Discussion 3: Algorithm V1,V2 & Experiment on Cases

June 2, 2020

Algorithm Version 1

Algorithm 1 Algorithm Version1: Learn Multiphase RF Incrementally

Require: Loop L, depthBound db

Ensure: `result` $\in \{$`FIN, INF, UNKNOWN`$\}$, a list of RFs `rf_list`

```
1: i := 0, result := UNKNOWN, rf_list := []
2: while i < db and result== UNKNOWN do
3:    result, rf = LearnRankerBounded(L)
4:    if result == INF or result == FIN then
5:       rf_list.append(rf)
6:       return result, rf_list
7:    else
8:       result, rf = LearnRankerNoBound(L)
9:       rf_list.append(rf)
10:   end if
11:   L = ConjunctConstraint(L, rf)
12:   i+=1
13: end while
14: return UNKNOWN, rf_list
```

Case 1: Multiphase

Example

`while(`x `>` `0` `or` y `>` `0)` `do` $x' = x + y - 1; y' = y - 1$
is a loop ranked by 2-phase multiphase ranking function: $\langle y, x \rangle$

Result of the algorithm: $\langle 0.9971y, 0.639x + 0.6774 \rangle$

```
------------------START INCREMENTAL LEARNING------------------
------------INCREASE TIMES: 0
--------LEARN BOUNDED
Failed to prove it is terminating

--------LEARN UNBOUNDED
Found Ranking Part: 0.9971 * x[1]^1

------------INCREASE TIMES: 1
--------LEARN BOUNDED
Found Ranking Function: 0.639 * x[0]^1 + 0.6774 * 1

------------------LEARNING MULTIPHASE SUMMARY-----------------
MULTIPHASE DEPTH: 2
LEARNING RESULT: FINITE
----------RANKING FUNCTIONS----------
0.9971 * x[1]^1
0.639 * x[0]^1 + 0.6774 * 1
```

Case 2: Multiphase plus Branch

Example

`while(`x `> 0 or` y `> 0) do`
`If y > 0:`$x' = x; y' = y - 1;$ `else:`$x' = x - 1; y' = y - 1;$

is a loop with 2-phase multiphase ranking function $\langle y, x \rangle$
Use template $ax + by + c$ to run:

```
--------------------LEARNING MULTIPHASE SUMMARY--------------------
MULTIPHASE DEPTH:  3
LEARNING RESULT:  FINITE
-----------RANKING FUNCTIONS----------
0.0045 *  x[0]^1 + 0.9955 *  x[1]^1
0.9901 *  x[1]^1
0.9846 *  x[0]^1 + 0.6667 * 1
```

where the learn unbound part first generated $x + y$ as a decreasing function, which is a spurious phase.

Way to solve the spurious decreasing function

Try different templates.
If we try $by + c$ as the first template and $ax + by + c$ as the second:

```
--------------------LEARNING MULTIPHASE SUMMARY-------------------
MULTIPHASE DEPTH:  2
LEARNING RESULT:  FINITE
-----------RANKING FUNCTIONS----------
1.0 *  x[1]^1
0.9804 *  x[0]^1 + 0.6667 * 1
```

We have to modify our algorithm to by adding back-tracking to try different templates when we fail to learn a ranking function out.

Case 3: Split Technique

Example

`while(`x `>` `0` `or` y `>` `0)` `do` $x' = x + y; y' = y - 1$

This is a false example for multiphase ranking function $\langle y, x \rangle$ for when $y < 0$

$$x - x' = -y > 0$$

not δ.

By setting the conjuncted formula to $y < -0.1$, $-y > 0.1$. The multiphase ranking function is $\langle y + 0.1, x \rangle$

Case 4: Not all cases can be solved

Example

`while(`$x \geq 1$ `and` $y \geq 1$ `and` $x \geq y$ `and` $2^B y \geq x$`) do`
$x' = 2x; y' = 3y$

According to "On Multiphase" paper, this loop has a multiphase ranking function:

$$\langle x - 2^B y, x - 2^{B-1} y, \ldots, x - y \rangle$$

We ran a experiment on $B = 2$. But failed to learn the RF out.

```
--------------------LEARNING MULTIPHASE SUMMARY-------------------
MULTIPHASE DEPTH:  5
LEARNING RESULT:  UNKNOWN
-----------RANKING FUNCTIONS----------
 - 0.1905 *  x[0]^1 - 0.3809 *  x[1]^1
 - 0.1905 *  x[0]^1 - 0.3809 *  x[1]^1
 - 0.1905 *  x[0]^1 - 0.3809 *  x[1]^1
 - 0.1905 *  x[0]^1 - 0.3809 *  x[1]^1
```

Reason: the first guess is incorrect.

Algorithm Version 2

Algorithm 2 Algorithm Version2: Learn Multiphase RF Backtracking

Require: Loop L, depthBound db, templateList `tpList`

Ensure: `result` $\in \{$`FIN, INF, UNKNOWN`$\}$, a list of RFs `rf_list`

1: $i = 1$
2: **while** $i \leq db$ and `result` $==$ `UNKNOWN` **do**
3: `rf_list = []`
4: `result, rf_list=train_backtracking_loopbody(`L`, rf_list, tpList, 0, 1,` i`)`
5: **if** `result != UNKNOWN` **then**
return result, rf_list
6: **end if**
7: **end while**

Algorithm Version 2 Continue

Algorithm 3 Algorithm Version2: Learn Multiphase RF Backtracking

1: `train_backtracking_loopbody`(L, `rf_list`, `tpList`, `tempId`, `currentDepth`, `maxDepth`):

Use dfs to learn multiphase ranking function, if for one layer the learning result is `UNKNOWN`, backtrack one depth and change the template.

Case 5: Jump In-and-Out

Example

`while(`x `>` `0 or` y `>` `0) do`
`If y > 0:`$x' = x; y' = y - 2;$ `else:`$x' = x - 1; y' = y + 1;$

- Not backtracking: cannot learn out
- Backtracking:

```
--------------------LEARNING MULTIPHASE SUMMARY------------------
MULTIPHASE DEPTH: 3
LEARNING RESULT: FINITE
-----------RANKING FUNCTIONS----------
1.4858 * x[0]^1 + 0.4977 * x[1]^1
0.9231 * x[0]^1
- 0.0491 * x[0]^1 + 0.4974 * x[1]^1 + 0.694 * 1
```

TODOs

- Implement the algorithm that extracts guards to be candidate ranking functions.
- Add to deal with nodeterministic loops.
- Find more cases and explore the advantage of our methods, and do some large scale experiments.