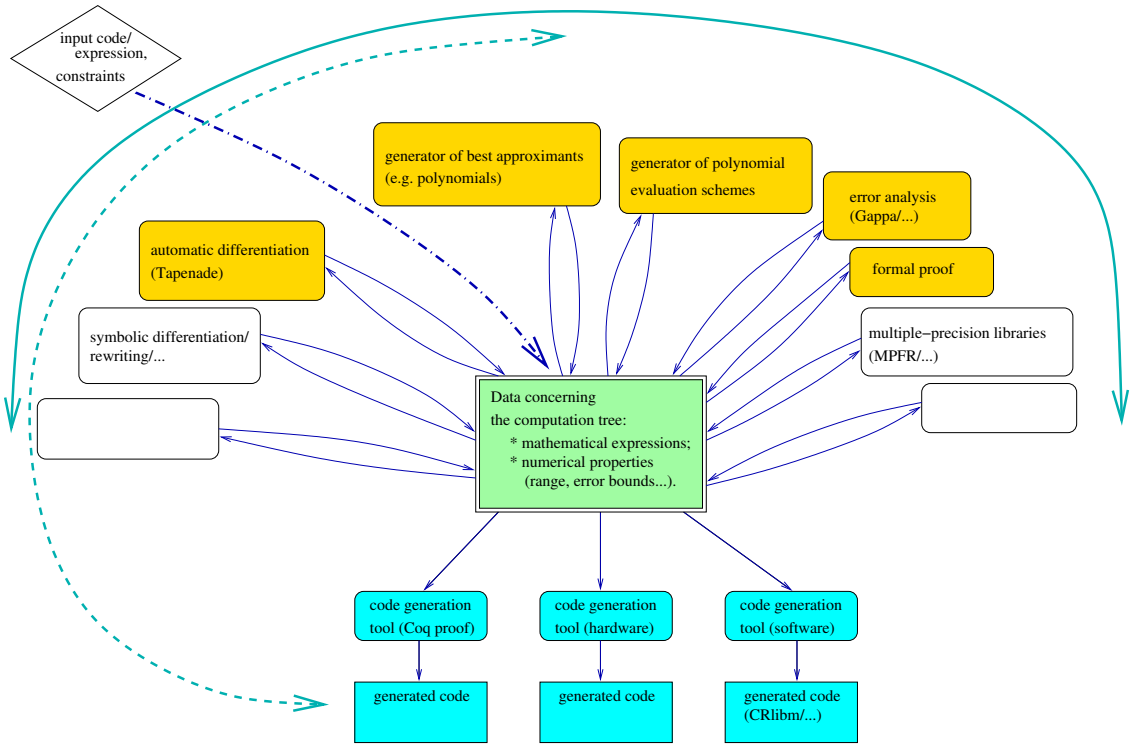
Définition du fonctionnement et des résultats du projet EVA-Flo

Compte rendu des discussions du 12 avril 2007
Première réunion EVA-Flo : 11-12 avril 2007

Secrétaire de séance : Nathalie Revol

Le résultat de cette discussion est le schéma suivant d'organisation d'EVA-Flo : le losange en haut à gauche est le point d'entrée, en vert au centre se trouve la représentation interne sur laquelle agissent la nébuleuse d'outils (développés par nous – en jaune – ou non) ; les résultats possibles sont représentés en turquoise.



Spécifications d'EVA-Flo

Chef d'orchestre ?

Matthieu : qu'est-ce qu'EVA-Flo ?

Nathalie rappelle le schéma avec un objet en représentation interne (LEMA) sur lequel agissent de multiples outils (voir ci-dessus schéma obtenu au cours de cette discussion).

Christoph pose la question du chef d'orchestre, c'est-à-dire du choix et de l'enchaînement des outils.

Nathalie : c'est prématuré, à élaborer dans le cadre du projet EVA-Flo.

David : comment spécifier ce que l'utilisateur veut comme résultat final ?

Christoph : bonne question, en effet tous les outils n'ont pas vocation à agir sur tous les problèmes posés en entrée.

Nathalie : on peut ajouter des attributs spéciaux *contraintes à respecter*.

Vincent : c'est une solution, est-ce la meilleure ? Il faudrait faire des tests pour savoir comment faire. Il y aura aussi interactivité, il faudra réfléchir pour savoir comment. Tests à faire à la main : comme les outils laissent des traces, on peut essayer différentes possibilités et les rejouer. À faire avec un script ou un makefile.
Il faudrait aussi refaire boucler à partir du code généré : par exemple du code généré pour tout un intervalle est compilé et exécuté sur un bout de l'intervalle. On établit un profil et on re-compile avec ces indications de profil : cela permet de gagner 20% sur un Opteron et 0% sur un Pentium.

Nathalie reformule : la méthodologie de Vincent consiste, dans un premier temps, à

(a) traduire le problème dans le langage d'entrée

(b) faire tourner à la main, en choisissant lesquels, les différents outils

pour satisfaire les contraintes que l'on a en tête.

Vincent : oui, au début on fera tout à la main et progressivement on écrira des scripts.

Ce qu'EVA-Flo saura faire, comment

David : demander à son ordinateur "génère-moi la fonction F en 64 bits..." juste en appuyant sur un bouton n'est pas réalisable.

ristoph & Sylvain Ch. : cela leur paraît réalisable dans un proche avenir, sauf les questions de pire cas.

Vincent envisage même de mettre un outil de recherche de pires cas dans la boîte à outils.

Matthieu : quels sont les utilisateurs ?

Guillaume : ST et en particulier l'équipe de compilation.

Christoph : ça arrive partout.

Matthieu : est-ce qu'EVA-Flo sera distribué aux utilisateurs ou est-ce que nous l'utiliserons pour répondre à leurs questions ?

Nathalie : dans l'idéal ou à terme, ce sera distribué.

Sylvain Ch. : ce sera à usage interne, et cela le restera dans un premier temps ; pour pouvoir le distribuer il faudra poursuivre un développement qui ne concerne que l'interface.

Nathalie rappelle que nous avons obtenu un an d'ingénieur, qui peut permettre de développer l'interface.

Sylvain Ch. : on a aussi besoin de pouvoir faire de la publicité pour chaque boîte prise séparément, avant d'en faire pour le gros outil.

Nathalie : cf. Maple, (presque) personne ne connaît les algorithmes particuliers, son intérêt est qu'il intègre de nombreux outils.

Quelques questions à propos de Fluctuat

Nathalie : Fluctuat est-il distribué ?

Matthieu : oui, cela évite à ses développeurs de se palucher les codes incompréhensibles des utilisateurs. On accompagne les utilisateurs (par une formation...) et on ajoute les fonctionnalités demandées dans Fluctuat plutôt que de modifier le code utilisateur.

Nathalie : comment s'inscrit Fluctuat dans le schéma EVA-Flo ? Pourrait-il lire et écrire dans ce langage ?

Matthieu : non, il faudrait traduire et on ne veut pas passer du temps sur cet aspect-là.

Nathalie : donc on n'a pas accès aux informations calculées par Fluctuat, telles que *ce résultat appartient à tel intervalle* ?

Matthieu : non, ces informations sont faciles à récupérer, mais pas question de changer le langage en entrée qui est le C. Nous avons le projet de réécrire tout Fluctuat d'ici quelques années. Ce qui est possible en attendant est de traduire le langage en entrée en C accompagné de commentaires et de retourner pour chaque résultat des informations complémentaires.

Vincent : l'utilisation de Fluctuat est valide uniquement sur certains types de codes, quelles sont les limitations ?

Matthieu : précisions simple et double uniquement.

Vincent : pas de précision étendue ?

Matthieu : non, parce que nous travaillons dans le contexte des systèmes embarqués.

Nathalie : quelle est la taille des codes traités ?

Matthieu : de quelques milliers à quelques dizaines de milliers de lignes de code. Par exemple, le système central de commande de vol d'Airbus fait 130 000 lignes.

Christoph est surpris par un si petit nombre, puisque pour un téléphone portable il faut compter de l'ordre du million de lignes de code.

Matthieu : un des plus gros codes traités est le module de ravitaillement de la station spatiale, module lancé par Ariane. La partie difficile est de trouver la station pour l'arrimage, cette partie comporte environ un million de lignes de codes : calculs de positions et de trajectoires, et c'est elle qui est hautement critique.

? : par rapport à CRlibm, il y a moins de demande côté précision, mais il y a beaucoup plus de lignes de code à traiter.

David : quelles sont les structures de contrôle en LEMA ?

Vincent : c'est récursif mais on peut donner des indications sur la génération des boucles.

Nathalie : dans un code long, est-ce que Fluctuat saurait identifier les petits passages qui posent problème et le signaler pour que les autres outils bossent dessus ?

Matthieu : oui et non. Fluctuat bosse sur tout le code, il sait dire à quel moment il ne sait pas faire mieux que "l'erreur est inférieure à $+\infty$".
La description d'une fonction doit comporter énormément de commentaires sur les spécifications et l'algorithme et des explications pour chaque ligne de code.

Christoph : pourquoi ne pourrait-on pas générer le code à partir de spécifications ?

Matthieu : parce qu'il est difficile de faire changer les modes de fonctionnement.

Christoph : même l'argument du prix de développement ne suffit pas ?

Matthieu : on ne nous a pas attendu pour faire voler des avions ! Le processus de développement est basé sur des outils tels que SCADE qui est un compilateur certifié, ou Simulink.

Nicolas L. : la conclusion est que nous devons travailler sur de petits codes, puisqu'on ne convaincra pas les utilisateurs qui ont déjà de gros codes.

Matthieu : CRlibm est un bon domaine d'utilisation, mais pas les systèmes embarqués. Pour Fluctuat, les utilisateurs sont ceux qui ont un produit fini et qui ont besoin de le valider, pas ceux qui conçoivent le produit.

Nathalie : si on désactive la génération de code, on devrait pouvoir valider du code sans le modifier.
On en revient donc à la question du contrôle dans EVA-Flo.

Différentiation automatique

Nathalie explique pourquoi on a besoin de différentiation automatique en arithmétique par intervalles. Une première justification est que l'on utilise des modèles de Taylor pour réduire les problèmes de décorrélation des variables et que l'on a besoin de pouvoir calculer les développements de Taylor. Une seconde justification est que certains algorithmes, comme la méthode de Newton, requièrent la connaissance de la différentielle de la fonction étudiée.

Laurent : il faut savoir évaluer le code correspondant à la représentation (dans la boîte centrale du schéma EVA-Flo) ; il faut mettre un interpréteur.

Vincent : ou générer de façon intermédiaire du code qu'on fera tourner.

Vincent : est-ce que Tapenade génère des dérivées de façon certifiée ?

Laurent : non.

Vincent : est-ce que le générateur produit des dérivées qui tiennent compte des erreurs d'arrondi et donc qui retournent, à l'évaluation, des intervalles.

Laurent : non, ou alors il faut surcharger.

Christoph : pourquoi ne pas utiliser un générateur rapide et non certifié et effectuer une preuve *a posteriori*, lente.

Laurent : Tapenade sait fournir des dérivées non certifiées pour la certification : l'approche par surcharge semble bien indiquée, c'est-à-dire utiliser le principe d'Adol-C pour écrire une librairie surchargée (chaque opérateur est surchargé et calcule, en même temps que l'opération, la différentielle de cette opération par rapport aux variables d'entrée).

Nathalie : ce n'est peut-être qu'une élucubration, mais il me semble qu'on pourrait utiliser la différentiation automatique pour estimer ou majorer un conditionnement (qui est après tout une constante de Lipschitz), en évaluant la différentielle sur tout un intervalle d'entrée.

Laurent : dans ce cas, la transformation source à source sera moins efficace que la surcharge : si la dérivée d'une instruction est une instruction, la transformation source à source est meilleure, si la dérivée d'une instruction est beaucoup d'instructions, la surcharge est meilleure.

Nathalie : pourtant la reverse différentiation donne des expressions plus courtes dans ce cas, ce qui devrait donner de meilleures évaluations en arithmétique par intervalles ?

Laurent : bonne remarque. Est-ce que la fonction est une expression ou un sous-programme ? Si c'est une expression, alors utiliser la différentiation symbolique. Si c'est un sous-programme, alors utiliser la différentiation automatique. Dans ce cas, comment fait-on des intervalles en mode reverse ?

Nicolas L. : à propos de l'utilisation de la différentiation automatique pour estimer un conditionnement, des travaux existent à ce sujet, mais par échantillonnage et donc non certifiés. Cf. Miller qui utilise des différences divisées : *W. Miller and C. Wrathall. Software for roundoff analysis of matrix algorithms. Academic Press. 1980.* Higham résume très bien la méthode utilisée, et donne également d'autres références concernant les travaux de Miller, à la page 484 de ASNA (Accuracy and Stability of Numerical Algorithms). Voir aussi les travaux de Iri et Kubota (Philippe doit avoir les papiers cités dans ASNA). Voir aussi les articles sur les approches en erreur absolue de Linnainnmaa et en erreur relative de Larson et Sameh, jeter aussi un coup d'oeil sur ce que Philippe en racontait dans son HDR.

Laurent : essai sur un exemple :

programme $r = f(x, y)$:

$$\begin{aligned} a &= x \times x \\ b &= y \times y \\ r &= a + 2 \times b \end{aligned}$$

programme avec reverse AD : $\bar{x} = 1.0$:

$$\begin{aligned} \bar{a} &= \bar{r} \\ \bar{b} &= 2\bar{r} \\ \boldsymbol{y} &= 2\boldsymbol{y}\bar{b} \\ \boldsymbol{x} &= 2\boldsymbol{x}\bar{a} \end{aligned}$$

où les intervalles sont écrits avec cette police de caractères : $\boldsymbol{x}$.
Cela ne devrait pas poser de problème grâce à la surcharge des opérateurs, on obtient bien un encadrement du gradient. C'est jouable en linkant Tapenade avec une librairie d'arithmétique par intervalles.

Vincent : est-ce que Tapenade peut prendre en entrée du MathML ?

Laurent : Tapenade lit une séquence d'instructions, c'est-à-dire du style impératif.
La question est de déterminer si on a besoin d'avoir une représentation interne en fonctionnel.

Vincent : oui, pour pouvoir effectuer des preuves.

Laurent : la question est alors de savoir si on peut traduire du fonctionnel en impératif.

Vincent : oui, en SSA.

Laurent : pour Tapenade, pas de problème pour lire du SSA.

Vincent : et a-t-on du SSA en sortie ?

Laurent : essai sur un exemple :

programme $r = f(x, y)$:

$$\begin{aligned} a &= x \times x \\ b &= y \times y \\ r &= a + 2 \times b + x \end{aligned}$$

programme avec reverse AD : $\bar{x} = 1.0$:

$$\begin{aligned} \bar{a} &= \bar{r} \\ \bar{b} &= 2\bar{r} \\ \boldsymbol{y} &= 2\boldsymbol{y}\bar{b} \\ \boldsymbol{x} &= 2\boldsymbol{x}\bar{a} + \bar{x} \end{aligned}$$

La réponse est non, mais on pourra traduire en SSA.

Laurent : et quid des boucles ?

Vincent : c'est récursif.

Laurent : dans ce cas il faut traduire les boucles en SSA et c'est aussi difficile.

Vincent : c'est possible sur des portions, sur des *straight-line programs*.

Laurent : a-t-on vraiment besoin de SSA ?

Vincent : oui (mais j'ai oublié la raison, note de NR).

Laurent : pour du reverse AD, avoir du SSA est intéressant pour économiser des *push* - *pop*.

? : SSA est généré par gcc et éliminé ensuite, c'est-à-dire que même avec un code SSA, le code compilé remet de l'écrasement.

Nicolas L. : pourquoi Tapenade intervient au niveau de la boîte centrale du schéma d'EVA-Flo plutôt que d'être appelé par chaque outil ?

Nathalie : pour éviter de multiples appels à l'AD.

LH et CL : pourquoi ne pas ajouter une boîte "différentiation symbolique" ?
La question est de savoir si elle fera du reverse AD pour factoriser les expressions dans le cas où on a beaucoup d'entrées et peu de sorties.

Questions diverses

question : fonctionnel versus impératif : apparemment il y a consensus pour adopter du fonctionnel pour la représentation interne.
Mais la question XML versus MathML se pose aussi : OK pour XML, ? ? ? pour MathML et le problème de la représentation par lambda-calcul ? C'est pour ceux qui font de la preuve.

question : format d'entrée ?

question : garde-t-on un schéma avec une nébuleuse d'outils branchés sur la boîte centrale (l'objet manipulé), ou le structure-t-on, par exemple en rattachant l'AD à un outil particulier ? On reste avec la nébuleuse.

question : contrôle.

question : peut-on fusionner des boîtes ? Peut-on avoir des courts-circuits, c'est-à-dire des boîtes en communication directe et non via le centre ? Oui pour les communications directes si l'information communiquée ne peut pas être utile aux autres.

Conclusions, à faire pour la prochaine fois

compte rendu : sur la page Web d'EVA-Flo.

rapport : à compléter par chaque équipe.

prochaine réunion : octobre-novembre à Perpignan.

à faire d'ici là : un prototype en MathML, que Sylvain Ch. et Christoph pourront tester (générer en sortie du MathML).

à Sophia : Laurent a besoin de la forme XML / MathML, il pourra alors la traduire en langage compréhensible par Tapenade, qui fera de la différentation en mode inverse et qui générera une expression XML / MathML avec des annotations intervalles.

à faire à Perpignan : David et Sylvain Co. réfléchissent aux spécifications de leurs arithmétiques exotiques, Marc et Matthieu à la vérification, Philippe et Nicolas L. à comment faire en sorte que leur méthode devienne l'une des boîtes à outils dans la nébuleuse des boîtes à outils (en jaune sur le schéma).