République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Chadli Bendjedid
Faculté des Sciences et de la Technologie
Département d'Informatique

الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة الشاذلي بن جديد
كلية العلوم و التكنولوجيا
قسم الاعلام الالي

MEMOIRE

Présenté par

BOUNADEUR ZAKARIA

Pour l'obtention de diplôme de

MASTER

Filière : Informatique

Spécialité : Systèmes Informatiques Intelligents

Thème

PRISM : un outil de vérification probabiliste des systèmes embarqués- Etude de cas-

Soutenu le : 15 / 09 / 2022 Devant le Jury composé de :

Qualité	Nom et Prénom	Grade	Université
Président	Mr. BENTRAD SASSI	MCB	Chadli Bendjedid El-Tarf
Rapporteur	**Mme. FERROUM A.**	**MCB**	**Chadli Bendjedid El-Tarf**
Examinateur	Mr. BETOUL	MCB	Chadli Bendjedid El-Tarf

Année Universitaire : 2021/2022

Remerciements

*Je tiens à remercier **Mme FERROUM Assia**, qui est mon encadrant, pour tout. Je lui suis reconnaissant de m'avoir encadré, éduqué et orienté vers ce cursus.*

Je remercie sincèrement tous les professeurs, conférenciers et autres personnes dont les remarques, réflexions, suggestions et critiques ont contribué à façonner mon point de vue et qui ont accepté de parler avec moi et de répondre à mes questions au cours de mon étude.

*Je remercie **mes parents** dévoués, qui m'ont soutenu de toutes les manières. J'apprécie le soutien de **mon frère et de mes sœurs**.*

*Je voudrais terminer en remerciant **mes copains** d'avoir été à mes côtés tout au long. Leur soutien indéfectible et leur inspiration ont été vraiment bénéfiques.*

J'exprime mon appréciation, mon admiration et mes remerciements à chacun de ces orateurs.

Dédicace

Je suis reconnaissant à ALLAH (mon dieu) de m'avoir fourni les compétences nécessaires pour penser et écrire, le courage d'y croire et la persévérance pour atteindre mon objectif.

Je vous confie cet effort.

Pour ma maman dévouée

Pour mon papa adoré

Tout au long de mes études, je suis reconnaissant pour tous leurs sacrifices, leur amour, leur gentillesse, leur soutien et leurs prières.

Mes frères

pour leur inspiration continue et leur aide morale.

A mes copains chéris

A toute ma famille

A tous mes autres potes

A tous ceux que j'aime et à tous ceux que j'adore.

ZAKARIA.

Table des matières

Liste des figures

Liste des tableaux

Liste des listings

Liste des acronymes

RTOS	**real-time operating system**
UI	**interface utilisateur**
CPU	**central processing unit**
BDDs	***Binary Decision Diagrams***
GUI	**interfaces utilisateur graphiques**
ASIC	***Application-Specific Integrated Circuit***
CSRL	***Continuous Stochastic Reward Logic***
DTMC	***Discrete-Time Markov Chains***
CTMC.	***Continuous-Time Markov Chains***
HMM.	***Hidden Markov Model***
PCTL	***Probabilistic Computation Tree Logic***
POCTL	***Probabilistic Observation branChing-time Temporal Logic***
PDA	***Probabilistic Automata***
MDP	***Markov Decision Process***
APC	***système de prévision des accidents***
PAs	***Personal Digital Assistant***
PRCTL.	***Probabilistic Reward Computation Tree Logic***
PTA	***Automates Temporisés Probabilistes***
TR	***Temps Réel***

Introduction Générale

En automatisant le fonctionnement des systèmes, les logiciels embarqués jouent un rôle de plus en plus important dans la gestion de nombreux systèmes. Ces systèmes sont utilisés dans des produits grand public tels que les gadgets électriques comme les téléphones portables, les PDA, les caméras et d'autres applications à haute valeur ajoutée (aérospatiale, aéronautique, militaire, etc.). Cependant, il existe d'autres aides à la conduite automobile (direction assistée, freinage automatique, en attendant les systèmes dits "drive-by-wire").

Alors que certains systèmes, comme ceux des avions, sont essentiels pour la sécurité, d'autres, comme ceux des montres ou des machines à laver, ne le sont pas.

1. Contexte du projet et problématique

Nous utilisons fréquemment une variété de kits et de circuits électriques et électroniques dans la vie quotidienne qui ont été créés à l'aide de la technologie des systèmes embarqués. Par rapport aux ordinateurs personnels, les erreurs de programmation des systèmes embarqués ont des répercussions plus graves.

Les accidents de la route font chaque année 1,24 million de morts et des dizaines de millions de blessés, selon l'Organisation mondiale de la santé (OMS), ce qui en fait la septième cause de décès d'ici 2030.

2. Motivations

Pour prévenir des incidents catastrophiques pouvant occasionner des pertes humaines et matérielles importantes, il est impératif de démontrer la sûreté de fonctionnement de ces technologies avant leur déploiement, notamment pour celles pouvant porter atteinte à la santé publique.

3. Objectifs

La vérification formelle est une approche importante pour résoudre ces problèmes. Il est courant d'aborder cette conformité à l'aide de méthodes telles que la vérification de modèle ou la vérification de modèle. Un model-checker, tel que PRISM, est un outil de vérification qui utilise le model-checking.

Faire une étude de cas d'un système embarqué probabiliste sélectionné à l'aide de PRISM est l'objectif de cet effort.

4. Contenu du mémoire

Cette thèse est divisée en quatre chapitres.

Le concept de systèmes embarqués est introduit dans le premier chapitre.

L'état de l'art du model-checking et sa forme probabiliste sont présentés dans le chapitre 2, « Model-checking probabiliste ». La couverture des nombreuses exécutions du système, le processus de vérification lui-même, le haut degré d'expertise requis pour son utilisation et la facilité de mise en œuvre semblent tous être extrêmement bien équilibrés par cette approche, à notre avis. Le formalisme de modélisation et le modèle HMM sont présentés dans ce chapitre.

Le model-checker PRISM fait l'objet de notre investigation, qui est présentée au chapitre 3.

Les résultats expérimentaux d'une implémentation de Prism sur un système embarqué APS réel sont présentés dans les sous-sections du chapitre 4 "Etude de cas" sur la présentation du système APS (Accident Prediction System), la sélection des propriétés à vérifier, la spécification des ces propriétés sélectionnées dans la logique POCTL, et la vérification par PRISM.

Cette thèse se termine par un résumé des idées clés et des suggestions pour des recherches futures.

Chapitre 1 : Etat de l'Art

1. Introduction

Dans le monde dans lequel nous vivons aujourd'hui, les logiciels sont essentiels. La majorité de ces logiciels, tels que ceux que l'on trouve dans les biens électroniques, les transports, les automobiles et les télécommunications, fonctionnent à l'intérieur des infrastructures et dans les systèmes et équipements que nous utilisons quotidiennement. Ce logiciel, qui s'immisce dans notre vie quotidienne, est en grande partie conçu pour faciliter l'existence humaine en éliminant les tâches laborieuses, compliquées et fastidieuses.

La majeure partie de ce logiciel émet des jugements en fonction du passage du temps et en est consciente. Le terme « système temps réel » est alors utilisé. Dans ce dernier cas, le respect de tous les délais avant l'exécution effective du système est tout aussi crucial que l'exactitude du résultat.

2.Historique :

L'ordinateur de guidage Apollo, créé dans les années 1960 par le Dr Charles Stark Draper au Massachusetts Institute of Technology pour le programme Apollo, a été le premier système informatique embarqué en temps réel contemporain. Les calculs critiques pour la mission du module de commande Apollo et du module lunaire sont fournis par l'ordinateur de guidage Apollo, qui a été créé pour collecter des données de manière autonome.

la National Engineering Manufacturers Association a publié une norme pour les microcontrôleurs programmables en 1978, améliorant la conception des systèmes embarqués. Au début des années 1980, les composants du système de mémoire, d'entrée et de sortie avaient été intégrés dans la même puce que le processeur, formant un microcontrôle. Au début des années 1980, les composants du système de mémoire, d'entrée et de sortie avaient été intégrés dans la même puce que le processeur. CPU, créant un microcontrôleur. Cette avancée dans la conception de systèmes embarqués a amélioré l'architecture du système embarqué.

Des lecteurs de cartes de crédit aux téléphones portables en passant par les feux de circulation et les thermostats, le système embarqué basé sur un microcontrôleur sera plus tard intégré à toutes les facettes de la vie quotidienne des clients.

3.Définition des systèmes embarqués :

système embarqué est une combinaison de matériel informatique et de logiciels créée dans un but particulier. De plus, les systèmes embarqués peuvent fonctionner dans le cadre d'un système plus vaste. Les systèmes peuvent être programmables ou ne remplir que certaines fonctions. Un système embarqué peut être trouvé dans les machines industrielles, l'électronique grand public, les équipements du secteur agricole et de transformation, les véhicules, les appareils médicaux, les appareils photo, les montres numériques, les appareils électroménagers, les avions, les distributeurs automatiques, les jouets et les appareils mobiles.

Même s'il s'agit de systèmes informatiques, les systèmes embarqués peuvent avoir de simples interfaces utilisateur graphiques (GUI), comme celles que l'on voit dans les appareils mobiles, ou ils peuvent ne pas avoir d'interface utilisateur (UI), comme celles que l'on trouve dans les appareils destinés à exécuter un travail spécifique. Le bouton, la LED (diode électroluminescente) et la détection de l'écran tactile sont des exemples d'interfaces utilisateur. Certains systèmes utilisent également des interfaces utilisateur à distance.

Il y a trois parties dans un système embarqué :

- Il a du matériel
- Il a un logiciel pour les applications.
- Son système d'exploitation en temps réel (RTOS) gère le logiciel d'application et offre un cadre pour permettre au processeur d'exécuter une tâche conformément à la planification en respectant un plan pour limiter les latences. Le RTOS spécifie le fonctionnement du système. Il établit les règles d'exécution du programme d'application. Il est possible qu'un petit système embarqué ne dispose pas d'un RTOS.

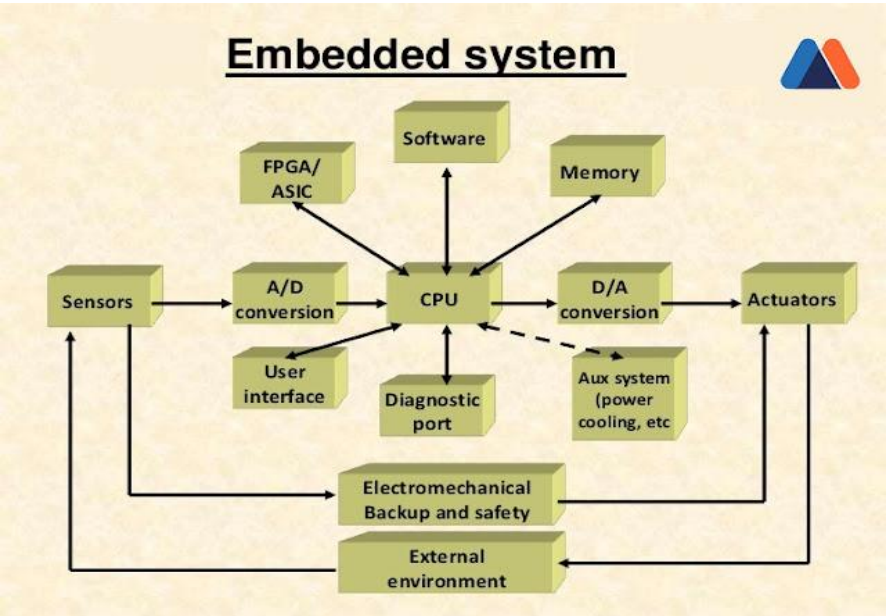


Figure 1. définition de systeme embarqué

4.Caractéristiques d'un système embarqué :

Un système embarqué n'exécute souvent qu'un seul type de fonction et le répète souvent. Un réveil fonctionne toujours comme un réveil, par exemple.

Les métriques de conception sont étroitement limitées dans tous les systèmes informatiques, mais elles pourraient l'être davantage dans les systèmes embarqués. Les métriques de conception sont un moyen d'évaluer le coût, la taille, la puissance et d'autres caractéristiques d'une implémentation. Il doit être suffisamment petit pour tenir sur une seule puce, suffisamment rapide pour analyser les données en temps réel et suffisamment économe en énergie pour prolonger la durée de vie de la batterie.

Réactif et en temps réel De nombreux systèmes embarqués doivent constamment réagir aux changements de leur environnement et calculer instantanément des sorties spécifiques. Prenons l'exemple d'un régulateur de vitesse dans une voiture ; il analyse et réagit en permanence aux capteurs de vitesse et de freinage. Il doit calculer à plusieurs reprises les accélérations et les décélérations dans un laps de temps défini ; si le calcul est retardé, l'automobile peut ne pas être contrôlée.

Il doit être basé sur un microprocesseur, via un microprocesseur ou un microcontrôleur.

Mémoire : Étant donné que son logiciel est souvent intégré dans la ROM, il doit avoir une mémoire. Le PC ne nécessite aucune mémoire supplémentaire.

Il doit avoir des périphériques qui sont connectés afin de relier les périphériques d'entrée et de sortie.

Systèmes avec HW-SW Plus de fonctionnalités et de flexibilité sont fournies par le logiciel. Le matériel est utilisé pour la sécurité et la performance.

5.Système embarqué : cas d’application :

Comme vous le savez probablement déjà, les systèmes embarqués sont utilisés pour exécuter des fonctions à l'intérieur d'un appareil. Mais de quels devoirs parle-t-on ? Tout, y compris l'équipement médical, les appareils électroménagers, l'informatique et le transport, dépend véritablement de l'appareil dans lequel le système est intégré. Nous utilisons des systèmes embarqués tous les jours sans même le reconnaître car ils sont employés dans de nombreuses industries différentes. HDTV, disque dur, console de jeu, imprimante, régulateur de vitesse, portail automatisé, lave-vaisselle Il existe plusieurs gadgets qui ont un ou plusieurs systèmes embarqués.

Le système embarqué dans le cas du portail automatique contient des capteurs qui peuvent identifier les choses. Lorsqu'une voiture approche, le système reconnaît l'automobile et ordonne au moteur d'ouvrir la porte.

Les systèmes embarqués peuvent réaliser une gamme de fonctions selon le matériel et les domaines d'application, ouvrant d'innombrables possibilités de créativité. Comparés aux plus complexes, qui peuvent être mis en réseau et qui disposent d'une interface tactile, les plus basiques ont simplement des boutons simples.

Figure.2 :Système embarqué : cas d'application

6. Différences avec un ordinateur de bureau :

Chaque application nécessite une IHM (Interface Homme Machine) différente. Des lumières et des boutons peuvent être utilisés, ou il peut être plus sophistiqué comme un écran tactile.
Il doit être robuste (étanche, aux chocs, etc.).

• Généralement, l'écran est plus petit et il n'y a pas de clavier.

• Le système embarqué dispose de périphériques et de capteurs adaptés à son utilisation. Cartes SD, mémoires flash, etc.

• Actionneurs, température et pression • Wifi, Bluetooth, un module GSM-GPS, etc (moteurs,...)

Les systèmes embarqués interagissent fréquemment avec leur environnement grâce à l'utilisation de capteurs et d'actionneurs, qui collectent des données de l'environnement et exécutent une tâche spécifique en réponse à l'analyse de ces données.

7.Architecture d'un système embarqué :

Cette conception varie selon les systèmes :

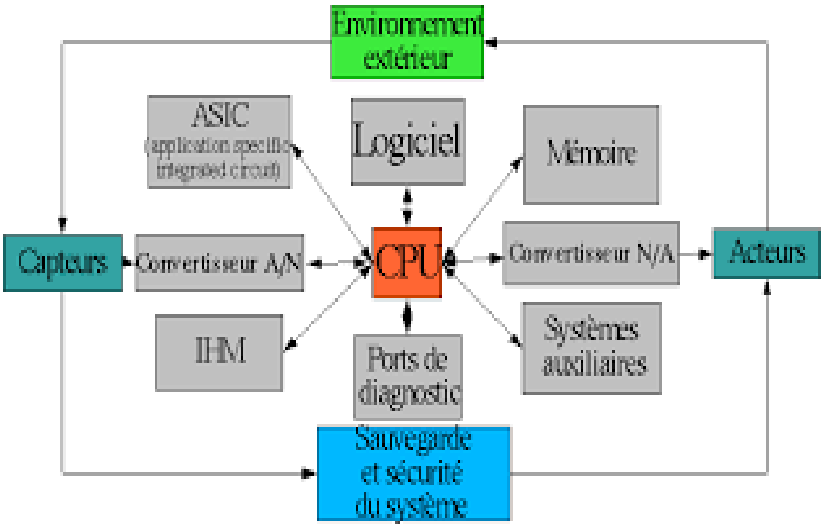


figure 3 :Architecture d'un système embarqué .

de nombreux systèmes embarqués autonomes et indépendants, par exemple, ne comportent pas de systèmes auxiliaires. L'architecture fondamentale, en revanche, est généralement constituée d'une unité centrale de traitement (CPU), d'un système d'exploitation (OS), que l'on ne trouve souvent que dans un logiciel particulier (ex : routeur), ou d'une boucle de données (ex : ABS). De même, l'interface IHM est fréquemment indisponible, bien qu'elle soit souvent nécessaire pour modifier ou tester le comportement du système.

Le fonctionnement du système peut être résumé comme suit :

Il accepte les informations du monde extérieur et les transforme en un signal numérique.

Les informations sont traitées par l'unité de traitement, qui se compose de la CPU, de la mémoire, du logiciel, de l'ASIC et peut-être d'autres systèmes.

Une fois le traitement terminé, une sortie est générée et transmise à la sortie, aux systèmes auxiliaires, aux ports de surveillance ou à l'IHM.

8.Composants du système embarqué :

CPU :

Il existe trois catégories de processeurs :

Les processeurs à objectif unique viennent en premier. C'est un circuit numérique destiné à exécuter un seul programme avec les caractéristiques suivantes :

- Il comprend simplement les composants dont nous avons besoin pour exécuter le programme.
- Il manque de mémoire.

Avantages:

- performances accélérées
- basse consommation, compacte
- De plus : Processeurs d'applications spéciales : Ce sont des dispositifs programmables qui, dans la majorité des situations, sont créés et construits pour exécuter un certain ensemble d'applications. Ils sont une combinaison du premier type de processeur que nous avons vu et du troisième type que nous verrons.
- lignes de données avec mémoire de programme améliorée.
- certaines unités fonctionnelles.

Avantages:

- assez adaptable; performance excellente
- petit et faible en puissance
- Troisièmement : la gestion des objectifs généraux

Ce sont des processeurs programmables qui sont produits pour être utilisés dans une variété d'applications. Les microprocesseurs sont ce qu'ils sont appelés. Voici leurs traits :

I. posséder une mémoire informatique
II. Il existe plusieurs registres publics, une ligne de données générique, une unité arithmétique et logique générale, ainsi qu'un grand nombre d'autres composants.

Avantages:

- Il produit des biens rapidement et à moindre coût, et est très polyvalent.
- Les processeurs Intel Pentium en sont une instance.

Unités de temporisation et de comptage : il s'agit de différents types de temporisateurs et de compteurs utilisés dans les systèmes embarqués pour synchroniser une tâche particulière.

Les unités de conversion des signaux de l'analogique au numérique et du numérique à l'analogique font partie des composants les plus cruciaux des systèmes embarqués. Conformément aux exigences des circuits installés dans le système numérique, il transforme les signaux analogiques en numériques et vice versa.

Les périphériques de sortie pour afficher les résultats et les messages ainsi que pour informer sur les états du système sont des exemples d'unités d'entrée et de sortie. Les panneaux LCD sont

les circuits les plus cruciaux. panneaux de sept pièces. Les LED sont des dispositifs de saisie de données qui peuvent être utilisés avec des claviers, des capteurs, etc.

La pertinence de ces composants s'est accrue récemment, d'autant plus que le monde entier passe aux technologies sans fil. Les plus cruciaux de ces circuits sont les émetteurs et récepteurs Bluetooth, ainsi que les émetteurs et récepteurs de signaux radio.

9.Propriétés:

Les capteurs et actionneurs qui collectent des données de l'environnement et exécutent une tâche spécifique en fonction de l'analyse de ces données sont fréquemment utilisés pour connecter des systèmes embarqués au monde extérieur.

Les principales caractéristiques des systèmes embarqués incluent :

- Fiabilité : Il s'agit de la probabilité qu'un système exécute sans défaillance la tâche qui lui a été déléguée.
- La maintenabilité est la capacité de remonter rapidement un système suite à un problème.
- Disponibilité : C'est la probabilité que le système soit opérationnel à un moment donné. Une excellente disponibilité nécessite une fiabilité et une maintenabilité élevées.
- La sécurité, ou l'assurance que des dommages ne résulteront pas d'une défaillance du système.
- Sécurité des données : Le système assure toute forme de communication qui nécessite une validation d'identité et s'assure que les données sensibles restent en sécurité.
- C'est abordable.
- Il doit être aussi léger que possible pour être efficace.
- Efficacité énergétique : il doit utiliser le moins d'énergie possible tout en remplissant sa fonction.
- Efficace en termes de taille de code de programme : étant donné que les systèmes embarqués ne disposent généralement pas d'un disque dur, toutes les instructions qui doivent être exécutées par le système doivent y être placées.
- Efficacité en temps réel : le système doit fonctionner dans des limites de temps réel tout en utilisant le moins de ressources physiques et le moins d'énergie possible pour exécuter la fonction requise.
- Ce sont des systèmes qui exécutent leurs tâches dans le temps imparti.

- Dans la majorité des cas, ce sont des systèmes hybrides, c'est-à-dire qu'ils intègrent à la fois des composants analogiques et numériques. Le premier emploie des valeurs de données numériques (intermittentes), tandis que le second utilise des valeurs de données analogiques.
- Ce sont des systèmes qui effectuent une seule tâche ; ils n'effectuent pas une variété de tâches.
- Ces systèmes sont parfois appelés ordinateurs internes car ils n'utilisent généralement pas les dispositifs de communication courants que l'on trouve dans les ordinateurs, tels que la souris, le pavé d'écriture et l'écran, mais utilisent plutôt des boutons, des molettes de commande et d'autres dispositifs. En conséquence, l'utilisateur n'est pas au courant qu'un schéma de traitement des données est en cours.

Les systèmes K-embedded sont souvent des systèmes interactifs en ce sens qu'ils interagissent avec leur environnement. De ce fait, ils sont constamment prêts à accepter et à analyser tout nouvel intrant environnemental et à fournir les extrants nécessaires conformément au programme qui leur est donné. Ils sont constamment engagés dans des interactions continues avec leur environnement à une vitesse définie par cet environnement.

10.Contraintes :

Les systèmes embarqués exécutent des tâches prédéfinies et doivent respecter certaines normes, qui peuvent aller de :

En termes de prix. Le prix de revient doit être aussi bas que possible, surtout si de grandes quantités sont produites.

L'espace mémoire est minime, avec seulement quelques mégaoctets disponibles au maximum. Pour réduire les dépenses inutiles, les systèmes embarqués doivent être construits pour répondre aux demandes aussi étroitement que possible.

Puissance de traitement informatique. Il est essentiel d'avoir juste assez de puissance de traitement pour répondre aux exigences et aux contraintes de temps de la tâche. Cela permet d'éviter les dépenses supplémentaires et la consommation d'énergie excessive de l'appareil .

Consommation d'énergie la plus faible possible, grâce à l'utilisation de batteries, de panneaux solaires et même de piles à combustible dans certains prototypes.

Les périodes d'exécution et l'échéance temporelle d'un job sont définies par temporelle (les échéances sont connues ou bornées a priori).

Sécurité au travail. Car si certaines de ces technologies embarquées échouent, cela met la vie des gens en danger ou compromet des investissements importants. Ils sont alors qualifiés de

"critiques" et ne peuvent tomber en panne. Par « ne jamais échouer », nous entendons que nous devons toujours réaliser que nous devons fournir des résultats justes et pertinents dans les délais prévus par les consommateurs (machines et/ou personnes) des résultats.
un sentiment de sécurité Ces systèmes peuvent contenir des informations sensibles sur leurs utilisateurs, qui doivent être conservées et protégées. En particulier, en ce qui concerne l'acquisition et la transmission d'informations médicales... (par exemple, les systèmes personnels permettant la collecte et la transmission à distance d'informations sensibles,telles que desdonnées médicales, par le patient lui-même, ou relevant de la vie privée de l'utilisateur (s) en général).

11.Classification de systémes embarqués :

- **Transformation du système :**

qui commence par lire ses données et ses entrées, puis produit ses sorties avant de cesser d'exister.

- **Système d'interaction :**

Système en interaction quasi permanente avec son environnement , la réponse du système est déterminée par les événements qu'il reçoit et par son état actuel (qui est fonction d'événements et de réactions antérieurs) ; le système lui-même, pas l'environnement,

- **Systèmes en temps réel vs systèmes réactifs :**

Le système lui-même, et non l'environnement, décide comment réagir en fonction des événements qu'il reçoit et de son état actuel.

12.Types de systèmes embarqués :

Il existe quelques types fondamentaux de systèmes embarqués, et chacun a des besoins fonctionnels uniques. Et ils sont:

- Les systèmes à petite échelle qui sont portables sont appelés systèmes embarqués mobiles. Cela peut être vu dans les appareils photo numériques.
- Afin de fournir une sortie à d'autres systèmes, les systèmes embarqués en réseau sont connectés à un réseau. Les systèmes de point de vente (POS) et les systèmes de sécurité à domicile en sont quelques exemples.
- Les systèmes embarqués indépendants sont ceux qui ne reposent pas sur un système hôte. Comme tout système embarqué, ils remplissent une certaine fonction. Contrairement à d'autres systèmes embarqués, ils ne font pas toujours partie d'un système hôte. Une illustration de ceci serait une calculatrice ou un lecteur MP3.

- La sortie nécessaire est fournie via des systèmes embarqués en temps réel dans un laps de temps prédéterminé. Ils effectuent fréquemment des travaux urgents, ce qui les rend utiles dans les domaines médical, industriel et militaire. Une illustration de ceci serait un système de contrôle du trafic.
- Les systèmes embarqués peuvent également être divisés en groupes en fonction de leurs performances :
- Les appareils embarqués à petite échelle n'utilisent souvent qu'un microprocesseur 8 bits.
- Des microcontrôleurs plus grands (16 à 32 bits) sont utilisés dans les systèmes embarqués à moyenne échelle, et les microcontrôleurs sont fréquemment reliés entre eux.
- Les systèmes embarqués à échelle compliquée peuvent utiliser de nombreux algorithmes, ce qui ajoute à la complexité du matériel et des logiciels. Ces systèmes peuvent également nécessiter des logiciels plus complexes, un processeur personnalisable ou un réseau logique programmable.
- Il existe un certain nombre de conceptions typiques de logiciels de systèmes embarqués, qui sont essentielles à mesure que les systèmes embarqués se développent et deviennent plus grands et plus complexes. Celles-ci consistent en :
- Des boucles de contrôle simples invoquent des sous-programmes, qui contrôlent un élément particulier de logiciel ou de matériel embarqué.
- Il existe deux boucles dans les systèmes contrôlés par interruption : une boucle primaire et une boucle secondaire. Les tâches sont lancées via des interruptions de boucle.
- Une interface de programmation d'application possède une boucle de contrôle simple qui est à la base du multitâche coopératif (API).
- Un RTOS utilise fréquemment le multitâche préemptif ou le multithreading, qui comprend des techniques de synchronisation et de commutation de tâche.

13.Tendances futures des systèmes embarqués :

Les progrès continus de l'intelligence artificielle (IA), de la réalité virtuelle (VR), de la réalité augmentée (AR), de l'apprentissage automatique (ML), de l'apprentissage profond (DL) et de l'Internet des objets devraient propulser la croissance du secteur des systèmes embarqués. (IdO). La réduction de la consommation d'énergie, l'amélioration de la sécurité des appareils intégrés, la connexion au cloud et les réseaux maillés, les applications d'apprentissage en profondeur et les outils de visualisation de données en temps réel seront tous pilotés par des systèmes cognitifs intégrés.

Une analyse de 2018 de QYResearch estime que le marché mondial des systèmes embarqués valait 68,9 milliards de dollars en 2017 et atteindra 105,7 milliards de dollars d'ici la fin de 2025.

14.Domaines d’application des systèmes embarqués :

Les systèmes embarqués se trouvent fréquemment dans : les centrales nucléaires, les circuits de contrôle de la robotique, les laboratoires et les industries.
Des circuits pour contrôler la circulation et faire respecter les règles de circulation se trouvent dans les rues et les routes.
Plus précisément, dans les appareils ménagers, y compris les machines à laver, les micro-ondes et les appareils électroniques
soit des PDA ou des téléphones portables
appareils pour Internet mobile et fixe
tels que les contrôleurs de missiles, est utilisé dans la guerre.
Les satellites sont des moyens de communication contemporains.
tous les types de gadgets médicaux

15.Conclusion

Les systèmes embarqués, initialement créés dans les années 1970, font désormais partie intégrante de la vie contemporaine et commencent à être utilisés dans toutes les industries où la réduction des effectifs, la mobilité, la puissance de traitement et les algorithmes sont nécessaires.

Les systèmes embarqués se distinguent des systèmes conventionnels par le fait qu'ils sont coûteux à créer en grande quantité et exigent une fiabilité absolue.

Il convient de souligner que l'intérêt des industriels pour les systèmes embarqués OpenSource (basés par exemple sur Linux) ne cesse de croître du fait des importantes économies de coûts qu'offre cette solution pour le développement de produits et la R&D.

Chapitre 2 : Etude de Projet

1. Introduction

La vérification de modèle est une approche de vérification formelle utilisée en informatique pour modéliser et analyser des systèmes qui présentent un comportement probabiliste ou pour déterminer si le modèle à états finis d'un système est conforme à une spécification donnée. Lorsque les spécifications des systèmes matériels ou logiciels incluent des critères de vivacité en plus des exigences de sécurité, c'est souvent le cas (comme l'évitement des états représentant un plantage du système).

Le modèle du système et sa spécification sont tous deux exprimés dans un langage mathématique précis afin de résoudre un tel problème de manière algorithmique. Pour ce faire, le problème est présenté comme un problème logique, à savoir déterminer si une structure correspond à une formule logique particulière. Cette idée globale est applicable à une grande variété de structures et de raisonnements. Vérifier qu'une structure donnée remplit une formule dans la logique propositionnelle est une tâche simple de vérification de modèle.

Ce chapitre développe les concepts clés pertinents à cette thèse et donne un aperçu de l'examen du modèle probabiliste.

Ce chapitre donnera également une introduction aux techniques de vérification de modèle probabiliste et au modèle probabiliste HMM, qui seront utilisées dans cette étude pour introduire la vérification probabiliste avec un vérificateur de modèle.

2. Model checking probabiliste :

La conformité d'un modèle de système à ses spécifications peut être vérifiée automatiquement via un processus appelé model-checking [6]. Des automates, des réseaux de Petri, des algèbres de processus et d'autres techniques de modélisation formelle sont utilisés pour représenter formellement le comportement du système, et les expressions formelles des spécifications - qui définissent les caractéristiques attendues du système - incluent des formules de logique temporelle.

Il est possible de déterminer par la vérification du modèle si une structure M répond à une propriété qui est énoncée sous la forme d'une formule logique. L'expression M |= indique que M a satisfait. Si le modèle formel du "comportement" du système remplit les formules logiques

représentant les attributs souhaités, le système, par extension, satisfait ses exigences. Selon les types de logiques prises en compte, le comportement correspond soit au graphe des états disponibles, soit à l'ensemble des séquences issues de l'état de départ (logiques linéaires) (logiques arborescentes).

Le cycle d'utilisation du modelchecking est résumé dans la Figure 2.1, qui est adaptée de [8]. Trois phases peuvent être identifiées dans le cycle :

1. Modélisation formelle du comportement du système

2. Une déclaration formelle des qualités attendues

3. Un contre-exemple est fourni qui décrit une situation d'erreur potentielle si une propriété n'est pas satisfaite (c'est-à-dire la violation de la propriété).

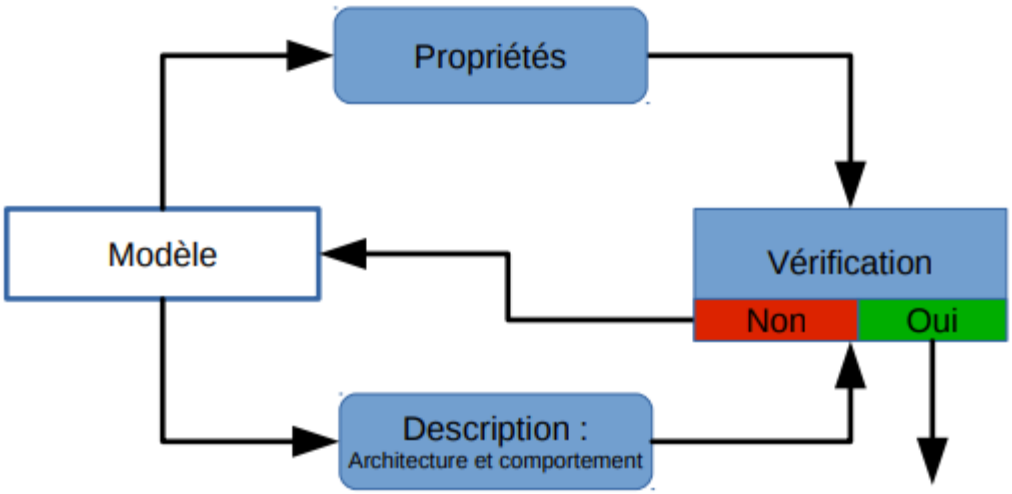


Figure 4: Cycle d'utilisation du model-checking

Le model-checking permet de décider si un modèle M vérifie une propriété φ, exprimée sous la forme d'une formule logique. La satisfaction de φ par M est notée M |= φ. Par extension, le système satisfait ses spécifications si le modèle formel de son comportement satisfait les formules logiques exprimant les propriétés attendues [8].

3.Chaîne de Markov en temps discret :

Une chaîne de Markov en temps discret (DTMC)[8,9], également connue sous le nom de processus stochastique, est une série de variables aléatoires où la valeur de chaque variable suivante ne dépend que de la précédente et ne prend en compte aucune variable précédente.

***Definition** : (DTMC) une chaine de markov à temps-discret est un tuple D = (S, I, P, L) avec:

- S un ensemble dénombrable d'états.
- I ∈ subDistr(S) une distribution initiale.
- P : S → subDistr(S) une matrice de probabilité de transition
- L : S → 2^{AP} une fonction d'étiquetage.

4.Processus de décision de Markov :

Le non-déterminisme ne peut pas être simulé à l'aide des DTMC. Le processus de décision de Markov est une sorte de DTMC qui permet une tarification décisionnelle probabiliste et non déterministe [9].

Markov Decision Process est le nom d'un modèle mathématique de tarification décisionnelle (MDP).

En plus d'avoir un caractère probabiliste, les actions sont choisies de manière non déterministe.

La définition d'un processus décisionnel de Markov est la suivante :

Définition 2.1.5 (Processus décisionnel de Markov). Un processus de décision de Markov est un tuple ⟨ S, Act, ι, π, Σ, L⟩

où:

• S est un ensemble fini et non vide d'états,

• Act est un ensemble fini d'étiquettes d'action,

• ι : S −→ [0, 1] est la distribution des états initiaux, telle que Σs∈ Sι(s) = 1

• π : S × Act × S −→ [0, 1] est la fonction de probabilité de transition, telle que ∀ s ∈ S et ∀ a ∈ Act, Σs ∈ Sπ(s, a, s') ∈ {0, 1} ,

• Σ = 2AP est l'alphabet état-étiquette représentant un ensemble de propositions atomiques,

• L : S −→ Σ est la fonction d'étiquetage d'état. Il attribue à chaque état un ensemble de propositions atomiques qui sont vraies à cet état.

5.Chaînes de Markov à temps continu :

Temps continu[8,9] Une variation en temps continu du processus de Markov est la chaîne de Markov. Le temps passé dans chacun des états est une variable aléatoire réelle positive qui suit une règle exponentielle dans ce modèle mathématique, qui a valeur dans une collection dénombrable appelée les états.

Cet objet est utilisé pour simuler l'évolution de certains systèmes, comme les files d'attente, dans le temps.

Définition : Soit *AP* un ensemble de propositions atomiques. Une *chaîne de Markov à temps continu* est un tuple $M = (S, s_0, R, L)$ où :

- S est un ensemble fini d'états,
- s_0 est l'état initial,
- $R : S \times S \to R_{\geq 0}$ est la matrice des taux de transition,
- $L : S \to 2^{AP}$ est une fonction d'étiquetage.

6.Modèles de Markov Cachés :

Les modèles de Markov cachés (HMM)[15,45] ont été introduits pour la première fois à la fin des années 1960 (Baum et Petrie 1966). Les modèles ont été mis en évidence par leur application à la reconnaissance vocale, comme dans Rabiner (1989). Depuis lors, cette classe de modèles a été utilisée dans de nombreux domaines d'étude, notamment la bioinformatique (Krogh, Mian et Haussler 1994), la finance (Hassan et Nath 2005) et la sismologie (Wang, Bebbington et Harte 2012 ; Wang et Bebbington 2013).Un modèle de Markov caché (HMM) est un modèle statistique commun qui est largement utilisé pour l'analyse des données de séquence biologique et d'autres phénomènes séquentiels.

Un modèle statistique populaire qui est fréquemment utilisé pour l'étude des données de séquence biologique et d'autres phénomènes séquentiels est le modèle de Markov caché (HMM). Dans le travail en cours, nous démontrons comment des contraintes secondaires peuvent être ajoutées aux HMM et proposons des méthodes de résolution de contraintes pour une inférence efficace. Il est avantageux de définir des HMM avec des restrictions latérales dans CLP car cela permet une expression plus efficace et un découpage d'inférence.

Nous proposons un cadre basé sur PRISM pour ajouter des contraintes latérales aux extensions HMM et démontrons comment des contraintes reconnaissables telles que la cardinalité et toutes différentes sont incluses. Sur le défi physiologique de l'alignement global par paires, nous vérifions expérimentalement notre méthode.

- **Définition :**

Définition : Une variante d'une machine à états finis ayant un ensemble d'états, Q, un alphabet de sortie, O, des probabilités de transition, A, des probabilités de sortie, B, et des probabilités d'état initial, Π. L'état actuel n'est pas observable. Au lieu de cela, chaque état produit une sortie avec une certaine probabilité (B). Habituellement, les états, Q, et les sorties, O, sont compris, donc un HMM est dit être un triplet, (A, B, Π).

- **Définition formelle :**

$A = \{aij = P(qj \text{ à } t+1 \mid qi \text{ à } t)\}$, où $P(a \mid b)$ est la probabilité conditionnelle d'un b donné, $t \geq 1$ est le temps et $qi \in Q$.

De manière informelle, A est la probabilité que le prochain état soit qj étant donné que l'état actuel est qi.

$B = \{bik = P(ok \mid qi)\}$, où $ok \in O$.

De manière informelle, B est la probabilité que la sortie soit ok étant donné que l'état actuel est qi.

$\Pi = \{pi = P(qi \text{ à } t=1)\}$.

Croyances : L'observation est stochastiquement et exclusivement dépendante de l'état actuel ; en général, une même observation pourra être émise par plusieurs états différents ; par conséquent, nous ne sommes pas sûrs de l'état actuel, mais nous pouvons collecter l'historique des observations dans un état de croyance (ou état d'information), qui est une distribution sur s.

Un état de croyance est un moyen de représenter ce que nous savons de l'état compte tenu de l'historique d'observation, plutôt qu'un état HMM. L'espace de croyance, indiqué par C, est la collection de tous les états de croyance potentiels.

Pour représenter l'état au temps t, nous utilisons st avec st S, Ot pour représenter l'observation au temps t, et ct pour représenter l'état de croyance au temps t. Soit Oi un nombre compris entre 1 et t, avec I = 1, 2,..., t. L'état de croyance ct au temps t est la distribution de S au temps t, qui donne l'historique des observations o0, o1,..., ot : ct(s) = p(st =s| O0=o0, O1= o1,...,Ot = ot,H), s S [4].

Maintenant que nous avons l'historique des observations o0,o1,...,ot, la question est de savoir comment calculer l'état de croyance cn : l'état de croyance au temps 0 ne dépend que de la

distribution initiale et de la première observation, alors que l'état de croyance à l'instant t prend toutes nos informations sur le passé, nous pouvons donc calculer l'état de croyance actuel ct basé sur l'état de croyance précédent ct-1 et l'observation actuelle ot, comme le montre la figure

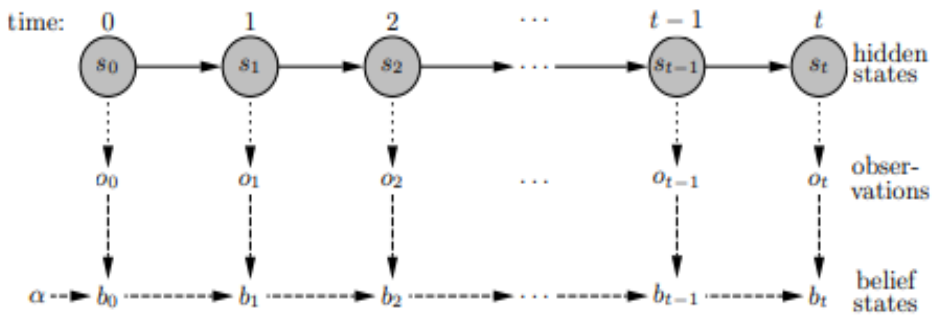


Figure 5 : Mise à jour des états de croyance

- Les flèches pointillées entre les états et les observations montrent que l'observation Ot est produite par l'état st accordé à la fonction d'observation, car un cas particulier C0 est une fonction de O0 et de la distribution initiale.

- • Les états reliés par des flèches continues entre eux représentent les états cachés d'une exécution.

6.1.Les trois problèmes des HMMs :

La réalisation par le modèle HMM H d'un processus Markovien de durée T est décrite par :

- Q = q 1 q 2 … q T , un chemin a priori caché parmi les N états,

- O = O 1 O 2 … O T , une suite d'observations appartenant à l'alphabet de M symboles.

Pour utiliser le modèle H, trois problèmes fondamentaux doivent être résolus à partir d'une séquence d'observations O prétendument émises par un modèle :

- ***Le problème d'évaluation :*** La probabilité que les observations continuent d'être diffusées par un modèle est évaluée. Lorsque de nombreux modèles sont disponibles, cette évaluation permet de sélectionner le modèle le plus probable. L'algorithme Forward [11] résout avec succès ce problème.

- ***Le problème de décodage ou de reconnaissance :***La recherche de la séquence cachée avec la plus grande probabilité et qui explique le mieux la série de données produite par un modèle, c'est-à-dire la recherche de la séquence cachée avec la plus grande

probabilité et qui explique le mieux ces observations. L'algorithme de Viterbi [] est utilisé pour résoudre ce problème rapidement.

- ***Le problème de re-estimation ou d'apprentissage :***

  • Connaître les paramètres d'un modèle. Nous recherchons les probabilités de transition et d'émission qui maximisent à la fois la probabilité des observations produites par un certain modèle a priori et la vraisemblance des observations que ce modèle est susceptible d'émettre.

6.2.Principes de fonctionnement des HMM :

L'algorithme est basé sur le principe de l'optimisation EM (Expectation Maximization) et alterne des phases d'estimation des probabilités t(i), t(i) et t(i), suivies d'une phase d'optimisation des paramètres du modèle[w8], qui consiste à les éléments suivants : l'ensemble A (matrice des transitions) de aij, le B de bi(O) (ensemble de fonctions de densité) et t(i) (vecteur de probabilités initiales),[11 ,15]

- **Quelques valeurs :**

– $\lambda = (A, B, \Pi)$ est le triplet caractérisant un HMM.

– A est la matrice des probabilités aij de transiter de l'état i à l'état j.

– B est l'ensemble des fonctions de densité de probabilité bi(o) d'émission d'une observation o par l'état i.

– Π est la distribution des états initiaux $\pi i = P(qt = Si \mid t = 0, \lambda)$.

$\alpha kt(i) = P(ok1, ok2, ..., okt, qkt = Si \mid \lambda)$ est la probabilité de réaliser une série d'observations données jusqu'à l'instant t et d'être dans l'état i à l'instant t connaissant la modèle λ.

- **Example :**

Two people, let’s call them Isla and Donnie, talk about food they like to eat. Donnie likes to eat pizza, pasta and pie. He tends to choose which to eat depending on his emotions. Isla has a rough understanding of the likelihood that Donnie is happy or upset and his tendency to pick food based on those emotions. Donnie’s food choice is the Markov process and Isla knows the parameters but she does not know the state of Donnie’s emotions; this is a hidden Markov model. When they talk, Isla can determine the probability of Donnie being either happy or upset based on which of the three foods he chose to eat at a given moment.

➢ **Applications of Hidden Markov Model :**

- Computational finance
- Cryptanalysis
- Speech recognition – Notably Apple’s Siri
- Handwriting recognition
- Time series analysis

7.Formalismes de spécification :

Les vérificateurs de modèles probabilistes les plus utilisés tirent parti des logiques qui permettent de créer et de vérifier des systèmes probabilistes. PCTL, CSL, PRCTL et CSRL font partie des systèmes logiques concernés. Dans le chapitre 3 de ce livre, nous examinerons la logique que chaque vérificateur de modèle devrait utiliser. [34]

7.1La logique PCTL :

délivrée par H. Hansson et B. Jonsson [12] est une modification probabiliste de CTL (Computational Tree Logic) [19], où le quantificateur (V) "Pour tous les chemins" et "Existe un chemin" a été remplacé par un opérateur probabiliste qui nécessite une probabilité donnée sur un certain ensemble de chemins qui est l'opérateur probabiliste P.P ; ou P E[0.1] est une liaison de probabilité, et ~e{>,<,52}.

7.1.1La syntaxe de PCTL :

H. Hanson et B. Johnson [12] ont introduit la logique PCTL, qui distingue les formules d'état et de chemin. Les propriétés et comportements attribués aux états sont exprimés dans des formules d'états, tandis que les propriétés et comportements attribués aux chemins sont exprimés dans des formules de chemins.(_çà)

- **Definition** :

La syntaxe de la logique PCTL est la suivante :

Formules d’´etats : $\Phi ::= tt \mid a \mid \neg\Phi \mid \Phi 1 \wedge \Phi 2 \mid P{\sim}p(\phi)$

Formules de chemins : $\phi ::= X\Phi \mid \Phi 1 \ U \ \Phi 2$

ou $a \in AP$, $p \in [0, 1]$ et $\sim \in \{\leq, \geq, <, >\}$.

PCTL est l’ensemble des formules d’etats construites de la facon suivante :

1. Formules d’etats

1- a est une formule d’etat.

2- Si Φ est une formule d'etat, alors ¬Φ est une formule d'etat

3- Si Φ1 et Φ2 sont des formules d'´etat, alors Φ1 ∧ Φ2 est une formule d'´etat..

4- Si ϕ est une formule de chemin, alors P∼p(ϕ) est une formule d'´etat. .

5- Rien d'autre n'est une formule d'´etat.

2. Formules de chemins :

1- Si Φ est une formule d'etat, alors X Φ est une formule de chemin.

2- Si Φ1 et Φ2 sont des formules d'etats, alors Φ1UΦ2 est une formule de chemin.

3- Rien d'autre n'est une formule de chemin.

7.1.2.La sémantique de PCTL

Soit M = (S, P, L) un modèle PCTL, s étant un état de M, AP désignant un ensemble de propositions atomiques et formule PCTL désignant une formule PCTL. « est remplie à l'état s dans le modèle M », dit la relation de satisfaction indiquée par M, s |=. La relation de satisfaction est également représentée par le symbole |=, qui utilise des formules de chemin.

2.3.1 **Définition** :Voici une définition de la sémantique de la logique PCTL :

Sémantique pour les formules d'états

$s \models \mathrm{T}$ *pour tout s*

$s \models a$ *ssi* $a \in \mathrm{L}(s)$

$s \models \neg \phi$ *ssi* $s \models \phi$

$s \models (\phi_1 \wedge \phi_2)$ *ssi* $s \models \phi_1 \wedge s \models \phi_2$

$s \models P_{\sim p}(\psi)$ *ssi* $Prob(\{\pi \in \mathrm{Paths(s)} \mid \pi \models \psi\}) \sim p$

Sémantique pour les formules de chemins

$\pi \models \mathrm{X}\phi$ *ssi* $\pi = s_0 s_1 s_2 \ldots$ et $s_1 \models \phi$

$\pi \models \phi_1 U \phi_2$ *ssi* $\pi = s_0 s_1 s_2 \ldots$ et $\exists k$ *tq.* $S_k \models \phi_2 \wedge \forall\, j < k, s_j \models \phi_1$

$\pi \models \phi_1 U^{\leq k} \phi_2$ *ssi* $\exists\, i \leq k$ tq. $s_i \models \phi_2 \wedge s_j \models \phi_1$, $\forall\, j < i$

Figure 6: La sémantique de PCTL

Les opérateurs booléens T et sont interprétés comme prévu. Par conséquent, si la formule 1 2 remplit à la fois la formule 1 et la formule 2, elle est valide à cet état. Si la probabilité que les voies maximales définies aient comme état de départ s0 et dont l'état suivant s1 satisfait est supérieure ou égale à p, la formule Pp(X) est satisfaite par le modèle M.

7.2.La logique POCTL :

POCTL est essentiellement une extension PCTL avec une restriction d'observation ajoutée à l'opérateur suivant. POCTL, d'autre part, peut être considéré comme une version de la logique temporelle ACTL (Action Computation Tree Logic) de De Nicola et d'autres, dans laquelle l'opérateur suivant standard est étendu pour confiner l'étiquette d'action de la transition. Les formules d'état, les formules de chemin et les formules d'état de croyance font toutes partie de POCTL, ce qui nous permet de décrire les attributs sur les HMM. Nous pouvons déclarer des qualités sur des observations de cette manière, par exemple, Xo dénote que la prochaine observation est o et que la route suivante satisfait [12].

7.2.1.La syntaxe de POCTL

H= (S, P, L, B,) est un HMM avec o O. La syntaxe logique pour POCTL est la suivante :

$$\phi := a \mid \phi_1 \wedge \phi_2 \mid \neg \phi \mid \varepsilon$$

$$\psi := X_o \phi \mid \phi_1 U^{\leq k} \phi_2$$

$$\varepsilon := P_{\sim p}(\psi) \mid \neg \varepsilon \mid \varepsilon_1 \wedge \varepsilon_2$$

Où : $k \in N$, $0 \leq p \leq 1$ et $\sim \in \{>, \geq, <, \leq\}$

Figure 7: La syntaxe de la logique POCTL

Les formules d'état, les formules de chemin et les formules d'état de croyance constituent la syntaxe POCTL. Pour la formule d'état et la formule de chemin, nous utilisons, et la formule est

connue sous le nom de formule d'état de croyance. pDans le cas d'un HMM, nous ne connaissons pas l'état actuel, mais nous connaissons l'état de croyance actuel. En conséquence, nous aimerions savoir si certaines caractéristiques (probabilistes) sont vraies dans les états de croyance. "Il y a au moins 90 % de chances que le modèle produise l'ordre des observations O = o0, o1,...,on", indique la propriété. Dans POCTL, cet attribut peut être représenté comme suit : Selon Rabiner dans [11], P0.9 (Xo0 Xo1... Xon) Une formule d'état de croyance peut s'écrire : = P0.9 (Xo0 Xo1.... Xon) [31].

7.2.2.La sémantique de POCTL :

H= (S, P, L, B,) est un HMM avec s S et Path. Une connexion de satisfaction (notée |=) entre un état s et une formule d'état, ou entre un chemin et une formule de chemin, ou entre un état de croyance b et une formule d'état de croyance définit la sémantique de POCTL (figure). Si l'état s, le chemin et l'état de croyance b satisfont respectivement la formule d'état, la formule de chemin et la formule d'état de croyance, nous écrivons H, s |=; H, |= et H, b |=.

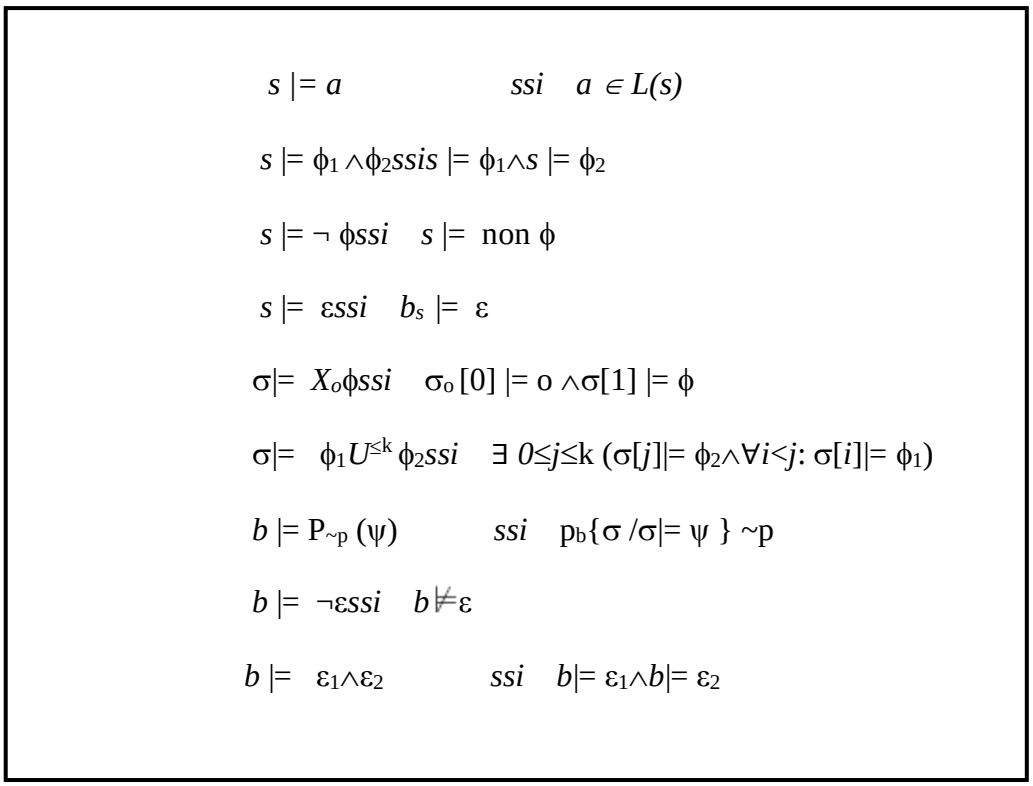


Figure 8: La sémantique de POCTL

Si une route commence par l'observation o et que le suffixe [12] satisfait, elle satisfait le nouvel opérateur X. Soit une collection d'observations, c'est-à-dire
Il est clair que nous avons si o[0] [1] |= X.
Le prochain opérateur commun est désigné par le symbole X.
Par conséquent, la logique PCTL peut être considérée comme la logique secondaire de POCTL [12].

8.Algorithmes de model-checking probabiliste :

Nous avons sélectionné des chaînes de Markov à temps discret comme modèle de système probabiliste et logique PCTL pour la description des propriétés nécessaires dans les sections précédentes. nous verrons comment déterminer quels états remplissent une certaine formule PCTL.[27,32].

Les entrées d'une méthode de vérification de modèle pour la logique PCTL ou CSL sont une logique de formule, un M probabiliste adapté au modèle (DTMC ou MDP pour PCTL et CTMC pour CSL) et l'ensemble Sat() contenant l'ensemble des états qui se rencontrent. L'algorithme de vérification de modèle CTL décrit dans 9] est la source de la structure générale de la méthode de vérification de modèle PCTL pour DTMC (ainsi que PCTL pour MDP et CSL pour CTMC). Construisez d'abord l'arbre syntaxique de la formule, selon la méthode.

Trouver l'ensemble Sat() = s S|s |= pour chaque sous-formule de en remontant récursivement l'arbre syntaxique de bas en haut (des feuilles à la racine).

Chaque nœud de l'arbre syntaxique d'une formule représente une sous-formule de cette formule, et la racine de l'arbre est la formule elle-même. Les propositions atomiques ou, dans le cas de PCTL, l'opération booléenne true constituent les feuilles de l'arbre. Avant de détailler en détail la mise en œuvre de la méthode, nous montrons son fonctionnement sur une formule de base (sans l'opérateur probabiliste).

9.Conclusion :

L'avantage de la vérification de modèle primaire est qu'elle est totalement automatisée et facile à utiliser. Il existe des méthodes qui implémentent des vérificateurs de modèle qui peuvent répondre à la question "Le modèle du système répond-il à la propriété ?" sans nécessiter d'intervention humaine si le modèle de système est exploitable en taille. De plus, la majorité des vérificateurs de modèle offrent un chemin d'exécution qui viole les attributs lorsque la réponse est non, ce qui simplifie la résolution du problème. L'inconvénient majeur de l'utilisation du model-checking est l'explosion dite combinatoire du nombre d'états, qui s'ajoute au fait que cette technique ne peut être utilisée que pour manipuler des modèles avec un nombre fini d'états.

La modélisation de systèmes réels (qui peut être compliquée) aboutit généralement à la génération d'un si grand nombre d'états qu'il est difficile de tous les conserver dans la mémoire d'un ordinateur, empêchant une exploration complète de l'espace d'états dans la pratique. Pour rendre praticable le model-checking de grands systèmes, des techniques de résistance aux phénomènes d'explosion combinatoire doivent être mises en œuvre.

La popularité des HMM est à la hausse. Cela est dû aux nombreux avantages qu'ils offrent, notamment la capacité d'intégrer correctement plusieurs sources de connaissances grâce à l'utilisation d'une théorie telle que la théorie des probabilités, la disponibilité d'algorithmes pour l'apprentissage des modèles et la facilité de représentation des régularités statistiques de l'application à en traitement.

Une sorte de logique appelée logiques temporelles peut être appliquée pour expliquer le changement de comportement. Pnueli a introduit son application en informatique car il pensait que les méthodes de vérification formelles conventionnelles n'étaient pas suffisantes pour les systèmes embarqués. Notre approche a tiré parti de la logique temporelle POCTL. Pour commencer, POCTL est simplement une extension de PCTL avec une contrainte d'observation ajoutée à l'opérateur après. POCTL, d'autre part, peut être comparé à la logique temporelle ACTL et utilise une extension de l'opérateur suivant pour limiter l'étiquette d'action de la transition.

Dans le chapitre suivant, nous examinerons de plus près l'un des outils les plus connus pour vérifier les modèles probabilistes.

Chapitre 3 : Réalisation

1. Introduction :

(PRISM) est le Probabilistic Model Checker [w10]. Un outil logiciel de vérification formelle pour la modélisation et l'analyse de systèmes à comportement probabiliste.[28] L'utilisation de la randomisation, par exemple dans des protocoles de communication comme Bluetooth et FireWire ou dans des protocoles de sécurité comme le routage Crowds et Onion, est l'une des sources de tels systèmes. De nombreux autres systèmes informatiques connaissent également un comportement stochastique, par exemple en raison de communications retardées de manière imprévisible ou de problèmes d'équipement. Les réseaux de réactions biochimiques sont une autre classe de systèmes accessibles à ce type d'investigation.
L'utilitaire lui-même prend deux fichiers. Le premier est un fichier modèle, qui est une description du système à examiner écrite en langage PRISM. Le deuxième fichier est un fichier de spécifications, qui comprend un ensemble de caractéristiques à comparer au système spécifié dans le fichier modèle. Il est écrit dans une logique temporelle appropriée.
Les systèmes peuvent être modélisés dans PRISM comme :
• Chaînes de Markov à temps discret (DTMC) ;
• processus décisionnels de Markov (MDP) ;
• Chaînes de Markov en temps continu (CTMC).
Les propriétés peuvent être spécifiées à l'aide des logiques temporelles suivantes :
• logique arborescente de calcul probabiliste (PCTL) pour les DTMC et les MDP ;
• Logique stochastique continue (CSL) pour les CTMC.
Avec l'utilisation de l'analyse numérique, pRISM crée une représentation basée sur la mémoire du modèle et vérifie les qualités par rapport à celle-ci. Les nombreux moteurs de vérification de modèles de PRISM effectuent cet examen.
Le système produit un ensemble de résultats qui indiquent soit si une propriété a été satisfaite1, soit la probabilité qu'elle soit satisfaite.
L'Université de Birmingham et l'Université d'Oxford sont celles où se fait la majorité du développement de PRISM. L'instrument est un logiciel libre distribué selon les termes de la licence publique générale GNU. PRISM a été sélectionné pour les programmes Google Summer of Code 2013 et 2014.

2. Historique :

- La création de la première théorie sous-jacente a eu lieu au début des années 1990.
- Recherche et développement de l'Université de Birmingham/Oxford depuis 1999.
- PRISM a été mis à la disposition du grand public en 2001.
- Systèmes probabilistes en temps réel, v4.0, 2011.
- Les jeux PRISM, un genre de jeux multijoueurs stochastiques, sont apparus en 2013.

3. Thèmes et tendances :

Les thèmes suivants ont été abordés tout au long de la création de PRISM :

1. Logiciels open source.
2. La capacité d'adapter des modèles plus grands a augmenté.
3. Plusieurs domaines différents avec lesquels il interagit ou se connecte.
4. Large collaboration (théorie, algorithmes, études de cas).
5. Modélisez des parcours avec plus de force et d'expression.
6. Problèmes de contrôle et de vérification.
7. Algorithmes distribués aléatoirement, réseaux/protocoles de communication, sécurité informatique, performance/fiabilité, biologie des systèmes, calcul ADN, robotique et véhicules autonomes, appareils portables/implantables, et bien d'autres domaines d'application.

4.Type de modèle :

Une variété de types de modèles probabilistes peuvent être décrits à l'aide du langage PRISM. Un mot-clé de type de modèle est généralement inclus dans les modèles PRISM :

- dtmc : chaîne de Markov à temps discret.
- ctmc : chaîne de Markov en temps continu.
- mdp : processus de décision de Markov (ou automate probabiliste).
- pta : automate temporisé probabiliste.

5.Language de Prism :

Un "langage de description simple basé sur des modules", ou "langage PRISM", est utilisé pour créer des descriptions de système qui sont entrées dans l'outil PRISM. La définition d'une description de système est "la composition parallèle de nombreux modules en interaction"[w10], qui peut être écrite comme un tuple (M, G, R) où : • M est une collection de N modules, où chaque module, m M, peut être considéré comme un processus qui s'exécute simultanément avec d'autres modules du modèle. Une collection non vide de variables entières ou booléennes, Vm,

est présente dans chaque module. Un ensemble de variables globales entières ou booléennes, G, est également présent.

L'état global, souvent appelé état actuel, est la configuration de Vm telle qu'elle est maintenant configurée, tandis que l'état local fait référence à la configuration actuelle de Vm. L'espace d'états du modèle est défini comme l'ensemble, S, de tous les états globaux atteignables.

De plus, il convient de mentionner que chaque module et variable a un nom unique2.

Le langage PRISM permet à chaque module d'avoir un ensemble, Commm de commandes pour représenter les changements dans l'état global du modèle. Chaque commande, c Commm, spécifie comment l'état local et l'ensemble de variables globales peuvent être modifiés pour un certain prédicat sur l'état global (appelé garde, Guardc).

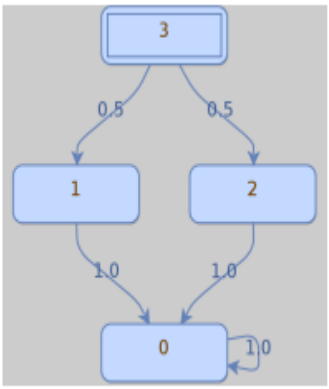
Un ou plusieurs modules peuvent être programmés pour effectuer une transition pour une certaine transition globale.

Une transition synchrone se produit lorsque deux ou plusieurs modules effectuent leurs transitions simultanément, par opposition à une transition asynchrone, qui se produit lorsqu'un module effectue sa transition individuellement. En étiquetant deux ou plusieurs instructions, provenant de différents modules, avec l'étiquette d'action a Act, où Act désigne l'ensemble de toutes les actions utilisées dans le modèle, il est possible de spécifier des transitions synchrones. Il est indiqué que les commandes indépendantes sont celles qui n'ont pas de noms d'action.

les noms de tous les autres modules, variables locales, variables globales, constantes et étiquettes d'action dans la description du système.Les modules et les variables constituent le cœur du langage PRISM. De nombreuses variables locales sont présentes dans un module. L'état du module est déterminé par les valeurs de ces variables à un instant donné. L'état local de chaque module affecte l'état global du modèle global. Le comportement de chaque module est décrit par un ensemble de commandes.
Une commandes exprime ainsi :
Probabilité1 :update1 + + probabilityn : updaten pour le garde
La garde est un prédicat sur toutes les variables du modèle (y compris celles appartenant à d'autres modules). Si la garde est vraie, chaque mise à jour explique une transition que le module peut effectuer. Une transition est définie en fournissant les valeurs mises à jour des variables du module, qui peuvent également être des fonctions d'autres variables. De plus, une probabilité est donnée à chaque mise à jour qui sera donnée à la transition qui l'accompagne.
Exemple La méthode illustrée par le code de la figure 3.1a sélectionne la variable a ou b et l'annule avec la même probabilité. L'autre variable est annulée à l'étape suivante :

```
dtmc

module main

a : bool init true;
b : bool init true;

[] a -> 1.0: (a'=false);
[] b -> 1.0: (b'=false);

endmodule
```

(a) Simple code



(b) Graph of code in Figure 3.1a

figure 9:exemple de dtmc

6.L'interface utilisateur graphique (GUI) du PRISM:

L'interface utilisateur graphique (GUI) du PRISM

L'interface de ligne de commande PRISM offre un accès à toutes ses fonctionnalités. Pour effectuer un grand nombre d'exécutions de tests de modèles ou pour intégrer PRISM à d'autres outils via des scripts, la ligne de commande PRISM est préférable [28].

Le PRISM graphique fournit :

• un éditeur de modèles pour le langage de modélisation PRISM avec coloration syntaxique et rapport d'erreurs,

• un éditeur de propriétés,

• un outil de simulation pour explorer et déboguer les modèles PRISM

• une simple fenêtre de sortie de journal,

• outils de tracé de graphes.

Il y a un menu en haut de la fenêtre principale de PRISM. Les aliments sont séparés en fonction des fonctionnalités PRISM proposées. Modèle, Propriétés et Simulateur sont les trois options de menu les plus importantes. Il est possible d'ouvrir, de recharger ou d'enregistrer un modèle depuis ou vers un fichier existant à l'aide du menu Modèle, qui possède des fonctionnalités de base pour travailler avec des modèles textuels. De plus, il permet au modèle d'être construit, exporté ou analysé. Le travail avec les propriétés est possible via le menu Propriétés . Enfin, l'option Simulateur permet de simuler le comportement nécessaire du système.

Il y a un deuxième menu situé sous le premier. Ce menu comprend des icônes d'éditeur de texte conventionnelles et facilite la gestion des composants de texte de l'interface graphique (par exemple, annuler, rétablir, copier, coller, etc.).

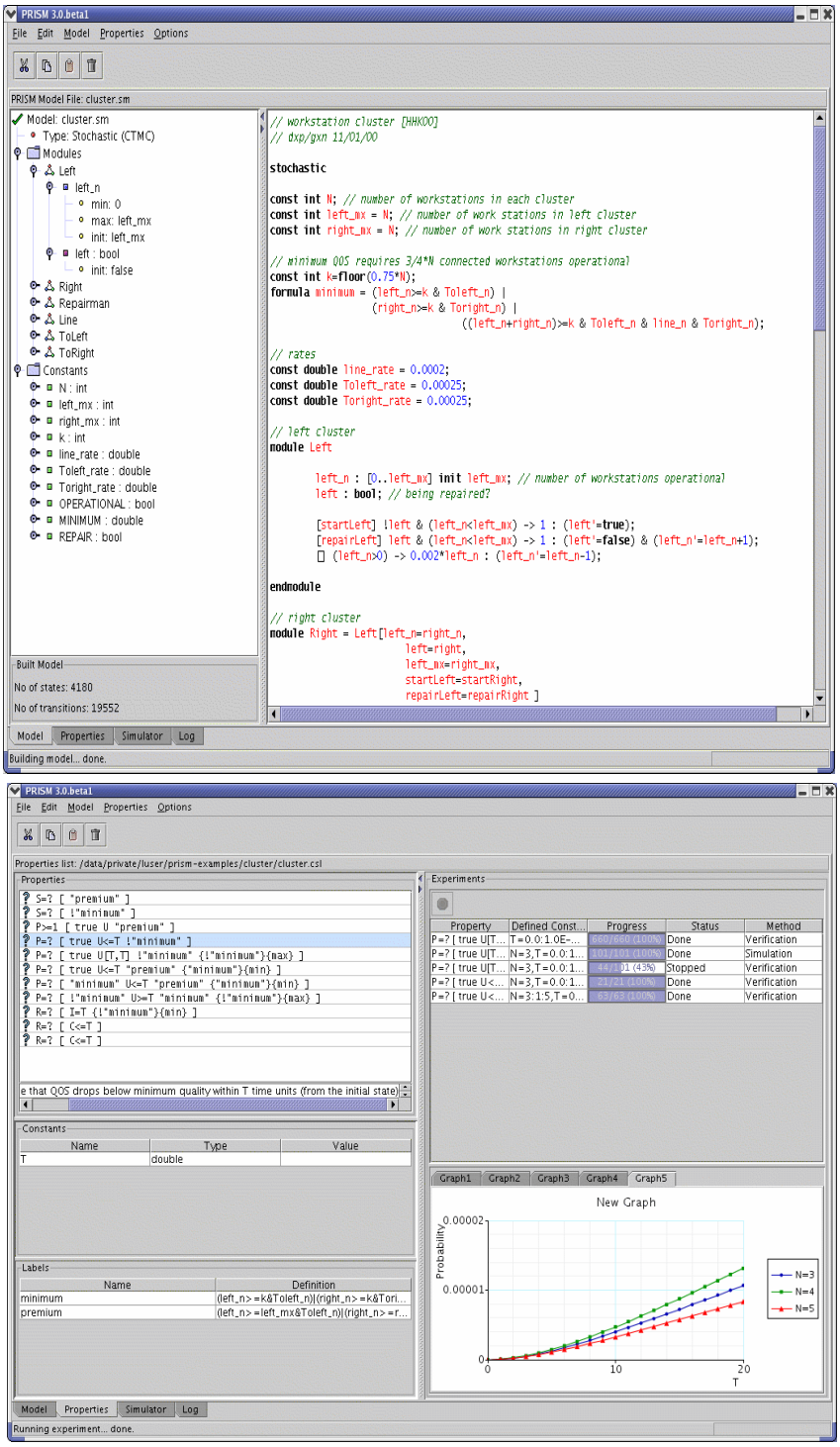


figure 10:L'interface utilisateur graphique du PRISM.

7.MODÈLES PROBABILISTES EN PRISME :

Ai représente la collection de toutes les actions qu'un module mi a utilisées. L'un des n modules effectue une transition indépendante dans chaque état global du modèle construit, ou pour certains a Act, l'ensemble des modules mi | a Ai effectuent tous des transitions synchrones, étiquetées a. [43]

Le composant de récompenses, R, spécifie comment les récompenses doivent être accumulées pour des états de modèle spécifiques et pour des transitions d'état spécifiques. Il est connu sous le nom de tuple (SR, T R).

SR est une collection de biens de récompense d'État, où. Un prédicat, Guard, et une expression, Value, qui évaluent un nombre réel sont liés à chaque élément de récompense d'état, ou SR. La valeur est évaluée et s'accumule en tant que valeur de récompense pour un état global où le modèle rencontre le prédicat Guard.

Un groupe d'éléments de récompense de transition est connu sous le nom de TR. Chaque élément de récompense de transition, désigné par les lettres T R, est connecté à un prédicat Guard, une expression appelée Value qui s'évalue à un montant réel et une étiquette d'action appelée Act. Lorsqu'une transition globale est effectuée et que le prédicat, Guard, est évalué à vrai, les éléments de récompense de transition commencent à accumuler des récompenses.

L'élément de récompense ne peut pas inclure d'étiquette d'action si la transition est asynchrone afin que la récompense soit accumulée. Si la transition est synchrone, la récompense ne sera accumulée que si an est égal à l'étiquette d'action de la transition, ac. La valeur du prix sera le résultat de l'évaluation de Valueτ.

Chaque type de modèle PRISM a une manière particulière d'interpréter les commandes. Les trois sections qui suivent examinent chaque type de modèle individuellement et expliquent comment les commandes peuvent être utilisées pour représenter les changements entre les états globaux.

8.Propriétés :

Afin d'étudier un modèle probabiliste qui a été créé et identifié dans PRISM, il est crucial d'identifier une ou plusieurs caractéristiques du modèle que l'outil peut évaluer. Le langage de spécification de propriétés PRISM prend en charge de nombreuses logiques spatio-temporelles probabilistes populaires, notamment PCTL, CSL, LTL probabiliste et PCTL.

9.Syntaxe :

Différentes logiques temporelles probabilistes, telles que PCTL, CSL, LTL (probabiliste), PCTL* et CTL, sont incluses dans la syntaxe du langage de spécification de propriété PRISM. Une propriété peut être n'importe quelle expression PRISM valide et correctement typée, qui (éventuellement) comprend les opérateurs probabilistes P, S et R qui ont été précédemment adressés, ainsi que les opérateurs non probabilistes (CTL) A et E. opérateurs peuvent donc être utilisés :

- `-` (unary minus)

- `*`, `/` (multiplication, division)
- `+`, `-` (addition, subtraction)
- <, <=, >=, >(relational operators)
- =, `!=` (equality operators)
- `!` (negation)
- `&`(conjunction)
- `|` (disjunction)
- <=>(if-and-only-if)
- =>(implication)
- `?` (condition evaluation: `condition ? a : b` means "if `condition` is true then `a` else `b`")
- P (probabilistic operator)
- S (steady-state operator)
- R (reward operator)
- A (for-all operator)
- E (there-exists operator)

10.RESSOURCES ET INFORMATIONS :

Obtention du PRISME. La licence publique générale GNU régit la distribution de l'outil, qui est un projet open source appelé PRISM (GPL). Sur le site Web, vous pouvez obtenir des versions de code binaire et source [28]. L'utilitaire est compatible avec tous les systèmes d'exploitation populaires, y compris Windows, Linux, Unix et Macintosh. Les versions "de développement" de PRISM, en plus des versions "publiques" régulières, sont rendues accessibles via

site internet Ils donnent accès aux mises à jour les plus récentes et en cours de l'outil.

Analyses de cas. La performance et la fiabilité des études de cas provenant de diverses sources ont été analysées à l'aide de PRISM. Cela couvre les systèmes de fabrication, les systèmes de contrôle embarqués, les systèmes de file d'attente, l'électronique à l'échelle nanométrique, les systèmes de gestion dynamique de l'alimentation et les réseaux informatiques. De plus, il a été utilisé pour analyser une variété de systèmes, y compris des processus biologiques, des mécanismes de sécurité aléatoires et des protocoles de communication et multimédias.

Un référentiel contenant des informations complètes sur plus de 45 études de cas est hébergé sur le site Web de l'outil. Il s'agit de contributions de sources internes et externes, y compris l'équipe PRISM. Des contributions supplémentaires sont toujours les bienvenues.

Le site Web contient également de nombreux liens vers des articles traitant de

d'autres études de cas, ainsi qu'une bibliographie complète

travaux scientifiques sur le sujet.

Donnée supplémentaire. Pour toute personne curieuse d'en savoir plus sur PRISM et les méthodes sur lesquelles il est fondé, le site Web [w10] offre une variété d'informations supplémentaires.

Il contient également du matériel technique supplémentaire, un manuel en ligne, une série de conférences en 11 parties et des publications associées.

De plus, un forum de discussion est disponible pour aider les utilisateurs de l'outil.

11. Conclusion :

PRISM est un vérificateur de modèle probabiliste. La version la plus récente (4.2.1) prend en charge la modélisation formelle et l'analyse des systèmes qui se comportent de manière aléatoire ou probabiliste . De nombreuses études de cas dans de nombreux secteurs d'application sont analysées à l'aide de PRISM.

L'interface graphique de PRISM est simple. Il comporte deux menus et quatre sections principales, chacune correspondant à un type différent de capacité offerte par PRISM.

L'apparence de l'interface graphique correspond à l'aspect et à la convivialité des composants Swing. L'inconvénient majeur est l'impossibilité de personnaliser entièrement l'interface utilisateur (par exemple, la disposition des composants de PRISM ne peut pas être modifiée) ; néanmoins, l'interface graphique de PRISM a une longue histoire et est créée et mise en œuvre par des experts qui travaillent avec PRISM depuis longtemps.

PRISM est gratuit et open source, "https://www.prismmodelchecker.org/".

Chapitre 4 : Évaluation et Discussion

1. Introduction

De nombreuses technologies ont vu le jour ces dernières années, permettant la réalisation d'innombrables études de cas. Notre objectif est d'évaluer un système authentique à l'aide d'une étude de cas de PRISM, l'instrument certifié le plus connu et le plus largement utilisé dans le domaine de la vérification.

L'aspect « modélisation » d'une étude de cas est difficile car il n'y a pas de directives universelles et tout est fait au cas par cas. Un cahier des charges décrivant le système réel étant fourni, la première étape consiste à effectuer une première modélisation assez grossière du système : cela permet de déterminer quelle classe de modèles paraît bien adaptée à la représentation du système, et cela permet aussi la possibilité de sélectionner un outil de modélisation à l'avance.

La deuxième phase consiste à créer un modèle complet du système et éventuellement à l'adapter aux décisions récentes.

2. Système de prévision des accidents :

Selon l'Organisation mondiale de la santé (OMS), les accidents de la route tuent 1,24 million de personnes chaque année et en blessent des dizaines de millions d'autres. [30]
En cas d'accident imminent, le système de prévision des accidents (APS) est un système de sécurité du véhicule qui peut soit alerter le conducteur, soit réagir de manière autonome en tournant ou en freinant sans l'aide du conducteur. Il existe deux formes d'APS, l'une basée sur le matériel et l'autre sur le logiciel. Le premier est réalisé en installant une gamme de capteurs et de caméras dans les voitures et en prédisant une collision probable sur la base des données acquises par le capteur, tandis que le second est basé sur des prédictions logicielles.

Plusieurs APS ont été proposés pour VANET. La majorité de celle-ci est déterminée uniquement par la vitesse du véhicule. Les conditions météorologiques, la fatigue du conducteur et d'autres éléments doivent cependant être pris en compte dans les scénarios d'effondrement. La reconnaissance vocale, la reconnaissance de l'écriture manuscrite, la reconnaissance de l'activité et la prédiction de la trajectoire du véhicule ne sont que quelques-unes des disciplines où le

modèle de Markov caché (HMM) a été utilisé. Dans cette étude, nous proposons un nouvel APS pour VANET dans le contexte d'un environnement urbain.

Le risque de collision est considéré comme une variable latente qui peut être mesurée à l'aide de divers facteurs tels que la vitesse, les conditions météorologiques, le lieu de la collision, la densité des véhicules et la fatigue du conducteur. La figure 1 montre comment HMM a été utilisé pour modéliser le lien entre ces observations et le risque d'effondrement.

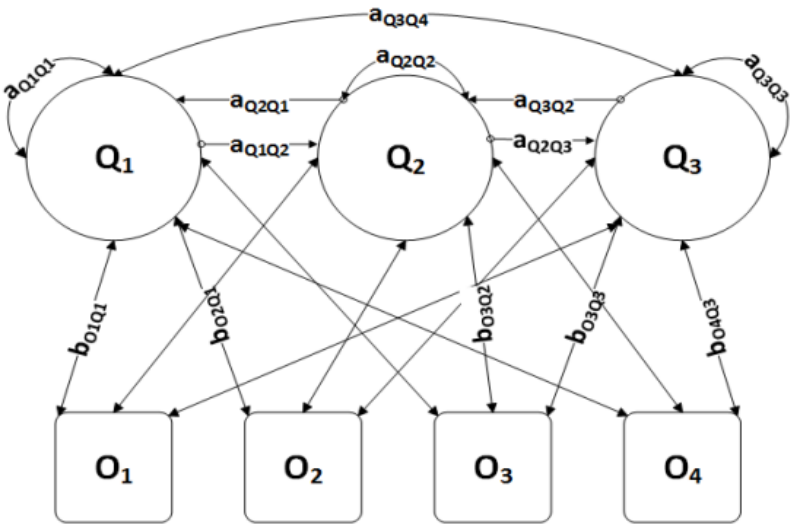


Figure 2. Example of Graphical Representation of Hidden Markov Model (HMM).

Figure 11:Exemple de modèle de markov cachée

La figure 11 fournit une illustration de la représentation graphique d'un HMM.

Un ensemble d'états et un ensemble d'observations constituent la majorité des HMM. Les états sont représentés sur la figure 2 par (Q1, Q2, Q3) et les observations sont représentées par (O1,O2,O3,O4).

3.Modélisation des systèmes :

L'APS suggéré a été modélisé comme un HMM, comme cela a déjà été noté, les États représentant le niveau de risque d'accident et les observations du HMM représentant les facteurs d'accident.

4.Paramètres HMM :

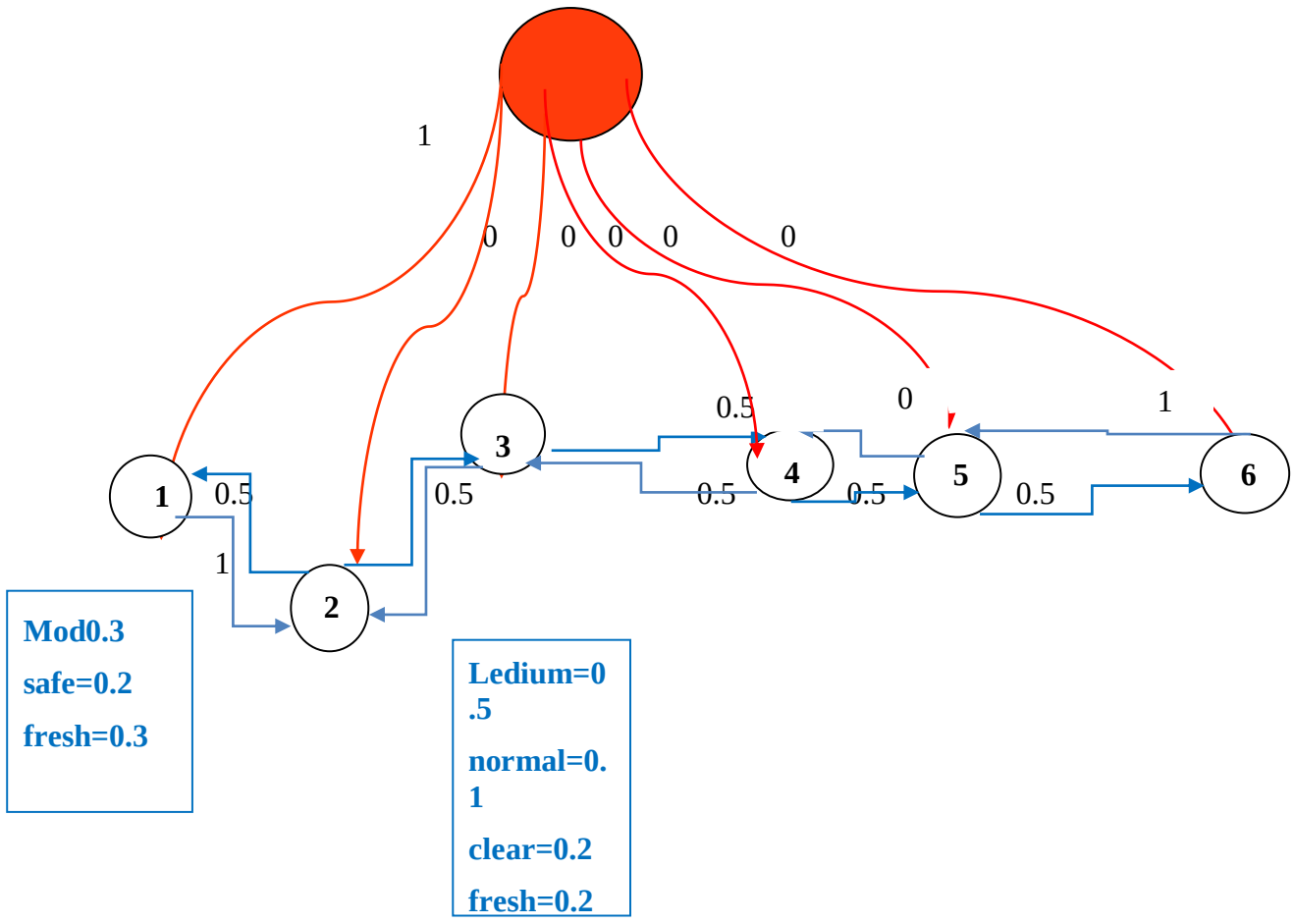
La stratégie d'évitement de collision suggérée a été modélisée sous la forme d'un HMM, où les états cachés correspondent aux niveaux de risque de collision potentiels et l'espace d'observation est déterminé par divers paramètres d'observation. Chaque composant de l'espace d'observation

est composé d'une variété d'éléments qui, pris dans leur ensemble, reflètent l'environnement de l'accident de manière holistique à un moment donné (voir tableau suivant).

Factor	Value Range
States	Negligible, Low, Moderate, High, Very high, Deadly
S	Very Slow, Slow, Medium, High, Very high, Extreme
L	Safe, Normal, Dangerous, Deadly
W	Clear, Sunny, Rainy, Foggy, Snowing
V	Low, Medium, High
D	Fresh, Medium, Tired

Figure 12 :Table Observation factors

S représente la vitesse actuelle du véhicule, L le risque d'accident à l'emplacement actuel, W les conditions météorologiques actuelles, V la densité actuelle des véhicules et D la fatigue actuelle du conducteur, respectivement. Les cinq éléments S, L, W, V et D définissent ensemble l'élément unique Ox dans l'espace d'observation, Ox = {S, L, W, V, D}



5.Intégration du modèle HMM dans PRISM :

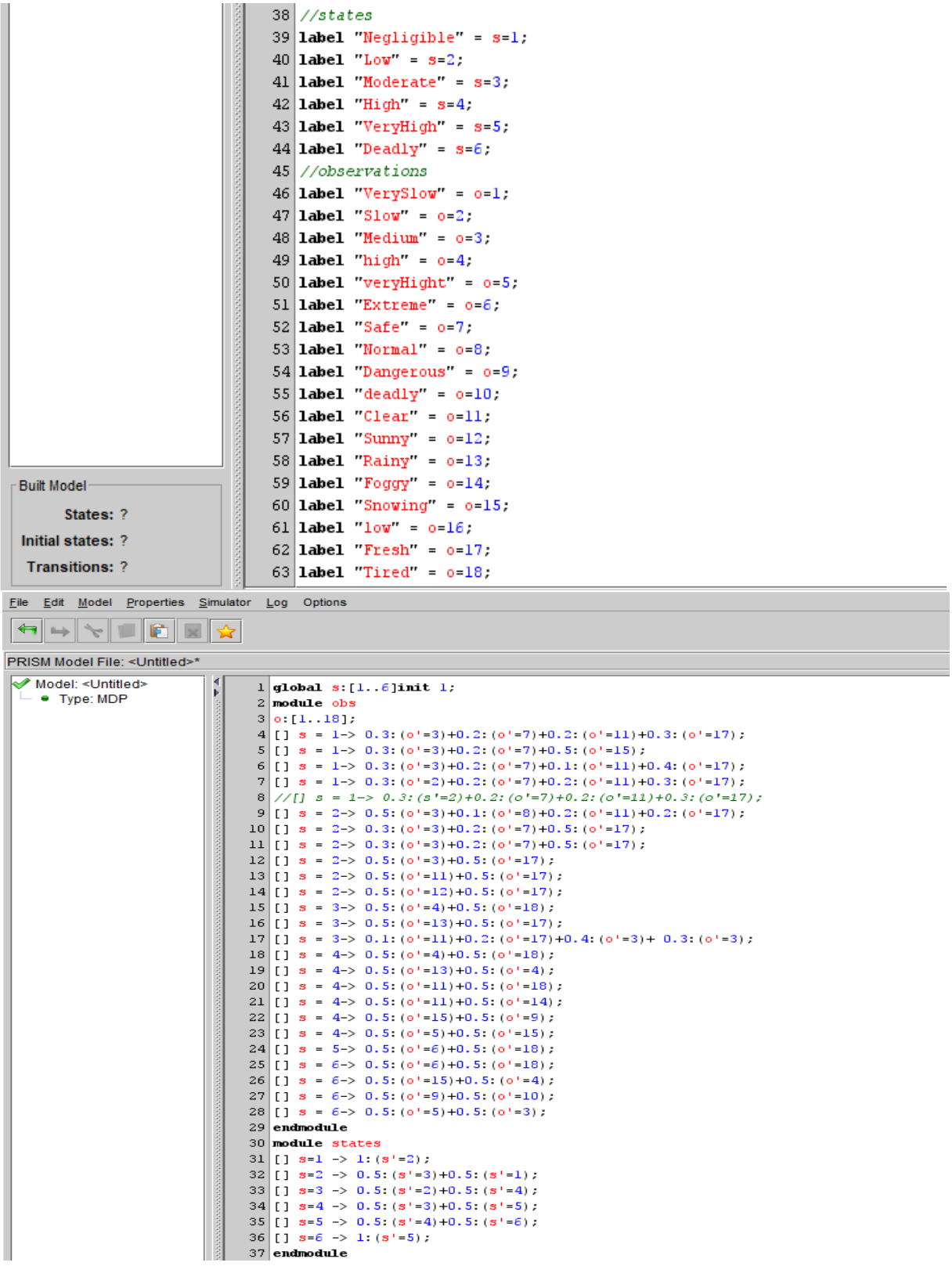


figure 13 : integration du modele hmm dans prism

1 er ligne : déclaration s global variable de type tableau de taille 6 du 1 au 6 initialiser a

6(déclaration de 6 states de hmm)

[s global car utiliser en model observation (2…29) et en model state (30…37); initialiser a 6 car state 6 la seule initiale dans hmm(.

2 eme …29 eme ligne (model observation) : c'est une traduction de matrice μ de probabilités

des observations définie le variable o exprime les observation

(o=1[observation very slow « ligne 46 »], o=2[observation slow « ligne 47»],

o=3[observation medium « ligne 48 ») par exemple dans la ligne 5 :si s= 1 alors

o=1(« veryslow »)avec probabilité de 0.3 ou o =7(« safe »)avec probabilité de 0.2 ou o=15(«snowing»)avec probabilité de 0.5.

30eme …. 37eme (model state) : c'est une traduction de matrice P de probabilités de transition entre les états0.

Les matrices de probabilité d'émission pour chaque facteur d'observation sont respectivement désignées par BS, BL, BW, BV et BD. Ils représentent chacun la matrice d'émission des facteurs suivants : vitesse, emplacement, météo, densité de véhicules et niveau de fatigue du conducteur.

$$B_S = \begin{pmatrix} P_{\{verySlow,Negligible\}} & \cdots & P_{\{verySlow,Deadly\}} \\ \vdots & \ddots & \vdots \\ P_{\{Extreme,negligible\}} & \cdots & P_{\{Extreme,Deadly\}} \end{pmatrix}$$

$$B_L = \begin{pmatrix} P_{\{Low,Negligible\}} & \cdots & P_{\{Low,Deadly\}} \\ \vdots & \ddots & \vdots \\ P_{\{High,Negligible\}} & \cdots & P_{\{High,Deadly\}} \end{pmatrix}$$

$$B_W = \begin{pmatrix} P_{\{Clear,Negligible\}} & \cdots & P_{\{Clear,Deadly\}} \\ \vdots & \ddots & \vdots \\ P_{\{Snowing,Negligible\}} & \cdots & P_{\{Snowing,Deadly\}} \end{pmatrix}$$

$$B_V = \begin{pmatrix} P_{\{Low,Negligible\}} & \cdots & P_{\{Low,Deadly\}} \\ \vdots & \ddots & \vdots \\ P_{\{High,Negligible\}} & \cdots & P_{\{High,Deadly\}} \end{pmatrix}$$

$$B_D = \begin{pmatrix} P_{\{Fresh,Negligible\}} & \cdots & P_{\{Fresh,Deadly\}} \\ \vdots & \ddots & \vdots \\ P_{\{Tired,Negligible\}} & \cdots & P_{\{Tired,Deadly\}} \end{pmatrix}$$

where:

$$\sum_{j=0}^{|B_x|} B_x(i,j) = 1 | x \epsilon \{S, L, W, V, D\}$$

La formule de la moyenne pondérée a été utilisée pour obtenir la probabilité fusionnée suite au calcul des matrices d'émission distinctes.

Les pondérations ont été déterminées à partir de données historiques sur les facteurs qui conduisent à des accidents.

L'effet du facteur actuel sur l'observation totale sera déterminé par chaque poids. Selon les statistiques antérieures de reconstitution des accidents de la circulation, le poids du facteur augmente à mesure qu'il contribue aux causes de l'accident.

BOxSj = (WsBs[Sx, j]) + (WLBL[Lx, j]) + (WwBw[Wx, j]) + (WvBv[Vx, j]) + (WDBD[Dx,j])

où:

Ox = {Sx, Lx, Wx, Vx, Dx}

Et où WS, WL, WW, WV et WD sont les poids de chaque facteur.

6.Une simulation de modèle en PRISM :

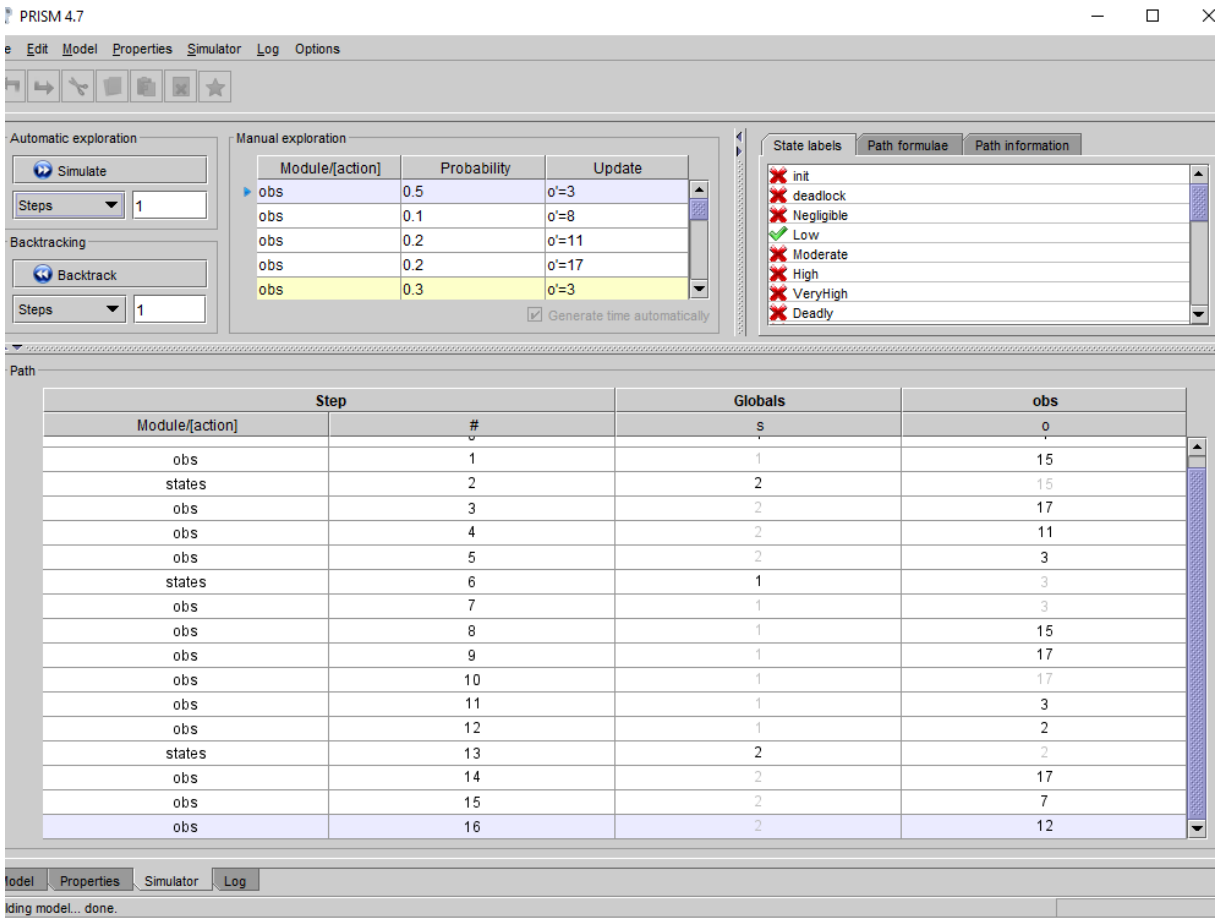


figure 14:Une simulation de modèle en PRISM :

Graphe 1 :

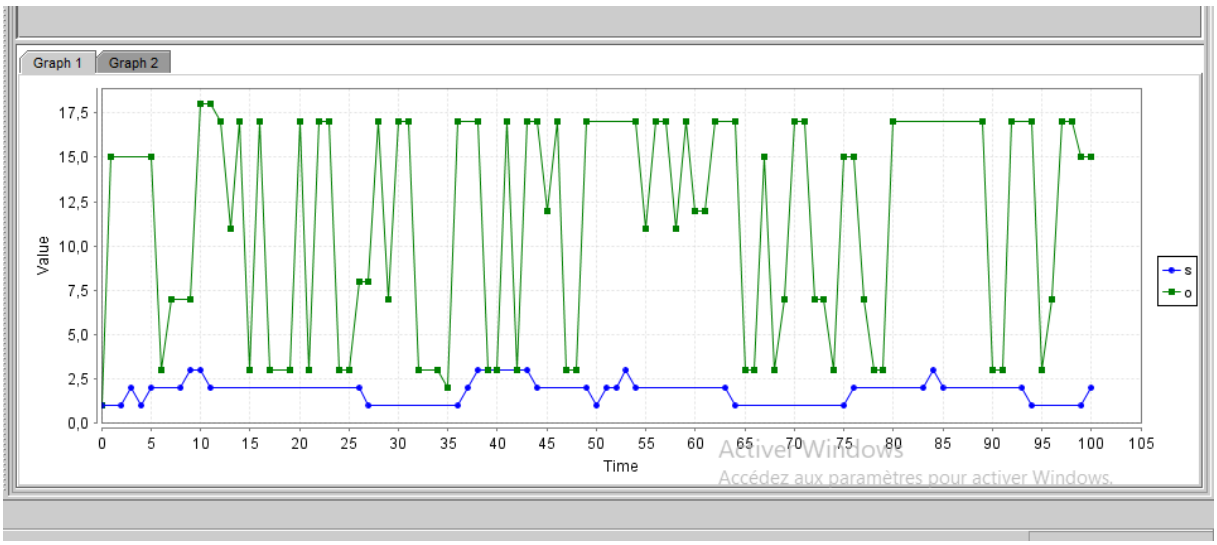


figure 15: capture d'ecran graphe numéro 1

Graphe 2 :

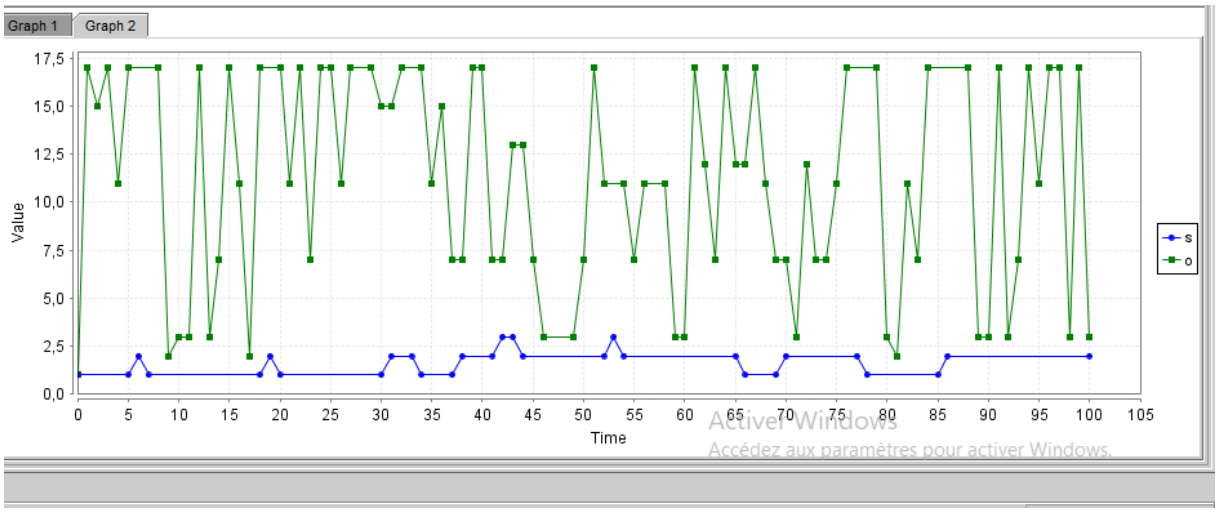


figure 16: capture d'ecran graphe numéro 2

7.Vérification des propriétés

7.1.Vérification manuelle des propriétés :

- **La propriété 1 :** $P_{<= 1}$ [X_e !(Medium)].

s satisfait P<=1 (Xe !(Medium)) ssi ps(Xe !(Medium)) <=1.

On a: ps(XΩ□) = μs(Ω) . ∑s'□ sat (□) p(s,s') , □!= Medium donc s' !={s6}

Dans ce qui suit, chaque ligne correspond à la satisfaction de la propriété au sein d'un état (si, i=1-6)

ps1 [Xe !(Medium)] = μs1(e) . ∑s'□ sat !(Medium) p(s1,s')

ps2 [Xe !(Medium)] = μs2(e) . ∑s'□ sat !(Medium) p(s2,s')

ps3 [Xe !(Medium)] = μs3(e) . ∑s'□ sat !(Medium) p(s3,s')

ps4 [Xe !(Medium)] = μs4(e) . ∑s'□ sat !(Medium) p(s4,s')

ps5 [Xe !(Medium)] = μs5(e) . ∑s'□ sat !(Medium) p(s5,s')

ps6 [Xe !(Medium)] = μs6(e) . ∑s'□ sat !(Medium) p(s6,s')

Nous venons de voir que la formule $P_{<= 1}$ [X_e !(Medium)]. est satisfaite dans tous les états du modèle (si, i=1 ... 6). Ainsi la formule (X_e !(Medium)) voire la propriété (atteignabilité) est vérifiée pour tout le modèle, et par conséquent nous déduisons la atteignabilité de fonctionnement du système.

- **La propriété 2 :** $P_{>=0}$ [X_E (Medium)]

s satisfait P>= 0 (Xe Medium) ssi ps(Xe Medium) >= 0.

On a: ps(XΩ□) = μs(Ω) . ∑s'□ sat (□) p(s,s') , □= Medium donc s'= {s6}.

Dans ce qui suit, chaque ligne correspond à la satisfaction de la propriété au sein d'un état (si, i=1-6)

ps1 [Xe (Medium)] = μs1(e) . ∑s'□ sat (Medium) p(s1,s')

ps2 [Xe (Medium)] = μs2(e) . ∑s'□ sat (Medium) p(s2,s') = 0.5

ps3 [Xe (Medium)] = μs3(e) . ∑s'□ sat (Medium) p(s3,s')

ps4 [Xe (Medium)] = μs4(e) . ∑s'□ sat (Medium) p(s4,s')

ps5 [Xe (Medium)] = μs5(e) . ∑s'□ sat (Medium) p(s5,s')

ps6 [Xe (Medium)] = μs6(e) . ∑s'□ sat (Medium) p(s6,s') =

Nous venons de voir que la formule P>= 0 [Xe (Medium)] est satisfaite dans tous les états du modèle (si, i=1 ... 6). Ainsi la formule (Xe Medium) voire la propriété (sûreté) est vérifiée pour tout le modèle, et par conséquent nous déduisons la sûreté de fonctionnement du système.

- **La propriété 3 :** $P_{<= 0.008}$ [X_E (Medium)]

s satisfait P<=0.008 (Xe Medium) ssi ps(Xe Medium) <= 0.008.

On a: ps(XΩ□) = μs(Ω) . ∑s'□ sat (□) p(s,s') , □= Medium donc s'= {s9}

Dans ce qui suit, chaque ligne correspond à la satisfaction de la propriété au sein d'un état (si, i=1-6)

ps1 [Xe (Medium)] = μs1(e) . ∑s'□ sat (Medium) p(s1,s')

ps2 [Xe (Medium)] = μs2(e) . ∑s'□ sat (Medium) p(s2,s')

ps3 [Xe (Medium)] = μs3(e) . ∑s'□ sat (Medium) p(s3,s') .

ps4 [Xe (Medium)] = μs4(e) . ∑s'□ sat (Medium) p(s4,s')

ps5 [Xe (Medium)] = μs5(e) . ∑s'□ sat (Medium) p(s5,s')

ps6 [Xe (Medium)] = μs6(e) . ∑s'□ sat (Medium) p(s6,s')

Nous venons de voir que la formule P<= 0.008 [Xe (Medium)] est satisfaite dans tous les états du modèle (si, i=1 ... 6). Ainsi la formule (Xe Medium) voire la propriété (sûreté) est vérifiée pour tout le modèle, et par conséquent nous déduisons la sûreté de fonctionnement du système.

7.2.Vérification automatique des propriétés par PRISM :

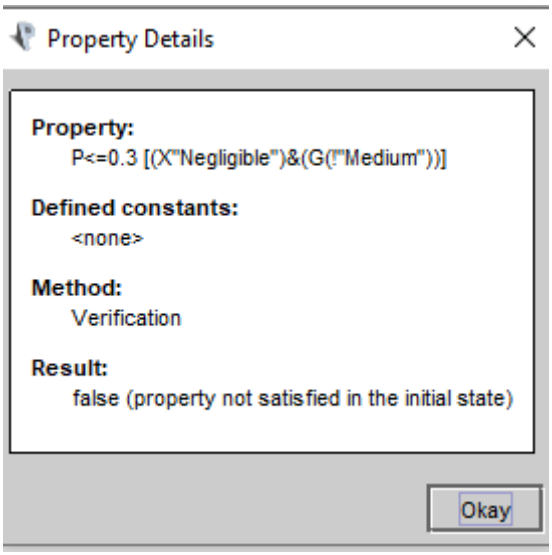


Figure 17:vérification de propriété 01 «P<= 0.5 [Xe !(Medium

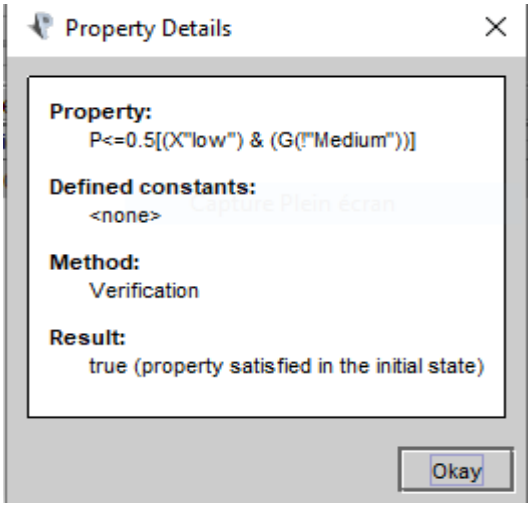


Figure 18 :vérification de propriété 02 «P<= 10.5[Xe !(Medium

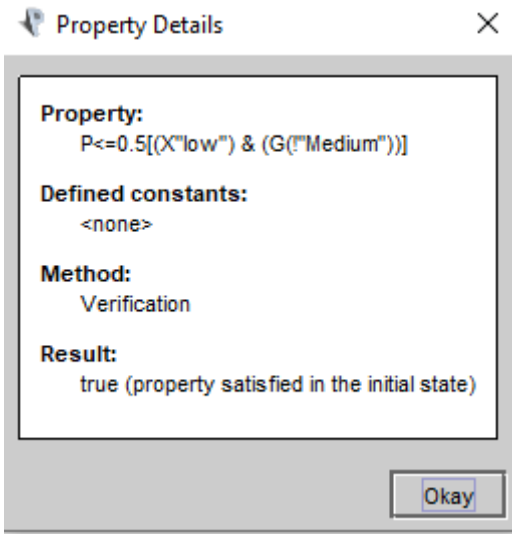


Figure 19 :vérification de propriété 03 «P<= 0.5 [Xe !(Medium

8.conclusion :

Nous avons présenté dans ce chapitre une étude de cas du model-checker PRISM utilisant le système de prévision des accidents (APC), un système embarqué réel très important. Nous avons commencé avec le modèle HMM de l'APC basé sur son modèle dans [34] et avons ajouté l'ensemble d'observations, la matrice de distribution de probabilité d'observation B et le vecteur de distribution de probabilité de transition initiale. Théoriquement, après avoir intégré le modèle et la propriété dans l'outil, il ne reste plus qu'à cliquer sur le bouton de l'outil pour obtenir le résultat escompté, à savoir la réponse à la requête « Le modèle vérifie-t-il les spécifications ? »

Mais ce calcul peut ne pas être précis en raison de contraintes machine (par exemple par manque de mémoire). La modélisation doit alors être considérablement simplifiée (tout en conservant les propriétés critiques du système) avant d'être relancée.

Nous avons décidé de proposer trois attributs dans l'étape de définition des propriétés permettant de valider le bon fonctionnement du système, et nous avons ensuite exprimé ces caractéristiques à l'aide de la logique POCTL. Parce que pour un HMM donné, on s'intéresse fréquemment aux propriétés du processus stochastique fondamental que l'on peut considérer comme un modèle DTMC, toutes les propriétés choisies sont de type Next suivies d'une observation, ce qui est le cas qui illustre notre approche dans comparaison avec d'autres model-checkers.

Nous nous intéressons également à déduire les caractéristiques d'autres ensembles de processus stochastiques qui aboutissent à des observations. Les trois propriétés spécifiées dans POCTL dans la troisième section du modèle HMM construit dans la deuxième section ont été vérifiées pour la satisfaction ou la non-satisfaction lors de la dernière étape de la vérification du modèle, connue sous le nom de vérification des propriétés. Cela a été fait manuellement en utilisant

l'algorithme de vérification de modèle pour POCTL sur le modèle HMM et automatiquement en utilisant PRISM pour valider le bon fonctionnement du système APC.

• Conclusion générale

Plusieurs sinistres de systèmes embarqués sont connus pour s'être produits. Étude du cas En utilisant une méthode qui facilite la validation du modèle probabiliste, on diminue la valeur des erreurs et des catastrophes (PRISM).

Dans cette étude, nous nous sommes concentrés sur une partie de la vérification automatique basée sur le modèle probabiliste, une méthode plus générale et réaliste, qui ne cesse de faire de plus en plus d'adeptes dans le domaine de la vérification probabiliste.

Cet effort bien fait a permis de :

Afin de modéliser différents systèmes probabilistes à partir d'un ensemble d'observations et dont le nombre d'états du système peut ne pas être connu, il est nécessaire d'introduire pour la première fois le formalisme mathématique "HMM" dans le domaine de la vérification probabiliste. D'une part, les HMM sont des outils simples, riches en propriétés, basés sur des statistiques solides, et ils peuvent être utilisés pour modéliser une variété de systèmes probabilistes. Cependant, ils sont également largement utilisés dans le domaine des ordinateurs.

 les études de cas PRISM doivent être améliorées.

Le modèle HMM, un nouveau formalisme de modélisation, et la logique POCTL, issue de PCTL, un nouveau formalisme de spécification, sont intégrés pour la première fois dans PRISM. Il n'existe pas de vérificateur de modèle probabiliste connu qui utilise POCTL et HMM.

Les travaux discutés dans ce livre peuvent être approfondis en poursuivant diverses directions de recherche. Nous avons noté plus précisément quelques aspects qui intriguent :

- L'outil PRISM a maintenant de nouvelles fonctionnalités.

Extension de la vérification à davantage d'attributs.

- La réalisation de plus d'études de cas utilisant de véritables systèmes embarqués.

- Visiter une plateforme de vérification de systèmes probabilistes.

- La création de langages de description de systèmes, car c'est dans le travail de modélisation que réside le vrai défi du model-checking.

A. Références Bibliographiques

[1] Nokovic B., 2016. Verification and implementation of embedded systems from high-level models, Ph.D. thesis, McMaster University, Canada. 164p.

[2] Belaidi, H. (2020). *Conception d'un disjoncteur électronique pour la protection des équipements et des personnes* (Doctoral dissertation, Université Mouloud Mammeri).

[3] Nleng C. H., 2014. Modélisation et vérification du flux d'information pour les systèmes orientés objets, Maîtrise ès sciences appliquées, département de génie informatique et génie logiciel, Ecole polytechnique de Montréal, Canada, 136p.

[4] Braham Lotfi Mediouni Modeling and Analysis of Stochastic Real-Time Systems Université Grenoble Alpes, 2019. English. f

[5] Les systèmes embarqués Introduction (Richard Grisel – Professeur des Universités – Université de RouenNacer Abouchi – Professeur – ESCPE Lyon)

[6] Parker D., 2002. Implementation of symbolic model checking for probabilistic systems,these de doctorat, University of Birmingham.

[7] Murat V., 2014. Extensions d'automates d'arbres pour la vérification de systèmes à états infinis. Thèse de doctorat en informatique, Université de Rennes 1, France, 220p.

[8] Christel B., Joost-Pieter K., & al., 2008. Principles of model checking, volume 26202649. MIT press Cambridge.

[9] Parker D., 2002. Implementation of symbolic model checking for probabilistic systems, these de doctorat, University of Birmingham.

[10] Muhimpundu J., 2014. Analyse de l'erreur dans la vérification probabiliste, Maîtrise en informatique, Université Laval, Québec, Canada. 106p.

11 Janeza Trdine 9, 51000 Rijeka, Croatia,HIDDEN MARKOV MODELS, THEORY AND APPLICATIONS Edited by Przemyslaw Dymarski

12 Hans Hansson .Bengt Jonsson A logic for reasoning about time and reliability

13 Sophie Demassey Méthodes hybrides de programmation par contraintes et programmation linéaire pour le problème d'ordonnancement de projet à contraintes de ressources .Université d'Avignon, 2003. Français

14 Gary Laski Le design : Théorie esthétique de l'histoire industrielle Université Paris-Est, 2011. Français.

15 Birney E., 2001. Hidden Markov models in biological sequence analysis. IBM Journal of Research and Development, Vol. 45(3), 449–454.

16 Nil geisweiller.2006.etud sur la modelisation et la vérification probabiliste.

17 Hansson H. & Jonsson B., 1994. A logic for reasoning about time and probability, Formal Aspects of Computing, Vol. 6(5), 512-535.

18 Ferroum A. & Boudour R., 2016. A tool for automatic verification of Embedded systems.3rd International Conference on Embedded Systems in

Telecommunications and Instrumentation (ICESTI'16), Annaba, Algeria.
19 Bouguerra Bilal .On the Verification of Internet of Things (IoT) Technologies using PRISM Model Checker
20 Nokovic B. & Sekerinski E., 2014. Verification and Code Generation for Timed Transitions in pCharts. Proceedings of the International Conference on Computer Science, Software Engineering, Montreal, Canada, 1-10.
21 Marta Kwiatkowska Gethin Norman .Quantitative Analysis With the Probabilistic Model Checker PRISM .
22 Nyothiri Aung 1 , Weidong Zhang 2, Sahraoui Dhelim 1 and Yibo Ai.accident Prediction System Based on Hidden Markov Model for Vehicular Ad-Hoc Network in Urban
23 Environments.School of Computer Science and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China .
24 Andrew Hinton .Software Project M60: Simulator for the Probabilistic Model Checker PRISM .University of Birmingham April 2005
25 DAVID ANTHONY PARKER.University of Birmingham 2002.IMPLEMENTATION OF SYMBOLIC MODEL CHECKING FOR PROBABILISTIC SYSTEMS
26 L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In A. Waibel and K.-F. Lee, editors, Readings in Speech Recognition, pages 267–296. Kaufmann, San Mateo, CA, 1990. M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. In K. Larsen and P. Niebert, editors, Proc. Formal Modeling and Analysis of Timed Systems (FORMATS'03), volume 2791 of LNCS, pages 105–120. Springer-Verlag, 2003.
27 Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. In K. Larsen and P. Niebert, editors, Proc. Formal Modeling and Analysis of Timed Systems (FORMATS'03), volume 2791 of LNCS, pages 105–120. Springer-Verlag, 2003.
28 PRISM web site. www.cs.bham.ac.uk/~dxp/prism.
29 M. Kwiatkowska, G. Norman, and D. Parker. PRISM users' guide. Available from www.cs.bham.ac.uk/~dxp/prism.
30 Instructors: Jacob Andreas and Davis Foote --- University of California, Berkeley These slides were created by Dan Klein and Pieter Abbeel for CS188 Intro to Al at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.)
31 Joël Muhimpundu..Québec2014, Canada .Analyse de l'erreur dans la vérification probabiliste .
32 Baier, C., & Katoen, J. P. (2008). *Principles of model checking*. MIT press.
33 DAVID ANTHONY PARKER .University of Birmingham August 2002.IMPLEMENTATION OF SYMBOLIC MODEL CHECKING FOR PROBABILISTIC SYSTEMS .
34 Matthieu Petit, Christiansen Henning .Un calcul de Viterbi pour un Modèle de Markov Caché Contraint
35 Alur, R., Courcoubetis, C. and Dill, D.: Model-checking for real-time systems. In*Proc. 5 th IEEE Int. Symp. on Logic in Computer Science*, pages 414–425, 1990.

36 Daniel Lehmann Michael O. Rabin.symposium on Principles of programming languagesJanuary.1981.

37 Nicolas markey.logique toporelle pour la vérification.

38 Nyothiri Aung ,Weidong Zhang ,Sahraoui Dhelim and Yibo Ai .Accident Prediction System Based on Hidden Markov Model for Vehicular Ad-Hoc Network in Urban Environments University of Science and Technology Beijing, Beijing 1 .

39 S. Haddad 1 13 octobre 2010 .Systèmes temporisés probabilistes 2009 - 2010 .

40 Analyses et preuves formelles d'algorithmes distribués probabilistes Allyx Fontaine .

41 Nokovic B., 2016. Verification and implementation of embedded systems from high-level models, Ph.D. thesis, McMaster University, Canada. 164p.

42 Claude Gravel.Échantillonnage des distributions continues non uniformes en précision arbitraire et protocole pour l'échantillonnage exact distribué des distributions discrètes quantiques.Université de Montréal .Mars, 2015 .

43 M. Kwiatkowska, G. Norman, and D. Parker. PRISM 2.0: A tool for probabilistic model checking. In Proc. 1st International Conference on Quantitative Evaluation of Systems (QEST'04), pages 322–323. IEEE Computer Society Press, 2004

44 Saadoune Nadia Université Hassan .bouchaib Radi Université Hassan.Probabilistic study an embedded system.2015.

45 Aung, Nyothiri, et al. "Accident prediction system based on hidden markov model for vehicular ad-hoc network in urban environments." *Information* 9.12 (2018): 311.

B. Références Web (Techniques)

[W1] https://fr.wikipedia.org/wiki/Syst%C3%A8me_embarqu%C3%A9

[W2] https://tel.archives-ouvertes.fr/tel-02467581/

[W4] https://link.springer.com/chapter/10.1007/3-540-46029-2_13

[W5] Calder, Muffy, et al. "Analysis of signalling pathways using the PRISM model checker." (2005).

[W6] Wilpon, J. G., and L. R. Rabiner. "Application of hidden Markov models to automatic speech endpoint detection." *Computer Speech & Language* 2.3-4 (1987): 321-341.

[W7] https://fr.wikipedia.org/wiki/Cha%C3%AEne_de_Markovhttps://en.wikipedia.org/wiki/Hidden_Markov_model

[W8] https://en.wikipedia.org/wiki/Hidden_Markov_model

[W9] ""https://www.prismmodelchecker.org/

W[10] https://www.prismmodelchecker.org/lectures/pmc/
]

Annexe A

Résumé

Un développement plus récent des méthodes de vérification de modèle conventionnelles est connu sous le nom de vérification de modèle probabiliste, qui permet l'étude intégrée des aspects qualitatifs et quantitatifs des systèmes stochastiques.

Dans cette lettre, nous définissons d'abord un système embarqué, puis discutons de ses nombreux types, de ses utilisations potentielles, de sa convivialité et de ses services. Ensuite, après avoir appris le langage de l'outil PRISM et comment l'utiliser, nous avons discuté de plusieurs modèles de probabilités aléatoires que nous pourrions utiliser pour simuler le comportement de ce système.

Ensuite, pour examiner le système de prévision des accidents, nous avons effectué une vérification de modèle probabiliste (APS). en utilisant un modèle de Markov caché généré par PRISM.

Abstract

A more recent development of conventional model-checking methods is known as probabilistic model checking, which allows for the integrated investigation of both qualitative and quantitative aspects of stochastic systems.

In this letter, we first define an embedded system and then discuss its many types, potential uses, usability, and services. Then, after learning the PRISM tool's language and how to use it, we discussed several random probability models that we might use to simulate the behavior of this system.

Next, to examine the Accident Prediction System, we performed probabilistic model verification (APS). utilizing a PRISM-generated hidden Markov model.

ملخّص

يُعرف التطوير الأكثر حداثة لطرق فحص النماذج التقليدية باسم فحص النموذج الاحتمالي ، والذي يسمح بالتحقيق المتكامل لكل من الجوانب النوعية والكمية للأنظمة العشوائية.

في هذه الرسالة ، نحدد أولاً نظامًا مضمنًا ثم نناقش العديد من أنواعه ، واستخداماته المحتملة ، وقابليته للاستخدام ، والخدمات. بعد ذلك ، بعد تعلم لغة أداة PRISMوكيفية استخدامها ، ناقشنا العديد من نماذج الاحتمالات العشوائية التي قد نستخدمها لمحاكاة سلوك هذا النظام.

بعد ذلك ، لفحص نظام التنبؤ بالحوادث ، أجرينا التحقق من النموذج الاحتمالي (APS). باستخدام نموذج ماركوف المخفي الذي تم إنشاؤه بواسطة PRISM.