Introduction to Program Synthesis (SS 25)

Exercise - Proof as Programs and Verification

Dr. rer. nat. Roman Kalkreuth

Chair for AI Methodology (**AIM**), Department of Computer Science,
RWTH Aachen University, Germany

Constructive Verification
Square of integer

Theorem
If n is an even integer, then n^2 is an even integer.

Proof.

- Let n be an even integer
- As n is even, there is some integer k such as $n = 2k$
- Naturally $n^2 = (2k)^2 = 4k^2 = 2(2k^2)$
- If $n^2 = 2(2k^2)$ is an even integer this means there is a number $m = 2k^2$ such as $n^2 = 2m$
- Thus n^2 is an even integer

$\square$

Constructive Verification
Square of integer

```
def square_of_integer(n: int) -> bool:
    k = n//2
    return n*n == 4*k*k

lambda n: n*n == 4*(n//2)*(n//2)
```

Inductive Verification

Sum of sequence of squares

Theorem

For $n \geq 1$ *and* $n \in \mathbb{N} : 1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$

Proof.

- **Base case:** $n = 1 : \sum_{i=1}^{1} i^2 = 1^2 \Leftrightarrow \frac{n(n+1)(2n+1)}{6} = \frac{1(1+1)(2\times1+1)}{6} = \frac{6}{6} = 1$
- **Inductive hypothesis:** $\sum_{i=1}^{k} i^2 = \frac{k(k+1)(2k+1)}{6}$
- **Inductive step:** $\sum_{i=1}^{k+1} i^2 = \frac{(k+1)(k+2)(2(k+1)+1)}{6}$
  - $\sum_{i=1}^{k+1} i^2 \Leftrightarrow \sum_{i=1}^{k} i^2 + (k+1)^2$

$$
\begin{aligned}
\sum_{i=1}^{k+1} i^2 &= \frac{k(k+1)(2k+1)}{6} + (k+1)^2 \\
&= \frac{k(k+1)(2k+1) + 6(k+1)^2}{6} \\
&= \frac{(k+1)(k(2+1) + 6(k+1)^2}{6} \\
&= \frac{(k+1)(2k^2 + 7k + 6)}{6} \\
&= \frac{(k+1)(k+2)(2(k+1)+1)}{6}
\end{aligned}
$$

Constructive Verification

Sum of sequence of squares

```python
def verify_sum(n: int) -> bool:
    if n == 1:
        return True
    powers = [i ** 2 for i in range(1, n + 1)]
    sum_ = lambda x: x * (x + 1) * (2 * x + 1) / 6
    pk = sum(powers) == sum_(n)
    return verify_sum(n - 1) if pk else False
```

Constructive Verification
Sum of sequence of squares

```
def verify_sum_optimized(n: int, power: int=0) -> bool:
    if n == 1:
        return True
    sum_ = lambda x: x * (x + 1) * (2 * x + 1) / 6
    pk = power == sum_
    power += n ** 2
    return verify_sum_optimized(n - 1, power) if pk else False
```

Exhaustive Verification
Multiple of three

Theorem

For $n \in \mathbb{Z} : n^2 - 1$ *is a multiple of* 3 *if* n *is not a multiple of* 3.

Proof.

- When n is not a multiple of 3, it is either $n = 3k + 1$ or $n = 3k + 2$
- **Case 1:**

$$\begin{aligned} n^2 - 1 &= (3k+1)^2 \\ &= (3k)^2 + 6k + 1 - 1 \\ &= 9k^2 + 6k = \mathbf{3(3k^2 + 2k)} \end{aligned}$$

- **Case 2:**

$$\begin{aligned} n^2 - 1 &= (3k+2)^2 \\ &= (3k)^2 + 2(3k)(2) + 2^2 - 1 \\ &= 9k^2 + 12k + 3 = \mathbf{3(3k^2 + 4k + 1)} \end{aligned}$$

$\square$

Exhaustive Verification
Multiple of three

```
def verify_term1(term, n):
    for i in range(-n, n + 1):
        if i % 3 > 0 and term(i) % 3 > 0:
            return False
    return True

def verify_term2(term, n):
    for i in range(-n, n + 1):
        if i % 3 > 0:
            print(f"{i} {term(i)} {term(i) % 3 == 0}")

verify_term1(lambda x: x ** 2 - 1, 100)
verify_term2(lambda x: x ** 2 - 1, 100)
```