Programming with Millions of Examples

Alon Mishne, Hila Peleg, Sharon Shoham,
Eran Yahav, Hongseok Yang

Components are everywhere

Components are Accessed Via APIs

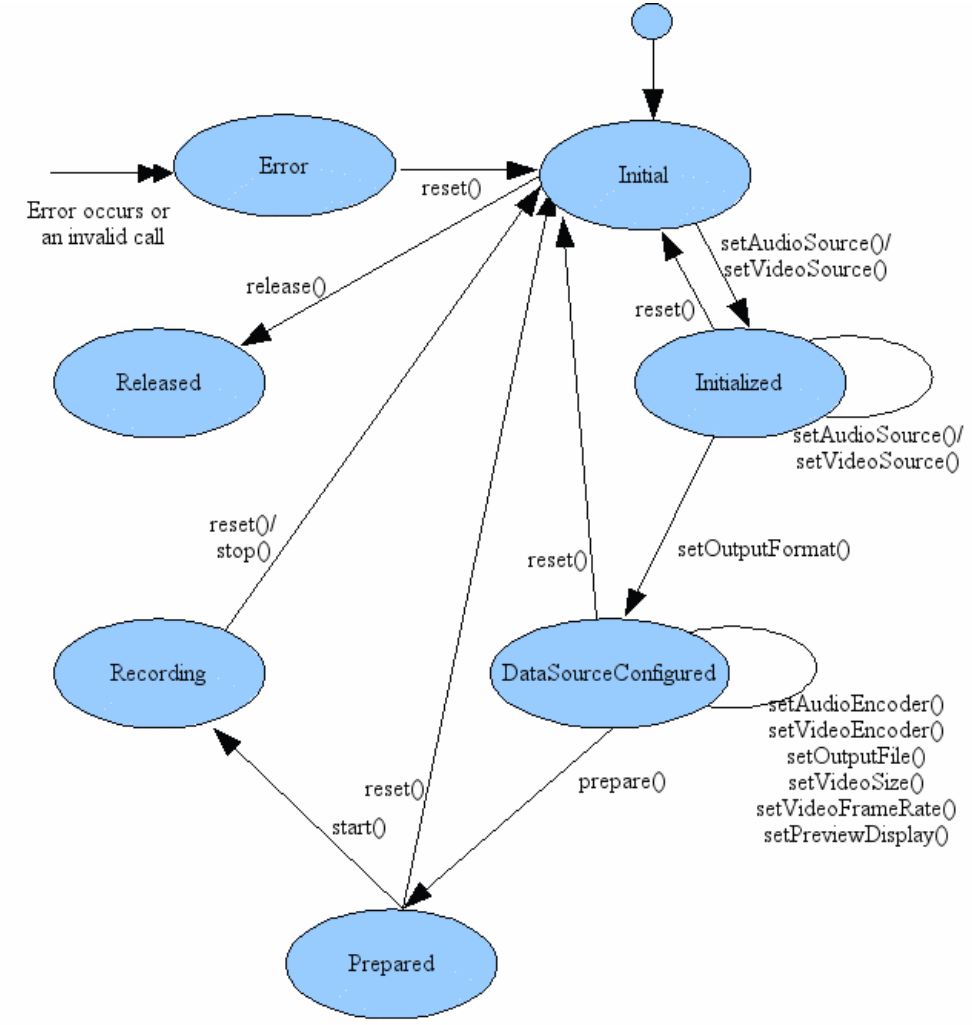
```
class File {
  File(String);
  void open();
  void close();
  byte read();
  void write(byte);
}
```

```
void writeBuff(File f, byte[] buff) {
  f.open();
  for (byte b : buff) f.write(b);
  f.close();
}
```

Component APIs are Complicated

There is only one thing more painful than learning from experience and that is not learning from experience.
– Archibald MacLeish

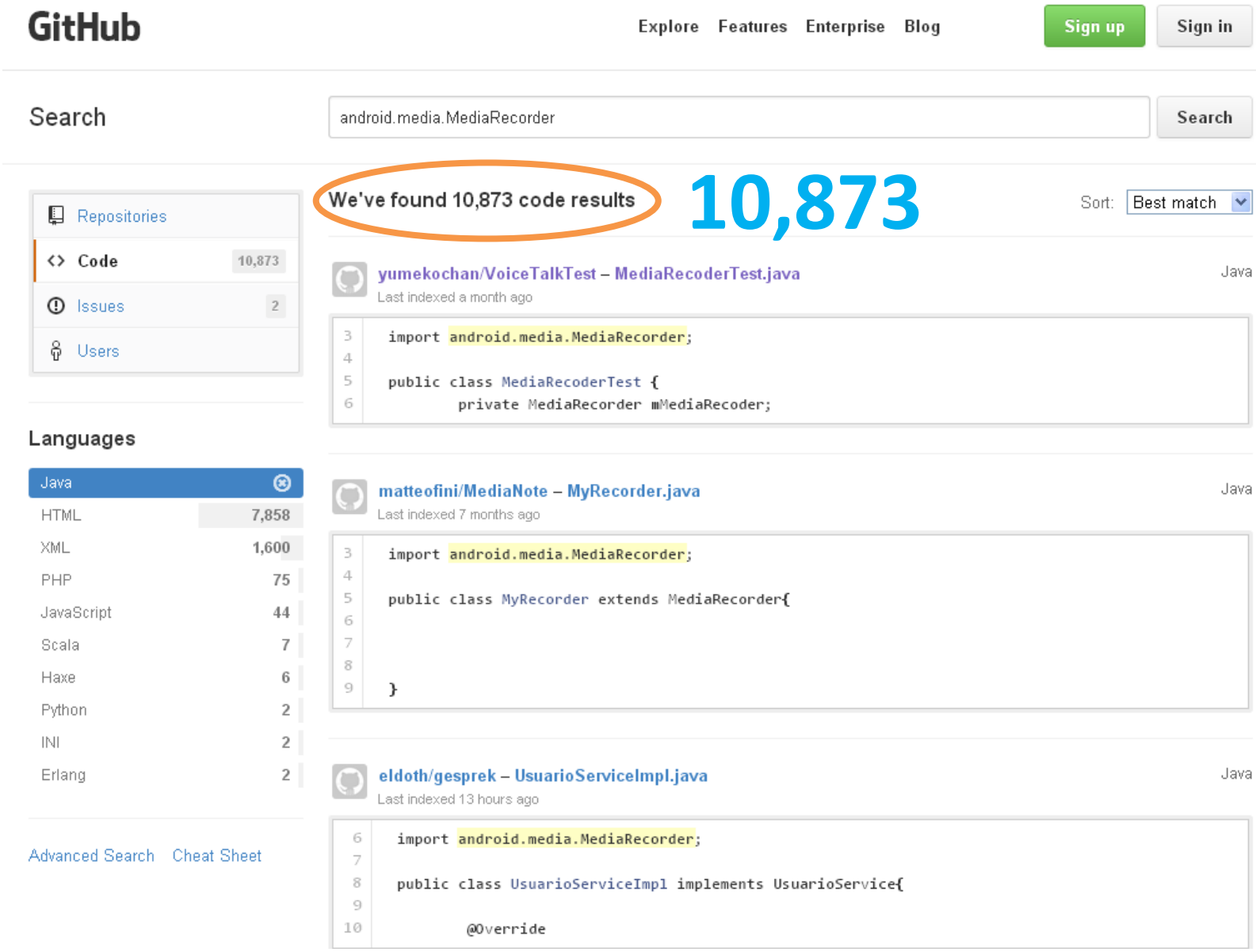
Component APIs are complicated



MediaRecorder state diagram

From the official Android documentation for MediaRecorder (http://developer.android.com/reference/android/media/MediaRecorder.html)

Examples are prevalent



Stats: > 3.5M users, over 8M repositories, growing daily

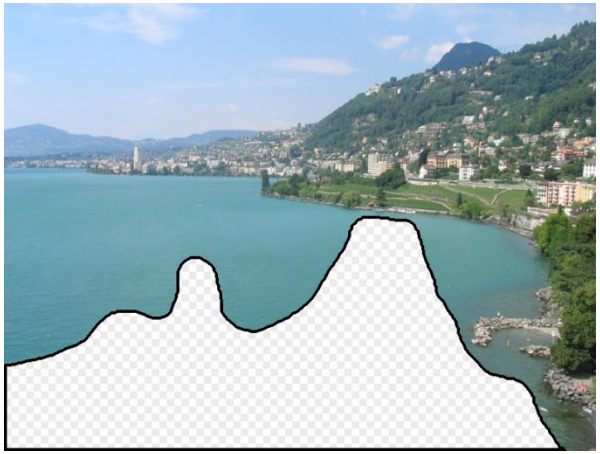
Examples are prevalent

Stats: > 2.3M registered users, > 5.6M questions, > 10M answers

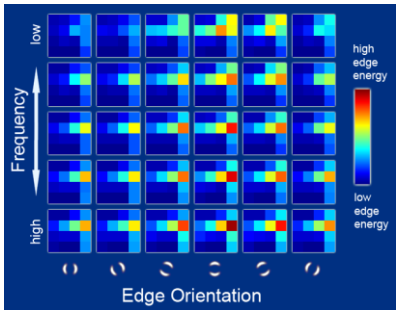
Challenge

How do we leverage the millions of examples to synthesize client code using the library?

Scene Completion from Millions of Photographs



Query



Semantic Descriptor

Semantic Index

200 matches

Context matching + blending

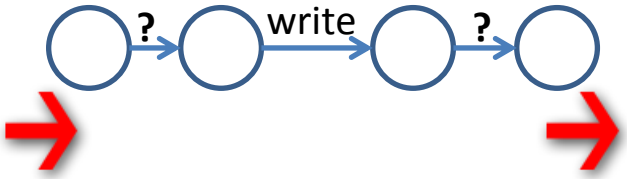
20 completions

Adapted from: Hays and Efros, SIGGRAPH 2007

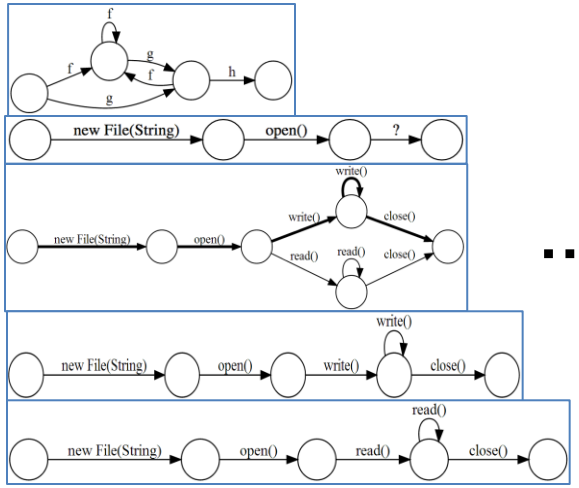
Code Completion from Millions of Examples

```
File f = ?
f.write("73");
f.?
```

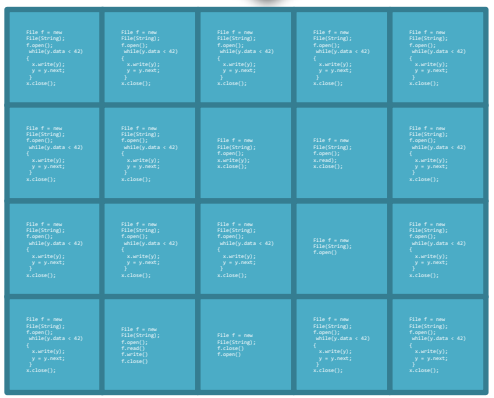
Query



Semantic Descriptor



Semantic Index




```
File x = new File();
x.open();
 while(y.data < 42) {
  x.write(y);
  y = y.next;
 }
x.close();
```

Context matching + blending

```
File f = new File();
f.open();
//while(y.data < 42) {
  f.write("73");
//  y = y.next;
//}
f.close();
```

1 of 20 completions

200 matches

Your wish is my command,
do you know **what is your wish**?

Discovery

Android location services

LocationManager x = ?
Location y = ?
y.getLastKnownLocation(?)

```
LocationManager manager =
    (LocationManager) context.getSystemService(LOCATION_SERVICE);
String provider = manager.getBestProvider(criteria, true);
boolean gpsEnabled =
    locationManager.isProviderEnabled(GPS_PROVIDER);
boolean networkEnabled =
    locationManager.isProviderEnabled(NETWORK_PROVIDER);
…
Location latestLocation = null;
if (networkEnabled)
  latestLocation = manager.getLastKnownLocation(GPS_PROVIDER);
else if (gpsEnabled)
  latestLocation = manager.getLastKnownLocation(NETWORK_PROVIDER);
…
return latestLocation;
```

Query

1 of 20 matches

Naïve specification does not lead to “common” solution

Semantic Index: what do we want?

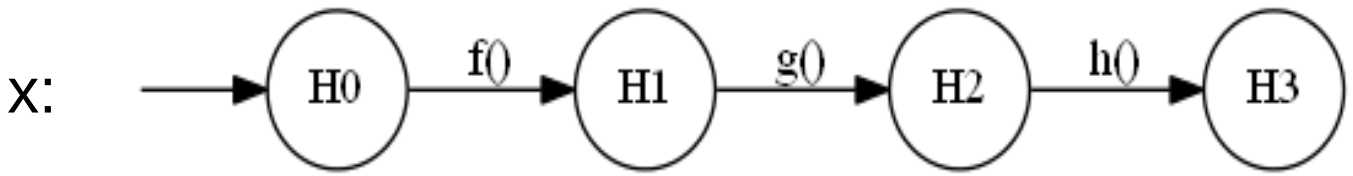
- Find semantically similar programs
  - Compare program behaviors
  - ***Represent temporal ordering of method invocations across multiple objects***

- Represent **partial information** due to partial programs
- Recover information using **consolidation**

- Feasible index construction and comparisons
- Precise enough to distinguish good from bad

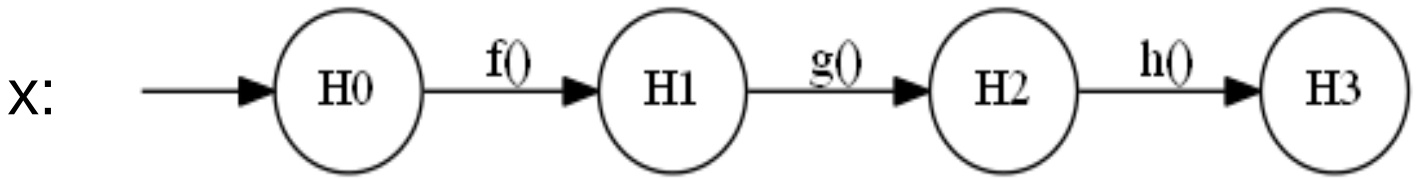
Concrete history

```
public void method(Something x) {
  x.f();
  x.g();
  x.h();
}
```



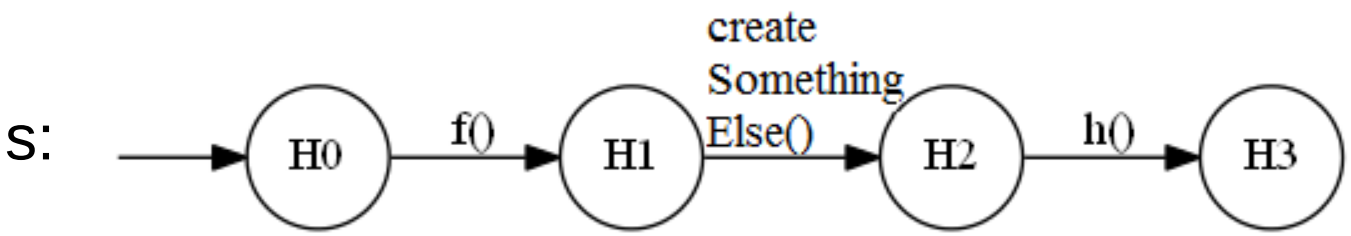
Aliasing

```
public void method(Something x) {
  x.f();
  BaseOfSomething y = x;
  y.g();
  y.h();
}
```



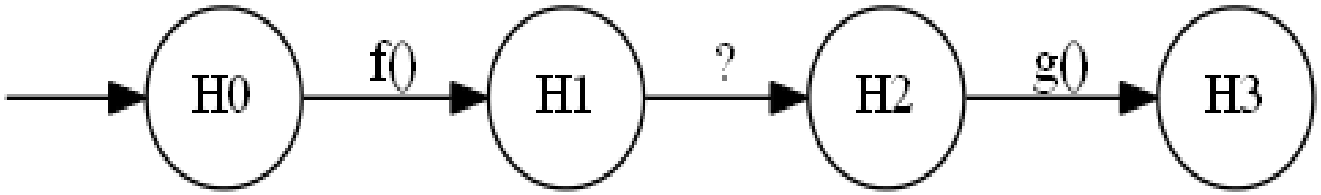
Creation context

```
public void method(Something x) {
  x.f();
  SomethingElse s = x.createSomethingElse();
  s.h();
}
```



Dealing with the unknown

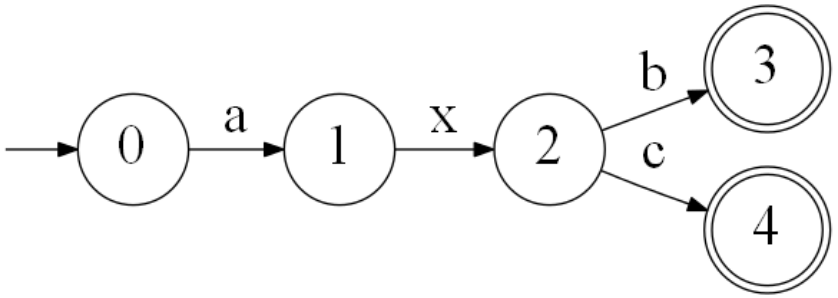
```
public void method1() {
  Something x = new Something();
  x.f();
  mystery(x);
  x.g();
}
```



Representing Histories: DSA

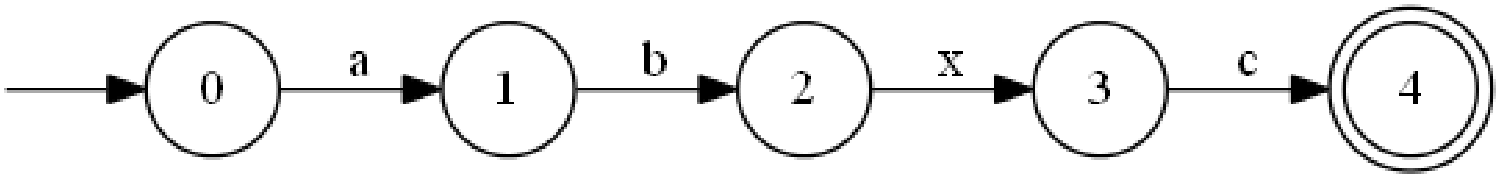
A Deterministic **Symbolic** Automaton is a tuple (Σ;Q;δ;ι;F;***Vars***)

- Σ is a finite alphabet
- Q is a finite set of states
- δ is the transition relation, $Q \times (\Sigma \cup Vars) \rightarrow Q$
- $\iota \in Q$ is the initial state
- $F \subseteq Q$ is the set of final states
- ***Vars* is the finite set of variables**

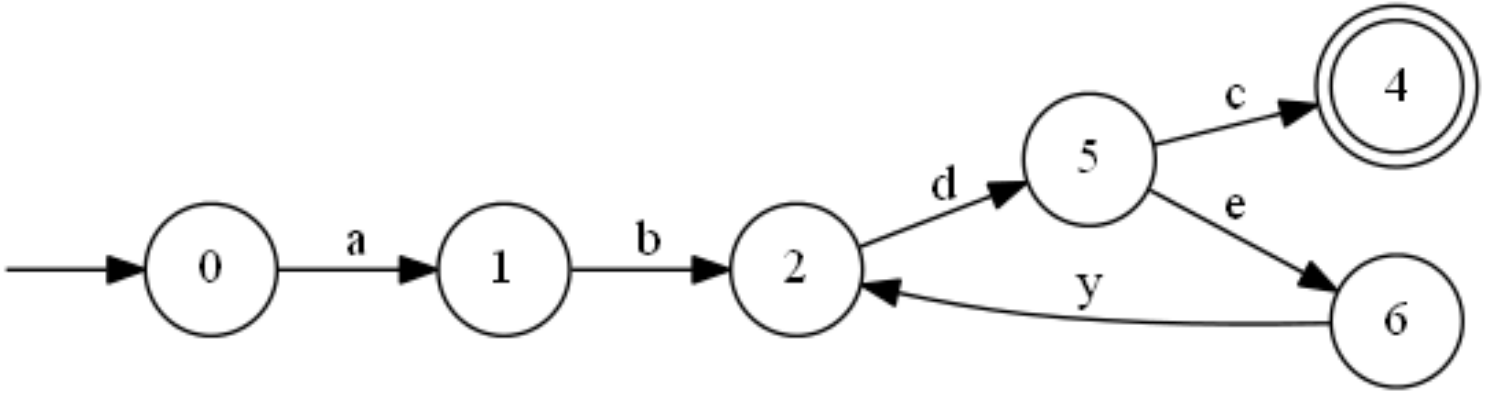


Assignment

An assignment σ maps a variable x in context sw1, sw2 (sw_1, x, sw_2) to a non-empty symbolic language



$$\sigma(\epsilon, x, \epsilon) = d(eyd)^*$$

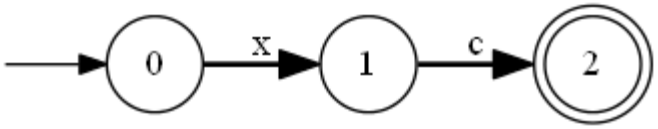


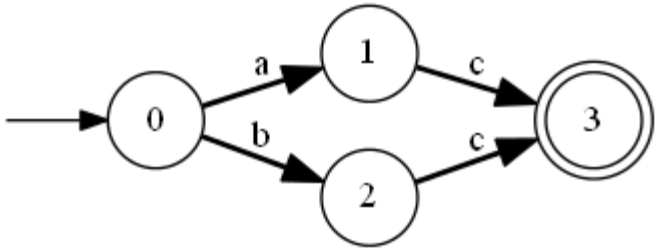
Desired Operations

- Analysis of individual example
  - Extend DSA with new call (known / unkown)
  - Join (later)
- Operations on multiple examples
  - Consolidation
  - Query matching

Order Between DSAs

- A natural way to define order between automata is language inclusion
- Won't work (directly) for symbolic automata:

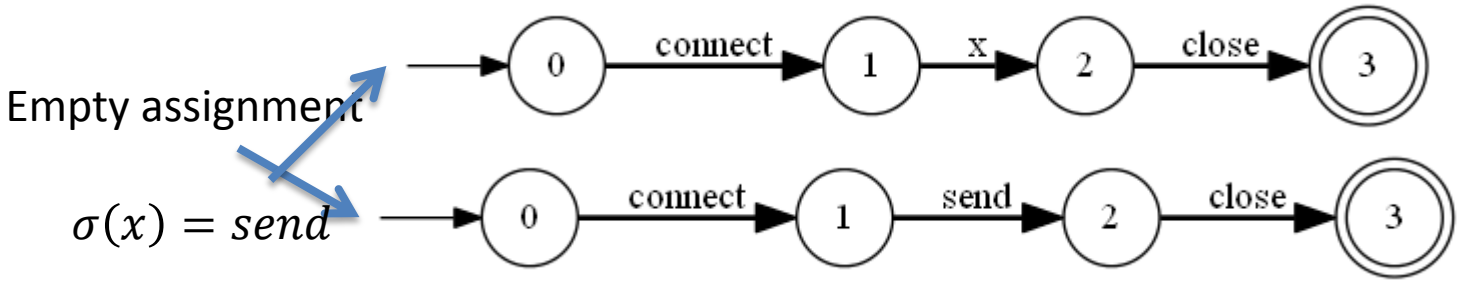




$\{xc\}$ $\not\subseteq$ $\{ac, bc\}$

Partialness

- A word is *less partial* (or *more complete*) than another if it represents a more concrete scenario

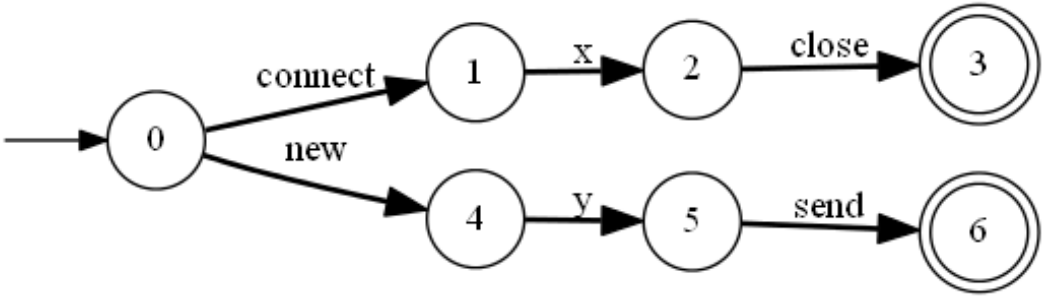


- Formally: w_1 is more partial than w_2 if for each assignment σ_2 to w_2 there is an assignment σ_1 to w_1 s.t. $\sigma_1(w_1) = \sigma_2(w_2)$

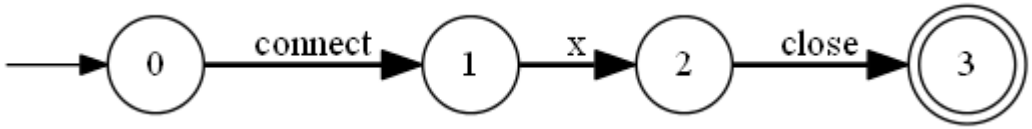
Partial Order

- We define the order over DSAs to capture both axes:
  - Precision: the natural concept of language inclusion
  - Partialness: of the individual words
- Intuitively: a DSA is smaller if it is more precise and more partial

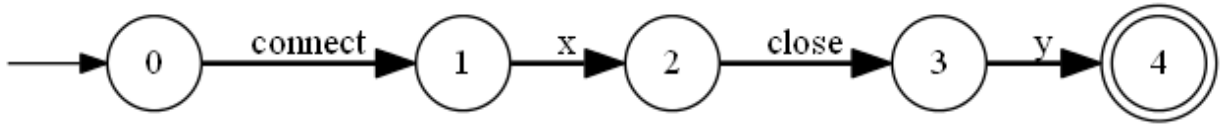
Precision vs. Partialness



The less *precise* you are, the more behaviors you describe



And the more *partial* you are, the more behaviors you describe…



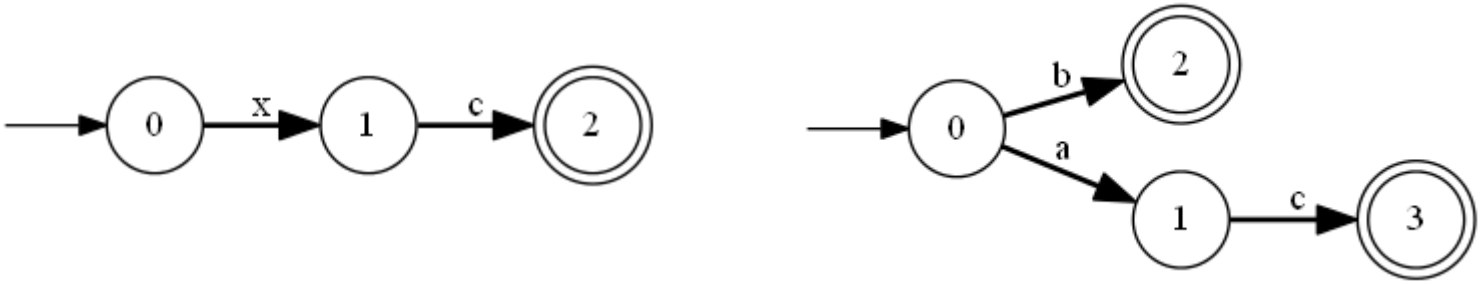
≤

DSA Order

$A_1 \leq A_2$ if

$$\forall \sigma_2 \exists \sigma_1 \; L(\sigma_1\,(A_1)) \subseteq L(\sigma_2(A_2))$$

where σ_1 and σ_2 are concrete assignments



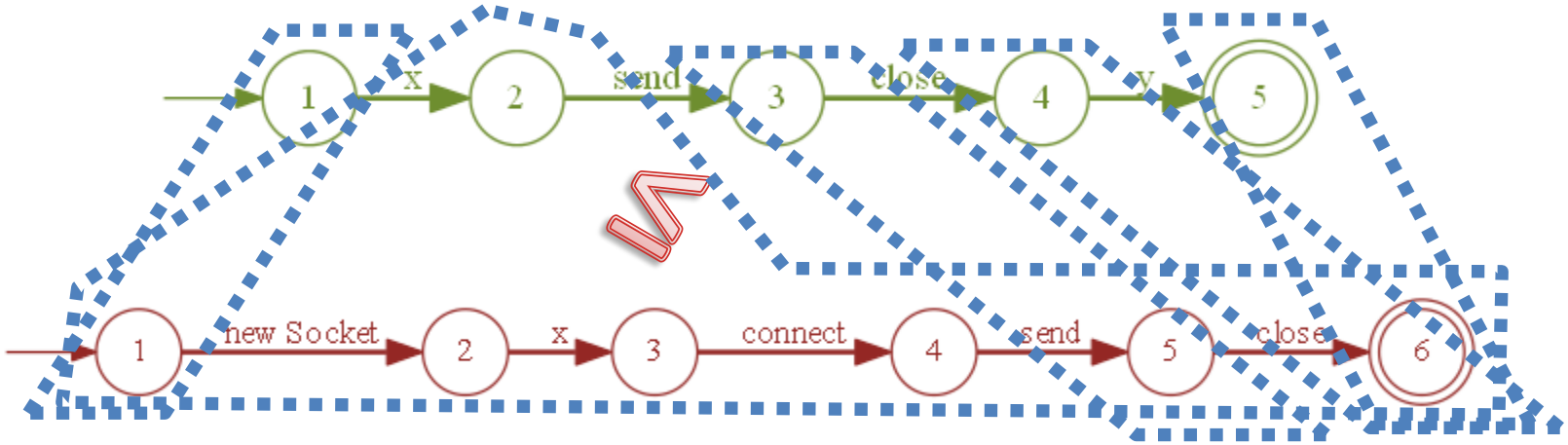
$$\sigma(x) = a, \text{ so } \{ac\} \subseteq \{ac, b\}$$

Calculating Inclusion via Simulation

- Want to determine whether $A_1 \leq A_2$
- Adapt DFA simulation to DSAs: **symbolic simulation**
  - Find pairs of **one state from A_1** (smaller) and a **set of states from A_2** (larger) that witness structural inclusion
  - Collect possible candidates using outgoing transitions
  - Shows us the matching word for each word in A_1

- DFA simulation already captures the notion of precision
- D**S**A simulation adds the notion of partialness
  - Symbols can "**swallow**" parts of the other DSA

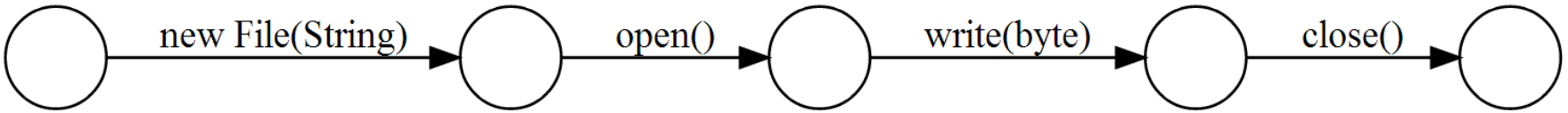
Simulation Example



✓(1,{1}),
(2,{1,2,3,4,5,6}),
(3,{5}),
(4,{6}),
(5,{6})

Index Construction: Client-Side Static Analysis

```
1    File f = new File(filename);
2    f.open();
3    f.write(z);
4    f.close();
```



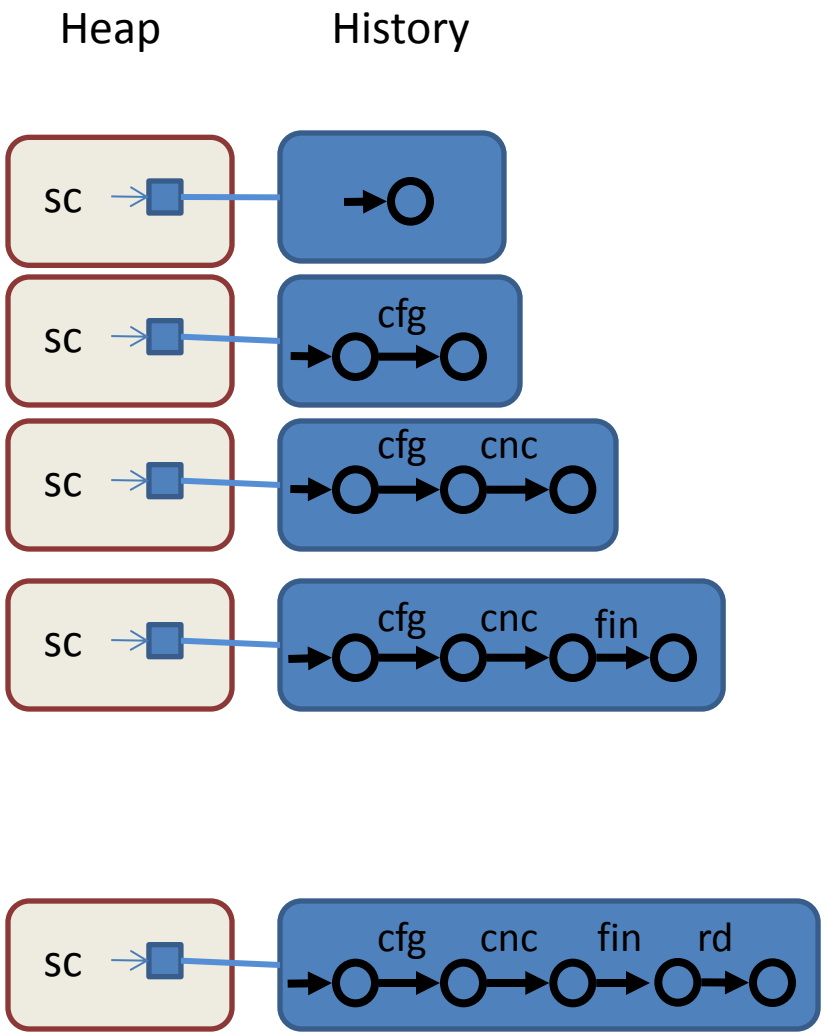
- Interprocedural
- Alias-aware with points-to information

Example

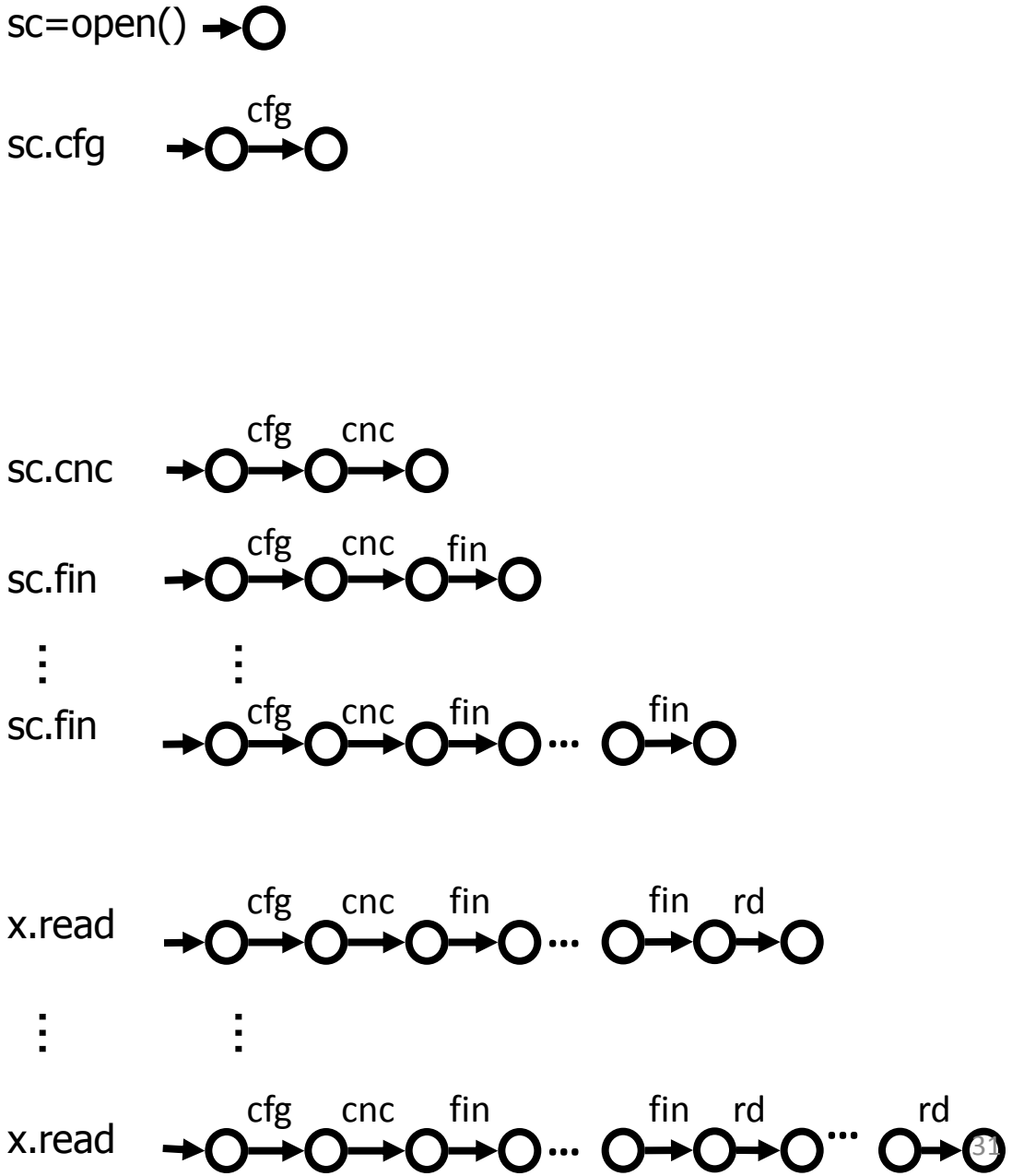
How should I use a java.nio.channels.SocketChannel?

Analyzing a Single Code Sample

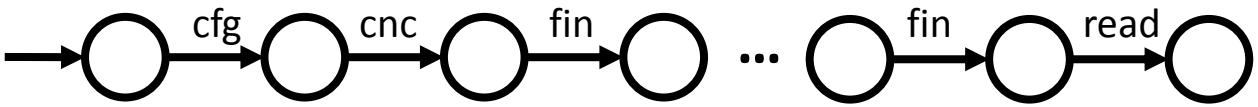
```
example1() {

  SocketChannel sc = SocketChannel.open();

  sc.configureBlocking(false);

 sc.connect();

 sc.finishConnect();

  ByteBuffer dst = …;

  sc.read(dst);
}
```



```
SocketChannel createChannel (…)
{
 SocketChannel sc =
     SocketChannel.open();
 sc.configureBlocking(false);
 return sc;
}

void example() {
 Collection<SocketChannel> chnls =
     createChannels();
 for (SocketChannel sc : chnls){
   sc.connect(new …);
   while (!sc.finishConnect()) { … }
   if (?) { receive(sc); }
     else { send(sc); }
 }
 closeAll(channels);
}


void receive(SocketChannel x) {
 …
 while (numBytesRead >= 0) {
   numBytesRead = x.read(dst);
   fos.write(dst.array());
 } …
}}
```



History Abstraction

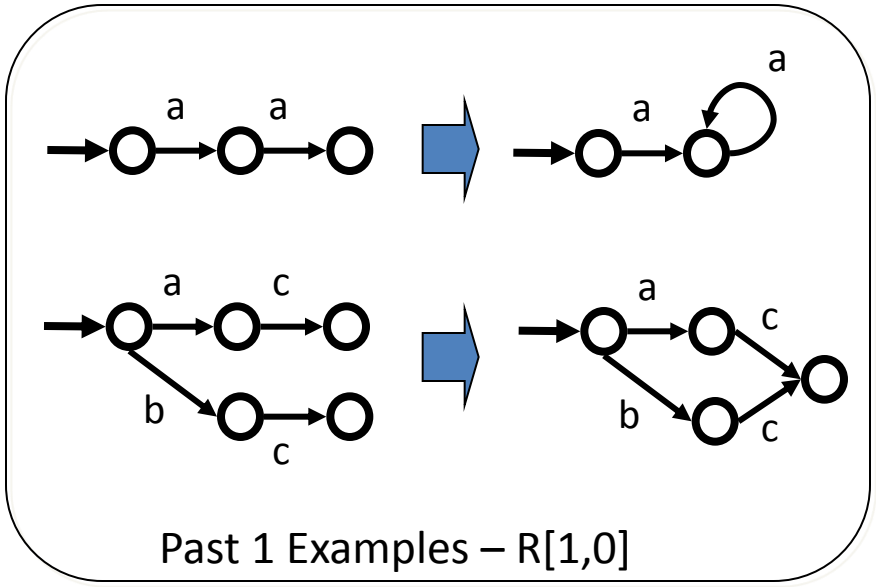


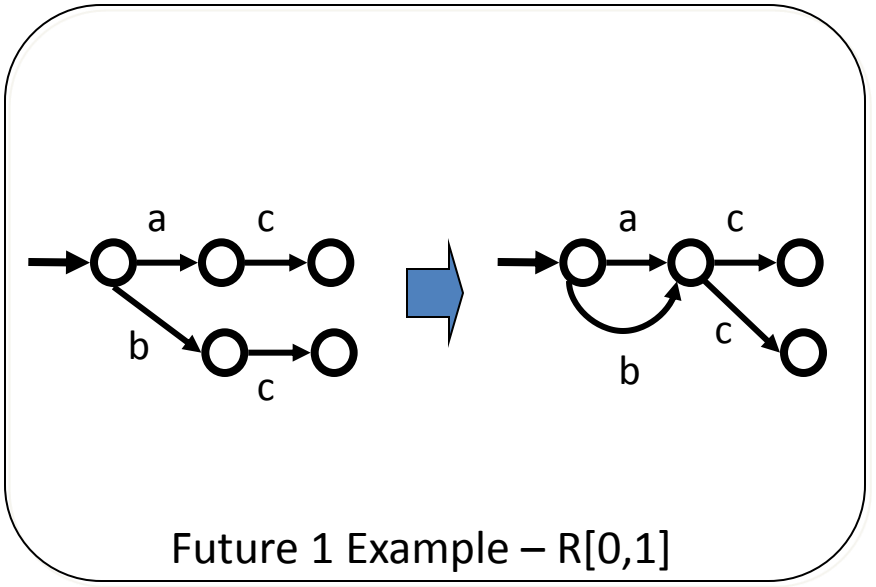
- Abstract history
  - DSA over-approximating unbounded histories
- **Quotient-based abstractions** for history
  - DSA states which are equivalent w.r.t. a given **equivalence relation R** are merged

History Abstraction

- Past-Future Abstraction

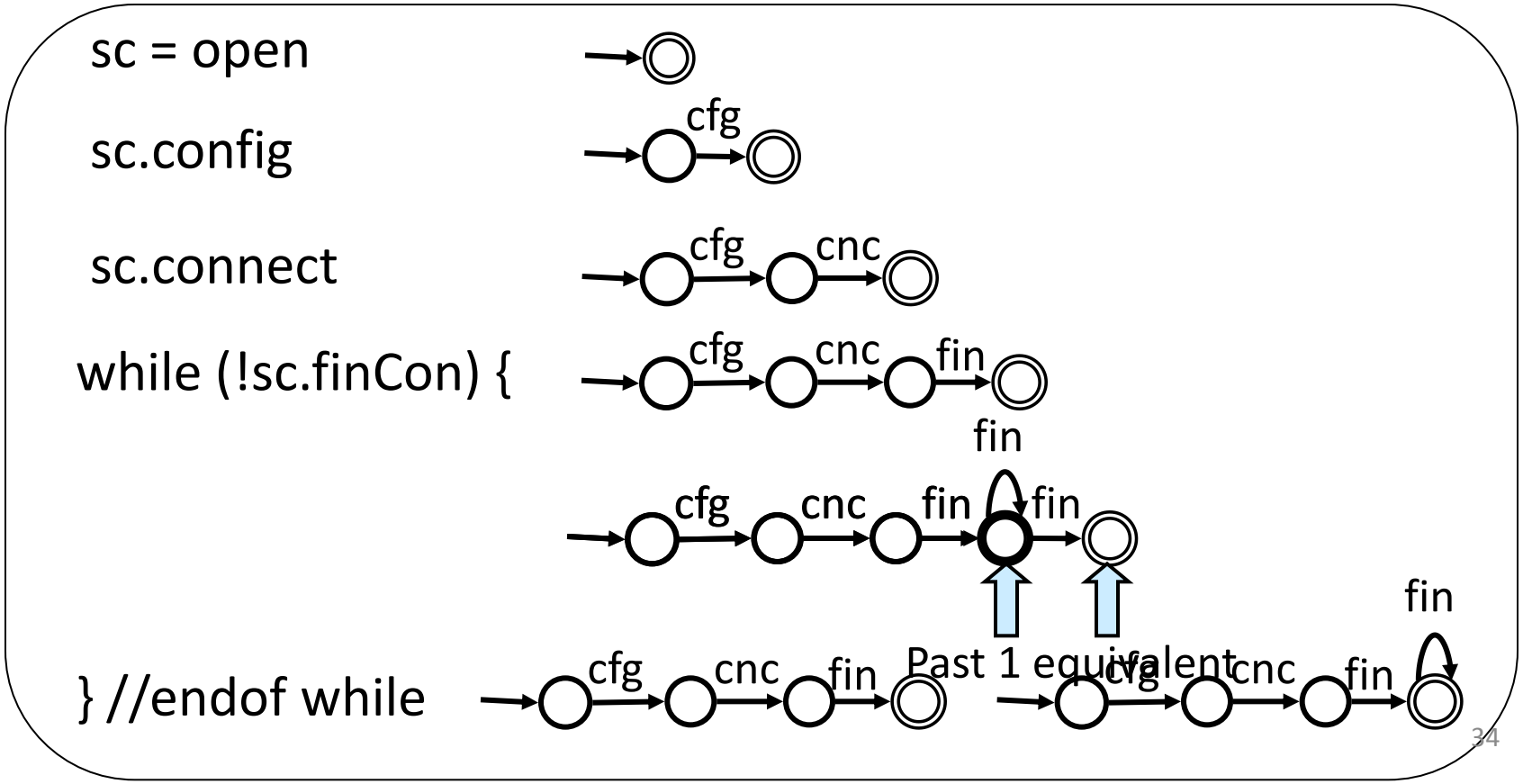
  $(q_1,q_2) \in R[k_{in},k_{out}]$ if q_1 and q_2 share both an incoming sequence of length k_{in} and an outgoing sequence of length k_{out}





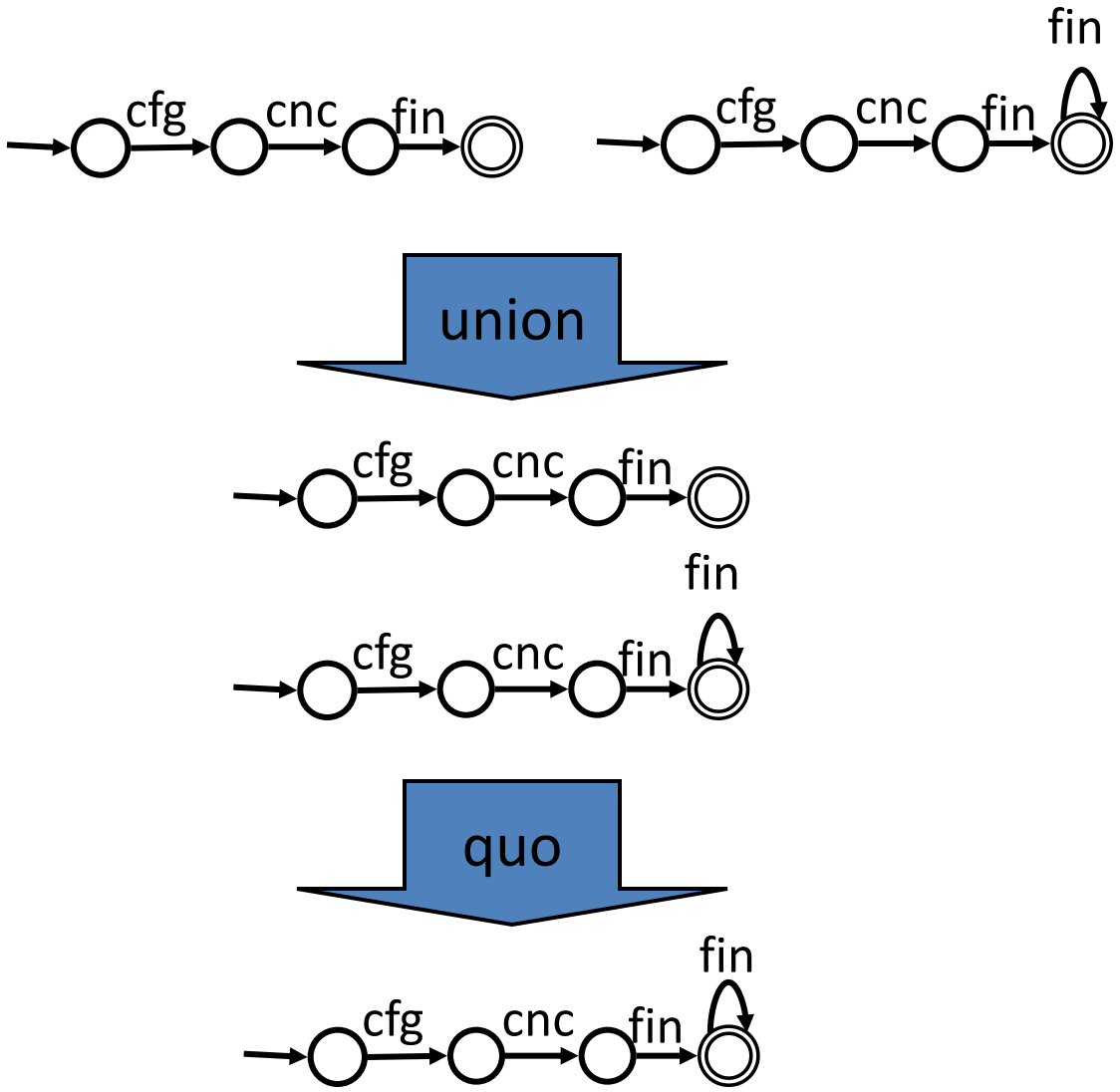
Abstract Semantics

- Initial abstract history
  - empty sequence automaton
- When an API method is invoked
  - history extended: append event and construct quotient

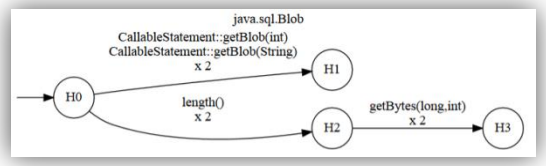


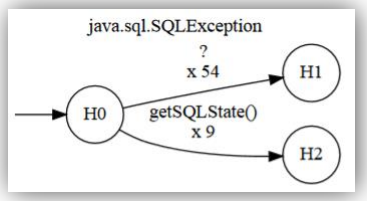
Join

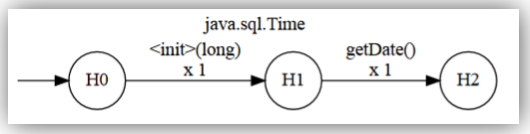
endof while

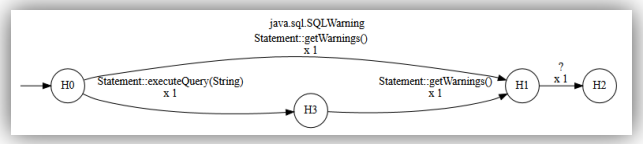


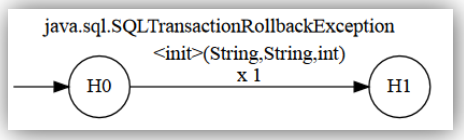
Consolidating the Index

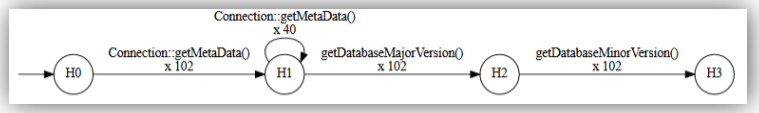




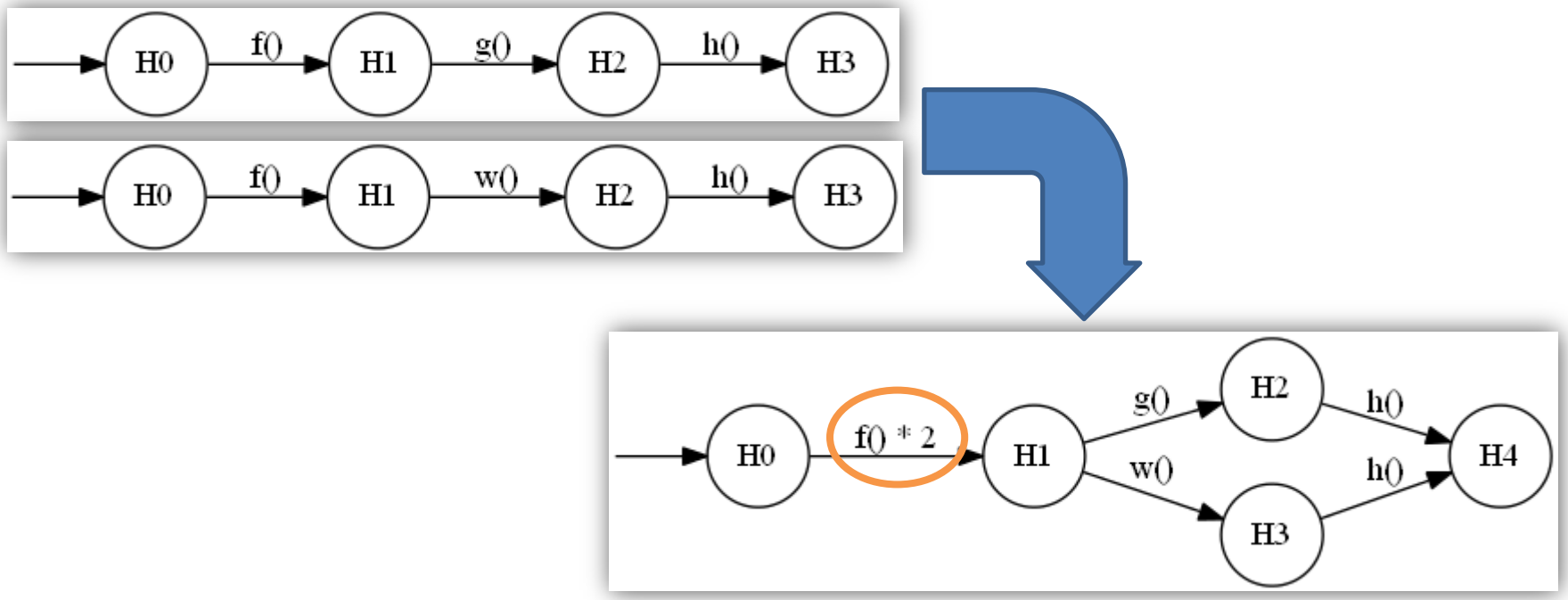




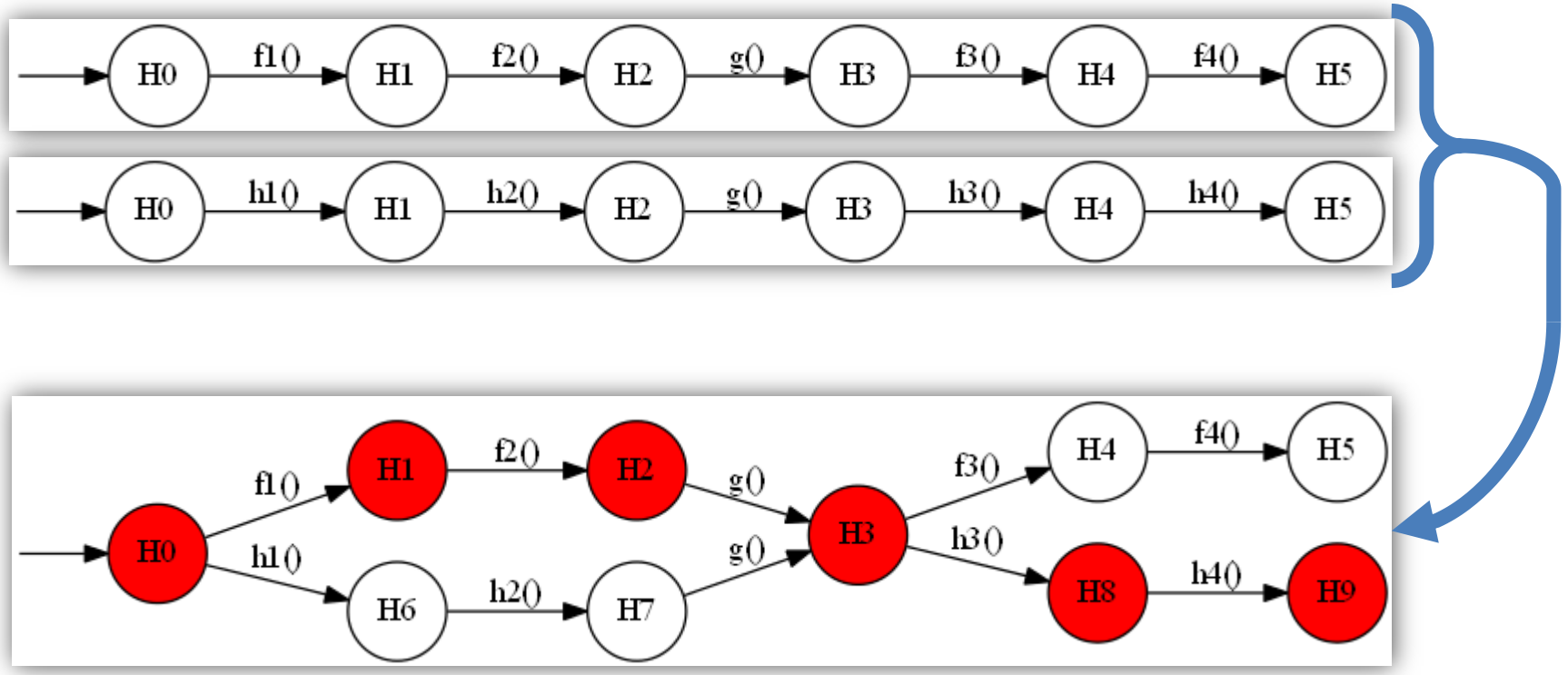




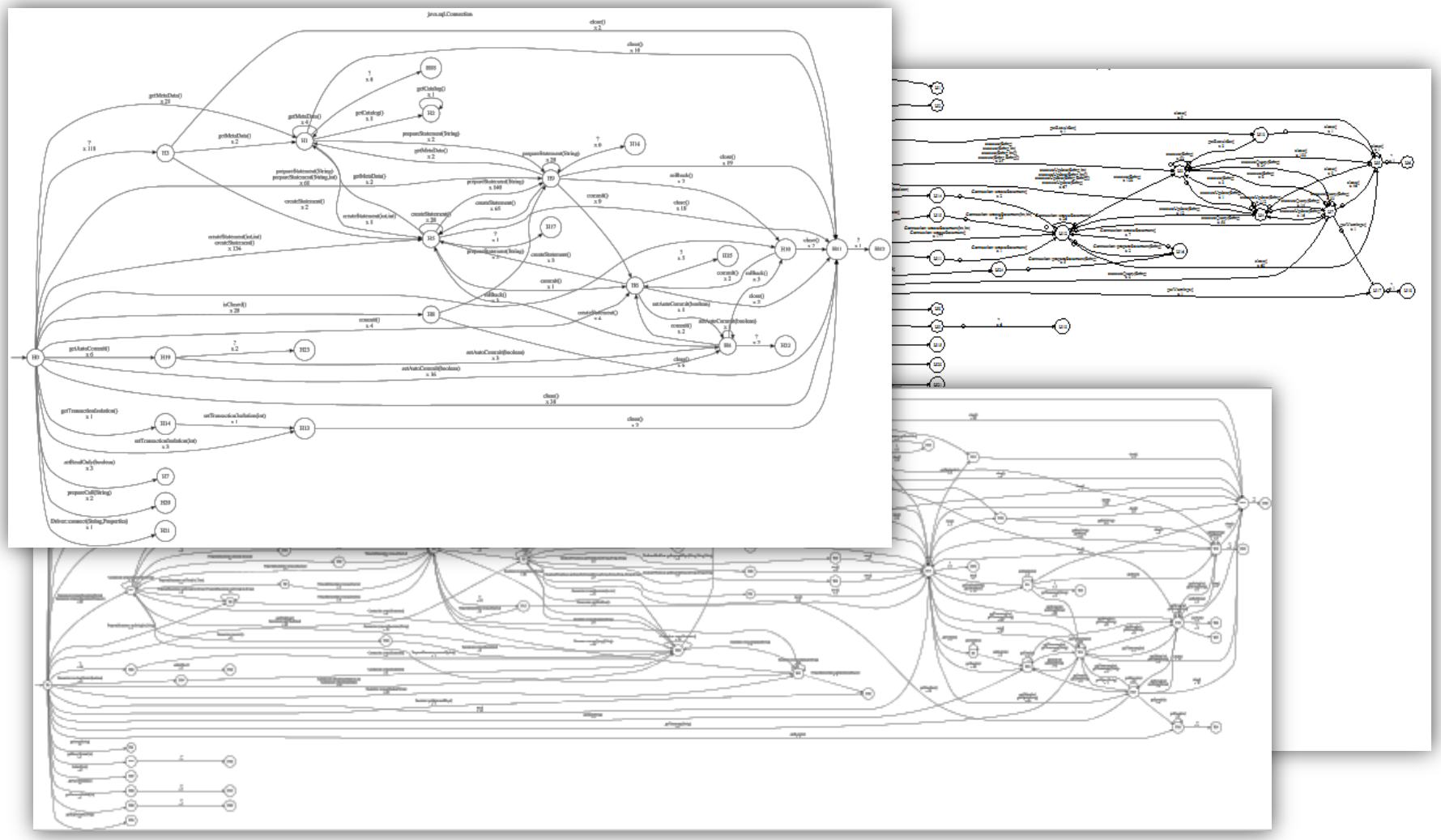
Merge Same Type Together



Merge All Samples of Same Type

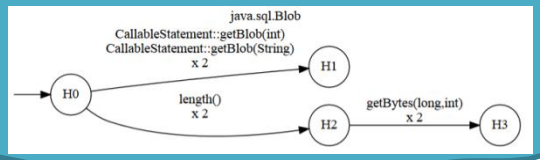


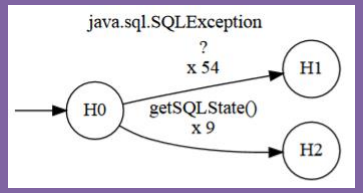
Merging all Samples of Same Type



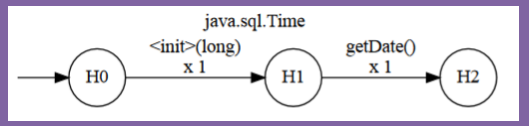
Merge by Use Case

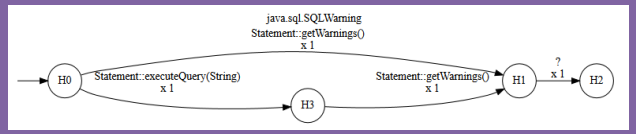
Use case 3



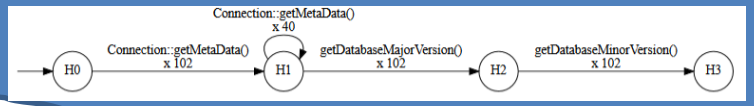


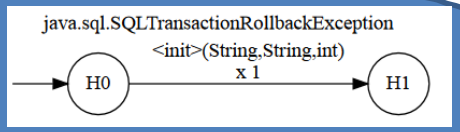
Use case 1





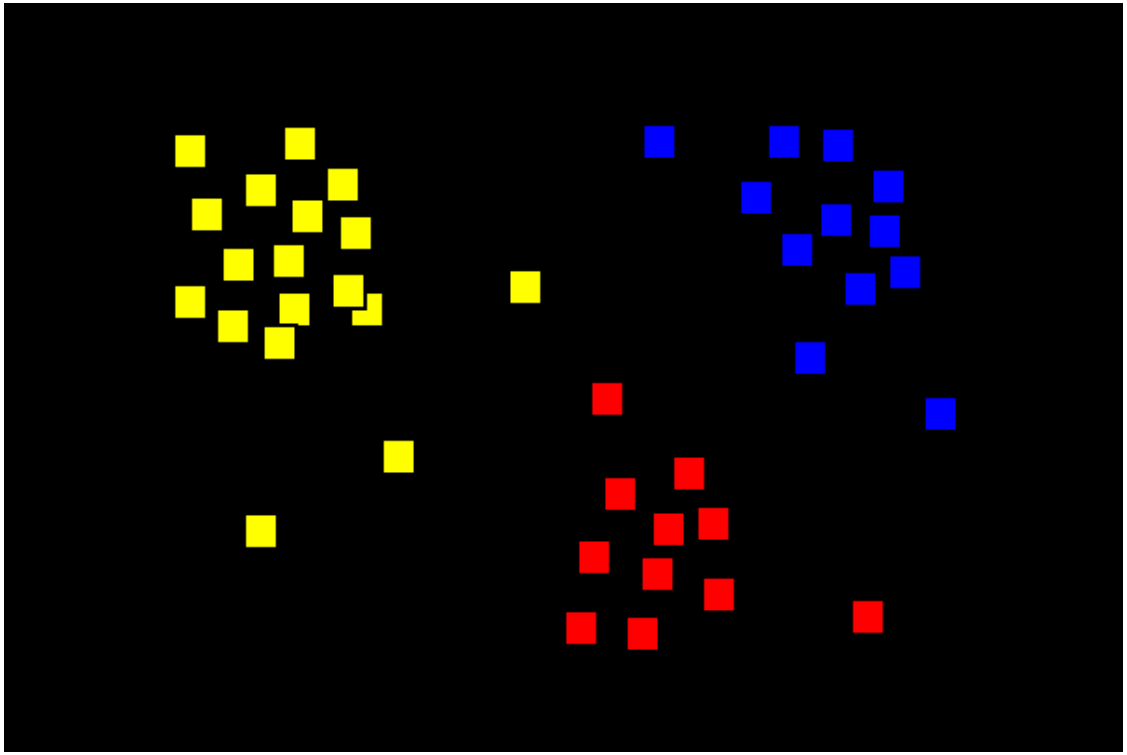
Use case 2





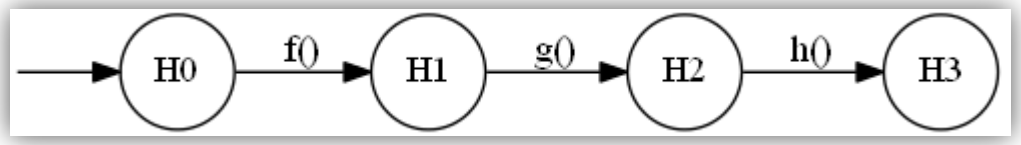
But how?

Clustering

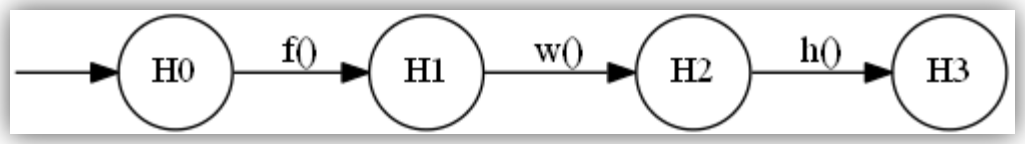


Example distance function

Different use-cases typically use different methods



f()	g()	w()	h()
1	1	0	1



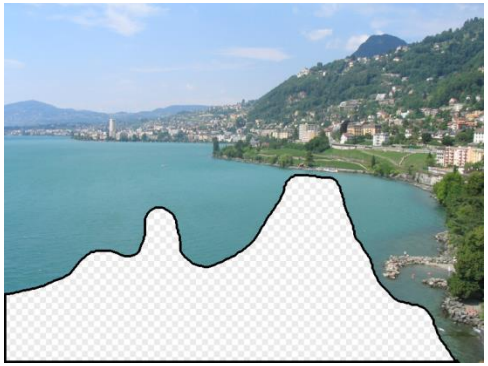
f()	g()	w()	h()
1	0	1	1

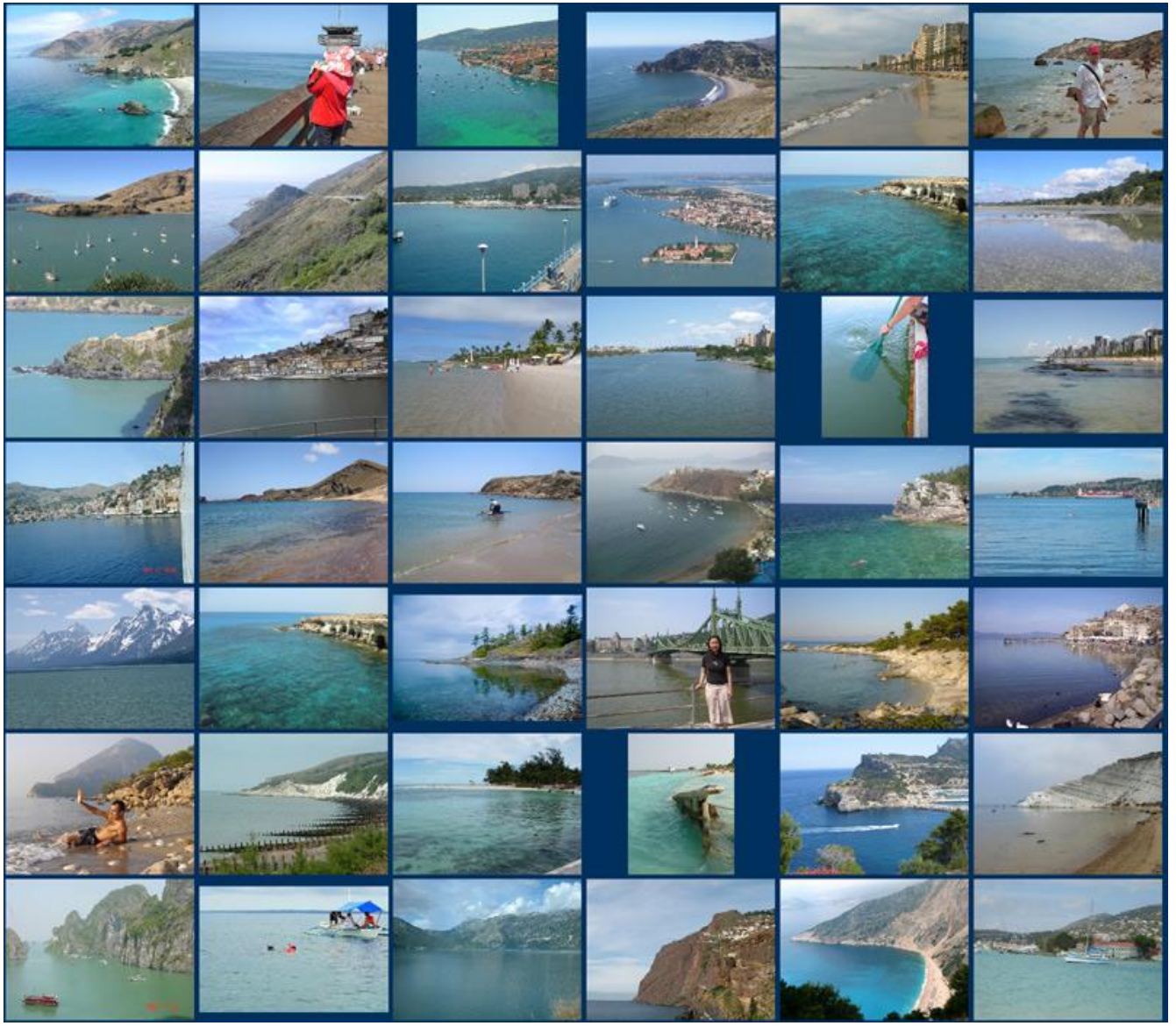
Distance = $\sqrt{(1-1)^2+(1-0)^2+(0-1)^2+(1-1)^2}$ = ~1.41

Clustering Results

	# samples	% samples	size
java.sql.Statement	396	18	40
Statement cluster 4	239	60	5
Statement cluster 6	77	19	7
Statement cluster 1	32	8	28
Statement cluster 5	27	7	5
Statement cluster 0	9	2	8
Statement cluster 3	6	2	6
Statement cluster 2	6	2	7
java.sql.Timestamp	62	3	22
Timestamp cluster 1	29	47	3
Timestamp cluster 2	13	21	9
Timestamp cluster 0	7	11	7
Timestamp cluster 4	7	11	10
Timestamp cluster 3	6	10	12

Query Matching

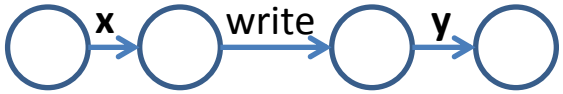




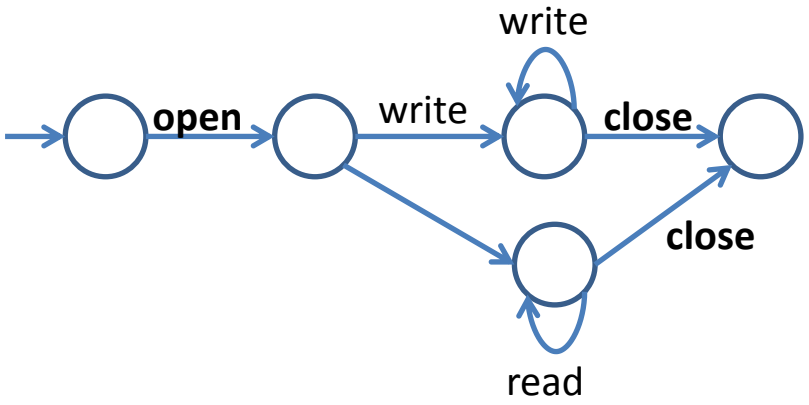
Hays and Efros, SIGGRAPH 2007

Query Matching

Query:



Index (partial):



Using unknown elimination we can get for example (assignment σ):

$$\sigma(var) = \begin{cases} open, & var = x \\ write^* \cdot close, & var = y \end{cases}$$

```
File f = ?
f.write(_);
f.?
```

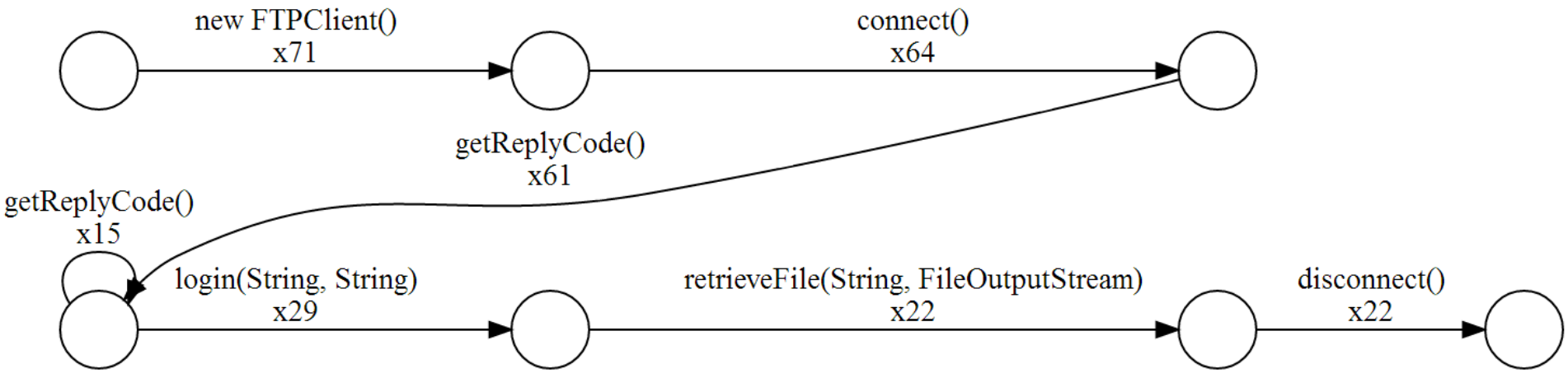
```
Public void print(String s) {
 File f = new File(“x”);
 String path = f.getAbsolutePath();
 f.write(s);
 f.close();
 record(path);
}
```

```
Public void print(String s) {
 File f = new File(“x”);
 if (s != null)
   f.write(s);
 else
   f.write(“error”);
 f.close();
}
```

⋮

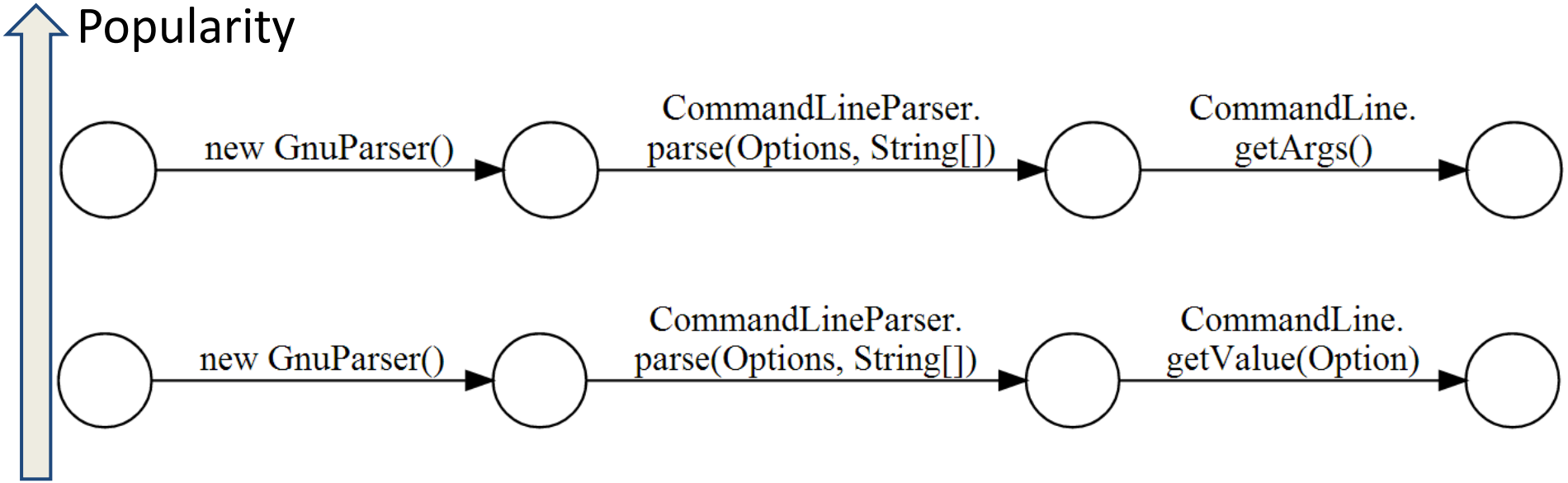
Example: Complete Sequence

```
FTPClient c = ?;
c.login("username", "password");
c.?;
c.retrieveFile("file path", out);
```



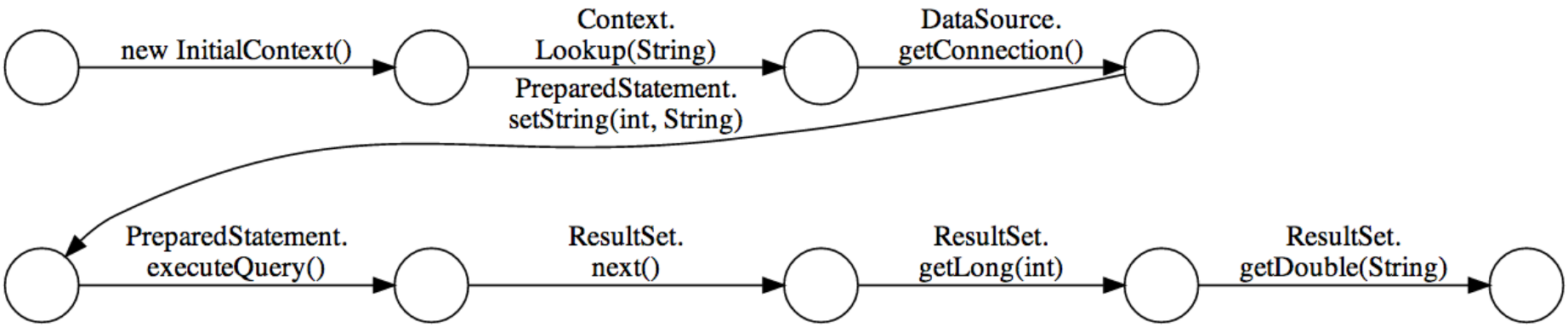
Example: Query Over Multiple Types

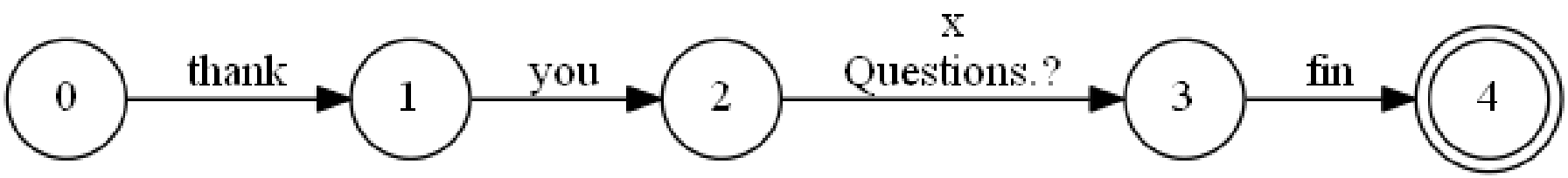
```
GnuParser p = new GnuParser();
p.?;
String s = p.?;
```



Example Result: Long Query

```
InitialContext ic =
new InitialContext();
ic.?;
PreparedStatement ps = ic.?;
ResultSet rs = ps.executeQuery();
rs.?;
double d = rs.getDouble("column");
```

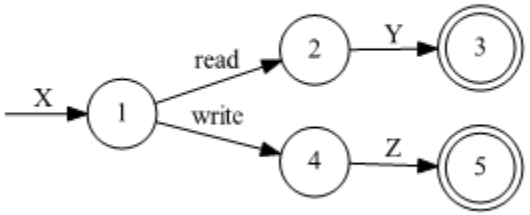



0
thank
1
you
2
x
Questions.?
3
fin
4

References

- [SAS13] Symbolic Automata for Static Specification Mining
- [OOPSLA12] Typestate-based semantic code search over partial programs
- [ISSTA07] Static Specification Mining using Automata Based Abstractions
- [ISSTA06] Effective Typestate Verification in the presence of Aliasing
- [Bodik-PLDI05] Jungloid mining: helping to navigate the API jungle
- [Kuncak-CAV11] Interactive Synthesis of Code Snippets
- [Perelman-PLDI12] Type-directed completion of partial expressions

context



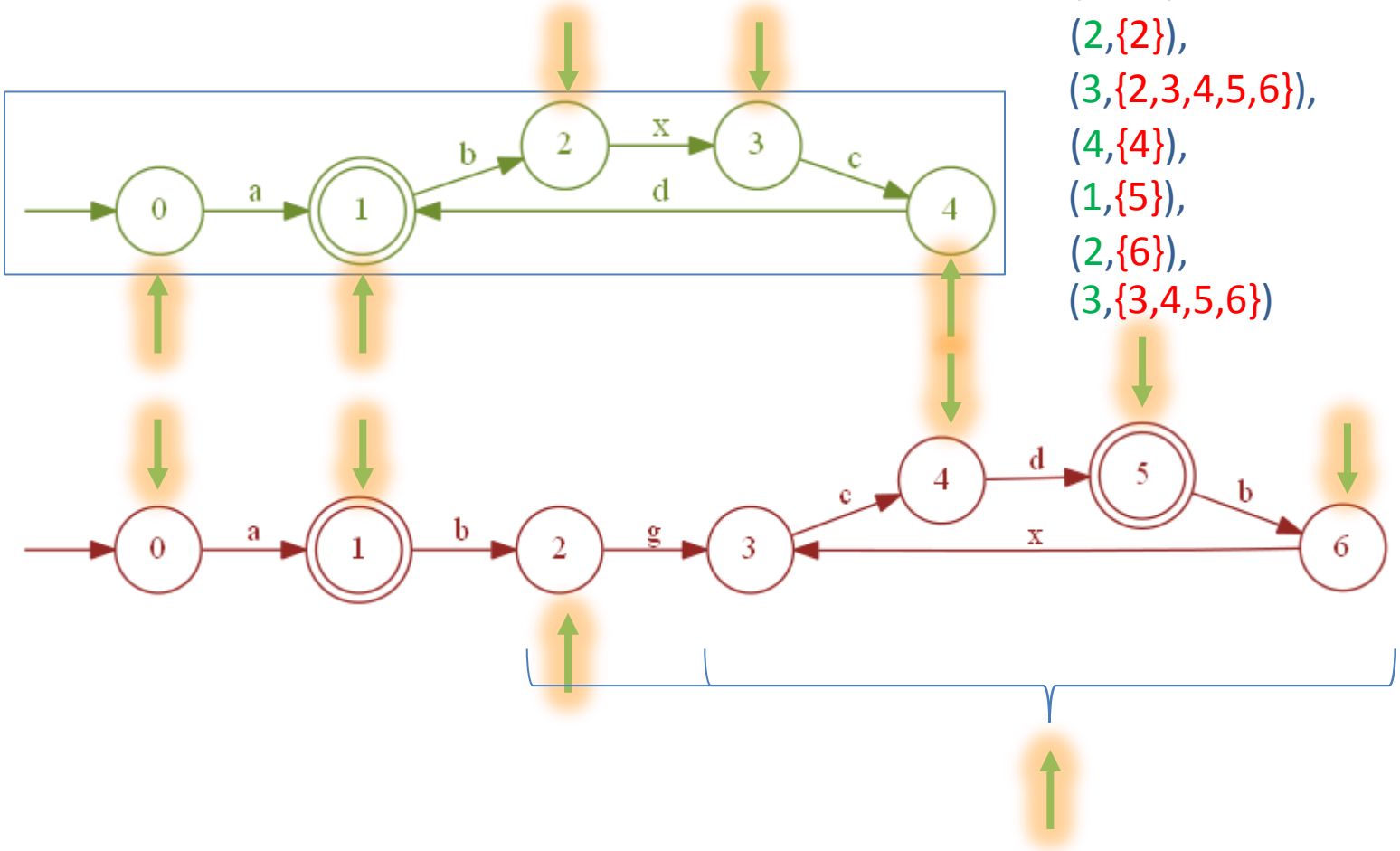
(a)

$(*,X,\texttt{read}) \mapsto \texttt{open,canRead}$
$(*,X,\texttt{write}) \mapsto \texttt{open}$
$\texttt{Y} \mapsto \texttt{close}$
$\texttt{Z} \mapsto \texttt{close}$

(b)

(a) Symbolic automaton representing the query for the behavior around read and write methods and (b) the assignment with contexts to its symbolic transitions which answers the query.

Simulation Example



Simulation: (0,{0}),
(1,{1}),
(2,{2}),
(3,{2,3,4,5,6}),
(4,{4}),
(1,{5}),
(2,{6}),
(3,{3,4,5,6})