

Precise Specification

- Sorting is a well-understood concept that is surprisingly hard to specify precisely
- Provide an English-language statement describing the expected behavior of a routine $\text{SORT}(X,Y)$ where X is an input vector of integers and Y is an output vector
 - You may assume the routine sorts in ascending order

Trouble Spots

- Does your specification handle the following cases?
 - $\langle 1, 2 \rangle \rightarrow \langle 4, 5 \rangle$
 - $\langle 1, 1, 2, 3 \rangle \rightarrow \langle 1, 2, 3 \rangle$
 - $\langle 1, 2, 3 \rangle \rightarrow \langle 1, 1, 2, 3 \rangle$
 - $\langle 1, 3, 3, 2, 2, 2 \rangle \rightarrow \langle 1, 2, 2, 3, 3, 3 \rangle$

English Specification

- Y is an ordered version of X
 - Y is ordered
 - Y and X have the same elements
 - With the same numbers of occurrences!

More Precise Specification

- Now express the same behavior more precisely
 - First give a signature (parameter and return types)
 - Next determine the situations in which you expect the program to work (its preconditions)
 - Then try to describe the relation between the input and the output in English (postconditions)
 - Finally, translate your specifications into predicate calculus

SIGNATURE

- ?

SIGNATURE

- `Vector<int> Y = SORT(Vector<int> X)`
 - X and Y are explicitly named so we can use them in the pre and post conditions

Preconditions

- ?

Preconditions

- For SORT any input vector of integers X qualifies
 - The preconditions assume the types are correct
- Thus, SORT is always able to run
 - This precondition is designated as `true`

Postconditions

- ?

Postconditions

- 1.Y must be the same length as X

Postconditions

- 1.Y must be the same length as X
- 2.The output vector Y must be ordered

Postconditions

- 1.Y must be the same length as X
- 2.The output vector Y must be ordered
- 3.The contents of Y must be the "same as" the contents of X

Postconditions

1. Y must be the same length as X
2. The output vector Y must be ordered
3. The contents of Y must be the "same as" the contents of X
 - Everything in X must be in Y
 - Everything in Y must have come from X
 - The number of occurrences of each item in Y must be the same as its number of occurrences in X
 - Another version of these properties is to say that Y is a *permutation* of X
 - Yet another version is to say that if the contents of X and Y are treated as bags, the two bags are equal
 - A *bag* is like a set, except that multiple instances of an element are allowed and that a count is kept of the occurrences

Review of Predicate Logic

- *Propositional logic*: assertions about the relationships among constants
 - $7 > 0 \wedge 6 > 0 \Rightarrow 7 * 6 > 0$
- *Predicate logic*: quantified assertions about the relationships among variables and constants
 - $\forall X \bullet \forall Y \bullet ((X > 0 \wedge Y > 0) \Rightarrow X * Y > 0)$

Syntax of Predicate Logic

sentence = simple proposition

- | predicate
- | \neg sentence // NOT
- | (sentence \wedge sentence) // AND
- | (sentence \vee sentence) // OR
- | (sentence \Rightarrow sentence) // IMPLIES
- | (sentence \Leftrightarrow sentence) // IFF (if and only if)
- | \forall variable \bullet sentence // ALL
- | \exists variable \bullet sentence // EXISTS

Syntax - 2

simple proposition = P | Q | R ...
predicate = predicate name (term+)
predicate name = f | g | h ...
term = constant | variable
constant = a | b | c ... 0 | 1 | 2 ...
variable = X | Y | Z ...

Predicate Calculus for SORT

Vector $Y = \text{SORT}(\text{Vector } X)$

pre: true

post: $\text{LENGTH}(X) == \text{LENGTH}(Y) \wedge$

$\text{ORDERED}(Y) \wedge$

$\text{PERMUTATION}(X, Y)$

1. Y is the Same Length as X

- X and Y are vectors
- Vectors (as a datatype) have properties one of which is their length
- $|X| == |Y|$ or $X.\text{length}() == Y.\text{length}()$ or $\#X == \#Y$
- This raises the question of who is responsible for allocating the space for Y
 - Caller preallocates or sort does the allocation
 - In the former case, the assertion would be a *precondition*; in the latter, it would be a *postcondition*

2. Y is ordered

- $\forall i : 1 \leq i < |Y| \bullet Y[i] \leq Y[i + 1]$
 - For all positions in Y from the first through the one before the last, the value stored in the vector at that position is no greater than the value at the next position
- Vectors have an indexing operation
- Counting starts from 1
- \forall reads "for all"

3. The Contents of Y is the "same as" the Contents of X

- Two vectors with the same contents but in which the values may be ordered differently are called *permutations*
- Y is a permutation of X; $\text{PERM}(X, Y)$
 - Precondition: $|X| == |Y|$
 - Postcondition
 - $|X| == 0 \Rightarrow \text{true}$
 - $X[1] == Y[1] \Rightarrow \text{PERM}(\text{tail}(X), \text{tail}(Y))$
 - $\exists j : 1 < j \leq |Y| \bullet X[1] == Y[j] \wedge \text{PERM}(\text{tail}(X), \text{concat}(Y[1 .. j - 1], Y[j + 1 .. |Y|]))$

Explanation

- \Rightarrow is an *if* expression, with the condition on the left, and the consequent on the right
- There is an implicit "else" between clauses
- `tail` is an operation on vectors that returns a new vector like the argument except that the first element has been removed
- `concat` is an operation on vectors that creates a new vector by pasting together the two vector arguments
- \exists reads "there exists"
- `i .. j` denotes the integers from `i` to `j` inclusive