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

entry

1

x:=5

2

y:=100

3

x>=0 ?

4

x<=0 ?

x:=x-1

y:=y+10

5

6

exit
Collecting Semantics:
Collects reachable environments at each control point

entry 1
   ↓
  2
      ↓
  3
         ↓
  4
             ↓
x>0?  x:=x-1
   ↓
  5
     ↓
x<0?  y:=y+10

exit 6

- $\text{Env} = \{x, y\} \rightarrow \mathbb{Z}$
- Invariants $X_i \in \wp(\text{Env})$
- Semantics of statements:
  $[\text{stm}] : \wp(\text{Env}) \rightarrow \wp(\text{Env})$
Program Analysis by Abstract Interpretation 1/3

Collecting Semantics:

Collects reachable environments at each control point

entry

1

x:=5

2

y:=100

3

x>=0 ?

x:=x-1

y:=y+10

4

x<0 ?

5

exit

6

- Env = \{x, y\} \rightarrow \mathbb{Z}
- Invariants \( X_i \in \wp(Env) \)
- Semantics of statements: \([\text{stm}] : \wp(Env) \rightarrow \wp(Env)\)
- Concrete equation system

\[
\begin{align*}
X_1 &= T_D = Env \\
X_2 &= [x := 5](X_1) \\
X_3 &= [y := 100](X_2) \cup [y := y + 10](X_5) \\
X_4 &= [x \geq 0?](X_3) \\
X_5 &= [x := x - 1](X_4) \\
X_6 &= [x < 0?](X_3)
\end{align*}
\]

The recursive system has a unique least solution (lfp)
Collecting Semantics:

Collects reachable environments at each control point.

- \( \text{Env} = \{x, y\} \rightarrow \mathbb{Z} \)
- Invariants \( X_i \in \wp(\text{Env}) \)
- Semantics of statements:
  \[ [\text{stm}] : \wp(\text{Env}) \rightarrow \wp(\text{Env}) \]
- Concrete equation system:
  \[
  \begin{cases} 
  X_1 = \top_{\mathcal{D}} = \text{Env} \\
  X_2 = [x := 5](X_1) \\
  X_3 = [y := 100](X_2) \cup [y := y + 10](X_5) \\
  X_4 = [x \geq 0?](X_3) \\
  X_5 = [x := x - 1](X_4) \\
  X_6 = [x < 0?](X_3)
  \end{cases}
  \]

The recursive system has a unique least solution (lfp).
Undecidability Issues:

- $\mathcal{D} = \phi(\text{Env})$ is not computer-representable
- $\llbracket \cdot \rrbracket$ and $\cup$ are not computable
- $\text{lfp}$ is not computable
Undecidability Issues:

- $\mathcal{D} = \emptyset(\text{Env})$ is not computer-representable
  $\llbracket \cdot \rrbracket$ and $\cup$ are not computable
- $\text{lfp}$ is not computable

Static approximation: Abstract Domain

- $\mathcal{D}^\#$: (simpler) set of computer-representable elements
  - $\gamma: \mathcal{D}^\# \rightarrow \mathcal{D}$: gives a meaning to abstract elements
  - $\llbracket \cdot \rrbracket^\#$ and $\cup^\#$: sound abstract counterparts to $\llbracket \cdot \rrbracket$ and $\cup$
Program Analysis by Abstract Interpretation 2/3

Undecidability Issues:

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

Static approximation: Abstract Domain

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

Dynamic approximation: Widening

- \( \blacktriangledown \) will ensure termination of \( \text{lfp} \) computation
Solving the abstract equation system

entry

1
x := 5

2
y := 100

3

x \geq 0?

4
x := x - 1
y := y + 10

x < 0?

5

exit

6

Static approximation:

\[
\begin{align*}
X_2 \supseteq \{ x := 5 \} \cup X_1 \\
X_3 \supseteq \{ y := 100 \} \cup X_2 \\
X_4 \supseteq \{ x \geq 0? \} \cup X_3 \\
X_5 \supseteq \{ x := x - 1 \} \cup X_4 \\
X_6 \supseteq \{ x < 0? \} \cup X_3 
\end{align*}
\]

solved iteratively from initial states
Solving the abstract equation system

**Static approximation:**

\[
\begin{align*}
X_2^\# & \supseteq [x := 5]^\#(X_1^\#) \\
X_3^\# & \supseteq [y := 100]^\#(X_2^\#) \cup^\#
\end{align*}
\]

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

$D^\# \text{abstracts } \phi(\text{Env}) \text{ with } \text{Env} = \text{Var} \rightarrow \mathcal{N}_{\mathbb{Z} \text{or } \mathbb{R}}$

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]

Linear Equalities
\[ \sum_i \alpha_i X_i = \beta \]
[Karr-76]

Simple Congruences
\[ X_i \equiv a_i [b_i] \]
[Granger-89]

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
\[ P(\bar{X}) = 0, \quad P \in \mathbb{R}[\text{Var}] \]
[Sankaranarayanan-Sipma-Manna-04]
Numerical Abstract Domains: Implementation

- Representation of abstract elements
- Logical/set operations:
  - conjunction ($\cap^\#$), disjunction ($\cup^\#$)
  - emptiness and inclusion test
  - introduction/elimination of variables
- Definition of a concrete semantics
  - $\llbracket\text{expr}\rrbracket : \mathcal{D} \rightarrow \wp(\mathcal{N})$
  - $\llbracket\text{cond}\rrbracket : \mathcal{D} \rightarrow \mathcal{D}$
  - $\llbracket\text{instr}\rrbracket : \mathcal{D} \rightarrow \mathcal{D}$
- And its abstraction in $\mathcal{D}^\#$
  - $\llbracket\text{cond}\rrbracket^\# : \mathcal{D}^\# \rightarrow \mathcal{D}^\#$
  - $\llbracket\text{instr}\rrbracket^\# : \mathcal{D}^\# \rightarrow \mathcal{D}^\#$
- Widening, Projection, Property extraction, ...
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 developed 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
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 $\equiv$ abstract data-type
- Effective underlying domain controlled by a manager
  - Domain-dependent code located in manager allocation
  - User options controlling the precision/efficiency tradeoff
Example

```c
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);
```
Support of non-linear, floating-point expressions

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

Two-level API

- Level 0: abstracts \( \mathbb{Z}^p \times \mathbb{R}^q \) (implementor level)
  Core functionalities, Efficiency
- Level 1: abstracts \( \text{Var} \rightarrow \mathbb{Z} \cup \mathbb{R} \) (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
  \[
  \text{ap\_manager\_t* man = oct\_manager\_alloc(\ldots) \implies}\n  \ldots = \text{ppl\_grid\_manager\_alloc(\ldots)}\n  \]

- Use of different domains at same time
  - Uniform API \implies easy
  - Thread-safe \implies 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. **Reduced 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. assignments)
  - 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]
    - Non-linear and floating-point expressions for free
  - Reduced product, ...

⇒⇒ only a small core of functions to implement
The APRON Library: Structure

Abstraction toolbox
- scalar & interval arithmetic
- linearization of expressions
- fall-back implementations

Underlying libraries & abstract domains
- box
- octagons
- NewPolka
- PPL + Wrapper

Data-types
- Coefficients
- Expressions
- Constraints
- Generators
- Abs. values

Semantics: \( A \rightarrow \varphi(\mathbb{Z}^n \times \mathbb{R}^m) \)
- dimensions and space dimensionality

Variables and Environments
Semantics: \( A \rightarrow \varphi(V \rightarrow \mathbb{Z} \cup \mathbb{R}) \)

Developer interface
User interface

C API
OCaml binding
C++ binding
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 development 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

Program annotated prog.

Front-end (INTERPROC)

Semantic Equations

Solver (FIXPOINT) → Abstract Domain (APRON)
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