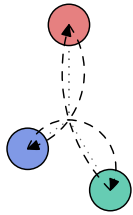


## **Dijkstra's Weakest Precondition**

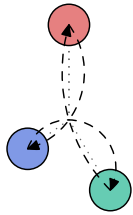
*Edsko de Vries*



## Introduction

### *Dijkstra's Weakest Precondition*

- We characterise a program state by listing the value of all variables in the program
- A *predicate* (or *condition*) is a function from program state to a boolean value
- We define the “goal” of the program as a predicate on its final state: the *postcondition*
- We are interested in the set of states  $I$ , such that when the program is started in a state  $i \in I$ , it is guaranteed to terminate in a state that meets the postcondition. This set is defined by the **weakest precondition** of the program and the postcondition.



## Notation

*Dijkstra's Weakest Precondition*

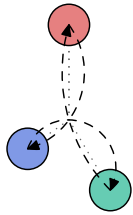
If the program is denoted by  $S$  (for System) and the post-condition is denoted by  $R$ , then the corresponding weakest precondition is given by

$$\mathbf{wp}(S, R)$$

Thus, if a program  $S$  is started in a state satisfying  $\mathbf{wp}(S, R)$ , it is guaranteed to terminate in a state satisfying  $R$ .

We introduce two constant predicates:

- All program states satisfy **true** (**T**)
- No program states satisfy **false** (**F**)



## Properties of **wp**

*Dijkstra's Weakest Precondition*

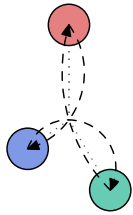
For any program  $S$ ,

$$\mathbf{wp}(S, F) = F \quad \text{Property (1)}$$

When a program  $S$  is started in a state satisfying  $\mathbf{wp}(S, F)$ , it will terminate in a state satisfying  $F$ .

However, no states satisfy  $F$ ; therefore, no states can satisfy  $\mathbf{wp}(S, F)$ .

This law is also called the “Law of the Excluded Miracle”. We will briefly come back to it later.



## Properties of **wp** (2)

*Dijkstra's Weakest Precondition*

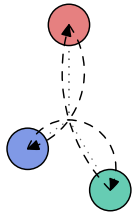
For any program  $S$  and postconditions  $Q, R$  s.t.  $Q \rightarrow R$ ,

$$\mathbf{wp}(S, Q) \rightarrow \mathbf{wp}(S, R) \quad \text{Property (2)}$$

When  $S$  is started in a state satisfying  $\mathbf{wp}(S, Q)$ , it will terminate in a state satisfying  $Q$ .

But  $Q \rightarrow R$ , so when  $S$  is started in a state satisfying  $\mathbf{wp}(S, Q)$  it will also terminate in a state satisfying  $R$ .

$\mathbf{wp}(S, R)$  is the weakest precondition for states that will result in  $R$ ; so the set of states that satisfy  $\mathbf{wp}(S, Q)$  must be a subset of the set of states that satisfy  $\mathbf{wp}(S, R)$ .



## Properties of **wp** (3)

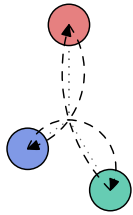
*Dijkstra's Weakest Precondition*

Properties (3) and (4) deal with conjunction (“and”) and disjunction (“or”) of **wp** respectively. I.e., for a program  $S$  and postconditions  $Q$  and  $R$ , we are interested in

$$\mathbf{wp}(S, Q) \wedge \mathbf{wp}(S, R) \quad (\text{conjunction})$$

and

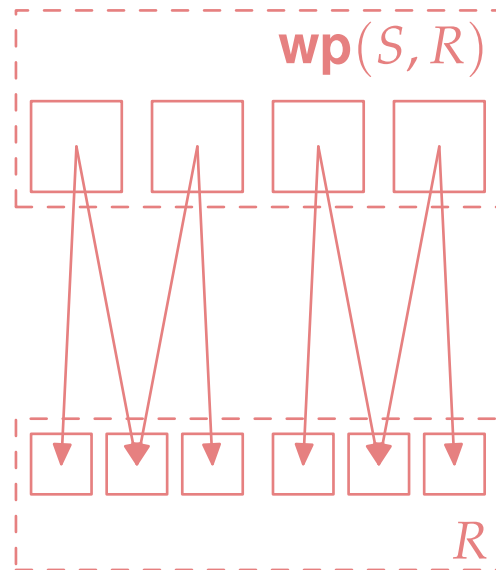
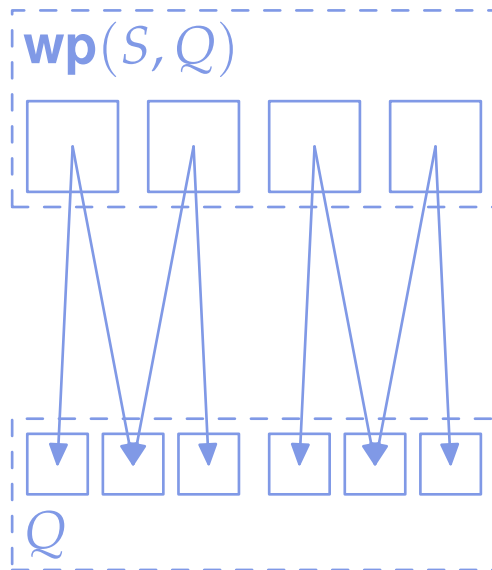
$$\mathbf{wp}(S, Q) \vee \mathbf{wp}(S, R) \quad (\text{disjunction})$$

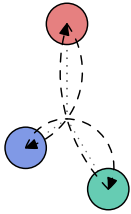


## Properties of $wp$ (4)

*Dijkstra's Weakest Precondition*

Every state (solid box) that satisfies  $wp(S, Q)$  (dashed box) is guaranteed to terminate in some state that satisfies  $Q$ .



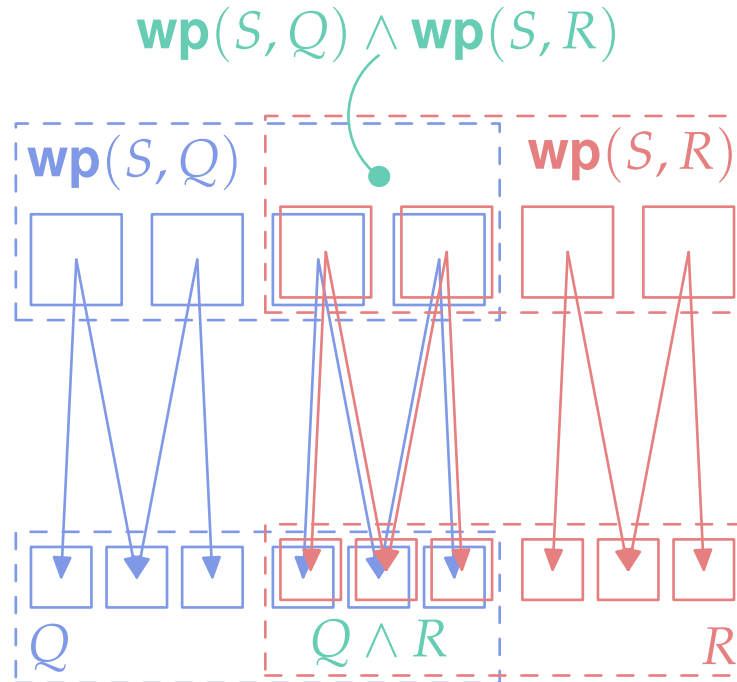


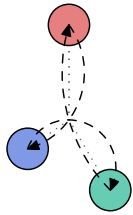
## Properties of **wp** (5)

*Dijkstra's Weakest Precondition*

$$\mathbf{wp}(S, Q) \wedge \mathbf{wp}(S, R) = \mathbf{wp}(S, Q \wedge R)$$

Property (3)



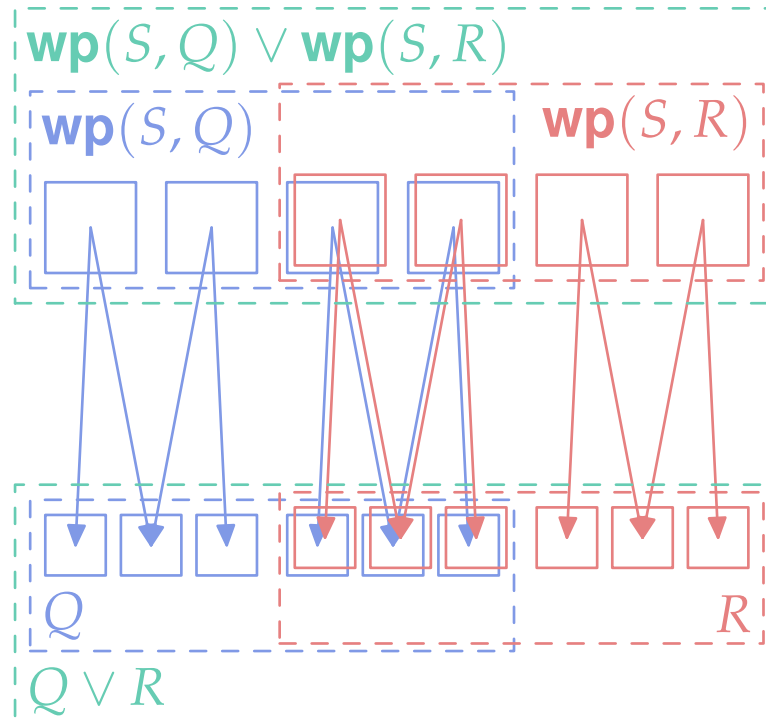


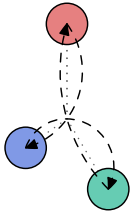
## Properties of **wp** (6)

*Dijkstra's Weakest Precondition*

$$\mathbf{wp}(S, Q) \vee \mathbf{wp}(S, R) \rightarrow \mathbf{wp}(S, Q \vee R)$$

Property (4)

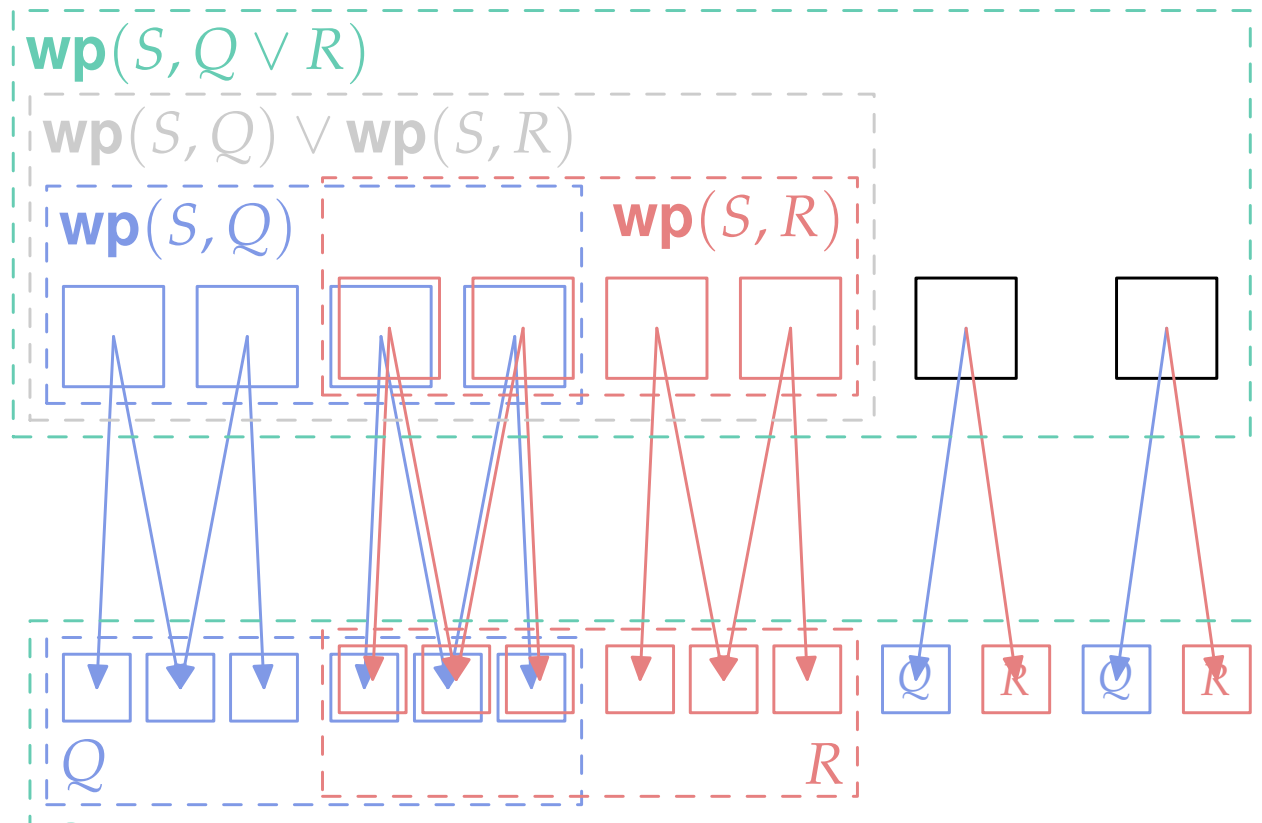


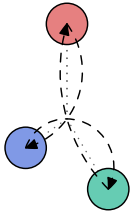


# Properties of wp (7)

*Dijkstra's Weakest Precondition*

$$\text{wp}(S, Q) \vee \text{wp}(S, R) \neq \text{wp}(S, Q \vee R)$$



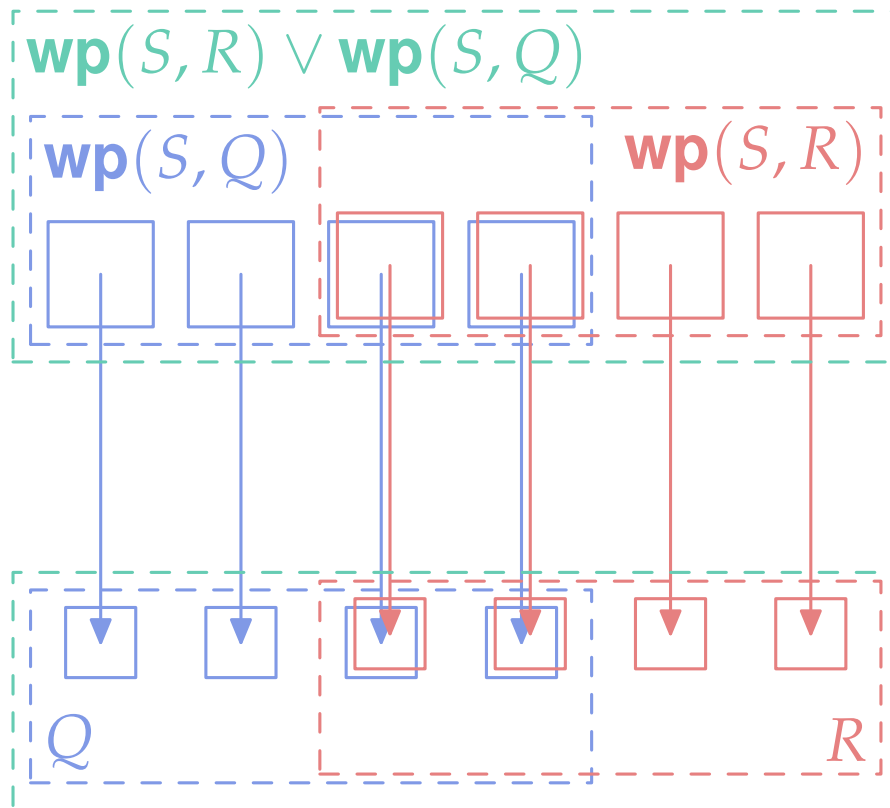


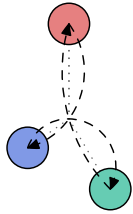
# Properties of wp (8)

*Dijkstra's Weakest Precondition*

$$\text{wp}(S, Q) \vee \text{wp}(S, R) = \text{wp}(S, Q \vee R)$$

Property (4') ( $S$  deterministic)





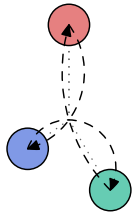
## The Derivation of $\text{wp}(S, R)$

*Dijkstra's Weakest Precondition*

The simplest program is one that does nothing. Let *Skip* be the null statement. Then

$$\text{wp}(\llbracket \text{Skip} \rrbracket, R) = R \quad (\text{skip})$$

In words,  $R$  will only hold after executing *Skip* when it held before executing *Skip*.



## The Derivation of $\text{wp}(S, R)$ (2)

*Dijkstra's Weakest Precondition*

Next we introduce two “programs” with a constant precondition (one that does not depend on the postcondition).

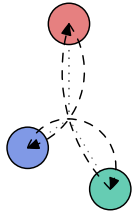
$$\text{wp}(\llbracket \text{Abort} \rrbracket, R) = \text{F} \quad (\text{abort})$$

By definition, *Abort* will fail to reach *any* state, so it cannot reach a state that satisfies any *R*.

The following is (explicitly) excluded by Dijkstra but is useful in a lattice theoretic framework:

$$\text{wp}(\llbracket \text{Miracle} \rrbracket, R) = \text{T} \quad (\text{miracle})$$

So, *Miracle* is guaranteed to terminate in a state that satisfies any postcondition you want, independent of the precondition.



## The Derivation of $\mathbf{wp}(S, R)$ (3)

*Dijkstra's Weakest Precondition*

Let  $R[x := E]$  denote  $R$  with  $E$  substituted for  $x$ . Then

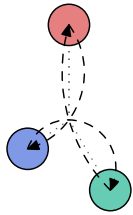
$$\mathbf{wp}(\llbracket x := E \rrbracket, R) = R[x := E] \quad (\text{assignment})$$

For example

$$\mathbf{wp}(\llbracket a := 7 \rrbracket, a = 7) = (7 = 7) = \mathbf{T}$$

$$\mathbf{wp}(\llbracket a := 7 \rrbracket, a = 6) = (7 = 6) = \mathbf{F}$$

So,  $a$  is always  $7$  after executing  $a := 7$ , and never  $6$ , as one would expect.



## The Derivation of $\mathbf{wp}(S, R)$ (4)

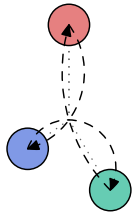
*Dijkstra's Weakest Precondition*

A few more examples

$$\mathbf{wp}(\llbracket a := 7 \rrbracket, b = c) = (b = c)$$

$$\mathbf{wp}(\llbracket a := 2 * b + 1 \rrbracket, a = 13) = (2 * b + 1 = 13) = (b = 6)$$

$$\mathbf{wp}(\llbracket a := a - b \rrbracket, a > b) = (a - b > b) = (a > 2 * b)$$



## The Derivation of $\text{wp}(S, R)$ (5)

*Dijkstra's Weakest Precondition*

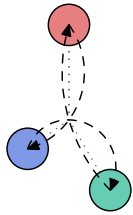
We must be able to compose multiple “atomic” instructions.

$$\text{wp}([S_1; S_2], R) = \text{wp}(S_1, \text{wp}(S_2, R)) \quad (\text{sequencing})$$

Intuitively, if  $S_1; S_2$  is to establish  $R$ , the last statement in the sequence ( $S_2$ ) should establish  $R$ , and the first statement in the sequence should “pave the way” for  $S_2$ .

But by definition  $S_2$  will be able to establish  $R$  if  $\text{wp}(S_2, R)$  holds; so this is exactly what  $S_1$  must establish.

Thus, the weakest precondition of  $S_1; S_2$  is the condition such that  $S_1$  is able to establish the weakest precondition of  $S_2$ . I.e.,  $S_1$  must be able to “pave the way” for  $S_2$ .



## The Derivation of $\mathbf{wp}(S, R)$ (6)

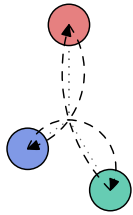
*Dijkstra's Weakest Precondition*

Example

$$\mathbf{wp}(\llbracket a := a + b; b := a * b \rrbracket, b > a) = \\ \mathbf{wp}(\llbracket a := a + b \rrbracket, \mathbf{wp}(\llbracket b := a * b \rrbracket, b > a))$$

$$\mathbf{wp}(\llbracket b := a * b \rrbracket, b > a) = (a * b > a) = \\ (b > 1 \wedge a \geq 0) \vee (b < 1 \wedge a \leq 0)$$

$$\mathbf{wp}(\llbracket a := a + b \rrbracket, (b > 1 \wedge a \geq 0) \vee (b < 1 \wedge a \leq 0)) = \\ (b > 1 \wedge (a + b) \geq 0) \vee (b < 1 \wedge (a + b) \leq 0) \\ = (b > 1 \wedge a \geq -b) \vee (b < 1 \wedge a \leq -b)$$



## The Derivation of $\text{wp}(S, R)$ (7)

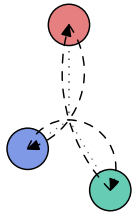
*Dijkstra's Weakest Precondition*

Two conditional statements:

$\text{if } B_1 \rightarrow S_1 \parallel B_2 \rightarrow S_2 \parallel \dots \parallel B_n \rightarrow S_n \text{ fi}$

and

$\text{do } B_1 \rightarrow S_1 \parallel B_2 \rightarrow S_2 \parallel \dots \parallel B_n \rightarrow S_n \text{ od}$



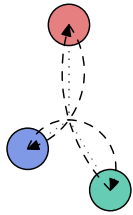
## The Derivation of $\mathbf{wp}(S, R)$ (8)

*Dijkstra's Weakest Precondition*

Let  $\mathbf{IF} = \text{if } B_1 \rightarrow S_1 \parallel B_2 \rightarrow S_2 \parallel \cdots \parallel B_n \rightarrow S_n \text{ fi}$ . Then

$$\mathbf{wp}(\mathbf{IF}, R) = \exists i \cdot B_i \wedge \forall i \cdot B_i \rightarrow \mathbf{wp}(S_i, R) \quad (\text{if})$$

Thus, at least one of the guards must be true, and the weakest precondition of all selected statements must be satisfied. (The if–statement will abort if all guards are false.)



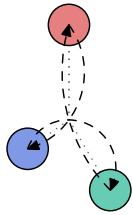
## The Derivation of $\text{wp}(S, R)$ (9)

*Dijkstra's Weakest Precondition*

Consider this program to calculate the maximum of three numbers<sup>1</sup>

```
if
   $x > y \wedge x > z \rightarrow \text{max} := x$ 
IF =  $y > x \wedge y > z \rightarrow \text{max} := y$ 
       $\neg((x > y \wedge x > z) \vee (y > x \wedge y > z)) \rightarrow \text{max} := z$ 
fi
```

(Example is due to Hugh Gibbons)



## The Derivation of $\mathbf{wp}(S, R)$ (10)

*Dijkstra's Weakest Precondition*

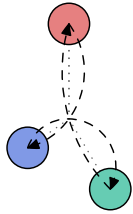
We use the postcondition  $R = \max \geq x \wedge \max \geq y \wedge \max \geq z$ . By definition,

$$\mathbf{wp}(\mathbf{IF}, R) = \exists i \cdot B_i \wedge \forall i \cdot B_i \rightarrow \mathbf{wp}(S_i, R)$$

$\exists i \cdot B_i$  is automatically satisfied because the conditions of  $\mathbf{IF}$  are exclusive. Left to show  $\forall i \cdot B_i \rightarrow R$ . For the first branch:

$$\begin{aligned} & x > y \wedge x > z \rightarrow \mathbf{wp}(\llbracket \max := x \rrbracket, \max \geq x \wedge \max \geq y \wedge \max \geq z) \\ \equiv & x > y \wedge x > z \rightarrow x \geq x \wedge x \geq y \wedge x \geq z \\ \equiv & x > y \wedge x > z \rightarrow x \geq y \wedge x \geq z \\ \equiv & \mathbf{T} \end{aligned}$$

The proof is similar for the second branch.

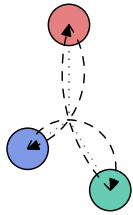


## The Derivation of $\text{wp}(S, R)$ (11)

*Dijkstra's Weakest Precondition*

The third branch is more interesting.

$$\begin{aligned}
 & \neg((x > y \wedge x > z) \vee (y > x \wedge y > z)) \rightarrow \\
 & \quad \text{wp}(\llbracket \text{max} := z \rrbracket, \text{max} \geq x \wedge \text{max} \geq y \wedge \text{max} \geq z) \\
 & \equiv \neg((x > y \wedge x > z) \vee (y > x \wedge y > z)) \rightarrow z \geq x \wedge z \geq y \\
 & \equiv \neg(x > y \wedge x > z) \wedge \neg(y > x \wedge y > z) \rightarrow z \geq x \wedge z \geq y \\
 & \equiv (x \leq y \vee x \leq z) \wedge (y \leq x \vee y \leq z) \rightarrow z \geq x \wedge z \geq y \\
 & \equiv ((x \leq y \vee x \leq z) \wedge y \leq x) \vee ((x \leq y \vee x \leq z) \wedge y \leq z) \rightarrow z \geq x \wedge z \geq y \\
 & \equiv (x = y) \vee (y \leq x \leq z) \vee (x \leq y \leq z) \vee (x \leq z \wedge y \leq z) \rightarrow z \geq x \wedge z \geq y \\
 & \equiv (x = y) \rightarrow (z \geq x \wedge z \geq y) \\
 & \equiv (x \neq y) \vee (z \geq x \wedge z \geq y) \\
 & \equiv (x \neq y) \vee (z \geq x)
 \end{aligned}$$



## The Derivation of $\text{wp}(S, R)$ (12)

*Dijkstra's Weakest Precondition*

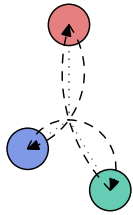
Thus, with  $\text{IF}$  defined as before,

```
if
     $x > y \wedge x > z \rightarrow \text{max} := x$ 
 $\text{IF} =$      $y > x \wedge y > z \rightarrow \text{max} := y$ 
     $\neg((x > y \wedge x > z) \vee (y > x \wedge y > z)) \rightarrow \text{max} := z$ 
fi
```

we have

$$\text{wp}(\text{IF}, \text{max} \geq x \wedge \text{max} \geq y \wedge \text{max} \geq z) = (x \neq y \vee z \geq x)$$

and the program fails if  $x = y \wedge z < x$ .



## The Derivation of $\mathbf{wp}(S, R)$ (13)

*Dijkstra's Weakest Precondition*

Let  $\mathbf{DO} = \mathbf{do} B_1 \rightarrow S_1 \parallel B_2 \rightarrow S_2 \parallel \cdots \parallel B_n \rightarrow S_n \mathbf{od}$ , and let  $\mathbf{IF}$  be the corresponding if-statement.

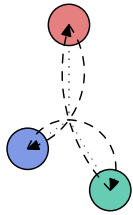
If  $H_k(R)$  is the weakest condition such that the loop will terminate in a state satisfying  $R$  after at most  $k$  iterations ( $k \geq 0$ ), then

$$\mathbf{wp}(\mathbf{DO}, R) = \exists k \cdot H_k(R) \quad (\mathbf{do})$$

We define  $H_k$  inductively.

$$H_0(R) = R \wedge \forall i \cdot \neg B_i$$

$$H_k(R) = \mathbf{wp}(\mathbf{IF}, H_{k-1}(R)) \vee H_0$$



## Invariant Properties

*Dijkstra's Weakest Precondition*

A predicate  $P$  is **invariant** in an if–statement **IF** if

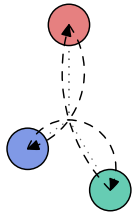
- $P$  holds before and after **IF**, and
- **IF** does not abort (at least one branch is selected)

i.e.,

$$P \wedge (\exists i \cdot B_i) \rightarrow \mathbf{wp}(\mathbf{IF}, P) \quad (\text{if-invariant})$$

This will hold if  $P$  is invariant in all selected branches:

$$\forall i \cdot (P \wedge B_i) \rightarrow \mathbf{wp}(S_j, P)$$



## Invariant Properties (2)

*Dijkstra's Weakest Precondition*

Let **DO** be a do-loop, and **IF** the corresponding if-statement.

**Theorem** (Fundamental Invariance Theorem for Loops)

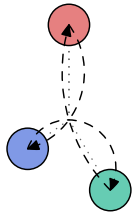
Let  $P$  be invariant in **IF**, i.e.

$$P \wedge (\exists i \cdot B_i) \rightarrow \mathbf{wp}(\mathbf{IF}, P)$$

Then for the corresponding **DO** construct, we can conclude

$$P \wedge \mathbf{wp}(\mathbf{DO}, T) \rightarrow \mathbf{wp}(\mathbf{DO}, P \wedge \forall i \cdot \neg B_i) \quad (\text{do-invariant})$$

(The condition  $\mathbf{wp}(\mathbf{DO}, T)$  forces the loop to terminate.)



## Loop Termination

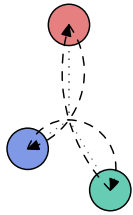
*Dijkstra's Weakest Precondition*

$wp(DO, T)$  is impossible to prove in the general case (halting problem).

The idea is to

- introduce a function  $t$  from program state to the set of integers, and
- to show that  $t$  is bounded below by some number and
- every execution of the loop reduces  $t$  by at least one.

This guarantees termination of the loop. We design our program in a way such that we can find a suitable function  $t$ .



## Loop Termination (2)

*Dijkstra's Weakest Precondition*

Let a property  $P$  be invariant in the loop. Let  $t$  be a function from program state to the set of integers, such that

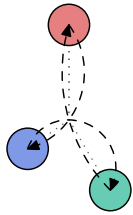
$$P \wedge (\exists i \cdot B_i) \rightarrow t > 0 \quad (t\text{-bounded})$$

Furthermore, for any value  $t_0$ ,

$$P \wedge (\exists i \cdot B_i) \wedge (t \leq t_0 + 1) \rightarrow \mathbf{wp}(\text{IF}, t \leq t_0) \quad (t\text{-decreasing})$$

This proves that  $P \wedge (t \leq k) \rightarrow \mathbf{wp}(H_k(\text{T}))$ , therefore  $P \rightarrow \mathbf{wp}(\text{DO}, \text{T})$ , and thus

$$P \rightarrow \mathbf{wp}(\text{DO}, P \wedge \forall i \cdot \neg B_i)$$



## Loop Termination (3)

*Dijkstra's Weakest Precondition*

To show ( $t$ -decreasing), we must show that every branch of the **DO** reduces  $t$ , i.e.

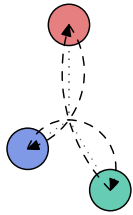
$$\forall i. P \wedge B_i \wedge (t \leq t_0 + 1) \rightarrow \mathbf{wp}(S_i, t \leq t_0)$$

Consider  $\mathbf{wp}(S_i, t \leq t_0)$ . Calculating this yields

$$\mathbf{wp}(S_i, t \leq t_0) = t' \leq t_0$$

where both  $t'$  and  $t$  are functions of the current state. We must show that  $t' \leq t - 1$ , i.e.,  $S_i$  decreases  $t$  by at least one. This is denoted by **wdec**:

$$\mathbf{wdec}(S_i, t) = t' \leq t - 1 \quad (\mathbf{wdec})$$



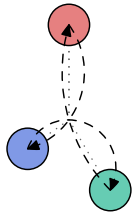
## Loop Example

*Dijkstra's Weakest Precondition*

As a final example, we consider a loop.

```
 $\tau, x, y := 0, 10, 10;$   
do  
   $\tau = 0 \wedge x > 0 \rightarrow \tau, x := 1, x - 1;$   
   $\tau = 1 \wedge y > 0 \rightarrow \tau, y := 0, y - 1;$   
od
```

Prove that  $\tau = x = y = 0$  when the loop terminates.



## Loop Example (2)

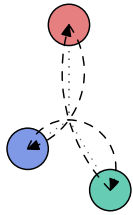
*Dijkstra's Weakest Precondition*

We need to find an invariant  $P$ .

```
 $\tau, x, y := 0, 10, 10;$   
do  
   $\tau = 0 \wedge x > 0 \rightarrow \tau, x := 1, x - 1;$   
   $\tau = 1 \wedge y > 0 \rightarrow \tau, y := 0, y - 1;$   
od
```

What is invariant throughout the program?

$$P \hat{=} (y - x = \tau) \wedge x \geq 0 \wedge y \geq 0$$

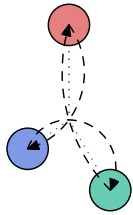


## Loop Example (3)

*Dijkstra's Weakest Precondition*

Let's verify that  $P$  holds after the initial assignment.

$$\begin{aligned} & \mathbf{wp}(\llbracket \tau, x, y := 0, 10, 10 \rrbracket, y - x = \tau \wedge x \geq 0 \wedge y \geq 0) \\ & \equiv 10 - 10 = 0 \wedge 10 \geq 0 \wedge 10 \geq 0 \\ & \equiv \mathbf{T} \end{aligned}$$



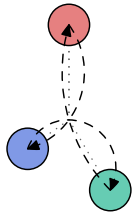
## Loop Example (4)

*Dijkstra's Weakest Precondition*

Now let's verify that  $P$  is invariant in the loop. For the first branch in the loop:

$$\begin{aligned} & (y - x = \tau \wedge x \geq 0 \wedge y \geq 0) \wedge \tau = 0 \wedge x > 0 \rightarrow \\ & \quad \mathbf{wp}(\llbracket \tau, x := 1, x - 1 \rrbracket, y - x = \tau \wedge x \geq 0 \wedge y \geq 0) \\ \equiv & (y - x = 0) \wedge x > 0 \wedge y \geq 0 \rightarrow \\ & \quad (y - (x - 1) = 1) \wedge (x - 1) \geq 0 \wedge y \geq 0 \\ \equiv & \mathbf{T} \end{aligned}$$

By a similar argument,  $P$  is also invariant in the second branch in the loop. Thus,  $P$  is invariant throughout the program.



## Loop Example (5)

*Dijkstra's Weakest Precondition*

To prove termination, we need to choose an appropriate  $t$ .

$\tau, x, y := 0, 10, 10;$

do

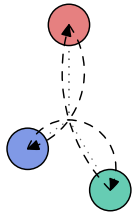
$\tau = 0 \wedge x > 0 \rightarrow \tau, x := 1, x - 1;$

$\tau = 1 \wedge y > 0 \rightarrow \tau, y := 0, y - 1;$

od

What would be a suitable function?

$$t \hat{=} x + y$$



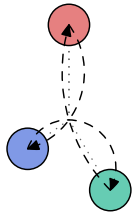
## Loop Example (6)

*Dijkstra's Weakest Precondition*

To show that  $t = x + y$  is a suitable choice, we need to show ( $t$ -bounded) and ( $t$ -decreasing). First, we show ( $t$ -bounded):

$$(y - x = \tau \wedge x \geq 0 \wedge y \geq 0) \wedge \tau = 0 \wedge x > 0 \rightarrow (x + y) > 0 \\ \equiv \text{T}$$

$$(y - x = \tau \wedge x \geq 0 \wedge y \geq 0) \wedge \tau = 1 \wedge y > 0 \rightarrow (x + y) > 0 \\ \equiv \text{T}$$



## Loop Example (7)

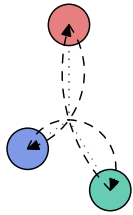
*Dijkstra's Weakest Precondition*

To show ( $t$ -decreasing), we have to show that every branch of the do-loop reduces  $t$ . We have

$$\mathbf{wp}(\llbracket t, x := 1, x - 1 \rrbracket, x + y < t_0) = ((x - 1) + y) < t_0$$

$$\mathbf{wdec}(\llbracket t, x := 1, x - 1 \rrbracket, x + y) = ((x - 1) + y) \leq (x + y) - 1 = T$$

By a similar argument, the second branch of the do-loop also decreases  $t$ . So, we have proven termination.



## Loop Example (8)

*Dijkstra's Weakest Precondition*

So, now we know that  $P \rightarrow \mathbf{wp}(\mathbf{DO}, P \wedge \forall i \cdot \neg B_i)$ . We know that  $P$  holds before the do-loop, and we can conclude that  $P \wedge \forall i \cdot \neg B_i$  holds when the program terminates. To be accurate, we know

$$\neg(t = 0 \wedge x > 0) \wedge \neg(t = 1 \wedge y > 0) \wedge (y - x = \tau \wedge x \geq 0 \wedge y \geq 0)$$

There are only two possibilities for  $\tau$ ; either  $\tau = 0$  or  $\tau = 1$ . We cannot have  $\tau = 1$ , because then  $y \leq 0$ ,  $y - x = 1$ , so  $x < 0$  which contradicts the invariant.

So,  $\tau = 0$ , therefore  $x \leq 0$ . But  $x \leq 0 \wedge x \geq 0 \rightarrow x = 0$ . We know  $y - x = \tau$ ;  $y - 0 = 0$ , so  $y = 0$ . ■