More on Disjoint Parallel Programs

Let's start with an example as the review of last lecture.

1. Is the program $S \equiv [S_1 \parallel S_2]$ a disjoint parallel program? Here, we have
   $S_1 \equiv a \coloneqq v$
   $S_2 \equiv \textbf{if } b \le 0 \textbf{ then } v \coloneqq c * b \textbf{ else } b \coloneqq b * 2 \textbf{ fi}$

   We can come up with the following table.

i	j	$vars(S_i)$	$changes(S_j)$	S_j interferes with S_i?
1	2	a, v	b, v	Yes
2	1	b, c, v	a	No

   Since S_2 can interfere with S_1, so S is not a disjoint parallel program.

- In the above example, we can see that S_2 only interferes with S_1 if we execute the true branch. In fact, this disjoint/interference test we use is static and can be too strict, it can over-estimate the inference:
  - If our test shows that each pair of threads are disjoint, then the program is **definitely** a disjoint parallel program.
  - If our test shows that each some thread can interfere with other threads, then it only means that this thread **might** interfere with other threads while executing.

- In general, with $[S_1 \parallel S_2]$ we can execute either S_1 or S_2 for the next step. In an evaluation graph, the current evaluation path splits into two paths. In general, there might be no way for those two paths to eventually merge back together into one path, but disjoint parallel programs are different.
- **[Diamond Property]** Let $[S_1 \parallel S_2]$ be a disjoint parallel program. If $\langle S_1, \sigma\rangle \to \langle T_1, \tau_1\rangle$ and $\langle S_2, \sigma\rangle \to \langle T_2, \tau_2\rangle$ then there is a state τ such that $\langle [T_1 \parallel S_2], \tau_1\rangle$ and $\langle [S_1 \parallel T_2], \tau_2\rangle$ both $\to \langle [T_1 \parallel T_2], \tau\rangle$.
  - It is easy to see the property is true since the order of execution of S_1 and S_2 does not matter: any change in state caused by S_1 will be the same whether we execute S_2 (and vice-versa).
  - It is also easy to see how the property is named if we draw the evaluation graph. Here I use different colors on directed edges: it means if red edges exist, then the green edges exist.

$$\begin{tikzcd}
 & \langle [S_1 \parallel S_2], \sigma\rangle \arrow[dl] \arrow[dr] & \\
\langle [T_1 \parallel S_2], \tau_1\rangle \arrow[dr] & & \langle [S_1 \parallel T_2], \tau_2\rangle \arrow[dl] \\
 & \langle [T_1 \parallel T_2], \tau\rangle &
\end{tikzcd}$$

- Diamond property can be generalized into a more general property called **confluence**: where the one-step arrows are replaced by any-step arrows ($\to$ becomes $\to^*$). Similarly, if red edges exist, then the green edges exist.

$$\begin{tikzcd}
 & \langle [S_1 \parallel S_2], \sigma\rangle \arrow[dl, "*"'] \arrow[dr, "*"] & \\
\langle [T_1 \parallel S_2], \tau_1\rangle \arrow[dr, "*"'] & & \langle [S_1 \parallel T_2], \tau_2\rangle \arrow[dl, "*"] \\
 & \langle [T_1 \parallel T_2], \tau\rangle &
\end{tikzcd}$$

- **[Confluence]** Let $[S_1 \parallel S_2]$ be a disjoint parallel program If $\langle S_1, \sigma\rangle \rightarrow^* \langle T_1, \tau_1\rangle$ and $\langle S_2, \sigma\rangle \rightarrow^* \langle T_2, \tau_2\rangle$ then there is a state τ such that $\langle [T_1 \parallel S_2], \tau_1\rangle$ and $\langle [S_1 \parallel T_2], \tau_2\rangle$ both $\rightarrow^* \langle [T_1 \parallel T_2], \tau\rangle$.
    - It is easy to see that the diamond property implies confluence (confluence is a generalized diamond property), but not the opposite.

- Because execution of disjoint parallel programs is confluent, if execution terminates, it terminates in a unique state. This gives us the following theorem.
- **[Theorem (Unique Result of Disjoint Parallel Program)]:** If S is a disjoint parallel program then $M(S, \sigma) = \{\tau\}$, where $\tau \in \Sigma \cup \{\bot_d, \bot_e\}$.
    - It is easy to prove this theorem using confluence: If $\langle S_1, \sigma\rangle \rightarrow^* \langle E, \tau_1\rangle$ and $\langle S_2, \sigma\rangle \rightarrow^* \langle E, \tau_2\rangle$ then there is a state τ such that $\langle [E \parallel S_2], \tau_1\rangle$ and $\langle [S_1 \parallel E], \tau_2\rangle$ both $\rightarrow^* \langle [E \parallel E], \tau\rangle$.

Inference Rules for Disjoint Parallel Programs

We need some inference rules that can prove the validity of triples with disjoint parallel programs.

- **Sequentialization Rule**
  If $S_1, S_2, \dots, S_n$ are pairwise disjoint:
    1 $\{p\}\ S_1; S_2; \dots; S_n\ \{q\}$
    2 $\{p\}\ [S_1 \parallel S_2 \parallel \cdots \parallel S_n]\ \{q\}$ Sequentialization 1

    - We call the **sequentialization** of the parallel statement $[S_1 \parallel S_2 \parallel \cdots \parallel S_n]$ (not necessarily disjoint parallel) is the sequence $S_1; S_2; \dots; S_n$. The **sequentialized execution** of the parallel statement is the execution of its sequentialization: We evaluate S_1 completely, then S_2 completely, and so on.
    - For a disjoint parallel program, the execution order of threads does not matter.

2. Prove that $\{T\}\ [a \coloneqq x + 1 \parallel b \coloneqq x + 2]\ \{a + 1 = b\}$.
    - First, we need to show all threads are pairwise disjoint to use the sequentialization rule, and we omit this proof here.
    - Then, we need to prove $\{T\}\ a \coloneqq x + 1;\ b \coloneqq x + 2\ \{a + 1 = b\}$. This can be proved using backward assignment axiom or forward assignment axiom:
      $$\{T\}\{x + 1 + 1 = x + 2\}\ a \coloneqq x + 1;\ \{a + 1 = x + 2\}\ b \coloneqq x + 2\ \{a + 1 = b\}$$
    - In the end, applying the sequentialization rule we have: $\{T\}[a \coloneqq x + 1 \parallel b \coloneqq x + 2]\{a + 1 = b\}$.

- Note that, since the execution order of threads does not matter, $[S_1 \parallel S_2]$ and $[S_2 \parallel S_1]$ work the same way. Thus, while using the sequentialization rule, you can get $\{p\}\ [S_1 \parallel S_2]\ \{q\}$ from either $\{p\}\ S_1; S_2\ \{q\}$ or $\{p\}\ S_2; S_1\ \{q\}$.
- Sequentialization rule is not easy to use when there are many threads, we need to calculate a lot of pre- or post-conditions to prove the sequentialization.

Let's come up with some rules of inference that are easy to use. Someone suggested that, using the fact that disjoint threads can write at the same time without affecting variables in other threads, we come up with the parallelism rule.

- **Parallelism Rule (v 1.0)?**
  If $S_1, S_2, \dots, S_n$ are pairwise disjoint:

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

- In the format of proof outline, we write: $\{p_1 \wedge p_2 \wedge \ldots \wedge p_n\}\; [\{p_1\}\, S_1\, \{q_1\} \parallel \{p_2\}\, S_2\, \{q_2\} \parallel \cdots \parallel \{p_n\}\, S_n\, \{q_n\}]\; \{q_1 \wedge q_2 \wedge \ldots \wedge q_n\}$

3. Prove $\{x = y = z = c\}\; [x := x^2 \parallel y := y^2 \parallel z := (z - d) * (z + d)]\; \{x = y = z + d^2\}$, where c, d are named constants.
   First, we need to prove that all threads are pairwise disjoint, and we omit this proof here. Using the parallelism rule, we can come up with the following full proof outline:

   $\{x = y = z = c\}\; \{x = c \wedge y = c \wedge z = c\}$
   $[\{x = c\}\; x := x^2\; \{x = c^2\} \parallel \{y = c\}\; y := y^2 \{y = c^2\} \parallel \{z = c\}\; z := (z - d) * (z + d)\; \{z = c^2 - d^2\}]$
   $\{x = c^2 \wedge y = c^2 \wedge z = c^2 - d^2\}$
   $\{x = y = z + d^2\}$

- The parallelism rule (v 1.0) can fail in some situations, even if all threads are disjoint.

4. For each of the following proof outlines, point out what is wrong in the proof and why parallelism rule (v 1.0) does not work.
   - $\{x = 0\}[\; \{x = 0\}\; x := 1\; \{x = 1\} \parallel \{x = 0\}\; y := 0\; \{x = y = 0\}]\; \{x = 1 \wedge x = y = 0\}$
     - The postcondition is wrong: $x = 1 \wedge x = y = 0$ is false, but this is not the case if we analyze the program.
     - It fails because the postcondition of thread 1 $\{x = 1\}$ violates the precondition of thread 2.

   - $\{x \geq y \wedge y = z\}\; [\{x \geq y\}\; x := x + 1\; \{x > y\} \parallel \{y = z\}\; y := y * 2; z := z * 2\; \{y = z\}]\; \{x > y \wedge y = z\}$
     - The postcondition is wrong: we don't always have $x > y$ in the postcondition. For example, if we start $x = y = z = 2$, then we will end up with $x = 3 \wedge y = z = 4$.
     - If we execute thread 2 first, then we can increase y which can make the precondition of the thread 1 not true anymore.

- We have the following observation. So that parallelism rule can work, we need not only "a thread doesn't change variables in other threads" (aka, pairwise disjoint), but also more: "a thread doesn't change variables appear in the conditions of other threads".

- For a set of predicates $p_1, p_2, \ldots, p_n$, we define $free(p_1, \ldots, p_n)$ = the set of variables that are free (have free occurrence) in $p_1, p_2, \ldots, p_n$.
- Given triples $\{p_1\}\, S_1\, \{q_1\}$ and $\{p_2\}\, S_2\, \{q_2\}$, we say **S_1 interferes with the conditions of S_2** if $change(S_1) \cap free(p_2, q_2) \neq \emptyset$. And we say **$S_1$ and S_2 have disjoint conditions** if $change(S_1) \cap free(p_2, q_2) = \emptyset$ and $change(S_2) \cap free(p_1, q_1) = \emptyset$.

Here, we give the real parallelism rule.

- **Parallelism Rule**
  If $S_1, S_2, \ldots, S_n$ are pairwise disjoint, and $S_1, S_2, \ldots, S_n$ have 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 $\{p_1 \wedge p_2 \wedge \ldots \wedge p_n\}\, [S_1 \parallel S_2 \parallel \cdots \parallel S_n]\, \{q_1 \wedge q_2 \wedge \ldots \wedge q_n\}$ parallelism 1,2, … , n

5. Are the following threads pairwise disjoint? Do they have disjoint conditions?
   - $\{p_1\}\, S_1\, \{q_1\} \equiv \{x \geq y\}\, x \coloneqq x + 1\, \{x > y\}$
     $\{p_2\}\, S_2\, \{q_2\} \equiv \{y = z\}\, y \coloneqq y * 2; z \coloneqq z * 2\, \{y = z\}$

     We can come up with the following table:

i	j	$change(S_i)$	$vars(S_j)$	$free(p_j, q_j)$	$S_i\ intf\ S_j$	$S_i\ intf\ cond_j$
1	2	x	y, z	y, z	No	No
2	1	y, z	x	x, y	No	Yes

     So, S_1 and S_2 are disjoint, but they don't have disjoint conditions. That's why we cannot use the parallelism rule to prove $\{p_1 \wedge p_2\}\, [S_1 \parallel S_2]\, \{q_1 \wedge q_2\}$.

- Since it is much easier to prove with the parallelism rule, we want to find a way to remove the interference of conditions among disjoint threads. This can be done by **aging all free variables in the precondition.**

6. Find a postcondition q and prove $\{x \geq y \wedge y = z\}\, [x \coloneqq x + 1 \parallel y \coloneqq y * 2; z \coloneqq z * 2]\, \{q\}$ is correct under total correctness.

   - Apparently, two threads in the composition are disjoint since they don't share variables; and example 5 shows that they don't have disjoint conditions.

   - We rewrite the precondition to $x_0 \geq y_0 \wedge y_0 = z_0 \wedge x = x_0 \wedge y = y_0 \wedge z = z_0$.
   - When we prove the first thread, we use the precondition but with all variables aged: $\{\, x_0 \geq y_0 \wedge y_0 = z_0 \wedge x = x_0\}\, x \coloneqq x + 1\, \{\, x_0 \geq y_0 \wedge y_0 = z_0 \wedge x = x_0 + 1\}$.
   - When we prove the second thread, we use the aged precondition as well: $\{\, x_0 \geq y_0 \wedge y_0 = z_0 \wedge y = y_0 \wedge z = z_0\}\, y \coloneqq y * 2; z \coloneqq z * 2\, \{\, x_0 \geq y_0 \wedge y_0 = z_0 \wedge y = y_0 * 2 \wedge z = z_0 * 2\}$.

   - We can recalculate the following table:

i	j	$change(S_i)$	$vars(S_j)$	$free(p_j, q_j)$	$S_i\ intf\ S_j$	$S_i\ intf\ cond_j$
1	2	x	y, z	y, z, x_0, y_0, z_0	No	No
2	1	y, z	x	x, x_0, y_0, z_0	No	No

   Thus, we can come up with the following full proof outline that uses parallelism rule:

$\{x \geq y \wedge y = z\}$
$\{x_0 \geq y_0 \wedge y_0 = z_0 \wedge x = x_0 \wedge y = y_0 \wedge z = z_0\}$
$[\{\, x_0 \geq y_0 \wedge y_0 = z_0 \wedge x = x_0\}\, x \coloneqq x + 1\, \{\, x_0 \geq y_0 \wedge y_0 = z_0 \wedge x = x_0 + 1\}$
$\parallel \{\, x_0 \geq y_0 \wedge y_0 = z_0 \wedge y = y_0 \wedge z = z_0\}\, y \coloneqq y * 2; z$
$\coloneqq z * 2\, \{\, x_0 \geq y_0 \wedge y_0 = z_0 \wedge y = y_0 * 2 \wedge z = z_0 * 2\}]$
$\{x_0 \geq y_0 \wedge y_0 = z_0 \wedge x = x_0 + 1 \wedge y = y_0 * 2 \wedge z = z_0 * 2\}$

7. Prove $\{x = y = z = c\}\, [x \coloneqq\ x^2 \parallel y \coloneqq y^2 \parallel z \coloneqq (z - d\,) * (z + d)]\, \{x = y = z + d^2\}$, where c, d are named constants.

- This is the same question as Example 3. Without the hint of the provided precondition in each thread, it is very natural to use $x = y = z = c$ as the precondition of every thread, then we have the following **wrong** partial proof outline if we use the parallelism rule:

  $\{x = y = z = c\}$
  $[\,\{x = y = z = c\}\; x \coloneqq x^2\; \{x = c^2 \wedge y = z = c\}$
  $\parallel \{x = y = z = c\}\; y \coloneqq y^2 \{y = c^2 \wedge x = z = c\}$
  $\parallel \{x = y = z = c\}\; z \coloneqq (z - d\,) * (z + d)\; \{z = c^2 - d^2 \wedge x = y = c\}]$
  $\{x = c^2 \wedge y = z = c \wedge y = c^2 \wedge x = z = c \wedge z = c^2 - d^2 \wedge x = y = c\}$

- It is wrong because interference in conditions is everywhere. We want to age the variables in the precondition: $\{x_0 = y_0 = z_0 = c \wedge x = x_0 \wedge y = y_0 \wedge z = z_0\}$
- For the first branch, we will use aged precondition: $\{x_0 = y_0 = z_0 = c \wedge x = x_0\}\; x \coloneqq x^2\; \{x_0 = y_0 = z_0 = c \wedge x = x_0^2\}$.
- Similarly, for the second thread: $\{x_0 = y_0 = z_0 = c \wedge y = y_0\}\; y \coloneqq y^2\; \{x_0 = y_0 = z_0 = c \wedge y = y_0^2\}$.
- And for the third thread: $\{x_0 = y_0 = z_0 = c \wedge z = z_0\}\;\; z \coloneqq (z - d) * (z + d)\;\; \{x_0 = y_0 = z_0 = c \wedge z = z_0^2 - d^2\}$.

- In the end, we get the following full proof outline:

  $\{x = y = z = c\}$
  $\{x_0 = y_0 = z_0 = c \wedge x = x_0 \wedge y = y_0 \wedge z = z_0\}$
  $[\,\{x_0 = y_0 = z_0 = c \wedge x = x_0\}\; x \coloneqq x^2\; \{x_0 = y_0 = z_0 = c \wedge x = x_0^2\}$
  $\parallel \{x_0 = y_0 = z_0 = c \wedge y = y_0\}\; y \coloneqq y^2\; \{x_0 = y_0 = z_0 = c \wedge y = y_0^2\}$
  $\parallel \{x_0 = y_0 = z_0 = c \wedge z = z_0\}\;\; z \coloneqq (z - d\,) * (z + d)\;\; \{x_0 = y_0 = z_0 = c \wedge z = z_0^2 - d^2\}]$
  $\{x_0 = y_0 = z_0 = c \wedge x = x_0^2 \wedge y = y_0^2 \wedge z = z_0^2 - d^2\}$
  $\{x = y = z + d^2\}$