Techniques for Algorithm Verification: Example

Stefan D. Bruda

CS 310, Winter 2024

Power

```
ASSERT(true)
ASSERT(0 >= 0 && 1 == power(x,0))
i = 0;
ASSERT(i >= 0 && 1 == power(x,i))
y = 1;
ASSERT(i >= 0 && y == power(x,i))
while (i != n) {
    ASSERT(i >= 0 && y == power(x,i) && i != n)
    ASSERT(i >= 0 && y == power(x,i))
    y = y * x;
    i ++;
    ASSERT(i >= 0 && y == power(x,i))
}
ASSERT(i >= 0 && y == power(x,i) && !(i != n))
ASSERT(i >= 0 && y == power(x,i) && i == n)
ASSERT(y == power(x,n))
```

Power (total correctness)

```
ASSERT(n >= 0)
ASSERT(0 >= 0 && 1 == power(x,0) && n >= 0)
i = 0;
ASSERT(i >= 0 && 1 == power(x,i) && n >= 0)
y = 1;
ASSERT(i >= 0 && y == power(x,i) && n >= 0)
while (i != n) {
    ASSERT(i >= 0 && y == power(x,i) && i != n && n >= 0)
    ASSERT(i >= 0 && y == power(x,i) && n >= 0)
    y = y * x;
    i ++;
    ASSERT(i >= 0 && y == power(x,i) && n >= 0)
}
ASSERT(i >= 0 && y == power(x,i) && !(i != n) && n >= 0)
ASSERT(i >= 0 && y == power(x,i) && i == n && n >= 0)
ASSERT(y == power(x,n) && n >= 0)
ASSERT(y == power(x,n))
```

Power (variant)

```
ASSERT(n >= 0)
ASSERT(0 >= 0 && 1 == power(x,0) && n >= 0 && 0 <= 0 <= n)
i = 0;
ASSERT(i >= 0 && 1 == power(x,i) && n >= 0 && 0 <= i <= n)
y = 1;
ASSERT(i >= 0 && y == power(x,i) && n >= 0 && 0 <= i <= n)
while (i != n) {
    ASSERT(i >= 0 && y == power(x,i) && n >= 0 && 0 <= i <= n && i != n)
    ASSERT(i >= 0 && y == power(x,i) && n >= 0 && 0 <= i+1 <= n && i != n)
    ASSERT(i >= 0 && y == power(x,i) && n >= 0 && 0 <= i+1 <= n)
    ASSERT(i+1 >= 0 && y == power(x,i) && n >= 0 && 0 <= i+1 <= n)
    ASSERT(i+1 >= 0 && y*x == power(x,i+1) && n >= 0 && 0 <= i+1 <= n)
    y = y * x;
    ASSERT(i+1 >= 0 && y == power(x,i+1) && n >= 0 && 0 <= i+1 <= n)
    i ++;
    ASSERT(i >= 0 && y == power(x,i) && n >= 0 && 0 <= i <= n)
}
ASSERT(i >= 0 && y == power(x,i) && !(i != n) && n >= 0 && 0 <= i <= n)
ASSERT(i >= 0 && y == power(x,i) && i == n && n >= 0 && 0 <= i <= n)
ASSERT(y == power(x,n) && n >= 0)

ASSERT(y == power(x,n))
```

- The additional assertion `0 <= i <= n` supports `n-i` as variant which in turn establishes loop termination