Program Verification by Abstract Interpretation

Patrick Cousot
pcousot@cs.nyu.edu  cs.nyu.edu/~pcousot

CMU, Pittsburgh  October 18, 2011

Content

- A lightweight informal introduction to Abstract Interpretation
- Application to the Verification of Embedded Control
- Commercial tools (ASTRÉE, CCCheck)
- Current and future research

Program concrete models/semantics

- Program executions are modelled by the language formal semantics (observed at discrete times)

\[ s(t) \]

\[ t \]
Verification of safety properties

- Program executions cannot reach a state in which computations can go wrong

Abstraction over-approximation

- Further approximations of the reachable states may introduce spurious states

Abstraction

- The computations are over-approximated in the abstract (e.g. by reachable states)

Machine-computable abstractions

- To scale up, machine computable abstraction must be very efficient and precise enough
Soundness

• No definite error is ever omitted (counter-examples: Coverity, Klocwork, etc)

Applications

• Verification of absence of runtime errors (arithmetic overflows, divisions by zero, buffer overruns, null and dangling pointers, user assertion violations, unreachability, ...) so specification is fully automatic

• Avionics, Spatial, Automotive, Medical, Systems on Chip (SoC), etc

• Use general abstractions for programming languages (integers, floats, arrays, structures, pointers, ...)

• Use domain-specific abstractions incorporating knowledge on control systems (filters, quaternions, integrators, etc)
**Abstractions**

- Collecting semantics: partial traces
- Intervals: $x \in [a, b]$
- Simple congruences: $x \equiv a[b]$
- Octagons: $\pm x \pm y \leq a$
- Ellipses: $x^2 + by^2 - axy \leq d$
- Exponentials: $-a^{bt} \leq y(t) \leq a^{bt}$

---

**Example of general purpose abstraction: octagons**

- Invariants of the form $\pm x \pm y \leq c$, with $O(N^2)$ memory and $O(N^3)$ time cost.
- Example:

  ```c
  while (1) {
    R = A-Z;
    L = A;
    if (R>V) {
      \star L = Z+V; \}
  \star}
  ```

  - At $\star$, the interval domain gives $L \leq \max(\max A, (\max Z)+(\max V))$.
  - In fact, we have $L \leq A$.
  - To discover this, we must know at $\star$ that $R = A-Z$ and $R > V$.

- Here, $R = A-Z$ cannot be discovered, but we get $L-Z \leq \max R$ which is sufficient.
- We use many octagons on small packs of variables instead of a large one using all variables to cut costs.

---

**Example of domain-specific abstraction: ellipses**

```c
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
BOOLEAN INIT; float P, X;
void filter () {
  static float E[2], S[2];
  if (INIT) { S[0] = X; P = X; E[0] = X; }
  else { P = ((((0.5 * X) - (E[0] * 0.7)) + (E[1] * 0.4))
               + (S[0] * 1.5)) - (S[1] * 0.7)); }
  E[1] = E[0]; E[0] = X; S[1] = S[0]; S[0] = P;
  /* S[0], S[1] in [-1327.02698354, 1327.02698354] */
}
void main () { X = 0.2 * X + 5; INIT = TRUE;
  while (1) {
    X = 0.9 * X + 35; /* simulated filter input */
    filter (); INIT = FALSE; }
}
```

**Example of domain-specific abstraction: exponentials**

```c
% cat count.c
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
volatile BOOLEAN I; int R; BOOLEAN T;
void main() {
  R = 0;
  while (TRUE) {
    if (I) { R = R + 1; }
    else { R = 0; }
    T = (R >= 100);
    \_\_ASTREE_wait_for_clock();
  }
% cat count.config
\_\_ASTREE_volatile_input((I [0,1]));
\_\_ASTREE_max_clock((3600000));
% astree -exec-fn main -config-sem count.config count.c|grep '|R|'
|R| <= 0. + clock *1. <= 3600001.
```
Commercial Tools

- AbsInt (www.absint.de)
- Astrée (run-time error analysis)

- Other abstract-interpretation-based tools: WCET, stack usage, memory safety analysis

Clousot/CCcheck in Visual Studio

- Modular code contract verification (and inference)

Research Challenges

- see online, www.rise4fun.com
CMACS achievements

- Static analysis of array content (POPL 2011)
- Necessary precondition inference for code contracts (VMCAI 2011)
- Abstract interpretation-based theory to combine abstract interpretation, model-checking and verifiers/SMT solvers (FOSSACS 2011)
- Termination analysis (POPL 2012)
- Probabilistic Abstract Interpretation

Research challenges

- Complex data structures
- Liveness, Closing the loop, ...
- Parallelism, Fairness, Scheduling, ... (AstréeA, www.astreea.ens.fr/)
- Security (AstréeS)

Other application domains:

Security

- Information flow analysis

Biology

- Cellular signaling networks
- Formal rule-based model reduction

Conclusion
Conclusion

• Does **scale** up (to $> 10^6$ LOCS)!

• **Find bugs** not found by simulation, testing, enumerative bug finding methods

• Can prove the **absence of** well-defined categories of bugs

• **Covers** new requirements on formal methods (e.g. DO 178 C)

• **Mandatory** in all embedded control systems of an European plane manufacturer

• Unfortunately **not so well-known and well-used in the US**

The End