

Directed Synthesis of Failing Concurrent Executions

Malavika Samak*, Omer Tripp⁺, Murali Krishna Ramanathan*

* Indian Institute of Science

+ Google Inc, Mountain View



Multi-threaded libraries are useful ...

Multi-threaded libraries are useful ...

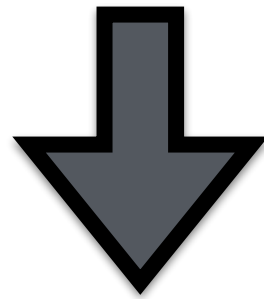
- Provides safety guarantees under concurrency

Multi-threaded libraries are useful ...

- Provides safety guarantees under concurrency
- Ensures performance benefits

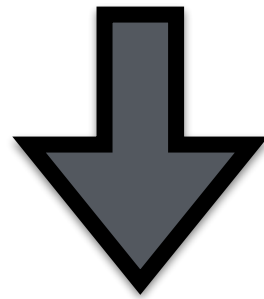
Multi-threaded libraries are useful ...

- Provides safety guarantees under concurrency
- Ensures performance benefits



Multi-threaded libraries are useful ...

- Provides safety guarantees under concurrency
- Ensures performance benefits



Makes building multithreaded applications easier

Building multi-threaded libraries



Building multi-threaded libraries

Class under construction

```
package exam2;

public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults() {
        Full_Name = "No Name Given";
        Exam_Name = "Unknown";
        Exam_Score = "No Score";
        Exam_Grade = "Unknown";
    }

    String fullName(String aName) {
        Full_Name = aName;
        return Full_Name;
    }

    String examName(String examCode) {
        if (examCode.equals("VB")) {
            Exam_Name = "Visual Basic .NET";
        }
        else if (examCode.equals("JV")) {
            Exam_Name = "Java";
        }
        else if (examCode.equals("CF")) {
            Exam_Name = "C# .NET";
        }
        else if (examCode.equals("FW")) {
            Exam_Name = "F#";
        }
        else {
            Exam_Name = "No Exam Selected";
        }
        return Exam_Name;
    }
}
```



Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String aName) {  
        Full_Name = aName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
  
        return Exam_Name;  
    }  
}
```



Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



Need to handle various issues associated with multithreading

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality**
- **Performance**
- **Concurrency**
- **Modularity**
- **Data Structures**
- **Shared state**
- **Lock field correlation**
- ...

Need to handle various issues associated with multithreading

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality**
- **Performance**
- **Concurrency**
- **Modularity**
- **Data Structures**
- **Shared state**
- **Lock field correlation**
- ...

Need to handle various issues associated with multithreading

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality**
- **Performance**
- **Concurrency**
- **Modularity**
- **Data Structures**
- **Shared state**
- **Lock field correlation**
- ...



Need to handle various issues associated with multithreading

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("PW")) {  
            Exam_Name = "PHP";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality**
- **Performance**
- **Concurrency**
- **Modularity**
- **Data Structures**
- **Shared state**
- **Lock field correlation**
- ...



Need to handle various issues associated with multithreading

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality** ✓
- **Performance** ✓
- **Concurrency** ✓
- **Modularity** ✓
- **Data Structures**
- **Shared state**
- **Lock field correlation**
- ...

Need to handle various issues associated with multithreading

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality** ✓
- **Performance** ✓
- **Concurrency** ✓
- **Modularity** ✓
- **Data Structures** ✓
- **Shared state**
- **Lock field correlation**
- ...

Need to handle various issues associated with multithreading

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality** ✓
- **Performance** ✓
- **Concurrency** ✓
- **Modularity** ✓
- **Data Structures** ✓
- **Shared state** ✓
- **Lock field correlation**
- ...

Need to handle various issues associated with multithreading

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality** ✓
- **Performance** ✓
- **Concurrency** ✓
- **Modularity** ✓
- **Data Structures** ✓
- **Shared state** ✓
- **Lock field correlation** ✓
- ... ✓

Need to handle various issues associated with multithreading

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResult {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResult() {  
        Full_Name = "John Doe";  
        Exam_Name = "Math 101";  
        Exam_Score = "85.5%";  
        Exam_Grade = "C+";  
    }  
  
    String fullName() {  
        Full_Name = "John Doe";  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("CV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("PW")) {  
            Exam_Name = "PHP";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
  
        return Exam_Name;  
    }  
}
```



- **Functionality** ❌
- **Performance** ✅
- **Concurrency** ❌
- **Modularity** ✅
- **Data Structures** ✅
- **Shared state** ✅
- **Lock field correlation** ✅
- ... ✅

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "John Doe";  
        Exam_Name = "Math 101";  
        Exam_Score = "85.5%";  
        Exam_Grade = "B+";  
    }  
  
    String fullName() {  
        Full_Name = "John Doe";  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("C#")) {  
            Exam_Name = "C#";  
        }  
        else if (examCode.equals("C++")) {  
            Exam_Name = "C++";  
        }  
        else if (examCode.equals("PHP")) {  
            Exam_Name = "PHP";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality** ❌
- **Performance** ✅
- **Concurrency** ❌
- **Modularity** ✅
- **Data Structures** ✅
- **Shared state** ✅
- **Lock field correlation** ✅
- ... ✅

Simultaneously handling the issues
can introduce bugs

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String aName) {  
        Full_Name = aName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



- **Functionality**
- **Performance**
- **Concurrency**
- **Modularity**
- **Data Structures**
- **Shared state**
- **Lock field correlation**
- ...

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("PW")) {  
            Exam_Name = "PHP";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
  
        return Exam_Name;  
    }  
}
```



- **Functionality**
- **Performance**
- **Concurrency**
- **Modularity**
- **Data Structures**
- **Shared state**
- **Lock field correlation**
- ...

Perform modular testing of various properties

Building multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
  
        return Exam_Name;  
    }  
}
```



- **Functionality**
- **Performance**
- **Concurrency**
- **Modularity**
- **Data Structures**
- **Shared state**
- **Lock field correlation**
- ...

Perform modular testing of various properties

Testing multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String aName) {  
        Full_Name = aName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



Testing multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String aName) {  
        Full_Name = aName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



Testing multi-threaded libraries

Class under construction

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String aName) {  
        Full_Name = aName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```



Testing multi-threaded libraries

Class under construction

```
package exam2;

public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults() {
        Full_Name = "No Name Given";
        Exam_Name = "Unknown";
        Exam_Score = "No Score";
        Exam_Grade = "Unknown";
    }

    String fullName(String fName) {
        Full_Name = fName;
        return Full_Name;
    }

    String examName(String examCode) {
        if (examCode.equals("VB")) {
            Exam_Name = "Visual Basic .NET";
        }
        else if (examCode.equals("JV")) {
            Exam_Name = "Java";
        }
        else if (examCode.equals("CF")) {
            Exam_Name = "C# .NET";
        }
        else if (examCode.equals("PW")) {
            Exam_Name = "PHP";
        }
        else {
            Exam_Name = "No Exam Selected";
        }
        return Exam_Name;
    }
}
```

```
1 class EcoTest
2 {
3     static void genException()
4     {
5         int num[] = new int[4];
6         System.out.println("before exception is generated.");
7         // generate an index out-of-bounds exception
8         num[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11 }
12 class StackTraceBubble
13 {
14     static void myMethod()
15     {
16         try
17         {
18             EcoTest.genException();
19         }
20         catch (ArrayIndexOutOfBoundsException exc)
21         {
22             exc.printStackTrace();
23         }
24     }
25     public static void main(String[] args)
26     {
27         try
28         {
29             StackTraceBubble.myMethod();
30             EcoTest.genException();
31         }
32         catch (ArrayIndexOutOfBoundsException exc)
33         {
34             // catch the exception
35             System.out.println("Standard message is: ");
36             System.out.println(exc);
37             System.out.println("\nStack trace: ");
38             exc.printStackTrace();
39         }
40         System.out.println("After catch statement.");
41     }
42 }
```

Client



Testing multi-threaded libraries

Class under construction

```
package exam2;

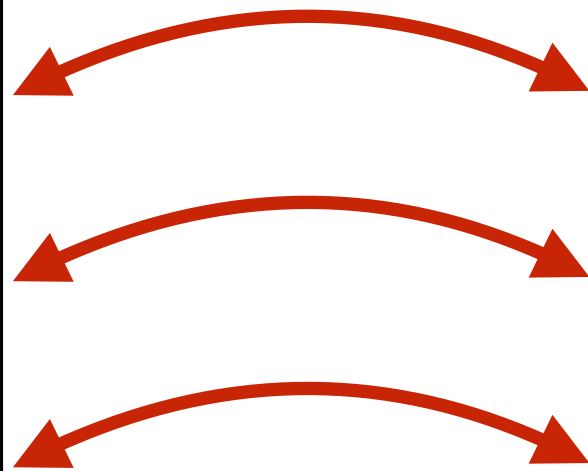
public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults() {
        Full_Name = "No Name Given";
        Exam_Name = "Unknown";
        Exam_Score = "No Score";
        Exam_Grade = "Unknown";
    }

    String fullName(String fName) {
        Full_Name = fName;
        return Full_Name;
    }

    String examName(String examCode) {
        if (examCode.equals("VB")) {
            Exam_Name = "Visual Basic .NET";
        }
        else if (examCode.equals("JV")) {
            Exam_Name = "Java";
        }
        else if (examCode.equals("CF")) {
            Exam_Name = "C# .NET";
        }
        else if (examCode.equals("PW")) {
            Exam_Name = "PHP";
        }
        else {
            Exam_Name = "No Exam Selected";
        }
        return Exam_Name;
    }
}
```



```
1 class EcoTest
2 {
3     static void genException()
4     {
5         int num[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         num[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11 }
12
13 class StackTraceBubble
14 {
15     static void myMethod()
16     {
17         try
18         {
19             EcoTest.genException();
20         }
21         catch (ArrayIndexOutOfBoundsException exc)
22         {
23             exc.printStackTrace();
24         }
25     }
26
27     public static void main(String[] args)
28     {
29         try
30         {
31             StackTraceBubble.myMethod();
32             EcoTest.genException();
33         }
34         catch (ArrayIndexOutOfBoundsException exc)
35         {
36             // catch the exception
37             System.out.println("Standard message is: ");
38             System.out.println(exc);
39             System.out.println("\nStack trace: ");
40             exc.printStackTrace();
41         }
42         System.out.println("After catch statement.");
43     }
44 }
```

Client



Testing multi-threaded libraries

- Invokes the APIs provided by the class.

**Class u
constru**

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CF")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("FW")) {  
            Exam_Name = "F#";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
        return Exam_Name;  
    }  
}
```

```
1 class EcoTest {  
2     static void genException() {  
3         int sum[] = new int[4];  
4         System.out.println("Before exception is generated.");  
5         // generate an index out-of-bounds exception  
6         sum[7] = 10;  
7         System.out.println("this won't be displayed");  
8     }  
9 }  
10  
11 class StackTraceBubble {  
12     static void myMethod() {  
13         try {  
14             EcoTest.genException();  
15         }  
16         catch (ArrayIndexOutOfBoundsException exc) {  
17             exc.printStackTrace();  
18         }  
19     }  
20  
21     public static void main(String[] args) {  
22         try {  
23             StackTraceBubble.myMethod();  
24             EcoTest.genException();  
25         }  
26         catch (ArrayIndexOutOfBoundsException exc) {  
27             // catch the exception  
28             System.out.println("Standard message is: ");  
29             System.out.println(exc);  
30             System.out.println("Stack trace: ");  
31             exc.printStackTrace();  
32         }  
33         System.out.println("After catch statement.");  
34     }  
35 }
```

Client



Testing multi-threaded libraries

Class under construction

- Invokes the APIs provided by the class.
- Exposes the broken contracts in the library.

```
package exam2;

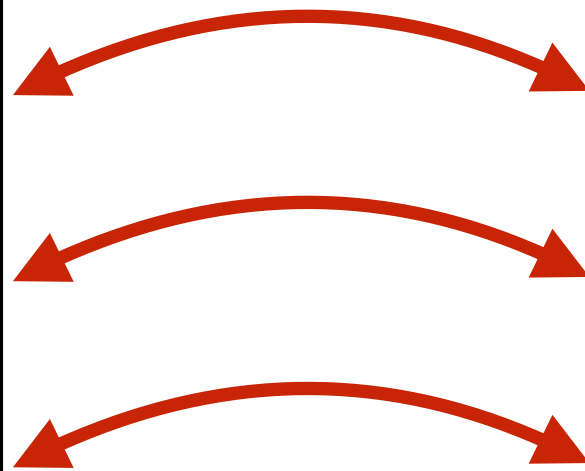
public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults() {
        Full_Name = "No Name Given";
        Exam_Name = "Unknown";
        Exam_Score = "No Score";
        Exam_Grade = "Unknown";
    }

    String fullName(String fName) {
        Full_Name = fName;
        return Full_Name;
    }

    String examName(String examCode) {
        if (examCode.equals("VB")) {
            Exam_Name = "Visual Basic .NET";
        }
        else if (examCode.equals("JV")) {
            Exam_Name = "Java";
        }
        else if (examCode.equals("CF")) {
            Exam_Name = "C# .NET";
        }
        else if (examCode.equals("FW")) {
            Exam_Name = "F#";
        }
        else {
            Exam_Name = "No Exam Selected";
        }
        return Exam_Name;
    }
}
```



```
1 class Ecotest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11 }
12
13 class StackTraceBubble
14 {
15     static void myMethod()
16     {
17         try
18         {
19             Ecotest.genException();
20         }
21         catch (ArrayIndexOutOfBoundsException exc)
22         {
23             exc.printStackTrace();
24         }
25     }
26
27     public static void main(String[] args)
28     {
29         try
30         {
31             StackTraceBubble.myMethod();
32             Ecotest.genException();
33         }
34         catch (ArrayIndexOutOfBoundsException exc)
35         {
36             // catch the exception
37             System.out.println("Standard message is: ");
38             System.out.println(exc);
39             System.out.println("\nStack trace: ");
40             exc.printStackTrace();
41         }
42         System.out.println("After catch statement.");
43     }
44 }
```

Client



Testing multi-threaded libraries

Class under construction

- Invokes the APIs provided by the class.
- Exposes the broken contracts in the library.
- Facilitates regression testing.

```
package exam2;

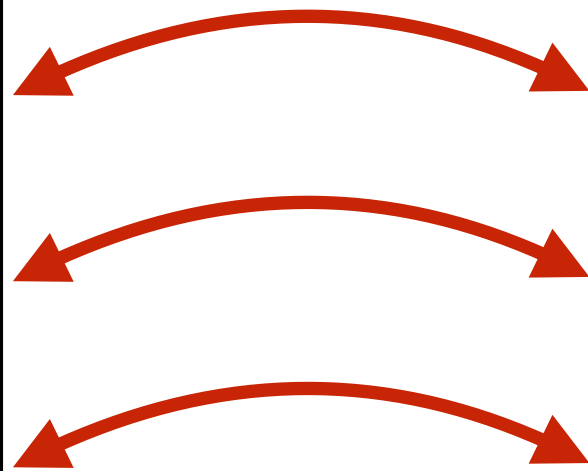
public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults() {
        Full_Name = "No Name Given";
        Exam_Name = "Unknown";
        Exam_Score = "No Score";
        Exam_Grade = "Unknown";
    }

    String fullName(String fName) {
        Full_Name = fName;
        return Full_Name;
    }

    String examName(String examCode) {
        if (examCode.equals("VB")) {
            Exam_Name = "Visual Basic .NET";
        }
        else if (examCode.equals("JV")) {
            Exam_Name = "Java";
        }
        else if (examCode.equals("CF")) {
            Exam_Name = "C# .NET";
        }
        else if (examCode.equals("PW")) {
            Exam_Name = "PHP";
        }
        else {
            Exam_Name = "No Exam Selected";
        }
        return Exam_Name;
    }
}
```



```
1 class Ecotest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11 }
12
13 class StackTraceBubble
14 {
15     static void myMethod()
16     {
17         try
18         {
19             Ecotest.genException();
20         }
21         catch (ArrayIndexOutOfBoundsException exc)
22         {
23             exc.printStackTrace();
24         }
25     }
26
27     public static void main(String[] args)
28     {
29         try
30         {
31             StackTraceBubble.myMethod();
32             Ecotest.genException();
33         }
34         catch (ArrayIndexOutOfBoundsException exc)
35         {
36             // catch the exception
37             System.out.println("Standard message is: ");
38             System.out.println(exc);
39             System.out.println("\nStack trace: ");
40             exc.printStackTrace();
41         }
42         System.out.println("After catch statement.");
43     }
44 }
```

Client



Testing multi-threaded libraries

```
1 class ExcTest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11 }
12
13 class StackTraceBubble
14 {
15     static void myMethod()
16     {
17         try
18         {
19             ExcTest.genException();
20         }
21         catch (ArrayIndexOutOfBoundsException exc)
22         {
23             exc.printStackTrace();
24         }
25     }
26
27     public static void main(String[] args)
28     {
29         try
30         {
31             StackTraceBubble.myMethod();
32             ExcTest.genException();
33         }
34         catch (ArrayIndexOutOfBoundsException exc)
35         {
36             // catch the exception
37             System.out.println("Standard message is: ");
38             System.out.println(exc);
39             System.out.println("\nStack traces ");
40             exc.printStackTrace();
41         }
42         System.out.println("After catch statement.");
43     }
44 }
```

Client



Developer

Testing multi-threaded libraries

```
1 class RecTest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11 }
12 class StackTraceBubble
13 {
14     static void myMethod()
15     {
16         try
17         {
18             RecTest.genException();
19         }
20         catch (ArrayIndexOutOfBoundsException eoc)
21         {
22             eoc.printStackTrace();
23         }
24     }
25     public static void main(String[] args)
26     {
27         try
28         {
29             StackTraceBubble.myMethod();
30             RecTest.genException();
31         }
32         catch (ArrayIndexOutOfBoundsException eoc)
33         {
34             // catch the exception
35             System.out.println("Standard message is: ");
36             System.out.println(eoc);
37             System.out.println("no stack trace");
38             eoc.printStackTrace();
39         }
40         System.out.println("After catch statement.");
41     }
42 }
```



Client

Developer

Testing multi-threaded libraries

```
1 class RecTest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11 }
12 class StackTraceBubble
13 {
14     static void myMethod()
15     {
16         try
17         {
18             RecTest.genException();
19         }
20         catch (ArrayIndexOutOfBoundsException eoc)
21         {
22             eoc.printStackTrace();
23         }
24     }
25     public static void main(String[] args)
26     {
27         try
28         {
29             StackTraceBubble.myMethod();
30             RecTest.genException();
31         }
32         catch (ArrayIndexOutOfBoundsException eoc)
33         {
34             // catch the exception
35             System.out.println("Standard message is: ");
36             System.out.println(eoc);
37             System.out.println("no stack traces");
38             eoc.printStackTrace();
39         }
40         System.out.println("After catch statement.");
41     }
42 }
```

Client



Developer

Testing multi-threaded libraries

```
1 class RecTest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11 }
12 class StackTraceBubble
13 {
14     static void myMethod()
15     {
16         try
17         {
18             RecTest.genException();
19         }
20         catch (ArrayIndexOutOfBoundsException eoc)
21         {
22             eoc.printStackTrace();
23         }
24     }
25     public static void main(String[] args)
26     {
27         try
28         {
29             StackTraceBubble.myMethod();
30             RecTest.genException();
31         }
32         catch (ArrayIndexOutOfBoundsException eoc)
33         {
34             // catch the exception
35             System.out.println("Standard message is: ");
36             System.out.println(eoc);
37             System.out.println("noStack trace");
38             eoc.printStackTrace();
39         }
40         System.out.println("After catch statement.");
41     }
42 }
```



Client

Developer

Challenges

Testing multi-threaded libraries

```
1 class RecTest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11 }
12 class StackTraceBubble
13 {
14     static void myMethod()
15     {
16         try
17         {
18             RecTest.genException();
19         }
20         catch (ArrayIndexOutOfBoundsException eoc)
21         {
22             eoc.printStackTrace();
23         }
24     }
25     public static void main(String[] args)
26     {
27         try
28         {
29             StackTraceBubble.myMethod();
30             RecTest.genException();
31         }
32         catch (ArrayIndexOutOfBoundsException eoc)
33         {
34             // catch the exception
35             System.out.println("Standard message is: ");
36             System.out.println(eoc);
37             System.out.println("stack trace ");
38             eoc.printStackTrace();
39         }
40         System.out.println("After catch statement.");
41     }
42 }
```

Client



Developer

How many **threads** need to be **created**?

Challenges

Testing multi-threaded libraries

```
1 class RecTest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("This won't be displayed");
10    }
11 }
12 class StackTraceBubble
13 {
14     static void myMethod()
15     {
16         try
17         {
18             RecTest.genException();
19         }
20         catch (ArrayIndexOutOfBoundsException eoc)
21         {
22             eoc.printStackTrace();
23         }
24     }
25     public static void main(String[] args)
26     {
27         try
28         {
29             StackTraceBubble.myMethod();
30             RecTest.genException();
31         }
32         catch (ArrayIndexOutOfBoundsException eoc)
33         {
34             // catch the exception
35             System.out.println("Standard message is: ");
36             System.out.println(eoc);
37             System.out.println("Stack trace: ");
38             eoc.printStackTrace();
39         }
40         System.out.println("After catch statement.");
41     }
42 }
```

Client



Developer

How many **threads** need to be **created**?

What **methods** need to be **invoked**?

Challenges

Testing multi-threaded libraries

```
1 class RecTest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("This won't be displayed");
10    }
11 }
12 class StackTraceBubble
13 {
14     static void myMethod()
15     {
16         try
17         {
18             RecTest.genException();
19         }
20         catch (ArrayIndexOutOfBoundsException eoc)
21         {
22             eoc.printStackTrace();
23         }
24     }
25     public static void main(String[] args)
26     {
27         try
28         {
29             StackTraceBubble.myMethod();
30             RecTest.genException();
31         }
32         catch (ArrayIndexOutOfBoundsException eoc)
33         {
34             // catch the exception
35             System.out.println("Standard message is: ");
36             System.out.println(eoc);
37             System.out.println("Stack trace: ");
38             eoc.printStackTrace();
39         }
40         System.out.println("After catch statement.");
41     }
42 }
```

Client



Developer

How many **threads** need to be **created**?

What **methods** need to be **invoked**?

What are the **parameters** to these methods?

Challenges

Testing multi-threaded libraries

```
1 class RecTest
2 {
3     static void genException()
4     {
5         int sum[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         sum[7] = 10;
9         System.out.println("This won't be displayed");
10    }
11 }
12
13 class StackTraceBubble
14 {
15     static void myMethod()
16     {
17         try
18         {
19             RecTest.genException();
20         }
21         catch (ArrayIndexOutOfBoundsException eoc)
22         {
23             eoc.printStackTrace();
24         }
25     }
26
27     public static void main(String[] args)
28     {
29         try
30         {
31             StackTraceBubble.myMethod();
32             RecTest.genException();
33         }
34         catch (ArrayIndexOutOfBoundsException eoc)
35         {
36             // catch the exception
37             System.out.println("Standard message is: ");
38             System.out.println(eoc);
39             System.out.println("Stack trace: ");
40             eoc.printStackTrace();
41         }
42         System.out.println("After catch statement.");
43     }
44 }
```

Client



Developer

How many **threads** need to be **created**?

What **methods** need to be **invoked**?

What are the **parameters** to these methods?

What is the **schedule** that needs to be followed?

Challenges

Client synthesis for multi-threaded libraries

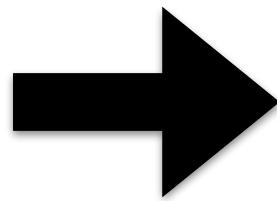
Client synthesis for multi-threaded libraries

```
package com;\n\npublic class StudentResults {\n\n    private String Full_Name;\n    private String Exam_Name;\n    private String Exam_Score;\n    private String Exam_Grade;\n\n    StudentResults() {\n        Full_Name = "No Name Given";\n        Exam_Name = "Unknown";\n        Exam_Score = "No Score";\n        Exam_Grade = "Unknown";\n    }\n\n    String fullName(String fName) {\n        Full_Name = fName;\n        return Full_Name;\n    }\n\n    String examName(String examCode) {\n        if (examCode.equals("VB")) {\n            Exam_Name = "Visual Basic .NET";\n        }\n        else if (examCode.equals("JV")) {\n            Exam_Name = "Java";\n        }\n        else if (examCode.equals("CS")) {\n            Exam_Name = "C# .NET";\n        }\n        else if (examCode.equals("PW")) {\n            Exam_Name = "PHP";\n        }\n        else {\n            Exam_Name = "No Exam Selected";\n        }\n\n        return Exam_Name;\n    }\n}
```

**Class under
test**

Client synthesis for multi-threaded libraries

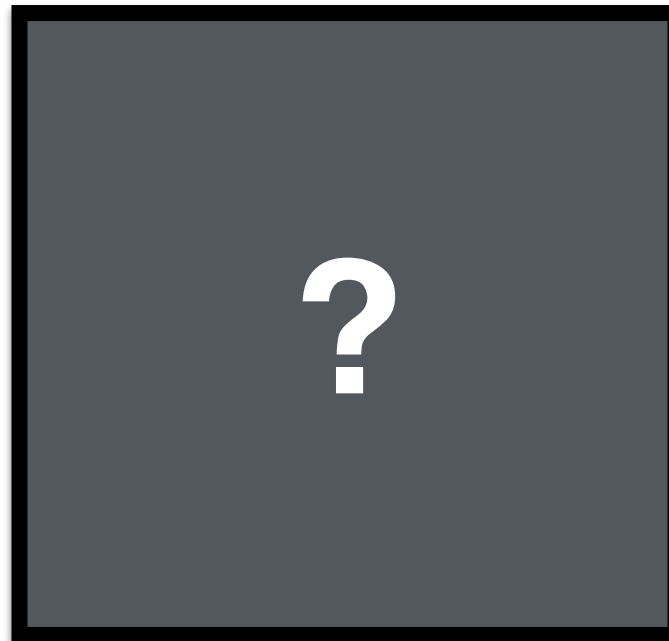
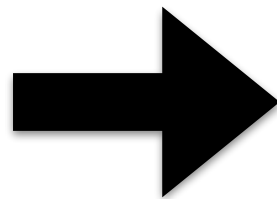
```
package com;\n\npublic class StudentResults {\n\n    private String Full_Name;\n    private String Exam_Name;\n    private String Exam_Score;\n    private String Exam_Grade;\n\n    StudentResults() {\n        Full_Name = "No Name Given";\n        Exam_Name = "Unknown";\n        Exam_Score = "No Score";\n        Exam_Grade = "Unknown";\n    }\n\n    String fullName(String fName) {\n        Full_Name = fName;\n        return Full_Name;\n    }\n\n    String examName(String examCode) {\n        if (examCode.equals("VB")) {\n            Exam_Name = "Visual Basic .NET";\n        }\n        else if (examCode.equals("JV")) {\n            Exam_Name = "Java";\n        }\n        else if (examCode.equals("CS")) {\n            Exam_Name = "C# .NET";\n        }\n        else if (examCode.equals("PW")) {\n            Exam_Name = "PHP";\n        }\n        else {\n            Exam_Name = "No Exam Selected";\n        }\n\n        return Exam_Name;\n    }\n}
```



**Class under
test**

Client synthesis for multi-threaded libraries

```
package com;\n\npublic class StudentResults {\n\n    private String Full_Name;\n    private String Exam_Name;\n    private String Exam_Score;\n    private String Exam_Grade;\n\n    StudentResults() {\n        Full_Name = "No Name Given";\n        Exam_Name = "Unknown";\n        Exam_Score = "No Score";\n        Exam_Grade = "Unknown";\n    }\n\n    String fullName(String fName) {\n        Full_Name = fName;\n        return Full_Name;\n    }\n\n    String examName(String examCode) {\n        if (examCode.equals("VB")) {\n            Exam_Name = "Visual Basic .NET";\n        }\n        else if (examCode.equals("JV")) {\n            Exam_Name = "Java";\n        }\n        else if (examCode.equals("CS")) {\n            Exam_Name = "C# .NET";\n        }\n        else if (examCode.equals("PW")) {\n            Exam_Name = "PHP";\n        }\n        else {\n            Exam_Name = "No Exam Selected";\n        }\n\n        return Exam_Name;\n    }\n}
```

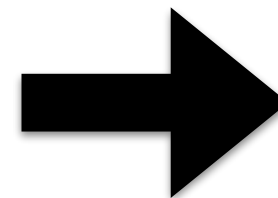
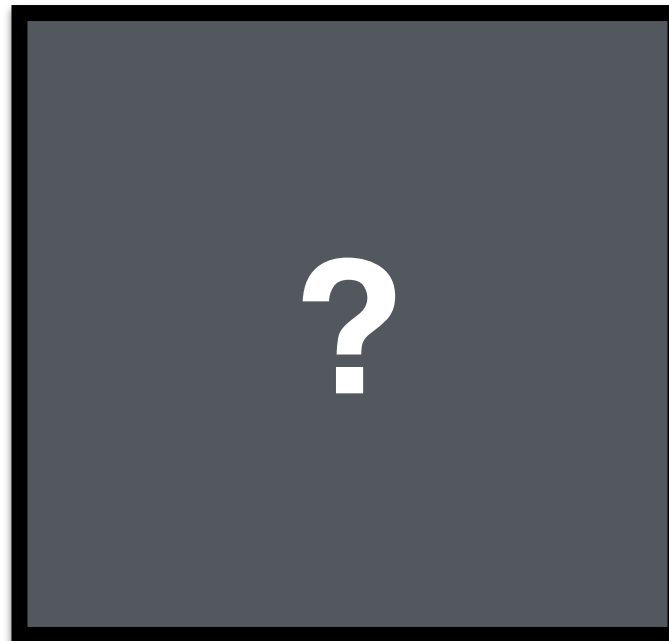
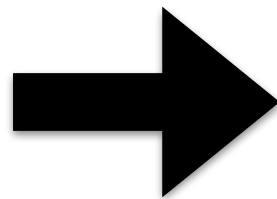


**Class under
test**

Client synthesis for multi-threaded libraries

```
package com;\n\npublic class StudentResults {\n\n    private String Full_Name;\n    private String Exam_Name;\n    private String Exam_Score;\n    private String Exam_Grade;\n\n    StudentResults() {\n        Full_Name = "No Name Given";\n        Exam_Name = "Unknown";\n        Exam_Score = "No Score";\n        Exam_Grade = "Unknown";\n    }\n\n    String fullName(String aName) {\n        Full_Name = aName;\n        return Full_Name;\n    }\n\n    String examName(String examCode) {\n        if (examCode.equals("VB")) {\n            Exam_Name = "Visual Basic .NET";\n        }\n        else if (examCode.equals("JV")) {\n            Exam_Name = "Java";\n        }\n        else if (examCode.equals("CS")) {\n            Exam_Name = "C# .NET";\n        }\n        else if (examCode.equals("PW")) {\n            Exam_Name = "PHP";\n        }\n        else {\n            Exam_Name = "No Exam Selected";\n        }\n\n        return Exam_Name;\n    }\n}
```

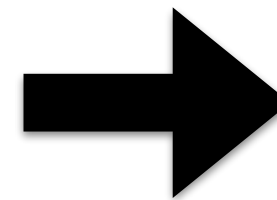
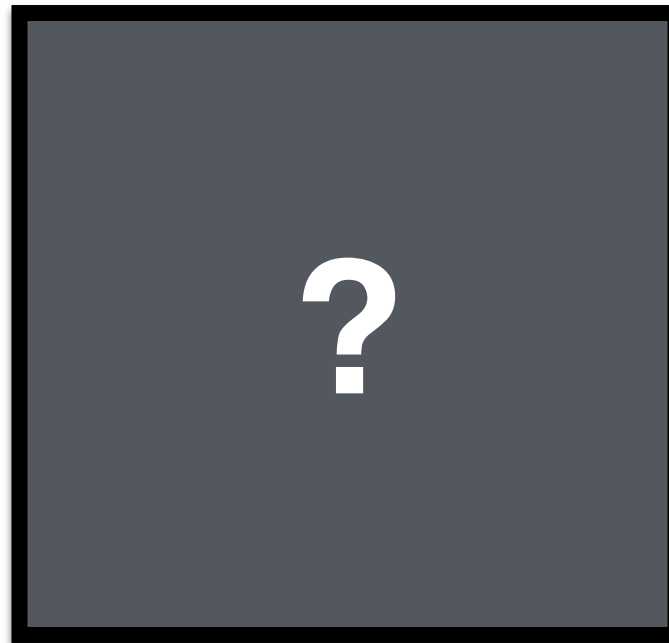
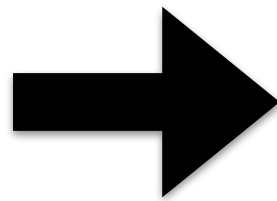
**Class under
test**



Client synthesis for multi-threaded libraries

```
package com;\n\npublic class StudentResults {\n\n    private String Full_Name;\n    private String Exam_Name;\n    private String Exam_Score;\n    private String Exam_Grade;\n\n    StudentResults() {\n        Full_Name = "No Name Given";\n        Exam_Name = "Unknown";\n        Exam_Score = "No Score";\n        Exam_Grade = "Unknown";\n    }\n\n    String fullName(String fName) {\n        Full_Name = fName;\n        return Full_Name;\n    }\n\n    String examName(String examCode) {\n        if (examCode.equals("VB")) {\n            Exam_Name = "Visual Basic .NET";\n        }\n        else if (examCode.equals("JV")) {\n            Exam_Name = "Java";\n        }\n        else if (examCode.equals("CS")) {\n            Exam_Name = "C# .NET";\n        }\n        else if (examCode.equals("PW")) {\n            Exam_Name = "PHP";\n        }\n        else {\n            Exam_Name = "No Exam Selected";\n        }\n\n        return Exam_Name;\n    }\n}
```

**Class under
test**



```
1 class Ecotest\n2 {\n3     static void genException()\n4     {\n5         int sum[] = new int[4];\n6         System.out.println("before exception is generated.");\n7         // generate an index out-of-bounds exception\n8         sum[7] = 10;\n9         System.out.println("this won't be displayed");\n10    }\n11\n12    class StackTraceBubble\n13    {\n14        static void myMethod()\n15        {\n16            try\n17            {\n18                Ecotest.genException();\n19            }\n20            catch (ArrayIndexOutOfBoundsException exc)\n21            {\n22                exc.printStackTrace();\n23            }\n24        }\n25        public static void main(String[] args)\n26        {\n27            try\n28            {\n29                StackTraceBubble.myMethod();\n30                Ecotest.genException();\n31            }\n32            catch (ArrayIndexOutOfBoundsException exc)\n33            {\n34                // catch the exception\n35                System.out.println("Standard message for ");\n36                System.out.println(exc);\n37                System.out.println("inStack trace ");\n38                exc.printStackTrace();\n39            }\n40            System.out.println("After catch statement.");\n41        }\n42    }\n43 }
```

Client

Story of the black box so far...

Story of the black box so far...

Randomized generation

Pradel and Gross, [PLDI'12](#)

Story of the black box so far...

Randomized generation

Pradel and Gross, [PLDI'12](#)



Crashes and Deadlocks

Story of the black box so far...

Randomized generation

Pradel and Gross, [PLDI'12](#)



Crashes and Deadlocks

Targeted generation

Story of the black box so far...

Randomized generation

Pradel and Gross, [PLDI'12](#)



Crashes and Deadlocks

Targeted generation

Samak and Ramanathan,
[OOPSLA'14](#)

Story of the black box so far...

Randomized generation

Pradel and Gross, [PLDI'12](#)



Crashes and Deadlocks

Targeted generation

Samak and Ramanathan,
[OOPSLA'14](#)

Samak, Ramanathan, Jagannathan
[PLDI'15](#)

Story of the black box so far...

Randomized generation

Pradel and Gross, [PLDI'12](#)



Crashes and Deadlocks

Targeted generation

Samak and Ramanathan,
[OOPSLA'14](#)

Samak, Ramanathan, Jagannathan
[PLDI'15](#)

Samak and Ramanathan,
[FSE'15](#)

Story of the black box so far...

Randomized generation

Pradel and Gross, [PLDI'12](#)

-- ➔ Crashes and Deadlocks

Targeted generation

Samak and Ramanathan,
[OOPSLA'14](#)

Samak, Ramanathan, Jagannathan
[PLDI'15](#)

Samak and Ramanathan,
[FSE'15](#)

-- ➔

-- ➔

-- ➔

**Dynamic
Analysis
Engine**

Story of the black box so far...

Randomized generation

Pradel and Gross, [PLDI'12](#)

-- ➔ Crashes and Deadlocks

Targeted generation

Samak and Ramanathan,
[OOPSLA'14](#)

-- ➔

-- ➔

Deadlocks

Samak, Ramanathan, Jagannathan
[PLDI'15](#)

-- ➔

-- ➔

Data races

Samak and Ramanathan,
[FSE'15](#)

-- ➔

-- ➔

Atomicity
violations

**Dynamic
Analysis
Engine**

Story of the black box so far...

Randomized generation

Pradel and Gross, [PLDI'12](#)

-- ➔ Crashes and Deadlocks

Targeted generation

Need a **targeted** approach to reveal **crashes**

Samak and Ramanathan, [OOPSLA'14](#)

-- ➔

-- ➔

Deadlocks

Samak, Ramanathan, Jagannathan
[PLDI'15](#)

-- ➔

-- ➔

Data races

Samak and Ramanathan, [FSE'15](#)

-- ➔

-- ➔

Atomicity violations

**Dynamic
Analysis
Engine**

and now...

and now...

- ◆ Detecting crashes in a library requires:

and now...

- ◆ Detecting crashes in a library requires:
 - ◆ Well designed multi-threaded clients

and now...

- ◆ Detecting crashes in a library requires:
 - ◆ Well designed multi-threaded clients
 - ◆ Specific thread interleavings

and now...

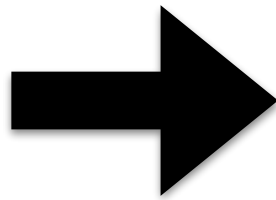
- ◆ Detecting crashes in a library requires:
 - ◆ Well designed multi-threaded clients
 - ◆ Specific thread interleavings
- ◆ We propose MINION





Class under test

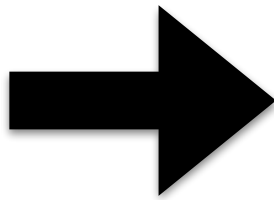
```
package examz;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String FullName(String fName) {  
        Full_Name = fName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CS")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("PW")) {  
            Exam_Name = "PHP";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
  
        return Exam_Name;  
    }  
}
```



Class under test

assert(p1)

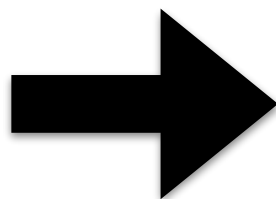
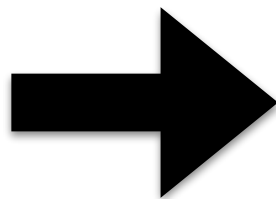
```
package examz;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String sName) {  
        Full_Name = sName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CS")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("PW")) {  
            Exam_Name = "PHP";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
  
        return Exam_Name;  
    }  
}
```



Class under test

assert(p1)

```
package exam2;  
  
public class StudentResults {  
  
    private String Full_Name;  
    private String Exam_Name;  
    private String Exam_Score;  
    private String Exam_Grade;  
  
    StudentResults() {  
        Full_Name = "No Name Given";  
        Exam_Name = "Unknown";  
        Exam_Score = "No Score";  
        Exam_Grade = "Unknown";  
    }  
  
    String fullName(String sName) {  
        Full_Name = sName;  
        return Full_Name;  
    }  
  
    String examName(String examCode) {  
  
        if (examCode.equals("VB")) {  
            Exam_Name = "Visual Basic .NET";  
        }  
        else if (examCode.equals("JV")) {  
            Exam_Name = "Java";  
        }  
        else if (examCode.equals("CS")) {  
            Exam_Name = "C# .NET";  
        }  
        else if (examCode.equals("PW")) {  
            Exam_Name = "PHP";  
        }  
        else {  
            Exam_Name = "No Exam Selected";  
        }  
  
        return Exam_Name;  
    }  
}
```



Random sequential test

```
1 import java.util.Scanner;  
2  
3 public class Main  
4 {  
5     public static void main(String[] args)  
6     {  
7         Scanner keys = new Scanner(System.in);  
8         int length;  
9         double width;  
10        System.out.print("Enter the length ");  
11        length=keys.nextInt();  
12        System.out.print("Enter the width ");  
13        width=keys.nextDouble();  
14        System.out.println("\nlength="+length+" width="+  
15        width+" area"+(length*width));  
16  
17        System.out.print("\nEnter your full name (first, last) ");  
18        String firstName;  
19        String lastName;  
20        firstName=keys.next();  
21        lastName=keys.next();  
22        System.out.println("Your name is "+lastName+", "+firstName);  
23    }  
24 }
```

Class under test

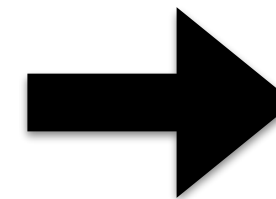
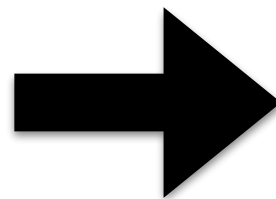
assert(p1)

Multi-threaded client

```
class EcTest {
    static void genException() {
        int num[] = new int[4];
        System.out.println("Before exception is generated.");
        // generate an index out-of-bounds exception
        num[7] = 10;
        System.out.println("This won't be displayed.");
    }
}

class StackTraceBubble {
    static void myMethod() {
        try {
            EcTest.genException();
        } catch (ArrayIndexOutOfBoundsException exc) {
            exc.printStackTrace();
        }
    }
}

public static void main(String[] args) {
    try {
        StackTraceBubble.myMethod();
        EcTest.genException();
    } catch (ArrayIndexOutOfBoundsException exc) {
        // catch the exception
        System.out.println("Standard message log.");
        System.out.println(exc);
        System.out.println("Stack trace.");
        exc.printStackTrace();
    }
    System.out.println("After catch statement.");
}
```



Random sequential test

```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner keys = new Scanner(System.in);
6         int length;
7         double width;
8         System.out.print("Enter the length ");
9         length=keys.nextInt();
10        System.out.print("Enter the width ");
11        width=keys.nextDouble();
12        System.out.println("\nlength="+length+" width="
13                             + width+" area="(length*width) );
14
15        System.out.print("\nEnter your full name (first, last) ");
16        String firstName;
17        String lastName;
18        firstName=keys.next();
19        lastName=keys.next();
20        System.out.println("Your name is "+lastName+", "+firstName);
21    }
22 }
```

Class under test

assert(p1)

Multi-threaded client



Random sequential test

```
1 import java.util.Scanner;
2
3 public class Main
4 {
5     public static void main(String[] args)
6     {
7         Scanner keys = new Scanner(System.in);
8         int length;
9         double width;
10        System.out.print("Enter the length ");
11        length=keys.nextInt();
12        System.out.print("Enter the width ");
13        width=keys.nextDouble();
14        System.out.println("\nlength="+length+" width="+
15        width+" area"+(length*width));
16
17        System.out.print("\nEnter your full name (first, last) ");
18        String firstName;
19        String lastName;
20        firstName=keys.next();
21        lastName=keys.next();
22        System.out.println("Your name is "+lastName+", "+firstName);
23    }
24 }
```

```
1 class Ecctest
2 {
3     static void genException()
4     {
5         int num[] = new int[4];
6         System.out.println("Before exception is generated.");
7         // generate an index out-of-bounds exception
8         num[7] = 10;
9         System.out.println("this won't be displayed");
10    }
11
12    class StackTraceBubble
13    {
14        static void myMethod()
15        {
16            try
17            {
18                Ecctest.genException();
19            }
20            catch (ArrayIndexOutOfBoundsException exc)
21            {
22                exc.printStackTrace();
23            }
24        }
25
26        public static void main(String[] args)
27        {
28            try
29            {
30                StackTraceBubble.myMethod();
31                Ecctest.genException();
32            }
33            catch (ArrayIndexOutOfBoundsException exc)
34            {
35                // catch the exception
36                System.out.println("Standard message is: ");
37                System.out.println(exc);
38                System.out.println("\nStack traces ");
39                exc.printStackTrace();
40            }
41            System.out.println("After catch statement.");
42        }
43    }
44 }
```

Class under test

assert(p1)

Multi-threaded client

```
class Ecotest {
    static void genException() {
        int num[] = new int[4];
        System.out.println("Before exception is generated.");
        // generate an index out-of-bounds exception
        num[7] = 10;
        System.out.println("this won't be displayed");
    }

    class StackTraceBubble {
        static void myMethod() {
            try {
                Ecotest.genException();
            } catch (ArrayIndexOutOfBoundsException exc) {
                exc.printStackTrace();
            }
        }

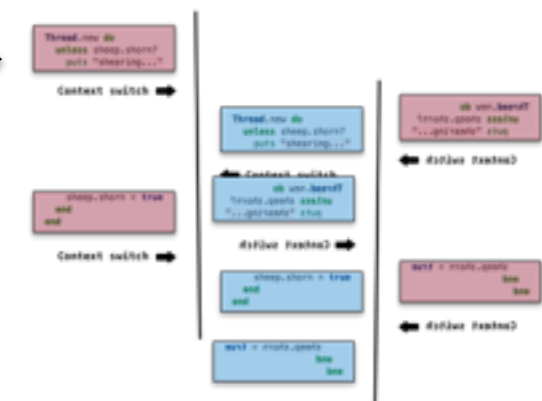
        public static void main(String[] args) {
            try {
                StackTraceBubble.myMethod();
                Ecotest.genException();
            } catch (ArrayIndexOutOfBoundsException exc) {
                // catch the exception
                System.out.println("Standard message for ");
                System.out.println(exc);
                System.out.println("StackTrace");
                exc.printStackTrace();
            }
            System.out.println("After catch statement.");
        }
    }
}
```

Random sequential test

```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner keys = new Scanner(System.in);
6         int length;
7         double width;
8         System.out.print("Enter the length ");
9         length=keys.nextInt();
10        System.out.print("Enter the width ");
11        width=keys.nextDouble();
12        System.out.println("\nlength*length* width*
13        = width* area*=(length*width)");
14
15        System.out.print("\nEnter your full name (first, last) ");
16        String firstName;
17        String lastName;
18        firstName=keys.next();
19        lastName=keys.next();
20        System.out.println("Your name is "+lastName+", "+firstName);
21    }
22 }
```



Thread interleaving



Class under test

assert(p1)

Multi-threaded client

```
class EcTest {
    static void genException() {
        int num[] = new int[4];
        System.out.println("Before exception is generated.");
        // generate an index out-of-bounds exception
        num[7] = 10;
        System.out.println("this won't be displayed");
    }

    class StackTraceBubble {
        static void myMethod() {
            try {
                EcTest.genException();
            } catch (ArrayIndexOutOfBoundsException exc) {
                exc.printStackTrace();
            }
        }

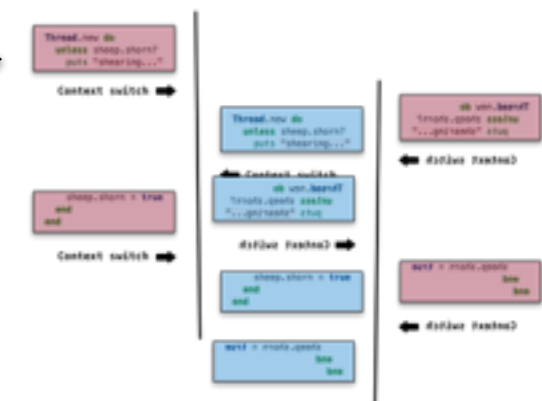
        public static void main(String[] args) {
            try {
                StackTraceBubble.myMethod();
                EcTest.genException();
            } catch (ArrayIndexOutOfBoundsException exc) {
                // catch the exception
                System.out.println("Standard message is: ");
                System.out.println(exc);
                System.out.println("Stack trace: ");
                exc.printStackTrace();
            }
            System.out.println("After catch statement.");
        }
    }
}
```

Random sequential test

```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner keys = new Scanner(System.in);
6         int length;
7         double width;
8         System.out.print("Enter the length ");
9         length=keys.nextInt();
10        System.out.print("Enter the width ");
11        width=keys.nextDouble();
12        System.out.println("\nlength*length* width*
13        = width* area*=(length*width) ");
14
15        System.out.print("\nEnter your full name (first, last) ");
16        String firstName;
17        String lastName;
18        firstName=keys.next();
19        lastName=keys.next();
20        System.out.println("Your name is "+lastName+", "+firstName);
21    }
22 }
```



Thread interleaving



Crash

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = buf.length - pos;
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122             System.arraycopy(buf, pos, cbuf, off, avail);
123             ...
126         }
127         ...
135     } catch (ArrayIndexOutOfBoundsException e) {
136         throw new IndexOutOfBoundsException();
137     }
138 }
139 }
```

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = buf.length - pos;
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122             System.arraycopy(buf, pos, cbuf, off, avail);
123             ...
126         }
127         ...
135     } catch (ArrayIndexOutOfBoundsException e) {
136         throw new IndexOutOfBoundsException();
137     }
138 }
139 }
```

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = buf.length - pos;
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122         assert(buf != null);
123         ...
126     }
127     ...
135 } catch (ArrayIndexOutOfBoundsException e) {
136     throw new IndexOutOfBoundsException();
137 }
138 }
139 }
```

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
                                   // returns on len <= 0
118         int avail = buf.length - pos;
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122         assert(buf != null);
                                   ...
126         }
                                   ...
135     } catch (ArrayIndexOutOfBoundsException e) {
136         throw new IndexOutOfBoundsException();
137     }
138 }
139 }
```

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
                                   // returns on len <= 0
118         int avail = assert(buf != null);
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122         assert(buf != null);
                                   ...
126         }
                                   ...
135     } catch (ArrayIndexOutOfBoundsException e) {
136         throw new IndexOutOfBoundsException();
137     }
138 }
139 }
```

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118             int avail = assert(buf != null);
119             if (avail > 0) {
120                 if (len < avail)
121                     avail = len;
122                 assert(buf != null);
123                 ...
126             }
127             ...
135         } catch (ArrayIndexOutOfBoundsException e) {
136             throw new IndexOutOfBoundsException();
137         }
138     }
139 }
```

PushbackReader.java


Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118             int avail = assert(buf != null);
119             if (avail > 0) {
120                 if (len < avail)
121                     avail = len;
122                 assert(buf != null);
123                 ...
126             }
127             ...
135         } catch (ArrayIndexOutOfBoundsException e) {
136             throw new IndexOutOfBoundsException();
137         }
138     }
139 }
```

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118             int avail = assert(buf != null);
119             if (avail > 0) {
120                 if (len < avail)
121                     avail = len;
122             assert(buf != null);
123             ...
126         }
127         ...
135     } catch (ArrayIndexOutOfBoundsException e) {
136         throw new IndexOutOfBoundsException();
137     }
138 }
139 }
```



PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = assert(buf != null);
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122         assert(buf != null);
123         ...
126     }
127     ...
135 } catch (ArrayIndexOutOfBoundsException e) {
136     throw new IndexOutOfBoundsException();
137 }
138 }
139 }
```

This assert
holds!

Fail this assert!

PushbackReader.java


Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118             int avail = assert(buf != null);
119             if (avail > 0) {
120                 if (len < avail)
121                     avail = len;
122                 assert(buf != null);
123                 ...
126             }
127             ...
135         } catch (ArrayIndexOutOfBoundsException e) {
136             throw new IndexOutOfBoundsException();
137         }
138     }
139 }
```

PushbackReader.java

Example from JDK 8


```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118             int avail = assert(buf != null);
119             if (avail > 0) {
120                 if (len < avail)
121                     avail = len;
122                 assert(buf != null);
123                 ...
126             }
127             ...
135         } catch (ArrayIndexOutOfBoundsException e) {
136             throw new IndexOutOfBoundsException();
137         }
138     }
139 }
```


 **buf != null**

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = assert(buf != null);
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122         assert(buf != null);
123         ...
126     }
127     ...
135 } catch (ArrayIndexOutOfBoundsException e) {
136     throw new IndexOutOfBoundsException();
137 }
138 }
139 }
```

 **buf != null**

 **buf == null**

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = buf.length - pos;
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122             assert(buf != null);
123             ...
126         }
127         ...
135     } catch (ArrayIndexOutOfBoundsException e) {
136         throw new IndexOutOfBoundsException();
137     }
138 }
139 }
```

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = buf.length - pos;
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122         }
123         ...
126     }
127     ...
135 } catch (ArrayIndexOutOfBoundsException e) {
136     throw new IndexOutOfBoundsException();
137 }
138 }
139 }
```

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = buf.length - pos;
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122             assert(buf != null);
123             ...
126         }
127         ...
135     } catch (ArrayIndexOutOfBoundsException e) {
136         throw new IndexOutOfBoundsException();
137     }
138 }
139 }
```

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = buf.length - pos;
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122             assert(buf != null);
123             ...
126         }
127         ...
135     } catch (ArrayIndexOutOfBoundsException e) {
136         throw new IndexOutOfBoundsException();
137     }
138 }
139 }
```

Evaluate to true

PushbackReader.java

Example from JDK 8

```
106 public int read(char cbuf[], int off, int len)
                                   throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
118         int avail = buf.length - pos;
119         if (avail > 0) {
120             if (len < avail)
121                 avail = len;
122             assert(buf != null);
123             ...
126         }
127         ...
135     } catch (ArrayIndexOutOfBoundsException e) {
136         throw new IndexOutOfBoundsException();
137     }
138 }
139 }
```

Don't care

Evaluate to true

PushbackReader.java

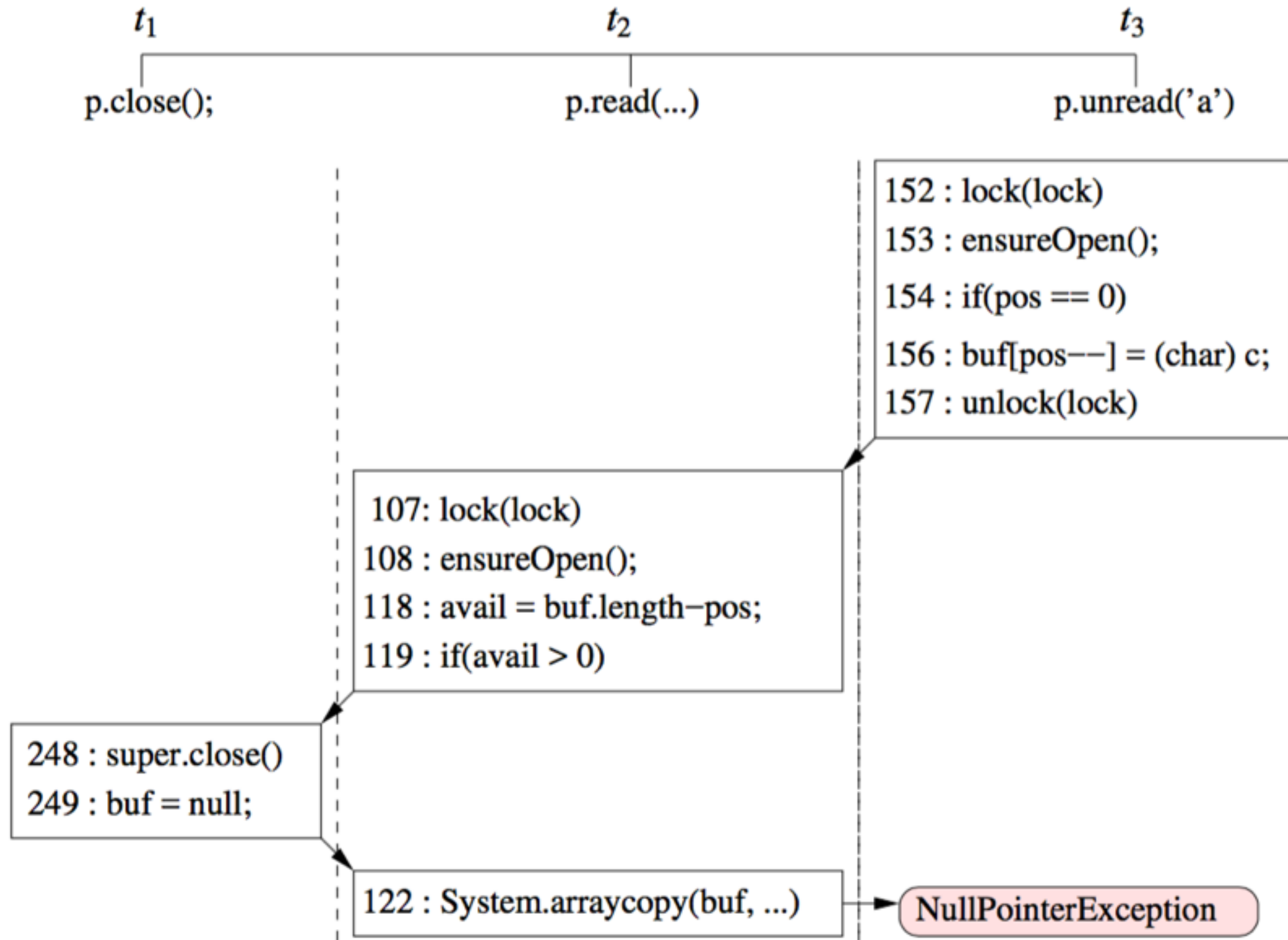
Example from JDK 8



```
106 public int read(char cbuf[], int off, int len)
    throws IOException {
107     synchronized (lock) {
108         ensureOpen();
109         try {
110             ... // returns on len <= 0
111             int avail = buf.length - pos;
112             if (avail > 0) {
113                 if (len < avail)
114                     avail = len;
115                 assert(buf != null);
116                 ...
117             }
118             ...
119         } catch (ArrayIndexOutOfBoundsException e) {
120             throw new IndexOutOfBoundsException();
121         }
122     }
123 }
124 }
```

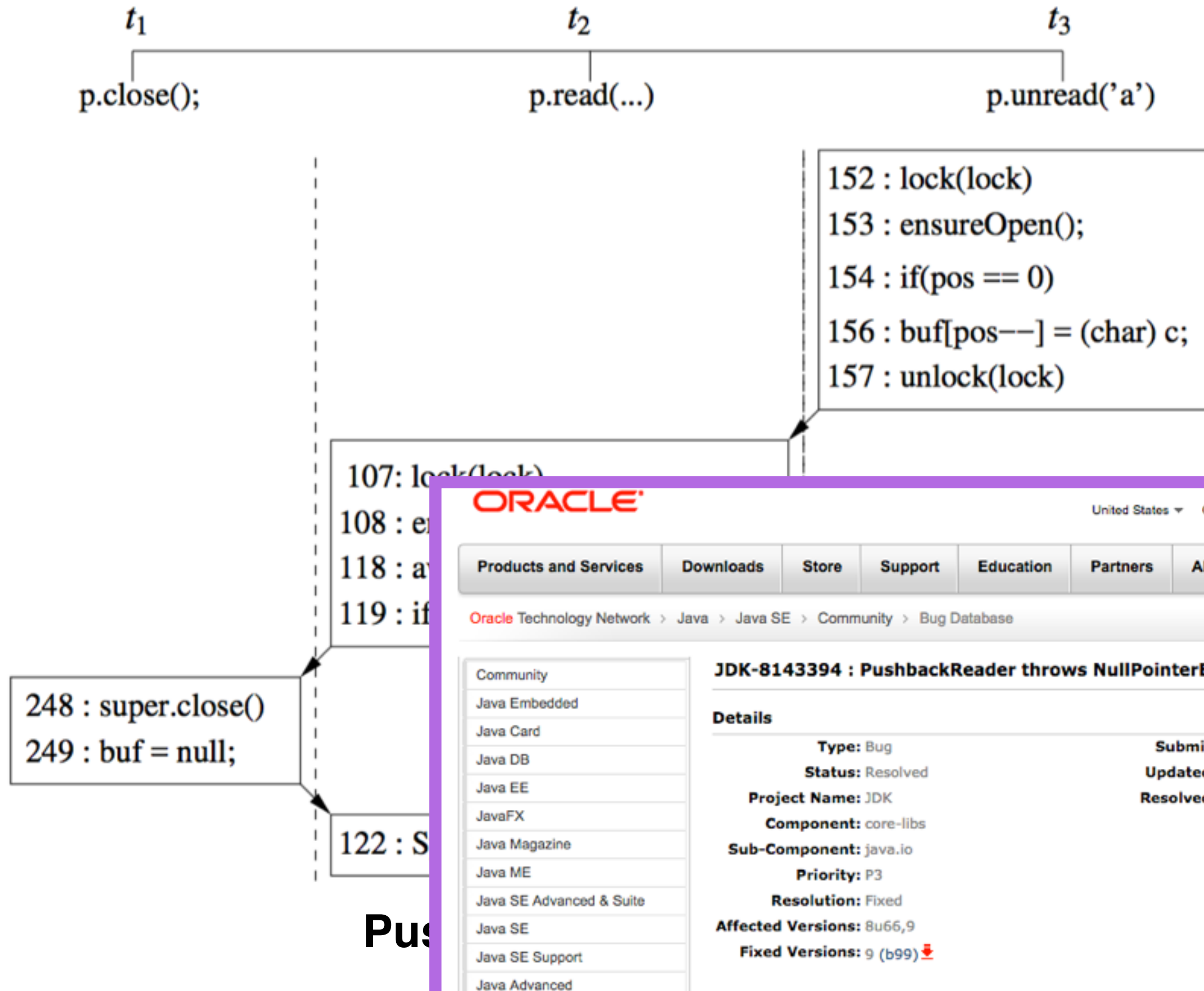
PushbackReader.java

Crash found by Minion in JDK8



PushbackReader.java

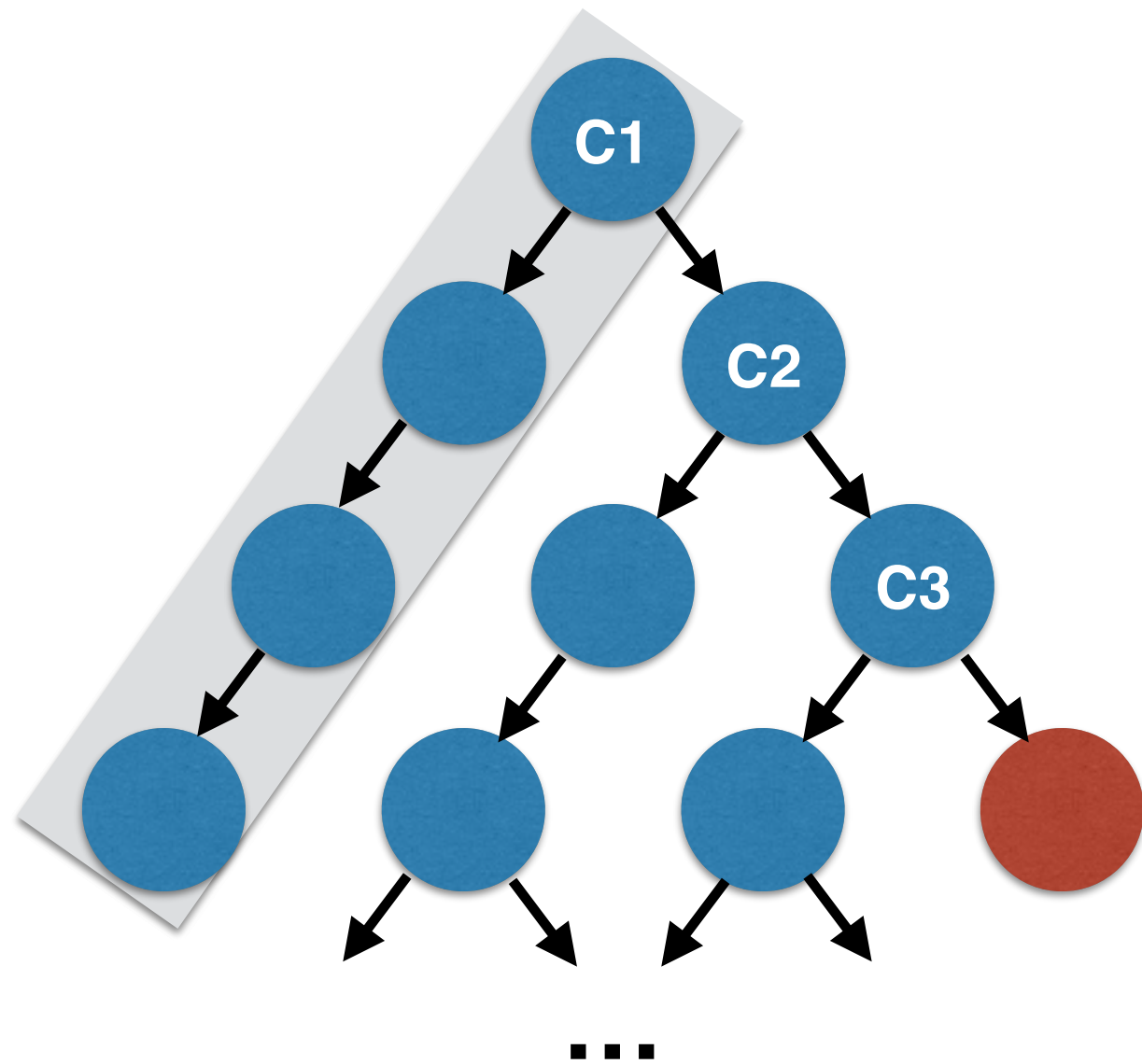
Crash found by Minion in JDK8



Execution synthesis

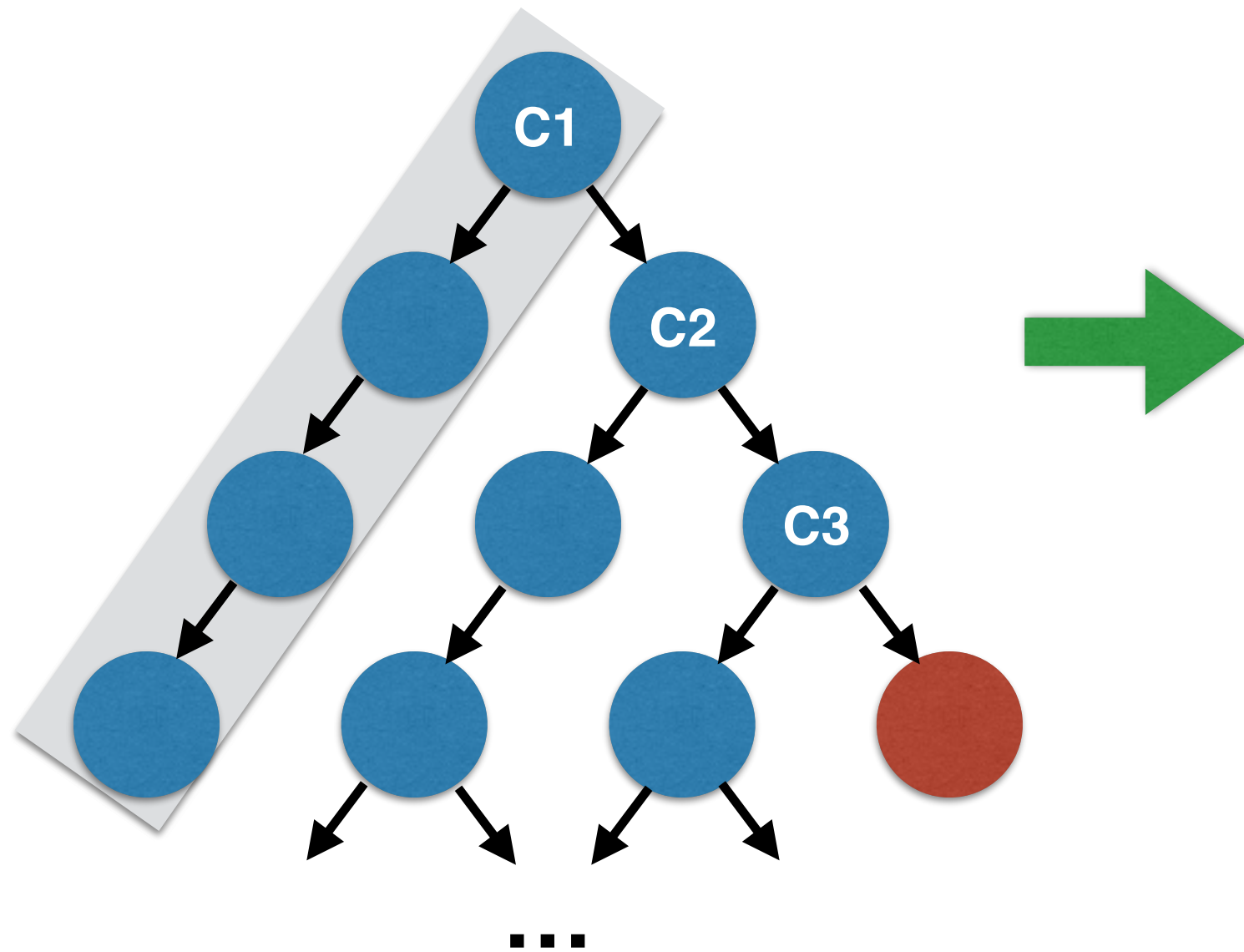
Execution synthesis

Random execution



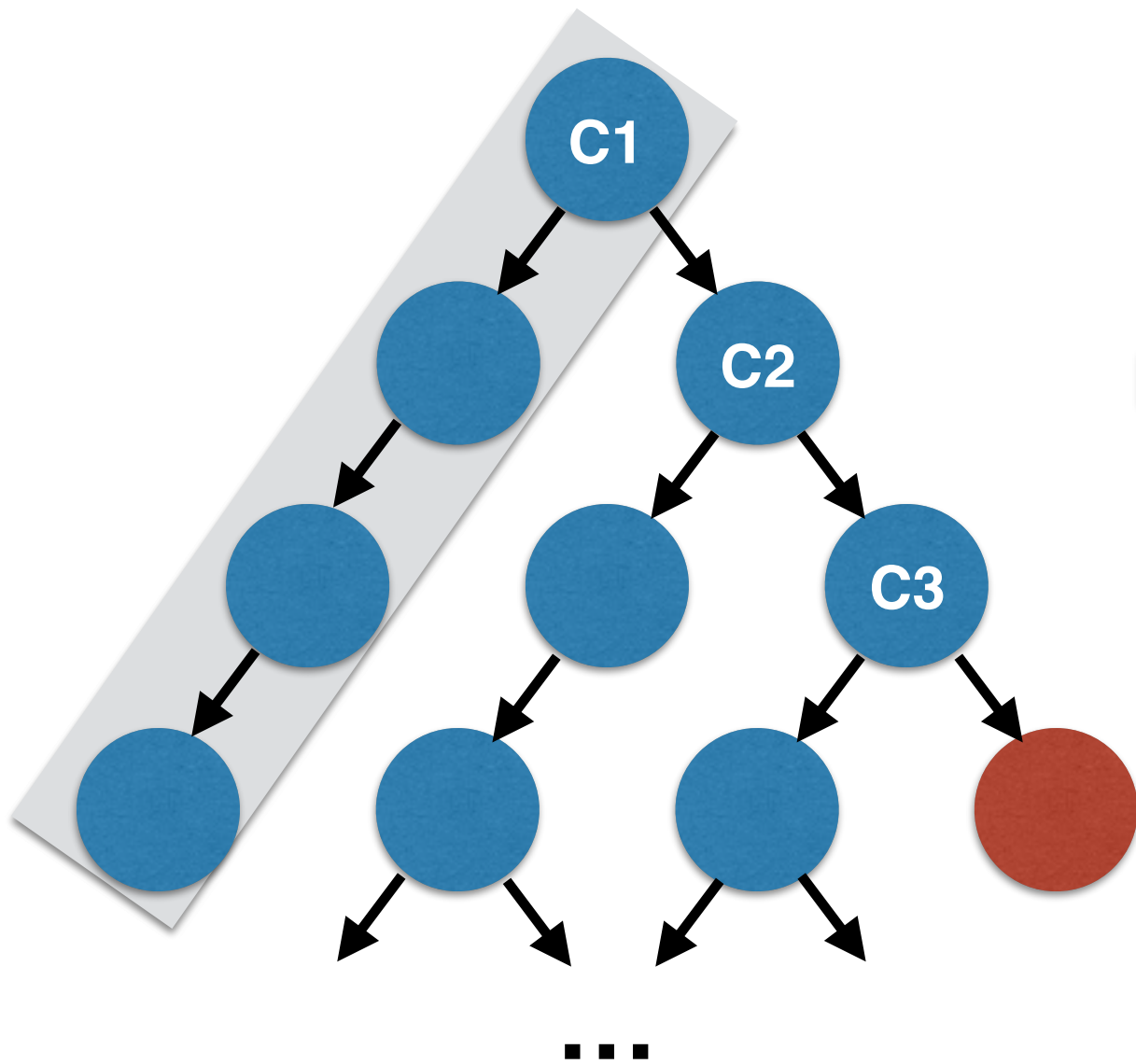
Execution synthesis

Random execution

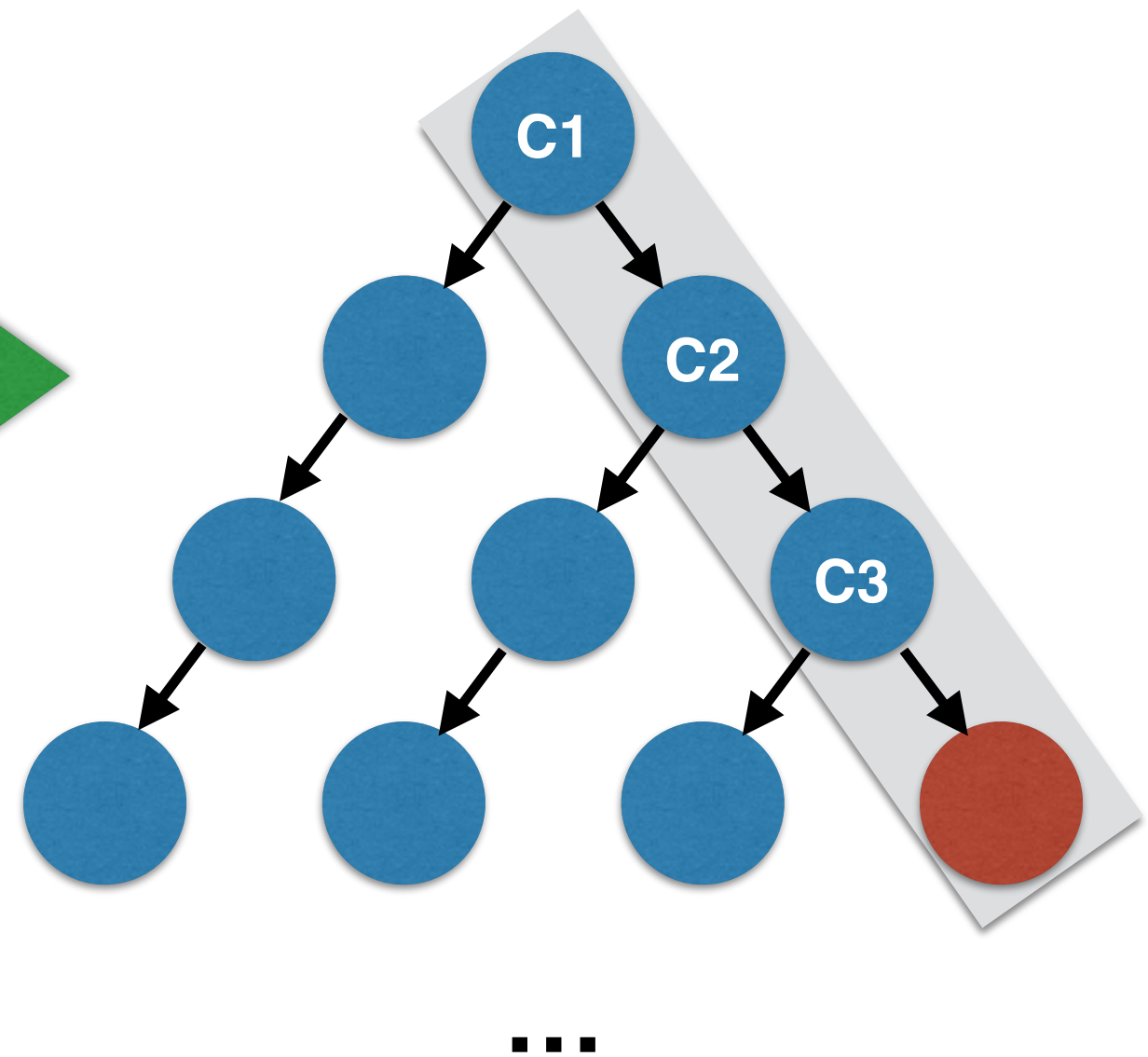


Execution synthesis

Random execution

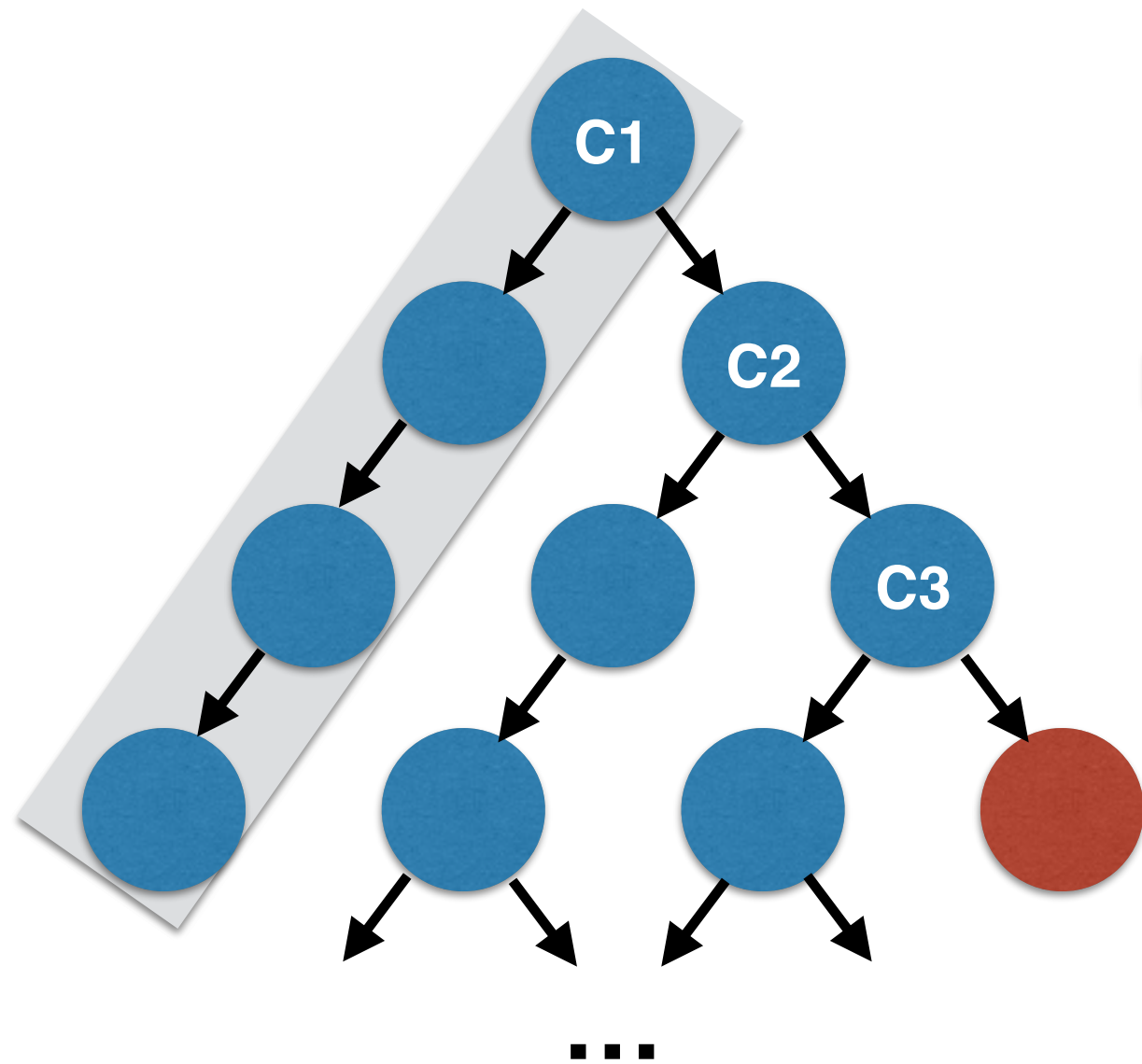


Target execution

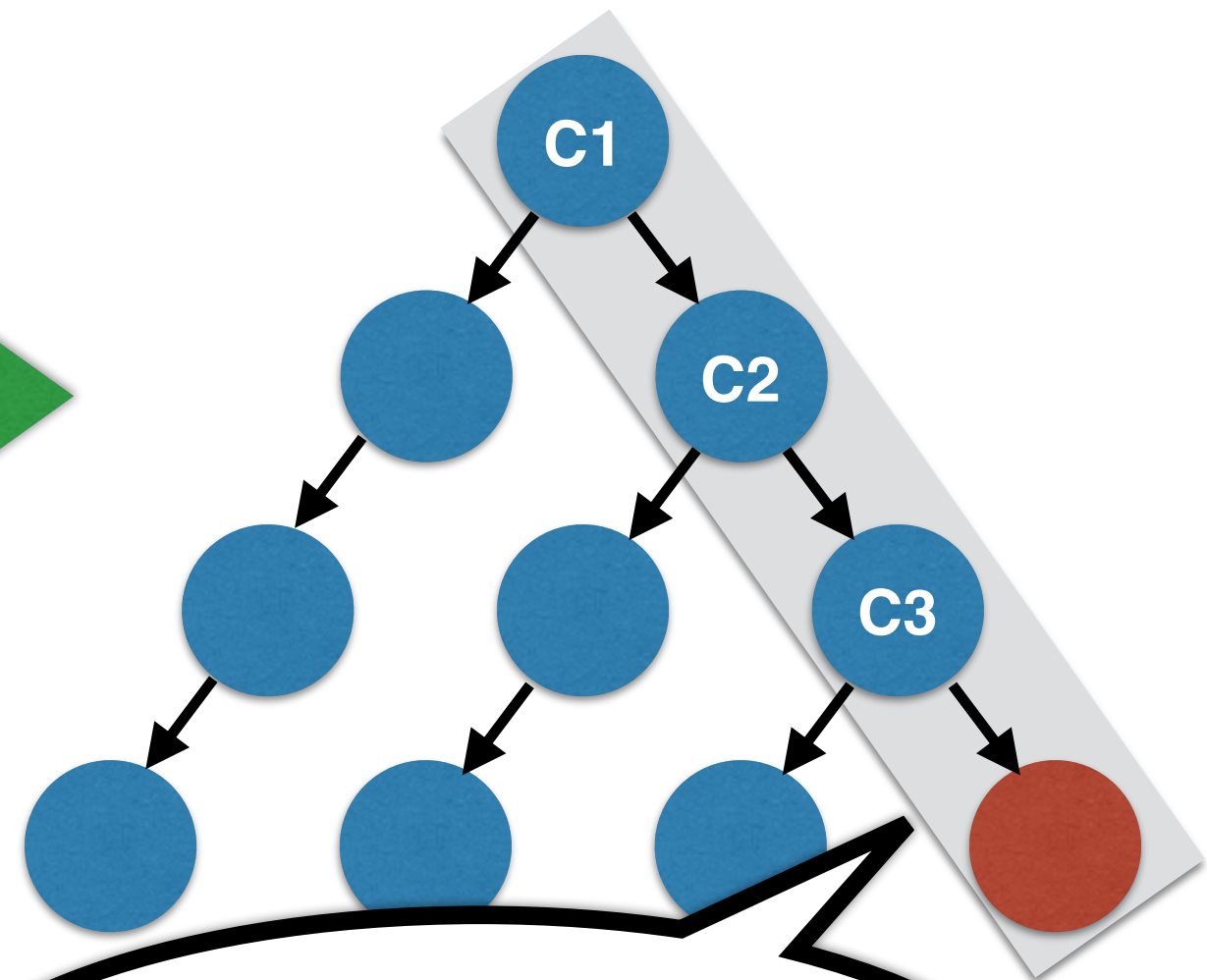
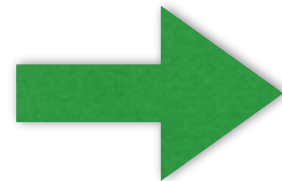


Execution synthesis

Random execution



Target execution



The path can be sequentially infeasible

Execution synthesis

t_1 : m3

t_2 : m1

t_3 : m2

Execution synthesis

- Methods invoked from different threads appropriately

t_1 : m3

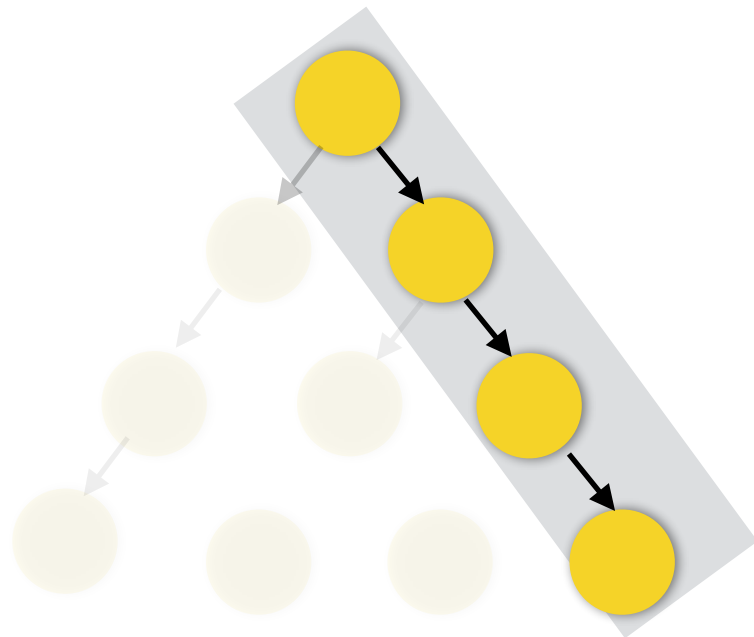
t_2 : m1

t_3 : m2

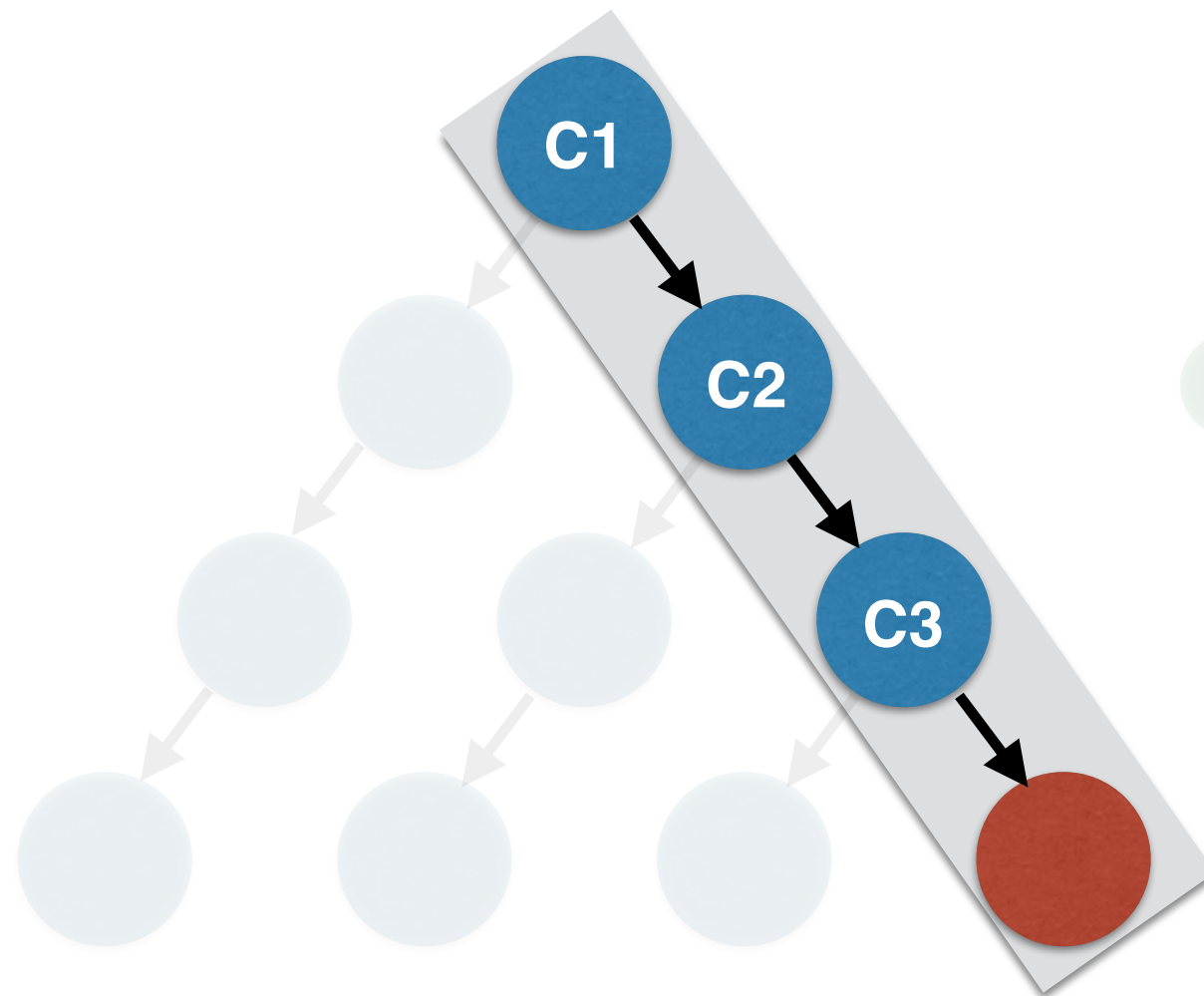
Execution synthesis

- Methods invoked from different threads appropriately

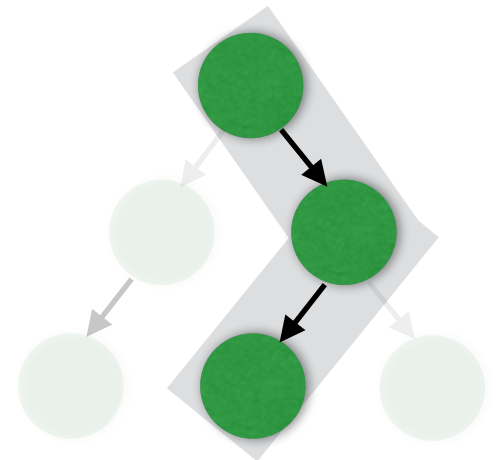
t₁: m3



t₂: m1

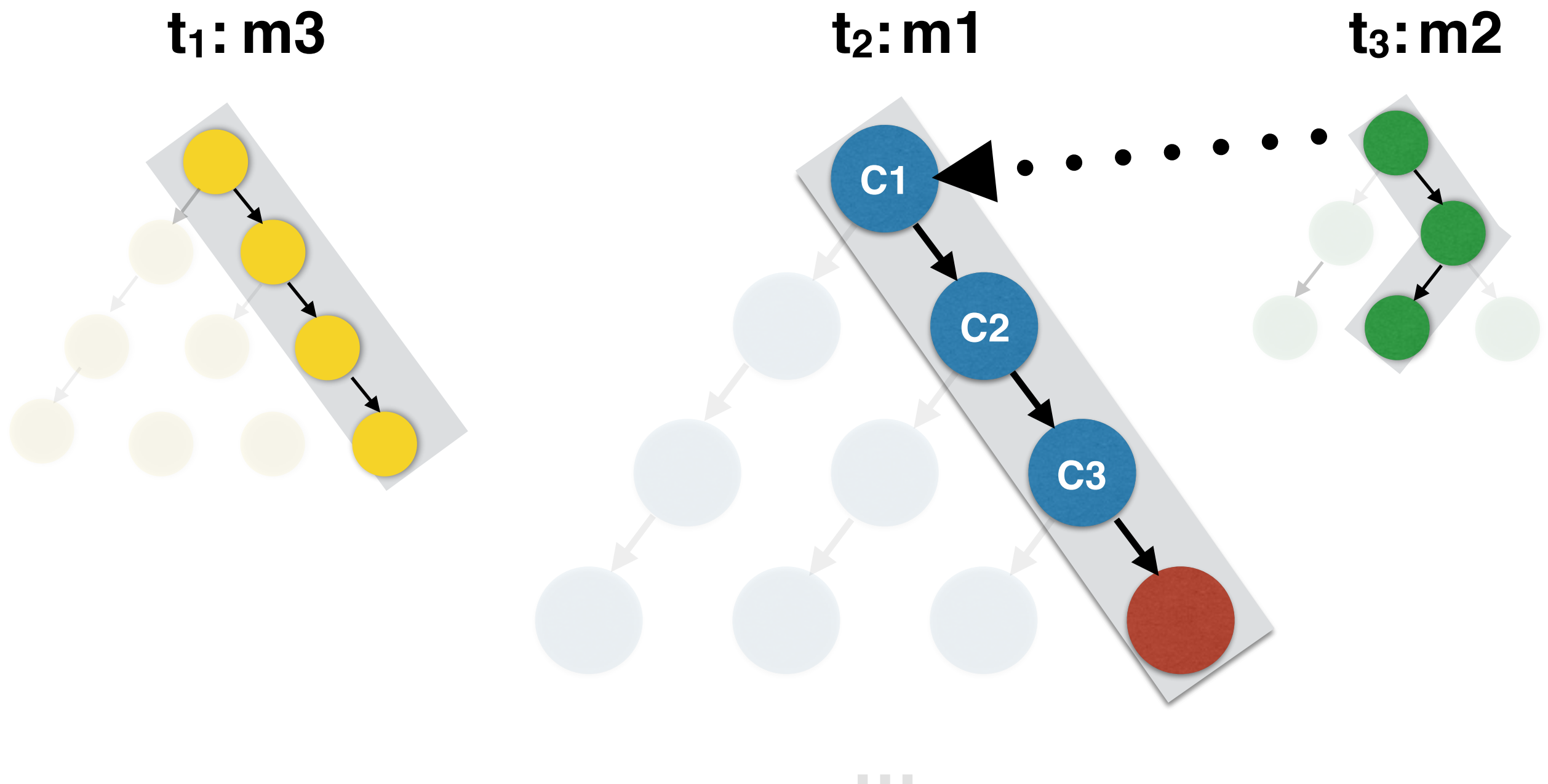


t₃: m2



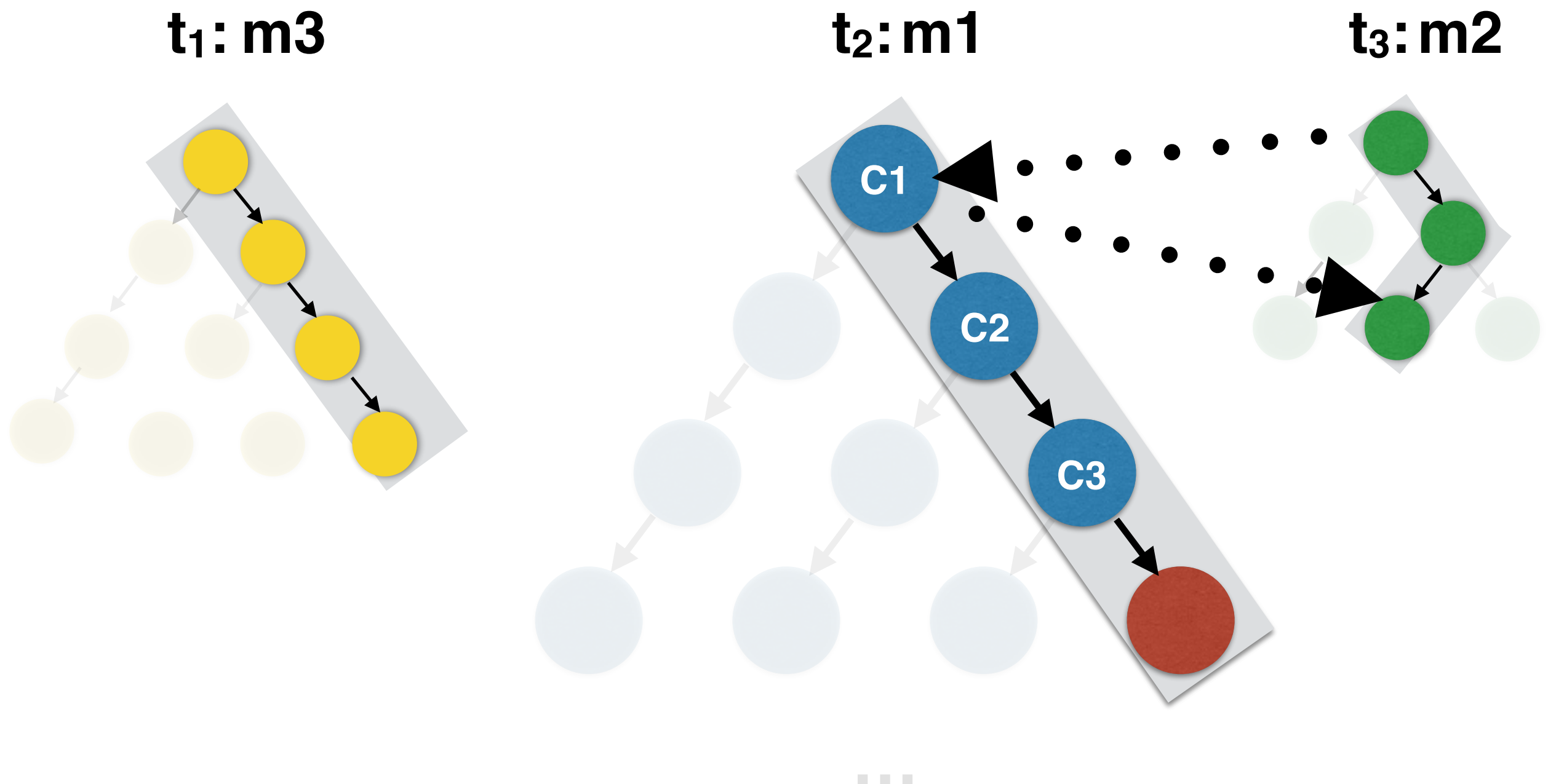
Execution synthesis

- Methods invoked from different threads appropriately
- Specific interleaving needs to be followed



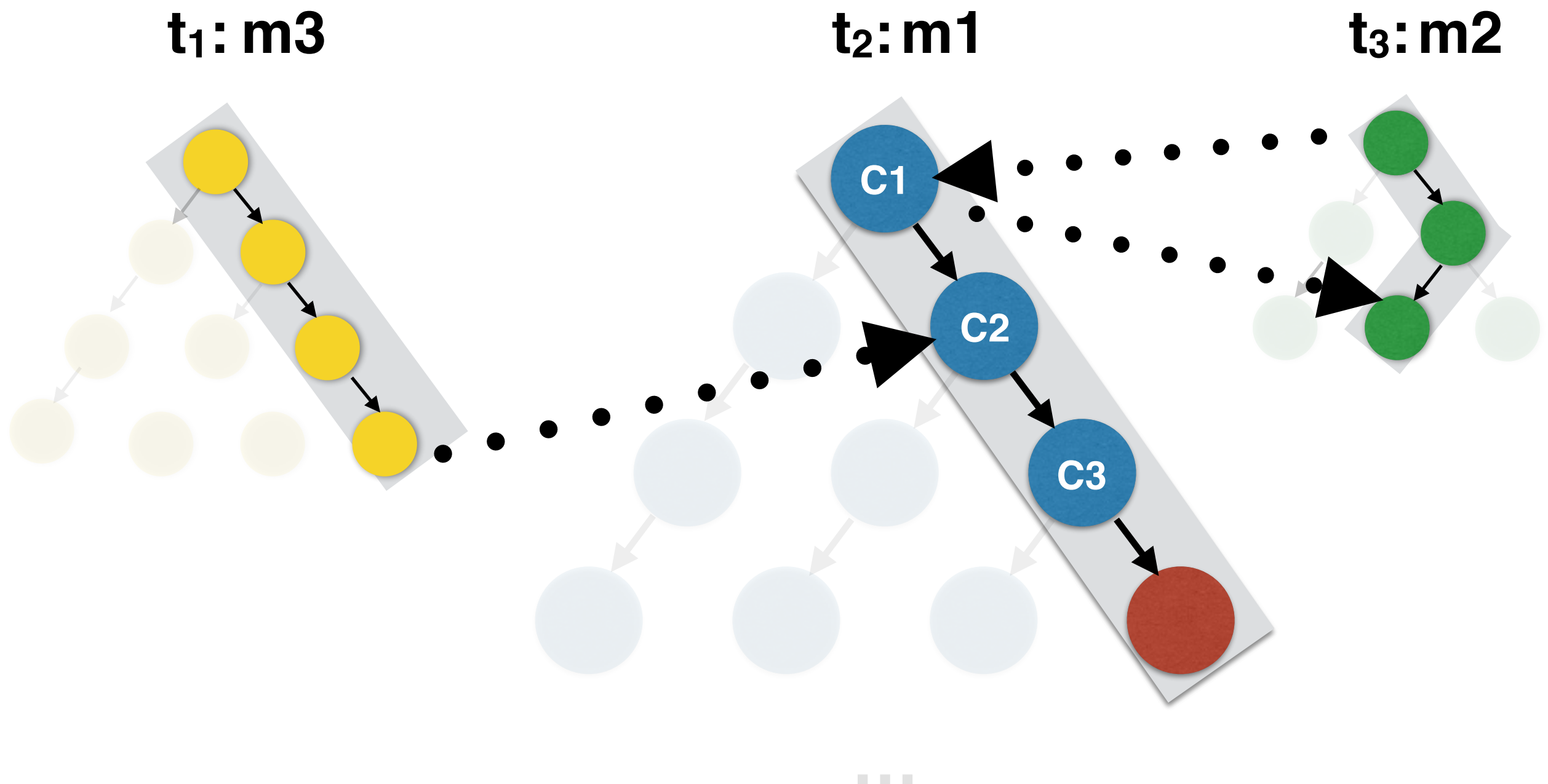
Execution synthesis

- Methods invoked from different threads appropriately
- Specific interleaving needs to be followed



Execution synthesis

- Methods invoked from different threads appropriately
- Specific interleaving needs to be followed



Approach Overview

Approach Overview

- ◆ **Static analysis**

Approach Overview

- ◆ **Static analysis**

- ◆ targets: locate assertions to violate and updates to fields

Approach Overview

- ◆ **Static analysis**

- ◆ targets: locate assertions to violate and updates to fields
- ◆ derive path conditions to reach target instructions

Approach Overview

- ◆ **Static analysis**

- ◆ targets: locate assertions to violate and updates to fields
- ◆ derive path conditions to reach target instructions

- ◆ **Dynamic analysis**

Approach Overview

- ◆ **Static analysis**

- ◆ targets: locate assertions to violate and updates to fields
- ◆ derive path conditions to reach target instructions

- ◆ **Dynamic analysis**

- ◆ obtain concrete data by executing provided tests

Approach Overview

- ◆ **Static analysis**

- ◆ targets: locate assertions to violate and updates to fields
- ◆ derive path conditions to reach target instructions

- ◆ **Dynamic analysis**

- ◆ obtain concrete data by executing provided tests
 - ◆ path conditions traversed, value of fields, etc

Approach Overview

- ◆ **Static analysis**

- ◆ targets: locate assertions to violate and updates to fields
- ◆ derive path conditions to reach target instructions

- ◆ **Dynamic analysis**

- ◆ obtain concrete data by executing provided tests
 - ◆ path conditions traversed, value of fields, etc

- ◆ **Constraint solvers**

Approach Overview

- ◆ **Static analysis**

- ◆ targets: locate assertions to violate and updates to fields
- ◆ derive path conditions to reach target instructions

- ◆ **Dynamic analysis**

- ◆ obtain concrete data by executing provided tests
 - ◆ path conditions traversed, value of fields, etc

- ◆ **Constraint solvers**

- ◆ encode path constraints, read-write constraints, lock constraints and parameter constraints using information from static and dynamic analysis

Approach Overview

- ◆ **Static analysis**

- ◆ targets: locate assertions to violate and updates to fields
- ◆ derive path conditions to reach target instructions

- ◆ **Dynamic analysis**

- ◆ obtain concrete data by executing provided tests
 - ◆ path conditions traversed, value of fields, etc

- ◆ **Constraint solvers**

- ◆ encode path constraints, read-write constraints, lock constraints and parameter constraints using information from static and dynamic analysis
- ◆ synthesize structure of new clients and schedules

Approach Overview

- ◆ **Static analysis**

- ◆ targets: locate assertions to violate and updates to fields
- ◆ derive path conditions to reach target instructions

- ◆ **Dynamic analysis**

- ◆ obtain concrete data by executing provided tests
 - ◆ path conditions traversed, value of fields, etc

- ◆ **Constraint solvers**

- ◆ encode path constraints, read-write constraints, lock constraints and parameter constraints using information from static and dynamic analysis
- ◆ synthesize structure of new clients and schedules

- ◆ **Leverage the above components; iterate until target is reached**

Iteration - 1 : Static analysis

m1

m2

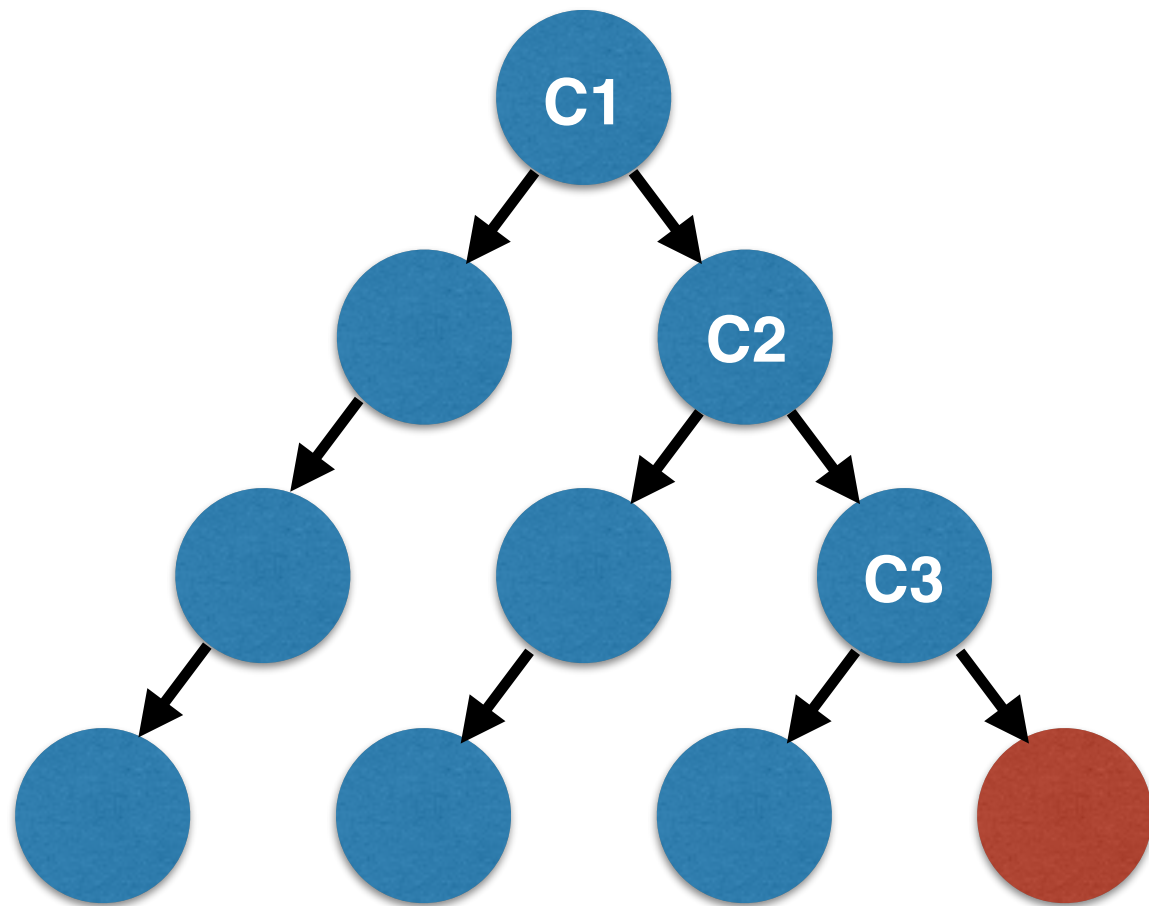
m3

Iteration - 1 : Static analysis

m1

m2

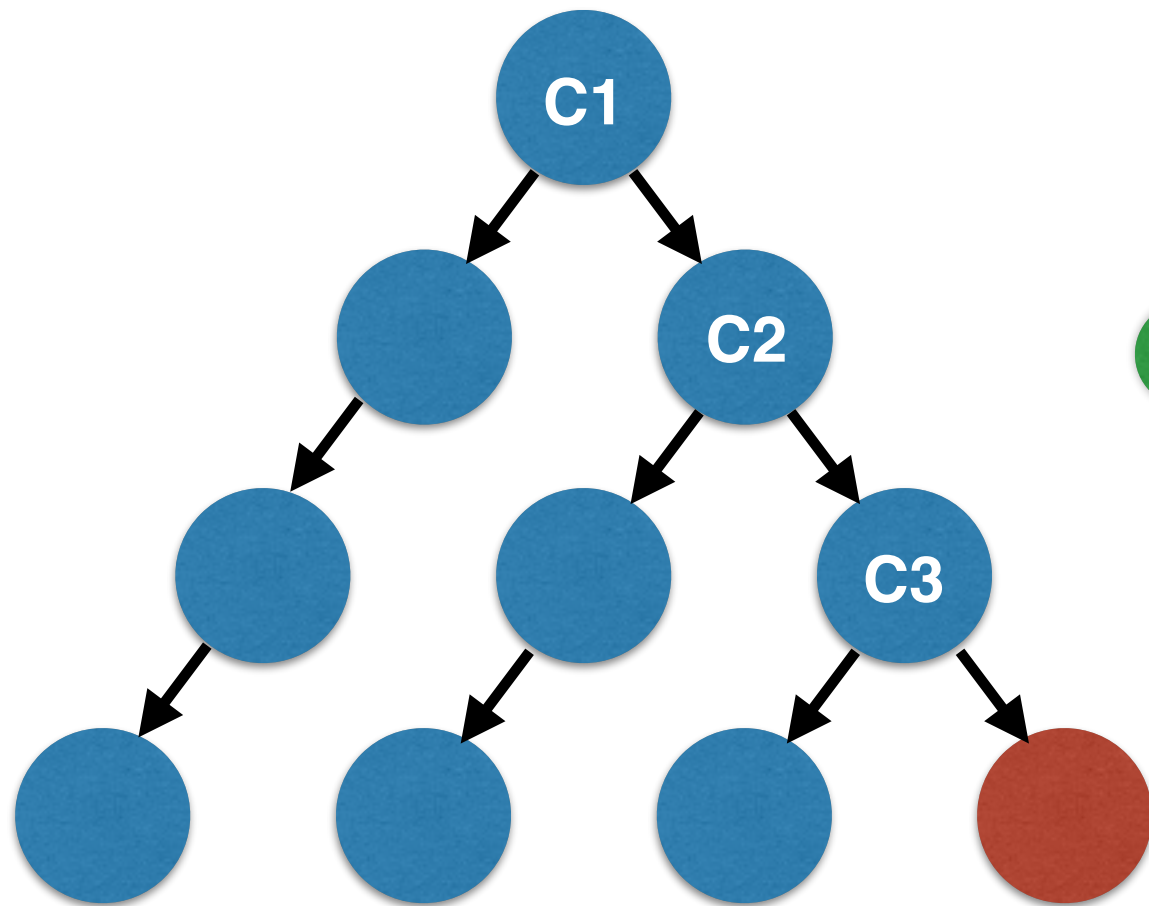
m3



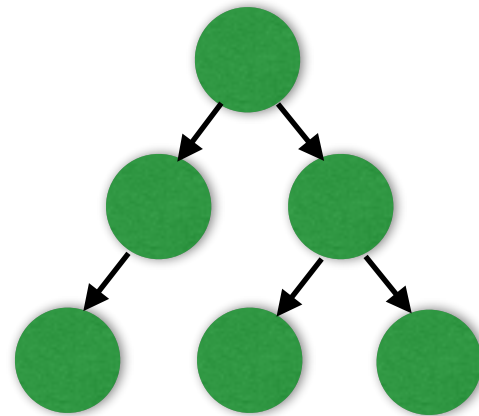
■ ■ ■

Iteration - 1 : Static analysis

m1



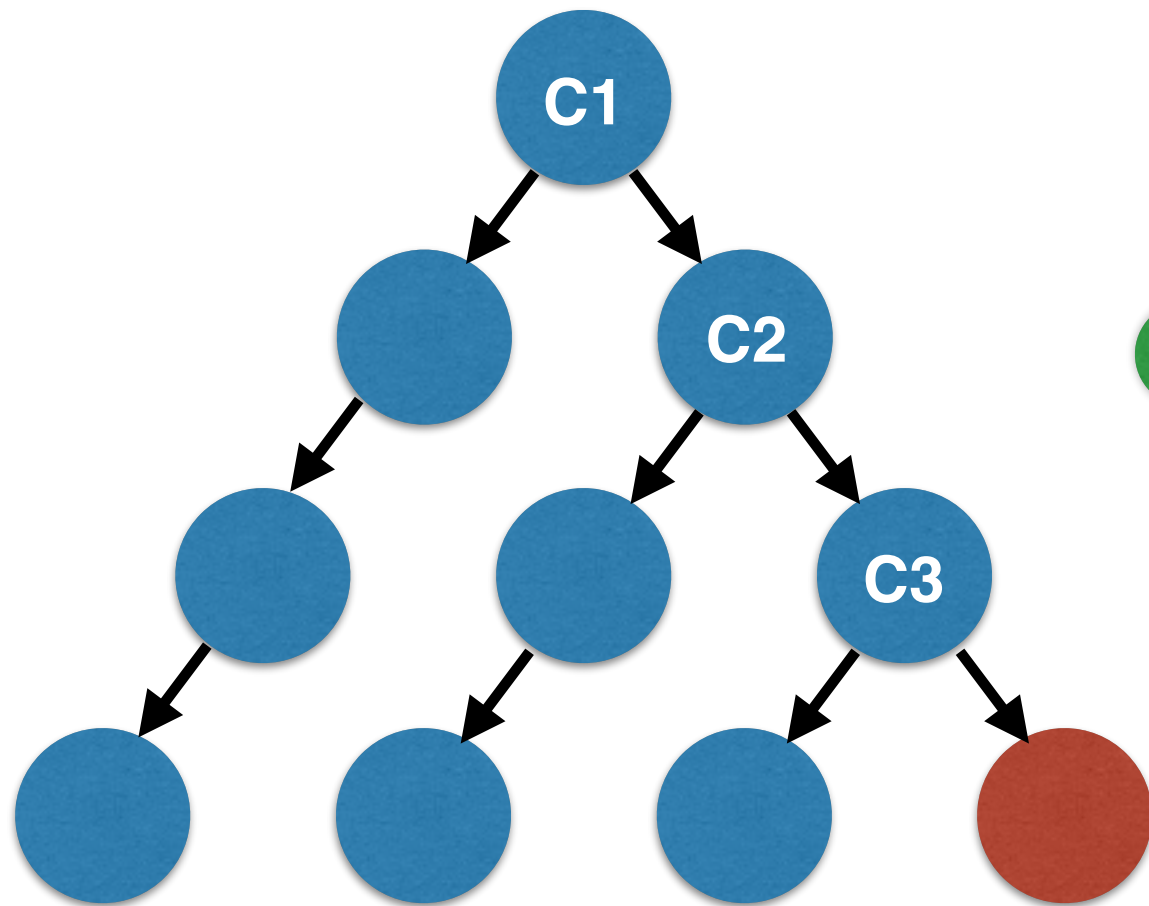
m2



m3

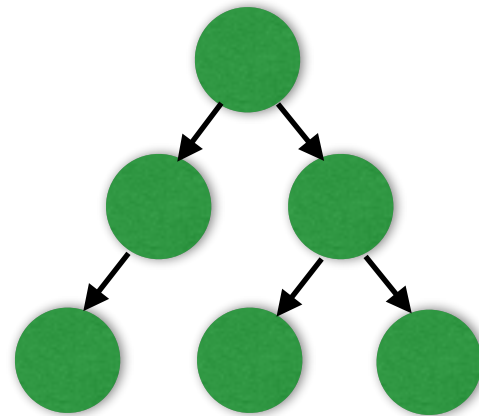
Iteration - 1 : Static analysis

m1

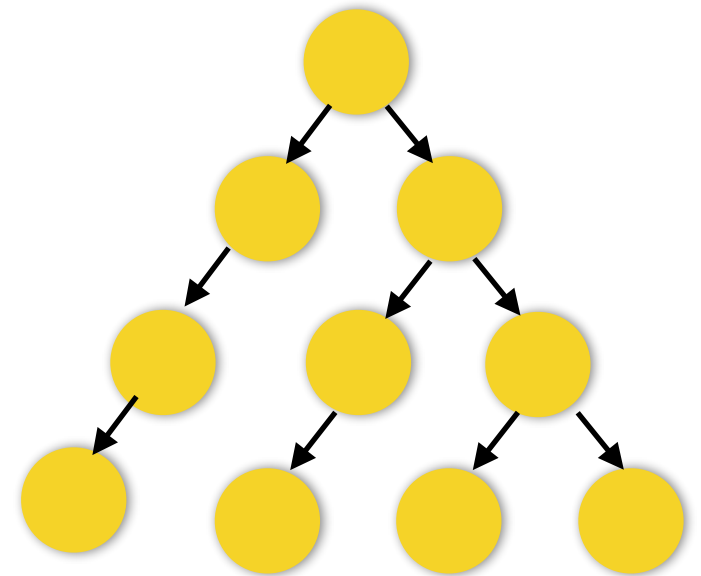


...

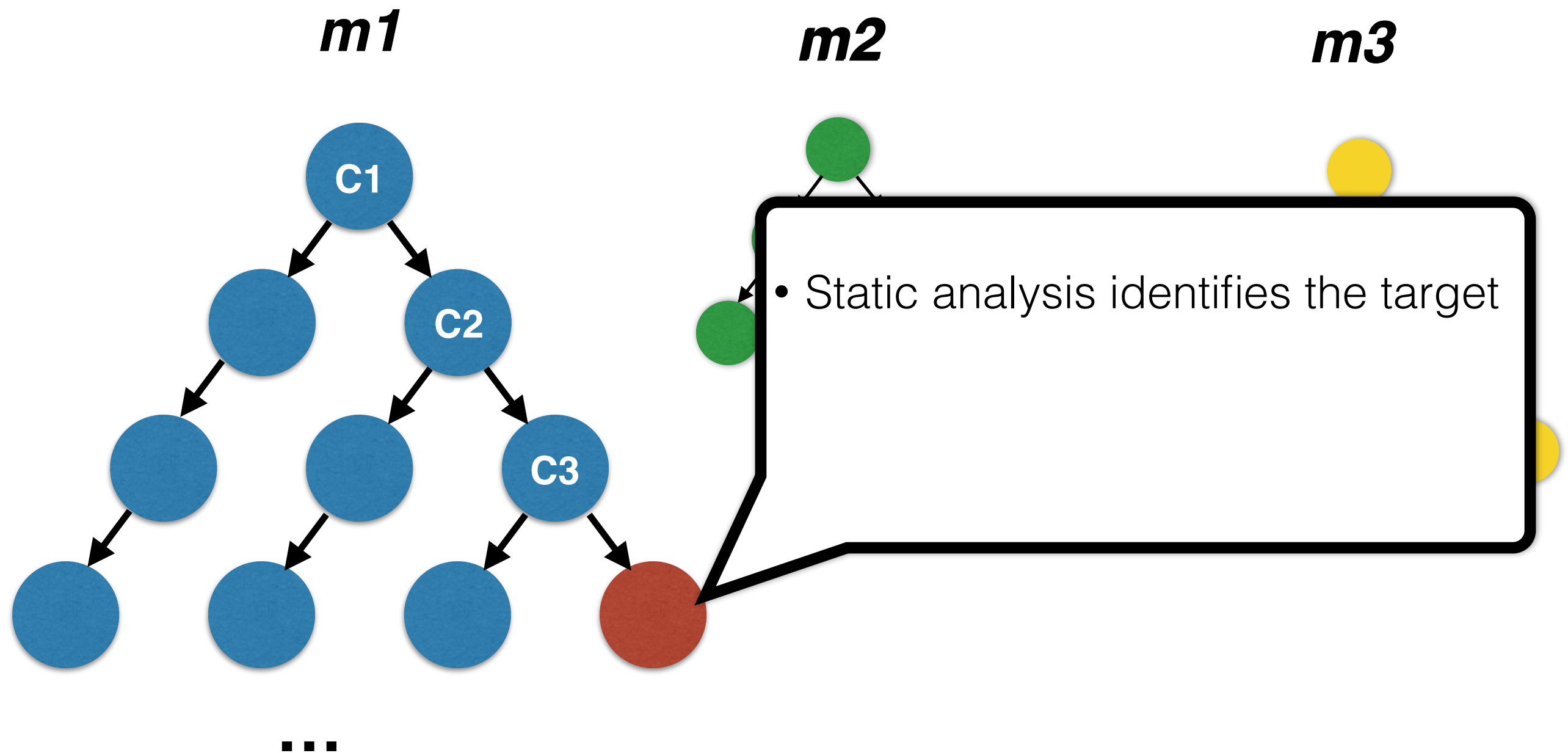
m2



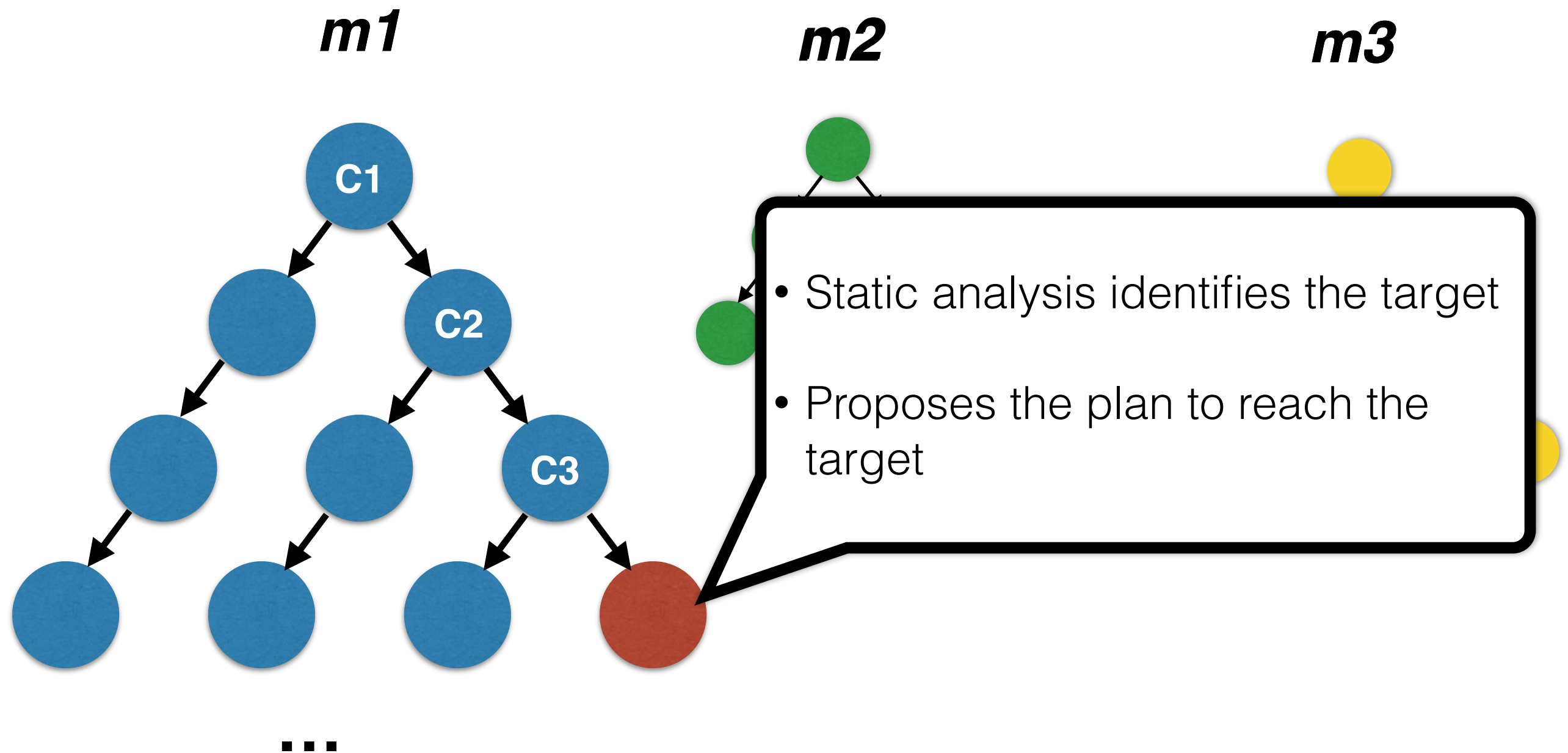
m3



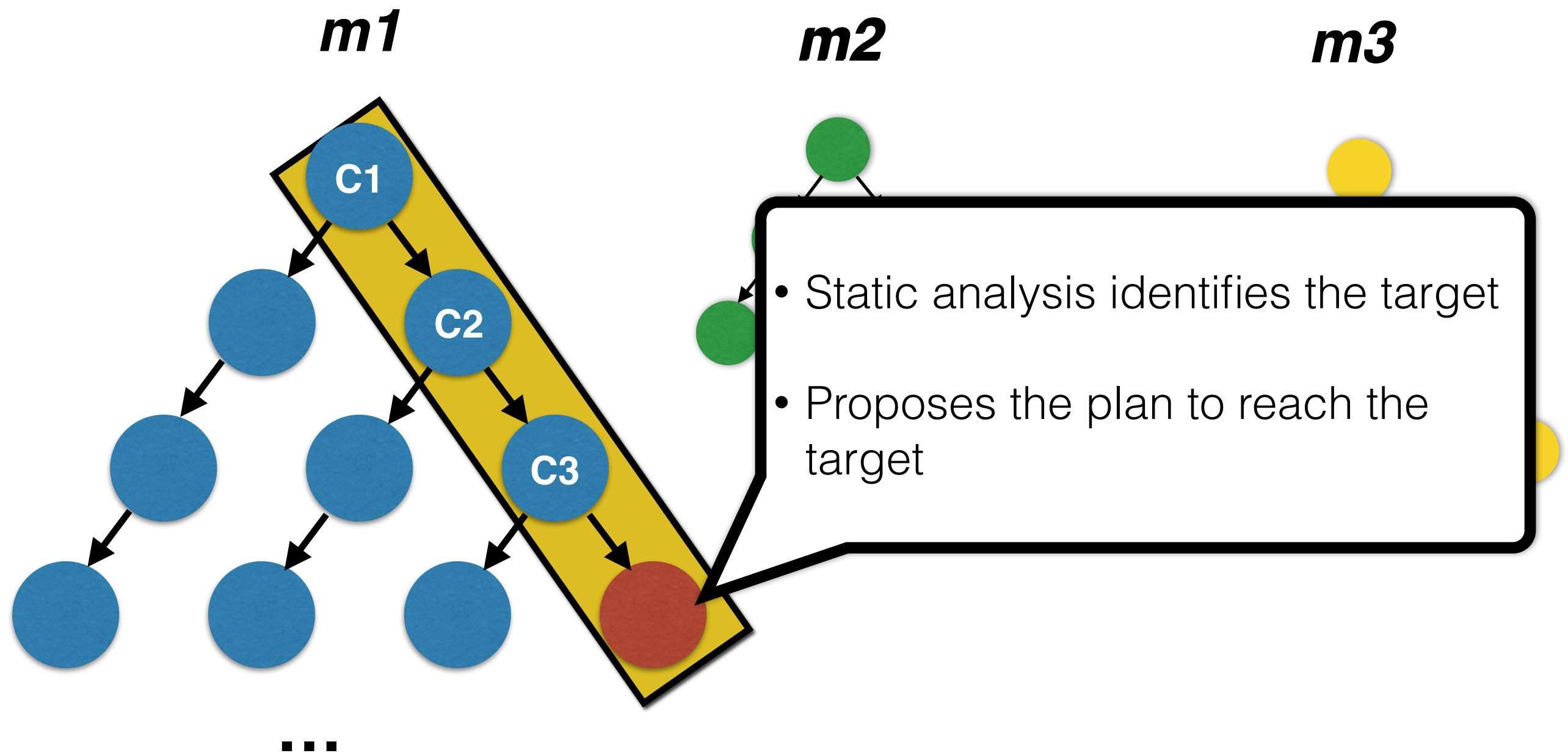
Iteration - 1 : Static analysis



Iteration - 1 : Static analysis

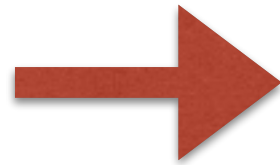
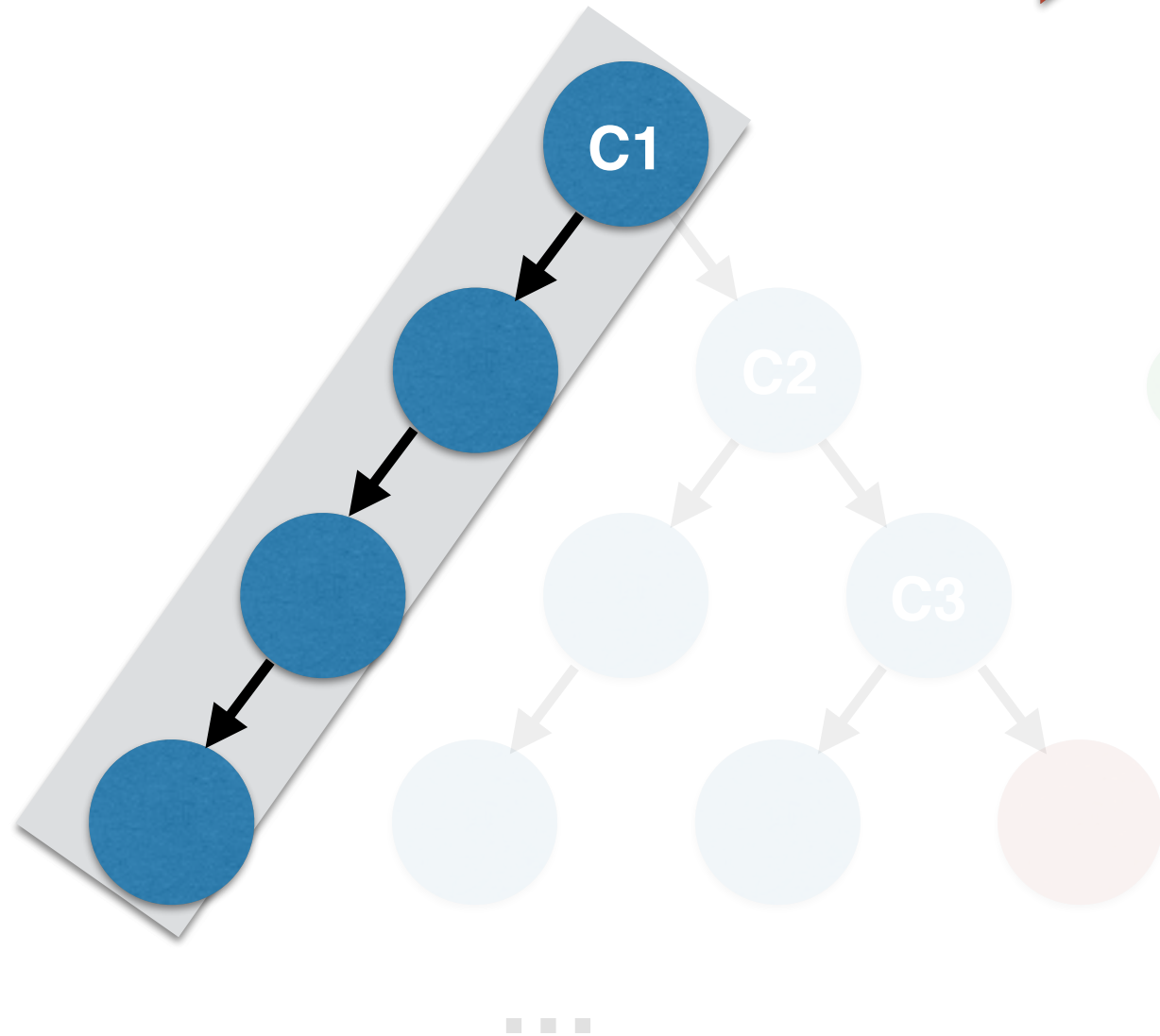


Iteration - 1 : Static analysis

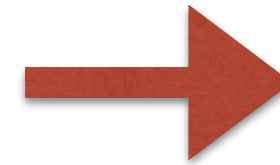
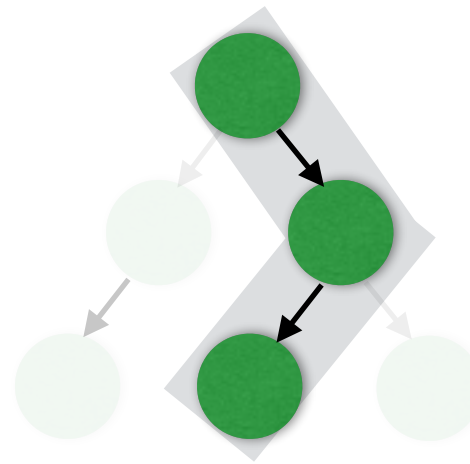


Iteration - 1 : Concrete Execution

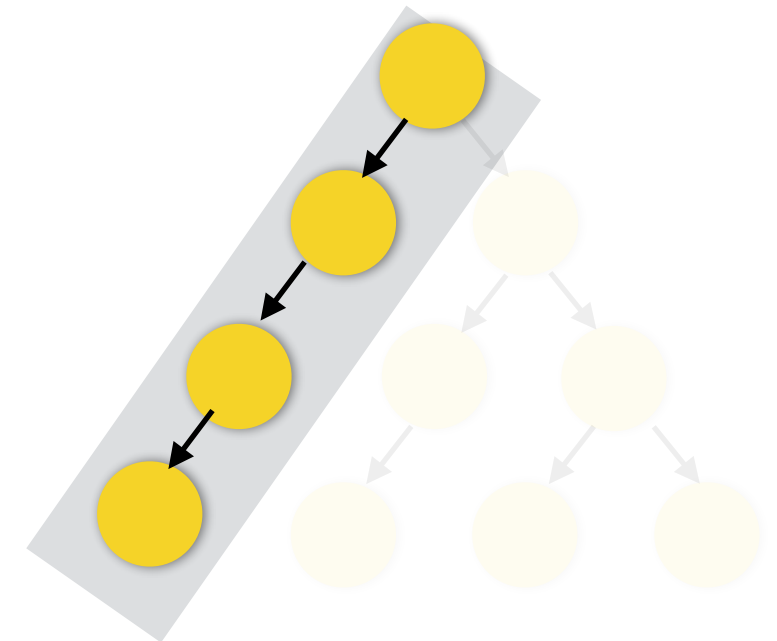
t_1 : *m1*



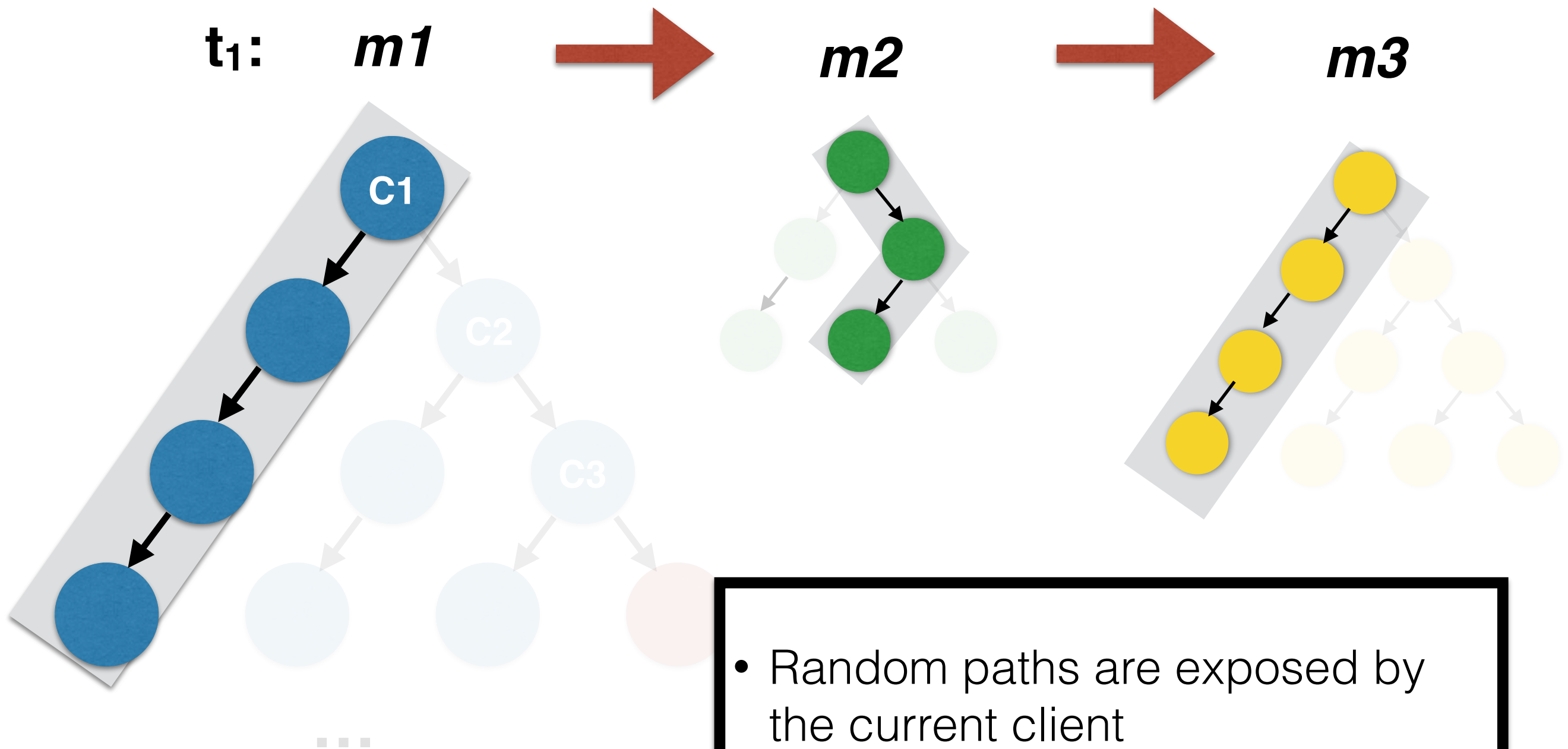
m2



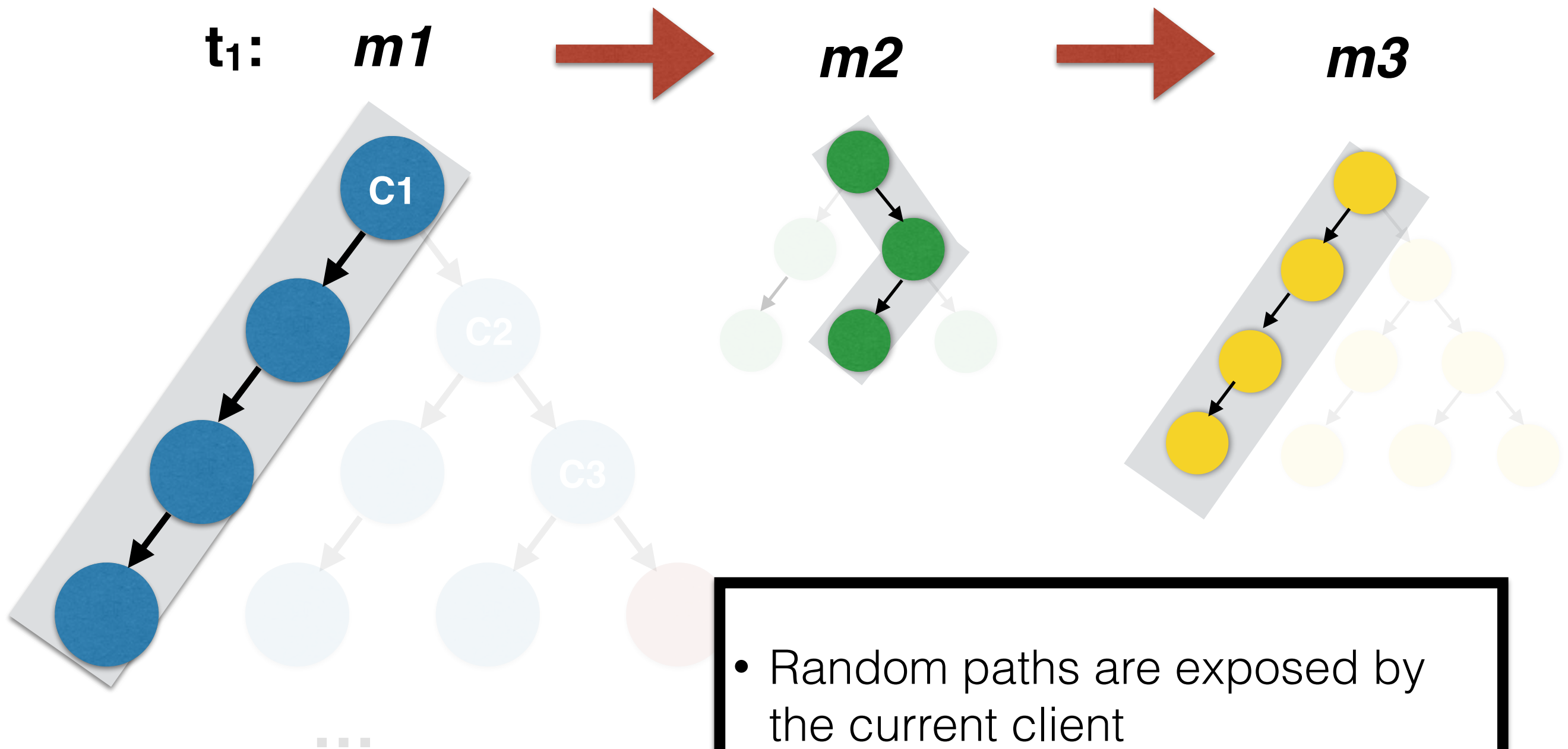
m3



Iteration - 1 : Concrete Execution

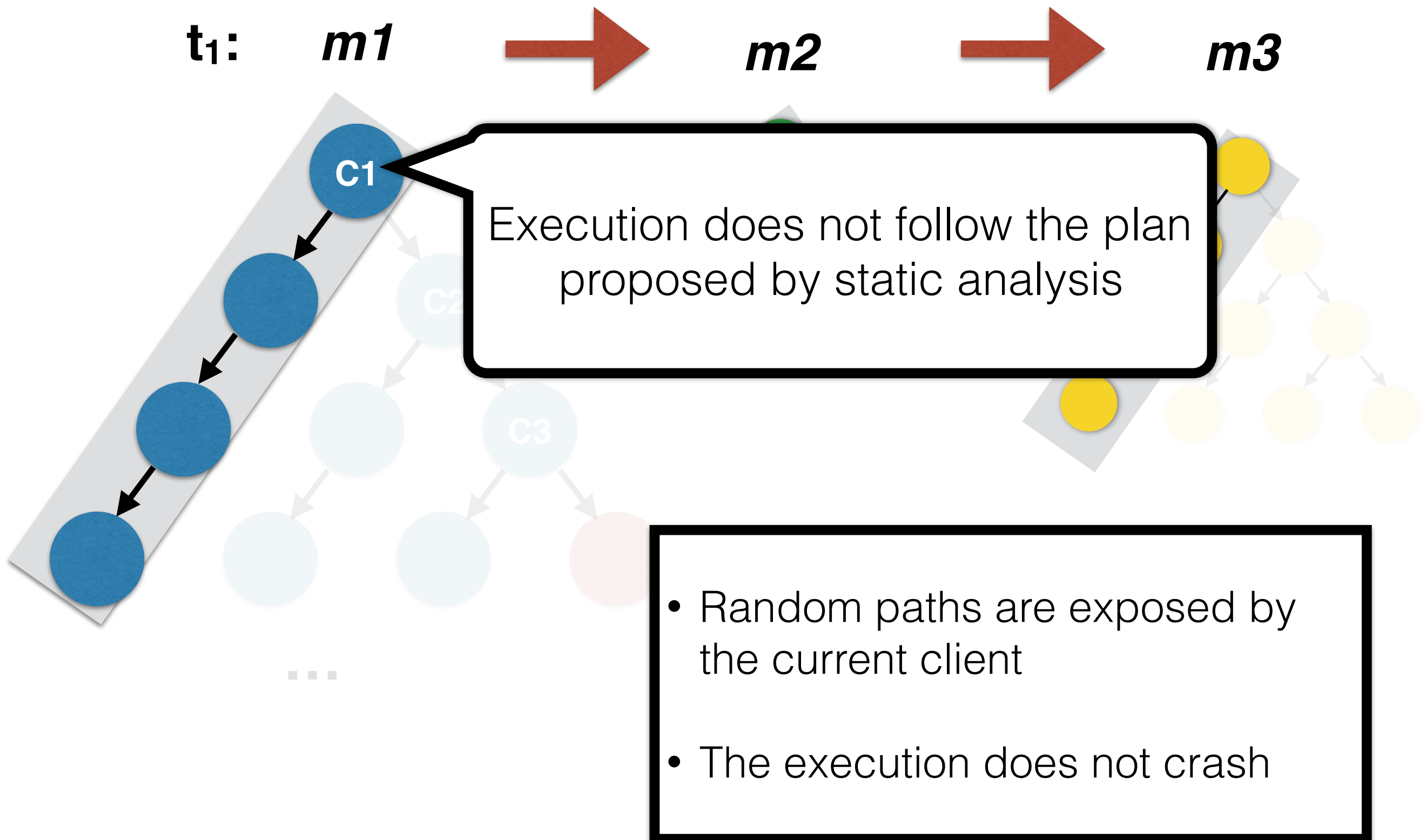


Iteration - 1 : Concrete Execution



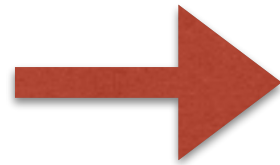
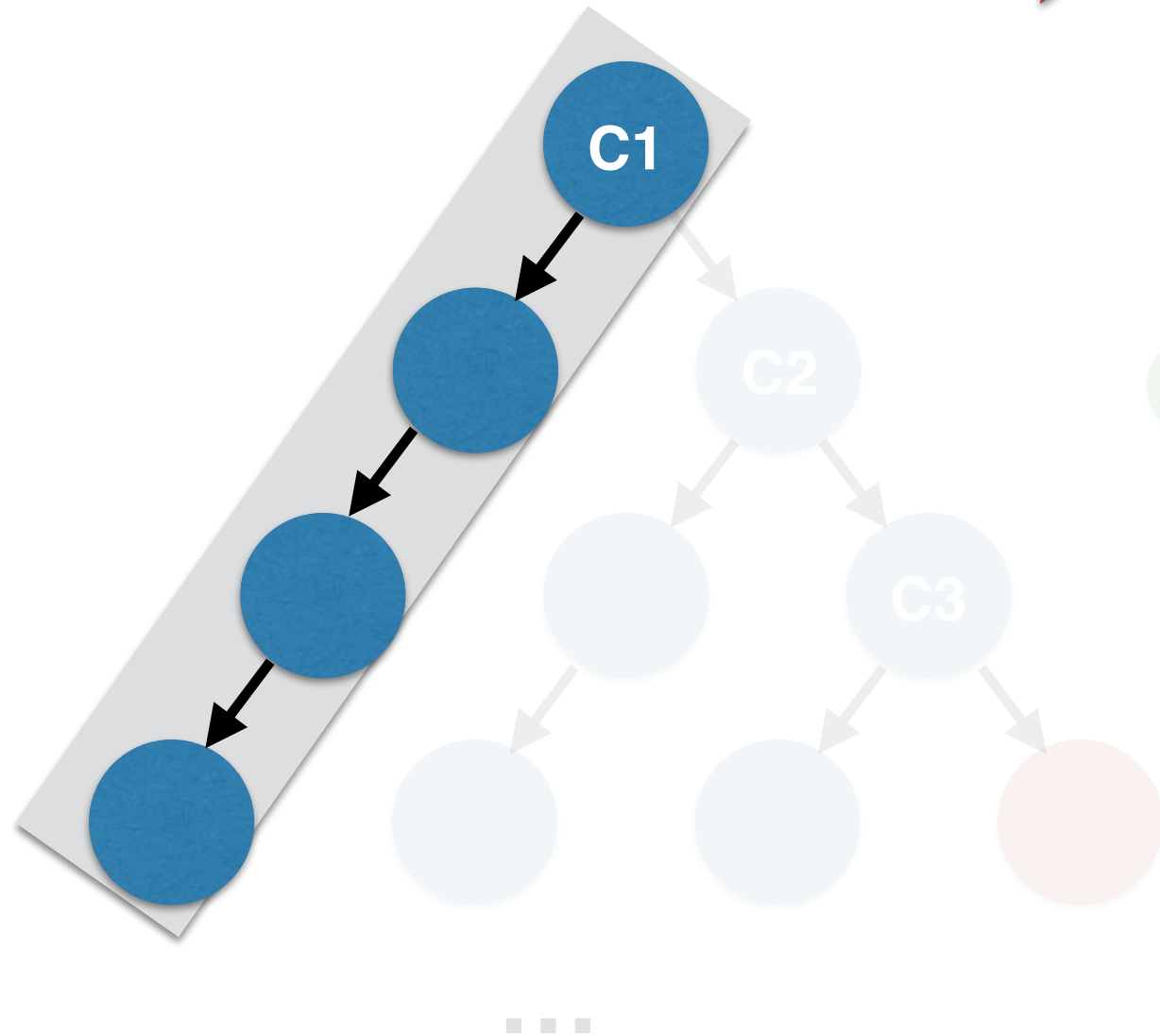
- Random paths are exposed by the current client
- The execution does not crash

Iteration - 1 : Concrete Execution

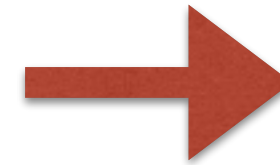
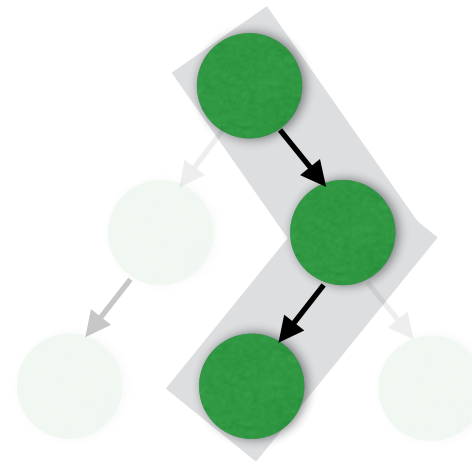


Iteration - 1 : Concrete Execution

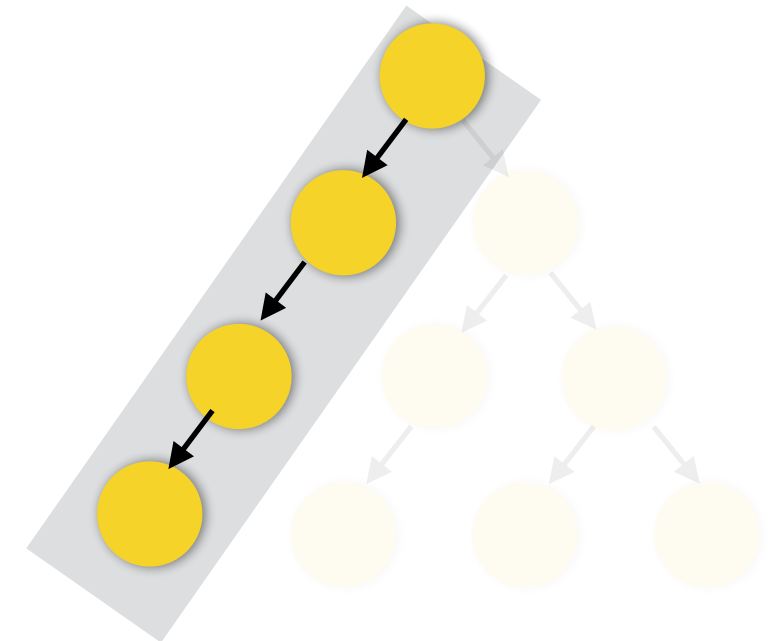
t_1 : *m1*



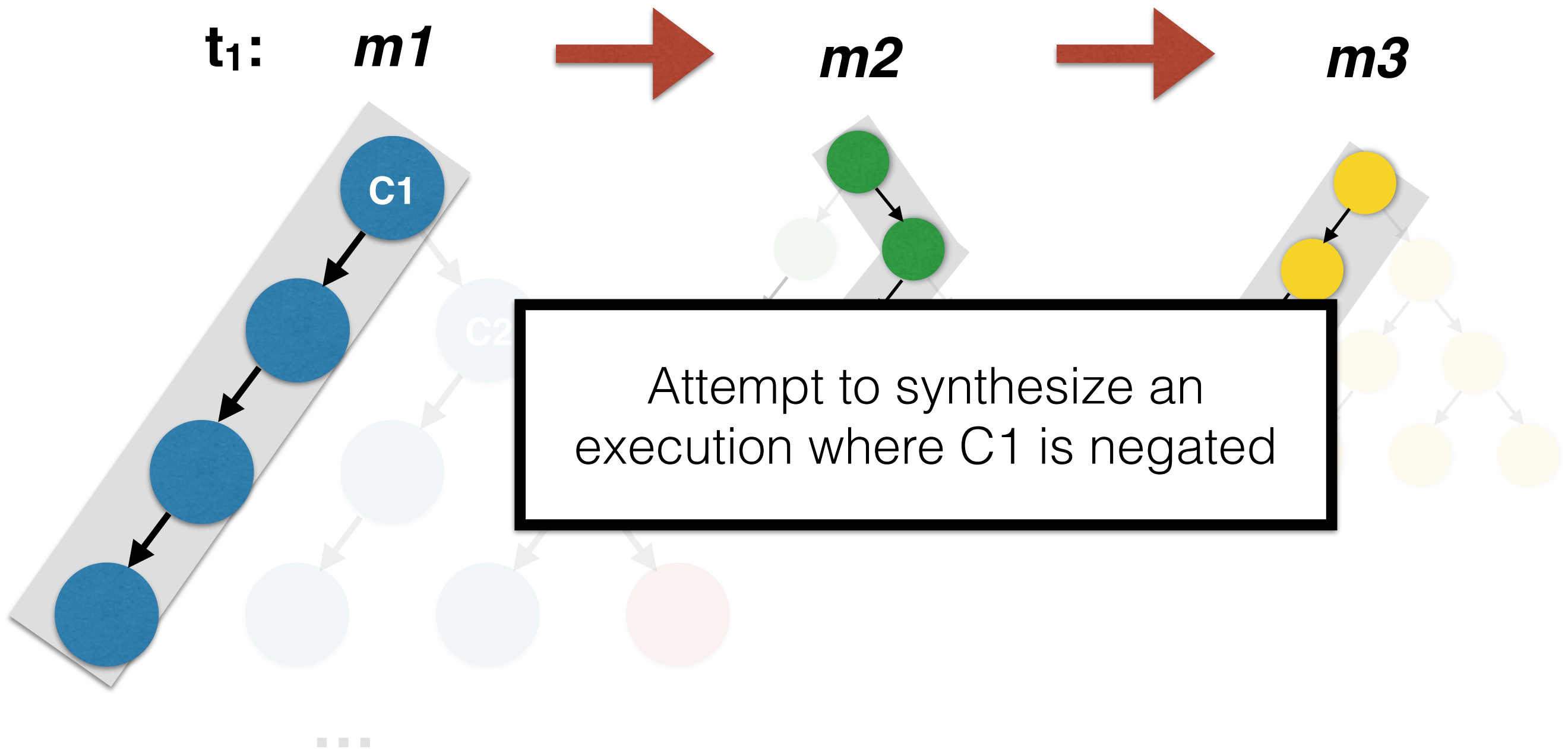
m2



m3

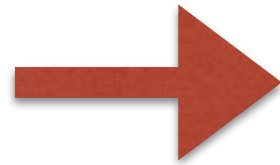
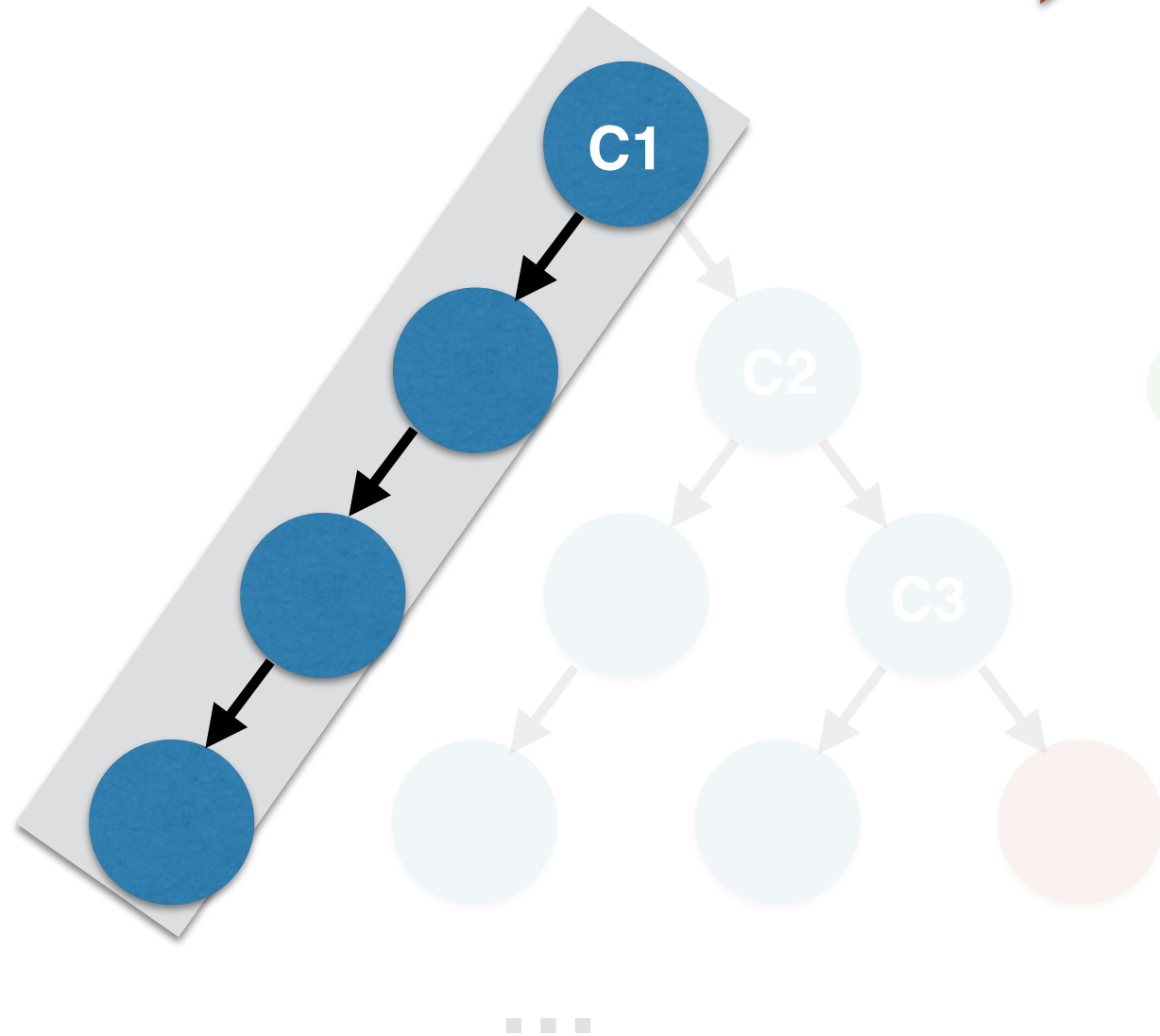


Iteration - 1 : Concrete Execution

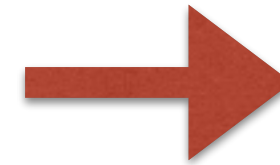
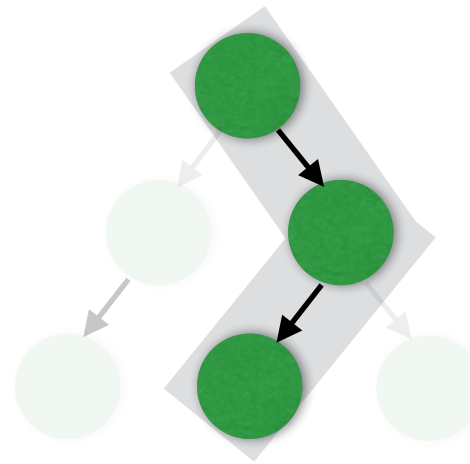


Iteration - 1 : Concrete Execution

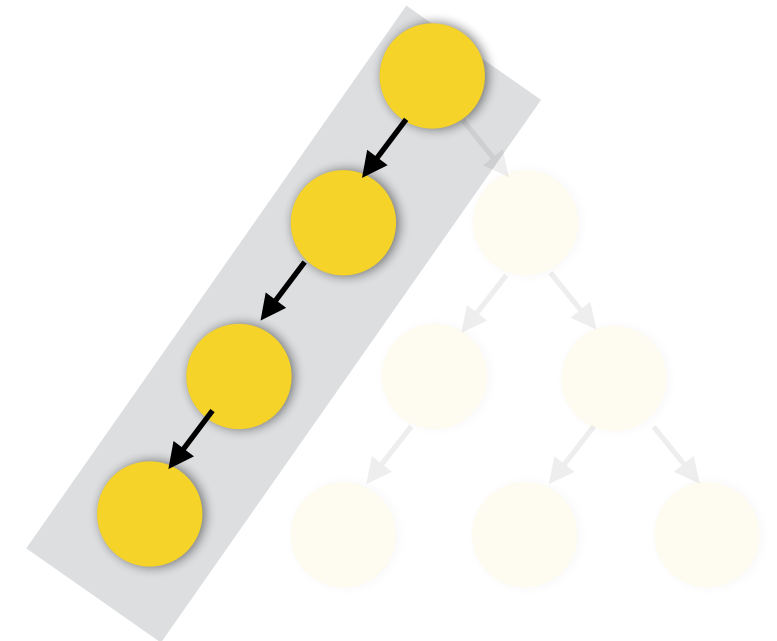
t_1 : *m1*



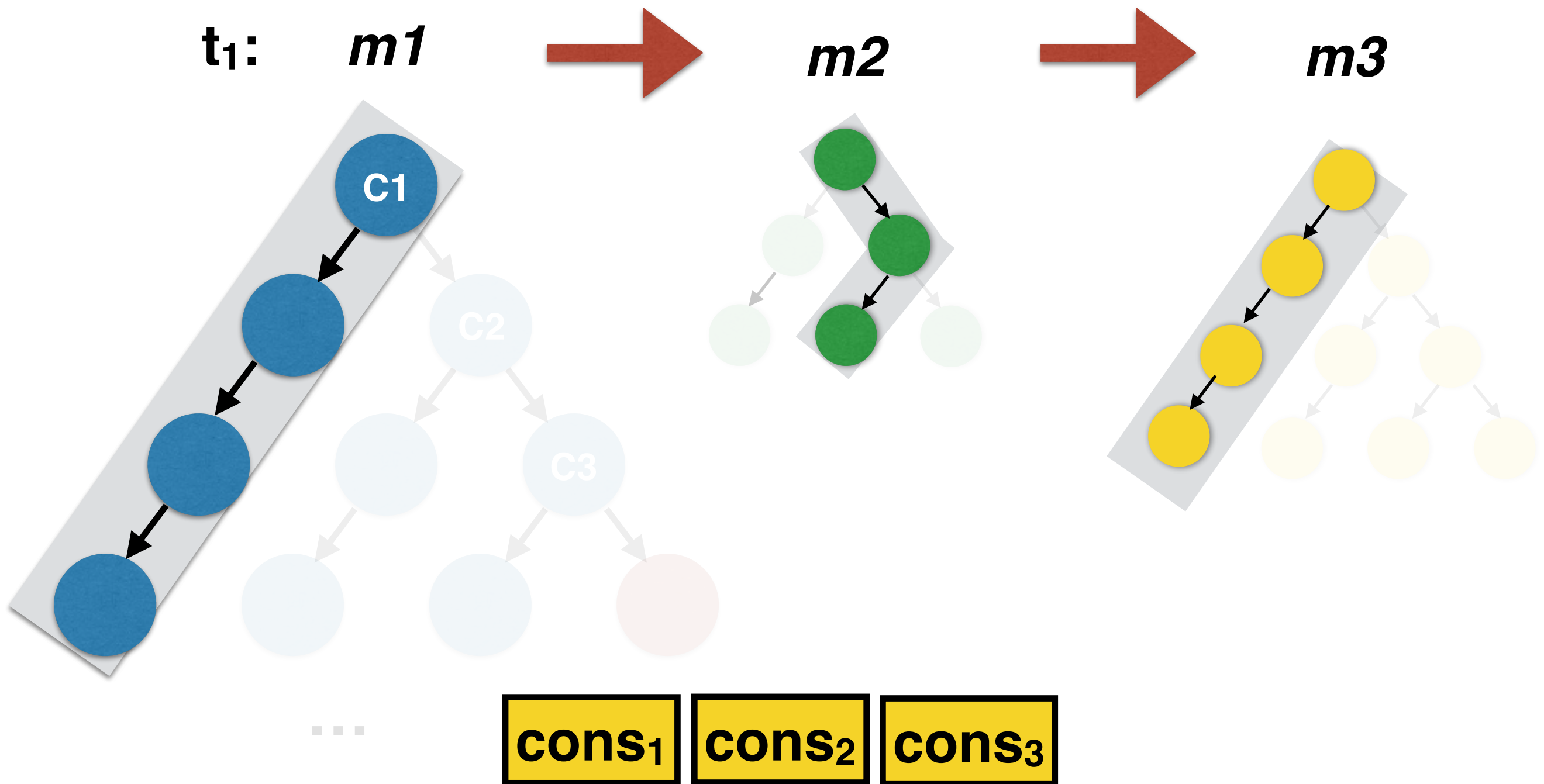
m2



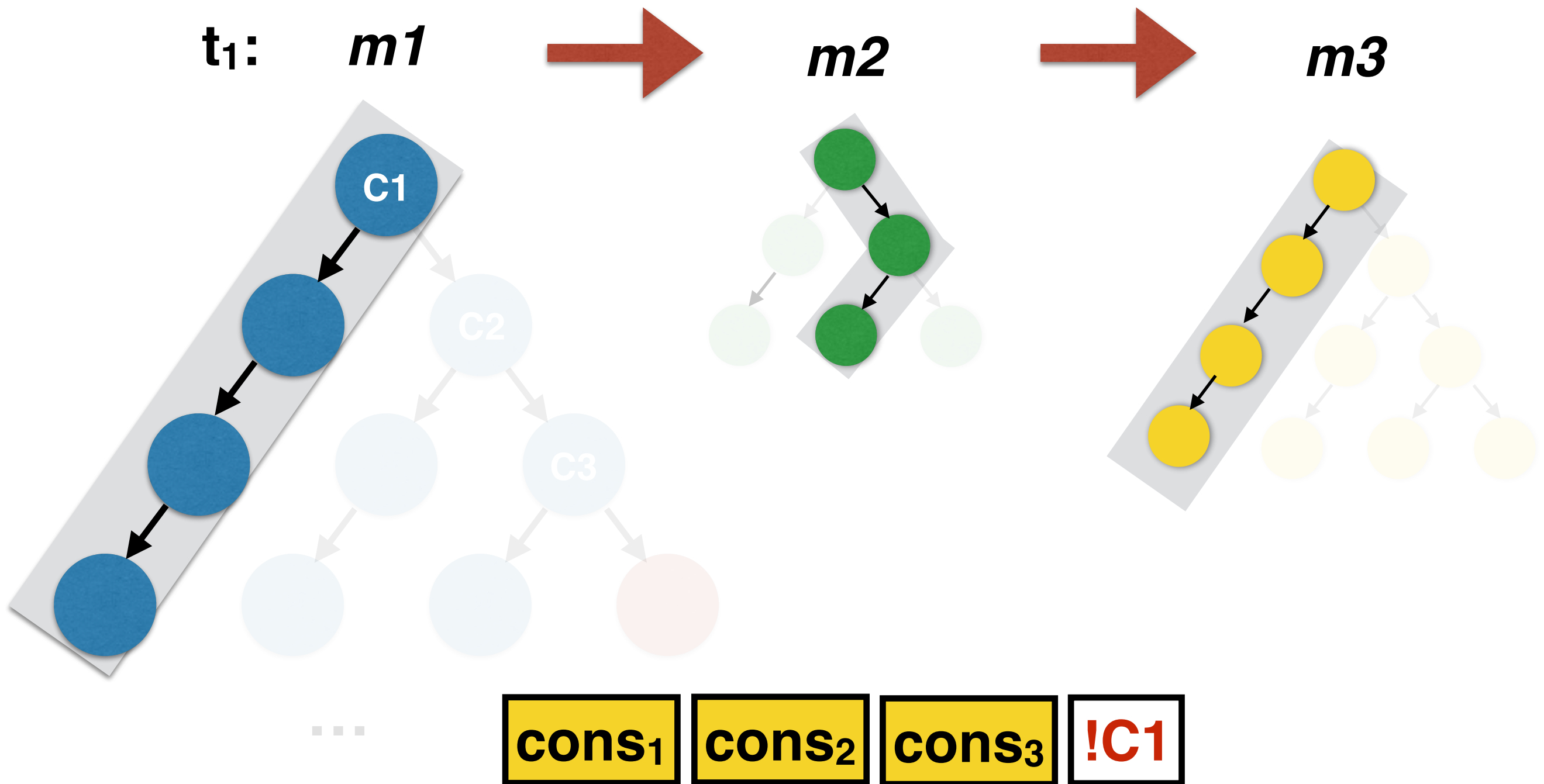
m3



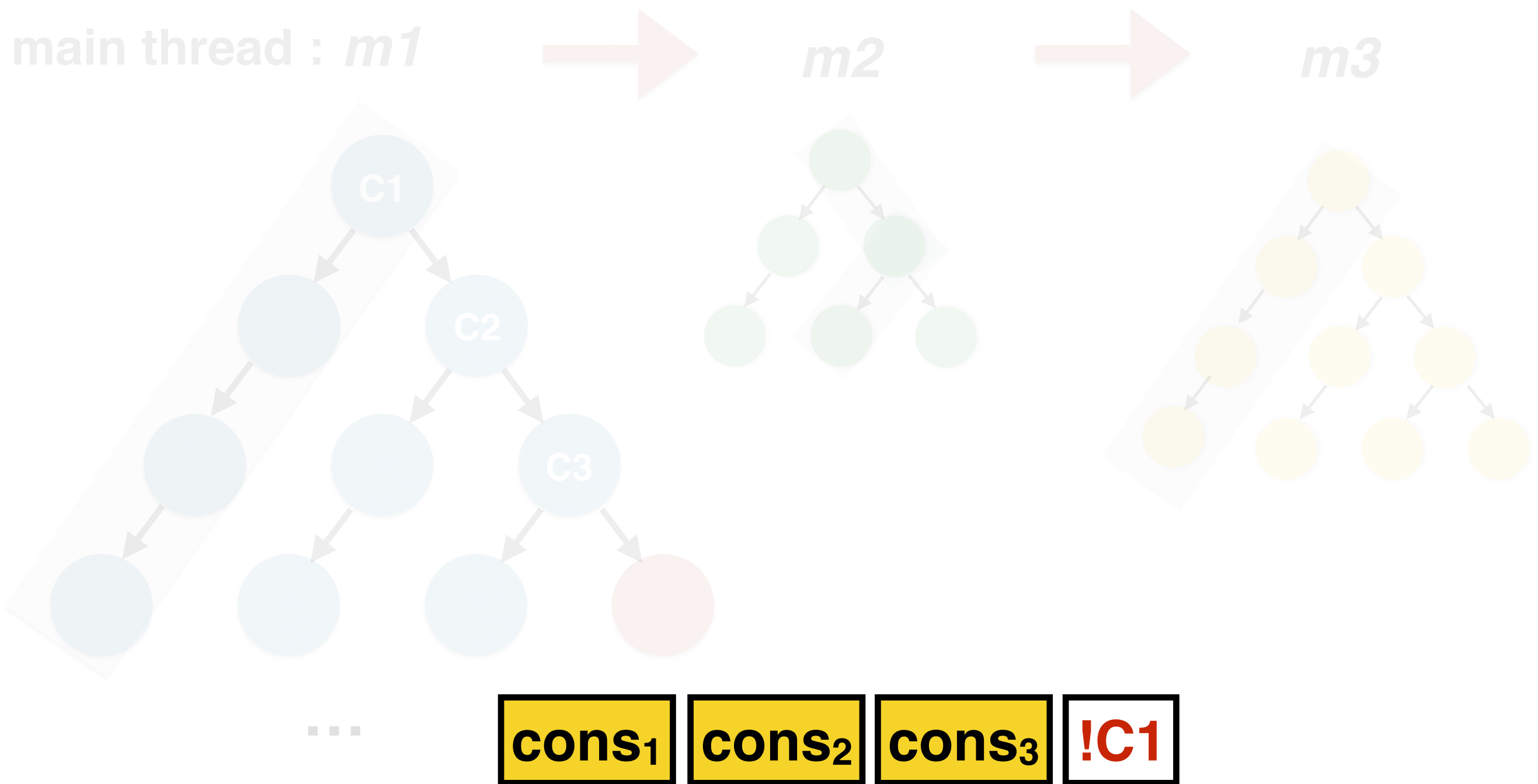
Iteration - 1 : Concrete Execution



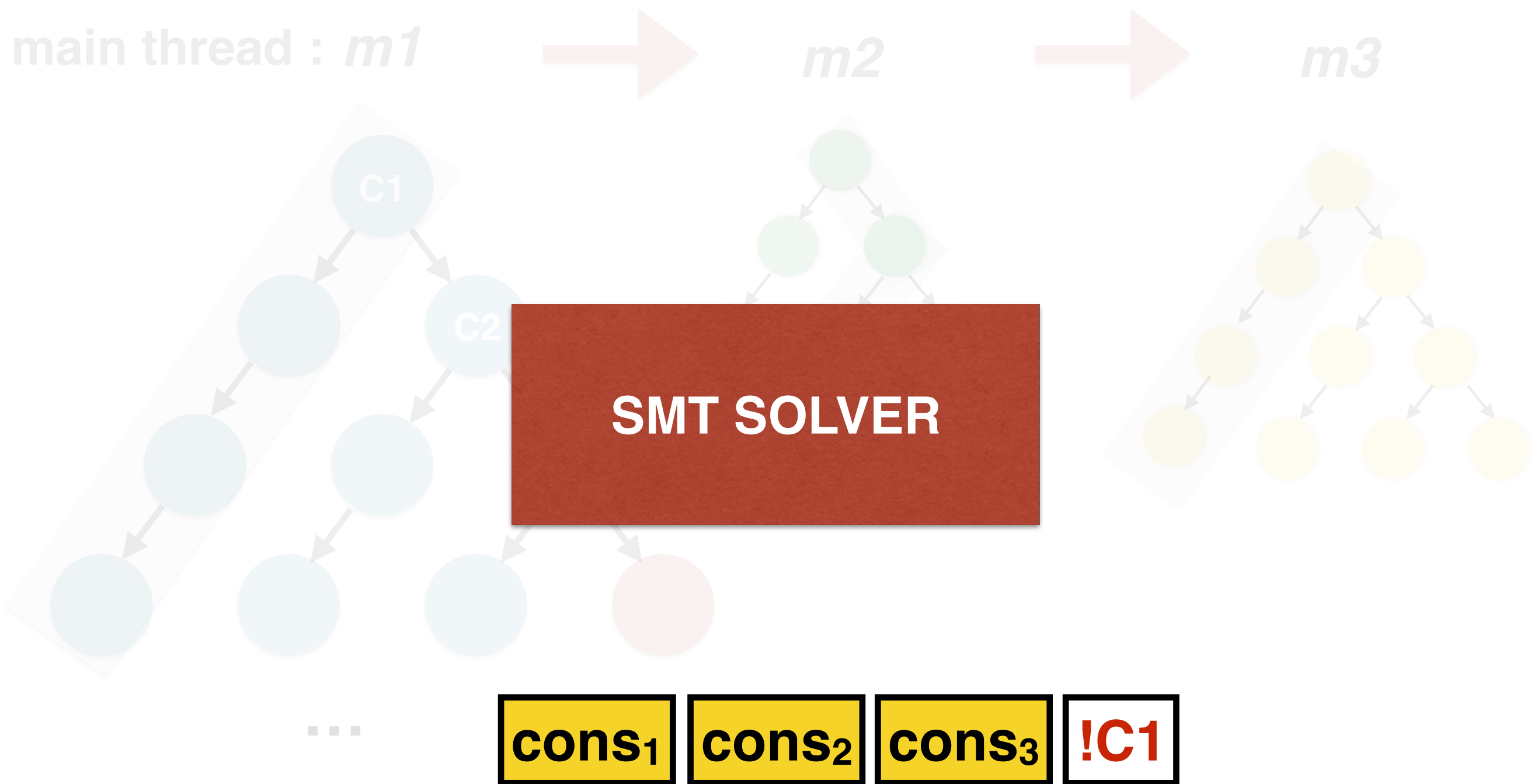
Iteration - 1 : Concrete Execution



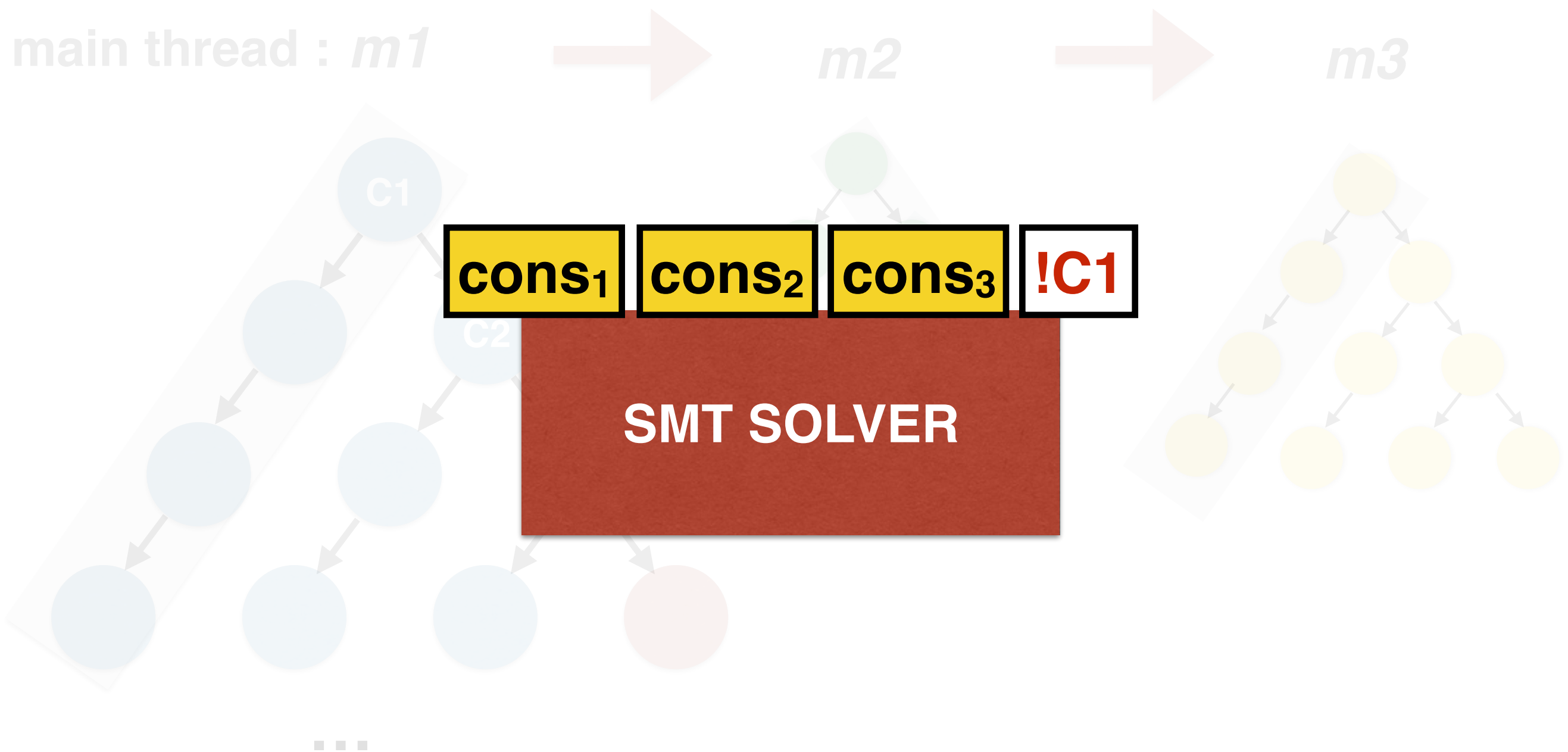
Iteration - 1 : Client Synthesis



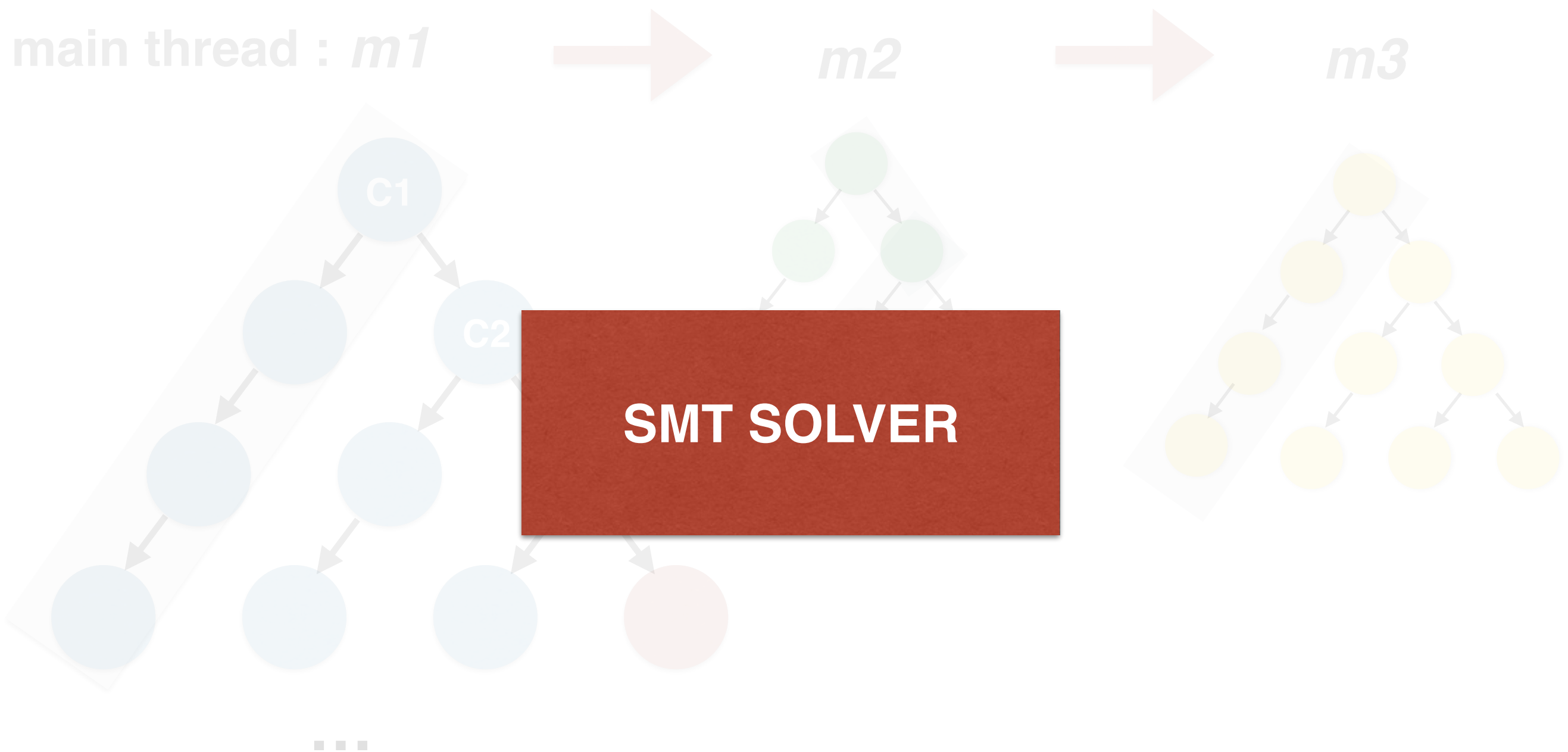
Iteration - 1 : Client Synthesis



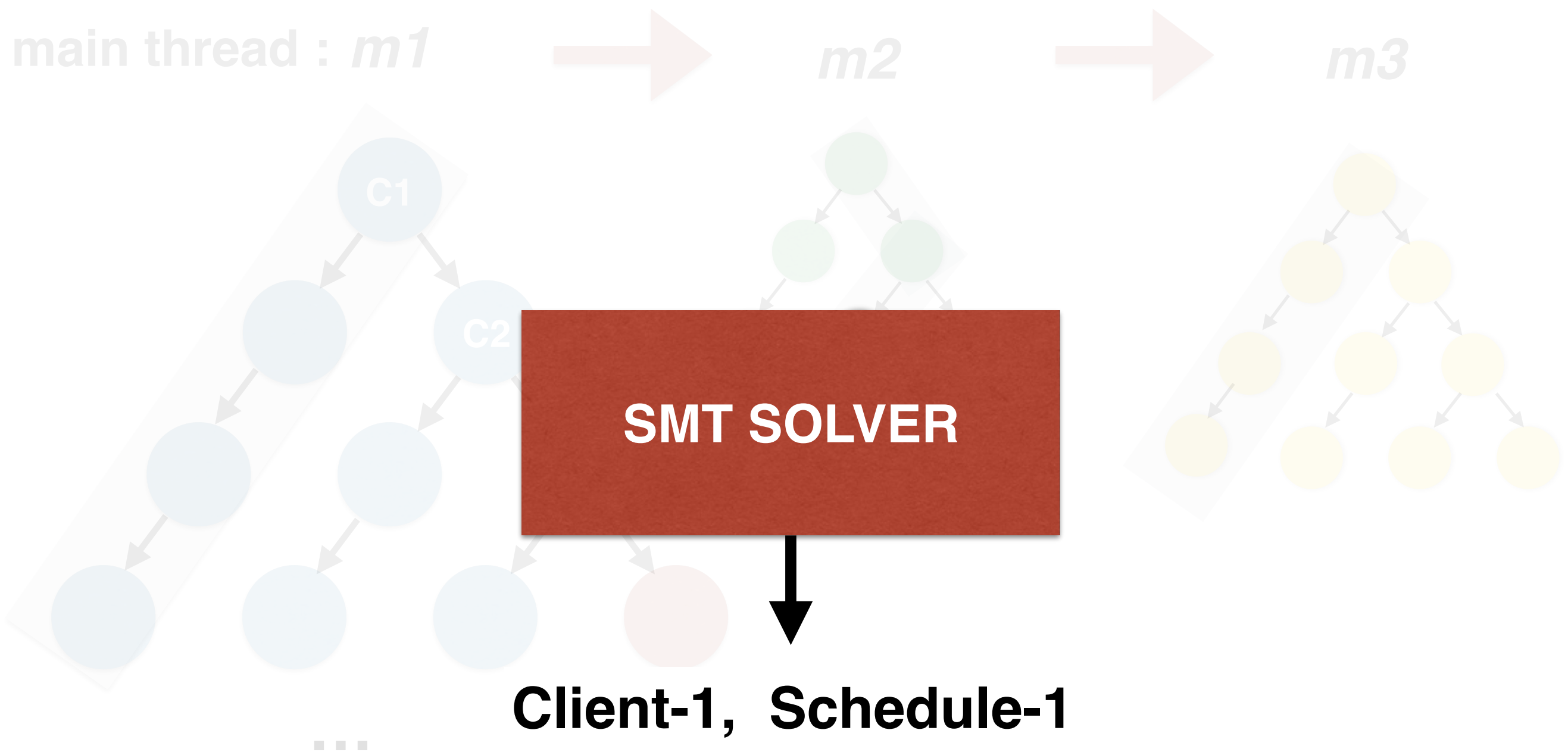
Iteration - 1 : Client Synthesis



Iteration - 1 : Client Synthesis



Iteration - 1 : Client Synthesis

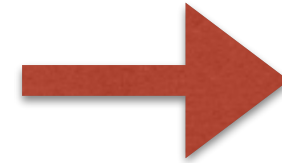


Iteration - 2 : Concrete Execution

New
client

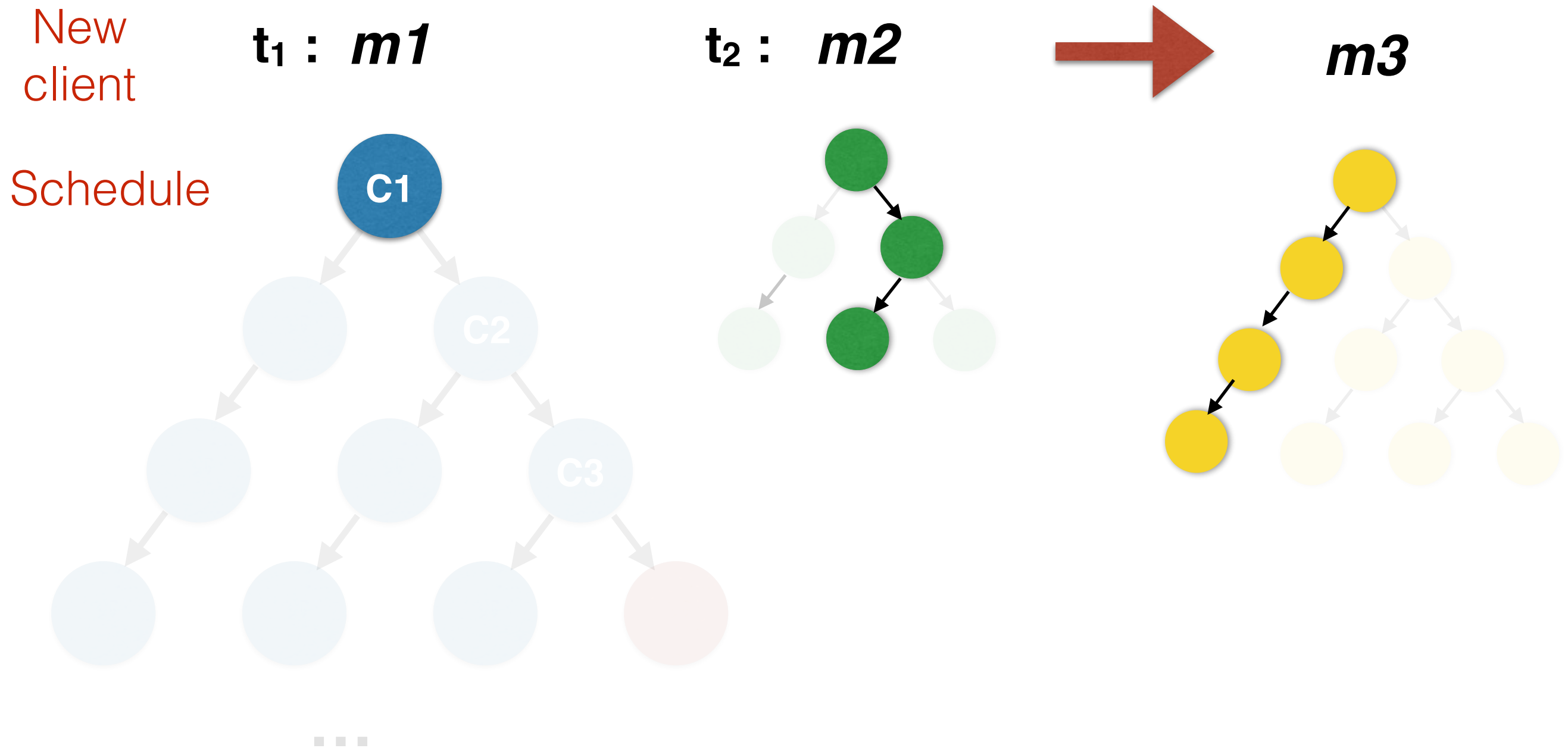
$t_1 : m1$

$t_2 : m2$

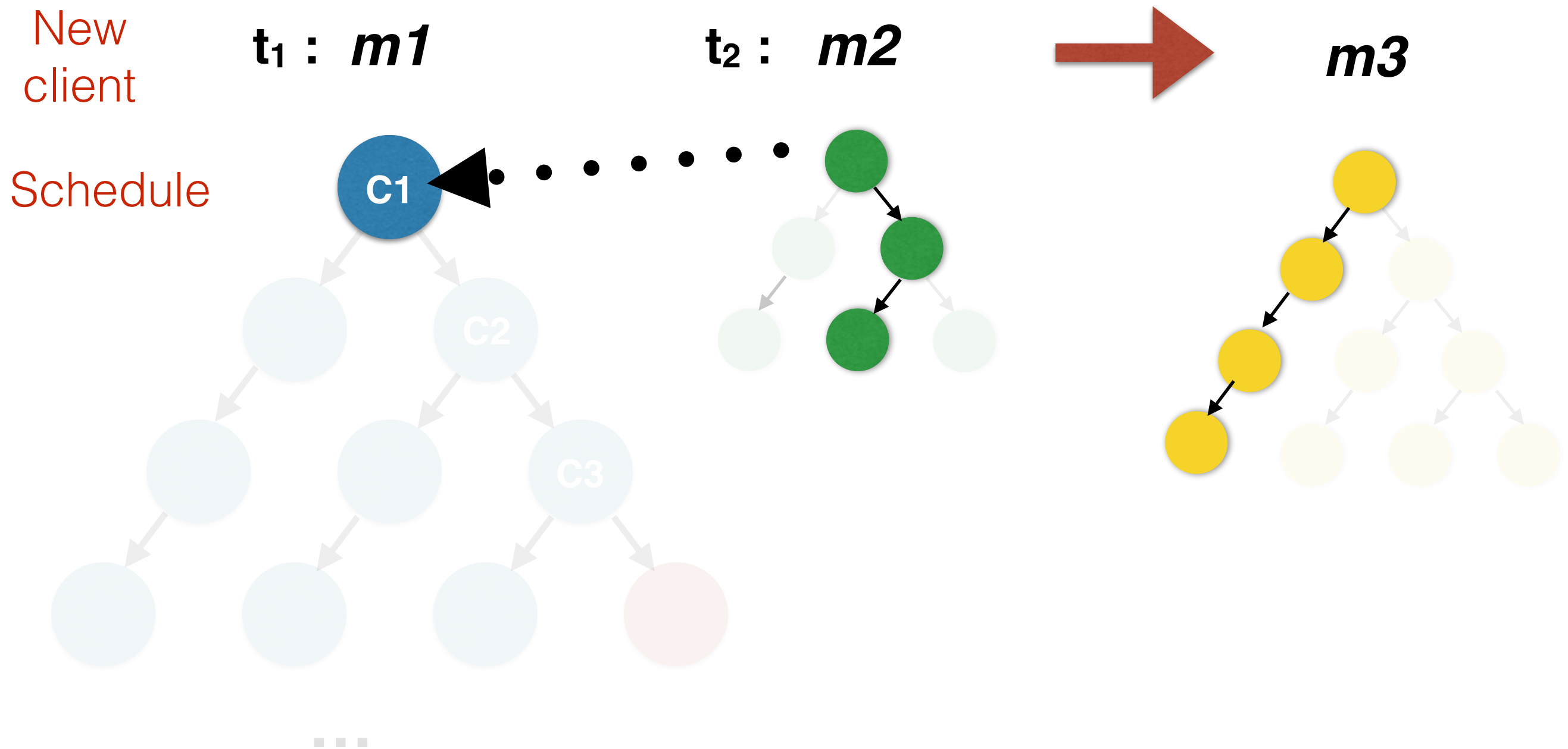


$m3$

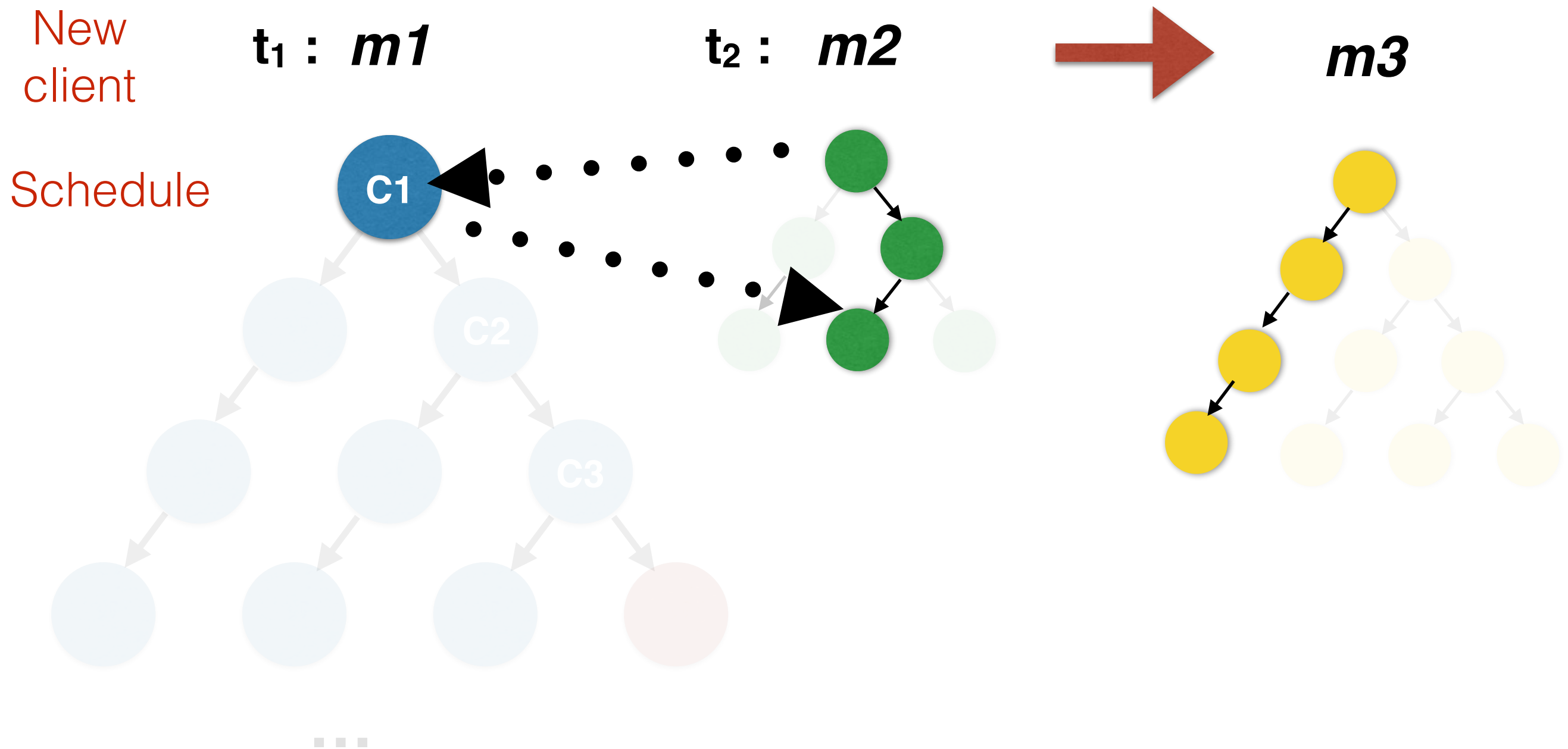
Iteration - 2 : Concrete Execution



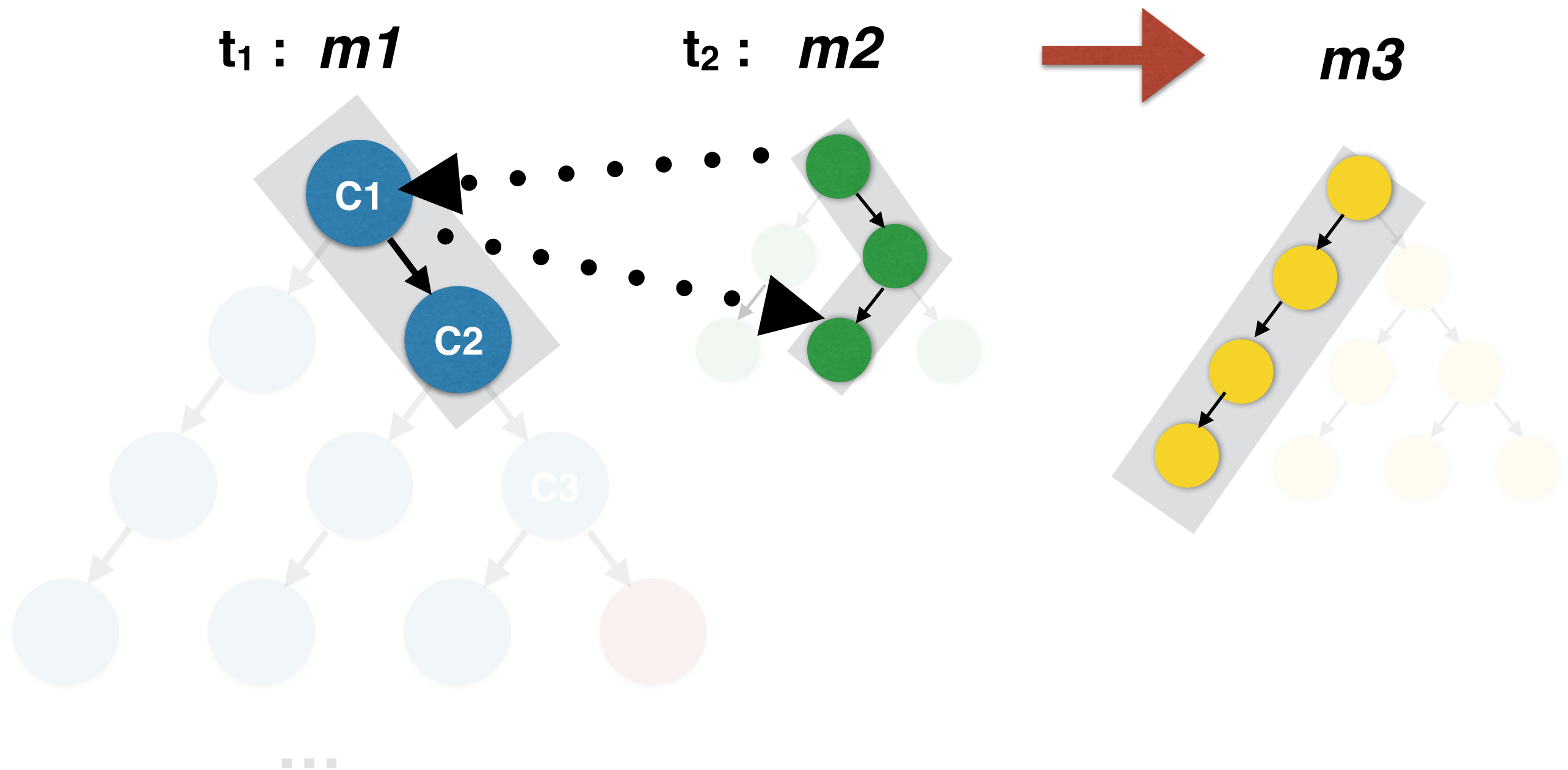
Iteration - 2 : Concrete Execution



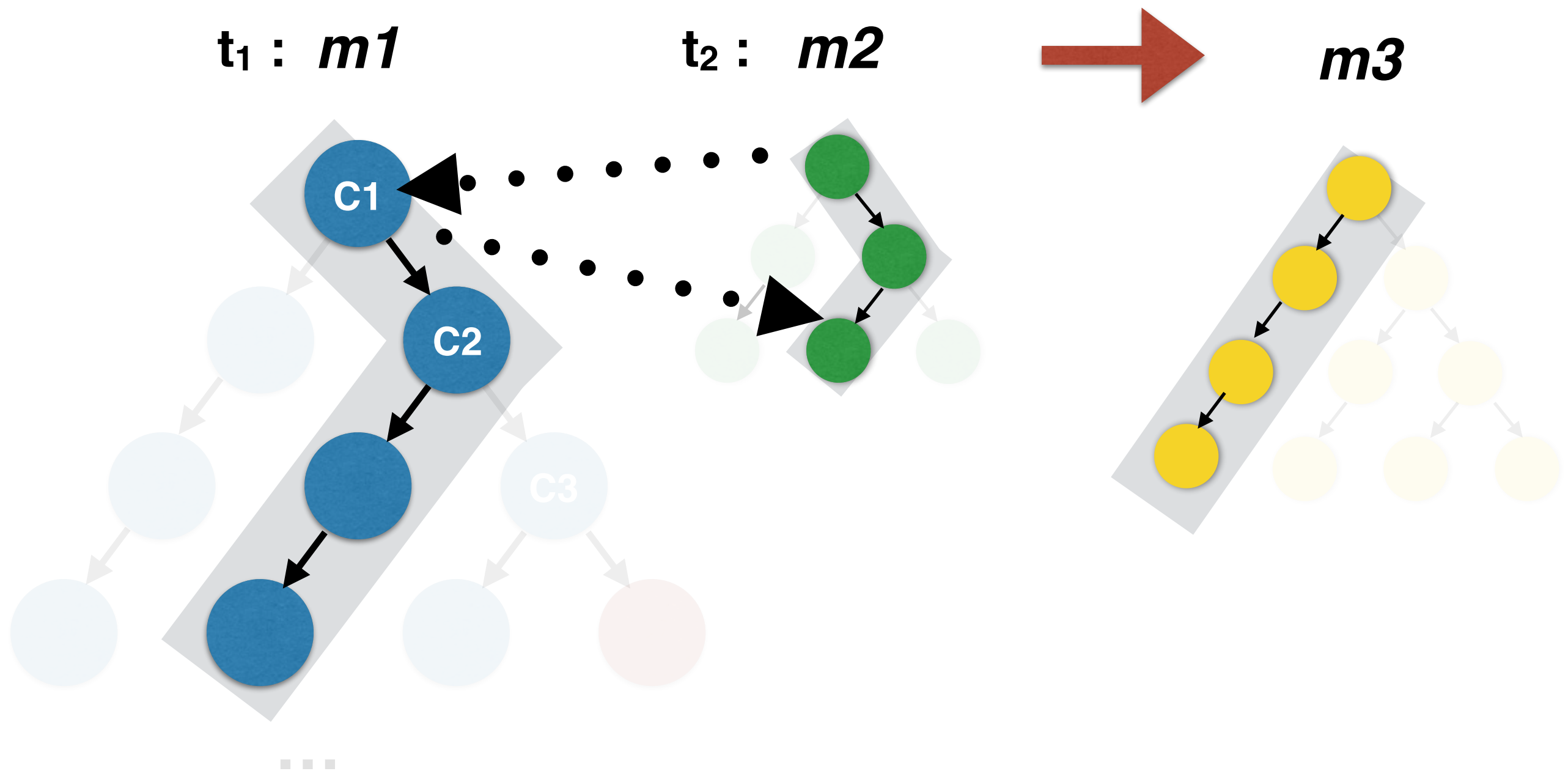
Iteration - 2 : Concrete Execution



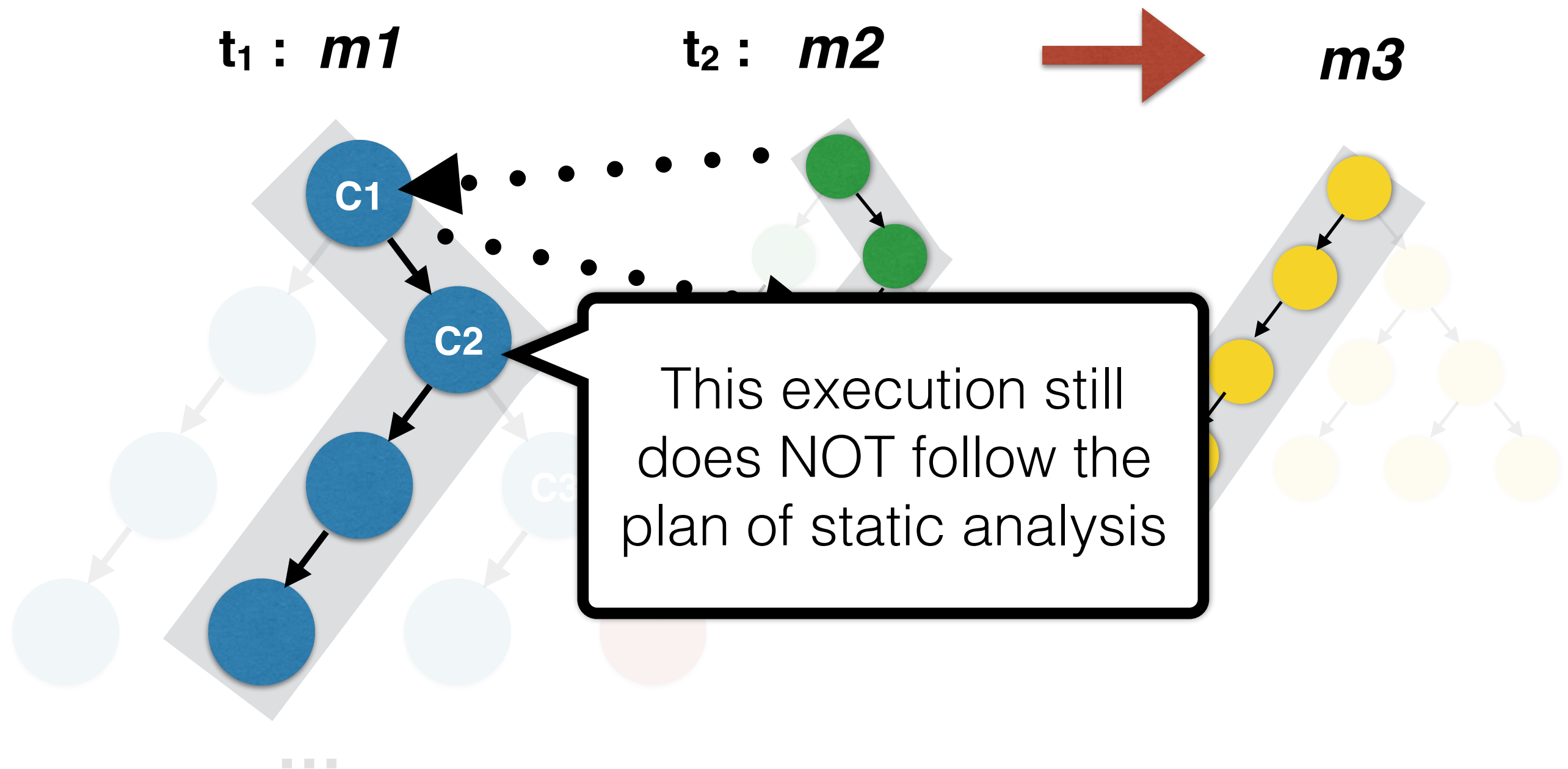
Iteration - 2 : Concrete Execution



Iteration - 2 : Concrete Execution



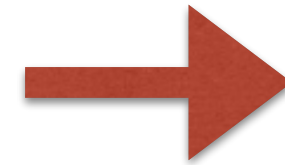
Iteration - 2 : Concrete Execution



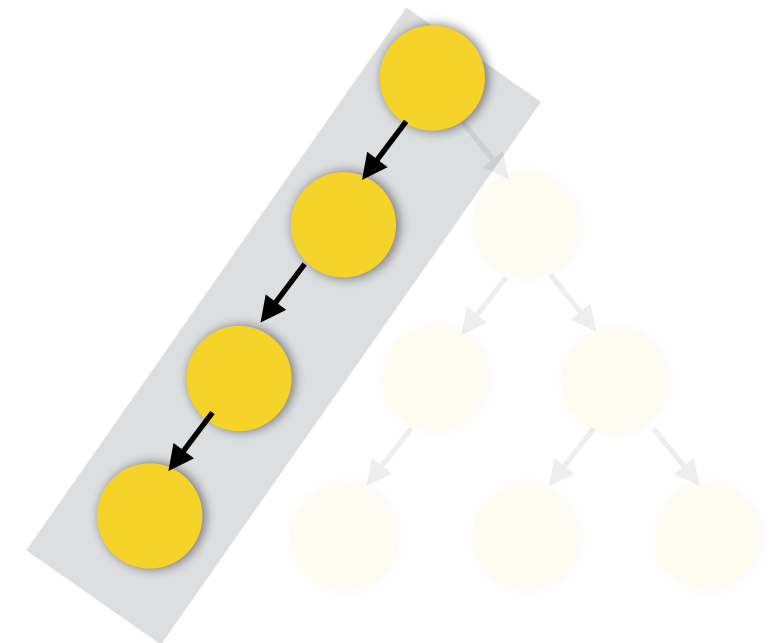
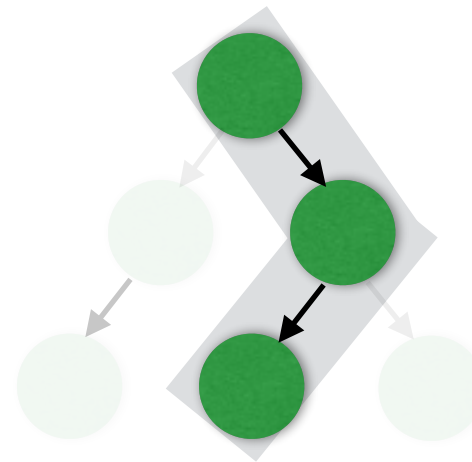
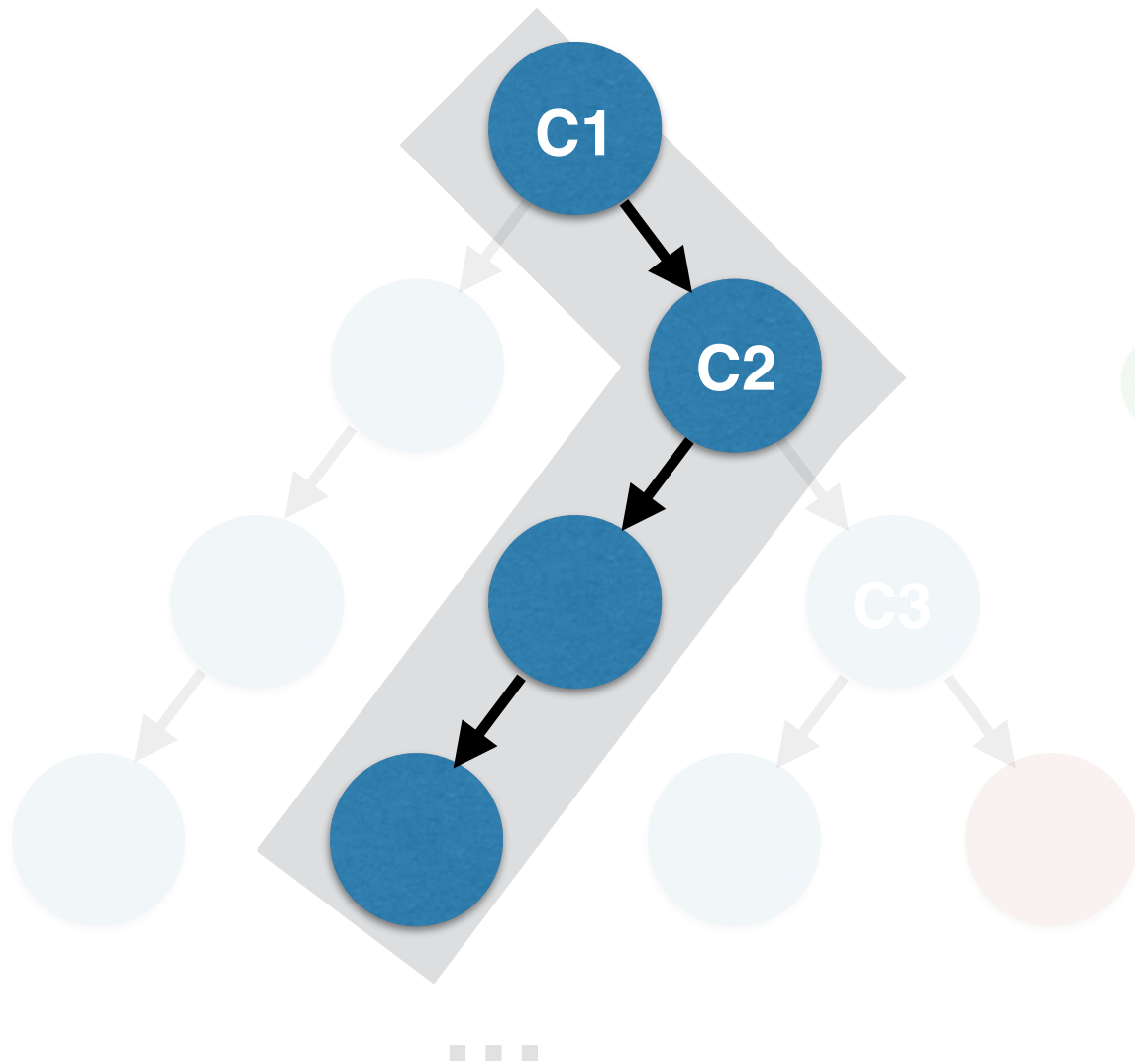
Iteration - 2 : Concrete Execution

$t_1 : m1$

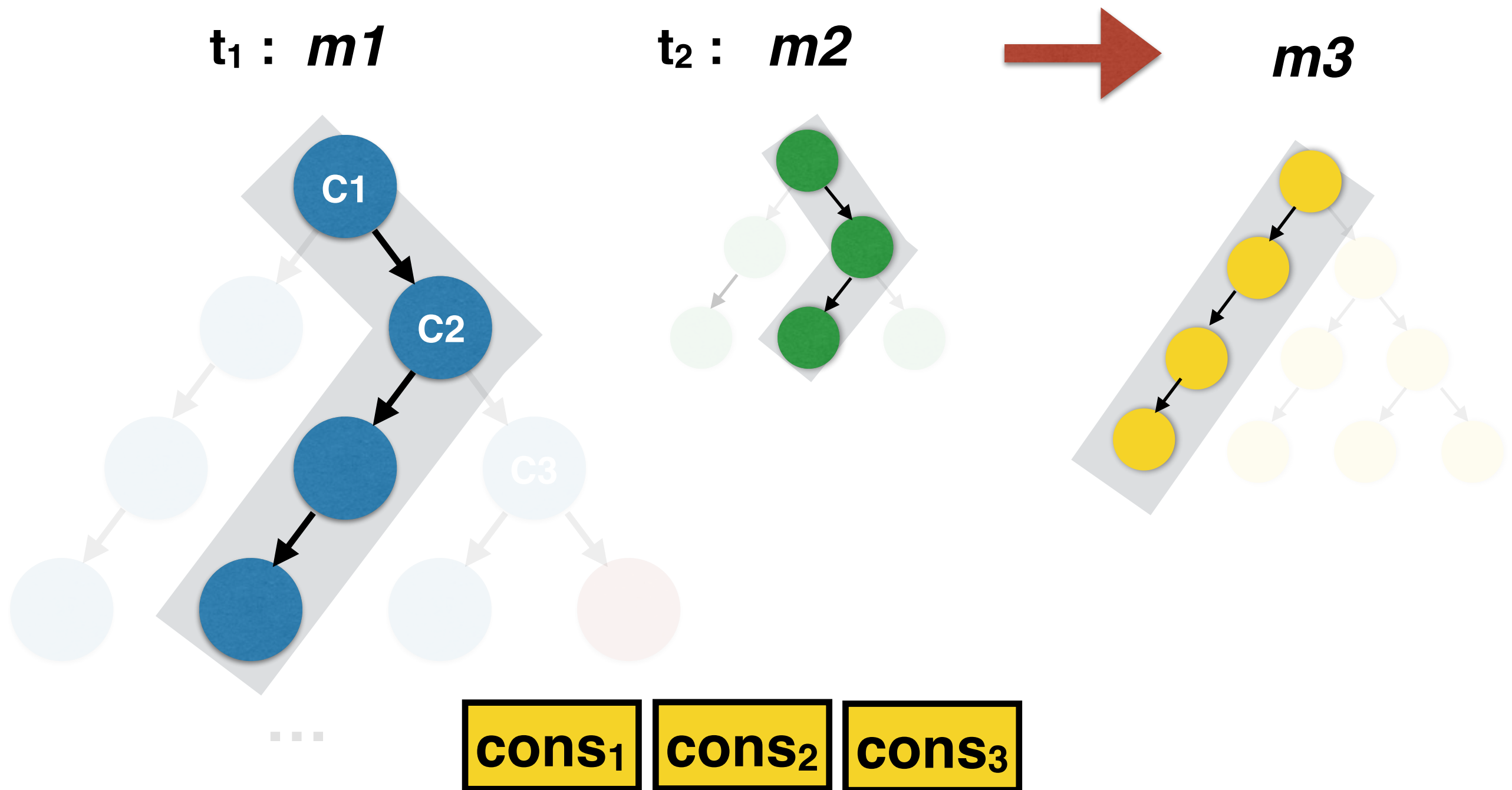
$t_2 : m2$



$m3$



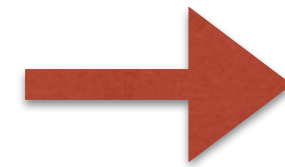
Iteration - 2 : Concrete Execution



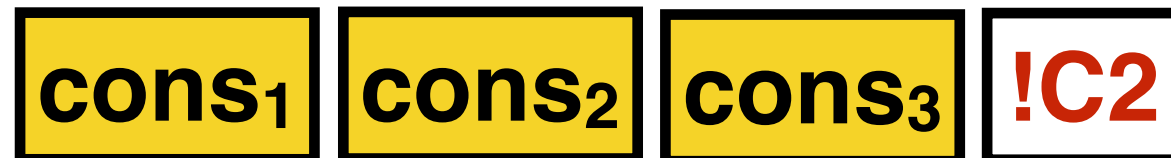
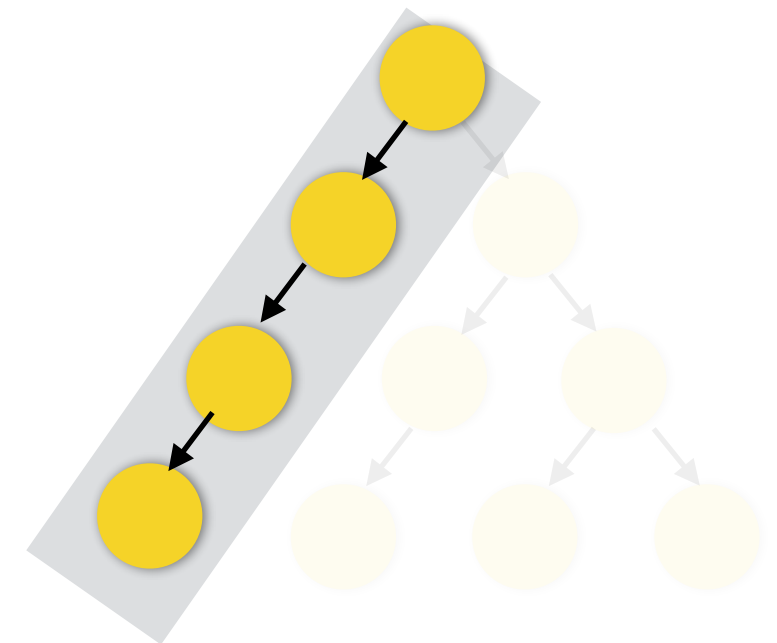
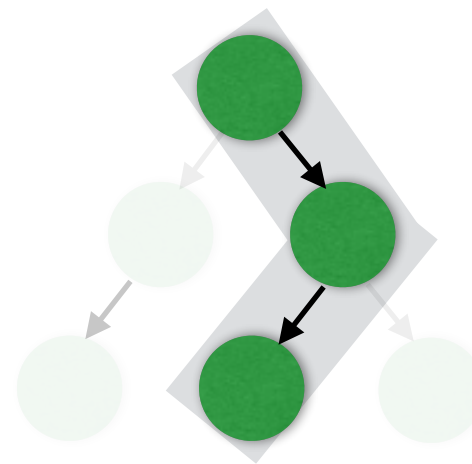
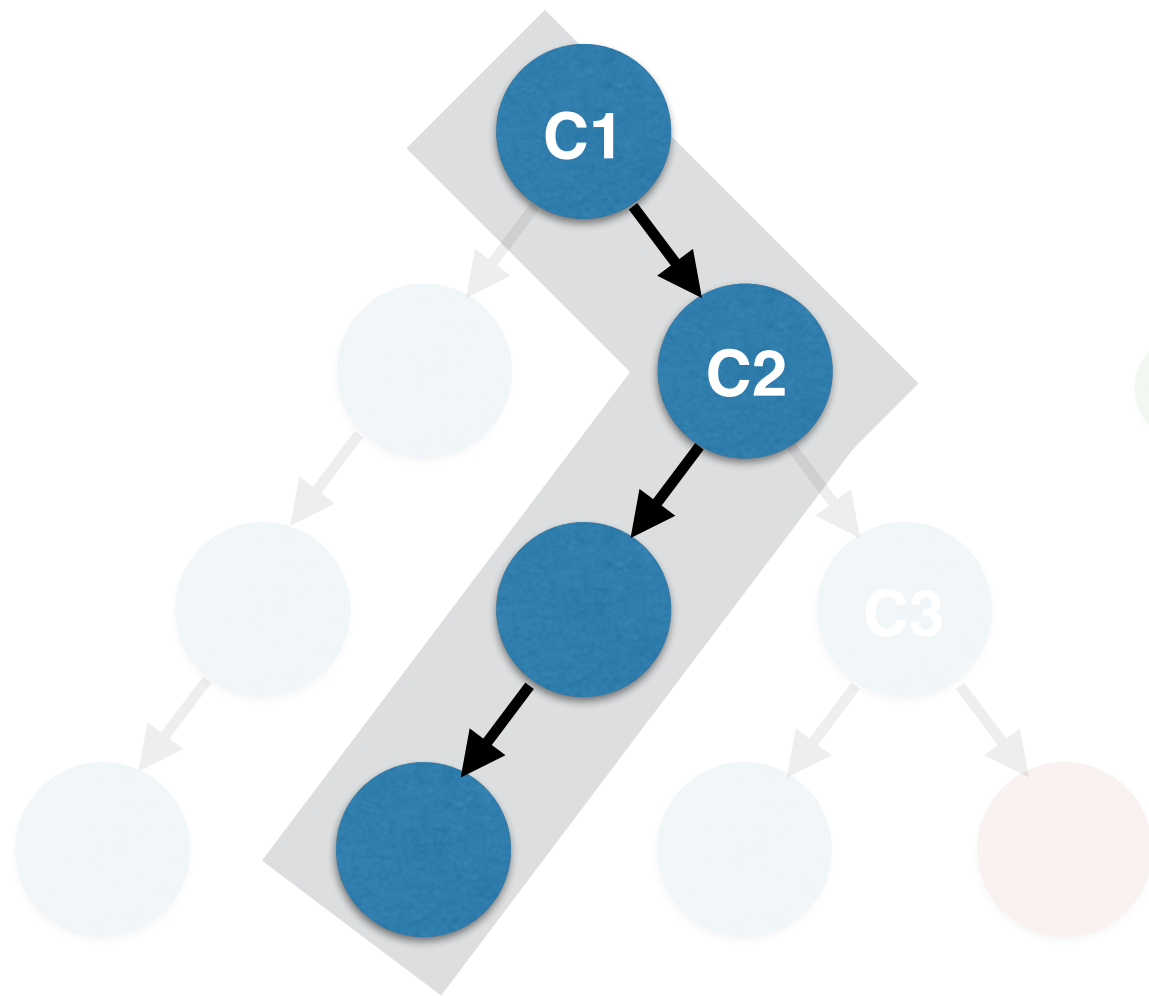
Iteration - 2 : Concrete Execution

$t_1 : m1$

$t_2 : m2$



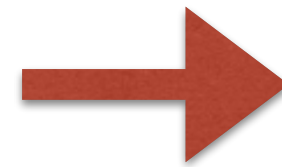
$m3$



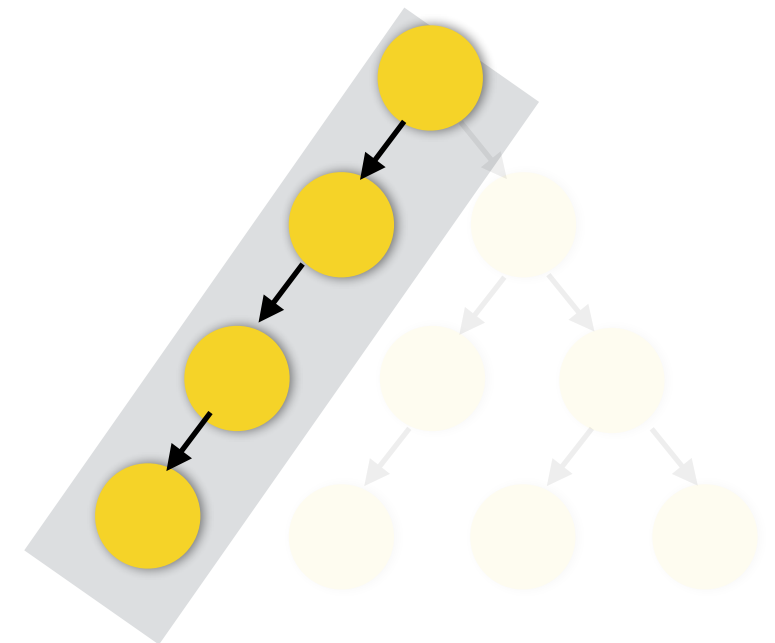
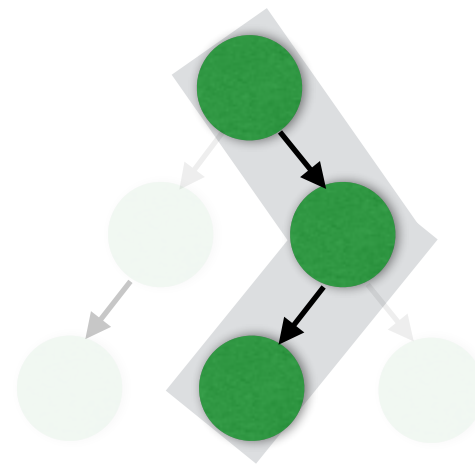
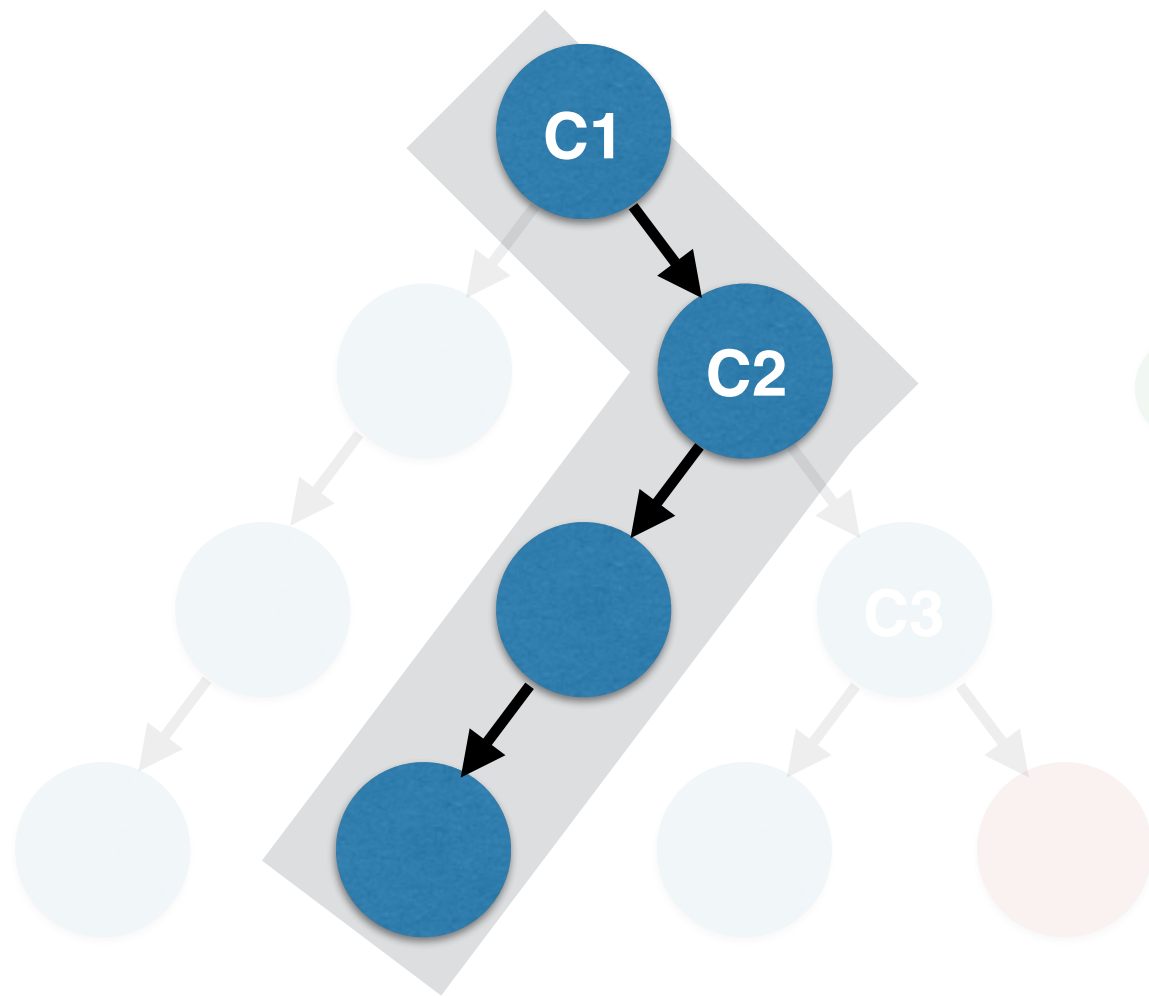
Iteration - 2 : Concrete Execution

$t_1 : m1$

$t_2 : m2$



$m3$



...

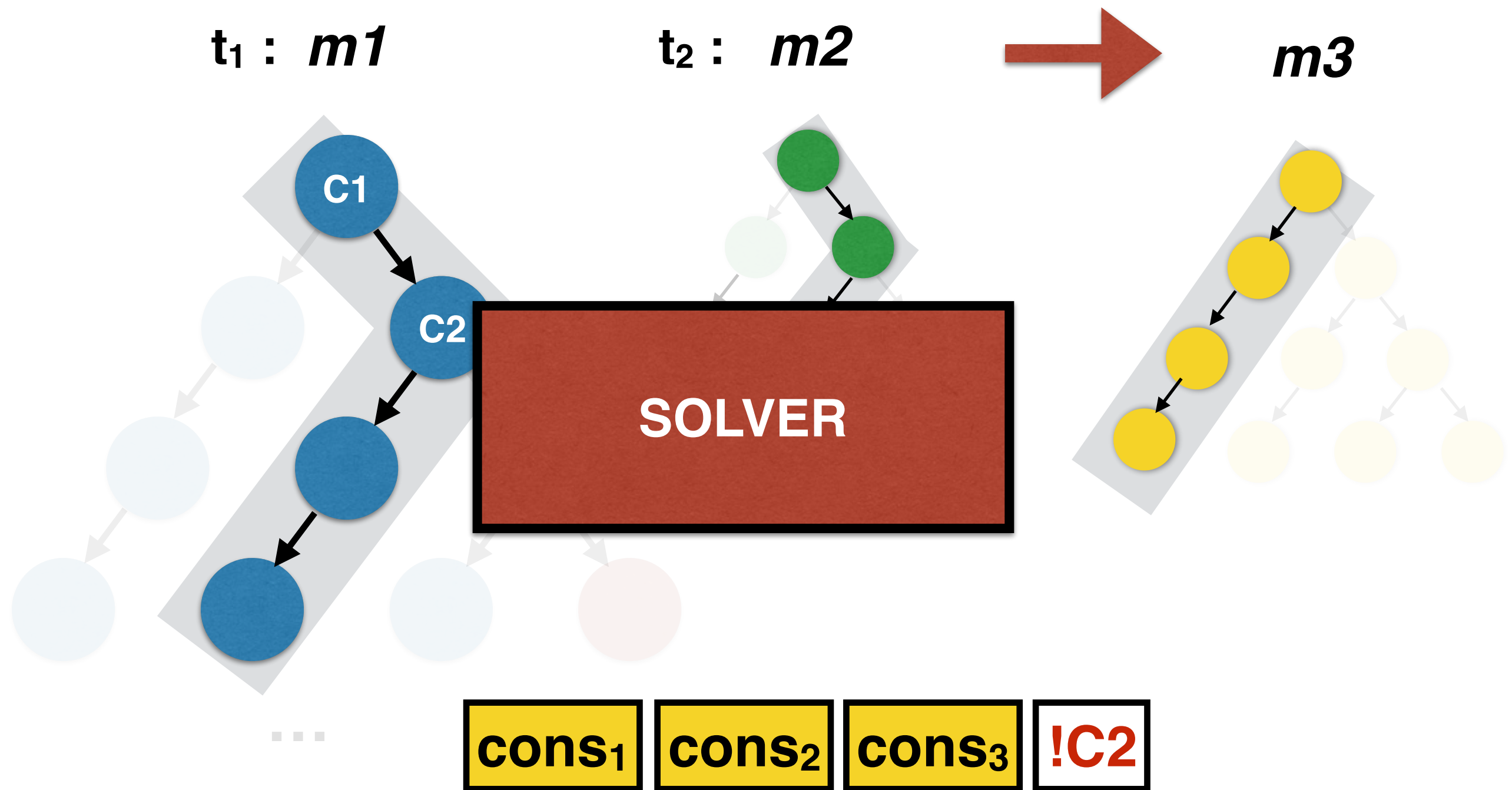
cons₁

cons₂

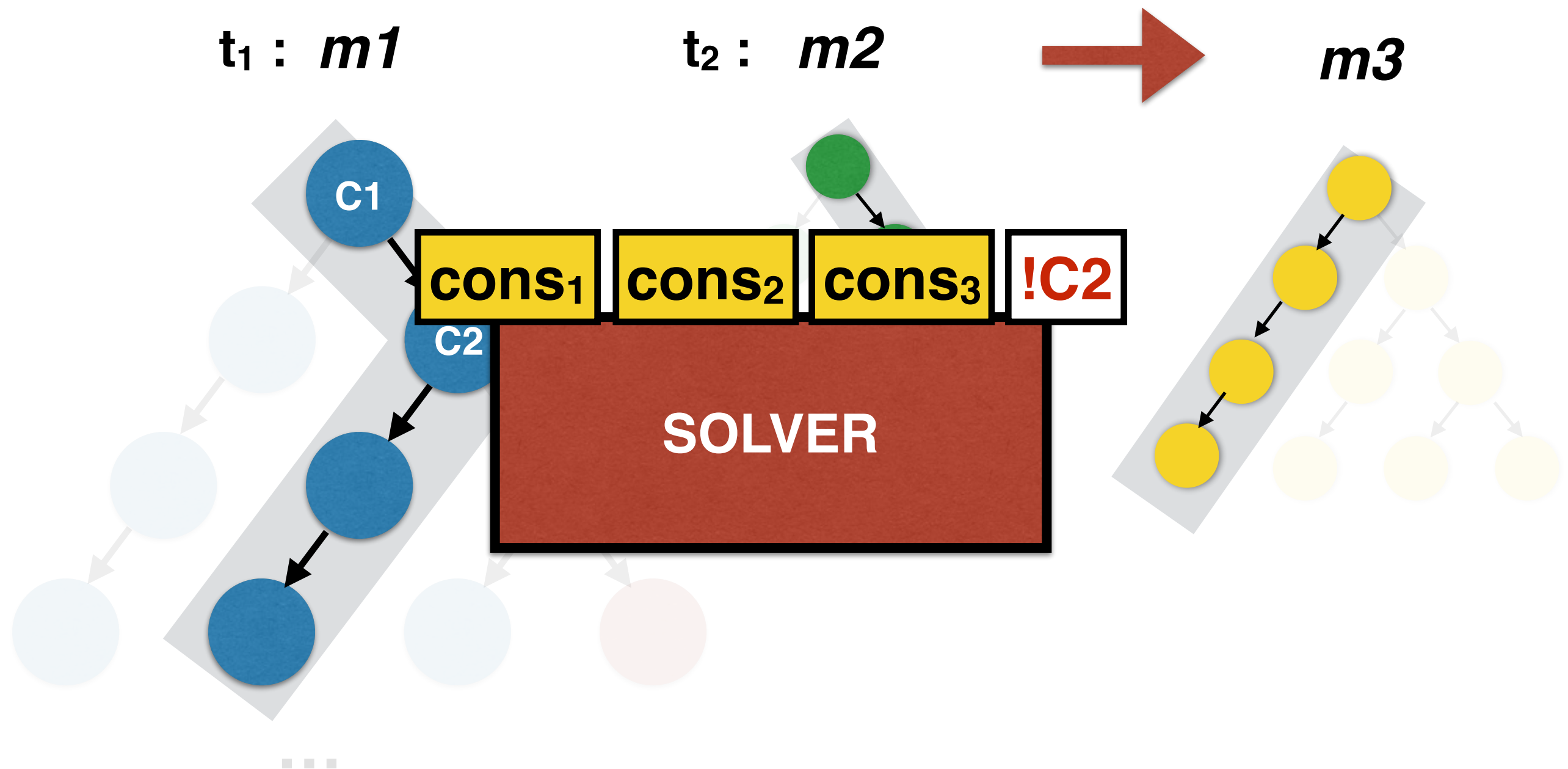
cons₃

!C2

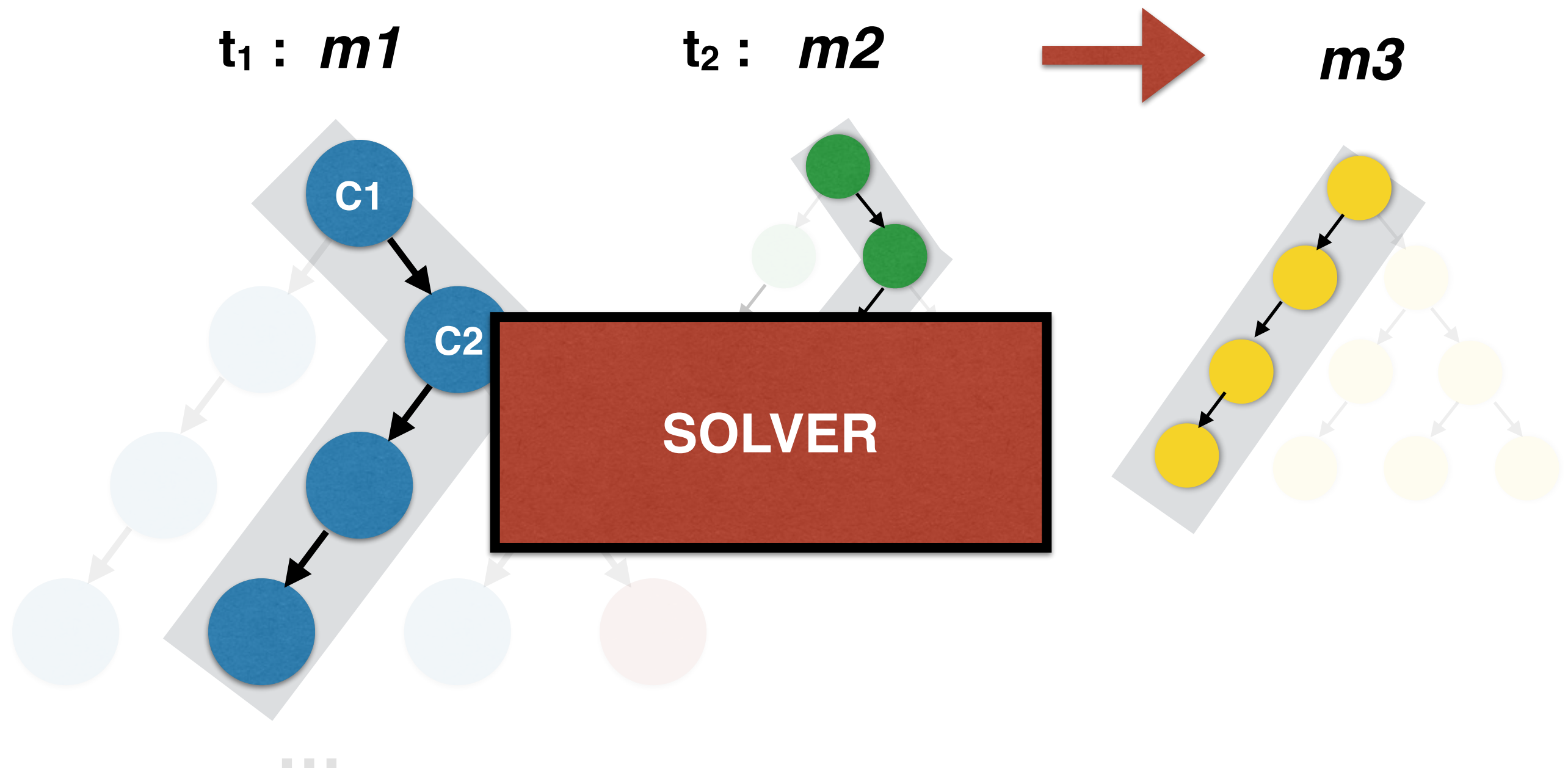
Iteration - 2 : Concrete Execution



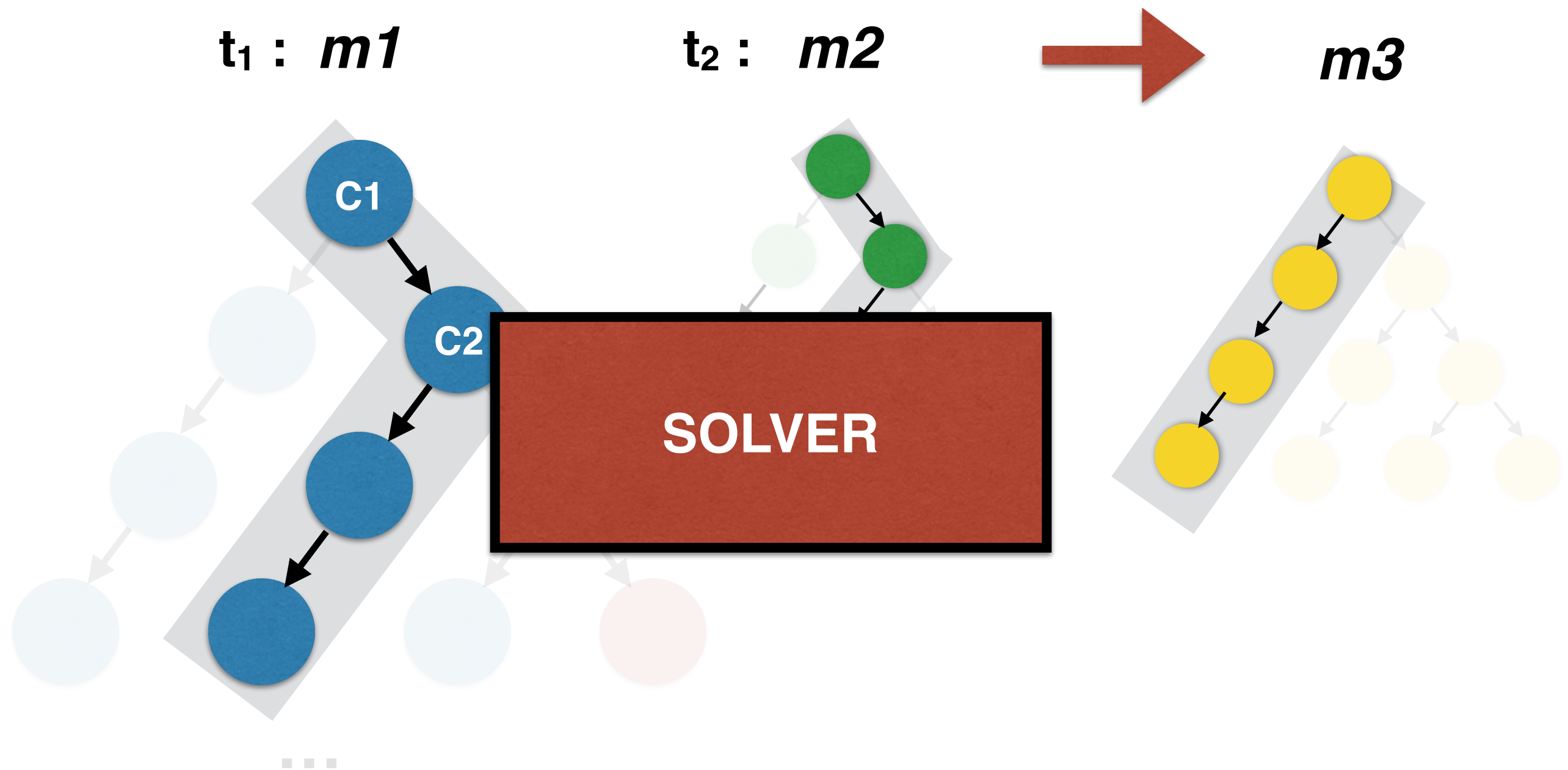
Iteration - 2 : Concrete Execution



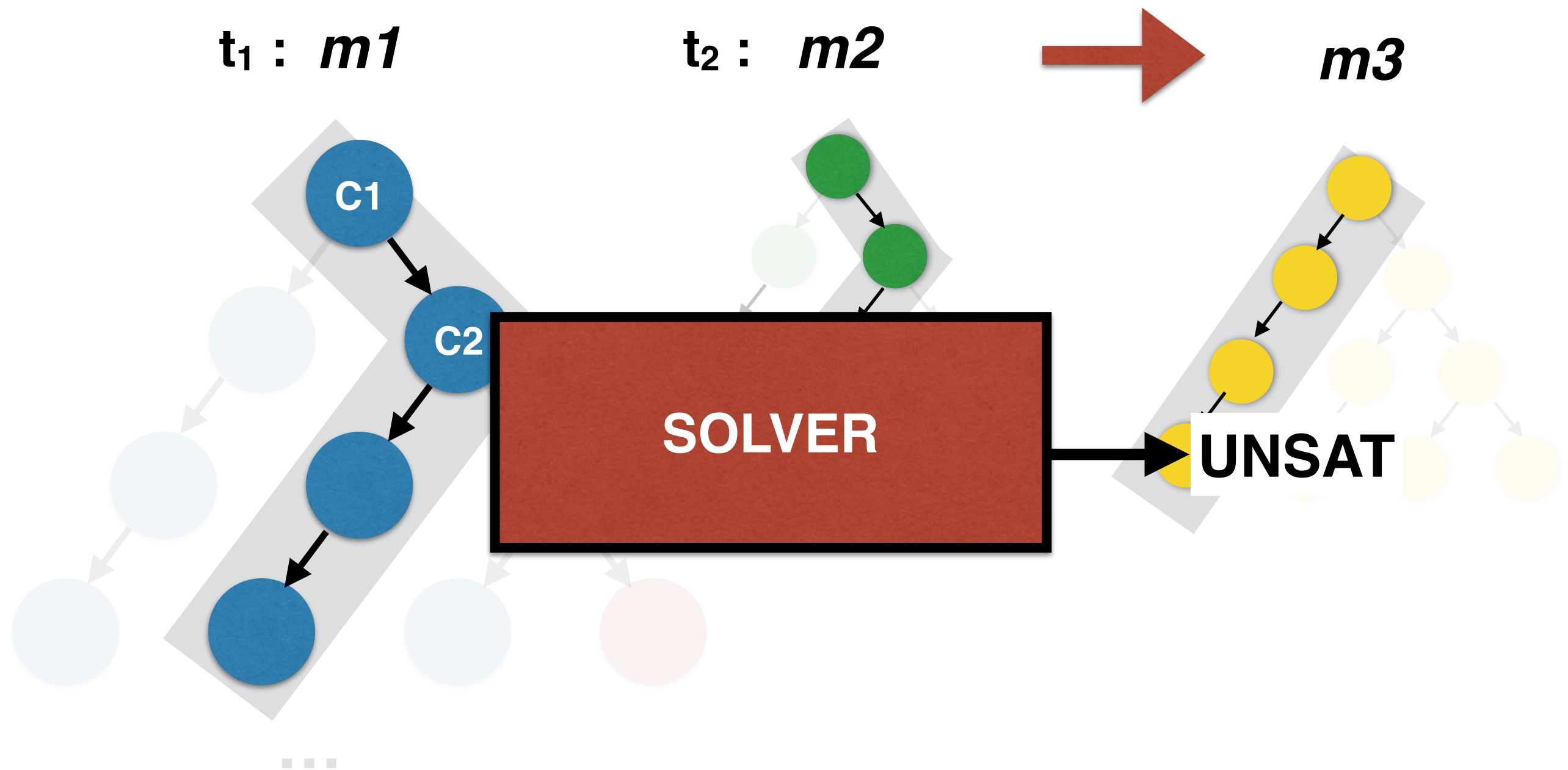
Iteration - 2 : Concrete Execution



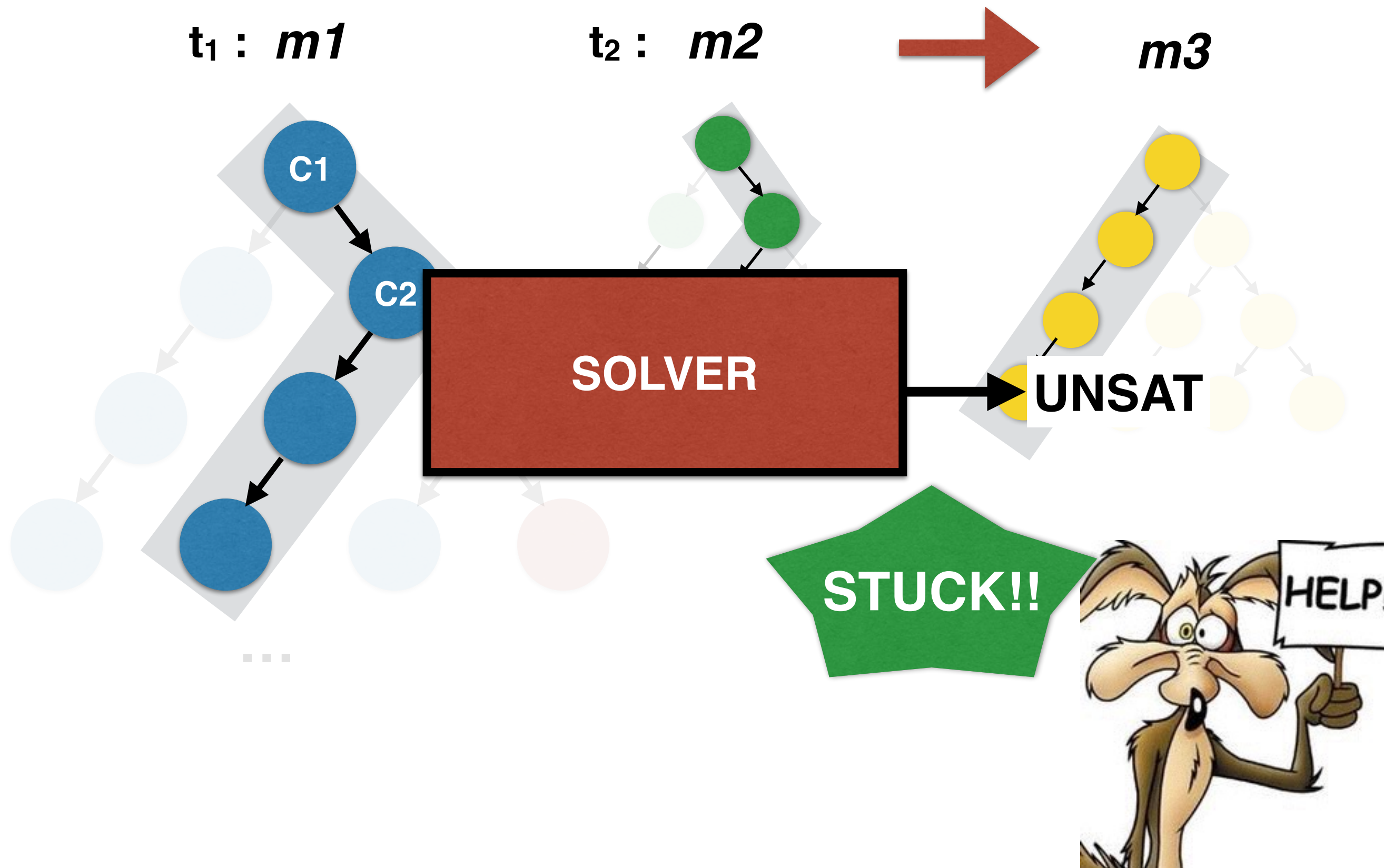
Iteration - 2 : Concrete Execution



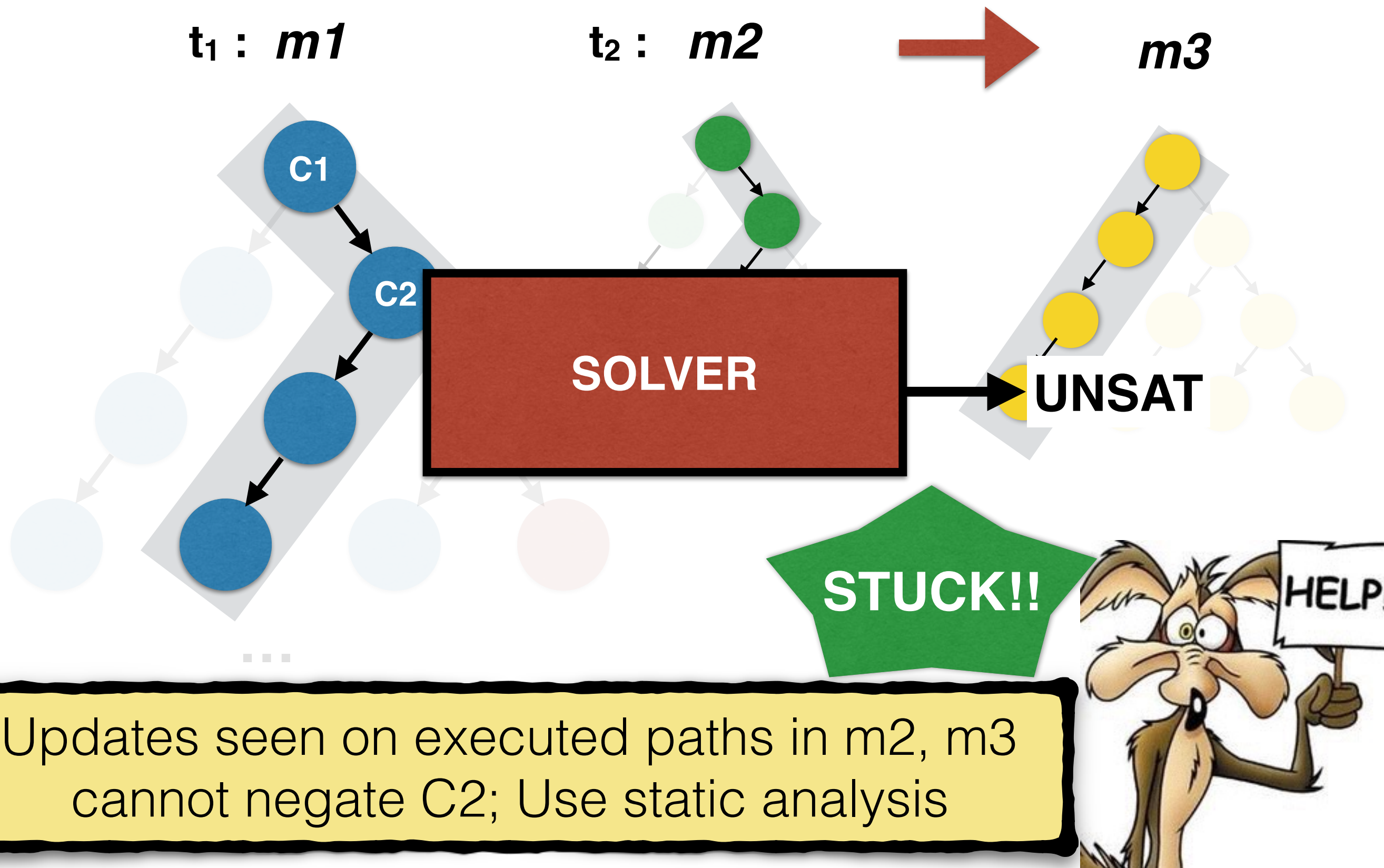
Iteration - 2 : Concrete Execution



Iteration - 2 : Concrete Execution



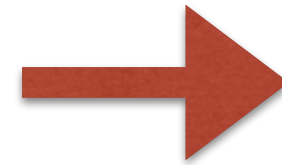
Iteration - 2 : Concrete Execution



Iteration - 3 : Static Analysis

$t_1 : m1$

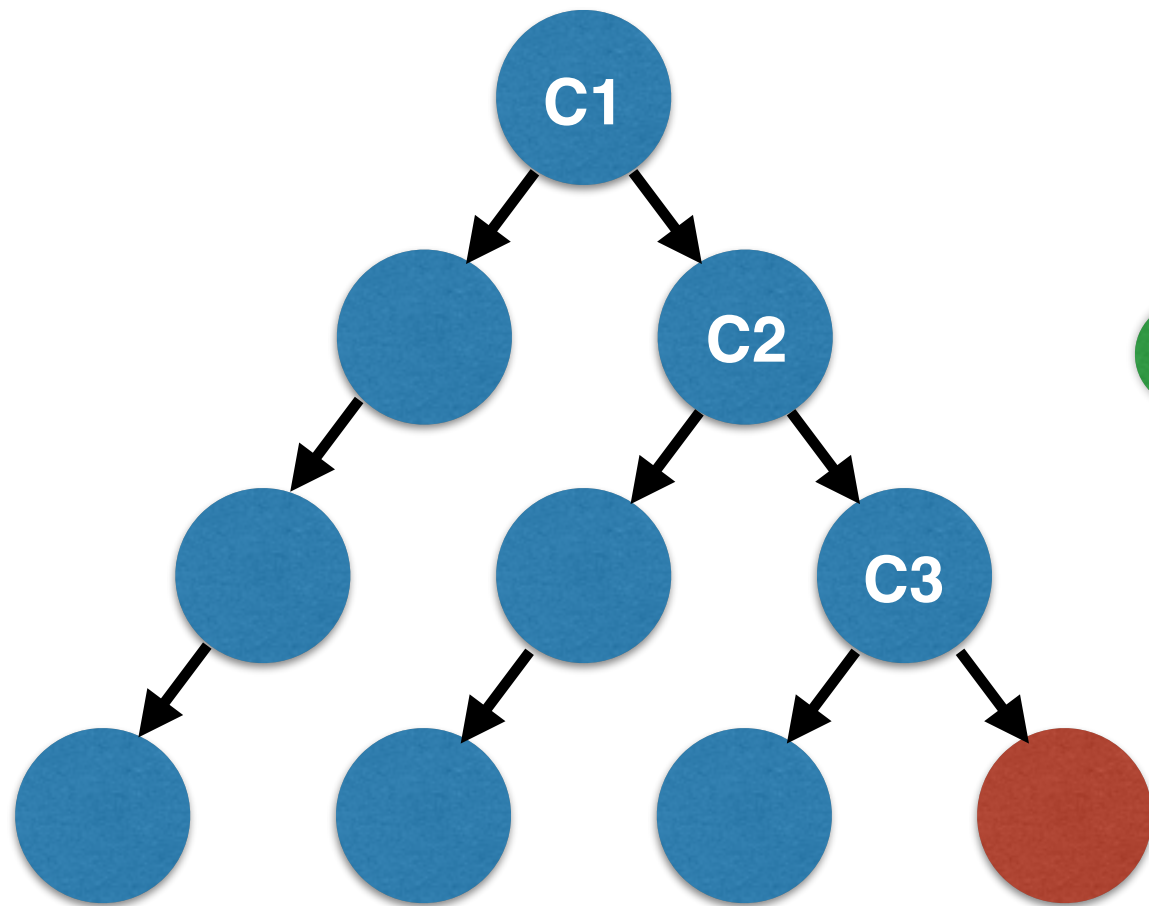
$t_2 : m2$



$m3$

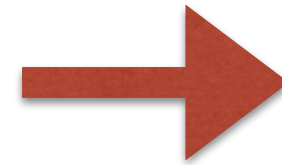
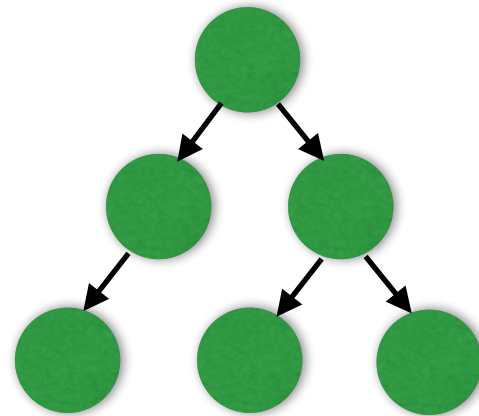
Iteration - 3 : Static Analysis

$t_1 : m1$

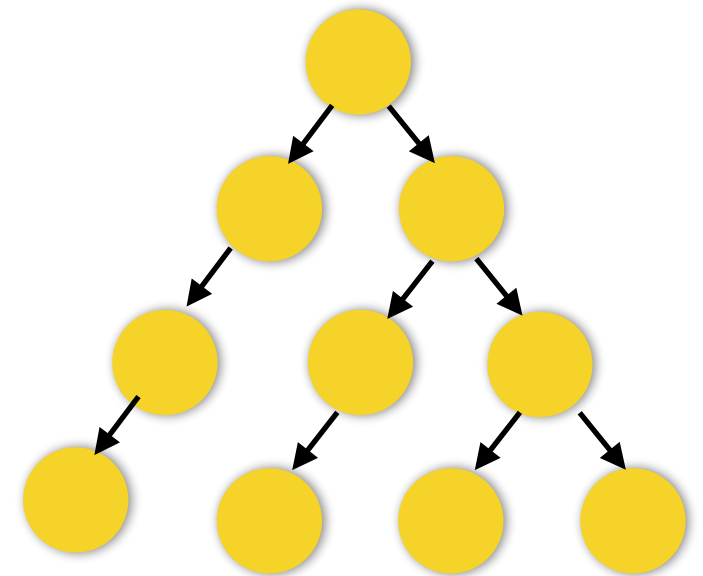


...

$t_2 : m2$



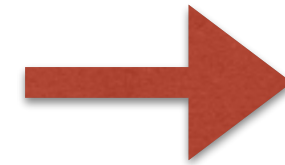
$m3$



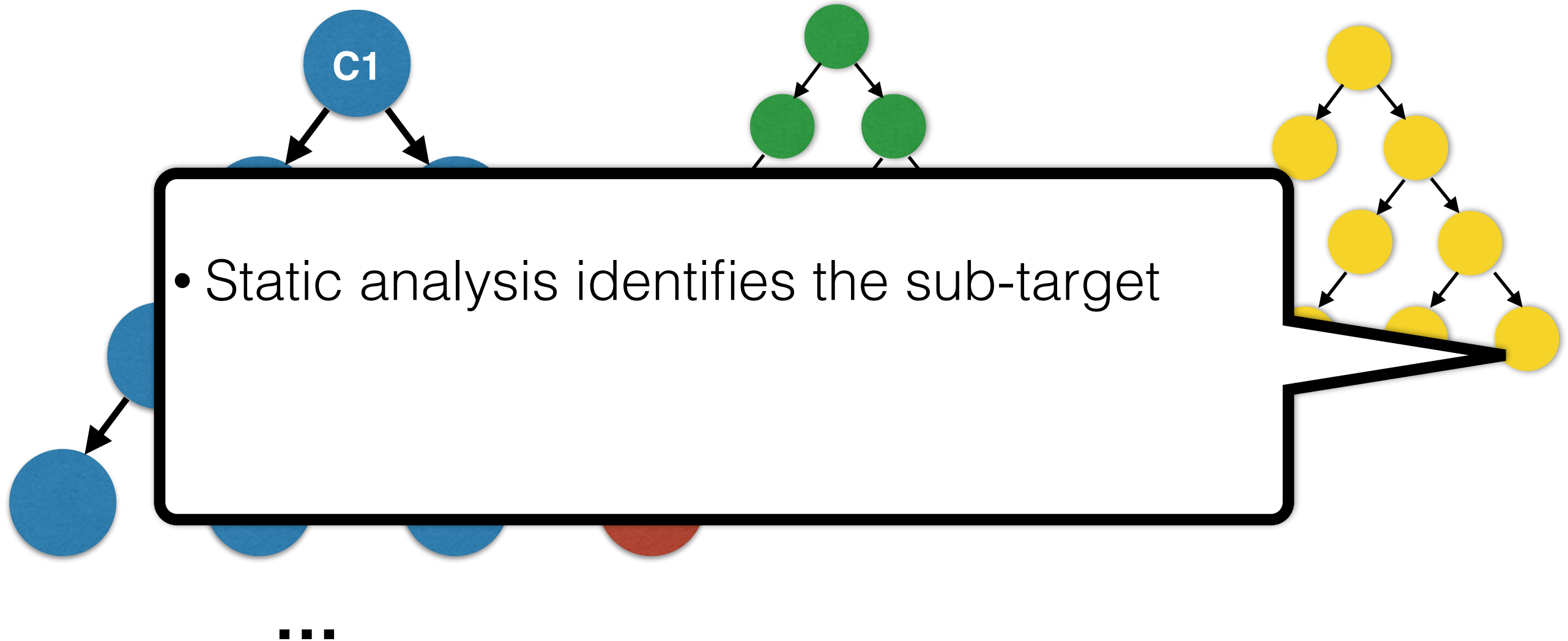
Iteration - 3 : Static Analysis

$t_1 : m1$

$t_2 : m2$



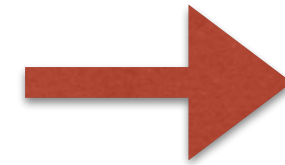
$m3$



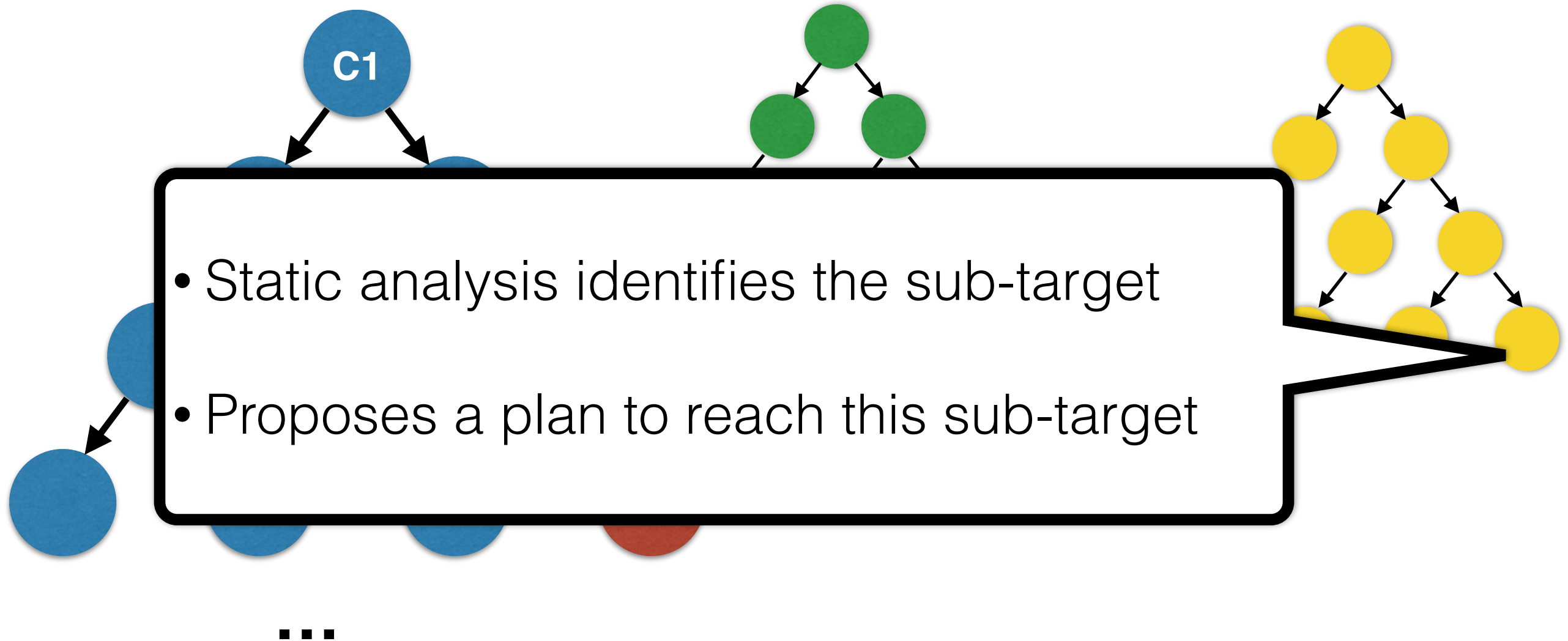
Iteration - 3 : Static Analysis

$t_1 : m1$

$t_2 : m2$



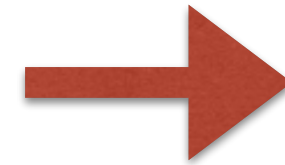
$m3$



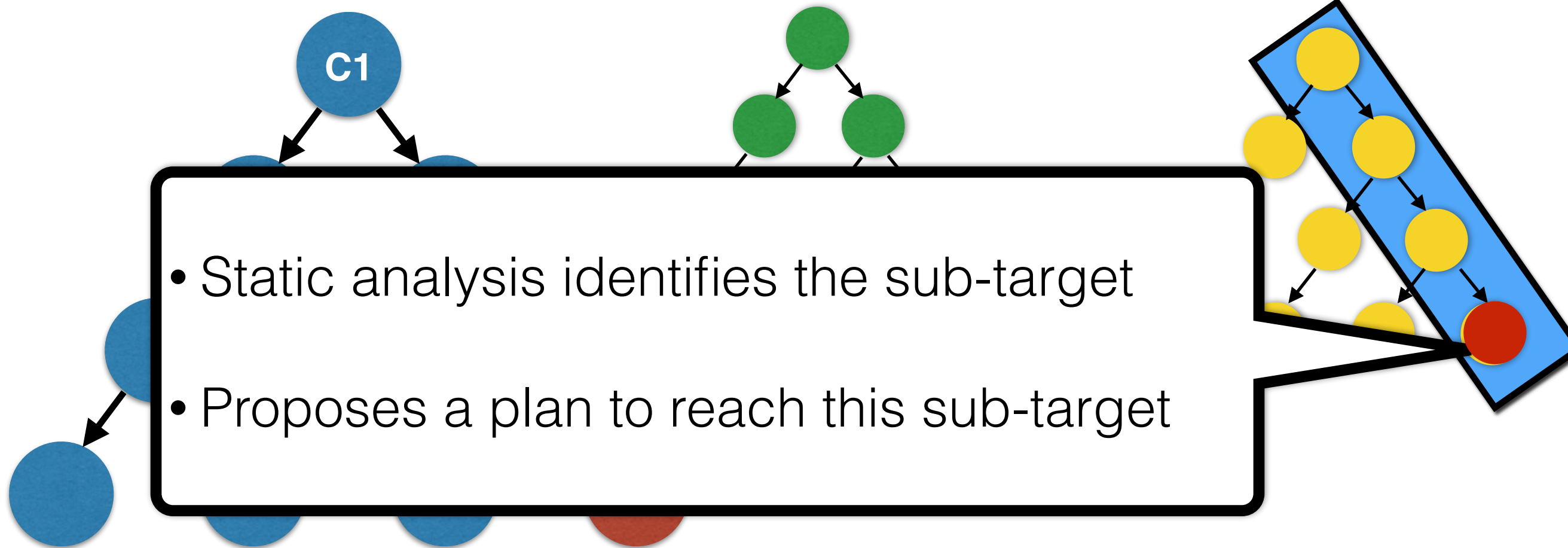
Iteration - 3 : Static Analysis

$t_1 : m1$

$t_2 : m2$



$m3$

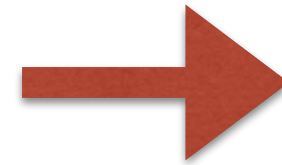


...

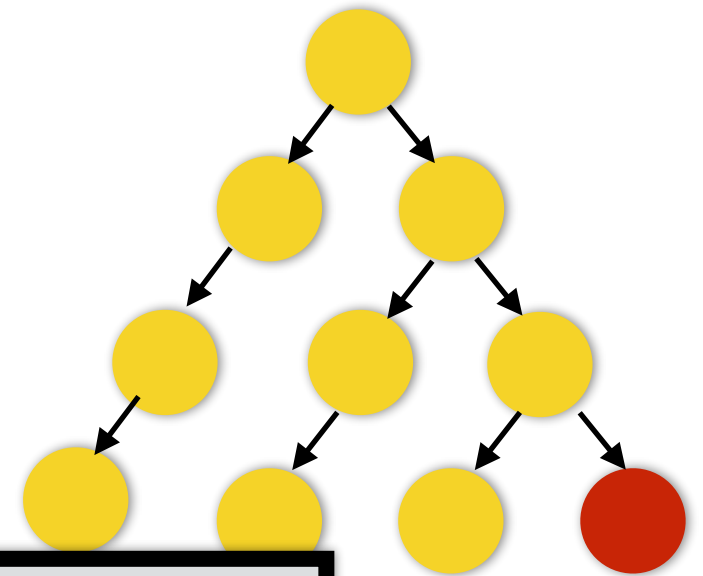
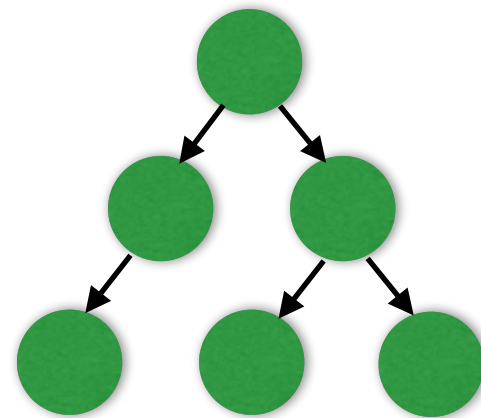
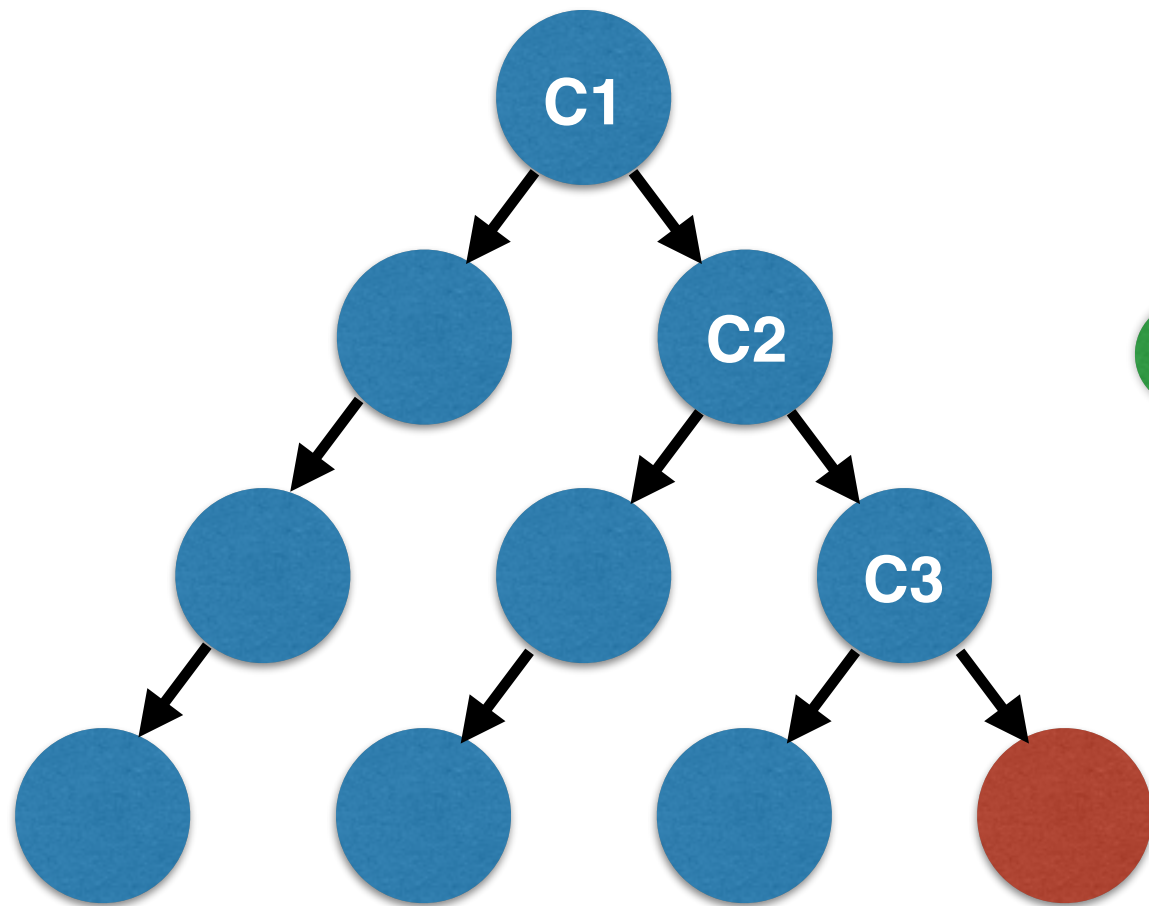
Iteration - 3 : Concrete Execution

$t_1 : m1$

$t_2 : m2$



$m3$



Apply MINION for
reaching this target

After a few more iterations...

Iteration - 4 : Concrete Execution

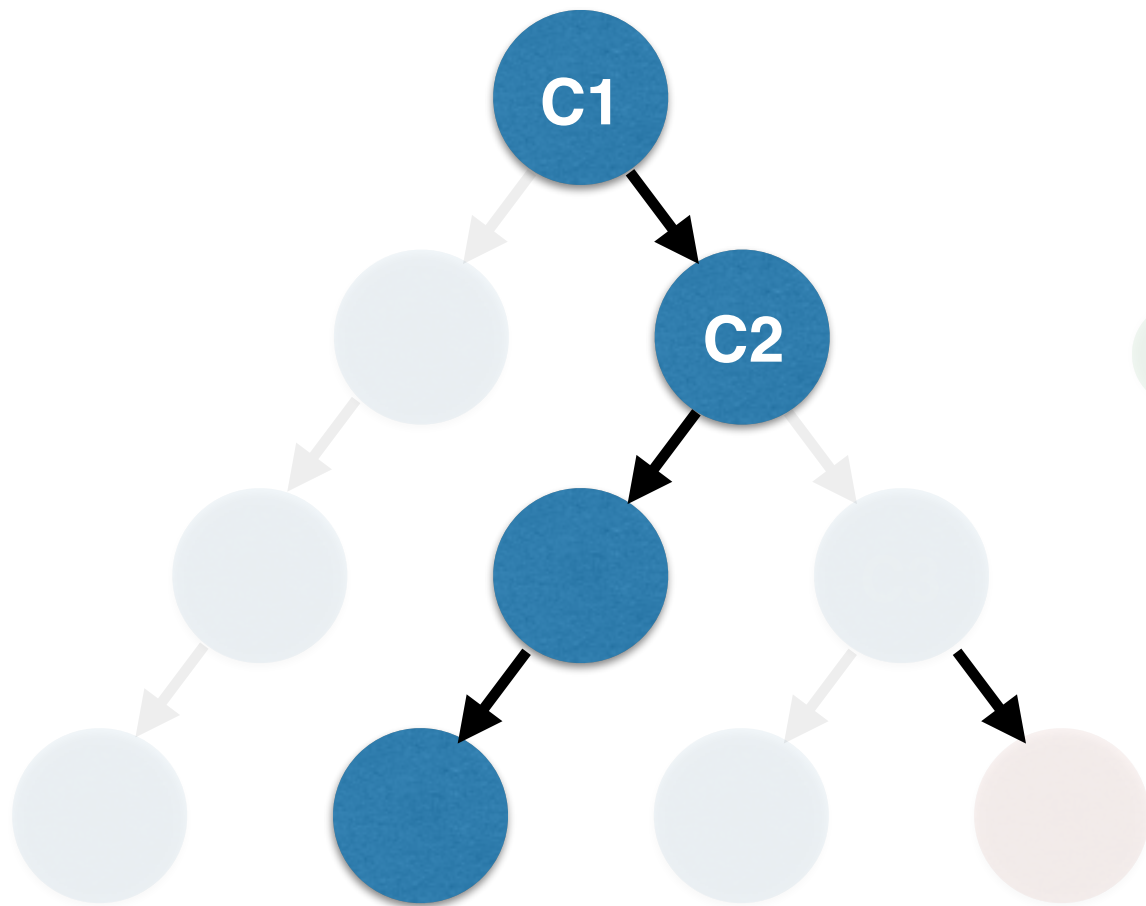
t_1 : *m1*

t_2 : *m2*

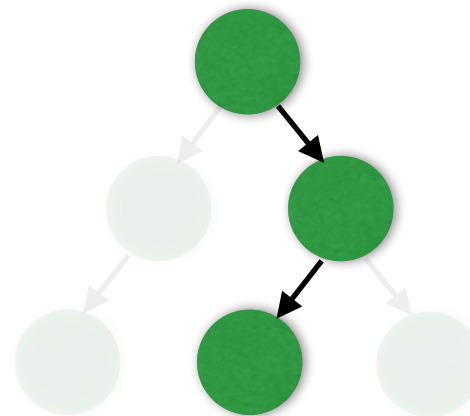
t_3 : *m3*

Iteration - 4 : Concrete Execution

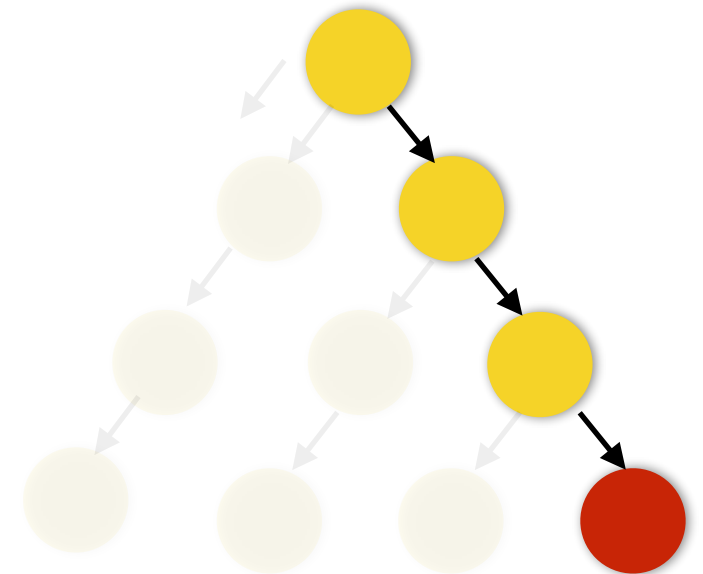
t_1 : $m1$



t_2 : $m2$

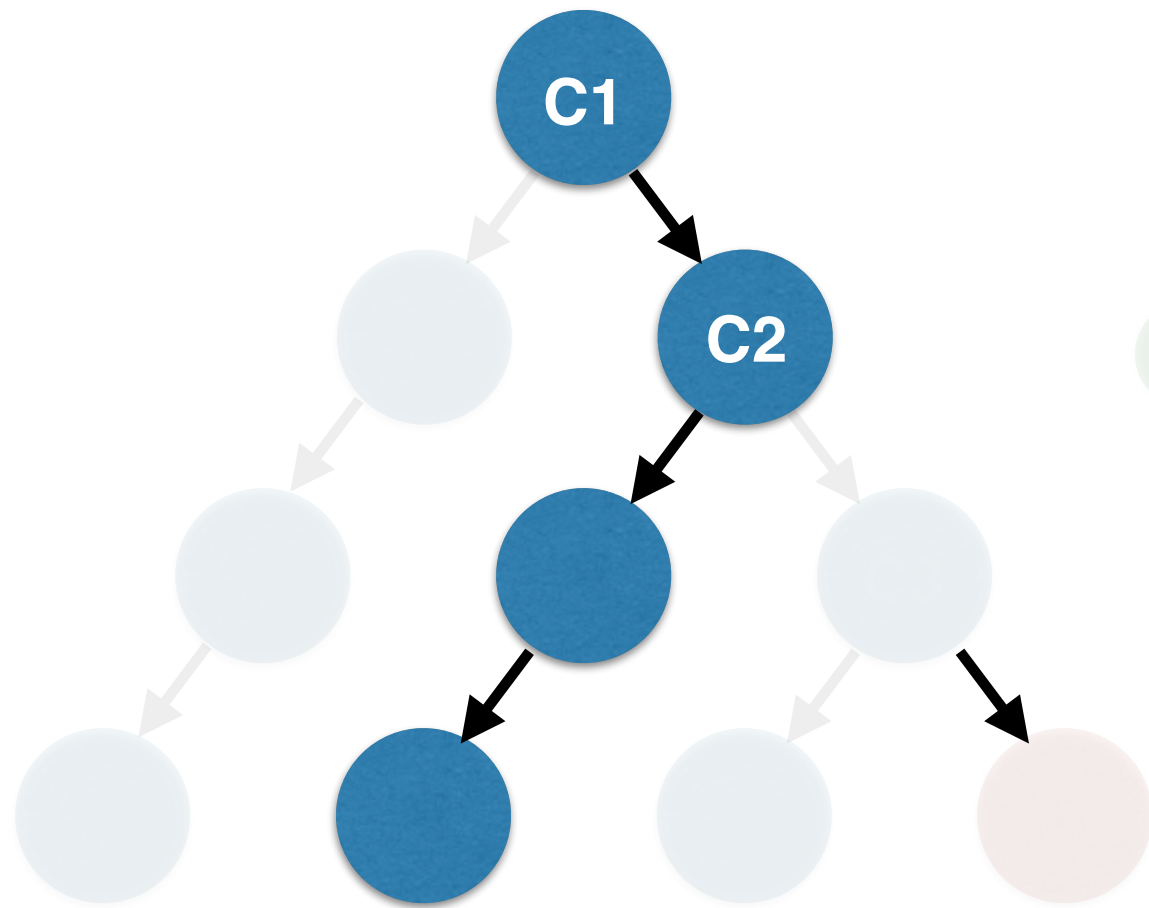


t_3 : $m3$

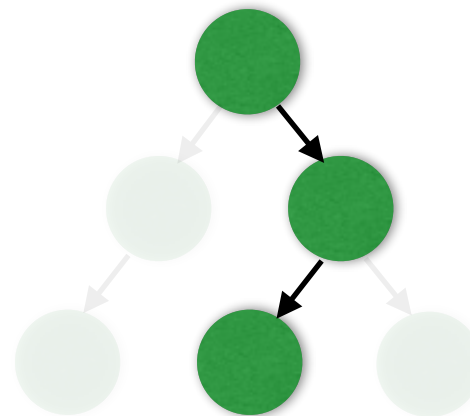


Iteration - 4 : Concrete Execution

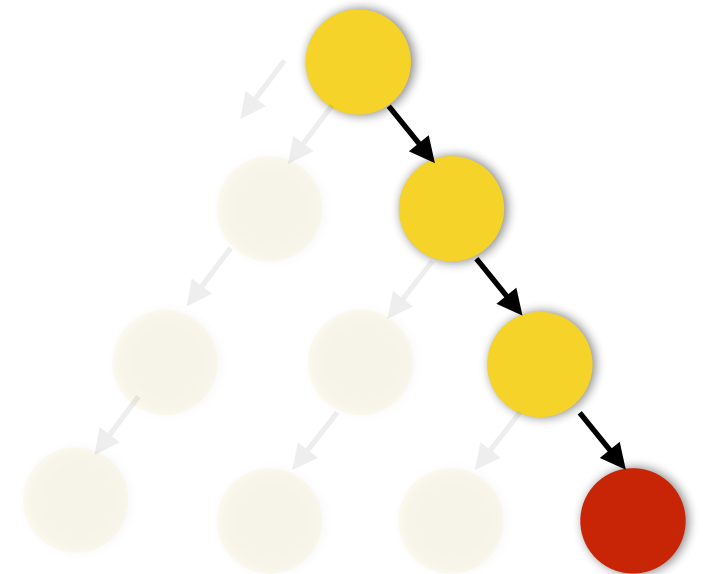
t₁ : *m1*



t₂ : *m2*



t₃ : *m3*



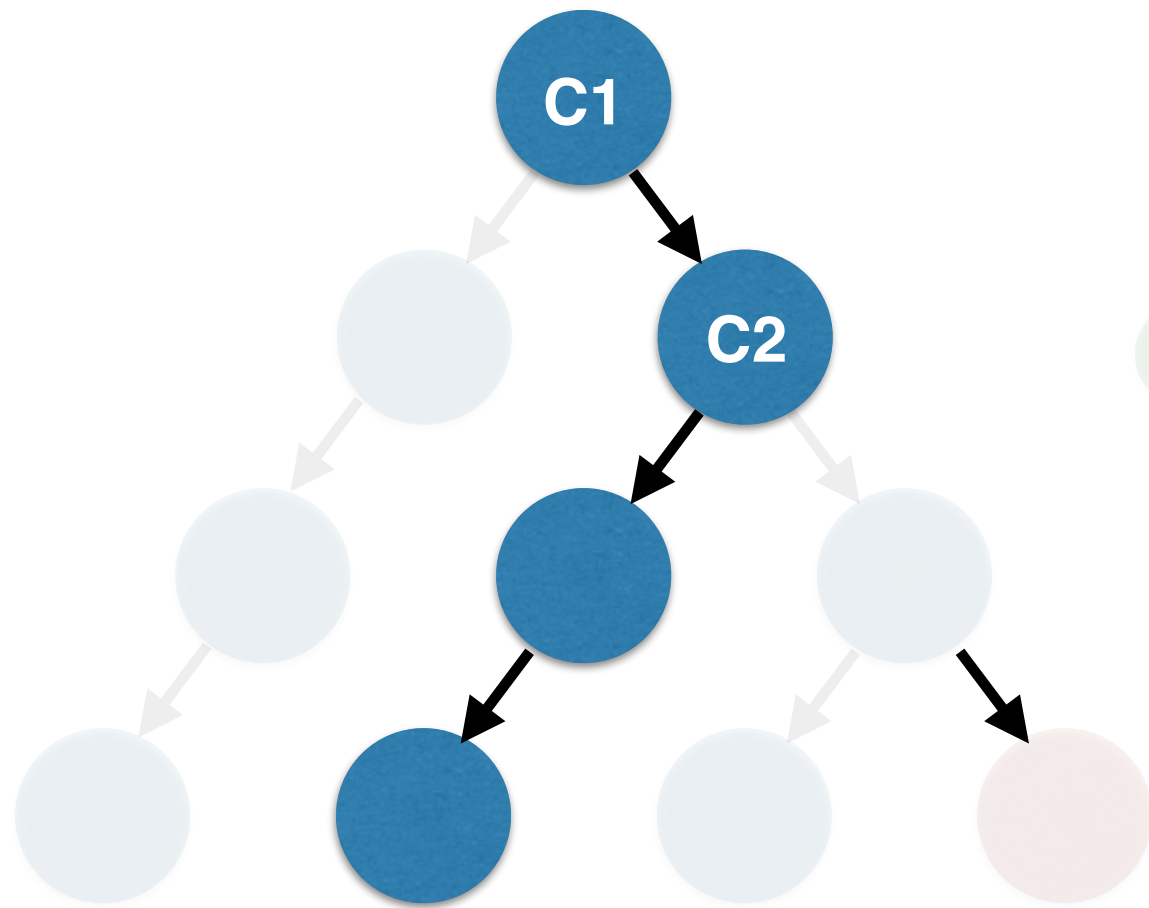
cons₁

cons₂

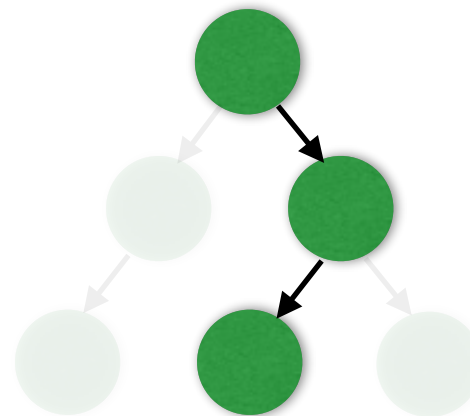
cons₃

Iteration - 4 : Concrete Execution

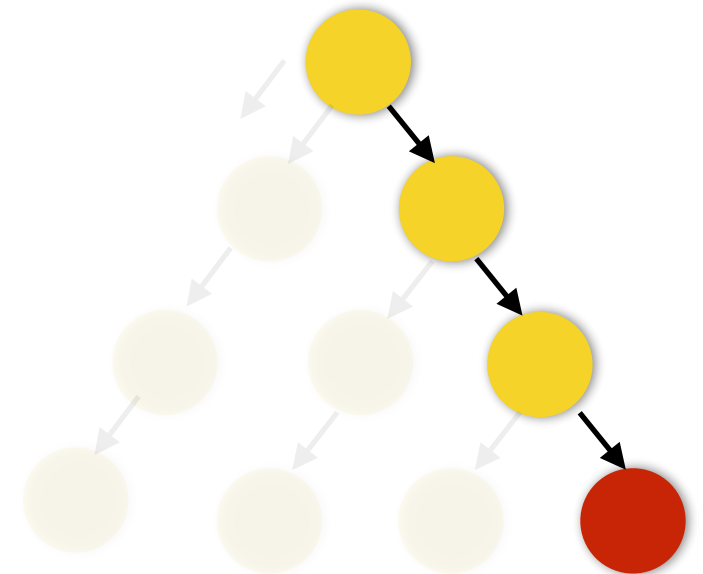
$t_1 : m1$



$t_2 : m2$



$t_3 : m3$



...

cons₁

cons₂

cons₃

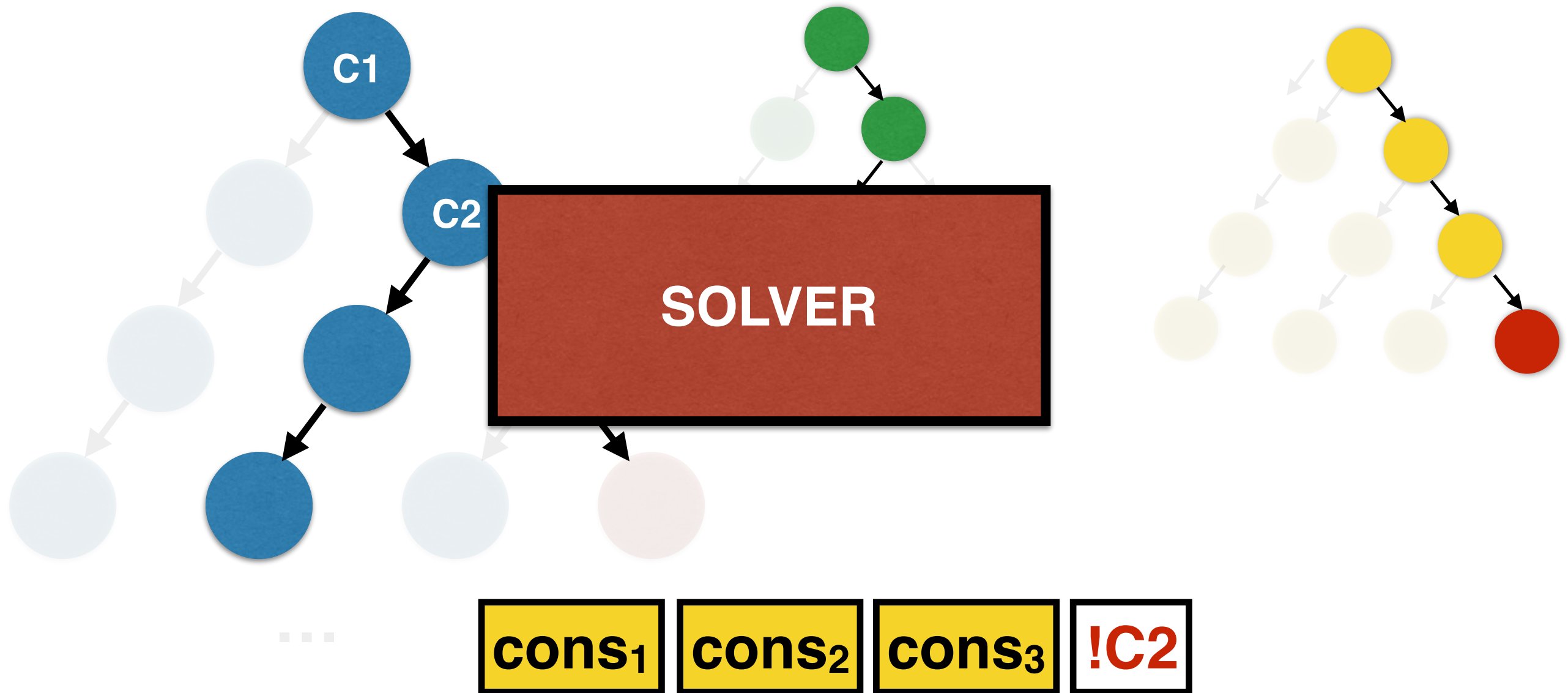
!C2

Iteration - 4 : Concrete Execution

$t_1 : m1$

$t_2 : m2$

$t_3 : m3$

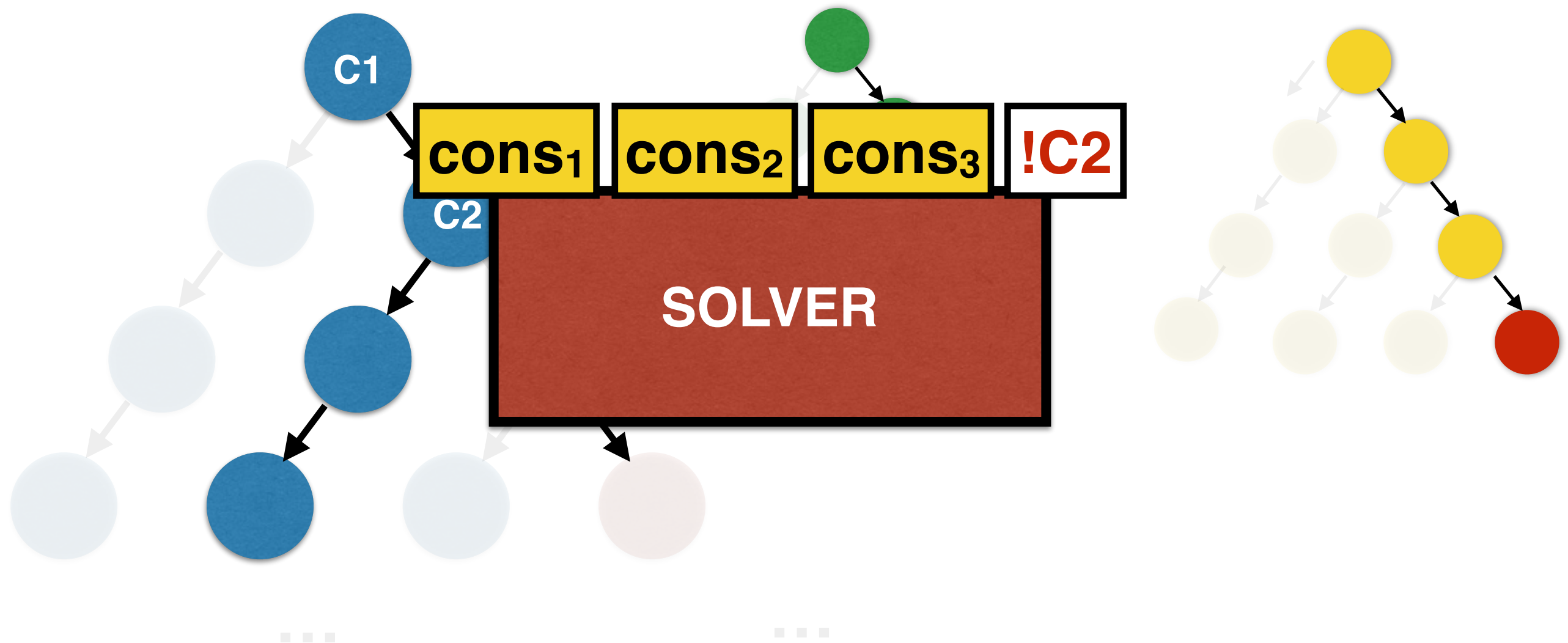


Iteration - 4 : Concrete Execution

$t_1 : m1$

$t_2 : m2$

$t_3 : m3$

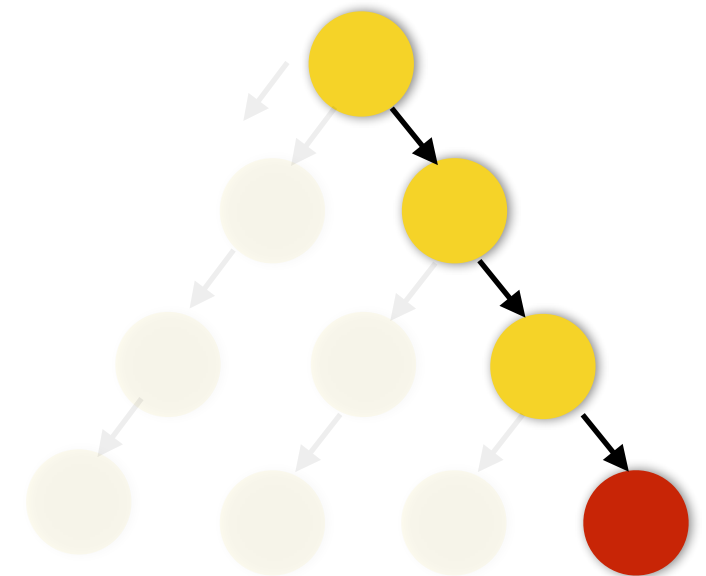
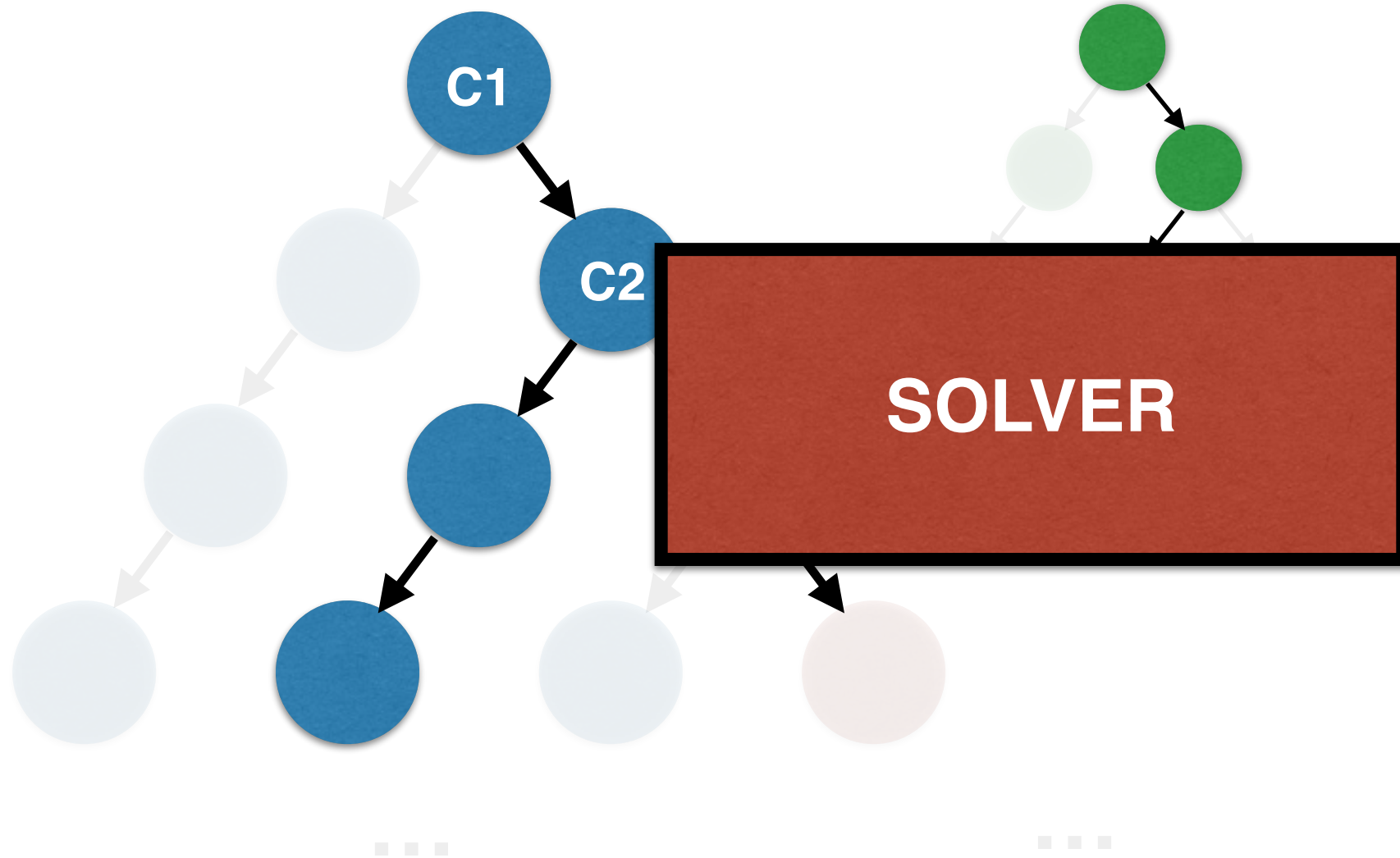


Iteration - 4 : Concrete Execution

$t_1 : m1$

$t_2 : m2$

$t_3 : m3$

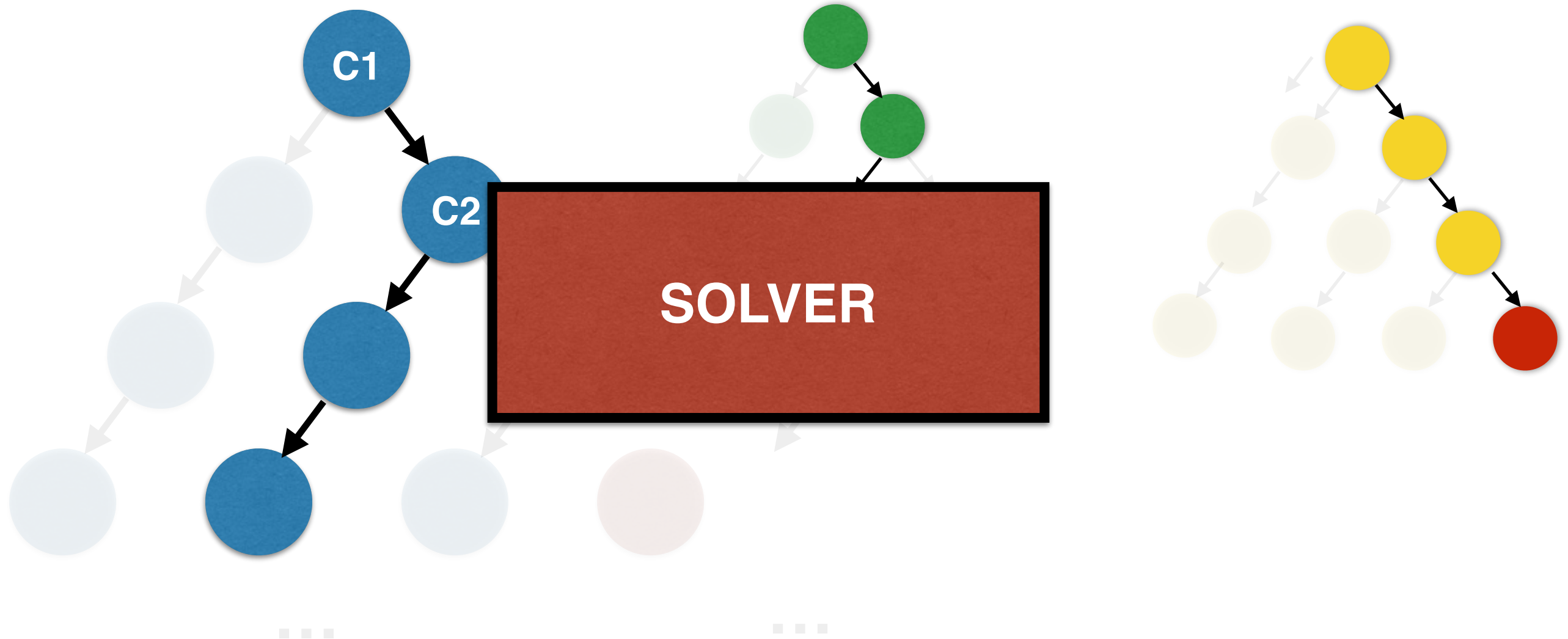


Iteration - 4 : Concrete Execution

$t_1 : m1$

$t_2 : m2$

$t_3 : m3$

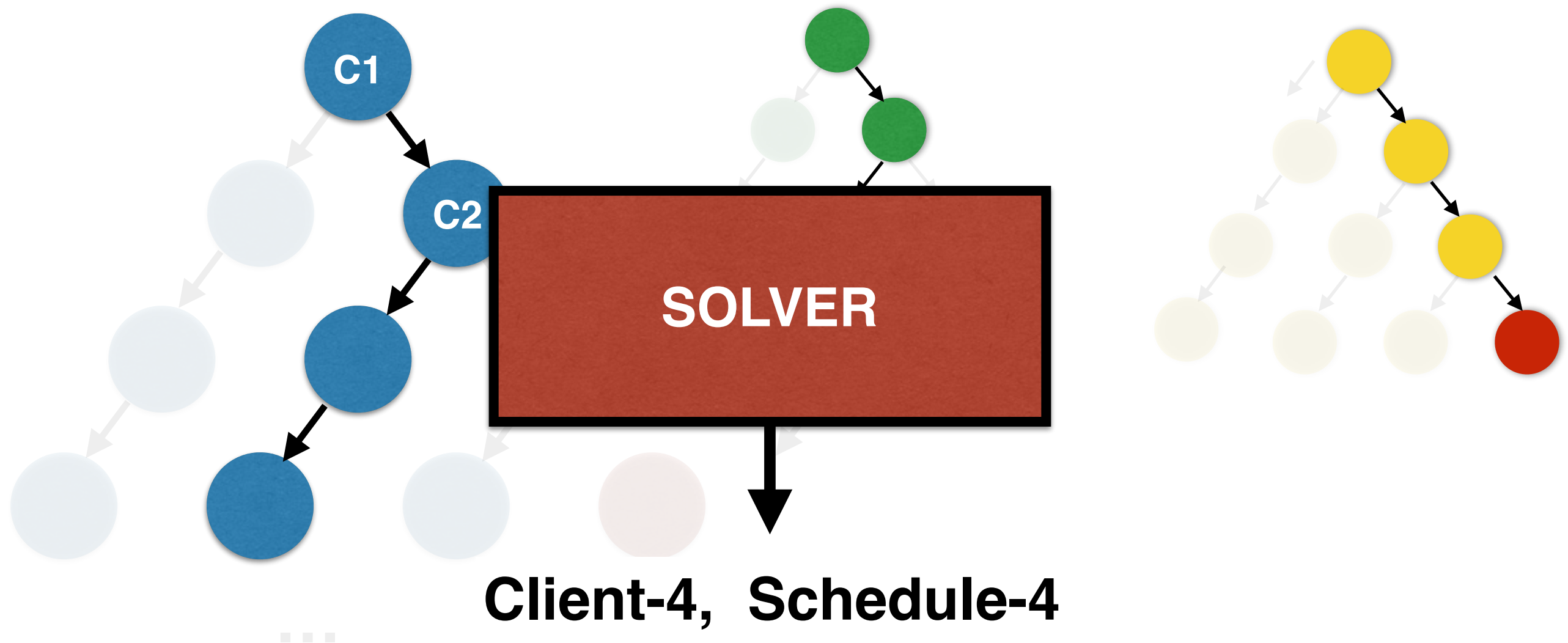


Iteration - 4 : Concrete Execution

t₁ : m1

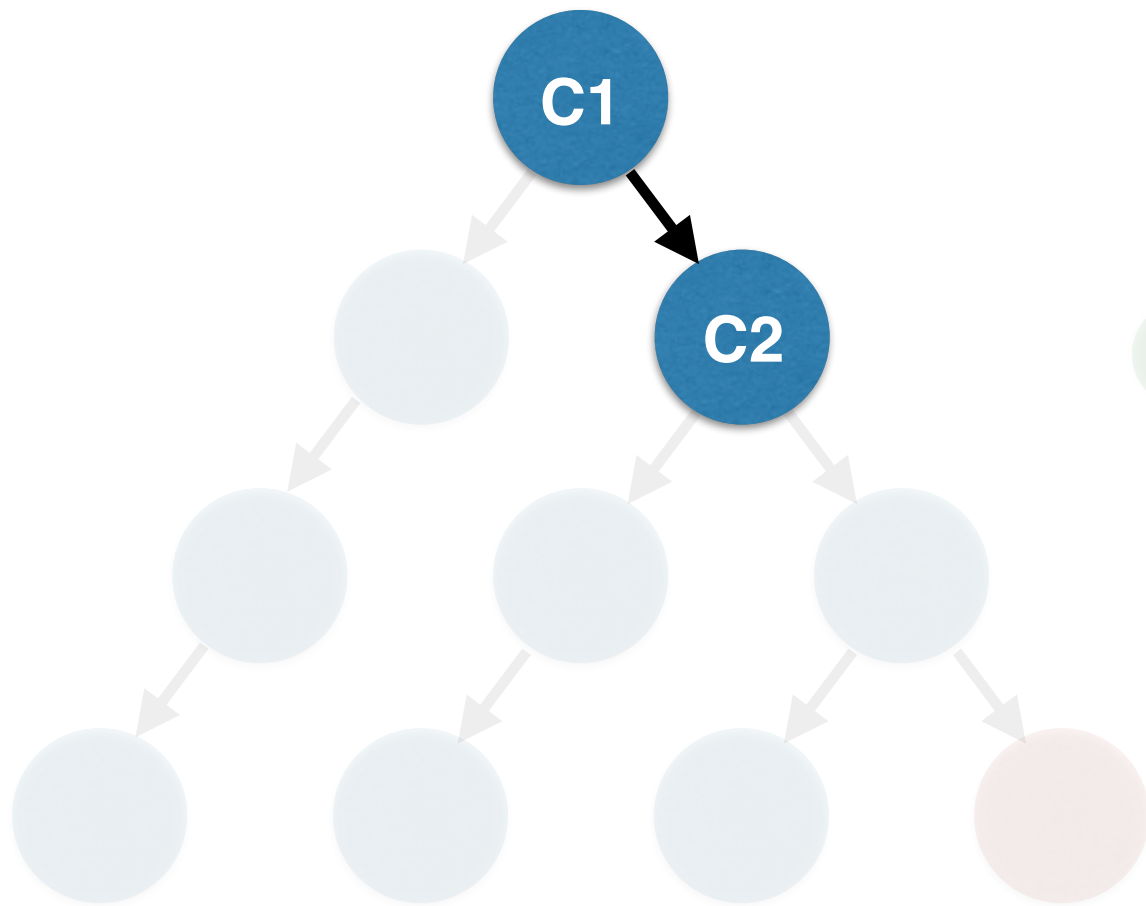
t₂ : *m2*

t₃ : *m3*

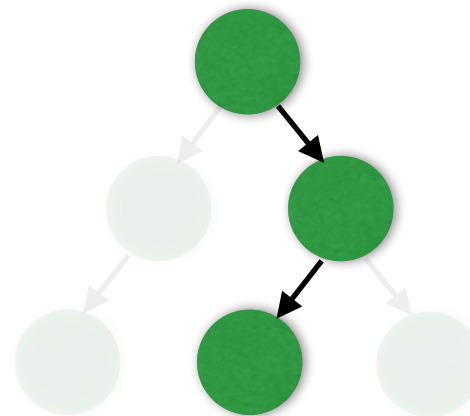


Iteration - 4 : Concrete Execution

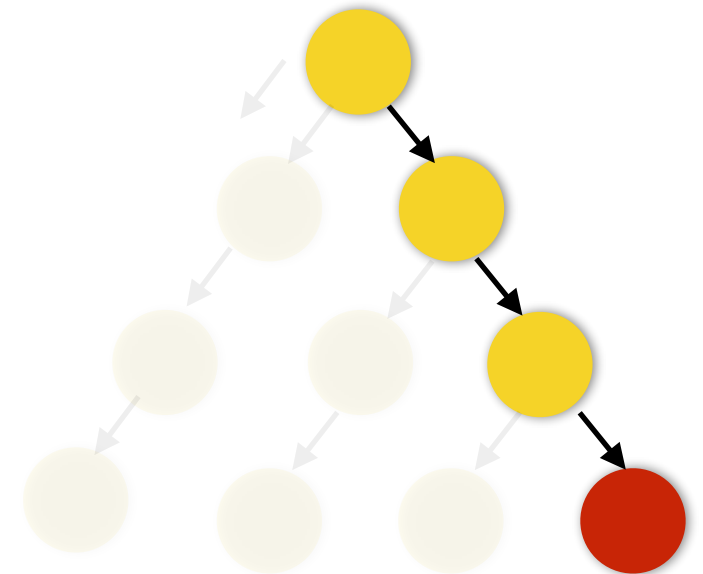
t₁ : *m1*



t₂ : *m2*



t₃ : *m3*

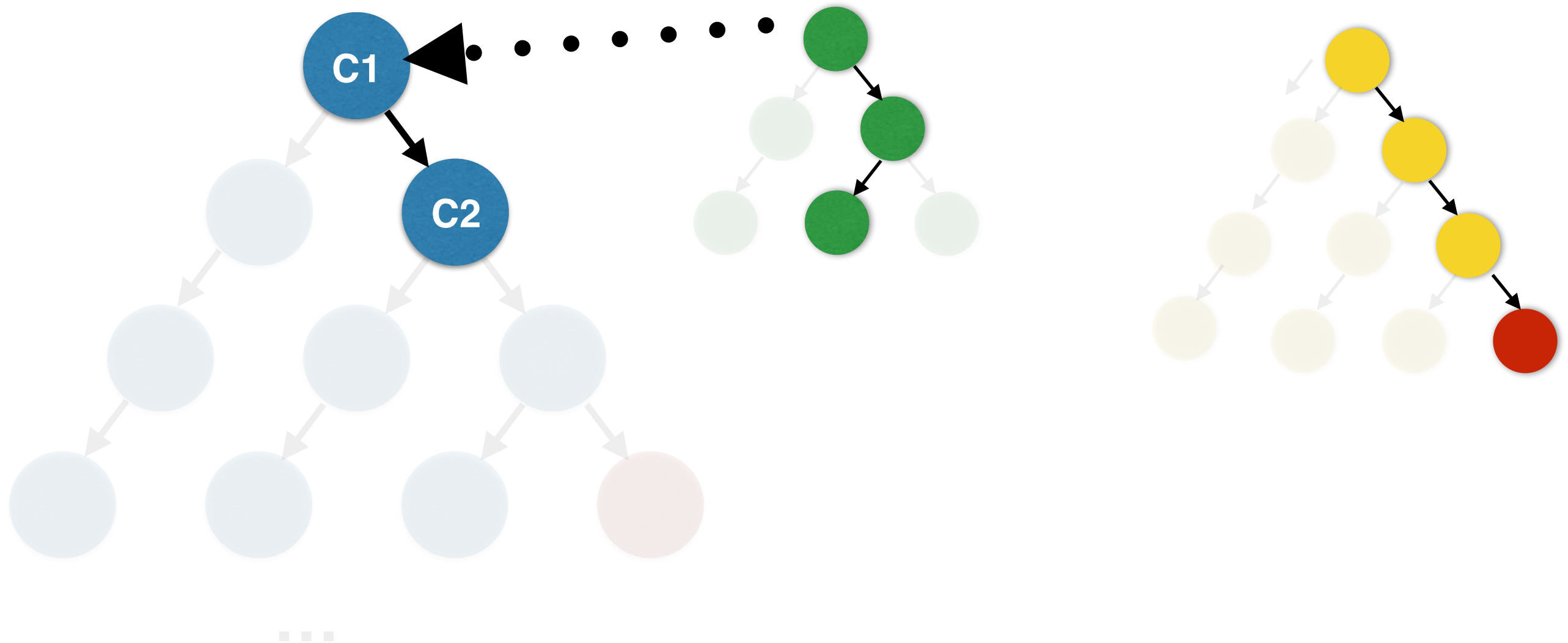


Iteration - 4 : Concrete Execution

t₁ : *m1*

t₂ : *m2*

t₃ : *m3*

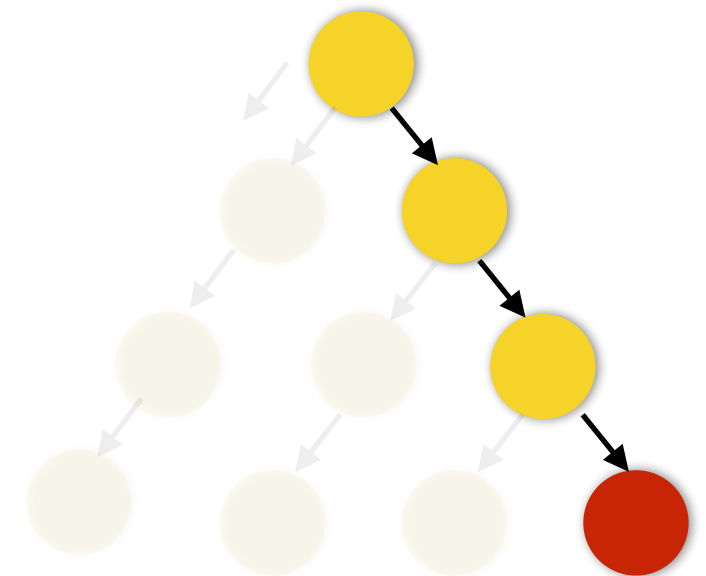
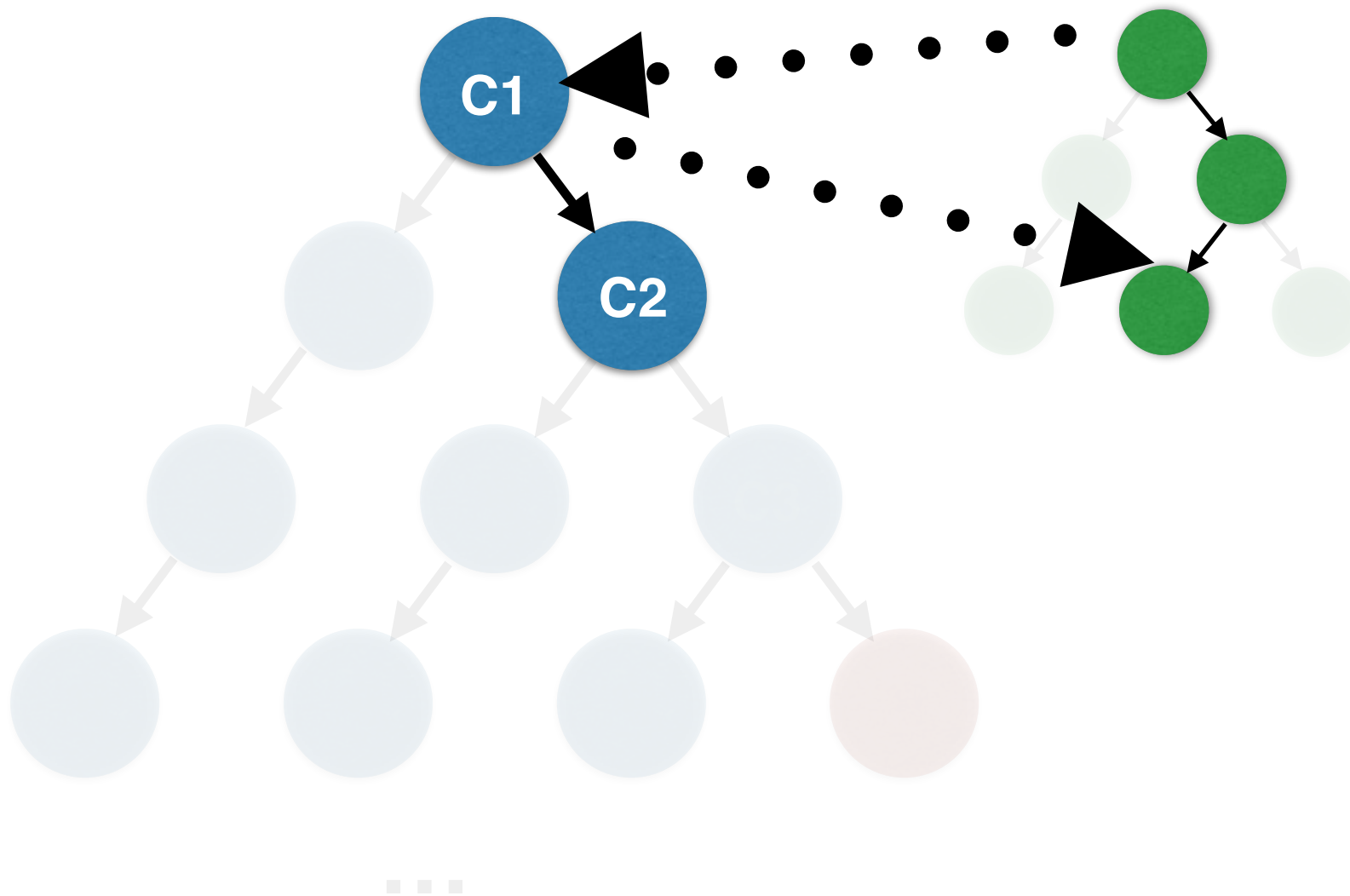


Iteration - 4 : Concrete Execution

$t_1 : m1$

$t_2 : m2$

$t_3 : m3$

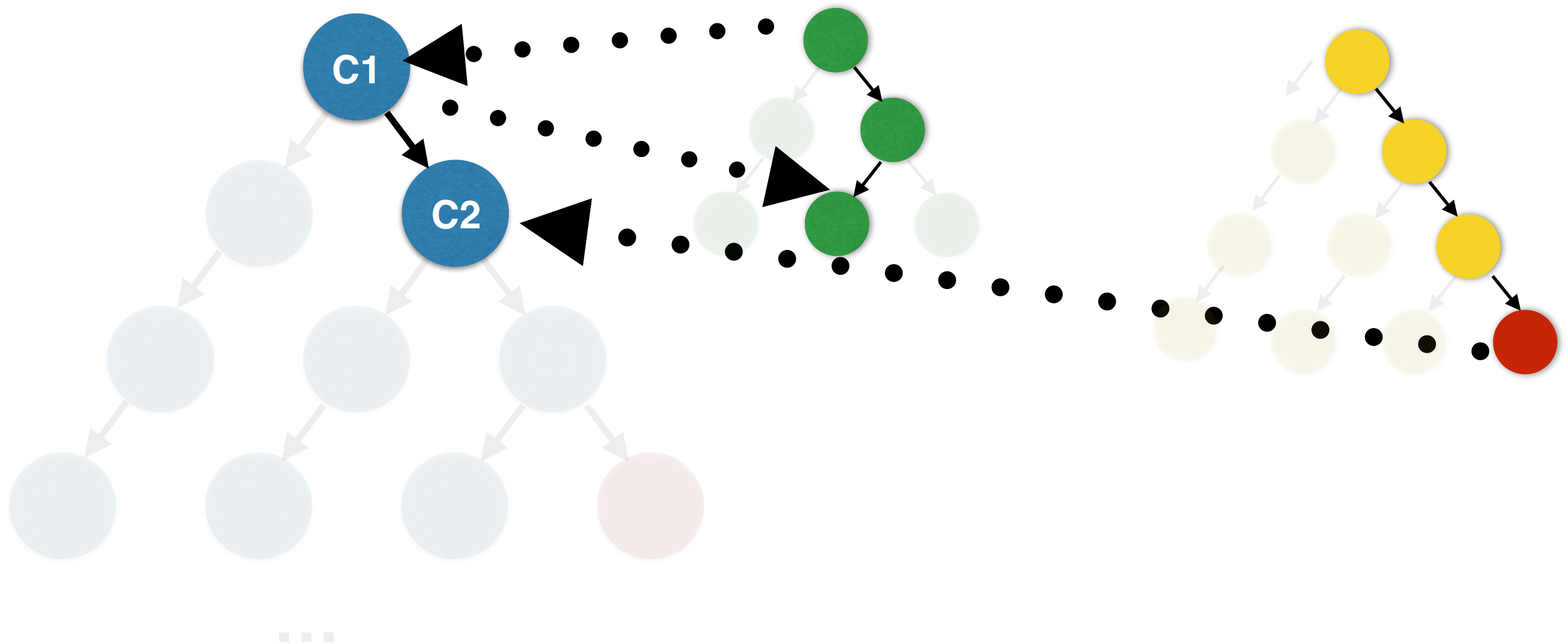


Iteration - 4 : Concrete Execution

t₁ : *m1*

t₂ : *m2*

t₃ : *m3*

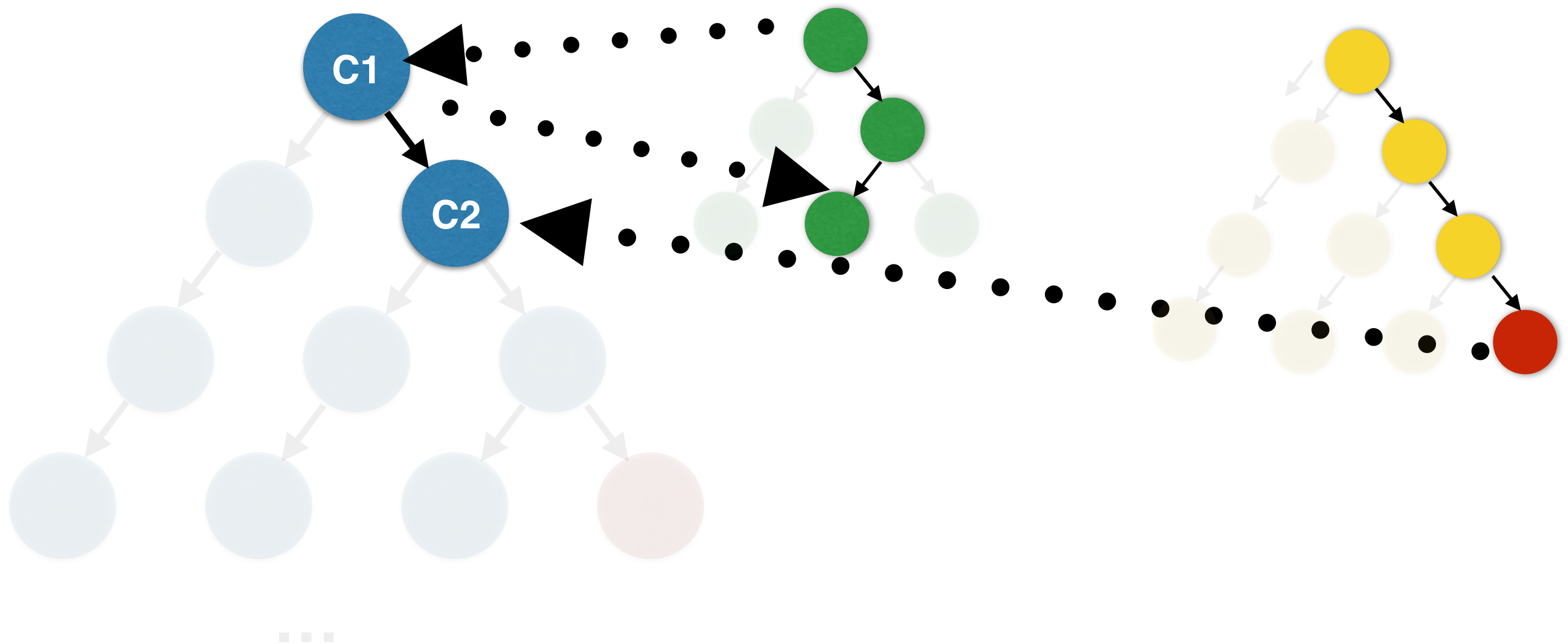


Iteration - 4 : Concrete Execution

t₁ : *m1*

t₂ : *m2*

t₃ : *m3*

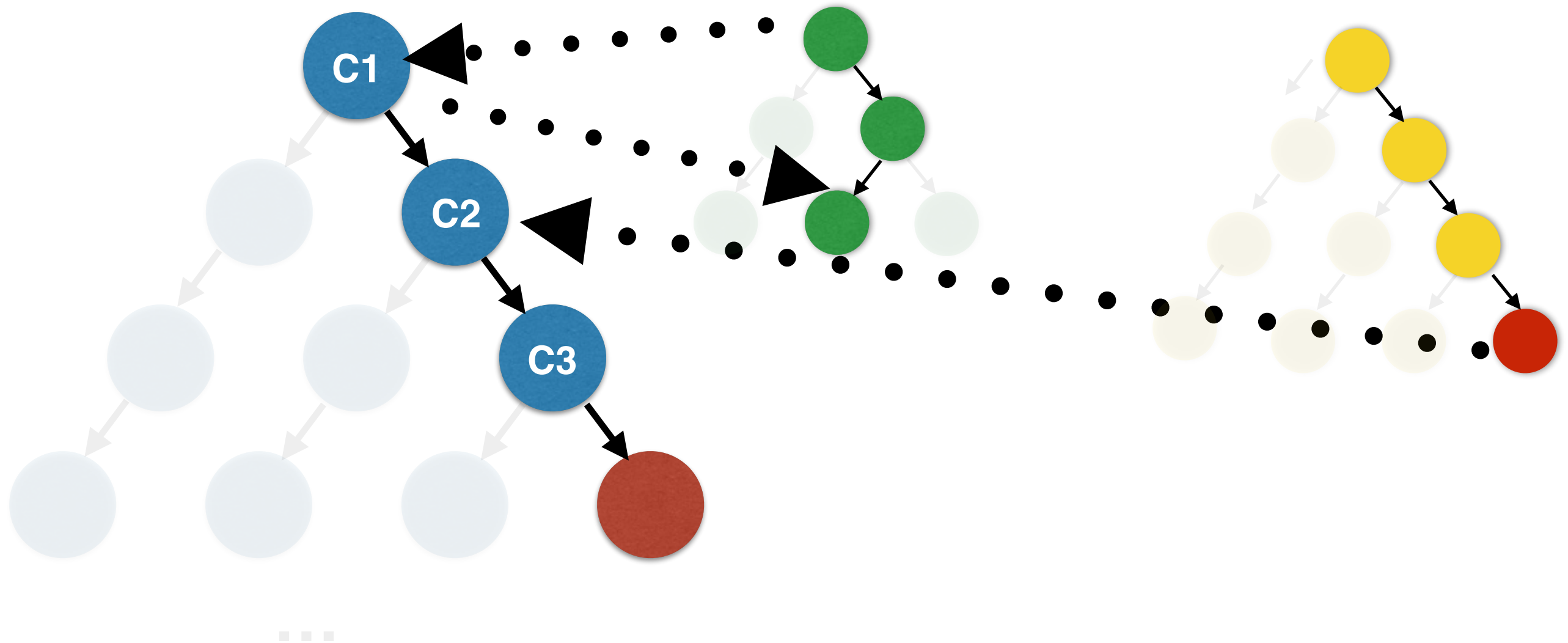


Iteration - 4 : Concrete Execution

$t_1 : m1$

$t_2 : m2$

$t_3 : m3$

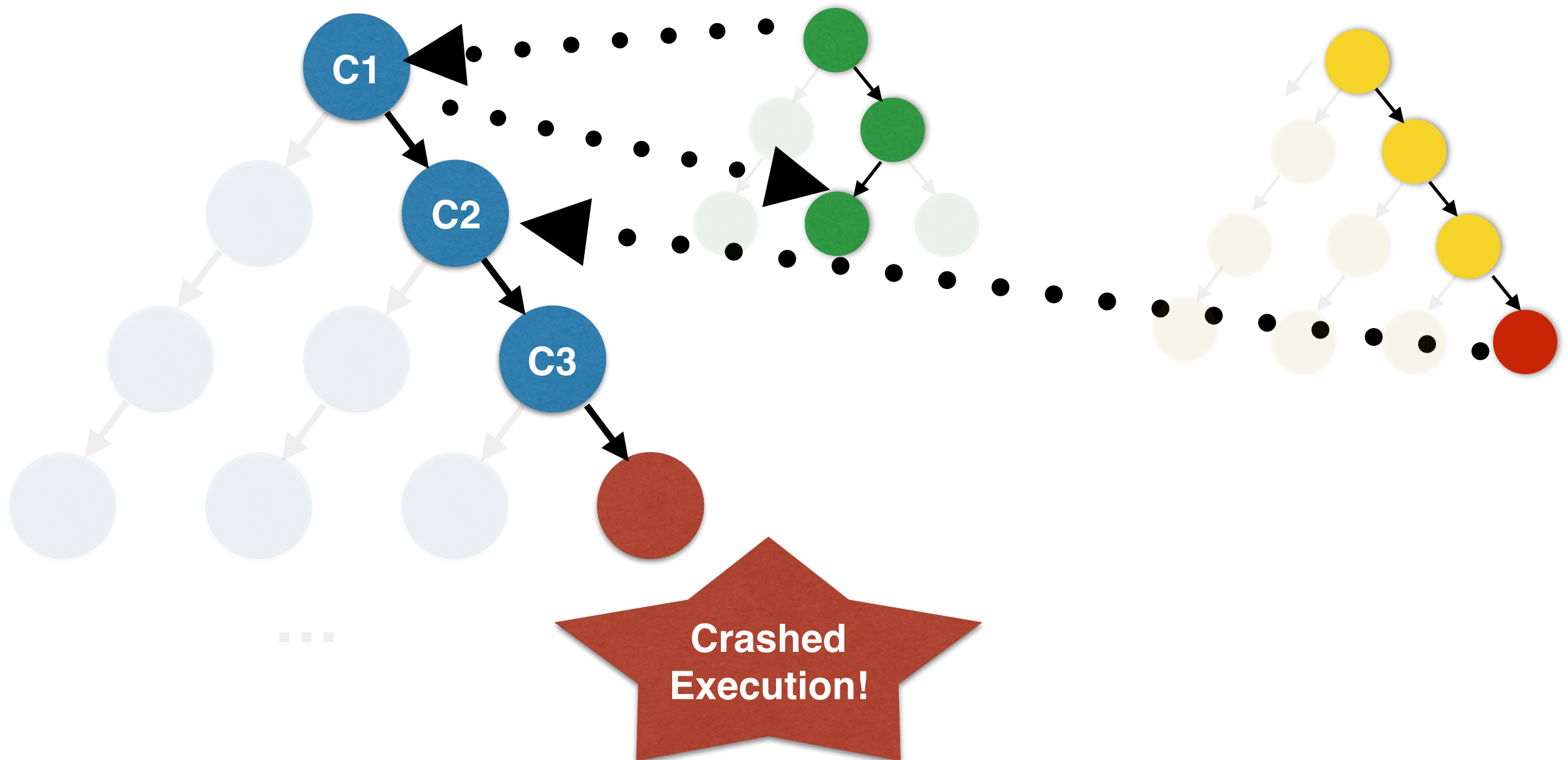


Iteration - 4 : Concrete Execution

$t_1 : m1$

$t_2 : m2$

$t_3 : m3$



Implementation

Implementation

- ◆ Built on top of the `soot` bytecode analysis framework

Implementation

- ◆ Built on top of the **soot** bytecode analysis framework
- ◆ Used the **Z3** constraint solver

Implementation

- ◆ Built on top of the **soot** bytecode analysis framework
- ◆ Used the **Z3** constraint solver
- ◆ Evaluated on open source Java libraries

Implementation

- ◆ Built on top of the **soot** bytecode analysis framework
- ◆ Used the **Z3** constraint solver
- ◆ Evaluated on open source Java libraries
- ◆ Input sequential client: invoke each method in a class once with random objects

Benchmark Information

Benchmark	Version	Class name	M
cache4j	0.4	CacheCleaner (C1)	3
classpath	0.99	BufferedInputStream (C2)	10
guava	18.0	SimpleStatsCounter (C3)	6
hsqldb	2.3.3	DoubleIntIndex (C4)	32
java.lang	1.7	StringBuffer (C5)	50
java.io	1.8	CharArrayReader (C6)	8
		PipedReader (C7)	5
		PushbackReader (C8)	11
		StringReader (C9)	8
java.util	1.7	Vector (C10)	43

Benchmark Information

Benchmark	Version	Class name	M
cache4j	0.4	CacheCleaner (C1)	3
classpath	0.99	BufferedInputStream (C2)	10
guava	18.0	SimpleStatsCounter (C3)	6
hsqldb	2.3.3	DoubleIntIndex (C4)	32
java.lang	1.7	StringBuffer (C5)	50
java.io	1.8	CharArrayReader (C6)	8
		PipedReader (C7)	5
		PushbackReader (C8)	11
		StringReader (C9)	8
java.util	1.7	Vector (C10)	43

Benchmark Information

Benchmark	Version	Class name	M
cache4j	0.4	CacheCleaner (C1)	3
classpath	0.99	BufferedInputStream (C2)	10
guava	18.0	SimpleStatsCounter (C3)	6
hsqldb	2.3.3	DoubleIntIndex (C4)	32
java.lang	1.7	StringBuffer (C5)	50
java.io	1.8	CharArrayReader (C6)	8
		PipedReader (C7)	5
		PushbackReader (C8)	11
		StringReader (C9)	8
java.util	1.7	Vector (C10)	43

Benchmark Information

Benchmark	Version	Class name	M
cache4j	0.4	CacheCleaner (C1)	3
classpath	0.99	BufferedInputStream (C2)	10
guava	18.0	SimpleStatsCounter (C3)	6
hsqldb	2.3.3	DoubleIntIndex (C4)	32
java.lang	1.7	StringBuffer (C5)	50
java.io	1.8	CharArrayReader (C6)	8
		PipedReader (C7)	5
		PushbackReader (C8)	11
		StringReader (C9)	8
java.util	1.7	Vector (C10)	43

Results

Class	# of Targets	# of Constraints	Schedule length	Time (sec)	Violations	
					Seq.	Conc.
C1	2	254	37	21	0	1
C2	8	8189	25	70	2	6
C3	3	1898	43	10	0	2
C4	8	4105	10	94	4	4
C5	12	14646	116	664	5	1
C6	4	717	14	20	1	3
C7	10	3355	30	6	2	1
C8	23	7276	19	324	8	9
C9	4	900	20	21	1	3
C10	6	56045	74	120	0	1
Total	80			1350	23	31

Results

Class	# of Targets	# of Constraints	Schedule length	Time (sec)	Violations	
					Seq.	Conc.
C1	2	254	37	21	0	1
C2	8	8189	25	70	2	6
C3	3	1898	43	10	0	2
C4	8	4105	10	94	4	4
C5	12	14646	116	664	5	1
C6	4	717	14	20	1	3
C7	10	3355	30	6	2	1
C8	23	7276	19	324	8	9
C9	4	900	20	21	1	3
C10	6	56045	74	120	0	1
Total	80			1350	23	31

Results

Class	# of Targets	# of Constraints	Schedule length	Time (sec)	Violations	
					Seq.	Conc.
C1	2	254	37	21	0	1
C2	8	8189	25	70	2	6
C3	3	1898	43	10	0	2
C4	8	4105	10	94	4	4
C5	12	14646	116	664	5	1
C6	4	717	14	20	1	3
C7	10	3355	30	6	2	1
C8	23	7276	19	324	8	9
C9	4	900	20	21	1	3
C10	6	56045	74	120	0	1
Total	80			1350	23	31

Results

Class	# of Targets	# of Constraints	Schedule length	Time (sec)	Violations	
					Seq.	Conc.
C1	2	254	37	21	0	1
C2	8	8189	25	70	2	6
C3	3	1898	43	10	0	2
C4	8	4105	10	94	4	4
C5	12	14646	116	664	5	1
C6	4	717	14	20	1	3
C7	10	3355	30	6	2	1
C8	23	7276	19	324	8	9
C9	4	900	20	21	1	3
C10	6	56045	74	120	0	1
Total	80			1350	23	31

Results

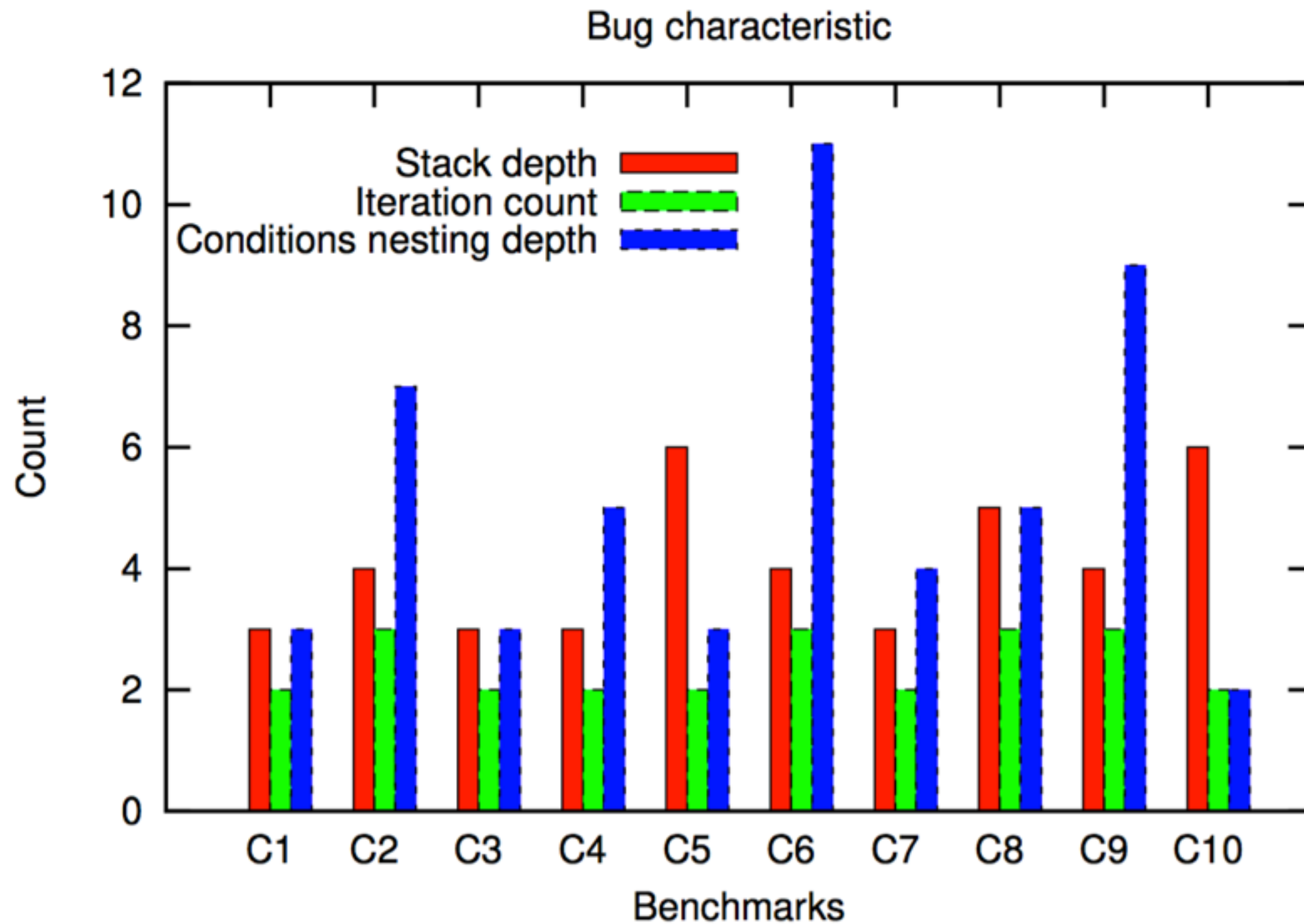
Class	# of Targets	# of Constraints	Schedule length	Time (sec)	Violations	
					Seq.	Conc.
C1	2	254	37	21	0	1
C2	8	8189	25	70	2	6
C3	3	1898	43	10	0	2
C4	8	4105	10	94	4	4
C5	12	14646	116	664	5	1
C6	4	717	14	20	1	3
C7	10	3355	30	6	2	1
C8	23	7276	19	324	8	9
C9	4	900	20	21	1	3
C10	6	56045	74	120	0	1
Total	80			1350	23	31

Results

Class	# of Targets	# of Constraints	Schedule length	Time (sec)	Violations	
					Seq.	Conc.
C1	2	254	37	21	0	1
C2	8	8189	25	70	2	6
C3	3	1898	43	10	0	2
C4	8	4105	10	94	4	4
C5	12	14646	116	664	5	1
C6	4	717	14	20	1	3
C7	10	3355	30	6	2	1
C8	23	7276	19	324	8	9
C9	4	900	20	21	1	3
C10	6	56045	74	120	0	1
Total	80			1350	23	31

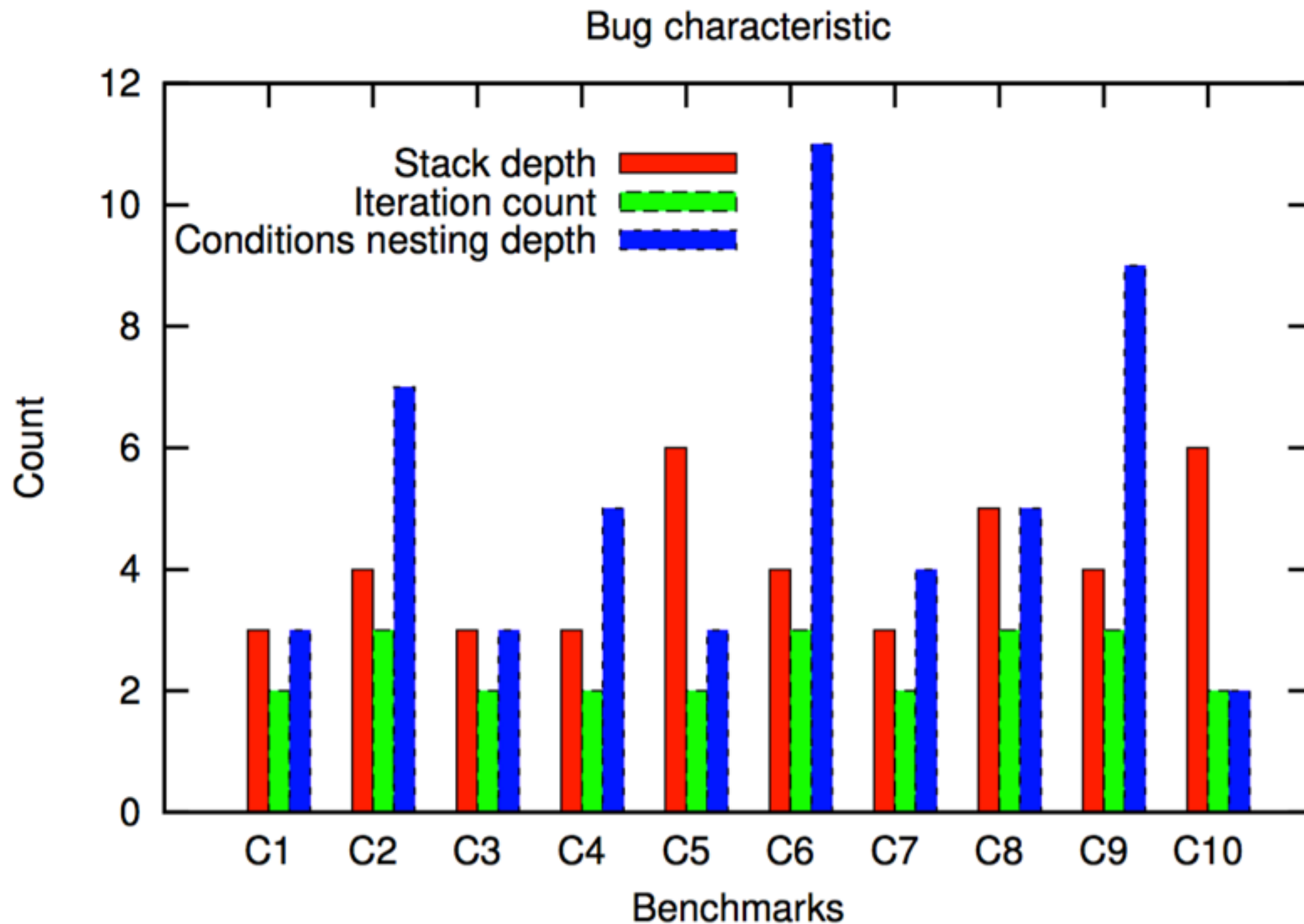
31 crashes due to concurrency

Bug Characteristics



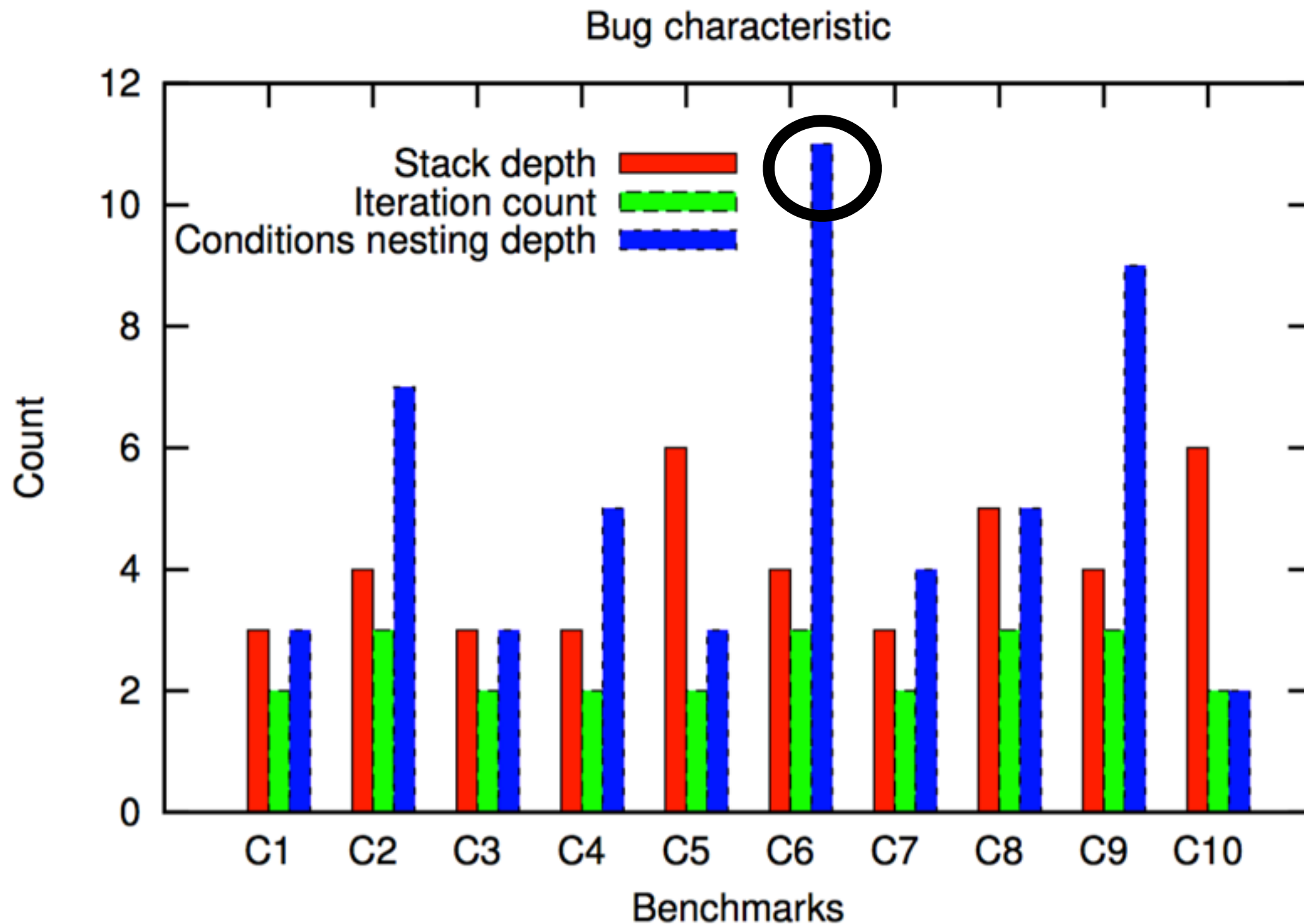
Bug Characteristics

MINION detects complex concurrency issues



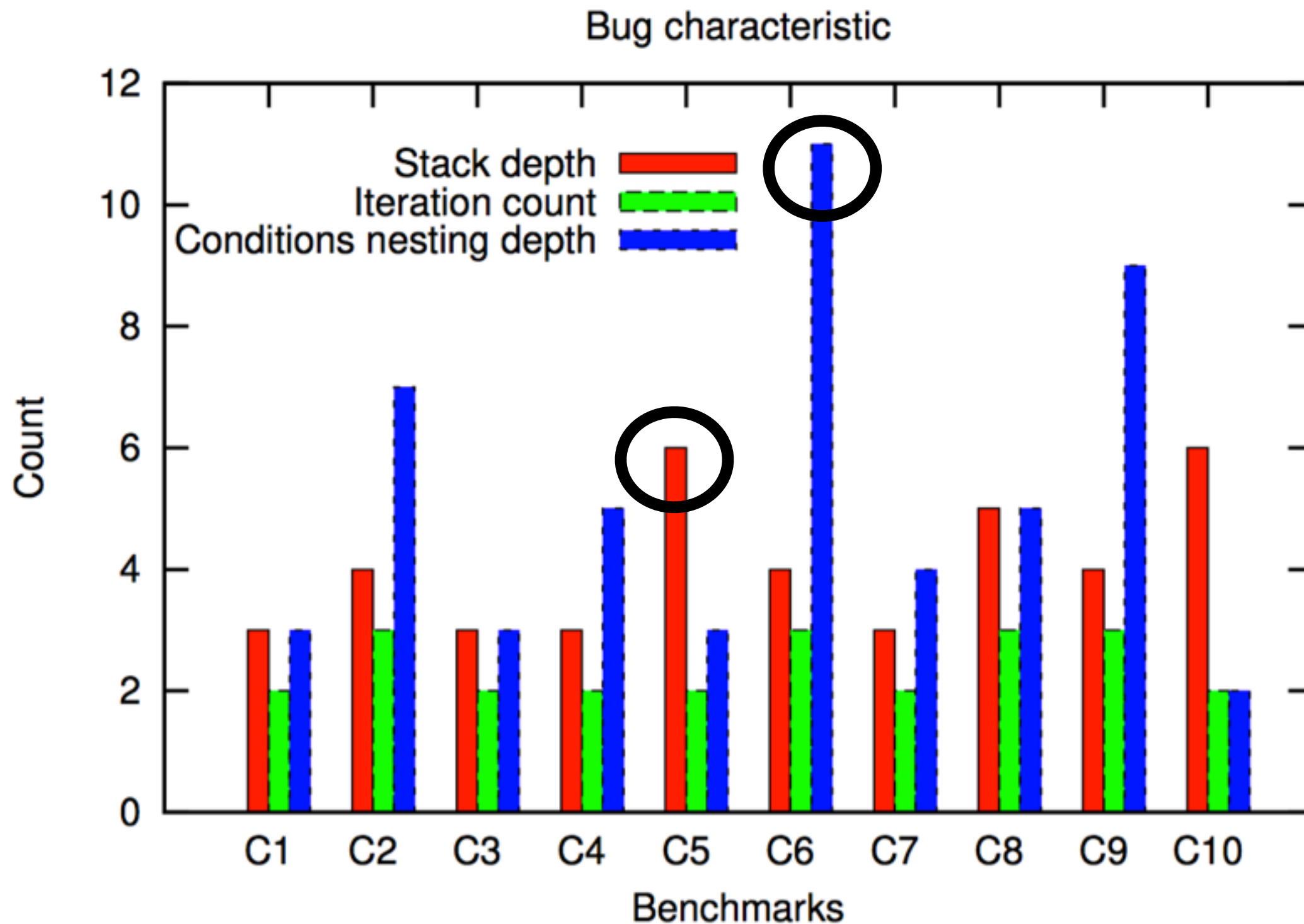
Bug Characteristics

MINION detects complex concurrency issues



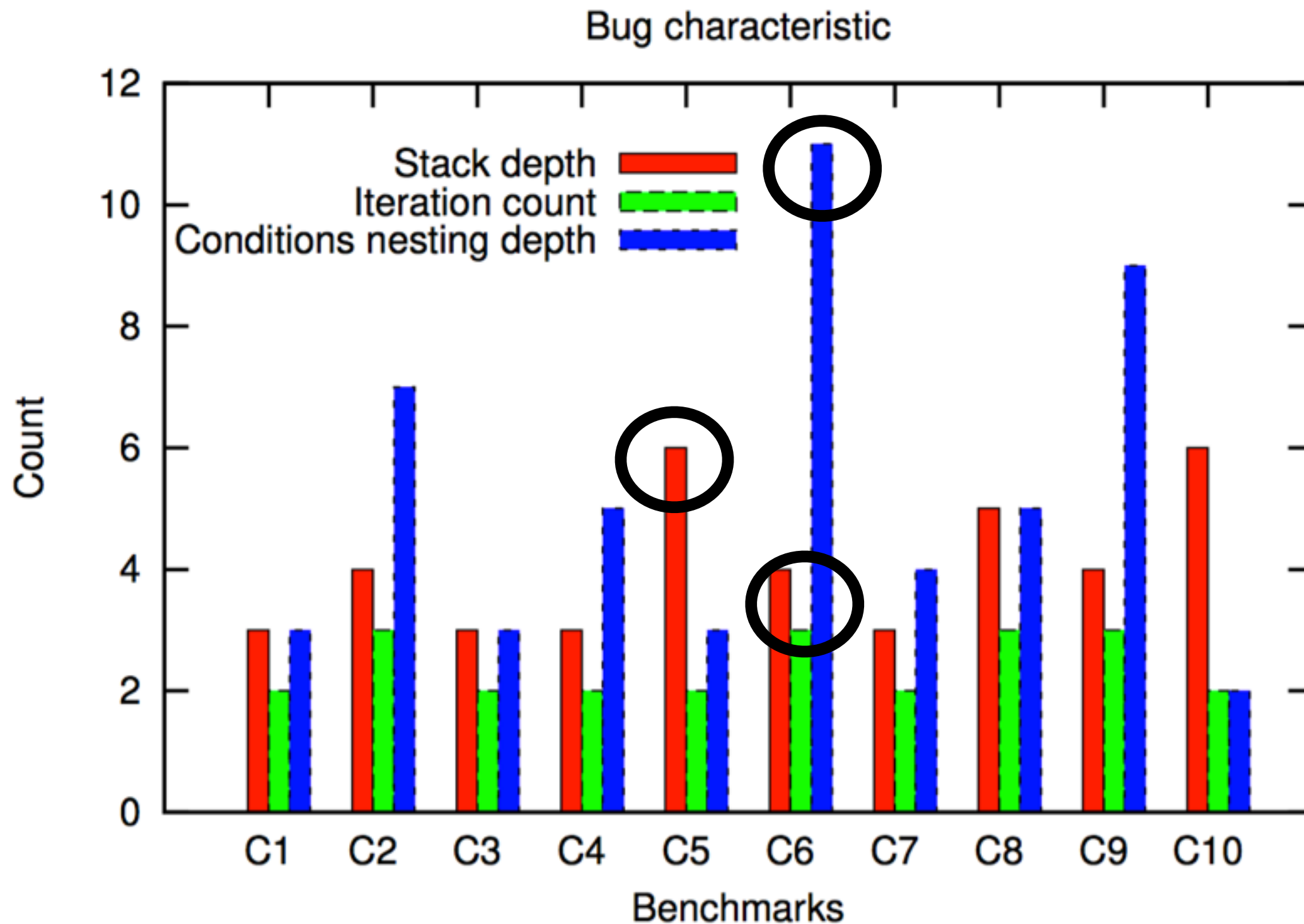
Bug Characteristics

MINION detects complex concurrency issues



Bug Characteristics

MINION detects complex concurrency issues



Summary

- ◆ Designed a directed synthesis of failing concurrent executions
 - ◆ Integrates testing, symbolic execution and static analysis
- ◆ Validated on **10** well tested and popular Java classes
- ◆ Detected **31** crashes .
 - ◆ Resulted in fixes (includes classes in **JDK 8**)
 - ◆ Total time for analyzing all classes is approximately 23 minutes
 - ◆ Maximum nested path conditions: 11, stack depth: 6