Poste d'ingénieur associé - détails de la mission

Isabelle Puaut - Inria-Rennes, Action ACES

1 Résumé des objectifs

Nom du logiciel : Heptane

Fonctionnalités : Heptane (Hades Embedded Processor Timing ANalyzEr) est un logiciel d'obtention automatique de pires temps d'exécution de logiciels temps-réel. Heptane, par analyse statique du code source d'un programme C, retourne une borne supérieure de son temps d'exécution sur une architecture Intel Pentium. Heptane exploite une modélisation fine de la micro-architecture du processeur (exécution pipelinée, cache d'instructions, prédicteur de branchement) de manière à évaluer les temps de manière la plus précise possible.

Statut actuel du logiciel et mode de diffusion envisagé : Le logiciel est en cours d'enregistrement à l'APP. Il sera diffusé sous licence publique dès que son niveau de documentation sera suffisant.

Responsable scientifique : Isabelle Puaut
Inria-Rennes, Action ACES (Ambient Computing and Embedded Systems), issue du projet Solidor (voir http://www.irisa.fr/solidor).
e-mail: puaut@irisa.fr
Tel: 02 99 84 73 10
Fax: 02 99 84 25 29

Public concerné par le logiciel : Académiques et industriels effectuant des recherches ou développant des logiciels ayant des contraintes de temps-réel strict (avionique, automobile, production d'énergie).

Objectif de la mission : L'objectif de la mission est de robustifier le code du logiciel d'analyse Heptane, ainsi que d'y ajouter des procédures d'installation et de la documentation, afin de pouvoir le diffuser sous forme open-source à l'extérieur de l'Irisa.

2 Contexte général d'utilisation d'Heptane

La connaissance de temps d'exécution au pire cas (WCET, pour Worst-Case Execution Times) est un paramètre d'entrée incontournable pour la validation temporelle de systèmes temps-réel strict, pour lesquels un dépassement d'échéance peut avoir des conséquences humaines ou économiques graves. Les WCET doivent être connus de manière sûre (majorer les temps d'exécution effectifs) tout en étant le moins élevés possibles, pour ne pas entraîner de sur-dimensionnement des ressources nécessaires à l'exécution du système. Par exemple, dans l'aéronautique, il n'est pas rare de trouver des processeurs jamais utilisés à plus de 20% de leurs capacités maximales.

Des méthodes d'analyse statique de programmes (*analyse statique de WCET*) ont été définies dans les dix dernières années pour obtenir de manière sûre les WCET de programmes par analyse de leur code source (voir [PB00] pour une vue d'ensemble de ces méthodes). Un des moyens pour éviter le pessimisme de ces méthodes est de prendre en compte la micro-architecture des processeurs. L'analyse statique de programmes permet d'obtenir des estimations sûres des WCET. Deux niveaux peuvent être distingués dans les méthodes d'analyse statique de WCET :

- *Caractérisation des chemins d'exécution (analyse de haut niveau).* Il s'agit ici d'identifier les différents chemins d'exécution dans un programme à partir de son code source (c'est pourquoi on parle d'analyse de *haut niveau*).
- *Obtention du temps d'exécution au niveau matériel (analyse de bas niveau).* Il s'agit ici d'obtenir la durée d'exécution de chaque instruction ou séquence d'instructions sur une architecture matérielle donnée. Cette analyse opère au niveau du code objet des programmes (analyse de *bas niveau*) et nécessite d'utiliser une modélisation du comportement temporel du processeur cible.

Heptane (voir paragraphe 4) est un outil mettant en œuvre de telles méthodes d'analyse. À partir d'un code source C, il génère, par parcours de sa structure syntaxique (méthode *tree-based*), son pire temps d'exécution sur une architecture Intel Pentium.

3 Domaines d'applications visés et originalité de l'outil

Heptane trouve son utilisation dans les domaines ou le temps de réponse des logiciels est un facteur critique de correction (avionique, certaines applications automobiles, production d'énergie).

À l'heure actuelle, peu d'industriels du domaine du temps-réel utilisent ce type d'outil pour analyser les performances pire-cas de leurs logiciels. Ceci s'explique partiellement par le manque d'outils disponibles dans le domaine. À notre connaissance, seuls deux outils commerciaux de ce type existent : Bound-T (Space Systems Finland, http://www.ssf.fi) et les outils de la société Absint (http://www.absint.com), les points faibles de ces deux outils étant la prise en compte de l'architecture matérielle. Le logiciel universitaire Cinderella (http://www.ee.princeton.edu/~yauli/cinderella-3.0), basé sur une méthode d'analyse différente de celle utilisée par Heptane souffre du même défaut (manque de précision de la modélisation de micro-architecture).

Contrairement à l'état de l'art des outils actuellement disponibles, Heptane a pour originalité d'intégrer une modélisation *fine* de la micro-architecture du processeur (cache d'instruction, exécution pipelinée, prédiction de branchements), de manière à générer des temps d'exécution qui sont à la fois sûrs et précis. Il a également la bonne propriété d'être modulaire, lui permettant d'être reciblé facilement d'un processeur à un autre.

4 Le logiciel Heptane

Le logiciel Heptane a été développé principalement par Antoine Colin pendant sa thèse [Col01]. Cette thèse s'est partiellement déroulée dans le cadre d'un partenariat avec la Direction Générale de l'Armement (DGA), impliquant la société Dassault-Aviation.

Heptane intègre les méthodes d'analyse de WCET proposées dans la thèse d'Antoine Colin (analyse de bas niveau par prise en compte statique de la micro-architecture [CP00, CP01a], analyse de haut niveau précise grâce à un système d'annotation symbolique des boucles).

Le logiciel est maintenant stable, puisqu'il a été utilisé avec succès pour analyser des codes de taille réelle (la partie centrale du code du noyau de système temps-réel RTEMS [CP01b]).

La structure interne d'Heptane est schématisée sur la figure 1. Les différents modules le composant sont : (i) un analyseur de code source, générant sous forme de fichier XML les structures de données utilisées par le cœur de l'analyseur ; (ii) le cœur de l'analyseur, intégrant de manière modulaire la prise en compte des caches d'instructions, prédicteurs de branchement, et exécution pipelinée. L'analyseur génère des équations Maple qui réalisent le calcul effectif du WCET.

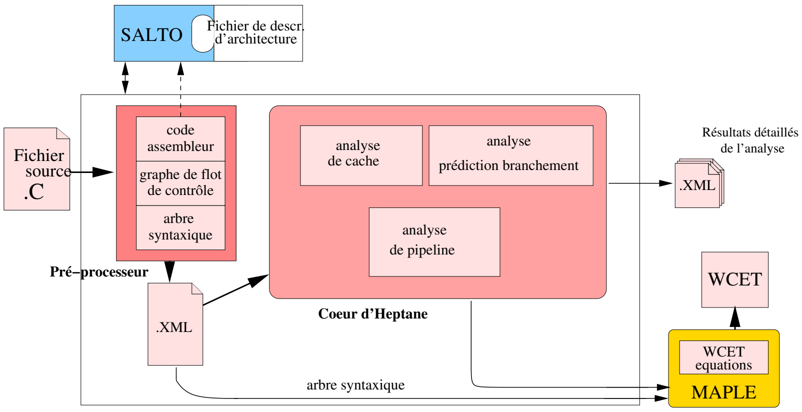

FIG. 1 – *Structure interne de l'analyseur Heptane*

Le préprocesseur est développé en C/Lex/Bison. Le cœur de l'analyseur est développé en utilisant le compilateur Objective Caml diffusé par le projet Cristal. Les informations sur la micro-architecture sont obtenues grâce à l'outil Salto, développé au sein du projet Caps.

5 Détails des développements à effectuer

Les développements à effectuer sont donnés ci-dessous en trois catégories, la première étant incontournable à l'utilisation du logiciel à l'extérieur, les deux autres ayant pour objectif d'augmenter le nombre de ses utilisateurs potentiels :

Ergonomie d'utilisation et d'installation. Dans son état actuel, l'installation, l'utilisation et la configuration d'Heptane ne sont réalisables que par ses concepteurs, car aucune documentation technique complète n'existe. Le traitement et l'explication des cas d'erreurs est également à améliorer. Un point important est aussi de rationaliser les langages et outils utilisés par Heptane, comme par exemple n'utiliser qu'un seul langage de programmation (Objective Caml) et réécrire les quelques parties de code encore développées en C/C++.

Multi-cible. Malgré sa conception volontairement modulaire, Heptane a actuellement un seul processeur cible (Intel Pentium), limitant son utilisation à plus grande échelle. L'objectif est ici de cibler plusieurs processeurs très répandus dans le monde de l'embarqué (ARM, PowerPC).

Éliminer l'utilisation de logiciels propriétaires. Heptane utilise actuellement le logiciel Maple pour le calcul final des WCET. Dans l'objectif d'une large diffusion, il nous semble plus opportun de le remplacer par un outil d'évaluation symbolique non commercial.

6 Planning envisagé pour les développements.

Le planning prévisionnel des développements est le suivant :

- **Phase 1 :** Prise en main de l'outil et remise à niveau de compétences de la personne recrutée. Étant donnée la gamme des compétences nécessaires à la modification de l'outil (méthodes d'analyse, langages fonctionnels, architectures matérielle, calcul symbolique), il semble nécessaire de mettre à niveau les compétences de la personne recrutée en l'initiant à notre problématique et aux outils et langages utilisés lors des développements.
- **Phase 2 :** Robustification, documentation et diffusion sous forme open-source. L'objectif visé est qu'au terme de cette phase, une version opérationnelle robuste et documentée d'Heptane soit diffusée, sans ajout à ce stade de fonctionnalités supplémentaires au logiciel. Différentes sous-étapes peuvent être distinguées :
  - Mise en place de procédures d'installation automatisées ou semi-automatisées (script configure)
  - Mise en place de tests et de vérification automatisée de la conformité des tests, afin de valider que l'installation s'est déroulée correctement. Ces tests seront utilisés comme tests de non régression lors des modifications du logiciel effectuées dans les phases suivantes.

  - rédaction de documentation d'installation et d'utilisation (manuel d'utilisation et tutoriel incluant des exemples d'utilisation). Toutes les documentations seront rédigées en langue anglaise.
  - amélioration des procédures de détection et d'explication des erreurs
  - rationalisation des outils de développement utilisés (suppression des quelques morceaux encore écrits en C et C++ pour adopter objective Caml comme langage unique de développement)
- **Phase 3 :** Extension d'Heptane à plusieurs cibles. Deux cibles additionnelles seront ajoutées à Heptane, en les sélectionnant de manière à ce qu'elles soient représentatives des processeurs utilisés dans le monde du temps-réel embarqué (e.g. ARM, PowerPC). Un minimum de 3 mois est à compter pour chaque cible, pour laquelle la personne recrutée devra :
  - comprendre le fonctionnement interne du processeur à partir de sa documentation et en complément le cas échéant avec un contact avec des spécialistes de l'architecture des processeurs (e.g. projet Caps, Inria-Rennes, projet Apara, IRIT Toulouse)
  - développer les fichiers de description d'architecture à fournir en entrée du logiciel Salto
  - adapter si nécessaire les modules Heptane de prise en compte de la micro-architecture
  - tester la validité des modifications de l'analyseur par confrontation avec des exécutions réelles des programmes. Ce point peut être d'une durée non négligeable, car il nécessite la mise en place d'outils d'instrumentation si l'on dispose à l'Inria Rennes de l'architecture ciblée, ou l'installation de simulateurs "cycle-true" dans la négative.
- **Phase 4 :** Changement du logiciel d'évaluation des résultats pour un logiciel non propriétaire. Le logiciel Maple, utilisé en phase ultime du calcul de WCET sera remplacé par un outil d'évaluation d'expressions symboliques non commercial (par exemple Maxima ou Ginac) pour augmenter l'impact d'Heptane dans la communauté.

7 Personnels impliqués

L'encadrement scientifique de l'ingénieur chargé des développements sera assuré par Isabelle Puaut (maître de conférences INSA, membre de l'action ACES), ayant participé au développement de la première version du logiciel.

L'organisation des développements et les aspects méthodes et qualité du code seront supervisés par Jean-Paul Routeau (ingénieur de recherche CNRS, membre de l'action ACES).

Références

[Col01] Antoine Colin. *Estimation de temps d'exécution au pire cas par analyse statique et application aux systèmes d'exploitation temps-réel.* Thèse de doctorat, Université de Rennes I, October 2001.

[CP00] A. Colin and I. Puaut. Worst case execution time analysis for a processor with branch prediction. *Real-Time Systems*, 18(2-3):249–274, May 2000.

[CP01a] A. Colin and I. Puaut. A modular and retargetable framework for tree-based wcet analysis. In *Proc. of the 13th Euromicro Conference on Real-Time Systems*, pages 37–44, Delft, The Netherlands, June 2001.

[CP01b] A. Colin and I. Puaut. Worst-case execution time analysis of the RTEMS real-time operating system. In *Proc. of the 13th Euromicro Conference on Real-Time Systems*, pages 191–198, Delft, The Netherlands, June 2001.

[PB00] P. Puschner and A. Burns. A review of worst-case execution-time analysis. *Real-Time Systems*, 18(2-3):115–128, May 2000. Guest Editorial.