**Disjoint Conditions**

*CS 536: Science of Programming, Spring 2021*

**A. Why?**

- Reducing the amount of interference between threads lets us reason about parallel programs by combining the proofs of the individual threads.
- 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**

After this class, you should know

- What disjoint conditions are.
- What the disjoint parallelism rule for disjoint parallel programs with disjoint conditions allows.

**C. Disjoint Parallelism Rule for DPPs?**

- The sequentialization proof rule for DPPs lets us reason about DPPs, which is nice, but it to use it, we have to develop many intermediate conditions: To prove \( \{ p \} S_1 ||| \ldots ||| S_n \{ q \} \), we need to prove \( \{ p \} S_1 ; \ldots ; S_n \{ q \} \), which (if we use \( \text{wp} \)) means finding a sequence of preconditions \( q_1, \ldots, q_n \) and proving
  \[
  \{ p \} \{ q_n \} S_1 ; \{ q_{n-1} \} S_2 ; \ldots ; \{ q_1 \} S_n \{ q \}
  \]

The proofs of \( q_1, q_2, \ldots, q_n \) can get increasingly complicated because each \( q_i \) can depend on all the threads and conditions to its right.
- Ideally, we'd like to prove correctness of the individual threads and then combine them to get correctness of the parallel program. I.e., we'd like something that lets us take sequential thread triples, combine their preconditions, run them in parallel, and conclude the conjunction of their postconditions

**Example 1:** As a proof rule application, we'd like something like

1. \( \{ x \geq 0 \} z := x \{ z \geq 0 \} \)
2. \( \{ y \leq 0 \} w := -y \{ w \geq 0 \} \)
3. \( \{ x \geq 0 \land y \leq 0 \} \{ z := x \mid w := -y \} \{ z \geq 0 \land w \geq 0 \} \) by ???

As a full proof outline, we would have

\[
\{ x \geq 0 \land y \leq 0 \}
\mid \{ x \geq 0 \} z := x \{ z \geq 0 \}
\mid \mid \{ y \leq 0 \} w := -y \{ w \geq 0 \}
\mid \{ z \geq 0 \land w \geq 0 \}
\]
• But we must be careful — this combination doesn't always work.

• **Example 2**: In the (invalid) proof outline below, we can't combine the $x = 1$ and $x = y$ postconditions because the first thread invalidates the $x = 0$ precondition that the second thread relies on.

$$\{x = 0\} \quad \{x = 0\} \ x := 1 \quad \{x = 1\}$$

$$\| \quad \{x = 0\} \ y := 0 \quad \{x = y\}$$

$$\} \ {x = 1} \land x = y$$  ← **Bad! Can't combine the postconditions!**

$$\{x = y = 1\}$$

• Even though the threads of the DPP can't affect each others runtime states, they can affect variables that appear in the conditions other threads. We need an additional restriction on our programs.

• **Definition**: \(\{p_1\} S_1 \{q_1\}\) and \(\{p_2\} S_2 \{q_2\}\) have **disjoint conditions** if neither program can affect the other's conditions: \(\text{Change}(S_1) \cap \text{Free}(p_2, q_2) = \emptyset\) and \(\text{Change}(S_2) \cap \text{Free}(p_1, q_1) = \emptyset\).

• **Example 3**: Some disjoint conditions:

• \(\{x \geq 0\} \ z := x \ {z \geq 0}\) and \(\{y \leq 0\} \ w := -y \ {w \geq 0}\), since \(\{z\} \cap \{w, y\} = \emptyset\) and \(\{w\} \cap \{x, z\} = \emptyset\).

• \(\{z = 0\} \ x := z + 1 \ {x \leq z}\) and \(\{z = 0\} \ y := z \ {z \geq y}\), since \(\{x\} \cap \{y, z\} = \{y\} \cap \{x, z\} = \emptyset\).

• **Example 4**: Some nondisjoint conditions:

• \(\{x = 0\} \ x := 1 \ {x = 1}\) and \(\{x = 0\} \ y := 0 \ {x = y}\), since \(\{x\} \cap \{x, y\} = \{x\}\) (the first thread interferes with the conditions of the second thread). Note that thread 2 doesn't interfere with thread 1, since \(\{y\} \cap \{x\} = \emptyset\).

• \(\{x \geq y\} \ x := x + 1 \ {x > y}\) and \(\{y = z\} \ y := y^*2 \ {y \geq z}\), since \(\{y, z\} \cap \{x, y\} = \{y\}\) (the second thread interferes with the conditions of the first thread). Note thread 1 doesn't interfere with thread 2.

• If we have a variable \(y_0\) that holds the value of \(y\) before running the second thread, then we can get disjoint conditions:

• \(\{x \geq y_0\} \ x := x + 1 \ {x > y_0}\)

• \(\{y = y_0 \land y = z\} \ y := y^*2 \ {y \geq z}\), \(z := z^*2 \ {y = z \land y = y_0 + 1}\).

• If two triples have disjoint programs and conditions, then neither can modify information used by the programs or conditions of the other. E.g., take the threads

• \(\{x = z\} \ x := x + 2; \ x := x \cdot 3 \ {x = 3z + 6}\)

• \(\{y^*2 > z \geq 1\} \ y := y^*2 \ {y > z \geq 1}\)

• The first thread changes \(x\), uses \(x\) in its program and uses \(x\) and \(z\) in its conditions. The second thread changes \(y\), uses \(y\) in its program, and uses \(y\) and \(z\) in its conditions. Therefore the threads have disjoint programs and disjoint conditions. No matter how we interleave execution, the first thread's changes to \(x\) will not affect \(y\) or \(z\), and the second thread's changes to \(y\) will not affect \(x\) or \(z\).
Disjoint Parallelism Rule (the parallelism rule for disjoint programs with disjoint conditions)

1. \(\{p_1\} S_1 \{q_1\}\)
2. \(\{p_2\} S_2 \{q_2\}\)
...
\(n\) \(\{p_n\} S_n \{q_n\}\)
\(n+1\) \(\{p_1 \land p_2 \land \ldots \land p_n\}\)
\([S_1 \parallel \ldots \parallel S_n]\)
\(\{q_1 \land q_2 \land \ldots \land q_n\}\)

Disjoint Parallelism, 1, 2, ..., \(n\)

where triples 1, 2, ..., \(n\) are pairwise disjoint programs with disjoint conditions.

• The program \(\{a := x + 1 \parallel b := x + 2\}\) can use disjoint parallelism, since the threads are disjoint parallel with disjoint conditions.

\[
\begin{align*}
\{T\} \\
\quad \text{[}\{T\} a := x + 1 \{a = x + 1\}\]}
\quad \text{||}\{T\} b := x + 2 \{b = x + 2\}\]
\quad \{a = x + 1 \land b = x + 2\}
\quad \{a + 1 = b\}
\end{align*}
\]

• Example 1 revisited: The triple from Example 1, \(\{x \geq 0 \land y \leq 0\} \{z := x \parallel w := -y\} \{z \geq 0 \land w \geq 0\}\) can also use disjoint parallelism.

\[
\begin{align*}
\{x \geq 0 \land y \leq 0\} \\
\quad \text{[}\{x \geq 0\} z := x \{z \geq 0\}\]}
\quad \text{||}\{y \leq 0\} w := -y \{w \geq 0\}\]
\quad \{z \geq 0 \land w \geq 0\}
\end{align*}
\]

• Example 2, revisited: The triple from Example 2, \(\{x = 0\} \{x := 1 \parallel y := 0\} \{x = 1 \land x = y\}\), has disjoint parallel threads but not disjoint conditions (thread 1 modifies \(x\), which appears in the conditions of thread 2).

\[
\begin{align*}
\{x = 0\} \\
\quad \text{[}\{x = 0\} x := 1 \{x = 1\}\]}
\quad \text{||}\{x = 0\} y := 0 \{x = y = 0\} \quad \quad \quad // \text{Bad: } x = 0 \text{ in conditions and } x := 1 \text{ in } S_1,
\quad \text{||}\{x = 1 \land x = y = 0\} \quad \quad \quad // \text{Can't use disjoint parallelism}
\quad \{x = y = 1 = 0\}
\end{align*}
\]

• Thread 1’s program changes variables in the conditions of thread 2, so we can fix the problem by changing the program or the conditions or both. However, merely introducing \(x_0 = 0\) as the starting value of \(x\) isn’t sufficient:

\[
\begin{align*}
\{x = 0 \land x_0 = 0\} \\
\quad \text{[}\{x_0 = 0 \land x = 0\} x := 1 \{x = 1\}\]}
\quad \text{||}\{x_0 = 0 \land x = 0\} y := 0 \{x_0 = 0 \land y = 0\} \quad \quad \quad // \text{Bad: } x = 0 \text{ still in precondition}
\quad \text{||}\{x = 1 \land x_0 = 0 \land y = 0\} \quad \quad \quad // \text{Can't use disjoint parallelism}
\quad \{x = 1 \land x_0 = 0 \land y = 0\}
\end{align*}
\]
Thread 1’s \( x := 1 \) still interferes with the \( x \) in the conditions of thread 2, so we need to remove it.

\[
\{ x = 0 \land x = x_0 \} \quad \text{// Only need } x_0 \text{ if we to use } sp
\]

\[
\{ x = 0 \land x = x_0 \} \ x := 1 \quad \{ x_0 = 0 \land x = 1 \}
\]

\[
|| \{ T \} \ y := 0 \quad \{ y = 0 \} \quad \text{// no info about } x \text{ in precondition or postcondition}
\]

\[
\{ x_0 = 0 \land x = 1 \land y = 0 \} \quad \text{// Can use disjoint parallelism}
\]

\[
\{ x = 1 \land y = 0 \}
\]

• Compressing the outline lets us see the overall structure of the program:

\[
\{ x = 0 \} \ { x = 0 \land x = x_0 \} \ { x := 1 \land y := 0 \} \ { x_0 = 0 \land x = 1 \land y = 0 \} \ { x = 1 \land y = 0 \}
\]
Disjoint Conditions

CS 536: Science of Programming, Spring 2021

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.
- Reducing the amount of interference between threads lets us reason about parallel programs by combining the proofs of the individual threads.
- 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

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

C. Questions

Class 24: Disjoint Conditions

To figure out whether a set of sequential threads is disjoint parallel and/or has disjoint conditions, we have to compare the changed, used, and free variables for various combinations of triples. In the tables below, the first two columns specify the pair of threads $i$ and $j$ we’re discussing, the next three columns specify the variables changed by thread $i$, used by thread $j$, and free in the conditions of thread $j$. The last two columns answer the questions “Does thread $i$ interfere with the program of thread $j$?” and “Does thread $i$ interfere with the conditions of thread $j$?”
1. Are the following programs parallel disjoint with disjoint conditions?
   - \( \{T\} \; x := 1 ; y := 1 \; \{x = 1\} \)
   - \( \{x = 0\} \; z := 0 \; \{x = z\} \)

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>Change i</th>
<th>Vars j</th>
<th>Free j</th>
<th>Disjoint Program?</th>
<th>Disjoint Cond?</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>Change i</th>
<th>Vars j</th>
<th>Free j</th>
<th>Disjoint Program?</th>
<th>Disjoint Cond?</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>Change i</th>
<th>Vars j</th>
<th>Free j</th>
<th>Disjoint Program?</th>
<th>Disjoint Cond?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Solution to Practice 24

### Class 24: Disjoint Conditions

(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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Change i</th>
<th>Vars j</th>
<th>Free j</th>
<th>Disjoint Program?</th>
<th>Disjoint Cond?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>x y</td>
<td>z</td>
<td>x z</td>
<td>Yes</td>
<td>No (because of) x</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>z</td>
<td>x y</td>
<td>x</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Change i</th>
<th>Vars j</th>
<th>Free j</th>
<th>Disjoint Program?</th>
<th>Disjoint Cond?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>x y</td>
<td>x z</td>
<td>z</td>
<td>No (because of) x</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>z</td>
<td>x y</td>
<td>x</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Change i</th>
<th>Vars j</th>
<th>Free j</th>
<th>Disjoint Program?</th>
<th>Disjoint Cond?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>y z</td>
<td>x y z</td>
<td>x y</td>
<td>No: y, z</td>
<td>No: y</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>y z</td>
<td>x y z</td>
<td>x z</td>
<td>No: y, z</td>
<td>No: z</td>
</tr>
</tbody>
</table>

4. Yes, these are parallel disjoint with disjoint conditions.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Change i</th>
<th>Vars j</th>
<th>Free j</th>
<th>Disjoint Program?</th>
<th>Disjoint Cond?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>x y</td>
<td>v z</td>
<td>v z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>x y</td>
<td>w</td>
<td>u w</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>v z</td>
<td>u x y</td>
<td>x y</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>v z</td>
<td>w</td>
<td>u w</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>w</td>
<td>u x y</td>
<td>x y</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>w</td>
<td>v z</td>
<td>v z</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>