# Pouzdanost i Sigurnost

## Software-a

Domagoj Babic
http://www.domagoj.info/

http://www.domagoj.info/fesb-july2012.ppt

# Lekcija 1.

- Abstracts govora se pišu u prvom licu
  - Govornik se obraća publici kroz abstract
- Znanstveni radovi s 1 autorom – isto
- Izbjegavati passiv i indirekciju po svaku cijenu
  - Teško čitljiv, dosadan, zauzima puno mjesta
- “We”
  - “Mi” autori (*In our implementation...*)
  - “Mi” pisac i čitatelj (*Let us prove...*)
  - Royal “We”

# Lekcija 1.

U predavanju će se sažeti glavni rezultati istraživanja predavača u području pouzdanosti i sigurnosti software-a u zadnjih 7-8 godina. Na početku će se dati kratki uvod u statičku analizu software-a, simboličku egzekuciju, te će se dotaknuti pitanja procedura odlučivanja koje se koriste u analizi pouzdanosti. U području sigurnosti software-a, fokus će biti na učenju formalnih jezika te će se pokazati kako se takve metode učenja mogu koristiti u analizi sigurnosti. Tijekom govora, posebno će se naglasak staviti na predavačeve uspjehe i neuspjehe u njegovom istraživanju, te iskustva koja je na svom istraživačkom putu stekao. U nastavku će bit prezentirane mogućnosti suradnje sa istraživačkom grupom predavača. Govor će biti prilagođen mlađim znanstvenicima i zahtijevat će samo osnovna predznanja iz područja računarstva.

U govoru cu sazeti glavne rezultate svog istrazivanja u podrucju pouzdanosti i sigurnosti softwera u zadnjih 7-8 godina. Pocet cu sa kratkim uvodom u staticku analizu softwera, simbolicku egzekuciju, te se dotaknuti procedura odlucivanja koje se koriste u analizi pouzdanosti. U podrucju sigurnosti softwera, fokusirat cu se na ucenje formalnih jezika i pokazati kako se takve metode ucenja mogu koristiti u analizi sigurnosti. Tijekom govora, posebno cu se fokusirati na uspjehe i neuspjehe u svom istrazivanju, te lekcije koje sam naucio uz put. Govor ce biti prilagodjen mladjim znanstvenicima i zahtjevat ce samo osnovna predznanja iz podrucja racunarstva.

_____

_____

_____

# Područja u Računarnim Znanostima

Computer vision and graphics

Software engineering

Machine learning (ML)

Distributed systems

Operating systems

Computational finance

Algorithms

Security

Theory

Bioinformatics

Scientific computing

Architecture

Programing languages (PL)

Formal methods

Human-computer interface (HCI)

# Kronologija Mog Istraživanja

- 2003, 2004: Decision procedures (SAT, SMT)
- 2005-2008: Extended static checking
- 2008-2010: SMT, automated test generation
- 2010-2012: Security, grammatical inference, automated test generation

# Osnovna Podjela SW Analiza

## Statička Analiza

- Bez izvršavanja koda
- Source ili binary
- Vrste:
  - Type checking
  - Static checking
  - Extended static checking
  - Model checking
  - AST pattern mining

## Dinamička Analiza

- Izvršava kod
- (Instrumented) binary
- Vrste:
  - Taint analysis
  - Delta debugging
  - Single path symbolic execution
  - Fault isolation
  - Memory consistency
  - Race conditions, deadlocks
  - Code coverage
  - Likely program invariants
  - ....

Car recalls (software):

2005: Mercedes-Benz, 1.3 million 2001-2004 E, SL, CLS

2005: Toyota, 160.000 Prius

2007: Ford, 1.8 million 7.3L PowerStroke trucks

Cost of software bugs (US economy only): $59.5 B / year

NIST survey: http://www.nist.gov/director/prog-ofc/report02-3.pdf

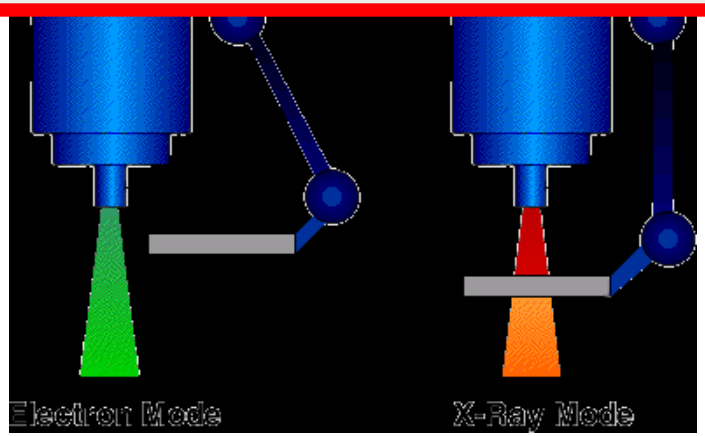
CryoSAT, Oct 8th, 2005

....

Defective medical devices:

1985: Therac-25 radiation therapy machine

5 deaths



# Static Checking

- Problem: dokazati određena svojstva programa
  - Npr.: assert(ptr != NULL);
- Generalno problem je undecidable
- Ideja: abstrahirati program tako da obuhvaća više ponašanja programa (over-approximation), i onda dokazati željeno svojstvo
- Računanje abstrakcije: abstract interpretation

# Primjer

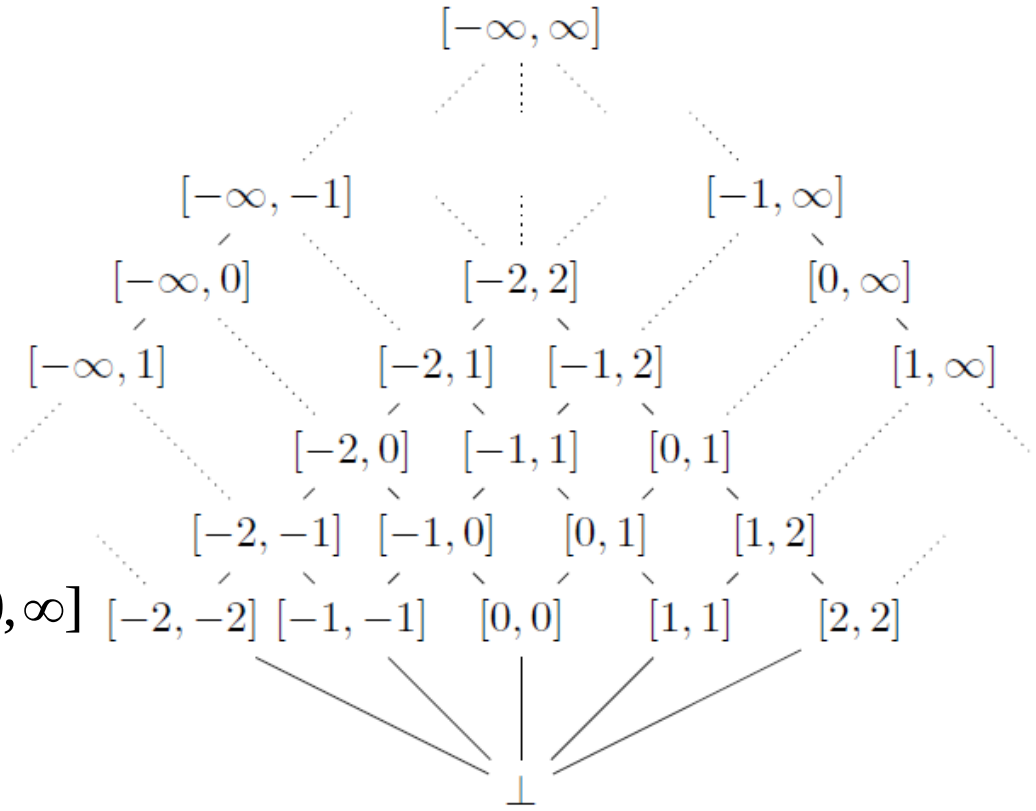
```
int abs(int x) {
   if (x < 0)
      x = -x
   else
      // do nothing
   assert(x >= 0);
   return x;
}
```

$[-\infty, \infty]$

$[-\infty, -1] \quad [1, \infty]$

$[0, \infty]$

$[0, \infty] \cup [1, \infty] = [0, \infty]$




Lattice (partially ordered set) with additional property:
-Any two elements have a join
-Any two elements have a meet

# Static Checking

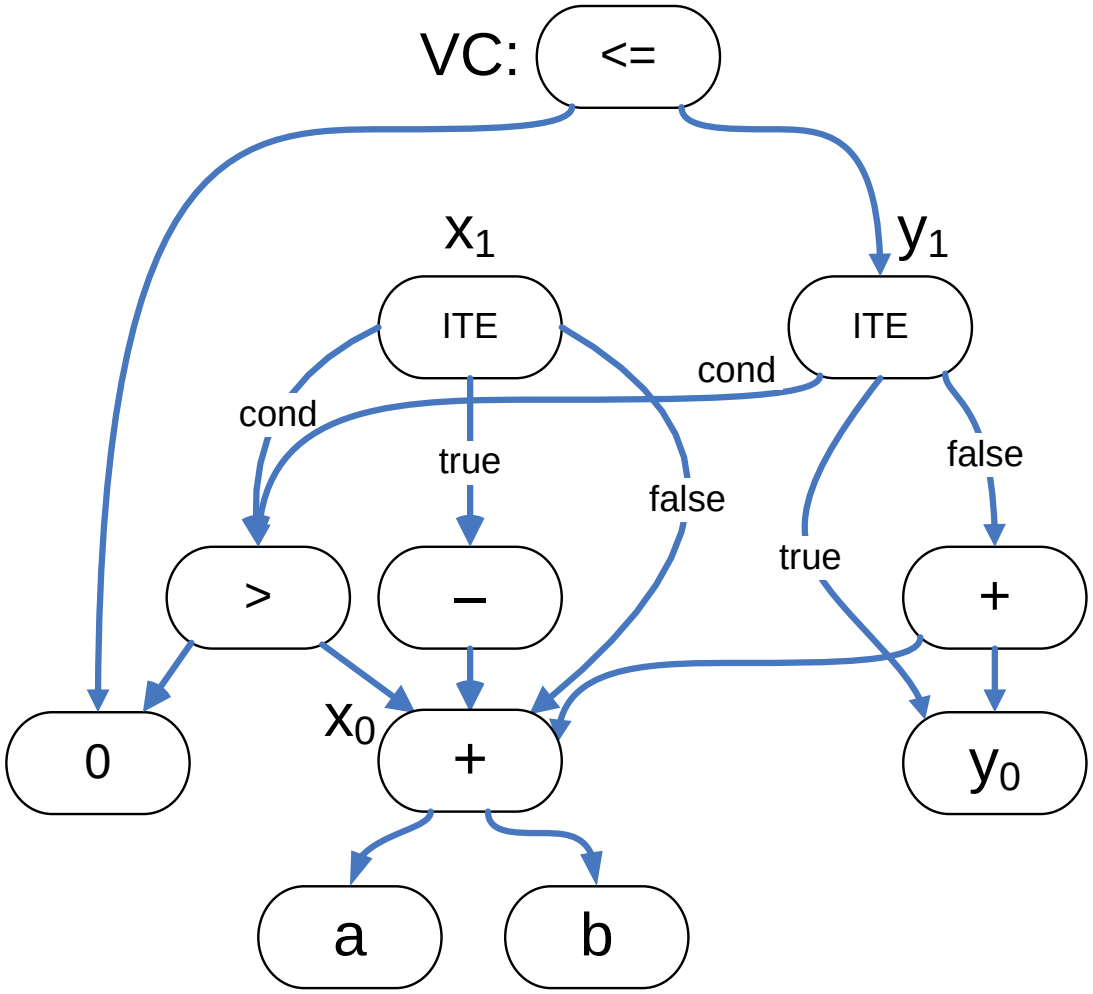
- Abstrakcija koda u petljama
  - Fixed point computation: abstract interpretation se ponavlja dok nema više promjene
  - Widening: $[0,1] \nabla [0,2] = [0, \infty]$
    - Forsira bržu konvergenciju, terminaciju po cijenu preciznosti
  - Join (unija na prošlom slideu)
    - Također izvor nepreciznosti
- Posljedice over-aproksimacije (abstrakcije)
  - Ako se svojstvo dokaže, onda je kod ispravan
  - Ako static analysis javi grešku, može biti false warning (false positive)
- Istraživanje u static analysis se vrti oko:
  - Nalaženja novih abstraktnih domena
  - Balansiranje preciznosti (performansi) i abstrakcije
- Upotreba:
  - Sve i jedan compiler
  - Airbus software, …

# Extended Static Checking

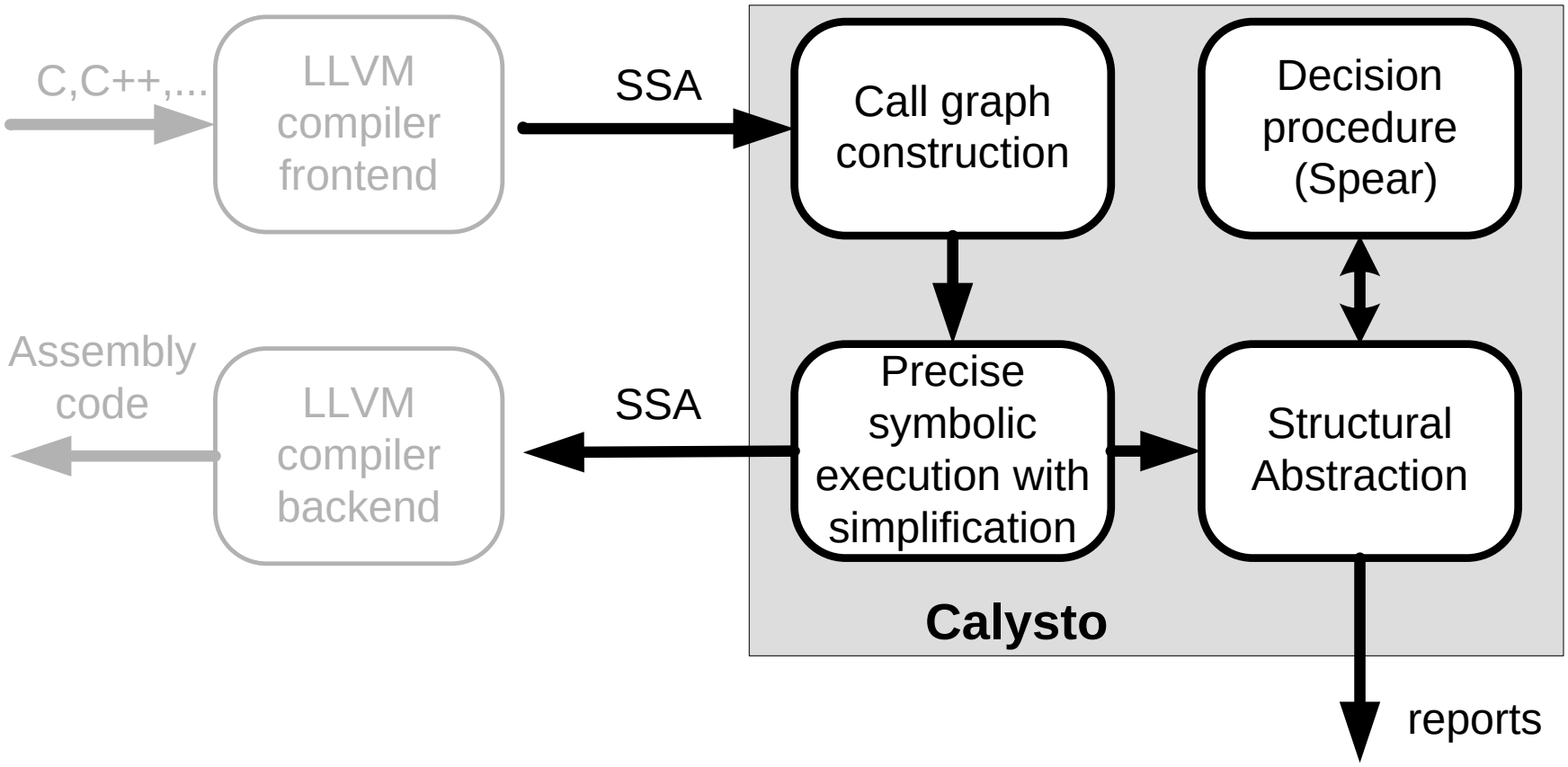
- Static checking – puno false positives
- Ideja:
  - Koristiti vrlo preciznu abstract domain
  - Izračunata abstrakcija nije više nužno over-approximation
    - Moguće da analiza neće naći nege greške u programu
- Vrlo precizna abstract domain:
  - Symbolic execution

# Symbolic Execution

$x_0$=a+b;
if $x_0$<0 then
    $x_1$ = -$x_0$;
    $y_1$ = $y_0$;
else
    $x_1$ = $x_0$;
    $y_1$ = $y_0$+$x_0$;
assert(0<=$y_1$);



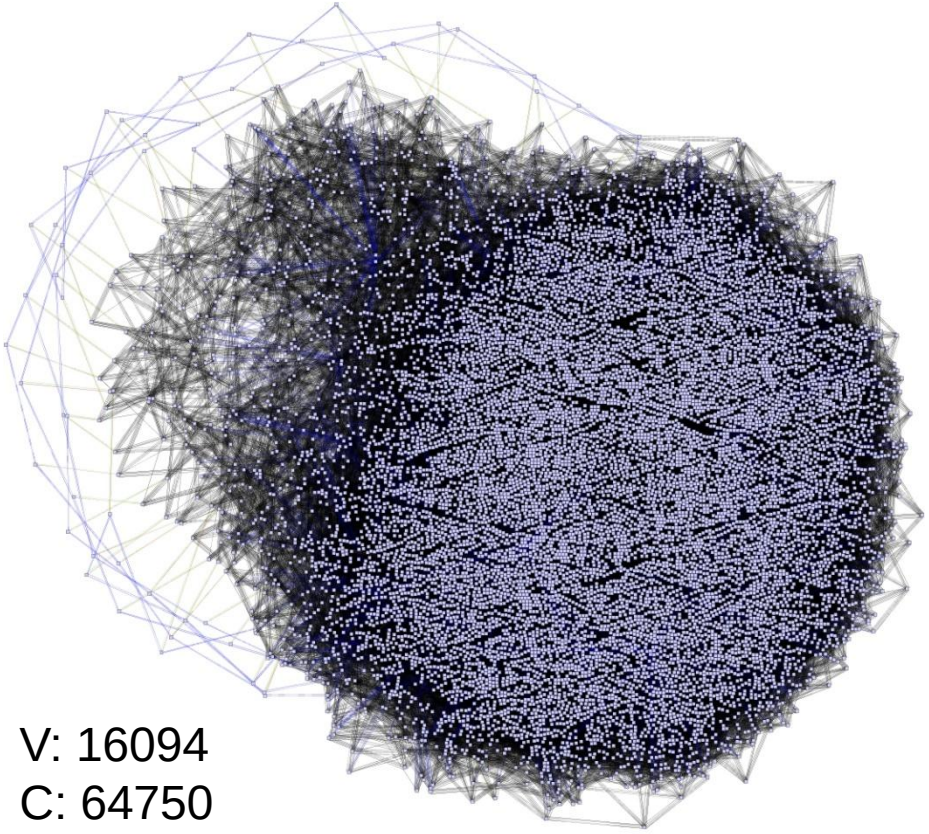
# Calysto Static Checker



# Abstrakcija Formula

- Formule su preteške za rješavanje
  - Simplifikacija
  - Abstrakcija
- Problem:
  - Što abstrahirati?
- Use problem structure

Calysto v1.3 Xinetd v2.3.14 vc18132 [uns]
after slicing, CSE, simplification



V: 16094
C: 64750

# Strukturna Abstrakcija

- Compute summaries of each side-effect
- Abstract function side-effects with unconstrained variables
- Let the decision procedure say what it needs
- Refine only the side-effect that is missing

# Strukturna Abstrakcija (Glavna Petlja)

```
f = Encode(F);
while (true) {
        <valid, falsifying assignment> = Solve(f);
        if (valid) {
                Report correct. Exit.
        }
        // Falsifying assignment found
        if (!Refine(F, falsifying assignment)) {
                Report bug. Exit.
        }
}
```

Abstraction comes for free!

# Calysto – Experimentalni Rezultati

| Benchmark | LOC | Reports | Bugs | Unkn. | FP Rate | Time [s] |
|---|---|---|---|---|---|---|
| Bftpd 1.8 | 4532 | 12 | 11 | 0 | 9% | 3.14 |
| Bftpd 1.9.2 | 4602 | 5 | 4 | 0 | 20% | 2.86 |
| HyperSAT 1.7 | 9123 | 0 | 0 | 0 | 0% | 14.57 |
| Spin 4.3.0 | 28394 | 0 | 0 | 0 | 0% | 6858.10 |
| Openssh 4.6p1 | 81908 | 4 | 1 | 0 | 75% | 8995.64 |
| Inn 2.4.3 | 122727 | 10 | 6 | 1 | 34% | 1312.33 |
| Ntp 4.2.4p2 | 185865 | 30 | 26 | 0 | 14% | 558.16 |
| Ntp 4.2.5p66 | 192019 | 13 | 4 | 3 | 56% | 493.39 |
| Openldap 2.4.4a | 374266 | 20 | 15 | 2 | 27% | 200.02 |
| Bind 9.4.1p1 | 393318 | 5 | 2 | 3 | 0% | *2436.88 |
| TOTAL | 1406754 | 99 | 69 | 9 | 23% | 20875.09 |

# Calysto

| Benchmark | LOC | Reports | Bugs | Unkn. | FP Rate | Time [s] |
|---|---|---|---|---|---|---|
| TOTAL | 1406754 | 99 | 69 | 9 | 23% | *20875.09 |

# Saturn

| Benchmark | LOC | Reports | Bugs | Unkn. | FP Rate | Time [s] |
|---|---|---|---|---|---|---|
| TOTAL | 1406754 | 1459 | 38 | 87 | 97% | 40226.40 |

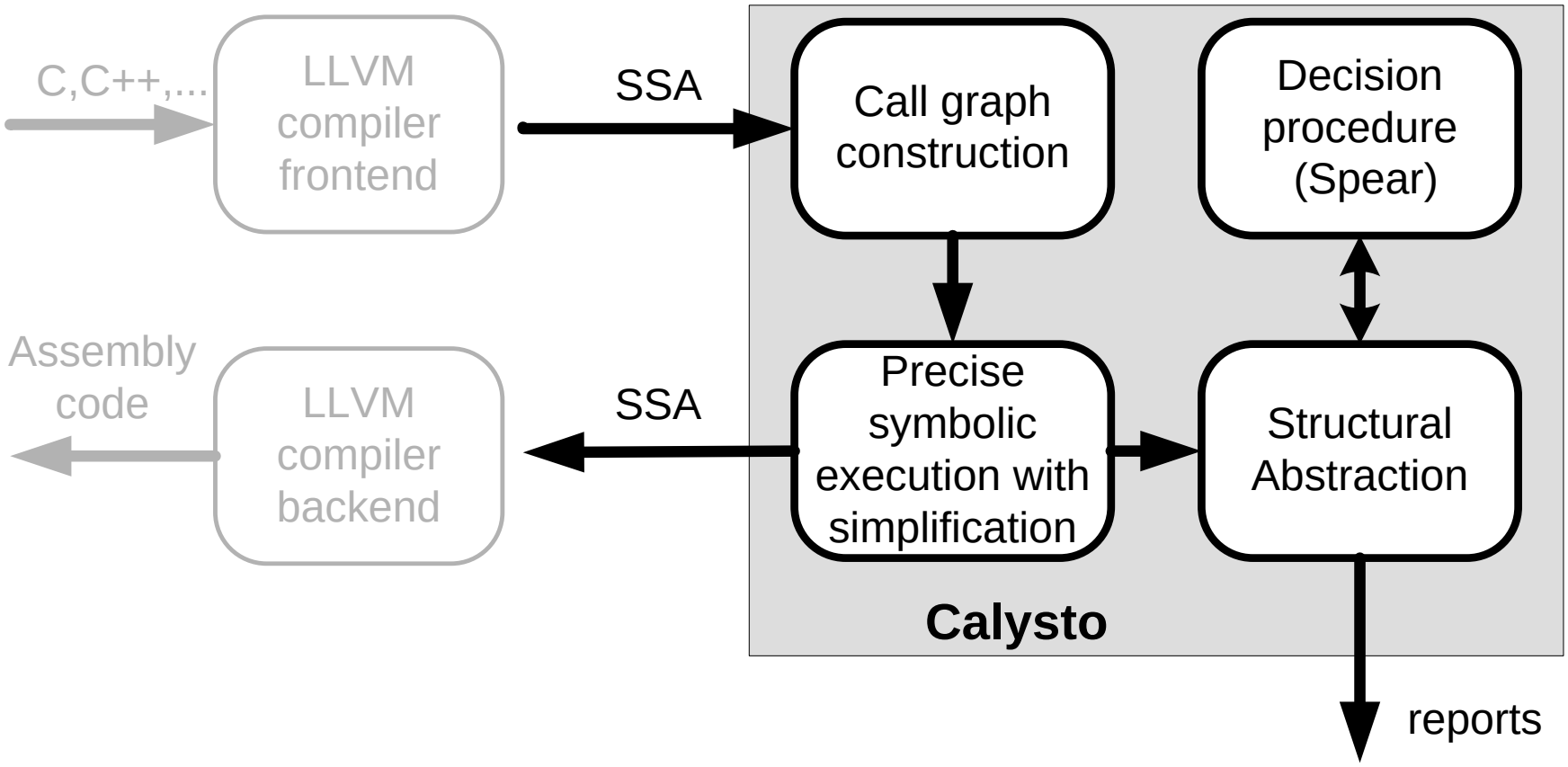
# 5 Godina Poslije

- Calysto papir dobio 65 citacija do sada
- Strukturna abstrakcija prihvaćena u industriji
  - Poirot static checker (MSR)
  - FSoft (NEC)
- Niz pozvanih govora širom svjeta...

# Lekcija 2.

- Kvalitetno pisanje papira izuzetno bitno!
  - Više škola koristi Calysto papir kao primjer dobrog pisanja papira
- Idealni papir (za mene)
  - Relevantan problem, teoretski doprinos, kvalitetna eksperimentalna potvrda
- Jaki eksperimentalni rezultati puno pomažu
  - Uspoređivanje sa *state-of-the-art*

# Decision Procedures (Solveri)



# Decision Procedures

- Koriste se za rješavanje logičkih formula
  - Ako formula *valid* – svojstvo vrijedi
  - NP-hard
- Vrste
  - SAT (propositional satisfiability)

$$a \wedge \big((a \wedge b) \Rightarrow (c \wedge \neg d)\big) \wedge \big(c \Rightarrow (b \wedge d)\big)$$

  - SMT (satisfiability-modulo theories)

$$(0 \le i \le size) \wedge \forall x \in \text{int} : (0 \le x < i) \Rightarrow (array[x] = 0)$$

$$\Rightarrow$$

$$\forall x \in \text{int} : (0 \le x < size) \Rightarrow (array[x] = 0)$$

# Boolean Satisfiability (SAT) Problem

- Find an assignment to Boolean variables such that a given propositional formula evaluates to TRUE
  - E.g., $a \wedge \big((a \wedge b) \Rightarrow (c \wedge \neg d)\big) \wedge \big(c \Rightarrow (b \wedge d)\big)$
- Applications: combinational equivalence checking, automatic test-pattern generation, model checking, planning, bioinformatics,...
- Solving Boolean satisfiability in practice:
  - Solved with SAT solvers (MiniSAT, Spear, …)

# Solving Purely Propositional Formula

$$a \wedge \big((a \wedge b) \Rightarrow (c \wedge \neg d)\big) \wedge \big(c \Rightarrow (b \wedge d)\big)$$

- Conjuncts:

$$a$$

$$(a \wedge b) \Rightarrow (c \vee \neg d)$$

$$c \Rightarrow (b \wedge d)$$

- Solutions:

$$a \equiv TRUE$$

$$b \equiv FALSE$$

$$c \equiv FALSE$$

$$d \equiv FALSE$$

# Solving Purely Propositional Formula

$$a \wedge \big((a \wedge b) \Rightarrow (c \wedge \neg d)\big) \wedge \big(c \Rightarrow (b \wedge d)\big)$$

- Conjuncts:

$$a$$

$$(a \wedge b) \Rightarrow (c \vee \neg d)$$

$$c \Rightarrow (b \wedge d)$$

- Solutions:

$$a \equiv TRUE \qquad a \equiv TRUE$$
$$b \equiv FALSE \qquad b \equiv FALSE$$
$$c \equiv FALSE \qquad c \equiv FALSE$$
$$d \equiv FALSE \qquad d \equiv TRUE$$

# SAT Modulo Theories: One Theory Case

$$a \wedge \big((a \wedge b) \Rightarrow (c \wedge \neg d)\big) \wedge \big(c \Rightarrow (b \wedge d)\big)$$

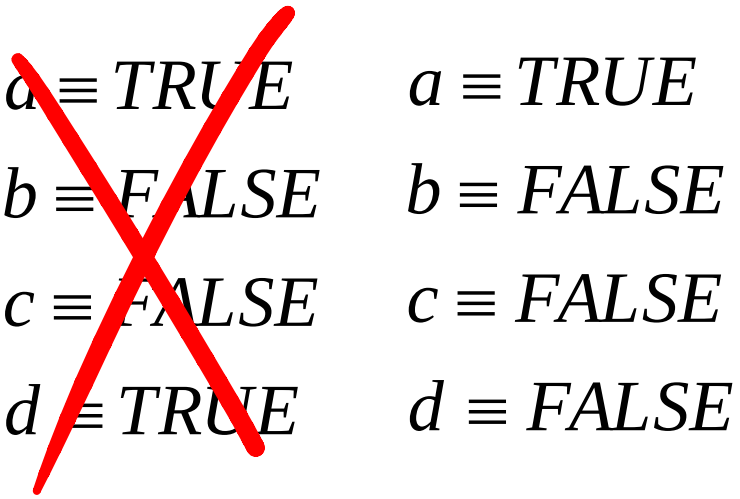
- Assume some Boolean variables represent real linear arithmetic (in)equalities…

$a \equiv (r \leq 5.0)$

$b \equiv (r < 5.0)$

$c$ remains unconstrained

$d \equiv (2r > 10.0)$




*( r is a real variable )*

# SAT Modulo One Theory

- Brute force approach
  1. Enumerate all propositional solutions
  2. Check every solution with the theory solver (e.g., linear arithmetic)
- Lazy approach
  1. Find a propositional solution
  2. Check it with the theory solver
  3. Add blocking clause (disjunction of negations of all decisions)
  4. Repeat the process until a satisfying assignment found
- Lazy approach + theory conflict analysis
  + Theory solver provides explanations for conflicts

# Multiple Theories

$$a \wedge \big((a \wedge b) \Rightarrow (c \wedge \neg d)\big) \wedge \big(c \Rightarrow (b \wedge d)\big)$$

- Now we add the theory of uninterpreted functions…
- Leibniz rule: $(x = y) \Rightarrow (f(x) = f(y))$
- Add interpretation: $c \equiv \big(f(r) = f(5.0)\big)$

**Important!**

**Constraints of each theory have a solution,**

**BUT a conjunction of constraints of both theories doesn't !**

# Moje Istraživanje o Decision Procedurama

- HyperSAT
  - SAT solver
  - Nova tehnika učenja (bcubing)
  - 3 mjesto na 2005 SAT natjecanju, hand-crafted category
- Tehnike rješavanja bitvector arithmetic formula (MSR 2005 tech report)
- Spear
  - Decision procedura za bitvector aritmetiku
  - Pobjedio na 2007 SMT natjecanju, bitvector arithmetic category
- Tehnike automatske optimizacije heurističkih prametara parametara

# Spear bit-vector decision procedure parameter space

- Large number of combinations:
  - After limiting the range of *double* & *unsigned*
  - After discretization of *double* parameters

  3.78£$10^{18}$
  - After exploiting dependencies

  **8.34£$10^{17}$ combinations**
  - Finding a good combination – hard!

Spear 1.9:

- 4 *heuristics* X
  22 optimization functions
- 2 *heuristics* X
  3 optimization functions
- 12 *double*
- 4 *unsigned*
- 4 *bool*

---

26 parameters

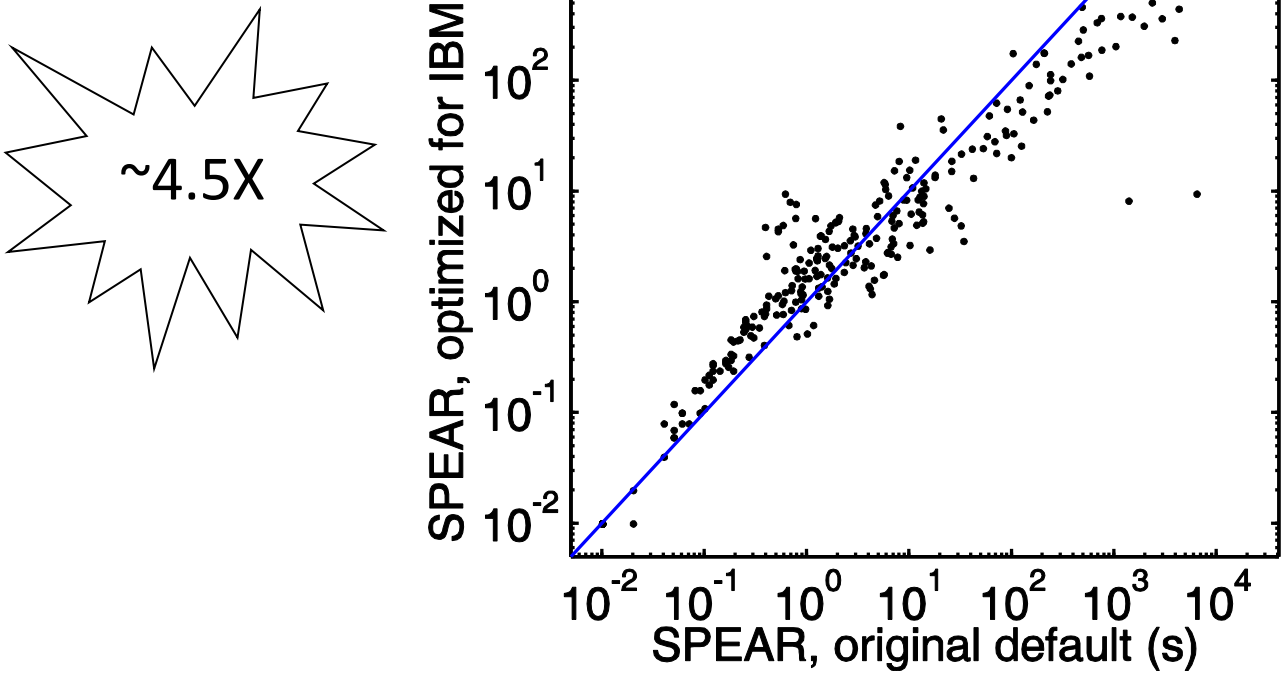
# Goal

- Find a good combination of parameters (and heuristics):
  - Optimize for different problem sets (minimizing the average runtime)
- Avoid time-consuming manual optimization
- Learn from found parameter sets
  - Apply that knowledge to design of decision procedures

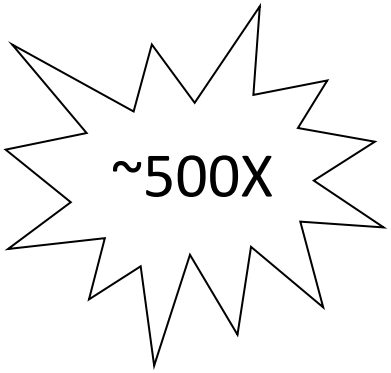
# Automatic tuning

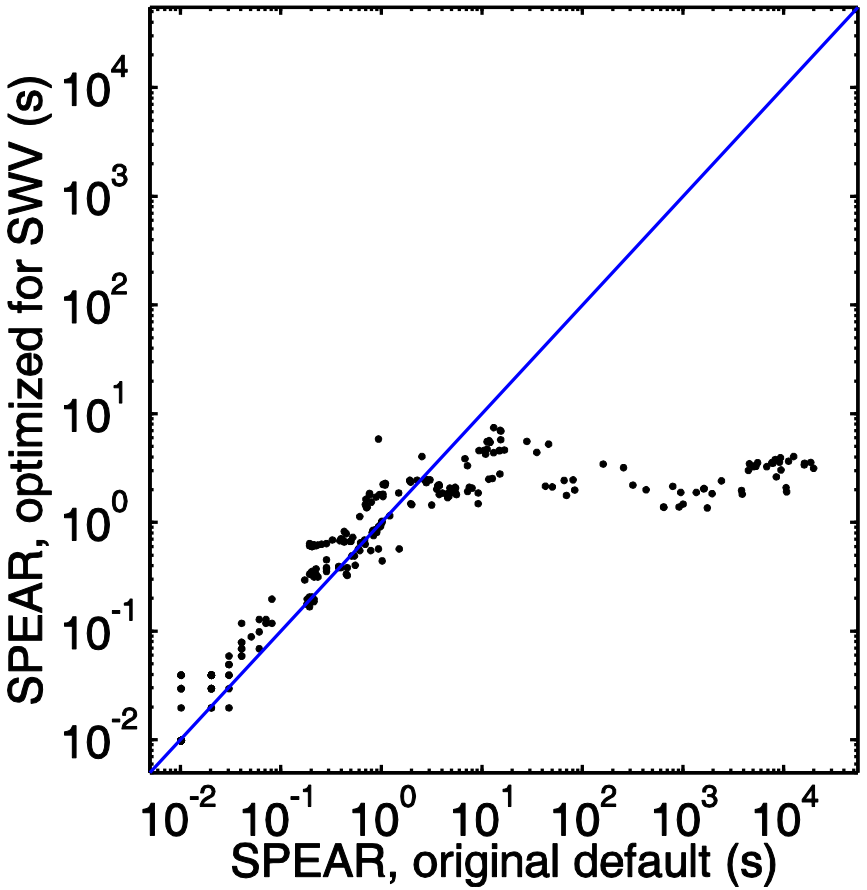
- Loop until happy (with found parameters)
  - Perturb existing set of parameters
  - Perform hill-climbing:
    - Modify one parameter at the time
    - Keep modification if improvement
    - Stop when a local optimum is found

# Tuning 1:
# Bounded model checking instances



# Tuning 2: Calysto instances



# Automated Tuning

- Moj najcitiraniji papir (70 citacija)
- Deseci pozvanih govora širom svjeta
- Postao standard u industriji
  - IBM ga koristi u RuleCheck model checkeru
    - Korišten za verifikaciju Z-series (MTTF several decades!)
  - MSR ga koristi za optimizaciju Z3 solvera
    - Koristi se u 30ak projekata i proizvoda širom MS

# Lekcija 3.

- Internacionalna natjecanja su odličan publicitet, pogotovo ako pobijedite!
- U istraživanju nikad ne znate što će funkcionirati, a što ne, inače, ne bi bilo istraživanje nego inžinjering!
- Greške:
  - Potrošio previše vremena na implementacije tijekom PhD-a, više bi mi koristilo da sam napisao više papira
  - Bcubing je intelektualno dubok rad, ali
    - Težak za shvatiti – 18 citacija za 4 papira ☹
    - Ne ostavlja puno mjesta za follow-up work
  - Nisam napisao pravi papir o Spear solveru, iako competition solver description ima 15 citacija
    - *When you create a wave, you have to surf it!*
  - Nisam publicirao MSR tech report o tehnikama za rješavanje bitvector arithmetic, iako je tech report dobio relativno dosta citacija (22 do sad)

# Hvala!