

Software Specification and Verification

Course Introduction: Reasoning about Programs

Yih-Kuen Tsay

Department of Information Management
National Taiwan University

The Coffee Can Problem

The Setting:

-  **Initially:** a coffee can contains some **black** beans and some **white** beans.
-  **Action:** the following steps are repeated as many times as possible.
 1. Pick **any two** beans from the can.
 2. If they have the **same color**, put another black bean in and throw anything else away. (Assume there is a sufficient supply of additional black beans.)
 3. **Otherwise**, put the white bean back in and throw the black one away.
-  **Finally:** only one bean remains in the can.

The Coffee Can Problem

The Setting:

-  **Initially:** a coffee can contains some **black** beans and some **white** beans.
 -  **Action:** the following steps are repeated as many times as possible.
 1. Pick **any two** beans from the can.
 2. If they have the **same color**, put another black bean in and throw anything else away. (Assume there is a sufficient supply of additional black beans.)
 3. **Otherwise**, put the white bean back in and throw the black one away.
 -  **Finally:** only one bean remains in the can.
-  **Question:** what can be said about the **color of the last** remaining bean?

The Coffee Can Problem as a Program

```
 $B, W := m, n; \quad // \quad m > 0 \wedge n > 0$   
do  $B \geq 0 \wedge W \geq 2 \rightarrow B, W := B + 1, W - 2 \quad // \text{both white}$   
   $\parallel B \geq 2 \wedge W \geq 0 \rightarrow B, W := B - 1, W \quad // \text{both black}$   
   $\parallel B \geq 1 \wedge W \geq 1 \rightarrow B, W := B - 1, W \quad // \text{different colors}$   
od
```

(Note: one of the three alternatives in the **do** loop is arbitrarily chosen and executed until none is “enabled”, at which time the loop terminates.)

The Coffee Can Problem as a Program

```
 $B, W := m, n; \quad // \quad m > 0 \wedge n > 0$   
do  $B \geq 0 \wedge W \geq 2 \rightarrow B, W := B + 1, W - 2 \quad // \text{ both white}$   
   $\parallel B \geq 2 \wedge W \geq 0 \rightarrow B, W := B - 1, W \quad // \text{ both black}$   
   $\parallel B \geq 1 \wedge W \geq 1 \rightarrow B, W := B - 1, W \quad // \text{ different colors}$   
od
```

(Note: one of the three alternatives in the **do** loop is arbitrarily chosen and executed until none is “enabled”, at which time the loop terminates.)

🌐 What are the values of B and W , when the program terminates?

The Coffee Can Problem as a Program

```
 $B, W := m, n; \quad // \quad m > 0 \wedge n > 0$   
do  $B \geq 0 \wedge W \geq 2 \rightarrow B, W := B + 1, W - 2 \quad // \text{ both white}$   
   $\parallel B \geq 2 \wedge W \geq 0 \rightarrow B, W := B - 1, W \quad // \text{ both black}$   
   $\parallel B \geq 1 \wedge W \geq 1 \rightarrow B, W := B - 1, W \quad // \text{ different colors}$   
od
```

(Note: one of the three alternatives in the **do** loop is arbitrarily chosen and executed until none is “enabled”, at which time the loop terminates.)

- 🌐 What are the values of B and W , when the program terminates?
- 🌐 Will the program actually terminate?

Invariants and Rank Functions

- 🌐 An *invariant* captures something that is never changed by the program.

Invariants and Rank Functions

- 🌐 An *invariant* captures something that is never changed by the program.
- 🌐 A *rank function* (or variant function) measures the progress made by the program.

Invariants and Rank Functions

- 🌐 An *invariant* captures something that is never changed by the program.
- 🌐 A *rank function* (or variant function) measures the progress made by the program.
- 🌐 For the Coffee Can problem,
 - ☀️ (Loop) Invariant: the parity of the number of white beans never changes, i.e., $W \equiv n \pmod{2}$. (in addition, $B + W \geq 1$)
 - ☀️ Rank Function: the total number of beans, i.e., $B + W$.
 - ☀️ The do loop decrements the rank function by one in each iteration and eventually terminates when $B + W = 1$ (i.e., $B = 0 \wedge W = 1$ or $B = 1 \wedge W = 0$).

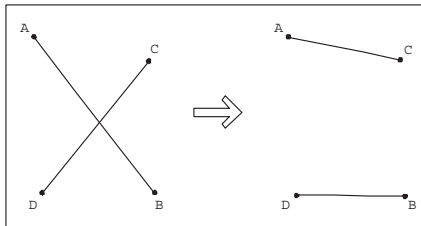
Invariants and Rank Functions

- 🌐 An *invariant* captures something that is never changed by the program.
- 🌐 A *rank function* (or variant function) measures the progress made by the program.
- 🌐 For the Coffee Can problem,
 - ☀️ (Loop) Invariant: the parity of the number of white beans never changes, i.e., $W \equiv n \pmod{2}$. (in addition, $B + W \geq 1$)
 - ☀️ Rank Function: the total number of beans, i.e., $B + W$.
 - ☀️ The do loop decrements the rank function by one in each iteration and eventually terminates when $B + W = 1$ (i.e., $B = 0 \wedge W = 1$ or $B = 1 \wedge W = 0$).
 - ☀️ So, what is the color of the remaining bean?

Another Example: Untangling Line Segments

🌐 The Setting:

- ☀ **Initially:** there are $2n$ points on the Euclidean plane. The points are **grouped in pairs** with a line segment connecting each pair.
- ☀ **Action:** the following untangling operation is repeatedly applied to the points.

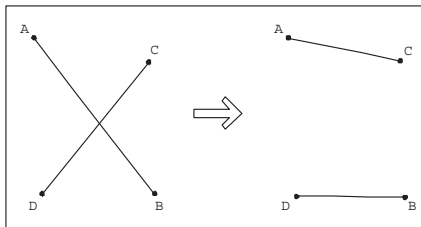


Note that new pairs of crossed line segments may result from this operation.

Another Example: Untangling Line Segments

🌐 The Setting:

- ☀️ **Initially:** there are $2n$ points on the Euclidean plane. The points are **grouped in pairs** with a line segment connecting each pair.
- ☀️ **Action:** the following untangling operation is repeatedly applied to the points.



Note that new pairs of crossed line segments may result from this operation.

🌐 Question: will this process terminate?

Untangling Line Segments (cont.)

- 🌐 **Rank Function:** the total length of all line segments. (Note: this needs to be refined.)
- 🌐 Each application of the untangling operation **reduces the total length** (thanks to the triangular inequality).
- 🌐 The above reduction in length must be greater than **some positive constant** which is determined in the initial state (by considering all possible groupings of four points).
- 🌐 The total length is finite and an infinite number of reductions by a positive constant is not possible.
- 🌐 Therefore, the untangling process will terminate.

Proving Termination Can Be Very Hard

```
function collatz( $n$ ): integer;  
begin  
  while  $n > 1$  do  
    if  $n$  is even then  $n := n/2$   
    else  $n := 3n + 1$   
  od  
  return  $n$   
end
```

🌐 What would be a suitable rank function for the while loop?

Proving Termination Can Be Very Hard

```
function collatz( $n$ ): integer;  
begin  
    while  $n > 1$  do  
        if  $n$  is even then  $n := n/2$   
        else  $n := 3n + 1$   
    od  
    return  $n$   
end
```

- 🌐 What would be a suitable rank function for the while loop?
- 🌐 Will the program terminate at all (for every possible input)?

Proving Termination Can Be Very Hard

```
function collatz( $n$ ): integer;  
begin  
  while  $n > 1$  do  
    if  $n$  is even then  $n := n/2$   
    else  $n := 3n + 1$   
  od  
  return  $n$   
end
```

- 🌐 What would be a suitable rank function for the while loop?
- 🌐 Will the program terminate at all (for every possible input)?

Note: if the Collatz conjecture is correct, the program will terminate.

-  I learned the Coffee Can and the Untangling Line Segments problems from Dijkstra's lectures.
-  The Coffee Can Problem also appeared in:
D. Gries. *The Science of Programming*, Springer-Verlag, 1981.
-  The Collatz Conjecture:
https://en.wikipedia.org/wiki/Collatz_conjecture