CMACS Industry Workshop on Verification of Embedded Control Systems

Program Verification by Abstract Interpretation

Patrick Cousot

pcousot@cs.nyu.edu

cs.nyu.edu/~pcousot

CMU, Pittsburgh

October 18, 2011

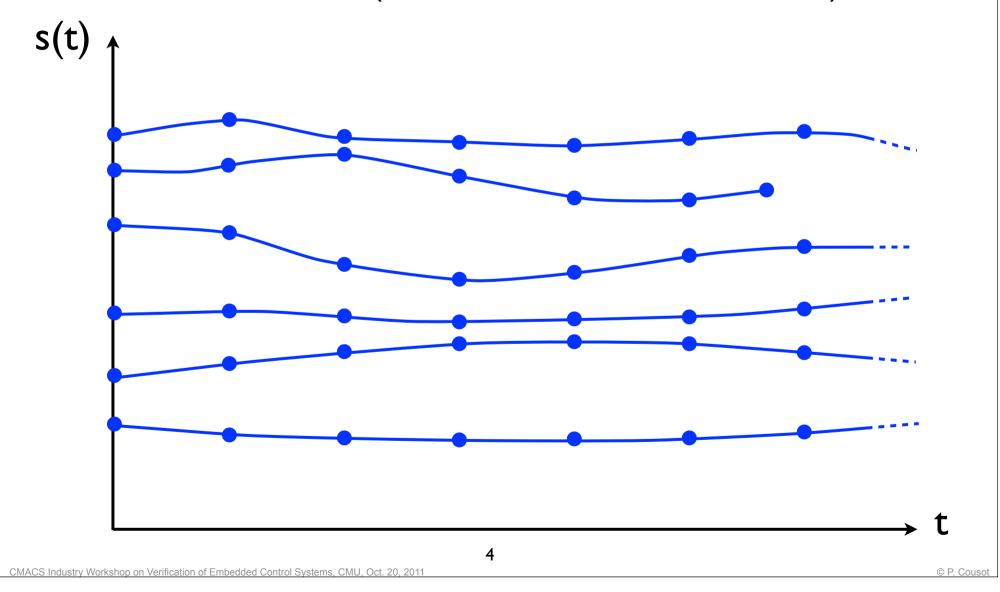
Content

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

An informal introduction to Abstract Interpretation

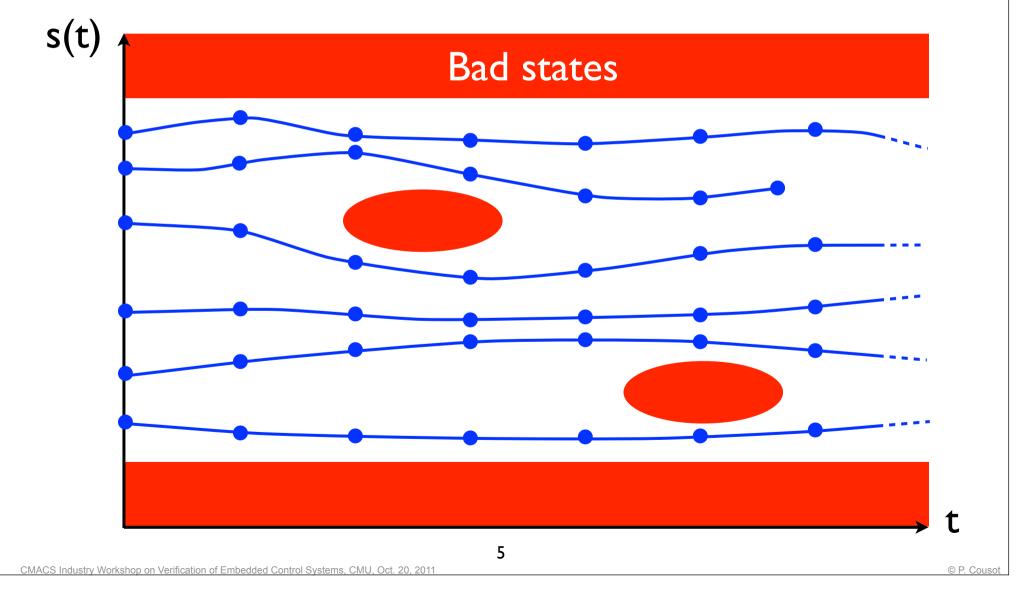
Program concrete models/semantics

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



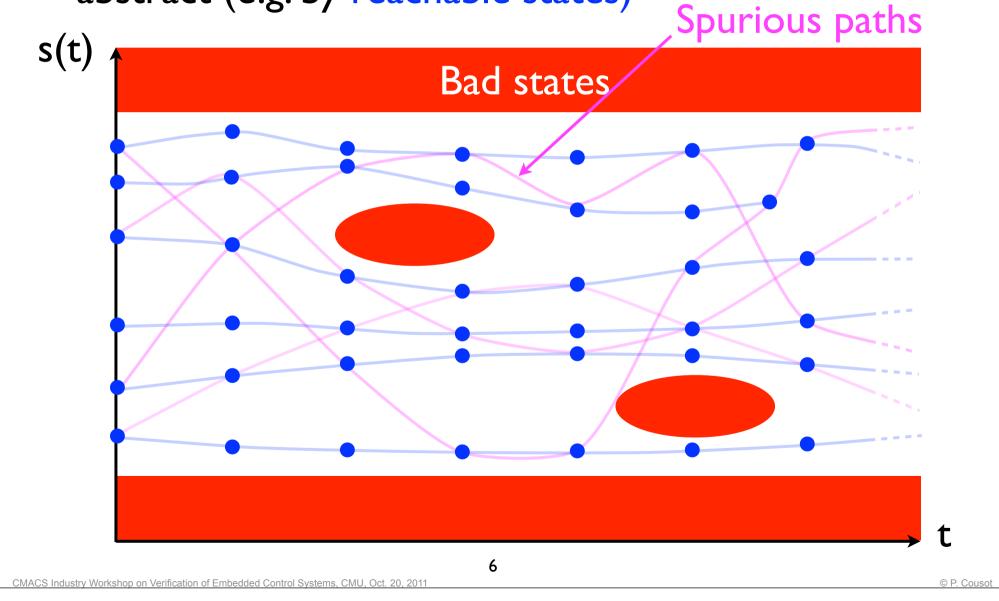
Verification of safety properties

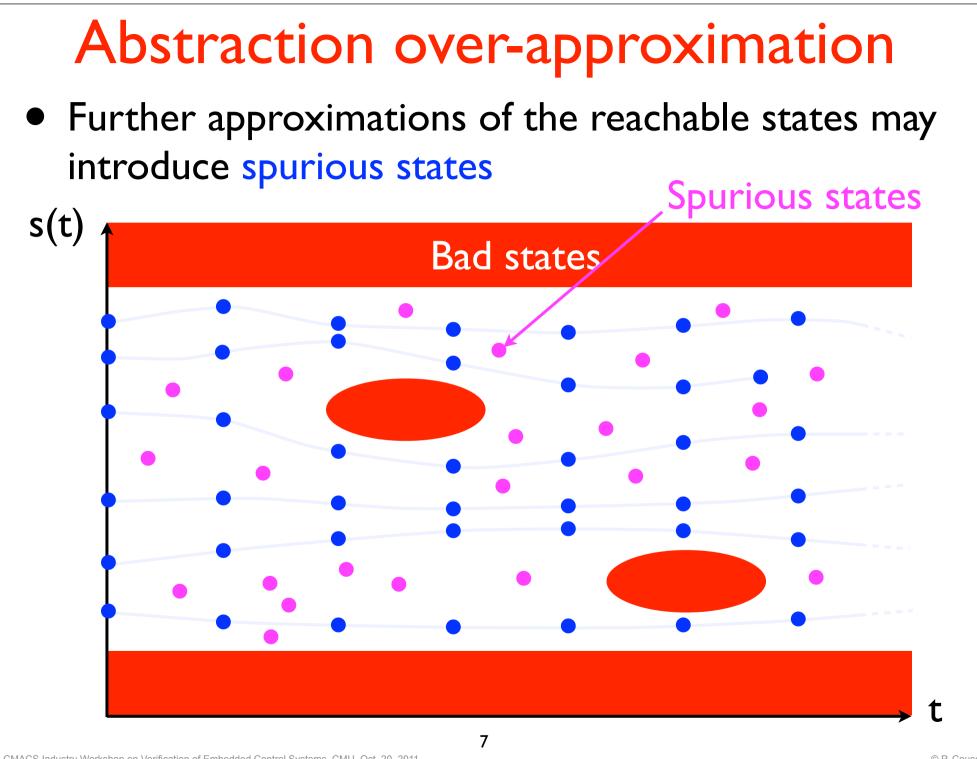
 Program executions cannot reach a state in which computations can go wrong



Abstraction

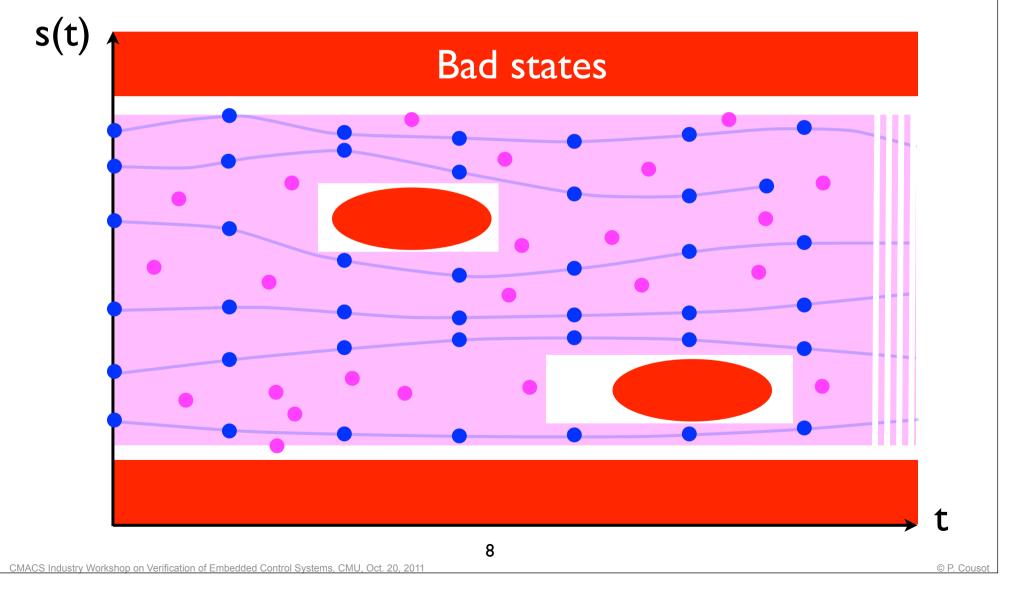
The computations are over-approximated in the abstract (e.g. by reachable states)





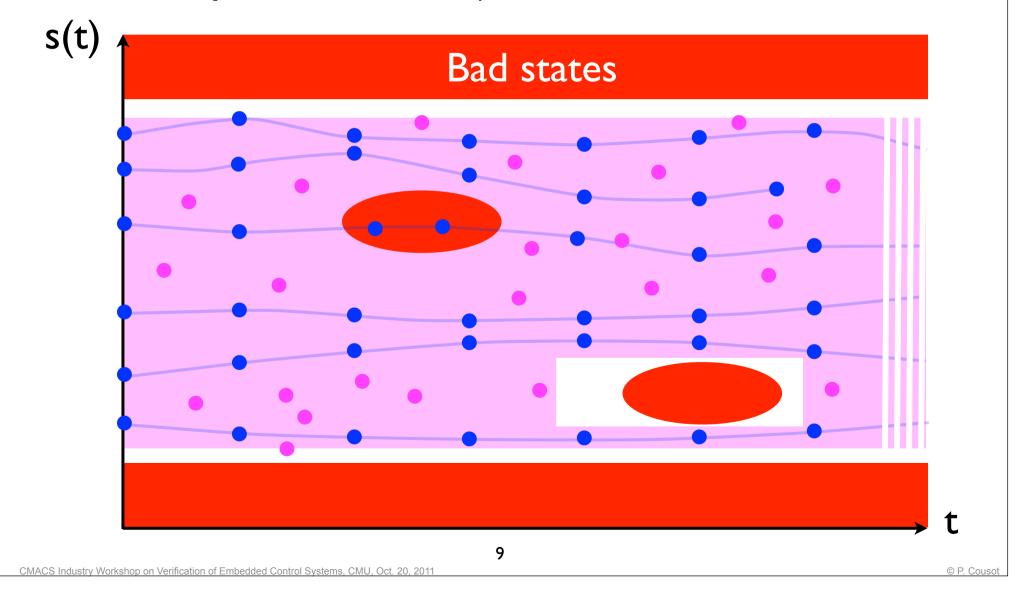
Machine-computable abstractions

• 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)



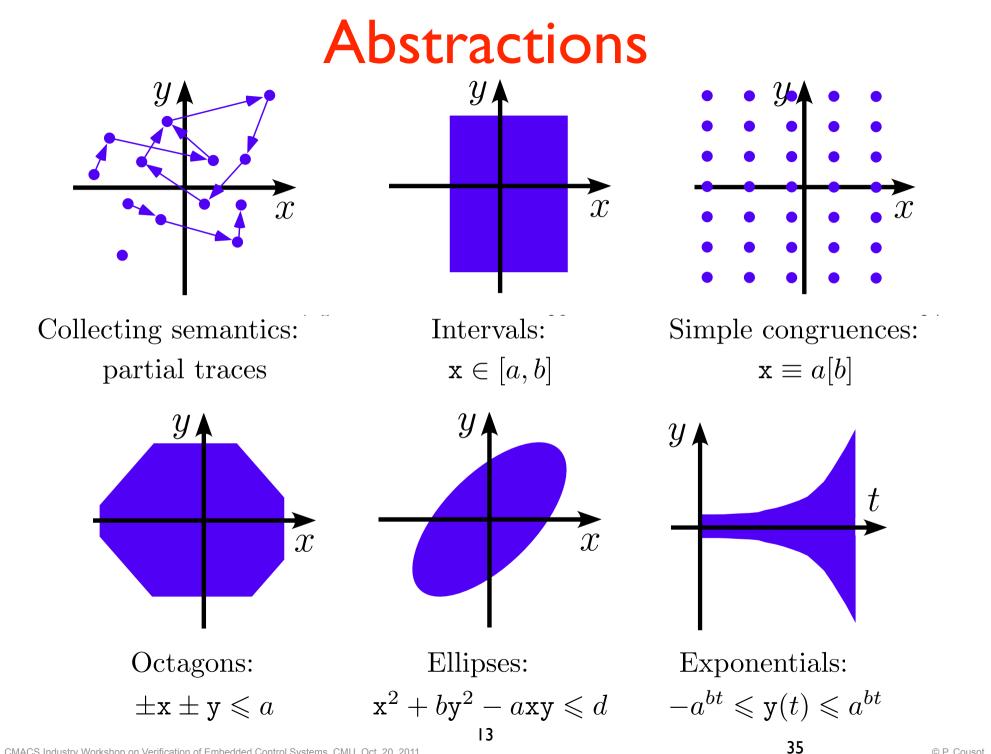
Incompleteness: false alarms • Spurious errors are possible (e.g. PolySpace) and may be eliminated by refining the abstraction (e.g. Astrée) s(t) **Bad** states 10 © P Couso CMACS Industry Workshop on Verification of Embedded Control Systems,

Application to the Verification of Embedded Control Systems

11

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)



Example of general purpose abstraction: octagons

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

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

36

Here, R = A-Z cannot be discovered, but we get L-Z ≤ 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.

14

Example of domain-specific abstraction: ellipses 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; } } 15 38

Example of domain-specific abstraction: exponentials

```
% cat count.c
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
volatile BOOLEAN I; int R; BOOLEAN T;
void main() {
  R = 0;
  while (TRUE) {
    __ASTREE_log_vars((R));
                                               \leftarrow potential overflow!
    if (I) { R = R + 1; }
    else { R = 0; }
    T = (R \ge 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.
                                      38
 CMACS Industry Workshop on Verification of Embedded Control Systems, CMU, Oct. 20
```

Commercial Tools

Commercialization

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

oject Analysis Edit Help		Ν
2 📑 📍 🏲 🕑 🗉 🖸	⇙▋▤▤◙◶◪▯▯;;;;;ノ▣▤	i π
temp [11458] Local settings Mapping to original sources Report Analysis	<pre>Analyzed file: C:/home/kaestner/Ab/Projects/Astree/Experiments/bsp1/bsp1pp.c 1 # 1 "bsp1.c" 2 # 1 "<built-in>" 3 # 1 "<command-line>" 4 # 1 "bsp1.c" 5 6 volatile int SwitchPosition; 7 7 8 int main() 9 { 10 11 int MODULE_NUMBER = 0x80000000 + 0x38343031; 12 13 char sp = SwitchPosition; 14 } 15</command-line></built-in></pre>	<pre>Original source: C:/home/kaestner/nt/Projects/Astree/Experiments/bsp1/bsp1.c 1 #define BASE 0x8000000 2 #define OFFSET 0x38343031 3 volatile int SwitchPosition; 4 int main() 5 { 6 /**/ 7 int MODULE_NUMBER = BASE + OFFSET; 8 /**/ 9 char sp = SwitchPosition; 10 } 11</pre>
Parallelism	Line 11, Column 0	Line 7, Column 0
 bsp1 bsp1pp.c 	/* Domains: Guard domain, and Packed (Boolean relations (based on Absolute value equality rn No ambiguity due to side effects in expressions [call#main@8 at ./drive_C/home/kaestner/AbsInt/Projects/Astree/Experiments/bsp1/bsp1pp.c. ALARM: implicit unsigned int>>signed int conversion range (3090427933) not included in [-21 [call#main@8 at ./drive_C/home/kaestner/AbsInt/Projects/Astree/Experiments/bsp1/bsp1pp.c. ALARM: implicit signed int>>unsigned char conversion range [-2147483648, 2147483647] not Time spent in analysis of function main: 0.234000 s (0 h 0 mn 0 s) 0 function(s) and 0 point(s) not reached during the analysis There were 2 alarm(s) in 2 context(s), 0 other enror(s), and 1 warning(s)	reproduct a convence of the second se
Project	/* 1 procedure(s) executed */	- •

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

18

Clousot/CCcheck in Visual Studio

• Modular code contract verification (and inference)

RiSE.Tmp				
<pre>namespace RiSE { public class Tmp { public static void VMCAIPaperExample(string[] strings) { for (var i = 0; i < strings.Length; i++) { Contract.Assert(strings[i] != null); strings[i] = null; } } //// * </pre>				
r List				-
0 Errors 1 0 Warnings 1 3 Messages				
Description	File	Line	Column	Project
1 CodeContracts: Suggested requires: Contract.Requires(strings != null);	Max.cs	11	12	StaticChecker
2 CodeContracts: Suggested precondition: Contract.Requires(Contract.ForAll(0, strings.Length, [i] != null));	i => strings Max.cs	11	12	StaticChecker
	Max.dll	1	1	StaticChecker
3 CodeContracts: Checked 10 assertions: 8 correct (2 masked)				

Research Challenges

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