

# Symbolic Extrapolation in Program Verification

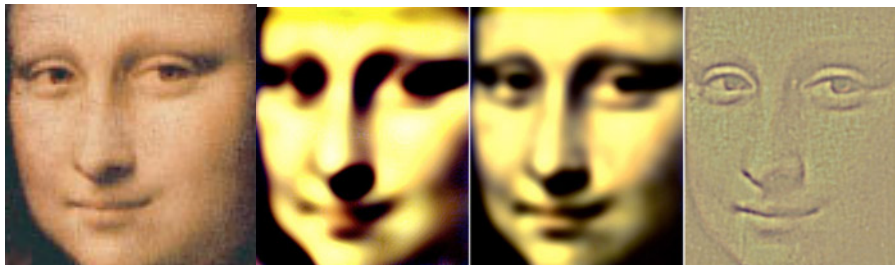
Vijay D'Silva, Daniel Kröning, David Basin, Daniel Kroening

ETH Zurich, University of Oxford

May 20, 2008

# Program Analysis 101: Abstract

Question: [Da Vinci, 1503-6] Does the Mona Lisa smile?

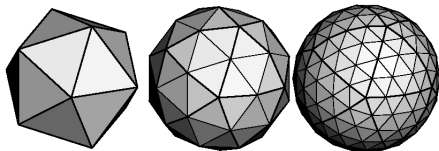


[Cousot & Cousot, POPL 1977] Abstract Interpretation: Approximate but sound answers.

Answer: [Livingstone, 2004] Yes, but only in low spatial frequencies.

# Program Analysis 101: Embrace Infinity

No finite abstraction for all problems. Infinite abstractions more expressive.



The price of infinity:

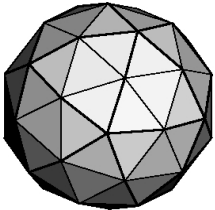
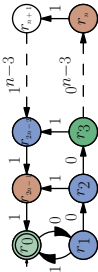
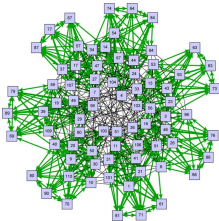
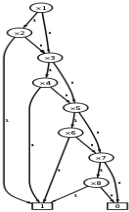
- Naïve iteration will not terminate.
- Limits may not exist.

[Cousot & Cousot, 1977] Widening: Accelerate convergence. Terminate. Guess the direction of growth and generalise. But how?

# Program Analysis 101: Be Symbolic

Enumerating facts is not efficient. Not possible for infinite sets.

- Deal with sets of facts.
- Smart data structures avoid redundancy.



Can we extract information from symbolic representations to cope with infinite computations? Can we *extrapolate*?

# Scoop of the Day

Abstraction: Regular Languages

Symbolic Representation: (Alternating) Finite Automata

Present an extrapolation framework

- Use binary relations and the quotient construction.
- Generate new operators by composition operations.
- Various instances exist in the literature.
- Study soundness, termination, representation issues.

# Extrapolation in Action

```
1 begin
2    $x \leftarrow 0$ 
3   repeat
4     if  $x$  is odd then
5        $x \leftarrow 2x$ 
6     else
7        $x \leftarrow 2x + 1$ 
8     end
9   until Eternity
10 end
11 assert(  $x \neq 511$  )
```

Encode as strings:

$\lambda, 1, 10, 101, 1010, 10101, \dots$

Values of  $x$ :

$0, 1, 2, 5, 10, 21, \dots$

# Extrapolation in Action

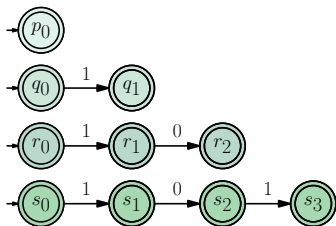
```
1 begin  
2    $x \leftarrow 0$   
3   repeat  
4     if  $x$  is odd then  
5        $x \leftarrow 2x$   
6     else  
7        $x \leftarrow 2x + 1$   
8     end  
9   until Eternity  
10 end  
11 assert(  $x \neq 511$  )
```

Values of  $x$ :

0, 1, 2, 5, 10, 21, ...

Encode as strings:

$\lambda, 1, 10, 101, 1010, 10101, \dots$



# Extrapolation in Action

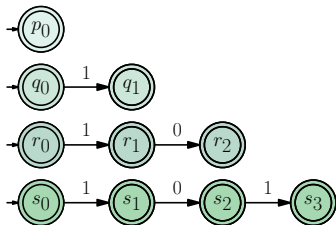
```
1 begin  
2    $x \leftarrow 0$   
3   repeat  
4     if  $x$  is odd then  
5        $x \leftarrow 2x$   
6     else  
7        $x \leftarrow 2x + 1$   
8     end  
9   until Eternity  
10 end  
11 assert(  $x \neq 511$  )
```

Values of  $x$ :

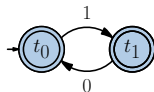
0, 1, 2, 5, 10, 21, ...

Encode as strings:

$\lambda, 1, 10, 101, 1010, 10101, \dots$



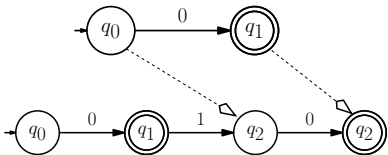
Voodoo



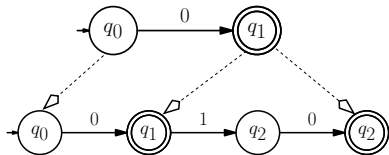
# Framework I: Identify Pattern

Relational templates identify states that are part of the same pattern.

$k$ -Suff- $\equiv$ : states with same suffixes of length at most  $k$



$k$ -Pre- $\cap$ : states with a common prefix of length at most  $k$

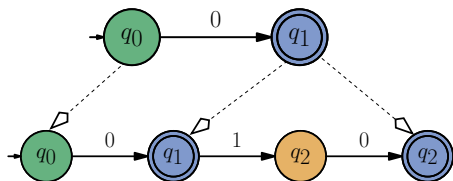


Relational Template:  $T$

A function such that  $T(\mathcal{A}, \mathcal{B}) \subseteq Q_A \times Q_B$

## Framework II: Collate Similarities

Identify states in one automaton that are part of the same pattern.



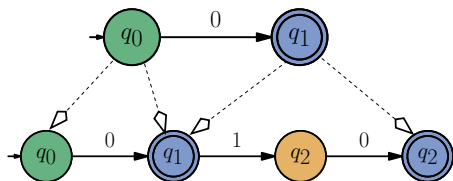
Equivalence Template:  $E_T(\mathcal{A}, \mathcal{B})$

Maps a relation over  $Q_A \times Q_B$  to an equivalence relation over  $Q_B$ .

$$E_T(\mathcal{A}, \mathcal{B}) \leftarrow [T^{-1}(\mathcal{A}, \mathcal{B}) \circ T(\mathcal{A}, \mathcal{A})]^* \cup id(Q_B)$$

## Framework II: Collate Similarities

Identify states in one automaton that are part of the same pattern.



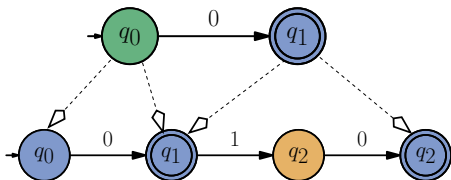
Equivalence Template:  $E_T(\mathcal{A}, \mathcal{B})$

Maps a relation over  $Q_A \times Q_B$  to an equivalence relation over  $Q_B$ .

$$E_T(\mathcal{A}, \mathcal{B}) \leftarrow [T^{-1}(\mathcal{A}, \mathcal{B}) \circ T(\mathcal{A}, \mathcal{A})]^* \cup id(Q_B)$$

## Framework II: Collate Similarities

Identify states in one automaton that are part of the same pattern.



Equivalence Template:  $E_T(\mathcal{A}, \mathcal{B})$

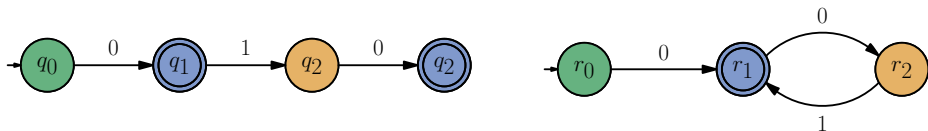
Maps a relation over  $Q_A \times Q_B$  to an equivalence relation over  $Q_B$ .

$$E_T(\mathcal{A}, \mathcal{B}) \leftarrow [T^{-1}(\mathcal{A}, \mathcal{B}) \circ T(\mathcal{A}, \mathcal{A})]^* \cup id(Q_B)$$

## Framework III: Generalise Pattern

Transform the automaton to generalise the observation.

[Huffman, '54] Quotient for minimization:  $Suff- \equiv$



Extrapolation Operator:  $\nabla_E(\mathcal{A}, \mathcal{B})$

The quotient of  $\mathcal{B}$  with respect to  $E(\mathcal{A}, \mathcal{B})$ .

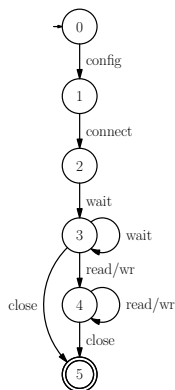
$$\nabla_E(\mathcal{A}, \mathcal{B}) \leftarrow \mathcal{B}/E(\mathcal{A}, \mathcal{B})$$

# Specification/Error Mining.

Can we infer API specifications/error patterns from use cases?

```
class SocketChannelClient {
void example() {
Collection<SocketChannel> channels
    = createChannels();
for (SocketChannel sc : channels) {
sc.connect(new InetSocketAddress(
    "tinyurl.com/23qct8", 80));
while (!sc.finishConnect()) {
// ... wait for connection ... }
if (?) {
receive(sc);
} else {
send(sc); } }
closeAll(channels); }
void closeAll(Collection<SocketChannel> chnls) {
for (SocketChannel sc : chnls) { sc.close(); }}

Collection<SocketChannel> createChannels() {
List<SocketChannel> list = new LinkedList<SocketChannel>();
list.add(createChannel("http://tinyurl.com/23qct8", 80));
list.add(createChannel("http://tinyurl.com/23qct8", 80));
return list;} .....
```



[Shoham, Yahav, Fink, Pistoia; ISSTA '07]  $k$ -Pre- $\cap$  and  $k$ -Suff- $\cap$ .  
[Cook, Wolf; TSE '98] Various language inference techniques.

# Inductive Inference from Positive Data.

*"I wish to construct a precise model for the intuitive notion 'able to speak a language' in order to be able to investigate theoretically how it can be achieved artificially"*

— E. Mark Gold, Language Identification in the Limit, 1967

[Angluin; JACM, '82]

Algorithm ZR

Input. a nonempty positive sample  $S$ .

Output. a zero-reversible acceptor  $A$ .

• Initialization

Let  $A_0 = (Q_0, I_0, F_0, \delta_0)$  be  $PT(S)$

Let  $\pi_0$  be the trivial partition of  $Q_0$

For each  $b \in U$  and  $q \in Q_0$  let  $s(\{q\}, b) = \delta_0(q, b)$  and  $p(\{q\}, b) = \delta_0(q, b)$

Choose some  $q' \in F_0$ .

Let LIST contain all pairs  $(q', q)$  such that  $q \in F_0 - \{q'\}$ .

Let  $i = 0$ .

• Merging

While LIST  $\neq \emptyset$  do

begin

Remove some element  $(q_1, q_2)$  from LIST

Let  $B_1 = B(q_1, \pi_i)$ ,  $B_2 = B(q_2, \pi_i)$

If  $B_1 \neq B_2$  then

begin

Let  $\pi_{i+1}$  be  $\pi_i$  with  $B_1$  and  $B_2$  merged

For each  $b \in U$ ,  $s$ -UPDATE( $B_1, B_2, b$ ) and  $p$ -UPDATE( $B_1, B_2, b$ )

Increase  $i$  by 1

end

end

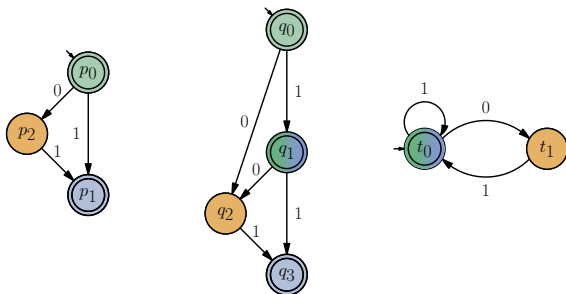
• Termination

Let  $f = i$  and output the acceptor  $A_0/\pi_f$

- Examples:  $e_1, e_2, e_3, \dots$
- Apply inference algo till hypothesis stabilizes.
- Reversible languages:  $Suff-\cap$
- Characterisation straightforward.
- Can describe over 10 inference algorithms.

# Verification is at the party too.

- [Bartzis, Bultan; CAV '04] Widening:  $Pre-\cap$  or  $Suff-\equiv$



- Identified and fixed unsoundness in the paper.
- On suffix-trees is inference algorithm for reversible languages!

# YACC for Automata Extrapolation

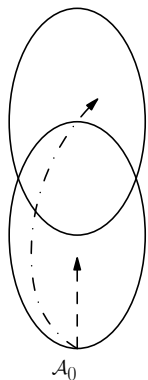
- $T ::= T \mid T_1 \cup T_2 \mid T_1 \cap T_2 \mid T_1 \setminus T_2 \mid \overline{T}$
- $E ::= E_T \mid E_1 \sqcap E_2 \mid E_1 \sqcup E_2$
- $\nabla_E = \mathcal{B}/E(\vec{\mathcal{A}}, \mathcal{B})$

Given a set  $\mathcal{T}$  of templates, can generate a family of operators by closure under compositions. Duality via *conjunctive* quotient.

## Theorem (Soundness)

For any automaton  $\mathcal{B}$  and vector of automata  $\vec{\mathcal{A}}$  and any extrapolation operator  $L(\mathcal{B}) \subseteq L(\nabla_E(\vec{\mathcal{A}}, \mathcal{B}))$ .

# Analysis with Extrapolation



Given  $\mathcal{A}_0$ , and monotone  $F : \mathbb{A} \rightarrow \mathbb{A}$ , generate

$$\mathcal{A}_0, \mathcal{A}_1 = \mathcal{A}_0 \cup F(\mathcal{A}_0), \mathcal{A}_2 = \mathcal{A}_1 \cup F(\mathcal{A}_1), \dots$$

Given  $\mathcal{A}_0$ , monotone  $F : \mathbb{A} \rightarrow \mathbb{A}$ ,  $\nabla_0, \nabla_1, \dots$  and  $\vec{\mathcal{B}}_0, \vec{\mathcal{B}}_1, \dots$  generate

$$\begin{aligned} \hat{\mathcal{A}}_0 &= \mathcal{A}_0 \\ \hat{\mathcal{A}}_{i+1} &= \begin{cases} \hat{\mathcal{A}}_i & \text{if } L(F(\hat{\mathcal{A}}_i)) \subseteq L(\hat{\mathcal{A}}_i) \\ \nabla_i(\vec{\mathcal{B}}_i, \hat{\mathcal{A}}_i \cup F(\hat{\mathcal{A}}_i)) & \text{otherwise.} \end{cases} \end{aligned}$$

## Theorem (Limit Soundness)

For  $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \dots$  and  $\hat{\mathcal{A}}_0, \hat{\mathcal{A}}_1, \hat{\mathcal{A}}_2, \dots$ :  $\bigcup_{i \geq 0} L(\mathcal{A}_i) \subseteq \bigcup_{i \geq 0} L(\hat{\mathcal{A}}_i)$

# Termination

$ind(\mathcal{A})$ : index of the Myhill-Nerode equivalence for  $L(\mathcal{A})$

$\|\mathcal{A}\|$ : number of states in  $\mathcal{A}$ .

## Lemma (Stablization)

An extrapolated sequence stabilizes iff for some  $n$  and infinitely many  $\hat{\mathcal{A}}_i$ ,  $ind(\hat{\mathcal{A}}_i) \leq n$ .



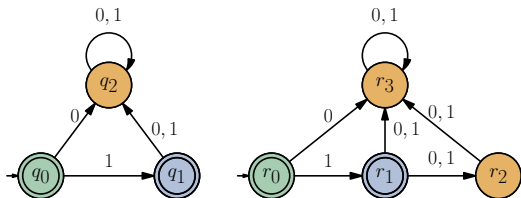
## Corollary (Conditional Stabilization)

For a predicate  $\mathcal{P}$ , if  $\mathcal{P}(\vec{\mathcal{B}}) \Rightarrow \|\nabla(\vec{\mathcal{B}}, \mathcal{A})\| \leq n$ , applying  $\nabla$  infinitely often with such  $\vec{\mathcal{B}}$  yields stabiliation.

# Termination: Examples

$Pre\text{-}\cap$ : Relate states with a common prefix.

- $\mathcal{P}(\mathcal{B})$  is true if  $\mathcal{B}$  is complete and  $\|\mathcal{B}\| \leq n$
- $\mathcal{P}(\mathcal{B}) \Rightarrow \|\nabla_T(\mathcal{A}, \mathcal{B})\| \leq n$
- Judicious choice of first parameter may work well.



What about composition of operators?

# Termination under Composition

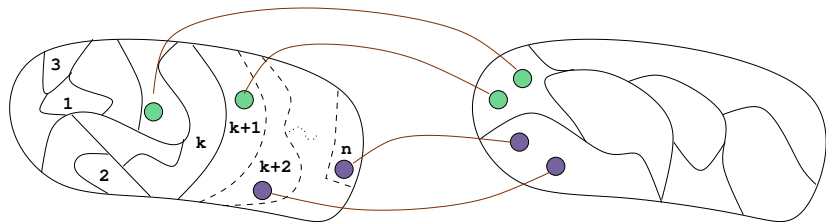
## Theorem (Bounded Equivalence Templates)

- $\|(E_1 \sqcap E_2)(\vec{B}_1, \vec{B}_2, \mathcal{A})\| \leq k$  iff there exist  $k_1, k_2$  such that  $k \leq k_1 \cdot k_2$ ,  $\|E_1(\vec{B}_1, \mathcal{A})\| \leq k_1$ , and  $\|E_2(\vec{B}_2, \mathcal{A})\| \leq k_2$ .

# Termination under Composition

## Theorem (Bounded Equivalence Templates)

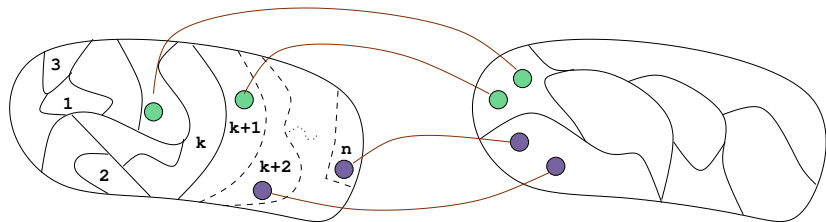
- $\|(E_1 \sqcap E_2)(\vec{B}_1, \vec{B}_2, \mathcal{A})\| \leq k$  iff there exist  $k_1, k_2$  such that  $k \leq k_1 \cdot k_2$ ,  $\|E_1(\vec{B}_1, \mathcal{A})\| \leq k_1$ , and  $\|E_2(\vec{B}_2, \mathcal{A})\| \leq k_2$ .



# Termination under Composition

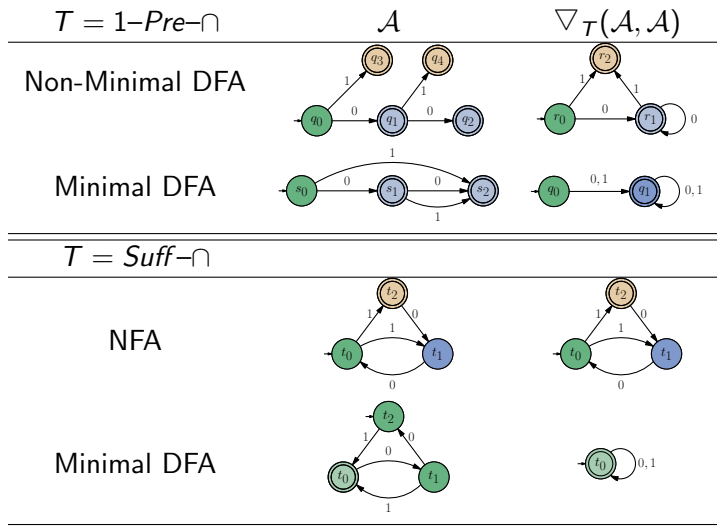
## Theorem (Bounded Equivalence Templates)

- $\|(E_1 \sqcap E_2)(\vec{\mathcal{B}}_1, \vec{\mathcal{B}}_2, \mathcal{A})\| \leq k$  iff there exist  $k_1, k_2$  such that  $k \leq k_1 \cdot k_2$ ,  $\|E_1(\vec{\mathcal{B}}_1, \mathcal{A})\| \leq k_1$ , and  $\|E_2(\vec{\mathcal{B}}_2, \mathcal{A})\| \leq k_2$ .
- $\|(E_1 \sqcup E_2)(\vec{\mathcal{B}}_1, \vec{\mathcal{B}}_2, \mathcal{A})\| \leq k$  iff for some  $n \geq \|(E_1(\vec{\mathcal{B}}_1, \mathcal{A})\| - k$  there exist states  $q_1, \dots, q_n, q'_1, \dots, q'_n \in Q_{\mathcal{A}}$  such that
  - ▶ for all  $1 \leq i < j \leq n$ ,  $(q_i, q_j) \notin E_1(\vec{\mathcal{B}}_1, \mathcal{A})$ , and
  - ▶ for all  $1 \leq i \leq n$ ,  $(q_i, q'_i) \in E_2(\vec{\mathcal{B}}_2, \mathcal{A})$ , and
  - ▶ for all  $1 \leq i \leq j \leq n$ ,  $(q_i, q'_j) \notin E_1(\vec{\mathcal{B}}_1, \mathcal{A})$ .



# Does Representation Matter?

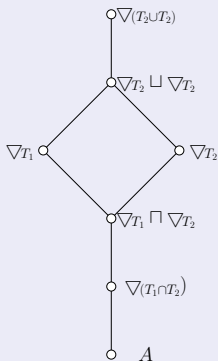
Verification Folk Wisdom: Stick to the minimal DFA.



# Refinement of Operators

$\nabla_1$  refines  $\nabla_2$  if for all  $\vec{B}$ ,  $\mathcal{A}$ ,  $L(\nabla_1(\vec{B}, \mathcal{A})) \subseteq L(\nabla_2(\vec{B}, \mathcal{A}))$ .

Lemma (Refinement under composition)



# To sum it all up.

Abstraction: cope with computational infeasibility.

- Concepts: Soundness, duality, refinement, completeness.
- State of the art: Mature theory. Omnipresent. Infinite abstractions problematic.

Symbolic Representation: cope with combinatorial explosion.

- Ideas: Canonicity, encoding, efficient algorithms.
- State of the art: Various stable tools and libraries.

Extrapolation: cope with absence of induction hypothesis.

- Concepts: Soundness, duality, termination, representation independence, refinement.
- State of the art?



# One for the Road

*“Suppose that you want to teach the ‘cat’ concept to a young child. Do you explain that a cat is a relatively small, primarily carnivorous mammal with retractible claws, a distinctive sonic output, etc.? No. You probably show the kid a lot of different cats, saying ‘kitty’ each time, until it gets the idea. To put it more generally, generalizations are best made by abstraction from experience.”*

— R. P. Boas, American Mathematical Monthly

*“Widening is considered as a kind of dirty heuristic”*

— Halbwachs, VMCAI '06, Invited tutorial

This need not be the case. Extrapolation operators are scarce because their effects are far from understood. We have made a step towards changing that.

Extrapolation can be given a rigorous yet accessible theoretical foundation.

The results we discover on the way may dispel the mysticism surrounding extrapolation. Other results may be of independent interest (learning algorithms, schema inference for XML, etc.).

There is much research to do, beauty to discover and systems to build.