6,7,5,7]) :-
cycle_simple_I(P0,normal,R0,normal), equal(R0,Suc0), qrs(R1,abnormal),
suc(R1,Suc0), equal(R1,Suc1),
cycle_complet_I(P2,normal,R2,normal,Dias2,short,Sys2,normal),
suc(P2,Suc1), equal(Sys2,Suc2), sd1(Sys1,Dias2,long).

class(bige,[0,3,0,0,0,0,0],[8,4,10,6,7,5,7]) :-
cycle_complet_I(P0,normal,R0,normal,Dias0,_,Sys0,normal),
equal(Sys0,Suc0), ds1(Dias0,Sys0,long), qrs(R1,abnormal),
```

```
suci(R1,Suc0), equal(R1,Suc1), rr1(R0,R1,normal),
cycle_complet_I(P2,normal,R2,normal,Dias2,normal,Sys2,normal),
suc(P2,Suc1), equal(Sys2,Suc2), ds1(Dias2,Sys2,normal).

class(esv,[0,1,9,0,0,0,0],[8,6,1,6,7,5,7]) :-
cycle_simple_I(P0,normal,R0,normal), equal(R0,Suc0),
qrs(R1,abnormal), suc(R1,Suc0), equal(R1,Suc1),
cycle_simple_I(P2,normal,R2,normal), suc(P2,Suc1), equal(R2,Suc2),
rr1(R1,R2,normal),
cycle_complet_I(P3,normal,R3,normal,Dias3,normal,Sys3,normal),
suc(P3,Suc2), equal(Sys3,Suc3), sd1(Sys2,Dias3,normal).

class(doublet,[0,0,0,5,0,0,0],[8,7,10,1,7,5,7]) :-
qrs(R0,abnormal), equal(R0,Suc0),
qrs(R1,abnormal), suc(R1,Suc0), equal(R1,Suc1),
cycle_simple_I(P2,normal,R2,normal), suc(P2,Suc1), equal(R2,Suc2),
cycle_complet_I(P3,normal,R3,normal,Dias3,normal,Sys3,normal),
suc(P3,Suc2), equal(Sys3,Suc3),
cycle_complet_I(P4,normal,R4,normal,Dias4,normal,Sys4,normal),
suci(P4,Suc3), equal(Sys4,Suc4), ds1(Dias4,Sys4,normal).

class(tv,[0,0,0,0,7,0,0],[8,7,10,6,0,5,7]) :-
qrs(R0,abnormal),
equal(R0,Suc0), qrs(R1,abnormal),
suc(R1,Suc0), equal(R1,Suc1), qrs(R2,abnormal),
suc(R2,Suc1), equal(R2,Suc2), rythm(R0,R1,R2,regular).

class(tsv,[0,0,0,0,0,5,0],[8,7,10,6,7,0,7]) :-
p_wav(P0,normal), equal(P0,Suc0), qrs(R1,normal),
suc(R1,Suc0), equal(R1,Suc1), rr1(R0,R1,short).

class(fa,[0,0,0,0,0,0,7],[8,7,10,6,7,5,0]) :-
qrs(R0,normal),
equal(R0,Suc0), qrs(R1,normal),
suc(R1,Suc0), equal(R1,Suc1).
```

C.3.3.2 V-ABP sans diastole

```
class(rs,[8,0,0,0,0,0,0],[0,7,10,6,7,5,7]) :-
cycle_qrs_sys(R0,normal,Sys0), equal(Sys0,Suc0),
cycle_qrs_sys(R1,normal,Sys1), equal(Sys1,Suc1),
suci(R1,Suc0), amp_ss(Sys0,Sys1,nul,normal), ss1(Sys0,Sys1,normal),
systole(Sys2), equal(Sys2,Suc2), suc(Sys2,Suc1),
amp_ss(Sys1,Sys2,nul,normal),
```

```
cycle_qrs_sys(R3,normal,Sys3), equal(Sys3,Suc3),
suci(R3,Suc2), amp_ss(Sys2,Sys3,nul,normal),
cycle_qrs_sys(R4,normal,Sys4), equal(Sys4,Suc4),
suci(R4,Suc3), rr1(R3,R4,normal), amp_ss(Sys3,Sys4,nul,normal),
systole(Sys5), equal(Sys5,Suc5), suc(Sys5,Suc4),
cycle_qrs_sys(R6,normal,Sys6), equal(Sys6,Suc6),
suci(R6,Suc5).
cpu(132530)

class(bige,[0,7,0,0,0,0,0],[8,0,10,6,7,5,7]) :-
qrs(R0,normal),
equal(R0,Suc0), qrs(R1,normal),
equal(R1,Suc1), suc(R1,Suc0), qrs(R2,abnormal),
equal(R2,Suc2), suc(R2,Suc1),
rythm(R0,R1,R2,regular), cycle_qrs_sys(R3,normal,Sys3),
equal(Sys3,Suc3), suc(R3,Suc2), qrs(R4,abnormal),
equal(R4,Suc4), suci(R4,Suc3).
cpu(6286.29)

class(esv,[0,0,8,0,0,0,0],[8,7,2,6,7,5,7]) :-
qrs(R0,normal),
equal(R0,Suc0), qrs(R1,normal),
equal(R1,Suc1), suc(R1,Suc0), qrs(R2,abnormal),
equal(R2,Suc2), suc(R2,Suc1),
rythm(R0,R1,R2,regular), qrs(R3,normal),
equal(R3,Suc3), suc(R3,Suc2),
rythm(R1,R2,R3,regular), cycle_qrs_sys(R4,normal,Sys4),
equal(Sys4,Suc4), suc(R4,Suc3).

class(esv,[0,0,3,0,0,0,0],[8,7,7,6,7,5,7]) :-
qrs(R0,normal),
equal(R0,Suc0), qrs(R1,abnormal),
equal(R1,Suc1), suc(R1,Suc0),
cycle_qrs_sys(R2,normal,Sys2),
equal(Sys2,Suc2), suci(R2,Suc1),
cycle_qrs_sys(R3,normal,Sys3),
equal(Sys3,Suc3), suci(R3,Suc2), ss1(Sys2,Sys3,normal).
cpu(26176.2)

class(doublet,[0,0,0,5,0,0,0],[8,7,10,1,7,5,7]) :-
cycle_qrs_sys(R0,normal,Sys0),
equal(Sys0,Suc0), qrs(R1,abnormal),
equal(R1,Suc1), suci(R1,Suc0), rr1(R0,R1,normal),
cycle_qrs_sys(R2,abnormal,Sys2),
```

```
equal(Sys2,Suc2), suc(R2,Suc1), systole(Sys3),
equal(Sys3,Suc3), suc(Sys3,Suc2), cycle_qrs_sys(R4,normal,Sys4),
equal(Sys4,Suc4), suci(R4,Suc3).
cpu(11897.4)

class(tv,[0,0,0,0,7,0,0],[8,7,10,6,0,5,7]) :-
qrs(R0,abnormal),
equal(R0,Suc0), qrs(R1,abnormal),
equal(R1,Suc1), suc(R1,Suc0), cycle_qrs_sys(R2,abnormal,Sys2),
equal(Sys2,Suc2), suc(R2,Suc1).
cpu(6065.09)

class(tsv,[0,0,0,0,0,5,0],[8,7,10,6,7,0,7]) :-
qrs(R0,normal),
equal(R0,Suc0), cycle_qrs_sys(R1,normal,Sys1),
equal(Sys1,Suc1), suc(R1,Suc0), rr1(R0,R1,short),
amp_ss(Sys0,Sys1,nul,normal), cycle_qrs_sys(R2,normal,Sys2),
equal(Sys2,Suc2), suci(R2,Suc1), amp_ss(Sys1,Sys2,neg,long).
cpu(97283.9)

class(fa,[0,0,0,0,0,0,6],[8,7,10,6,7,5,1]) :-
qrs(R0,normal),
equal(R0,Suc0), qrs(R1,normal),
equal(R1,Suc1), suc(R1,Suc0), qrs(R2,normal),
equal(R2,Suc2), suc(R2,Suc1), rythm(R0,R1,R2,irregular).
cpu(351231)
```

C.3.3.3 V sans forme du QRS-ABP sans diastole

```
class(rs,[8,0,0,0,0,0,0],[0,7,10,6,7,5,7]) :-
cycle_qrs_sys(R0,Sys0),
equal(Sys0,Suc0), cycle_qrs_sys(R1,Sys1),
equal(Sys1,Suc1), suci(R1,Suc0), amp_ss(Sys0,Sys1,nul,normal),
ss1(Sys0,Sys1,normal),
cycle_qrs_sys(R2,Sys2),
equal(Sys2,Suc2), suci(R2,Suc1), rr1(R1,R2,normal),
amp_ss(Sys1,Sys2,nul,normal), cycle_qrs_sys(R3,Sys3),
equal(Sys3,Suc3), suci(R3,Suc2), amp_ss(Sys2,Sys3,nul,normal),
cycle_qrs_sys(R4,Sys4), equal(Sys4,Suc4), suci(R4,Suc3),
amp_ss(Sys3,Sys4,nul,normal),
qrs(R5), equal(R5,Suc5), suci(R5,Suc4), r
ythm(R3,R4,R5,regular), cycle_qrs_sys(R6,Sys6),
equal(Sys6,Suc6), suc(R6,Suc5).
```

```
class(bige,[0,6,0,0,0,0,0],[8,1,10,6,7,5,7]) :-
qrs(R0), equal(R0,Suc0), systole(Sys1),
equal(Sys1,Suc1), suci(Sys1,Suc0), amp_ss(Sys0,Sys1,pos,short),
ss1(Sys0,Sys1,long), systole(Sys2),
equal(Sys2,Suc2), suc(Sys2,Suc1), systole(Sys3),
equal(Sys3,Suc3), suc(Sys3,Suc2), ss1(Sys2,Sys3,long).

class(esv,[0,0,6,0,0,0,0],[8,7,4,6,7,5,7]) :-
systole(Sys0), equal(Sys0,Suc0),
cycle_qrs_sys(R1,Sys1), equal(Sys1,Suc1), suci(R1,Suc0),
amp_ss(Sys0,Sys1,neg,short), ss1(Sys0,Sys1,normal),
systole(Sys2), equal(Sys2,Suc2), suc(Sys2,Suc1),
amp_ss(Sys1,Sys2,nul,normal),
cycle_qrs_sys(R3,Sys3), equal(Sys3,Suc3), suci(R3,Suc2),
ss1(Sys2,Sys3,normal),
systole(Sys4), equal(Sys4,Suc4), suc(Sys4,Suc3),
systole(Sys5), equal(Sys5,Suc5), suc(Sys5,Suc4),
amp_ss(Sys4,Sys5,pos,short).

class(esv,[0,0,2,0,0,0,0],[8,7,8,6,7,5,7]) :-
qrs(R0), equal(R0,Suc0),
systole(Sys1), equal(Sys1,Suc1),
suci(Sys1,Suc0), amp_ss(Sys0,Sys1,neg,long),
ss1(Sys0,Sys1,normal), systole(Sys2),
equal(Sys2,Suc2), suc(Sys2,Suc1),
amp_ss(Sys1,Sys2,nul,normal),
cycle_qrs_sys(R3,Sys3), equal(Sys3,Suc3), suci(R3,Suc2),
ss1(Sys2,Sys3,normal),
cycle_qrs_sys(R4,Sys4), equal(Sys4,Suc4), suci(R4,Suc3),
ss1(Sys3,Sys4,normal),
systole(Sys5), equal(Sys5,Suc5), suc(Sys5,Suc4),
amp_ss(Sys4,Sys5,neg,short),
qrs(R6), equal(R6,Suc6), suci(R6,Suc5),
rythm(R4,R5,R6,regular).

class(esv,[0,1,7,4,3,0,0],[8,6,3,2,4,5,7]) :-
systole(Sys0), equal(Sys0,Suc0),
cycle_qrs_sys(R1,Sys1), equal(Sys1,Suc1), suci(R1,Suc0),
amp_ss(Sys0,Sys1,neg,short), ss1(Sys0,Sys1,normal),
systole(Sys2), equal(Sys2,Suc2), suc(Sys2,Suc1),
amp_ss(Sys1,Sys2,pos,normal), ss1(Sys1,Sys2,normal),
qrs(R3), equal(R3,Suc3), suci(R3,Suc2),
rythm(R1,R2,R3,regular),
systole(Sys4), equal(Sys4,Suc4), suci(Sys4,Suc3),
```

```
amp_ss(Sys3,Sys4,nul,normal),
cycle_qrs_sys(R5,Sys5), equal(Sys5,Suc5), suci(R5,Suc4),
ss1(Sys4,Sys5,normal).

class(doublet,[0,0,1,4,0,0,0],[8,7,9,2,7,5,7]) :-
qrs(R0),
equal(R0,Suc0), qrs(R1),
equal(R1,Suc1), suc(R1,Suc0), qrs(R2),
equal(R2,Suc2), suci(R2,Suc1), rythm(R0,R1,R2,irregular),
cycle_qrs_sys(R3,Sys3), equal(Sys3,Suc3), suc(R3,Suc2),
cycle_qrs_sys(R4,Sys4), equal(Sys4,Suc4), suci(R4,Suc3),
cycle_qrs_sys(R5,Sys5), equal(Sys5,Suc5), suci(R5,Suc4),
ss1(Sys4,Sys5,normal).

class(doublet,[0,0,0,3,0,0,0],[8,7,10,3,7,5,7]) :-
cycle_qrs_sys(R0,Sys0), equal(Sys0,Suc0),
cycle_qrs_sys(R1,Sys1), equal(Sys1,Suc1), suci(R1,Suc0),
rr1(R0,R1,normal), amp_ss(Sys0,Sys1,neg,short),
ss1(Sys0,Sys1,short),
systole(Sys2), equal(Sys2,Suc2), suc(Sys2,Suc1),
ss1(Sys1,Sys2,normal),
cycle_qrs_sys(R3,Sys3), equal(Sys3,Suc3), suci(R3,Suc2),
amp_ss(Sys2,Sys3,pos,long), systole(Sys4),
equal(Sys4,Suc4), suc(Sys4,Suc3), qrs(R5),
equal(R5,Suc5), suci(R5,Suc4),
rythm(R3,R4,R5,regular), cycle_qrs_sys(R6,Sys6),
equal(Sys6,Suc6), suc(R6,Suc5).

class(tv,[0,0,0,0,4,0,0],[8,7,10,6,3,5,7]) :-
qrs(R0), equal(R0,Suc0),
qrs(R1), equal(R1,Suc1), suc(R1,Suc0),
rr1(R0,R1,short), qrs(R2),
equal(R2,Suc2), suc(R2,Suc1),
rythm(R0,R1,R2,irregular), qrs(R3),
equal(R3,Suc3), suc(R3,Suc2),
rythm(R1,R2,R3,regular),
cycle_qrs_sys(R4,Sys4), equal(Sys4,Suc4), suc(R4,Suc3),
rr1(R3,R4,normal), amp_ss(Sys3,Sys4,pos,long).

class(tv,[0,0,0,2,3,0,0],[8,7,10,4,4,5,7]) :-
cycle_qrs_sys(R0,Sys0), equal(Sys0,Suc0),
cycle_qrs_sys(R1,Sys1), equal(Sys1,Suc1),
suci(R1,Suc0), amp_ss(Sys0,Sys1,pos,short),
ss1(Sys0,Sys1,short), systole(Sys2),
```

```
equal(Sys2,Suc2), suc(Sys2,Suc1),
ss1(Sys1,Sys2,normal).

class(tsv,[0,0,0,0,0,5,0],[8,7,10,6,7,0,7]) :-
systole(Sys0), equal(Sys0,Suc0),
cycle_qrs_sys(R1,Sys1), equal(Sys1,Suc1),
suci(R1,Suc0), rr1(R0,R1,short),
amp_ss(Sys0,Sys1,nul,normal),
systole(Sys2), equal(Sys2,Suc2),
suc(Sys2,Suc1), amp_ss(Sys1,Sys2,neg,long).

class(fa,[1,0,0,0,0,0,5],[7,7,10,6,7,5,2]) :-
systole(Sys0), equal(Sys0,Suc0),
cycle_qrs_sys(R1,Sys1), equal(Sys1,Suc1), suci(R1,Suc0),
amp_ss(Sys0,Sys1,nul,normal),
ss1(Sys0,Sys1,long), cycle_qrs_sys(R2,Sys2),
equal(Sys2,Suc2), suci(R2,Suc1).

class(fa,[0,0,0,0,0,0,3],[8,7,10,6,7,5,4]) :-
systole(Sys0), equal(Sys0,Suc0),
cycle_qrs_sys(R1,Sys1), equal(Sys1,Suc1),
suci(R1,Suc0), qrs(R2),
equal(R2,Suc2), suci(R2,Suc1),
rythm(R0,R1,R2,irregular),
rr2(R0,R2,short), rr1(R1,R2,normal).
```

C.3.3.4 P-QRS

```
class(rs,[8,10,0,6,4,0,0],[0,0,7,0,3,5,7]) :-
cycle_I(P0,normal,R0),
cycle_I(P1,normal,R1), suc(P1,R0),
rr1(R0,R1,normal),
cycle_I(P2,normal,R2), suc(P2,R1),
cycle_I(P3,normal,R3), suc(P3,R2),
cycle_I(P4,normal,R4), suc(P4,R3),
cycle_I(P5,normal,R5), suc(P5,R4),
cycle_I(P6,normal,R6), suc(P6,R5).

class(esv,[0,10,0,0,0,0,0],[8,0,7,6,7,5,7]) :-
cycle_I(P0,normal,R0),
cycle_I(P1,normal,R1), suc(P1,R0),
cycle_I(P2,normal,R2), suc(P2,R1),
 qrs(R3), suc(R3,R2),
cycle_I(P4,normal,R4), suc(P4,R3),
```

```
cycle_I(P5,normal,R5), suc(P5,R4).

class(bige,[0,0,7,0,0,0,0],[8,10,0,6,7,5,7]) :-
cycle_I(P0,normal,R0), qrs(R1), suc(R1,R0),
cycle_I(P2,normal,R2), suc(P2,R1),
qrs(R3), suc(R3,R2).

class(doublet,[0,0,0,6,0,0,0],[8,10,7,0,7,5,7]) :-
cycle_I(P0,normal,R0), qrs(R1),
suc(R1,R0), qrs(R2),
suc(R2,R1), cycle_I(P3,normal,R3), suc(P3,R2).

class(tv,[0,0,0,0,7,0,0],[8,10,7,6,0,5,7]) :-
cycle_I(P0,normal,R0), qrs(R1),
suc(R1,R0), qrs(R2),
suc(R2,R1), qrs(R3),
suc(R3,R2).

class(tsv,[0,0,0,0,0,5,0],[8,10,7,6,7,0,7]) :-
cycle_I(P0,normal,R0), p_wav(P1,normal), suc(P1,R0),
pp1(P0,P1,short).

class(fa,[0,0,0,0,0,0,6],[8,10,7,6,7,5,1]) :-
qrs(R0), qrs(R1), suc(R1,R0),
qrs(R2), suc(R2,R1),
qrs(R3), suc(R3,R2),
rythm(R1,R2,R3,irregular),
qrs(R4), suc(R4,R3),
qrs(R5), suc(R5,R4),
qrs(R6), suc(R6,R5).
```

C.4 Apprentissages multisources biaisés avec ICL

L'apprentissage multisource nécéssite un biais différent (construit automatiquement) pour chacune des classes, on ne peut donc pas utiliser la méthode d'apprentissage multiclasse standard d'ICL puisque celle ci nécessite un biais global pour toutes les classes.

Le fait d'effectuer une session d'apprentissage par classe (même si cela correspond au fonctionnement d'ICL) ne nous permet pas de disposer de statistiques globales pour chaque règle. Les statistiques présentées en section 3.2.2.5 sont remplacées par des statistiques locales et totales présentées ci-dessous.

local(A,B,C,D) :
A : nb d'exemples positifs couverts par la règle qui n'étaient pas couvert pas les règles

précédentes
B : nb d'exemples positifs qu'il reste à couvrir pour cette classe
C : nb d'exemples négatifs couverts par la règle
D : nb d'exemples négatifs non couverts par la règle

total(A,B,C,D) :
A : nb d'exemples positifs couverts par la règle
B : nb d'exemples positifs non couverts par cette règle
C : nb d'exemples négatifs couverts par la règle
D : nb d'exemples négatifs non couverts par la règle

C.4.1 I-ABP

```
rule(rs):-
    cycle_ABP(Dias0, _, Sys0,normal), suc(Sys0, Dias0),
    cycle_ABP(Dias1,normal, Sys1,normal), amp_ss(Sys0, Sys1,nul,normal),
    ss1(Sys0, Sys1,normal), ds1(Dias1, Sys1,normal), suc(Dias1, Sys0),
    suc(Sys1, Dias1),
    cycle_ABP(Dias2,normal, Sys2,normal), amp_ss(Sys1, Sys2,nul,normal),
    ds1(Dias2, Sys2,normal), suc(Dias2, Sys1),
    suc(Sys2, Dias2),
    cycle_ABP(Dias3,normal, Sys3,normal), suc(Dias3, Sys2),
    suc(Sys3, Dias3),
    cycle_ABP(Dias4,normal, Sys4,normal), amp_ss(Sys3, Sys4,nul,normal),
    suc(Dias4, Sys3), suc(Sys4, Dias4).
Nb de noeuds visités : 415, cpu(470.61), local((8,0,0,42)), total((8,0,0,42))


rule(bige):-
    cycle_I(P0,normal, R0,normal), suc(R0, P0),
    qrs(R1,abnormal), suc(R1, R0),
    cycle_I(P2,normal, R2,normal), suc(P2, R1),
    suc(R2, P2),
    qrs(R3,abnormal), suc(R3, R2).
Nb de noeuds visités : 77, cpu(23.4), local((7,0,0,43)), total((7,0,0,43))


rule(esv):-
    cycle_I(P0,normal, R0,normal), suc(R0, P0),
    cycle_I(P1,normal, R1,normal), suc(P1, R0),
    suc(R1, P1),
    qrs(R2,abnormal), suc(R2, R1),
    cycle_I(P3,normal, R3,normal), rr1(R2, R3,normal), suc(P3, R2),
```

```
    suc(R3, P3),
    cycle_I(P4,normal, R4,normal), suc(P4, R3),
    suc(R4, P4).
 cpu(498.47), local((9,1,0,40)), total((9,1,0,40))
rule(esv):-
    cycle_ABP(Dias0, _, Sys0,short), suc(Sys0, Dias0),
    cycle_ABP(Dias1,normal, Sys1,normal), amp_ss(Sys0, Sys1,pos,long),
    sd1(Sys0, Dias1,normal), suc(Dias1, Sys0),
    suc(Sys1, Dias1),
    cycle_ABP(Dias2,normal, Sys2,normal), sd1(Sys1, Dias2,normal),
    suc(Dias2, Sys1),
    suc(Sys2, Dias2),
    cycle_ABP(Dias3,normal, Sys3,normal), sd1(Sys2, Dias3,normal),
    suc(Dias3, Sys2),
    suc(Sys3, Dias3),
    cycle_I(P0,normal, R0,normal), suci(P0, Sys3),
    suc(R0, P0).
Nb de noeuds visités : 2020, cpu(951.77), local((1,0,0,40)), total((5,5,0,40))


rule(doublet):-
    cycle_I(P0,normal, R0,normal), suc(R0, P0),
    qrs(R1,abnormal), suc(R1, R0),
    qrs(R2,abnormal), suc(R2, R1),
    cycle_I(P3,normal, R3,normal), suc(P3, R2),
    suc(R3, P3).
Nb de noeuds visités : 346, cpu(104.94), local((6,0,0,44)), total((6,0,0,44))


rule(tsv):-
    cycle_I(P0,normal, R0,normal), suc(R0, P0),
    cycle_I(P1,normal, R1,normal), suc(P1, R0),
    suc(R1, P1),
    cycle_I(P2,normal, R2,normal), rr2(R0, R2,short), suc(P2, R1),
    suc(R2, P2).
Nb de noeuds visités : 75, cpu(113.86), local((5,0,0,45)), total((5,0,0,45))


rule(atv):-
    qrs(R0,abnormal), qrs(R1,abnormal), suc(R1, R0),
    qrs(R2,abnormal), suc(R2, R1).
Nb de noeuds visités : 140, cpu(205.62), local((7,0,0,43)), total((7,0,0,43))
```

```
rule(fa):-
    qrs(R0,normal), qrs(R1,normal), suc(R1, R0).
Nb de noeuds visités : 16, cpu(47.02), local((7,0,0,43)), total((7,0,0,43))
```

C.4.2 Voie V-ABP sans diastole

	multisource naïf			multisource biaisé		
	PrecAp	PrecT	Nbcycl	PrecAp	PrecT	Nbcycl
rs	1	0.96	5	1	0.94	5/4
esv	1	0.94	5/6	1	0.94	5/6
bige	0.98	0.94	4	1	1	4
doublet	0.773	0.58	4	1	1	4
tv	1	0.96	3	1	1	3
tsv	0.94	0.94	2	1	**1**	5
fa	0.99	0.9	2/2	0.99	0.9	3/6/5

TAB. C.3 – Apprentissages monosources et multisources sur les données de l'ECG sans onde P et sur les données de pression sans diastole

```
rule(rs):-
    systole(Sys0), systole(Sys1),
    amp_ss(Sys0, Sys1,nul,normal), suc(Sys1, Sys0),
    systole(Sys2), amp_ss(Sys1, Sys2,nul,normal), suc(Sys2, Sys1),
    qrs(R0,normal), suci(R0, Sys2),
    systole(Sys3), amp_ss(Sys2, Sys3,nul,normal), suci(Sys3, R0),
    qrs(R1,normal), rr1(R0, R1,normal), suci(R1, Sys3),
    systole(Sys4), suci(Sys4, R1),
    qrs(R2,normal), suci(R2, Sys4),
    qrs(R3,normal), suc(R3, R2),
    qrs(R4,normal), suc(R4, R3).
cpu(222.27), local((8,0,0,42)), total((8,0,0,42))


rule(bige):-
    qrs(R0,normal), qrs(R1,abnormal), suc(R1, R0),
    qrs(R2,normal), suc(R2, R1),
    qrs(R3,abnormal), suc(R3, R2).
cpu(65.55), local((7,0,0,43)), total((7,0,0,43))


rule(esv):-
    qrs(R0,normal), qrs(R1,normal), suc(R1, R0),
```

```
    qrs(R2,normal), suc(R2, R1),
    qrs(R3,abnormal), suc(R3, R2),
    qrs(R4,normal), suc(R4, R3),
    qrs(R5,normal), suc(R5, R4).
cpu(81.78), local((10,0,0,40)), total((10,0,0,40))


rule(doublet):-
    qrs(R0,normal), qrs(R1,abnormal), suc(R1, R0),
    qrs(R2,abnormal), suc(R2, R1),
    qrs(R3,normal), suc(R3, R2).
cpu(75.09), local((6,0,0,44)), total((6,0,0,44))


rule(tsv):-
    qrs(R0,normal), systole(Sys0), suci(Sys0, R0),
    qrs(R1,normal), rr1(R0, R1,short), suci(R1, Sys0),
    systole(Sys1), amp_ss(Sys0, Sys1,nul,normal), suci(Sys1, R1).
cpu(334.19), local((5,0,1,44)), total((5,0,1,44))


rule(tv):-
    qrs(R0,abnormal), qrs(R1,abnormal), suc(R1, R0),
    qrs(R2,abnormal), suc(R2, R1).
cpu(21.16), local((7,0,0,43)), total((7,0,0,43))


rule(fa):-
    qrs(R0,normal), qrs(R1,normal), suc(R1, R0),
    qrs(R2,normal), rythm(R0, R1, R2,irregular), suc(R2, R1).
cpu(541.59), local((6,1,0,43)), total((6,1,0,43))
```

C.4.3 V sans forme du QRS-ABP sans diastole

```
rule(rs):-
    systole(Sys0), systole(Sys1),
    amp_ss(Sys0, Sys1,nul,normal), suc(Sys1, Sys0),
    systole(Sys2), amp_ss(Sys1, Sys2,nul,normal),
    ss2(Sys0, Sys2,normal), suc(Sys2, Sys1),
    systole(Sys3), amp_ss(Sys2, Sys3,nul,normal), suc(Sys3, Sys2),
    systole(Sys4), amp_ss(Sys3, Sys4,nul,normal), suc(Sys4, Sys3).
Nb de noeuds visités : 712, cpu(96.32), local((8,0,0,42)), total((8,0,0,42))
```

```
rule(bige):-
    systole(Sys0), systole(Sys1), amp_ss(Sys0, Sys1,pos,long),
    ss1(Sys0, Sys1,long), suc(Sys1, Sys0),
    systole(Sys2), suc(Sys2, Sys1),
    systole(Sys3), ss1(Sys2, Sys3,long), suc(Sys3, Sys2).
 cpu(107.94), local((5,2,0,43)), total((5,2,0,43))
rule(bige):-
    systole(Sys0), systole(Sys1),
    amp_ss(Sys0, Sys1,nul,normal), suc(Sys1, Sys0),
    systole(Sys2), ss1(Sys1, Sys2,short), suc(Sys2, Sys1),
    systole(Sys3), amp_ss(Sys2, Sys3,pos,long),
    ss2(Sys1, Sys3,normal), suc(Sys3, Sys2),
    systole(Sys4), suc(Sys4, Sys3).
Nb de noeuds visités : 2116, cpu(273), local((2,0,0,43)), total((3,4,0,43))


rule(esv):-
    systole(Sys0), systole(Sys1), amp_ss(Sys0, Sys1,neg,long),
    ss1(Sys0, Sys1,normal), suc(Sys1, Sys0),
    systole(Sys2), amp_ss(Sys1, Sys2,nul,normal),
    ss2(Sys0, Sys2,normal), suc(Sys2, Sys1),
    systole(Sys3), ss1(Sys2, Sys3,normal), suc(Sys3, Sys2),
    systole(Sys4), ss2(Sys2, Sys4,normal), suc(Sys4, Sys3),
    qrs(R0), suci(R0, Sys4),
    qrs(R1), rr1(R0, R1,normal), suc(R1, R0),
    qrs(R2), suc(R2, R1),
    qrs(R3), suc(R3, R2).
cpu(148.23), local((5,5,1,39)), total((5,5,1,39))
rule(esv):-
    systole(Sys0), systole(Sys1), ss1(Sys0, Sys1,normal), suc(Sys1, Sys0),
    qrs(R0), suci(R0, Sys1),
    qrs(R1), suci(R1, R0),
    systole(Sys2), amp_ss(Sys1, Sys2,nul,normal), suci(Sys2, R1),
    qrs(R2), suci(R2, Sys2),
    systole(Sys3), ss1(Sys2, Sys3,normal), suci(Sys3, R2).
cpu(351.92), local((2,3,0,40)), total((2,8,0,40))
rule(esv):-
    systole(Sys0), systole(Sys1), amp_ss(Sys0, Sys1,neg,long),
    ss1(Sys0, Sys1,normal), suc(Sys1, Sys0),
    systole(Sys2), amp_ss(Sys1, Sys2,nul,normal),
    ss2(Sys0, Sys2,normal), suc(Sys2, Sys1),
    systole(Sys3), ss1(Sys2, Sys3,normal), suc(Sys3, Sys2),
    systole(Sys4), ss2(Sys2, Sys4,normal), suc(Sys4, Sys3).
Nb de noeuds visités : 5014, cpu(450.92), local((3,0,3,37)), total((9,1,3,37))
```

```
rule(doublet):-
    systole(Sys0), qrs(R0), suci(Sys0, R0),
    qrs(R1), rr1(R0, R1,normal), suci(R1, Sys0),
    qrs(R2), rr2(R0, R2,short), suci(R2, R1).
cpu(89.37), local((3,3,0,44)), total((3,3,0,44))
rule(doublet):-
    systole(Sys0), systole(Sys1), ss1(Sys0, Sys1,short), suc(Sys1, Sys0),
    systole(Sys2), ss1(Sys1, Sys2,normal), suc(Sys2, Sys1),
    systole(Sys3), amp_ss(Sys2, Sys3,pos,long), suc(Sys3, Sys2),
    systole(Sys4), ss1(Sys3, Sys4,normal), suc(Sys4, Sys3).
Nb de noeuds visités : 2709, cpu(267.73), local((3,0,0,44)), total((4,2,0,44))


rule(tsv):-
    systole(Sys0), systole(Sys1), amp_ss(Sys0, Sys1,nul,normal),
    ss1(Sys0, Sys1,short), suc(Sys1, Sys0),
    systole(Sys2), suc(Sys2, Sys1),
    systole(Sys3), ss1(Sys2, Sys3,short), suc(Sys3, Sys2),
    systole(Sys4), suc(Sys4, Sys3),
    qrs(R0), suci(R0, Sys4),
    qrs(R1), suc(R1, R0),
    qrs(R2), rr1(R1, R2,short), suc(R2, R1),
    qrs(R3), rr1(R2, R3,short), suc(R3, R2).
Nb de noeuds visités : 2495, cpu(290.36), local((5,0,0,45)), total((5,0,0,45))


rule(tv):-
    qrs(R0), qrs(R1), rr1(R0, R1,short), suc(R1, R0),
    qrs(R2), rr1(R1, R2,normal), rr2(R0, R2,normal), suc(R2, R1),
    qrs(R3), rr1(R2, R3,normal), suc(R3, R2),
    qrs(R4), rr1(R3, R4,normal), suc(R4, R3),
    systole(Sys0), suci(Sys0, R4),
    qrs(R5), suci(R5, Sys0),
    systole(Sys1), ss1(Sys0, Sys1,short), suci(Sys1, R5).
 cpu(214.45), local((2,5,0,43)), total((2,5,0,43))
rule(tv):-
    systole(Sys0), systole(Sys1), ss1(Sys0, Sys1,normal), suc(Sys1, Sys0),
    systole(Sys2), amp_ss(Sys1, Sys2,pos,long),
    ss1(Sys1, Sys2,short), suc(Sys2, Sys1),
    systole(Sys3), amp_ss(Sys2, Sys3,pos,long), suc(Sys3, Sys2).
cpu(464.7), local((3,2,1,42)), total((4,3,1,42))
rule(tv):-
```

```
    systole(Sys0), systole(Sys1), ss1(Sys0, Sys1,short), suc(Sys1, Sys0),
    systole(Sys2), amp_ss(Sys1, Sys2,pos,long),
    ss1(Sys1, Sys2,short), suc(Sys2, Sys1),
    qrs(R0), suci(R0, Sys2),
    systole(Sys3), amp_ss(Sys2, Sys3,pos,long), suci(Sys3, R0),
    qrs(R1), suci(R1, Sys3),
    qrs(R2), rr1(R1, R2,normal), rr2(R0, R2,normal), suc(R2, R1),
    qrs(R3), suc(R3, R2).
Nb de noeuds visités : 6231, cpu(720.09), local((2,0,1,42)), total((2,5,1,42))


rule(fa):-
    systole(Sys0), systole(Sys1),
    amp_ss(Sys0, Sys1,nul,normal), suc(Sys1, Sys0),
    systole(Sys2), ss1(Sys1, Sys2,long), suc(Sys2, Sys1),
    systole(Sys3), suc(Sys3, Sys2),
    systole(Sys4), ss2(Sys2, Sys4,normal), suc(Sys4, Sys3).
cpu(136.37), local((5,2,0,43)), total((5,2,0,43))
rule(fa):-
    qrs(R0), qrs(R1), rr1(R0, R1,short), suc(R1, R0),
    systole(Sys0), suci(Sys0, R1),
    qrs(R2), rr1(R1, R2,short), suci(R2, Sys0),
    systole(Sys1), suci(Sys1, R2),
    qrs(R3), rr2(R1, R3,normal), suci(R3, Sys1),
    systole(Sys2), ss1(Sys1, Sys2,short), suci(Sys2, R3).
Nb de noeuds visités : 4134, cpu(577.48), local((2,0,1,42)), total((3,4,1,42))
```

C.4.4 P-QRS sans forme

```
rule(rs):-
    p_wav(P0,normal), p_wav(P1,normal), suc(P1, P0),
    p_wav(P2,normal), pp2(P0, P2,normal), suc(P2, P1),
    p_wav(P3,normal), suc(P3, P2),
    p_wav(P4,normal), suc(P4, P3),
    p_wav(P5,normal), pp1(P4, P5,normal), suc(P5, P4),
    qrs(R0), suci(R0, P5),
    qrs(R1), suc(R1, R0),
    qrs(R2), rr1(R1, R2,normal), suc(R2, R1),
    qrs(R3), rr1(R2, R3,normal), suc(R3, R2),
    qrs(R4), rr1(R3, R4,normal), suc(R4, R3),
    qrs(R5), rr1(R4, R5,normal), suc(R5, R4).
Nb de noeuds visités : 1331, cpu(595.4), local((8,0,18,24)), total((8,0,18,24))
```

```
rule(bige):-
    qrs(R0), p_wav(P0,normal), suci(R0, P0),
    qrs(R1), suci(R1, R0),
    p_wav(P1,normal), suci(P1, R1),
    qrs(R2), suci(R2, P1),
    qrs(R3), suci(R3, R2).
Nb de noeuds visités : 401, cpu(163.32), local((7,0,0,43)), total((7,0,0,43))


rule(esv):-
    qrs(R0), p_wav(P0,normal), suci(R0, P0),
    qrs(R1), suci(R1, R0),
    p_wav(P1,normal), suci(P1, R1),
    qrs(R2), rr1(R1, R2,normal), suci(R2, P1),
    p_wav(P2,normal), suci(P2, R2),
    qrs(R3), suci(R3, P2),
    p_wav(P3,normal), suci(P3, R3),
    qrs(R4), suci(R4, P3),
    p_wav(P4,normal), suci(P4, R4),
    qrs(R5), suci(R5, P4),
    p_wav(P5,normal), suci(P5, R5).
Nb de noeuds visités : 3274, cpu(746), local((7,3,1,39)), total((7,3,1,39))
rule(esv):-
    qrs(R0), p_wav(P0,normal), suci(P0, R0),
    qrs(R1), rr1(R0, R1,normal), suci(R1, P0),
    p_wav(P1,normal), suci(P1, R1),
    qrs(R2), suci(R2, P1),
    qrs(R3), suci(R3, R2),
    p_wav(P2,normal), suci(P2, R3),
    qrs(R4), suci(R4, P2),
    p_wav(P3,normal), suci(P3, R4).
Nb de noeuds visités:3274, cpu(1271.24),local((3,0,0,40)),total((10,0,0,40))


rule(doublet):-
    qrs(R0), p_wav(P0,normal), suci(R0, P0),
    qrs(R1), suci(R1, R0),
    qrs(R2), suci(R2, R1),
    p_wav(P1,normal), suci(P1, R2).
Nb de noeuds visités : 349, cpu(140), local((6,0,0,44)), total((6,0,0,44))


rule(tsv):-
    p_wav(P0,normal), p_wav(P1,normal), pp1(P0, P1,short), suc(P1, P0).
```

```
Nb de noeuds visités : 44, cpu(16.05), local((5,0,0,45)), total((5,0,0,45))


rule(tv):-
    qrs(R0), p_wav(P0,normal), suci(R0, P0),
    qrs(R1), rr1(R0, R1,short), suci(R1, R0),
    qrs(R2), suci(R2, R1).
cpu(225.89), local((3,4,0,43)), total((3,4,0,43))
rule(tv):-
    p_wav(P0,normal), p_wav(P1,normal), pp1(P0, P1,verylong), suc(P1, P0).
Nb de noeuds visités : 2305, cpu(548.4), local((4,0,3,40)), total((6,1,3,40))


rule(fa):-
    qrs(R0), qrs(R1), rr1(R0, R1,normal), suc(R1, R0),
    qrs(R2), suc(R2, R1),
    qrs(R3), rr2(R1, R3,long), suc(R3, R2).
cpu(21.85), local((3,4,0,43)), total((3,4,0,43))
rule(fa):-
    qrs(R0), qrs(R1), rr1(R0, R1,short), suc(R1, R0),
    qrs(R2), rr1(R1, R2,short), suc(R2, R1),
    qrs(R3), rr2(R1, R3,normal), suc(R3, R2).
Nb de noeuds visités : 228, cpu(42.59), local((3,1,2,41)), total((3,4,2,41))
```

Table des figures

Bibliographie

[Benferhat et Lang, 2001] Benferhat, S. et Lang, J. (2001). Conference paper assignment. *International Journal of Intelligent Systems*, 16:1183–1192.

[Benzaken, 1991] Benzaken, C. (1991). *Systèmes formels : introduction à la logique et à la théorie des langages*. Masson.

[Bloch *et al.*, 2001] Bloch, I., Hunter, A., Appriou, A., Ayoun, A., Benferhat, S., Besnard, P., Cholvy, L., Cooke, R., Cuppens, F., Dubois, D., Fargier, H., Grabisch, M., Kruse, R., Lang, J., Moral, S., Prade, H., Saffiotti, A., Smets, P. et Sossai, C. (2001). Fusion: General concepts and characteristics. *International Journal of Intelligent Systems*, 16:1107–1134.

[Blockeel, 1998] Blockeel, H. (1998). *Top-down induction of first order logical decision trees*. Phd, Department of Computer Science, K.U.Leuven, Leuven, Belgium.

[Blockeel et De Raedt, 1998] Blockeel, H. et De Raedt, L. (1998). Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297.

[Blockeel *et al.*, 2004] Blockeel, H., Dzeroski, S., Kompare, B., Kramer, S., Pfahringer, B. et Van Laer, W. (2004). Experiments in predicting biodegradability. *Applied Artificial Intelligence*, 18(2):157–181.

[Blondeau et Hiltgen, 1980] Blondeau, M. et Hiltgen, M. (1980). *Électrocardiographie clinique*. Masson.

[Blum et Mitchell, 1998] Blum, A. et Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *In COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, pages 92–100.

[Bratko *et al.*, 1989] Bratko, I., Mozetič, I. et Lavrač, N. (1989). *Kardio: A Study in Deep and Qualitative Knowledge for Expert Systems*. MIT Press, Cambridge, MA.

[Calvelo *et al.*, 2000] Calvelo, D., Chambrin, M.-C., Pomorski, D. et Ravaux, P. (2000). Towards symbolization using data-driven extraction of local trends for ICU monitoring. *Artificial Intelligence in Medicine*, 19(3):203–223.

[Carrault *et al.*, 2003] Carrault, G., Cordier, M., Quiniou, R. et Wang, F. (2003). Temporal abstraction and inductive logic programming for arrhythmia recognition from ECG. *Artificial Intelligence in Medicine*, 28:231–263.

[Cauvin *et al.*, 1998] Cauvin, S., Cordier, M.-O., Dousson, C., Laborie, P., Lévy, F., Montmain, J., Porcheron, M., Servet, I. et Travé-Massuyès, L. (1998).

The alarm research group, monitoring and alarm interpretation in industrial environnements. *AI Communications*, 11(3,4):139–173.

[Chambrin, 2001] Chambrin, M.-C. (2001). Alarms in the intensive care unit: how can the number of false alarms be reduced? *Critical Care*, 5(4):184–188.

[Cholvy, 2003] Cholvy, L. (2003). Information evaluation in fusion: a case study. *In ECSQARU-03 Workshop "Uncertainty, Incompleteness, Imprecision and Conflict in Multiple Data Sources*, Aalborg, Denmark.

[Cholvy et Hunter, 1997] Cholvy, L. et Hunter, A. (1997). Information fusion in logic: A brief overview. *In Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU)*, pages 86–95.

[Clark et Boswell, 1991] Clark, P. et Boswell, R. (1991). Rule induction with CN2: Some recent improvements. *In Proc. Fifth European Working Session on Learning*, pages 151–163, Berlin. Springer.

[Clark et Niblett, 1989] Clark, P. et Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, 3:261–283.

[Coiera, 1993] Coiera, E. (1993). Intelligent monitoring and control of dynamic physiological systems. *Artificial Intelligence in Medicine*, 5:1–8.

[Cordier, 2005] Cordier, M. (2005). Sacadeau: A decision-aid system to improve stream-water quality. *ERCIM News*, (61):35–37.

[Cornuéjols et Miclet, 2003] Cornuéjols, A. et Miclet, L. (2003). *Apprentissage artificiel*. Eyrolles.

[Cussens et Džeroski, 2000] Cussens, J. et Džeroski, S., éditeurs (2000). *Learning Language in Logic*, volume 1925 de *Lecture Notes in Computer Science*. Springer.

[De Raedt, 1997] De Raedt, L. (1997). Logical settings for concept-learning. *Artif. Intell.*, 95(1):187–201.

[De Raedt et Bruynooghe, 1993] De Raedt, L. et Bruynooghe, M. (1993). A theory of clausal discovery. *In Proceedings of the 13th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann. CLAUDIEN.

[De Raedt et Dehaspe, 1997] De Raedt, L. et Dehaspe, L. (1997). Clausal discovery. *Machine Learning*, 26:99–146.

[De Raedt et Džeroski, 1994] De Raedt, L. et Džeroski, S. (1994). First order jk-clausal theories are PAC-learnable. *Artif. Intell.*, 70:375–392.

[De Raedt et Van Laer, 1995] De Raedt, L. et Van Laer, W. (1995). Inductive constraint logic. *In Proceedings of the 6th Conference on Algorithmic Learning Theory*, LNCS 997, pages 80–94. Springer Verlag.

[Dechter *et al.*, 1991] Dechter, R., Meiri, I. et Pearl, J. (1991). Temporal constraint networks. *Artif. Intell.*, 49(1-3):61–95.

[Dehaspe, 1999] Dehaspe, L. (1999). Frequent pattern discovery in first-order logic. *AI Commun.*, 12(1-2):115–117.

[Dehaspe et De Raedt, 1996] Dehaspe, L. et De Raedt, L. (1996). DLAB: A declarative language bias formalism. *In International Syposium on Methodologies for Intelligent Systems*, pages 613–622.

[Delahaye, 1988] Delahaye, J. (1988). *Outils logiques pour l'IA*. Eyrolles.

[Dempster *et al.*, 1977] Dempster, A. P., Laird, N. M. et Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statististical Society*, 39-B:1–38.

[Denis *et al.*, 2003] Denis, F., Gilleron, R., Laurent, A. et Tommasi, M. (2003). Text classification and co-training from positive and unlabeled examples. *In Proceedings of the ICML Workshop: the Continuum from Labeled Data to Unlabeled Data in Machine Learning and Data Mining*, pages 80 – 87.

[Dojat et Chittaro, 1997] Dojat, M. et Chittaro, L. (1997). Using a general theory of time and change in patient monitoring : experiment and evaluation. *Computers in Biology and Medecine*, 27(5).

[Dojat *et al.*, 1997] Dojat, M., Pachet, F., Guessoum, Z., Touchard, D., Harf, A. et Brochard, L. (1997). Néoganesh: a working system for the automated control of assisted ventillation in icus. *Artificial Intelligence in Medicine*, 11:97–117.

[Dojat *et al.*, 1998] Dojat, M., Ramaux, N. et Fontaine, D. (1998). Scenario recognition for temporal reasoning in medical domains. *Artificial Intelligence in Medicine*, 14(5).

[Dolsak et Muggleton, 1992] Dolsak, B. et Muggleton, S. (1992). The application of Inductive Logic Programming to finite element mesh design. *In* Muggleton, S., éditeur : *Inductive Logic Programming*. Academic Press, London.

[Dougherty *et al.*, 1995] Dougherty, J., Kohavi, R. et Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *In International Conference on Machine Learning*, pages 194–202.

[Dousson, 1996] Dousson, C. (1996). Alarm driven supervision for telecommunication networks. ii- on-line chronicle recognition. *Annales des Télécommunications*, 51(9-10):501–508.

[Dousson *et al.*, 1993] Dousson, C., Gaborit, P. et Ghallab, M. (1993). Situation recognition: representation and algorithms. *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 166–172, Chambéry, France. Morgan Kaufman.

[Dubois *et al.*, 2001] Dubois, D., Grabisch, M., Prade, H. et Smets, P. (2001). Using the transferable belief model and a qualitative possibility theory approach on an illustrative example: the assessment of the value of a candidate. *International Journal of Intelligent Systems*, 16:1183–1192.

[Duchêne, 2004] Duchêne, F. (2004). *Fusion de données multicapteurs pour un système de télésurveillance médicale de personnes à domicile*. Thèse, Université Joseph Fourier.

[Džeroski et Bratko, 1992] Džeroski, S. et Bratko, I. (1992). Handling noise in inductive logic programming. *In* Muggleton, S., éditeur : *ILP92*, Report ICOT TM-1182.

[Džeroski et Lavrač, 2001] Džeroski, S. et Lavrač, N. (2001). *Relational Data Mining*. Springer Verlag.

[Fayyad et Irani, 1993] Fayyad, U. et Irani, K. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. *In International Joint Conference on Artificial Intelligence IJCAI-93*.

[Fildier, 2001] Fildier, C. (2001). Apprentissage et reconnaissance de scénarios à partir d'ecg multivoies. Mémoire de D.E.A., IRISA.

[Fromont *et al.*, 2005a] Fromont, E., Cordier, M. et Quiniou, R. (2005a). Extraction de connaissances provenant de données multisources pour la caractérisation d'arythmies cardiaques. *RNTI-E-4, Cepaduès Editions*, Fouille de données complexes.

[Fromont *et al.*, 2004] Fromont, E., Cordier, M.-O. et Quiniou, R. (2004). Learning from multi source data. *In PKDD'04 (Knowledge Discovery in Databases)*, volume 3202 de *Lecture Notes in Artificial Intelligence*, pages 503–505, Pise, Italie. Springer.

[Fromont *et al.*, 2003] Fromont, E., Cordier, M.-O., Quiniou, R. et Hernandez, A. (2003). Kardio and calicot: a comparison of two cardiac arrhythmia classifiers. *In AIME'03 Workshop: Qualitative and Model-based Reasoning in Biomedicine*, Protaras, Cyprus.

[Fromont et Portet, 2005] Fromont, E. et Portet, F. (2005). Pilotage d'un système de monitoring cardiaque multisource. *In Actes de conférence de MajecStic'2005 (MAnifestation des JEunes Chercheurs STIC)*, pages 346–354, Rennes, France.

[Fromont *et al.*, 2005b] Fromont, E., Quiniou, R. et Cordier, M. (2005b). Learning rules from multisource data for cardiac monitoring. *In* Miksch, S., Hunter, J. et Keravnou, E., éditeurs : *AIME'05 (Artificial Intelligence in Medicine)*, volume 3581 de *LNAI*, pages 484–493, Aberdeen, Scotland. Springer Verlag.

[Fromont *et al.*, 2006] Fromont, E., Quiniou, R. et Cordier, M.-O. (2006). Apprentissage multisource par programmation logique inductive. *In RFIA'2006 : 15ème Congrès Reconnaissance des Formes et Intelligence Artificielle*, page à paraître, Tours, France.

[Fürnkranz, 1997] Fürnkranz, J. (1997). Dimensionality reduction in ILP: A call to arms. *In* De Raedt, L. et Muggleton, S., éditeurs : *Proceedings of the IJCAI-97 Workshop on Frontiers of Inductive Logic Programming*, pages 81–86. Nagoya, Japan.

[Fürnkranz, 1998] Fürnkranz, J. (1998). Integrative windowing. *Journal of Artificial Intelligence Research*, 8(0):129–164.

[Garvey, 1990] Garvey, T. D. (1990). A survey of ai approaches to the integration of information. Rapport technique 481, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025.

[Genesereth et Nilsson, 1987] Genesereth, M. et Nilsson, N. (1987). *Logical Foundation of Artificial Intelligence*. Morgan Kaufmann, Los Altos, California.

[Ghallab, 1996] Ghallab, M. (1996). On chronicles: representation, on-line recognition and learning. *In Proceedings of KR-96*, pages 597–606. Morgann-Kauffmann.

[Gritzali, 1988] Gritzali, F. (1988). Towards a generalized scheme for QRS detection in ECG waveforms. *Signal Processing*, 15:183–192.

[Helft, 1989] Helft, N. (1989). Induction as nonmonotonic inference. *In Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 149–156.

[Hernández, 2000] Hernández, A. (2000). *Fusion de signaux et de modèle pour la caractérisation d'arythmies cardiaques*. Phd, Université de Rennes 1, Rennes, France.

[Hernández *et al.*, 2000] Hernández, A., Carrault, G., Mora, F. et Bardou, A. (2000). Overview of carmen: A new dynamic quantitative cardiac model for ECG monitoring and its adaptation to observed signals. *Acta Biotheoretica*, 48(3-4):303–322.

[Hernández *et al.*, 1999] Hernández, A., Carrault, G., Mora, F., Thoraval, L., Passariello, G. et Schleich, J. M. (1999). Multisensor fusion for atrial and ventricular activity detection in coronary care monitoring. *IEEE Transactions on Biomedical Engineering*, 46(10):1186–1190.

[Hernández *et al.*, 2002] Hernández, A. I., Carrault, G., Mora, F. et Bardou, A. (2002). Model-based interpretation of cardiac beats by evolutionary algorithms: signal and model interaction. *Artificial Intelligence in Medicine*, 23(211-235).

[Hernández et Carrault, 2004] Hernández, A. et Carrault, G. (2004). Intégration modèle-observation pour l'interprétation des battements cardiaques. *In Reconnaissance de Formes et Intelligence Artificielle (RFIA'04)*, pages 447–456, Toulouse.

[Hoeksel *et al.*, 1997] Hoeksel, S., Jansen, J., Blom, J. et Schreuder, J. (1997). Detection of dicrotic notch in arterial pressure signals. *Journal of Clinical Monitoring and Computing*, 13:309.

[Joachims, 1998] Joachims, T. (1998). Making large-scale support vector machine learning practical. *In* B. Schölkopf, C. Burges, A. S., éditeur : *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA.

[Kapur et Narendran, 1986] Kapur, D. et Narendran, P. (1986). Np-completeness of the set unification and matching problems. *In Proc. of the 8th international conference on Automated deduction*, pages 489–495, New York, NY, USA. Springer-Verlag New York, Inc.

[Karalic et Bratko, 1997] Karalic, A. et Bratko, I. (1997). First order regression. *Machine Learning*, 26(2-3):147–176.

[Kietz et Wrobel, 1992] Kietz, J.-U. et Wrobel, S. (1992). Controlling the complexity of learning in logic through syntactic and task-oriented models. *In* Muggleton, S., éditeur : *Inductive Logic Programming*. Academic Press.

[King *et al.*, 2004] King, R., Whelan, K., Jones, F., Reiser, P., C.H.Bryant, Muggleton, S., Kell, D. et Oliver, S. (2004). Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252.

[Kiritchenko et Matwin, 2001] Kiritchenko, S. et Matwin, S. (2001). Email classification with co-training. *In CASCON '01: Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, page 8. IBM Press.

[Kramer, 1995] Kramer, S. (1995). Predicate invention: A comprehensive view. Rapport technique OFAI-TR-95-32, Austrian Research Institute for Artificial Intelligence, Vienna.

[Larsson et Hayes-Roth, 1998] Larsson, J. et Hayes-Roth, B. (1998). Guardian: An intelligent autonomous agent for medical monitoring and diagnosis. *IEEE Intelligent Systems*, 13(1):58–64.

[Lavrač et Džeroski, 1993] Lavrač, N. et Džeroski, S. (1993). *Inductive Logic Programming: Techniques and Applications*. Routledge, New York, NY, 10001.

[Le Moulec, 1991] Le Moulec, F. (1991). *Etude et réalisation d'un modèle qualitatif profond de l'activité éléctrique du coeur pour un système de monitoring intelligent en Unités de Soins Intensifs pour Coronariens*. Thèse, Université de Rennes 1.

[Lloyd, 1987] Lloyd, J. (1987). *Foundations of Logic Programming*. Springer-Verlag, Heidelberg.

[Luo et Kay, 1989] Luo, R. et Kay, M. (1989). Multisensor integration and fusion in intelligent systems. *IEEE Transactions on Systems, Man and Cybernetics*, 19(5):901–931.

[Maguy, 2002] Maguy, I. (2002). Detection et apprentissage de séquence multivoies. Rapport technique. rapport de DESS.

[Michalski, 1993] Michalski, R. S. (1993). A theory and methodology of inductive learning. pages 323–348.

[Michalski *et al.*, 1986] Michalski, R. S., Mozetic, I. et Hong, J. (1986). The multipurpose incremental learning system aq15 and it testing application to three medical domains. *In Proc. of AAAI-86*, pages 1041–1045, Philadelphia, PA.

[Miksch *et al.*, 1996] Miksch, S., Horn, W., Popow, C. et Paky, F. (1996). Utilizing temporal data abstraction for data validation and therapy planning for artificially ventilated newborn infants. *Artificial Intelligence in Medicine*, 8(6):543–576.

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, New York.

[Mitchell, 1982] Mitchell, T. M. (1982). Generalization as search. *Artif. Intell.*, 18(2):203–226.

[Moody, 1997a] Moody, G. B. (1997a). Ecg database applications guide. Harvard-MIT Division of Health Sciences and Technology Biomedical Engineering Center, Ninth Edition.

[Moody, 1997b] Moody, G. B. (1997b). Ecg database programmer's guide. Harvard-MIT Division of Health Sciences and Technology Biomedical Engineering Center, Ninth Edition. http://www.physionet.org/physiobank/database/mitdb/.

[Moody et Mark, 1996] Moody, G. B. et Mark, R. G. (1996). A database to support development and evaluation of intelligent intensive care monitoring. *Computers in Cardiology*, 23:657–660. http://ecg.mit.edu/mimic/mimic.html.

[Morik *et al.*, 2000] Morik, K., Imboff, M., Brockhausen, P., Joachims, T. et Gather, U. (2000). Knowledge discovery and knowledge validation in intensive care. *Artificial Intelligence in Medicine*, 19(3):225–249.

[Morik *et al.*, 1993] Morik, K., Wrobel, S., Kietz, J.-U. et Emde, W. (1993). *Knowledge Acquisition and Machine Learning: Theory, Methods and Applications*. AP.

[Muggleton, 1995] Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286.

[Muggleton et Buntine, 1988a] Muggleton, S. et Buntine, W. (1988a). Machine invention of first-order predicates by inverting resolution. *In Proceedings of the Fifth International Conference on Machine Learning*, pages 339–352. Kaufmann. CIGOL.

[Muggleton et Buntine, 1988b] Muggleton, S. et Buntine, W. L. (1988b). Machine invention of first order predicates by inverting resolution. *In ML*, pages 339–352.

[Muggleton et De Raedt, 1994] Muggleton, S. et De Raedt, L. (1994). Inductive Logic Programming: Theory and methods. *The Journal of Logic Programming*, 19 & 20:629–680.

[Muggleton et Feng, 1990] Muggleton, S. et Feng, C. (1990). Efficient induction of logic programs. *In Proceedings of the First Conference on Algorithmic Learning Theory*, Tokyo. Ohmsha. GOLEM.

[Nienhuys-Cheng et de Wolf, 1996] Nienhuys-Cheng, S.-H. et de Wolf, R. (1996). Least generalisations and greatest specializations of sets of clauses. *Journal of Artificial Intelligence Research*, 4:341–363.

[Nigam et Ghani, 2000] Nigam, K. et Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. *In CIKM*, pages 86–93.

[Nédellec et Rouveirol, 1994] Nédellec, C. et Rouveirol, C. (1994). Specifications of the haiku system. Rapport interne 928, Laboratoire de Recherche en Informatique, Université de Paris Sud, France.

[Nédellec *et al.*, 1996] Nédellec, C., Rouveirol, C., Adé, H., Bergadano, F. et Tausend, B. (1996). Declarative bias in ILP. *In* De Raedt, L., éditeur : *Advances in Inductive Logic Programming*, pages 82–103. IOS Press.

[Oncina et García, 1991] Oncina, J. et García, P. (1991). *Inferring regular languages in polynomial update time*. World Scientific Publishing. Pattern Recognition and Image Analysis, 1991,.

[Pan et Tompkins, 1985] Pan, J. et Tompkins, W. J. (1985). A real-time QRS detection algorithm. *IEEE Transactions on Biomedical Engineering*, BME-32(3):230–236.

[Pierson, 1998] Pierson, D. J. (1998). *Principles and practice of intensive care monitoring*, chapitre Goals and indications for monitoring, pages 33–44. McGraw-Hill, Health Professions Division, New York.

[Plotkin, 1970] Plotkin, G. (1970). A note on inductive generalisation. *In* Meltzer, B. et Michie, D., éditeurs : *Machine Intelligence 5*, pages 153–163. Elsevier North Holland, New York.

[Plotkin, 1971] Plotkin, G. (1971). A further note on inductive generalisation. *In Machine Intelligence 6*, pages 101–124. Elsevier North Holland.

[Portet, 2005] Portet, F. (2005). *Pilotage d'algorithmes pour la reconnaissance en ligne d'arythmies cardiaques*. Thèse de doctorat, Université de Rennes 1.

[Portet *et al.*, 2005a] Portet, F., Carrault, G., Cordier, M.-O. et Quiniou, R. (2005a). Pilotage en ligne d'algorithmes de traitement du signal guidé par le contexte courant. *In 7e Rencontres des Jeunes Chercheurs en Intelligence Artificielle (RJCIA'05)*, pages 1–14, Nice.

[Portet *et al.*, 2005b] Portet, F., Hernández, A. et Carrault, G. (2005b). Evaluation of real-time QRS detection algorithms in variable contexts. *Medical & Biological Engineering & Computing*, 43(3):381–387.

[Quiniou *et al.*, 2001] Quiniou, R., Cordier, M., Carrault, G. et Wang, F. (2001). Application of ILP to cardiac arrhythmia characterization for chronicle recognition. *In* Rouveirol, C. et Sebag, M., éditeurs : *Proceedings of ILP-2001*, volume 2157 de *LNAI*, pages 193–215. Springer Verlag.

[Quinlan et Cameron-Jones, 1993] Quinlan, J. et Cameron-Jones, R. (1993). FOIL: a midterm report. *In* Brazdil, P., éditeur : *Proceedings of the 6th European Conference on Machine Learning*, volume 667 de *Lecture Notes in Artificial Intelligence*, pages 3–20. Springer-Verlag.

[Quinlan, 1983] Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. *In* Michalski, R. S., Carbonell, J. G. et Mitchell, T. M., éditeurs : *Machine Learning: An Artificial Intelligence Approach*, pages 463–482. Springer, Berlin, Heidelberg.

[Quinlan, 1990] Quinlan, R. (1990). Learning logical definitions from relations. *Machine Learning*, 5:239–266.

[Rouveirol et Puget, 1990] Rouveirol, C. et Puget, J.-F. (1990). Beyond inversion of resolution. *In Proceedings of the seventh international conference (1990) on Machine learning*, pages 122–130, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Schapire, 2001] Schapire, R. (2001). The boosting approach to machine learning: An overview.

[Senhadji *et al.*, 2002] Senhadji, L., Wang, F., Hernandez, A. et Carrault, G. (2002). Wavelet extrema representation for qrs-t cancellation and p wave detection. pages 37–40.

[Shapiro, 1983] Shapiro, E. Y. (1983). *Algorithmic Program Debugging*. MIT Press, Cambridge, MA. ISBN 0-262-19218-7.

[Sharshar *et al.*, 2005] Sharshar, S., Allart, L. et Chambrin, M.-C. (2005). A new approach to the abstraction of monitoring data in intensive care. *In* Miksch, S.,

Hunter, J. et Keravnou, E., éditeurs : *AIME'05 (Artificial Intelligence in Medicine)*, volume 3581 de *LNAI*, pages 13–22, Aberdeen, Scotland. Springer Verlag.

[Shortliffe, 1976] Shortliffe, E. (1976). *Computer-Based Medical Consultations: MYCIN*. Elsevier, New York.

[Siregar *et al.*, 1995] Siregar, P., Chahine, M., Lemoulec, F. et Beux, P. L. (1995). An interactive qualitative model in cardiology. *Computers and Biomedical Research*, 28(6):443–478.

[Siregar *et al.*, 1989] Siregar, P., Coatrieux, J. et Beux., P. L. (1989). Kiss : Knowledge based interactive signal monitoring system.

[Sittig *et al.*, 1989] Sittig, D., Pace, N., Gardner, R., Beck, E. et Morris, A. (1989). Implementation of a computerized patient advice system using the help clinical information system. *Computers and Biomedical Research*, 22(5):474–487.

[Srinivasan, 2003] Srinivasan, A. (2003). Aleph manual v4 and above. http ://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/.

[Srinivasan *et al.*, 1996] Srinivasan, A., Muggleton, S., King, R. et Sternberg, M. (1996). Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence Journal*, 85(1,2):277–299.

[Stein, 1999] Stein, E. (1999). *Rapid analysis of arrhytmias*. Lippincott Williams ans Wilkins.

[Thoraval *et al.*, 1997] Thoraval, L., Carrault, G., Schleich, J., Summers, R., de Velde, M. V. et Diaz, J. (1997). Data fusion of electrophysiological and haemodynamic signals for ventricular rhythm tracking. *IEEE Engineering in Medicine and Biology Magazine*, 16 (6):48–55.

[Tsien et Fackler, 1997] Tsien, C. et Fackler, J. (1997). Poor prognosis for existing monitors in the intensive care unit. *Critical Care*, 25:614–619.

[Uckun, 1993] Uckun, S. (1993). Intelligent systems in patient monitoring and therapy management.

[Valiant, 1984] Valiant, L. G. (1984). A theory of the learnable. *In STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 436–445, New York, NY, USA. ACM Press.

[Van Laer, 2002] Van Laer, W. (2002). *From Propositional to First Order Logic in Machine Learning and Data Mining - Induction of first order rules with ICL*. Phd, Department of Computer Science, K.U.Leuven, Leuven, Belgium.

[Van Laer et De Raedt, 2001] Van Laer, W. et De Raedt, L. (2001). How to upgrade propositional learners to first order logic: A case study. *Lecture Notes in Computer Science*, 2049:102–131.

[Van Laer *et al.*, 1997] Van Laer, W., De Raedt, L. et Džeroski, S. (1997). On multi-class problems and discretization in inductive logic programming. *In International Syposium on Methodologies for Intelligent Systems*, pages 277–286.

[Van Laer *et al.*, 1996] Van Laer, W., Džeroski, S. et De Raedt, L. (1996). Multi-class problems and discretization in ICL. *In* Pfahringer, B. et Fuernkranz, J., éditeurs : *Proceedings of the MLnet Familiarization Workshop on Data Mining with Inductive Logic Programming*, pages 53–0.

[Vilhelm *et al.*, 2000] Vilhelm, C., Ravaux, P., Calvelo, D., Jaborska, A., Chambrin, M.-C. et Boniface, M. (2000). Think!: a unified numerical-symbolic knowledge representation scheme and reasoning system. *Artif. Intell.*, 116(1-2):67–85.

[Wang, 2002] Wang, F. (2002). *Abstraction temporelle de signal ECG, apprentissage inductif de contraintes temporelles et reconnaissance des arythmies cardiaques*. Phd, Université de Rennes 1, Rennes, France.

[Wemmert et Gançarski, 2001] Wemmert, C. et Gançarski, P. (2001). A new method of data fusion and its integration in a multi-agent system of automatic refinement of classifications. *In 5th World Multi-Conference on Systemics, Cybernetics and Informatics*, Orlando, USA.

[World Health Organization, 2004] World Health Organization (2004). The world health report 2004 : changing history. Rapport technique. http://www.who.int/whr/2004/en/.

[Wrobel, 1997] Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. *In* Komorowski, J. et Zytkow, J., éditeurs : *Proc. First European Symposion on Principles of Data Mining and Knowledge Discovery (PKDD-97)*, pages 78–87, Berlin. Springer Verlag.

[Yarowsky, 1995] Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. *In Meeting of the Association for Computational Linguistics*, pages 189–196.

[Zelle et Mooney, 1993] Zelle, J. M. et Mooney, R. J. (1993). Learning semantic grammars with constructive inductive logic programming. *In Proceedings of the 11th National Conference on Artificial Intelligence*, pages 817–822, Washington, D.C. AAAI Press/MIT Press.

Résumé

Ce travail a pour thème l'extraction de connaissances à partir de données provenant de plusieurs sources reflétant un même phénomène. L'objectif visé est l'amélioration de la qualité des systèmes de surveillance. Lorsque les données sont redondantes, l'utilisation de plusieurs sources permet de pallier aux problèmes de perte de signal et de bruit. Lorsque les données sont complémentaires, l'utilisation conjointe des différentes sources permet d'augmenter les performances en détection de ces systèmes.

Nous appliquons nos travaux au domaine du diagnostic d'arythmies cardiaques. Nous utilisons une technique d'apprentissage artificiel relationnel (la programmation logique inductive) pour apprendre des règles discriminantes permettant de caractériser les arythmies à partir de plusieurs voies d'un électrocardiogramme et de mesures de pression artérielle. Pour exploiter la redondance des sources, nous apprenons dans un premier temps, des règles à partir des données des différentes sources prises séparément. Pour exploiter la complémentarité des sources, un apprentissage multisource naïf consisterait à apprendre globalement sur l'ensemble des données et avec un langage d'expression des concepts permettant de couvrir toute la richesse des données représentées. En alternative à un tel type d'apprentissage, nous proposons une méthode plus efficace qui s'appuie sur des apprentissages monosources, ie. effectués sur chacune des sources séparément, pour biaiser l'espace de recherche multisource. Le fait de s'appuyer sur les règles monosources permet de restreindre le langage des hypothèses ainsi que le nombre de relations possibles entre les objets représentés sur les différentes sources.

Ce travail a été effectué dans le cadre du projet RNTS (Réseau National des Technologies et de la Santé) Cepica. Les résultats montrent que les règles apprises par apprentissage multisource sont au moins aussi bonnes que les règles monosources dans le cas où les données sont redondantes et meilleures dans les cas où les sources sont complémentaires. La technique d'apprentissage biaisé permet en outre d'apprendre des règles de manière beaucoup plus efficace que dans le cas naïf en bénéficiant d'un biais de langage généré automatiquement. Ces nouvelles règles sont incorporées au système Calicot pour la surveillance de patients souffrant de troubles du rythme cardiaque.

Abstract

This work belongs to the field of knowledge discovery from data coming from several sources that describe a sole phenomenon. The aim is to improve the quality of monitoring systems. When data are redundant, several sources can help the system to cope with the loss of a signal or the presence of important noise. When sources are complementary, the joint use of all the data coming from the different sources can improve the detection performances of the systems.

Our work is applied to cardiac arrhythmia diagnosis. We use a relational learning technique (inductive logic programming) to learn discriminating rules that characterize cardiac arrhythmias from different leads of an electrocardiogram and a measure of arterial blood pressure. To exploit redundant sources, we designed appropriate solutions to learn accurate monosource rules from data coming from each source separately. If the sources are complementary, a naive multisource learning method that consists in aggregating the data of all the sources and then learn globally from the whole set of data with a learning bias as few restrictive as possible, is a first solution to produce rules that contain relationships between events occurring on the different sources. However, this technique is not efficient and the huge search space associated with the set of possible solutions for the multisource problem leads to unsatisfactory solutions when using a relational learner. To solve those dimensionality problems, we have proposed a new multisource learning method that uses monosource rules to bias automatically and efficiently a new learning process from the aggregated data.

The work has been done in the context of the Cepica project. The results show that the proposed method is much more efficient than learning directly from the aggregated data. Furthermore, it yields rules having better or equal accuracy than rules obtained by monosource learning. Besides, the learned rules have been transformed into chronicles and implemented in the cardiac monitoring system Calicot and tested with encouraging results on real noisy signal.