A simple strategy for finding invariants

Maximum product subarray

Octobre 2018, Pierre-Edouard Portier

http://peportier.me/teaching_2018_2019/
$a(i: j - 1)$ denotes all adjacent elements $a(k)$ in array $a(0: n - 1)$ with $0 \leq i \leq k \leq j \leq n - 1$.

It is called a segment (or subarray) of the array.

When $i = j$, $a(i: j - 1)$ is an empty segment.
\langle Q \ i \colon r(i) \colon f(i) \rangle \text{ denotes a quantifier.}

To each quantifier $Q$ is associated an associative and commutative binary operator $q$.

For example, the quantifiers for binary operators $\land, \lor, +, \times, min, max$ are $\forall, \exists, \Sigma, \Pi, \downarrow, \uparrow$.

$i$ is a bounded variable with $r(i)$ its range, and $f(i)$ a function.

For example, $\langle \Sigma i : 1 \leq i \leq n : 2 \times i \rangle = n \times (n + 1)$. 
Here are a few useful rules that can be used with quantifiers:

Cartesian product:
\[
\langle Qi, j : r(j) \land s(i, j) : f(i, j) \rangle = \langle Qi : r(i) : \langle Qj : s(i, j) : f(i, j) \rangle \rangle
\]

Range split:
\[
\langle Qi : r(i) : f(i) \rangle = \langle Qi : r(i) \land b(i) : f(i) \rangle q \langle Qi : r(i) \land \neg b(i) : f(i) \rangle
\]

Singleton: \(\langle Qi : i = k : f(i) \rangle = f(k)\)

Range disjunction, if \(q\) is idempotent, we have:
\[
\langle Qi : r(i) \lor s(i) : f(i) \rangle = \langle Qi : r(i) : f(i) \rangle q \langle Qi : s(i) : f(i) \rangle
\]

Associativity and commutativity
\[
\langle Qi : r(i) : s(i) q f(i) \rangle = \langle Qi : r(i) : s(i) \rangle q \langle Qi : r(i) : f(i) \rangle
\]
A very general strategy, proposed by Xue Jinyun[1], for developing programs correct by construction:

Formulate as precisely as possible what the algorithm should do.

Partition the problem into sub-problems with the same structure as the original one.

Formulate the algorithm as a recurrence relation.

Develop loop invariants from the recurrence relation to transform the algorithm into a program.

Given \( a(0 : n - 1) \) an array of integers, compute \( p = \text{maxprod}(a(0 : n - 1)) \).

\[
\text{maxprod}(a(0 : n - 1)) = \langle \uparrow r, t : 0 \leq r \leq t \leq n : \text{prod}(r, t) \rangle
\]

\[
\text{prod}(r, t) = \langle \prod i : r \leq i \leq t - 1 : a(i) \rangle
\]

NB. For \( r = t \), \( \text{prod}(r, t) = 1 \) (the identity element of \( \times \))
We look for a function $F$ such that:

$$\text{maxprod}(a(0 : i - 1)) = F(\text{maxprod}(a(0 : i - 2)), a(i - 1))$$

with $1 \leq i \leq n$. 


Can \( \text{maxprod}(a(0 : i - 1)) \) be a function of \( \text{maxprod}(a(0 : i - 2)) \)?

\[
\text{maxprod}(a(0 : i - 1)) \\
= \{\text{def. of maxprod}\} \\
\langle \uparrow r, t : 0 \leq r \leq t \leq i : \text{prod}(r, t) \rangle \\
= \{\text{split with } t = i ; \text{def. of maxprod} ; \text{intro: } ma(i) = \langle \uparrow r : 0 \leq r \leq i : \text{prod}(r, i) \rangle\} \\
\text{maxprod}(a(0 : i - 2)) \uparrow ma(i)
\]
Can \( ma(i) \) be a function of \( ma(i - 1) \).

\[
ma(i) \\
= \{ \text{def. of } ma \} \\
= \{ \text{split with } r = i \} \\
= \{ \text{def. of prod} \} \\
= \{ \text{arithmetic; intro: } mi(i) = \langle \downarrow r : 0 \leq r \leq i : prod(r, i) \rangle \uparrow 1 \} \\
= \begin{cases} 
ma(i - 1) \times a(i - 1) & \text{if } a(i - 1) > 0 \\
1 & \text{if } a(i - 1) = 0 \\
mi(i - 1) \times a(i - 1) \uparrow 1 & \text{if } a(i - 1) < 0 
\end{cases}
\]
Now, we wish to express $mi(i)$ as a function of $mi(i - 1)$.

$$mi(i) = \{\text{def. of } mi\} \langle \downarrow r : 0 \leq r \leq i : \text{prod}(r, i) \rangle = \{\text{split with } r = i\} \langle \downarrow r : 0 \leq r \leq i - 1 : \text{prod}(r, i - 1) \times a(i - 1) \rangle \downarrow 1 = \{\text{arithmetic}\} \begin{cases} mi(i - 1) \times a(i - 1) \downarrow 1 & \text{if } a(i - 1) > 0 \\ 0 & \text{if } a(i - 1) = 0 \\ ma(i - 1) \times a(i - 1) & \text{if } a(i - 1) < 0 \end{cases}$$
We introduced 3 computations: \textit{maxprod}, \textit{ma} and \textit{mi}.
By associating variables to each of them, we obtain a straightforward invariant:

\[ I: \quad p = \text{maxprod}(a(0 : i - 1)) \land x = \text{ma}(i) \land y = \text{mi}(i) \land 1 \leq i \leq n \]

\( I \) can be trivially satisfied with the initialization:

\[ i, p, x, y := 1, 1, 1, 1 \]
The recurrence relation tells us what to do in the loop to maintain $I$. We obtain the following program:

```
i, p, x, y := 1, 1, 1, 1;
do i≠n →
    if a(i-1) >0 → x, y := x*a(i-1), min(y*a(i-1), 1)
    | a(i-1)==0 → x, y := 1, 0
    | a(i-1) <0 → x, y := max(y*a(i-1), 1), x*a(i-1)
fi
    p := max(p, x);
    i := i+1;
od
```
By a change of variable, we can rewrite this program as:

```plaintext
i,p,x,y := 0,1,1,1;
do i\neq n \rightarrow
  if a(i) > 0 \rightarrow x,y := x \times a(i), \min(y \times a(i), 1)
  | a(i) == 0 \rightarrow x,y := 1, 0
  | a(i) < 0 \rightarrow x,y := \max(y \times a(i), 1), x \times a(i)
  fi
p := \max(p, x);
i := i+1;
od
```