

Manufacturing and Mathematics (ものづくりと数学) - Symbolic Approach -

Ryusuke Masuoka (益岡 竜介)
Fujitsu Laboratories of America, Inc. (FLA)
(米国富士通研究所)
October 21, 2011

数学に基づく設計・開発

(1) モデル化



(2) 解析



(3) 設計



(4) 検証

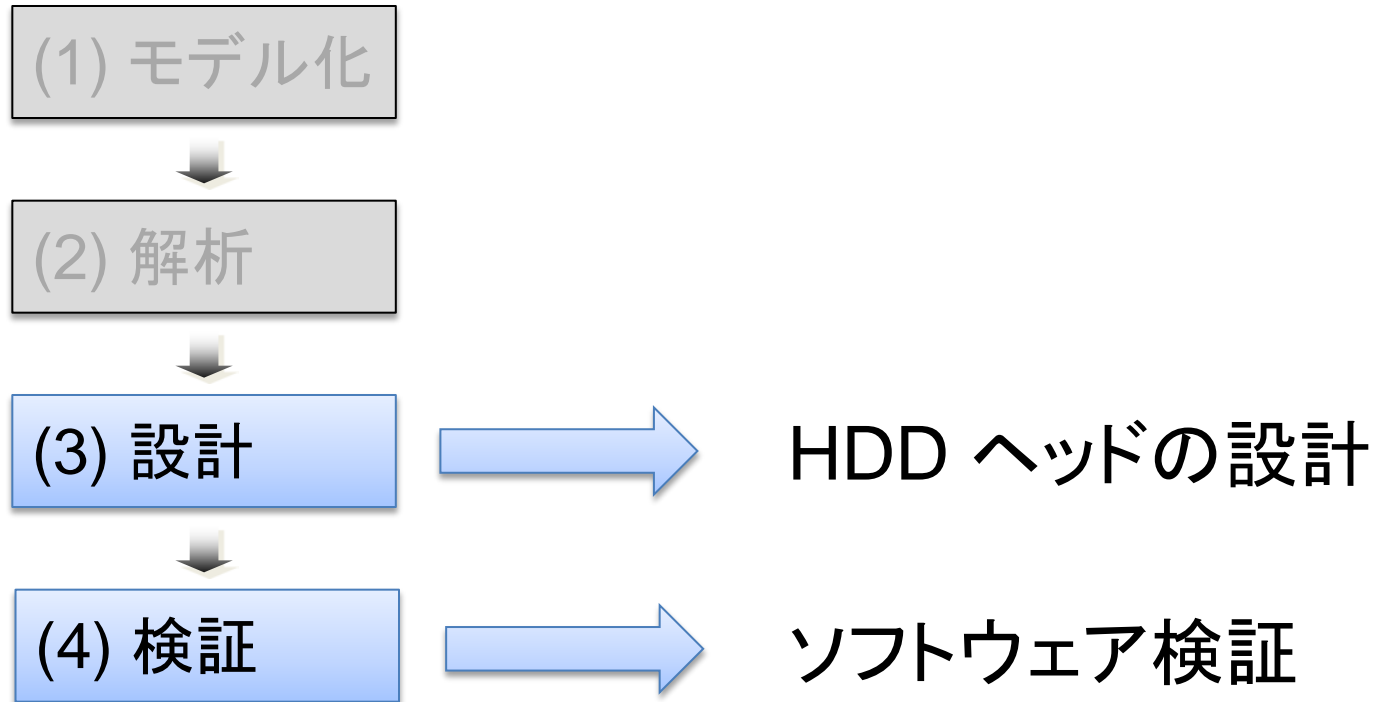
- どのように対象のシステムをモデル化すべきか?

- どのような解析をそのシステムで行うべきか?

- 問題をどのように定式化し、何を設計目的とするか?
- どのように設計を行う (設計問題を解く) か?

- システムの設計は正しいか、不具合は?
- システムが望ましくない状態に陥らないか?

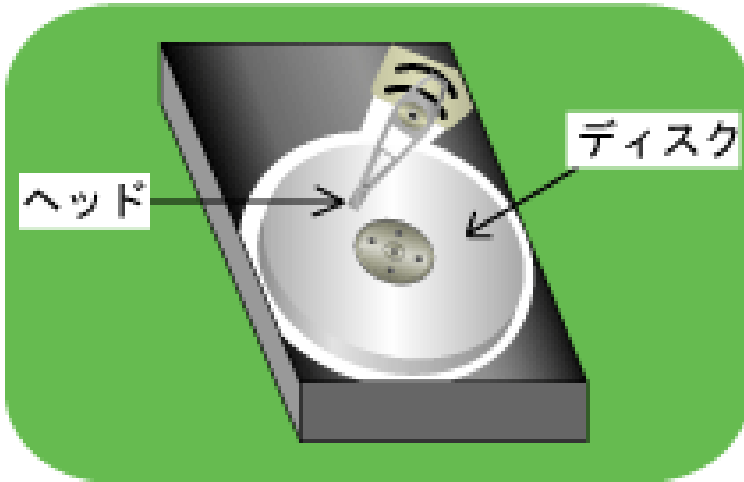
数学に基づく設計・開発



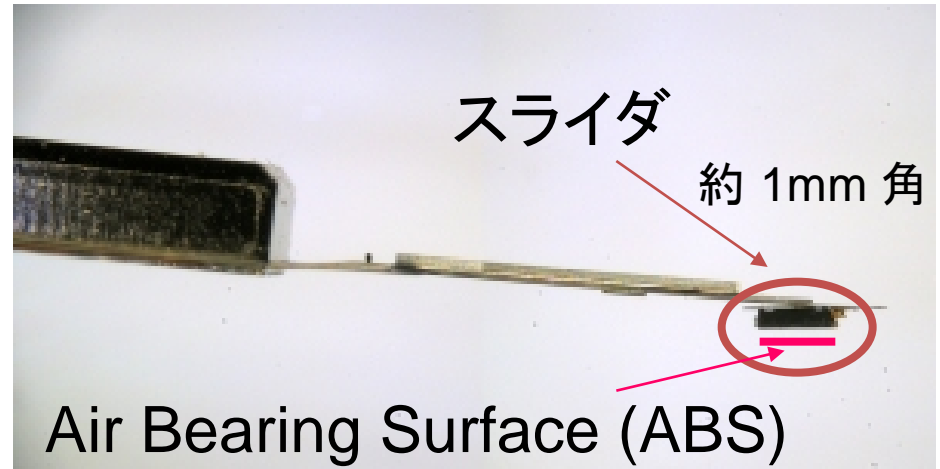
HDD: Hard Disk Drive

HDD ヘッドの設計

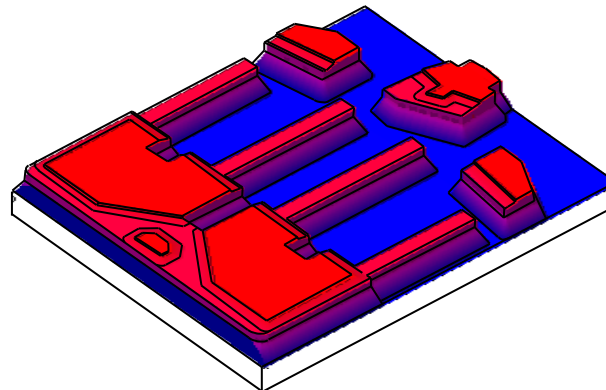
HDD



ヘッド部分の拡大図



スライダ拡大図 (上面が ABS)



■ スライダの役割

- 先端の磁気ヘッドで情報の読み取り／書き込みを行う
 - ディスクに対する相対位置が重要
 - ディスクに近いほど読み取り／書き込みエラー少ない
 - ディスクに接触するとクラッシュの原因に

■ スライダとディスクの相対位置

■ 浮上量

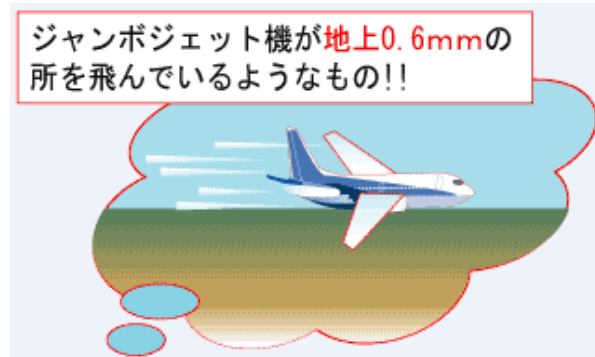
- スライダはディスクの回転で生じる空気の流れて 10 ~ 20 nm 程度浮上
- 高度(気圧)などの環境変化により浮上量は変化

■ 角度

- アームの位置により空気の流れが変わる
- スライダに縦・横方向への回転が生じる

■ ABS 形状設計

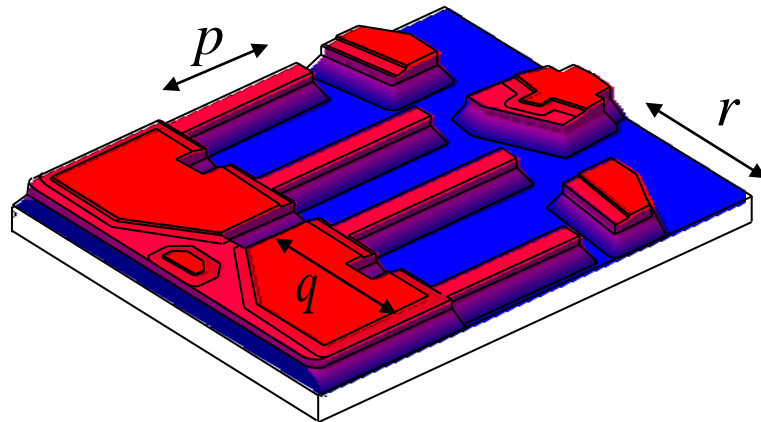
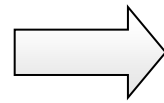
- ABS に溝を掘ることでスライダの浮上を制御
- 浮上量が小さく、安定した位置を保てる ABS 形状を見つける



スライダをジャンボジェット機の大きさに拡大すると…

■ ABS形状 $x = (p, q, r, \dots)$

p, q, r, \dots はスライダ形状を決めるパラメタ



浮上計算(レイノルズ方程式)

スライダ形状から、
フライハイト (@0m, @4000m)、ロール、ピッチ...
を計算

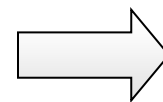


目的関数
(コスト関数)

フライハイト (@0 m, @4000m)
ロール
ピッチ
...

■ ABS 形状最適化

目的関数 (たち) を小さくする
ABS 形状 (パラメタ x) を求める



多目的最適化

$$\begin{cases} \min_{x \in \mathbb{R}^N} & f = \{f_1(x), \dots, f_M(x)\} \\ \text{subject to} & g(x) \leq 0 \\ & h(x) = 0 \end{cases}$$

Minimize $F = (f_1(x, y), f_2(x, y))$

$$f_1(x, y) = 4x^2 + 4y^2$$

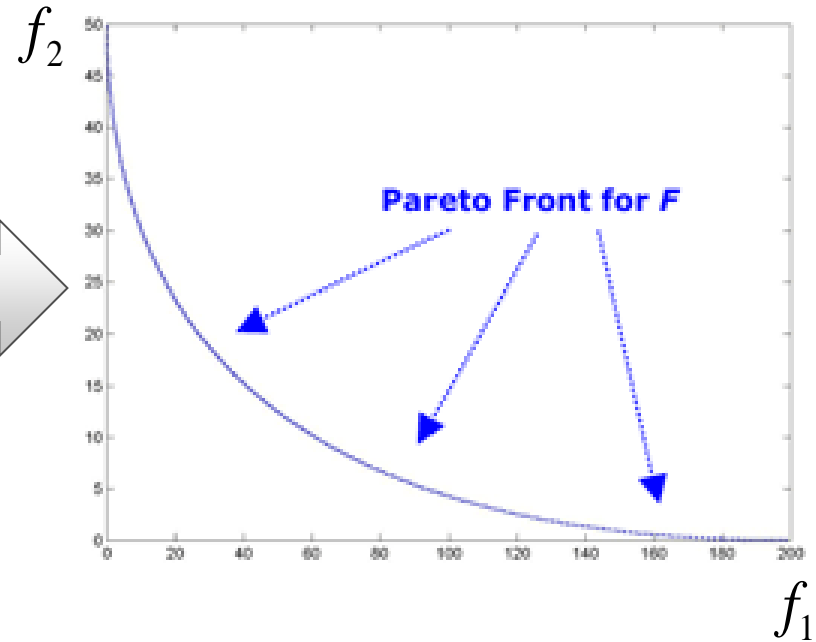
$$f_2(x, y) = (x - 5)^2 + (y - 5)^2$$

$$0 \geq (x - 5)^2 + y^2 - 25$$

$$0 \geq -(x - 8)^2 - (y + 3)^2 + 7.7$$

$$0 \leq x \leq 5$$

$$0 \leq y \leq 3$$



数打ちやあたる

限定記号を使った定式化

Minimize $F = (f_1(x, y), f_2(x, y))$

$$f_1(x, y) = 4x^2 + 4y^2$$

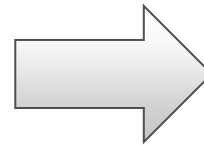
$$f_2(x, y) = (x - 5)^2 + (y - 5)^2$$

$$0 \geq (x - 5)^2 + y^2 - 25$$

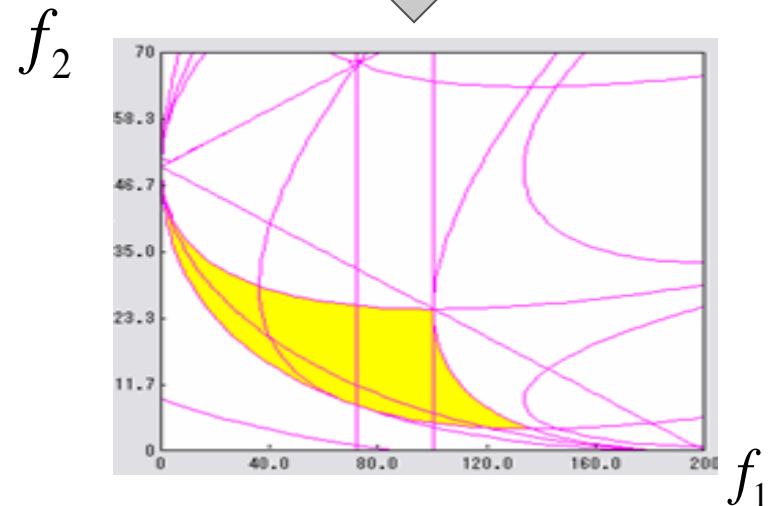
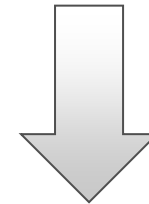
$$0 \geq -(x - 8)^2 - (y + 3)^2 + 7.7$$

$$0 \leq x \leq 5$$

$$0 \leq y \leq 3$$



$$\begin{aligned} & \exists x \exists y (4x^2 + 4y^2 - f_1 = 0 \wedge \\ & (x - 5)^2 + (y - 5)^2 - f_2 = 0 \wedge \\ & (x - 5)^2 + y^2 - 25 \leq 0 \wedge \\ & -(x - 8)^2 - (y + 3)^2 + 7.7 \leq 0 \wedge \\ & x \geq 0 \wedge x \leq 5 \wedge \\ & y \geq 0 \wedge y \leq 3) \end{aligned}$$



■ パラメトリックな最適化の基礎アルゴリズム

■ 限量記号消去 (Quantifier Elimination)

入力

限定記号がついた論理式

例

$$\forall x (x \mid$$

$$\exists x (a \mid$$

$$\forall x \exists y (x^2 + xy + b > 0 \wedge \\ x + ay^2 + b \leq 0)$$

出力

限定記号がない等価な論理式

$$b \mid$$

$$(a \neq 0 \wedge b^2 - 4ac \geq 0) \vee$$

$$(a = 0 \wedge b \neq 0) \vee$$

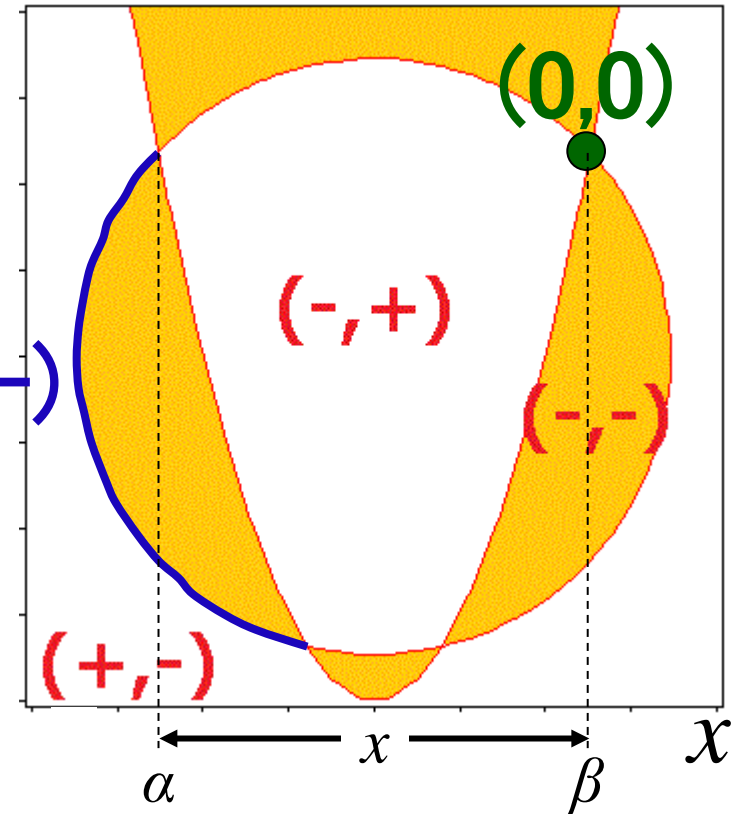
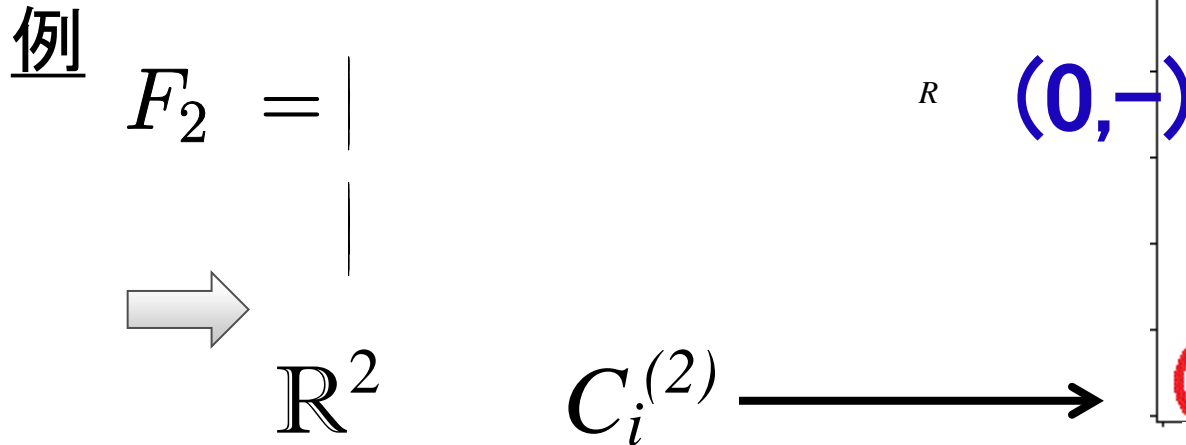
$$(a = 0 \wedge b = 0 \wedge c = 0)$$

$$a \mid$$

■ QE の基礎となる代数的アルゴリズム

■ G. E. Collins が導入 (1975年) y

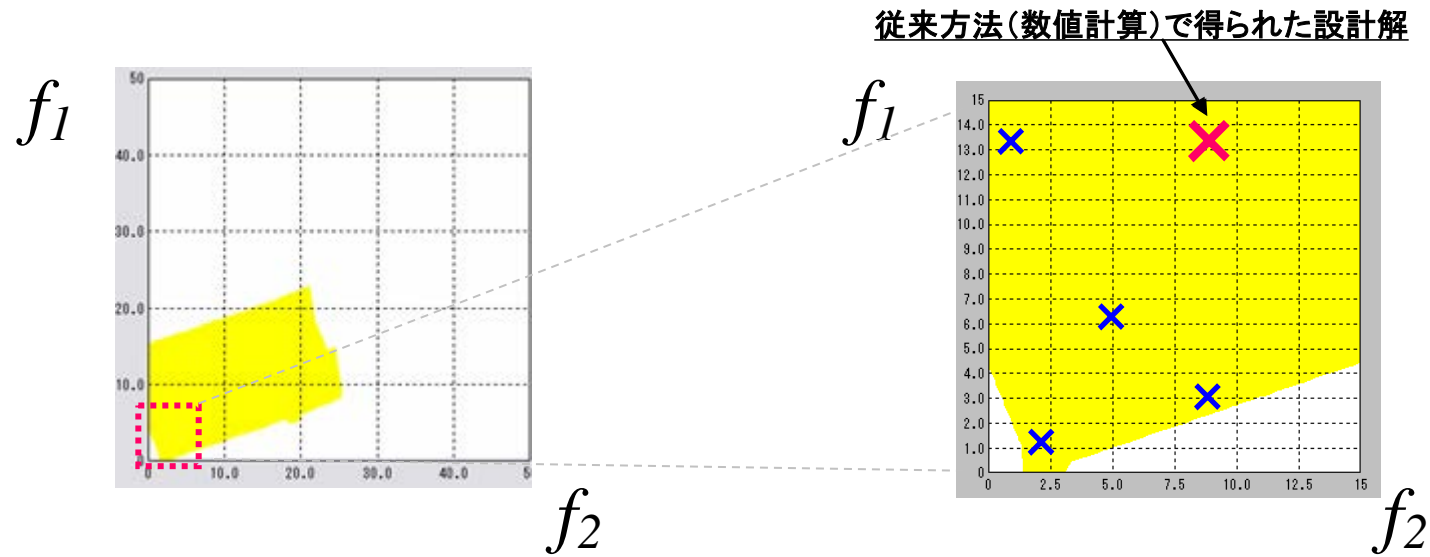
- 入力: $F_r \subset \mathbb{Q}[x_1, \dots, x_r]$
- 出力: 入力の多項式が符号不変で互いに交わらない集合 (セル) に分解



問 以下と同値な式をもとめよ

$$\exists y (x^2 + y^2 - 3 < 0 \wedge y - 2x^2 + 2 > 0)$$

答 $\alpha < x < \beta$



$\{f_1, f_2\}$ の可能領域の正確な可視化により、
多目的最適化を見通しよく実現



CAD, VS, SDC

Fujitsu Laboratories Ltd.

(H.Yanami, H.Iwane, H.Anai)



CAD, VS

Wolfram Research, Inc.

(A.Strzebonski)



CAD

RISC-Linz + etc.

(G.Collins, H.Hong, C.Brown)



CAD, VS

Univ. Passau

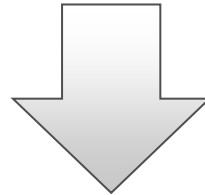
(T.Sturm, A.Dolzmann, A.Seidl)

- 穴井 宏和他, “数式処理を用いた設計技術,” 雑誌 Fujitsu, Vol.60, No.5, pp. 514 – 521, Sep. 2009
- 穴井 宏和, 横山 和弘, “QEの計算アルゴリズムとその応用 ～数式処理による最適化,” 東京大学出版会, Aug. 2011



ソフトウェア検証

What day is September 9th?




“Debugging Day!”

9/9

0800 Andan started
 1000 " stopped - andan ✓
 1300 (032) MP-AC { 1.2700 9.032 847 025
 033) PRO 2 2.130476415 9.037 846 995 const
 const 2.130676415 4.615925059(-2)
 Relays 6-2 in 033 failed special speed test
 in relay 11.000 test.
 Relays changed

1100 Started Cosine Tape (Sine check)
 1525 Started Multy Adder Test.

1545  Relay #70 Panel F
 (moth) in relay.

First actual case of bug being found.

1630 Andan started.
 1700 closed down.

Relay 2145
 Relay 3370

Is a Bug an Inconvenience?



Calls dropped

- 1990 AT&T 4ESS long-distance switches

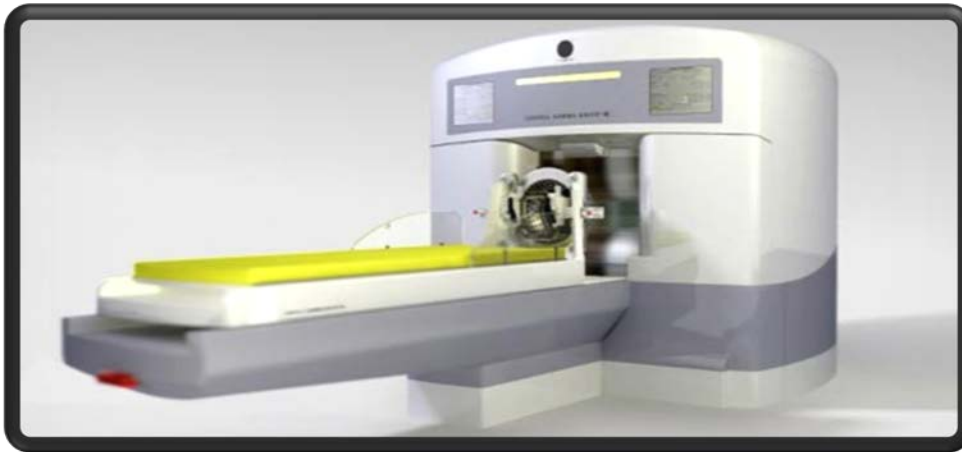
Money lost

- 2010 German Bank Card
Year 2010 Bug



Lives lost

- 1985 Therac-25
Medical Accelerator



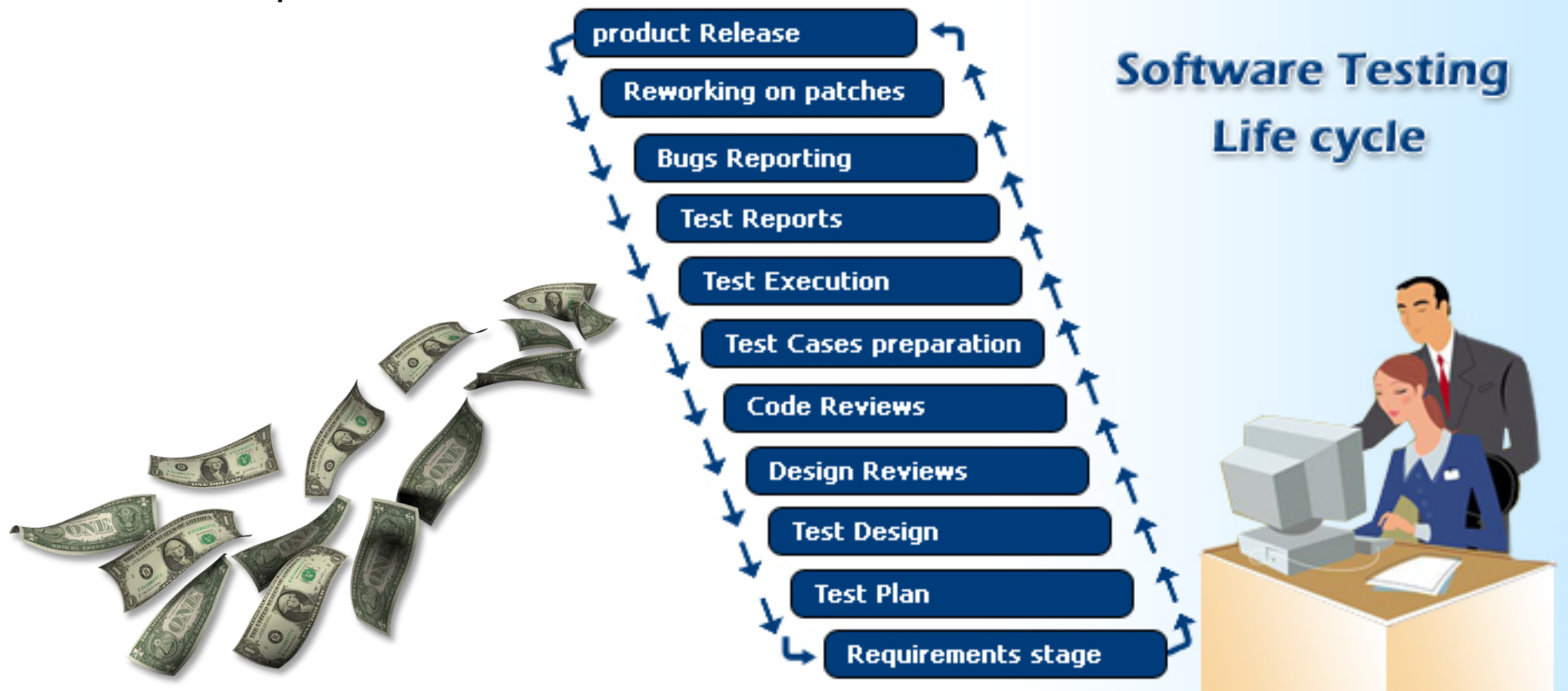


False Alarms

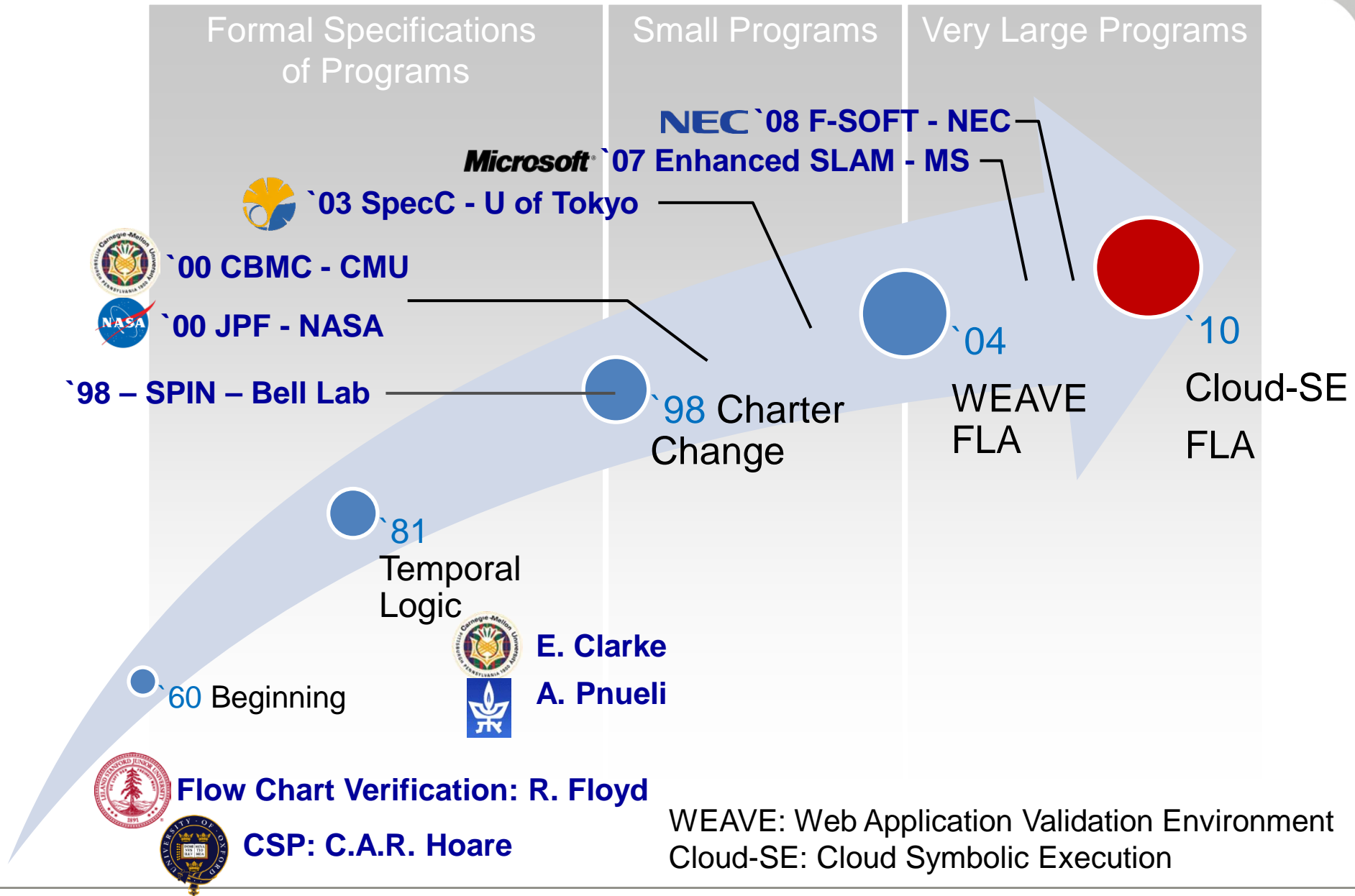
- 1980 NORAD
- 1983 Soviet Union

Testing, Of Course, But ... It Is Expensive

- Largely manual labor-intensive process
- Software getting larger and more complex every day
 - Beyond human capacity to grasp!
 - More expensive

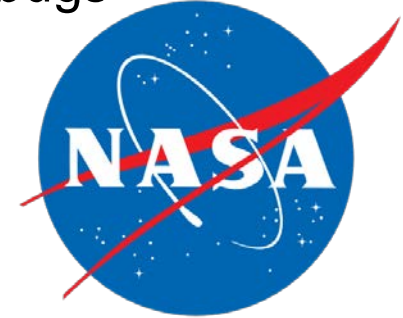


History of Software Validation



■ NASA

- Why NASA? - History of blowing up rockets to software bugs
- Checking Toyota software
- FLA closely collaborating – JPF, String



■ Fujitsu

- As a provider of ICT for critical social infrastructure

■ IBM, NEC, HP, Intel, Siemens, MS

■ Startups

- Coverity, RealIntent

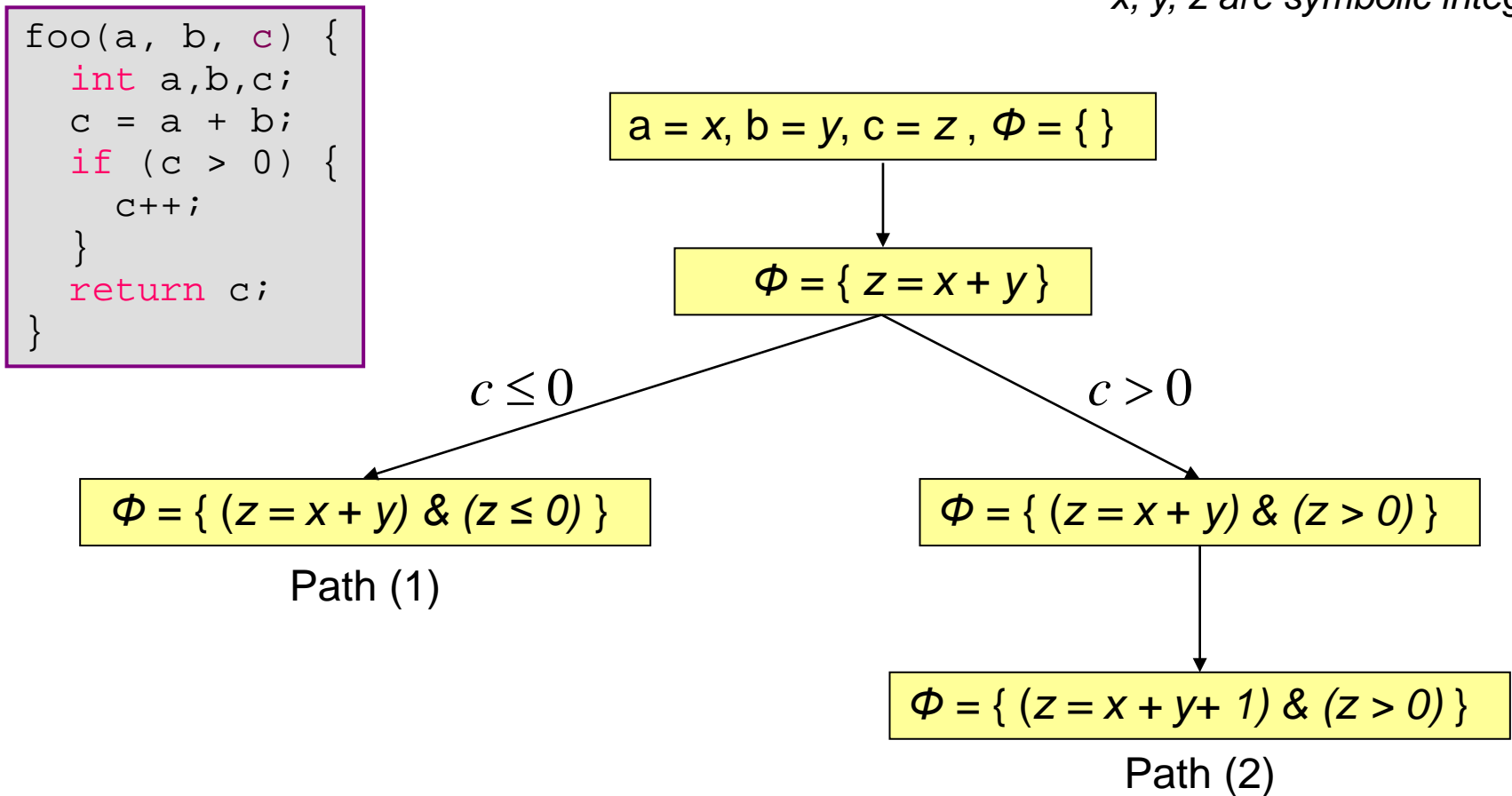
■ Universities

- UC Berkeley, Oxford Univ., Stanford, CMU, NYU, UT Austin, Kansas U
- University of Tokyo, JAIST

- Calculate whether a system satisfies a certain behavioral property:
 - Is the system deadlock free?
 - Whenever a packet is sent will it eventually be received?
- So it is like testing? No, there is a major difference:
 - Try exploring *all* possible behaviors of a system in *one* run
 - Two different techniques:
 - Model checking (suitable for hardware, finite state)
 - Symbolic execution (suitable for software, infinite state)
- Automation
 - Push-button technology needs deep algorithmic research
 - Most important for commercial acceptance

- Each path in the tree represents a (possibly infinite) set of execution paths

Φ is symbolic expression
 x, y, z are symbolic integers



- Property to check : if $((a > 1) \& (b > 0)) \rightarrow (c > 4)$
- Negate property:
 - if $((a > 1) \& (b > 0)) \rightarrow (c \leq 4)$
- Check at the end of each path of the symbolic execution

Equations at the end of path (1)

$x > 1$
 $y > 0$ } Preconditions
 $z = x + y$
 $z \leq 0$
 $z \leq 4$ } Post condition

Solve using ILP
- No solutions
- Property holds

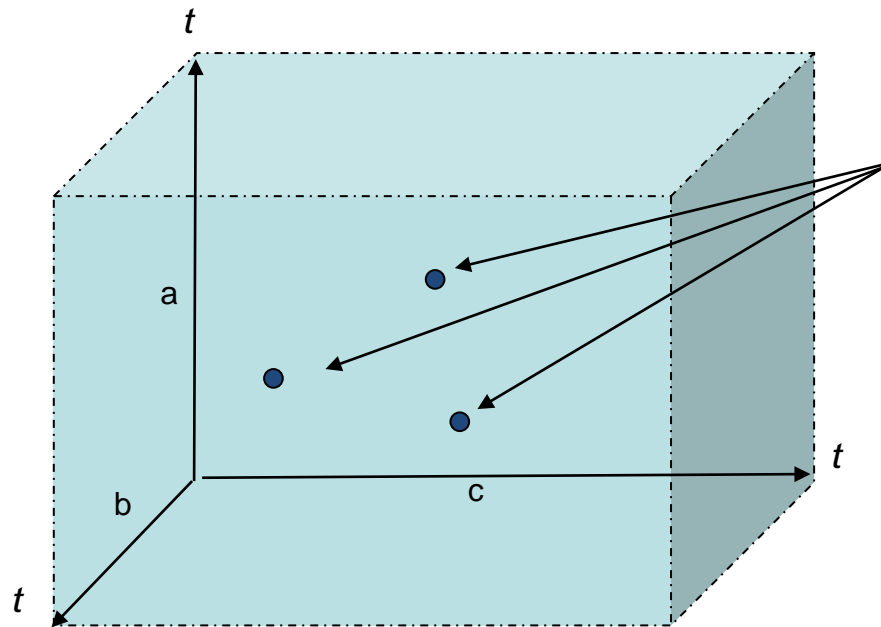
Equations at the end of path (2)

$x > 1$
 $y > 0$ } Preconditions
 $z = x + y + 1$
 $z > 0$
 $z \leq 4$ } Post condition

Solve using ILP
- **SOLUTION FOUND !!**
- Counter example: $x = 2, y = 1, z = 4$

- Bug Uncovered: if $a = 2$ and $b = 1$ then $c > 4$ does not hold in this path

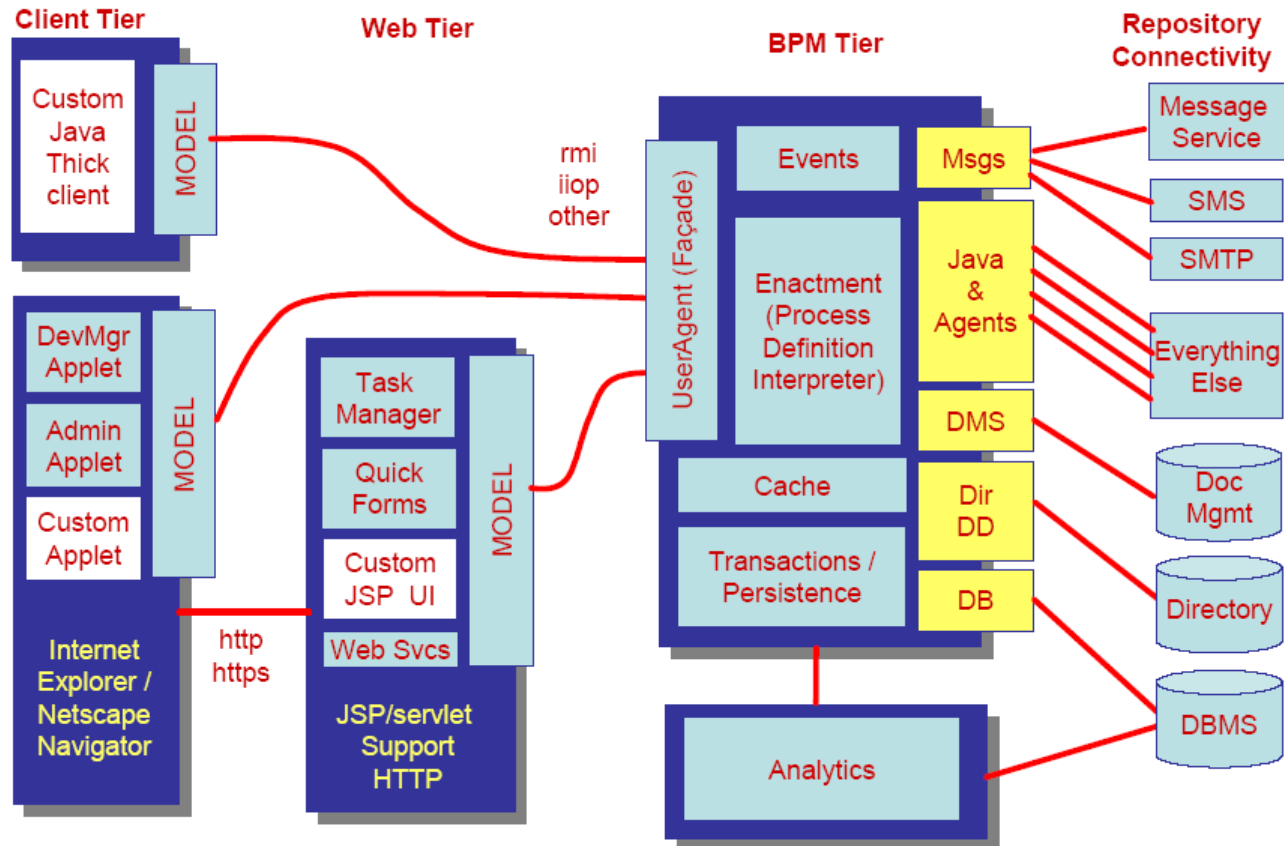
- Solution of these symbolic expressions can give concrete traces representing all distinct paths in the program tree
 - Complete input space examined in one shot



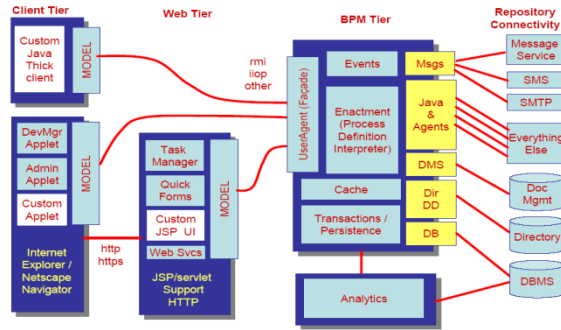
Specific simulation test cases for (a, b, c);
[Each test case examines only one point in input space]

But, Industry Software too Large to Process

Scenario: Given a large software code base, consisting of interacting heterogeneous components such as client, server, and database. Its server “locks up” some times and clients receive incorrect data

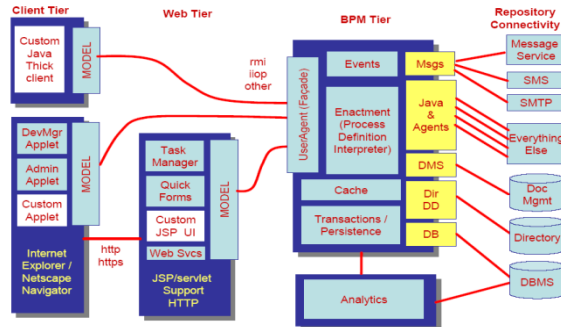
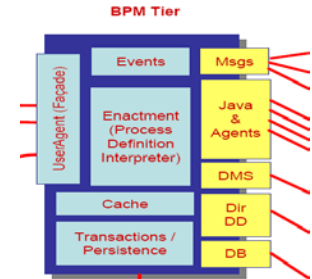
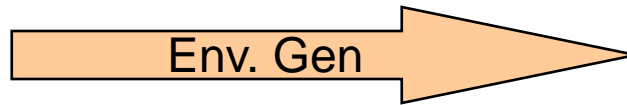


Three-Pronged Approach for Reduction



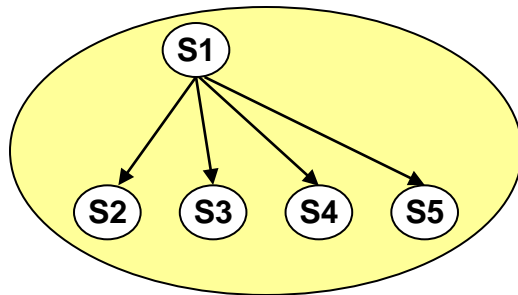
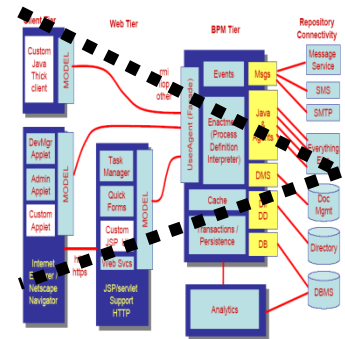
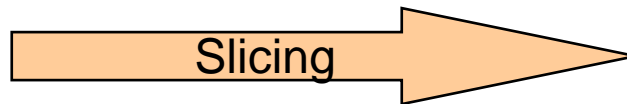
Environment Generation

Reduce scenarios of the system behavior



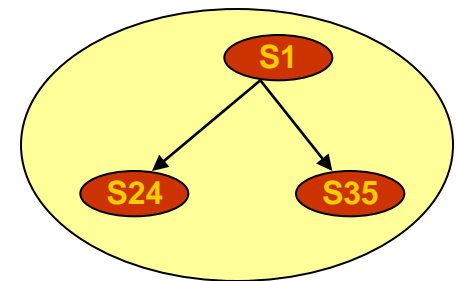
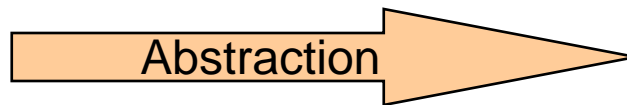
Slicing

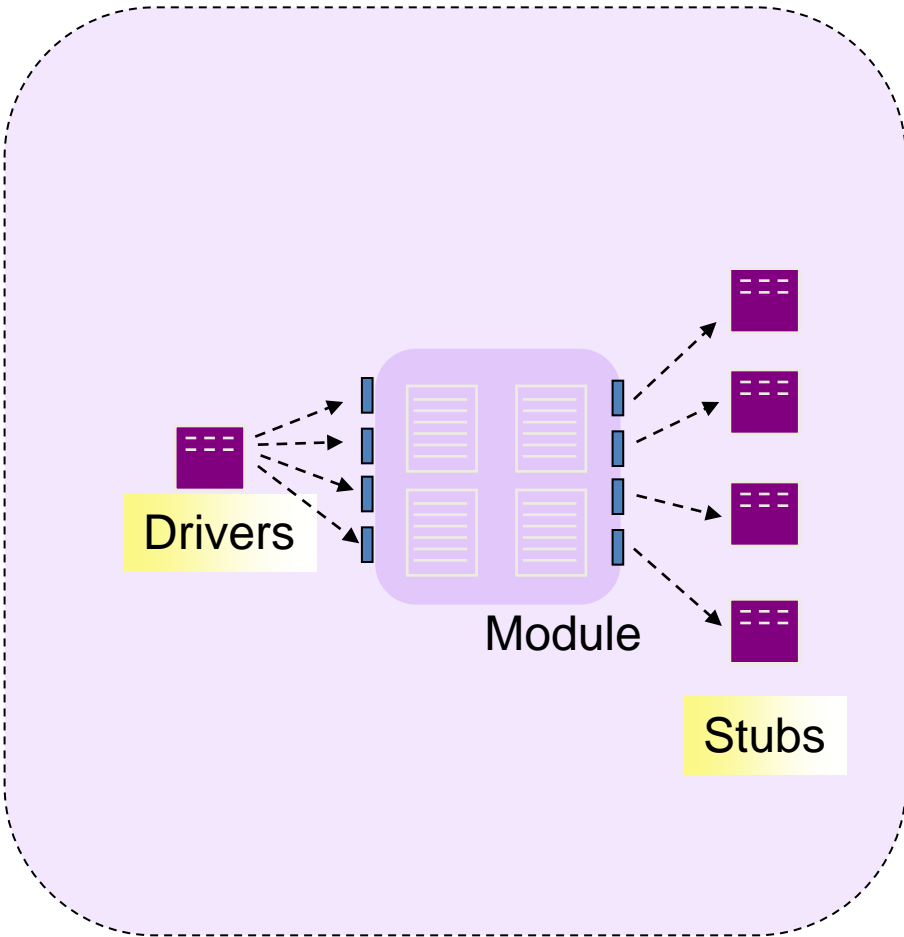
Reduce irrelevant statements through static analysis



Abstraction

Reduce the state space by consolidating states





Code Base

- Drivers (Test harness, test drivers)
 - Call the module
 - Java classes/threads that instantiate classes inside the module and perform sequences of method calls to the module

- Stubs
 - Are called by the module
 - Java classes/methods called by classes inside the module, e.g., library methods

Slicing: Reduces Code Size to Process

- Chop irrelevant program statements through static analysis

Example below taken from slide 4 of
“The Bandera Model Reduction Tools”
[Linked from “slicing in Bandera” in
<http://bandera.projects.cis.ksu.edu/talks.shtml>]

Raise m to the power of n :

```
init: m      := 5;      [init. 1]
      n      := 2;      [init. 2]
      result := 1;      [init. 3]
      goto test;
test: if (n<1)
      then end
      else loop;
loop: result := result*m; [loop. 1]
      n := n-1;          [loop. 2]
      goto test;        [loop. 3]
end:  return;           [end. 1]
```

Data dependence

Control dependence

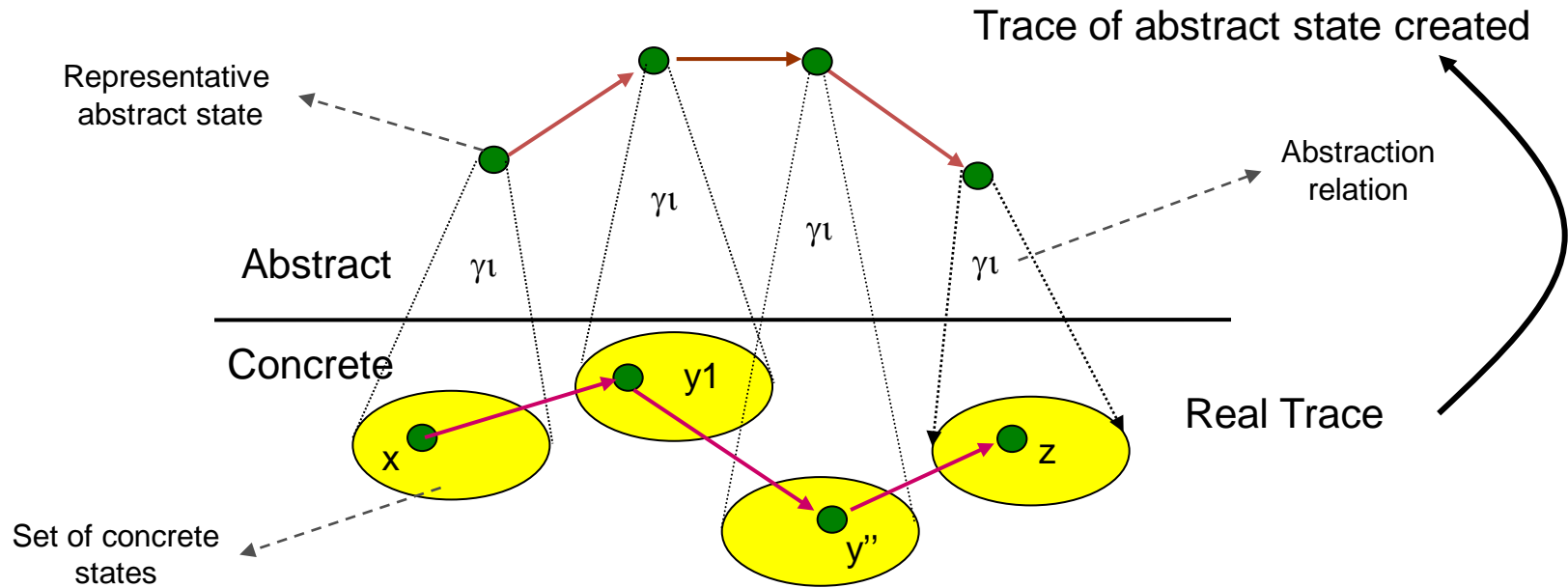
Original Program

```
init:      n      := 2;      [init. 2]
goto test;
test: if (n<1)
      then end
      else loop;
loop:      n := n-1;          [loop. 2]
      goto test;        [loop. 3]
end:  return;           [end. 1]
```

Sliced Program

Slicing Criterion: $C = \{ [loop. 2] \}$

Abstraction: Reduces State Space



- Abstract state transition relation is conservative
 - Abstract next states *must* contain all concrete successors
 - And possibly more states

Abstraction: Example

Example Code

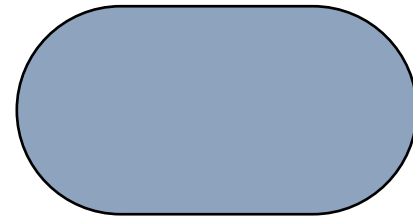
```
int x = 0;
if (x == 0)
    x = x + 1;
```



```
Signs x = ZERO;
if (Signs.eq(x, ZERO))
    x = Signs.add(x, POS);
```

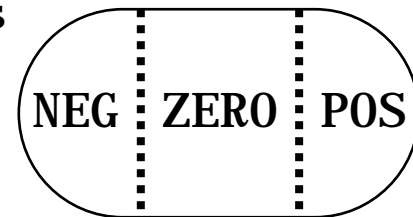
Data Type Abstraction

int

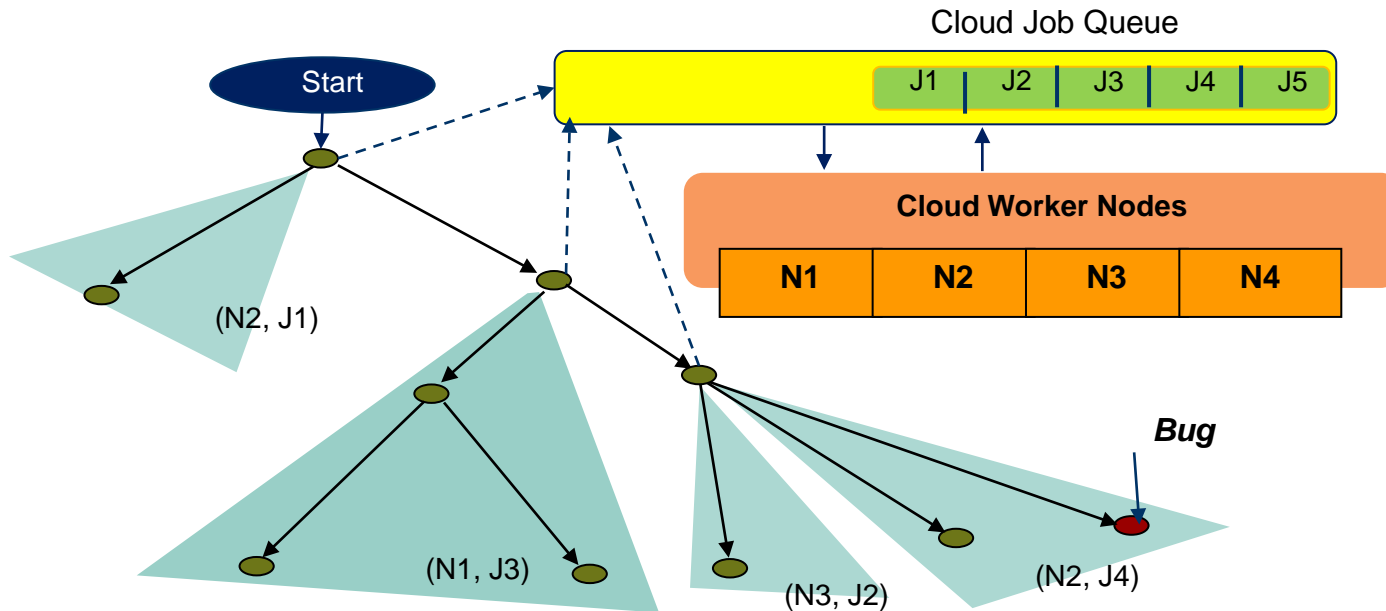


(n<0) : NEG
(n==0) : ZERO
(n>0) : POS

Signs



One Last Trick: Distribution on Cloud



- Distributed it over compute nodes in cloud to enhance scalability

Performance improvement: 15x speed-up with 25 nodes

Conclusion (Software Validation)

To check suspicious modules

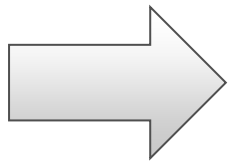
To reduce computational loads

Symbolic Execution

**Environment Generation
Slicing
Abstraction**

+
On Cloud


To enhance scalability



Software validation for large software

Future Work: Higher Branch Coverage

- “Epic failures: 11 infamous software bugs,” *ComputerWorld*, Sep. 9, 2010, <http://www.computerworld.com/s/article/9183580/>
- “The Year 2010 Bug Strikes German Bank Cards,” *TIME*, Jan. 7, 2010, <http://www.time.com/time/business/article/0,8599,1952305,00.html>
- “10 historical software bugs with extreme consequences,” *Royal Pingdom*, Mar. 9, 2009, <http://royal.pingdom.com/2009/03/19/10-historical-software-bugs-with-extreme-consequences/>
- D'Silva, V., Kroening, D. & Weissenbacher, G., "A Survey of Automated Techniques for Formal Software Verification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, July 2008.
- Rajan, S. P., Tkachuk, O., Prasad, M. R., Ghosh, I., Goel, N., & Uehara T., "WEAVE: WEb Applications Validation Environment," *ICSE 2009*
- Rajan, S. P., Ghosh, I., Tkachuk, O., Prasad, M. R., Murthy, P. K., Masuoka, R., Uehara, T., Munakata, K., Oki, K., & Hara, H., "Software Applications Validation Environment: SAVE," *Fujitsu Scientific & Technical Journal (FSTJ)*, 2007-10, 2007-10 (Vol.43, No.4)



FUJITSU

shaping tomorrow with you

WORKING IN US

■ India, China, Korea, Taiwan, Columbia, France, Spain, Russia, Saudi Arabia, Nigeria, Vietnam, Iran, ... Of course, some from US

■ Status もばらばら、

■ Visa Status

- 前の代から/自分の代から/途中から米国市民
- グリーンカード (米国永住権)
- 学生ビザ、就労ビザ
- 一時訪問者

■ FLA での Status

- 研究者 (正社員)、テンプ、インターン (学生)、コンサルタント

■ 米国の特に研究開発現場では当たり前

■ 国防関係以外

- 米国でも以外にコネは重要
 - ただしそれから後は、自分の能力しだい
- High turnover
 - 出来る人ほどすぐいなくなる
 - 最先端なので、バックグラウンドのある人などいない
 - いかに満足度を高めるか
 - インターン: 新しい技術の習得、レジメ
- 使えるものは何でも使う
- 自主独立
 - 明確な Job Description
- 柔らかく、しかししっかりと、煙たがれない程度に頻繁に、手を変え、品を変え、押す



■ 管理職はサービス業

- 担当者が効率よく (気持ちよく) 仕事ができるように、段取りよく準備しておく
- 判断などをすばやく、はっきりと下す
 - 日本とのプロジェクトではここが難しい

■ 簡単に Okay というが ...

- 分担がはっきりしている ...
 - どこかで間に落ちて止まる
- 何度もチェックを入れる

■ 日本とのプロジェクト

- うまく回せば 24 時間動かし続けられる
- 逆に何日も無駄に過ぎることも



- 英会話だけで意図を伝えるのは難しい
 - 特にセマンティックスといった抽象的なものは ...
 - Visualization
 - モックアップ
- Email
 - 何でも文書にして、関係する範囲に流しておく
 - 後で言った、言わないがなくなる
 - 自分の記録にもなる
 - 打ち合わせなどのまとめを Email で送る
 - 聞き取れていない、聞き間違っている可能性を排除
 - 最初のパラグラフに用件を遠慮せずはっきりと書く
 - 背景や詳細は別に後ろに書く。
 - そういう文を読み書きする訓練を受けてきている
 - 何種類もの依頼と感謝の言い方を持っておく

■ 知的所有権 (IPR)

- 最近、大学も IPR に厳しくなってきた
- Open Source のソフト

■ 標準化

- 同じ船作戦

- 自分が何をやりたいかの明確かつ強い意志を持ち、
 - しかし頑迷にならず、大きな目的のためには柔軟に対処する
- 一度ではあきらめないで、
 - 手を変え、品を変え、試してみる
- 誠意を尽くし、がんばる
- 対立しているように見えても、考えればいくらでも両者のメリットになりうる形はありうる
 - 共同研究やインターン