Interference Freedom

- In the previous lecture, we discussed how to decide whether a parallel program is disjoint, and we discussed how to decide whether threads have disjoint condition. We learned that is a program is disjoint parallel then it has confluence, and we can prove its correctness using the sequentialization rule; and if threads in a disjoint parallel program also have disjoint condition, then we can use the parallelism rule to prove its correctness.
- This class, we focus on the situations that threads are not disjoint with each other. We start with the following example.

1. Draw the evaluation graph for $\langle [x \coloneqq x+1;\ x \coloneqq x*2 \parallel x \coloneqq x+3], \{x=10\}\rangle$.

$$\begin{tikzcd}
 & \langle [x \coloneqq x+1;\ x \coloneqq x*2 \parallel x \coloneqq x+3], \{x=10\}\rangle \arrow[dl] \arrow[dr] & \\
\langle [x \coloneqq x*2 \parallel x \coloneqq x+3], \{x=11\}\rangle \arrow[d] \arrow[drr] & & \langle [x \coloneqq x+1;\ x \coloneqq x*2 \parallel E], \{x=13\}\rangle \arrow[d] \\
\langle [E \parallel x \coloneqq x+3], \{x=22\}\rangle \arrow[d] & & \langle [x \coloneqq x*2 \parallel E], \{x=14\}\rangle \arrow[d] \\
\langle [E \parallel E], \{x=25\}\rangle & & \langle [E \parallel E], \{x=28\}\rangle
\end{tikzcd}$$

- We have some observations in the above example.
    - Although both threads write the same variable, there exists some diamond shape in the graph. Which implies that, even if threads are not disjoint with each other, their execution might not matter under some circumstances.
    - We can **interleave** a thread: we can execute one thread for some steps then execute another thread for some steps.

- With shared variables and interleaving, threads can have **critical section problems**. Let's use an example to explain this.

2. Let $S \equiv [x \coloneqq y+y \parallel x \coloneqq y; x \coloneqq x+y]$ and we execute S in state $\{y=a\}$, we need postcondition $x=2a$. Is there any race condition?
    - We can get the following evaluation graph:

$$\begin{tikzcd}
 & & \langle [x \coloneqq y+y \parallel x \coloneqq y; x \coloneqq x+y], \{y=a\}\rangle \arrow[dl] \arrow[dr] & & \\
 & \langle [E \parallel x \coloneqq y; x \coloneqq x+y], \{y=a, x=2a\}\rangle \arrow[dl] & & \langle [x \coloneqq y+y \parallel x \coloneqq x+y], \{y=a, x=a\}\rangle \arrow[dl] \arrow[dr] & \\
\langle [E \parallel x \coloneqq x+y], \{y=a, x=a\}\rangle \arrow[dr] & & \langle [x \coloneqq y+y \parallel E], \{y=a, x=2a\}\rangle \arrow[dl] & & \langle [E \parallel x \coloneqq x+y], \{y=a, x=2a\}\rangle \arrow[d] \\
 & \langle [E \parallel E], \{y=a, x=2a\}\rangle & & & \langle [E \parallel E], \{y=a, x=3a\}\rangle
\end{tikzcd}$$

- We can see that $M(S, \{y = a\}) = \big\{\{y = a, x = 2a\}, \{y = a, x = 3a\}\big\}$, and $x = 3a$ conflicts with the postcondition we need, so it is a race condition.
- If we look at these two threads separately, we find that $x \coloneqq y + y$ and $x \coloneqq y; x \coloneqq x + y$ can both give us $x = 2a$ (since $y = a$ in the starting state). So, this race condition comes from interleaving the second thread.
- Informally, we define that a **critical section** is a part of the thread, which uses shared memory (either read or write), that must execute atomically to get the expected condition. The critical section problem comes from interleaving a critical section.

- To avoid critical section problems,
  - We control where interleaving is allowed to happen using **atomic regions**.
  - We ensure that when interleaving occurs, it causes no harm (those threads are **interference-free**).

- If S is a statement, then $< S >$ is an **atomic region** statement with body S. The evaluation of S is considered as one step: If $\langle S, \sigma\rangle \rightarrow^* \langle E, \tau\rangle$ then $\langle < S >, \sigma\rangle \rightarrow \langle E, \tau\rangle$.
- Using atomic regions gives us control over the size of pieces of code that can be interleaved. However, making more or larger atomic sections is no panacea: The more or larger atomic regions code has, the less interleaving and hence parallelism we have.

- For different types of statement S, can we interleave S without creating a problem?
  - A single assignment statement or a single $\mathbf{skip}$ statement is atomic itself.
  - We have seen that interleaving a sequential statement can cause critical section problems.
  - For $\mathbf{if} - \mathbf{else}$ and $\mathbf{while}$ statements, interleaving the execution of loop body or the execution of a branch can be harmful. Also, although evaluation of a Boolean expression is atomic, interleaving can occur between inspection of the result and the jump to the next configuration.
    - For example, if $\sigma(B) = T$, then $\langle \mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{fi},\ \sigma\rangle \rightarrow \langle S_1,\ \sigma\rangle$. Thus, in parallel, another statement can be executed between the if test and the start of the true branch. If statement U changes σ to τ, then we can have $\langle [\mathbf{if}\ B\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2\ \mathbf{fi} \parallel\ < U >], \sigma\rangle \rightarrow \langle [S_1 \parallel < U >], \sigma\ \rangle \rightarrow \langle [\ S_1 \parallel\ E\ ], \tau\rangle$.
    - In this last configuration, we're about to execute S_1 not in σ, but in τ. There's no guarantee that B is still true when the true branch begins executing so we cannot use conditional rules to prove its correctness.

- We have learned that a thread can interfere with another thread or another thread's (pre- and post-) condition. Let's generalize the definition of interference today.

3. Let $\{x > 0\}\ S_1\ \{x \geq 0\}$ and $\{x > 0\}\ S_2\ \{x > 0\}$ be two threads, do they interfere with each other?
   - If S_1 runs before S_2, then S_1terminates with $x \geq 0$, which interferes with the precondition $x > 0$ of S_2.
   - If S_1 runs after S_2, then S_1 terminates with $x \geq 0$, which interferes with the postcondition $x > 0$ of S_2.
   - On the other hand, no matter runs S_2 before or after S_1, its postcondition $x > 0$ doesn't interfere with pre- and post- conditions of S_1.

- Our observation is that, if two threads both executed atomically, and their execution doesn't affect the pre- and post- conditions of each other, then their execution order doesn't matter.

- We define **interference-freedom check** for $\{p\} < S > \{\ldots\}$ versus a predicate q is the triple $\{p \land q\} < S > \{q\}$. If this $\{p \land q\} < S > \{q\}$ is valid, then we say that $\{p\} < S > \{\ldots\}$ does not interfere with q.

- Like the disjoint / interference test we studied for disjoint threads; the interference-freedom check is also too strong. Passing the check means "no interference", but failing the check only means that "$\{p\} < S > \{\ldots\}$ MIGHT interfere with q at runtime".

4. Do the following $\{p\} < S > \{\ldots\}$ and q pass the interference-freedom check?
   a. $\{x > 0\}\ S_1\ \{\ldots\}$ and $x > 0$ in example 3.
      The triple we need to check is $\{x > 0\}\ S_1\ \{x > 0\}$ which is not necessarily valid, since we only know $\{x > 0\}\ S_1\ \{x \geq 0\}$, so it doesn't pass the test.

   b. $\{p\} < S > \{\ldots\}$ and F.
      Yes. The precondition is $p \wedge F$ so the triple is valid.

   c. $\{p\}\ x \coloneqq x + 1\ \{\ldots\}$ and $x \geq 0$.
      Yes. $p \wedge x \geq 0$ is either F or $\{p \wedge x \geq 0\}\ x \coloneqq x + 1\ \{x \geq 0\}$ is valid.

   d. $\{p\}\ x \coloneqq x + 1\ \{\ldots\}$ and $x < 0$.
      No.

   e. $\{x \% 4 = 2\}\ x \coloneqq x + 4\ \{\ldots\}$ and $even(x)$.
      Yes.

- Once we have a notion of interference freedom of an atomic triple versus a predicate, we can build up to a notion of interference between threads.

- We use S^* to represent the proof outline of S.
- **The atomic statement** $\{p_1\}\ S_1\ \{\ldots\}$ **does not interfere with the proof outline** $\{p_2\}\ S_2^*\ \{q_2\}$ if it doesn't interfere with p_2 nor with q_2, nor with any precondition before an atomic statement in S_2^*, I.e., $\{p_1\}\ S_1\ \{\ldots\}$ does not interfere with any r where $\{r\} < \cdots > \{\ldots\}$ appears in S_2^*.
- **A proof outline** $\{p_1\}\ S_1^*\ \{q_1\}$ **does not interfere with the proof outline** $\{p_2\}\ S_2^*\ \{q_2\}$ if every atomic statement $\{r\} < S > \{\ldots\}$ in S_1^* it doesn't interfere with $\{p_2\}\ S_2^*\ \{q_2\}$.
- Two proof outlines $\{p_1\}\ S_1^*\ \{q_1\}$ and $\{p_2\}\ S_2^*\ \{q_2\}$ are **interference-free** if they don't interfere with each other.

5. Are $\{x \% 4 = 0\}\ x \coloneqq x + 3\ \{x \% 4 = 3\}$ and $\{even(x)\}\ x \coloneqq x + 1;\ \{odd(x)\}\ x \coloneqq x\ \{odd(x)\}$ interference-free?
   - We need to check the following triples:

▪ $\{x \% 4 = 0 \wedge even(x)\}\ x \coloneqq x + 3\ \{even(x)\}$	invalid
▪ $\{x \% 4 = 0 \wedge odd(x)\}\ x \coloneqq x + 3\ \{odd(x)\}$	valid
▪ $\{even(x) \wedge x \% 4 = 0\}\ x \coloneqq x + 1\ \{x \% 4 = 0\}$	invalid
▪ $\{even(x) \wedge x \% 4 = 3\}\ x \coloneqq x + 1\ \{x \% 4 = 3\}$	valid
▪ $\{odd(x) \wedge x \% 4 = 0\}\ x \coloneqq x\ \{x \% 4 = 0\}$	valid
$\{odd(x) \wedge x \% 4 = 3\}\ x \coloneqq x\ \{x \% 4 = 3\}$	valid

   - They are not interference-free.

<u>Parallelism Rule with Shared Variables and Interference-Freedom</u>

- **Parallelism Rule with Shared Variables**
  If for $k = 1 \ldots n$, we have $\{p_k\}\ S_k^*\ \{q_k\}$ are all interference free:
  1 $\{p_1\}\ S_1^*\ \{q_1\}$
  2 $\{p_2\}\ S_2^*\ \{q_2\}$
  …

n $\{p_n\}\, S_n^* \,\{q_n\}$

n + 1 $\{p_1 \land p_2 \land \ldots \land p_n\}\, [S_1^* \parallel S_2^* \parallel \cdots \parallel S_n^*]\, \{q_1 \land q_2 \land \ldots \land q_n\}$ parallelism 1,2, … , n

6. Show that provable triples $\{x = 0\}\, x \coloneqq x + 2\, \{x = 0 \lor x = 2\}$ and $\{T\}\, x \coloneqq 0\, \{x = 0 \lor x = 2\}$ are interference free.
   - We need to prove the following triples being valid:
     - $\{x = 0 \land T\}\, x \coloneqq x + 2\, \{T\}$
     - $\{x = 0 \land (x = 0 \lor x = 2)\}\, x \coloneqq x + 2\, \{x = 0 \lor x = 2\}$
     - $\{T \land x = 0\}\, x \coloneqq 0\, \{x = 0\}$
     - $\{T \land (x = 0 \lor x = 2)\}\, x \coloneqq 0\, \{x = 0 \lor x = 2\}$

   - Applying the parallelism rule, we can prove the following triple is valid:

$$\{x = 0\}\, [x \coloneqq x + 2 \parallel x \coloneqq 0]\, \{x = 0 \lor x = 2\}$$