Software Method Level Speculation for Java

Christopher John Francis Pickett
chris.pickett@mail.mcgill.ca

Sable Research Group
School of Computer Science
McGill University

February 17th, 2012

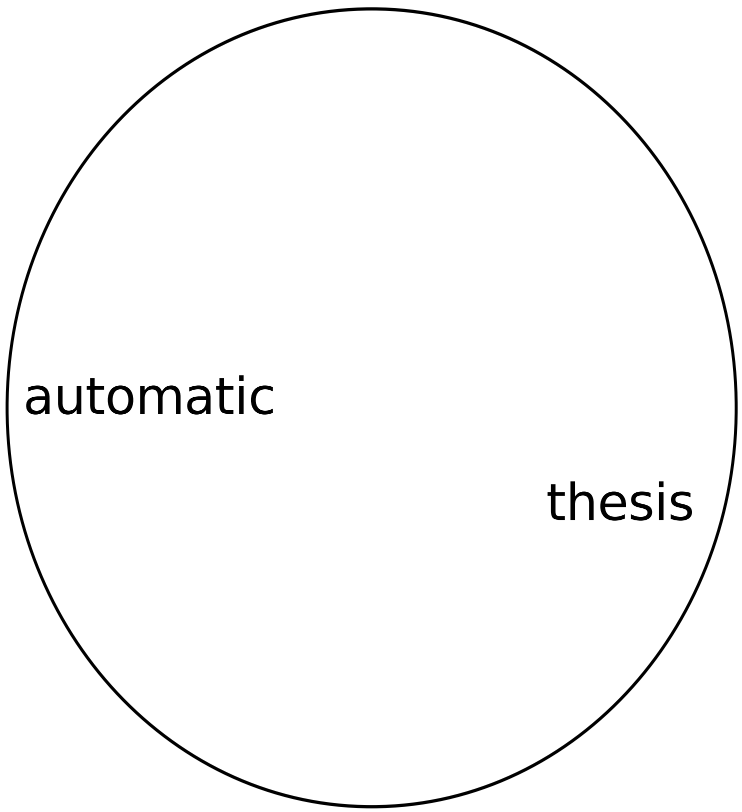
Ph.D. Oral Defense

Parallelization

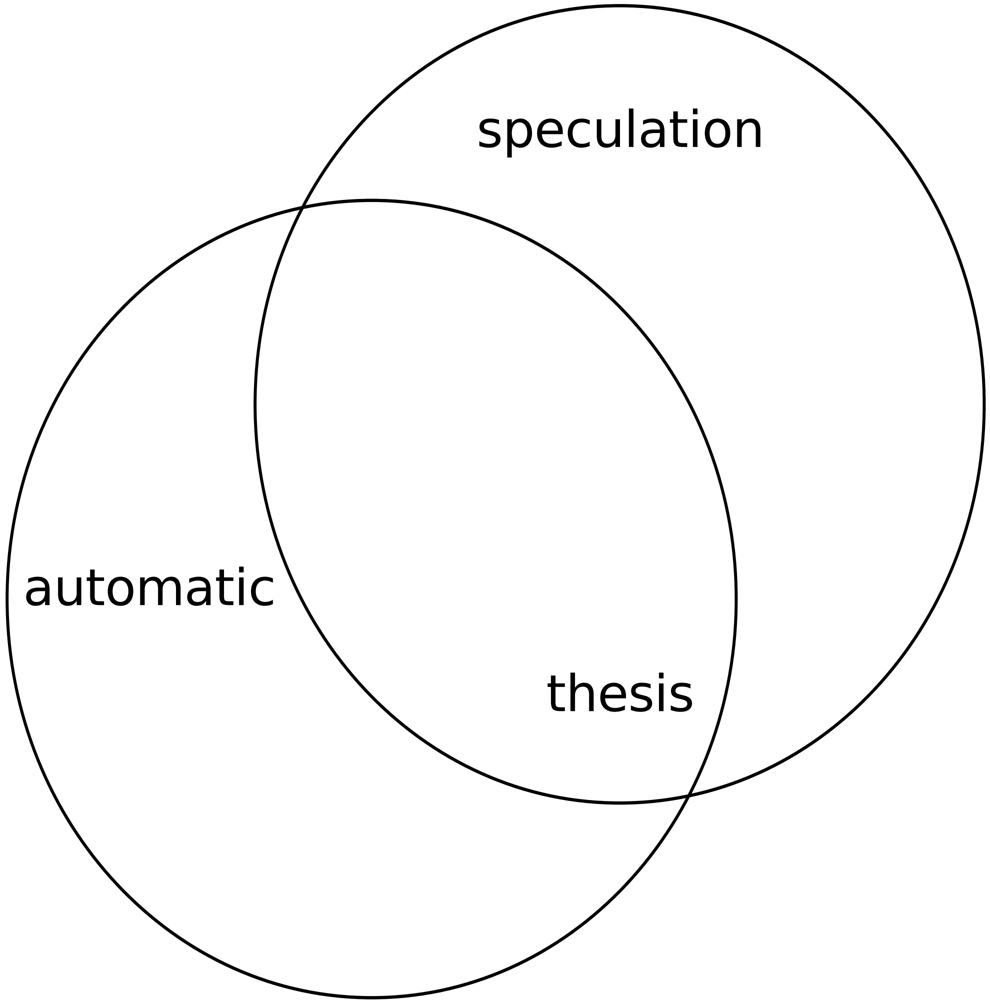
Parallelization

thesis

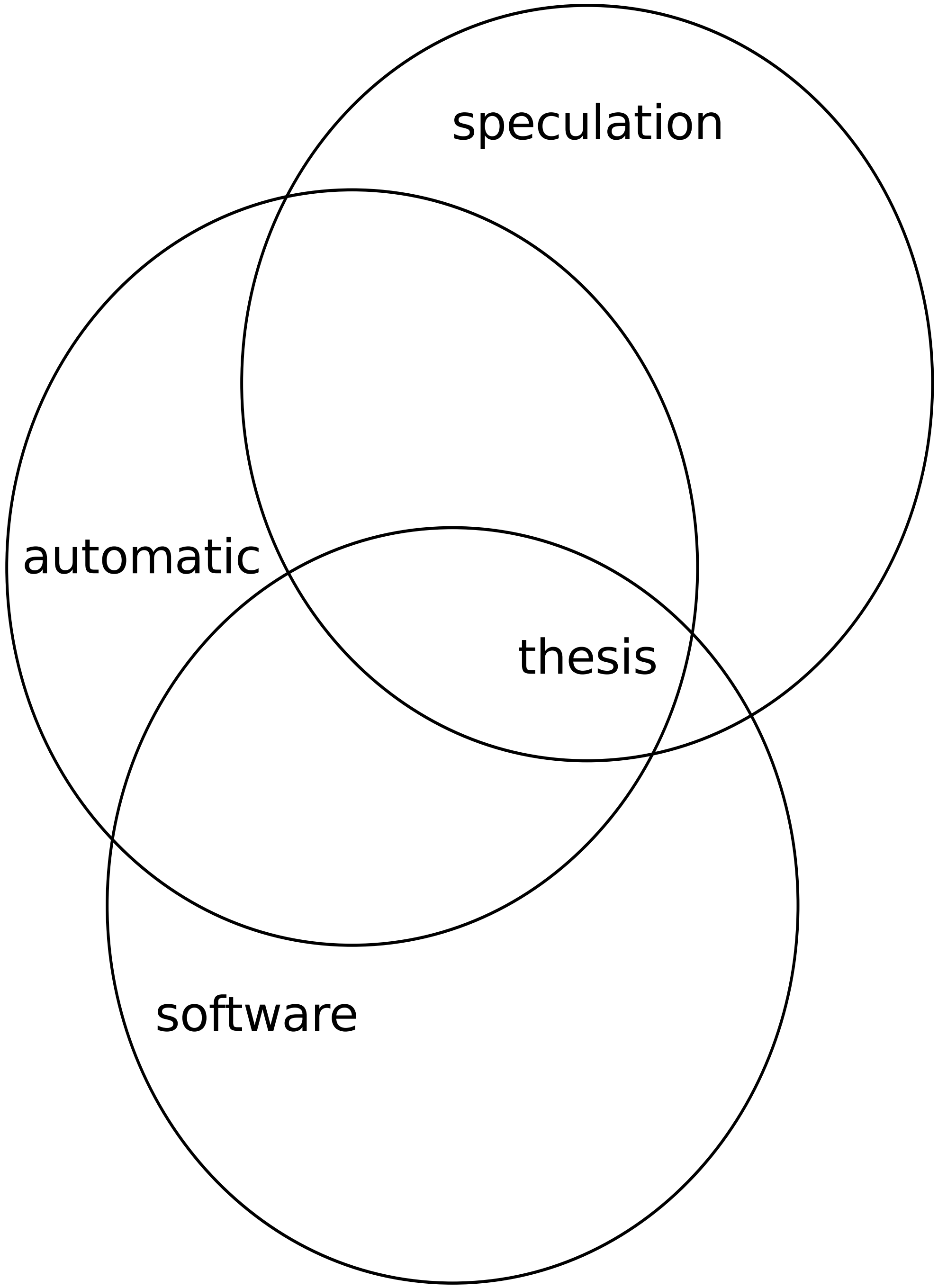
Parallelization



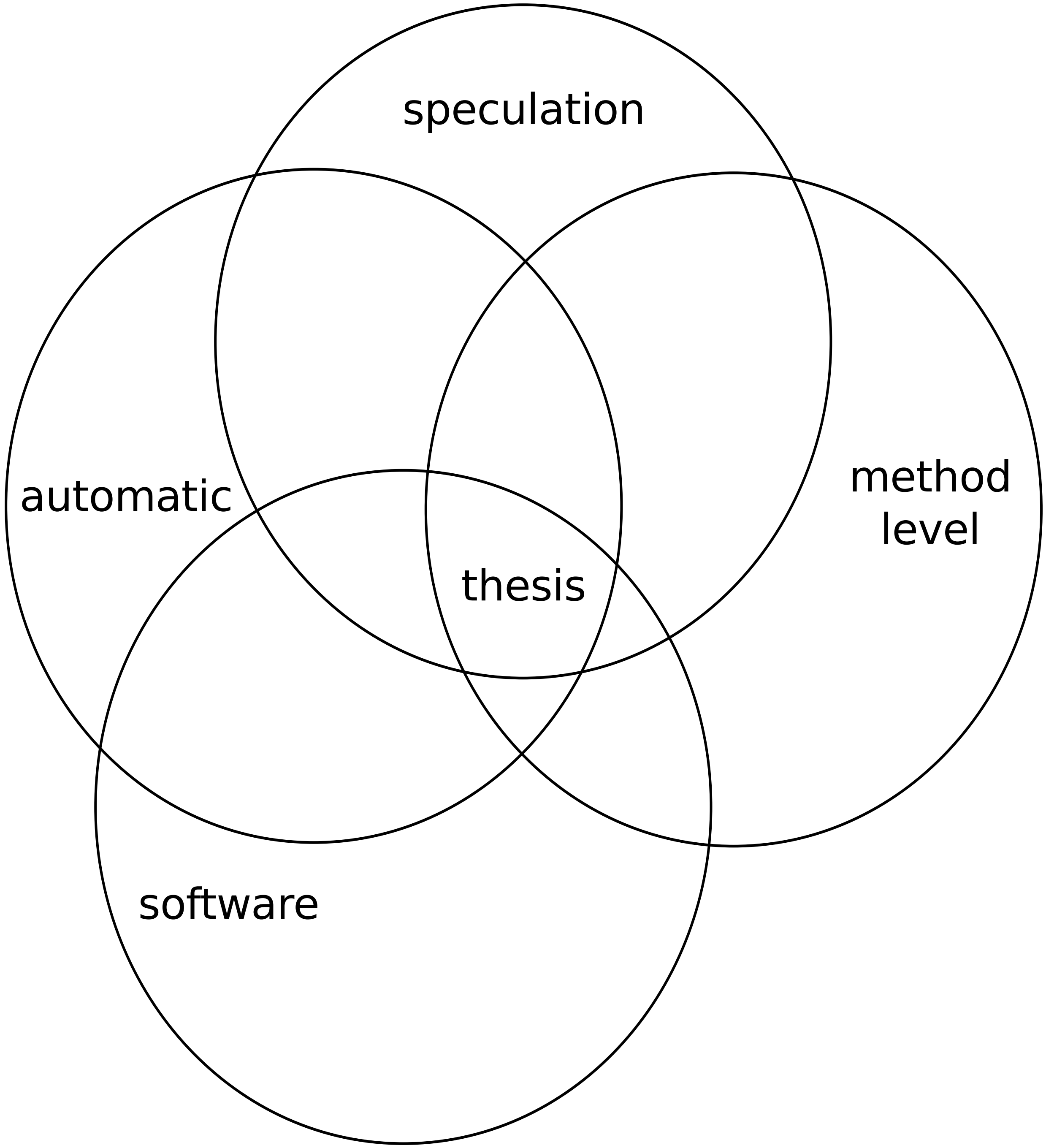
Parallelization



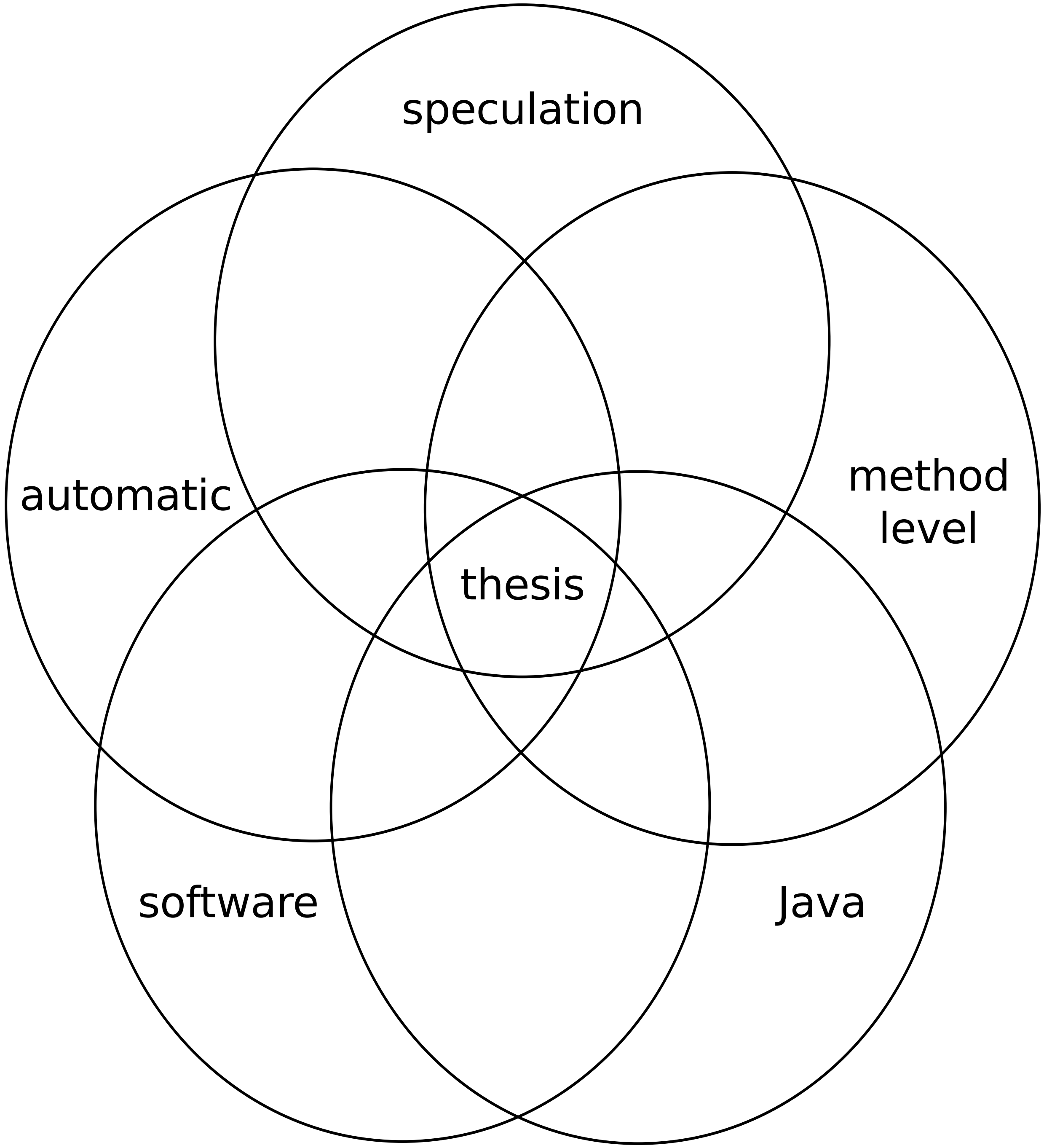
Parallelization



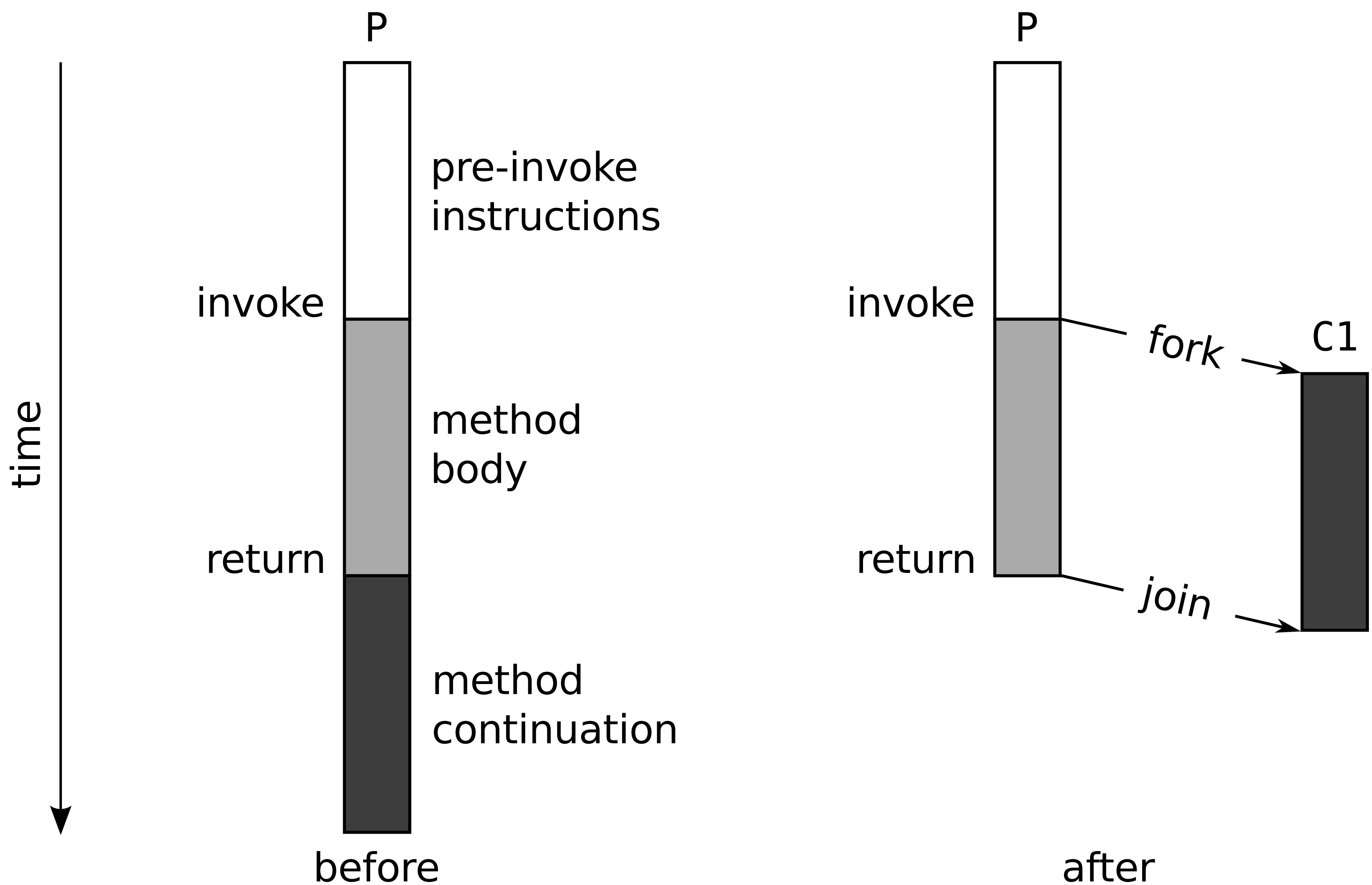
Parallelization



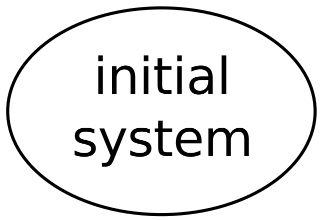
Parallelization



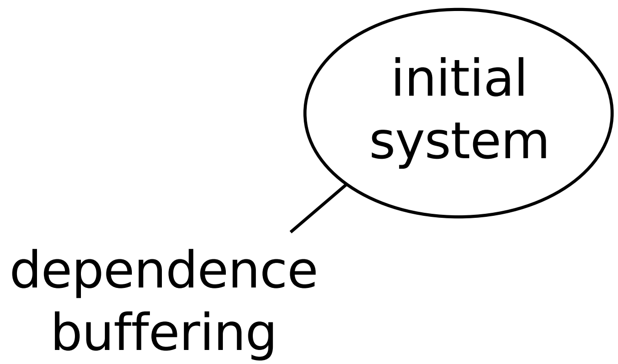
Method Level Speculation



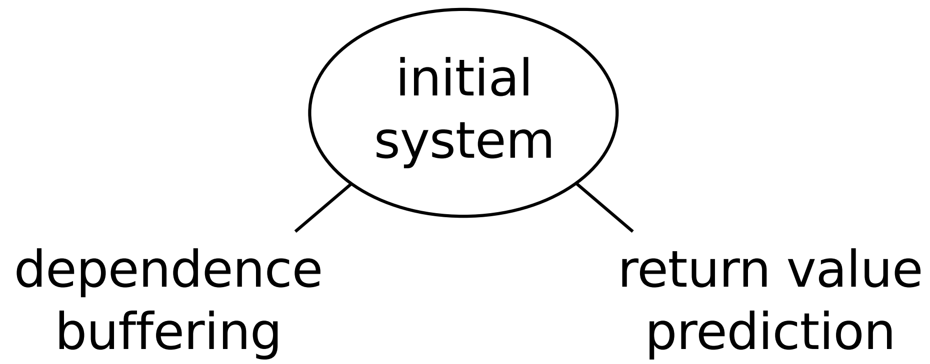
Initial System



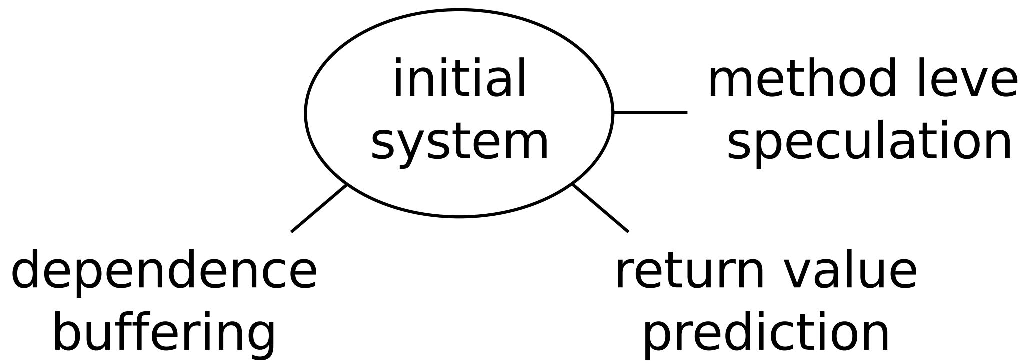
Initial System



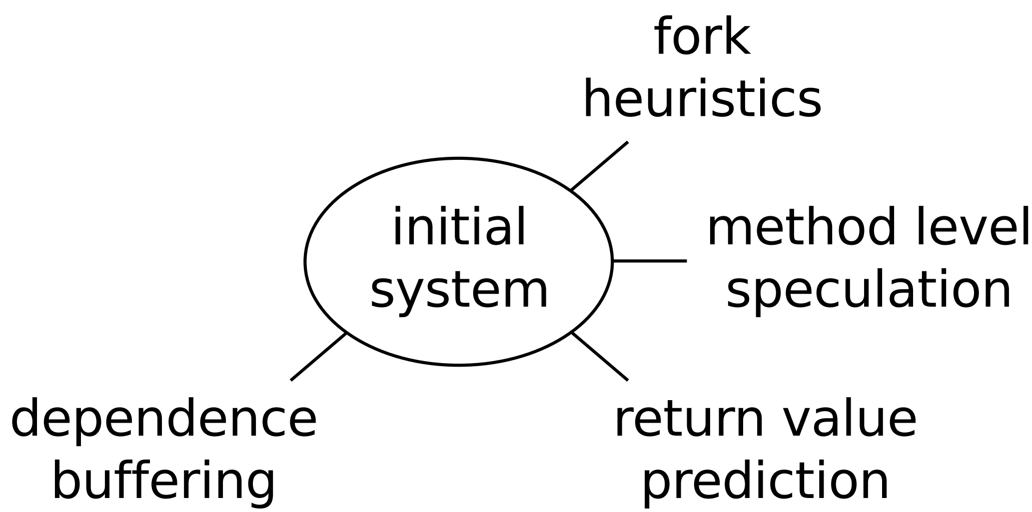
Initial System



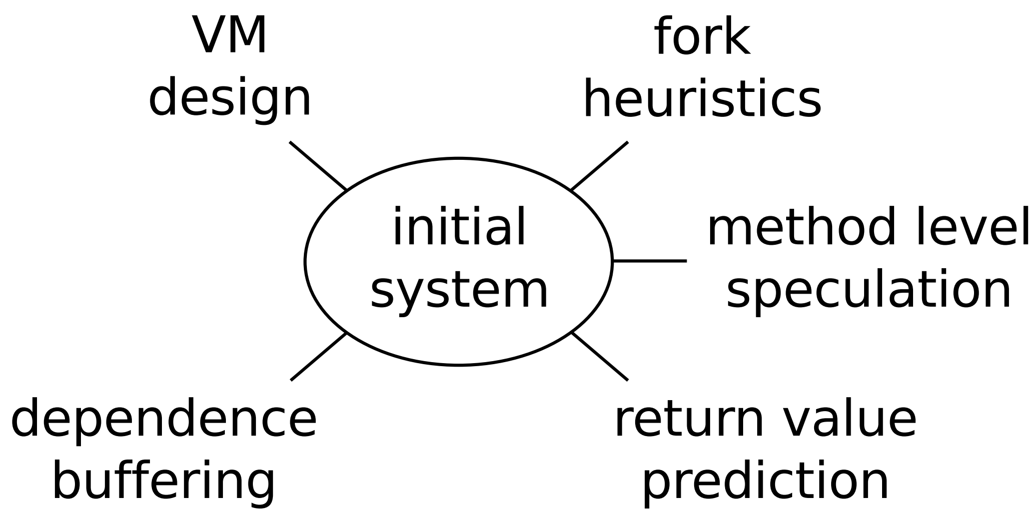
Initial System



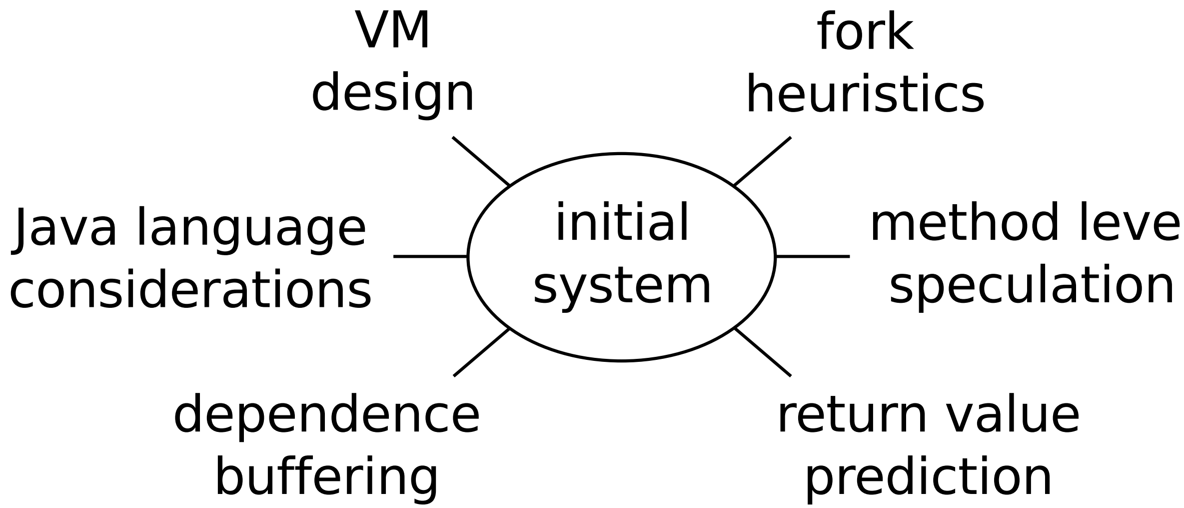
Initial System



Initial System



Initial System



Example

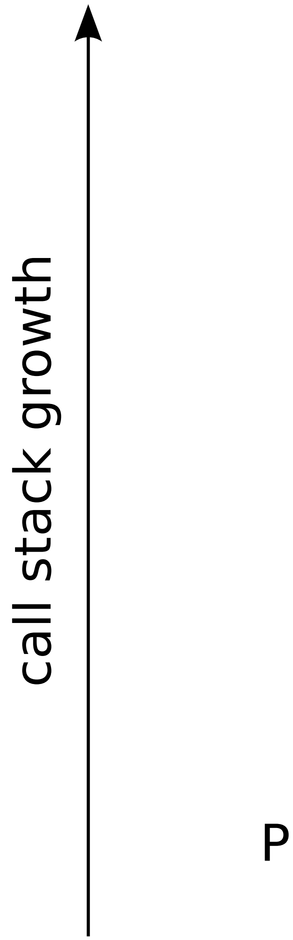
```
// execute foo() non-speculatively
r = foo (x, y, z);
```

Example

```
// execute foo() non-speculatively
r = foo (x, y, z);

// execute foo()'s continuation speculatively?
```

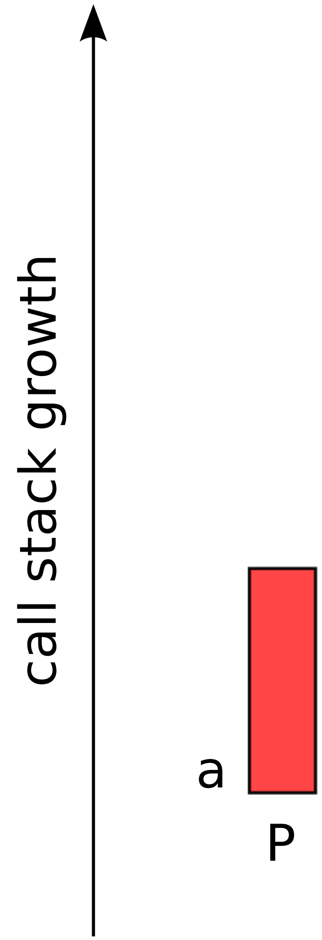
Single Child Speculation

```
A(){
   a;
   B(){
      b;
   }
   a';
}
```



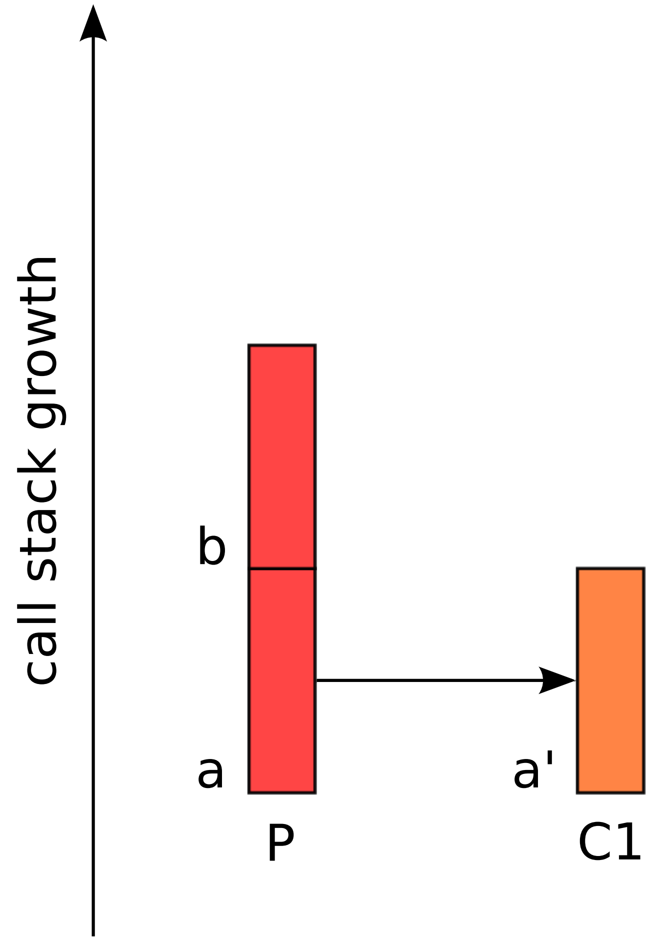
Single Child Speculation

```
A(){
  a;
  B(){
    b;
  }
  a';
}
```



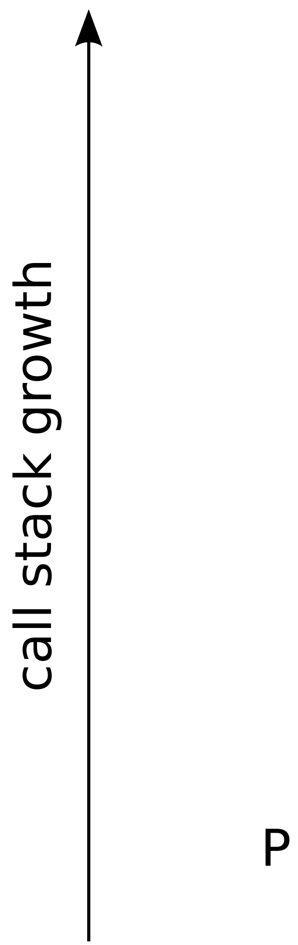
Single Child Speculation

```
A(){
   a;
   B(){
      b;
   }
   a';
}
```



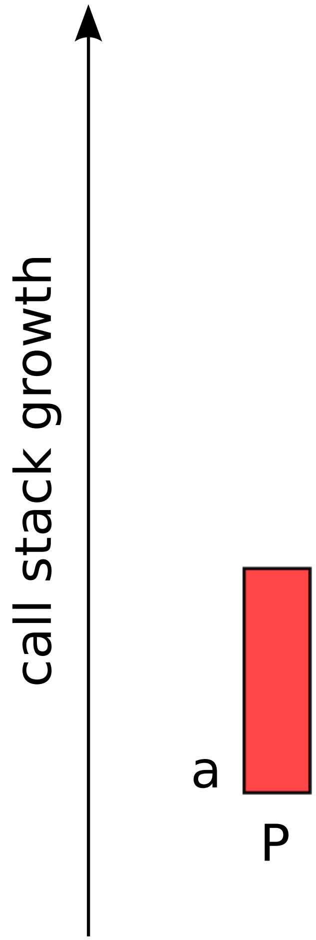
Out-of-Order Nesting

```
A(){
   a;
   B(){
      b;
      C(){
         c;
      }
      b';
   }
   a';
}
```



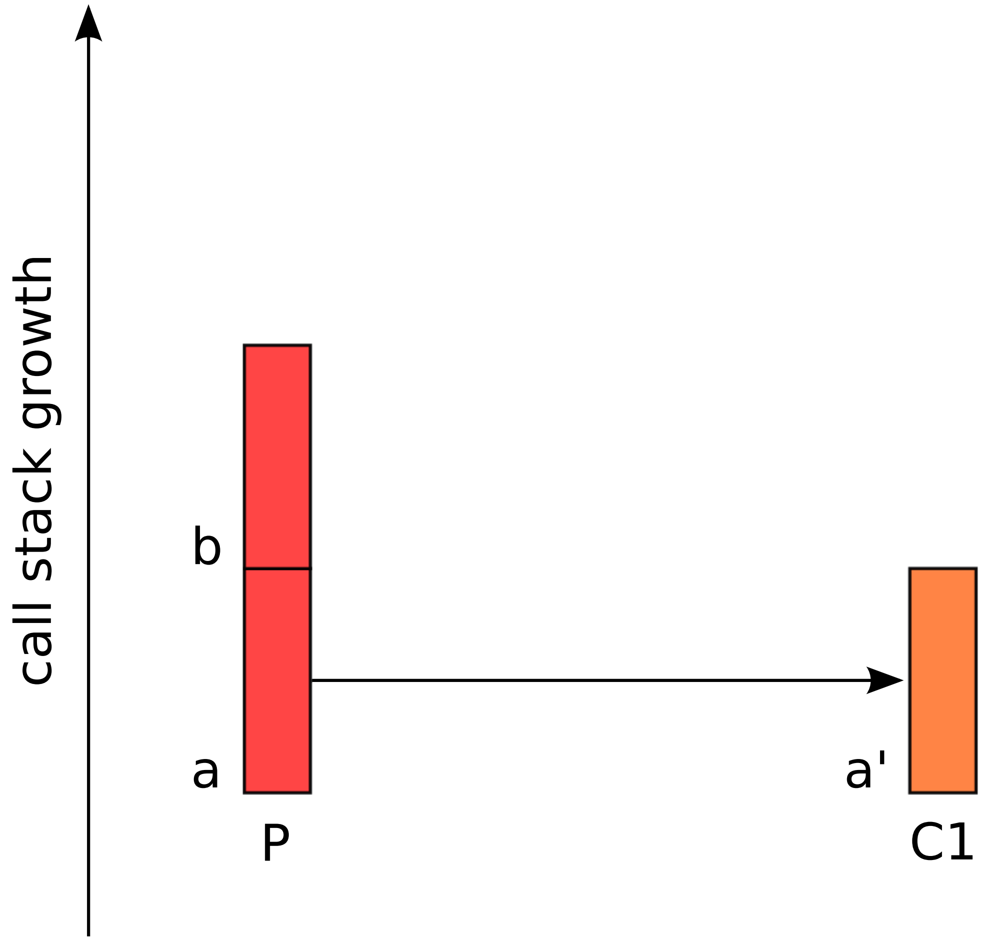
Out-of-Order Nesting

```
A(){
  a;
  B(){
    b;
    C(){
      c;
    }
    b';
  }
  a';
}
```



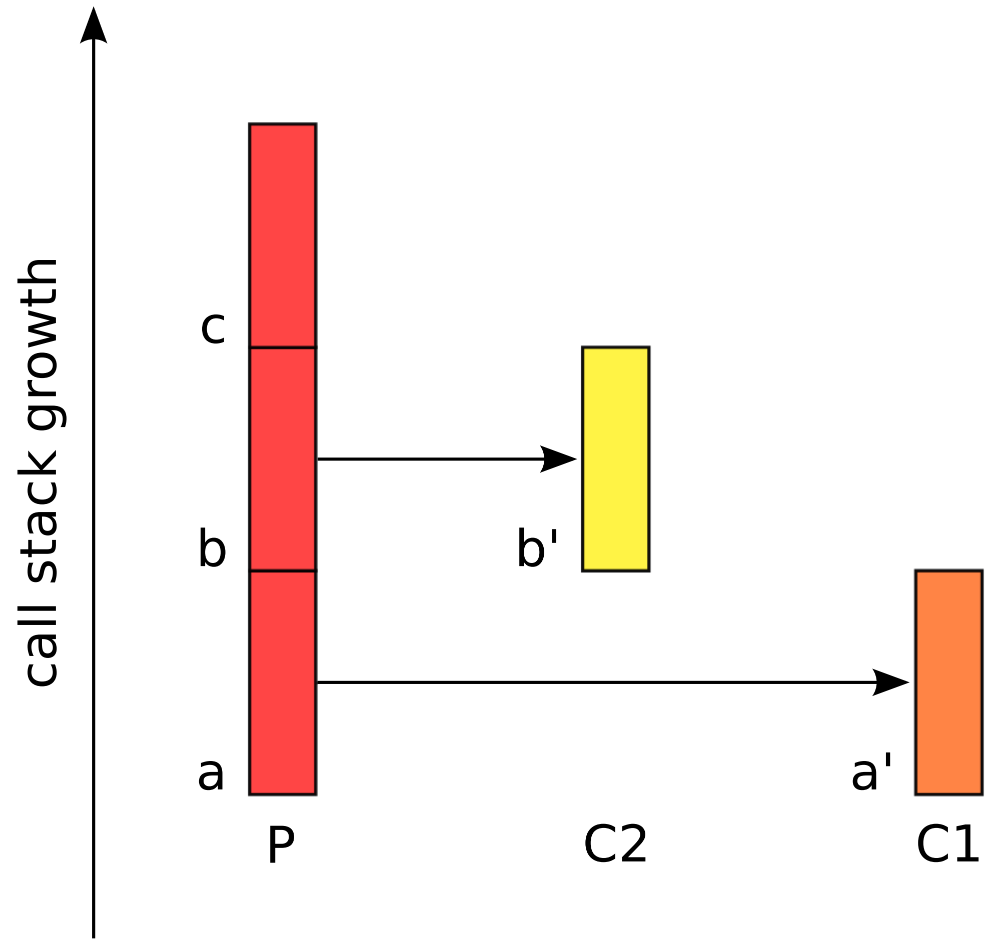
Out-of-Order Nesting

```
A(){
   a;
   B(){
      b;
      C(){
         c;
      }
      b';
   }
   a';
}
```



Out-of-Order Nesting

```
A(){
   a;
   B(){
      b;
      C(){
         c;
      }
      b';
   }
   a';
}
```



Example

```
// execute foo() non-speculatively
r = foo (x, y, z);

// execute foo()'s continuation speculatively
if (r > 10)        // predict return value
  {
    ...
  }
```

Return Value Prediction

predictor	history	prediction
last value	$1, 1, 1$	1

Return Value Prediction

predictor	history	prediction
last value	$1, 1, 1$	1
stride	$1, 2, 3$	4

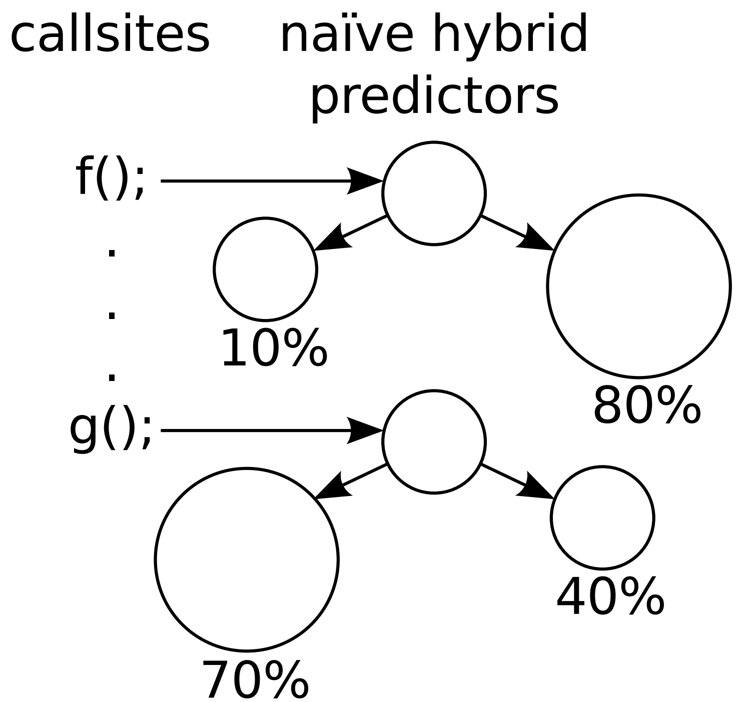
Return Value Prediction

predictor	history	prediction
last value	$1, 1, 1$	1
stride	$1, 2, 3$	4
context	$1, 5, 6, 8, \ldots, 1, 5, 6$	8

Return Value Prediction

predictor	history	prediction
last value	$1, 1, 1$	1
stride	$1, 2, 3$	4
context	$1, 5, 6, 8, \ldots, 1, 5, 6$	8
memoization	$f(2, 4) : 7, \ldots, f(2, 4)$	7

Software Hybrid RVP



Example

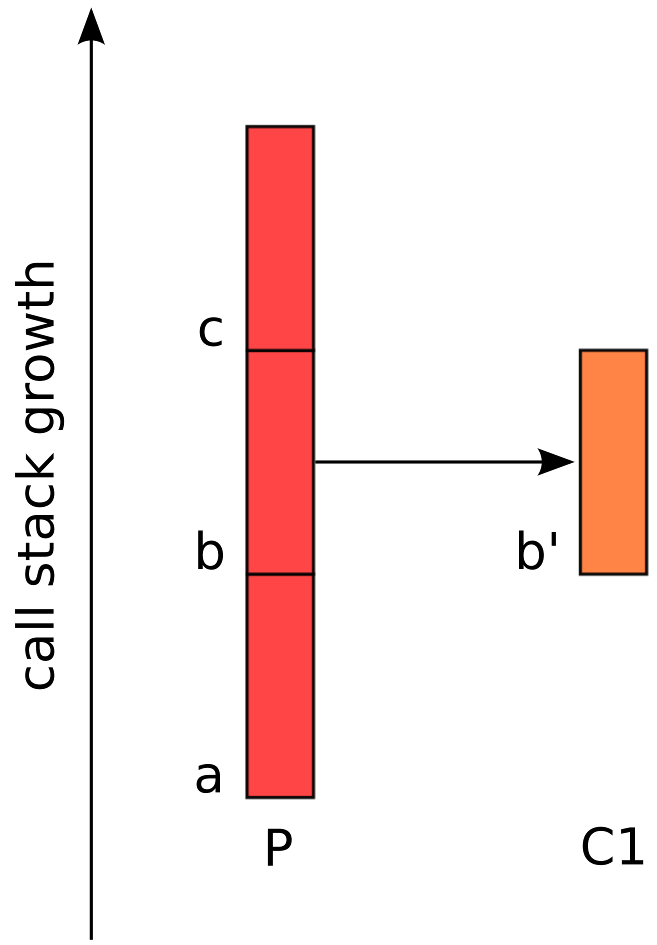
```
// execute foo() non-speculatively
r = foo (x, y, z);

// execute foo()'s continuation speculatively
if (r > 10)        // predict return value
  {
    s = o1.f;      // buffer heap & static reads
    o2.f = r;      // buffer heap & static writes
  }
```

Example

```
// execute foo() non-speculatively
r = foo (x, y, z);

// execute foo()'s continuation speculatively
if (r > 10)        // predict return value
  {
    s = o1.f;      // buffer heap & static reads
    o2.f = r;      // buffer heap & static writes
  }

// invoke bar () speculatively
o3.bar ();
```

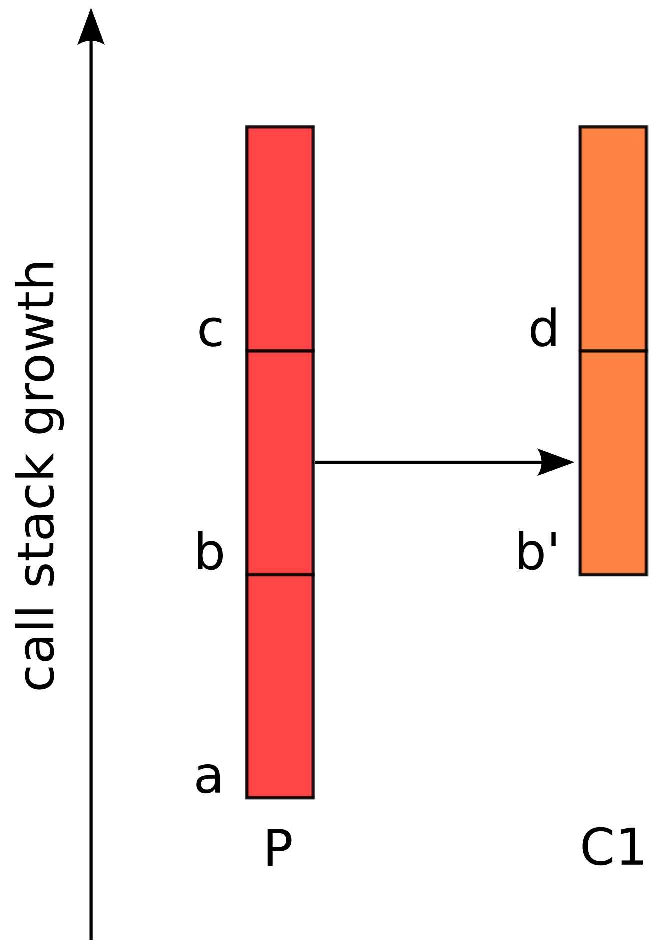
Speculative Method Invocation

```
A(){
  a;
  B(){
    b;
    C(){
      c;
    }
    b';
    D(){
      d;
    }
  }
  a';
}
```



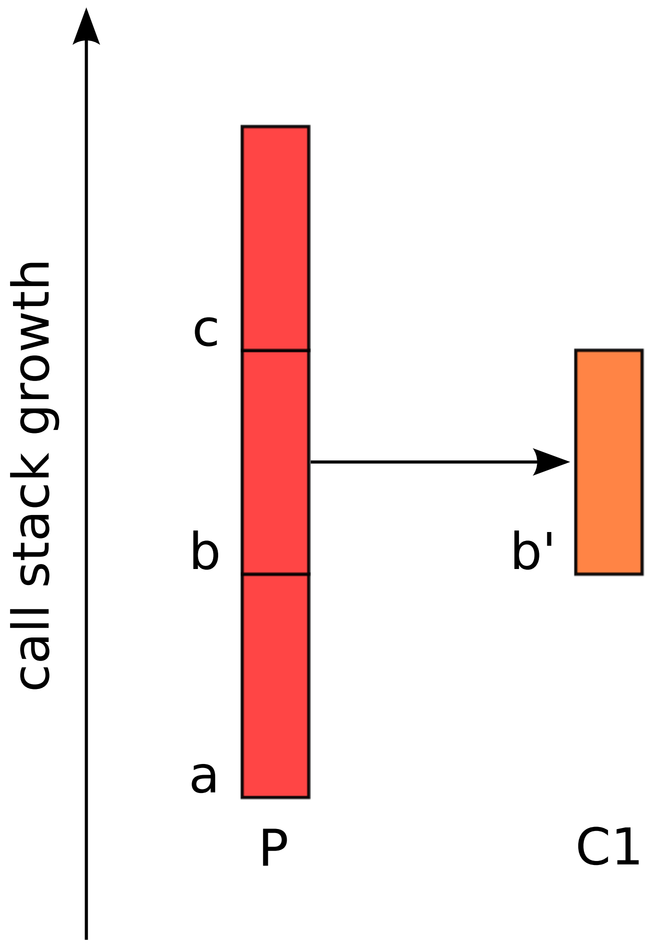
Speculative Method Invocation

```
A(){
   a;
   B(){
      b;
      C(){
         c;
      }
      b';
      D(){
         d;
      }
   }
   a';
}
```



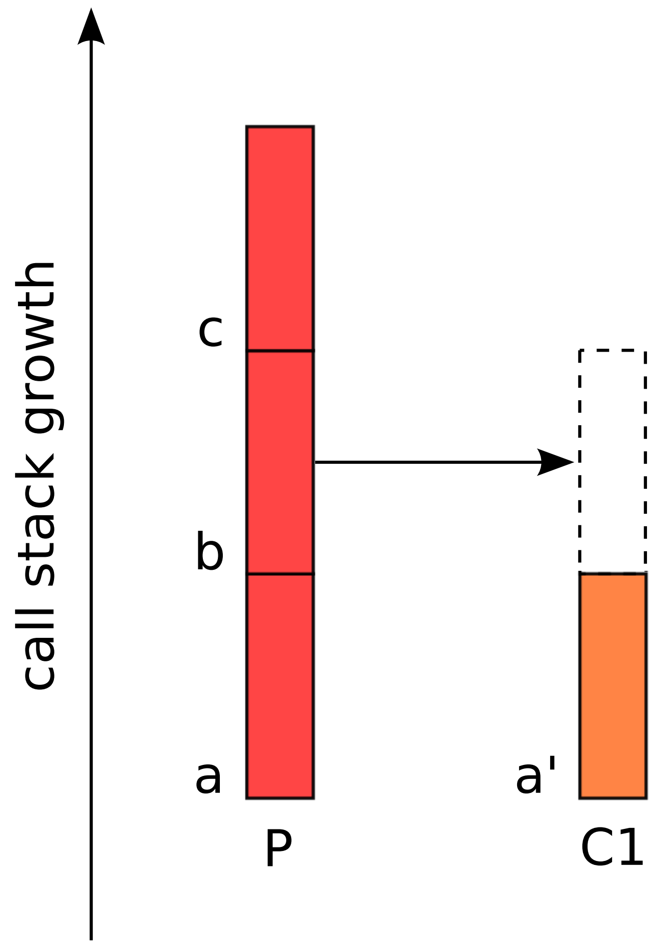
Speculative Method Invocation

```
A(){
   a;
   B(){
      b;
      C(){
         c;
      }
      b';
      D(){
         d;
      }
   }
   a';
}
```



Speculative Method Invocation

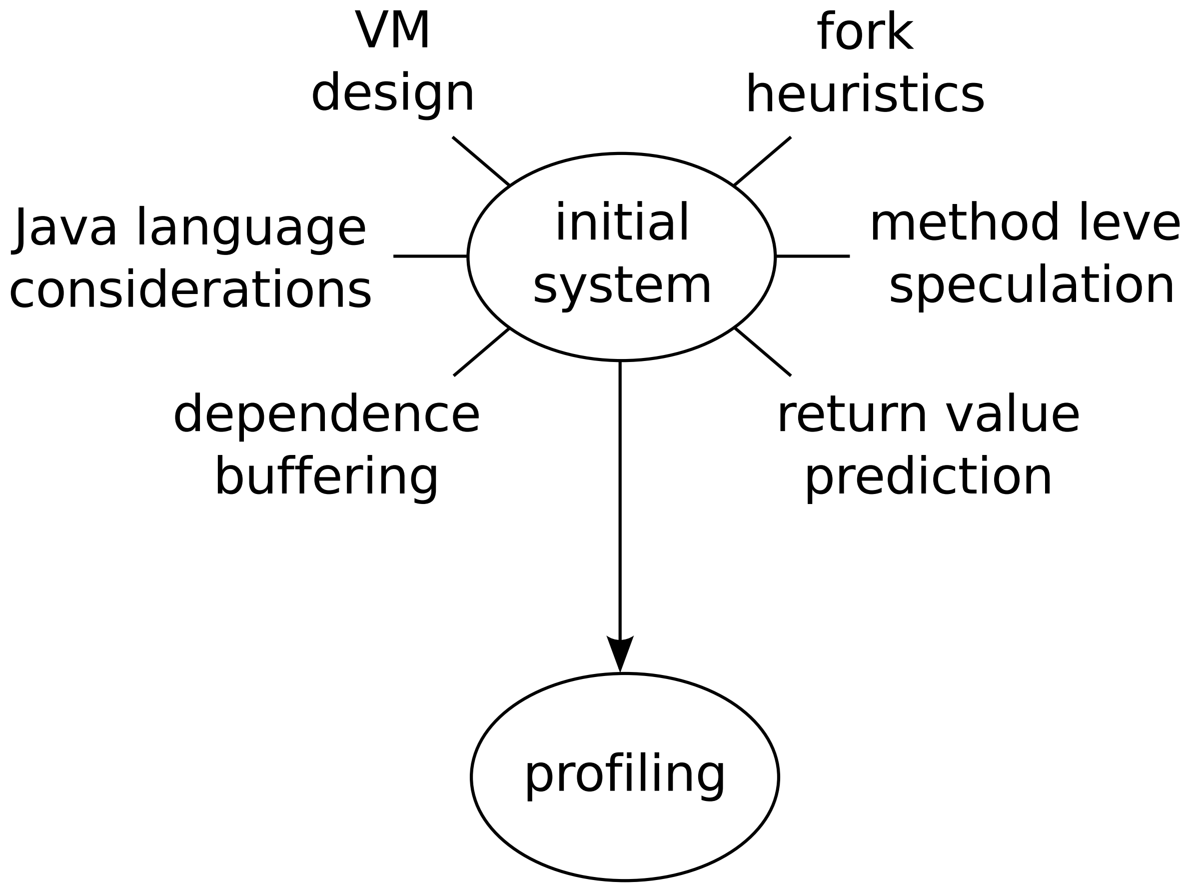
```
A(){
   a;
   B(){
      b;
      C(){
         c;
      }
      b';
      D(){
         d;
      }
   }
   a';
}
```



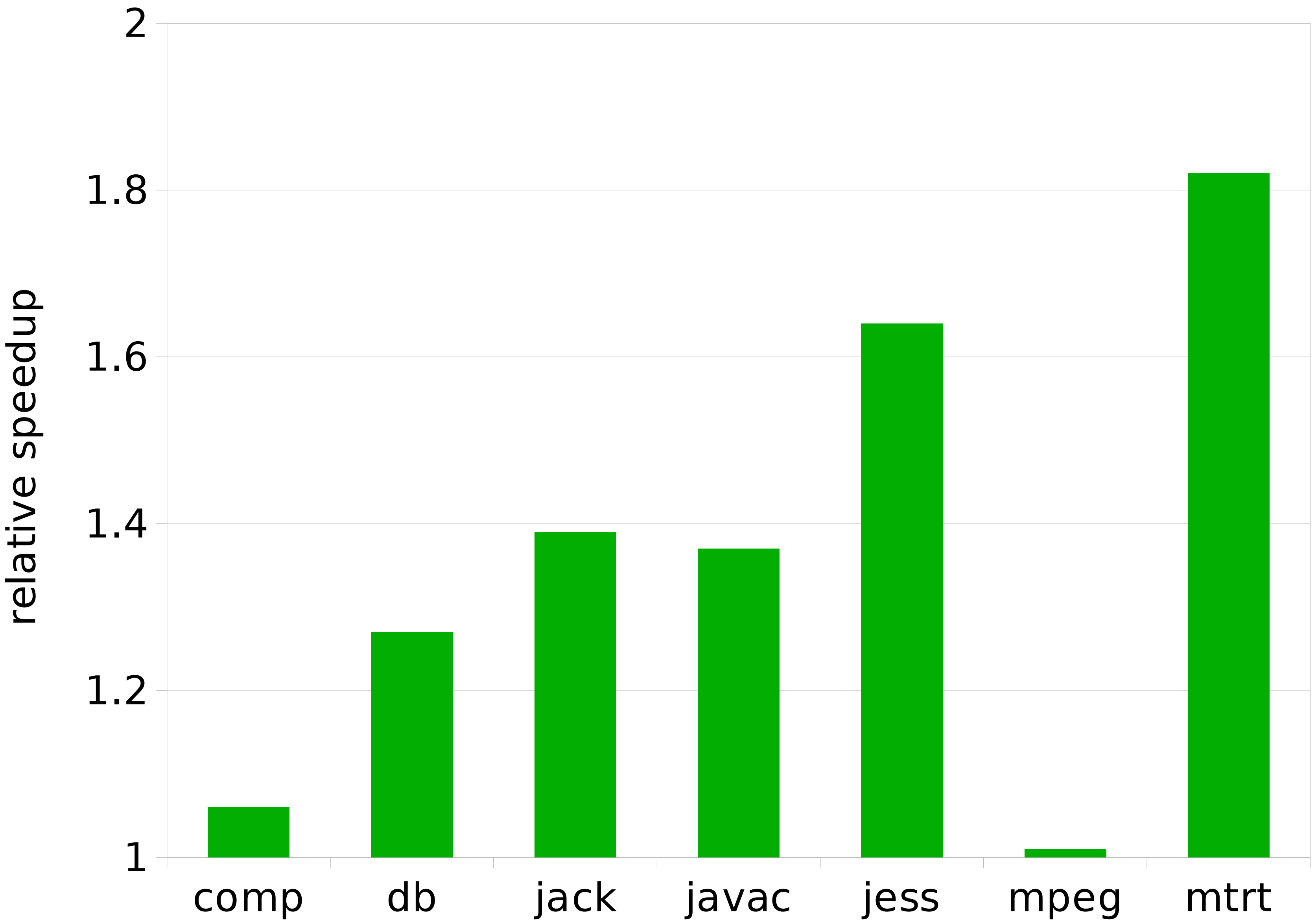
Example

```
// execute foo() non-speculatively
r = foo (x, y, z);

// execute foo()'s continuation speculatively
if (r > 10)      // predict return value
  {
    s = o1.f;    // buffer heap & static reads
    o2.f = r;    // buffer heap & static writes
  }

// invoke bar () speculatively
o3.bar ();

// stop speculation due to unsafe operation
synchronized (o4) { ... }
```

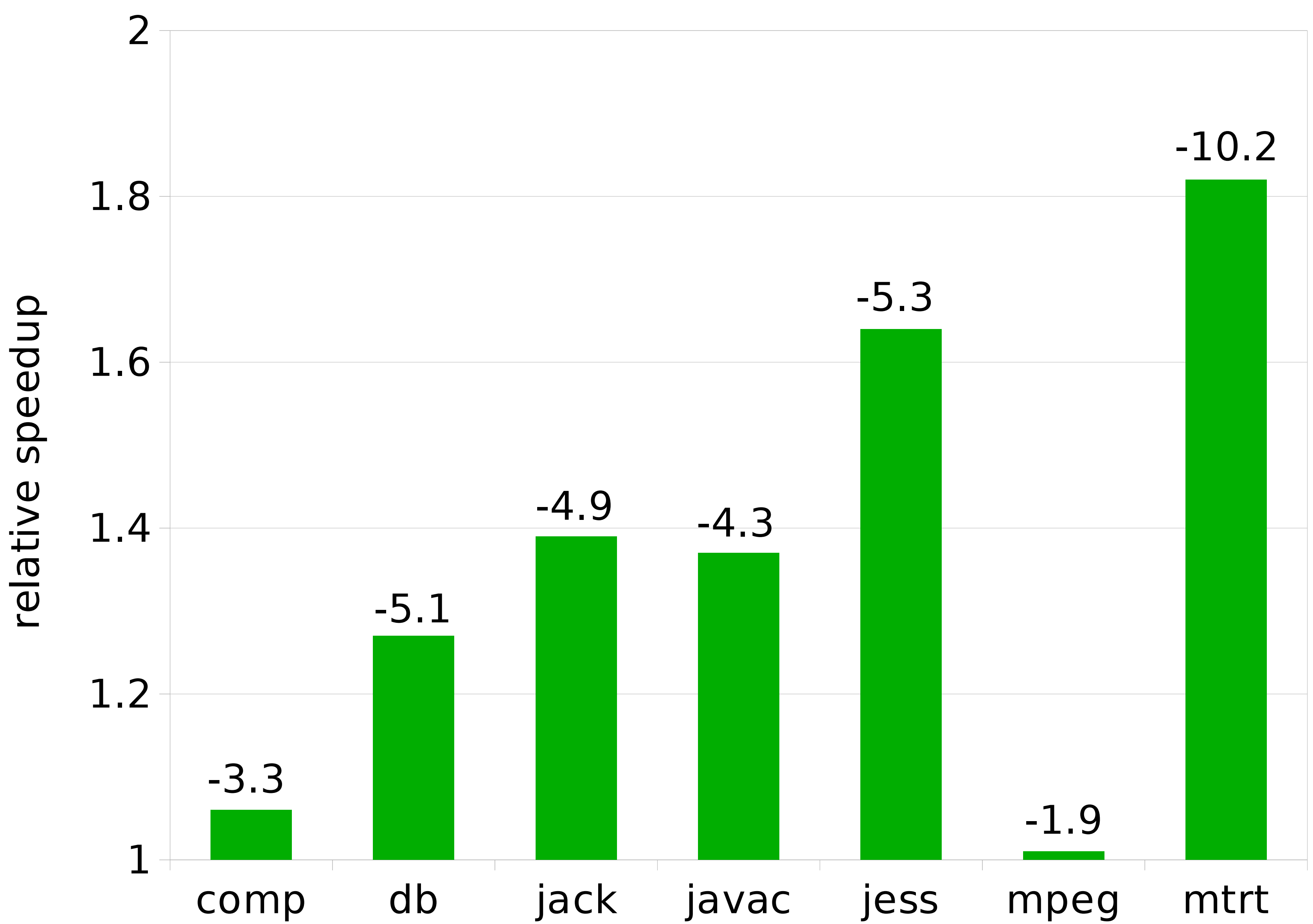
Profiling


VM design
fork heuristics
Java language considerations
initial system
method level speculation
dependence buffering
return value prediction
profiling

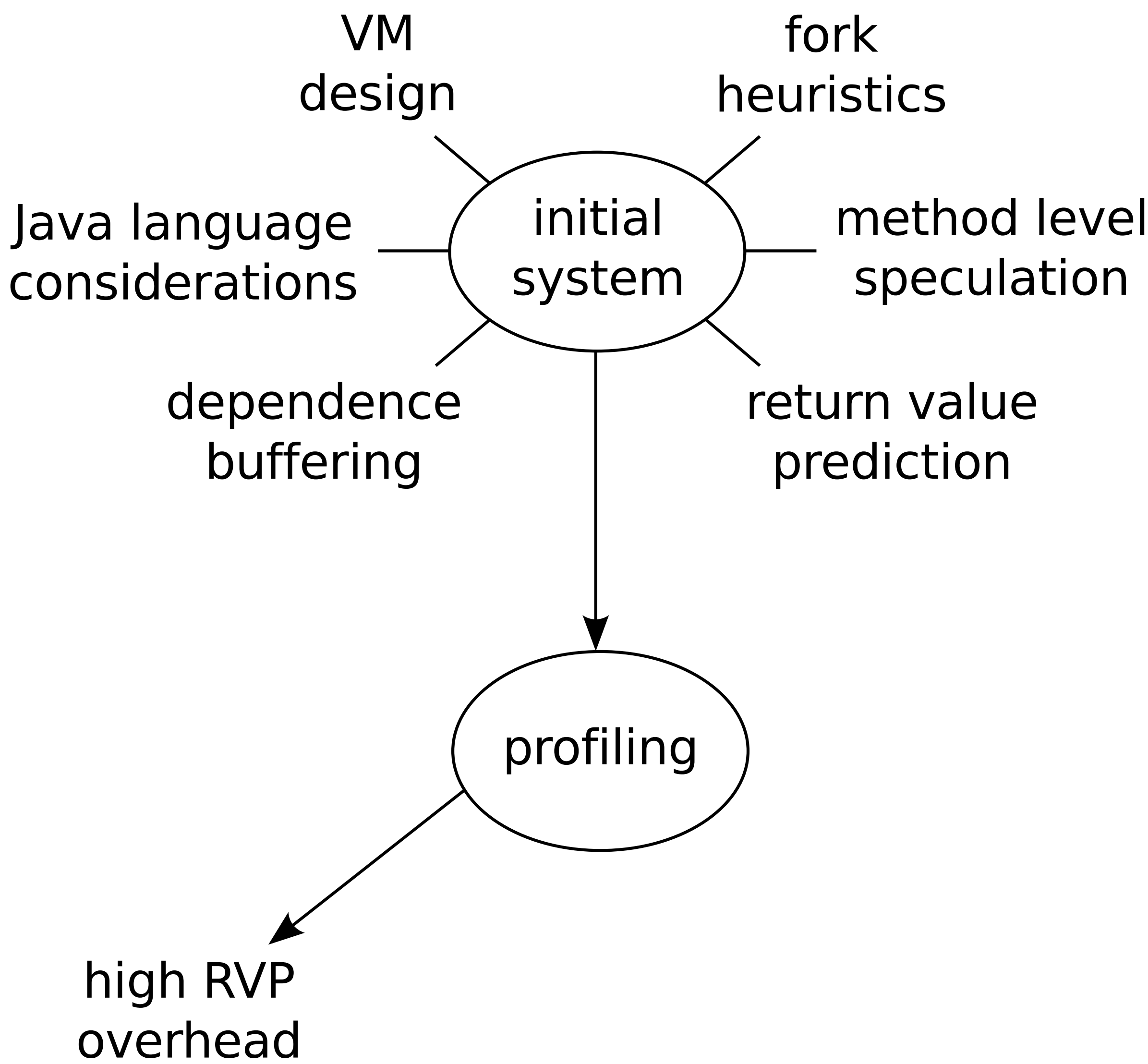
Speedup...


relative speedup
2
1.8
1.6
1.4
1.2
1
comp
db
jack
javac
jess
mpeg
mtrt

Speedup... and Slowdown

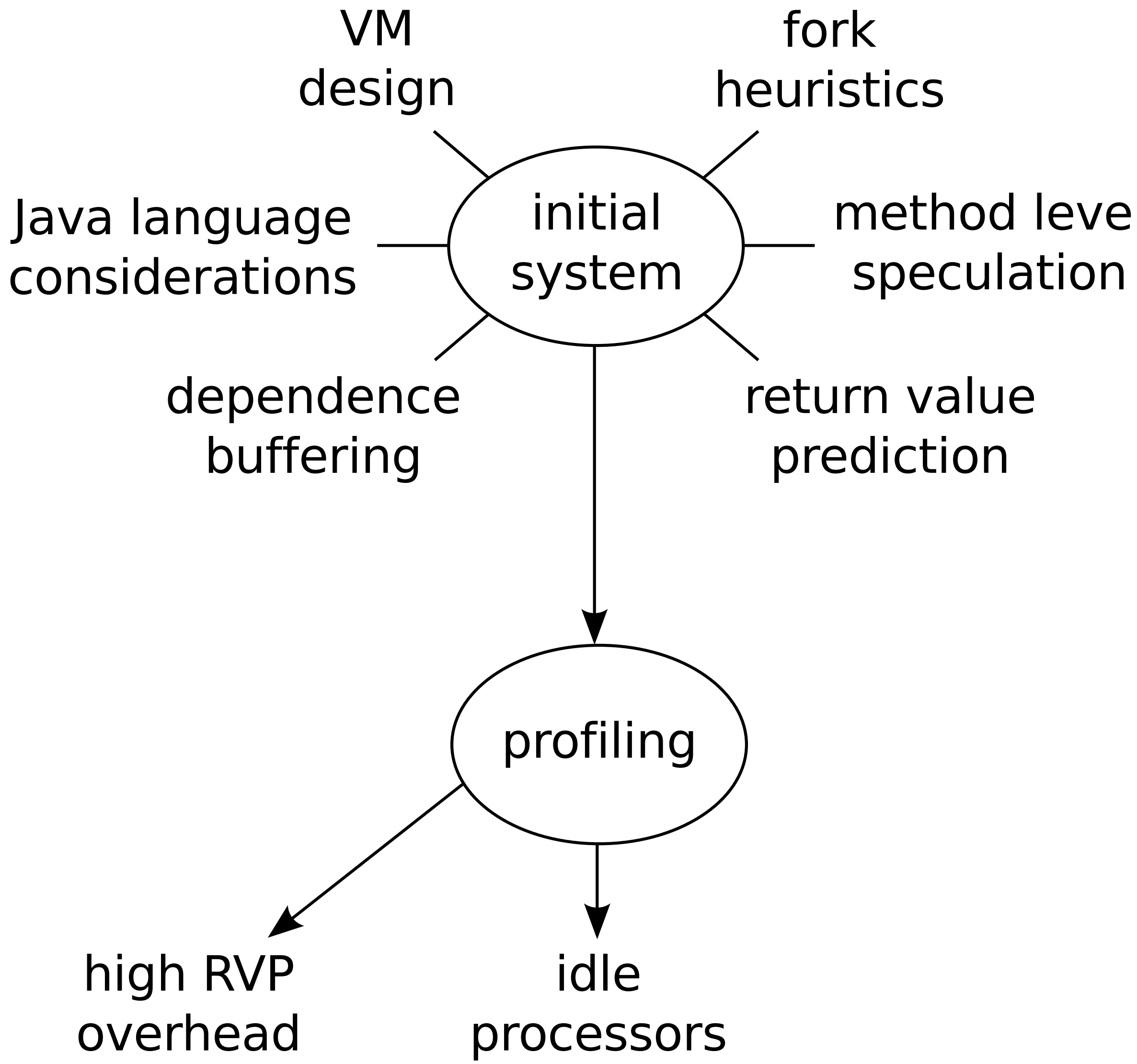


Profiling

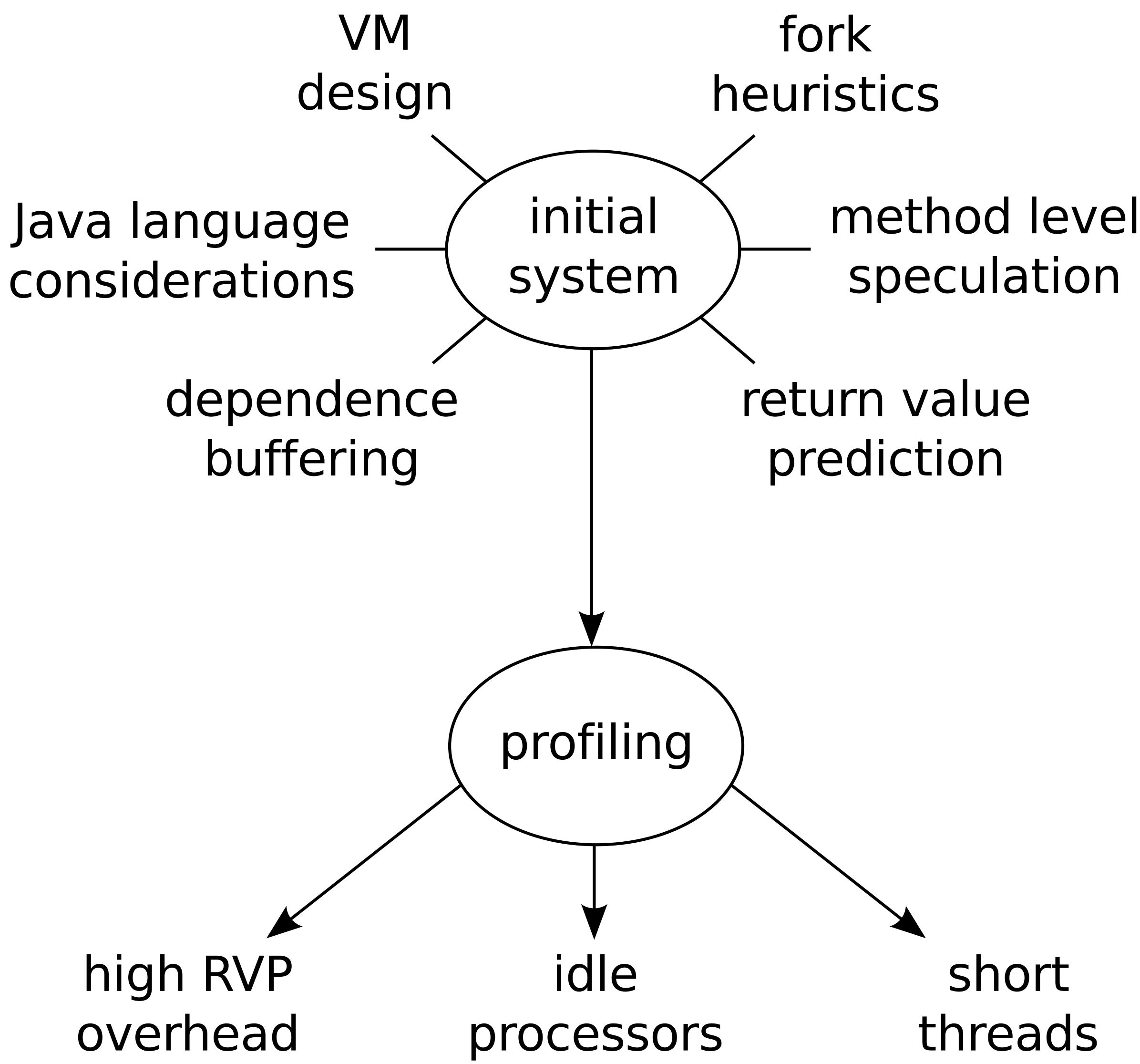


VM design
fork heuristics
Java language considerations
initial system
method level speculation
dependence buffering
return value prediction
profiling
high RVP overhead

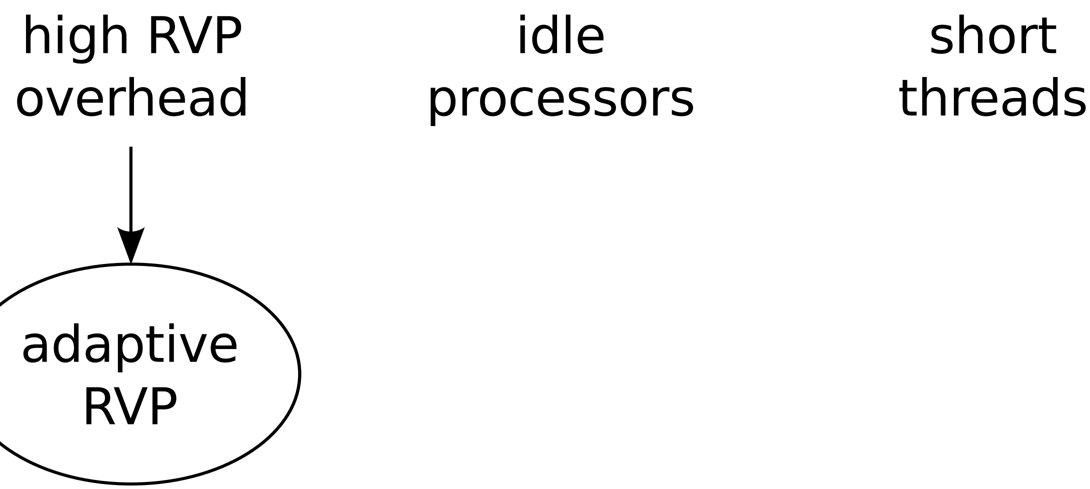
Profiling


VM
design
fork
heuristics
Java language
considerations
initial
system
method level
speculation
dependence
buffering
return value
prediction
profiling
high RVP
overhead
idle
processors

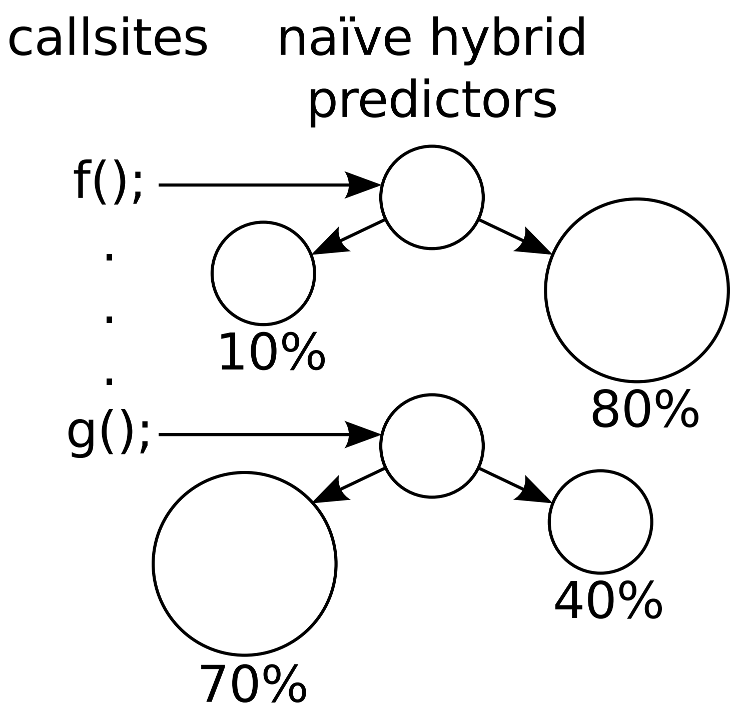
Profiling


VM design
fork heuristics
Java language considerations
initial system
method level speculation
dependence buffering
return value prediction
profiling
high RVP overhead
idle processors
short threads

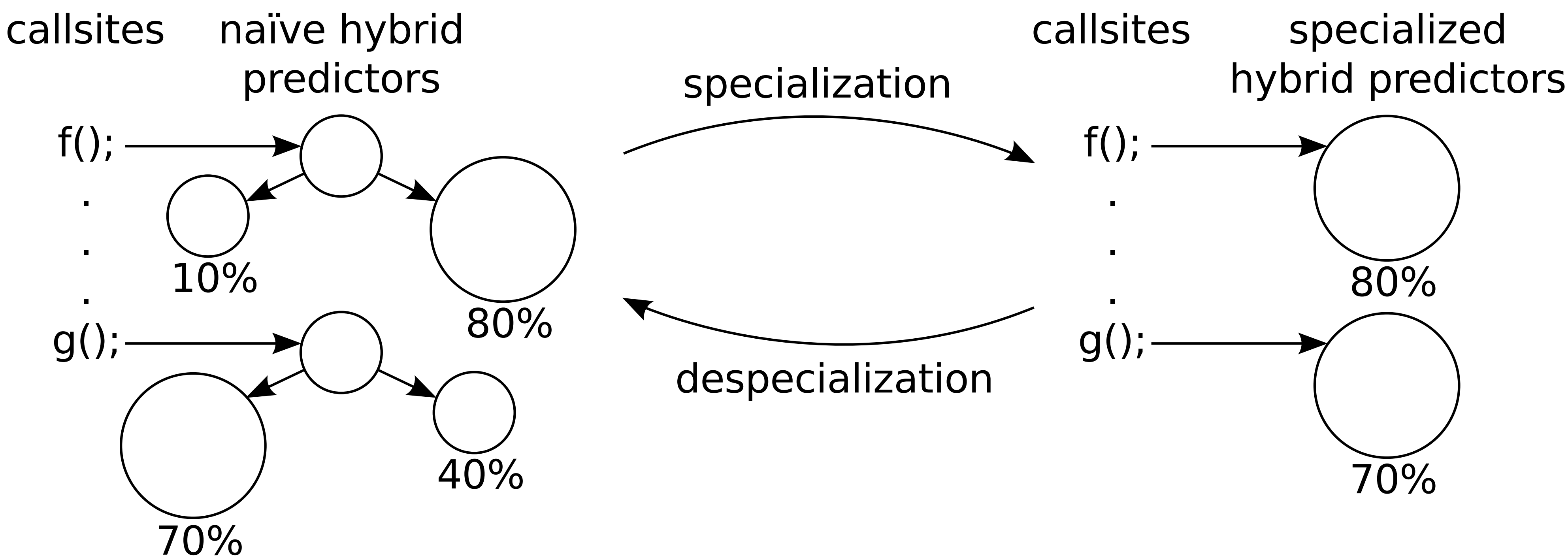
Optimization



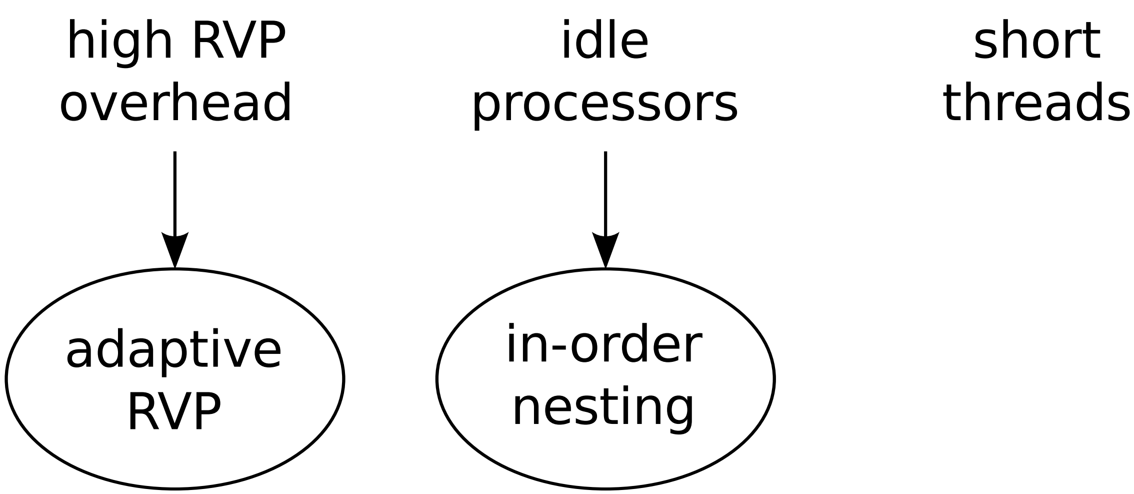
Adaptive RVP



Adaptive RVP

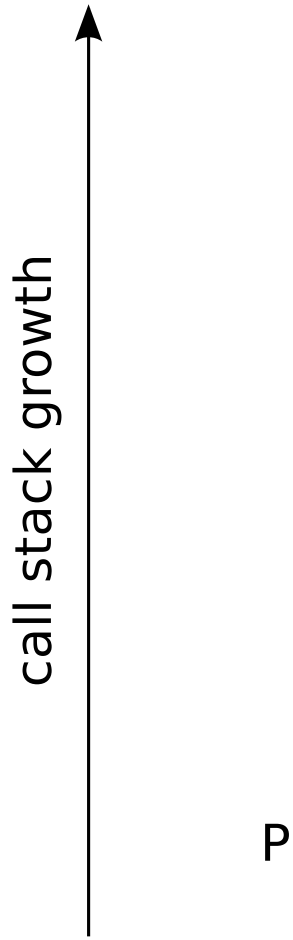


Optimization



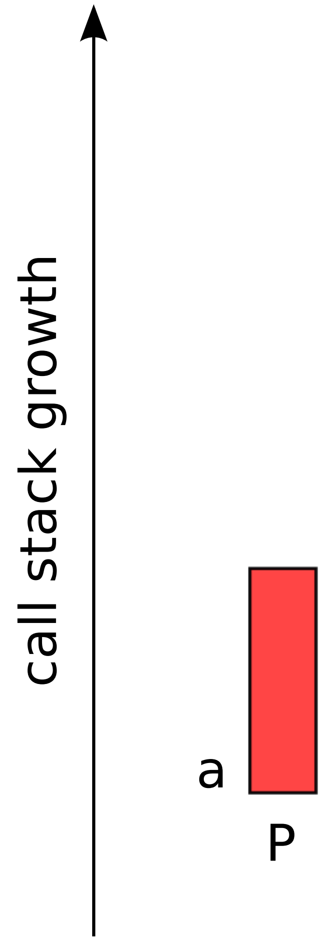
In-Order Nesting

```
A(){
   a;
   B(){
      b;
   }
   a';
   C(){
      c;
   }
   a'';
}
```



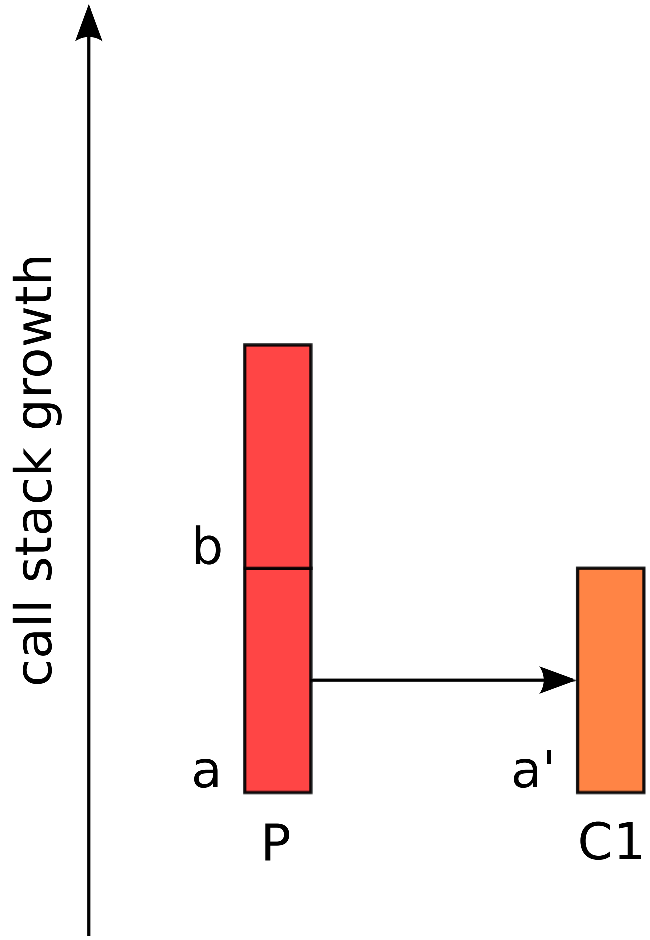
In-Order Nesting

```
A(){
  a;
  B(){
    b;
  }
  a';
  C(){
    c;
  }
  a'';
}
```



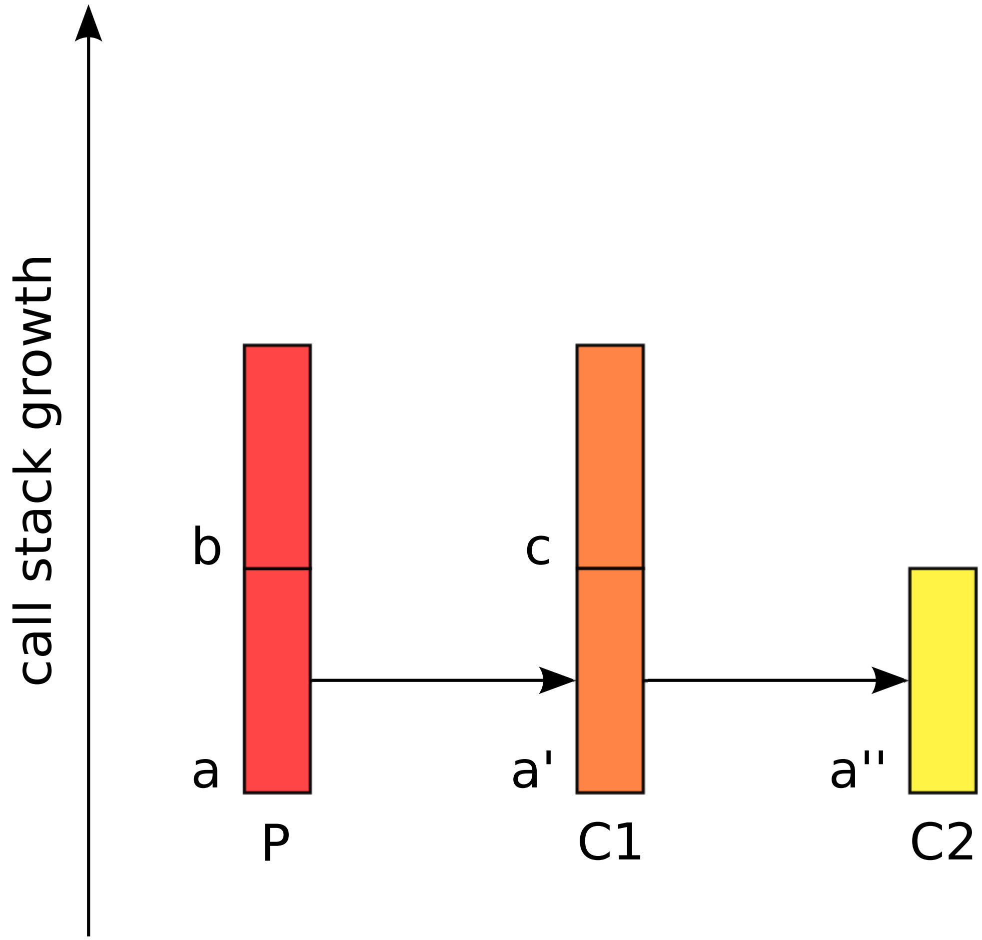
In-Order Nesting

```
A(){
   a;
   B(){
      b;
   }
   a';
   C(){
      c;
   }
   a'';
}
```



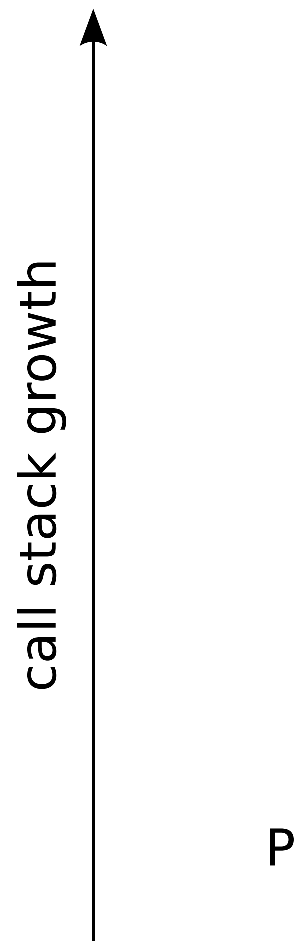
In-Order Nesting

```
A(){
   a;
   B(){
      b;
   }
   a';
   C(){
      c;
   }
   a'';
}
```



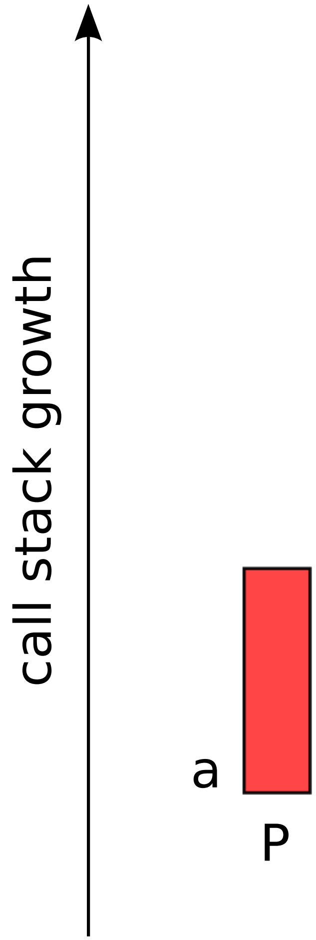
Mixed Nesting

```
A(){
  a;
  B(){
    b;
    C(){
      c;
    }
    b';
    D(){
      d;
    }
    b'';
  }
  a';
}
```



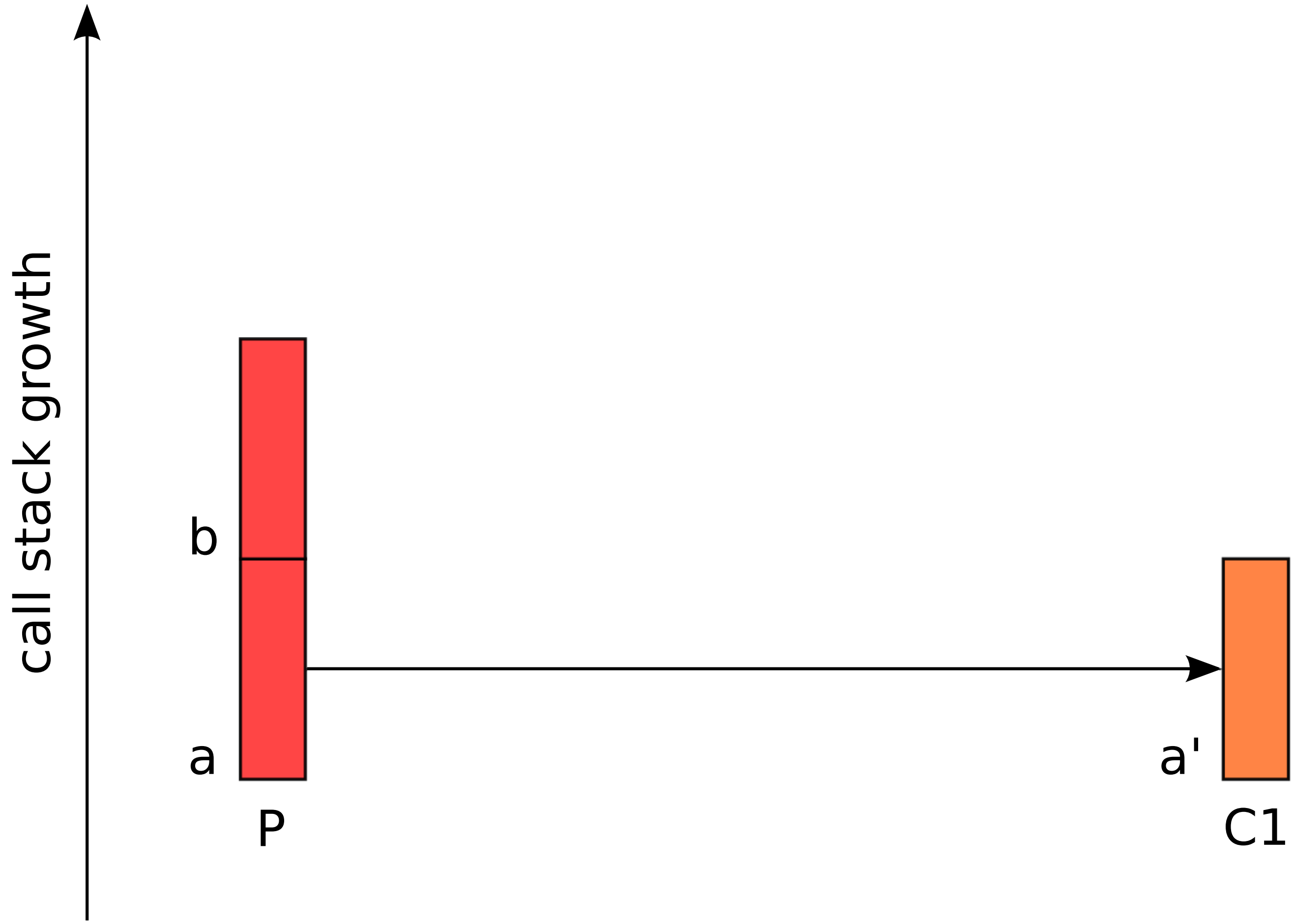
Mixed Nesting

```
A(){
  a;
  B(){
    b;
    C(){
      c;
    }
    b';
    D(){
      d;
    }
    b'';
  }
  a';
}
```



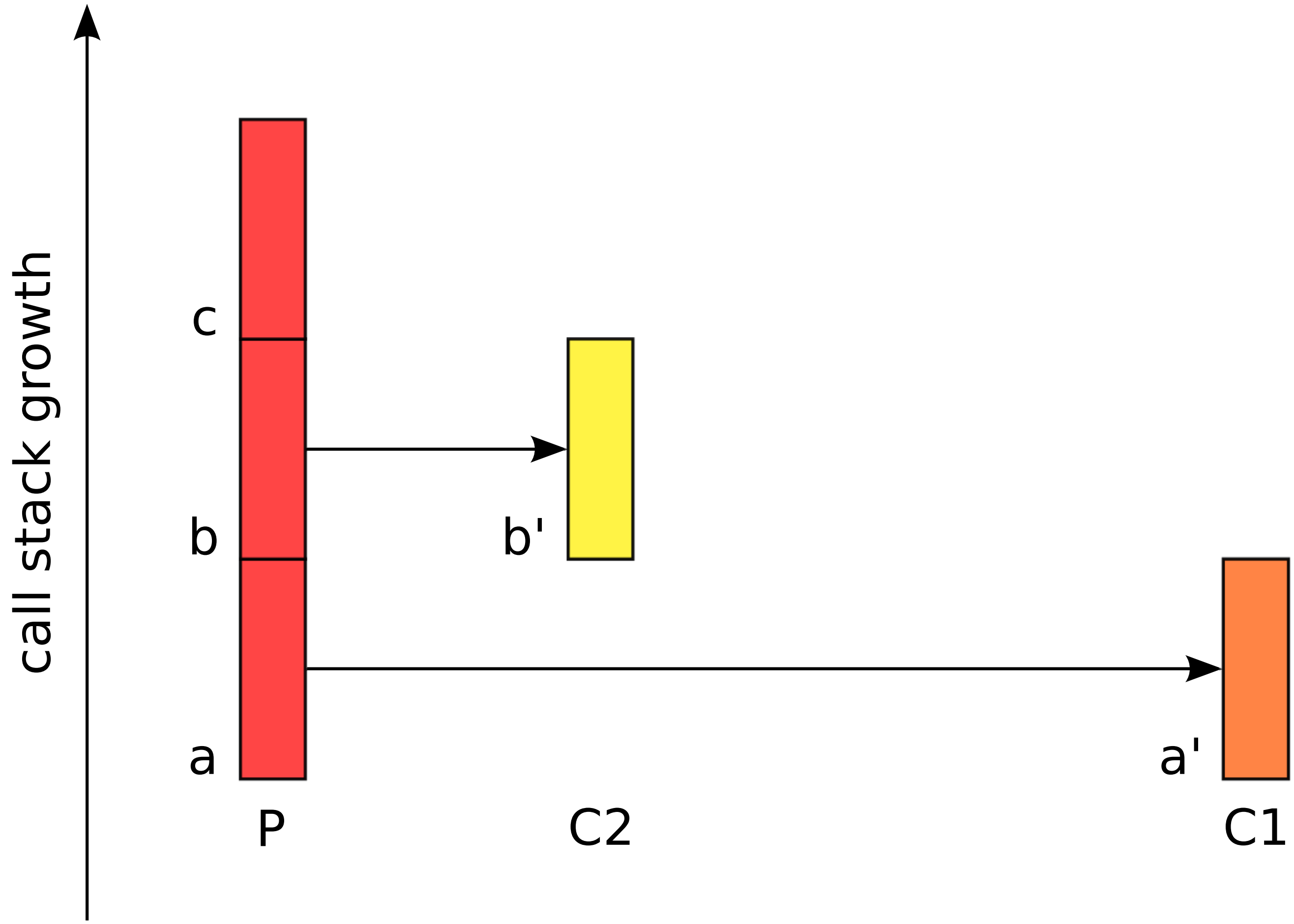
Mixed Nesting

```
A(){
   a;
   B(){
      b;
      C(){
         c;
      }
      b';
      D(){
         d;
      }
      b'';
   }
   a';
}
```



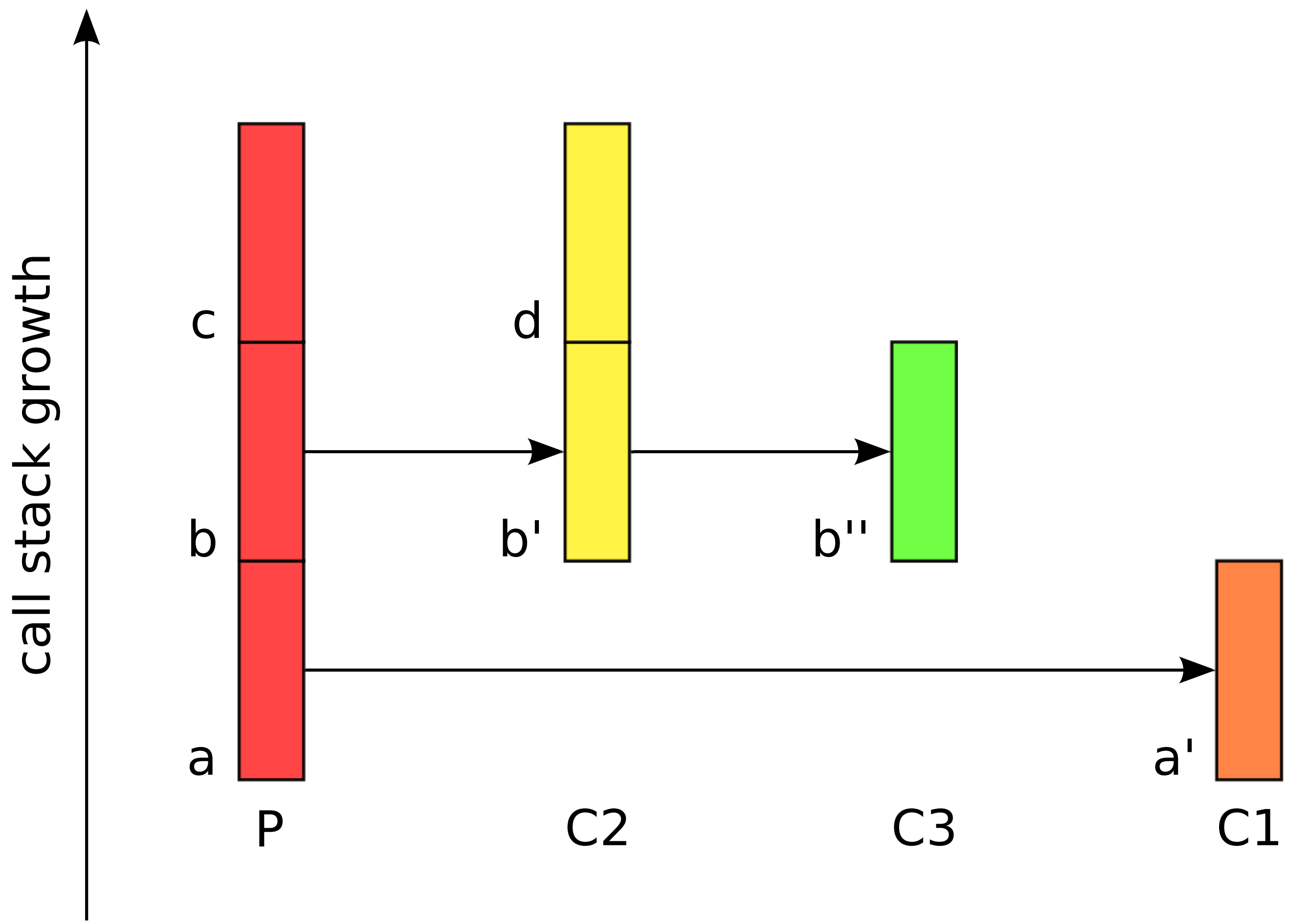
Mixed Nesting

```
A(){
   a;
   B(){
      b;
      C(){
         c;
      }
      b';
      D(){
         d;
      }
      b'';
   }
   a';
}
```

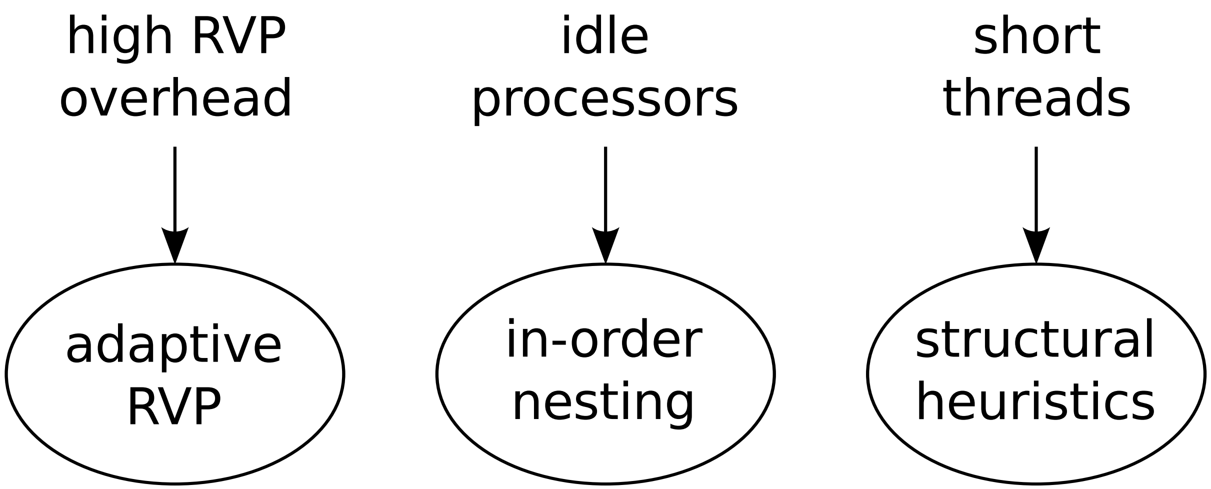


Mixed Nesting

```
A(){
  a;
  B(){
    b;
    C(){
      c;
    }
    b';
    D(){
      d;
    }
    b'';
  }
  a';
}
```



Optimization



Structural Fork Heuristics

```
tail (i, n) {
  work (i);
  if (i < n)
    tail (i + 1, n);
}
```

```
iterate (n) {
  for (i = 1; i <= n; i++)
    work (i);
}
```

```
head (i, n) {
  if (i < n)
    head (i + 1, n);
  work (i);
}
```

```
tree (node *n) {
  if (!leaf (n)) {
    tree (n->left);
    tree (n->right);
  }
  work (n);
}
```

Structural Fork Heuristics

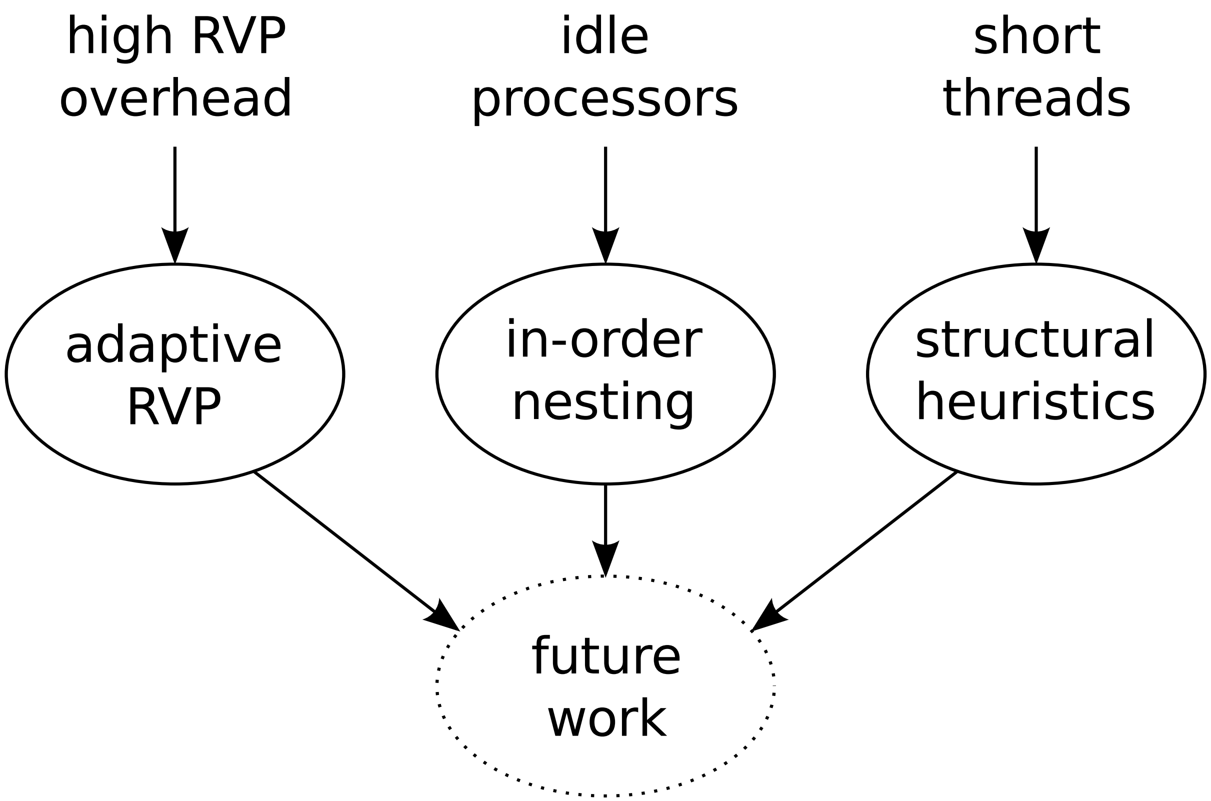
```
// in-order nesting
tail (i, n) {
  spec work (i);
  if (i < n)
    tail (i + 1, n);
}
```

```
// in-order nesting
iterate (n) {
  for (i = 1; i <= n; i++)
    spec work (i);
}
```

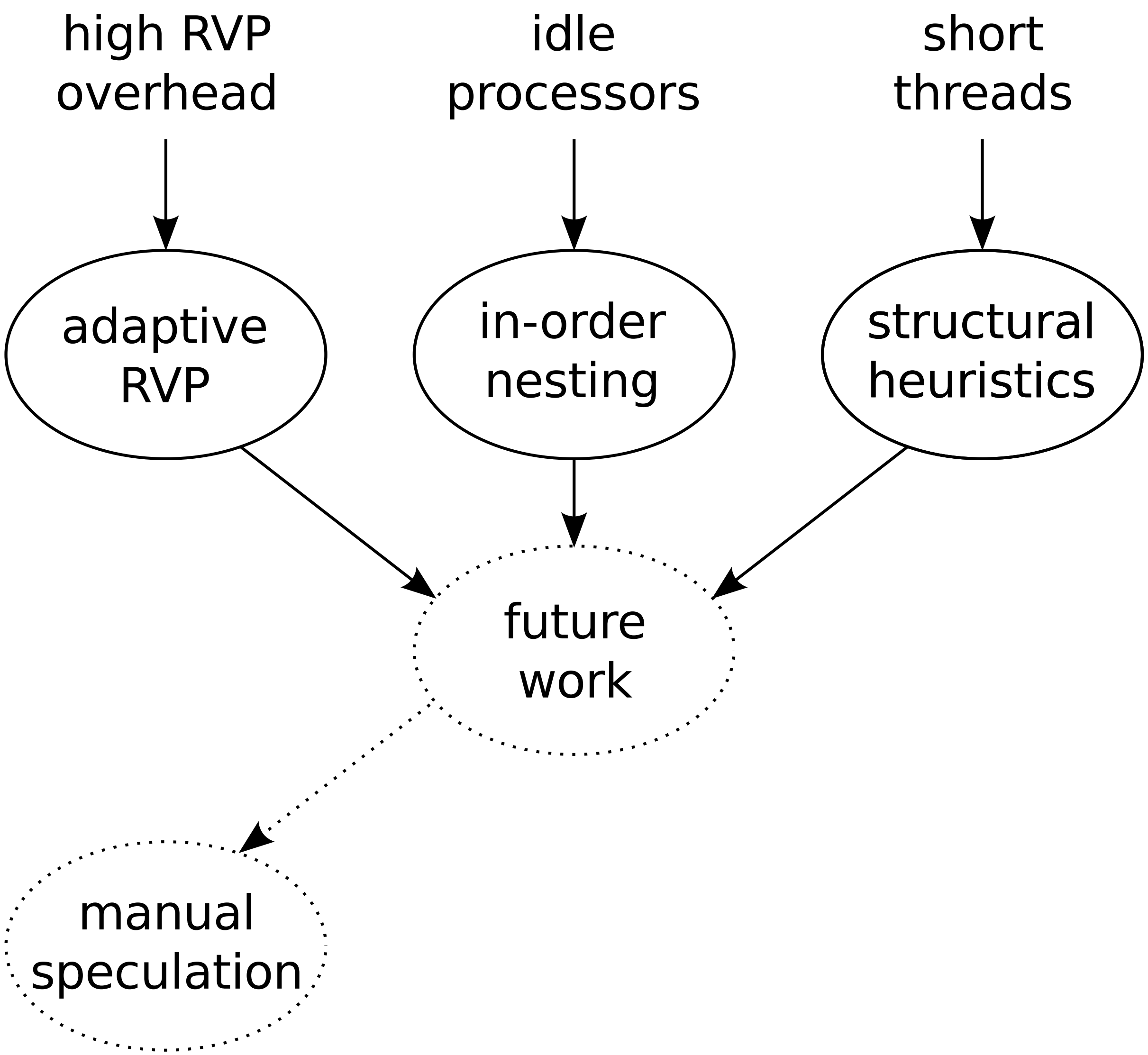
```
// out-of-order nesting
head (i, n) {
  if (i < n)
    spec head (i + 1, n);
  work (i);
}
```

```
// mixed nesting
tree (node *n) {
  if (!leaf (n)) {
    spec tree (n->left);
    spec tree (n->right);
  }
  work (n);
}
```

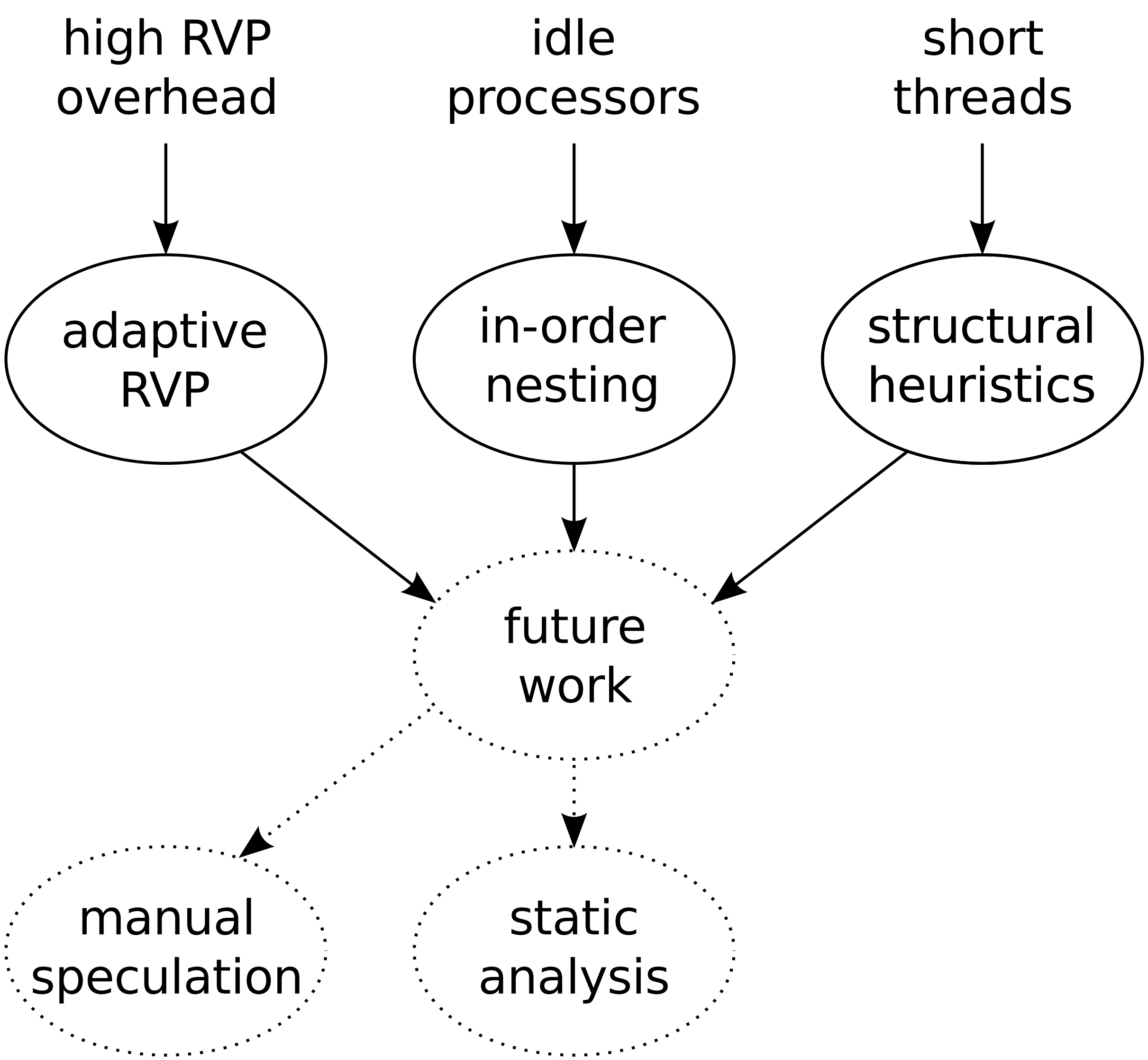
Future Work


high RVP overhead
idle processors
short threads
adaptive RVP
in-order nesting
structural heuristics
future work

Future Work

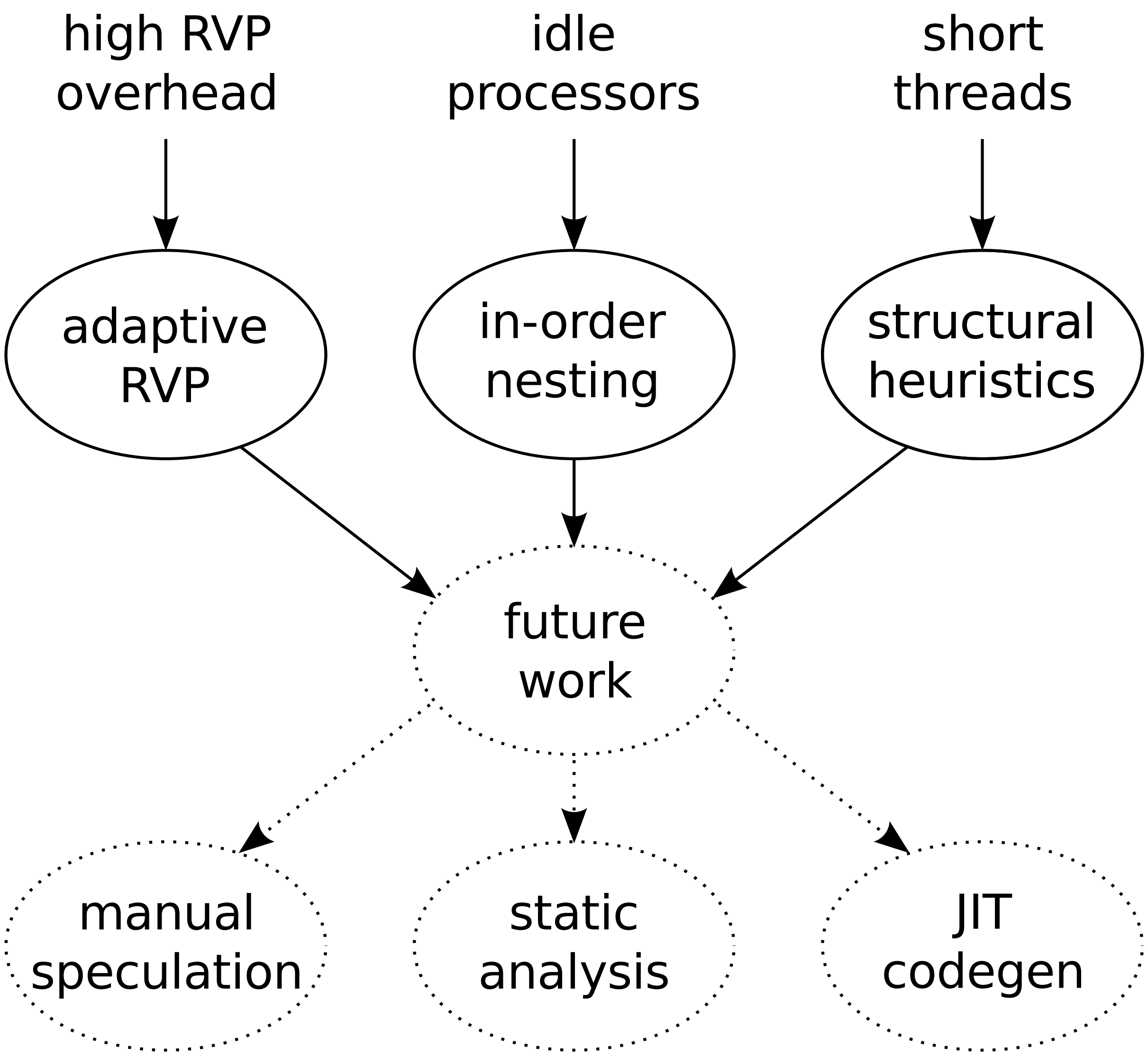

high RVP overhead
idle processors
short threads
adaptive RVP
in-order nesting
structural heuristics
future work
manual speculation

Future Work

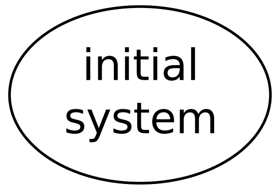


high RVP overhead
idle processors
short threads
adaptive RVP
in-order nesting
structural heuristics
future work
manual speculation
static analysis

Future Work


high RVP overhead
idle processors
short threads
adaptive RVP
in-order nesting
structural heuristics
future work
manual speculation
static analysis
JIT codegen

Conclusions

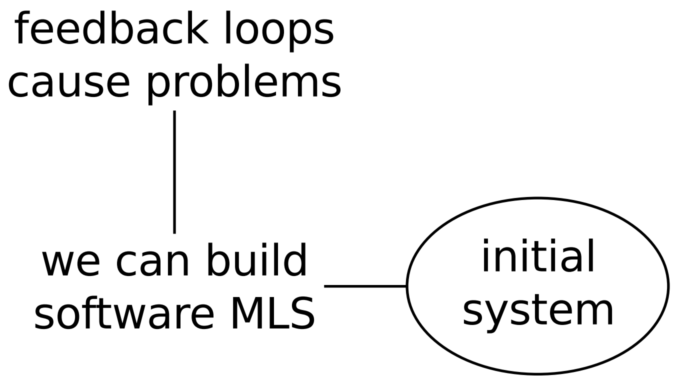


Conclusions

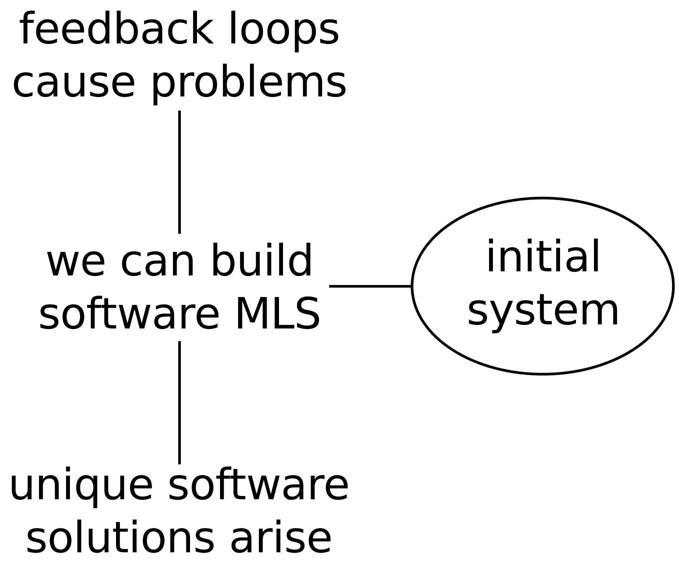
we can build software MLS

initial system

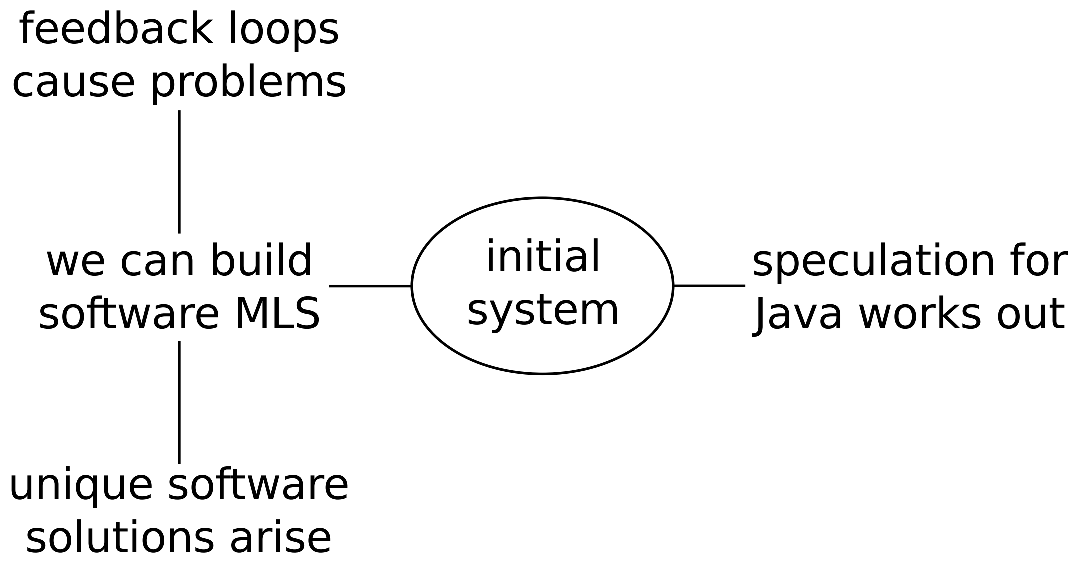
Conclusions



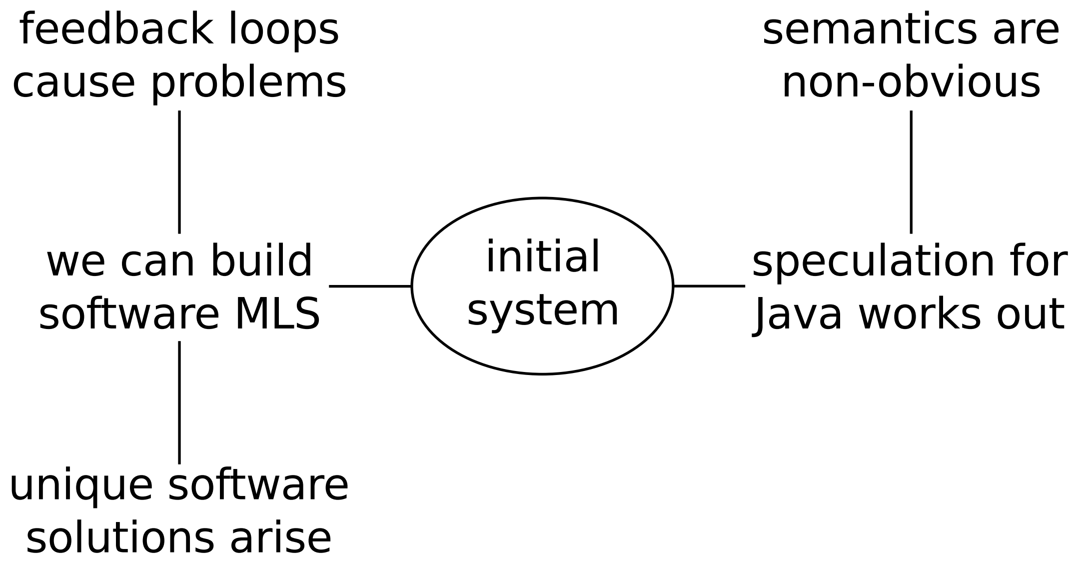
Conclusions



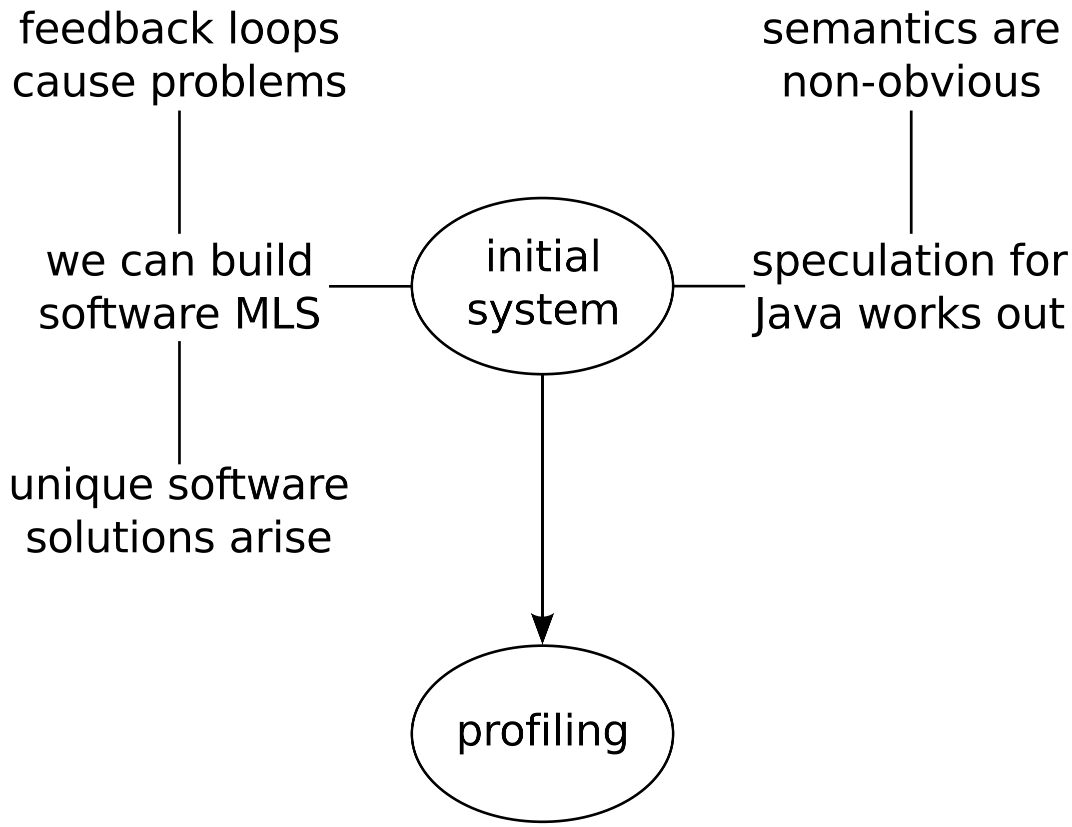
Conclusions



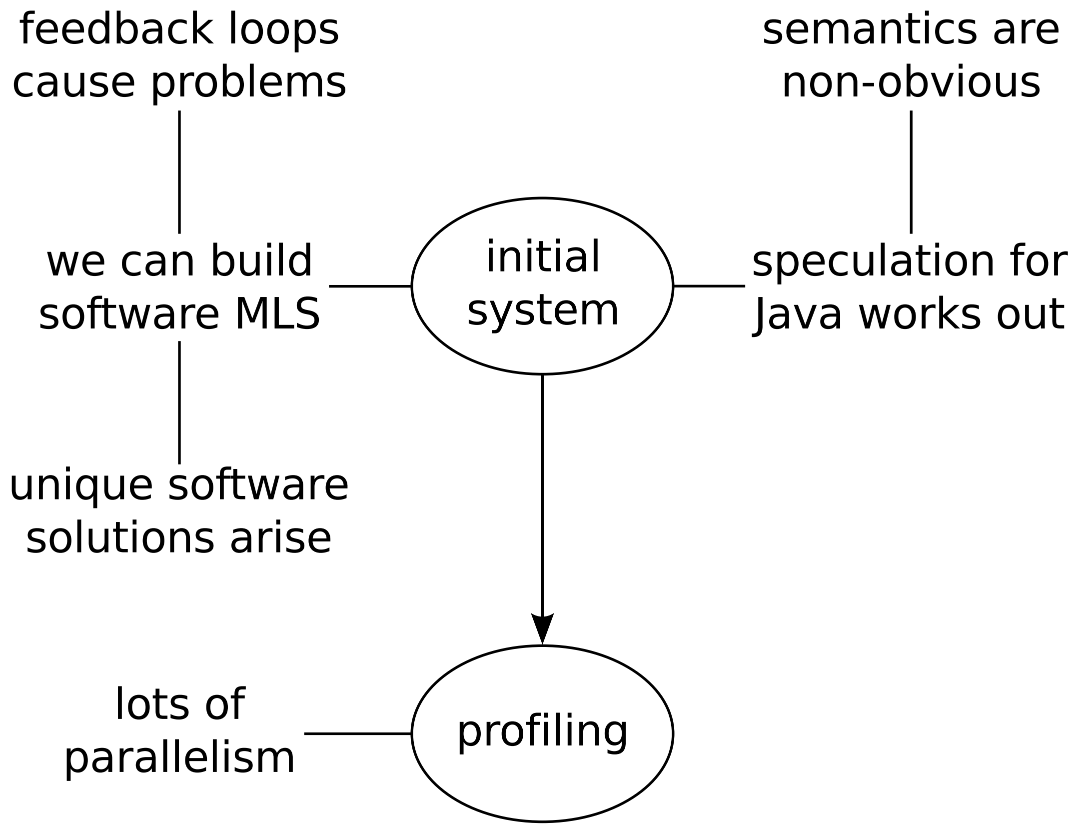
Conclusions



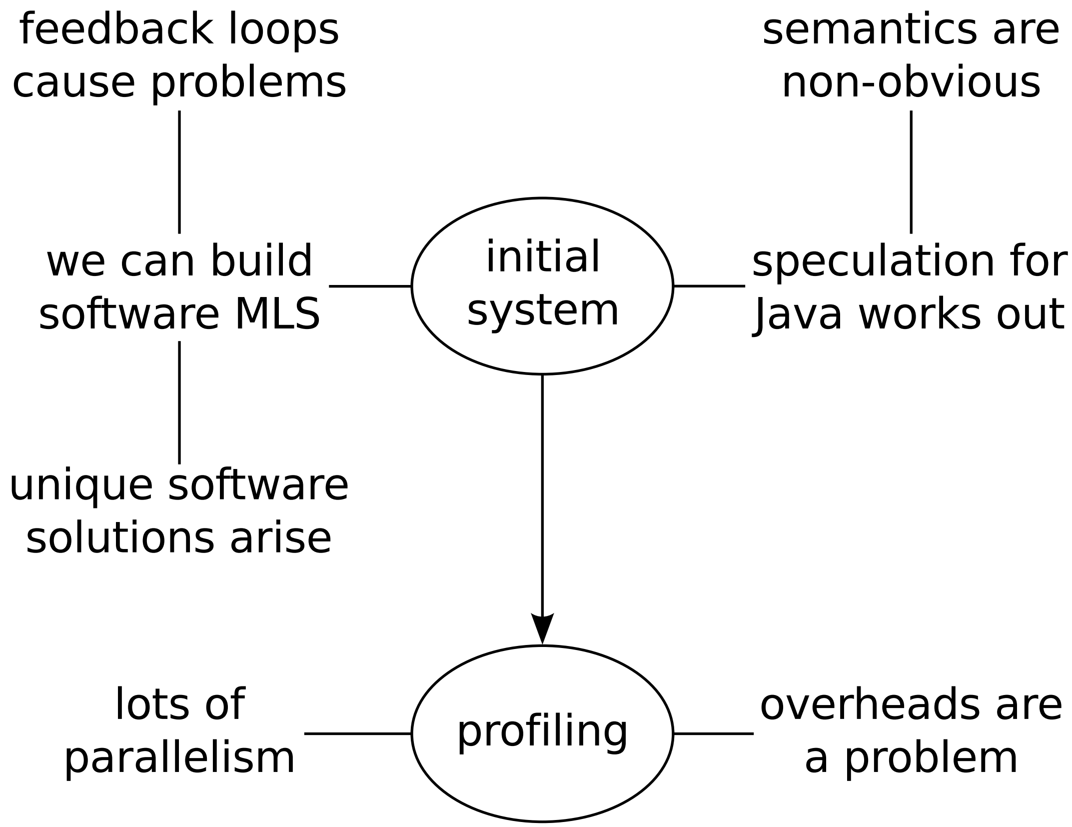
Conclusions



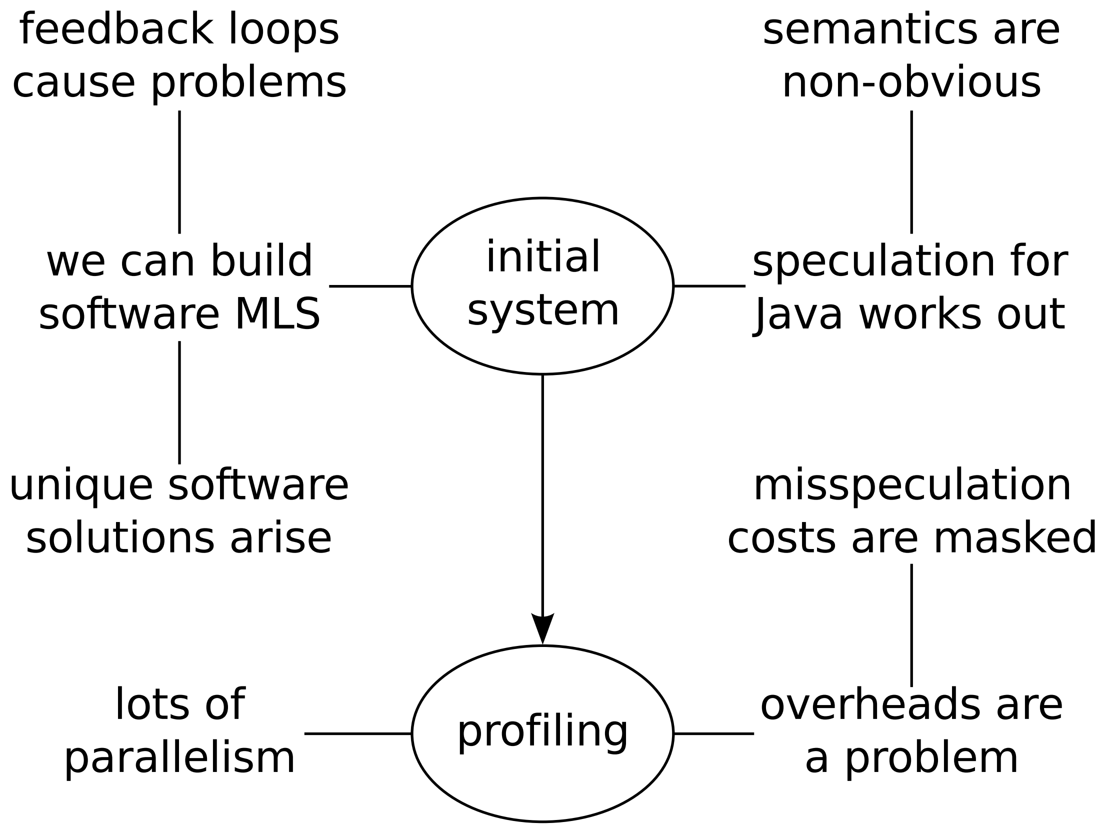
Conclusions



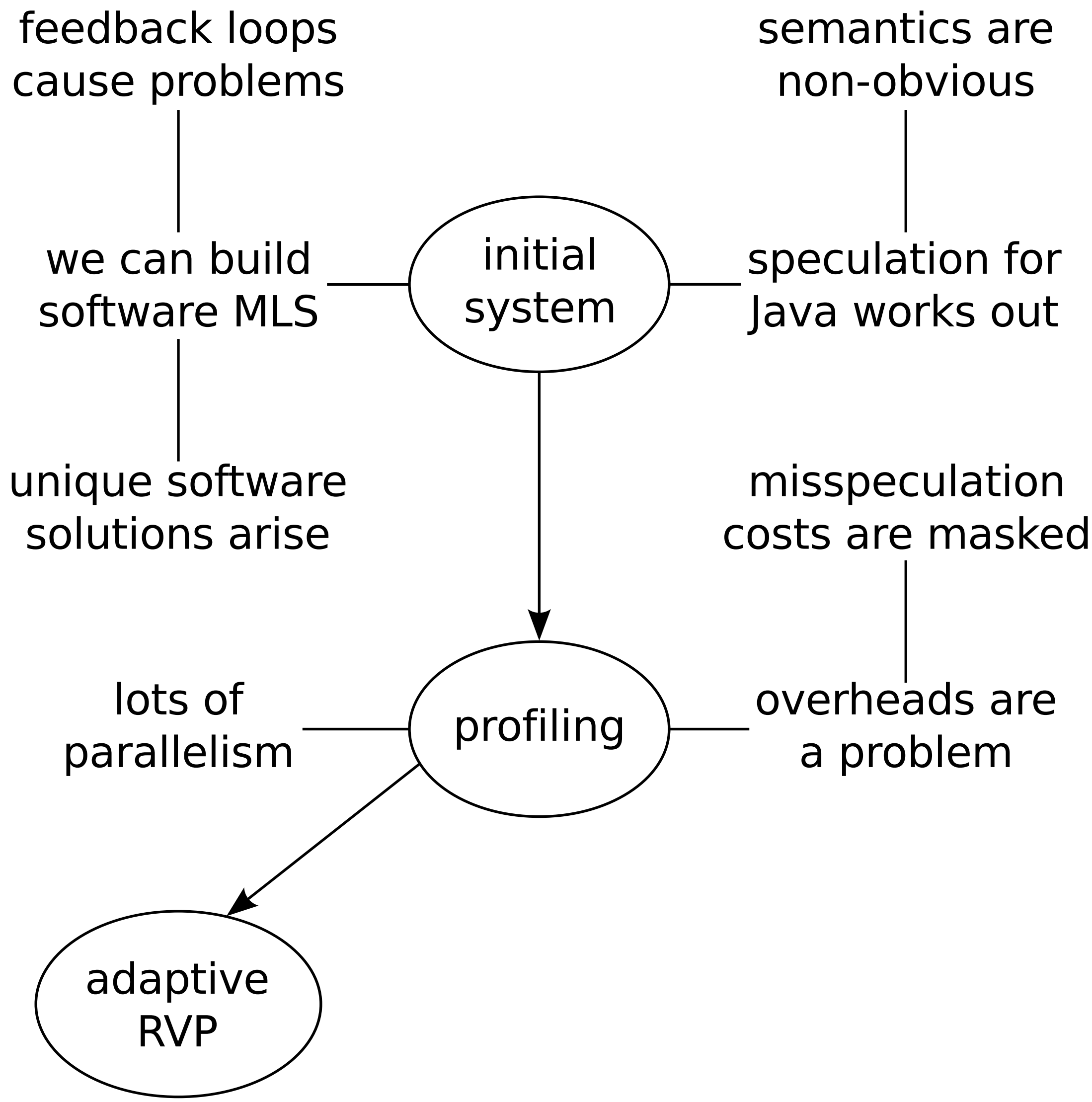
Conclusions



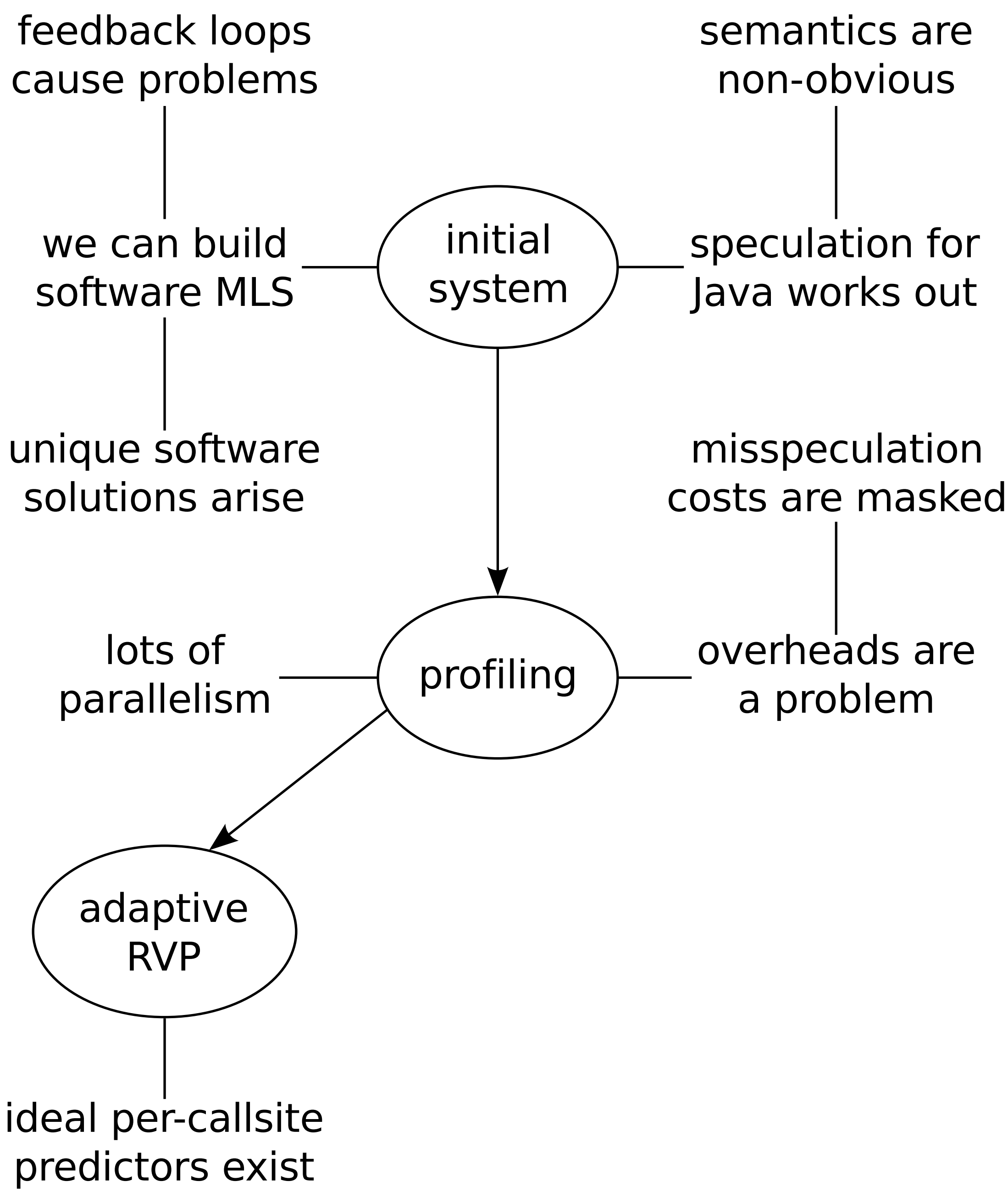
Conclusions



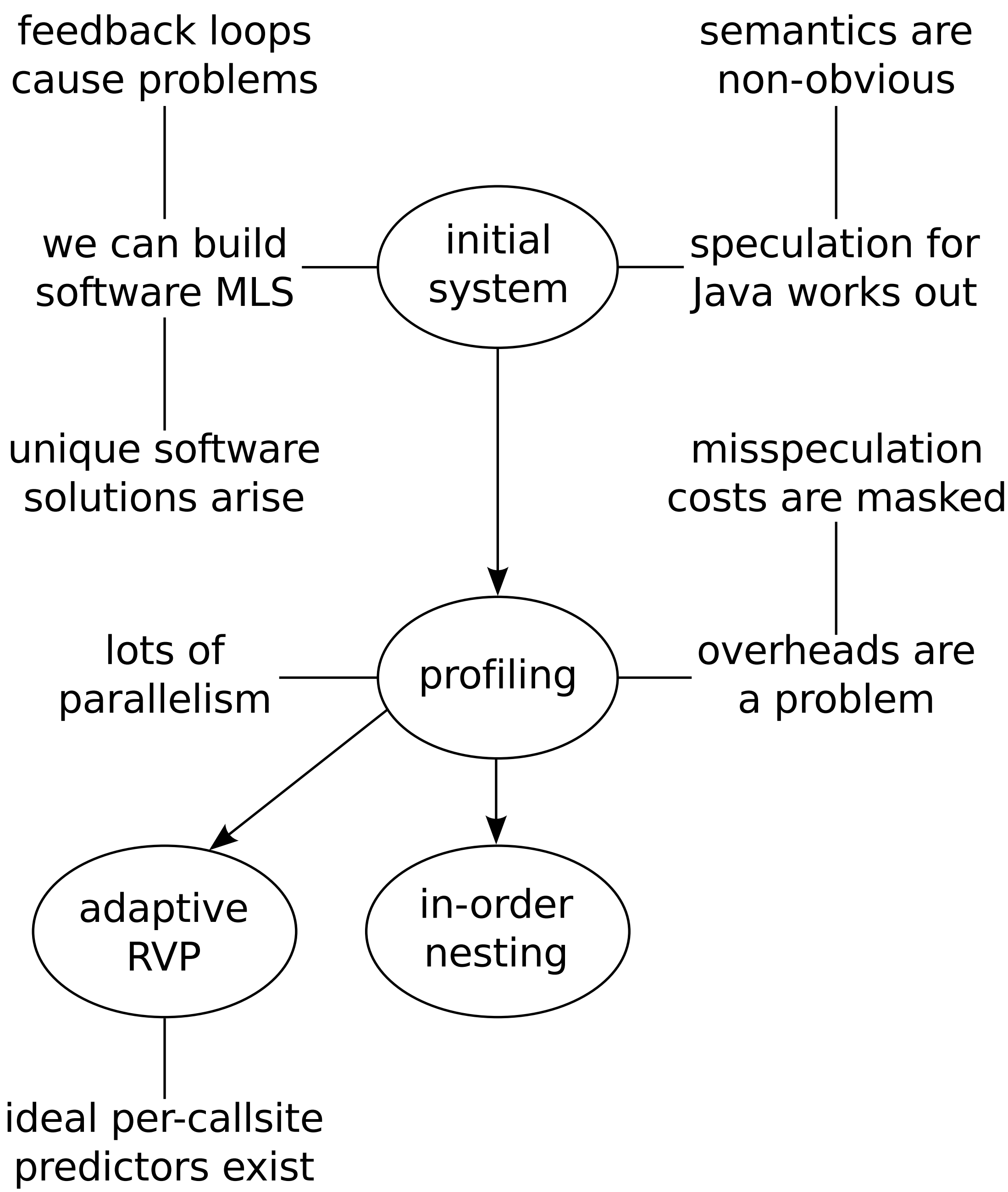
Conclusions



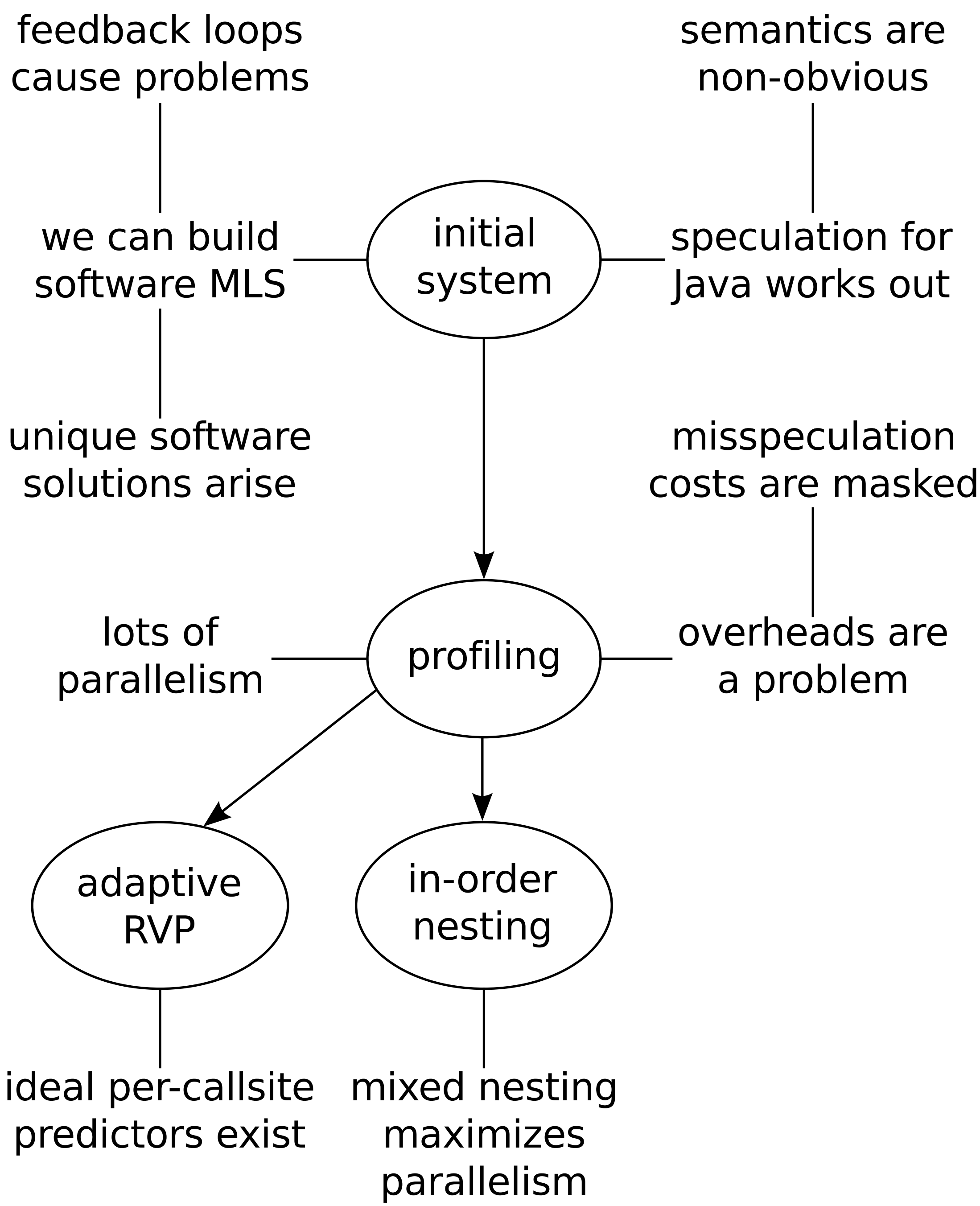
Conclusions



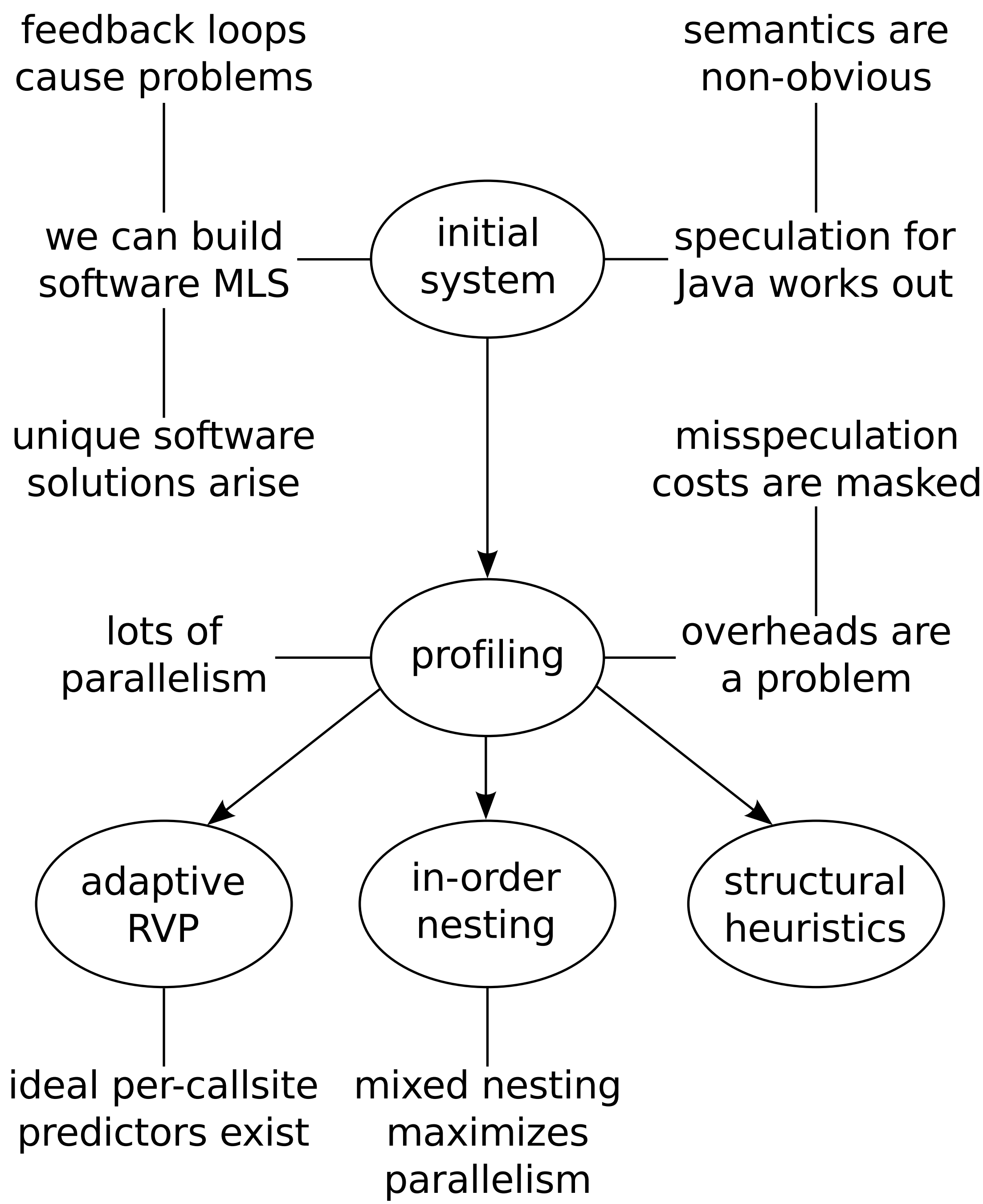
Conclusions



Conclusions



Conclusions



Conclusions