Generalization

Mathematics —and computer science— often involve the process of *generalizing* statements. By *generalizing* a statement, we mean it to include new situations, or states.

Here's an example. A kid observes these statements:

Adding 1 to 0 yields an odd integer, 1.
Adding 1 to 2 yields an odd integer, 3.

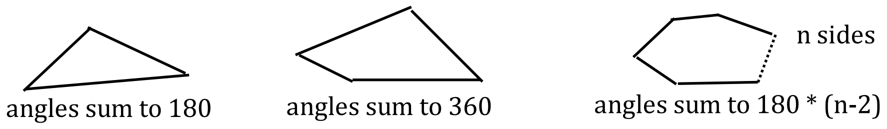
The kid sees a pattern and guesses that

adding 1 to an even integer yields an odd integer.

The kid *generalized* the statements "adding 1 to 0 and adding 1 to 2 …" to "adding 1 to an even integer …".

Here's a silly generalization. The 5 Irish boys I know all have red hair. I generalize to: All Irish boys have red hair."

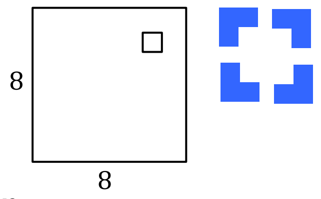
Here's a good example. Some mathematician knew that the sum of the angles of a triangle was 180 degrees, figured out that the sum of the angles of a 4-sided convex figure was 360 degrees, and generalized these facts to the case of n sides. That mathematician saw a pattern, generalized, and then proved the generalization. It was a generalization of a statement about n = 3 and n = 4 to a statement about all n ≥ 3.



In the two cases discussed above, the generalization was good; one could prove the two generalizations. But sometimes a generalization is not true. Here's a tongue-in-cheek physicist's generalization that all odd numbers at least 3 are prime. Well, 3, 5, and 7 are prime. 9 isn't, but that's just an experimental error. 11 and 13 are prime. That's enough: 3, 5, 7, 11, 13 —we see a pattern and generalize to: all odd numbers bigger than 1 are prime.

Here's a neat example. We have an 8 by 8 kitchen with a 1 by 1 refrigerator on one of the squares of the grid. Elaine wants the kitchen tiled with L-shaped squares —2-by-2 squares with one corner removed— covering every square except the one on which the refrigerator sits. Let's write this task as: Tile an 8 by 8 kitchen with one square covered. How do we do that? That's difficult! We don't even know whether it can be done.

We can get a handle on the problem by generalizing. We have to tile a 2^3 x 2^3 kitchen. Generalize that to: Tile a 2^n by 2^n kitchen with one square covered, where 0 ≤ n. Then will be able to tile the kitchen using recursion (not done here!).



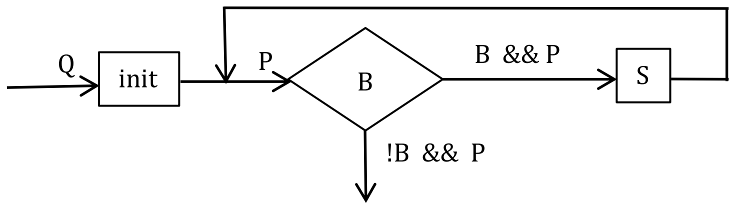
Without the generalization, this is a very hard problem. With the generalization, anyone who fully understands recursion can solve it.

Generalizing too much may lead to an unsolvable problem. For example, suppose we generalize to: Tile an n by n kitchen with one square covered, for 0 ≤ n. That can't be done, as you can see by trying to tile a 3 x 3 kitchen.

Generalization and loop invariants

Generalization in math helps people to solve problems and to discover new hypotheses, which, hopefully, can be proved to be theorems. In the domain of loops, generalization will help us find loop invariants.

When we start developing a loop, we have a precondition Q and a postcondition R, that's all. R is true after the last iteration. Invariant P is supposed to be true after *every* iteration. Hence, P is a *generalization* of R. We try to generalize R to a situation in which simple initialization can make P true. In the next videos, we'll show ways to generalize a postcondition to arrive at a suitable invariant.



!B && P implies R