Exploitation du Raffinement pour Vérifier les Modèles Hiérarchiques

Mohammed Al Achhab
Laboratoire d'Informatique
de l'Université de Franche-Comté
16, route de Gray 25030 Besançon Cedex
alachhab@lifc.univ-fcomte.fr

Ahmed Hammad, Hassan Mountassir
Laboratoire d'Informatique
de l'Université de Franche-Comté
16, route de Gray 25030 Besançon Cedex
{hammad, mountass}@lifc.univ-fcomte.fr

Résumé

Cet article traite de la spécification et de la vérification par model-checking des systèmes hiérarchiques réactifs. Le modèle utilisé est celui des automates hiérarchiques dans lesquels nous exploitons la décomposition d'un état en un ensemble d'automates. Pour pallier au problème de l'explosion combinatoire du nombre d'états induit par le model-checking, nous proposons d'utiliser la technique du raffinement. La contribution de ce papier est de définir les conditions de raffinement entre automates hiérarchiques avec cycles.

1. Motivations et problématique

Ces dernières années, la vérification des systèmes informatiques critiques est devenue un sujet de recherche important en raison du développement croissant de logiciels appliqués à la médecine, aux moyens de transports ou aux centrales nucléaires. Dans ces domaines, une erreur de programmation peut coûter très cher financièrement ou en vies humaines. Il convient donc, lors du développement de telles applications, de s'assurer qu'elles satisfont un certain nombre de *propriétés*, notamment des propriétés de sûreté (comme l'absence de défaillance grave).

Notre travail s'articule autour de la spécification et la vérification de systèmes hiérarchique réactifs. Grâce à la popularité grandissante de *StateCharts* [6] et d'*Unified Modeling Language (UML)* [5], la modélisation hiérarchique devient quasiment incontournable pour les développements logiciels industriels. Cette modélisation possède, en outre, des concepts puissants 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.

Dans [10], les auteurs ont proposé une sémantique opérationnelle de StateCharts, ce sont *les automates hiérarchiques* qui couvrent le raffinement d'états, la priorité entre les transitions et l'exécution simultanée de transitions. Dans [11], les auteurs montrent comment les StateCharts peuvent être traduits en *Promela* (le langage de programmation de SPIN [7]) en utilisant les automates hiérarchiques comme format intermédiaire. Cette méthode est également employée pour la vérification des diagrammes d'états transitions d'UML [8]. Les techniques de vérification utilisées sont algorithmiques, en particulier le *model-checking*. Les propriétés sont exprimées à l'aide d'une logique comme la LTL (Linear Temporal Logic) [9]. Le principal problème réside dans la complexité des procédures de décision lors de la vérification de propriétés.

Nous proposons le raffinement d'automates hiérarchiques comme solution pour vérifier les systèmes hiérarchiques de grande taille. L'idée est d'introduire pas à pas des détails supplémentaires sur le système en éclatant *les états de base*[1] du système abstrait. Cette approche permet de décrire des spécifications abstraites de plus petite taille sur lesquelles on vérifie les propriétés exprimables au niveau des détails considérés. Puis le système est raffiné par introduction de nouveaux détails sous forme de composants qui peuvent être des automates ou un ensemble d'automates parallèles. Cette démarche ne résout le problème qu'à condition que l'opération de raffinement préserve sur le système raffiné les propriétés vérifiées sur le système abstrait. La notion de raffinement de systèmes de transitions, définies dans [4], issue de celle de raffinement de systèmes d'événements [1], préserve les propriétés de logique temporelle linéaire. Donc, si une propriété LTL est vérifiée au niveau abstrait alors elle l'est également au niveau raffiné.

Dans [3], nous avons défini une relation de raffinement entre automates hiérarchiques, limités à des nouveaux automates acycliques. Ce raffinement permet de préserver les propriétés LTL du niveau abstrait. L'hypothèse de l'absence de cycles dans les nouveaux automates est très forte. La contribution de ce travail est de définir une relation de raffinement entre automates hiérarchiques avec cycles.

La suite de ce papier est organisée en trois sections. Dans la section deux, nous présentons les concepts de base : automate hiérarchique et structure de Kripke. Dans la section trois, nous définissons la relation de raffinement

[1] Nous appelons état de base l'état qui n'a pas une structure interne.

entre automates hiérarchiques et dans la section quatre, nous concluons et nous traçons quelques perspectives à ce travail.

2. Préliminaires

2.1. Automate séquentiel

Soit V un ensemble de variables d'états, chacune de domaine fini. Soit AP_V l'ensemble des propositions atomiques de la forme $v = d$, $v \in V$ et $d \in domaine(v)$.

Définition 1 *Un automate A est un 5-uplet $A = \langle S, s_0, \Sigma, \rightarrow, L\rangle$ où : S est un ensemble fini d'états, $s_0 \in S$ est l'état initial, Σ est un alphabet fini de noms d'actions, $\rightarrow \subseteq S \times \Sigma \times S$ est l'ensemble des transitions, $L : S \rightarrow 2^{AP_V}$ est l'interprétation des états de S sur V.*

Une exécution σ de A est une séquence finie ou infinie des états et des actions $s_0 \stackrel{a_0}{\rightarrow} \cdots \stackrel{a_i}{\rightarrow} s_i \stackrel{a_i}{\rightarrow} \ldots$ telle que s_0 est l'état initial et pour chaque $i \geq 0$, nous avons $s_i \stackrel{a_i}{\rightarrow} s_{i+1} \in \rightarrow$. On note $Exec(A)$ l'ensemble des exécutions de l'automate A.

Soit $\sigma \stackrel{def}{=} s_0 \stackrel{a_1}{\rightarrow} \cdots \stackrel{a_i}{\rightarrow} s_i \stackrel{a_{i+1}}{\rightarrow} \ldots$ une exécution de A. Nous appelons trace de σ, $tr(\sigma) \stackrel{def}{=} a_0 \ldots a_i \ldots$, la séquence d'actions, telle que $tr(\sigma) \in \Sigma^\omega$.

On appelle *cycle d'un automate* A une séquence finie d'états et d'actions $cy \stackrel{def}{=} s_0 \stackrel{a_0}{\rightarrow} \cdots \stackrel{a_{n-1}}{\rightarrow} s_n$, telle que $s_n = s_0$, et pour chaque $0 \leq i < n$ nous avons $s_i \stackrel{a_i}{\rightarrow} s_{i+1} \in \rightarrow$.

2.2. Automate hiérarchique

Un automate hiérarchique, notée 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 et un ensemble d'automates.

Définition 2 *AH est un trip-let $\langle F, E, \gamma\rangle$, où : $F = \{A_1, \ldots A_n\}$ est un ensemble d'automates ayant un ensemble d'états distincts, E est un alphabet fini de noms d'actions, $\gamma : \bigcup_{A \in F} S_A \rightarrow 2^F$ est une fonction de composition sur F telle que : (1) dans l'ensemble F il y a un seul automate racine A_{racine}, (2) chaque automate $A \in F \backslash \{A_{racine}\}$ a un état père et (3) la fonction de composition ne doit pas contenir de cycle.*

On considére l'exemple de TV-Set décrit dans la figure 1. Au premier niveau d'observation le TV-Set peut être dans deux états : $EnService$ et $EnAttente$ (état initial). L'état $EnAttente$, qui est raffiné par l'automate $Power$, peut être aussi dans deux états : $Veille$ (état initial) et $Deconnecte$. L'automate hiérarchique, $AH_1 = \langle F_1, E_1, \gamma_1\rangle$, qui modélise le fonctionnement de TV-Set, est composé de deux automates $F_1 = \{TV, Power\}$ qui sont liés avec la fonction de composition $\gamma_1 = \{s_0 \rightarrow \{Power\}\} \cup \{s \rightarrow \emptyset | s \in \{s_1, s_2, s_3\}\}$.

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 et χ^{-1} est *la fonction d'ancêtre* où : $\chi^{-1}(s) = s'$ si $s \in \chi(s')$.

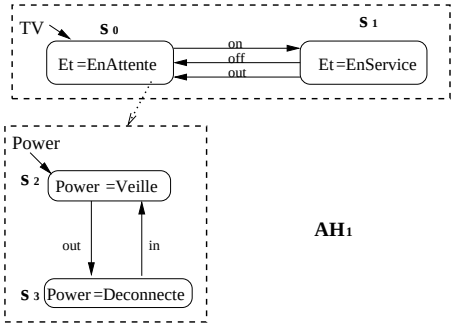

FIG. **1. TV-Set au niveau abstrait**

Soit S_t l'ensemble des états de l'automate hiérarchique AH. On note $\gamma^{-*} : S_t \rightarrow F$ l'application qui constitue le lien entre un état s et son ancêtre A.

Dans l'automate hiérarchique AH_1 de la figure 1 : $\chi(s_0) = \{s_2, s_3\}$, $\chi^{-1}(s_3) = \{s_0\}$ et $\gamma^{-*}(s_3) = \{TV\}$.

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).

Définition 3 *Soit AH un automate hiérarchique et soit S_t l'ensemble des états de AH. $C \subseteq S_t$ est une configuration[2] de AH ssi : (1) l'automate racine participe par un seul état à la configuration et (2) 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.*

L'étiquetage d'une transition $t = s \rightarrow s'$ 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 t.

Les transitions de l'automate hiérarchique AH_1 sont définies comme suit : $(s_0 \stackrel{\{s_2\},on,\emptyset}{\rightarrow} s_1)$, $(s_1 \stackrel{\emptyset,off,\{s_2\}}{\rightarrow} s_0)$ et $(s_1 \stackrel{\emptyset,out,\{s_3\}}{\rightarrow} s_0)$.

Remarque 2.1 *(Priorité entre les transitions) Nous commençons par exécuter les transitions de l'automate racine, puis les transitions des sous-automates sont effectuées si aucune transition n'est activable à partir de l'automate père.*

[2]Une configuration d'un automate hiérarchique permet de décrit l'état global de système hiérarchique à un instant précis.

2.3. Structure de Kripke

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 par le model-checking des systèmes réactifs à nombre d'états fini.

Soient $AH = \langle F, E, \gamma \rangle$ un automate hiérarchique, A un automate dans F, C une configuration de AH et e un ensemble d'actions dans E. On dit que la transition $s \overset{sr,a,td}{\rightarrow} s'$ de l'automate A est activable à partir de la configuration C et sur l'ensemble e si $C \cup \{s\} \in sr$ et $a \in e$.

Définition 4 *La sémantique de AH est une structure de Kripke $SK = \langle Conf, C_0, \rightarrow_K, E, LK \rangle$ où : $Conf$ est l'ensemble des configurations de AH, C_0 est la configuration initiale, E est l'ensemble d'actions, $LK : Conf \rightarrow 2^{AP_V}$ tel que $LK(C) = \cup_{s_i \in C} L(s_i)$ et $\rightarrow_K \subseteq Conf \times 2^E \times Conf$ est la relation des transitions de SK.*

Soit $t \overset{def}{=} C \overset{e}{\rightarrow} C'$ une transition, $t \in \rightarrow_K$ ssi elle est définie par une des trois règles suivantes :

- ***Règle de progrès :***

$$\frac{\{s\} = C \cap S_A \quad \exists t \in \rightarrow_A .(activable_{(C|e)}(t)) \wedge t = (s \overset{sr,a,td}{\rightarrow} s')}{A :: C \overset{e}{\rightarrow} (\{s'\} \cup td)}$$

- ***Règle de composition :***

$$\frac{\begin{array}{c}\{s\} = C \cap S_A \\ \forall t.(t \in \rightarrow_A .(t = (s \overset{sr,a,td}{\rightarrow} s') \Rightarrow \neg activable_{(C|e)}(t))) \\ \gamma(s) = \{A_1, \ldots A_m\} \neq \emptyset \\ A_1 :: C \overset{e}{\rightarrow} C'_1 \\ A_m :: C \overset{e}{\rightarrow} C'_m\end{array}}{A :: C \overset{e}{\rightarrow} \{\{s\} \cup C'_1 \cup \cdots \cup C'_m\}}$$

- ***Règle de bégaiement :***

$$\frac{\{s\} = C \cap S_A \; Basic(s) \quad \forall t \in \rightarrow_A .(t = (s \overset{sr,a,td}{\rightarrow} s')) \Rightarrow \neg activable_{(C|e)}(t)}{A :: C \overset{e}{\rightarrow} \{s\}}$$

La structure de Kripke associée à AH_1 de la figure 1 est représentée dans la figure 2.

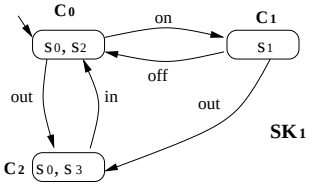

FIG. **2.** SK_1 **associe** AH_1

3. Résultat : raffinement entre automates hiérarchiques

Dans cette section, nous définissons les conditions de raffinement entre automates hiérarchiques. Le raffinement consiste à éclater les états de base du système abstrait. Ces états sont remplacés par un ou plusieurs automates. Ces automates sont notés A_τ. Les transitions de A_τ sont désignées par τ.

Nous définissons d'abord *la relation d'invariant de collage* sur (V_1 et V_2)[3] et *la relation de collage* μ entre les états de système raffiné et ceux du système abstrait.

Définition 5 *Un invariant de collage, notée I_{12}, est une relation entre les variables d'état du système abstrait ($\in V_1$) et celles du système raffiné ($\in V_2$) définie par une proposition de la forme suivante :*

$$q ::= ap_1 \,|\, ap_2 \,|\, x_1 = x_2 \,|\, q \wedge q \,|\, \neg q$$

où : $ap_1 \in AP_{V_1}$, $ap_2 \in AP_{V_2}$, $x_1 \in V_1$ et $x_2 \in V_2$.

Définition 6 *La relation de collage entre les états de $AH_1 = \langle F_1, E_1, \gamma_1 \rangle$ et $AH_2 = \langle F_2, E_2, \gamma_2 \rangle$, notée μ, est une relation binaire $\mu \subseteq \cup_{A_2 \in F_2} S_{A_2} \times \cup_{A_1 \in F_1} S_{A_1}$ qui exprime le fait que les états sont collés si leurs propriétés d'états satisfont le collage. Autrement dit, les états $s_2 \in \cup_{A_2 \in F_2} S_{A_2}$ et $s_1 \in \cup_{A_1 \in F_1} S_{A_1}$ sont collés par la relation μ, notée $s_2 \,\mu\, s_1$, ssi :*

$$\left(\bigwedge_{ap_2 \in L_2(s_2)} ap_2 \wedge I_{12} \right) \Longrightarrow \left(\bigwedge_{ap_1 \in L_1(s_1)} ap_1 \right)$$

Définition 7 *On dit que AH_1 est raffiné par AH_2 et on note $AH_1 \sqsubseteq_H AH_2$ ssi $s_{racine_{02}} \,\rho\, s_{racine_{01}}$ où $s_{racine_{01}}$ et $s_{racine_{02}}$ sont respectivement les états initiaux de l'automate racine de AH_1 et de AH_2 et ρ est la plus grande relation incluse dans μ vérifiant les conditions suivantes :*

1. *Raffinement de transitions*
   (a) *les anciennes transitions sont raffinées :*

$$\begin{array}{c} s_2 \,\rho\, s_1 \;\wedge\; s_2 \overset{sr_2,a,td_2}{\rightarrow} s'_2 \in \rightarrow_2 \Rightarrow \\ \exists s'_1 \,.\, (s_1 \overset{sr_1,a,td_1}{\rightarrow} s'_1 \in \rightarrow_1 \wedge s'_2 \,\rho\, s'_1 \wedge \\ (\forall s \in sr_2 \Rightarrow \exists s' \in sr_1 \wedge s \rho s' \vee \exists A_\tau \in \gamma(s_2)) \wedge \\ (\forall s \in td_2 \Rightarrow \exists s' \in td_1 \wedge s \rho s' \vee \exists A_\tau \in \gamma(s'_2)) \wedge \end{array}$$

   (b) *les τ-transitions bégaient :*

$$s_2 \,\rho\, s_1 \;\wedge\; s_2 \overset{\tau}{\rightarrow} s'_2 \in \rightarrow_2 \Rightarrow s'_2 \,\rho\, s_1$$

   (c) *pour les états terminaux[4] :*

$$\begin{array}{c} s_2 \,\rho\, s_1 \wedge s_2 \not\rightarrow \Rightarrow s_1 \not\rightarrow \vee \\ (s_2 \in S_{A_\tau} \wedge \exists t.(t \in \rightarrow_{\gamma^{-*}(s_2)} \wedge s_2 \in sr(t))) \end{array}$$

   (d) *conditions sur les nouveaux cycles :*

$$\begin{array}{c} cy \overset{def}{=} s_i \overset{\tau_1}{\rightarrow} \cdots \overset{\tau_n}{\rightarrow} s_n \in Exec(A_\tau) \Rightarrow \\ \exists t, j.(t \in \rightarrow_{\gamma^{-*}(s_i)} \wedge i \leq j < n \wedge s_j \in sr(t)) \end{array}$$

2. *Préservation de la hiérarchie entre les états*
   (a) *dans le système abstrait :*

$$s_2 \rho s_1 \wedge A_1 \in \gamma(s_1) \Rightarrow \exists A_2 \in \gamma(s_2) \wedge s_{0_{A_2}} \rho s_{0_{A_1}}$$

   (b) *dans le système raffiné :*

$$\begin{array}{c} s_2 \rho s_1 \wedge A_2 \in \gamma(s_2) \wedge \exists t \in \rightarrow_{A_2} \wedge \\ action(t) \in E_1 \Rightarrow \exists A_1 \in \gamma(s_1) \wedge s_{0_{A_2}} \rho s_{0_{A_1}} \end{array}$$

[3] V_1 dénote l'ensemble des variables de la spécification de système abstrait et V_2 l'ensemble des variables de la spécification raffinée.

[4] s est dit terminal s'il n'est l'état source d'aucun transition et on le note $s \not\rightarrow$.

4. Conclusion et Perspectives

Dans ce travail, nous avons défini une relation de raffinement entre automates hiérarchiques. Dans [3], nous prouvons que si les nouveaux automates A_τ ne contient pas de cycles alors la relation de raffinement préserve les propriétés LTL du niveau abstrait.

Nous envisageons d'étendre la préservation des propriétés LTL au raffinement présenté dans cet article. Pour généraliser cette préservation, nous pensons pouvoir exploiter la notion de priorité entre les actions du système abstrait pour établir des conditions d'équité sur la structure de Kripke associée, c'est-à-dire, définir une sémantique d'automate hiérarchique sous forme d'une *structure de Kripke équitable*. Ce choix nous permettera de représenter les priorités entre les transitions de l'automate hiérarchique, sous forme d'hypothèses d'équité.

Références

[1] J.-R. Abrial. *The B-book : assigning programs to meanings*. Cambridge University Press, New York, NY, USA, 1996.

[2] M. Al'Achhab. Specification and verification of hierarchical systems by refinement. In *Winter School on Modelling and Verifying Parallel Processes, MOVEP'04*, pages 103–109, Bruxelles, Belgique, Dec. 2004.

[3] M. Al'Achhab, A. Hammad, and H. Mountassir. Vérification de systèmes hiérarchiques par raffinement. Grenoble, France, Oct. 2005. À paraître In Colloque MSR'05, Modélisation des Systèmes Réactifs.

[4] F. Bellegarde, J. Julliand, and O. Kouchnarenko. Ready-simulation is not ready to express a modular refinement relation. In *Fondamental Aspects of Software Engineering 2000, FASE'2000*, volume 1783 of *LNCS*, pages 266–283, Berlin, Mar. 2000.

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

[6] D. Harel. Statecharts : A visual formalism for complex systems. *Science of Computer Programming*, 8(3) :231–274, June 1987.

[7] G. J. Holzmann. The model checker SPIN. *Software Engineering*, 23(5) :279–295, 1997.

[8] 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.*, 11(6) :637–664, 1999.

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

[10] 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*, pages 181–196, London, UK, 1997. Springer-Verlag.

[11] 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*, pages 90–101. IEEE Computer Society, October 1998.