Disjoint Conditions

CS 536: Science of Programming, Spring 2022

A. Why

- Disjoint parallel programs ensure that no thread can interfere with the execution of another thread.
- Disjoint conditions ensure that no thread can interfere with the conditions of a triple.
- Disjoint parallel programs with disjoint conditions can be proved correct by combining the proofs of their individual threads.

B. Objectives

At the end of this work you should be able to

- Recognize disjoint parallel programs and correctness triples with disjoint conditions
- Use the rules for sequentialization and disjoint parallelism

C. Questions

1. Are the following programs parallel disjoint with disjoint conditions?
   - *{T} x := 1 ; y := 1 {x = 1}*
   - *{x = 0} z := 0 {x = z}*

i	*j*	*Change i*	*Vars j*	*Free j*	*i interfere w/ j?*	*Interfere w/ conds?*
1	2					
2	1					

2. Are the following programs parallel disjoint with disjoint conditions?
   - *{T} x := 1 ; y := 0 {x = 1}*
   - *{z = 0} z := z*x {z = 0}*

i	*j*	*Change i*	*Vars j*	*Free j*	*i interfere w/ j?*	*Interfere w/ conds?*
1	2					
2	1					

3. Are the following programs parallel disjoint with disjoint conditions?
   - $\{T\}$ *if* $x > 0$ *then* $y := 1; z := 2$ *fi* $\{x \leq 0 \rightarrow z = 2\}$
   - $\{T\}$ *if* $x \leq 0$ *then* $z := 2; y := 3$ *fi* $\{x \leq 0 \rightarrow y = 3\}$

i	*j*	*Change i*	*Vars j*	*Free j*	*i interfere w/ j?*	*Interfere w/ conds?*

4. Are the following programs parallel disjoint with disjoint conditions?
   - $\{T\}$ $x := u$; $y := u$ $\{x = y\}$
   - $\{z > 0\}$ $z := z–1$; $v := z$ $\{v = z\}$
   - $\{w \geq u\}$ $w := w+1$ $\{w > u\}$

i	*j*	*Change i*	*Vars j*	*Free j*	*i interfere w/ j?*	*Interfere w/ conds?*
1	2					
1	3					
2	1					
2	3					
3	1					
3	2					

5. Design a parallel program $\{p_1 \wedge p_2\}$ $[\{p_1\}$ s_1 $\{q_1\} \parallel \{p_2\}$ s_2 $\{q_2\}]$ $\{q_1 \wedge q_2\}$ where (1) the threads are not disjoint programs, (2) the threads don't have disjoint conditions, but (3) in fact, running the program in any state works correctly. There are any number of possible answers — it's easiest if you write the threads so that each one produces the same results whether you run it sequentially or in parallel with the other thread.

Solution to Practice 22

Feel free to abbreviate *{x, y}* to *x, y* or just *x y* (which is what I used below.)

1. No: Thread 1 interferes with the conditions of thread 2

i	*j*	*Change i*	*Vars j*	*Free j*	*i interfere w/ j?*	*Interfere w/ conds?*
1	2	*x y*	*z*	*x z*	*No*	*Yes (because of) x*
2	1	*z*	*x y*	*x*	*No*	*No*

2. No: Thread 1 interferes with the program of thread 2.

i	*j*	*Change i*	*Vars j*	*Free j*	*i interfere w/ j?*	*Interfere w/ conds?*
1	*2*	*x y*	*x z*	*z*	*Yes (because of) x*	*No*
2	*1*	*z*	*x y*	*x*	*No*	*No*

3. No: Each interferes with the other's programs and conditions.

i	*j*	*Change i*	*Vars j*	*Free j*	*i interfere w/ j?*	*Interfere w/ conds?*
1	2	*y z*	*x y z*	*x y*	*Yes: y, z*	*Yes: y*
2	1	*y z*	*x y z*	*x z*	*Yes: y, z*	*Yes: z*

4. Yes, these are parallel disjoint with disjoint conditions

i	*j*	*Change i*	*Vars j*	*Free j*	*i interfere w/ j?*	*Interfere w/ conds?*
1	2	*x y*	*v z*	*v z*	*No*	*No*
1	3	*x y*	*w*	*u w*	*No*	*No*
2	1	*v z*	*u x y*	*x y*	*No*	*No*
2	3	*v z*	*w*	*u w*	*No*	*No*
3	1	*w*	*u x y*	*x y*	*No*	*No*
3	2	*w*	*v z*	*v z*	*No*	*No*

5. It helps if the threads change the state at opposite times. Here's one solution:

{T}
[{T} if x > 0 then {y := 0 } {x > 0 → y = 0)
|| {T} if x ≤ 0 then { y := 0 } {x ≤ 0 → y = 0}]
{(x > 0 → y = 0) ∧ (x ≤ 0 → y = 0)}
{y = 0}

More generally, if neither s_1 nor s_2 modify x, then the outline below is correct.

{T}
[{T} if x > 0 then {s_1} {x > 0 → sp(x > 0, s_1)}
|| {T} if x ≤ 0 then {s_2 }{x ≤ 0 → sp(x ≤ 0, s_2)}]
{(x > 0 → sp(x > 0, s_1)) ∧ (x ≤ 0 → sp(x ≤ 0, s_2))}