Nu: a Dynamic Aspect-Oriented Intermediate Language Model and Virtual Machine for Flexible Runtime Adaptation

Robert Dyer and Hridesh Rajan

Department of Computer Science
Iowa State University
{rdyer,hridesh}@cs.iastate.edu

7th International Conference on
Aspect-Oriented Software Development

- Motivation: Supporting dynamic aspect-oriented constructs
- Approach: Nu
- Evaluation: Expressiveness **and** performance
- Technical Contributions:
  - Flexible, dynamic AO intermediate language model
  - Implementation in industrial strength VM (Sun Hotspot)
  - Dedicated AO caching mechanism

Need For a Dynamic IL Model

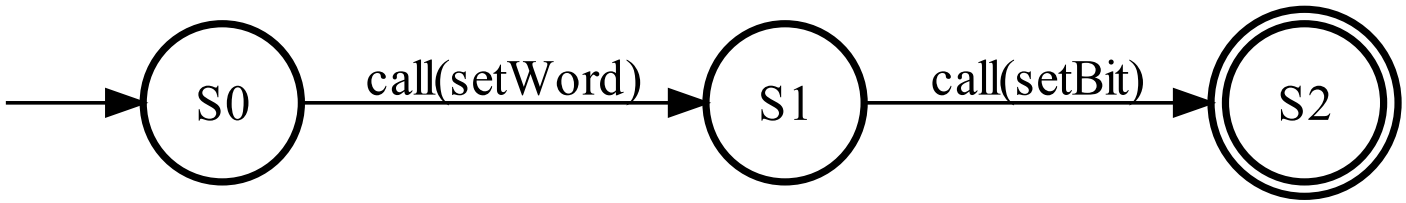
- Currently: Dynamic, high-level AO constructs $\longrightarrow$ low-level OO representation
- AO compilers need “building blocks”!
- Perhaps an example...

History-based Pointcuts

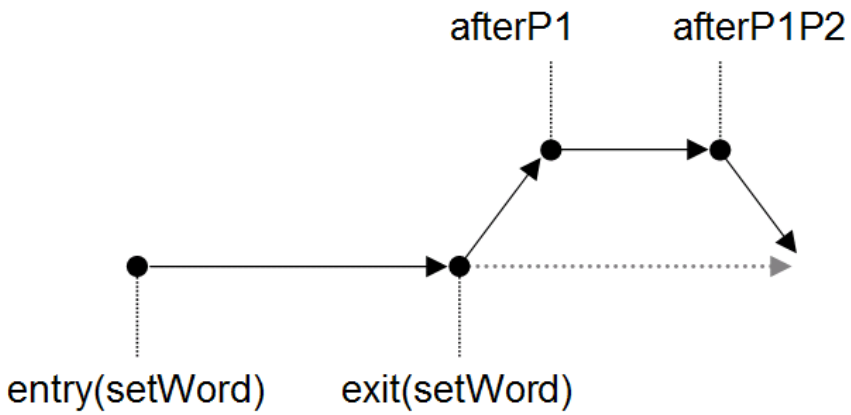
- History-based pointcuts [Douence, Fradet, and Südholt]
- Temporal constructs in AspectJ [Stolz and Bodden]

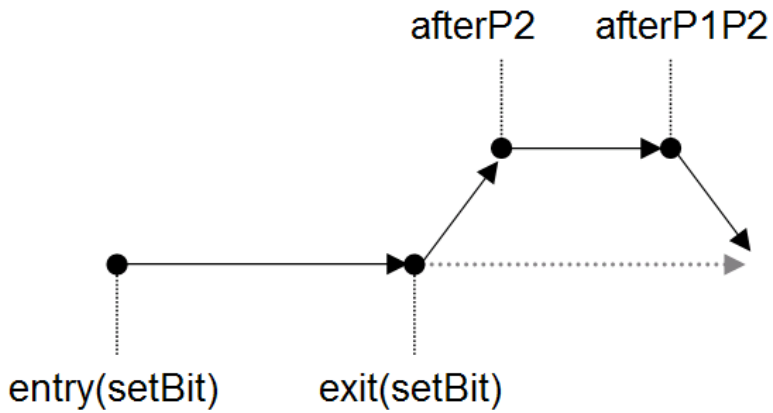
History-based Pointcuts

- History-based pointcuts [Douence, Fradet, and Südholt]
- Temporal constructs in AspectJ [Stolz and Bodden]

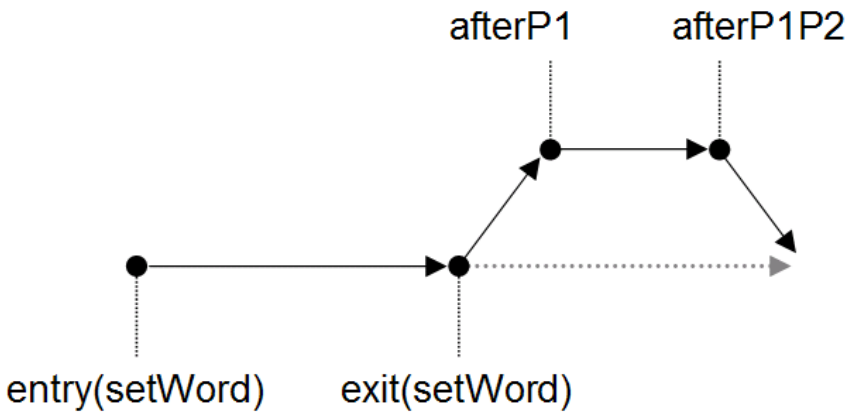


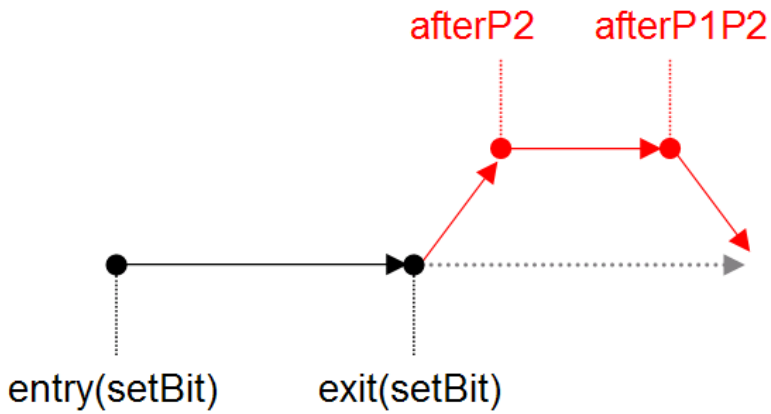
History-based Pointcuts - Static Translation



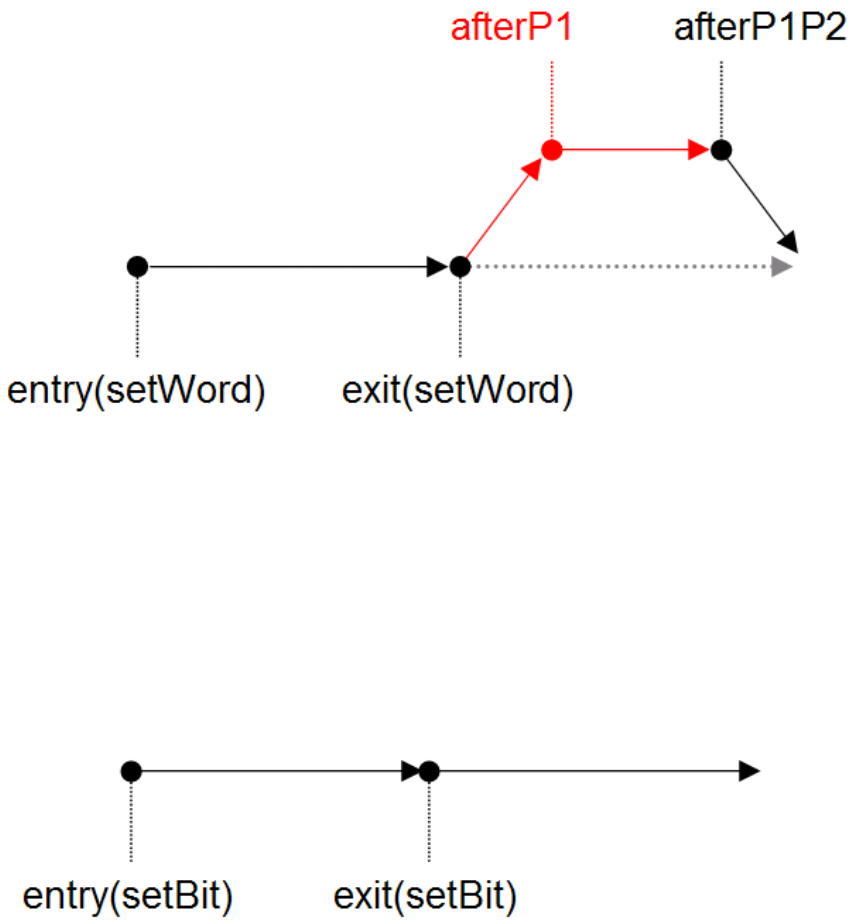


History-based Pointcuts - Static Translation

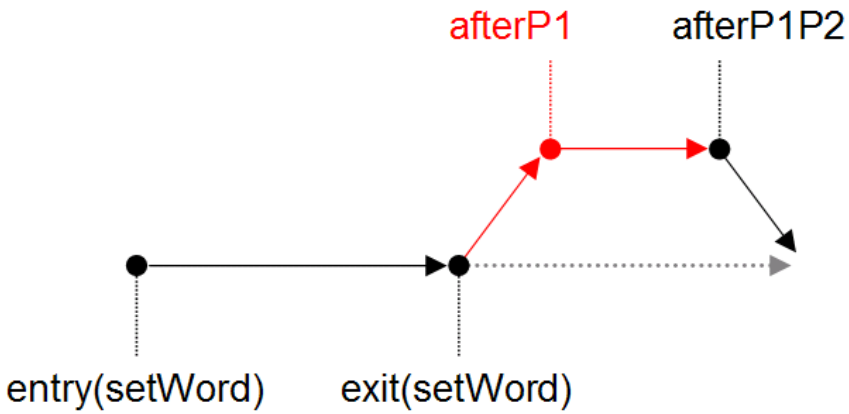


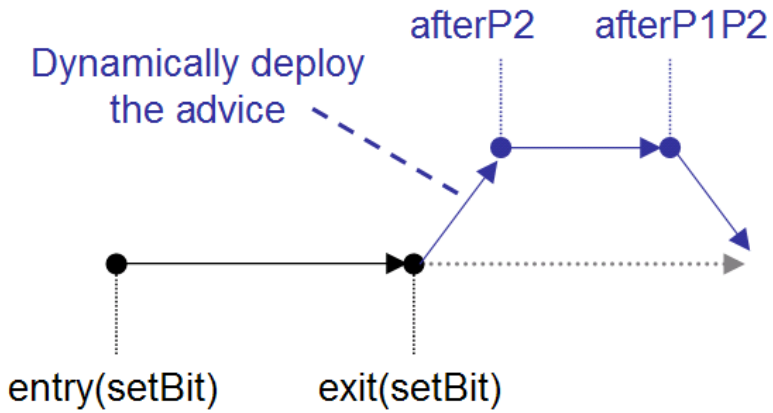


History-based Pointcuts - Dynamic Support

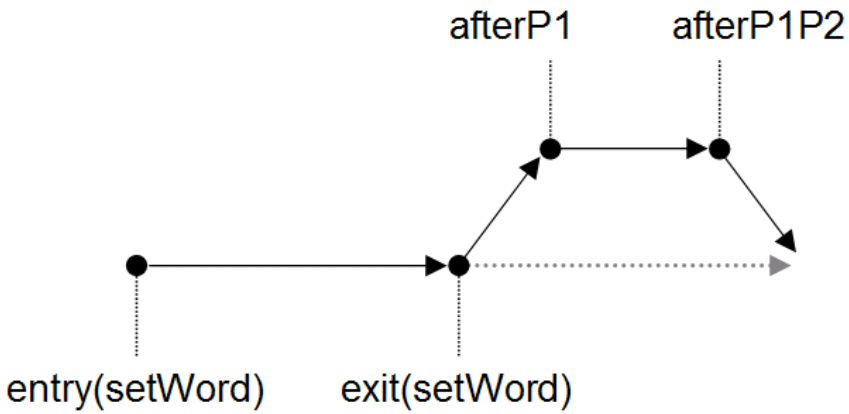


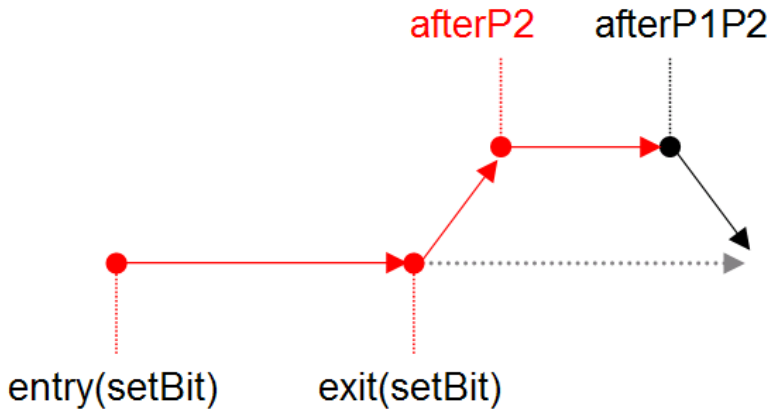
History-based Pointcuts - Dynamic Support



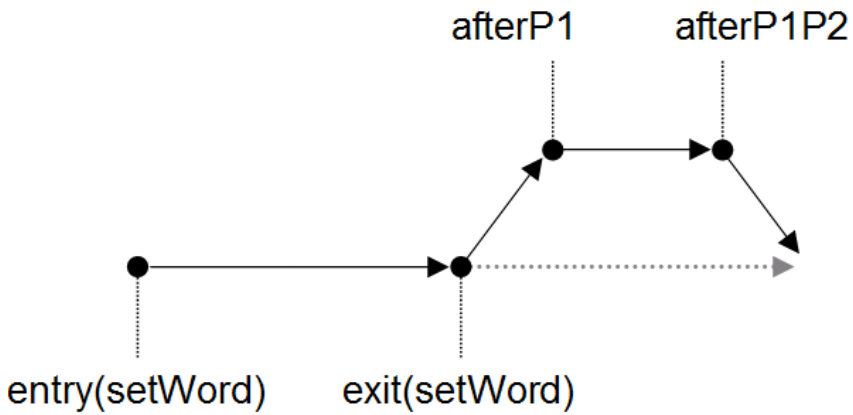


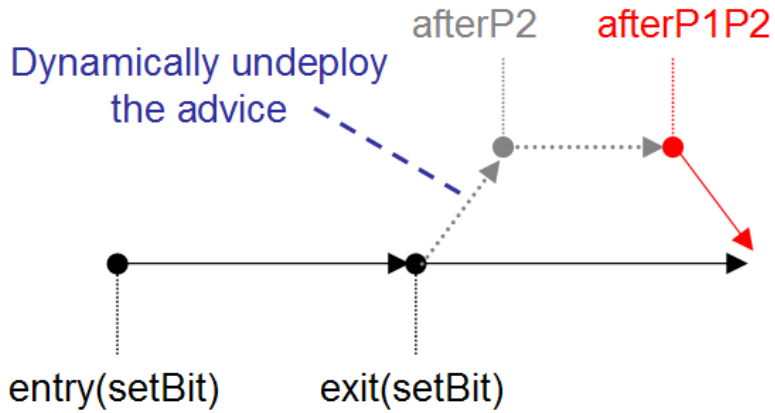
History-based Pointcuts - Dynamic Support





History-based Pointcuts - Dynamic Support





Need For a Dynamic IL Model

- Inadequate support for dynamic use cases in current ILs
  - Dynamic deployment,
  - Dynamic adaptation,
  - Policy changes, etc
- Dynamically adapting set of advised join points
  - Morphing aspects [Hanenberg et al.]
  - Open Aspects [Hirschfeld and Hanenberg]

Overview of the Nu Model

- JPM: Point-in-time model [Masuhara et al.]
- New primitives: *bind* and *remove*
- Advice as delegate methods
- Library of patterns
  - First-class, immutable objects

Two New Primitives: Bind and Remove

	Stack Transition	Exceptions
bind	..., `Pattern`, `Delegate` ⟶ ..., `BindHandle`	`NullPointerEx`
remove	..., `BindHandle` ⟶ ...	`IllegalArgumentEx`

This talk uses source representation for ease.

Bind and Remove Example

```
public class AuthLogger {



}
```

Bind and Remove Example

```
public class AuthLogger {
  protected static Pattern p;
  protected static Delegate d;


  static {
    p = new Execution(new Method("*.login"));
    d = new Delegate(AuthLogger.class, "log");
  }



  public static void log() { // record the time of login }
}
```

Bind and Remove Example

```
public class AuthLogger {
  protected static Pattern p;
  protected static Delegate d;


  static {
    p = new Execution(new Method("*.login"));
    d = new Delegate(AuthLogger.class, "log");
  }



  public static void log() { // record the time of login }
}
```

Bind and Remove Example

```
public class AuthLogger {
  protected static Pattern p;
  protected static Delegate d;


  static {
    p = new Execution(new Method("*.login"));
    d = new Delegate(AuthLogger.class, "log");
  }



  public static void log() { // record the time of login }
}
```

Bind and Remove Example

```
public class AuthLogger {
  protected static Pattern p;
  protected static Delegate d;
  protected static BindHandle id = null;

  static {
    p = new Execution(new Method("*.login"));
    d = new Delegate(AuthLogger.class, "log");
  }

  public static void enable() { id = bind(p, d); }


  public static void log() { // record the time of login }
}
```

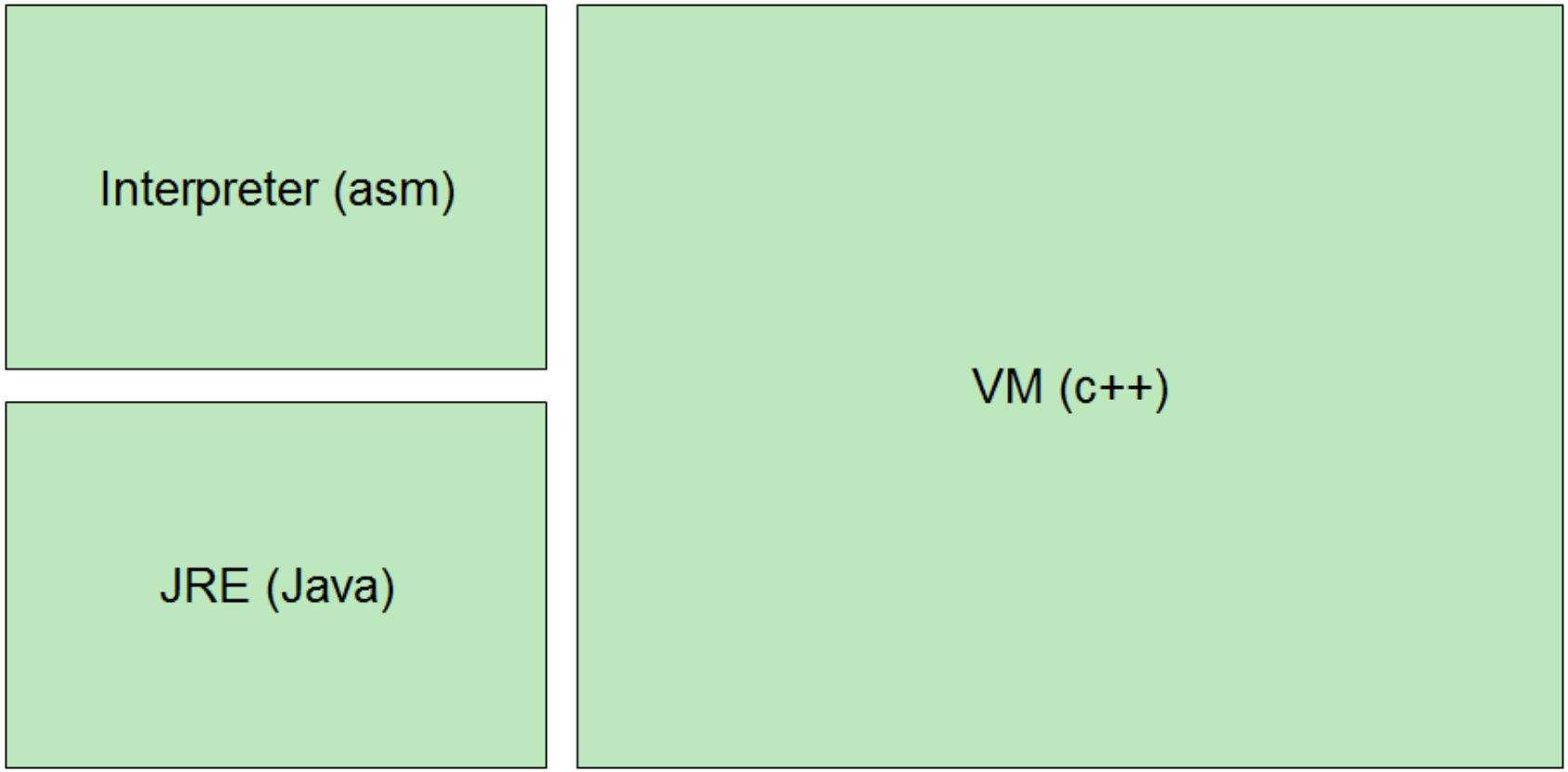
Bind and Remove Example

```
public class AuthLogger {
  protected static Pattern p;
  protected static Delegate d;
  protected static BindHandle id = null;

  static {
    p = new Execution(new Method("*.login"));
    d = new Delegate(AuthLogger.class, "log");
  }

  public static void enable() { id = bind(p, d); }
  public static void disable() { remove(id); }

  public static void log() { // record the time of login }
}
```

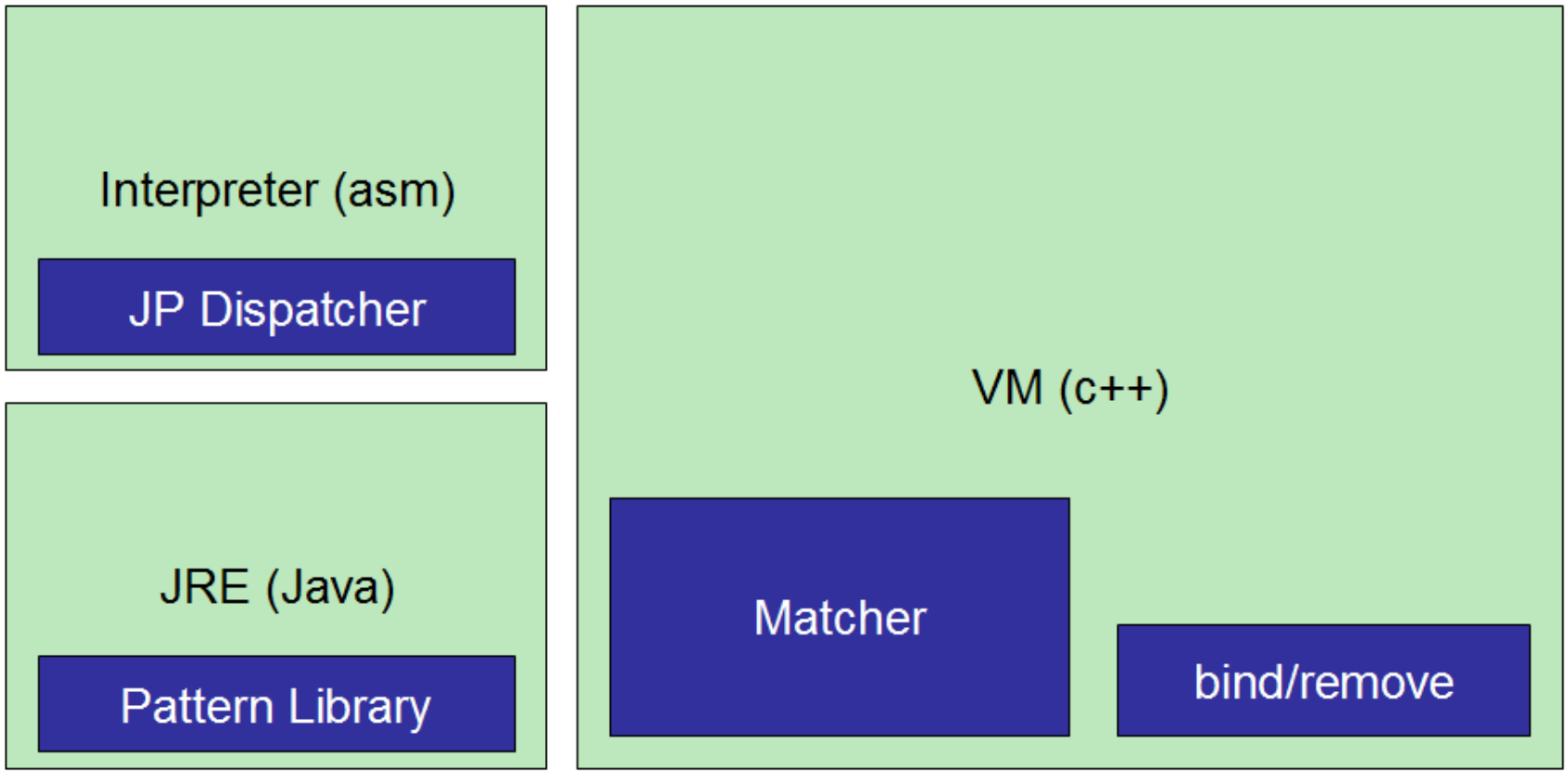
Implementation Overview

- Built on top of Sun Hotspot VM
- Adds code to the interpreter for join point dispatch
- bind and remove implemented as native methods

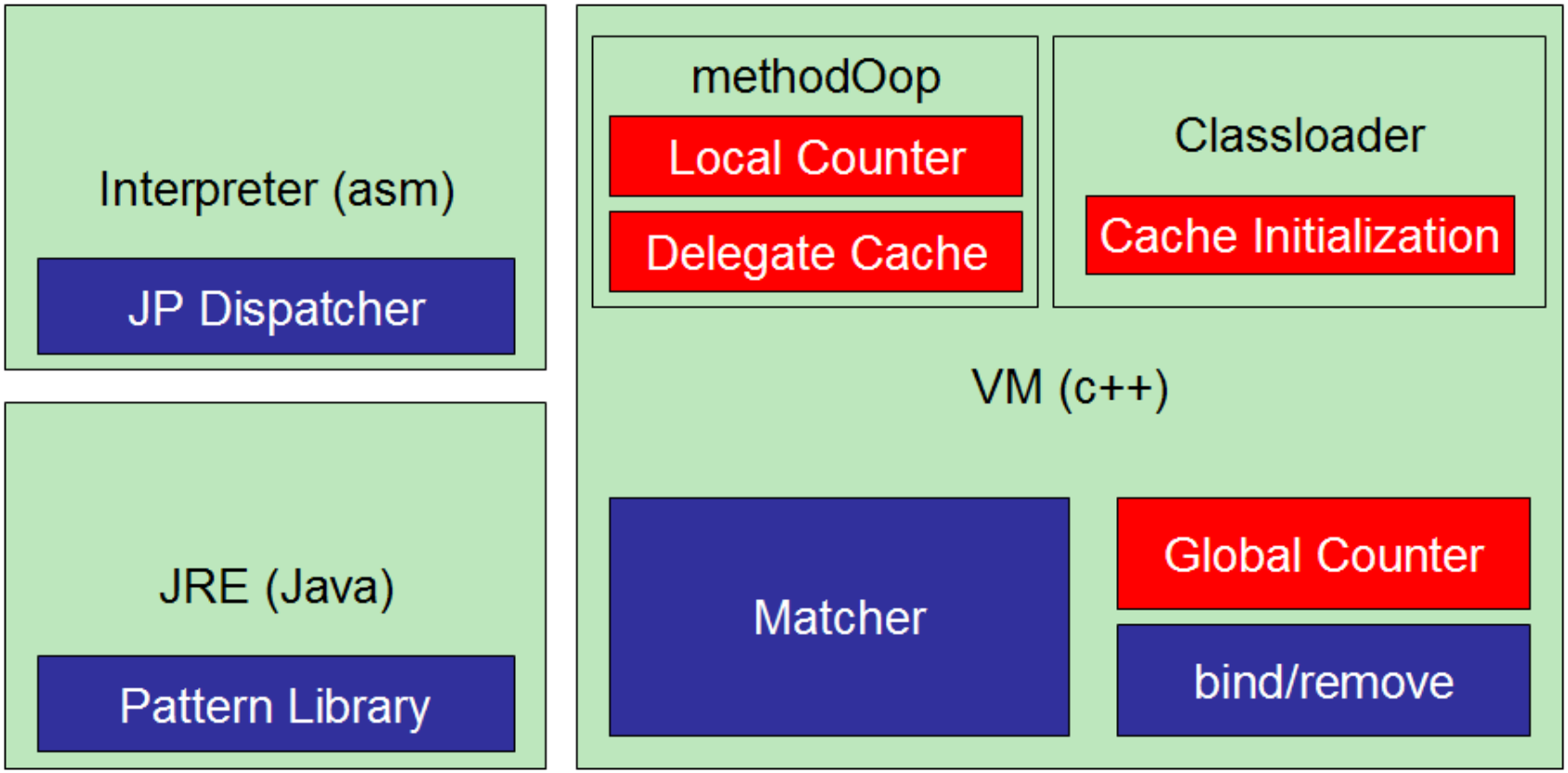
Implementation Overview



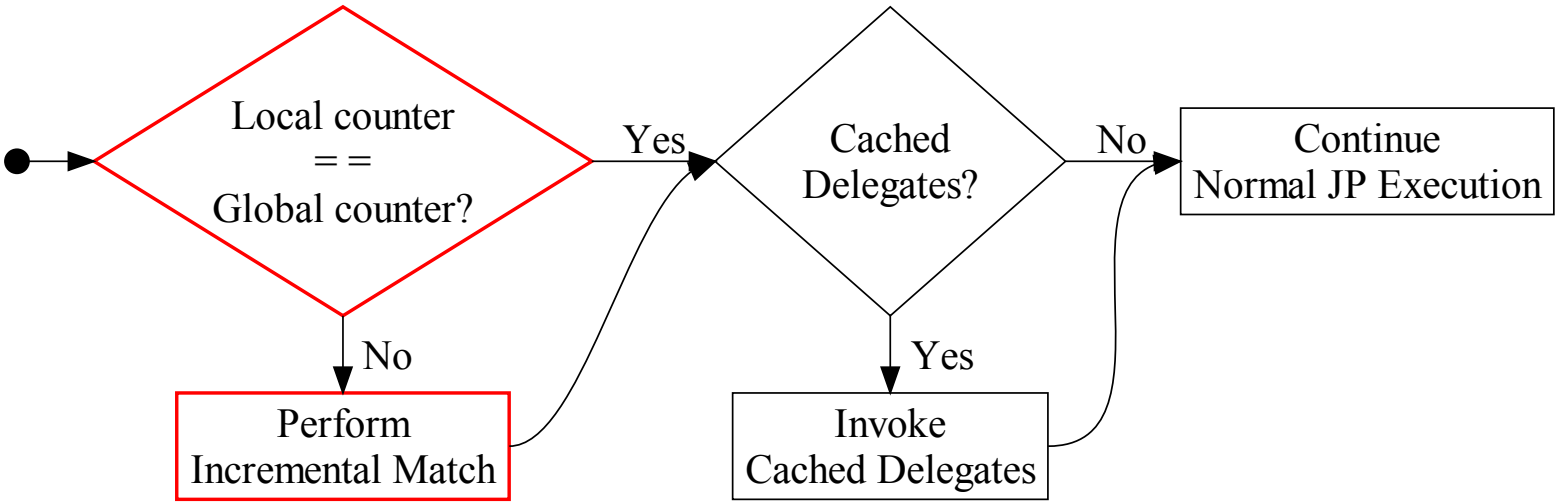
Implementation Overview



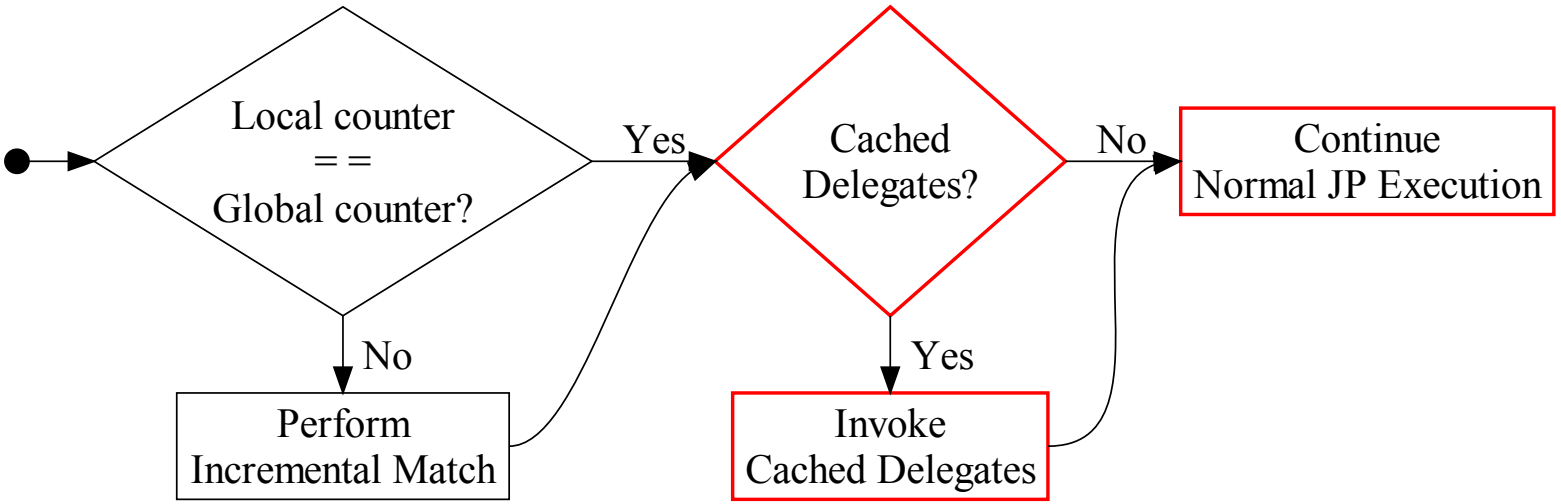
Novel AO Caching Mechanism



Caching Mechanism



Caching Mechanism



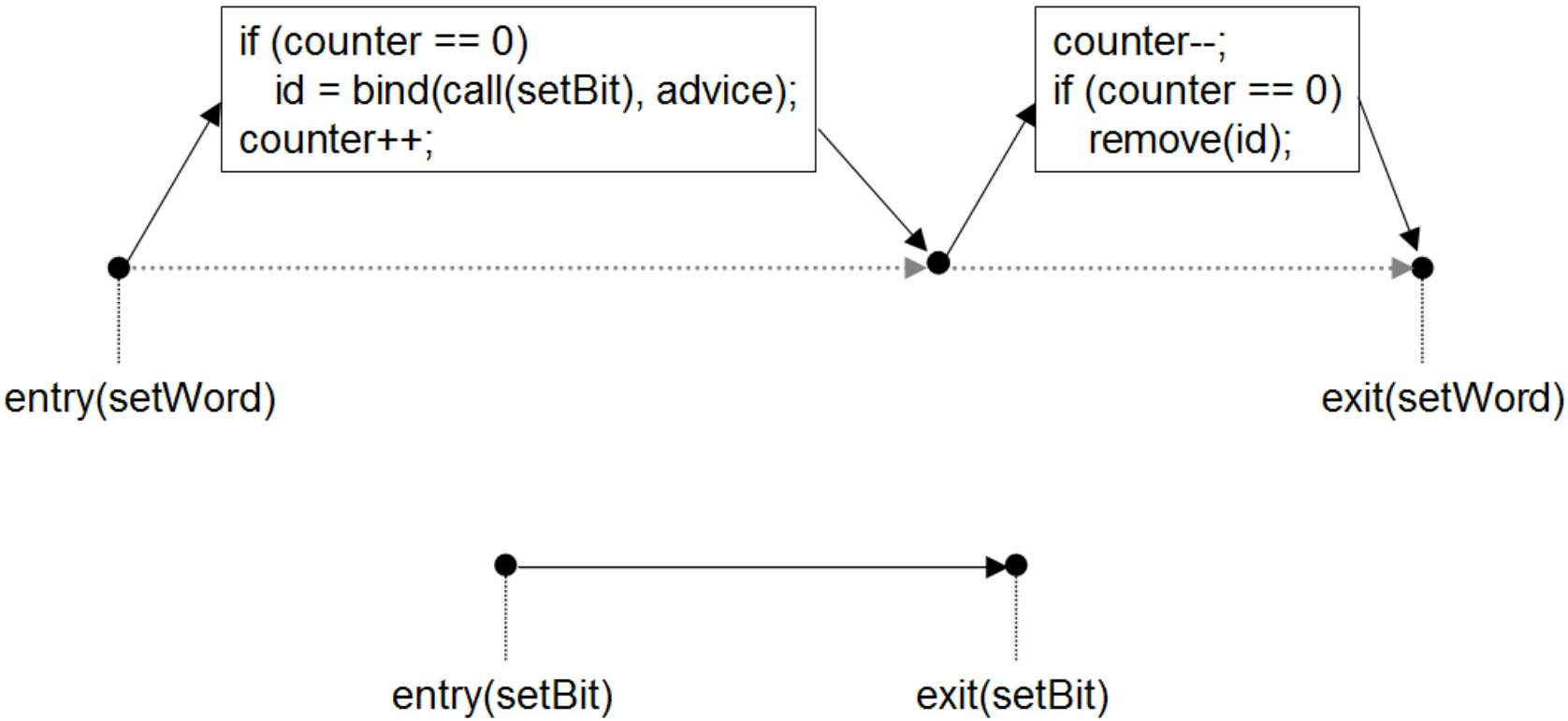
Expressing Constructs

- Supports multiple language constructs:
  - AspectJ aspects, pointcuts and advice
  - cflow, cflowbelow
  - History-based pointcuts
  - CaesarJ's deploy

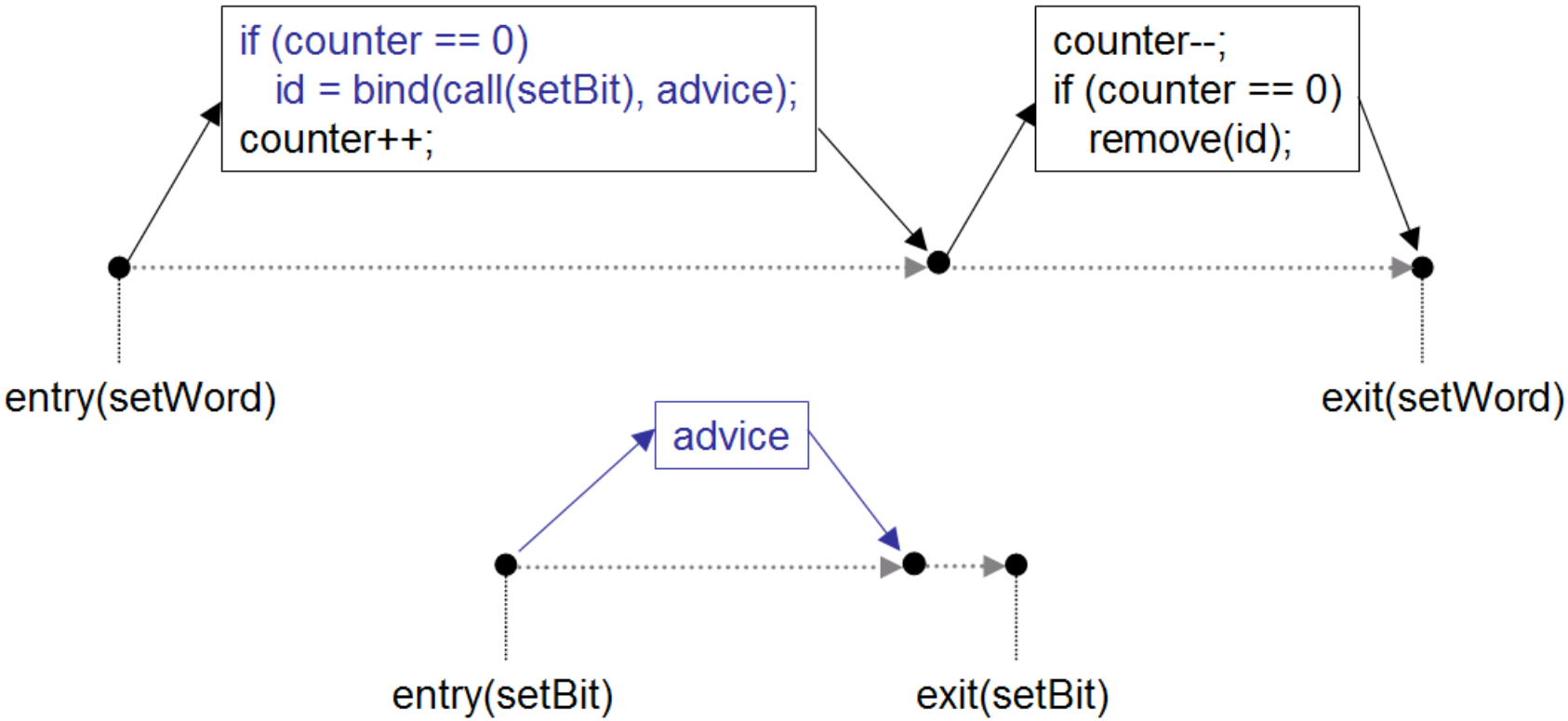
Expressing Constructs

- Supports multiple language constructs:
  - AspectJ aspects, pointcuts and advice
  - cflow, cflowbelow
  - History-based pointcuts
  - CaesarJ's deploy

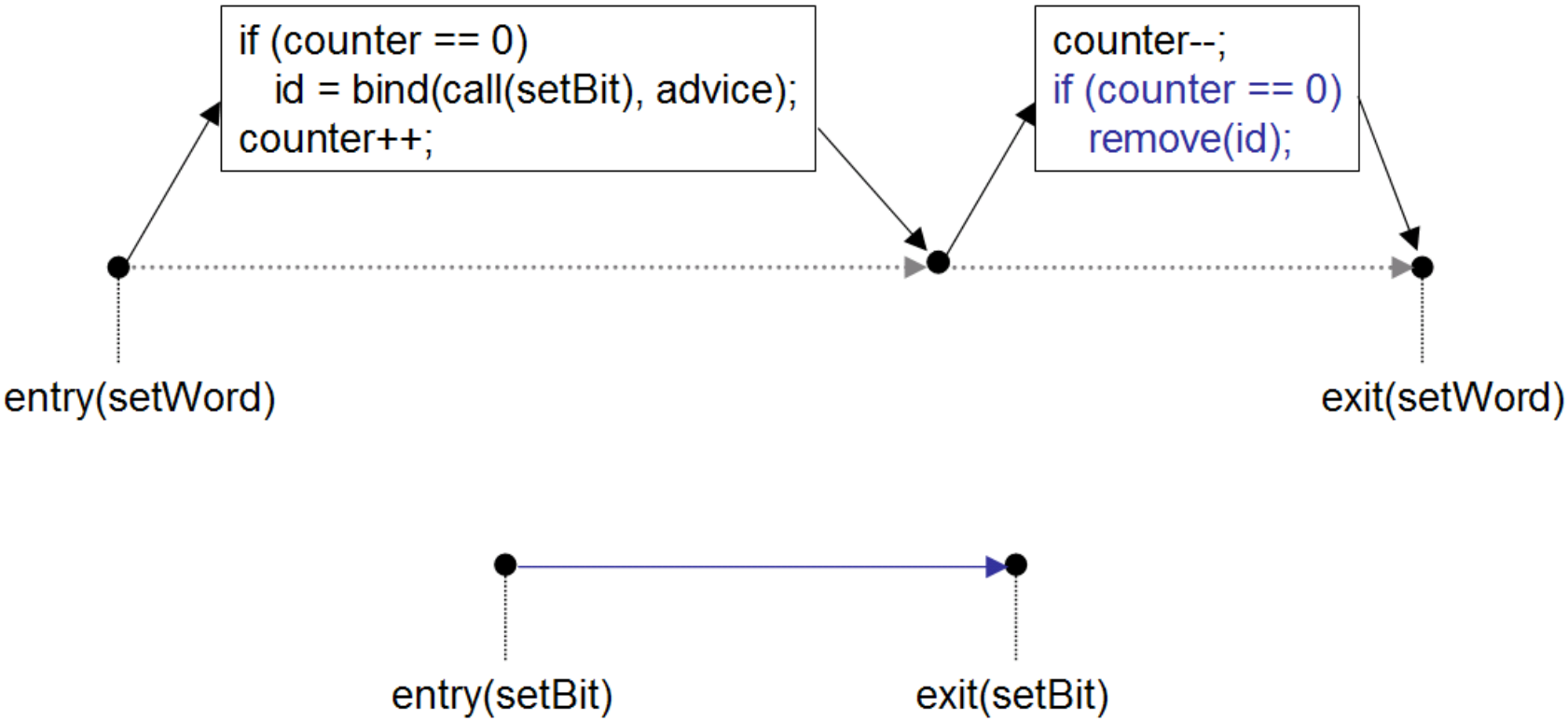
cflow(execution(setWord)) && execution(setBit)



cflow(execution(setWord)) && execution(setBit)

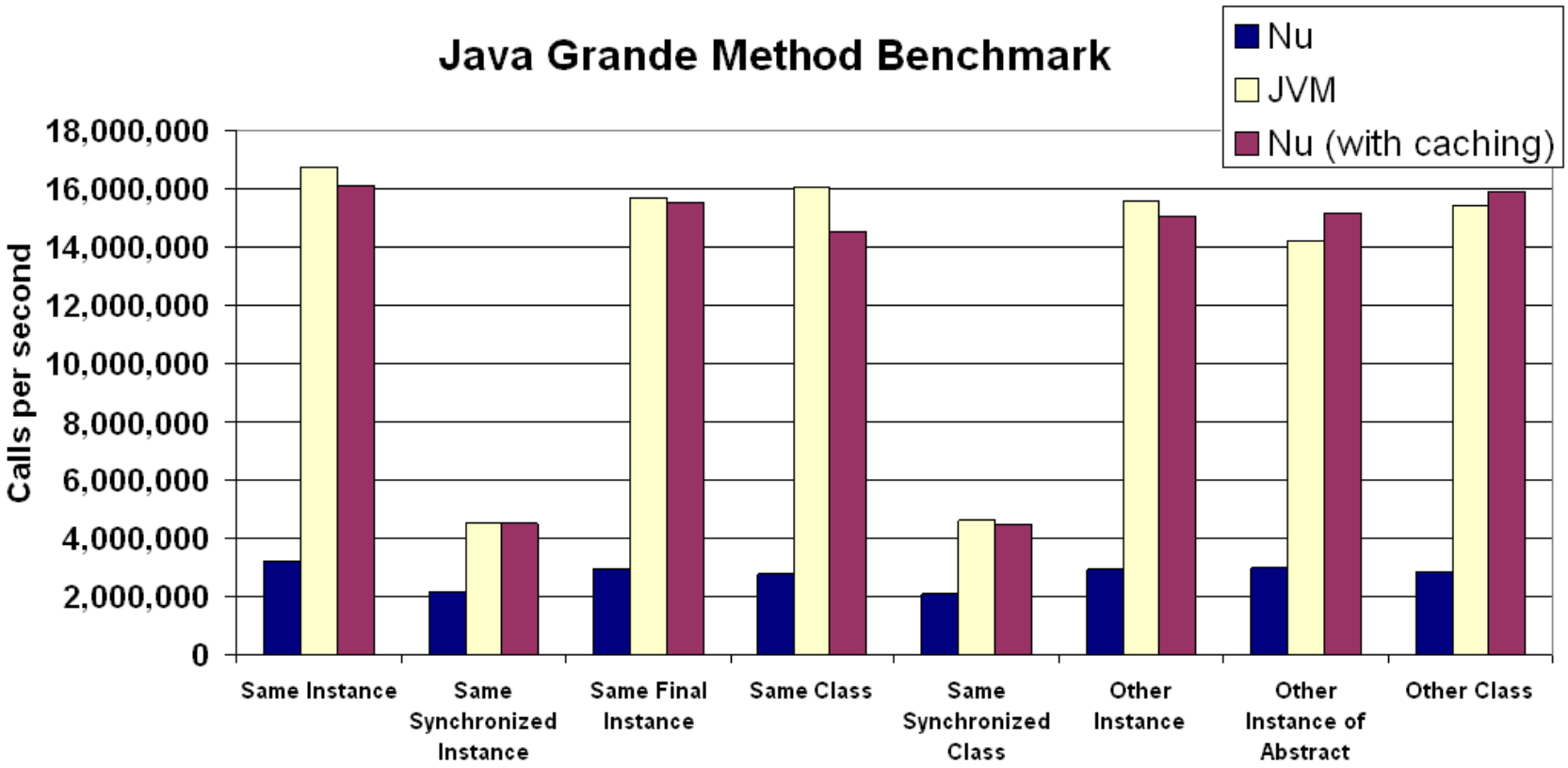


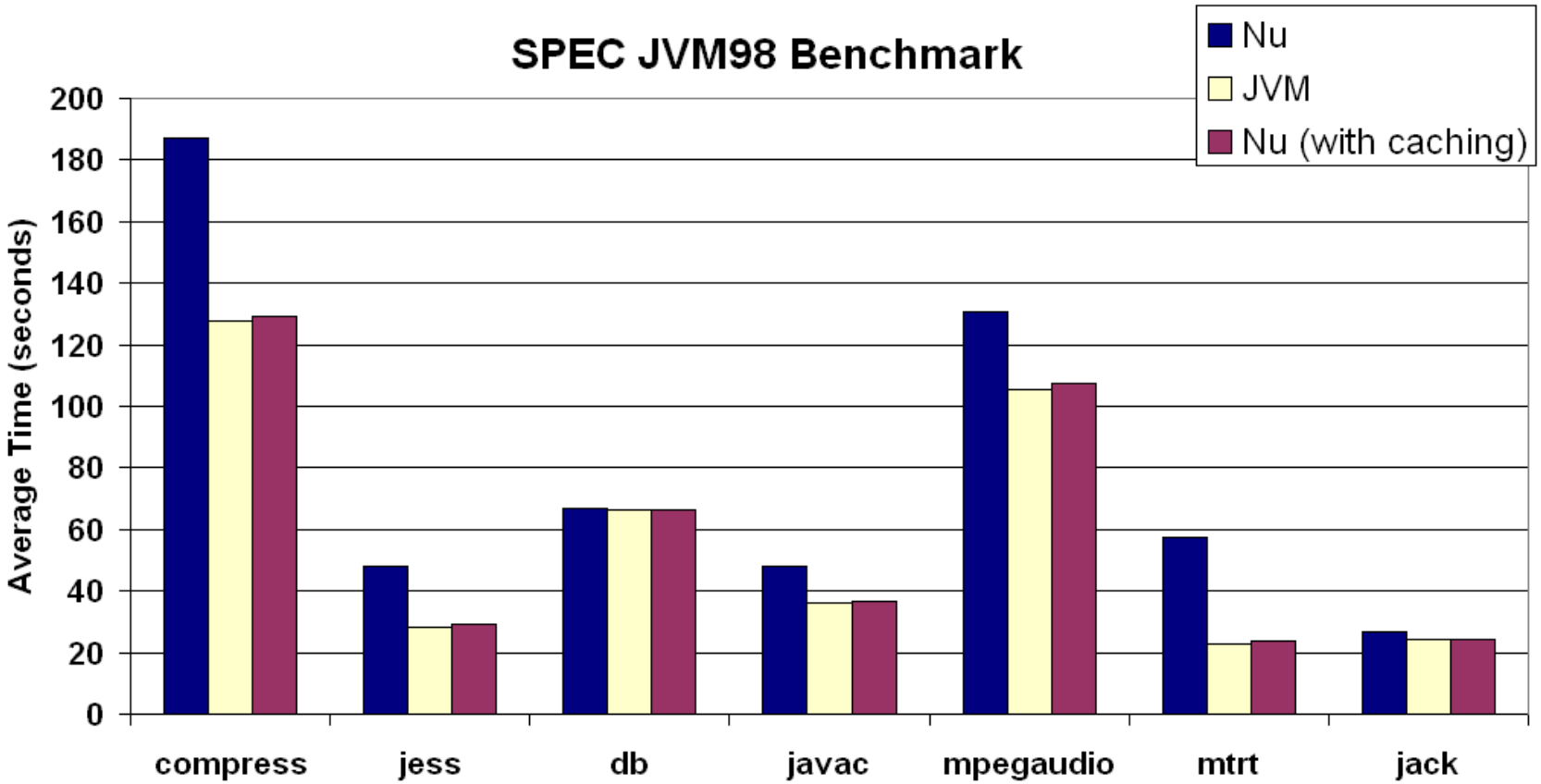
cflow(execution(setWord)) && execution(setBit)

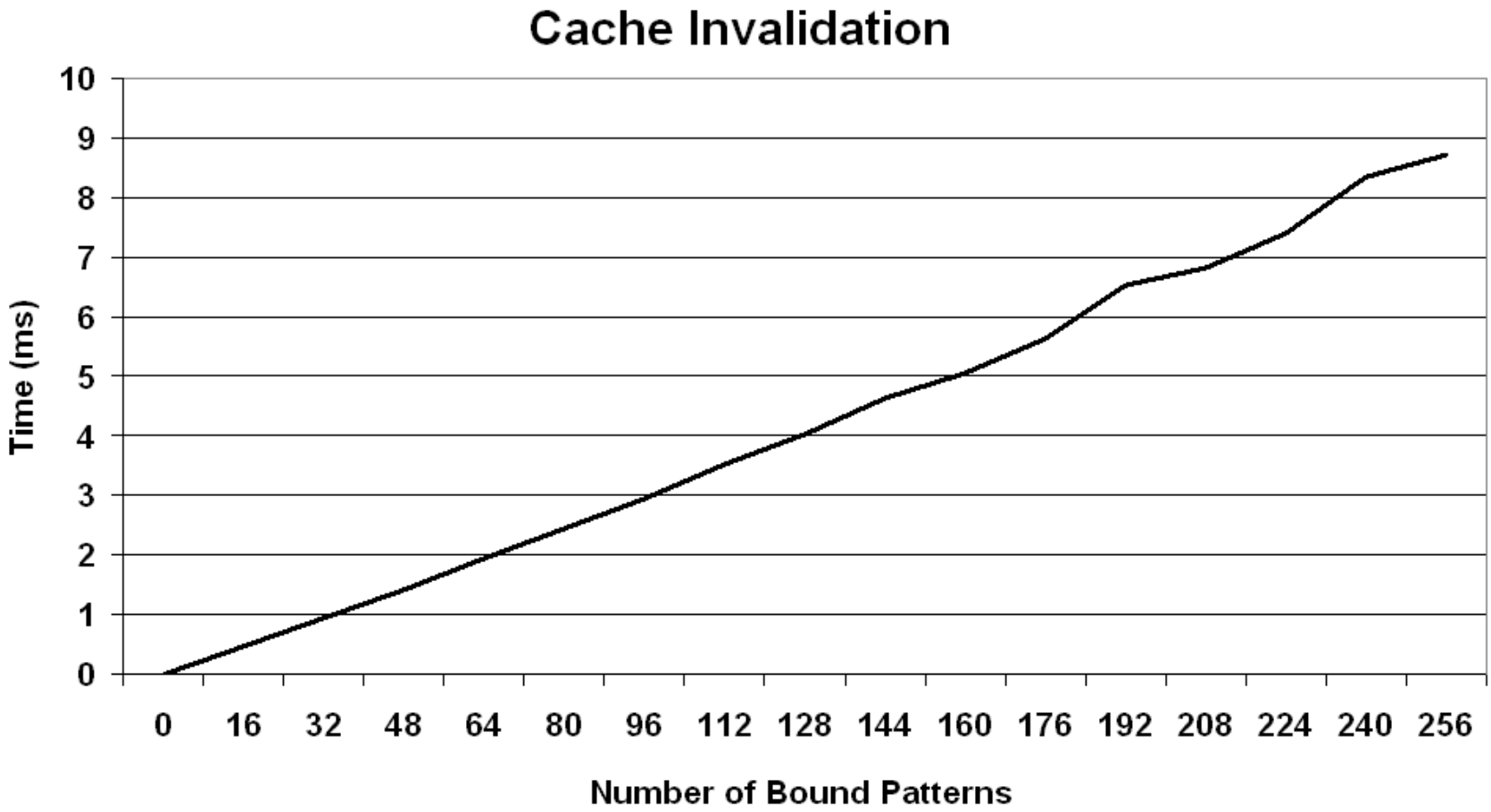


Performance Evaluation Overview

- Java Grande, SPEC JVM98, and custom micro-benchmarks
- Unmodified VM, Nu VM (caching), Nu VM (no caching)
- Only about 1.5% overhead


Java Grande Method Benchmark
Nu
JVM
Nu (with caching)
Calls per second
18,000,000
16,000,000
14,000,000
12,000,000
10,000,000
8,000,000
6,000,000
4,000,000
2,000,000
0
Same Instance
Same Synchronized Instance
Same Final Instance
Same Class
Same Synchronized Class
Other Instance
Other Instance of Abstract
Other Class

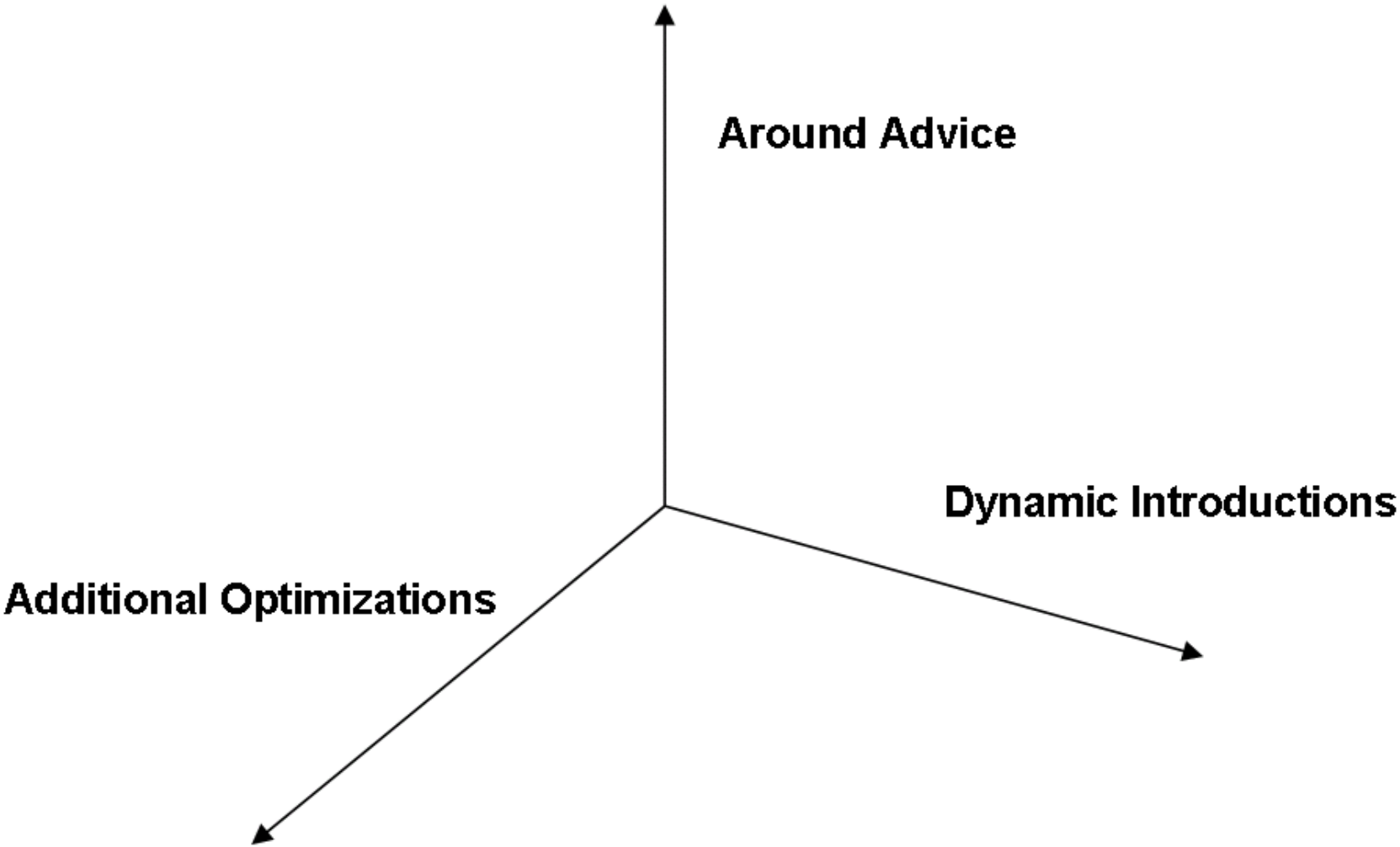

SPEC JVM98 Benchmark
Nu
JVM
Nu (with caching)
Average Time (seconds)
200
180
160
140
120
100
80
60
40
20
0
compress
jess
db
javac
mpegaudio
mtrt
jack


Cache Invalidation
Time (ms)
0
1
2
3
4
5
6
7
8
9
10
0
16
32
48
64
80
96
112
128
144
160
176
192
208
224
240
256
Number of Bound Patterns

Related Work

- Steamloom - Bockisch, Haupt, Mezini, and Ostermann
- Prose - Popovici, Alonso, and Gross
- Delegation-based Machine Model - Haupt and Schippers
- Morphing Aspects and Continuous Weaving - Hanenberg, Hirschfeld, and Unland

Future Work


Around Advice
Dynamic Introductions
Additional Optimizations

- Motivation: Supporting dynamic aspect-oriented constructs
  - cflow, deploy, history-based, etc
  - Compiled into static constructs
  - Lower-level support may yield run-time benefits
- Approach: Nu
  - IL-level primitives for dynamic deployment
  - Dedicated caching mechanism (low overhead)
- Evaluation: Expressiveness **and** performance
  - Supports large subset of dynamic AO constructs
  - Only about 1.5% overhead
- Technical Contributions:
  - Flexible, dynamic AO intermediate language model
  - Implementation in industrial strength VM (Sun Hotspot)
  - Dedicated AO caching mechanism

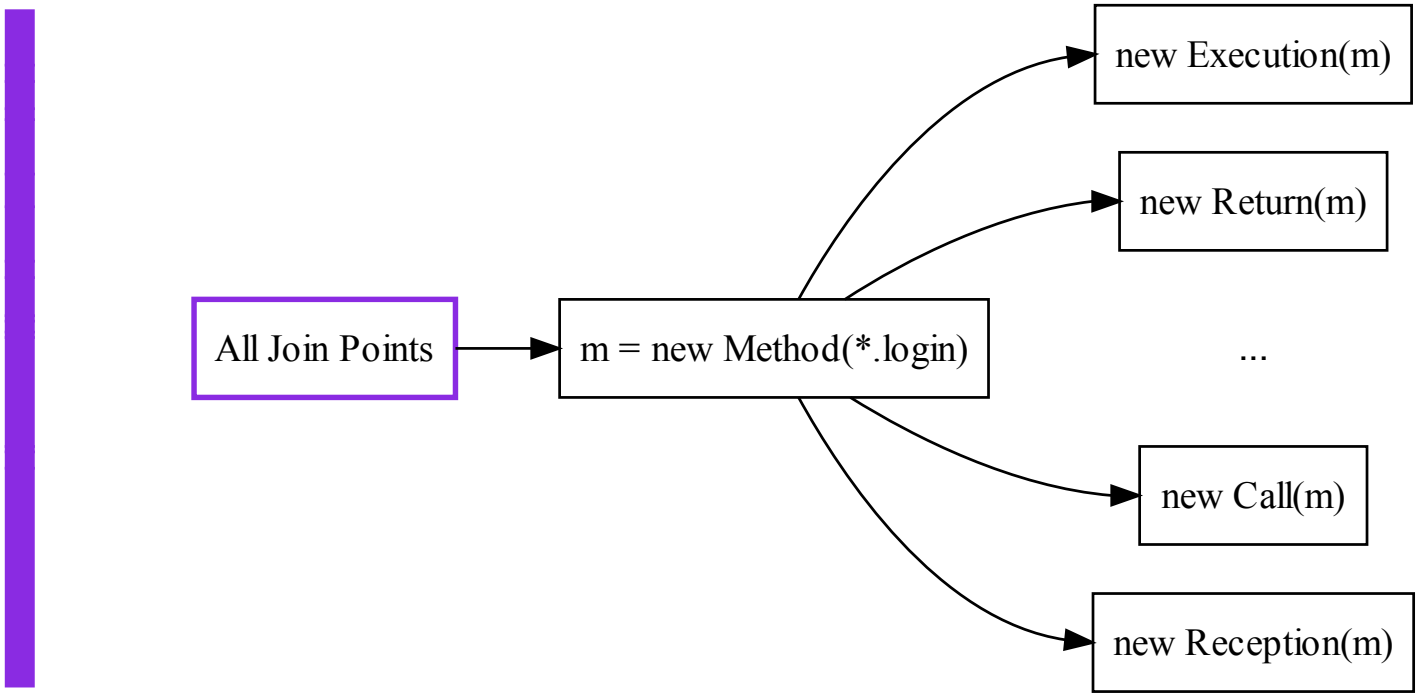
Questions?

`http://www.cs.iastate.edu/~nu/`

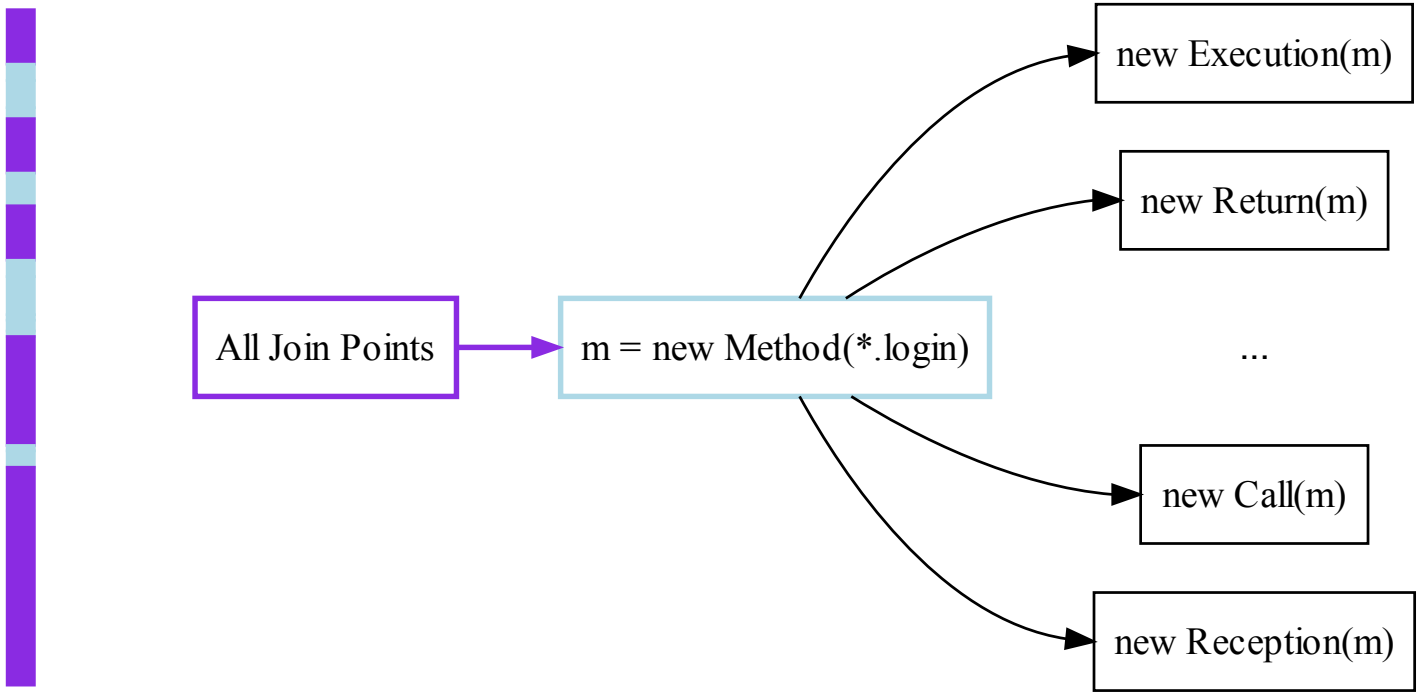
Caching Mechanism Assembly Code

```
    movl   eax, methodOop.counter
    movl   ecx, globalCounter

    // methodOop.counter == globalCounter?
    cmpl   eax, ecx
    jcc    equals, InvokeDelegates

    call_VM incrementalMatcher

InvokeDelegates:

    movl   eax, methodOop.cache.head

    // cache.head == NULL?
    testl eax, eax
    jcc    zero, ContinueJP

    // invoke the cached delegates

ContinueJP:
```

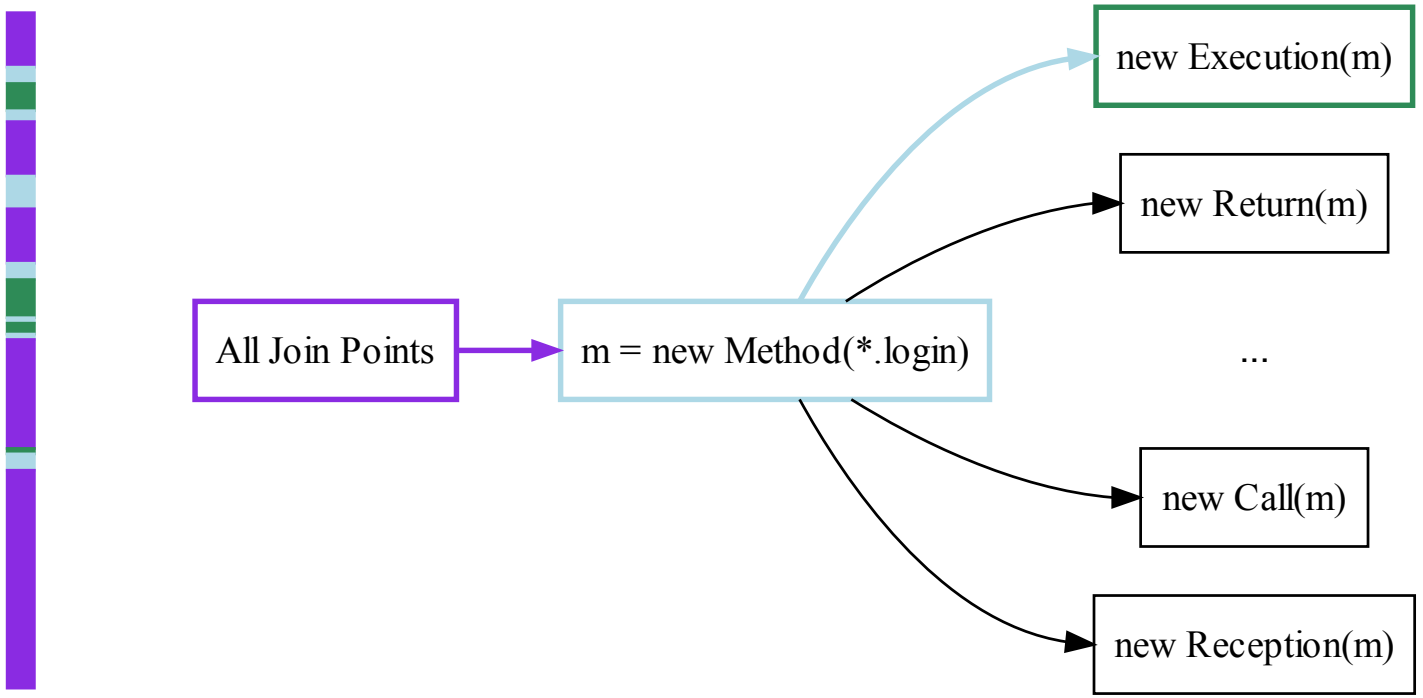
Pattern Library Example



Pattern Library Example



Pattern Library Example



Temporal Constructs - AspectJ

```
aspect Tcheck {
  pointcut p1(): call(* Word.set(..));
  int p1 = 1;

  pointcut p2(): call(* Bit.set(..));
  int p2 = 2;

  Formula state = Globally(p1, Finally(p2));
  Set<int> propSet = new Set<int>();

  ...
```

Temporal Constructs - AspectJ

```
aspect Tcheck {
  pointcut p1(): call(* Word.set(..));
  int p1 = 1;

  pointcut p2(): call(* Bit.set(..));
  int p2 = 2;

  Formula state = Globally(p1, Finally(p2));
  Set<int> propSet = new Set<int>();

  ...
```

Temporal Constructs - AspectJ

```
aspect Tcheck {
  pointcut p1(): call(* Word.set(..));
  int p1 = 1;

  pointcut p2(): call(* Bit.set(..));
  int p2 = 2;

  Formula state = Globally(p1, Finally(p2));
  Set<int> propSet = new Set<int>();

  ...
```

Temporal Constructs - AspectJ

```
...
after(): p1() { propSet.add(p1); }
after(): p2() { propSet.add(p2); }

after(): p1() || p2() {
  state = state.transition(propSet);
  if (state.equals(Formula.TT))
    // report formula as satisfied
  else if(state.equals(Formula.FF))
    // report formula as falsified
  state.clear(); //reset proposition vector
}
```

Temporal Constructs - AspectJ

```
...
after(): p1() { propSet.add(p1); }
after(): p2() { propSet.add(p2); }

after(): p1() || p2() {
  state = state.transition(propSet);
  if (state.equals(Formula.TT))
    // report formula as satisfied
  else if(state.equals(Formula.FF))
    // report formula as falsified
  state.clear(); //reset proposition vector
}
```

Temporal Constructs - Nu

```
class Tcheck {
  int p1 = 1;
  int p2 = 2;

  Formula state = Globally(p1, Finally(p2));
  Set<int> propSet = new Set<int>();

  ...
```

Temporal Constructs - Nu

```
class Tcheck {
  int p1 = 1;
  int p2 = 2;

  Formula state = Globally(p1, Finally(p2));
  Set<int> propSet = new Set<int>();

  ...
```

Temporal Constructs - Nu

```
class Tcheck {
  int p1 = 1;
  int p2 = 2;

  Formula state = Globally(p1, Finally(p2));
  Set<int> propSet = new Set<int>();

  ...
```

Temporal Constructs - Nu

```
...

protected static Pattern prop2;
protected static Delegate d2;

static {
  Pattern prop1 = new Call("* Word.set(..)");
  d1 = new Delegate(Tcheck.class, "afterP1");
  bind(prop1, d1);

  prop2 = new Call("* Bit.set(..)");
  d2 = new Delegate(Tcheck.class, "afterP2");

  Pattern afterP1P2 = new Or(prop1, prop2);
  d3 = new Delegate(Tcheck.class, "afterP1P2");
  bind(afterP1P2, d3);
}
```

Temporal Constructs - Nu

```
...

protected static Pattern prop2;
protected static Delegate d2;

static {
  Pattern prop1 = new Call("* Word.set(..)");
  d1 = new Delegate(Tcheck.class, "afterP1");
  bind(prop1, d1);

  prop2 = new Call("* Bit.set(..)");
  d2 = new Delegate(Tcheck.class, "afterP2");

  Pattern afterP1P2 = new Or(prop1, prop2);
  d3 = new Delegate(Tcheck.class, "afterP1P2");
  bind(afterP1P2, d3);
}
```

Temporal Constructs - Nu

```
...

protected static Pattern prop2;
protected static Delegate d2;

static {
  Pattern prop1 = new Call("* Word.set(..)");
  d1 = new Delegate(Tcheck.class, "afterP1");
  bind(prop1, d1);

  prop2 = new Call("* Bit.set(..)");
  d2 = new Delegate(Tcheck.class, "afterP2");

  Pattern afterP1P2 = new Or(prop1, prop2);
  d3 = new Delegate(Tcheck.class, "afterP1P2");
  bind(afterP1P2, d3);
}
```

Temporal Constructs - Nu

```
...
protected static BindHandle id;
public void afterP1() {
  propSet.add(p1);
  id = bind(prop2, d2);
}
public void afterP2() {
  propSet.add(p2);
  remove(id);
}
public void afterP1P2() {
  ... // same as before
}
```

Temporal Constructs - Nu

```
...
protected static BindHandle id;
public void afterP1() {
  propSet.add(p1);
  id = bind(prop2, d2);
}
public void afterP2() {
  propSet.add(p2);
  remove(id);
}
public void afterP1P2() {
  ... // same as before
}
```

Temporal Constructs - Nu

```
...
protected static BindHandle id;
public void afterP1() {
  propSet.add(p1);
  id = bind(prop2, d2);
}
public void afterP2() {
  propSet.add(p2);
  remove(id);
}
public void afterP1P2() {
  ... // same as before
}
```