Sequential Nondeterminism

CS 536: Science of Programming, Fall 2021

A. *Why*

- Nondeterminism can help us avoid unnecessary determinism.
- Nondeterminism can help us develop programs without worrying about overlapping cases.

B. *Objectives*

At the end of these practice questions you should

- Be able to evaluate nondeterministic conditionals and loops.

C. *Nondeterminism*

1. Let $IF \equiv if\ B_1 \rightarrow S_1 \square B_2 \rightarrow S_2 \square \dots \square B_n \rightarrow S_n\ fi$ and $BB \equiv B_1 \vee B_2 \vee \dots B_n$.
   a. What property does BB have to have for us to avoid a runtime error when executing IF?
   b. Does it matter if we reorder the guarded commands? (E.g., if we swap $B_1 \rightarrow S_1$ and $B_2 \rightarrow S_2$.)

2. Let $U_1 \equiv if\ B_1 \rightarrow S_1 \square B_2 \rightarrow S_2\ fi$ and $U_2 \equiv if\ B_1\ then\ S_1\ else\ if\ B_2\ then\ S_2\ fi\ fi$.
   a. Fill in the table below to describe what happens for each combination of B_1 and B_2 being true or false.

If $\sigma \vDash \dots$	U_1	U_2
$B_1 \wedge B_2$	Executes S_1 or S_2	
$B_1 \wedge \neg B_2$		
$\neg B_1 \wedge B_2$		
$\neg B_1 \wedge \neg B_2$.		

b. For what kinds of states σ can statements U_1 and U_2 behave differently?

3. Let $DO \equiv do\ B_1 \rightarrow S_1 \square B_2 \rightarrow S_2 \square \dots \square B_n \rightarrow S_n\ od$ and $BB \equiv B_1 \vee B_2 \vee \dots B_n$. What property does *BB* have to have for us to avoid an infinite loop when executing *DO*?

4. Consider the loop $i := 0$; $do\ i < 1000 \rightarrow S_1;\ i := i+1 \square i < 1000 \rightarrow S_2;\ i := i+1\ od$ (where neither S_1 nor S_2 modifies i). Do we know anything about how many times or in what pattern we will execute S_1 vs S_2?

5. Consider the loop $x := 1;\ do\ x \geq 1 \rightarrow x := x+1 \square x \geq 2 \rightarrow x := x-2\ od$. Can running it lead to an infinite loop?

6. What are the reasons mentioned in the text for why using nondeterminism might be helpful?

7. What is $M(S, \{x = 1\})$ where $S \equiv do\ x \leq 20 \rightarrow x := x*2 \square x \leq 20 \rightarrow x := x*3\ od$?

Solution to Practice 7 (Nondeterministic Sequential Programs)

1. (Basic properties of nondeterministic if)
   a. We need $\sigma \vDash BB$, because if $\sigma \vDash \neg BB$, then $M(IF, \sigma) = \{\bot_e\}$. (In English: At least one guard must be true; if none of them are true, we get a runtime error.)
   b. The order of the guarded commands doesn't matter: If more than one guard is true, we nondeterministically choose one element from the set of corresponding statements, and in a set, the elements aren't ordered.
2. (Deterministic vs nondeterministic conditionals) Recall $U_1 \equiv$ *if* $B_1 \rightarrow S_1 \,\Box\, B_2 \rightarrow S_2$ *fi* and $U_2 \equiv$ *if* B_1 *then* S_1 *else if* B_2 *then* S_2 *fi*.
   a. Execution of U_1 and U_2:
   b. U_1 and U_2 behave the same when one of B_1 and B_2 is true and the other is false. When both are true, U_2 always executes S_1 but U_1 will execute S_1 or S_2. When both of B_1 and B_2 are false, U_1 yields a runtime error but U_2 does nothing.
3. The nondeterministic *do-od* loop halts if BB is false at the top of the loop; an infinite loop occurs when BB is always true at the top of the loop.
4. Say S_1 is run m times and S_2 is run n times. We know $0 \le m, n \le 1000$ and $m+n = 1000$, but that's all. At each iteration, the choice is nondeterministic (i.e., unpredictable). The choice does not have to be random (like with a coin flip), and the sequence of choices don't have to follow an pattern or distribution or be fair, etc. We can't even assign a probability to any particular sequence of choices (like "always choose S_1").
5. It's possible that the loop could run forever. There's no guaranteed fairness in nondeterministic choice, so we could increment x by 1 many more times than we decrement it by 2.
6. Reason 1: Nondeterminism Makes It Easy to Combine Partial Solutions.
   Reason 2: Nondeterminism Makes it Easy to Ignore Overlapping Cases

