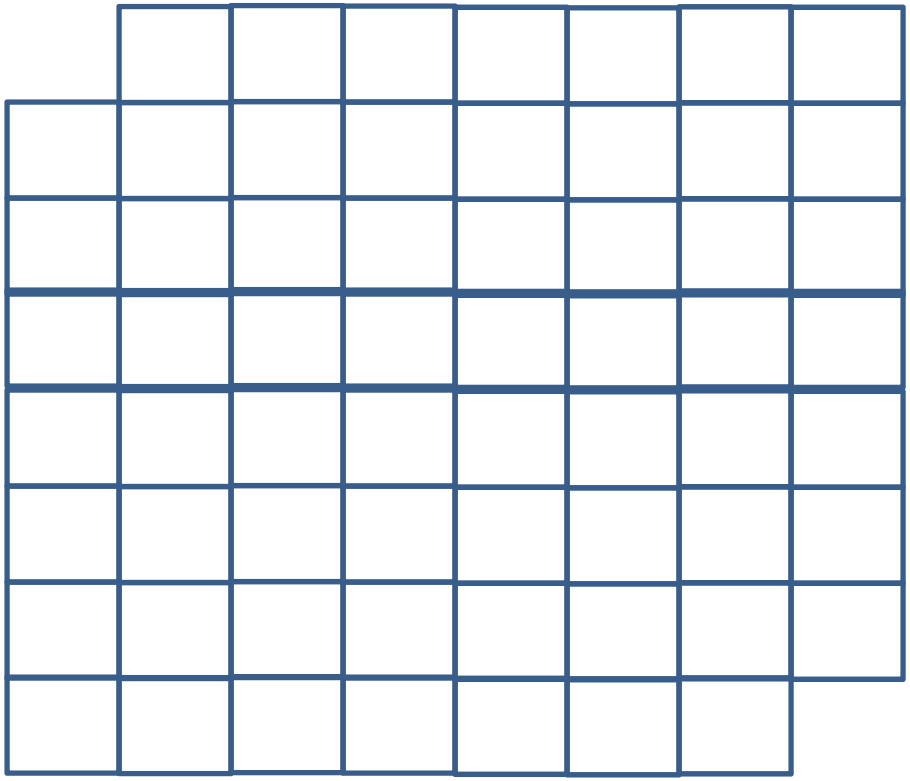
A puzzle

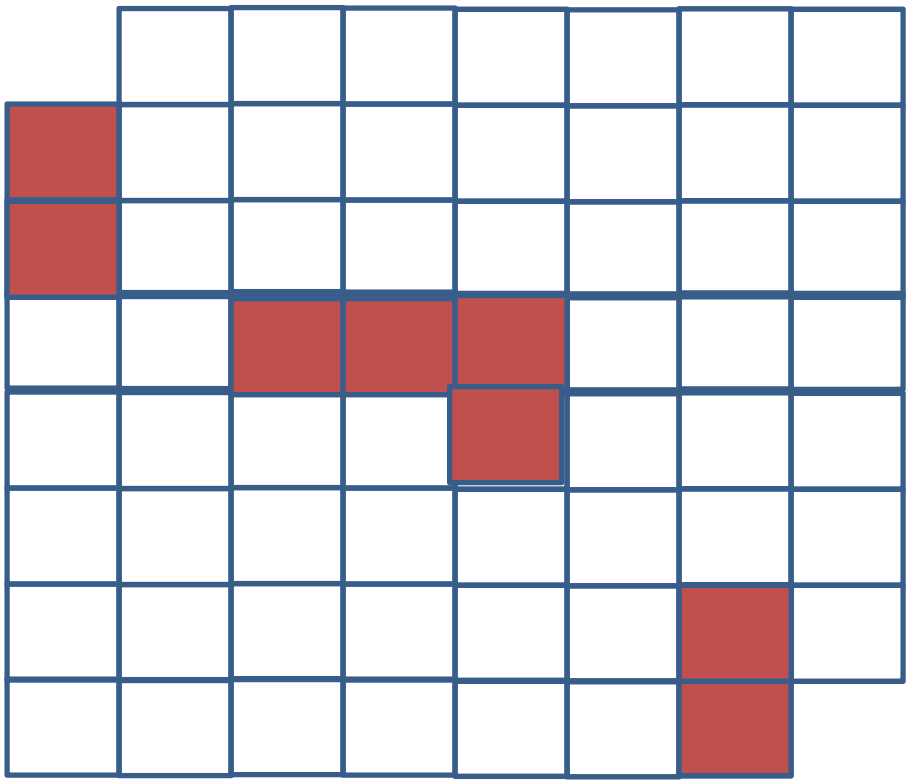
- A robot places 2x1 domino tiles on the following 8x8 checker board with top-left corner and bottom right-corner removed.
- Prove that it will never be the case that all spaces are covered.



Domino

A puzzle

- Example state reached



Domino

Proof

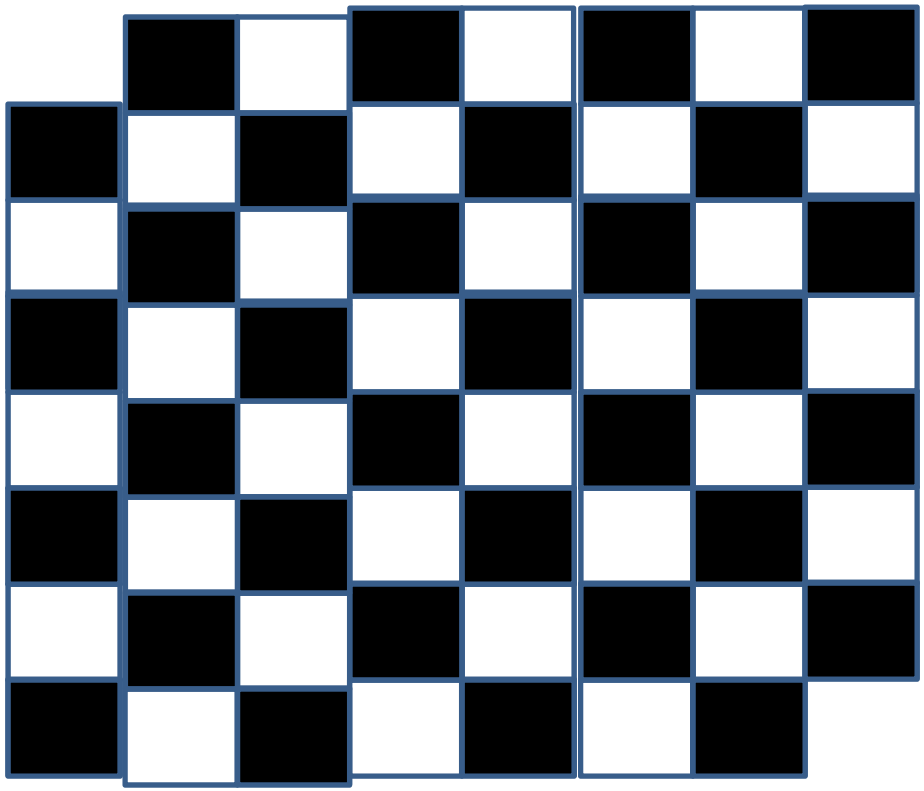
- Label the board with alternating black/white (like a chess board)
- Notice #blacks= #whites+2

  (since two white squares are removed)

- Invariant:

  No matter what state is reached, the number of white sq covered = number of black squares covered

- Hence all squares can never get covered



Proof

Initially, invariant holds.

From a state where the invariant holds, when a domino is placed, the new state will continue to satisfy the invariant.

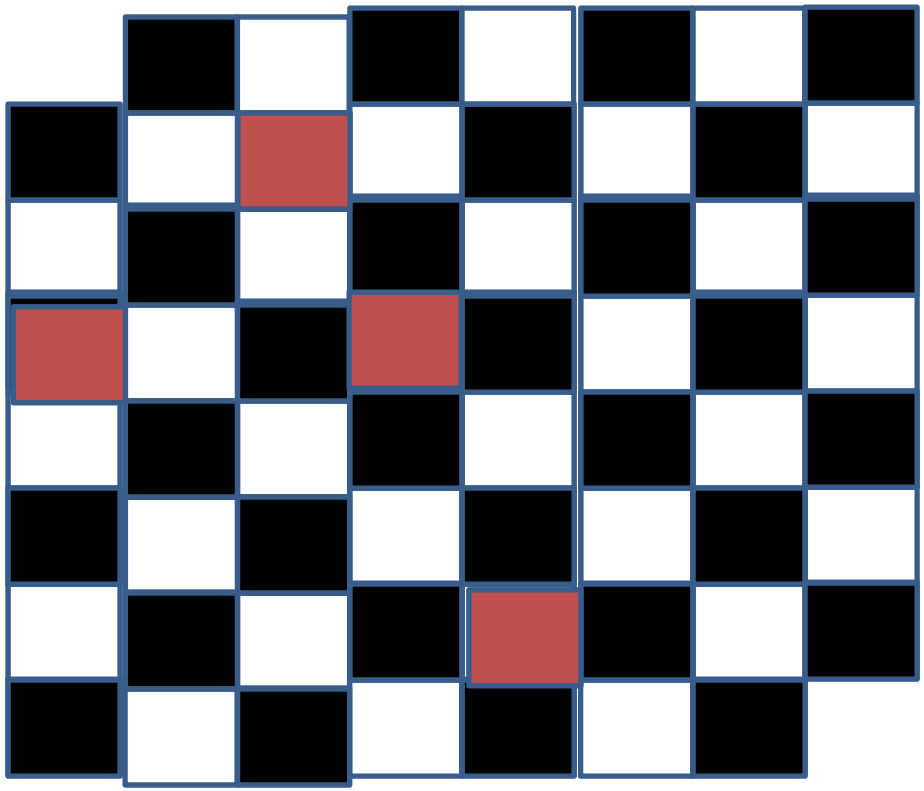
The state where all spaces are covered is not included in the invariant.

Hence, by induction on number of steps the robot takes, we can show that it will never reach the state where all spaces are covered (the invariant being the induction hypothesis).

Invariant need not be set of reachable states

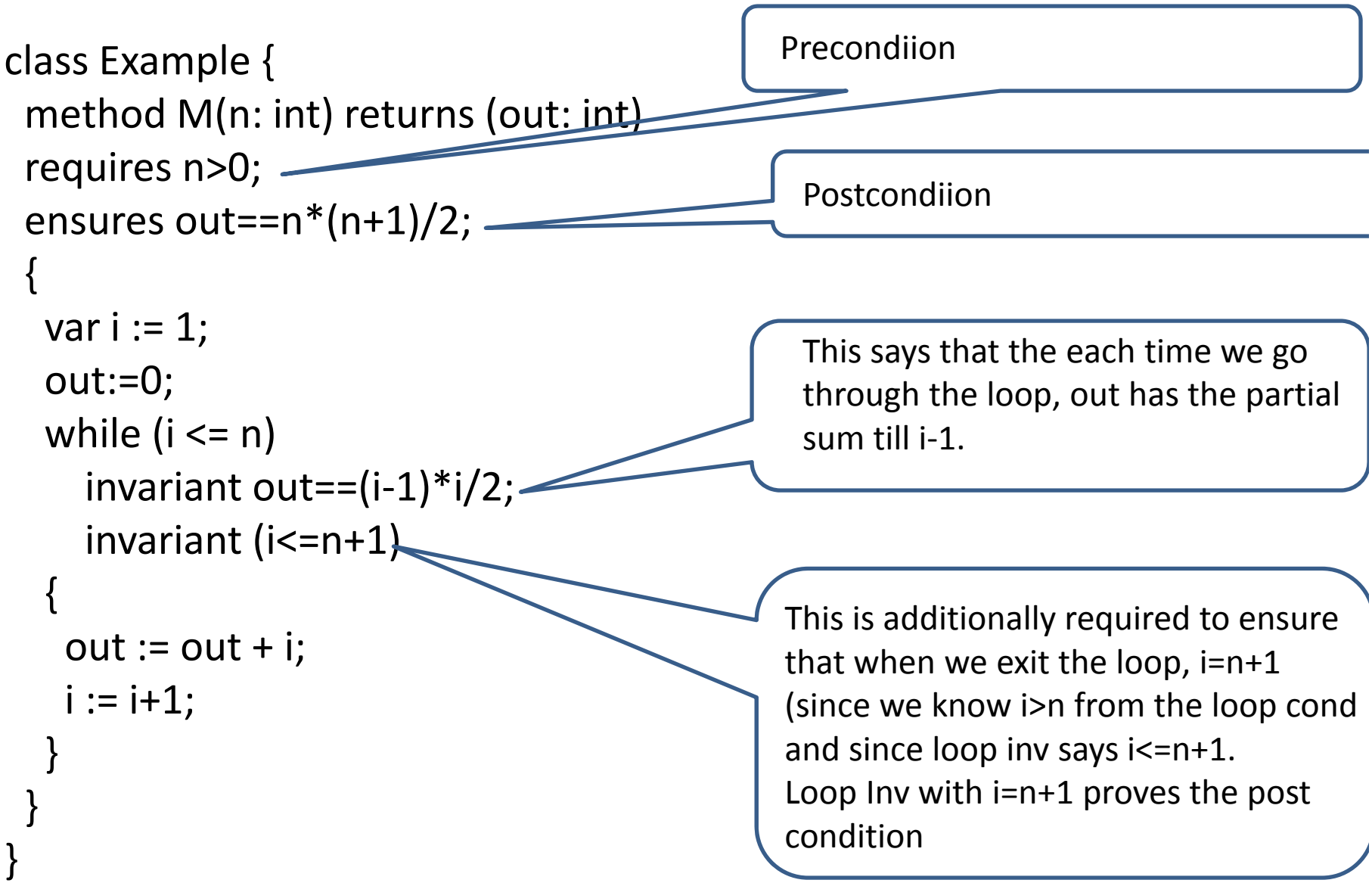
- Note that the invariant is not precisely the set of reachable states.

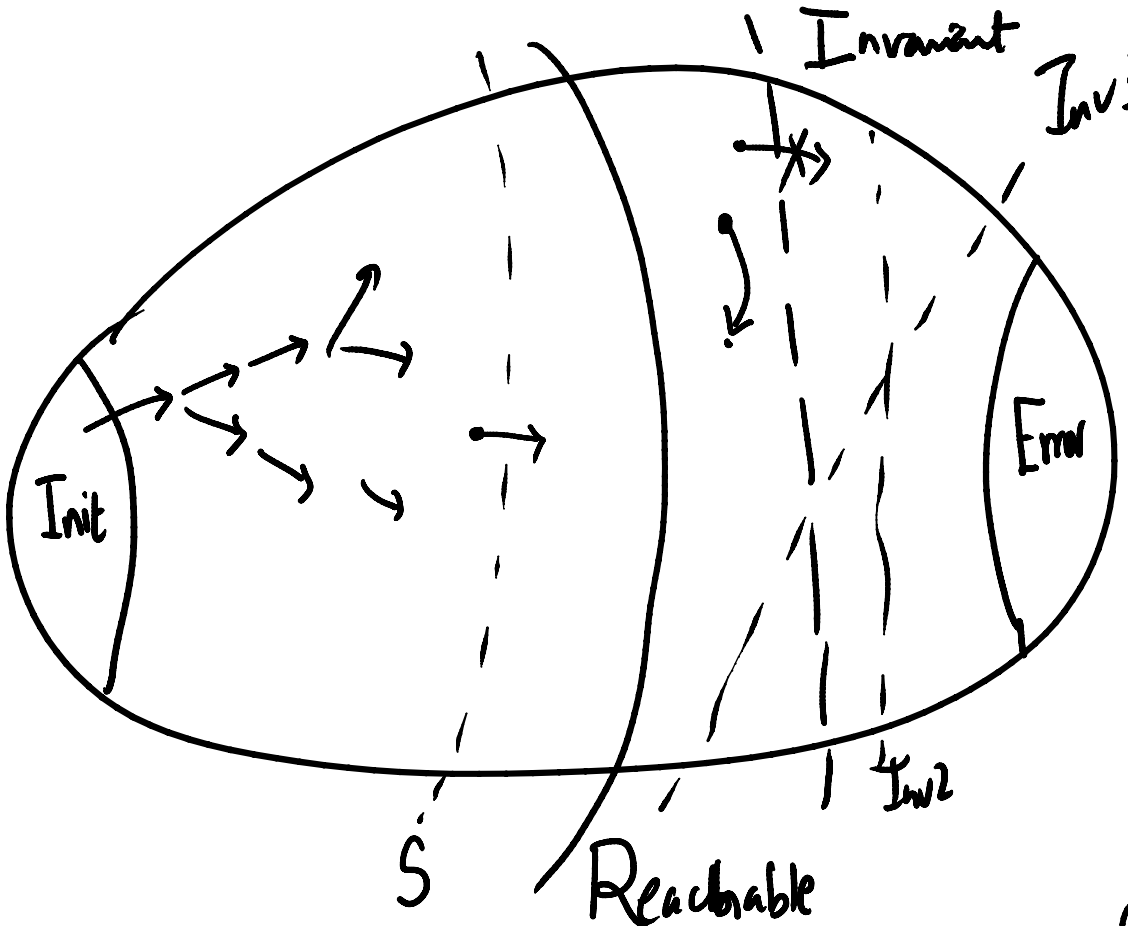
- E.g., the following is satisfied by the invariant but is not reachable



Example using Dafny

- DAFNY online:


class Example {
method M(n: int) returns (out: int)
requires n>0;
ensures out==n*(n+1)/2;
{
var i := 1;
out:=0;
while (i <= n)
invariant out==(i-1)*i/2;
invariant (i<=n+1)
{
out := out + i;
i := i+1;
}
}
}
Precondiion
Postcondiion
This says that the each time we go through the loop, out has the partial sum till i-1.
This is additionally required to ensure that when we exit the loop, i=n+1 (since we know i>n from the loop cond and since loop inv says i<=n+1.
Loop Inv with i=n+1 proves the post condition



$Reachable \cap Err = \phi$

$\underline{Init \subseteq Inv}$

$Inv \cap Err = \phi$

$\forall s \in Inv \not\vdash \left(R(s,s') \Rightarrow s' \in Inv \right)$

Key properties

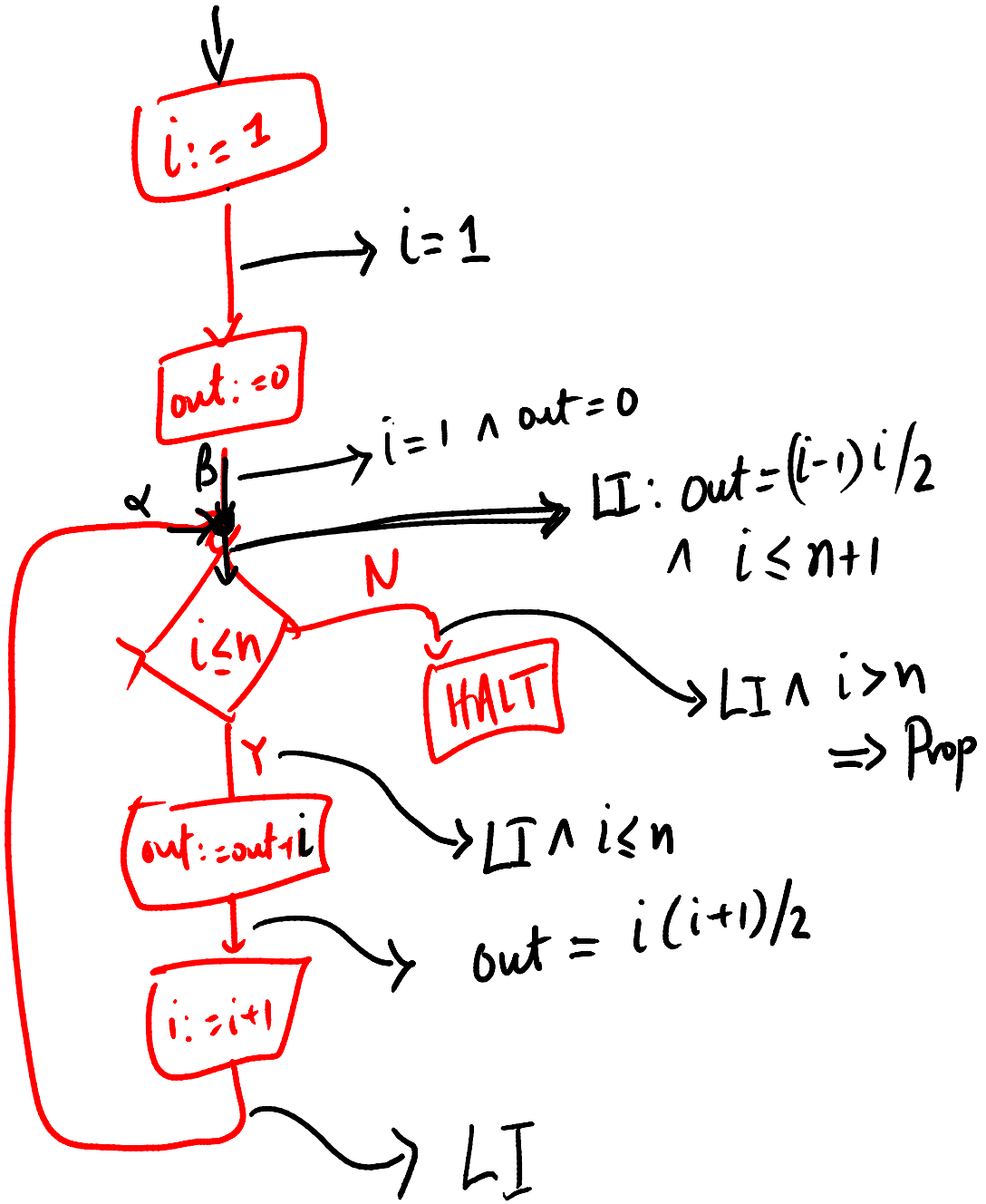
System: Init, Err, $R(\cdot,\cdot)$

Inductive Invariant: I

$$\begin{cases} Init \subseteq I \\ Err \cap I = \emptyset \\ \forall s \in I,\ R(s,s') \Rightarrow s' \in I \end{cases}$$

A program is correct if it has an inductive invariant.

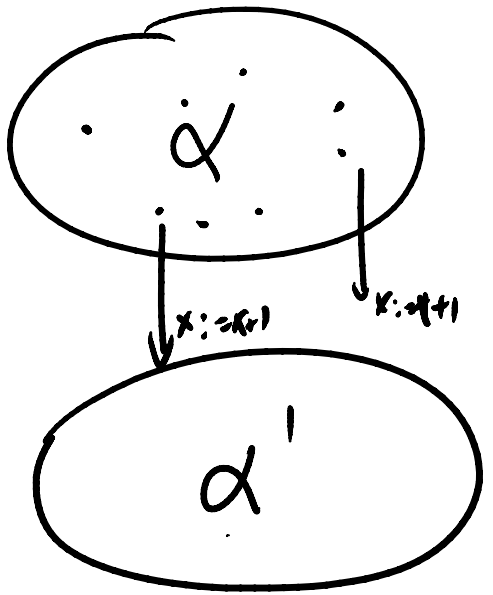
$Reach \subseteq I$ for every inductive invariant I.




```
Input : n
assume (n>0)
i := 1

Out := 0

while (i≤n) {

      Out := out+
                i;

      i := i+1;

}

assert( out =

          n*(n+1)
          ------
            2
        )
```

$\alpha \quad [x := x+1] \quad \gamma$

$\alpha' : sp(\alpha, x := x+1)$



$\alpha' \Rightarrow \gamma$

$\alpha \xrightarrow{R(\bar{x}, \bar{x}')} sp : \alpha'(x) \equiv \exists \bar{x}_0 (\alpha(\bar{x}_0) \wedge R(\bar{x}_0, x))$