A Decision Procedure for Subset Constraints over Regular Languages

Pieter Hooimeijer
Westley Weimer

University of Virginia

Motivation

"String values have lost their innocence..." [Thiemann05]

An Example Program

```
// v1 and v2 are user inputs

if (!ereg('o(pp)+', v1)){exit;}
if (!ereg('p*q', v2)){exit;}

v3 = v1 . v2; // concat
if (v3 != 'oppppq'){exit;}
```

An Example Program

```
// v1 and v2 are user inputs

1 if (!ereg('o(pp)+', v1)){exit;}
2 if (!ereg('p*q', v2)){exit;}

v3 = v1 . v2; // concat
3 if (v3 != 'oppppq'){exit;}
```

An Example Program

$c_1 = L(\texttt{o(pp)+})$ ① $v_1 \subseteq c_1$

$c_2 = L(\texttt{p*q})$ ② $v_2 \subseteq c_2$

$c_3 = L(\texttt{opppq})$ ③ $v_1 \circ v_2 \subseteq c_3$

Some Applications

- Automated randomized testing
- Abstraction refinement
- Fault localization
- Optimization & refactoring

Some Questions

- Is the theory expressive enough?
- Does it apply to interesting problems?
- Is the decision procedure fast enough?

Outline

- Problem definition
- Algorithm
  - example execution
  - correctness
- Experiment

Subset Constraints

$$
\begin{array}{lcl}
S & ::= & E \subseteq C \\
E & ::= & E \circ E \\
 & | & C \\
 & | & V
\end{array}
$$

constants

variables

Subset Constraints

$$
\begin{array}{lll}
S & ::= & E \subseteq C \\
E & ::= & E \circ E \\
 & | & C \\
 & | & V
\end{array}
$$

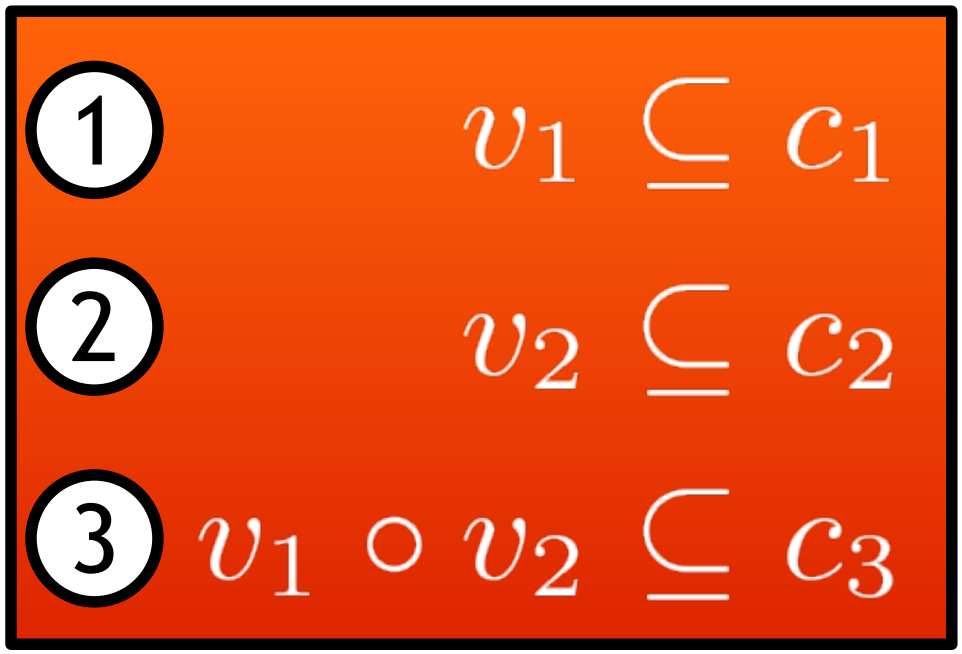
concatenation

Regular Matching Assignments

Definition: Let $P = \{s_1, \ldots, s_p\}$ be a system of subset constraints over a set of shared variables V. Find all assignments that simultaneously satisfy each constraint $s_1, \ldots, s_p$

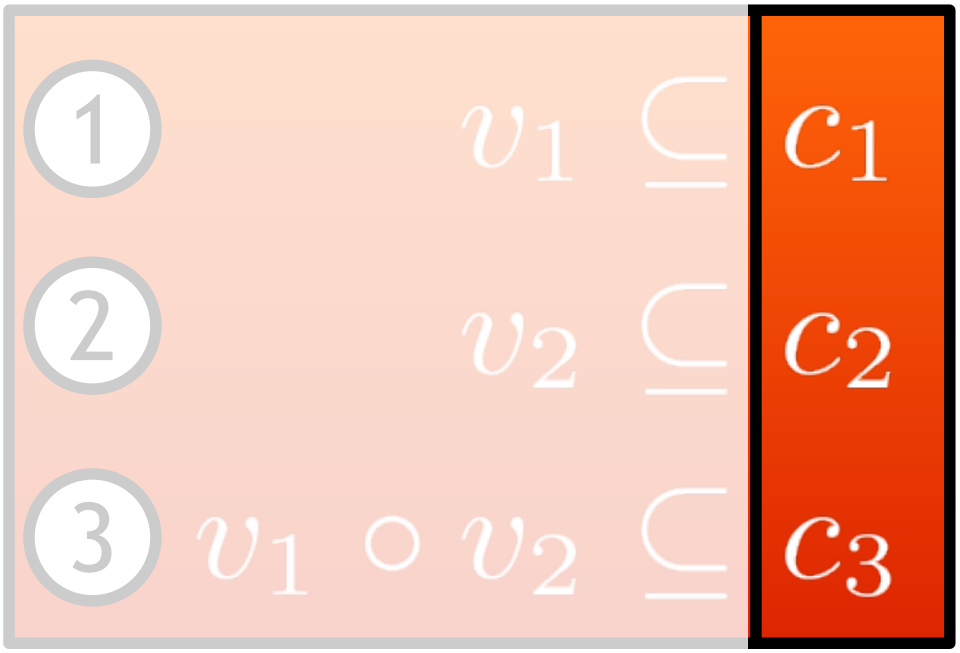
Algorithm

Input



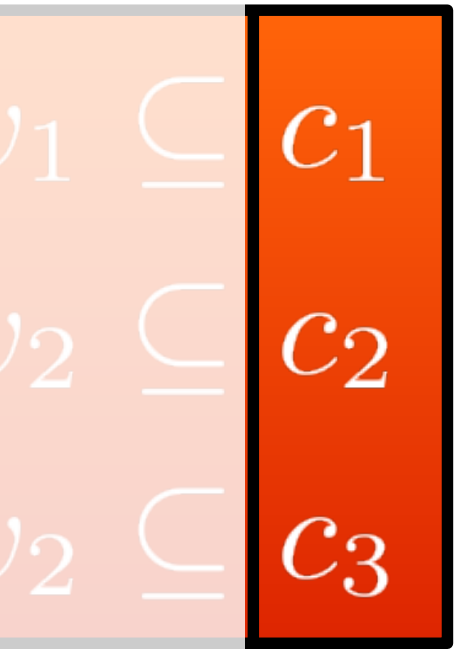
Algorithm

Input

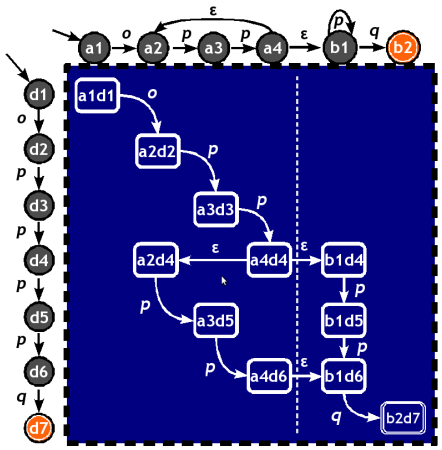


Algorithm

Input



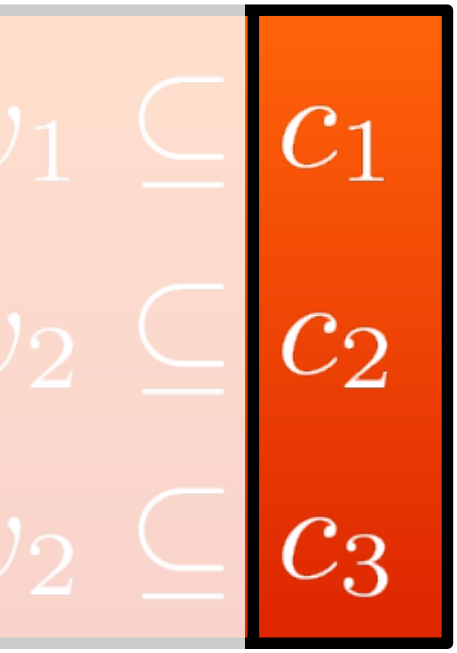
Cross Product



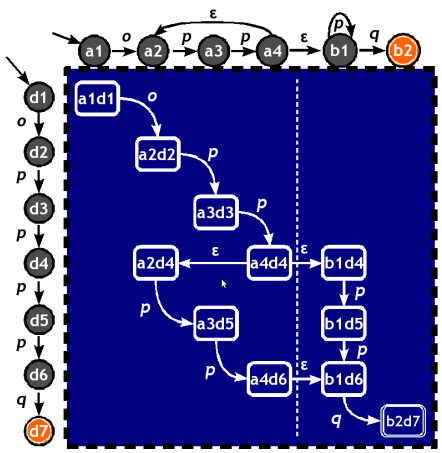
$(c_1 \circ c_2) \cap c_3$

Algorithm

Input



Cross Product



$(c_1 \circ c_2) \cap c_3$

✓ Sat.

✗ Unsat.

An Example

Solution I:

v_1 = { opp }

v_2 = { ppq }

Solution II:

v_1 = { opppp }

v_2 = { q }

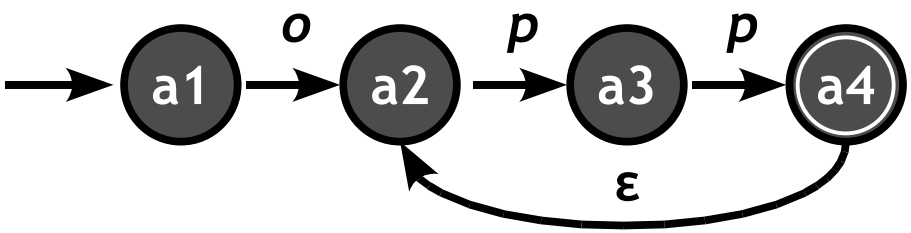
$$c_1 = L(\mathtt{o(pp)+}) \qquad v_1 \subseteq c_1$$

$$c_2 = L(\mathtt{p*q}) \qquad v_2 \subseteq c_2$$

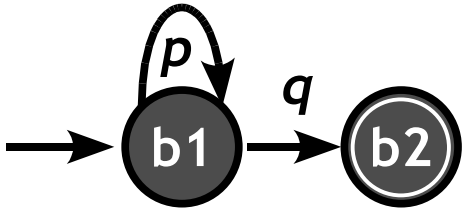
$$c_3 = L(\mathtt{oppppq}) \qquad v_1 \circ v_2 \subseteq c_3$$

An Example

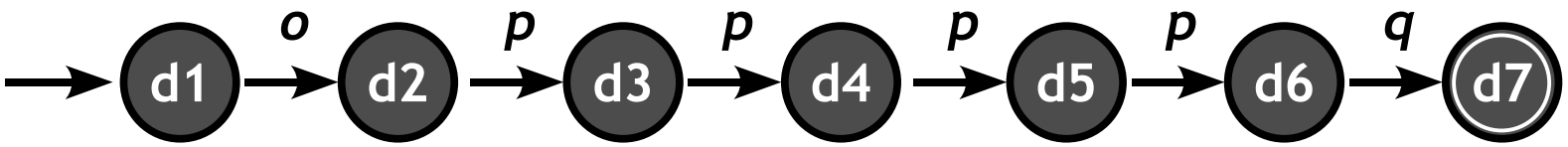
$c_1 = L(o(pp)+)$



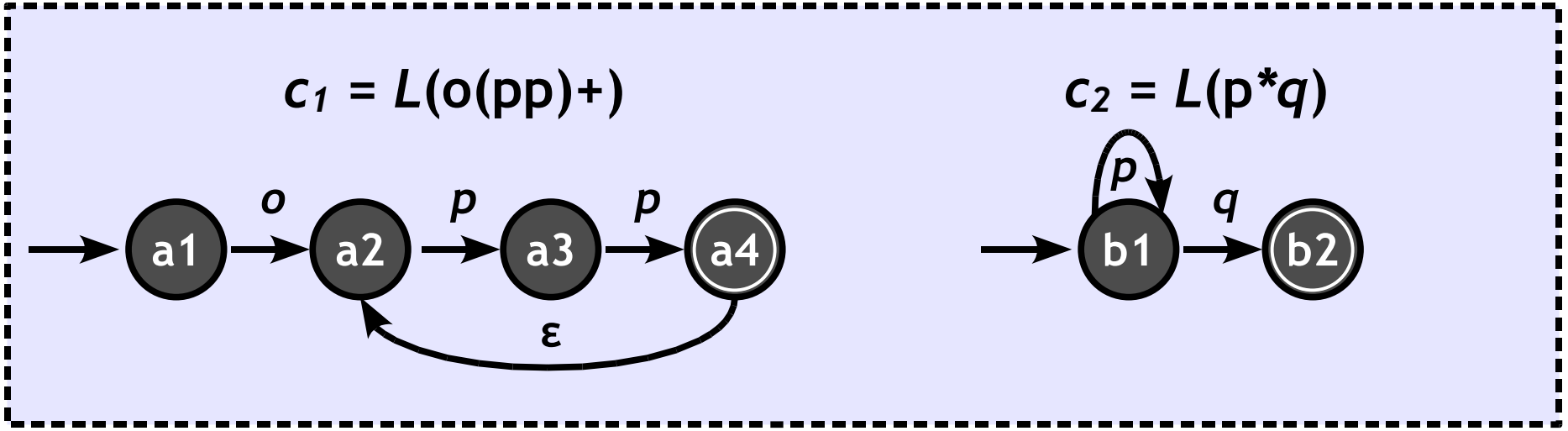
$c_2 = L(p^*q)$

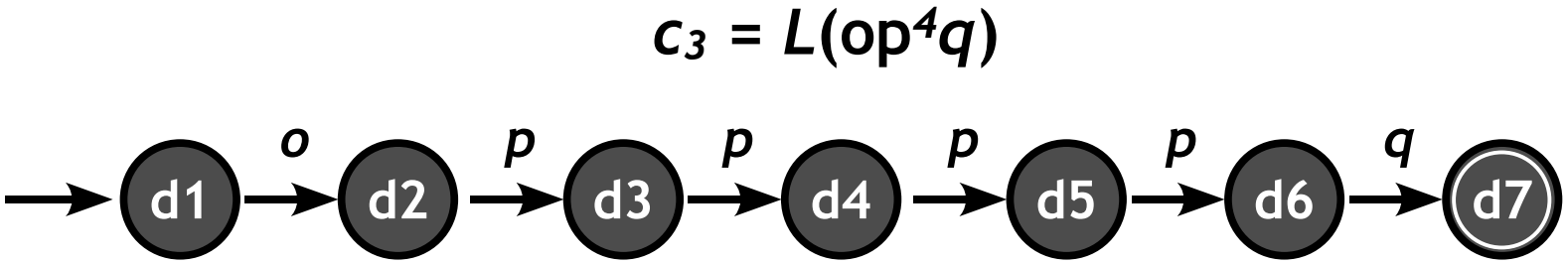


$c_3 = L(op^4q)$

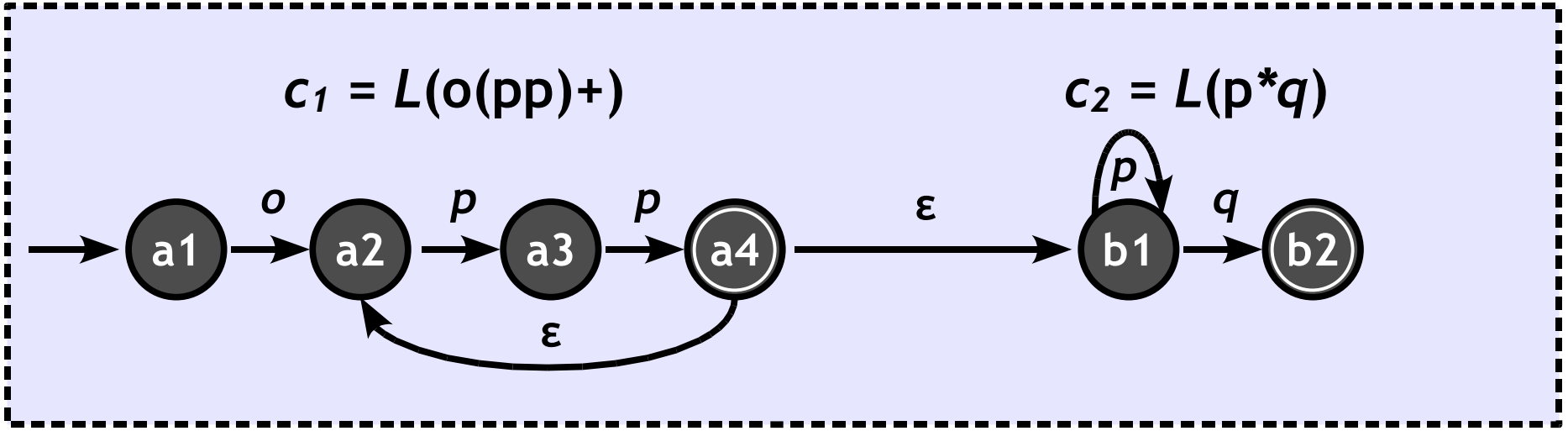


An Example

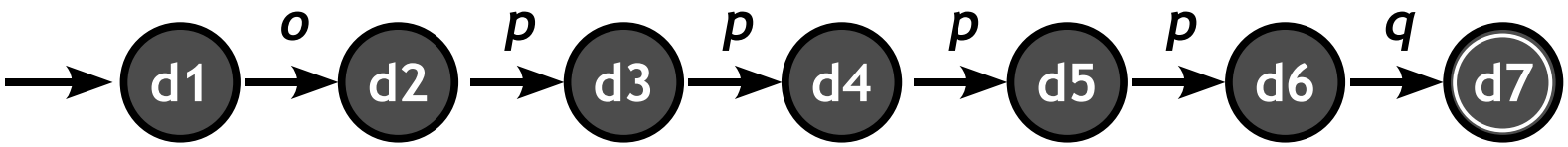

$c_1 = L(o(pp)+)$
a1
o
a2
p
a3
p
a4
ε
$c_2 = L(p^*q)$
p
q
b1
b2

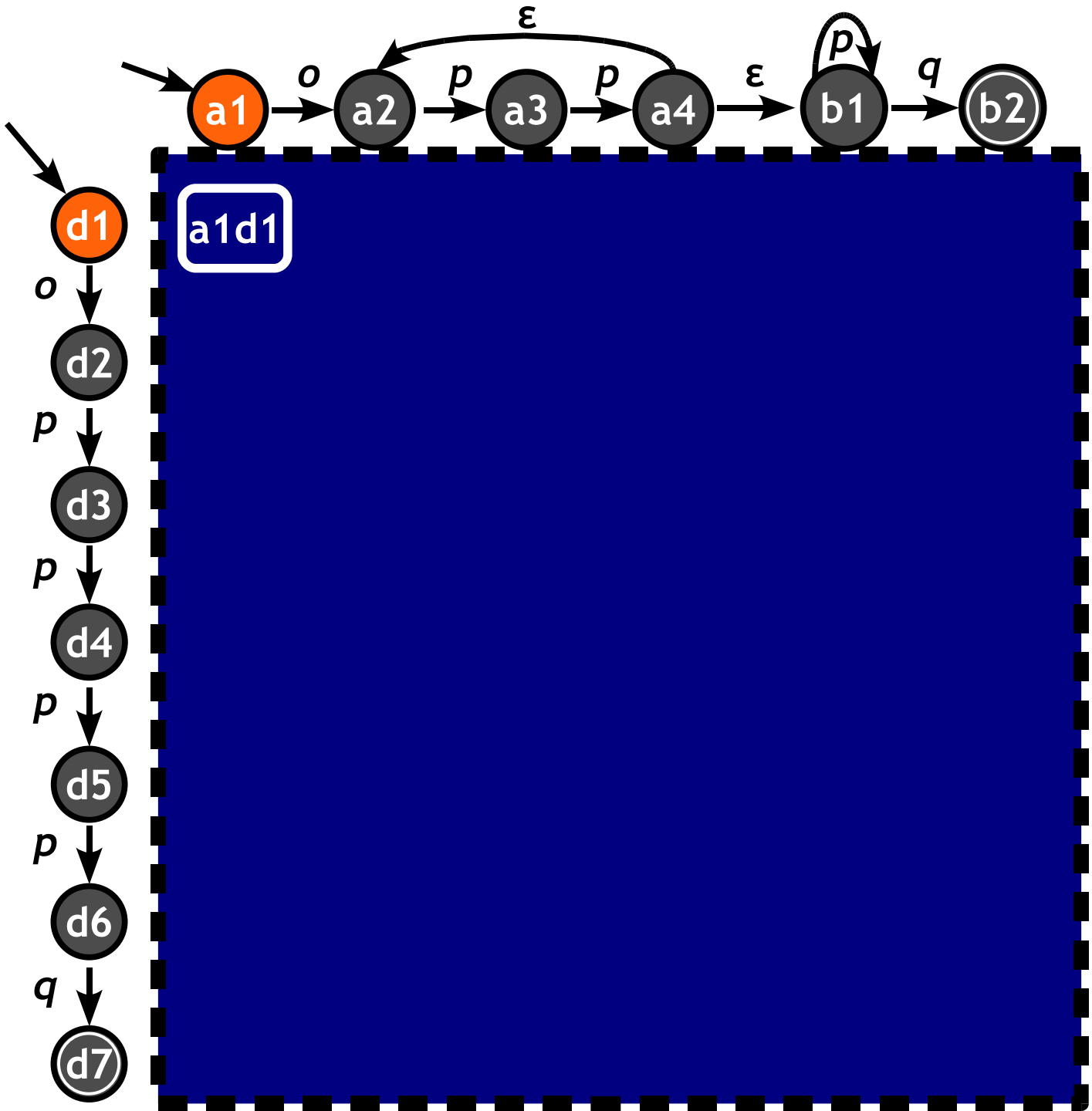

$c_3 = L(op^4q)$
d1
o
d2
p
d3
p
d4
p
d5
p
d6
q
d7

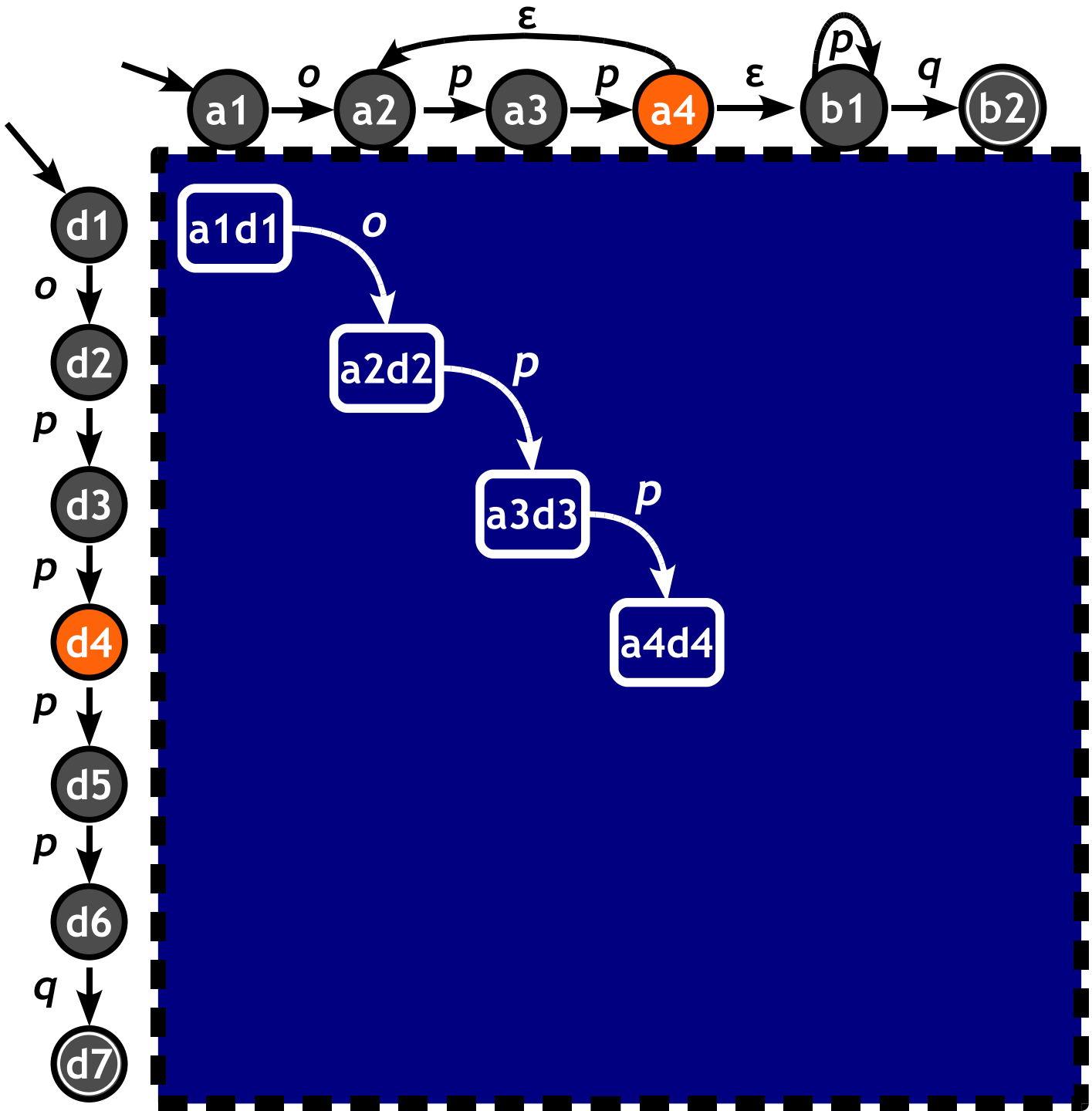
An Example

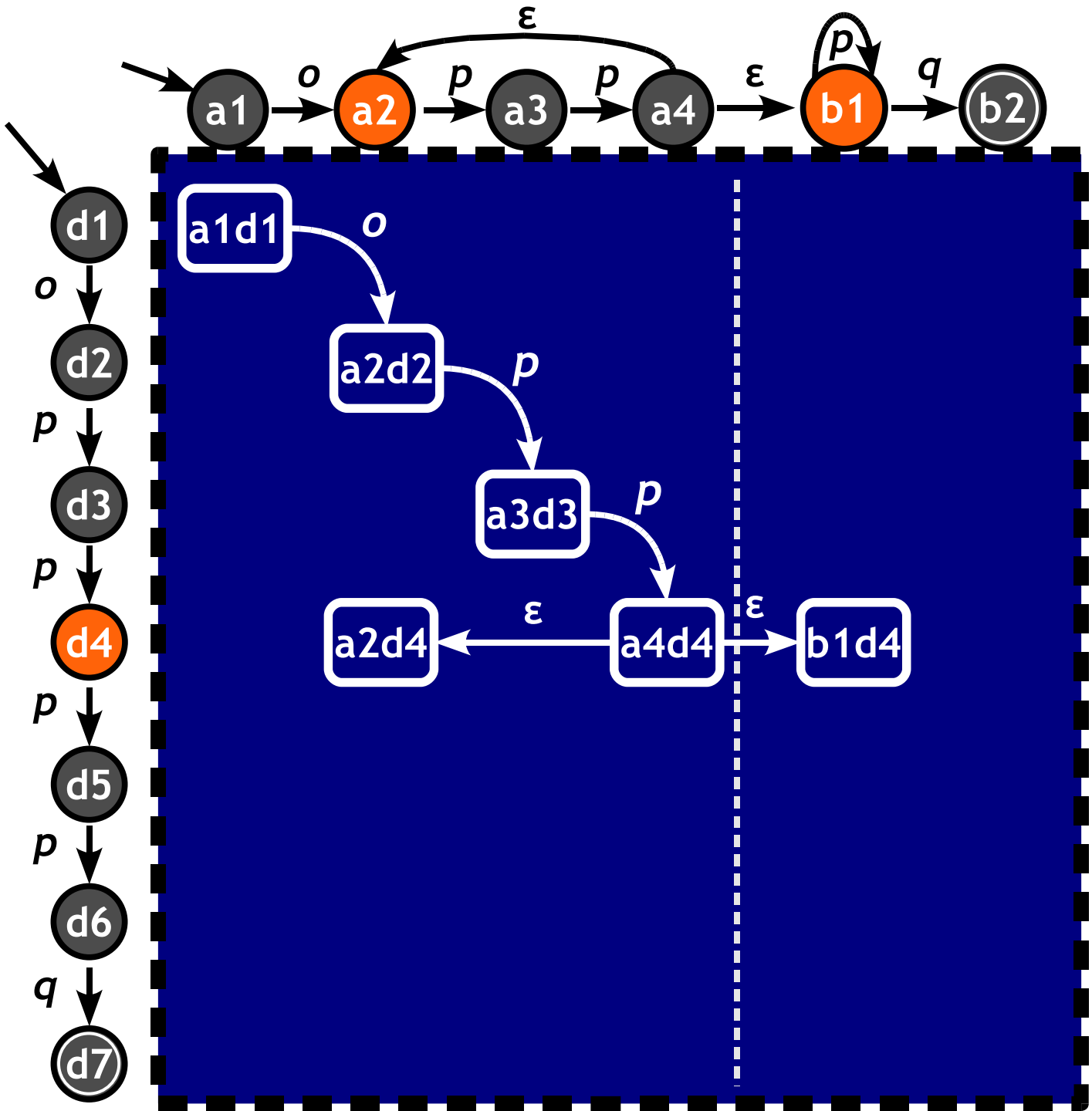


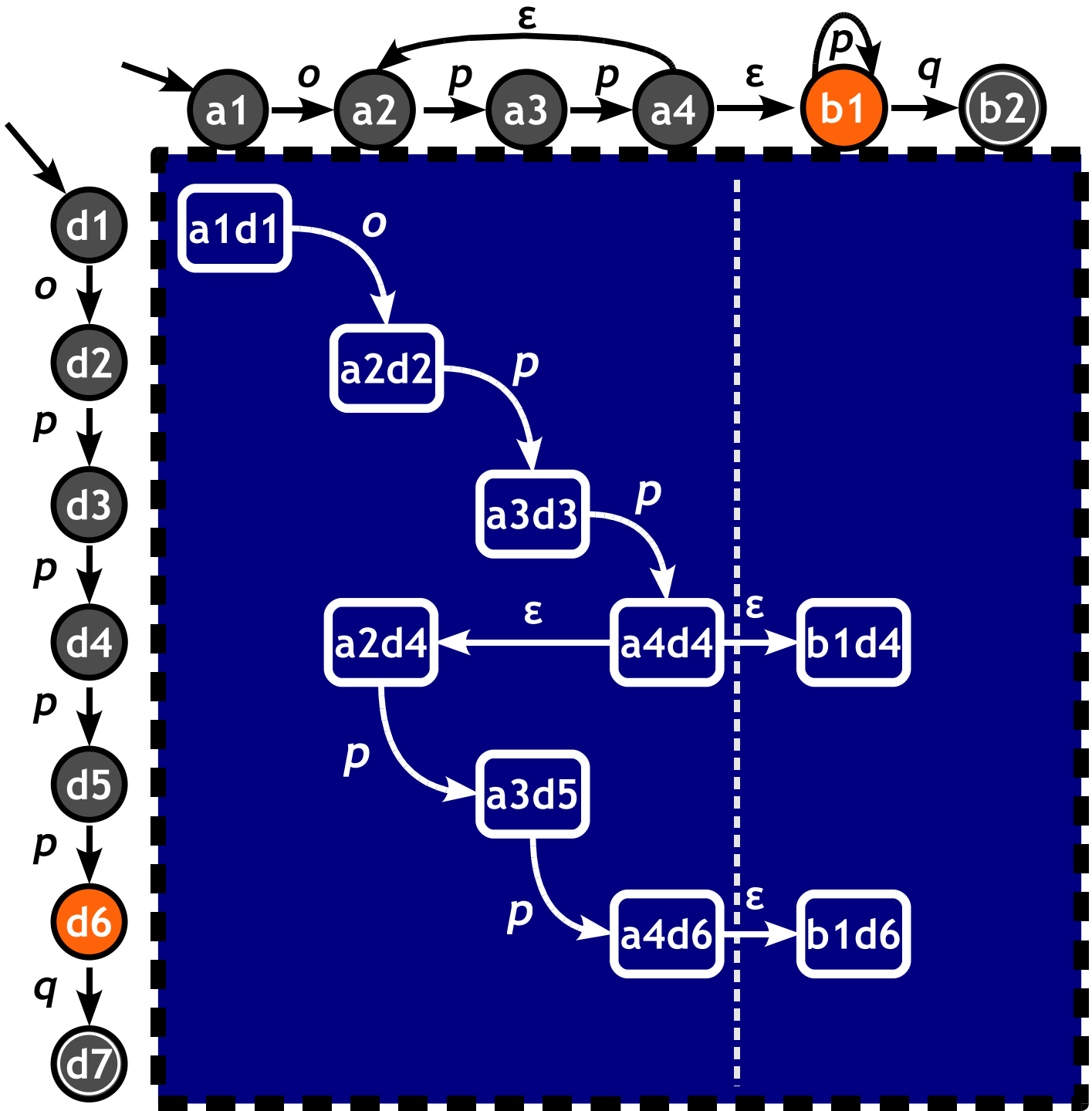
$c_3 = L(op^4q)$

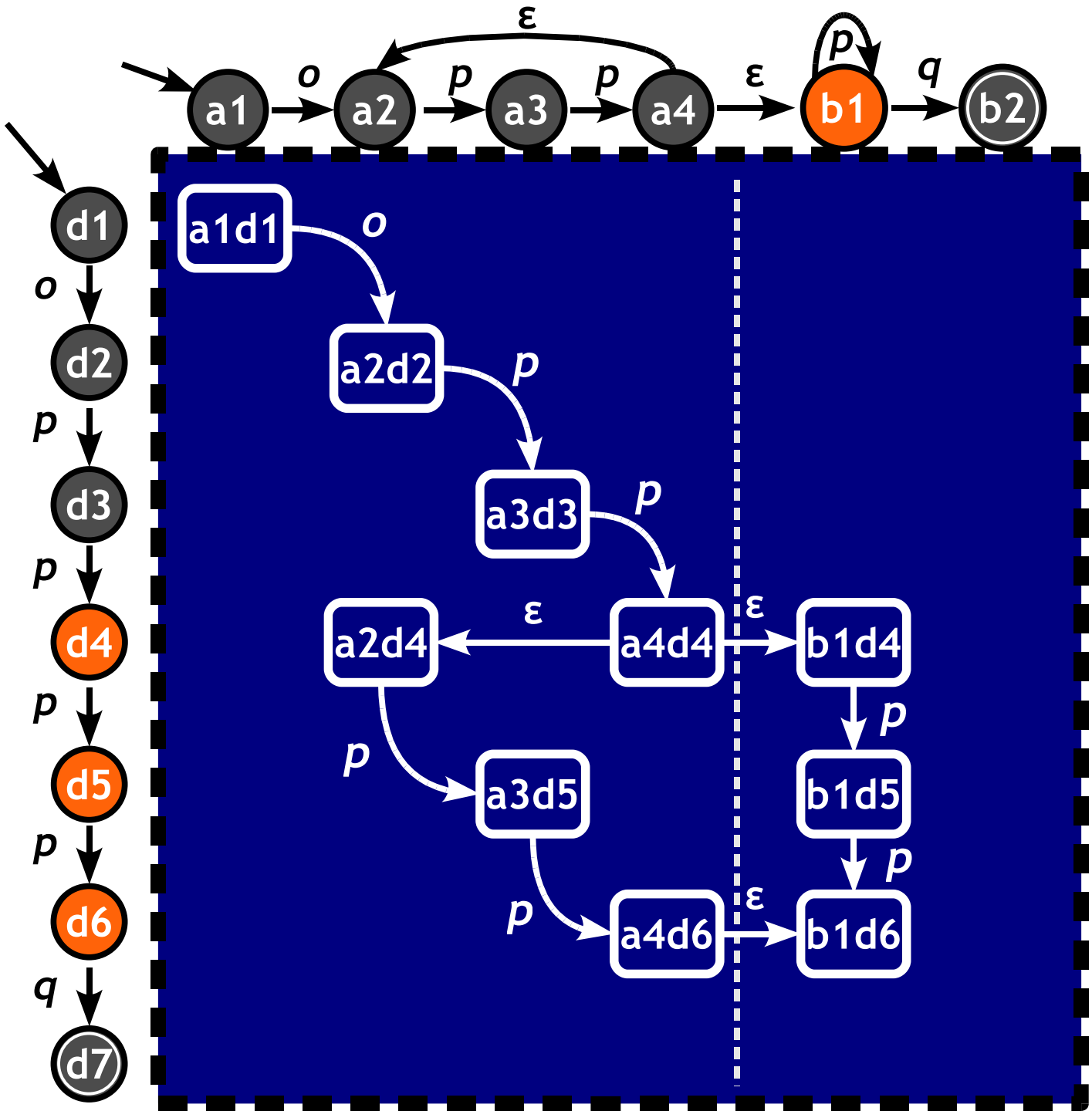


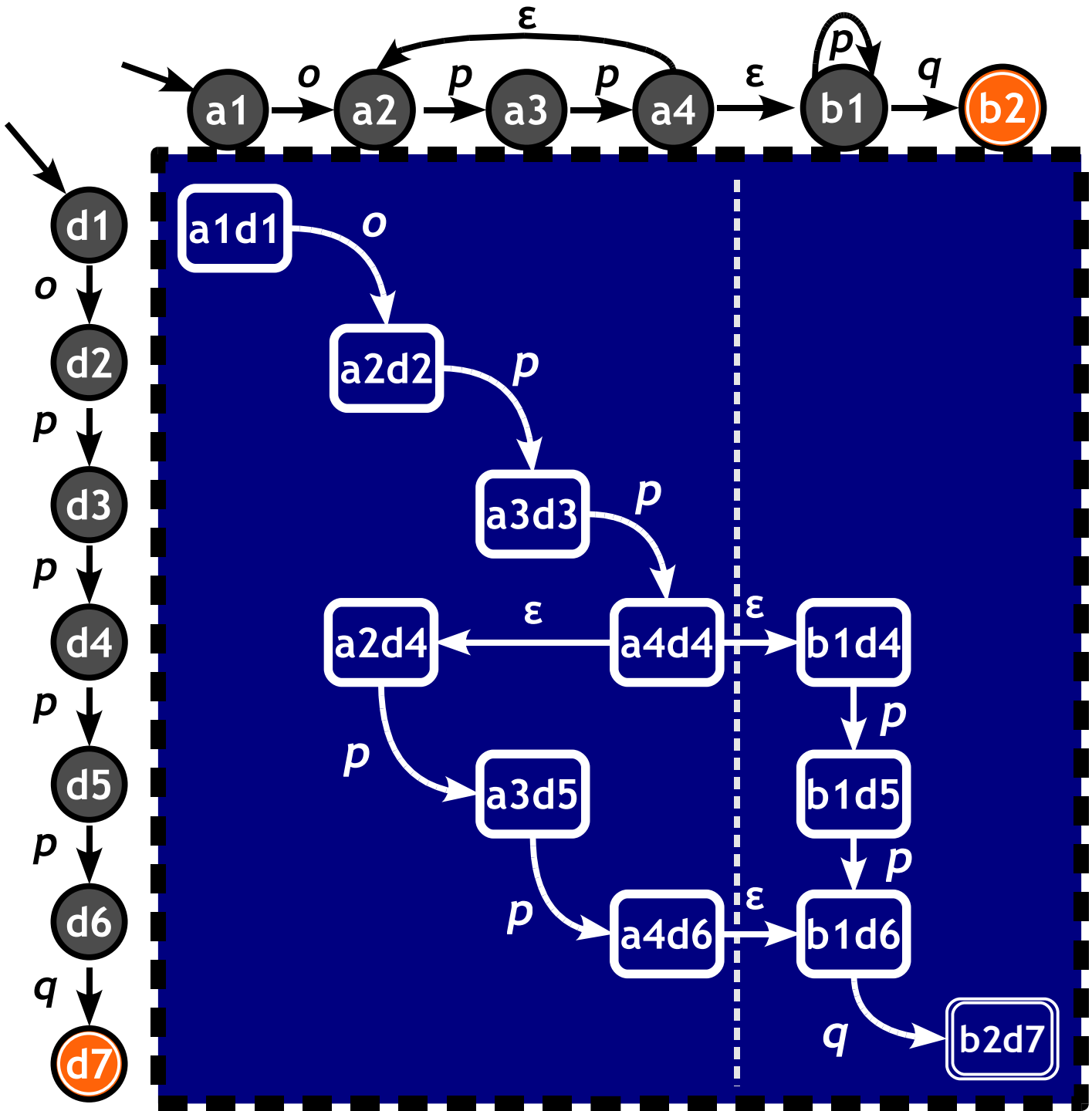

ε
o
p
p
ε
p
q
a1
a2
a3
a4
b1
b2
a1d1
d1
o
d2
p
d3
p
d4
p
d5
p
d6
q
d7

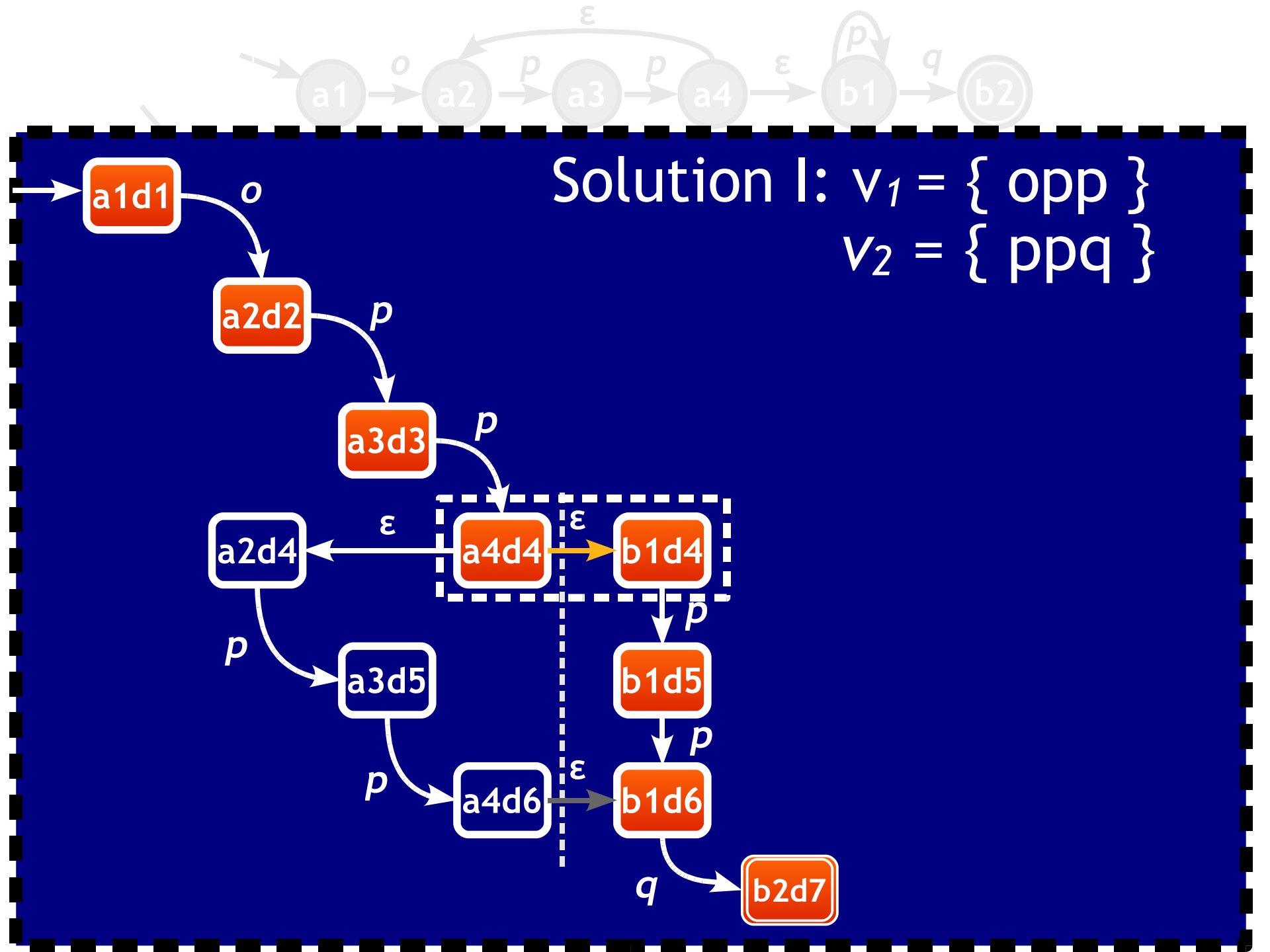

ε
o
p
p
ε
p
q
a1
a2
a3
a4
b1
b2
d1
o
d2
p
d3
p
d4
p
d5
p
d6
q
d7
a1d1
o
a2d2
p
a3d3
p
a4d4

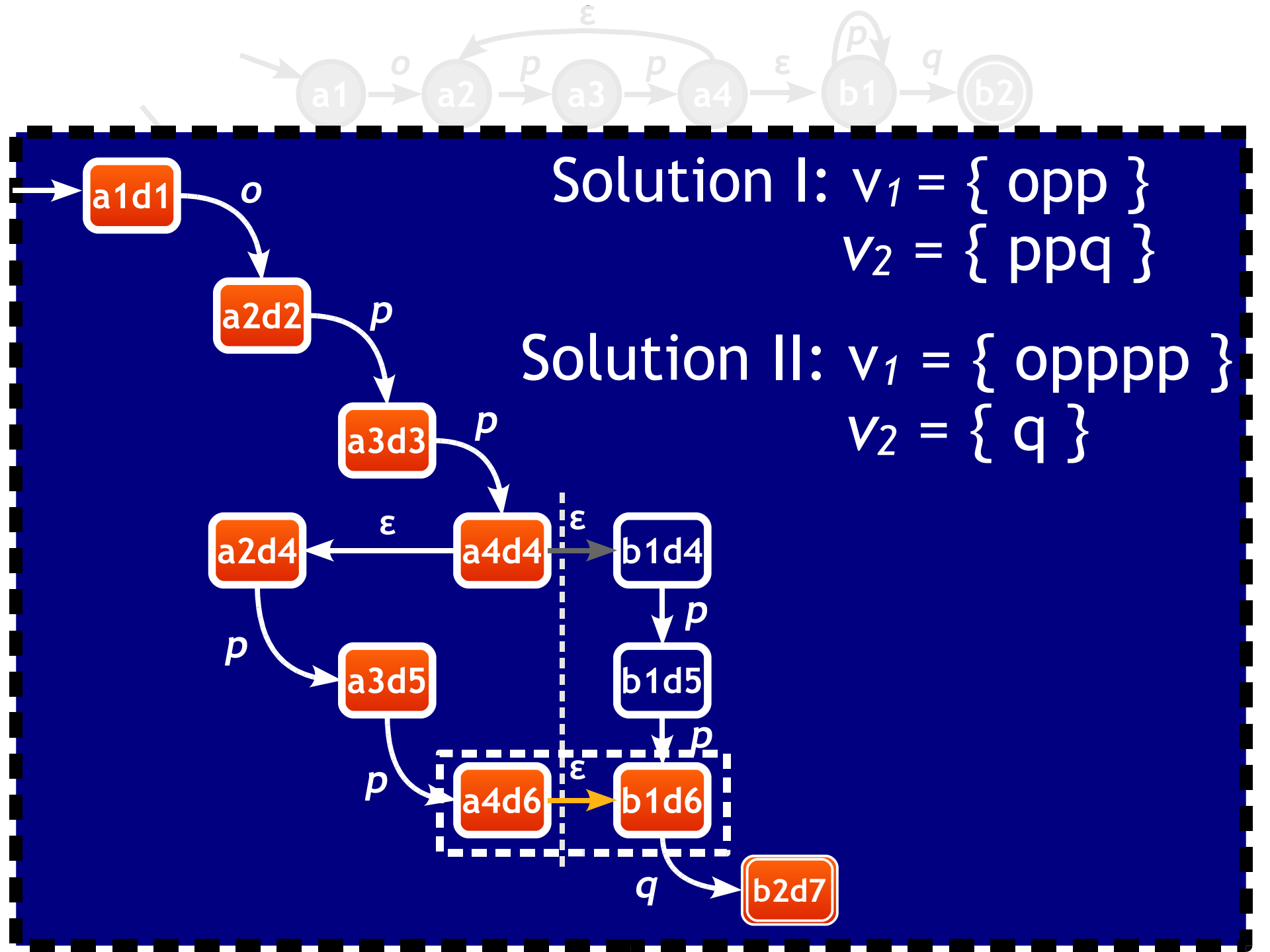

ε
o
p
p
ε
p
q
a1
a2
a3
a4
b1
b2
d1
o
d2
p
d3
p
d4
p
d5
p
d6
q
d7
a1d1
o
a2d2
p
a3d3
p
a2d4
ε
a4d4
ε
b1d4


ε
o
p
p
ε
p
q
a1
a2
a3
a4
b1
b2
d1
o
d2
p
d3
p
d4
p
d5
p
d6
q
d7
a1d1
o
a2d2
p
a3d3
p
a2d4
ε
a4d4
ε
b1d4
p
a3d5
p
a4d6
ε
b1d6


ε
o
p
p
ε
p
q
a1
a2
a3
a4
b1
b2
d1
o
d2
p
d3
p
d4
p
d5
p
d6
q
d7
a1d1
o
a2d2
p
a3d3
p
ε
ε
a2d4
a4d4
b1d4
p
p
a3d5
b1d5
p
p
ε
a4d6
b1d6


ε
o
p
p
ε
p
q
a1
a2
a3
a4
b1
b2
d1
o
d2
p
d3
p
d4
p
d5
p
d6
q
d7
a1d1
o
a2d2
p
a3d3
p
a2d4
ε
a4d4
ε
b1d4
p
b1d5
p
b1d6
q
b2d7
p
a3d5
p
a4d6
ε


a1
o
a2
p
a3
p
a4
ε
b1
p
q
b2
Solution I: v1 = { opp }
v2 = { ppq }
a1d1
o
a2d2
p
a3d3
p
a2d4
ε
a4d4
ε
b1d4
p
p
a3d5
b1d5
p
p
a4d6
ε
b1d6
q
b2d7


a1
o
a2
p
a3
p
a4
ε
b1
p
q
b2
Solution I: v1 = { opp }
v2 = { ppq }
Solution II: v1 = { opppp }
v2 = { q }
a1d1
o
a2d2
p
a3d3
p
a2d4
ε
a4d4
ε
b1d4
p
b1d5
p
a3d5
p
a4d6
ε
b1d6
q
b2d7

The Algorithm

- Concat-Intersect (CI) algorithm:
  - two variables, three constants; fixed form
  - mechanically verified proof in Coq 8.1pl3
  - proof size is ~1300 lines

- Regular Matching Assignments (RMA):
  - implemented in a tool, DPrle
  - applies *CI* procedure inductively

Outline

- Problem definition
- Algorithm
  - example execution
  - correctness
- Experiment

Experimental Design

- Find SQL injection vulnerabilities [Wassermann and Su; PLDI07]
- For each vulnerability:
  - generate bad SQL + path
  - check path consistency (Simplify)
  - solve string constraints (*this*)

The Data

Name	LOC	No. Vuln.
eve 1.0	905	1
utopia 1.3.0	5438	4
warp 1.2.1	24365	12

Example

```
// $userid is untrusted
if (!eregi('[0-9]+', $userid)) {
   exit;
}

$user = query("SELECT * FROM `unp_user`".
              "WHERE userid='$userid'");
```

Example

```
// $userid is untrusted
if (!eregi('[0-9]+', $userid)) {
   exit;
}

$user = query("SELECT * FROM `unp_user`".
              "WHERE userid='$userid'");
```

'^[0-9]+$'

	|C|	T(s)
1	29	0.320
2	16	0.052
3	16	0.001
4	156	0.650
5	102	0.260
6	10	0.054
7	4	0.010
8	10	0.530
9	10	0.063

	|C|	T(s)
10	31	0.170
11	41	0.430
12	81	577.000
13	10	0.057
14	66	0.220
15	387	0.500
16	10	0.052
17	10	0.180

	|C|	T(s)
1	29	0.320
2	16	0.052
3	16	0.001
4	156	0.650
5	102	0.260
6	10	0.054
7	4	0.010
8	10	0.530
9	10	0.063

	|C|	T(s)
10	31	0.170
11	41	0.430
12	81	577.000
13	10	0.057
14	66	0.220
15	387	0.500
16	10	0.052
17	10	0.180

	|C|	T(s)
1	29	0.320
2	16	0.052
3	16	0.001
4	156	0.650
5	102	0.260
6	10	0.054
7	4	0.010
8	10	0.530
9	10	0.063

	|C|	T(s)
10	31	0.170
11	41	0.430
12	81	577.000
13	10	0.057
14	66	0.220
15	387	0.500
16	10	0.052
17	10	0.180

Conclusion

- Presented a theory and decision procedure for modeling string values
- Algorithm relies on basic NFA operations and tracking transitions
- Implementation is available now!

Future Work

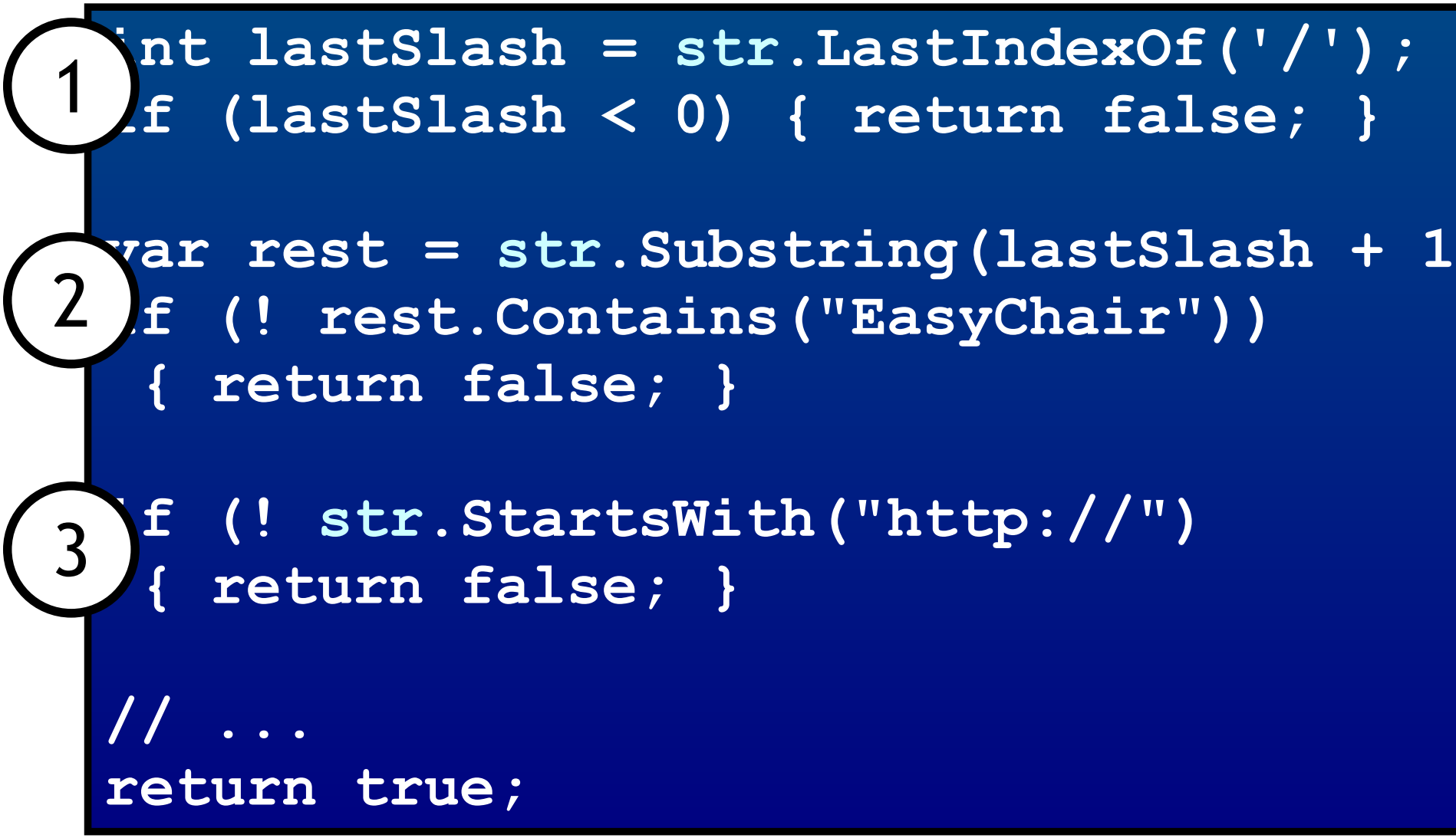
- Finding single string assignments quickly (*cf*. Hampi [ISSTA '09])
- Support for common idioms, including integer indexing
- Applications beyond testcase generation

```
int lastSlash = str.LastIndexOf('/');
if (lastSlash < 0) { return false; }

var rest = str.Substring(lastSlash + 1);
if (! rest.Contains("EasyChair"))
  { return false; }

if (! str.StartsWith("http://")
  { return false; }

// ...
return true;
```


1
2
3
int lastSlash = str.LastIndexOf('/');
if (lastSlash < 0) { return false; }
var rest = str.Substring(lastSlash + 1
if (! rest.Contains("EasyChair"))
{ return false; }
if (! str.StartsWith("http://")
{ return false; }
// ...
return true;

1 $v_1 \subseteq \Sigma^*/$

2 $v_2 \subseteq \Sigma^*\texttt{EasyChair}\Sigma^*$

3 $v_1 \circ v_2 \subseteq \texttt{http://}\Sigma^*$

Tool: Decision Procedure for Regular Language Equations

http://dprle.sf.net/

http://www.cs.virginia.edu/~ph4u/dprle/