The APRON Library

Bertrand Jeannet and Antoine Miné

INRIA, CNRS/ENS

CAV'09 conference
02/07/2009

Context : Static Analysis

What is it about ?

Discover properties of a program statically and automatically.

How : Abstract Interpretation

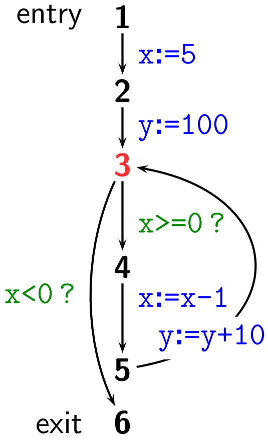
Theoretical tool to design and compare analyses that :

- always terminate
- are sound (no behavior is omitted)
- are approximate (solve undecidability and efficiency issues)

Applications

- compilation and optimisation
  - e.g., alias analysis
- verification and debugging
  - infer invariants
  - prove properties

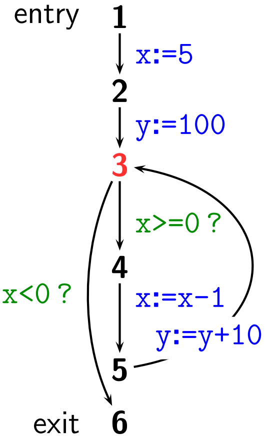
Program Analysis by Abstract Interpretation 1/3



Program Analysis by Abstract Interpretation 1/3

Collecting Semantics :

Collects reachable environments at each control point

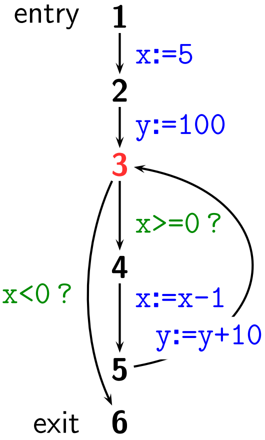


- $\mathbf{Env} = \{\mathrm{x}, \mathrm{y}\} \rightarrow \mathbb{Z}$
- Invariants $\boldsymbol{X_i} \in \wp(\mathbf{Env})$
- Semantics of statements :
  $[\![\mathbf{stm}]\!] : \wp(\mathbf{Env}) \rightarrow \wp(\mathbf{Env})$

Program Analysis by Abstract Interpretation 1/3

Collecting Semantics :

Collects reachable environments at each control point



- $\mathbf{Env} = \{x, y\} \longrightarrow \mathbb{Z}$
- Invariants $\boldsymbol{X_i} \in \wp(\mathbf{Env})$
- Semantics of statements :
  $[\![\mathbf{stm}]\!] : \wp(\mathbf{Env}) \longrightarrow \wp(\mathbf{Env})$
- Concrete equation system

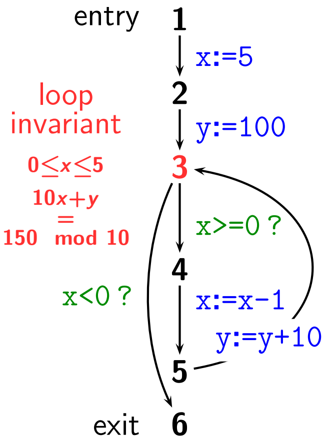
$$\begin{cases} X_1 = \top_{\mathcal{D}} = \mathbf{Env} \\ X_2 = [\![x := 5]\!](X_1) \\ X_3 = [\![y := 100]\!](X_2) \cup \\ \qquad [\![y := y + 10]\!](X_5) \\ X_4 = [\![x \geq 0?]\!](X_3) \\ X_5 = [\![x := x - 1]\!](X_4) \\ X_6 = [\![x < 0?]\!](X_3) \end{cases}$$

The recursive system has a unique least solution (**lfp**)

Program Analysis by Abstract Interpretation 1/3

Collecting Semantics :

Collects reachable environments at each control point



- $\mathbf{Env} = \{\mathrm{x}, \mathrm{y}\} \longrightarrow \mathbb{Z}$
- Invariants $X_i \in \wp(\mathbf{Env})$
- Semantics of statements :
  $[\![\mathbf{stm}]\!] : \wp(\mathbf{Env}) \longrightarrow \wp(\mathbf{Env})$
- Concrete equation system

$$\begin{cases} X_1 = \top_{\mathcal{D}} = \mathbf{Env} \\ X_2 = [\![\mathrm{x} := 5]\!](X_1) \\ X_3 = [\![\mathrm{y} := 100]\!](X_2) \cup \\ \qquad [\![\mathrm{y} := \mathrm{y} + 10]\!](X_5) \\ X_4 = [\![\mathrm{x} \geq 0?]\!](X_3) \\ X_5 = [\![\mathrm{x} := \mathrm{x} - 1]\!](X_4) \\ X_6 = [\![\mathrm{x} < 0?]\!](X_3) \end{cases}$$

The recursive system has a unique least solution (**lfp**)

Program Analysis by Abstract Interpretation 2/3

Undecidability Issues :

- $\mathcal{D} = \wp(\mathbf{Env})$ is not computer-representable
  $[\![\cdot]\!]$ and $\cup$ are not computable
- **lfp** is not computable

Program Analysis by Abstract Interpretation 2/3

Undecidability Issues :

- $\mathcal{D} = \wp(\mathbf{Env})$ is not computer-representable
  $[\![\cdot]\!]$ and $\cup$ are not computable
- **lfp** is not computable

Static approximation : Abstract Domain

- $\mathcal{D}^\sharp$: (simpler) set of computer-representable elements
  - $\gamma : \mathcal{D}^\sharp \to \mathcal{D}$: gives a meaning to abstract elements
- $[\![\cdot]\!]^\sharp$ and $\cup^\sharp$: sound abstract counterparts to $[\![\cdot]\!]$ and $\cup$

Program Analysis by Abstract Interpretation 2/3

Undecidability Issues :

- $\mathcal{D} = \wp(\mathbf{Env})$ is not computer-representable
  $[\![\cdot]\!]$ and $\cup$ are not computable
- **lfp** is not computable

Static approximation : Abstract Domain

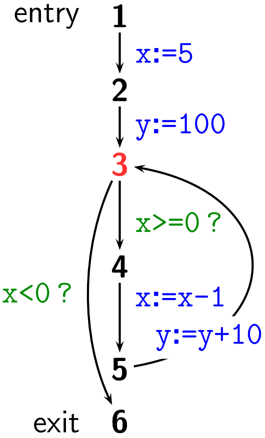
- $\mathcal{D}^\sharp$: (simpler) set of computer-representable elements
  - $\gamma : \mathcal{D}^\sharp \rightarrow \mathcal{D}$: gives a meaning to abstract elements
- $[\![\cdot]\!]^\sharp$ and $\cup^\sharp$: sound abstract counterparts to $[\![\cdot]\!]$ and $\cup$

Dynamic approximation : Widening

- ∇ will ensure termination of **lfp** computation

Program Analysis by Abstract Interpretation 3/3

Solving the abstract equation system



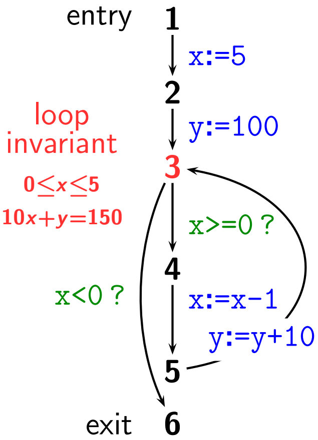
- Static approximation :

$$\begin{cases} \boldsymbol{X}_2^\sharp \supseteq^\sharp [\![\mathtt{x} := \mathbf{5}]\!]^\sharp(\boldsymbol{X}_1^\sharp) \\ \boldsymbol{X}_3^\sharp \supseteq^\sharp [\![\mathtt{y} := \mathbf{100}]\!]^\sharp(\boldsymbol{X}_2^\sharp) \cup^\sharp \\ \qquad\quad [\![\mathtt{y} := \mathtt{y} + \mathbf{10}]\!]^\sharp(\boldsymbol{X}_5^\sharp) \\ \boldsymbol{X}_4^\sharp \supseteq^\sharp [\![\mathtt{x} \geq \mathbf{0?}]\!]^\sharp(\boldsymbol{X}_3^\sharp) \\ \boldsymbol{X}_5^\sharp \supseteq^\sharp [\![\mathtt{x} := \mathtt{x} - \mathbf{1}]\!]^\sharp(\boldsymbol{X}_4^\sharp) \\ \boldsymbol{X}_6^\sharp \supseteq^\sharp [\![\mathtt{x} < \mathbf{0?}]\!]^\sharp(\boldsymbol{X}_3^\sharp) \end{cases}$$

solved iteratively from initial states

Program Analysis by Abstract Interpretation 3/3

Solving the abstract equation system



- Static approximation :

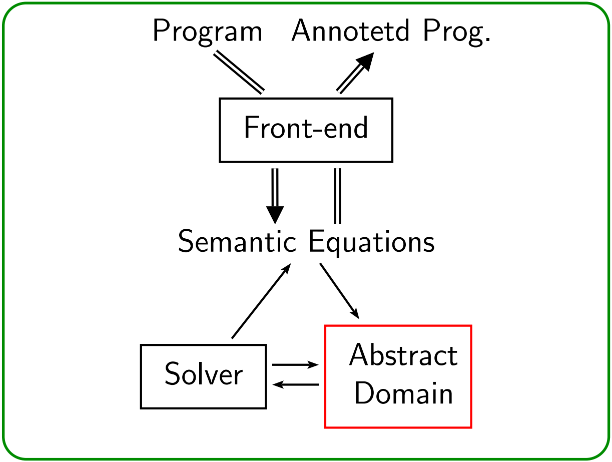
$$\begin{cases} X_2^\sharp \sqsupseteq^\sharp [\![x := 5]\!]^\sharp(X_1^\sharp) \\ X_3^\sharp \sqsupseteq^\sharp [\![y := 100]\!]^\sharp(X_2^\sharp) \cup^\sharp \\ \qquad [\![y := y + 10]\!]^\sharp(X_5^\sharp) \\ X_4^\sharp \sqsupseteq^\sharp [\![x \geq 0?]\!]^\sharp(X_3^\sharp) \\ X_5^\sharp \sqsupseteq^\sharp [\![x := x - 1]\!]^\sharp(X_4^\sharp) \\ X_6^\sharp \sqsupseteq^\sharp [\![x < 0?]\!]^\sharp(X_3^\sharp) \end{cases}$$

  solved iteratively from initial states

- Dynamic approximation :

  applying widening at loop heads

Typical Architecture of a Static Analyzer



Numerical Abstract Domains

Important case : numerical variables

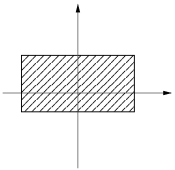
$\mathcal{D}^\sharp$ abstracts $\wp(\mathbf{Env})$ with $\mathbf{Env} = \mathbf{Var} \rightarrow \mathcal{N}$

$\mathcal{N}$: $\mathbb{Z}$ or $\mathbb{R}$

Applications

- Discover numerical properties on program variables
- Prove the absence of a large class of run-time errors
  - Division by zero, overflow, out-of-bound array access
- Parametrize non-numerical analysis
  - Pointer analysis, shape analysis

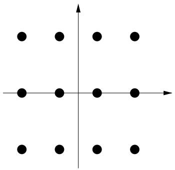
Some Existing Numerical Abstract Domains



Intervals

$X_i \in [a_i, b_i]$

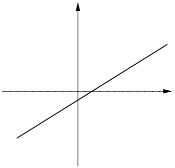
[Cousot-Cousot-76]



Simple Congruences

$X_i \equiv a_i\,[b_i]$

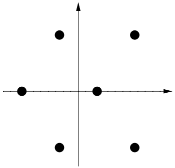
[Granger-89]



Linear Equalities

$\sum_i \alpha_i X_i = \beta$

[Karr-76]

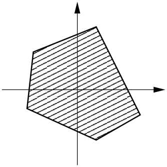


Linear Congruences

$\sum_i \alpha_i X_i \equiv \beta\,[\gamma]$

[Granger-91]

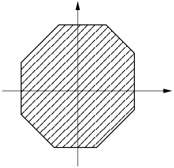
Some Existing Numerical Abstract Domains (cont.)



Polyhedra

$\sum_i \alpha_i X_i \geq \beta$

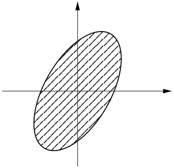
[Cousot-Halbwachs-78]



Octagons

$\pm X_i \pm X_j \leq \beta$

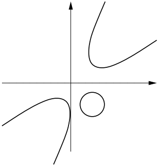
[Miné-01]



Ellipsoids

$\alpha X^2 + \beta Y^2 + \gamma XY \leq \delta$

[Feret-04]



Varieties

$P(\vec{X}) = 0,\ P \in \mathbb{R}[\mathrm{Var}]$

[Sankaranarayanan-Sipma-Manna-04

Numerical Abstract Domains : Implementation

- Representation of abstract elements
- Logical/set opérations :
  - conjunction ($\cap^\sharp$), disjunction ($\cup^\sharp$)
  - emptiness and inclusion test
  - introduction/elimination of variables
- Definition of a concrete semantics
  - $[\![\textbf{expr}]\!] : \mathcal{D} \rightarrow \wp(\mathcal{N})$
  - $[\![\textbf{cond}]\!] : \mathcal{D} \rightarrow \mathcal{D}$
  - $[\![\textbf{instr}]\!] : \mathcal{D} \rightarrow \mathcal{D}$
- And its abstraction in $\mathcal{D}^\sharp$
  - $[\![\textbf{cond}]\!]^\sharp : \mathcal{D}^\sharp \rightarrow \mathcal{D}^\sharp$
  - $[\![\textbf{instr}]\!]^\sharp : \mathcal{D}^\sharp \rightarrow \mathcal{D}^\sharp$
- Widening, Projection, Property extraction, ...

Numerical Abstract Domains : Implementation

Some problems with most implementations

- Have low-level API
  - *e.g.*, former versions of OCTAGON and NEWPOLKA libraries by ourselves...
- That are incompatible (and tight to the domain)
  - *e.g.*, NEWPOLKA and PPL, both implementing convex polyhedra
- Sometimes lack important features
  - *e.g.*, POLYLIB developped by IRISA/Strasbourg university, dedicated to automatic parallelisation of programs
- Often duplicate code

The APRON Library

Goals of the APRON library

- Ready-to-use numerical abstract domains under a common and high-level API
  - Easing the design of new analysers
  - Easing the comparison of domains
- A platform for integration of new domains
  - Toolbox for domain implementors
- Teaching, demonstration, dissemination tools
  - InterProc static analyzer

The APRON Library : Distinctive Features I

Domain-neutral API and concrete data-types

- Supports the concrete semantics
  (safely abstracted by abstract domain)
- Independent of the implementation of domains

Object orientation

- Abstract value = abstract data-type
- Effective underlying domain controlled by a manager
  - Domain-dependent code located in manager allocation
  - User options controlling the precision/efficiency tradeoff

The APRON Library : Distinctive Features II

Example

```
ap_manager_t* man = oct_manager_alloc(...);

ap_abstract1_t val = ap_abstract1_top(man,env);
ap_abstract1_t val =
        ap_abstract1_assign_linexpr(man,val,var,expr);
```

The APRON Library : Distinctive Features III

Support of non-linear, floating-point expressions

- E.g., assignement $y := 2x^2z + \sqrt{yz} \ +_{f,+\infty} \ e$
- Full IEEE754 support (except NaN)

Two-level API

- Level 0 : abstracts $\boxed{\mathbb{Z}^p \times \mathbb{R}^q}$ (implementor level)
  Core functionalities, Efficiency

- Level 1 : abstracts $\boxed{\mathbf{Var} \longrightarrow \mathbb{Z} \cup \mathbb{R}}$ (user level)
  User convenience, Shared services

The APRON Library : Benefits

For domain users

- Higher-level API
  - Variables ("x","y") replace dimensions (0,1)
  - Abstract values typed by environments (["x";"y"])
  - User-convenient functions
    - Change of environment,*.e.g.* from ["x";"y"] to ["y";"z"], involving introduction & elimination of variables (+ permutation of dimensions)
  - Non-linear and floating-point expressions

The APRON Library : Benefits

For domain users

- Higher-level API
  - Variables ("x","y") replace dimensions (0,1)
  - Abstract values typed by environments (["x";"y"])
  - User-convenient functions
    - Change of environment,*.e.g.* from ["x";"y"] to ["y";"z"], involving introduction & elimination of variables (+ permutation of dimensions)
  - Non-linear and floating-point expressions
- Switching domain made easy

```
ap_manager_t* man = oct_manager_alloc(...) ⟹
     ...          = ppl_grid_manager_alloc(...)
```

The APRON Library : Benefits

For domain users

- Higher-level API
  - Variables (`"x"`,`"y"`) replace dimensions (0,1)
  - Abstract values typed by environments (`["x";"y"]`)
  - User-convenient functions
    - Change of environment,.*e.g.* from `["x";"y"]` to `["y";"z"]`, involving introduction & elimination of variables (+ permutation of dimensions)
  - Non-linear and floating-point expressions
- Switching domain made easy

```
ap_manager_t* man = oct_manager_alloc(...) ⟹
     ...          = ppl_grid_manager_alloc(...)
```

- Use of different domains at same time
  - Uniform API $\Longrightarrow$ easy
  - Thread-safe $\Longrightarrow$ enables concurrent use

The APRON Library : Benefits (cont.)

For domain users

- Provides a set of reference implementations for 6 domains

1. Intervals [Cousot-Cousot-76] with BOX (Jeannet-Miné-07)
2. Octagons [Miné-01] with OCTAGON [Miné-01]
3. Convex Polyhedra [Cousot-Halbwachs-78] with
   - NEWPOLKA (Wilde-93, Halbwachs-94, Jeannet-00)
   - PPL [... + Bagnara & al - 02])
4. Linear equalities [Karr-76] with NEWPOLKA
5. Linear congruences [Granger-91]

   with PPL [Bagnara & al - 05]
6. Reducted product polyhedra/congruences

   with NEWPOLKA + PPL

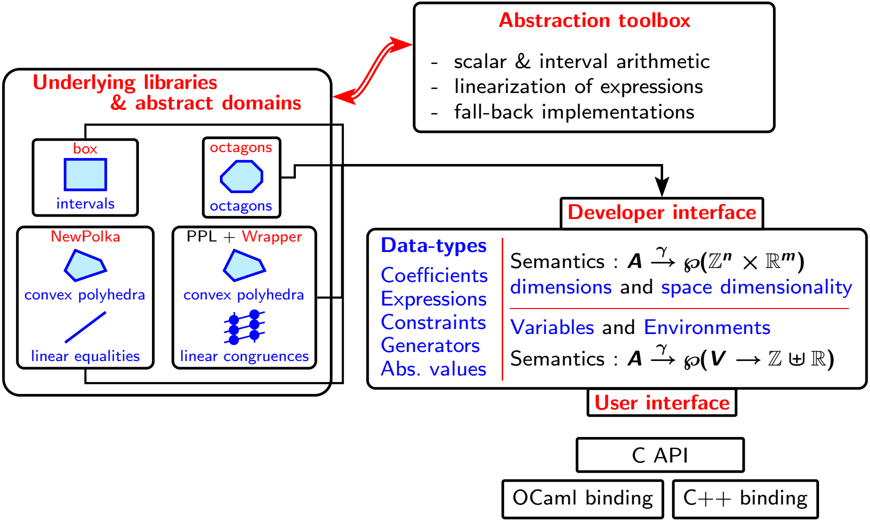
The APRON Library : Benefits (cont.)

For domain implementors

- Only level 0 API to implement (core functionalities)
- Still some redundant functions (*e.g.* assignements)
  - Kept for efficiency reasons in the API
  - But fallback functions provided
- Ready-to-use convenience libraries
  - Numbers (machine int, float, GMP, MPFR) and interval arithmetic
  - **Linearization of non-linear expressions** [Miné-04]
    $\Longrightarrow$ Non-linear and floating-point expressions for free
  - Reduced product, ...

$\Longrightarrow$ only a small core of functions to implement

The APRON Library : Structure



The APRON Library : Distribution

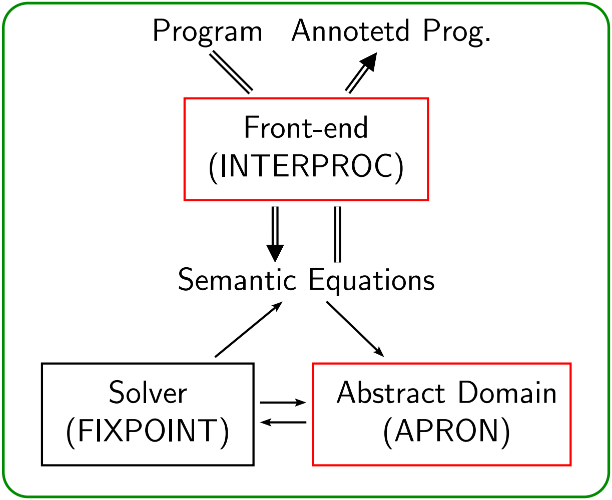
http://apron.cri.ensmp.fr/library/

- Released under LGPL license
- 50 000 lines of C
- Current language bindings : C, C++, OCaml

Some perspectives

- Two innovative domains under developpment by external teams (next talk describes one of them)
- BDDAPRON : combining
  - finite datatypes using BDDs (booleans, bounded integers, enumerated types)
  - with numerical datatypes using APRON domains

Typical Architecture of a Static Analyzer



The INTERPROC analyzer

`http://pop-art.inrialpes.fr/interproc/interprocweb.cgi`

- A demonstration analyzer for a toy language
  - Control : conditionals, while loops, recursive procedures
  - Data : integer and real variables, full support of APRON expressions
- Infers numerical properties using APRON
  - Choice by the user of the underlying abstract domain
  - Exploited *e.g.* by InvGen tool [Rybalchenko & al - 09]
- Simple (3000 LOC of OCaml)
  - Thanks to APRON high-level API
- Online WEB version available
- Released under GPL license