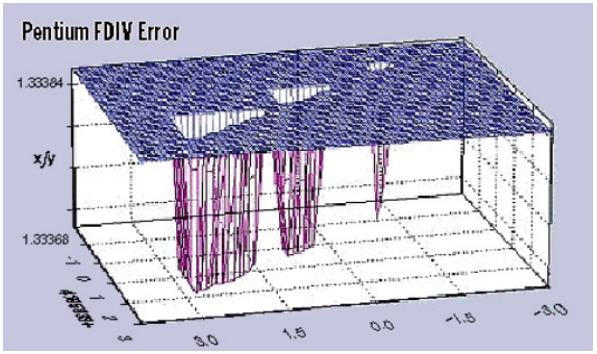
浙江大学
ZHEJIANG UNIVERSITY

基于符号抽象的程序分析

rainoftime
rainoftime.github.io
pyaoaa@zju.edu.cn

保障软件质量的重要性

数值溢出漏洞导致Ariane5火箭升空数秒后爆炸



Intel Pentium漏洞导致声誉和巨额经济损失

纳斯达克OMX系统发生故障造成1300 万美元损失

软件漏洞导致美国东北部大面积停电

软件漏洞导致丰田回收120万辆Prius汽车

软件漏洞导致放疗仪使用超过量的放射物

程序分析: 保障软件质量的重要手段

静态分析

动态分析

SonarQube Coverity Cppcheck PVS-Studio ...

报告提纲

抽象解释理论

符号抽象框架

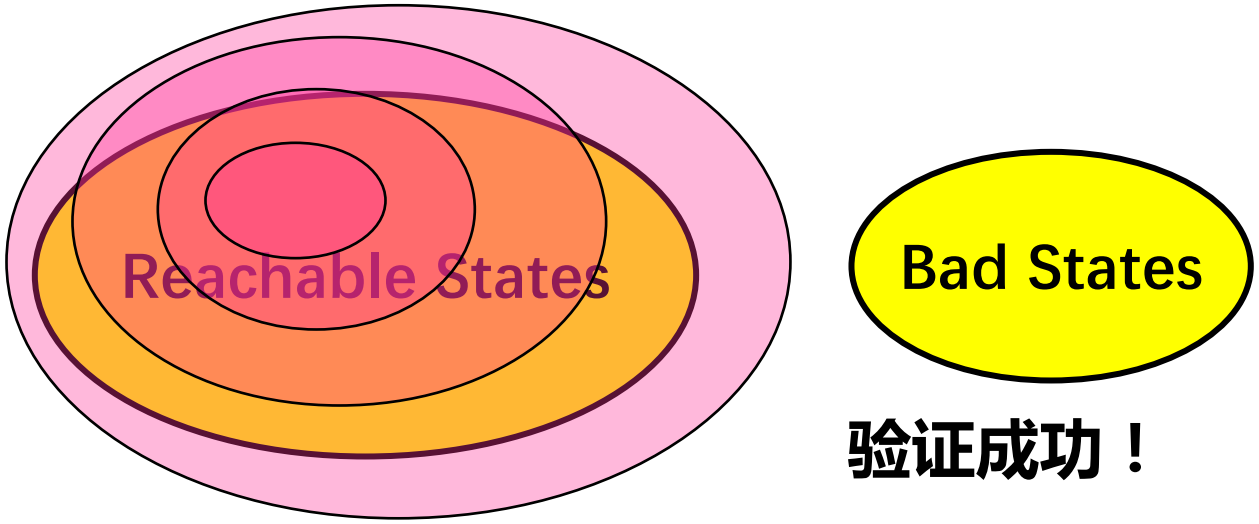
个人近期工作

一些相关问题

抽象解释(Abstract Interpretation)

- 静态程序分析的核心理论 [Cousot & Cousot, POPL'77]
- 对程序的可达状态做上近似

验证程序是否安全



验证成功！

ABSTRACT INTERPRETATION : A UNIFIED LATTICE MODEL FOR STATIC ANALYSIS OF PROGRAMS BY CONSTRUCTION OR APPROXIMATION OF FIXPOINTS

Patrick Cousot* and Radhia Cousot** @ POPL 1977

抽象解释的一些应用

- 1996~ : 涌现了多个基于抽象解释的程序分析与验证工具

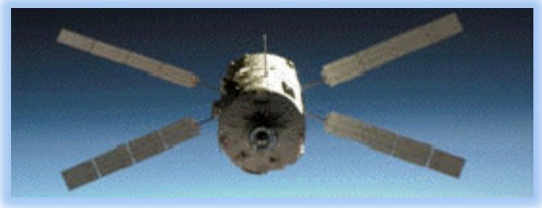
Astrée
数值分析

TVLA
形态分析

CodeSurfer/x86
二进制分析

空中客车A340主飞控软件
（13.2万行C 代码）

空中客车A380主飞控软件
（约35万行C 代码）



欧空局自动货运飞船ATV“儒勒凡尔纳”号（约19万行C 代码）

抽象解释的一些应用

- 1996~ : 涌现了多个基于抽象解释的程序分析与验证工具

Astrée
数值分析

TVLA
形态分析

CodeSurfer/x86
二进制分析

与分离逻辑分庭抗礼 → “收编”基于分离逻辑的静态分析

抽象解释的一些应用

- 1996~ : 涌现了多个基于抽象解释的程序分析与验证工具

Astrée
数值分析

TVLA
形态分析

CodeSurfer/x86
二进制分析

Value-Set Analysis (值集分析) 已成为二进制分析框架标配

BitBlaze BAP angr BINSEC

抽象解释所获部分荣誉

- 2013: SIGPLAN程序语言成就奖

SIGPLAN

To explore programming language concepts and tools focusing on design, implementation and efficient use.

Home | Awards | Conferences | Resources | Membership | SIGPLAN Research Highlights | Student Information | Publications | Announcem

Programming Languages Achievement Award

Given by ACM SIGPLAN to recognize an individual or individuals who has made a significant and lasting contribution to the field of programming languages. The contribution can be a single event or a life-time of achievement. The award includes a prize of $5,000. The award is presented at SIGPLAN's PLDI conference the following June.

Nominations

- Details of the nomination and award process (pdf).
- Please use http://awards.sigplan.org/ to submit nominations.

Recipients of the Achievement Award

2013: Patrick and Radhia Cousot

Patrick and Radhia Cousot are the co-inventors of abstract interpretation, a unifying theory of sound abstraction and approximation of structures involved in various domains of computer science, such as formal semantics, specification, proof, and verification. In particular, abstract interpretation has had a major impact on the development of the static analysis of software. In their original work, the Cousots showed how to relate a static analysis to a language's standard semantics by means of a second, abstract semantics that makes precise which features of the full language are being modeled and which are being discarded (or abstracted), providing for the first time both a formal definition of and clear methodology for designing and proving the correctness of static analyses. Subsequently, the Cousots contributed many of the building blocks of abstract interpretation in use today, including chaotic iteration, widening, narrowing, combinations of abstractions, and a number of widely used abstract domains. This work has developed a remarkable set of intellectual tools and has found its way into practice in the form of widely used libraries and frameworks. Finally, the Cousots and their collaborators have contributed to demonstrating the utility of static analysis to society. They led the development of the Astrée static analyzer, which is used in the medical, automotive, and aerospace industry for verifying the absence of a large class of common programming errors in low-level embedde systems code. This achievement stands as one of the most substantial successes of program verification to date.

- 2018: 约翰·冯诺依曼奖

Patrick Cousot awarded John von Neumann Medal

Patrick Cousot is the recipient of the IEEE John von Neumann medal, given "for outstanding achievements in computer-related science and technology".

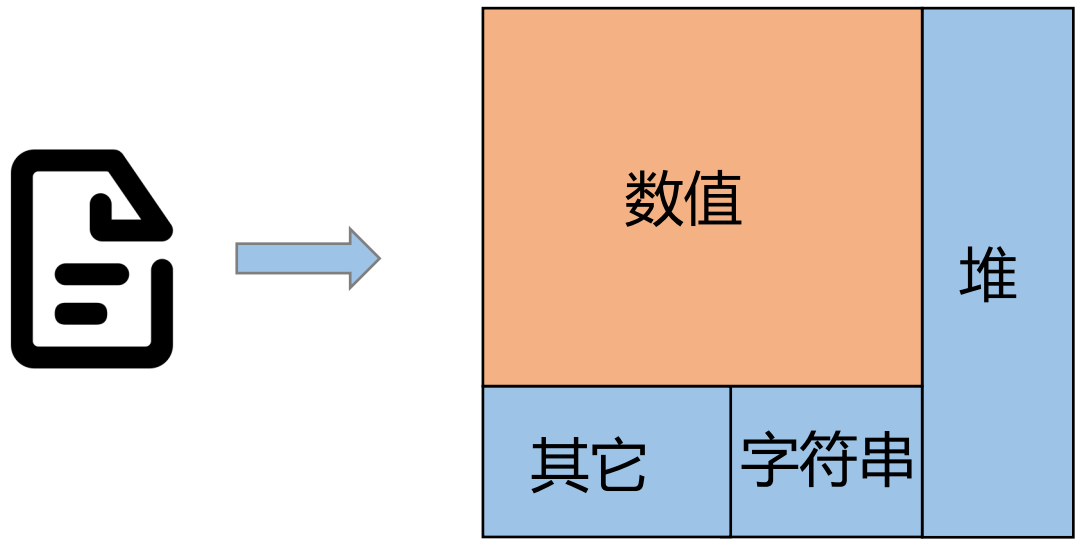
Read More

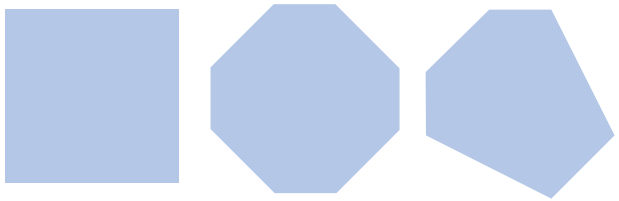
IEEE JOHN VON NEUMANN MEDAL
RECIPIENTS

2018	PATRICK COUSOT Professor, New York University, New York, New York, USA	"For introducing abstract interpretation, a powerful framework for automatically calculating program properties with broad application to verification and optimization."

抽象域及其要素

- 对程序语义的一种数学近似





抽象元素

“程序的状态是什么”

$M \quad \sqcap \quad \sqsubseteq \quad \triangledown$

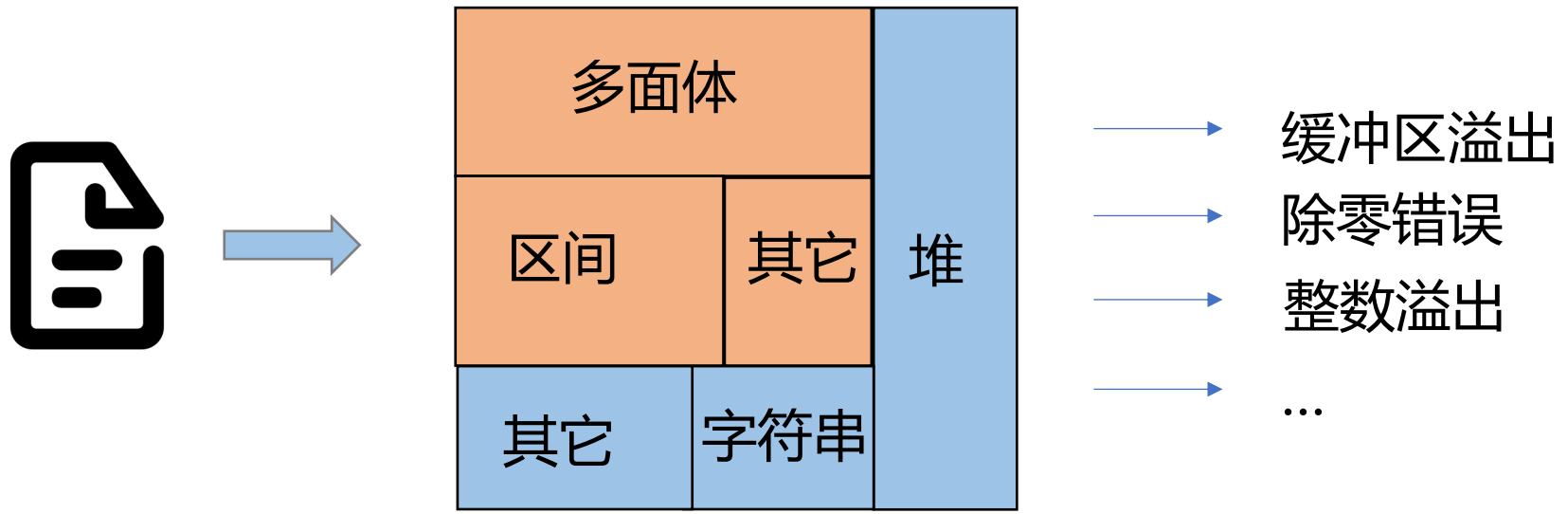
$[x := e] \quad [x < e]$

抽象迁移函数

“状态是怎么变迁的”

抽象域及其要素

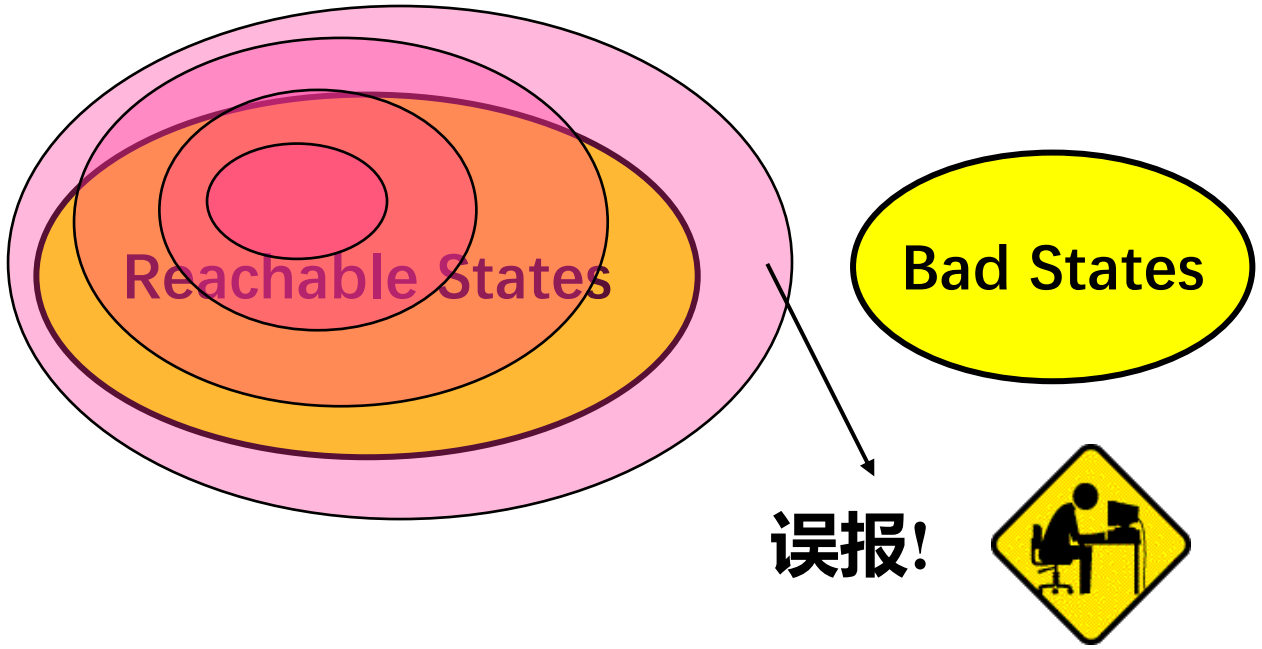
- 捕捉程序数值相关的属性



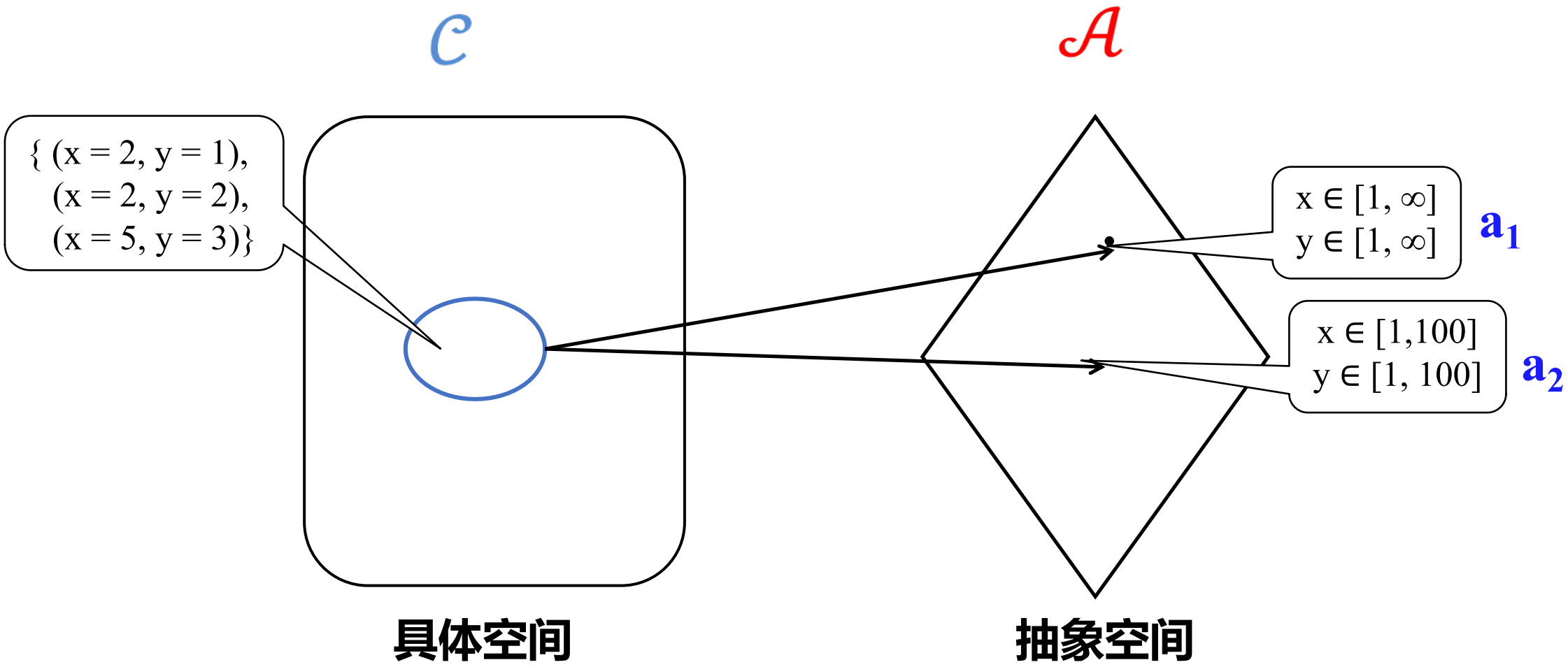
抽象域	抽象元素特征
区间域 [Cousot & Cousot, POPL'77]	$\pm x_i \leq c$
多面体域 [Cousot & Halbwach, POPL'78]	$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \leq c$

抽象解释的精度问题

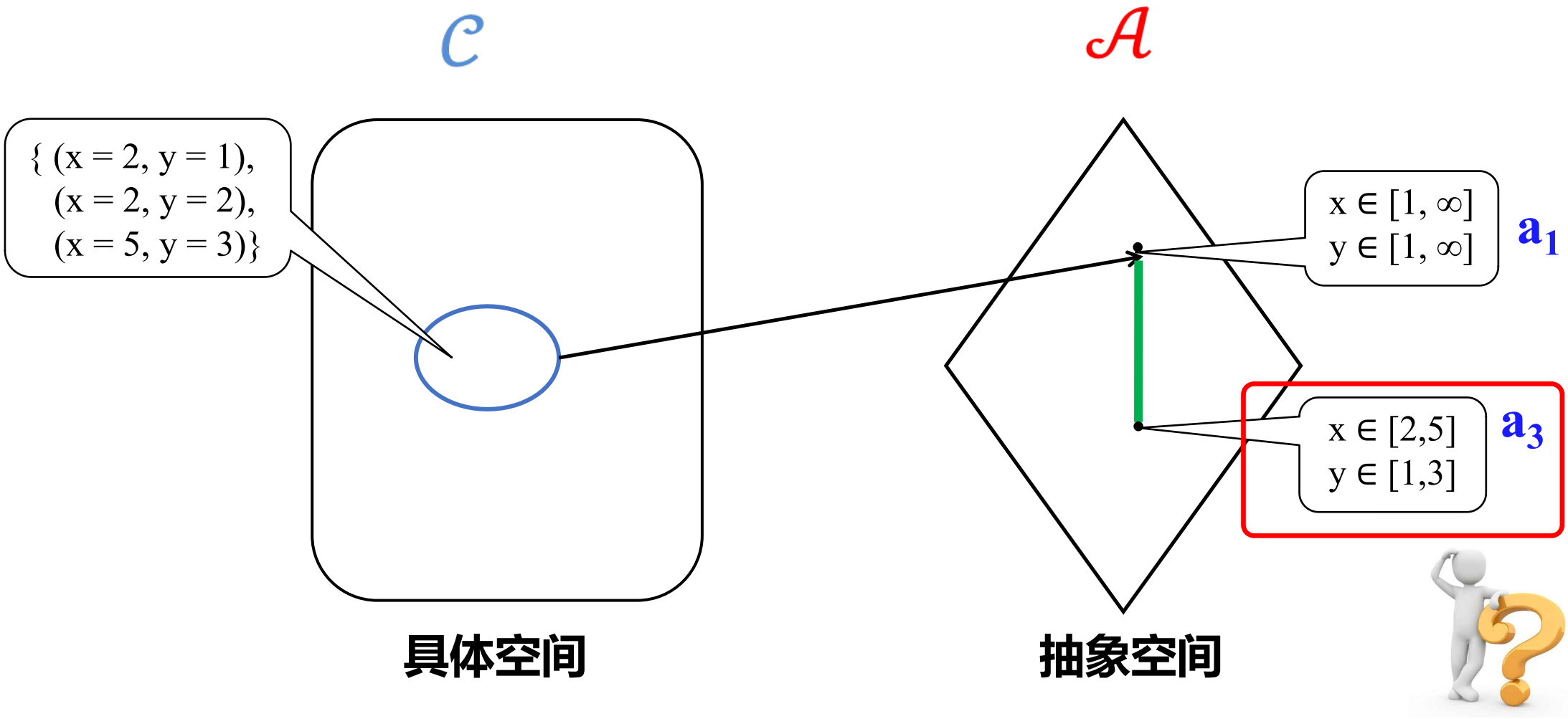
- 静态程序分析的核心理论 [Cousot & Cousot, POPL'77]
- 对程序的可达状态做上近似



例: 区间域的精度问题


$\mathcal{C}$
$\mathcal{A}$
{ (x = 2, y = 1),
(x = 2, y = 2),
(x = 5, y = 3)}
x ∈ [1, ∞]
y ∈ [1, ∞]
a_1
x ∈ [1,100]
y ∈ [1, 100]
a_2
具体空间
抽象空间

区间域下的最佳(最精确)抽象



最佳抽象的形式化定义

- 给定具体域C和抽象域$\mathcal{A}$之间的Galois连接

$$(C, \subseteq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathcal{A}, \leq)$$

- 具体迁移函数 $F: C \to C$ 的最佳近似是 $F^{\#}$

$$F^{\#} \equiv \alpha \circ F \circ \gamma$$

问题: 以上 $F^{\#}$的定义是“非构造性”的

报告提纲

抽象解释理论

符号抽象框架

个人近期工作

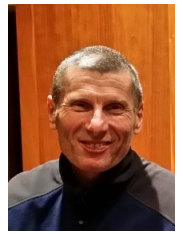
一些相关问题

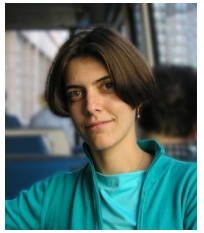
符号抽象: 计算最佳抽象的框架

- *Symbolic implementation of the best transformer* [RSY, VMCAI'04]

输入: (1) 编码了程序具体语义的约束 φ; (2) 抽象域$\mathcal{A}$,
输出: (2) 抽象领$\mathcal{A}$ 中能***over-approximate*** φ的最小元素


Thomas Reps


Mooly Sagiv


Greta Yorsh

符号抽象: 逻辑意义上的刻画

- *Automating abstract interpretation* [Reps & Thakur, VMCAI'16]

$\mathcal{L}$ 是表达力丰富的逻辑, 抽象域$\mathcal{A}$ 对应于$\mathcal{L}$的一个子集 $\mathcal{L}'$.
给定约束$\varphi \in \mathcal{L}$, 找到它在$\mathcal{L}'$中的*最强逻辑后承*

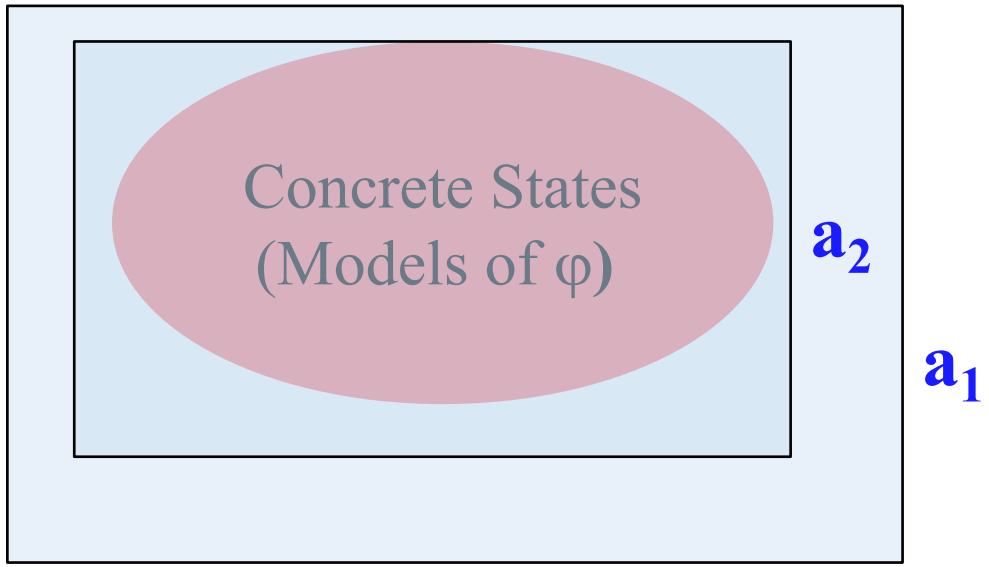
Thomas Reps

Aditya Thakur

例: 符号抽象

- *Automating abstract interpretation* [Reps & Thakur, VMCAI'16]

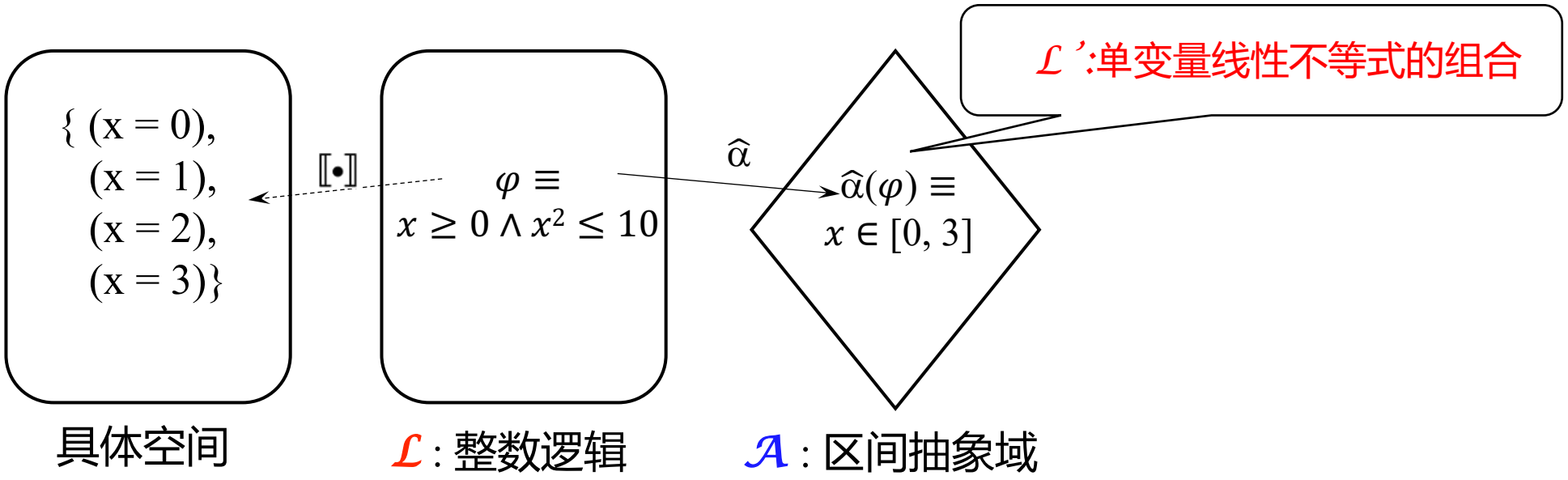
$\mathcal{L}$ 是表达力丰富的逻辑, 抽象域$\mathcal{A}$ 对应于$\mathcal{L}$的一个子集 $\mathcal{L}'$
给定约束$\varphi \in \mathcal{L}$, 找到它在$\mathcal{L}'$中的*最强逻辑后承*



例: 符号抽象

- *Automating abstract interpretation* [Reps & Thakur, VMCAI'16]

$\mathcal{L}$ 是表达力丰富的逻辑, 抽象域$\mathcal{A}$ 对应于$\mathcal{L}$的一个子集 $\mathcal{L}'$
给定约束$\varphi \in \mathcal{L}$, 找到它在$\mathcal{L}'$中的*最强逻辑后承*



符号抽象的理论和实际意义

```
#include <algorithm>
#include <boost/bimap.hpp>
#include <boost/optional.hpp>
name
name
usin
/*
 * T
 * 1
 * 2
 *
 * F
 * s
 * s
 *
 * F
 */
temp
clas
  us
  us
  us
publ
  us
  us
  us
  us
  us
  us
```

```
                                << Params::ldd_size << "\n";);
      s_ldd_man = Ldd_Init(cudd, theory);
      if (Params::dynamic_reordering) {
        Cudd_AutodynEnable(cudd, CUDD_REORDER_GROUP_SIFT);
        // XXX: the value of Params::convexify_threshold is quite arbitrary. A
        // good value seems around 1000000
        unsigned threshold = num_of_vars() * Params::convexify_threshold;
        // unsigned num_paths = Ldd_PathSize(NULL, &m_ldd);
        CRAB_LOG("boxes", boxes_domain_t tmp(*this); crab::outs()
                                                  << "Performed complement \n"
                                                  << "**" << tmp << "\n"
                                                  << "= " << res << "\n";);

      return res;
    }

    void operator-=(const
      crab::CrabStats::cou
      crab::ScopedCrabStat

      if (is_bottom() || i
        return;

      int id = get_var_dim(var);
      m_ldd =
          lddPtr(get_ldd_man(), Ldd_ExistsAbstract(get_ldd_man(), &*m_ldd, id));
    }

    void forget(const variable_vector_t &variables) override {
      crab::CrabStats::count(domain_name() + ".count.forget");
      crab::ScopedCrabStats __st__(domain_name() + ".forget");

      if (is_bottom() || is_top())
        return;

      std::vector<int> qvars;
      qvars.reserve(variables.size());
      for (variable_t v : variables) {
        qvars.push_back(get_var_dim(v));
      }

      m_ldd = lddPtr(get_ldd_man(), Ldd_MvExistAbstract(get_ldd_man(), &*m_ldd,
                                                        &qvars[0], qvars.size()));
    }

    void project(const variable_vector_t &variables) override {
```

...

```
bool BilateralAnalyzer::strongestConsequence(AbstractValue* result,
                                             z3::expr phi,
                                             const ValueMapping& vmap) const {
    z3::solver solver(phi.ctx());
    solver.add(phi);
    auto lower = std::unique_ptr<AbstractValue>(result->clone());
    result->havoc();
    while (!((*result) <= (*lower))) {
        auto p = std::unique_ptr<AbstractValue>(lower->clone());
        p->abstractConsequence(*result);

        solver.push();
        solver.add(!p->toFormula(vmap, phi.ctx()));

        } else {
            auto cstate = ConcreteState(vmap, solver.get_model());
            if (lower->updateWith(cstate))
                changed = true;
        }
        solver.pop();
    }
    return changed;
```

障碍: 性能问题导致较少用在真实分析器中!

符号抽象的(少量)应用

- 源代码分析
  - 形态分析 [RSY, VMCAI'04; Yorsh et al., SAS'04]
  - 数值验证 [Li et al., POPL'14; Jiang el al., VMCAI'17]

- 二进制分析
  - 区间和集合域 [Barrett et al., ENTCS'10; Brauer et al., ESOP'11]
  - 线性等式域 [King et al., VMCAI'10; Thakur et al., SAS'12; Elder et al., TOPLAS'14]

应用于**工业级**二进制分析产品

GRAMMATECH

在形式化/编程语言、甚至抽象解释社区都还不够普及

报告提纲

抽象解释理论

符号抽象框架

个人近期工作 (这部分略过)

一些相关问题

报告提纲

抽象解释理论

符号抽象框架

个人近期工作

一些相关问题

符号抽象相关问题

$\mathcal{L}$ 是表达力丰富的逻辑, 抽象域$\mathcal{A}$ 对应于$\mathcal{L}$的一个子集 $\mathcal{L}'$.
给定$\varphi \in \mathcal{L}$, 找到它在$\mathcal{L}'$中的最强逻辑后承

- **形式化方法:** 谓词抽象 [Graf & Saidi 97]
  - $\varphi \in \mathcal{L}$: 编码了一个(不带循环的)程序块的约束
  - $\mathcal{L}'$: 给定谓词集合$\{P_1,\ldots, P_n\}$的任意布尔组合

Software Model Checking领域的“开山之作”

符号抽象相关问题

$\mathcal{L}$ 是表达力丰富的逻辑, 抽象域$\mathcal{A}$ 对应于$\mathcal{L}$的一个子集 $\mathcal{L}'$.
给定$\varphi \in \mathcal{L}$, 找到它在$\mathcal{L}'$中的最强逻辑后承

- **形式化方法**
  - 谓词抽象 [Graf & Saidi 97] $\mathcal{L}'$: 给定谓词集合$\{P_1,\ldots, P_n\}$的布尔组合
- **自动推理:** 量词消去
  - $\varphi \in \mathcal{L}$: 任意形式的整数/实数/浮点/字符/...约束
  - $\mathcal{L}'$: 只使用给定约束变量集合 $S \subseteq Vars(\varphi)$的约束

符号抽象相关问题

$\mathcal{L}$ 是表达力丰富的逻辑, 抽象域$\mathcal{A}$ 对应于$\mathcal{L}$的一个子集 $\mathcal{L}'$.
给定$\varphi \in \mathcal{L}$, 找到它在$\mathcal{L}'$中的最强逻辑后承

- **形式化方法**
  - 谓词抽象 [Graf & Saidi 97] $\mathcal{L}'$: 给定谓词集合$\{P_1,\ldots, P_n\}$的布尔组合
- **自动推理**
  - 量词消去 $\mathcal{L}'$: 使用给定约束变量集合 $S \subseteq Vars(\varphi)$的约束
- **人工智能**
  - Forgetting [Lin & Reita 94] $\mathcal{L}'$: 类似量词消去问题

符号抽象相关问题

$\mathcal{L}$ 是表达力丰富的逻辑, 抽象域$\mathcal{A}$ 对应于$\mathcal{L}$的一个子集 $\mathcal{L}'$.
给定$\varphi \in \mathcal{L}$, 找到它在$\mathcal{L}'$中的最强逻辑后承

- **形式化方法**
  - 谓词抽象 [Graf & Saidi 97] $\mathcal{L}'$: 给定谓词集合$\{P_1,\ldots, P_n\}$的布尔组合
- **自动推理**
  - 量词消去 $\mathcal{L}'$: 使用给定约束变量集合 $S \subseteq Vars(\varphi)$的约束
- **人工智能**
  - Forgetting [Lin & Reita 94] $\mathcal{L}'$: 类似量词消去问题
- **运筹学**
  - 线性规划 $\mathcal{L}$ 线性约束集合; $\mathcal{L}'$: $f(X) \leq c$ (f是线性目标函数)

总结与展望: 程序合成视角

Automating abstract interpretation [Reps & Thakur, VMCAI'16]

程序合成: 如何自动从规约中生成程序

"One of the most central problems in the theory of programming."

----Amir Pneuli

1996年图灵奖获得者

"(软件自动化)提升软件生产率的根本途径"

----徐家福先生

中国软件先驱

感谢聆听！

pyaoaa@zju.edu.cn

部分胶片参考自陈立前、熊英飞等老师的报告