Finding Parallel Regions with Temporal Planning

Claudio Scheer[1]

[1]Pontifical Catholic University of Rio Grande do Sul - PUCRS
claudio.scheer@edu.pucrs.br

Final Presentation, July 2020

Table of Contents

Table of Contents

Finding parallel regions

- It takes a lot of time;

Common approaches

Static analysis of the source code:

- loops detection;
- variable dependencies;
- identifying whether the arguments are read or written;

Table of Contents

Approach

- PDDL domain executes the instructions;
- PDDL problem defines the instructions dependency tree;
- Simultaneous Temporal Planner to find a temporal plan;

PDDL domain - assignment

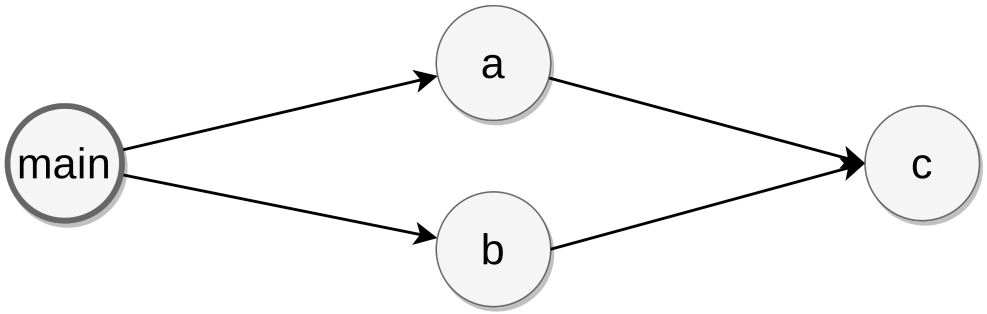
```
(:durative-action assignment
  :parameters (?instruction_id - id ?id - assignment)
  :duration (= ?duration 1)
  :condition (and
    (at start (assignment_id ?id ?instruction_id))
    (at start (not (executed_assignment ?id)))
    (at start (forall (?parent - id)
      (or
        (not (dependency_tree ?parent ?instruction_id))
        (executed_instruction ?parent)
      )
    ))
  )
  :effect (and
    (at end (executed_instruction ?instruction_id))
    (at end (executed_assignment ?id))
  )
)
```

PDDL domain - binary_operation

```
(:durative-action binary_operation
  :parameters (
    ?instruction_id - id ?idA - assignment
    ?idB - assignment ?operation_id - operation ?idC - assignment
  )
  :duration (= ?duration 1)
  :condition (and
    (at start (operation_id ?operation_id ?instruction_id))
    (at start (forall (?parent - id)
      (or
        (not (dependency_tree ?parent ?instruction_id))
        (executed_instruction ?parent)
      )
    ))
    (at start (not (executed_operation ?operation_id)))
    (at start (not (executed_binary_operation ?idA ?idB ?operation_id ?idC)))
    (at start (executed_assignment ?idA))
    (at start (executed_assignment ?idB))
  )
  :effect (and
    (at end (executed_instruction ?instruction_id))
    (at end (executed_operation ?operation_id))
    (at end (executed_binary_operation ?idA ?idB ?operation_id ?idC))
  )
)
```

PDDL problem 1

```
int main()
{
  int a = 3;
  int b = 3;
  int c = a + b;
  return 0;
}
```



PDDL problem 1

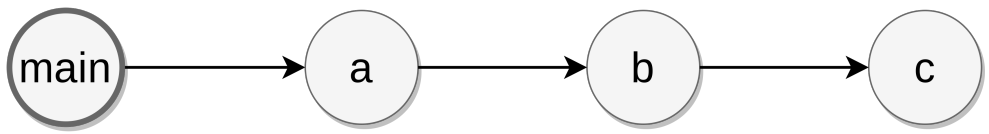
```
(:init
    (executed_instruction id0)

    (assignment_id assignmentA id1)
    (assignment_id assignmentB id2)
    (operation_id sumAB id3)
    (assignment_id assignmentC id4)

    (dependency_tree id0 id1)
    (dependency_tree id0 id2)
    (dependency_tree id1 id3)
    (dependency_tree id2 id3)
    (dependency_tree id3 id4)
)

(:goal (and
    (executed_assignment assignmentA)
    (executed_assignment assignmentB)
    (executed_binary_operation assignmentA assignmentB sumAB
        assignmentC)
    (executed_assignment assignmentC)
))
```

PDDL problem 1

0.000	1.000	2.000
assignmentA		
assignmentB		
	sumAB	
		assignmentC

PDDL problem 2

```
int main()
{
  int a = 3;
  int b = a + 1;
  int c = a + b;
  return 0;
}
```



PDDL problem 2

```
(:init
    (executed_instruction id0)

    (assignment_id assignmentA id1)
    (assignment_id assignmentB id2)
    (operation_id sumAB id3)
    (assignment_id assignmentC id4)

    (dependency_tree id0 id1)
    (dependency_tree id1 id2)
    (dependency_tree id1 id3)
    (dependency_tree id2 id3)
    (dependency_tree id3 id4)
)

(:goal (and
    (executed_assignment assignmentA)
    (executed_assignment assignmentB)
    (executed_binary_operation assignmentA assignmentB sumAB
        assignmentC)
    (executed_assignment assignmentC)
))
```

PDDL problem 2

0.000	1.000	2.000	3.000
assignmentA			
	assignmentB		
		sumAB	
			assignmentC

Table of Contents

How to handle `for` loops?

```
int main()
{
  int s = 0;
  std::vector<int> x = {1, 2, 3};
  for (int i = 0; i < x.size(); i++)
  {
    s += x[i];
  }
  return 0;
}
```

```
int main()
{
  int a[3] = {0};
  a[0] = rand();
  for (int i = 1; i < 3; ++i)
  {
    a[i] = a[i - 1] + rand();
  }
  return 0;
}
```

Table of Contents

Questions/Ideas

1. Is the compiler domain capable of handling fluent variables and predicates?

Questions/Ideas

1. Is the compiler domain capable of handling fluent variables and predicates?
2. Is the compiler domain capable of performing operations with strings?

Questions/Ideas

1. Is the compiler domain capable of handling fluent variables and predicates?
2. Is the compiler domain capable of performing operations with strings?
3. Which planners should I test the compiler domain on?

Questions/Ideas

1. Is the compiler domain capable of handling fluent variables and predicates?
2. Is the compiler domain capable of performing operations with strings?
3. Which planners should I test the compiler domain on?
4. How does a planner find a parallel region?

Questions/Ideas

1. Is the compiler domain capable of handling fluent variables and predicates?
2. Is the compiler domain capable of performing operations with strings?
3. Which planners should I test the compiler domain on?
4. How does a planner find a parallel region?
5. Can I set a weight for the planner to get regions that are really worth running in parallel?

Table of Contents

Conclusion

- This is not a conventional approach;

Conclusion

- This is not a conventional approach;
- If the results are positives, the approach may reduce the amount of time to find parallel regions.