Model Checking :Etude et implantation

Laurent VAN BEGIN

9 octobre 2000

Des systèmes informatiques de plus en plus perfectionnés sont présents, chaque jour, dans de nouveaux secteurs d'activités. Aux machines informatiques utilisées comme machines de calculs et de recherches d'information, s'est ajouté la machine de contrôle d'équipements. Les systèmes réactifs obtenus sont composés d'équipements électroniques ou mécaniques et d'un système informatique qui les contrôle ; les interactions entre la partie contrôle et les équipements s'effectuant par des stimulis appelés généralement "évenements". De façon générale, les équipements étant distribués géographiquement, la partie contrôle l'est également. Un système de contrôle de passage à niveau est un exemple classique de système réactif. De part la nature de ces systèmes réactifs, ceux-ci sont bien souvent *critiques*, c'est-à-dire que leur fiabilité est crutiale. Le mémoire réalisé traite de la conception fiable de tels systèmes.
Généralement, pour contruire des systèmes fiables, plusieurs niveaux de descriptions existent. Une description abstraite donne les différentes caractéristiques fonctionnelles ou non requises pour le système. A côté de cela,une description plus détaillée décrit plus en détail la structure et le fonctionnement interne du système.
La technique du model-checking (contrôle de modèle) permet de vérifier qu'une description détaillée d'un système a bien les caractéristiques,propriétés ,qui lui sont requises. Par exemple pour le système de passage à niveau, une propriété souhaitée est que la barrière soit toujours fermée lorsqu'un train passe. Spécifier cette propriété en language naturel a comme avantage que l'idée sous jascente est facilement compréhensible. Cette idée malheureusement peut varier d'une personne à l'autre suivant l'interprétation particulière de chaque personne. Si effectivement des propriétés sont vérifiées, se pose alors le problème de savoir ce qui a été effectivement vérifié. Formaliser les descriptions de systèmes et leurs caractéristiques souhaitées à l'aide de l'outil mathématique semble pour cela indispensable. Une logique temporelle est en général le formalisme choisi pour la spécification de propriétés. Une telle logique peut être vue comme une logique *classique* à laquelle a été rajoutée un ou plusieurs opérateurs temporels. Reprenant l'exemple du passage à niveau, on peut écrire la formule $\Box(in \rightarrow \Diamond out)$ qui signifie qu'à tout instant ("$\Box$"), si le train se trouve dans le passage à niveau ("in") alors

inévitablement ("$\Diamond$") il en sortira ("out"). Dans le cadre de notre travail de fin d'étude, nous nous sommes tout d'abord proposés de faire une étude de différents modèles mathématiques utilisés pour modéliser les comportements d'un système ; différents types d'automates ont ainsi été étudiés. Nous avons également fait l'étude d'un certain nombre de logiques temporelles adaptées à la description de propriétés sur ces modèles. En gros, les modèles d'automates et les logiques temporelles étudiées sont

- a)orientés état ou b)orientés action, ce qui signifie qu'on s'intéresse soit aux propriétés des différents états traversés par le système lors de ses exécutions soit aux différentes actions effectuées lors des changements d'états du système.
- a)temporisés ou b) non temporisés, ce qui signifie qu'une notion de temps quantifié existe ou non. C'est-à-dire la possibilié ou non, par exemple, pour une logique de spécifier un temps d'attente maximum et/ou minimun avant qu'une propriété ne soit vérifiée.

Un certain nombre d'algorithmes de Model Checking utilisant des formalismes aux caractéristiques différentes ont été étudiés. Ces algorithmes permettent de répondre de manière automatique à la question qui est de savoir si les exécutions d'un automate satisfont bien une propriété spécifiée sous la forme d'une formule de logique temporelle. Un tel algorithme a été implanté dans le cadre du mémoire. La logique utilisée dans ce cadre a été la logique TCTL. Cette logique est temporisée orientée état et permet de spécifier des propriétés de choix, c'est-à-dire que sa sémantique est dite "branchante". Un exemple de propriété spécifiable en TCTL est que les choix lors d'une exécution sont tels qu'il est toujours possible d'atteindre un état satisfaisant une propriété p donnée avant n unités de temps. Le but de cette implantation étant d'avoir un outil de base pouvant être amélioré par la suite.

Construire un automate à la main représentant les comportements d'un système n'est pas réaliste lorsque le système considéré est un tant soit peu complexe. Une solution à ce problème est de modéliser les comportements du système à l'aide d'un formalisme *de plus haut niveau*, c'est-à-dire plus proche du langage humain. Le processus de vérification de propriétés peut être décomposé comme suit :

- spécifier les comportements du système à l'aide d'un outil de spécification de *haut niveau* ;
- appliquer une fonction à cette spécification afin d'obtenir un automate *équivalent* ;
- vérifier la validité de formules temporelles sur cet automate.

Il est intéressant que la spécification de haut niveau et la logique temporelle utilisée ait des propriétés similaires. C'est-à-dire, lorsque par exemple le modèle de haut niveau définit l'ensemble des exécutions du système en termes de séquences d'actions, la logique utilisée devrait permettre de spécifier des propriétés sur ces séquencements d'actions et non sur les propriétés des états traversés lors de ces exécutions. Ceci n'est pas techniquement in-

dispensable mais souhaitable pour des raisons de cohérence, de simplicité de spécification et de compréhension des propriétés exprimées. Des formalismes temporisés orientés actions permettant de spécifier les comportements d'un système existent, comme par exemple les algèbres de processus LOTOS et son extension temporisée ET-LOTOS. Par contre une logique temporelle temporisée orientée action possédant une sémantique branchante n'a pas été trouvé dans la littérature. Cela nous a insité à définir une logique à partir de TCTL ayant ces propriétés. Nous l'avons logiquement appelé ATCTL(pour Action TCTL). Définir une logique consiste à définir sa syntaxe (comment écrire une formule) et la sémantique associée (ce que veut dire la formule). Le but du travail étant de pouvoir vérifier des propriétés, deux algorithmes permettant de vérifier des formules ATCTL sur des automates temporisés ont été défini. Malheureusement, faute de temps, seul un des deux algorithmes a pu être implanté dans le cadre du mémoire. Pour cette raison, aucune comparaison sur le comportement en pratique de ces deux algorithmes n'a pu être faite.

Dans le futur, il serait intéressant de faire une telle comparaison. Les algorithmes définis dans le cadre du mémoire peuvent avoir des performances, en terme de temps d'exécution et de mémoire utilisée, grandement améliorées en pratique. Une comparaison entre notre outil de Model Checking utilisant TCTL et l'outil KRONOS implantant le même algorithme montre des différences de performance plus que notable. Le choix d'une bonne structure de données peut par exemple apporter un gain de performance notable, tant au niveau de l'espace mémoire utilisé que du temps d'exécution.