Analyse statique des programmes

Alors que le texte d'un programme est un objet statique, qui ne change pas au cours de son exécution, cette exécution elle-même est un processus dynamique dans lequel un état évolue.

Le but de l'analyse statique est de vérifier des propriétés de l'exécution d'un programme par la seule analyse de son texte. C'est l'une des voies utilisées pour aller vers la production de programmes corrects.

Déterminer des propriétés de l'exécution d'un programme (et notamment vérifier l'absence de certaine erreurs) par la seule analyse de son texte est une idée qui remonte aux débuts de la programmation : l'auteur d'un programme en fait un examen attentif, avant de le faire exécuter. Cette méthode trouve vite ses limites, dès que le programme est un peu long et complexe. L'analyse statique poursuit le même objectif, avec une approche systématique fondée sur une base rigoureuse. La relecture reste néanmoins utilisée, notamment de manière croisée (chaque programmeur relit le programme d'un de ses collègues).

Deux grandes voies sont utilisée pour l'analyse statique. Elles reposent sur une démarche d'abstraction (remplacer le programme par un modèle abstrait plus facilement analysable).

- Le *model checking*. Cette méthode repose sur une représentation du système analysé par un graphe de transition, qui modélise l'évolution du système par un passage d'un état à un autre, sous certaines conditions. Un exemple simple est un distributeur de boissons, dont les états représentent les diverses valeurs possibles de la somme d'argent collectée et les transitions sont déclenchées par l'insertion d'une pièce. La transition finale déclenche la délivrance de la boisson et le rendu éventuel de la monnaie. Si un tel système est représenté par quelques états, les systèmes réels peuvent en avoir des dizaines ou des centaines de milliers. Les propriétés à vérifier sont par exemple l'absence de blocage, la terminaison effective de l'exécution, le passage par tel ou tel état, etc. La principale difficulté que doivent surmonter les outils de *model checking* réside dans la croissance exponentielle, avec le nombre d'états, du nombre potentiel de transitions. Le *model checking* est davantage utilisé pour la vérification de systèmes matériels que pour la vérification de programmes.
  Le *model checking* a été inventé au début des années 1980 par Edmund Clarke, Allen Emerson et Joseph Sifakis, qui ont reçu en 2007 le prix Turing.

- L'interprétation abstraite. Dans l'interprétation abstraite, on s'intéresse aux «trajectoires» qui schématisent les exécutions possibles d'un programme, dans un espace représentant les variables qu'il utilise. Dans cet espace, certaines régions sont interdites (telle variable doit rester dans telle plage de valeurs, etc.). Pour un programme un peu complexe, le nombre de trajectoires possibles est très grand et empêche tout examen exhaustif. Le principe de l'interprétation abstraite consiste à construire une approximation de ces trajectoires permettant de vérifier si elles restent ou non dans les limites autorisées, sans avoir à les examiner toutes.
  Le principe de l'interprétation abstraite a été posé en 1977 par Patrick Cousot.

Ces méthodes d'analyse statique, pour lesquelles de nombreux outils ont été développés, ne sont pas infaillibles. Elles permettent néanmoins d'améliorer considérablement le processus de vérification des systèmes informatiques.