Spécification et Validation d'un Contrôleur de Performances Sportives

Mohammed Al'Achhab, Ahmed Hammad, Hassan Mountassir

Spécification et Validation d'un Contrôleur de Performances Sportives

Mohammed Al Achhab, Ahmed Hammad et Hassan Mountassir
LIFC-Laboratoire d'Informatique de l'Université de Franche-Comté
16, route de Gray 25030 Besançon Cedex
Email : {alachhab, hammad, mountass}@lifc.univ-fcomte.fr

***Abstract*— Ce papier a pour objectif de montrer l'intérêt d'utiliser les méthodes formelles pour spécifier et valider un système médical, nous utilisons les automates hiérarchiques pour modéliser ces systèmes. La sémantique utilisée est celle d'une structure de Kripke où les états sont valués par des propositions atomiques. Cette structure peut être de grande taille en nombre d'états. Par nature, les systèmes hiérarchiques sont définis de manière hiérarchique par un ensemble de sous systèmes en éclatant à chaque fois un ou plusieurs états en un ensemble d'automates. Dans cet article, nous proposons de vérifier des proporiétés, seulement sur ces sous-systèmes. Pour pallier au problème de l'explosion combinatoire et la vérification de propriétés, nous considérons que les sous systèmes concernés par la propriété à vérifier et en déduire sa vérification sur le système global. Les résultats sont illustrés sur l'exemple d'un contrôleur de performances sportives.**

I. Motivations et problématique

Au cours de la dernière décennie, le développement de logiciels n'a pas cessée de croitree dans tous les domaines. En effet l'introduction de composantes informatiques s'avère de plus en plus importante dans les secteurs comme les systèmes de gestion pour l'aviation, les réseaux de téléecommunication, les banques, les systèmes de transport, les protocoles de paiement électronique et bien d'autres secteurs. L'être humain est et va de plus en plus être confronté à des exigences croissantes quant au bon fonctionnement de ces systèmes, et plus particulièrement pour les systèmes informatiques critiques. Nous qualifions de système critique, tout système dont les défaillances peuvent avoir des conséquences dramatiques et catastrophiques en termes humains ou économiques. Les systèmes de contrôle et les outils médicaux sont les systèmes critiques les plus répandus. Un exemple malheureux de système médical est la machine de radiation $THERAC25$ [1], développée par la compagnie AECL[1]. Cette machine, utilisée entre 1985 et 1987 pour le traitement du cancer, permettait de détruire les tumeurs en émettant des radiations sur celles-ci sans affecter l'organe touché par la tumeur. Cependant, une anomalie dans son fonctionnement a entraîné la mort de 6 patients ; en effet, une erreur dans la programmation du logiciel conduisait dans certaines situations à utiliser erronément la machine dans un mode Rayon-X au lieu du mode faisceau d'électrons, ce qui envoyait 125 fois plus de radiations que dans la dose prescrite par le médecin. Ainsi, il est important de spécifier avec précision ces systèmes et par la suite vérifier les propriétés exigées par l'utilisateur, afin d'assurer un développement fiable de tels systèmes, de réduire le coût de maintenance et de minimiser le risque de pertes économiques et humaines.

Notre travail s'articule autour de la spécification et la vérification de systèmes hiérarchique réactifs issus des travaux sur les $StateCharts$ [2] et Unified Modeling Language (UML) [3]. Nous utilisons les automates hiérarchiques [4] pour modéliser les systèmes hiérarchiques réactifs. Cette modélisation possède, en outre, des concepts tels que le raffinement d'états, des transitions qui ont plusieurs états de départ et plusieurs états d'arrivée (Interlevel Transitions), la priorité entre les transitions et l'exécution simultanée des transitions. Ces concepts rendent difficile l'utilisation directe de méthodes formelles.

Diverses méthodes et techniques ont été introduites afin d'effectuer la vérification de propriétés par model-checking sur les modèles hiérarchiques [5], [6], [7]. Dans [8], les auteurs montrent comment les StateCharts peuvent être traduits en $PROMELA$ (le langage de spécification qu'utilise $SPIN$ [9]) en utilisant les automates hiérarchiques comme format intermédiaire. Les techniques de vérification utilisées sont algorithmiques et sont basées souvent sur la détection de cycles pour montrer la satisfaction ou non d'une propriété. Les propriétés sont exprimées à l'aide d'une logique comme la LTL (Linear Temporal Logic) [10]. Le principal problème réside dans la complexité des procédures de décision et l'explosion combinatoire du nombre d'états.

Dans ce papier, nous proposons d'utiliser la vérification par modules (ou modulaire) sur notre exemple. Cette méthode est utilisée pour vérifier les systèmes de grande taille. Il consiste à découper le modèle de système en un ensemble de modules et de vérifier si une propriété est satisfaite sur chaque module. La vérification de chaque module se fait d'une manière séparée et indépendante [11]. Le choix des propriétés à vérifier et la méthode de découpage sont deux points très importants, parce que ce n'est pas toujours vrai que : si une propriété est vérifiée sur tout les modules alors elle est satisfaite sur le modèle global du système. Cela découle de la manière de découper le modèle global en modules. Par nature, les automates hiérarchiques sont définis par des sous automates hiérarchiques.

Nous proposons des schémas de propriétés vérifiées sur chaque module du système et qui le sont sur le système

[1]Atomic Energy of Canada Limited

global. Les modules du systèmes sont appelés sous structure de Kripke, notée SSK. Chaque sous structure de Kripke représente un aplatissement d'une partie de système représenté par un ensemble des automates associés à un état. La vérification des propriétés sera donc effectuée sur les sous structures de Kripke associées aux états raffinés par ces automates. Si les SSK vérifient la propriété ' alors la structure de Kripke globale, associée à l'automate hiérarchique global, la vérifie également. Cette méthode est appelée la vérification modulaire ou par parties.

La suite de ce papier est organisée en quatre sections. Dans la section deux, nous présentons les concepts de base : automate séquentiel et automate hiérarchique. Dans la section trois, nous présentons la sémantique d'automate hiérarchique sous forme de structure de Kripke. Dans la section quatre, nous présentons la démarche de vérification modulaire : définition de la partie à vérifier et schémas de propriétés. Dans la section cinq, nous traçons quelques perspectives à ce travail. Nous illustrons nos propos par un exemple de contrôleur de performances des sportives.

A. Automate hiérarchique

Un automate hiérarchique, noté AH, est un ensemble d'automates qui sont liés entre eux par une fonction de composition. La fonction de composition fait le lien entre un état s d'un automate séquentiel (état père) et un ensemble d'automates (automates fils).

Définition 1: (automate hiérarchique) AH est défini par le 3-uplet $\langle F, E, \gamma \rangle$, où : $F = \{A_1, \dots A_n\}$ est un ensemble d'automates ayant un ensemble d'états distincts, $E = \cup_{A \in F} \Sigma_A$ est un alphabet fini de noms d'actions et $\gamma : \bigcup_{A \in F} S_A \longrightarrow 2^F$ est une fonction de composition sur F tel que : dans l'ensemble F il y a un seul automate racine A_{racine}, chaque automate $A \in F \backslash \{A_{racine}\}$ a *un état père* et la fonction de composition ne doit pas contenir de cycles : $\forall S \subseteq \bigcup_{A \in F} S_A . \exists s \in S . S \cap \bigcup_{A \in \gamma(s)} S_A = \emptyset$.

On note χ l'application qui lie un état raffiné avec les états de l'automate fils. Nous définissons χ^+ comme la fermeture transitive non-réflexive, χ^* comme la fermeture transitive réflexive de χ.

Proposition 1: Le décor de l'état fils implique celui de son ascendant : $\forall s' \in \chi(s) \Longrightarrow (L(s') \Longrightarrow L(s))$.

Définition 2: (sous-automate hiérarchique) Soit $AH = \langle F, E, \gamma \rangle$ un automate hiérarchique, la restriction de la fonction de composition γ aux états d'un automate $A \in F$ nous permet de définir *le sous-automate hiérarchique* $AH_A = \langle F_A, E_A, \gamma_A \rangle$ telle que : $F_A = F \backslash \{A_i | S_{A_i} \cap \chi^*(S_A) = \emptyset\}$, $E_A = E$ et $\gamma_A = \gamma | \chi^*(S_A)$ (On considère A comme l'automate racine de AH_A).

B. Exemple d'un contrôleur de performances sportives

Nous donnons un exemple de contrôleur de performances sportives des cyclistes, c'est un appareil d'entraînement basé sur un cardiofréquencemètre. Cet appareil permet d'enregistrer principalement les fréquences cardiaques d'un cycliste pendent ces entraînements. Il se compose d'un récepteur des signaux

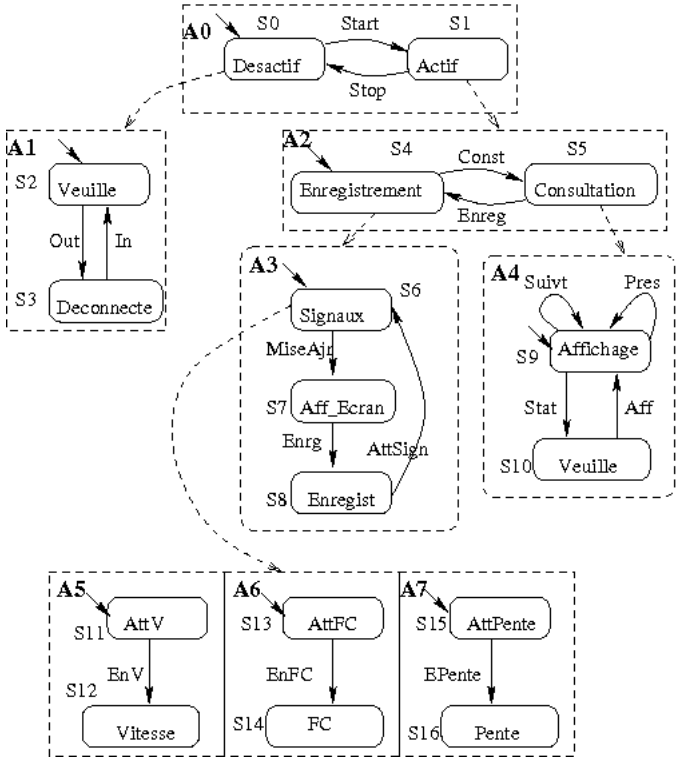

Fig. 1. Exemple de AH

(vitesse, fréquence cardiaque,...) venant d'une ceinture émettrice destinée aux cyclistes.

Ce contrôle est muni d'une unité de calcul, un écran, des boutons et une mémoire pour stocker les données. Le cycliste peut éventuellement consulter les données enregistrées.

Ce système est conçu principalement pour enregistrer la fréquence cardiaque, la vitesse et la mesure de la pente. L'objectif est d'améliorer les performances des cyclistes en analysant ces données.

Au départ, le contrôleur est désactivé, dans cet état, il peut être soit en veuille, soit en maintenance. Dans l'état actif, l'athlète peut choisir soit le mode enregistrement des signaux, soit le mode consultation. Chaque attribut enregistré est composé de (vitesse, fréquence cardiaque et la mesure de pente). Dans le mode consultation, le cycliste peut choisir soit les statistiques, soit affichage des attributs.

La figure 1 représente un modèle simplifié, décrit par un automate hiérarchique.

L'automate hiérarchique $AH = \langle F, E, \gamma \rangle$ est composé de huit automates séquentiels $F = \{A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$ qui sont liés entre eux avec la fonction de composition $\gamma = \{s_0 \longrightarrow \{A_1\} \wedge s_1 \longrightarrow \{A_2\} \wedge s_4 \longrightarrow \{A_3\} \wedge s_5 \longrightarrow \{A_4\} \wedge s_6 \longrightarrow \{A_5, A_6, A_7\} \cup \{s \longrightarrow \emptyset | s \in \{s_2, s_3, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}, s_{15}, s_{16}\}\}\}$. Nous avons $\chi(s_0) = \{s_2, s_3\}$.

II. Sémantique des automates hiérarchiques

Dans cette section, nous décrivons la sémantique de l'automate hiérarchique définie comme une structure de Kripke.

Cette présentation nous permet de vérifier les systèmes hiérarchiques à nombre fini d'états par model-checking.

Une structure de Kripke est définie par un ensemble de configurations reliées par une relation de transition. Nous commençons par définir la notion de configuration, étiquetage des transitions dans AH et transition activable.

A. Configuration

Une configuration d'un automate hiérarchique permet de décrire l'état global de celui-ci à un instant précis.

Définition 3: (configuration) Soit $S_t = \cup_{A \in F} S_A$ l'ensemble des états de $AH = \langle F, E, \gamma \rangle$ et soit C un ensemble d'états dans S_t. C est une configuration si :

- l'automate racine participe par un seul état à la configuration C : $\exists!s.(s \in S_{A_{racine}} \wedge s \in C)$ et
- la fermeture en bas : pour chaque état s dans C si s est raffiné par un automate A, alors, A participe aussi par un seul état à C : $\forall s, A.(s \in C \wedge A \in \gamma(s) \Rightarrow \exists!s'.(s' \in S_A \wedge s' \in C))$

Dans l'automate hiérarchique de la figure 1, les ensembles (s_0, s_2), (s_0, s_3), (s_1, s_5, s_9), $(s_1, s_4, s_6, s_{11}, s_{13}, s_{15})$, (s_1, s_4, s_7) et (s_1, s_4, s_8) sont des configurations. (s_0, s_2) est la configuration initiale.

On note $Conf_A$ l'ensemble des configurations de l'automate hiérarchique AH_A où A est l'automate racine de AH_A.

B. L'étiquetage des transitions dans un automate hiérarchique

Pour représenter les diagrammes StateCharts à l'aide des automates hiérarchiques, *Mikk et al* [4] ont ajouté deux gardes, *sr(source restriction)* et *td(target determinator)* aux transitions d'automates séquentiels. L'objectif est de préserver les informations des transitions qui ont plusieurs états de départ et plusieurs états d'arrivée ($Interlevel\ Transitions$).

Soit $AH = \langle F, E, \gamma \rangle$ un automate hiérarchique. L'étiquetage d'une transition $t \stackrel{def}{=} s \longrightarrow s'$ dans un automate $A \in F$ est défini par le triplet $(sr,\ a,\ td)$ tel que : $source(t) = s$ est la source de la transition, $but(t) = s'$ est l'état d'arrivée de la transition t, $action(t) = a$ est l'action de t, sr est employée pour déterminer dans quelles configurations t est activable, et td est employée pour déterminer les états d'arrivée de la transition t.

Dans la figure 1, les transitions $s_0 \stackrel{Start}{\longrightarrow} s_1$, $s_4 \stackrel{Const}{\longrightarrow} s_5$, $s_6 \stackrel{MiseAjr}{\longrightarrow} s_7$ sont définies comme suit :

- $s_0 \stackrel{Start}{\longrightarrow} s_1 \stackrel{def}{=} s_0 \stackrel{\{s_2\},Start,\emptyset}{\longrightarrow} s_1$;
- $s_4 \stackrel{Const}{\longrightarrow} s_5 \stackrel{def}{=} s_4 \stackrel{\{s_8\},Const,\emptyset}{\longrightarrow} s_5$;
- $s_6 \stackrel{MiseAjr}{\longrightarrow} s_7 \stackrel{def}{=} s_6 \stackrel{\{(s_12,s_14,s_16)\},MiseAjr,\emptyset}{\longrightarrow} s_7$.

Soient s et s' deux états d'un automate $A \in F$ et $t \stackrel{def}{=} s \stackrel{sr,a,td}{\longrightarrow} s'$ une transition dans A. Si s est raffiné par un ou plusieurs automates et $sr = \emptyset$, alors, t est activable de n'importe quel état *des automates fils* de s. Si s' est raffiné par un ou plusieurs automates et $td = \emptyset$, alors, les états d'arrivée de t sont les états initiaux des *automates fils* de s'.

Dans l'exemple, les transitions $s_1 \stackrel{Stopt}{\longrightarrow} s_0$ et $s_5 \stackrel{Enreg}{\longrightarrow} s_4$ sont activables à partire de tout les états des automates fils.

1) Ensemble de transitions sans conflit: Soit $AH = \langle F, E, \gamma \rangle$ un automate hiérarchique, A un automate dans F, C une configuration dans $Conf_A$ et a une action dans E. On dit que la transition $t \stackrel{def}{=} s \stackrel{sr,\,a,\,td}{\longrightarrow} s'$ de l'automate A est activable à partir de la configuration C, notée $activable_{(C|t)}$, si l'état s est dans la configuration C et C est une configuration dans sr ou $sr = \emptyset$.

Soient t_1 et t_2 deux transitions activables à partir de $C|e$. On dit que t_2 est prioritaire par rapport à t_1 si : $(source(t_1) \in \chi^+(source(t_2)))$.

Définition 4: (ensemble de transitions activables sans conflit) Soit $ET_{(C|e)}$ l'ensemble des transitions étiquetées par l'ensemble des actions $e \subseteq E$ activables à partir de la configuration C. On appelle $trs \subseteq ET_{(C|e)}$ l'ensemble maximal des transitions sans conflit activables à partir de $C|e$ si : (1) $\forall A \in F.(|trs \cap \longrightarrow_A\ | \leq 1)$ (chaque automate participe au plus avec une seule transition) (2) $\forall t \in ET_{(C|e)}.(t \in trs)$ s'il n'y a pas de transition $t' \in ET_{(C|e)}$ avec une priorité élevée.

Dans la figure 1, si la configuration actuelle est $\{s_1,\ s_5,\ s_9\}$ et l'ensemble des actions à exécuter est $\{Stop, Enreg, Stat\}$), alors l'ensemble des transitions activables est $\{(s_1 \stackrel{Stop}{\longrightarrow} s_0), (s_5 \stackrel{Enreg}{\longrightarrow} s_4), (s_9 \stackrel{Stat}{\longrightarrow} s_{10})\}$ et l'ensemble maximal des transitions sans conflit est $\{(s_1 \stackrel{Stop}{\longrightarrow} s_0)\}$.

2) Structure de Kripke:

Définition 5: (structure de Kripke) La sémantique de AH est une structure de Kripke $SK = \langle Conf, C_0, \longrightarrow_K, E, LK \rangle$ où :

- $Conf$ est l'ensemble des configurations de AH,
- C_0 est la configuration initiale,
- E est l'ensemble d'actions,
- $LK : Conf \longrightarrow 2^{AP}$ telle que $LK(C) = \cup_{s_i \in C} L(s_i)$,
- $\longrightarrow_K \subseteq Conf \times 2^E \times Conf$ est la relation des transitions de SK.

Une transition $t \in \longrightarrow_K$ est un ensemble de transitions sans conflit définie par les règles suivantes :

- **Règle de progrès :** Cette règle s'applique à un automate A si un de ses états s est dans la configuration C et si l'une des transitions partante de s est activables alors une de ces transitions a lieu.

$$\{s\} = C \cap S_A$$

$$\frac{\exists t \in \longrightarrow_A .(activable_{(C|e)}(t)) \wedge t = (s \stackrel{sr,a,td}{\longrightarrow} s')}{A :: C \stackrel{e}{\longrightarrow} (\{s'\} \cup td)}$$

- **Règle de composition :** Cette règle explique comment un automate délègue sa transition à ses automates fils : elle s'applique à un automate A ayant un état s dans la configuration C tel que toutes les transitions partantes de s ne sont pas activables et l'état s est raffiné par un ou plusieurs automates. *Les automates fils* peuvent exécuter leurs transitions.

$$\{s\} = C \cap S_A$$
$$\forall t.(t \in \longrightarrow_A .(t = (s \stackrel{sr,a,td}{\longrightarrow} s') \Rightarrow \neg activable_{(C|e)}(t)))$$
$$\gamma(s) = \{A_1, \ldots A_m\} \neq \emptyset$$
$$A_1 :: C \stackrel{e}{\longrightarrow} C'_1$$

$$\frac{A_m :: C \stackrel{e}{\longrightarrow} C'_m}{A :: C \stackrel{e}{\longrightarrow} \{\{s\} \cup C'_1 \cup \cdots \cup C'_m\}}$$

– **Règle de bégaiement :** Cette règle est appliquée à un automate A ayant un état s dans la configuration C mais toutes les transitions partantes de s ne sont pas activables et s n'est pas raffiné par un autre automate.

$$\{s\} = C \cap S_A$$

$$Basic(s)$$

$$\frac{\forall t \in \longrightarrow_A .(t = (s \overset{sr,a,td}{\longrightarrow} s')) \Rightarrow \neg activable_{(C|e)}(t)}{A :: C \overset{e}{\longrightarrow} \{s\}}$$

Une exécution de SK est une séquence finie ou infinie de configurations et de noms d'actions de la forme : $C_0 \overset{e_1}{\longrightarrow} C_1 \overset{e_2}{\longrightarrow} \cdots \overset{e_i}{\longrightarrow} C_i \overset{e_{i+1}}{\longrightarrow} \ldots$. La séquence de configurations $\sigma = C_0, C_1, \ldots C_i \ldots$ définie un chemin d'exécution de SK. On note $\Sigma(SK)$ l'ensemble des chemin d'exécution de SK.

La structure de Kripke SK associée à l'automate hiérarchique AH de la figure1 est représentée dans la figure 2.

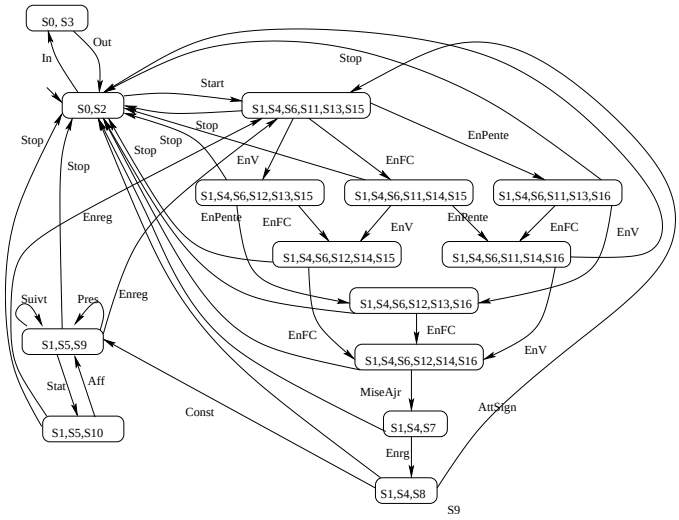

Fig. 2. SK associée à AH

Pour faciliter la lisibilité de SK nous avons dessiné qu'une partie des transitions définies par les règles de progrès et de composition, aussi nous avons donné que les noms des états des configurations.

III. Vérification de propriétés

Dans ce papier, nous nous intéressons à la vérification de propriétés exprimées en LTL sur les systèmes hiérarchiques. Pour faire face au problème d'explosion combinatoire dû à la complexité de ces systèmes, nous présentons une approche de vérification par modules pour éviter l'aplatissement de tout l'automate hiérarchique. Cette méthode est utilisée pour vérifier les systèmes de grande taille. Il consiste à découper le modèle de système en un ensemble de modules et de vérifier si une propriété est satisfaite sur chaque module. La vérification d'une propriété sur chaque module se fait d'une manière séparée et indépendante.

Le choix des propriétés à vérifier et la méthode de découpage sont deux points très importants, parce que ce n'est pas toujours vrai que : si une propriété φ est vérifiée sur tous les modules alors elle l'est sur le modèle global du système. Cela découle de la manière de découper le modèle global du système en modules.

Par nature, les automates hiérarchiques sont définis par des partitions des espaces d'états du système. C'est à dire, chaque état peut être décomposé en un un ensemble d'automates, cette décomposition définit un module. Nous appelons cette partie de système une sous structure de Kripke, notée SSK. Chaque sous structure de Kripke représente un aplatissement des automates fils associés à un état décomposé.

Un certain nombre de propriétés LTL ont pour particularité d'être satisfaites globalement lorsqu'elles sont satisfaites sur tous les modules du système. Ainsi, il faut s'assurer qu'une propriété est vérifiable modulairement avant de la vérifier de manière modulaire.

La vérification d'une propriété φ sera donc effectuée sur les sous structures de Kripke. Si les SSK vérifient la propriété φ alors la structure de Kripke globale, associée à l'automate hiérarchique, la vérifie également. Cette méthode est appelée la vérification modulaire ou par parties. Formellement, pour tout état s raffiné, si $SSK_s \models \varphi$ alors $SK \models \varphi$.

Avant de définir les schémas de propriétés qui peuvent être vérifiées d'une manière modulaire, nous commençons par définir l'ensemble des sous structures de Kripke associées à un automate hiérarchique.

A. L'ensemble des modules de AH

Soient $AH = \langle F, E, \gamma \rangle$ un automate hiérarchique. Soit $SK = \langle Conf, C_0, \longrightarrow_K, E, LK \rangle$ la structure de Kripke associée à AH. Soit S_r l'ensemble des états raffinés (c'est à dire : un état $s \in S_r$ si $\gamma(s) \neq \emptyset$). Nous associons à chaque état s de S_r une sous structure de Kripke, notée SSK_s, de la structure de Kripke globale SK.

La sous structure de Kripke SSK_s représente l'aplatissement des automates fils de l'état $s \in S_r$ et les transitions sortantes de s appartenant aux automates ancêtres de s.

Définition 6: (Sous strucure de Kripke) Soit s un état de S_r, la sous structure de Kripke $SSK_s = \langle Conf_s, C_{0s}, \longrightarrow_{Ks}, E, LK \rangle$ associée à l'état s est définie comme suit :

- $Conf_s$ est un sous ensemble de configurations de SK, telle que : C est une configuration dans $Conf_s$ si C est une configuration des automates fils de s ou C est une configuration cible d'une transition sortante de s ;
- C_{0s} est la configuration initiale tels que tous les états initiaux des automates fils de s appartiennent à C_{0s} ;
- E_2 est l'ensemble d'actions ;
- $LK_2 : Conf_s \longrightarrow 2^{APv}$ est la fonction de décor des configurations ;
- $\longrightarrow_{Ks} \subseteq \longrightarrow_{K2}$ est la relation de transitions de SSK_s.

$\longrightarrow_{Ks}$ est un sous ensemble de la relation $\longrightarrow_{K2}$. $t \overset{def}{=} C_i \overset{a}{\longrightarrow} C_j$ est une transition dans $\longrightarrow_{Ks}$ si et seulement si : $t \in \longrightarrow_{K2}$, $C_i \in Conf_s$ et $C_j \in Conf_s$.

Les sous structures de Kripke SSK_{s_1} et SSK_{s_4} associées aux états s_1 et s_4 de l'automate hiérarchique AH de la figure 1 sont représentées dans la figure 3.

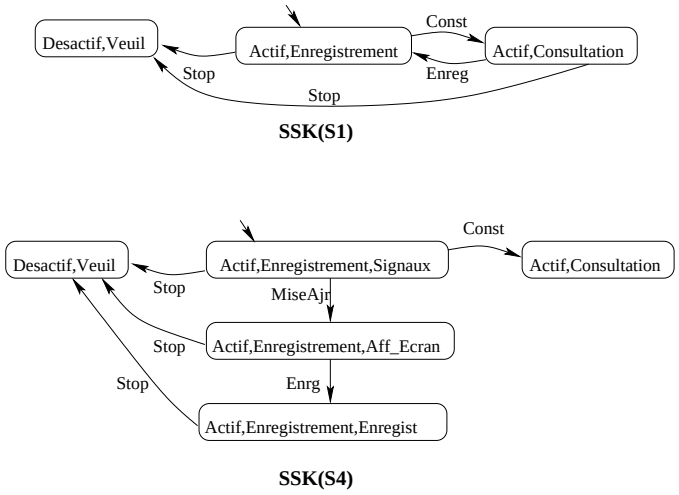

Fig. 3. Les sous structures de Kripke SSK_{s_1} et SSK_{s_4}

La sous structure de Kripke SSK_{s_1} se compose de trois configurations et quatre transitions. Les transitions étiquetées par l'actions *Stop* sont des transitions de l'automate ancêtre A_0, sortantes de l'état s_1.

Définition 7: (L'ensemble des modules de AH) Soient $AH = \langle F, E, \gamma \rangle$ un automate hiérarchique. A_{racine} l'automate racine de AH. Soit S_r l'ensemble des états raffinés. Nous appellons $\mathcal{M}$ l'ensemble des modules de AH si :

- l'automate racine appartient à $\mathcal{M}$,
- pour chaque état s de S_r il existe une sous structure de Kripke SSK_s associée à s dans $\mathcal{M}$.

L'ensembe des modules de l'automate hiérarchique de la figure 1 est défini par six sous structure de Kripke : SSK_{racine}, SSK_{s_0}, SSK_{s_1}, SSK_{s_4}, SSK_{s_5} et SSK_{s_6}. SSK_{s_1} et SSK_{s_4} sont représentées dans la figure 3, les autres sous structure de Kripke sont représentées dans la figure 4.

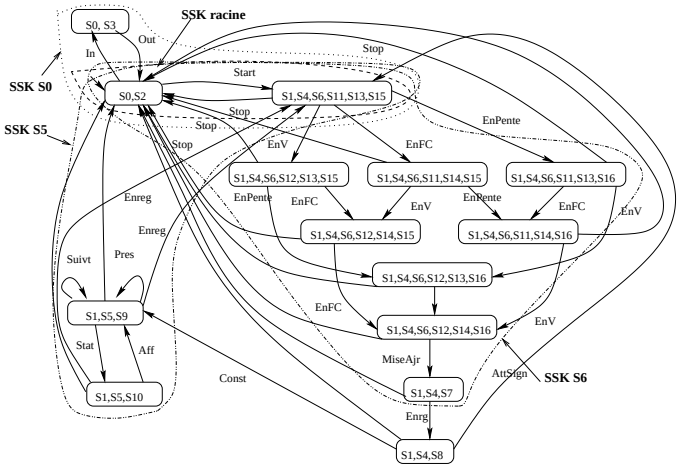

Fig. 4. Les sous structures de Kripke de l'automate hiérarchique de la figure 1

B. Schémas de propriétés

Nous proposons dans cette section quatre patterns de propriétés qui peuvent être vérifiées d'une manière modulaire. Nous commençons par rappeler la définition de la logique LTL.

1) Logique temporelle linéaire: Soit p une proposition atomique de l'ensemble AP. Une formule temporelle φ de LTL [2] est définie par la grammaire suivante :

$$\varphi ::= p | \neg\varphi | \varphi \wedge \varphi | \bigcirc \varphi | \varphi \mathcal{U} \varphi$$

Soient φ, φ_1, φ_2 des formules temporelles. Soit $\sigma = C_0, C_1, \ldots C_i, \ldots$ un chemin d'exécution d'une structure de Kripke SK. Nous définissons que "φ est satisfaite à la configuration $C_{i_{i\geq 0}}$ de l'exécution σ", noté $\sigma(i) \models \varphi$, comme suit :

- $\sigma(i) \models p$ si et seulement si $p \in LK(C_i)$,
- $\sigma(i) \models \neg\varphi$ si et seulement si il n'est pas vrai que $\sigma(i) \models \varphi$,
- $\sigma(i) \models \varphi_1 \wedge \varphi_2$ si et seulement si $\sigma(i) \models \varphi_1 \wedge \sigma(i) \models \varphi_1$,
- $\sigma(i) \models \bigcirc\varphi$ si et seulement si $\sigma(i+1) \models \varphi$,
- $\sigma(i) \models \varphi_1 \mathcal{U} \varphi_2$ si et seulement si $\exists k.(k \geq i \wedge \sigma(k) \models \varphi_2 \wedge \forall j.(i \leq j < k \Rightarrow \sigma(j) \models \varphi_1))$.

Les autres opérateurs temporels se redéfinissent comme suit : $\Diamond\varphi$ peut être redéfini par $true \mathcal{U} \varphi$, $\Box\varphi$ peut être redéfini par $\neg\Diamond\neg\varphi$, $\varphi_1 \Rightarrow \varphi_2$ redéfini par $\neg\varphi_1 \vee \varphi_2$.

2) Schémas de propriétés: Soient $AH = \langle F, E, \gamma \rangle$ un automate hiérarchique, SK la structure de Kripke associée à AH, S_r l'ensemble des états raffinés et $\mathcal{M}$ l'ensembe des modules de AH. Soient s un état de S_r, p et q deux propositions exprimées sur AP.

Dans le théorème suivant, nous énonçons des schémas de propriétés vérifiables modulairement.

Théorème 3.1: Soit φ une formule LTL de type $\Box p$, $\Box(p \Rightarrow \Diamond q)$ et $\Box(p \Rightarrow p \mathcal{U}\, q)$. Si toutes les sous structures de Kripke SSK de $\mathcal{M}$ satisfont φ alors SK satisfait φ.

Preuve du théorème 3.1 :

Cas 1 :
Pour toute $SSK \in \mathcal{M}$, si $SSK \models \Box p$ alors $SK \models \Box p$.

SK est définie par l'ensemble des configurations de SSK, c'est à dire, dans SK il n'y a pas une configuration C qui n'appartient pas à une sous structure de Kripke de $\mathcal{M}$.
Nous avons $SSK \models \Box p$, d'après la sémantique de la LTL toutes les configurations de SSK satisfont p.
Comme SK est définie par l'ensemble des configurations de SSK alors $SK \models \Box p$.

Cas 2 :
Pour toute $SSK \in \mathcal{M}$, si $SSK \models \Box(p \Rightarrow \Diamond q)$ alors $SK \models \Box(p \Rightarrow \Diamond q)$.

Nous avons $SSK \models \Box(p \Rightarrow \Diamond q)$, alors $\forall \sigma \in \Sigma(SSK)$ et

[2]Pour plus de détails sur la logique LTL, voir [10], [12]

$\forall i.(i \geq 0 \wedge \sigma(i) \models p \wedge \sigma(j)_{j>i} \models \Diamond q)$.
Pour chaque SSK_s, toutes les transitions sortantes de l'état s (transitions prioritaires) sont dans SSK_s, donc chaque chemin d'exécution de SSK_s est un morceau de chemin de SK, alors les configurations satisfont q seront dans le chemin d'exécution de SK.
Donc, si chaque SSK vérifie $\Box(p \Rightarrow \Diamond q)$, alors SK la vérifie également.

Cas 3 :
Pour toute $SSK \in \mathcal{M}$, si $SSK \models \Box(p \Rightarrow p\,\mathcal{U}\,q)$ alors $SK \models \Box(p \Rightarrow p\,\mathcal{U}\,q)$.

$SSK \models \Box(p \Rightarrow p\,\mathcal{U}\,q)$ alors $\forall\sigma.(\sigma \in \Sigma(SSK) \Rightarrow \sigma \models \Box(p \Rightarrow p\,\mathcal{U}\,q))$.
Par définition de la LTL $\forall i.(i \geq 0 \wedge \sigma(i) \models p)$ et $\exists k.(k \geq i \wedge \sigma(k) \models q \wedge \forall j.(i \leq j < k \Rightarrow \sigma(j) \models p))$.
Pour chaque SSK_s, toutes les transitions sortantes de l'état s sont dans SSK_s, donc chaque chemin d'exécution de SSK_s est un morceau de chemin SK.
D'après la proposition 1, les états fils des états raffinés dans $\sigma(i)$ vérifiont aussi $\Box p$. Donc, si les SSK satisfont $(p \Rightarrow p\,\mathcal{U}\,q)$, alors nous concluons que la propriété est vérifiée sur SK.

Nous donnons quelques exemples de propriétés vérifiables modulairement et par le système global. Dans le cas 2, *"Au départ, le contrôleur est inactif, il devient actif plus tard"*. Dans le cas 3, *"Le contrôleur continue à enregistrer les performances d'un sportif pour qu'elles soient consultées ultérieurement"*.

IV. Conclusion et perspectives

Dans ce papier nous nous sommes intéressés à la vérification de systèmes hiérarchiques, qui proviennent du domaine médical, en l'occurence un contrôleur de performances sportives. Pour faire face au problème d'explosion combinatoire dû à la complexité de ces systèmes, nous présentons une approche de vérification par modules (sous-structures) pour éviter l'aplatissement de tout l'automate hiérarchique. Cette méthode est utilisée pour vérifier les systèmes de grande taille. Il consiste à découper le modèle de système en un ensemble de modules et de vérifier si une propriété est satisfaite sur chaque module. La vérification d'une propriété sur chaque module se fait d'une manière séparée et indépendante.

Nous travaillons actuellement pour étendre ces schémas de propriétés car notre objectif est d'exploiter la notion d'hiérarchie pour utiliser le moins possible la technique du model-checking ou éventuellement l'utiliser sur des sous modèles de taille raisonnable.

References

[1] N. Leveson and C. S. Turner., "An investigation of the therac-25 accidents," *IEEE Computer*, vol. 25, No. 7, pp. 13–17, 1993.

[2] D. Harel, "Statecharts : A Visual Formalism for Complex Systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, June 1987.

[3] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language user guide*. Redwood City, CA, USA : Addison Wesley Longman Publishing Co., Inc., 1999.

[4] E. Mikk, Y. Lakhnech, and M. Siegel, "Hierarchical Automata as Model for Statecharts," in *ASIAN '97 : Proceedings of the Third Asian Computing Science Conference on Advances in Computing Science*. London, UK : Springer-Verlag, 1997, pp. 181–196.

[5] D. Latella, I. Majzik, and M. Massink, "Automatic verification of a behavioural subset of uml statechart diagrams using the spin model-checker." *Formal Asp. Comput.*, vol. 11, no. 6, pp. 637–664, 1999.

[6] M. Al'Achhab, A. Hammad, and H. Mountassir, "Vérification de systèmes hiérarchiques par raffinement," in *Journal Européen des Systèmes Automatisés : Modélisation des Systèmes Réactifs (MSR'05)*, vol. 39/1-3. Grenoble, France : Hermès Science, oct 2005, pp. 239–254.

[7] P. Bhaduri and S. Ramesh, "Model checking of statechart models : Survey and research directions," *CoRR*, vol. cs.SE/0407038, 2004.

[8] E. Mikk, Y. Lakhnech, M. Siegel, and G. J. Holzmann, "Implementing Statecharts in Promela/SPIN," in *Proceedings of the 2nd IEEE Workshop on Industrial-Strength Formal Specification Techniques*. IEEE Computer Society, October 1998, pp. 90–101.

[9] G. J. Holzmann, "The model checker SPIN," *Software Engineering*, vol. 23, no. 5, pp. 279–295, 1997.

[10] Z. Manna and A. Pnueli, *The temporal logic of reactive and concurrent systems*. New York, NY, USA : Springer-Verlag New York, Inc., 1992.

[11] P.-A. Masson, H. Mountassir, and J. Julliand, "Modular verification for a class of PLTL properties," in *2nd Int. Conf. on Integrated Formal Methods, IFM'2000*, ser. LNCS, vol. 1945. Dagstuhl, Saarland, Germany : Springer-Verlag, Nov. 2000, pp. 398–419.

[12] A. Pnueli, "The temporal semantics of concurrent programs," in *Proceedings of the International Symposium on Semantics of Concurrent Computation*. London, UK : Springer-Verlag, 1979, pp. 1–20.