Génération des événements primitifs dans une plateforme de reconnaissance d'activités.

Narjes Ghrairi

Année universitaire : 2012/2013

REPUBLIQUE TUNISIENNE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR, DE LA RECHERCHE SCIENTIFIQUE ET DE LA TECHNOLOGIE
UNIVERSITE DE SOUSSE
ECOLE NATIONALE D'INGENIEURS DE SOUSSE

MEMOIRE
DE PROJET DE FIN D'ETUDES

Présenté en vue de l'obtention du diplôme
d'INGENIEUR EN GENIE INFORMATIQUE INDUSTRIELLE
Option : Systèmes de télécommunications embarquées

Par

Narjes GHRAIRI

Sujet

Génération des événements primitifs dans une plateforme de reconnaissance d'activités

Organisme d'accueil : **STARS, INRIA Sophia Antipolis-France**

Soutenu en **octobre 2013**

Dirigé par :
Annie RESSOUCHE, Chercheuse à l'Institut Nationale de Recherche en Informatique et Automatique, France.
Daniel GAFFE, Maitre de conférence à la Faculté des Sciences de l'Université de Nice Sophia-Antipolis, France.
Saoussen BEN JABRA, Maitre assistante à l'École Nationale d'Ingénieur de Sousse, Tunisie.

À mes parents,
pour leur patience et leurs encouragements.

Remerciements

Ce travail a été effectué au sein de l'équipe Spatio-Temporal Activity Recognition Systems (STARS) du centre de recherche Inria Sophia Antipolis - Méditerranée. Je voudrais remercier tout d'abord mes encadrants Madame Annie RESSOUCHE et Monsieur Daniel GAFFE pour leurs soutiens et leurs orientations judicieuses. Je suis très reconnaissante de m'avoir fait profiter de leurs expériences.

Je tiens également à exprimer ma gratitude et mon amitié à Madame Saoussen BEN JABRA, pour sa disponibilité constante et pour ses conseils donnés avec sérénité et patience pendant ce stage.

Je remercie tous les membres du jury pour l'attention et l'intérêt qu'ils ont manifestés en acceptant d'assumer la charge de discuter ce travail.

Je souhaiterais remercier tous le personnel administratif et technique de l'équipe STARS et tous ceux qui ont contribué à la réussite de ce travail.

Mes remerciements vont également à Campus de France et à INRIA pour leur soutien financier.

Pour finir, un grand merci mes parents, mon cher frère et ma chère sœur pour leur soutien et leurs sacrifices.

Table des matières

Liste des figures

Liste des tableaux

Liste des algorithmes

Résumé

Les systèmes de détection et d'analyse d'activités, deviennent, aujourd'hui, assez importants dans diverses domaines comme les applications de surveillance des espaces privés et publiques ainsi que les technologies intelligentes, de maintien et de suivi des personnes, utilisées essentiellement dans le domaine de santé. Ils se basent sur les outils de vision par ordinateur et les technologies intégrant l'utilisation intelligente de capteurs environnementaux.

Cette classe de systèmes que l'on considère, s'appuie sur un modèle synchrone. Les données écoutées proviennent de la plateforme de vision cognitive SUP du projet Stars ainsi que d'autres capteurs environnementaux. L'arrivée de ces signaux étant asynchrone, chacun a sa propre horloge. L'interfaçage avec un module synchrone pose des problèmes théoriques et techniques. L'objectif de cette étude est donc la conception d'une machine d'exécution qui calcule les instants synchrones d'évènements asynchrones.

Les tests de cette machine effectués sur des vrais exemples élaborés du centre hospitalier universitaire (CHU) de Nice, aboutit à des résultats encourageants, vu que la machine a réussi à calculer les bons instants logiques nécessaires pour le fonctionnement des moteurs synchrones de reconnaissance d'activités.

Mots clés : reconnaissance d'activités, asynchrone, plate-forme SUP, capteurs environnementaux, instant logique, système synchrone réactif.

Abstract

Detection and activity-analysis systems have become important in various applications such as public or private monitoring space as well as intelligent technologies of maintenance and monitoring of people, mainly used in the field of health care. These systems are based on tools for computer vision technologies and integrate the intelligent use of environmental sensors.

This type of systems is based on a synchronous model. The used data come from the cognitive vision platform SUP, developed in Stars team, and other environmental sensors. These signals come in an asynchronous mode, each has its own clock, and thus interfacing these entries with a synchronous module raises theoretical and technical problems. So, the objective of this study is to design a machine that could calculate the synchronous instants of asynchronous events.

The machine's tests executed on real examples developed at the University Hospital (CHU) of Nice have led to positive results, as the machine has been able to calculate the right logic moment necessary for the synchronous engines of activity recognition.

Keywords : activity recognition, asynchronous, SUP platform, environmental sensors, logic instant, reactive synchronous system

مُلَخَّصْ

تَعُدُّ أَنْظِمَةُ كَشْفِ وَ تَحْلِيلِ ٱلأَنْشِطَةِ فِي يَوْمِنَا هَذَا ذَاتِ أَهَمِيَّةٍ بَالِغَةٍ فِي عِدَّةِ مِجَالاَتٍ كَمُرَاقَبَةِ ٱلفَضَاءَاتِ ٱلعُمُومِيَّةِ وِ ٱلخَاصَّةِ إِضَافَةً إِلَى تَطْبِيقَاتِ ٱلتِّكْنُولُوجِيَةِ ٱلذَّكِيَّةِ ٱلمُسْتَعْمَلَةِ أَسَاسًا فِي مَجَالِ ٱلصِّحَّةِ كَمُسَاعَدَةِ وَ مُرَاقَبَةِ ذَوِي ٱلقُدُرَاتِ ٱلمَحْدُودَةِ. إِذْ تَسْتَنِدُ هَذِهِ ٱلأَنْظِمَةِ عَلَى أَدَوَاتِ مُرَاقَبَةٍ مَرْئِيَّةٍ وَ فِي ٱلآن ذَاتِهِ مُدْمَجَةٍ مَعَ أَجْهِزَةِ اسْتِشْعَارٍ ذَكِيَّةٍ.

يَسْتَنِدُ هَذَا ٱلنَوْعُ مِنَ ٱلأَنْظِمَةِ عَلَى نَمُوذَجٍ مُتَزَامِنٍ. عِلْمًا أَنَّ مَصْدَرَ ٱلمَعْلُومَاتِ مِنْ جِهَةٍ هُوَ بَرْمَجِيَّةٌ ٱلرُؤْيَةِ ٱلمَعْرِفِيَّةِ وَ أَجْهِزَةِ ٱلاسْتِشْعَارِ ٱلبِيئِيَّةِ مِنْ جِهَةٍ أُخْرَى، فَإِنَّ وُصُولَهَا سَيَكُونُ غَيْرُ مُتَزَامِنٍ. لِذَلِكَ فَإِنَّ ٱلتَّوَاصُلَ مَعَ تَقْنِيَةٍ تَعْمَلُ بِصِفَةٍ مُتَزَامِنَةٍ يَنْطَوِي حَتْمًا عَلَى أَبْحَاثَ نَظَرِيَّةٍ وَ أُخْرَى تَقْنِيَةٍ. إِذْ أَنَّ ٱلهَدَفُ مِنْ هَذِهِ ٱلدِرَاسَةِ هُوَ تَصْمِيمِ بَرْنَامَجٍ يَقُومُ بِحِسَابِ ٱللَّحَظَاتِ ٱلمُتَزَامِنَةِ لِأَحْدَاثٍ غَيرَ مُتَزَامِنَة.

ٱلإِخْتِبَارَاتِ ٱلتِي أُجْرِيَتْ لِهَذَا ٱلبَرْنَامَجِ بِاسْتِعْمَالِ أَمْثِلَةٍ حَقِيقِيَّةٍ أُخِذَتْ مِنْ مَرْكَزِ ٱلطِّبِّ ٱلجَامِعِي بِمَدِينَةِ نِيسْ أَعْطَتْ نَتَائِجَ مُشَجِعَّةٍ، كَمَا أَنَّ ٱلجِهَازَ كَانَ قَادِرًا عَلَى حِسَابِ ٱلمُدَّةِ ٱلأَزِمَةِ لِعَمَلِ مُحَرِكَاتِ مَعْرِفَةِ ٱلأَنْشِطَةِ ٱلمُتَزَامِنَةِ.

ٱلكَلِمَاتِ ٱلمَفَاتِيح : أَجْهِزَةُ ٱلإِسْتِشْعَارْ، لَحْظَةٌ مَنْطِقِيَّةٌ، جِهَازٌ مُتَزَامِنٌ ، ٱلتَّعَرُفُ عَلَى ٱلأَنْشِطَة.

Introduction générale

Les systèmes de vision cognitive sont généralement utilisés pour la reconnaissance d'activités. Leur fiabilité est reconnue pour fournir des informations précises sur le comportement des êtres humains, des animaux, ou/et des équipements, etc.

Ces systèmes de vision deviennent, aujourd'hui, de plus en plus importants dans diverses domaines comme les applications de surveillance des espaces privés et publiques ainsi que les technologies intelligentes, de maintien et de suivi des personnes, utilisées essentiellement dans le domaine de santé.

En premier lieu, les intérêts des systèmes de vidéo-surveillance sont très diverses en particulier la sécurité des personnes et la protection des biens constituent les utilisations majeures.

En deuxième lieu, les évolutions démographiques associées au vieillissement de la population conduisent à un changement dans la structure sociale et économique. En effet, nous prévoyons, selon l'étude de Robert-Bobée[22], une croissance soutenue de la population des personnes âgées de 60 ans et plus, passant de 10 actuellement à 22 millions en 2015. En conséquence, le nombre de personnes qui risquent de perdre leur indépendance et qui nécessitent des soins augmentera.

Ces exemples mettent en évidence l'importance de détecter les évènements anormaux avant qu'ils deviennent critiques pour surveiller, d'une façon continue, les personnes, et contrôler les espaces publiques.

Notre étude concerne un système permettant de détecter et d'analyser les activités, en s'appuyant seulement sur des outils de vision. Pour le rendre de plus en plus fiable, nous avons complété la vision par d'autres capteurs. En effet, la reconnaissance de l'activité automatique devient plus réaliste, en attachant des capteurs sur différents objets, des lieux ou le corps humain. Les activités peuvent être de cette façon suivies et surveillées en permanence.

Nous considérons dans cette étude un système de reconnaissance d'activités s'appuyant sur un modèle synchrone. Les données proviennent de la plateforme de vision cognitive SUP (Scene Understading Platform) du projet Stars et d'autres capteurs environnementaux. Ils sont asynchrones dans le sens oú chacun a son propre horloge et qu'il faut interfacer ce module avec d'autres modules synchrones. L'objectif de cette étude est la conception d'une machine d'exécution qui construit les instants synchrones d'évènements asynchrones.

Le mémoire est organisé de la manière suivante :

- Nous présentons l'équipe-projet d'accueil, dans le chapitre 1 ;
- Dans le chapitre 2, nous étudions, d'une part, les concepts de la reconnaissance d'activités en présentant des travaux existants. D'autre part, nous introduisons les différents aspects de la programmation synchrone des systèmes réactifs ;

- Dans le chapitre 3, nous présentons la machine d'exécution proposée, la spécification, la conception et la mise en œuvre ;
- Au niveau du chapitre 4, nous évaluons nos solutions, nous testons la machine d'exécution proposée en utilisant un ensemble de scénarios réels et nous présentons des perspectives à court terme et à long terme ;
- Enfin, nous concluons notre travail, en résumant les contributions de ce stage.

CHAPITRE 1

Présentation du laboratoire d'accueil

1.1 INRIA

L'Institut National de Recherche en Informatique et Automatique [1], Inria, est un établissement public de recherche à caractère scientifique et technologique (EPST) sous la double tutelle des ministères en charge de la recherche et de l'industrie, il a pour mission de produire une recherche de qualité en informatiques, en mathématiques, en sciences du numérique et de transférer cette connaissance dans le tissu industriel.

Créé en 1967 à Rocquencourt, INRIA, s'est depuis étendue et dispose à présent de 8 sites en France (Rocquencourt, Rennes, Sophia Antipolis, Nancy, Grenoble, Bordeaux, Lille et Saclay).

Il emploie plus de 4000 personnes, dont près de 3500 scientifiques regroupés en 171 équipes-projets, et dispose d'un budget annuel de 250 millions d'euros.

1.2 INRIA Sophia Antipolis

Le centre de recherche Inria Sophia Antipolis-Méditerrané [2] est présent sur la Technopole depuis 1981.

En s'impliquant dans différents réseaux, les équipes du centre poursuivent leurs engagements en faveur du développement et de l'innovation dans leurs domaines d'expertise, notamment la création de richesse et d'emplois sur le territoire dans les sciences du numérique en menant des recherches dans des domaines aussi diversifiés que les mathématiques appliquées, l'algorithmique, les langages de programmation, les réseaux et les systèmes distribués, ainsi que l'application des technologies de l'information aux sciences de la vie et à l'environnement.

1. http://www.inria.fr/
2. http://www.inria.fr/centre/sophia

1.3 Équipe-projet d'accueil : STARS

L'équipe de recherche STARS [3] (Spatio-Temporal Activity Recognition Systems) se focalise sur la conception et le développement des systèmes pour la reconnaissance d'activités.

Cette recherche s'appuie notamment sur l'interprétation sémantique et en temps réel des scènes dynamiques contrôlées par des capteurs.

Plus généralement, les buts de l'équipe couvrent entre autres ces différents points :

- La reconnaissance complète (la perception, la compréhension et l'apprentissage) allant de l'analyse du signal de bas niveau à la description sémantique de ce qui se passe dans une scène donnée contrôlée par des caméras de vidéo et éventuellement d'autres capteurs environnementaux ;
- L'élaboration d'une méthodologie de conception des systèmes de reconnaissance des activités en assurant la généricité, la modularité, la réutilisabilité, l'extensibilité, la fiabilité et la maintenabilité.

3. http://team.inria.fr/stars

CHAPITRE 2

Contexte général

2.1 Introduction

Dans ce chapitre nous allons présenter le contexte de cette étude, sans oublier un état de l'art indispensable à la bonne compréhension de cette problématique. Seront abordés les concepts de la reconnaissance d'activités, leurs historiques, leurs domaines d'application, et les différentes approches sur lesquelles nous nous basons (cognitive, capteurs, multi-capteurs). Dans une autre partie, nous aborderons l'étude des systèmes réactifs synchrones et leur fonctionnement.

2.2 Contexte du projet

Les techniques de la reconnaissance d'activités se sont améliorées pour répondre à la demande croissante de confort (thermique, acoustique, visuel, etc.), de sécurité et de simplicité d'utilisation des équipements dans la vie quotidienne tout en assurant l'hygiène et la sureté. Ainsi le vieillissement de la population exige d'offrir des logements adaptés aux besoins des personnes âgées vivant seules et garantissant le soutien et le maintien de l'ouverture à l'extérieur.

Pour rendre ces techniques effectives, une solution est de les intégrer dans des systèmes réactifs qui vérifient l'hypothèse du modèle synchrone (voir 2.4.3).

Ce travail a été mené dans l'équipe STARS d'INRIA Sophia Antipolis, en France. STARS se concentre sur deux domaines d'application : la sécurité/sureté et la santé. Elle vise à reconnaitre les activités humaines ou des objets mobiles pour des applications de soins de santé ou de vidéo-surveillance. Dans cette étude, nous collaborons avec le centre hospitalier universitaire de Nice afin d'effectuer des expériences en utilisant des données réelles comme la surveillance des activités des personnes âgées.

2.3 Cahier des charges

La planification de notre travail respectent ces aspects théoriques et pratiques du développement :

- Étudier des modèles synchrones et se familiariser avec les systèmes de reconnaissance d'activités ;
- Interfacer les systèmes asynchrones / synchrones ;
- Se familiariser avec la plate-forme SUP et son fonctionnement ;
- Améliorer la plate-forme SUP avec un module de génération d'événements asynchrones et les adapter avec les moteurs de reconnaissance d'activités synchrones.

Nous avons adopté un planning de travail donné par la figure 2.1.

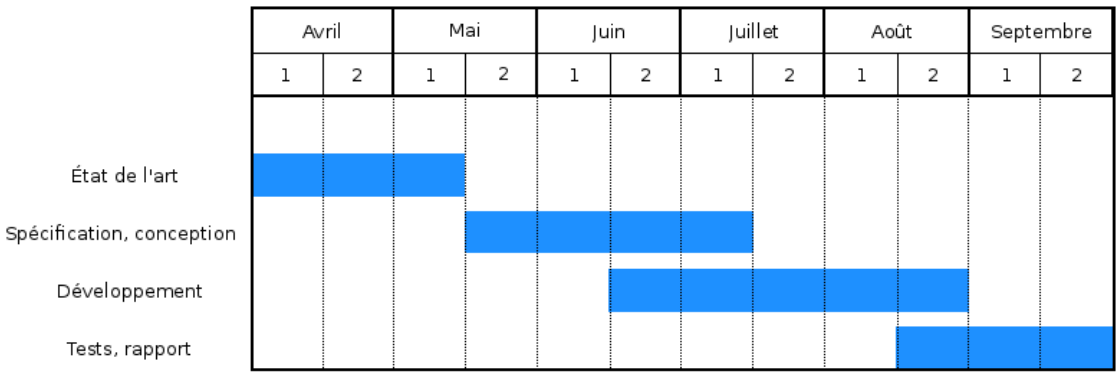


Figure 2.1 — Planning du stage.

2.4 État de l'art

Nous présentons dans cette section d'une part le concept de la reconnaissance d'activités, d'autre part l'approche synchrone.

2.4.1 Reconnaissance d'activités

2.4.1.1 Concept de la reconnaissance d'activités

La reconnaissance d'activités, humaines, animales, et des objets mobiles dans un espace donné, nécessite l'interaction des systèmes informatiques vis-à-vis de leurs environnements.

En fait, cette notion est basée sur le concept de vision par ordinateur [1], une branche de l'intelligence artificielle, permettant à une machine d'intégrer des informations lorsqu'on la connecte à un ou plusieurs capteurs (caméra, kinect, capteurs environnementaux, ect).

D'ailleurs, l'installation des capteurs multimodaux, dans un environnent donné, permet à ce dernier d'être surveillé en fournissant des informations précises. Le traitement de ces informations, donne un rapport de toutes les activités des personnes ou des objets mobiles dans cet environnent.

1. http://perception.inrialpes.fr/people/Horaud/livre-hermes.html

La reconnaissance de l'activité peut être classifiée en trois grandes approches, à savoir la reconnaissance de l'activité basée sur la vision, la reconnaissance de l'activité à base de capteurs et une troisième comme une fusion des deux premiers dite multi-capteurs.

La reconnaissance de l'activité est applicable dans diverses domaines. Cependant, une application de la reconnaissance d'activité pour un certain besoin reste toujours spécifique, et n'est pas utilisable pour d'autres. Parmi les domaines d'application nous pouvons citer [9] :

- La surveillance intelligente : Détection et suivi d'une personne donnée tout en identifiant son visage en utilisant plusieurs caméras. Cette détection peut même s'enrichir du comportement voire des intentions de la personne ;
- La réalité virtuelle et/ou la réalité augmentée : Nous considérons ici les mondes virtuels interactifs qui récupèrent les gestes, les poses corporelles et les expressions faciales. Ainsi, les systèmes de reconnaissance deviennent de plus en plus intégrés aux jeux vidéo, aux studios virtuels, à la conception d'animations corporelles et la téléconférence ;
- Les interfaces utilisateur avancées : Parmi les applications nous pouvons citer les traducteurs linguistiques intelligents, les applications de pilotage de gestes en suivant les squelettes et aussi les systèmes de contrôle dans des environnements bruyants comme les usines ou les aéroports ;
- L'analyse du mouvement : Généralement utilisée pour les séquences vidéo des exercices sportifs dont les formations personnalisées (golf, tennis, etc.), les chorégraphies de danse et de ballet et les études cliniques en orthopédie ;
- Le codage des images : À l'instar des visiophones, nous pouvons suivre les visages dans les images et les coder d'une façon plus détaillée avec une compression efficace des vidéos.

Revenons maintenant en détail sur les différentes approches de reconnaissance d'activités dans les paragraphes suivants, tout en présentant quelques travaux existants.

2.4.1.2 Reconnaissance d'activités basée sur des approches cognitives

Les recherches effectuées sur les applications de vidéo-surveillance, des communications multimédias et des diagnostics médicaux comptent, dans la plus importante partie, sur la reconnaissance des activités, en se basant sur des séquences vidéo [20].

En fait, dans la littérature, il existe des travaux de détection des évènements de Zelnik Manor et Irani [26] et de modélisation de l'activité humaine de Ivanov et Bobick en 2000 [25], et de Chowdhury et Chellappa en 2003 [1]. Des telles recherches reposent sur des méthodes différentes, parmi lesquelles :

- Les chaines de Markov : des méthodes à base de graphes, autrement dit, des machines d'états finies stochastiques modélisant les activités en tenant compte des probabilités de transition entre leurs états pour modéliser des gestes plus ou moins complexes. De plus, elles traitent les interactions entre plusieurs objets mobiles [18] ;
- Les réseaux de neurones : S'appuyant sur un modèle de comportements des neurones humains, cette classe de méthodes construit des graphes à poids stochastique. Ces réseaux diffèrent des précédents par le fait que ces poids sont dynamiques et sont remis à jour classiquement par des lois de rétro-propagation [18] ;
- Les réseaux bayésiens : Ce sont des systèmes qui utilisent classiquement des entrées vidéo, et travaillent avec des probabilités et combinent un cadre d'estimation bayésienne séquentielle du suivi des cibles données [16], en fait, un réseau bayésien est un graphe servant à modéliser une partie du monde réel en fonction des différents états possibles des éléments de ce monde (personnes, biens, etc.) et l'influence d'un état sur les autres.

2.4.1.3 Reconnaissance d'activités basée sur la technologie des capteurs

L'utilisation des capteurs est une approche de plus en plus utilisée. Dans la littérature nous trouvons plusieurs travaux qui cherchent à améliorer cette technologie, parmi lesquels :

- Les recherches de Guralnik[23] décrivent des méthodes de collecte de données d'un ensemble de capteurs installés dans un environnement spécifique, en définissant des algorithmes d'apprentissage pour reconnaître les comportements des personnes dans cet environnement. Cependant, la seule utilisation des détecteurs de mouvements est insuffisante pour déduire des activités spécifiques avec une grande précision ;
- L'étude de Kern et al.[17] a abouti à une implémentation d'une plate-forme matérielle équipée d'accéléromètres. Malgré leur effort, les résultats présentés montrent que seulement la reconnaissance des activités simples est possible, notamment assis, debout, marchant ;
- Une autre initiative est venue de l'université d'Aarhus en Denmark[14] en 2004. Mais, ils ont conclu que la capture fiable des activités nécessitent des infrastructures et des traitements complexes.

2.4.2 Reconnaissance d'activités basée sur l'approche multi-capteurs

2.4.2.1 Présentation de l'approche multi-capteurs

D'après l'étude effectuée par Elmenreich [7], un capteur physique présente un certain nombre de problèmes parmi lesquels :

- La perte du capteur : Dans ce cas, nous risquons de perdre l'information des cibles surveillées ;
- La couverture spatiale limitée : Un capteur couvre seulement une zone restreinte ;
- La couverture temporelle limitée : La fréquence de production de mesures est faible et insuffisante ;
- L'imprécision : manque de précision physique du capteur ;
- L'incertitude : peut survenir lorsque le capteur ne mesure pas des attributs pertinents.

L'utilisation d'un ensemble de capteurs et la combinaison de leurs données est un véritable bénéfice, en termes de précision et de robustesse, qui limite ces problèmes et leurs conséquences.

2.4.2.2 Avantages de l'approche multi-capteurs

Le but fondamental de l'utilisation de plusieurs capteurs, est à la fois, de fournir des informations sur l'environnement, intéressantes pour contrôler son fonctionnement. Dans ce cas, l'information devient plus précise, tout en gagnant du temps et par conséquent du cout.

Déjà le gain est obtenu, ici, en fusionnant les informations fournies par un groupe de capteurs hétérogènes chacun avec son degré de confiance, ce qui permet de garantir un fonctionnement sûr même en mode dégradé [12]. De cette façon un système à multiples capteurs reste toujours plus fiable et robuste.

Pour un système conçu de façon traditionnelle, les flux de données restent imprécis, ambigüs et incomplets. Par contre la fusion de capteurs a l'avantage de réduire la complexité du système, car moins d'informations issues des capteurs sont présentés au système mais leur fiabilité est plus grande[7].

2.4.2.3 Exemples de travaux basés sur l'approche de fusion de capteurs dans le domaine de santé

La vie à domicile des personnes âgées peut être plus confortable avec les nouvelles technologie : de télésurveillance médicale, de sécurité à domicile, d'assistance aux personnes âgées. Toutes ces technologies de téléassistance permettent de maintenir le lien social.

Listons, quelques travaux de recherche dans ce domaine :

- La fusion Multi-capteurs [24] : C'est une étude qui cherche à combiner les données des capteurs (caméras vidéo et capteurs environnementaux) hétérogènes pour la reconnaissance des activités des personnes âgées à domicile pour contrôler l'interaction des personnes avec leurs environnements. Mais, cette étude n'a pas résolu les problèmes de détection des activités multitâches et multi-personnes ;
- Le projet GERHOME [11] : l'objectif de ce projet est de concevoir et d'expérimenter des solutions techniques supportant des services d'aide, des personnes âgées, au maintien à domicile, en utilisant des technologies domotiques intelligentes pour assurer l'autonomie, le confort de vie, la sécurité, la surveillance et l'assistance à domicile. Dont le but de réduire les risques d'accidents domestiques (risques de chutes, de brulures, etc.), garder le lien avec les membres de la famille, l'entourage, le médecin, ainsi que le suivi médical et la gestion de l'urgence. Ce laboratoire, cherche encore des méthode plus intelligentes servant à caractériser l'état de fragilité à partir de l'interprétation de données collectées ;
- L'habitat intelligent [15] : C'est une étude qui cherche à reconnaitre les comportements d'une personne âgée vivant seule dans un "habitat intelligent pour la santé". Le travail propose une méthode de traitement de données provenant des capteurs infrarouges passifs installés dans un habitat intelligent, afin de reconnaitre les activités de la vie quotidienne réalisées par la personne âgée dans une journée, et de suivre l'évolution de son état d'autonomie. Cette méthode a été implémentée en Matlab et appliquée à des données réelles provenant de ces habitats occupés par des personnes âgées vivant seules, mais les traitement des signaux et la détermination des variables pertinentes servant à la reconnaissance reste encore expérimentales, ainsi , cette recherche est orientée seulement pour les personnes âgées et n'est plus valable pour d'autres besoins.

2.4.3 Programmation réactive synchrone

2.4.3.1 Systèmes réactifs

Les systèmes réactifs ou systèmes réflexes sont des systèmes dont le comportement est décrit par un ensemble d'états et de transitions, autrement dit, par une séquence de réactions à des stimuli [3].

Ces derniers sont externes au système : l'environnement est le maitre de l'interaction. Donc le rôle du système est de réagir, régulièrement, aux stimuli externes en produisant des réponses adaptées.

Comme la figure 2.2 le montre les réactions *Si* aux événements *Ei* sont générées par des fonctions *Fi* du système qui peuvent chacune être soumises à des contraintes temporelles *di* imposées par l'environnement. Nous parlons alors de systèmes réactifs temps réel.

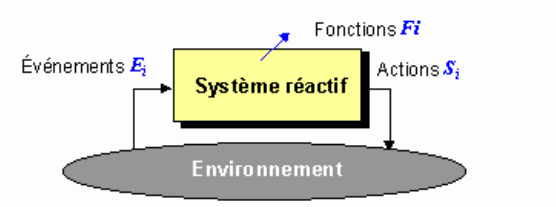


Figure 2.2 — Architecture générale d'un système réactif.

Ces systèmes diffèrent des systèmes transformationnels (par exemple les filtres numériques) par le fait que leur vitesse de réaction est imposée par l'environnement. Soit l'environnement ne peut pas attendre (système temps réel), soit la non prise en compte d'un évènement peut entrainer des conséquences très graves (arrêt d'urgence) impliquant la vie des personnes.

Des exemples de tels systèmes incluent le contrôle de processus industriels, les surveillances médicales, les systèmes embarqués, les pilotes de dispositifs, le traitement du signal, les contrôleurs de centrales nucléaires, de systèmes de vol avion, etc.

Les caractéristiques essentielles des systèmes réactifs sont les suivantes :

- Criticité : C'est une discipline qui vise à prévenir les risques de réaction non désirée et à garantir les réactions voulues ;
- Parallélisme : De nombreux systèmes sont conçus comme un ensemble de sous-système qui peuvent être concurrents et dont la bonne coopération est indispensable ;
- Déterminisme : Un système réactif détermine une séquence de signaux de sortie à partir d'une séquence de signaux d'entrée de manière unique ;
- Robustesse : La capacité du système de garder un comportement correct malgré la défaillance de certains capteurs.

2.4.3.2 Programmation synchrone

2.4.3.2.1 Présentation de la Programmation synchrone

La programmation des systèmes réactifs est mal résolue par les outils de programmation classiques. Par exemple, les systèmes à base d'automates communicants, induisent beaucoup d'indéterminisme à cause de l'asynchronisme des communications. De même la gestion de variables partagées entre plusieurs processus parallèles peut devenir rapidement indéterministe malgré les mécanismes de protection mis en place par les langages de programmation.

Dans ce cadre, nous pouvons citer quelques langages tel que OCCAM [2] basé sur le modèle DSP (Communicating Sequential Processes) de Tony Hoare et Ada[1980] [3], SDL (Specification and Description Language) qui utilise des files d'attente inspirées par le modèle des réseaux de Petri.

Prenons l'exemple d'une activité définie par un évènement a en parallèle avec un autre évènement b, deux mis en œuvre sont possibles : un a puis b ou b puis a. Si les deux évènements touchent la même variable par exemple :

$$x \leftarrow 3 : \text{évènement } a$$
$$y \leftarrow x + 1 : \text{évènement } b$$

Le système n'est plus déterministe car nous ne connaissons pas l'entrelacement des processus choisis.

D'autre part, dans les langages synchrones basés sur le principe de simultanéité, cette activité sera implémentée comme un fusion ab, et le compilateur résoudra le problème ou rejettera le programme [8][5].

2. Occam est un langage de programmation parallèle, développé par David May, en 1983 pour programmer les transputeurs

3. Ada est un langage de programmation de CII-Honeywell Bull qui résout le cahier des charges proposé par le département de la Défense des États-Unis

2.4.3.2.2 Fondements

- L'hypothèse synchrone : Les langages synchrones reposent sur l'hypothèse synchrone forte qui suppose une échelle de temps logique discrète faite d'instants indépendants. En fait ces instants correspondent aux réactions du système définis par l'ensemble des évènements d'entrée, de sortie et internes au système. La réaction complète et la diffusion des informations sont instantanées et assurent le déterminisme.
  Dans une réaction, les signaux d'entrée sont prêts, des calculs internes sont exécutés et les données de contrôle sont propagées jusqu'à ce que les valeurs de sortie soient calculées. La réaction est dite "atomique" et a une durée nulle, ceci est schématisé par la figure 2.3.

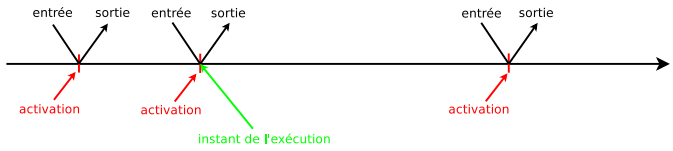


Figure 2.3 — Instants d'exécution et l'activation des systèmes synchrones réactifs.

- Les signaux : Dans un instant, un signal doit être soit présent ou soit absent et peut porter une valeur. Ce signal est diffusé à toutes les entités synchrones parallèles même celles qui n'y sont pas sensibles. Les signaux internes peuvent avoir une localité (cas de l'instruction "signal" d'Esterel ou du constructeur "let...tel" de Lustre).
- Les conditions d'activation et les horloges : Chaque signal externe ou généré en interne peut être considéré comme une nouvelle horloge. Autrement dit, les signaux sont des entités de contrôle, responsables de l'activation ou non de sous-parties du système.

2.4.3.3 Exemple introductif de système synchrone réactif

Comme étant un exemple introductif, prenons un système d'acquittement définit par la machine d'état donné par la figure 2.4.

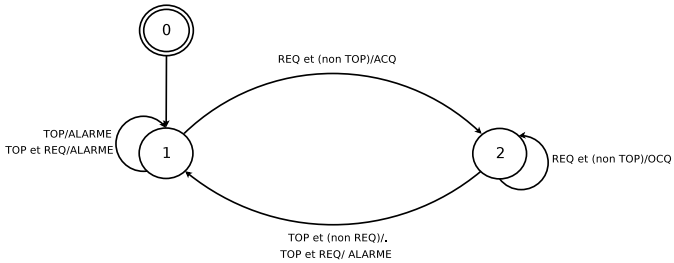


Figure 2.4 — Machine d'états de l'exemple : système d'acquittement.

Ce système fonctionne de telle façon qu'entre deux signaux TOP, à la réception du premier signal REQ (demande), le système acquitte la demande en envoyant un signal ACQ. Ensuite, à la réception des demandes suivantes, il signale que c'est occupé en envoyant le signal OQP, et si entre deux TOP, il n'y a pas de demande, le système considère que c'est une situation non souhaitée et déclenche le signal ALARME. De même, si une demande arrive en même temps avec un TOP, le système considère que c'est une situation non souhaitée, dans ce cas, il déclenche aussi le signal ALARME. Enfin, l'initialisation du système est considérée comme un premier TOP.

La figure 2.5 schématise quelques instances de l'exécution de ce système d'acquittement.

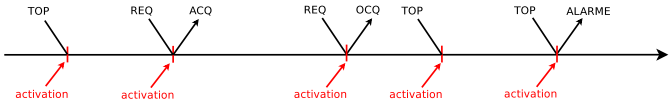


Figure 2.5 — Instances d'exécution de l'exemple : système d'acquittement.

2.4.3.4 Exemples de langages synchrones

Nous présentons dans ce paragraphe trois exemples de langages synchrones : Esterel, SYNCCHARTS et Lustre/SCADE.

2.4.3.4.1 Esterel

Esterel [1980][10] est le premier langage de programmation, de type synchrone réactif, conçu par un groupe dirigé par Gérard Berry [4]. Il a été créé par le centre de mathématiques appliquées (CMA) de l'école des Mines de Paris à Sophia-Antipolis, et par l'INRIA de Sophia Antipolis.

La dernière version du compilateur *v7* a étè développée au sein de l'entreprise Esterel-Technologies [5].

Ce langage est d'une part impératif, c'est à dire il décrit les opérations en termes de séquences d'instructions exécutées par la machine pour modifier la réaction globale du système, et d'autre part exprime d'une façon assez simple les notions de parallélisme et de préemption.

En tant que langage synchrone réactif, Esterel est :

- Séquentiel ;
- Réagit infiniment vite par rapport à l'environnement (hypothèse synchrone forte) ;
- Le parallélisme lui permet de gérer plusieurs tâches simultanément.

Le langage Esterel fournit au programmeur un ensemble de structures de contrôle de haut niveau pour la réalisation de systèmes réactifs synchrones, et il est constitué de deux types d'instructions :

- Les instructions de contrôle : permettant de définir les données de contrôle du système et de manipuler le temps logique ;
- Les instructions de données : permettant la manipulation de données, leurs types et les opérations élémentaires à travers des traitements externes non décrits dans le programme Esterel.

Exemple illustratif

À travers L'exemple suivant appelé ABRO, nous allons faire une brève description détaillée de l'ensemble des instructions dont dispose le langage [6].

```
module ABRO
    input A,B,R
    output O
    loop
        await A
      ||
        await B;
      emit O
    each R
end module
```

4. http://www-sop.inria.fr/members/Gerard.Berry/
5. http://www.esterel-technologies.com

La construction englobante *loop p each R* permet de relancer l'exécution de p à chaque instant où R est présent. Notons que R est préemptif pour tous les instants : l'occurrence de R interdit l'exécution de p. Dans notre exemple le corps p correspond aux instructions :

```
   await A
||
   await B ;
emit O
```

La construction *await A || await B* permet d'attendre les signaux A et B en parallèle. Lorsque ces deux évènements, A et B, sont survenus et si l'évènement R n'a pas eu lieu, l'émission du signal O est exécutée en séquence (;).

2.4.3.4.2 SYNCCHARTS

SyncCharts est un outil graphique dédié à la modélisation des systèmes réactifs synchrone. Ce langage a été créé par le professeur Charles André [6] en 1996 [2].

SyncCharts est l'acronyme de Synchronous Charts. Ils s'inspirent de nombreuses caractéristiques de diagrammes d'états, mais offrent une grande richesse au niveau des préemptions et ils bénéficient d'une sémantique plus stricte basée sur celle du langage Esterel, et ils s'approchent également d'Argos [7] de point de vue sémantique[3].

Les SyncCharts permettent la spécification du comportement réactif, ainsi que la programmation synchrone des systèmes réactifs tout en assurant la hiérarchie, la simultanéité, la communication, et la préemption entre les différents composants.

Exemple illustratif

L'exemple présenté par la figure 2.6, est un exemple type de SyncCharts. En effet, il s'agit d'un contrôleur d'occurrences synchronisées. Chaque fois les signaux A et B sont présents le signal AB est émis comme signal de sortie. D'une part, A et B peuvent être émis dans n'importe quel ordre ou bien en même temps. L'occurrence de A ou B déclenche un timer qui réinitialise le système au bout de 2 T à moins que le signal *Inhibit* soit présent. Le système peut également être réinitialisé par *Reset* à n'importe quel instant [4].

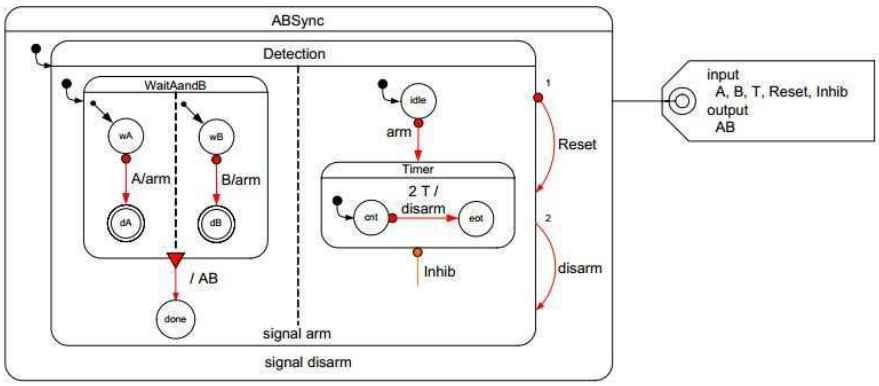


Figure 2.6 — Exemple de syncChart.

6. http://www-sop.inria.fr/members/Charles.Andre/
7. Un langage graphique pour la conception, la description et la validation des système réactifs

Instances / Signaux	i0	i1	i2	i3	...
A	false	false	true	false	
neverA	true	true	false	false	

Tableau 2.1 — Tableau des instances d'exécution du programme de l'exemple Lustre.

2.4.3.4.3 Lustre/SCADE

Lustre [19] est un langage fonctionnel déclaratif manipulant de flots de données, bien adapter à la spécification d'applications dans le domaine du traitement du signal.

Ce langage est inventé par P. Caspi [8] et N. Halbwachs [9] vers 1984 à Grenoble.

Lustre a premièrement été développé par Verilog, ensuite, l'outil SCADE (vision graphique de Lustre) fut commercialisé par Esterel-Technologies et utilisé par Aerospatiale.

Tous les deux fournissent une spécification formelle et permettent la vérification de propriété des programmes par Model-Checking (méthode des observateurs [13]).

Contrairement aux langages impératifs qui décrivent la séquence des opérations à réaliser, Lustre/SCADE décrivent la façon dont les entrées sont transformées en sorties, nous disons qu'il s'agit d'un langage déclaratif.

Toute variable ou expression est représentée par une suite infinie de valeurs et est considérée comme une fonction du temps. Le temps dans ce cas reste toujours logique et discret : les informations sont diffusées en temps nul. Le compilateur se charge de trouver un ordre d'évaluation de tous les signaux à un instant logique donné.

Par conséquent, les variables sont associées à une horloge globale, et prennent des valeurs du premier jusqu'au énième cycle de cette horloge.

Exemple illustratif

Prenons ce simple exemple de programme Lustre [21] :

```
node Never(A : bool) returns (neverA : bool) ;
let
    neverA = not(A) -> not(A) and pre(neverA) ;
tel ;
```

Au premier instant, neverA est la négation de A. Aux instants suivants, si la valeur de A est vraie, la sortie est fausse. À partir de cet instant *neverA* sera toujours faux quelque soit l'entrée.

Des instances de son exécution sont élaborés dans le tableau 2.1.

2.5 Conclusion

Le but principal de ce chapitre fut de cadrer notre étude en présentant, dans une première partie, le concept de la reconnaissance d'activités, ses différents avantages et ses domaines d'application. Dans la partie suivante de ce chapitre, nous avons étudié les trois approches de la reconnaissance d'activités à savoir la vision par ordinateur et la technologie de capteurs mais l'accent fut mis surtout sur l'intérêt d'utiliser un système multi-capteurs. Enfin,

8. http://www-verimag.imag.fr/ caspi/
9. http://www-verimag.imag.fr/ halbwach/

nous avons abordé les systèmes synchrones réactifs et leurs rôles de modélisation des scénarios de reconnaissance des évènements du monde réel.

Comme présenté précédemment, notre objectif est de proposer une machine d'exécution qui traite des données, calcule leurs instants synchrones et les interface avec des outils synchrones.

Dans le chapitre suivant, nous allons détailler la mise en œuvre de cette machine proposée.

CHAPITRE 3

Spécification et mise en œuvre de la machine d'exéxution

3.1 Introduction

Comme nous avons vu dans le chapitre 2, l'objectif de la reconnaissance d'activités est de fournir des informations précises sur le comportement d'une cible contrôlée dans un environnement donné. Cette reconnaissance est basée sur un modèle des scénarios synchrones modélisant des évènements réels provenant d'une façon asynchrone de diverses sources. Dans ce cadre, nous proposons une machine d'exécution dont le fonctionnement permet de calculer l'instant logique de n'importe quel modèle synchrone en fonction des évènements donnés en entrée. Dans ce chapitre, nous détaillerons la réalisation de cette machine proposée, en illustrant les méthodes développées et en présentant les outils utilisés.

3.2 Architecture générale de la machine d'exécution

Dans cette étude, nous visons à interfacer les moteurs de reconnaissance d'activités synchrones avec les différentes sources d'informations. Ces sources peuvent être des caméras de vidéo et/ou des capteurs environnementaux pour surveiller l'interaction des cibles (personnes, objets mobiles, animaux, etc.) avec leurs environnements.

Comme il est décrit par la figure 3.1, les entrées de la machine proposée sont les données fournies par les différents capteurs (de vision et d'environnement) (voir 3.3.1). Ensuite, ces entrées passent par un pré-traitement (tri et filtre) avant d'être transférées à la machine d'exécution (voir 3.4.1).

Une fois les données filtrées et triées, la machine d'exécution s'appuie sur les signaux attendus ("*Awaited*") du système de reconnaissance d'activités pour calculer l'instant logique (voir 3.3.2). En effet, comme déjà mentionné dans le chapitre 2, paragraphe 2.4.3, un instant logique est atomique, et le système réactif synchrone doit réagir à un état (présence ou absence) d'un ensemble d'évènements externes qui constituent cet instant logique.

À la réception des signaux "*Awaited*", la machine déclenche le calcul de cet instant et envoie le résultat obtenu vers le système de reconnaissance d'activités. De cette façon la machine d'exécution reste en permanence et en interaction avec ce dernier (voir 3.12).

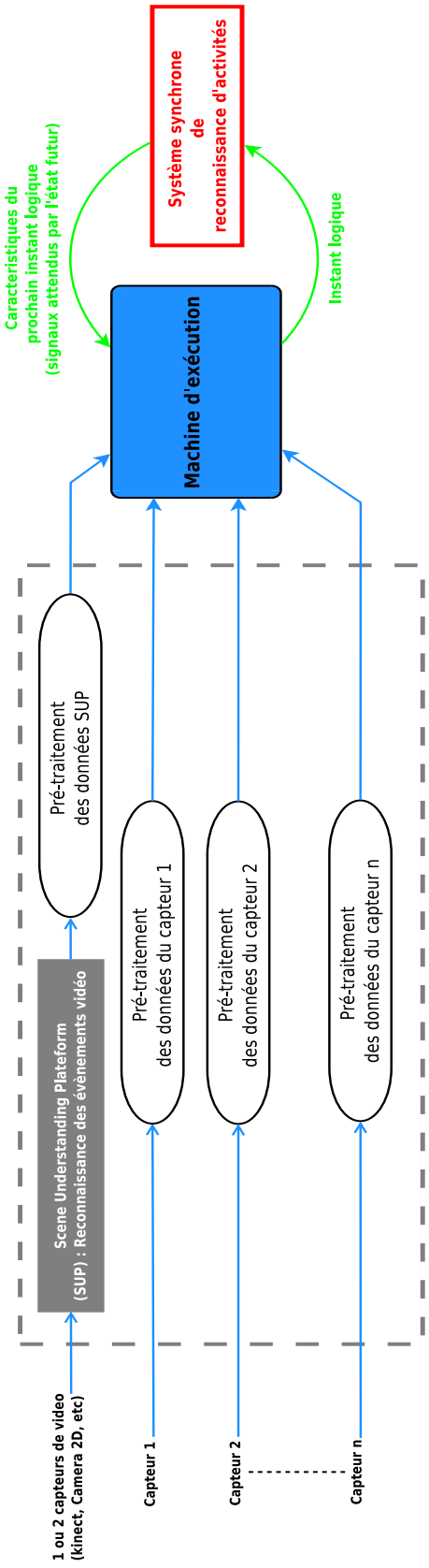


Figure 3.1 — Architecture générale de la machine d'exécution.

3.3 Environnement de la machine d'exécution

L'environnement de la machine d'exécution est composé de deux principales partie : les données des différentes sources (vidéo et environnementaux) et les signaux attendus ("*Awaited*") par le prochain état correspondants aux évènements attendus.

3.3.1 Différentes sources de données

Nous proposons deux sources de données : la plate-forme SUP et des capteurs environnementaux.

3.3.1.1 Plate-forme SUP : Scene Understanding Platform

3.3.1.1.1 Présentation de SUP

SUP (ou Scene Understanding Platform) est une plateforme, temps réel qui permet de percevoir, d'analyser et d'interpréter des scènes observées à travers un réseau de capteurs.

Une scène traitée peut contenir un certain nombre d'objets physiques de différents types (par exemple des personnes, des objets mobiles) en interaction les uns avec les autres ou avec leurs environnements.

Les capteurs sont généralement des caméras 2D ou caméras Kinect [1].

En fait, la plateforme SUP est une chaine d'algorithmes bien structurée comme la figure 3.2 le montre.

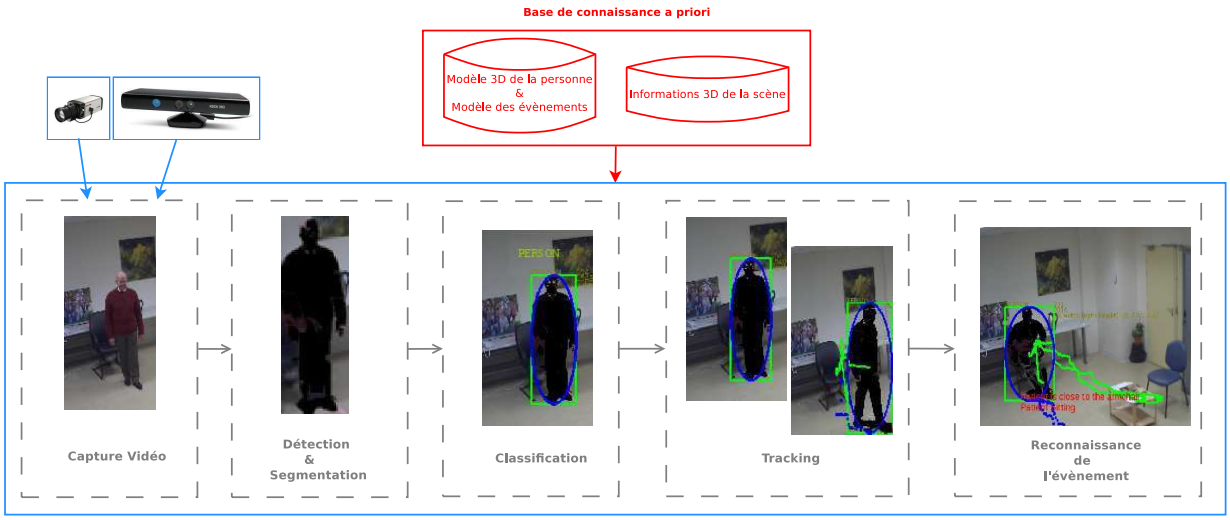


Figure 3.2 — Fonctionnement de la plate-forme SUP.

➤ La capture vidéo : Obtenue en général, à l'aide d'une ou deux caméras 2D ou d'une caméra Kinect. En fait une vidéo est une succession d'images qui défilent à un rythme fixe (FPS : Frame Per Seconde) pour

1. Kinect est un produit de Microsoft, il joue le rôle d'une caméra infra-rouge et d'un projecteur de points permettant d'obtenir une mesure de la profondeur

donner l'illusion du mouvement, chaque image est décomposée en lignes, chaque ligne étant une succession de points appelés pixels ;

- La détection et la segmentation : C'est un module qui détecte les objets mobiles (personnes, équipements mobiles, etc.), colore en noir les pixels correspondants et regroupe ces derniers dans un seul corps en mouvement ;
- La classification : C'est un algorithme capable de classifier et identifier l'objet mobile détecté par les algorithmes précédents (par exemple l'objet mobile est une personne, une voiture, un animal, etc.) ;
- Le tracking : Montre la trajectoire suivie par une personne ;
- La reconnaissance de l'activité : De cette façon, les algorithmes précédents mis en séquence permettent à la plate-forme de reconnaitre les activités de la scène contrôlée tout en affichant un modèle 3D et en générant des fichiers XML dans lesquelles les activités sont bien détaillées.

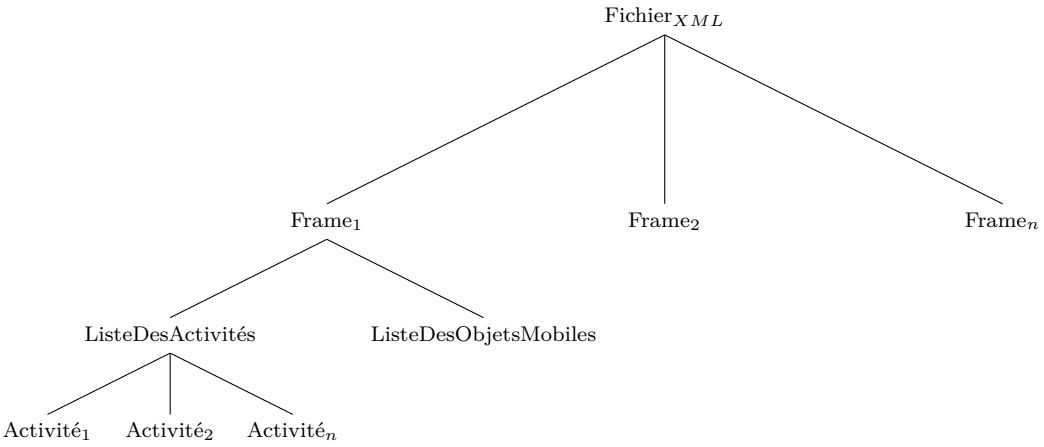
3.3.1.1.2 Données de la plate-forme SUP

Concernant la plate-forme SUP, deux types de données sont pertinents : les activités vidéo organisées dans un fichier XML et la période d'échantillonnage de fonctionnement de la plate-forme.

3.3.1.1.2.1 Évènements vidéo

La plate-forme SUP est une suite de modules servant à modéliser un environnement contrôlé (voir le paragraphe précédent). La plate-forme a comme sortie des fichiers XML dans lesquelles figurent tous les évènements détectés et nous présentons dans l'annexe B un exemple de ces fichiers.

L'arbre suivant schématise la structure des fichiers XML standards :



Dans cet arbre nous définissons :

- Une Frame : C'est l'unité d'une vidéo dans laquelle nous pouvons avoir un ensemble d'activités ou non ;
- Une liste des objets mobiles : C'est l'ensemble des objets mobiles autres que les personnes ;
- Une liste des activités : C'est la liste des évènements détectés par le capteur vidéo.

De même une activité vidéo est caractérisée par :

- Un identificateur : Une activité est identifiée par le numéro du frame où elle est apparue ;
- Un nom : C'est une description succincte de l'activité ;
- Un temps de début : C'est la date de commencement de l'activité en prenant l'année, le mois, le jour, l'heure, la minute, la seconde et la milliseconde ;

- Un temps de fin : C'est la date de la dernière détection de l'activité en prenant l'année, le mois, le jour, l'heure, la minute, la seconde et la milliseconde ;
- Une liste d'objets physiques : C'est l'ensemble des zones et des personnes choisies pour définir une activité.

À son tour un objet physique est défini par les éléments suivants :

- Un type : Les objets physiques ont des classes bien définies. Les plus intéressantes sont OBJECT pour les personnes et STATIC_ZONE pour les zones. Par exemple :
  <ActivityPhysicalObject **type="OBJECT"** ID="104" name="p1" dynamicObjectID="0"/>
  <ActivityPhysicalObject **type="STATIC_ZONE"** ID="2" name="exercisewalking" dynamicObjectID="0"/>

- Un identificateur : Etant donné que la même scène nous risquons d'identifier des objets de même type, pour les différencier les uns des autres, nous associons à chaque objet un identifiant unique. Par exemple :
  <ActivityPhysicalObject type="OBJECT" **ID="104"** name="p1" dynamicObjectID="0"/>
  <ActivityPhysicalObject type="OBJECT" **ID="105"** name="p1" dynamicObjectID="0"/>

- Un nom : Il s'agit d'un nom caractérisant un objet appartenant à une même classe. Par exemple :
  <ActivityPhysicalObject type="STATIC_ZONE" ID="104" **name="z1"** dynamicObjectID="0"/>
  <ActivityPhysicalObject type="STATIC_ZONE" ID="2" **name="exercisewalking"** dynamicObjectID="0"/>

3.3.1.1.2.2 Période d'échantillonnage de la plateforme SUP (Δt_{SUP})

Le Δt_{SUP} de la plate-forme SUP est fonction du nombre d'images traitées par seconde (FPS). Ce dernier dépend lui même du temps d'exécution des algorithmes cités dans le paragraphe 3.3.1.1.1 (dès la lecture des images jusqu'à la reconnaissance des évènements vidéo) et il dépend aussi du nombre d'objets mobiles dans la scène contrôlée. Si nous activons la séquence complète de traitement, nous obtenons 4 frames/seconde plus ou moins. Dans le cas idéal, où la plate-forme est assez rapide, FPS est égal à 15.

D'une façon générale, pour calculer le Δt_{SUP} nous avons appliqué la formule suivante :

$$\Delta t_{SUP}(en\ milliseconde) = \frac{1000}{FPS}$$

Nous calculons le tableau 3.1 Δt_{SUP} en fonction $N(images/seconde)$.

FPS	4	6	10	15
Δt_{SUP} (milliseconde)	250	167	100	67

Tableau 3.1 — Tableau de calcul du Δt_{SUP} en fonction de $N(images/seconde)$.

Nous allons voir dans la partie 3.4 comment le calcul de l'instant logique dépend de plusieurs variables parmi lesquels Δt_{SUP} de fonctionnement de la plate-forme SUP.

3.3.1.2 Capteurs environnementaux

Comme le montre la figure 3.3, un capteur est un dispositif convertissant une grandeur physique variable selon l'état observé en un signal numérique.

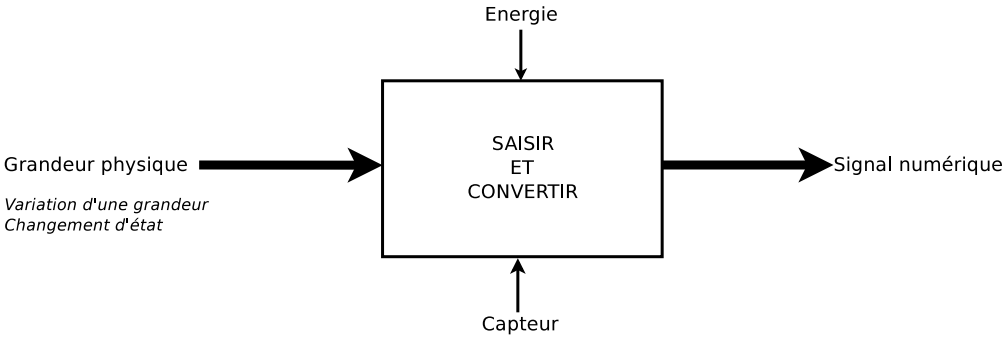


Figure 3.3 — Représentation fonctionnelle d'un capteur.

3.3.1.2.1 Classification des capteurs environnementaux

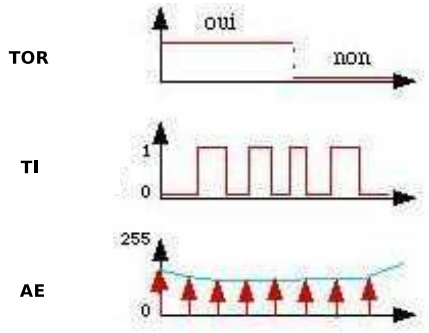


Figure 3.4 — Différentes formes des signaux des capteurs considérés.

Dans cette étude nous allons considérer trois grandes familles de capteurs différenciées par le type de signal qu'ils transmettent comme la figure 3.4 le montre :

- Capteurs à signal TOR "Tout Ou Rien" : Il reflète l'état d'un système et il ne peut avoir que deux états significatifs relatifs à la présence du signal. Ce type de montage est dit binaire dû au fait que la sortie du montage ne peut prendre que deux valeurs (ouvert/fermé, 1/0) ;
- Capteurs à signal en TI "Train d'Impulsion" : Chaque impulsion est l'image d'un changement d'état. Par exemple, un codeur incrémental donne un nombre fini et connu d'impulsions par tour ;
- Capteurs à signal AE "Analogique Échantillonné" : C'est l'image numérique d'un signal analogique sous la forme d'un «escalier» dû à la résolution du capteur.

3.3.1.2.2 Données des capteurs environnementaux

Concernant les capteurs environnementaux, deux types de données sont pertinentes : les évènements des capteurs et les périodes d'échantillonnage de fonctionnement de chaque capteur.

3.3.1.2.2.1 Évènements des capteurs

Nous appelons évènement environnemental chaque état et/ou un évènement détecté par des capteurs environnementaux (par exemple des capteurs de contact, les capteurs de pression, capteurs électriques, capteurs de présence et des capteurs d'eau, etc.) à l'exception des caméras vidéo.

Les capteurs environnementaux fournissent des données lorsqu'il y a un changement d'état. Par exemple, le capteur de contact détermine une ouverture et une fermeture d'un équipement (par exemple armoire, tiroir, réfrigérateur, etc.).

Pour notre approch, nous avons représenté dans les fichiers d'entrées les données des capteurs environnementaux avec une syntaxe proche de celle des évènements vidéo, enrichie et modifiée par le type du capteur (TOR, TI ou AE). Nous présentons dans l'annexe C un exemple de ces fichiers.

Pour tous les évènements des capteurs nous définissons :

- ➤ Un temps de début : C'est la date de commencement de l'état en prenant l'année, le moi, le jour, l'heure, la minute, la seconde et la milliseconde ;
- ➤ Un temps de fin : C'est la date de la dernière détection de l'état en prenant l'année, le moi, le jour, l'heure, la minute, la seconde et la milliseconde ;
- ➤ Un type du capteur : TOR, TI ou AE ;
- ➤ Un nom du capteur : qui va servir d'identifiant ;
- ➤ Un nom d'évènement : C'est une description succincte de l'évènement contrôlé ;
- ➤ Une valeur : La valeur du signal capturé.

Mais les évènements, des capteurs à signal analogique échantillonné (AE) différent des deux premiers types (TOR et TI). En effet, un signal analogique est significatif dans un certain intervalle limité par :

- ➤ Une valeur minimale ;
- ➤ Une valeur maximale.

Pour cela nous avons ajouté un champs, pour les capteurs qui ont ce type, dans lequel nous précisons les deux valeurs considérées.

3.3.1.2.2.2 Périodes d'échantillonnage des capteurs ($\Delta t_{capteur}$)

Le théorème de Shannon donne une condition sur la période d'échantillonnage $\Delta t_{capteur}$, nécessaire à la reconstruction d'un signal continu à partir de ses échantillons régulièrement espacés.

Soit le théorème : Nous ne perdons pas d'information en reconstruisant un signal à partir de ses échantillons si la fréquence d'échantillonnage est au moins égale à deux fois la plus élevée des fréquences attendues contenues dans le spectre du signal qu'on échantillonne :

$$F_e = \frac{1}{\Delta t_{capteur}} \geqslant 2 \times F_{max}$$

On se basant sur ce théorème, nous avons considéré que la période d'échantillonnage d'un capteur est le temps de réponse de référence de cet outil.

Nous allons voir dans la partie 3.4 que l'instant logique, lorsqu'on parle d'un système multi-capteurs, est une fonction des $\Delta t_{capteur}$ des capteurs donnés.

3.3.2 Signaux attendus par le prochain état : "*Awaited*"

La machine d'exécution peut a priori définir un instant logique à chaque occurrence d'un évènement attendu. En fait, il est beaucoup plus intéressant de construire de vrais instants significatifs. Pour cela la machine d'exécution peut tenir compte de la liste des signaux attendus pour le prochain état. Notons "*Awaited*" un tel signal,

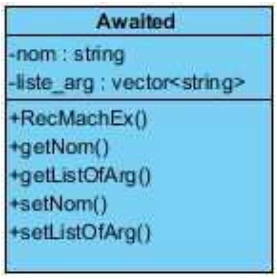


Figure 3.5 — Classe "*Awaited*" des signaux attendus par le prochain état.

tel que l'ensemble des "*Awaited*" donné à un cycle soit considéré comme un paramètre caractérisant un instant logique.

Dans notre approche, chaque signal "*Awaited*" est une instance de la classe donnée par la figure 3.5.

Cette classe présente deux attributs : le nom de l'évènement attendu et une liste d'arguments. Pour les signaux attendus des capteurs environnementaux, nous n'avons qu'un seul argument : la valeur du signal capturé, pour les signaux vidéo attendus de la plate-forme SUP ces arguments correspondent aux noms des objets physiques caractérisant les évènements vidéo attendus.

3.4 Développement de la machine d'exécution

D'un point de vue matériel, la machine d'exécution proposée est un ensemble d'algorithmes développés en langage C++ dans un environnement Linux, Fedora, compilés par l'outil GNU g++.

D'un point de vue algorithmique, nous avons organisé le développement en deux parties :

- Le pré-traitement ;
- Le fonctionnement de la machine d'exécution.

3.4.1 Pré-traitement de la machine d'exécution

Nous abordons pour les évènements vidéo de SUP et les évènements des capteurs une série de pré-traitements sur les données : une extraction, un tri et un filtre.

3.4.1.1 Extraction de données

3.4.1.1.1 Extraction de donnés de SUP

Pour extraire les données des fichiers XML, nous avons utilisé les outils GNU Flex/Bison :

- Flex : Construit un analyseur lexical à partir d'un ensemble de règles/actions décrites par des expressions régulières ;
- Bison : C'est un compilateur de compilateur décrit par un ensemble de règles grammaticales et d'actions sémantiques.

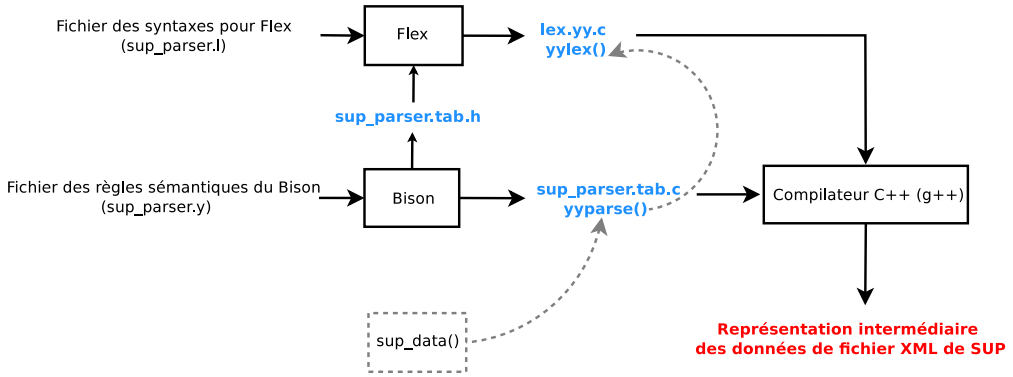


Figure 3.6 — Démarche Flex/Bison d'extraction des données d'un fichier sortie de SUP.

Comme le montre la figure 3.6, la fonction "*sup_data*()" responsable du pré-traitement des évènements vidéo de SUP, appelle la routine *yyparse*() pour faire la lecture du fichier XML dont les tags et les différents éléments décrits dans le fichier "*sup_parser.l*".

Le code C "*y.tab.c*" généré à partir de "*yacc sup_parser.y*", vérifie que le fichier suit bien la grammaire attendue. Prenons une portion du fichier XML de SUP dans laquelle nous marquons les champs à récupérer :

```
<?xml version="1.0" encoding="UTF-8"?>
<Cofriend version="3.00">
<SUVideoFrame frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54"
timeSec="40" timeMs="16">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="Person_Inside_Zone_ExerciseWalking" confidence="1" type="PrimitiveState"
AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54"
timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54"
timeSec="40" timeMs="16" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
</Cofriend>
```

En récupérant ces valeurs, nous construisons des objets, instances, de la classe "*SupEvent*" (voir figure 3.7) :

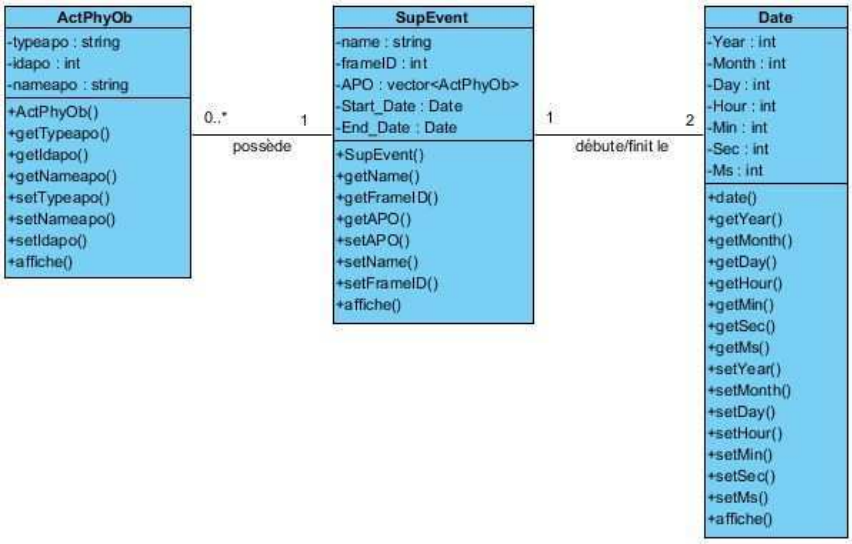


Figure 3.7 — Classe "*SupEvent*" des évènements vidéo de la plate-forme SUP.

3.4.1.1.2 Extraction des donnés des capteurs

Le format des fichiers des capteurs est plus simple que celui des fichiers XML de SUP. Prenons une portion du fichier d'un capteur environnemental dans laquelle nous marquons les champs à récupérer :

```
03 : 11 : 2011 : 14 : 34 : 15 : 56 / 03 : 11 : 2011 : 14 : 34 : 15 : 56 / Tor / sensor1 / event1 / 1
03 : 11 : 2011 : 14 : 34 : 30 : 65 / 03 : 11 : 2011 : 14 : 34 : 30 : 65 / Tor / sensor1 / event1 / 1
```

La routine "*sensor_data*()", se charge de faire le parsing de ces fichiers et permet de construire des objets, instances, classes "*TOR_TI_Event*" et "*AE_Event*" (voir figures 3.8 et 3.9) :

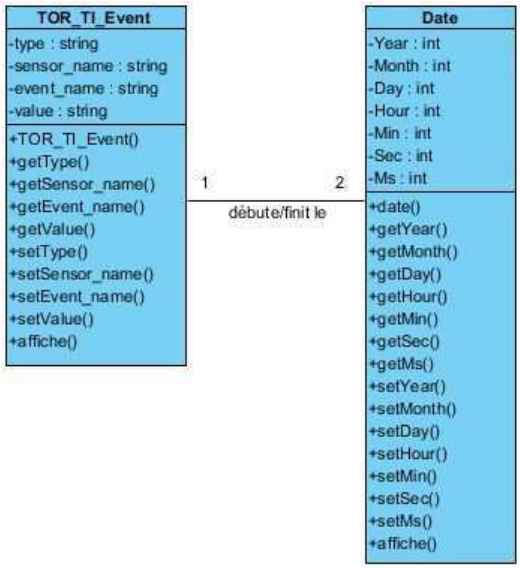


Figure 3.8 — Classe *TOR_TI_Event* des évènements des capteurs de type TOR ou TI.

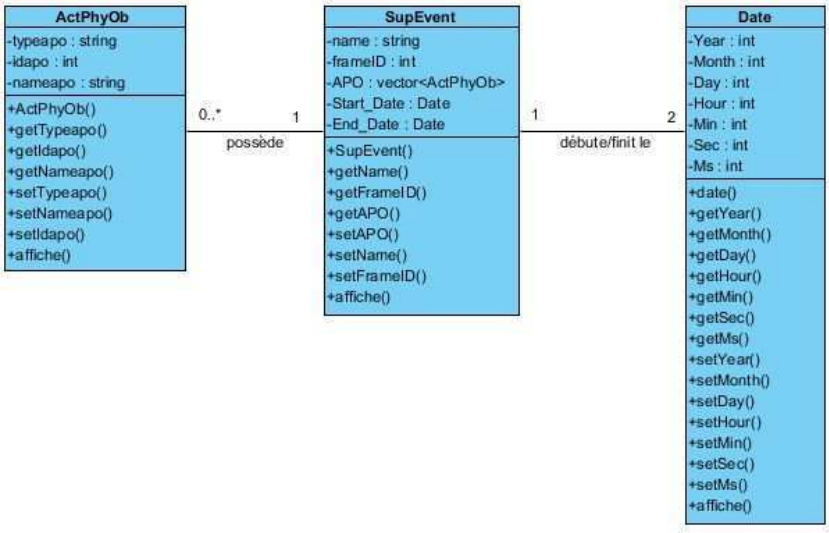


Figure 3.9 — Classe *AE_Event* des évènements des capteurs de type AE.

3.4.1.2 Tri des données

Puisque le temps logique est déduit du temps physique, pour diminuer les risques d'erreur, nous considérons des routines servant à trier les données de chaque source dans l'ordre croissant du temps physique car il peut arriver qu'un évènement retardé soit mal-ordonné dans un fichier de données.

3.4.1.3 Filtrage des données

Le transfert des évènements, vers les systèmes de reconnaissance d'activités, portant la même information pour des temps d'apparition très proches est inutile. De plus dans ce cas nous risquons de diminuer la vitesse de réaction de la machine d'exécution. Pour modérer ces risques, nous avons proposé des méthodes, servant à filtrer les données, qui diffèrent selon le type d'évènements donnés en entrée.

3.4.1.3.1 Filtrage des données de SUP

Pour les évènements vidéo de SUP, le filtre consiste à définir un paramètre générique noté "$paramFiltrage$" qui est égal à un certain nombre de frames. Comme le montre l'algorithme 1, le filtre élimine chaque réapparition du même évènement tel que son numéro de frame soit inférieur ou égal au numéro du frame de la première apparition en ajoutant le paramètre "$paramFiltrage$".

Nous pouvons augmenter (resp. diminuer) la valeur de ce paramètre si le nombre de FPS est grand (resp.

petit).

Algorithme 1 Algorithme de filtrage des données de SUP.

Entrées: vecteur v de données de SUP

Sorties: v filtré

pour it **de** 1 **jusqu'à** $taille(v)$ **faire**
  $existe \leftarrow faux$
  $i \leftarrow 1$
  tantque $i < it$ **et** $existe = faux$ **faire**
    si $v(i) = v(it)$ **alors**
      $existe \leftarrow vrai$
    finsi
    $i \leftarrow i + 1$
  fin tantque
  si $existe = faux$ **alors**
    $n \leftarrow FrameID(v(it))$
    pour $it1$ **de** $it + 1$ **jusqu'à** $taille(v)$ **faire**
      $f \leftarrow FrameID(it1)$
      si ($Start\_Date(it) = Start\_Date(it1)$ **et** $End\_Date(it) = End\_Date(it1)$ **et** $APO(it) = APO(it1)$ **et** $frameID(it) = frameID(it1)$ **et** $name(it) = name(it1)$) **alors**
        éliminer($v(it1)$)
      sinon si ($APO(it) = APO(it1)$ **et** $name(it) = name(it1)$ **et** $f <= n + paramFiltrage$ **et** $it1 <>$ taille(v)) **alors**
        éliminer($v(it1)$)
      sinon si ($APO(it) = APO(it1)$ **et** $name(it) = name(it1)$ **et** $v(it) = v(it1)$ **et** $f <= n + paramFiltrage$ **et** $it1 =$ taille(v)) **alors**
        éliminer($v(it1)$)
        $it1 \leftarrow it1 + 1$
      sinon si ($APO(it) = APO(it1)$ **et** $name(it) = name(it1)$ **et** $v(it) = v(it1)$ **et** $f = n + paramFiltrage + 1$) **alors**
        $it1 \leftarrow it1 + 1$
        $n \leftarrow FrameID(it1)$
      sinon
        $it1 \leftarrow it1 + 1$
      finsi
    fin pour
  finsi
fin pour

APO : *ActivityPhysicalObject*

Prenons un exemple d'exécution de cet algorithme sur un ensemble d'évènements identifiés par leurs numéros de frames où ils sont apparus. Pour cet exemple nous considérons que "*paramFiltrage*" est égal à 3.

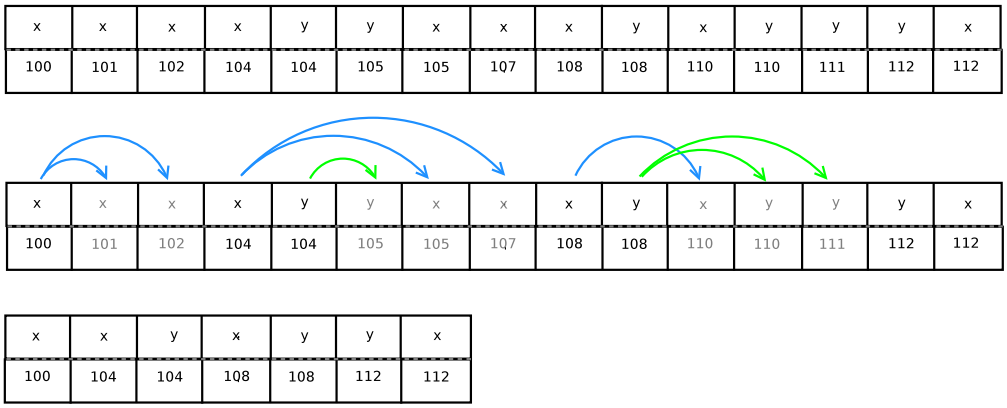


Figure 3.10 — Filtrage d'un ensemble d'évènements vidéo SUP ($paramFiltrage = 3$).

3.4.1.3.2 Filtrage des donnés des capteurs environnementaux

Comme déjà mentionné dans le paragraphe 3.3.1, nous considérons trois type de capteurs : TOR, TI et AE.

Pour les évènements TOR et AE, il est inutile de reprendre le même évènement pour deux temps successifs. De ce fait, le filtre des capteurs à signal TOR ou AE élimine toute réapparition d'un évènement pour deux temps successifs. Dans ce cas, nous proposons l'algorithme 2.

Algorithme 2 Algorithme de filtrage des données d'un capteur de type TOR ou AE.

Entrées: vecteur v de données d'un capteur de type TOR ou AE

Sorties: v filtré

pour it **de** 2 **jusqu'à** $taille(v)$ **faire**
  si $(type(it) = type(it-1)$ **et** $sensor\_name(it) = sensor\_name(it-1)$ **et** $event\_name(it)$ $= event\_name(it-1)$ **et** $value(it) = value(it-1))$ **alors**
    éliminer$(v(it))$
  finsi
fin pour

Pour les évènements TI, nous n'avons pas droit de les éliminer parce que chaque impulsion nous renseigne d'un changement de l'état.

3.4.2 Fonctionnement de la machine d'exécution

3.4.2.1 Calcul de l'instant logique : Définition standard

Le challenge est de trouver les temps réels de début et de fin qui caractérisent l'instant logique courant. Pour déterminer des valeurs, nous avons proposé les deux étapes suivantes :

➤ **Calcul du début du temps logique** : Comme le montre l'algorithme 3, pour localiser le début d'un instant logique, nous cherchons parmi l'ensemble des "*Awaited*", généré par le système de reconnaissance d'activités, lequel correspond à un évènements donné en entrée (statut : présent) et il doit être aussi le plus ancien possible du point de vue temps d'apparition.

Comme le montre la figure 3.11, nous disons qu'un signal "*Awaited*" correspond à un évènement vidéo de SUP s'ils ont les mêmes noms et s'il y a une correspondance entre les arguments du signal "*Awaited*" et les noms des objets physiques de cet évènement vidéo.

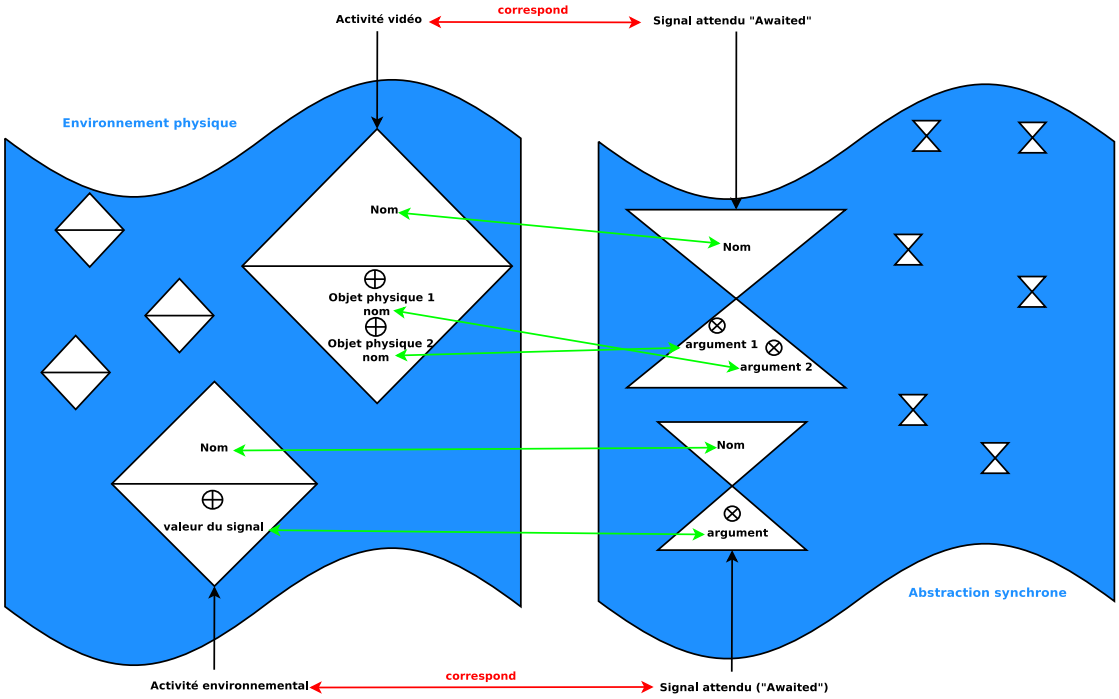


Figure 3.11 — Correspondances des activités vidéos et environnementales aux signaux attendus "Awaited".

De même nous disons qu'un signal "*Awaited*" correspond à un évènement de capteur environnemental s'ils ont les mêmes noms et si le seul argument du signal "*Awaited*" peut être mis en correspondance avec la valeur courante du signal capturé par ce capteur.

Algorithme 3 Algorithme de calcul du début du temps logique.

Entrées:

vecteur v de signaux "*Awaited*"

vecteur vse d'évènements de SUP

vecteur $vstor$ d'évènements des capteurs TOR

vecteur $vstimp$ d'évènements des capteurs TI

vecteur $vsana$ d'évènements des capteurs AE

Sorties: temps $DébutTempsLogique$

pour i **de** 1 **jusqu'à** $taille(v)$ **faire**
  $DébutTempsLogique \leftarrow \infty$
  pour j **de** 1 **jusqu'à** $taille(vse)$ **faire**
    si $v(i)$ **correspond** $vse(j)$ **et** $dateDébut(j) < DébutTempsLogique$ **alors**
      $DébutTempsLogique \leftarrow dateDébut(j)$
    finsi
  fin pour
  pour j **de** 1 **jusqu'à** $taille(vstor)$ **faire**
    si $v(i)$ **correspond** $vse(j)$ **et** $dateDébut(j) < DébutTempsLogique$ **alors**
      $DébutTempsLogique \leftarrow dateDébut(j)$
    finsi
  fin pour
  pour j **de** 1 **jusqu'à** $taille(vstimp)$ **faire**
    si $v(i)$ **correspond** $vse(j)$ **et** $dateDébut(j) < DébutTempsLogique$ **alors**
      $DébutTempsLogique \leftarrow dateDébut(j)$
    finsi
  fin pour
  pour j **de** 1 **jusqu'à** $taille(vsana)$ **faire**
    si $v(i)$ **correspond** $vse(j)$ **et** $dateDébut(j) < DébutTempsLogique$ **alors**
      $DébutTempsLogique \leftarrow dateDébut(j)$
    finsi
  fin pour
fin pour

retourner $DébutTempsLogique$

➤ **Calcul de la fin du temps logique** : Nous avons déjà vu dans le paragraphe 3.3.1 que chaque source d'informations doit avoir un temps d'échantillonnage (les Δt gérés par SUP et par les capteurs environnementaux).

Parmi ces paramètres, nous avons décidé de choisir le plus petit, de telle sorte qu'un instant logique dure un temps égal à ce paramètre.

Comme le montre l'algorithme 4 nous commençons à comparer les différents "Δt" pour choisir le plus petit. Nous décidons de cette façon que la fin du temps logique correspond à un décalage du début du temps logique déjà calculé par la valeur de ce Δt choisi.

Algorithme 4 Algorithme de calcul du fin du temps logique.

Entrées:

entier $DeltaTsup$ de SUP

vecteur d'entiers $DeltaTvstor$ des capteurs TOR

vecteur d'entiers $DeltaTvstimp$ des capteurs TI

vecteur d'entiers$DeltaTvsana$ des capteurs AE

Sorties: entier $FinTempsLogique$

$\Delta t_{min} \leftarrow \infty$

pour j **de** 1 **jusqu'à** $taille(DeltaTvstor)$ **faire**

si $DeltaTvstor(j) < \Delta t_{min}$ **alors**

$\Delta t_{min} \leftarrow DeltaTvstor(j)$

finsi

fin pour

pour j **de** 1 **jusqu'à** $taille(DeltaTvstimp)$ **faire**

si $DeltaTvstimp(j) < \Delta t_{min}$ **alors**

$\Delta t_{min} \leftarrow DeltaTvstimp(j)$

finsi

fin pour

pour j **de** 1 **jusqu'à** $taille(DeltaTvsana)$ **faire**

si $DeltaTvsana(j) < \Delta t_{min}$ **alors**

$\Delta t_{min} \leftarrow DeltaTvsana(j)$

finsi

fin pour

si $DeltaTsup < \Delta t_{min}$ **alors**

$\Delta t_{min} \leftarrow DeltaTsup$

finsi

$FinTempsLogique \leftarrow DébutTempsLogique + \Delta t_{min}$

retourner $FinTempsLogique$

Nous proposons dans la figure 3.12 un système synchrone modélisé par deux automates (*automate* 1 et *automate* 2). Initialement, les deux automates sont dans leurs états 1, l'*automate* 1 attend les évènements $v25$ et $v14$ pour transiter et l'*automate* 2 attend de même l'évènement $v33$. À partir de ces signaux "*Awaited*", la machine est capable de construire l'instant logique en intégrant le plus d'informations pertinentes ("*Awaited*").

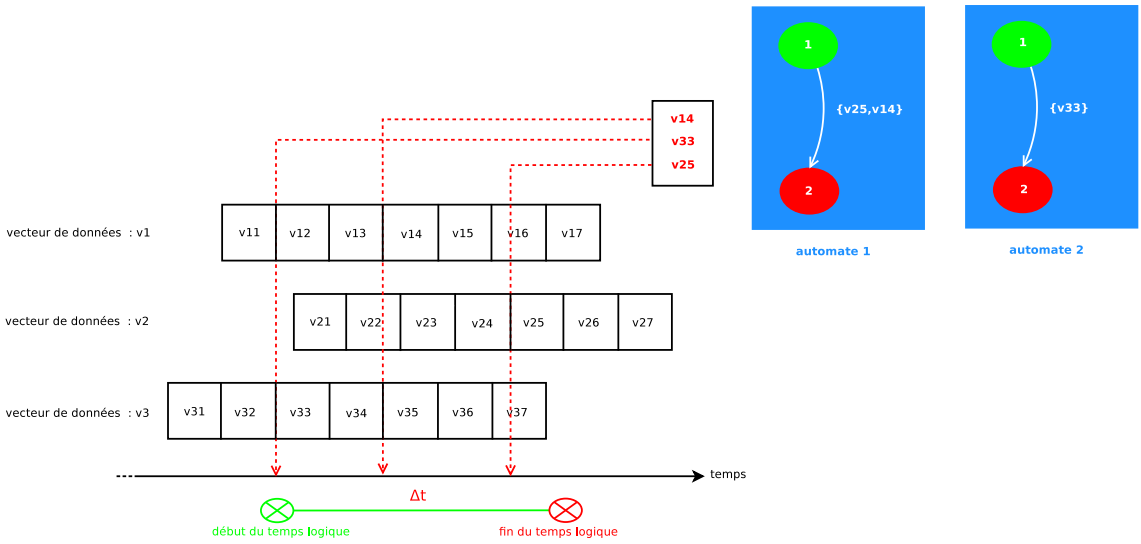


Figure 3.12 — Fonctionnement de la machine d'exécution dans le cas normal : Définition standard de l'instant logique.

Après calcul de l'instant logique par la machine d'exécution, voici l'état final du système (figure 3.13).

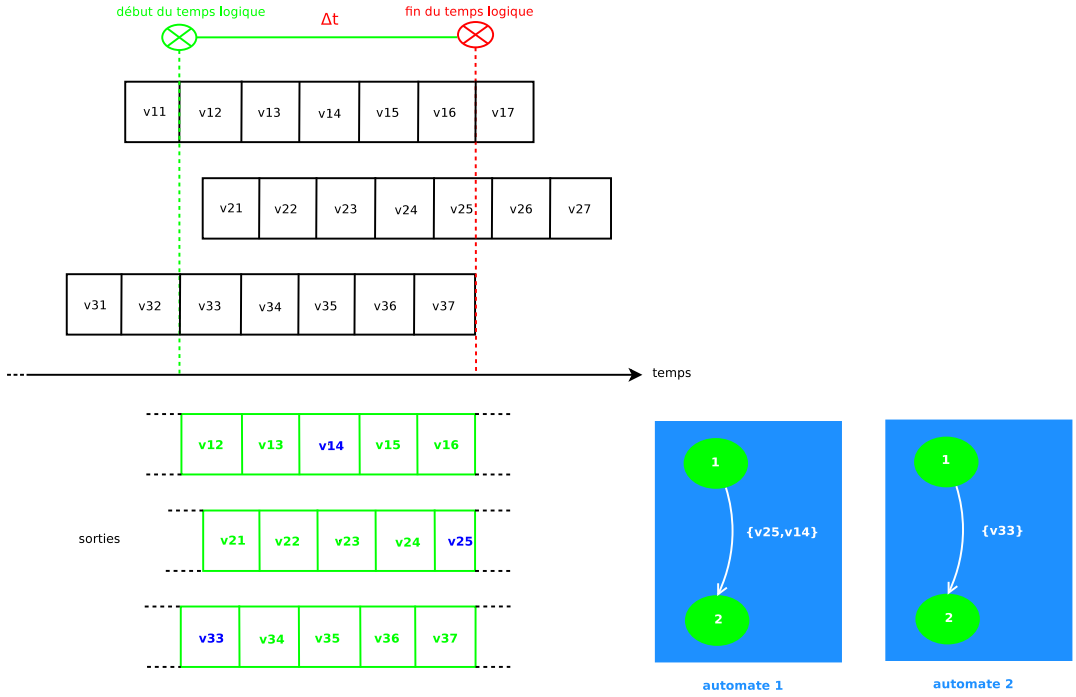


Figure 3.13 — Résultat du fonctionnement de la machine d'exécution dans le cas normal : Définition standard de l'instant logique.

Dans l'instant logique, nous récupérons les signaux attendus $v25$, $v14$ et $v33$, de cette façon les automates peuvent transiter leurs états 1.

3.4.2.2 Redéfinition de l'instant logique : Cas des exceptions

Un instant logique se calcule d'une façon standard en décalant par Δt_{min} le temps d'apparition de la première occurrence du signal parmi ceux attendus.

Les systèmes de reconnaissance d'activités synchrones doivent réagir suivant les statuts des signaux d'entrée (présent/absent). Mais, nous avons constaté que dans l'instant logique défini d'une façon standard, certains signaux peuvent avoir différents statuts, ce qui est contraire à l'hypothèse synchrone. Pour éviter ce problème, nous avons défini des exceptions qui caractérisent la nécessitéé d'avoir une redéfinition du temps logique :

- Apparition d'une seconde occurrence d'un évènement dans l'instant logique calculé d'une façon standard (voir paragraphe 3.4.2.2.1) ;
- Apparition d'un évènement après la fin d'une suite d'apparitions consécutives d'occurrences de cet évènement dans l'instant logique calculé d'une façon standard (voir paragraphe 3.4.2.2.2) ;
- Chevauchement d'occurrences d'un évènement dans l'instant logique calculé d'une façon standard (voir paragraphe 3.4.2.2.3).

En fait, si le système présente l'une de ces exceptions alors l'instant logique standard est modifié pour tenir compte de ces exceptions.

En effet, pour vérifier que le système présente l'une des exceptions, prenons par exemple une source (vidéo ou environnemental) dont les données sont stockées dans un vecteurs v. Dans un instant logique défini d'une façon standard à partir de Δt_{min}, nous cherchons les redondances du même signal attendu et nous les enregistrons dans *tab*. Comme le montre l'algorithme 5.

Algorithme 5 Algorithme de calcul du temps du coupure.

Entrées:

vecteur $vrec$ des "*Awaited*"

vecteur v de données

Sorties: temps $dateCoupure$

pour i **de** 1 **jusqu'à** $taille(vrec)$ **faire**
  pour j **de** 1 **jusqu'à** $taille(v)$ **faire**
    si $vrec(i) = v(j)$ **et** $dateDébut(j) < FinTempsLogique$ **alors**
      insérer($vse(j)$,tab)
    finsi
  fin pour
fin pour
si $taille(tab) = 1$ **alors**
  $dateCoupure \leftarrow dateFin(tab(0))$
sinon
  $dateCoupure \leftarrow TestTroisExceptions(tab)$
finsi
retourner $dateCoupure$

Dans un deuxième temps, l'algorithme 6, prend le résultat de l'algorithme précédent, si nous avons un nombre d'occurrences supérieur à 1 ($taille(tab) > 1$), alors nous testons quel traitement a été choisi précédemment parmi les exceptions possibles.

Algorithme 6 Algorithme de test des trois exceptions.

Entrées:

vecteur tab de données identiques dans un temps logique

Sorties: temps $dateCoupure$

tantque $i < taille(tab) - 1$ **et** $trouve =$ faux **faire**

si $dateFin(tab(i)) < dateDébut(tab(i+1))$ **et** $dateFin(tab(i)) < FinTempsLogique$ **alors**

$dateCoupure \leftarrow dateDébut(tab(i+1))$

sinon si $dateFin(tab(i)) >= dateDébut(tab(i+1))$ **alors**

$i \leftarrow i+1$

sinon si $i = taille(tab) - 1$ **et** $dateDébut(tab(i)) <= FinTempsLogique$ **alors**

$dateCoupure \leftarrow dateFin(tab(i))$

finsi

fin tantque

$FinTempsLogique \leftarrow DébutTempsLogique + dateCoupure$

retourner $FinTempsLogique$

Notons que si nous avons plus d'un évènement qui présente des cas d'exceptions, alors nous devons calculer les temps de coupure imposés par chaque évènement. Enfin nous décidons que le temps de coupure de l'instant logique final correspond au plus ancien temps parmi ces temps calculés.

Nous traitons dans les paragraphes suivants les trois exceptions en détaillant pas à pas quelques exemples et montrons les aspects dynamiques.

3.4.2.2.1 Redéfinition de l'instant logique : cas d'une apparition d'une seconde occurrence d'un évènement dans l'instant logique calculé d'une façon standard

Nous reprenons la même modélisation du système de reconnaissance d'activités que dans le paragraphe (paragraphe 3.12), en modifiant les évènements donnés en entrée de la machine d'exécution de telle sorte que le vecteur de données $v1$ présente le cas d'une réapparition de l'évènement x à l'indice (1 ;4) après la fin d'une première occurrence à l'indice (1 ;2) dans le même instant. De cette façon, la machine doit arrêter le temps logique avant l'apparition du deuxième puisque cela est considéré comme un changement d'état.

Après avoir calculé l'instant logique, réponse de la machine d'exécution à cette exception, nous présentons l'état final du système dans la figure 3.14. Dans l'instant logique, nous récupérons le signal $v33$, de cette façon l'automate 2 peut transiter dans son état 1. Notons que dans la prochaine exécution les signaux $v25$ et $v14$ restent attendus.

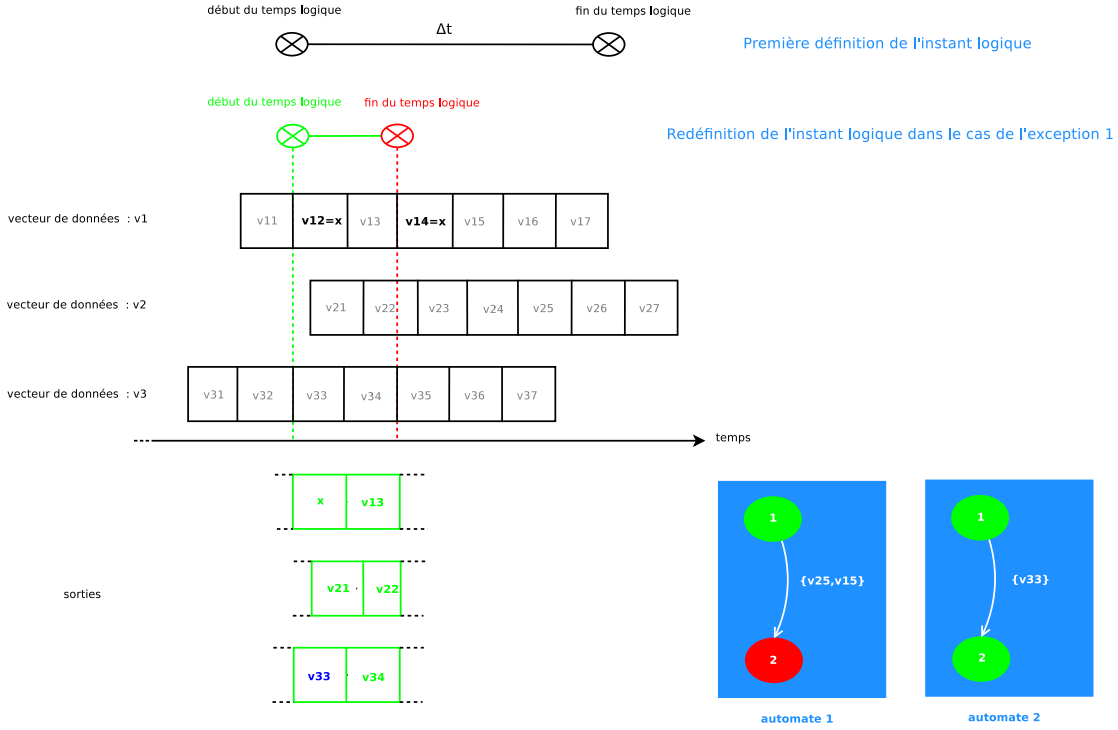


Figure 3.14 — Résultat du fonctionnement de la machine d'exécution lors de l'exception 1.

3.4.2.2.2 Redéfinition de l'instant logique : cas d'une apparition d'un évènement après la fin d'une suite d'apparitions consécutives d'occurrences de cet évènement dans l'instant logique calculé d'une façon standard

Nous reprenons la même modélisation du système de reconnaissance d'activités que dans le paragraphe (paragraphe 3.12), en modifiant les évènement donné en entrée de la machine d'exécution de telle sorte que le vecteur de données $v1$ présente le cas d'une réapparition de l'évènement x à l'indice (1 ;6) après la fin d'une suite d'apparitions de cet évènement à l'indice (1 ;4) dans le même instant. De cette façon, la machine doit arrêter le temps logique avant une occurrence isolée de l'évènement après la suite des premiers occurrences successives de celui-ci car cela est considéré comme un changement d'état.

Après avoir calculé l'instant logique, réponse de la machine d'exécution à cette exception, nous présentons l'état final du système dans la figure 3.15. Dans l'instant logique, nous récupérons les signaux $v15$ et $v33$, de cette façon l'automate 2 peut transiter dans son état 1. L'automate 1 ne peut transiter qu'à la présence des deux signaux $v15$ et $v25$. Notons que dans la prochaine exécution les signaux $v15$ et $v25$ restent attendus par le système synchrone de reconnaissance.

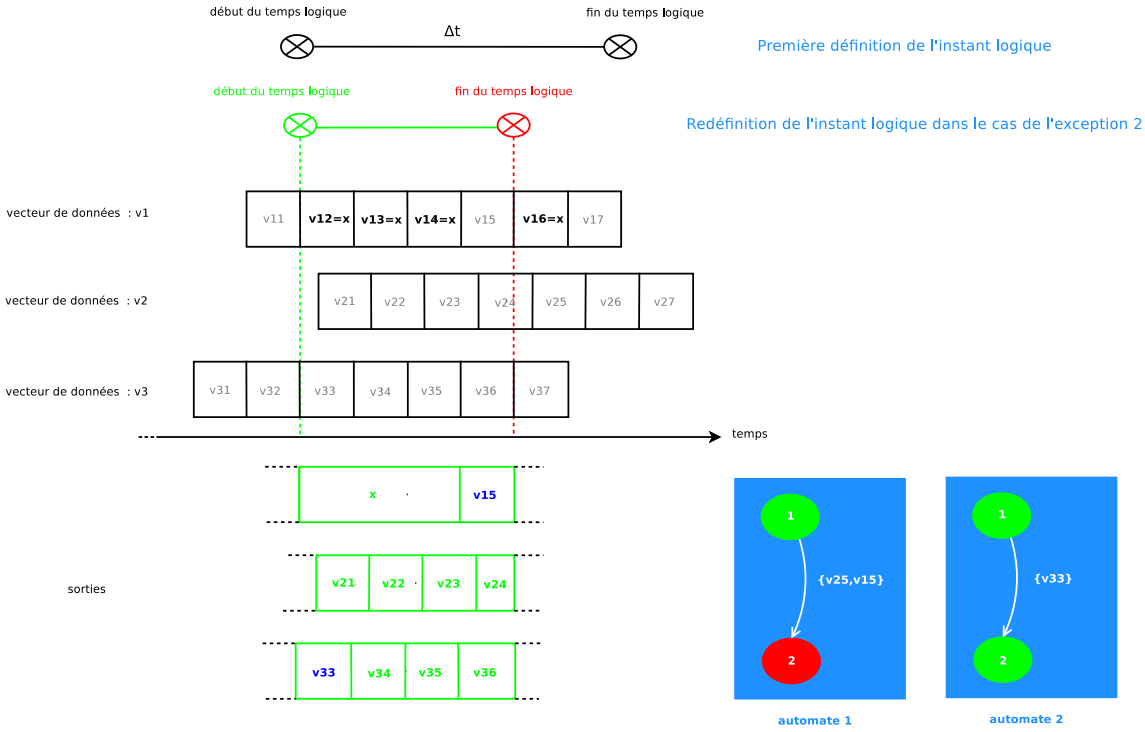


Figure 3.15 — Résultat du fonctionnement de la machine d'exécution lors de l'exception 2.

3.4.2.2.3 Redéfinition de l'instant logique : cas du chevauchement d'occurrences d'un évènement dans l'instant logique calculé d'une façon standard

Nous reprendrons la même modélisation du système de reconnaissance d'activités que dans le paragraphe (voir 3.12), en modifiant les évènements donnés en entrée de la machine d'exécution de telle sorte que le vecteur de données $v1$ présente le cas d'une apparition de l'évènement x à l'indice (1 ;3) avant la fin de la première occurrence à l'indice (1 ;2) dans le même instant. De cette façon, la machine doit les considérer comme une seule apparition et arrête le temps logique qu'après la seconde occurrence, puisque cela est considéré comme un changement d'état.

Après avoir calculé l'instant logique, réponse de la machine d'exécution à cette exception, nous présentons l'état final du système dans la figure 3.16. Dans l'instant logique, nous récupérons que les signaux $v15$ et $v33$, de cette façon que l'automate 2 peut transiter son état 1. L'automate 2 ne peut transiter qu'en présence des deux signaux $v15$ et $v25$. Notons que dans la prochaine exécution les signaux $v15$ et $v25$ restent attendus par le système synchrone de reconnaissance.

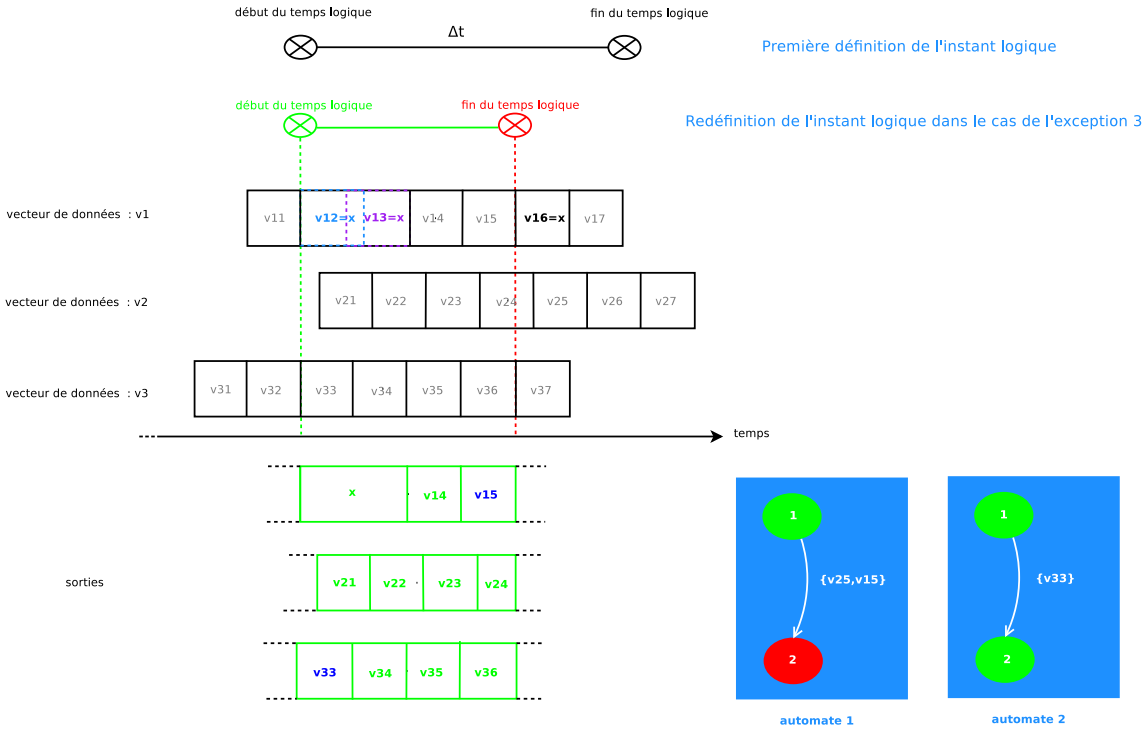


Figure 3.16 — Résultat du fonctionnement de la machine d'exécution lors de l'exception 3.

3.5 Conclusion

Ce chapitre a été consacré à la spécification, la conception et implémentation de notre machine d'exécution tout en l'interfaçant avec les systèmes de reconnaissance synchrones modélisant des évènements réels provenant d'une façon asynchrone de diverses sources. En premier lieu, nous avons présenté le pré-traitement des données en entrée, tri et filtrage.En deuxième lieu, nous avons expliqué le fonctionnement de la machine d'exécution dans son cas de fonctionnement standard ainsi ses réactions en cas où le système présente des conditions exceptionnelles.

Nous abordons dans le chapitre suivant, le test de la machine proposée sur des données réelles provenant du centre hospitalier universitaire de Nice.

CHAPITRE 4

Évaluation et résultats

4.1 Introduction

Afin d'évaluer le fonctionnement de la machine d'exécution, plusieurs expériences ont été réalisées. Le calcul des instants logiques est l'objectif principal de ces expériences. Pour cela nous avons modélisé des systèmes synchrones de reconnaissance d'activités. En communiquant avec ces modèles, nous allons tester l'interaction de la machine d'exécution dans un environnement de données élaborées au centre hospitalier de Nice.

Dans la première partie de ce chapitre, nous présentons les outils utilisés pour modéliser les systèmes synchrones. Dans la deuxième partie, nous avons choisi de présenter la réaction de la machine vis-à-vis des modèles de scénarios : "$Before$" , "$Overlaps$" et un troisième test de la machine en interaction avec un système modélisant deux scénarios "$Before$" synchrones en parallèle. Enfin, nous abordons les problèmes résolus et non résolus rencontrés au cours de cette étude et les amélioration futures de la machine que nous avons proposé.

4.2 Modélisation des systèmes de reconnaissance d'activités

4.2.1 Outils de modélisation

Pour concevoir le système de reconnaissance d'activités, nous avons utilisé deux outils de recherche développés par Monsieur Daniel Gaffé [1] : Galaxy et Autom2circuit.

1. http://webs.unice.fr/dgaffe/recherche/outils_automate.html

4.2.1.1 Galaxy

Galaxy est un éditeur d'automates d'états finis : automates simples, automates parallèles synchrones, automates hiérarchiques ou SyncCharts. Nous intersessions dans notre étude des éditions des automates simples.

4.2.1.2 Autom2circuit

Autom2circuit est un synthétiseur d'automates explicites Galaxy en machines de Mealy ou Moore booléennes. Comme le montre la figure 4.1, il génère de multiples formats de sortie (C, Vhdl, Blif, etc.), de plus l'option "with awaited" de cet outil donne en sortie les évènements attendus pour atteindre l'état futur. C'est un facteur qui nous a permis de calculer les caractéristiques du prochain instant logique.

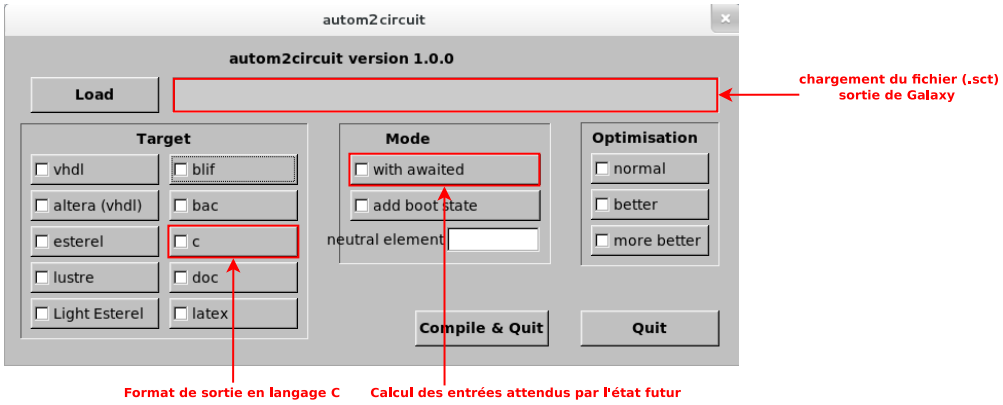


Figure 4.1 — Interface de l'outil Autom2circuit : génération du code C avec calcul des évènements attendus au prochain état.

4.2.2 Interfaçage des outils Galaxy/Autom2circuit avec la machine d'exécution proposée

Galaxy/Autom2circuit permettent de modéliser et de synthétiser un système synchrone pour une cible en langage C par exemple.

Le code C généré est basé sur deux principales fonctions : Une fonction d'initialisation et une fonction de calcul de l'état futur et des sorties dont les "*Awaited*". Les deux fonctions ont respectivement cette forme :

$extern\ void\ NAME\_reset\_automaton(int\ entrée_1,\ int\ entrée_2,\ ...,\ int\ entrée_n,\ int\ *sortie_1,\ int\ *sortie_2,\ ...,\ int\ *sortie_m,\ int\ *awaited_1,\ intawaited_2,\ ...,\ int\ *awaited_p);$

$extern\ void\ NAME\_automaton(int\ entrée_1,\ int\ entrée_2,\ ...,\ int\ entrée_n,\ int\ *sortie_1,\ int\ *sortie_2,\ ...,\ int\ *sortie_m,\ int\ *awaited_1,\ intawaited_2,\ ...,\ int\ *awaited_p);$

Ces fonctions manipulent les entrées, les sortie et les "*Awaited*" comme des variables booléennes, alors que les entrées et les sorties que nous avons considérées, même les signaux "*Awaited*" sont des instances de classes bien structurées comme nous avons vu dans le chapitre 2. Pour adapter les différentes variables booléennes aux évènements vidéo et aux évènements des capteurs nous introduisons une table de correspondance. De cette façon, le statut des vrais évènements reste cohérent avec celui des variables booléennes (absent/présent).

4.2.3 Autres outils

Autre que Galaxy et Autom2circuit, nous avons utilisé :

- Blif_Simul : Un simulateur graphique d'automate implicite au format blif ;
- Merge_Blif : Un outil de concaténation de fichiers blif ;
- BlifTo : Un convertisseur de fichier blif vers d'autre formats (C, etc.).

4.3 Scénario "$Before$"

$e1$ $Before$ $e2$ est un scénario qui teste que l'évènement $e1$ arrive et se termine avant l'évènement $e2$. Il lève une alarme $TERM$ si le scénario est satisfait.

Nous avons utilisé l'éditeur d'automate Galaxy comme le montre la figure 4.2, pour modéliser le scénario $e1$ $Before$ $e2$. Nous avons appliqué ce modèle de scénario pour tester le scénario concret d'une personne qui doit être dans la zone où il y a la table de lecture avant qu'elle soit dans le coin de café. Nous considérons alors : $e1$ $Before$ $e2$, avec $e1$: *La personne p1 est dans la zone où il y a la table de lecture* et $e2$: *La personne p1 est dans le coin de café*.

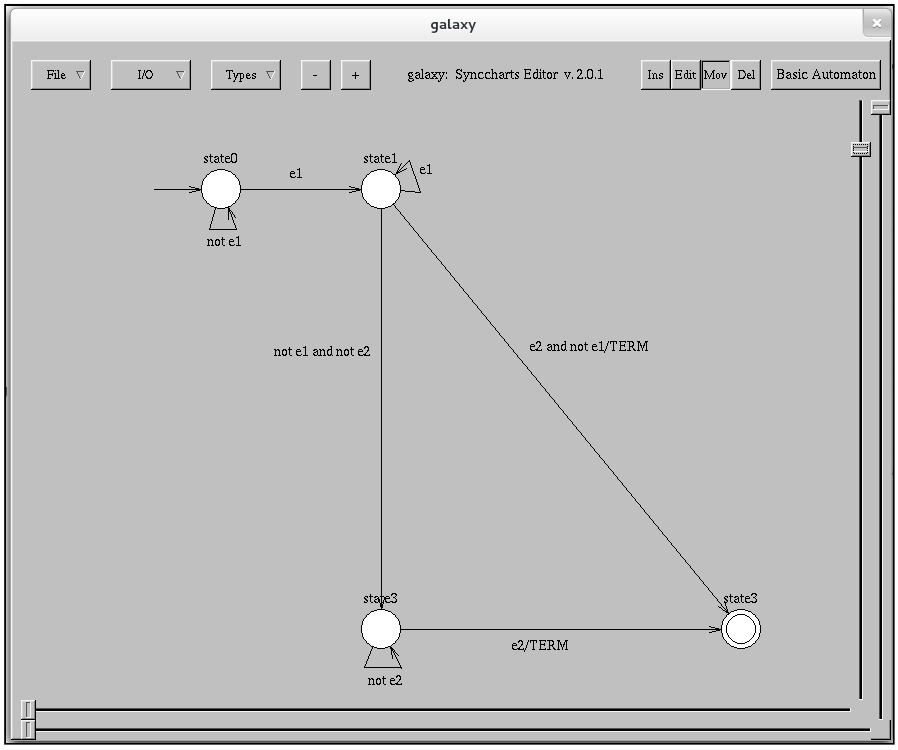


Figure 4.2 — Édition du modèle $e1$ $Before$ $e2$ en utilisant l'outil Galaxy.

Le format interne de galaxy est sous forme d'un fichier "*.sct*". En utilisant *Autom2circuit*, nous pouvons générer deux représentations sous forme de machine de Mealy, l'une au format "*.blif*" l'autre au format "*.c*".

Nous considérons la table de correspondance entre les signaux attendus de type booléen du code "*.c*" généré par l'outil *Autom2circuit* et les évènements concrets que considère la machine d'exécution

($Person\_Inside\_Zone\_UseReandingTable$ $Before$ $Person\_Inside\_Zone\_UseCoffeeCorner$), comme le montre le tableau 4.1.

Variables booléennes	Objets
e1	Person_Inside_Zone_UseReandingTable (p1,zoneUseReadingTable)
e2	Person_Inside_Zone_UseCoffeeCorner (p1,zoneUseCoffeeCorner)

Tableau 4.1 — Tableau de correspondance du scénario "$Before$".

Nous pouvons simuler le fonctionnement de cet automate en format "$.blif$" en utilisant l'outil $Blif\_simul$, comme le montre la figure 4.3.

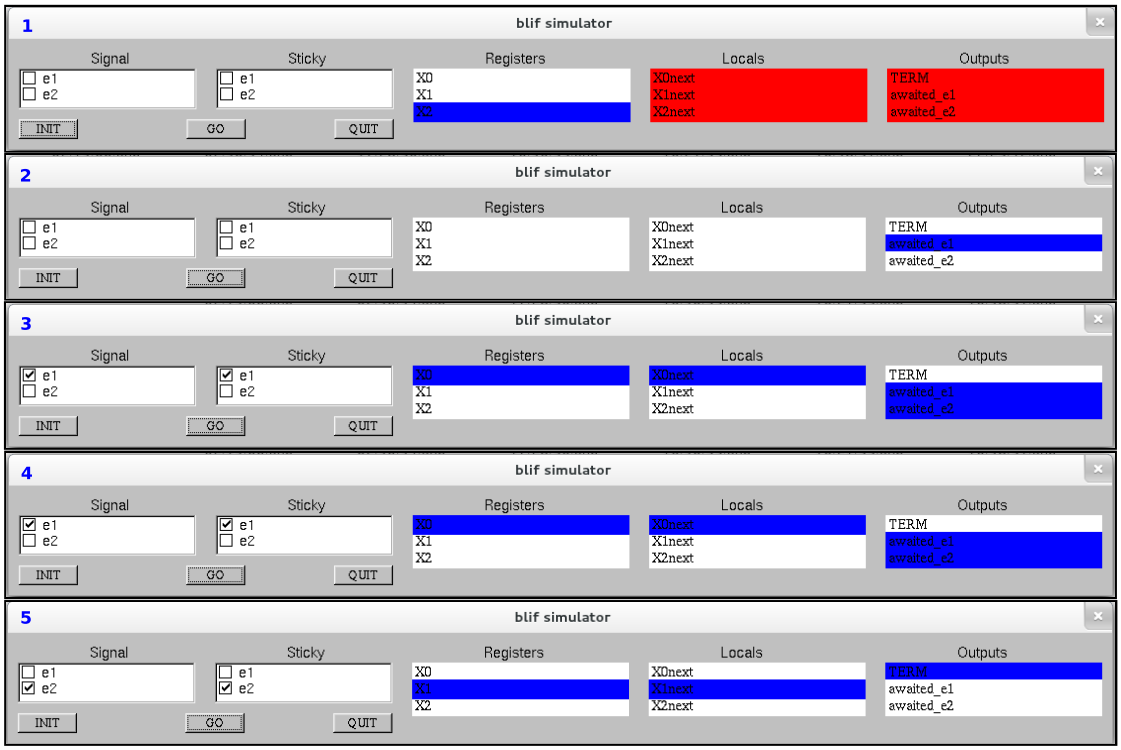


Figure 4.3 — Simulation du fonctionnement du scénario $e1$ $Before$ $e2$ en utilisant l'outil $Blif\_simul$.

Rouge signifie non évalué, sans coloration signifie absent et bleu signifie présent.

De cette façon la machine d'exécution, en utilisant le code "$.c$" et la table de correspondance, peut calculer les instants logiques à partir des signaux attendus et transférer l'ensemble des évènements qui caractérise l'instant logique calculé.

Comme le montre l'exécution dans la figure 4.4, au début nous attendons le premier évènement "$Person\_Inside\_Zone\_UseReandingTable$", une fois reçu, nous attendons dans le prochain instant le deuxième évènement "$Person\_Inside\_Zone\_UseCoffeeCorner$".

En fait, dans le premier et deuxième les deux signaux "*Person_Inside_Zone_UseReandingTable*" et "*Person_Inside_Zone_UseCoffeeCorner*" sont attendus, mais dans le troisième le signal attendu "*Person_Inside_Zone_UseCoffeeCorner*" est trouvé dans le fichier d'entrée et le signal "*TERM*" est émis car le scénario est reconnu et terminé.

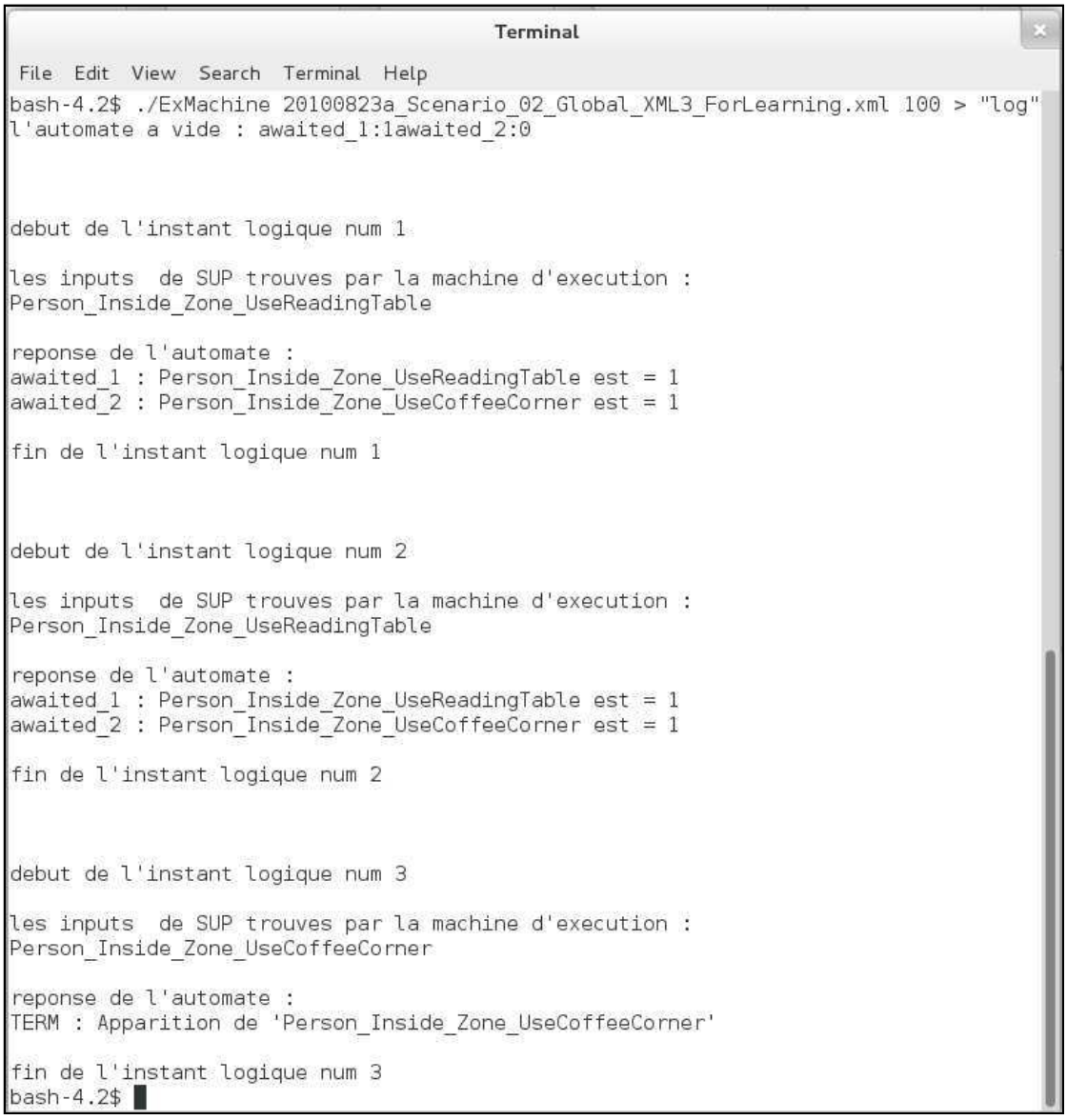


Figure 4.4 — Réaction de la machine d'exécution vis-à-vis le scénario "*Before*".

Nous avons extrait du fichier *log* (voir figure 4.5) quelques portions, dans lesquelles il y a une succession des calculs des instants logiques et les évènements construisant cet instant.

```
############liste des evenements de la SUP##############
la taille de la liste avant filtre est :648
la taille de la liste apres filtre est :57
```

```
L'ensemble des signaux attedus :
Signal attendu num:1
        Name: Person_Inside_Zone_UseReadingTable
        Arguments: p1/zoneUseReadingTable

L'instant logique va commencer apartir de cette date :2010/8/23 10:40:24:106
Normalement l'instant logique se termine a cette date:  2010/8/23       10:40:24:206

Apres les tests des exceptions, l'instant logique se termine a cette date:      2010/8/23       10:40:24:206
###les outputs SUP###

Output Activity num 1 est:
1       Person_Inside_Zone_UseReadingTable      1       PrimitiveState  NOTURGENT       NOT MEITIONED   NOT MEITIONED   1       6468    2010/8/23       10:40:24:106    0       2010/8/23       10:40:24:226    0
la liste des Activites des objets physique est :
        Activity de l'objet num 1 est: OBJECT/0/p1/0
        Activity de l'objet num 2 est: STATIC_ZONE/7/zoneUseReadingTable/0
on va eliminer les valeur jusqu'a le temps:     2010/8/23       10:40:24:206
Il nous reste que :
la taille de la liste sup est :50
```

```
L'ensemble des signaux attedus :
Signal attendu num:1
        Name: Person_Inside_Zone_UseReadingTable
        Arguments: p1/zoneUseReadingTable
Signal attendu num:2
        Name: Person_Inside_Zone_UseCoffeeCorner
        Arguments: p1/zoneUseCoffeeCorner

L'instant logique va commencer apartir de cette date :2010/8/23 10:40:24:726
Normalement l'instant logique se termine a cette date:  2010/8/23       10:40:24:826

Apres les tests des exceptions, l'instant logique se termine a cette date:      2010/8/23       10:40:24:826
###les outputs SUP###

Output Activity num 1 est:
2       Person_Inside_Zone_UseReadingTable      1       PrimitiveState  NOTURGENT       NOT MEITIONED   NOT MEITIONED   1       6477    2010/8/23       10:40:24:726    0       2010/8/23       10:40:25:357    0
la liste des Activites des objets physique est :
        Activity de l'objet num 1 est: OBJECT/1/p1/0
        Activity de l'objet num 2 est: STATIC_ZONE/7/zoneUseReadingTable/0
on va eliminer les valeur jusqu'a le temps:     2010/8/23       10:40:24:826
Il nous reste que :
la taille de la liste sup est :49
```

```
L'ensemble des signaux attedus :
Signal attendu num:1
        Name: Person_Inside_Zone_UseReadingTable
        Arguments: p1/zoneUseReadingTable
Signal attendu num:2
        Name: Person_Inside_Zone_UseCoffeeCorner
        Arguments: p1/zoneUseCoffeeCorner

L'instant logique va commencer apartir de cette date :2010/8/23 10:40:34:855
Normalement l'instant logique se termine a cette date:  2010/8/23       10:40:34:955

Apres les tests des exceptions, l'instant logique se termine a cette date:      2010/8/23       10:40:34:955
###les outputs SUP###

Output Activity num 1 est:
5       Person_Inside_Zone_UseCoffeeCorner      1       PrimitiveState  NOTURGENT       NOT MEITIONED   NOT MEITIONED   1       6596    2010/8/23       10:40:34:855    0       2010/8/23       10:40:40:226    0
la liste des Activites des objets physique est :
        Activity de l'objet num 1 est: OBJECT/1/p1/0
        Activity de l'objet num 2 est: STATIC_ZONE/13/zoneUseCoffeeCorner/0
on va eliminer les valeur jusqu'a le temps:     2010/8/23       10:40:34:955
Il nous reste que :
la taille de la liste sup est :37
```

Figure 4.5 — Fichier *log* de sortie : Réaction de la machine d'exécution vis-à-vis le scénario "$Before$".

4.4 Scénario "$Overlaps$"

$e1$ *Overlaps* $e2$ est un scénario qui teste que l'évènement $e1$ arrive lorsque l'évènement $e2$ est déjà présent. Il lève une alarme $TERM$ si c'est le cas.

Nous avons utilisé l'éditeur d'automate Galaxy comme le montre la figure 4.6, pour modéliser le scénario $e1$ *Overlaps* $e2$. Comme précédemment, nous avons appliqué ce modèle au scénario concret d'une personne qui ne doit pas être dans la zone où il y a la table de lecture en même temps que dans le coin de café.

Si $e1$ *Overlaps* $e2$, avec $e1$: *La personne* $p1$ *est dans la zone où il y a la table de lecture* et $e2$: *La personne* $p1$ *dans le coin de café*, alors nous déclenchons une alarme "$TERM$" indiquant que le scénario est reconnu.

Pour ce scénario, nous construisons la table de correspondance entre les signaux attendus de type booléen du code "*.c*", généré par l'outil *Autom2circuit*, et les objets réels que considère la machine d'exécution ($(Person\_Inside\_Zone\_UseReandingTable\ Overlaps\ Person\_Inside\_Zone\_UseCoffeeCorner)$), comme le montre le tableau 4.2.

Rappelons que l'abstraction synchrone du monde physique est définie par le fait d'associer à un évènement concert deux évènements abstraits, un modélise l'apparition et l'autre la disparition. C'est le cas de cet exemple qui utilise l'évènement attendu $Person\_Inside\_Zone\_UseReandingTable(p1, zoneUseCoffeeCorner)$ où e2 correspond à sa présence et ne2 correspond à son absence.

Variables booléennes	Objets
e1	Person_Inside_Zone_UseReandingTable (p1,zoneUseReadingTable)
e2 et ne2 (=not e2)	Person_Inside_Zone_UseCoffeeCorner (p1,zoneUseCoffeeCorner)

Tableau 4.2 — Tableau de correspondance du scénario "*Overlaps*".

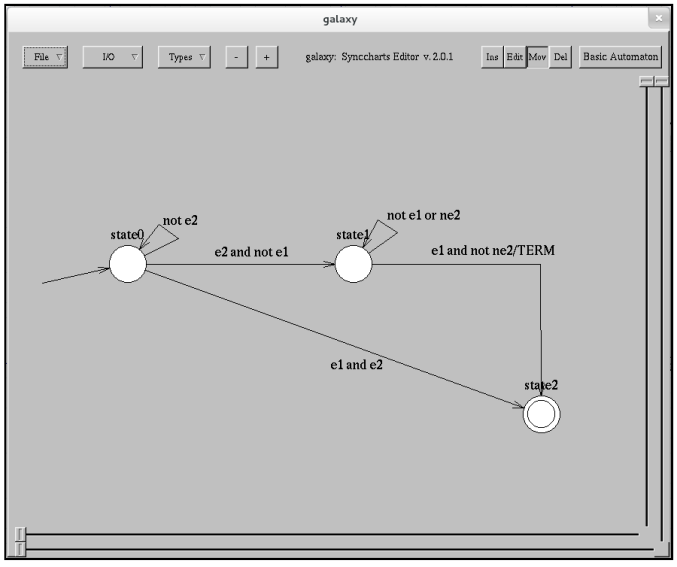


Figure 4.6 — Édition du modèle $e1$ *Overlaps* $e2$ en utilisant l'outil Galaxy.

Nous simulons le "*.blif*" de cet automate en utilisant l'outil *Blif_simul*, comme le montre la figure 4.7.

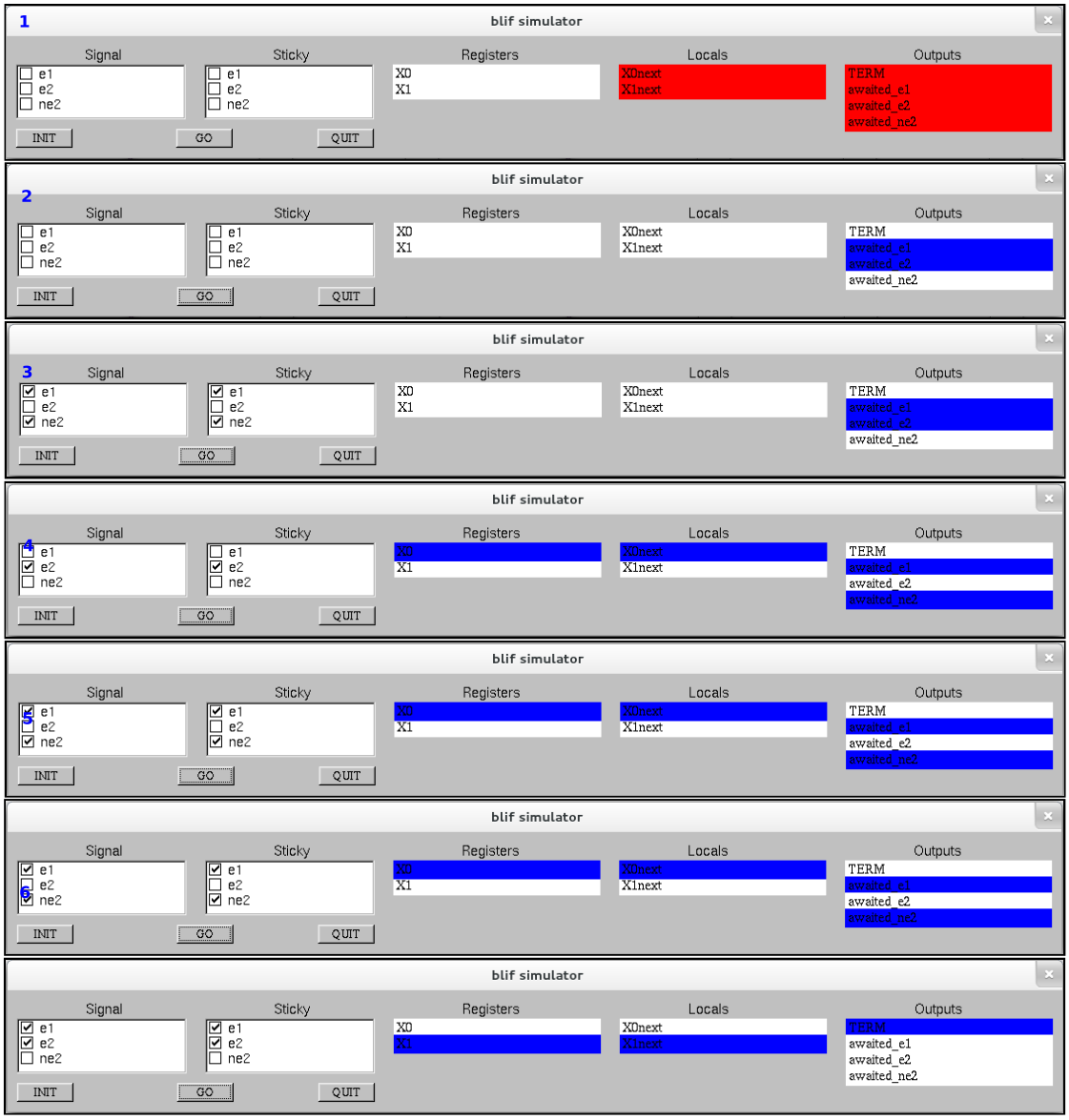


Figure 4.7 — Simulation du fonctionnement du scénario $e1$ *Overlaps* $e2$ en utilisant l'outil $Blif\_simul$.

Rouge signifie non évalué, sans coloration signifie absent et bleu signifie présent.

Comme le montre l'exécution dans la figure 4.8, au début nous attendons les deux évènements "$Person\_Inside\_Zone\_UseReandingTable$" et "$Person\_Inside\_Zone\_UseCoffeeCorner$". Ensuite, nous avons l'évènement "$Person\_Inside\_Zone\_UseReandingTable$"dans l'instant logique numéro 1, de même pour l'instant logique numéro 2. Pour le troisième il y a apparition de l'évènement "$Person\_Inside\_Zone\_UseCoffeeCorner$" et ainsi de suite. Nous avons pas reconnu le scénario $e1$ *Overlaps* $e2$ jusqu'à la fin de transfert des donnée physiques.

Figure 4.8 — Réaction de la machine d'exécution vis-à-vis le scénario "*Overlaps*".

Nous avons élaboré du fichier *log* (voir figure 4.9) quelques portions, dans lesquelles il y a une succession des calculs des instants logiques et les évènements "*Outputs*" construisant cet instant.

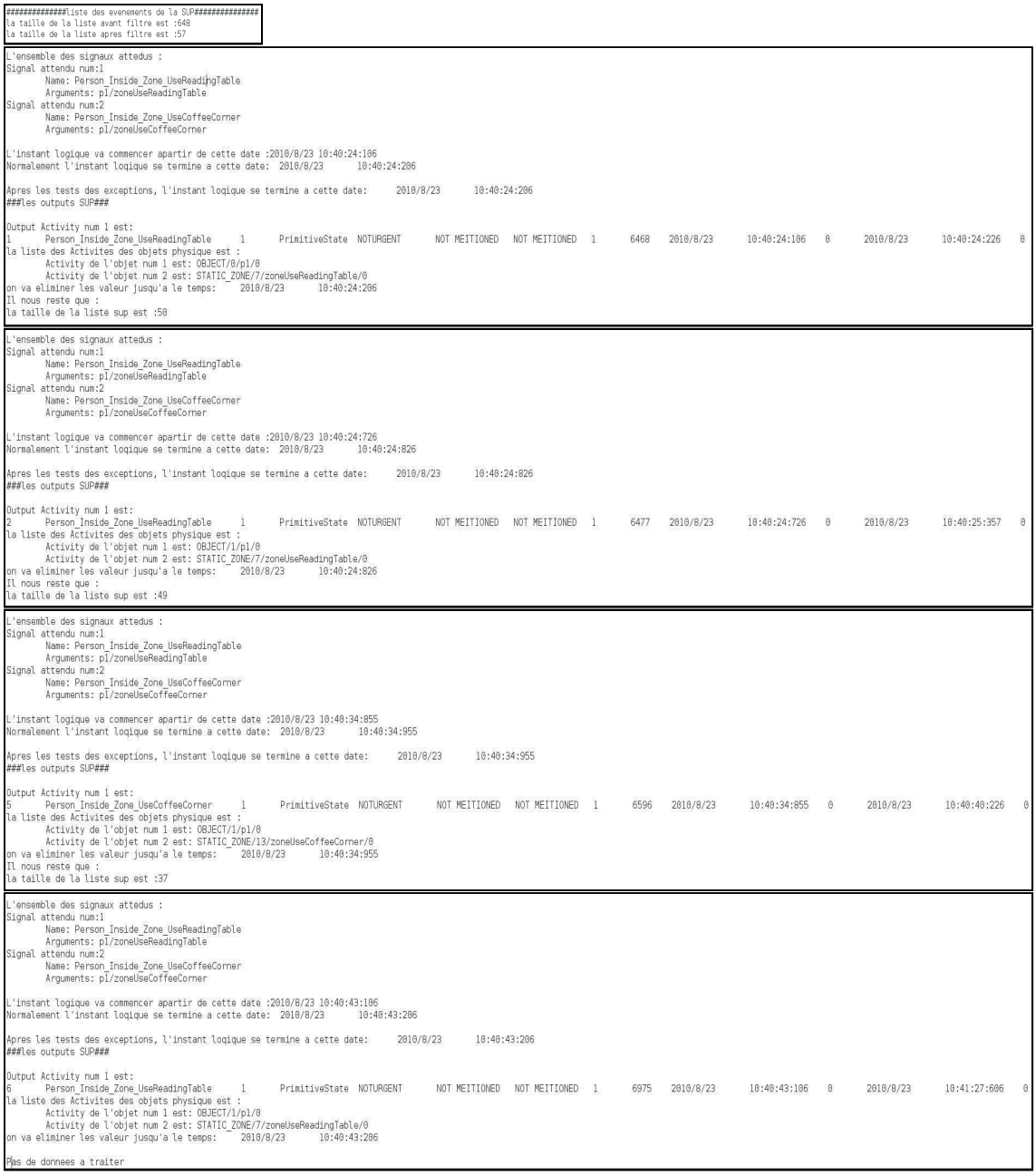


Figure 4.9 — Fichier *log* de sortie : Réaction de la machine d'exécution vis-à-vis le scénario "*Overlaps*".

4.5 Modèle de deux scénarios "$Before$" synchrones

Ce modèle consiste à contrôler deux scénarios en parallèle :

- $e3$ $Before$ $e1$ est un modèle qui teste que l'évènement $e3$ arrive et se termine avant l'évènement $e1$. Il lève une alarme $TERM$ si c'est le cas.

  Nous appliquons ce modèle à la reconnaissance du scénario d'une personne qui doit être à l'entrée avant d'être dans la zone où il y a la table de lecture. On considère : $e3$ $Before$ $e1$, avec $e3$: *La personne* $p1$ *est à l'entrée* et $e2$: *La personne* $p1$ *est à la zone où il y a la table de lecture* ;

- $e1$ $Before$ $e2$ est un scénario qui teste que l'évènement $e1$ arrive et se termine avant l'évènement $e2$. Il lève une alarme $TERM1$ si c'est le cas.

  Nous appliquons ce modèle à la reconnaissance du scénario d'une personne qui doit être t dans la zone ou il y a la table de lecture avant dêtre dans la zone où il y a le coin caffé.On considère : $e1$ $Before$ $e2$, avec $e1$: *La personne* $p1$ *est à l zone où il y a la table de lecture* et $e2$: *La personne* $p1$ *est à la zone du coin de caffé*.

Nous avons utilisé l'éditeur d'automate Galaxy comme le montre la figure 4.10 pour modéliser ces deux modèles des scénarios. Pour obtenir le modèle global qui est la mise en parallèle de ces deux sous modèles, nous avons utilisé les outils décrits précédemment. Nous avons construits les représentations respectives de ces sous deux modèles sous forme d'automates de Mealy, au format "*.blif*". Ensuite, l'outil *Merge_blif* a réalisé la mise en parallèle des deux automates et généré l'automate résultat, également au format "*.blif*". Enfin, l'outil *Blifto* nous a permis de générer la représentation en "*c*" du modèle global, pour dialoguer avec la machine d'exécution.

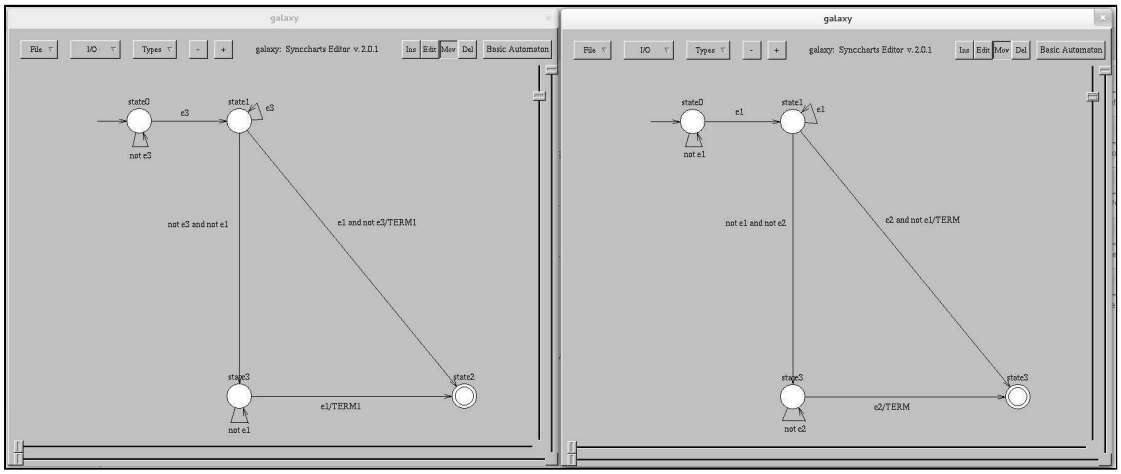


Figure 4.10 — Modèles de deux scénarios "$Before$" synchrones ($e3$ $Before$ $e1$, $e1$ $Before$ $e2$) en utilisant l'outil Galaxy.

Nous simulons ces modèles en utilisant l'outil *Blif_simul*, comme le montre la figure 4.7.

Figure 4.11 — Simulation du fonctionnement du modèle de deux scénarios "$Before$" synchrones ($e3$ $Before$ $e1$, $e1$ $Before$ $e2$) en utilisant l'outil $Blif\_simul$.

Rouge signifie non évalué, sans coloration signifie absent et bleu signifie présent.

En suivant le même démarche que pour les deux scénarios précédents, nous construisons la table de correspondance entre les signaux attendus de type booléen du code "$.c$", généré par l'outil $Autom2circuit$, et les objets concrets que considèrent la machine d'exécution, comme le montre le tableau 4.3.

Comme le montre l'exécution de la machine dans la figure 4.12, au début, les signaux attendus suite à l'exécution à vide de l'automate correspondent aux évènements : "$Person\_Inside\_Zone\_UseReandingTable$" et "$Person\_Inside\_Zone\_Entrance$". D'une manière constante la machine déclenche une alarme (TERM1) si les données physiques présentent une apparition de l'évènement "$Person\_Inside\_Zone\_Entrance$" (qui correspond à e3) avant l'évènement "$Person\_Inside\_Zone\_UseReandingTable$" (qui correspond à e1) . De même une autre alarme (TERM) est déclanchée si les données physiques présentent une apparition de l'évènement "$Person\_Inside\_Zone\_UseReandingTable$" (qui correspond à e1) avant "$Person\_Inside\_Zone\_UseCoffeeCorner$" (qui correspond à e2). Conformément à la sémantique du parallèle synchrone, le scénario est terminé et reconnu lorsque les deux alarmes ont été émises.

Variables booléennes	Objets
e1	Person_Inside_Zone_UseReandingTable (p1,zoneUseReadingTable)
e2	Person_Inside_Zone_UseCoffeeCorner (p1,zoneUseCoffeeCorner)
e3	Person_Inside_Zone_Entrance (p1,Entrance)

Tableau 4.3 — Tableau de correspondance du modèle de deux scénarios "$Before$" synchrones et parallèle.

Figure 4.12 — Réaction de la machine d'exécution vis-à-vis les deux scénarios "$Before$" synchrones et parallèle.

Nous avons extrait du fichier *log* (voir figure 4.13) quelques portions, dans lesquelles il y a une succession des calculs des instants logiques et les évènements "$Inputs$" construisant cet instant.

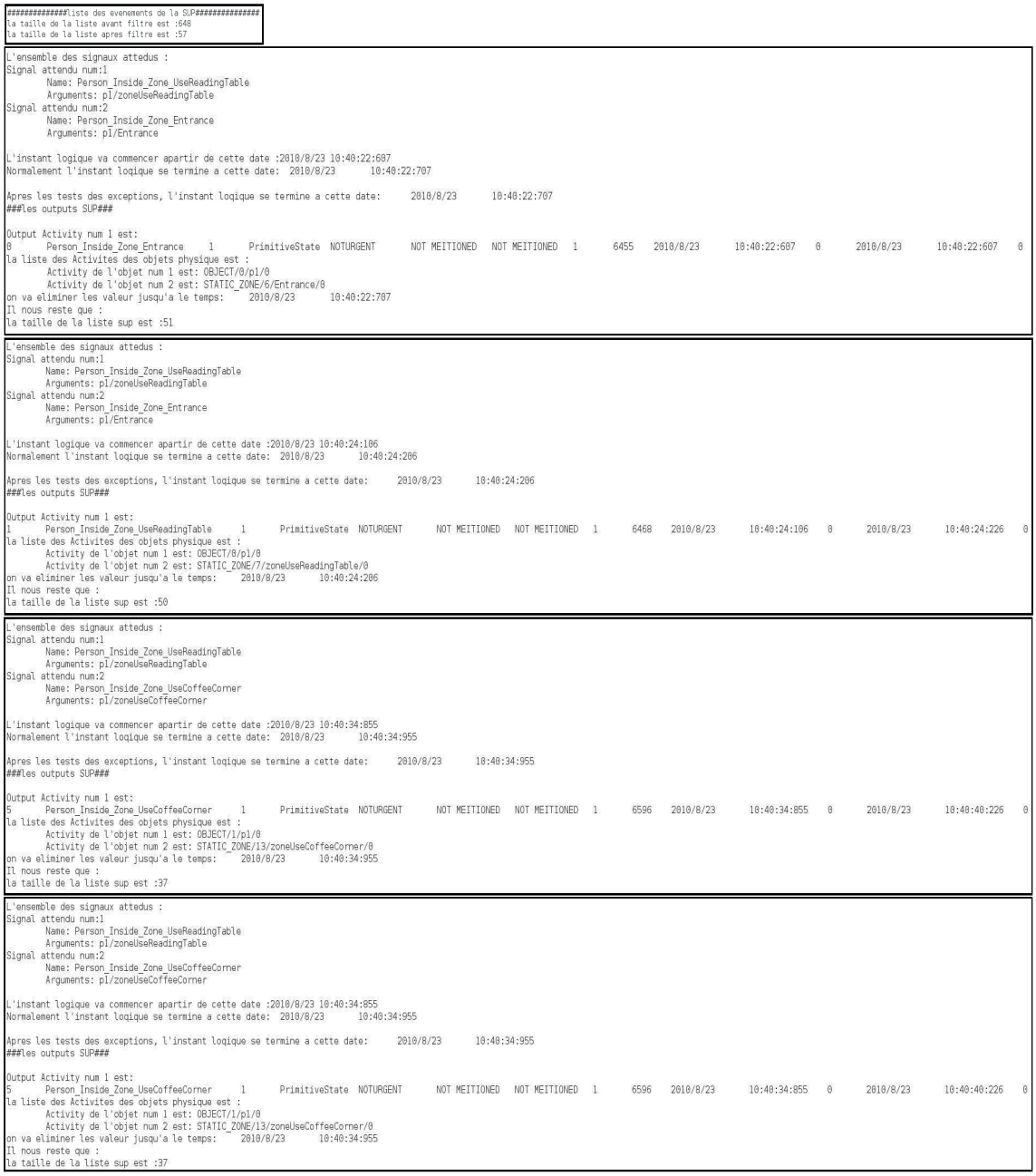


Figure 4.13 — Fichier *log* de sortie : Réaction de la machine d'exécution vis-à-vis les deux scénarios "$Before$" synchrones et parallèle.

4.6 Limites et améliorations

À mesure que l'étude avançait, nous avons pu observer quelques limitations à ce travail, parmi lesquelles :

- Un système de reconnaissance d'activités est un système qui réagi immédiatement selon les changements des statuts des signaux : il est sensible à la présence et à l'absence. De cette façon, nous avons adapté au début de notre étude une méthode qui consiste à considérer que la fin de premier signal n'appariant une autre fois dans un Δt choisi, mais le fiat qu'un évènement est caractérisé par une date de fin il se peut que cette dernière être en retard si Δt est achevé, donc nous serons obligé à chaque fois de retardé le fin de l'instant jusqu'au la fin de cet évènement, dans ce cas si le même évènement réapparaisse avant la fin de la première apparition nous risquons de retardé l'instant logique et ce démarche peut être fait infiniment. Pour cela nous avons décidé de traité les trois exceptions citées précédemment ;
- Pour tester le bon fonctionnement de la machine d'exécution proposée, nous avons essayé d'avoir des vrais valeurs modélisant une scène du monde réel de diverses sources (capteurs environnementales et plate-forme SUP) et vu les contraintes de temps nous avons obligé de testé q'avec des donnée de SUP provenant du centre hospitalier universitaire de NICE.

Bien que la machine ait réussi à calculer les instants logiques en passant d'une interface asynchrone en une autre synchrone, plusieurs améliorations de cette solution proposée sont envisageables :

- En premier lieu, la machine d'exécution peut être un outil plus fiable, générique, et utilisable pour satisfaire n'importe quel besoin de passage de l'asynchrone en synchrone, pour cela nous devons modifier le mode de lecture des données (Flex/Bison : grammaire statique) ;
- En deuxième lieu, nous pouvons proposé d'autres fonctionnalités plus intelligentes, de telle façon que la machine d'exécution pourra gérer l'incompatibilité des évènements, ceci sert à raffiner l'instant logique.

4.7 Conclusion

Ce chapitre a été consacré pour tester la réaction de la machine d'exécution vis-à-vis des scénarios bien définis ($Before$, $Overlaps$ et un modèle de deux $Before$ synchrones), en présence des données élaborées au centre hospitalier de Nice. En effet, ces scénarios sont modélisés à l'aide de Galaxy/Autom2circuit, outils développés par Monsieur Daniel Gaffé dans le cadre de recherche. Enfin nous avons cité les problèmes rencontrés au cours de la conception de la machine d'exécution, et nous avons proposé des idées servant de l'améliorer.

Conclusion générale

L'objectif premier de ce travail était de développer une machine d'exécution qui joue le rôle d'une interface intelligente, écoute des données provenant, d'une manière asynchrone, de la plateforme de vision cognitive SUP du projet Stars ainsi que d'autres capteurs environnementaux et communique avec des systèmes de reconnaissance d'activités s'appuyant sur un modèle synchrone.

Dans le premier chapitre de ce rapport, nous avons présenté l'équipe-projet d'accueil, STARS d'INRIA-Sophia antipolis. Nous avons abordé dans le deuxième chapitre, d'une part, les concepts de la reconnaissance d'activités en présentant des travaux existants. D'autre part, nous introduisons les différents aspects de la programmation synchrone des systèmes réactifs. Dans le chapitre suivant, nous avons présenté la machine d'exécution proposée, sa spécification, sa conception et sa mise en œuvre. Enfin, au niveau du cinquième chapitre, nous avons évalué cette solution et nous avons testé la machine d'exécution proposée en utilisant un ensemble de scénarios réels.

Ce stage était fort intéressant, tout d'abord, il m'a permis d'améliorer mes compétences d'élaborer une recherche bibliographique approfondie et synthétiser mon propre travail. Ensuite, Il a été très bénéfique en matière d'acquis techniques et en matière de confrontation de problèmes réels. Enfin, nous remercions encore une fois toutes les personnes qui ont contribué de près et de loin à la réussite de ce travail.

Bibliographie et Netographie

[1] R. Chellappa A. Chowdhury. Advanced a factorization approach for activity recognition. 2003.

[2] C. André. Computing syncchart réactions. *Electronic Notes in Theoretical Computer Science*, 9(88) :3–19, 2004.

[3] Charles André. *Systèms réactifs et programmation synchrone*. 1998.

[4] Charles André. Modélisation des systèmes réactifs par une approche graphique synchrone : Syncharts, 2003.

[5] G. Berry. Preemption in concurrent systems. *Proc FSTTCS, Lecture notes in Computer Science*, 761 :72–93, 1993.

[6] Gérard Berry. The esterel v5.91 system manual. http://www.esterel-technologies.com/v3/?id=18162, 2000.

[7] W. Elmenreich. Smart transducers-principles, communications, and configuration. 2003.

[8] D. Gaffe et A. Resouche. Les langages synchrones. http://www-sop.inria.fr/members/Annie.Ressouche/clem.html.

[9] D. M. Gavrila. The visual analysis of human movement : A survey. 1998.

[10] Jean-Paul Marmorat Gerard Berry, Xavier Fornari. The esterel language. http://www-sop.inria.fr/meije/esterel/esterel-eng.html.

[11] L'équipe Gerhome. Projet gerhome. http://gerhome.cstb.fr/, 2013.

[12] P. Grossmann. Multisensor data fusion. 1998.

[13] N. Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer Academic Publishers, Amsterdam, 1993.

[14] H. hristensen J. Bardram. Open issues in activity-based and task-level computing. 2004.

[15] Xuan Hoa Binh LE. *Reconnaissance des comportements d'une personne âgée vivant seule dans un habitat intelligent pour la santé*. PhD thesis, Université Joseph Fourier de Grenoble, 2008.

[16] J. Ferryman N. Carter, D. Young. A combined bayesian markovian approach for behaviour recognition. 2006.

[17] A. Schmidt N. Kern, B. Schiele. Multi-sensor activity context detection for wearable computing. 2003.

[18] A. Garg N. Oliver, E. Horvitz. Layered representations for human activity recognition. 2002.

[19] N. Halbwachs et J. A. Plaice P. Caspi, D. Pilaud. Lustre : A declarative language for programming synchronous systems. 1987.

[20] A. Halme P. Harmo, T. Taipalus. Needs and solutions-home automation and service robots for the elderly and disabled. 2005.

[21] Virginia Papailiooulou. *Test automatique de programmes Lustre/SCADE*. PhD thesis, Université De Grenoble, Ecole Doctorale MSTII(Mathématiques, Sciences et Technologies de l'Information, Informatique), 2010.

[22] I. Robert-Bobée. Projections de population 2005-2050 vieillissement de la population en france métropolitaine, 2006.

[23] K. Haigh V. Guralnik. Learning models of human behavior with sequential patterns. 2002.

[24] Nadia Zouba Valentin. *Fusion multi-capteurs pour la reconnaissance d'activités des personnes agées á domicile*. PhD thesis, Université de Nice-Sophia Antipolis, École doctorale STIC, 2010.

[25] A. Bobick Y. Ivanov. Recognition of visual activities and interactions by stochastic parsing. 2000.

[26] Irani Zelnik-Manor. Event-based video analysis. 2001.

ANNEXE A

Fonctionnement de la plate-forme SUP

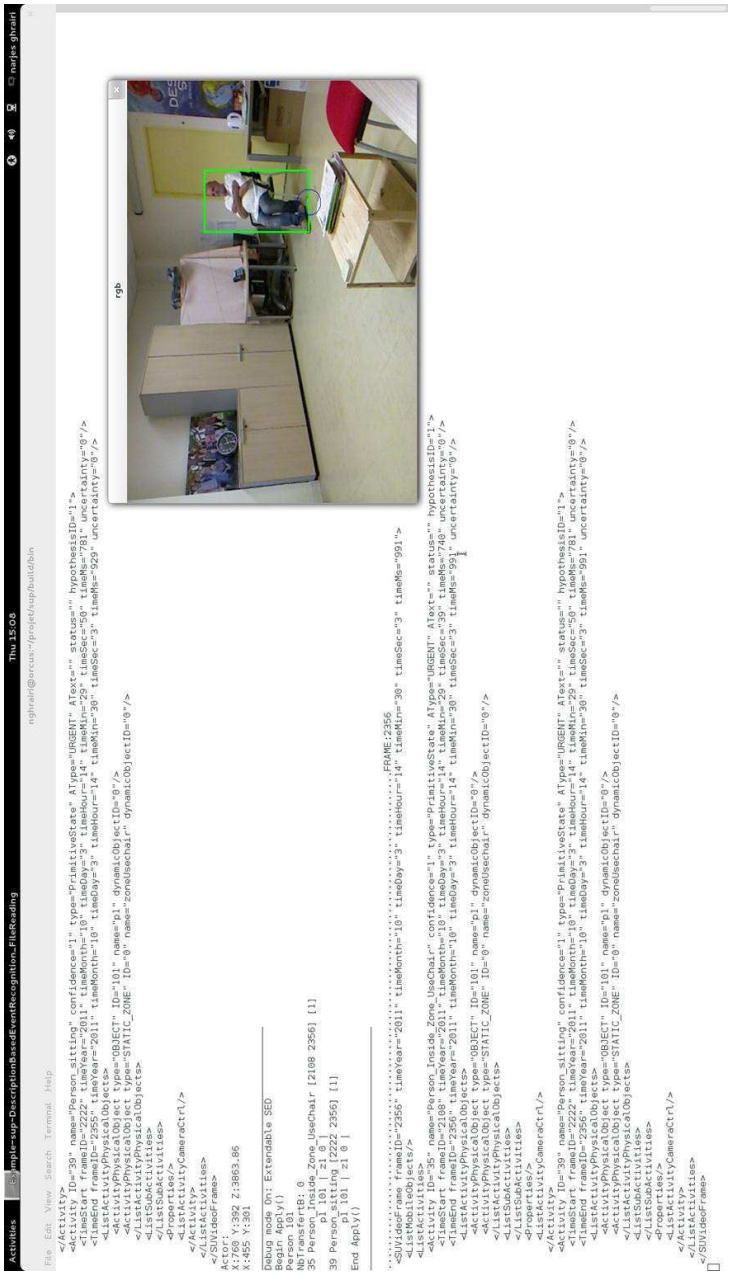


Figure A.1 — Exemple 1 de fonctionnement de la plate-forme SUP et génération des évènements vidéo.

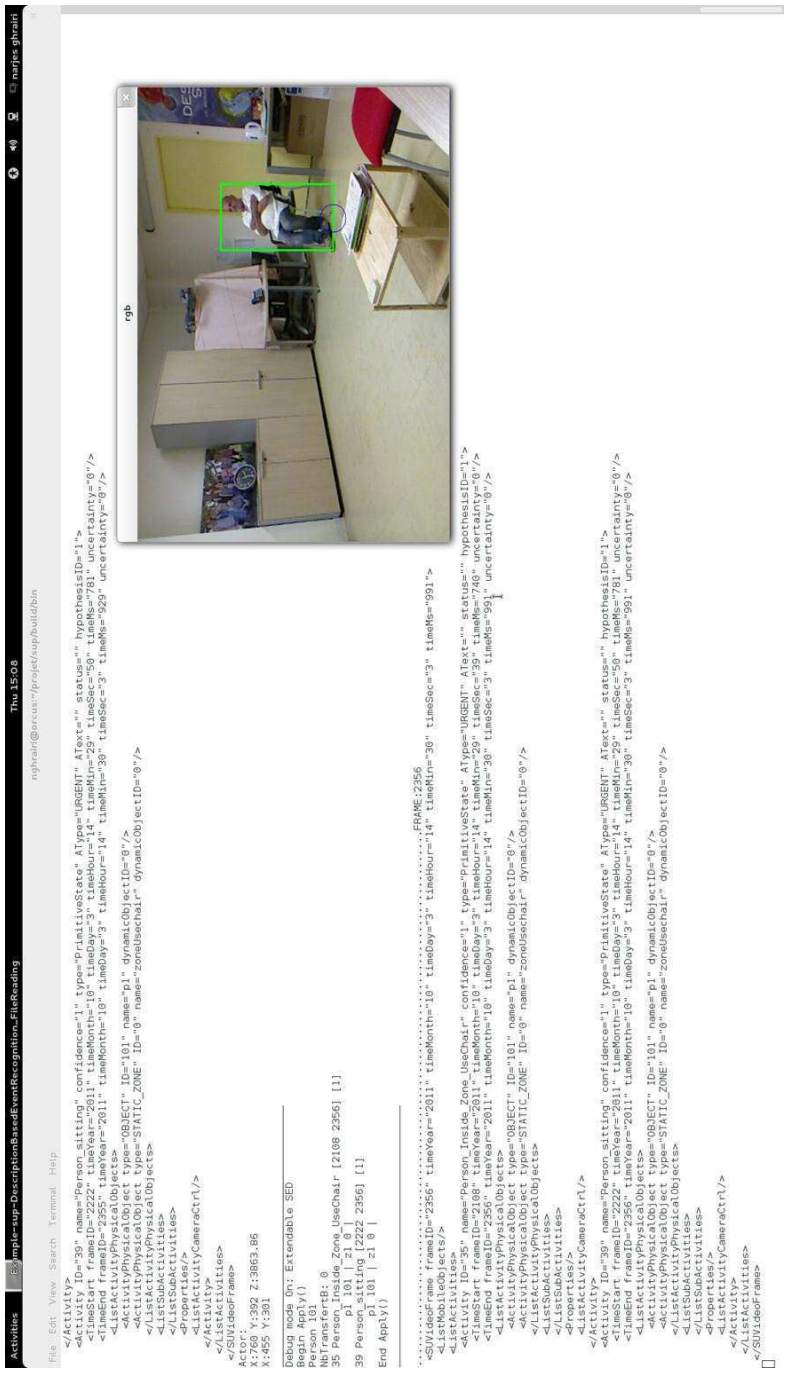


Figure A.2 — Exemple 2 de fonctionnement de la plate-forme SUP et génération des évènements vidéo.

ANNEXE B

Un exemple de fichier XML sortie de la plate-forme SUP

< ?xml version="1.0" encoding="UTF-8" ?>
<Cofriend version="3.00">
<SUVideoFrame frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="2" name="exercisewalking" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
<Activity ID="1" name="Person_farFrom_chair" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="3" name="z1" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="39" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54"

timeSec="40" timeMs="117">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="Person_Inside_Zone_ExerciseWalking" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="39" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="117" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="2" name="exercisewalking" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="40" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="209">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="Person_Inside_Zone_ExerciseWalking" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="40" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="209" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="2" name="exercisewalking" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="41" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="318">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="Person_Inside_Zone_ExerciseWalking" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="41" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="318" uncertainty="0"/>
<ListActivityPhysicalObjects>

<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="2" name="exercisewalking" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="42" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="416">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="Person_Inside_Zone_ExerciseWalking" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="42" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="416" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="2" name="exercisewalking" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="43" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="517">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="Person_Inside_Zone_ExerciseWalking" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="43" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="517" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="2" name="exercisewalking" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>

<SUVideoFrame frameID="44" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="619">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="Person_Inside_Zone_ExerciseWalking" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="44" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="619" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="2" name="exercisewalking" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="45" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="711">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="Person_Inside_Zone_ExerciseWalking" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="45" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="711" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="2" name="exercisewalking" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="46" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="782">
<ListMobileObjects/>
<ListActivities>
<Activity ID="0" name="Person_Inside_Zone_ExerciseWalking" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="38" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="16" uncertainty="0"/>
<TimeEnd frameID="46" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="782" uncertainty="0"/>

<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="2" name="exercisewalking" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="47" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="888">
<ListMobileObjects/>
<ListActivities>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="48" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="40" timeMs="987">
<ListMobileObjects/>
<ListActivities>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="49" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="41" timeMs="80">
<ListMobileObjects/>
<ListActivities>
</ListActivities>
</SUVideoFrame>
<SUVideoFrame frameID="50" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="41" timeMs="181">
<ListMobileObjects/>
<ListActivities>
<Activity ID="2" name="Person_Inside_Zone_UseChair" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="50" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="41" timeMs="181" uncertainty="0"/>
<TimeEnd frameID="50" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="41" timeMs="181" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="0" name="zoneUsechair" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
<Activity ID="3" name="Person_sitting" confidence="1" type="PrimitiveState" AType="URGENT" AText="" status="" hypothesisID="1">
<TimeStart frameID="50" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="41" timeMs="181" uncertainty="0"/>

<TimeEnd frameID="50" timeYear="2011" timeMonth="10" timeDay="3" timeHour="14" timeMin="54" timeSec="41" timeMs="181" uncertainty="0"/>
<ListActivityPhysicalObjects>
<ActivityPhysicalObject type="OBJECT" ID="104" name="p1" dynamicObjectID="0"/>
<ActivityPhysicalObject type="STATIC_ZONE" ID="0" name="zoneUsechair" dynamicObjectID="0"/>
</ListActivityPhysicalObjects>
<ListSubActivities>
</ListSubActivities>
<Properties/>
<ListActivityCameraCtrl/>
</Activity>
</ListActivities>
</SUVideoFrame>
</Cofriend>

ANNEXE C

Un exemple de fichier d'entrée des capteurs (cas d'un capteur TOR)

03 :11 :2011 :14 :33 :44 :560/03 :11 :2011 :14 :33 :44 :56/Tor/sensor1/event1/1
03 :11 :2011 :14 :33 :44 :65/03 :11 :2011 :14 :33 :44 :65/Tor/sensor1/event1/1
03 :11 :2011 :14 :33 :44 :66/03 :11 :2011 :14 :33 :44 :66/Tor/sensor1/event1/1
03 :11 :2011 :14 :33 :44 :66/03 :11 :2011 :14 :33 :44 :66/Tor/sensor1/event1/2
03 :11 :2011 :14 :33 :44 :155/03 :11 :2011 :14 :33 :44 :155/Tor/sensor1/event2/2
03 :11 :2011 :14 :33 :44 :59/03 :11 :2011 :14 :33 :44 :159/Tor/sensor1/event2/2
03 :11 :2011 :14 :33 :44 :222/03 :11 :2011 :14 :33 :44 :222/Tor/sensor1/event1/1
03 :11 :2011 :14 :33 :55 :65/03 :11 :2011 :14 :33 :55 :65/Tor/sensor1/event1/1
03 :11 :2011 :14 :33 :55 :787/03 :11 :2011 :14 :33 :55 :787/Tor/sensor1/event1/1
03 :11 :2011 :14 :33 :56 :66/03 :11 :2011 :14 :33 :56 :66/Tor/sensor1/event1/2
03 :11 :2011 :14 :33 :56 :801/03 :11 :2011 :14 :33 :56 :801/Tor/sensor1/event2/2
03 :11 :2011 :14 :33 :57 :1/03 :11 :2011 :14 :33 :57 :1/Tor/sensor1/event2/2
03 :11 :2011 :14 :33 :57 :2/03 :11 :2011 :14 :33 :57 :2/Tor/sensor1/event1/1
03 :11 :2011 :14 :33 :57 :23/03 :11 :2011 :14 :33 :57 :23/Tor/sensor1/event1/1
03 :11 :2011 :14 :33 :57 :401/03 :11 :2011 :14 :33 :57 :401/Tor/sensor1/event1/1
03 :11 :2011 :14 :33 :57 :660/03 :11 :2011 :14 :33 :57 :660/Tor/sensor1/event1/2
03 :11 :2011 :14 :33 :57 :70/03 :11 :2011 :14 :33 :57 :702/Tor/sensor1/event2/2
03 :11 :2011 :14 :33 :57 :801/03 :11 :2011 :14 :33 :57 :801/Tor/sensor1/event2/2
03 :11 :2011 :14 :33 :57 :901/03 :11 :2011 :14 :33 :57 :901/Tor/sensor1/event1/
03 :11 :2011 :14 :34 :01 :65/03 :11 :2011 :14 :34 :01 :65/Tor/sensor1/event1/1
03 :11 :2011 :14 :34 :01 :62/03 :11 :2011 :14 :34 :01 :62/Tor/sensor1/event1/1
03 :11 :2011 :14 :34 :01 :66/03 :11 :2011 :14 :34 :01 :66/Tor/sensor1/event1/2
03 :11 :2011 :14 :34 :01 :155/03 :11 :2011 :14 :34 :01 :155/Tor/sensor1/event2/2
03 :11 :2011 :14 :34 :01 :159/03 :11 :2011 :14 :34 :01 :159/Tor/sensor1/event2/2
03 :11 :2011 :14 :34 :10 :40/03 :11 :2011 :14 :34 :10 :40/Tor/sensor1/event1/1
03 :11 :2011 :14 :34 :10 :50/03 :11 :2011 :14 :34 :10 :50/Tor/sensor1/event1/1
03 :11 :2011 :14 :34 :10 :62/03 :11 :2011 :14 :34 :10 :62/Tor/sensor1/event1/1
03 :11 :2011 :14 :34 :10 :66/03 :11 :2011 :14 :34 :10 :66/Tor/sensor1/event1/2
03 :11 :2011 :14 :34 :15 :03/03 :11 :2011 :14 :34 :15 :03/Tor/sensor1/event2/2
03 :11 :2011 :14 :34 :15 :10/03 :11 :2011 :14 :34 :15 :10/Tor/sensor1/event2/2
03 :11 :2011 :14 :34 :15 :56/03 :11 :2011 :14 :34 :15 :56/Tor/sensor1/event1/1
03 :11 :2011 :14 :34 :30 :65/03 :11 :2011 :14 :34 :30 :65/Tor/sensor1/event1/1
03 :11 :2011 :14 :34 :30 :62/03 :11 :2011 :14 :34 :30 :62/Tor/sensor1/event1/1
03 :11 :2011 :14 :34 :30 :66/03 :11 :2011 :14 :34 :30 :66/Tor/sensor1/event1/2
03 :11 :2011 :14 :34 :44 :155/03 :11 :2011 :14 :34 :44 :155/Tor/sensor1/event2/2
03 :11 :2011 :14 :34 :44 :159/03 :11 :2011 :14 :34 :44 :159/Tor/sensor1/event2/2