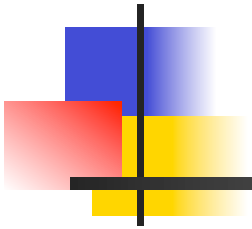
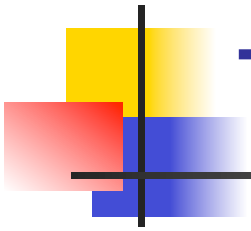


Nondeterminism and Search Exploration

Pascal Van Hentenryck

Brown University



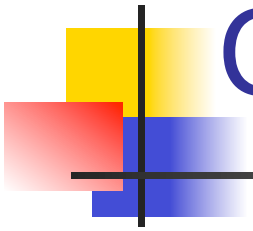
The Search

Search Procedure

=

Nondeterministic Program + Exploration Strategy

- Nondeterministic program
  - specify (implicitly) an and-or tree
- Exploration strategy
  - specify how to explore the tree



Continuation

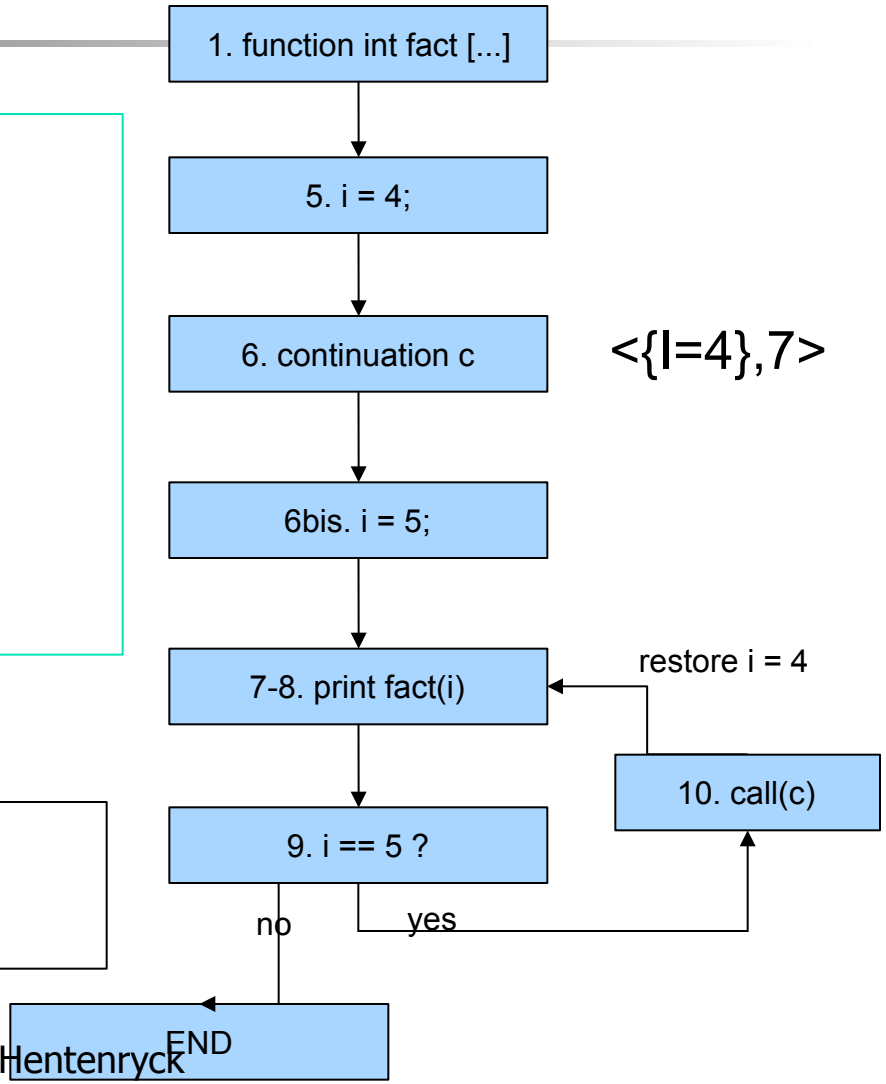
- A snapshot of the runtime data structures
  - Store a pair <I,S>
  - I points to the next instruction
  - S is the previous stack
  - When called:
    - S is restored
    - execution continues from I

Example

```
1. function int fact(int n) {
2.             if (n == 0) return 1;
3.             else return n * fact(n - 1);
4. }
5. int i = 4;
6. continuation c { i = 5; }
7. int r = fact(i);
8. cout << "fact(" << i << ") = " << r << endl;
9. if (i == 5)
10.            call(c);
```

Output:

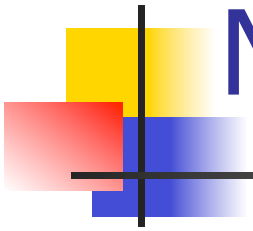
```
fact(5) = 120
fact(4) = 24
```



Nondeterministic Search

- try instruction
  - **try**<sc> left | right

```
bool rightBranch = true;
continuation c {
    sc.addChoice(c);
    rightBranch = false;
    left
}
if rightBranch
    right
```



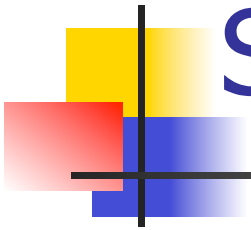
Nondeterministic Search

- tryall instruction
  - idem that try except that more than two choices can be specified

Nondeterministic Search

- solveall instruction
  - find all solutions of a nondeterministic program
  - **solveall** <sc> body

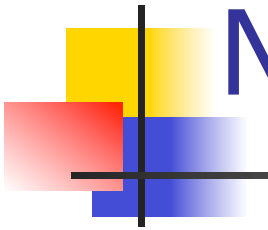
```
continuation e {
    continuation s {
        sc.start(s,e);
    }
    body
    sc.fail();
}
```



Search Exploration

- search controllers
  - defines the search exploration
  - search controllers are implementations SearchController

```
interface SearchController {
    void start(Continuation s,Continuation e);
    void exit();
    void addChoice(Continuation c);
    void fail();
}
```



Nondeterministic Search

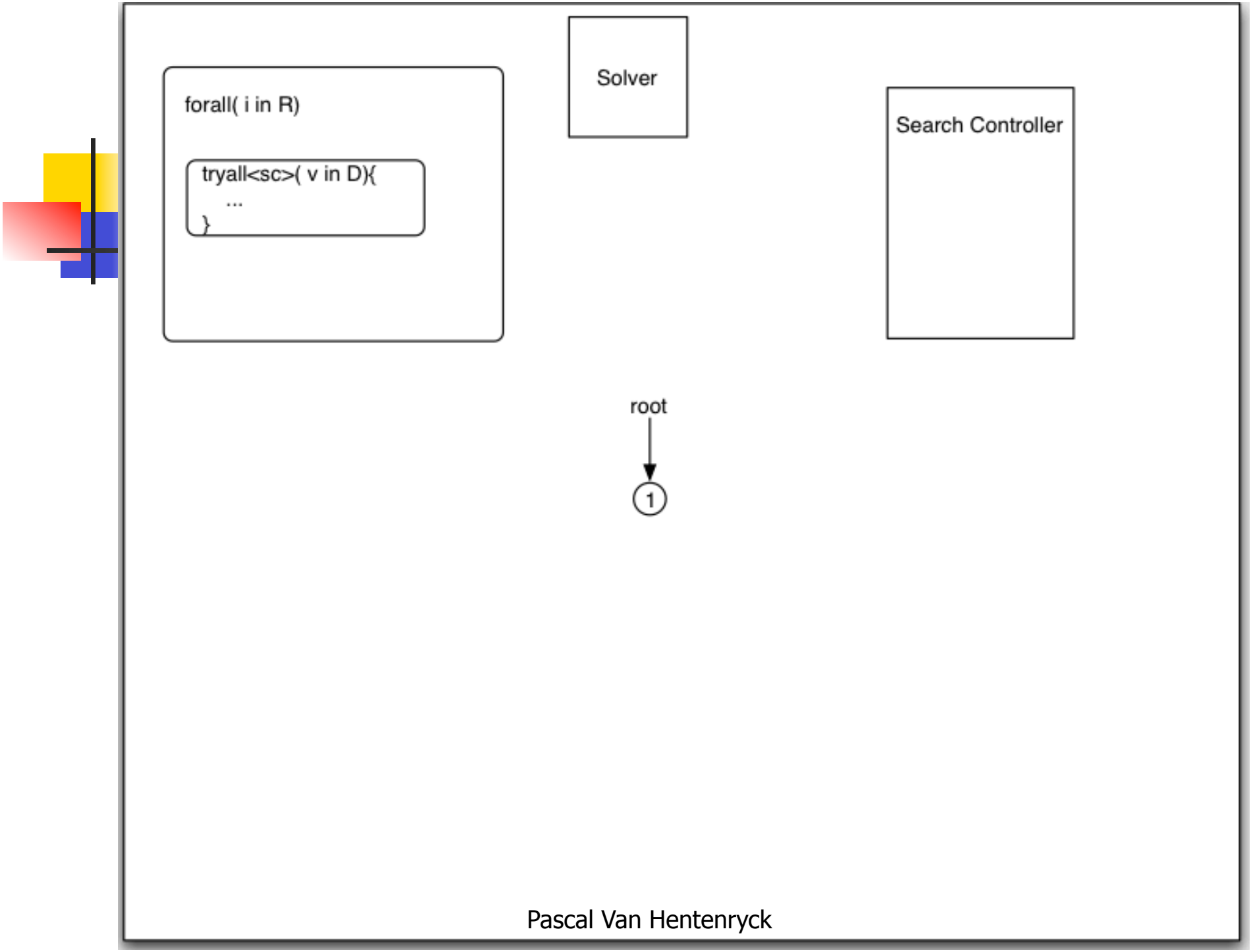
- Example of search controller : DFSearch

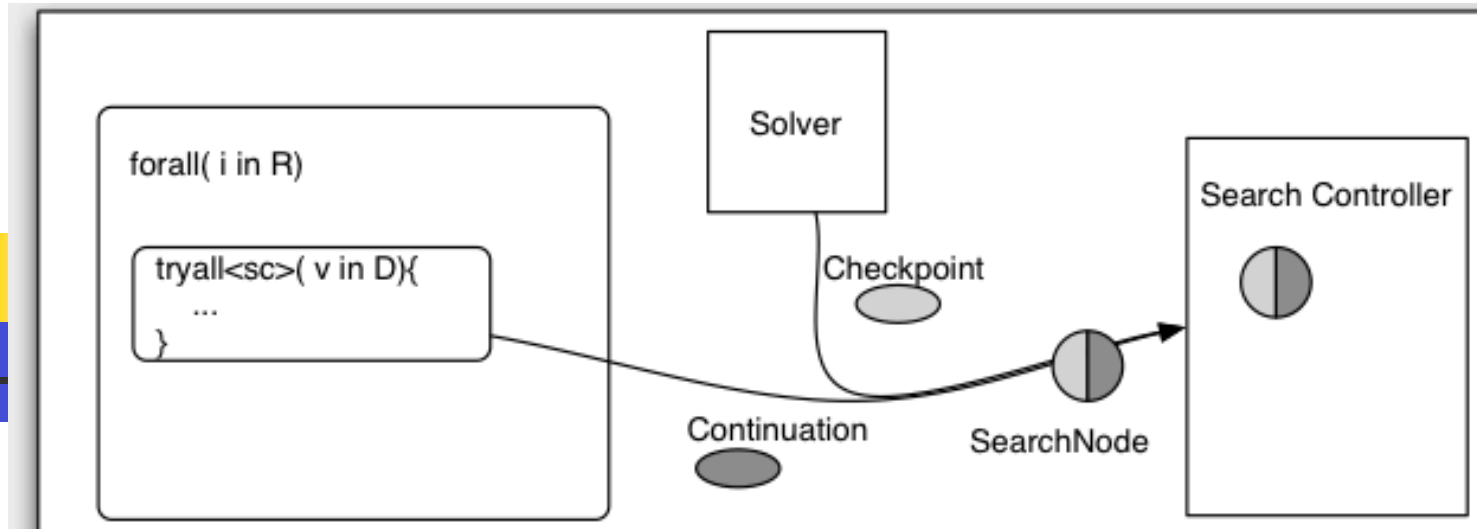
```
class DFSearch implements SearchController {
    Stack{Continuation} st;
    void addChoice(Continuation c) {
        st.push(c);
    }
    void fail() {
        if (st.empty())
            exit();
        else
            call(st.pop());
    }
}
```

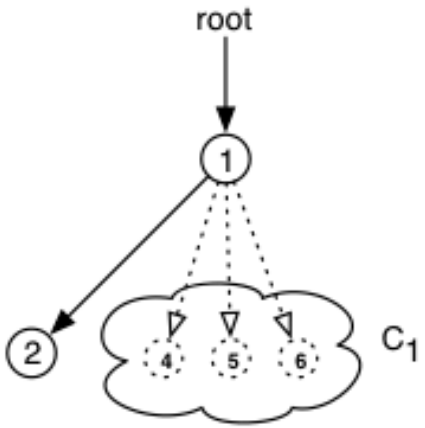
Nondeterministic Search

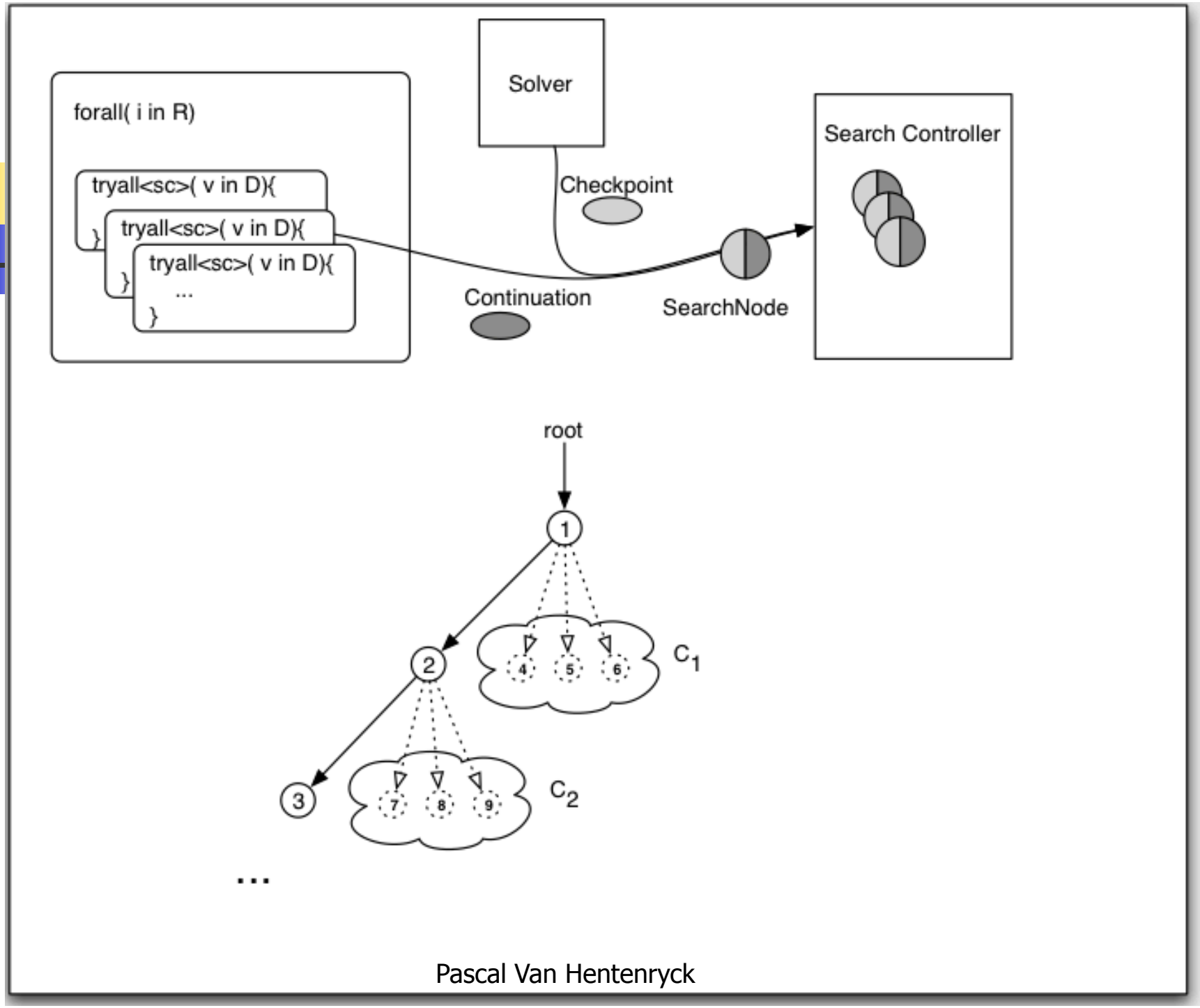
- Example of search controller : LDSearch

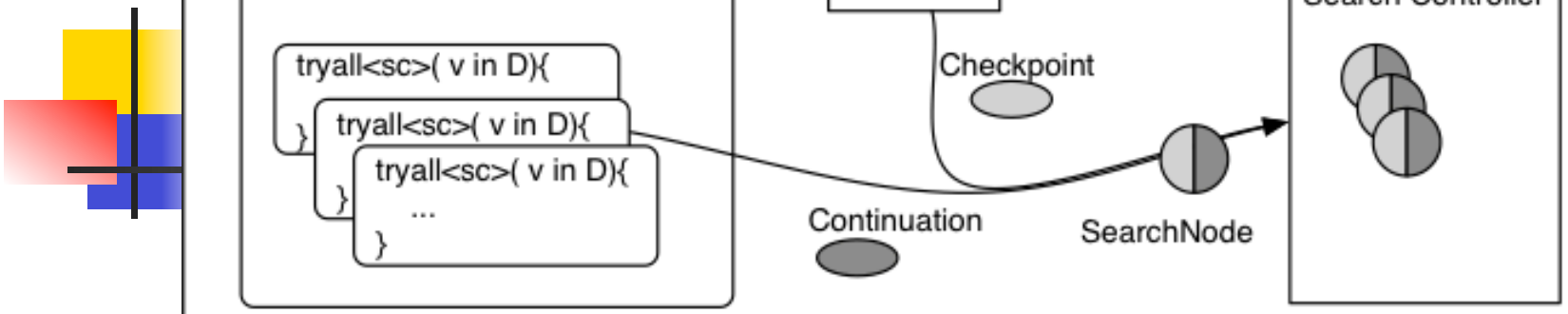
```
class LDSearch implements search controller{
    Queue{Continuation} st;
    void addChoice(Continuation c) {
        st.push(c);
    }
    void fail() {
        if (st.empty())
            exit();
        else
            call(st.pop());
    }
}
```

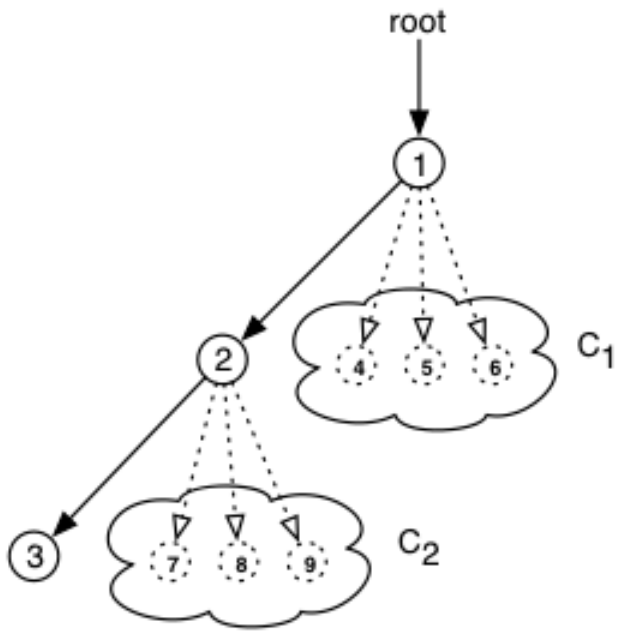

forall(i in R)
tryall<sc>(v in D){
...
}
Solver
Search Controller
root
1


forall(i in R)
tryall<sc>(v in D){
...
}
Solver
Checkpoint
Continuation
SearchNode
Search Controller

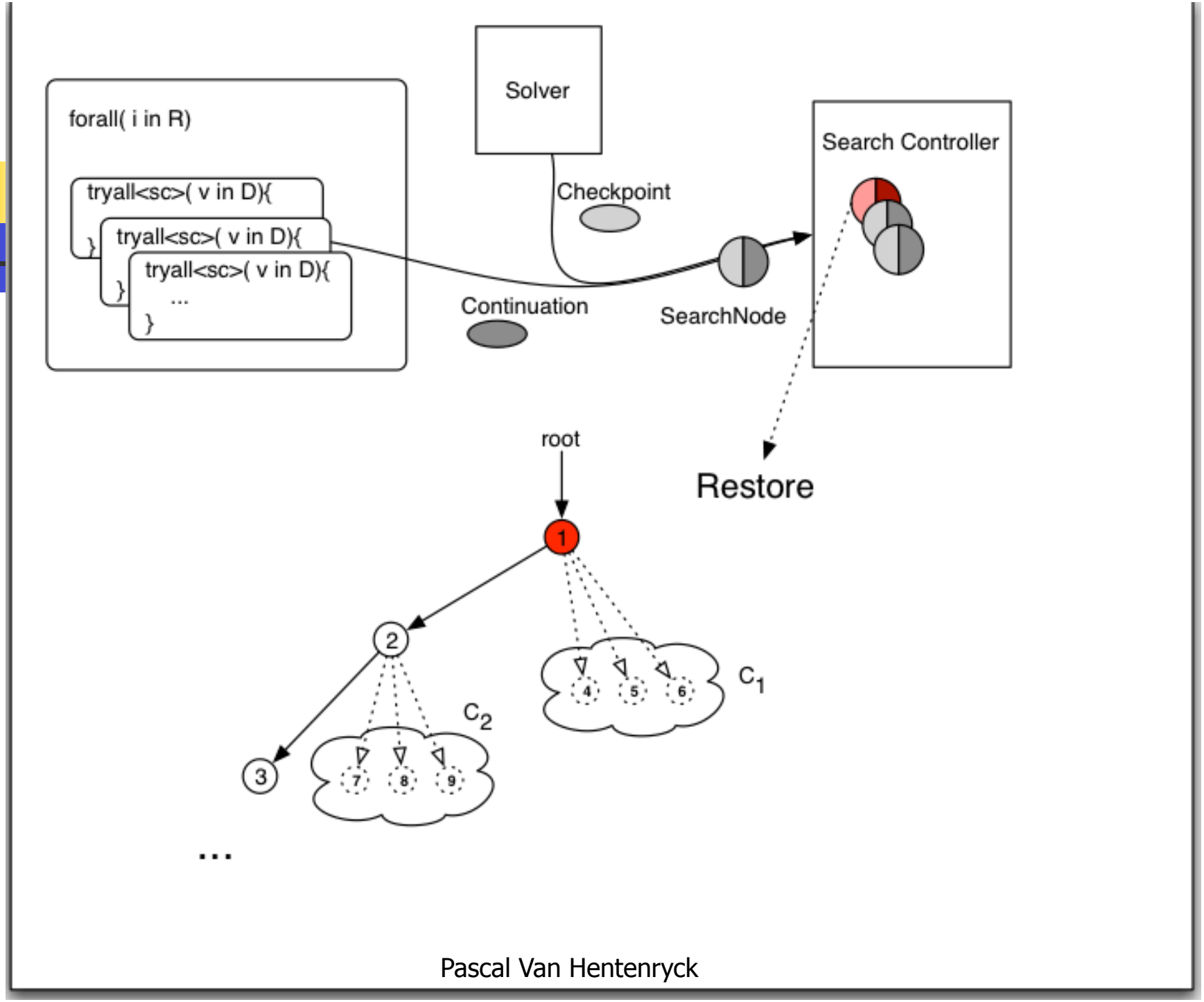

root
1
2
4
5
6
C1

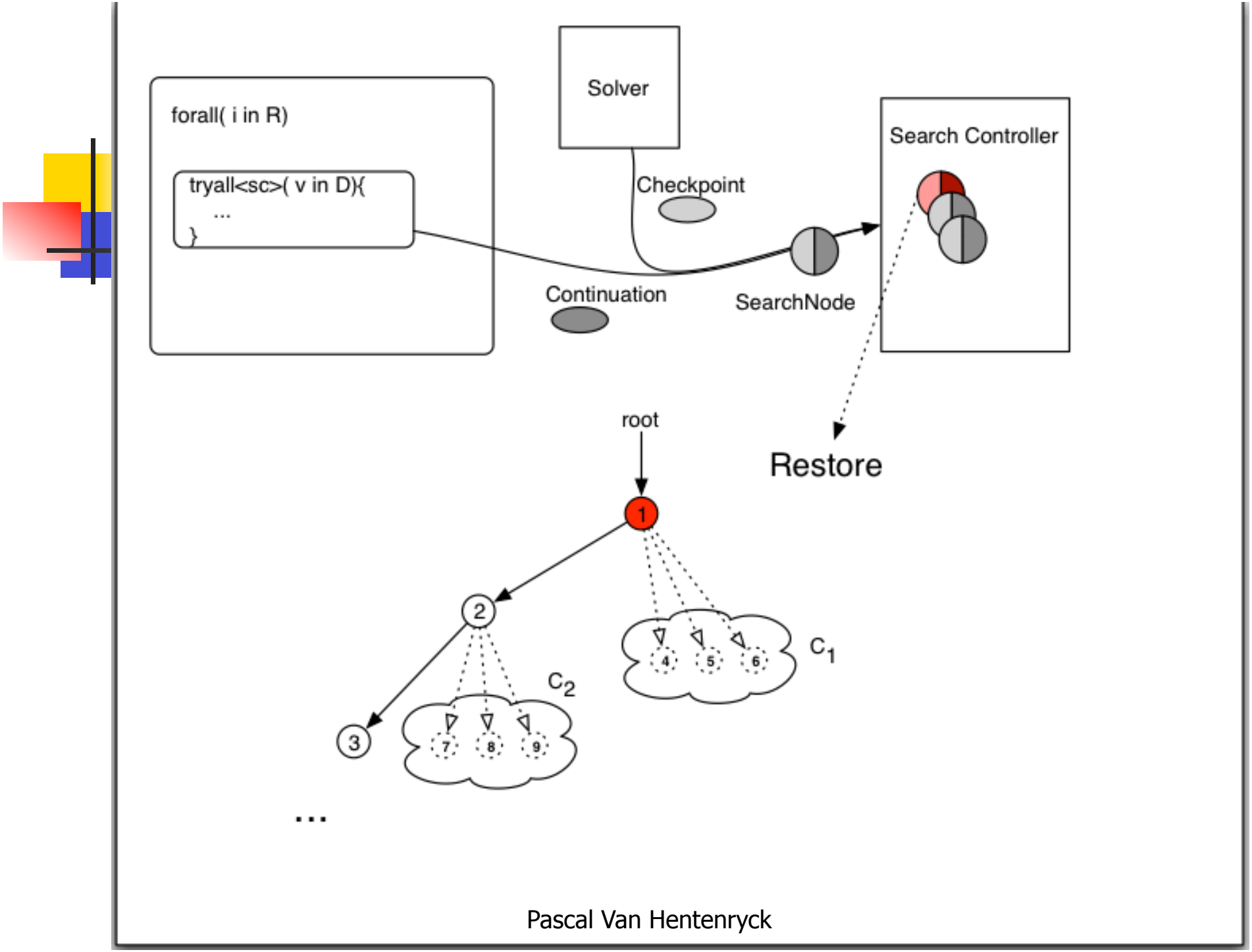

forall(i in R)
tryall<sc>(v in D){
tryall<sc>(v in D){
tryall<sc>(v in D){
...
}
}
}
Solver
Checkpoint
Continuation
SearchNode
Search Controller
root
1
2
3
4
5
6
7
8
9
C1
C2
...

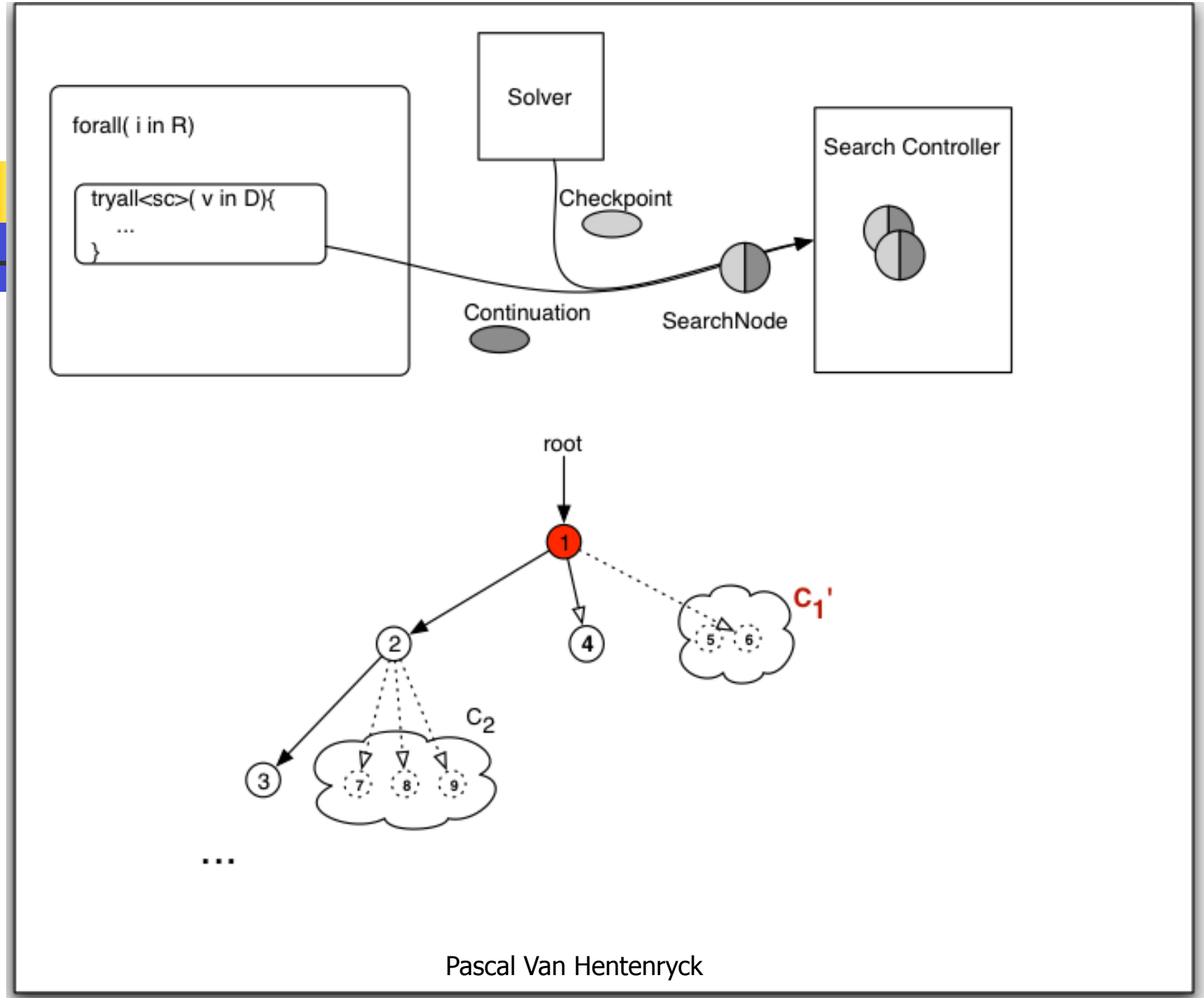

forall(i in R)
tryall<sc>(v in D){
tryall<sc>(v in D){
tryall<sc>(v in D){
...
Solver
Checkpoint
Continuation
SearchNode
Search Controller

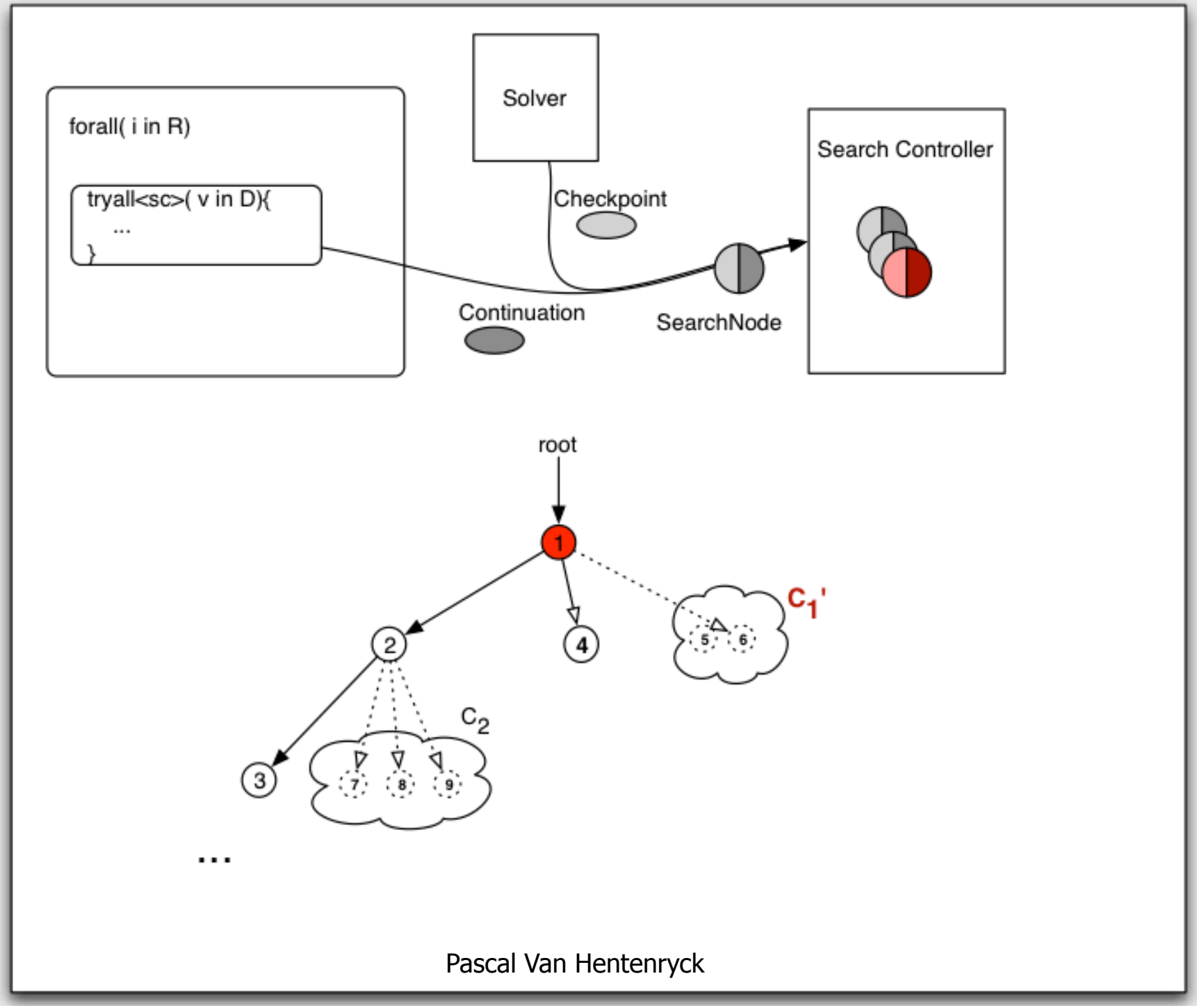

root
1
2
3
4
5
6
7
8
9
C1
C2
...

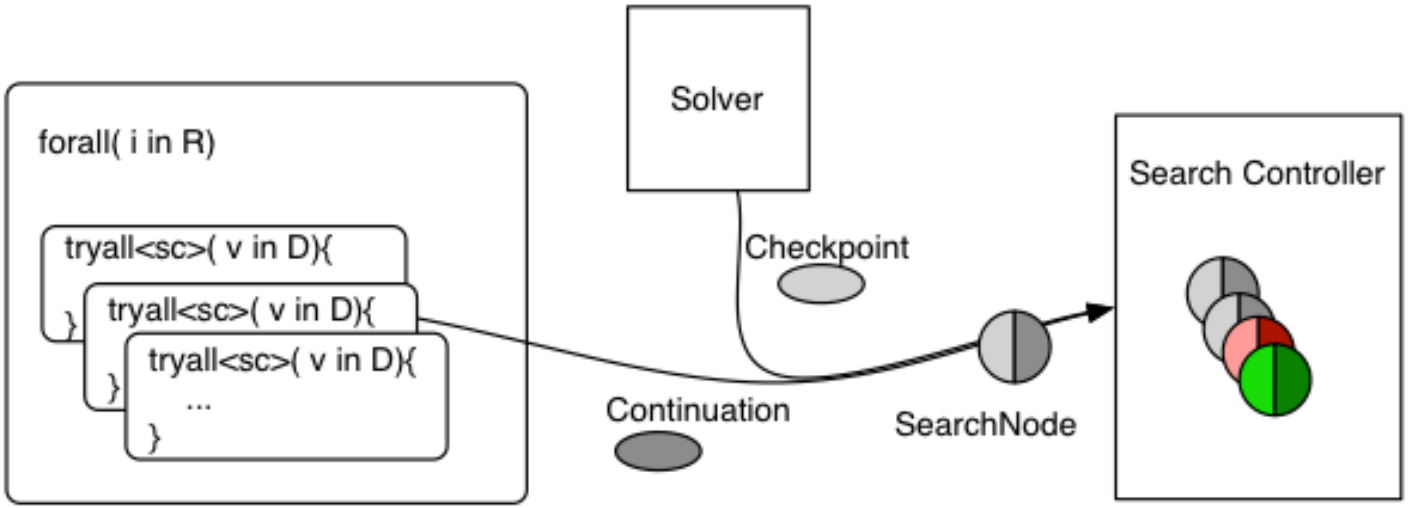
FAIL!

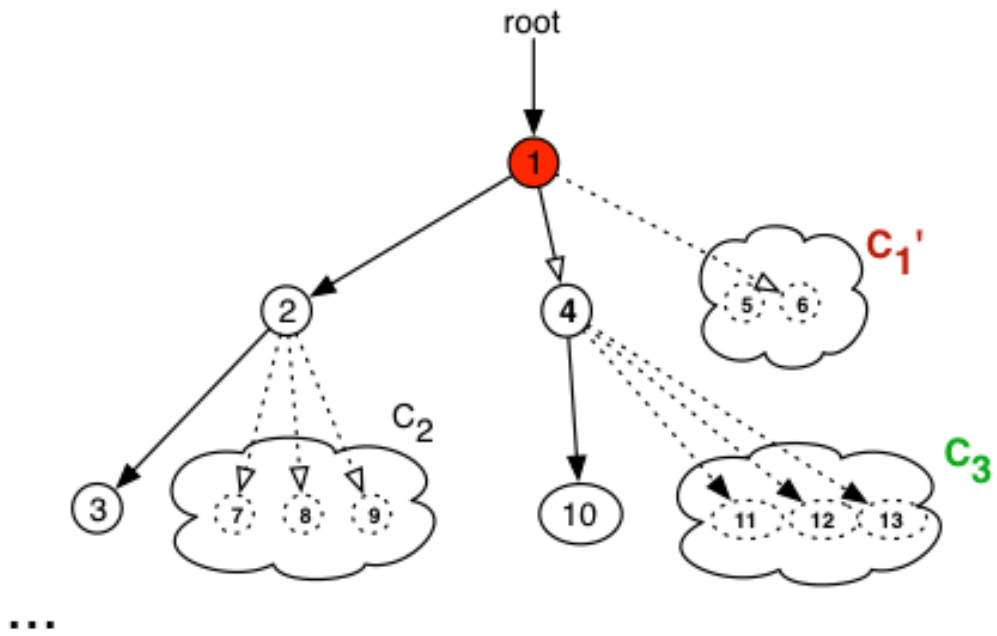

forall(i in R)
tryall<sc>(v in D){
tryall<sc>(v in D){
tryall<sc>(v in D){
...
}
Solver
Checkpoint
Continuation
SearchNode
Search Controller
Restore
root
1
2
3
4
5
6
7
8
9
C1
C2
...


forall(i in R)
tryall<sc>(v in D){
...
}
Solver
Checkpoint
Search Controller
Continuation
SearchNode
root
Restore
1
2
3
4
5
6
7
8
9
C1
C2
...


forall(i in R)
tryall<sc>(v in D){
...
}
Solver
Checkpoint
Search Controller
Continuation
SearchNode
root
1
2
3
4
5
6
7
8
9
C1'
C2
...


forall(i in R)
tryall<sc>(v in D){
...
}
Solver
Checkpoint
Search Controller
Continuation
SearchNode
root
1
2
3
4
5
6
7
8
9
C_1'
C_2
...


forall(i in R)
tryall<sc>(v in D){
tryall<sc>(v in D){
tryall<sc>(v in D){
...
}
Solver
Checkpoint
Continuation
SearchNode
Search Controller


root
1
2
3
4
5
6
7
8
9
10
11
12
13
C1'
C2
C3
...

```
class BFSearch implements SearchController {
    heap{ObjectiveValue -> Continuation} st;
    ObjectiveValue bestValue;
    Objective obj;

    void addChoice(Continuation c) {
      st.insert(obj.getValue,c);
    }
    void fail() {
      if (st.empty()) exit();
      if (st.getKeyMin().compare(bestValue) >= 0)
        exit();
      Continuation c = st.getDataMin();
      st.pop();
      call(c);
     }
     ...
}
```

```
class BFSearch implements SearchController {
    heap{ObjectiveValue -> Continuation} st;
    ObjectiveValue bestValue;
    Objective obj;
    ...
    void exitTry() {
      if (!st.empty()) {
         ObjectiveValue m = st.getKeyMin();
         ObjectiveValue cm = obj.getValue();
         if (cm.degrade(m,0.0)) {
            continuation f {
               st.insert(cm,f);
               fail();
            }
         }
      }
   }
}
```