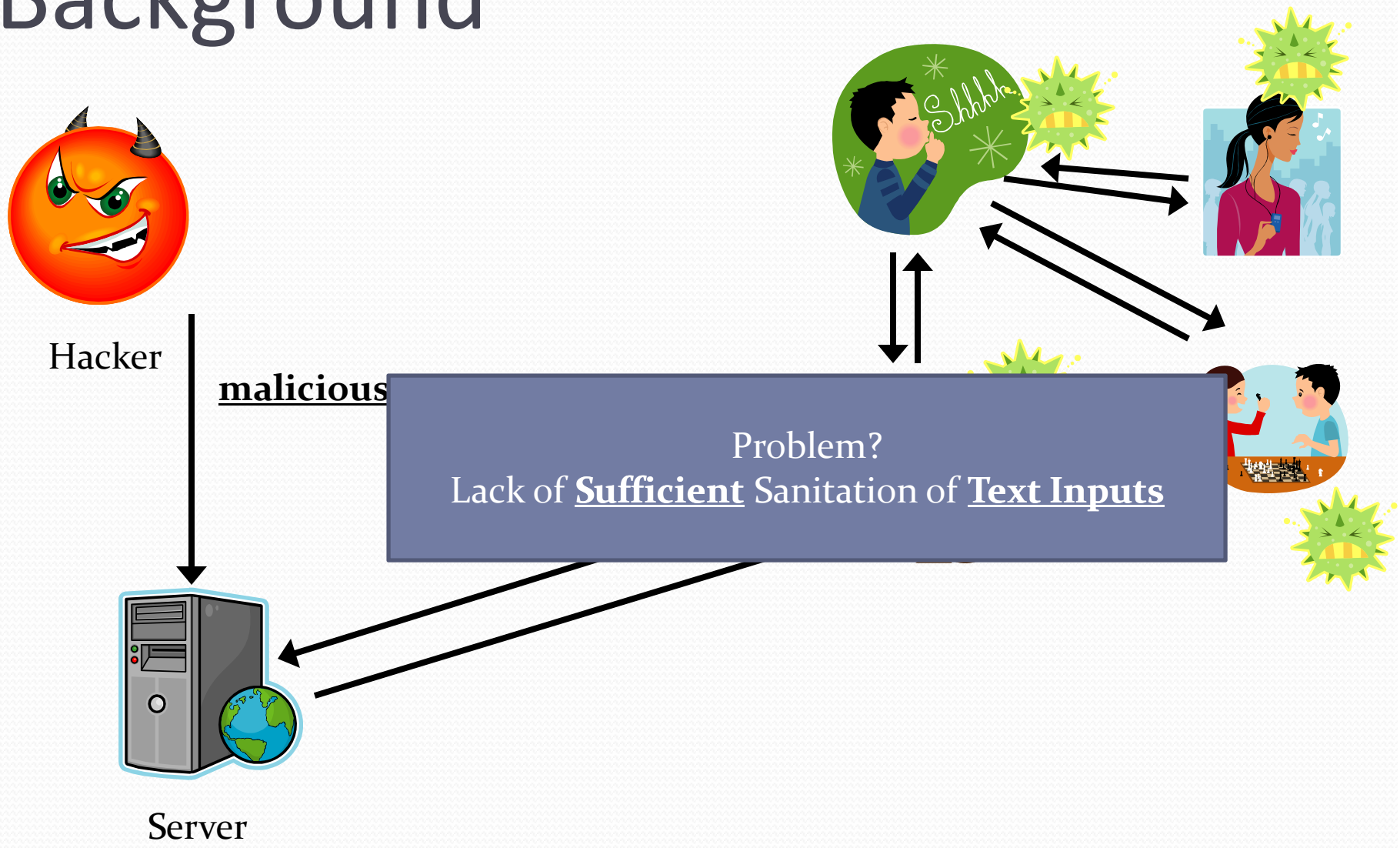
Modeling Regular Replacement
for String Constraints Solving

Xiang Fu
Hofstra University

Chung-Chih Li
Illinois State University

Background

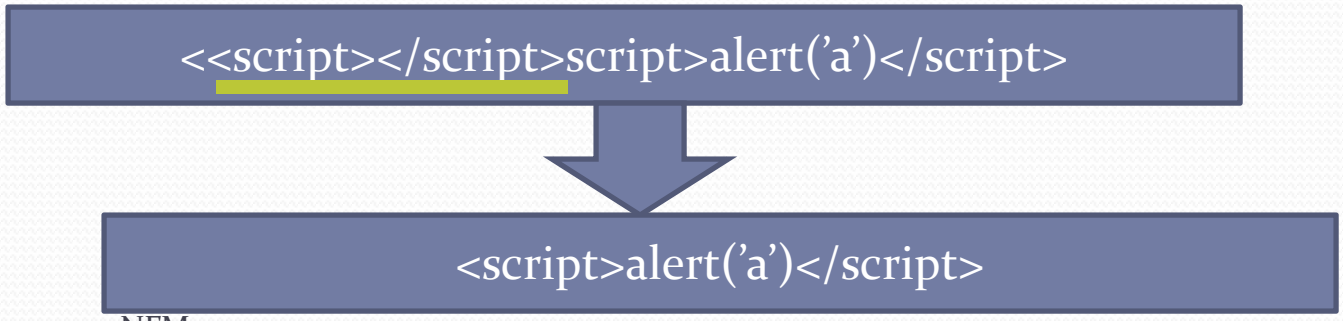

Shhhh...
Hacker
malicious
Problem?
Lack of Sufficient Sanitation of Text Inputs
Server

One Typical Error

```
1  <?php
2     $msg = $_POST["msg"];
3     $sanitized = pregreplace(
4       "/\< s c r i p t .*?\>.*?\<\/ s c r i p t .*?\ >/ i ",
5       " " ,
6       $msg ) ;
7     savetodb($sanitized )
8  ?>
```

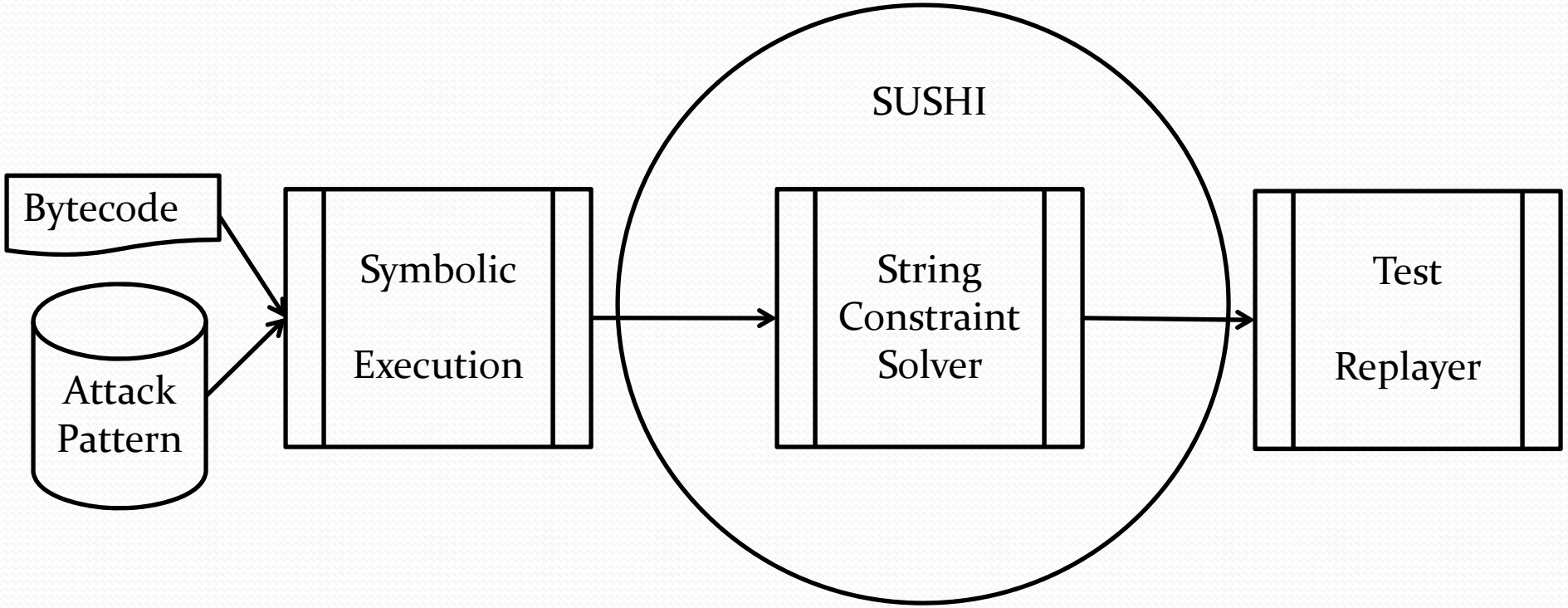
Reluctant Kleene Star

Attacker's Input



Bigger Picture

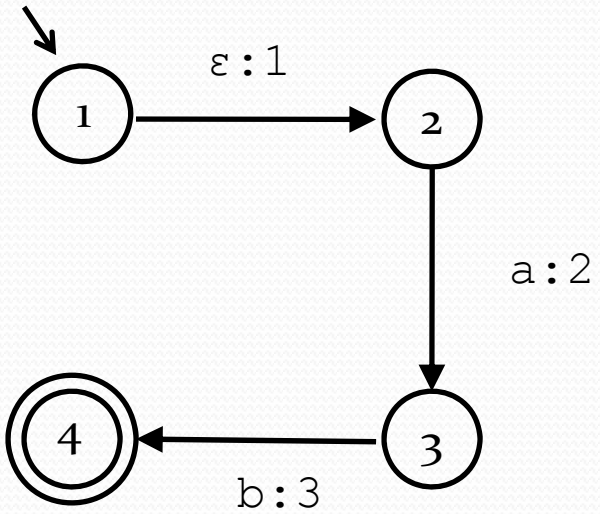
- Objective: Automatic Discovery of Vulnerabilities



Our Contribution

- Atomic **Replacement** Constraints
- Consider Two Semantics
  - Greedy
  - Reluctant
- Modeling Using Finite State Transducer (FST)
- Compact Representation of FST
- Security Analysis

Finite State Transducer



A

(ab,123) ∈ L(A)

- Accepts Regular Relation
- Union, Concat, Composition
- Intersection, Complement

- Used for Modeling Rewriting Rules [Kaplan94, Karttunen96]

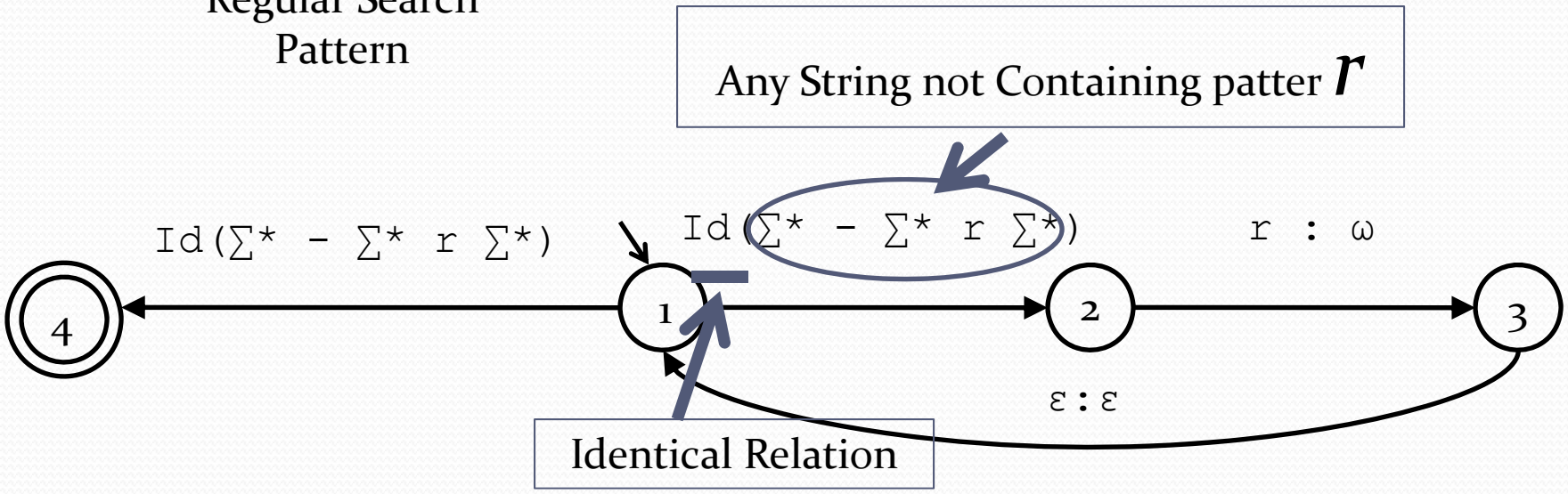
Hierarchical FST & Modeling Declarative Semantics

Goal: $S_{r \rightarrow \omega}$

$\text{aaa}_{a^+ \rightarrow b} \Rightarrow \{b, bb, bbb\}$

Regular Search Pattern

Replacement



Modeling Reluctant Semantics

Goal: $S^{-}_{r \to \omega}$ $\quad aaa^{-}_{a^{+} \to b}$ ⇒ {bbb}

```
Key: Left-Most Matching
```

- 2 Steps
  - Mark the beginning of pattern
  - Do the replacement

Input Word

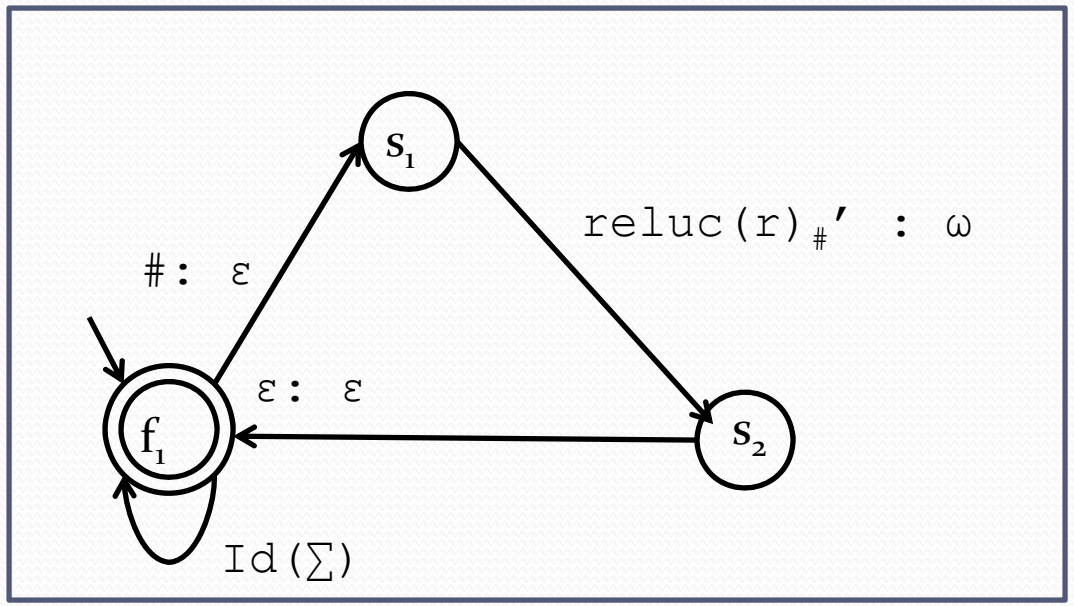
a a b b c d a b c a b d

Search Pattern

$a^+b^+c \rightarrow x$

Begin Marker

a # a b b c d # a b c a b d



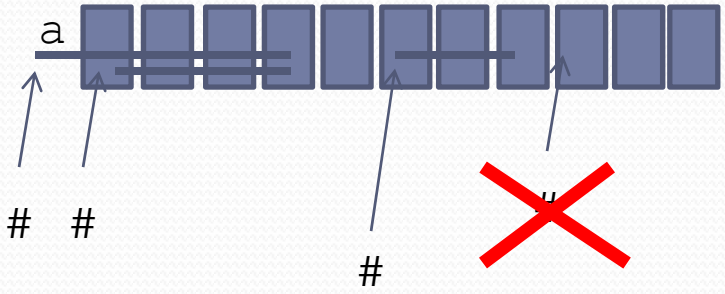
x d x a b d

The Challenge: Begin Marker

Search Pattern

$a^+b^+c \rightarrow x$

Input Word

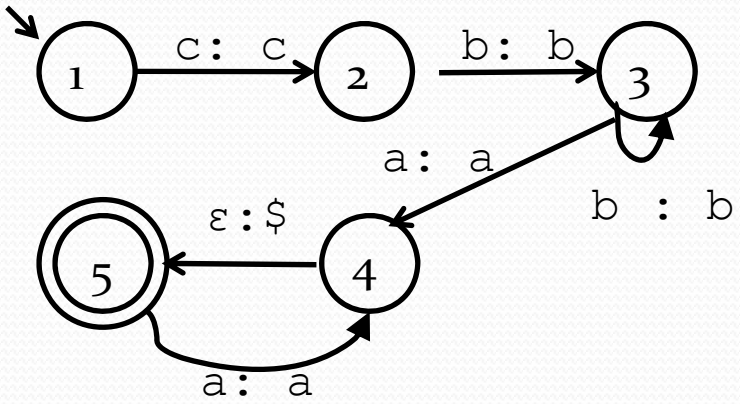


Look-ahead Capability?

3 Steps:
(1) End marker
(2) Generic end marker
(3) Begin marker

Non-determinism

Preliminary End Marker



A_1

Search Pattern

$a^+b^+c \rightarrow x$

Idea: Start with **End Marker** for **Reverse** of Search Pattern

Reversed Pattern

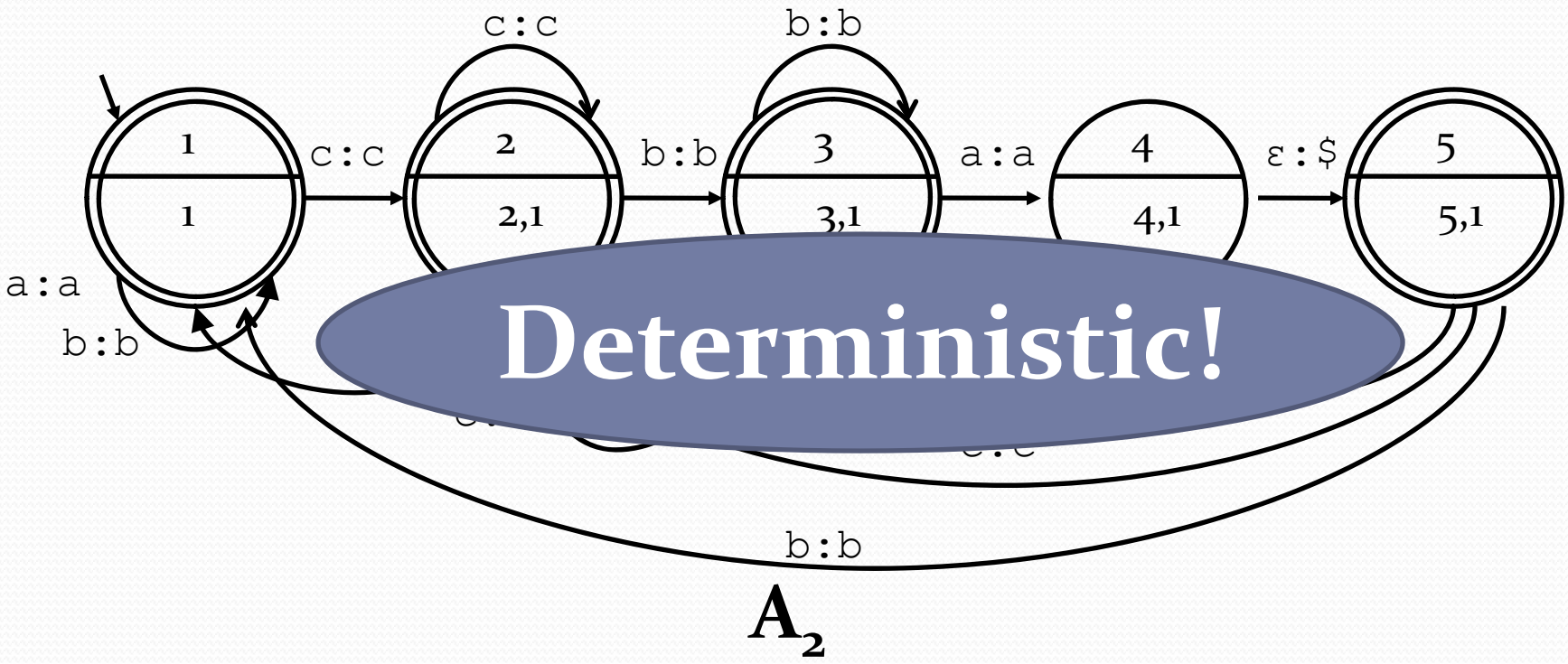
cb^+a^+

Problem:
Input tape accepts cb^+a^+ only!

Generic End Marker

Pattern

cb^+a^+



A_2

c c b a a
Input Word

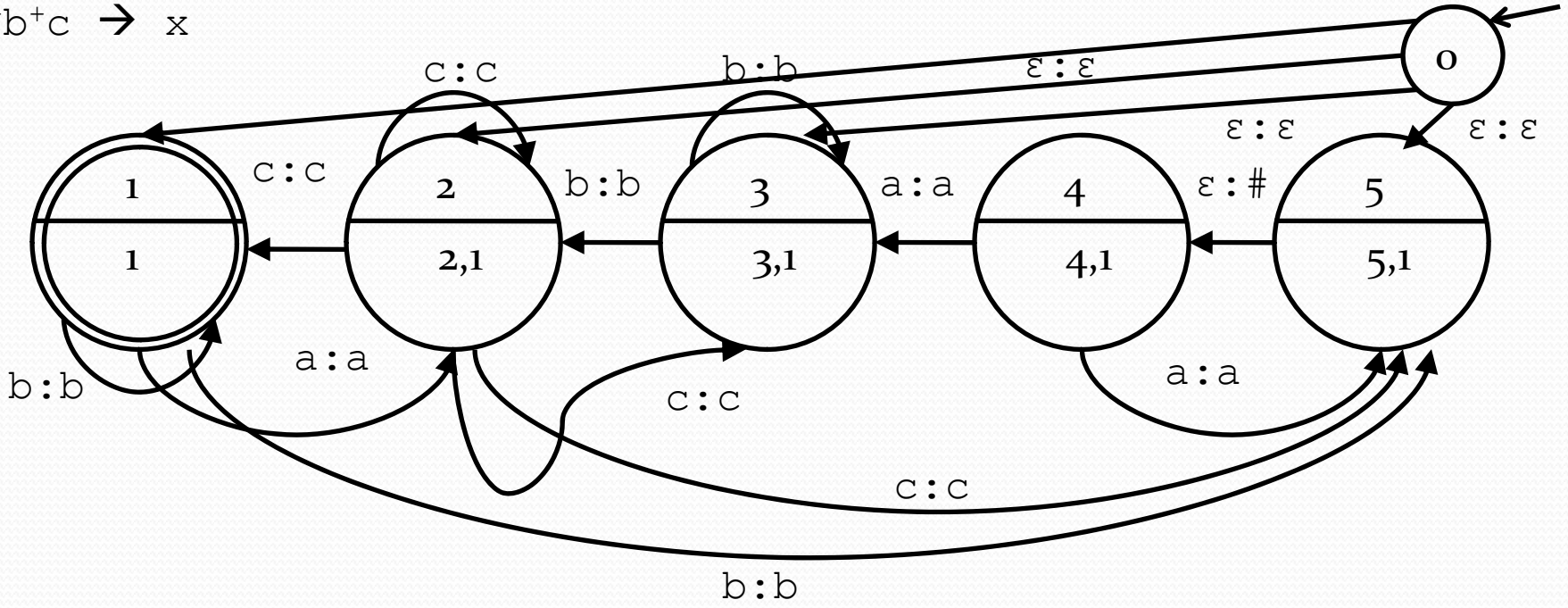
→

c c b a **$** a **$**
Output Word

Finally, the Begin Marker

Search Pattern

$a^+b^+c \rightarrow x$



A_3

Input Word

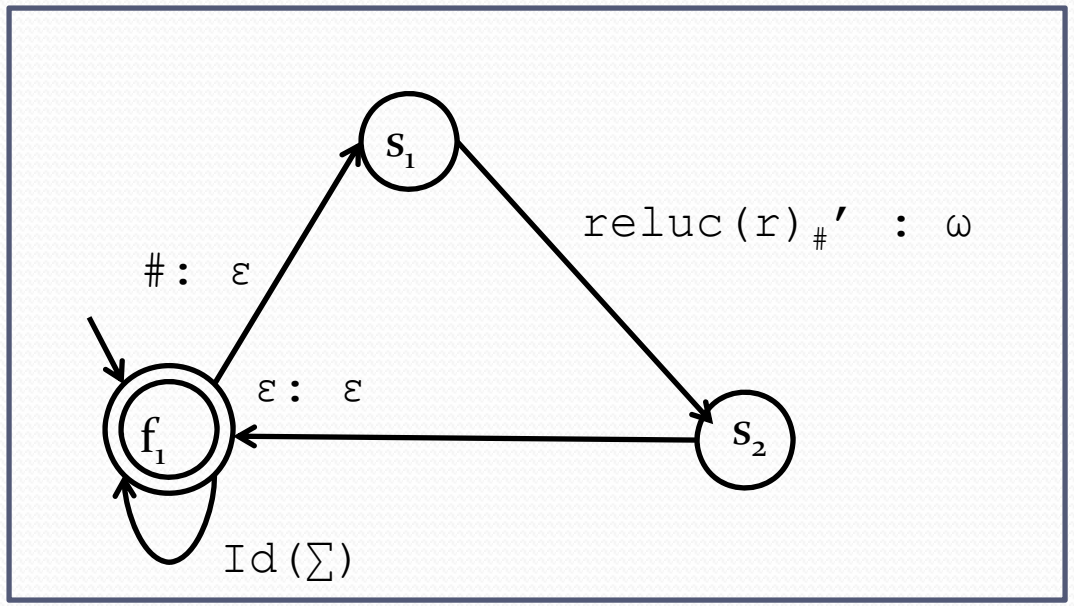
a a b b c d a b c a b d

Search Pattern

$a^+b^+c \rightarrow x$

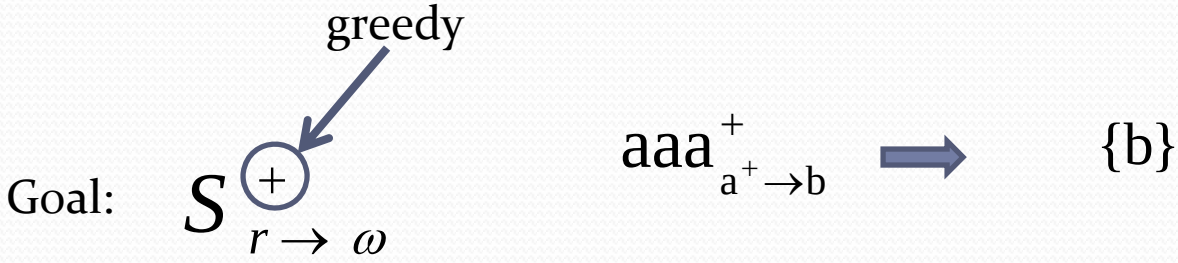
Begin Marker

a # a b b c d # a b c a b d



x d x a b d

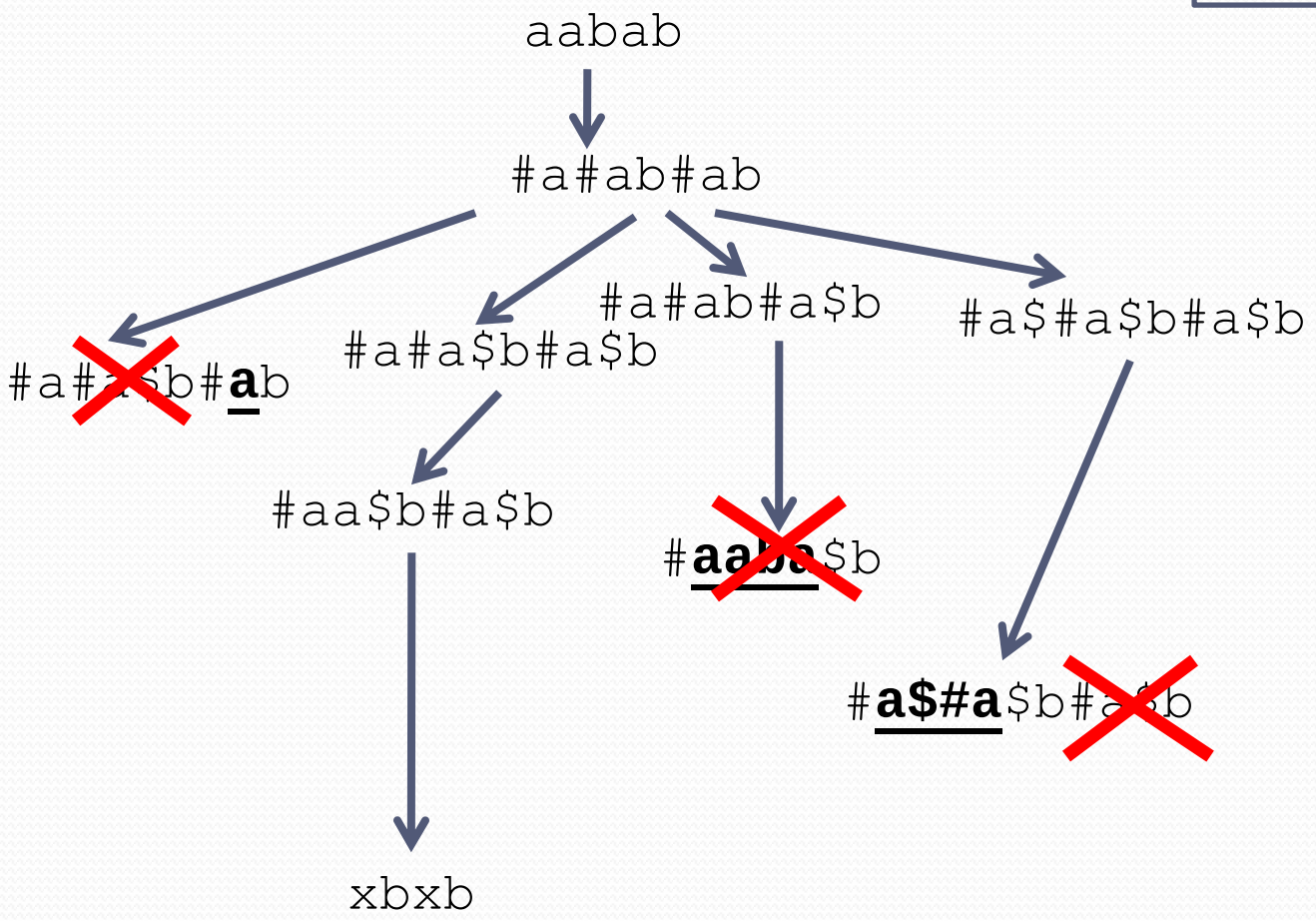
Greedy Semantics



Challenge:
Look-ahead ***longest*** match

Search Pattern

$a^+ \rightarrow x$



Step 1: Begin Marker

Step 2: ND End Marker

Step 3: Pairing Markers

Step 4: Checking Match

Step 5: Check Longest

Step 6: Replacement

Applications

- Solve String Constraints

```
1  protected void processRequest(
2   HttpServletRequest request ...)
3   throws ServletException{
4   PrintWriter out = response.getWriter();
5   try {
6     String sUname = request.getParameter("sUname");
7     String sPwd = request.getParameter("sPwd");
8     Connection con = DriverManager.getConnection("...");
9     Statement stmt = con.createStatement();
10    String strCmd= "SELECT * FROM users\nWHERE uname='"
11      + massage(sUname) + "' AND pwd='"
12      + massage(sPwd) + "'";
13    ResultSet srs = stmt.executeQuery(strCmd);
14    if(srs.next()){
15     out.println("Welcome " + sUname);
16    }else{
17     out.println("Login fail!");
18    }
19   }catch(Exception exc){...}
20 }
21
22 protected String massage(String str){
23  String strOut = str.replaceAll("'", "''");
24  if(strOut.length()>16) return strOut.substring(0,16);
25  return strOut;
26 }
```

Input: user name
After filtering **single quote** and **length restriction**

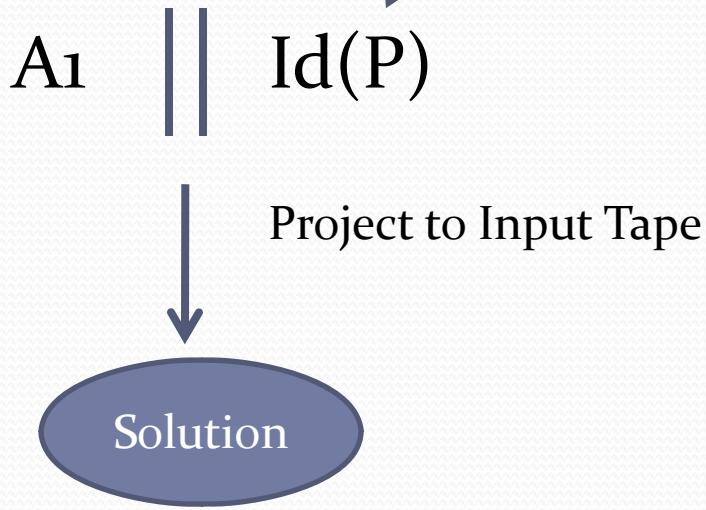
$$'SELECT...WHERE\ ...' + x^{+}_{\rightarrow''}[0,16] + ''\ AND\ pwd = '' + y^{+}_{\rightarrow''}[0,16] + '''$$

$$\equiv$$

$$uname = '([\hat{}'] \mid '').*'\ OR\ .*\ uname <> ''$$

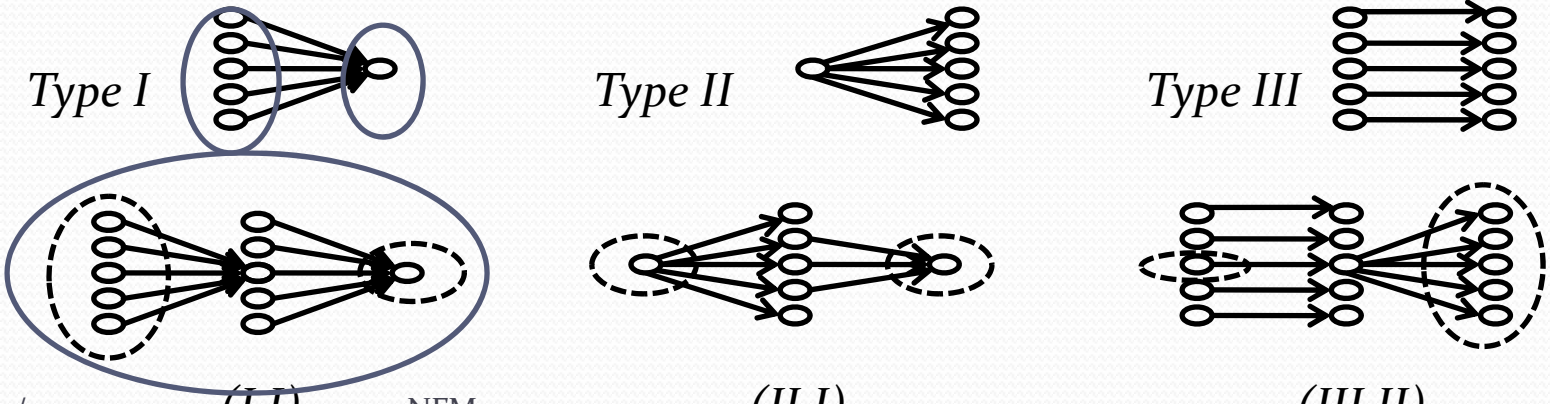
Solving Atomic Constraint

Goal: $S_{r \to \omega} \equiv P$



SUSHI Constraint Solver

- Solves Simple Linear String Constraints (SISE)
- Relies on
  - `dk.brics.automaton` for FSA operations
  - Self-made Java package for FST operations
- Supports 16-bit Unicode
- Compact Transition Representation



Efficiency of Solver

Login Servlet

```
1 protected void processRequest(
2  HttpServletRequest request ...)
3  throws ServletException{
4  PrintWriter out = response.getWriter();
16   }else{
17    out.println("Login fail!");
18   }
19  }catch(Exception exc){...}
20 }
21
22 protected String massage(String str){
23  String strOut = str.replaceAll("'", "''");
24  if(strOut.length()>16) return strOut.substring(0,16);
```

1.4 Seconds on 2Ghz PC

Benchmark Equations

1. $x^{+}_{a^{+}\to b\{n,n\}} = b\{2n,2n\}$
2. $x^{-}_{a^{+}\to b\{n,n\}} = b\{2n,2n\}$
3. $x^{+}_{a^{*}\to b\{n,n\}} = b\{$
4. $x^{-}_{a^{*}\to b\{n,n\}} = b\{$

Equation Size: 565
74 Seconds
Shorter than Security Track #1022748

	/script>.*
EffectiveScript	.*[a-zA-Z0-9_]+ *= *"[^"]*<script.*>.*
MatchTag	.*<embed[^<>]*>.* ∩ .*</embed[^<>]*>.*

Equation	FST States	FST Tra			
eq1(41)	5751	16(			
eq2(41)	5416	57			
eq3(41)	631	1565	2	2	492.281
eq4(41)	126	177	0	0	14.016

Related Work

- Forward String Analysis
  - Christensen & Møller [SAS'03]
  - Wasserman & Su [PLDI'07, ICSE'08]
  - Bjørner & Tillmann [TACAS'09]
- Backward String Analysis
  - Kiezun & Ganesh [ISSTA'09]
  - Yu & Bultan [SPIN'08, ASE'09]
  - Fu [COMPSAC'07, TAVWEB'08]
- Natural Language Processing
  - * Kaplan and Kay [CL'1994]

Our Contribution:

Precise Modeling of Various Regular Substitution Semantics

Limitations

- SISE String Constraints
  - All Variables Appear on LHS (Once)
  - No Easy Solution for Equation System Yet
  - No string length
- Future Directions
  - Encoding string length in automata
  - Finite model on bit-vector

Questions?