Intl. Conf. RIVF'04
February 2-5, Hanoi, Vietnam

Génération d'un Simulateur Stochastique Guidée par la Description du Système

Minh-Thang Khuu

***Abstract*— Stochastic simulation is an approach usually used in system dependability studies when systems are too large or too complex to be solved by analytical methods. By using states/transitions formalisms (transitions systems, Markov chains, Petri nets, etc.) for modeling, analytical methods are hardly accessible due to the state space explosion problem and simulation seems to be the most effective way for system evaluation. However, this approach suffers from the following disadvantages :**

- **it provides statistical estimates rather than the exact characteristics and performance mesures of the model ;**
- **it is very time consuming.**

In this paper, we show that a compilation of the system 's stochastic model into programming language 's codes before carrying out simulations will reduce execution time of the simulation processus. This compilation generates simulators which are based on the structure of the systems. Experimental results carried out on the *AltaRica Data Flow* model show that the effective duration of the simulation can be considerably reduced.

Abréviations :

ADF *AltaRica Data Flow*
RdPSG *Réseaux de Petri Stochastiques Généralisés*

I. INTRODUCTION

La simulation stochastique est une des méthodes de l'évaluation d'une mesure de la sûreté de fonctionnement d'un système, notamment quand les systèmes sont trop grands ou trop complexes pour être résolus de manière analytique. En raison du problème de l'explosion du nombre d'états, elle est presque la seule approche accessible permettant d'utiliser les formalismes de type états/transitions (systèmes de transitions, chaînes de Markov, réseaux de Petri, etc.) pour la modélisation. Cependant, cette approche souffre des inconvénients suivants :

- nous n'obtenons que des résultats estimés, pas exacts. Autrement dit ils sont exprimés dans une intervalle de confiance.
- le temps de simulation est énorme pour atteindre une stabilisation des résultats.

Pour réduire le premier problème, des techniques de réduction de variance sont utilisées [9]. Dans cet article, nous examinons le deuxième problème.

Considérons la simulation stochastique des formalismes de type états/transitions. De manière intuitive, une simulation stochastique effectue une « marche aléatoire »à travers l'espace d'états du système. Cette marche nécessite des parcours de la structure modélisant le système pour choisir l'état suivant et mettre à jour des informations du système après chaque étape de simulation (liste des transitions qui sont activées/désactivées, variables d'observation, etc.). Ces parcours consomment de grandes quantités de temps de simulation. Nous proposons que le modèle décrivant le système soit compilé en code exécutable d'un langage de programmation avant d'effectuer la simulation. L'expérimentation est effectuée sur le modèle *AltaRica Data Flow*.

Nous présentons d'abord la simulation stochastique et le modèle *ADF*. Ensuite nous proposons une compilation de la description d'ADF en langage de programmation. Pour terminer nous abordons des perspectives et des travaux restant à faire.

Minh Thang Khuu est doctorant à l'Université de la Méditerranée, faculté des sciences de Luminy, 163, av. de Luminy - 13288 Marseille, France, (email: khuu@iml.univ-mrs.fr)

II. SIMULATION PAR ÉVÉNEMENTS DISCRETS

Les concepts fondamentaux de la modélisation pour la simulation s'appuient méthodologiquement sur les notions de base : *système* et *modèle*. Le premier désigne un ensemble de composants reliés d'une certaine manière alors que le second signifie une abstraction du système réel. Le choix du modèle est guidé par des caractéristiques du système et les propriétés à évaluer. Les modèles de simulations peuvent être classifiés de différentes manières :

- Modèles statiques vs modèles dynamiques : ils se distinguent par la présence de représentation de l'évolution du temps.
- Modèles déterministes vs modèles stochastiques : dans les modèles stochastiques, il existe au moins un élément (ou variable) aléatoire qui influence l'évolution du système étudié.
- Modèles continus vs modèles discrets : cette classification caractérise le changement continu ou discret au cours du temps des grandeurs modélisant les états du système.

La simulation par événements discrets désigne la modélisation d'un système réel tel qu'il évolue dans le temps, par une représentation dans laquelle les grandeurs caractérisant le système ne changent qu'en nombre fini ou dénombrable de points isolés dans le temps.

Le temps est géré par un échéancier et par une horloge centrale. L'échéancier est une liste d'événements ordonnés chronologiquement selon l'heure du système de simulation.

Horloge de simulation : il s'agit d'une variable qui enregistre le temps de simulation actuel. Ce dernier est avancé lorsqu'un événement a lieu.

Liste d'événements : les événements de cette liste sont maintenus dans un ordre chronologique. L'événement le plus imminent, c.-à-d dont le temps précédent l'occurrence (ou la durée de temporisation) est le plus petit, se situe à la tête de la liste.

Initialement, l'horloge est mise à zéro et les événements qui peuvent avoir lieu sont insérés dans la liste selon l'ordre chronologique. Ensuite, l'événement le plus imminent en est enlevé pour l'exécution, et l'horloge de simulation est avancée de la durée de temporisation de l'événement. Pendant l'exécution d'événement, l'état du système est mis à jour. De nouveaux

événements sont insérés dans la liste et certains en sont enlevés. Ce processus est répété jusqu'au moment où une des deux conditions suivantes se vérifie :

– il n'y a plus de transition qui peut être franchie.
– le temps de mission est expiré.

III. Modèle AltaRica Data Flow

Le modèle ADF est proposé par A.Rauzy dans le projet appelé *AltaRica*, qui a pour ambition de définir un modèle de haut niveau pour les études de sûreté de fonctionnement des systèmes et de construire une boîte d'outils supportant ce modèle. Une fois défini, ce modèle permet de :

– faciliter la communication entre plusieurs groupes de travail utilisant des formalismes différents.
– accéder à des méthodes/outils différents dans l'étude des systèmes critiques (Aralia [1], Moca-RP [1], MEC [2]).

L'accès à des méthodes/outils est réalisé grâce à des compilations du modèle ADF vers des formalismes de base. Ces compilations sont présentées dans [4], [6], [8].

A. Modèle fondamental

Un système modélisé par ADF est constitué de composants. Les états du composant sont modélisés par les évaluations d'un ensemble de variables appelées *variables d'états*. Les *variables de flux d'entrée et de sortie* définissent les entrées et les sorties du composant. La valeur des flux de sortie dépend de celle des flux d'entrée et de l'état actuel du composant. Cette dépendance est exprimée sous forme d'*assertions*. Le changement d'états s'exprime en termes de *transitions* déclenchées par des *événements*. La sémantique du modèle ADF s'exprime en termes d'automates de mode [8].

Le langage ADF permet l'écriture d'automates de mode sous forme textuelle et compréhensible [7]. Par exemple, la description suivante décrit un composant qui se trouve dans un des trois états : *en-marche*, *en-panne* ou *en-arrêt*

```
node unit
  state
    s : {en-marche, en-panne, en-arret};
  flow
    entree:bool:in; sortie:bool:out;
  event
    panne, repare, demarre;
  trans
    s=en-marche |- panne    -> s:=en-panne;
    s=en-panne  |- repare   -> s:=en-arret;
    s=en-arret  |- demarre  -> s:=en-marche;
  assert
    sortie = entree and (s=en-marche);
  init
    s = en-arret;
end
```

Les composants interagissent selon deux mécanismes : *la coordination de flux* et *la synchronisation d'événements*. Leurs sémantiques sont exprimées en terme d'opérations de composition d'automates de mode. A.Rauzy [8] définit trois opérations de base :

[1] http ://www.gfic-oasys.com/french/grif/moca_desc.htm

– la composition parallèle : la composition parallèle consiste à placer des automates de mode l'un à côté l'autre. La figure 1(a) illustre la composition parallèle de trois composants *A, B et C*. L'état du système est constitué de celui des composants. Cette composition exprime un produit libre des automates. Elle s'effectue en premier dans toutes les séquences de compositions et résulte en un seul automate. La figure 1(a) illustre la composition parallèle de trois composants *A, B et C*.

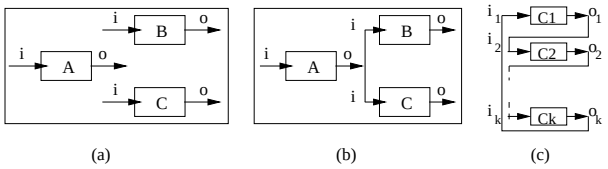

Fig. 1. Composition parallèle et connexion des automates de mode

– la connexion : Il s'agit d'un branchement d'une sortie à plusieurs entrées. La figure 1(b) illustre la connexion de la sortie *A.o* aux entrées *B.i* et *C.i*.
Le modèle ADF ne traite pas la dépendance cyclique des variables de flux. Il s'agit du problème de causalité étudié dans le cadre de langages réactifs [5]. Ce problème est qualifié de dépendance causale dans la terminologie des automates de mode [8]. La figure 1(c) montre un exemple où $o_1, o_2, \ldots, o_k$ font un cycle de dépendance.
– la synchronisation : il s'agit de synchroniser des événements du système. A titre d'exemple, prenons un *système en redondance passive*, courant dans les études sur la sûreté de fonctionnement. Ce système est constitué de deux composants identiques décrits dans l'exemple précédent. Initialement, le premier est en marche et le deuxième est en arrêt. Lorsque le premier tombe en panne, le deuxième est démarré automatiquement. La description en langage ADF du système est :

```
node systeme
  sub
    U1, U2 : unit;
  sync
    <U1.panne, U2.demarre>;
end
```

Les priorités peuvent être associées aux événements. Il s'agit d'un ordre partiel défini sur l'ensemble des événements. Considérons le système de redondance passive décrits ci-dessus. Supposons qu'initialement le premier soit dans l'état de marche, l'autre dans l'état d'arrêt et que le commutateur fonctionne parfaitement. Lorsque le premier tombe en panne, le deuxième sera démarré immédiatement. L'événement *U2.demarre* a donc une priorité plus élevée que celle de *U1.repare*.

B. Modèle hiérarchique vs modèle plat

En utilisant ADF comme langage de modélisation, le système est décomposé en sous-systèmes et composants. La description des interactions de ces sous-systèmes/composants forme une hiérarchie décrivant le système entier.

L'application des opérations de composition d'automate de modes aboutit à un automate de mode unique. L'automate de mode obtenu est un modèle « plat »du système examiné. Le processus d'application des opérations est donc appelé *mise à plat*

de hiérarchie. Ce modèle est pris comme entrée dans la plupart des outils associés au modèle ADF.

C. Aspect stochastique d'ADF

Il est introduit au modèle par l'association des fonctions de distribution probabiliste aux événements. Ces fonctions définissent, de manière aléatoire, la temporisation des événements. Comme chaque transition est étiquetée par un événement, cette temporisation désigne le *délai de franchissement* des transitions. La transition ayant le délai de franchissement le plus court sera franchie.

L'introduction du délai de franchissement scinde les transitions en deux catégories : transitions temporisées et transitions instantanées. Ces dernières sont raffinées en transitions immédiates et transitions conditionnelles. Étant dans un état donné, l'ensemble des transitions conditionnelles franchissables désigne l'ensemble des transitions en conflit : la somme de leur probabilités doit être égale à 1.

Le modèle stochastique ADF est celui des RdPSG [3]. Pour que le processus stochastique engendré à partir du modèle ADF soit déterminé, nous devons déterminer les politiques de choix de mémoire et de service. Les deux dernières dépendent du système modélisé. Nous examinons la première.

Nous empruntons des terminologies utilisées dans des RdPSG. Le choix de transition à franchir s'appuie sur la *politique de course* et le principe *post-sélection*. Les transitions franchissables sont triées en ordre croissant de la durée de franchissement. Étant donné un ensemble des transitions franchissables, nous examinons la nature de ces transitions.

- Les transitions temporisées : la durée de franchissement est générée par la génération de variables aléatoires présentée ci-dessus. La transition ayant la durée de franchissement la plus courte sera choisie.
- Les transitions immédiates : la durée est égale à zéro et elles sont donc franchies immédiatement.
- Les transitions conditionnelles : Soient $pr_1, pr_2, ..., pr_k$ sont les probabilités des transitions conditionnelles franchissables $t_1, t_2, ..., t_k$, $pr_1 + pr_2, ... + pr_k = 1$. Étant donné $r(0 \leq r \leq 1)$ un nombre aléatoire généré, si $pr_1 + pr_2 + ... + pr_j < r \leq pr_1 + pr_2 + ... + pr_{j+1}$ alors la transition t_{j+1} sera choisie.

Les lois de distribution probabiliste associées aux événements sont déclarées dans la clause *extern* des descriptions d'ADF [7].

IV. Génération de simulateurs stochastiques

L'objectif de la projection d'ADF en code exécutable est, à partir d'une description d'ADF, d'engendrer un simulateur stochastique *propre* à la description. Le schéma de la génération du simulateur est décrit dans la figure 2.

- Le cadre général réalise la boucle principale de la simulation, la génération des nombres aléatoires en fonction d'une distribution probabiliste.
- La description en langage de programmation C du système fournit le *paramètrage* du cadre général : la liste des événements, les variables dont l'évaluation caractérise l'état du système, les mesures à estimer, etc...

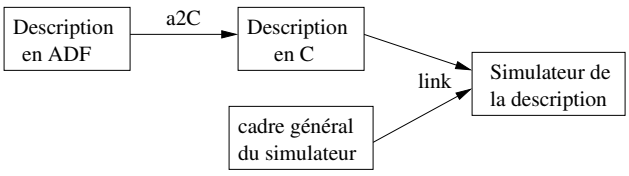

Fig. 2. Processus de génération du simulateur d'une description d'ADF

A. Détails de la projection

Les éléments de base du modèle ADF mis à plat sont compilés en structures de données ou fonction du langage de programmation C :

- *Les variables d'états et de flux :* elles sont transformées en variables du langage C.
- *Les événements :* ils sont aussi transformés en variables du langage C. Chacune de ces variables est une structure contenant le nom de l'événement, sa priorité et la fonction de distribution probabiliste.
- *Les transitions :* chaque transition est transformée sous la forme de deux fonctions :*enable_()* et *fire_()*. La première retourne une valeur booléenne permetant de vérifier si la transition est activée ou désactivée. La deuxième fonction effectue des affectations à des variables d'états inclueses dans la transition. La mise à jour de variables de flux est aussi effectuée en appelant des fonctions correspondantes.
- *Les assertions :* chaque assertion est transformée en une fonction *update_()*. Ces fonctions sont appelées par des fonctions *fire_()*.
- *L'état initial :* une fonction *init_state()* est définie pour effectuer les affectations des valeurs initiales aux variables d'états.

B. Résultats

La projection de la description ADF en langage de programmation C est réalisée par la programme *a2C*, qui est implémenté en langage de programmation OCaml.

Le tableau 3 effectue une comparaison de temps d'exécution du simulateur stochastique *alta_sto*, qui parcourt directement les structures de données du modèle, avec les simulateurs générés à partir des description d'ADF. Le système *KR02* contient 8 variables d'états, 8 transitions. Le système *TFE02-3* contient 50 variables d'états, 50 transitions. Le tableau montre que le gain de temps d'exécution est d'autant plus grand que le temps de mission est long.

V. Conclusion

Nous avons présenté une génération de simulateurs stochastiques guidée par le modèle du système. Elle peut être améliorée en intégrant d'autres techniques d'optimisation de simulation. Nous sommes en train d'étudier la construction de la machine virtuelle de simulation en transformant les descriptions d'ADF vers son ensemble de commandes. Le schéma dans la figure 2 est réutilisé avec les modifications suivantes : un environnement d'exécution de la machine virtuelle à la place du cadre général de simulation et l'ensemble de commandes de la machine à la place de la description C.

Exemple	Temps de mission	Nombre d'histoires	alta_sto	simulateur généré	Gain
KR02	1e+3	1e+5	3.83 (s)	0.69 (s)	5.55
		5e+5	18.87	3.09	6.10
		1e+6	37.70	6.20	6.08
	1e+4	1e+5	6.48	0.86	7.95
		5e+5	32.31	3.98	8.11
		1e+6	64.46	7.98	8.07
	1e+5	1e+5	10.74	1.08	9.94
		5e+5	53.04	5.12	10.35
		1e+6	105.96	10.15	10.43
TFE02-3	1e+3	1e+4	3.72	0.51	7.29
		1e+5	36.65	4.43	8.23
		1e+6	363.95	43.53	8.36
	1e+4	1e+4	9.42	1.15	8.88
		1e+5	93.97	10.75	8.7
		1e+6	938	107	8.7
	1e+5	1e+4	13.35	1.5	8.9
		1e+5	133	14.32	9.3
		1e+6	1334	143	9.3

Fig. 3. Comparaison de temps d'exécution des simulateurs (CPU 1.6MHz, 256MB RAM)

REFERENCES

[1] Group Aralia. Computation of prime implicants of a fault tree within aralia. In *Proceeding of European Safety and Reliability Association conference (ESREL)*, 1995.

[2] A. Arnold, D. Begay, and P. Crubille. *Construction and Analyse of transition systems with MEC*. World Scientific Publishing Co. Pte. Ltd., 1994. AMAST Series in Computing : Vol. 3.

[3] M.Ajmone Marsan et al. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995. ISBN 0-471-93059-8.

[4] G.Point. *Contribution à l'unification des méthodes formelles et de la sûreté de fonctionnement*. PhD thesis, Université Bordeaux I, 2000.

[5] Nicolas Halbwachs. *Synchronous Programming of Reactive Systems*. Kluwer, 1993. ISBN 0-7923-9311-2.

[6] MT.Khuu and A.Rauzy. A generalized stochastic petri net semantic for mode automata. In *ICAPL Satellite workshop on GSPNs*, 2003.

[7] A. Rauzy. The altarica data-flow language. Technical report, CNRS/IML, 2002.

[8] A. Rauzy. Mode automata and their compilation into fault trees. *Reliability Engineering and System Safety*, April 2002.

[9] RY. Rubinstein and B.Melamed. *Modern Simulation and Modeling*. John Wiley & Son, 1998.