Basics of Parallel Programs

CS 536: Science of Programming, Spring 2022

A. *Why*

- Parallel programs are more flexible than sequential programs but their execution is more complicated.
- Parallel programs are harder to reason about because parts of a parallel program can interfere with other parts.

B. *Objectives*

At the end of this work you should be able to

- Draw evaluation graphs for parallel programs.

C. *Problems*

In general, for the problems below, if it helps you with the writing, feel free to define other symbols. ("Let *s* ≡ *some program*," for example.)

1. What is the sequential nondeterministic program that corresponds to the program $[x := v \mid\mid y := v+2 \mid\mid z := v*2]$.

2. Let configuration $C \equiv \langle s, \sigma\rangle$ *where* $s \equiv [x := 1 \mid\mid x := -1]; y := y+x$.
   a. What is the sequential nondeterministic program that corresponds to *s?*
   b. Draw an evaluation graph for C_2.

3. Repeat Problem 2 on $C_3 \equiv \langle\ S_3, \sigma[v \mapsto 0]\ \rangle$ where $S_3 \equiv [x := v+3;\ v := v*4 \,\|\, v := v+2]$. Note that in the first thread, the two assignments must be done with *x* first, then *v*. Because adding 3 and adding 2 are commutative, two of the (normally-different) nodes will merge.

4. Repeat Problem 2 on $C_4 \equiv \langle S4, \sigma[v \mapsto \delta]\rangle$ where $S_4 \equiv [v := v*\gamma;\ v := v+\beta \,\|\, v := v+\alpha\,]$. This problem is similar to Problem 3 but is symbolic, and the commutative plus operator has been moved, so the shape of the graph will be different from Problem 3.

5. Let $C_5 \equiv \langle W, \sigma\rangle$ where $W \equiv$ *while* $x \le n$ $\{[x := x+1 \; || \; y := y*2] \}$ and let σ of x, y, and z be *0, 1,* and *2* respectively. Note the parallel construct is in the body of the loop.
   a. Draw an evaluation graph for C_5. (Feel free to to say something like "Let $T \equiv \ldots$" for the loop body, to cut down on the writing.
   b. Draw another evaluation graph for C_5, but this time, use the $\rightarrow^3$ notation to get a straight line graph. Concentrate on the configurations of the form $\langle\; W, \ldots \;\rangle$.

6. In $[s_1 \,\|\, s_2 \,\|\, \ldots \,\|\, s_n]$ can any of the threads $s_1, s_2, \ldots, s_n$ contain parallel statements? Can parallel statements be embedded within loops or conditionals?

7. Say we know $\{p_1\}\ s_1\ \{q_1\}$ and $\{p_2\}\ s_2\ \{q_2\}$ under partial or total correctness.
   a. In general, do we know how $\{p_1 \wedge p_2\}\ [s_1 \,\|\, s_2]\ \{q_1 \wedge q_2\}$ will execute? Explain briefly.
   b. What if $p_1 \equiv p_2$? I.e., if we know $\{p\}\ s_1\ \{q_1\}$ and $\{p\}\ s_2\ \{q_2\}$, then do we know how $\{p\}\ [s_1 \,\|\, s_2]\ \{q_1 \wedge q_2\}$ will work?
   c. What if in addition, $q_1 \equiv q_2$? I.e., If we know $\{p\}\ s_1\ \{q\}$ and $\{p\}\ s_2\ \{q\}$, do we know how
      $\{p\}\ [s_1 \,\|\, s_2]\ \{q\}$ will work? (This problem is harder)
   d. For parts (a) – (c), does it make a difference if we use $\vee$ instead of $\wedge$?

Solution to Practice 22

Class 22: Basics of Parallel Programs

1. Sequential nondeterministic equivalent of [*x* := *v* ‖ *y* := *v*+2 ‖ *z* := *v**2]:

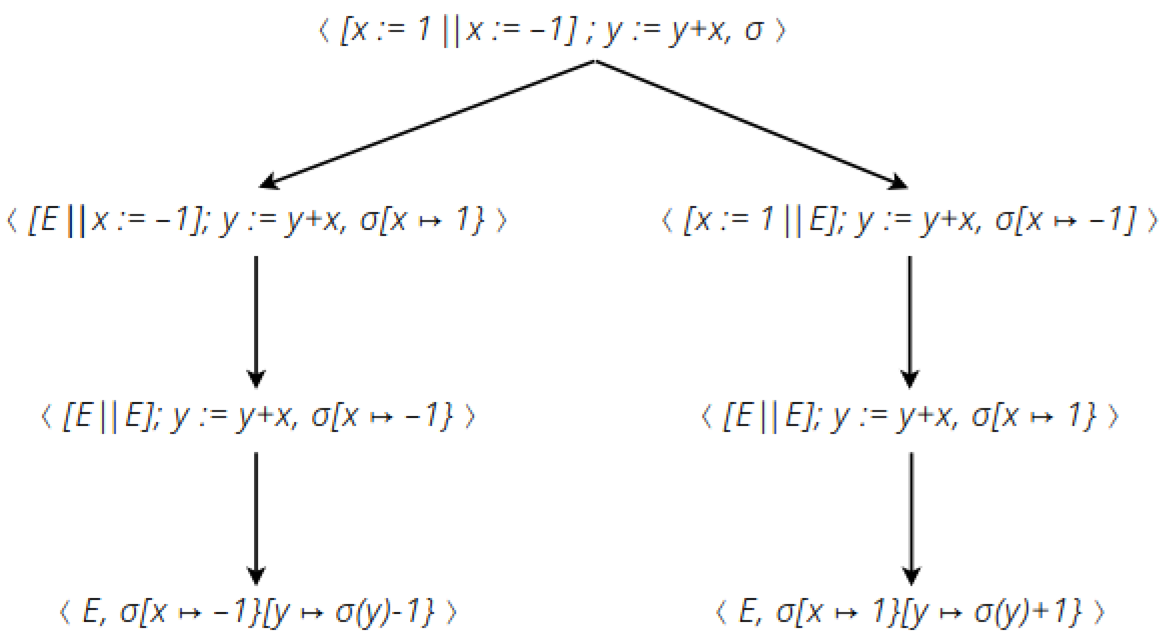
    *branch { T → x := v; y := v+2; z := v*2*
    *☐ T → x := v; z := v*2; y := v+2*
    *☐ T → y := v+2; x := v; z := v*2*
    *☐ T → y := v+2; z := v*2; x := v*
    *☐ T → z := v*2; x := v; y := v+2*
    *☐ T → z := v*2; y := v+2; x := v*
    }

2. (Program *[x := 1 || x := –1] ; y := y+x]*)
    a. Equivalent sequential nondeterministic program

        branch { T → x := 1; x := –1 ☐ T → x := –1; x := 1 }

    b. Evaluation graph for ⟨ [*x* := *1* || *x* := *–1*]; *y* := *y*+*x*, σ ⟩

Note: This uses Prof. Sasaki's notation, which uses E instead of skip.



3. (Program *[v := v+3; v := v*4‖ v := v+2]*)

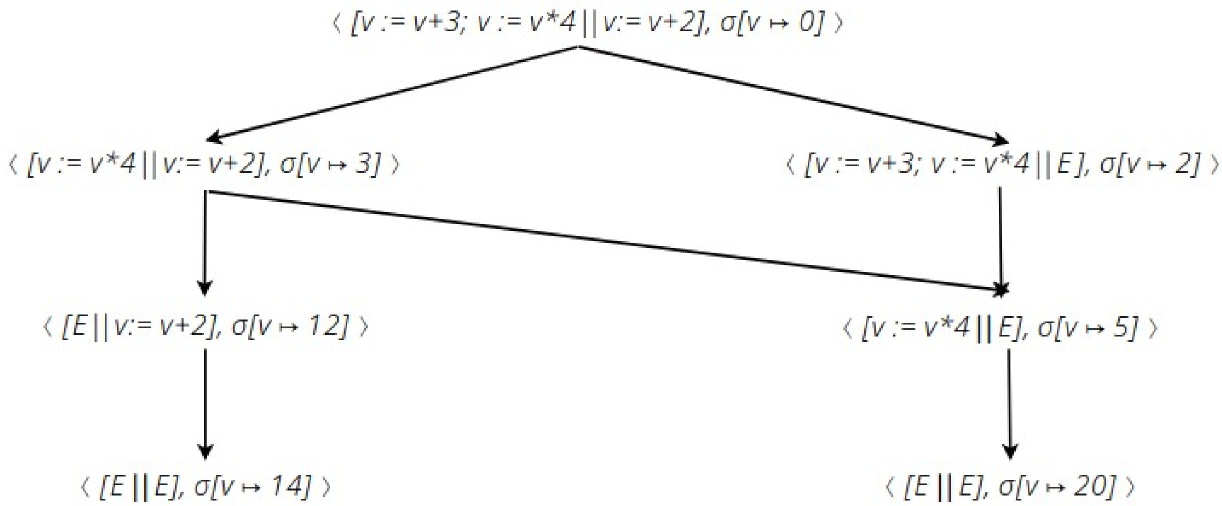
a. Equivalent sequential nondeterministic program

*branch { T → v := v+3; branch { T → v := v*4; v := v+2* ☐ T → *v := v+2; v := v*4 }*

☐ *T → v := v+2; v := v+3; v := v*4*

}

b. Evaluation graph for ⟨ *[v := v+3; v := v*4|| v := v+2]*, σ*[v ↦ 0]* ⟩. Note that two of the execution paths happen to merge, so there are only two final states instead of three.



4. (Program *[v := v*γ; v := v+β‖ v:= v+α]*).

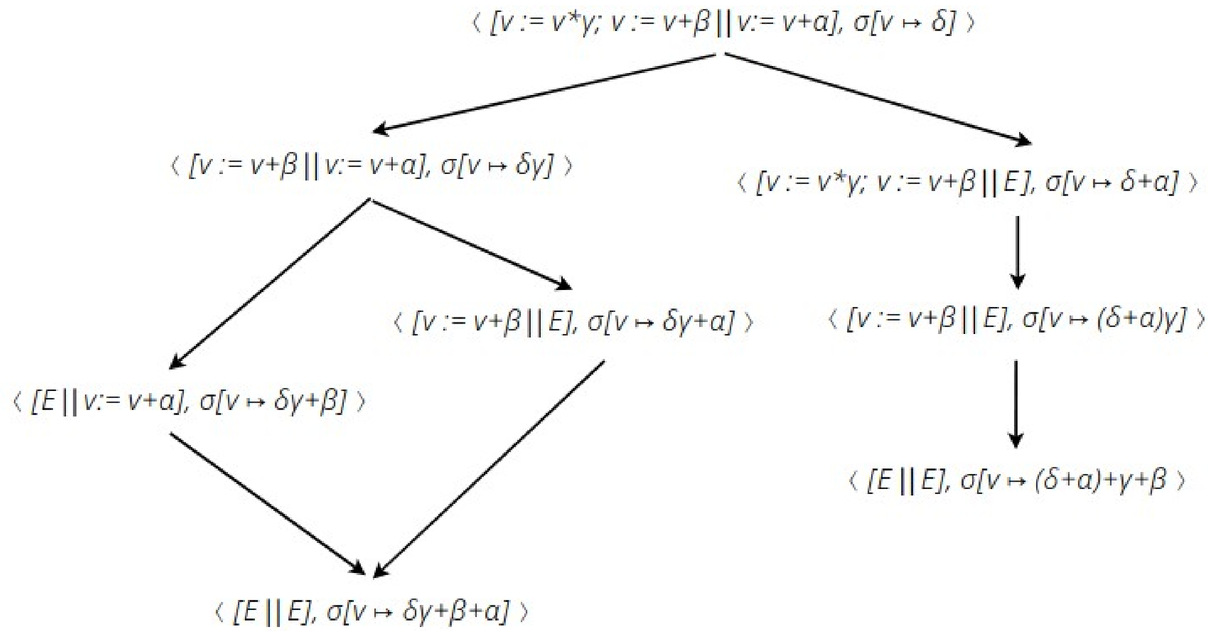
a. Equivalent sequential nondeterministic program

branch { T → v := vγ; branch { T → v := v+β; v := v+α* ☐ T → *v := v+α; v := v+β }*

☐ *T → v := v+α; v := v*γ; v := v+β*

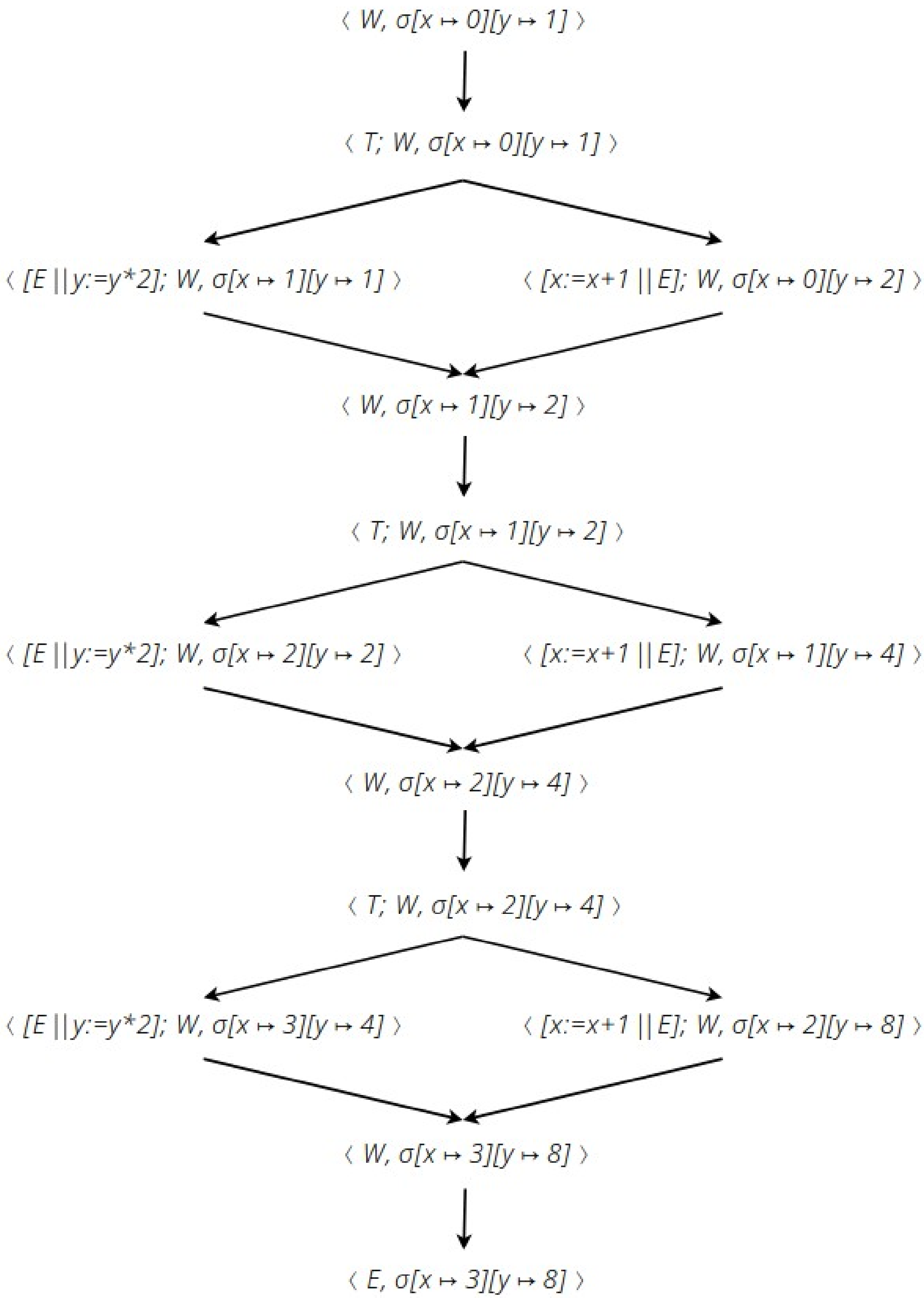
}

b. Evaluation graph for ⟨ *[v := v*γ; v := v+β‖ v := v+2]*, σ*[v ↦ δ]* ⟩



5.

a. A full evaluation graph. Just to be explicit, I wrote *σ[x ↦ 0][y ↦ 1]* below but just σ is fine.


⟨ W, σ[x ↦ 0][y ↦ 1] ⟩
⟨ T; W, σ[x ↦ 0][y ↦ 1] ⟩
⟨ [E || y:=y*2]; W, σ[x ↦ 1][y ↦ 1] ⟩
⟨ [x:=x+1 || E]; W, σ[x ↦ 0][y ↦ 2] ⟩
⟨ W, σ[x ↦ 1][y ↦ 2] ⟩
⟨ T; W, σ[x ↦ 1][y ↦ 2] ⟩
⟨ [E || y:=y*2]; W, σ[x ↦ 2][y ↦ 2] ⟩
⟨ [x:=x+1 || E]; W, σ[x ↦ 1][y ↦ 4] ⟩
⟨ W, σ[x ↦ 2][y ↦ 4] ⟩
⟨ T; W, σ[x ↦ 2][y ↦ 4] ⟩
⟨ [E || y:=y*2]; W, σ[x ↦ 3][y ↦ 4] ⟩
⟨ [x:=x+1 || E]; W, σ[x ↦ 2][y ↦ 8] ⟩
⟨ W, σ[x ↦ 3][y ↦ 8] ⟩
⟨ E, σ[x ↦ 3][y ↦ 8] ⟩

b. Evaluation graph abbreviated using $\rightarrow^3$ notation:

$\langle\ W, \sigma[x \mapsto 0][y \mapsto 1]\ \rangle \rightarrow^3 \langle\ W, \sigma[x \mapsto 1][y \mapsto 2]\ \rangle \rightarrow^3 \langle\ W, \sigma[x \mapsto 2][y \mapsto 4]\ \rangle$
$\rightarrow^3 \langle\ W, \sigma[x \mapsto 3][y \mapsto 8]\ \rangle \rightarrow \langle\ skip, \sigma[x \mapsto 3][y \mapsto 8]\ \rangle$

6. No, in $[s_1 \,\|\, s_2 \| \ldots \|\, s_n]$ the threads cannot contain parallel statements, but yes, parallel statements can be embedded within loops and conditionals.

7. In general, even if $\{p_1\}\ s_1\ \{q_1\}$ and $\{p_2\}\ s_2\ \{q_2\}$ are both valid sequentially, we can't compose them in parallel, even if $p_1 \equiv p_2$ and $q_1 \equiv q_2$. An example is how $\{x > 0\}\ x := x\text{-}1\ \{x \geq 0\}$ is valid but $\{x > 0\}\ [x := x\text{-}1 || x := x–1]\ \{x \geq 0\}$ is not. The first $x := x–1$ to execute ends with $x \geq 0$, which is too weak for the second $x := x–1$ to work correctly.