Une interface commune pour les treillis numériques

ACI Sécurité Informatique APRON
Réunion plénière du 8 novembre 2006

Bertrand Jeannet
INRIA Rhône-Alpes

Plan

1. Introduction
2. Fonctionnalités offertes
3. Architecture générale de l'interface
4. État d'avancement et perspectives

I. Introduction

Analyse de programme par Interprétation Abstraite 1/4

Un cas simple:

- programme constitué d'une seule procédure
- pas d'allocation dynamique
- types scalaires (booléens, types énumérés, entiers, réels, flottants)

Modélisation:

- Un ens. $\boldsymbol{V}$ de variables
- Un graphe $(\boldsymbol{K}, \boldsymbol{I})$ avec
  - $\boldsymbol{K}$: ens. des points de contrôle
  - $\boldsymbol{I} : \boldsymbol{K} \times \boldsymbol{K} \longrightarrow \mathbf{Cond} \cup \mathbf{Instr} \cup \{\bot\}$:
    associe à un arc une condition ou une instruction

Analyse de programme par Interprétation Abstraite 2/4

Sémantique opérationelle:

- Espace d'état $S = K \times \mathbf{Env}$
  où $\mathbf{Env} = (V \longrightarrow \mathbb{R}) \simeq \mathbb{R}^n$ définit la valuation des variables
- Sémantique d'une condition: $[\![\mathbf{cond}]\!] : \mathbf{Env} \longrightarrow \mathbb{B}$
  Sémantique d'une instruction: $[\![\mathbf{instr}]\!] : \mathbf{Env} \longrightarrow \mathbf{Env}$
- Relation $\longrightarrow$ entre états:

$$\frac{I(k,k') = \mathbf{cond} \quad [\![\mathbf{cond}]\!](e) = true}{(k,e) \longrightarrow (k',e)}$$

$$\frac{I(k,k') = \mathbf{instr} \quad [\![\mathbf{instr}]\!](e) = e'}{(k,e) \longrightarrow (k',e')}$$

- Système dynamique discret $(S, S_{\mathbf{init}}, \longrightarrow)$

Analyse de programme par Interprétation Abstraite 3/4

Objectif: analyser des propriétés d'invariance

invariant global : ens. d'états $\boldsymbol{X \subseteq S}$

espace des propriétés: $\boldsymbol{\wp(S) = \wp(K \times \mathbf{Env}) \simeq K \longrightarrow \wp(\mathbf{Env})}$

Sémantique collectrice:

Sémantique d'une condition: $[\![\mathbf{cond}]\!] : \wp(\mathbf{Env}) \longrightarrow \wp(\mathbf{Env})$

Sémantique d'une instruction: $[\![\mathbf{instr}]\!] : \wp(\mathbf{Env}) \longrightarrow \wp(\mathbf{Env})$

$$\boldsymbol{X^k \supseteq X^k_{\mathbf{init}}}$$

$$\boldsymbol{X^{k'} \supseteq [\![I(k,k')]\!](X^k)}$$

$$\Leftrightarrow$$

$$\boldsymbol{X^{k'} = X^{k'}_{\mathbf{init}} \cup \bigcup_{(k,k')} [\![I(k,k')]\!](X^k)}$$

Analyse de programme par Interprétation Abstraite 4/4

L'espace des propriétés concrètes $C = \wp(env)$ trop complexes !
Approximation statique:

- On substitue à C un treillis abstrait A plus simple

$$\wp(\mathbf{Env}) \leftrightarrows A$$

- On transpose les équations dans A:

$$Y^{k'} = Y^{k'}_{\mathbf{init}} \sqcup \bigsqcup_{(k,k')} [\![I(k,k')]\!]^\sharp(Y^k)$$

Approximation dynamique: widening

Implémentation d'un treillis abstrait A

- Représentation des éléments de $\boldsymbol{A}$, avec
  $\gamma : \boldsymbol{A} \longrightarrow \wp(\boldsymbol{V} \longrightarrow \mathbb{R})$
- Opérations logiques/ensemblistes:
  - conjonction (borne inf), disjonction (borne sup)
  - test du vide et de l'inclusion
  - introduction/élimination d'une variable
- Abstraction dans $\boldsymbol{A}$ de la sémantique des instructions
  - $[\![\textbf{cond}]\!] : \boldsymbol{A} \longrightarrow \boldsymbol{A}$
  - $[\![\textbf{instr}]\!] : \boldsymbol{A} \longrightarrow \boldsymbol{A}$

  dépend du langage analysé
- ...

Motivations pour une interface commune 1/2

Analyse des variables numériques d'un programme: nombreux treillis abstraits implémentés:

- Intervalles (. . .)
- Egalités linéaires (Müller-Olms & Seidl)
- Combinaison des intervalles et égalités (Venet)
- Zones et Octogones (Miné, Bagnara & al)
- Octaèdres (Cortadella)
- Polyèdres convexes (NewPolka, Parma, PolyLib, CRI)
- "Templates" (VMCAI'05)
- Égalités polynomiales (Müller-Olms & Seidl)
- Inégalités polynomiales (Bagnara & al, SAS'05)
- Ellipsoïdes (Férêt)
- Inégalités $\neq$ entre variables (Peron, VMCAI'07)

Motivations pour une interface commune 2/2

Fonctionnalités

- Noyau minimal de fonctions offertes
  (meet, join, affectation par expression linéaire, ...)
- + Fonctions plus spécifiques pour des applications particulières (ex: PolyLib, CRI)

Mais API très diverses !

- Certaines fonctionnalités de base peuvent manquer
- Certaines interfaces trop liées aux structures de données internes (ex: NewPolka)

$\Rightarrow$ Complique la diffusion de ces implémentations
$\Rightarrow$ Difficile de comparer

- l'efficacité de 2 implémentations du même treillis
- la précision de deux treillis différents

Objectifs pour une interface commune 1/2

- Identifier les fonctionnalités de base que doit fournir (l'implémentation d') un treillis numérique
- Élaborer une interface détaillée (types de données, signatures des fonctions, sémantique associée)
- Implémenter une telle interface pour les bibliothèques maintenues par les membres du projet
- Diffuser auprès de la communauté analyse statique (utilisateurs et implémenteurs)

Objectifs pour une interface commune 2/2

Sous les contraintes suivantes:

- Satisfaire les besoins de tous les membres du projet, tout en restant générique !
- Souci de simplicité et de minimalité, sans trop pénaliser les performances
- Minimiser le travail d'adaptation d'une librairie existante

Initiatives comparables en analyse/vérification

L'interface SIS/VIS pour manipuler les BDDs (Université de Californie, Berkeley)

Analyseurs génériques:

- Se préoccupent du calcul de point-fixe
- Souvent un peu spécialisés:
  - en fonction du type d'analyse (ex: analyse type bitvector)
  - incorporent la sémantique du langage source (Marktoberdorf'98 Generic Abstract Interpreter)

II. Principes généraux

Notion de niveau d'interface

On a pour objectifs: performance des implantations, confort de l'utilisateur, et non-duplication de code entre $\neq$ bibliothèques

- Niveau 0: on se préoccupe des performances, et de la précision
- Niveau 1: on se préoccupe du confort, et des fonctionnalités factorisables entre les $\neq$ domaines abstraits

Notion de niveau d'interface

On a pour objectifs: performance des implantations, confort de l'utilisateur, et non-duplication de code entre $\neq$ bibliothèques

- Niveau 0: on se préoccupe des performances, et de la précision
- Niveau 1: on se préoccupe du confort, et des fonctionnalités factorisables entre les $\neq$ domaines abstraits

Niveau 0:

- En prise directe avec la librairie sous-jacente
- Contient toutes les opérations spécifiques au domaine abstrait (ne pouvant être partagées)
- Interface minimale, sauf si avantage algorithmique fort à y inclure une combinaison

Notion de niveau d'interface

On a pour objectifs: performance des implantations, confort de l'utilisateur, et non-duplication de code entre $\neq$ bibliothèques

- Niveau 0: on se préoccupe des performances, et de la précision
- Niveau 1: on se préoccupe du confort, et des fonctionnalités factorisables entre les $\neq$ domaines abstraits

Niveaux supérieurs: fonctions factorisables
Deux exemples envisagés:

- abstraction d'expressions non linéaires en expressions linéaires d'intervalle
- Appel automatique aux opérations de redimensionnement et de permutation pour calculer $\boldsymbol{P}(\boldsymbol{x},\boldsymbol{y}) \sqcap \boldsymbol{Q}(\boldsymbol{z},\boldsymbol{y})$

Notion de niveau d'interface

Combinaison de treillis/bibliothèques: deux interfaces/bibliothèques de niveau 0 pourront être combinées pour offrir une nouvelle bibliothèque de niveau 0.

- implémentations différentes du même domaine
- produit cartésien ou réduit de domaines
- décomposition de polyèdres en produit cartésien de polyèdres (thèse de D. Merchat)

Langage de programmation

La version de référence est la version C de l'interface

- C s'interface aisément avec la plupart des langages
- La plupart des librairies existantes sont en C (ou C++)

Une version OCaml a également été élaborée (ENS, INRIA)
L'interfaçage OCaml/C est générique pour toutes les librairies

Architecture générale (1/3)

Compatibilité avec les threads

Un contexte d'appel qui sera explicitement passé à chaque fonctions afin d'assurer:

- La transmission de données spécifiques à chaque librairie (options non standard, espace de travail, ...)
- La transmission des options (leviers de réglages des algos, de la précision)
- La gestion des exceptions (`not_implemented`, `invalid_argument`, `overflow`, `timeout`, `out_of_space`)

Architecture générale (2/3)

Gestion mémoire:

- Pas de ramasse-miettes dans l'interface C
  (hormis comptage de référence pour certains types)
- Utilisation des mécanismes du runtime pour les interfaces OCaml, Prolog, ...

Mécanismes d'interruption en cas de temps trop long ou de consomnation mémoire trop élevée (`timeout`, `out_of_space`, `overflow`)

Signatures fonctionnelles et impératives (effet de bord) supportées

Architecture générale (3/3)

Représentation des nombres:

- En interne, propre à chaque bibliothèque
- Dans l'interface, choix entre
  - rationels multi-précision GMP (arithmétique exacte)
  - `double` (arithmétique flottante)

III. Fonctionnalités offertes

(Au niveau 0 de l'interface)

Fonctionnalités offertes (1/6)

Sémantique d'une valeur abstraite:
sous-ensemble $\boldsymbol{X} \subseteq \mathbb{N}^{\boldsymbol{p}} \times \mathbb{R}^{\boldsymbol{q}}$

Les valeurs abstraites sont typées selon leur dimensionalité

Les dimensions sont donc typées entier ou réel
Elles sont numérotées de $\mathbf{0}$ à $\boldsymbol{p} + \boldsymbol{q} - \mathbf{1}$

Autres types de données manipulés:

- nombres (rationnels ou flottants)
  intervalles
  coefficients (nombre ou intervalle)
- expressions linéaires (d'intervalle)
- contraintes
  rayons

Fonctionnalités offertes (2/6)

Contrôle de la représentation interne:

- Forme canonique
- Forme minimale (en terme d'occupation mémoire)
- Notion d'approximation laissé au libre choix de l'implémentation (prise en compte des entiers ou non, ...)

Impression:

- d'une valeur abstraite
- de la différence de deux valeurs abstraites

Sérialisation/désérialisation binaire vers une zone mémoire

Fonctionnalités offertes (3/6)

Constructeurs: $\bot(\boldsymbol{p}, \boldsymbol{q})$, $\top(\boldsymbol{p}, \boldsymbol{q})$, abstraction d'un hypercube, abstraction d'un polyèdre convexe

Tests:

- du vide, de l'univers
- inclusion, égalité
- inclusion d'une dimension dans un intervalle, satisfaction d'une contrainte linéaire

Extraction de propriétés:

- intervalle de variation d'une dimension, d'une expression linéaire, dans une valeur abstraite
- conversion vers un hypercube, un polyèdre convexe

Fonctionnalités offertes (4/6)

Opérations de treillis et variations:

- $\sqcup$, $\sqcap$ de deux valeurs, d'un tableau de valeurs
- intersection avec une plusieurs contraintes linéaires
- ajout de rayons (opérateur de passage du temps généralisé)

Affectation/Substitution:

- d'une dimension par une expression linéaire
- en parallèle de plusieurs dimensions par des expressions linéaires

Élargissement avec seuil (ensemble de contraintes linéaires)

Fonctionnalités offertes (5/6)

Projection/Oubli d'une ou plusieurs dimensions
à dimensionalité constante

Ajout/Retrait/Permutation de dimensions

Expansion et pliage de dimensions
(utiles pour l'abstraction de tableaux, par ex)

Clôture topologique (relaxation des contraintes strictes)

Fonctionnalités offertes (6/6)

Représentation d'une valeur abstraite:
Abstract0=(Manager,void*)

Le contexte d'appel explicite Manager:

- est alloué par la bibliothèque sous-jacente
- contient une table de pointeurs vers les fonctions du niveau 0 offertes par celle-ci
- détermine donc le type effectif d'une valeur abstraite

validité des arguments vérifiés avant l'appel des fonctions effectives
⇒ soulage l'implanteur d'une librairie sous-jacente

III bis. Fonctionnalités offertes

Au niveau 1 de l'interface
Ajoutées à toute librairie se connectant au niveau 0

Fonctionnalités offertes (1/3)

Notion de variable:

Les dimensions sont remplacées par des variables
Les variables sont définies par un type C générique
(chaîne de caractère, type structuré, ...)
équipé des opérations `compare, copy, free, to_string`

Notion d'environnement:

Les environnement gère la correspondance
dimensions du niveau 0 $\leftrightarrow$ variables du niveau 1

Sémantique d'une valeur abstraite:

sous-ensemble $\boldsymbol{X} \subseteq (\boldsymbol{V} \longrightarrow \mathbb{N} \cup \mathbb{R})$

Représentation:

Abstract1=(environnement, Abstract0)
Expr1=(environnement, Expr0) ...

Fonctionnalités offertes (2/3)

Opérations sur les environnements: (niveau utilisateur)

- création, fusion, destruction
- ajout/retrait/renommage de variables

Typage dynamique:

- Pour les opérations binaires (borne sup, test d'inclusion), on vérifie que les environnements sont les mêmes
- Pour les opérations mixtes (Abstract1 $\times$ Expr1), on vérifie que l'env. de l'expression est un sous-environnement de la valeur abstraite, et on retaille si nécessaire

Fonctionnalités offertes (3/3)

Projection/Oubli d'une ou plusieurs variables
à environnement constant

Ajout/Retrait/Renommage de variables
(engendrant une modification corresp. de l'env. associé)
Changement d'environnement (pouvant combiner retrait et ajout de variables)

Expansion et pliage de variables
(engendrant une modification corresp. de l'env. associé)

Conclusion: confort largement accru !

IV. Avancement et perspectives

État d'avancement 1/2

Document de travail synthétisant discussions et choix

Programmation:

Interface niveau 0:

- Implantation des types de données et opérations communes (7000 lignes de C)
- Interfaçage de 4 bibliothèques:
  - treillis d'intervalle (écrit pour l'occasion)
  - Octagons (octogones)
  - NewPolka (polyèdres convexes)
  - PPL, partie polyèdres convexes

Interface niveau 1: 3000 lignes de C

Interface OCaml: 3800 lignes de source CamlIDL

Distribution sous license LGPL, déposé à l'APP

État d'avancement 2/2

Connexion de l'outil NBac à l'interface (au niveau 1)
Connexion imminente de l'outil Astree ?

Premiers utilisateurs extérieurs

Projets

Interfacer d'autres bibliothèques/domaines:

- NewPolka, surcouche pour ne faire que des égalités linéaires
- PPL, partie congruences linéaires ($\sum_i a_i x_j \equiv b[c]$)
- Treillis des inégalités $\neq$, [Peron & Halbwachs 07]
- Omega (arithmétique de Presburger)
- NewPolka, ajout d'opérations "alternatives"
  (join moins précis mais plus efficace ([Manna and al], ...)

Ajout des expressions et contraintes non linéaires dans l'interface

Diffusion dans la communauté analyse statique:

- Auprès des utilisateurs
- Mais aussi des implanteurs