Chair of Software Modeling and Verification (Computer Science 2)

RWTH AACHEN UNIVERSITY

Weakest Precondition Reasoning and Its Generalizations

Emma Ahrens, Christina Gehnen
RWTH Aachen University
05.03.2026

About us

Emma Ahrens
ahrens@cs.rwth-aachen.de

- PhD student at RWTH Aachen University, Germany
- Group: Software Modeling and Verification
- Research interests: program verification, weighted programming

Christina Gehnen
christina.gehnen@cs.rwth-aachen.de

- PhD student at RWTH Aachen University, Germany
- Group: Software Modeling and Verification & Quantum Information Systems
- Research interests: program verification, quantum computing

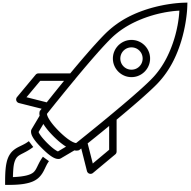
Structure

1. Classical programs - Boolean predicates
2. Probabilistic programs - Quantitative predicates
3. Probabilistic programs - Expectations
4. Break
5. Weighted programs - Expectations
6. Quantum programs - Observables

Example: Rocket Landing

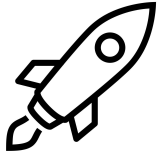
We need to check some things before landing

- Fuel level
- Engine temperature
- Fuel pressure
- ...



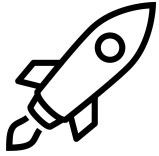
Program for Rocket Landing

```
while (fuel > 100) {
    fuel := fuel − 10
};
if (engineTemp > 900){
    engineOn := False
}
else {
    engineOn := True
};
fuelPressure := fuelPressure + 50
```



Program for Rocket Landing

```
while (fuel > 100) {
    fuel := fuel − 10
};
if (engineTemp > 900){
    engineOn := False
}
else {
    engineOn := True
};
fuelPressure := fuelPressure + 50
```



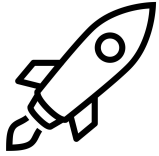
Postcondition:

```
// [fuelPressure > 500 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

Program for Rocket Landing

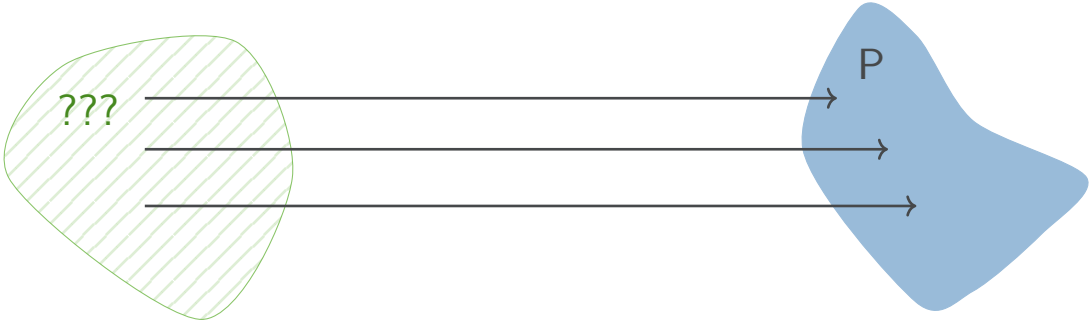
Precondition:

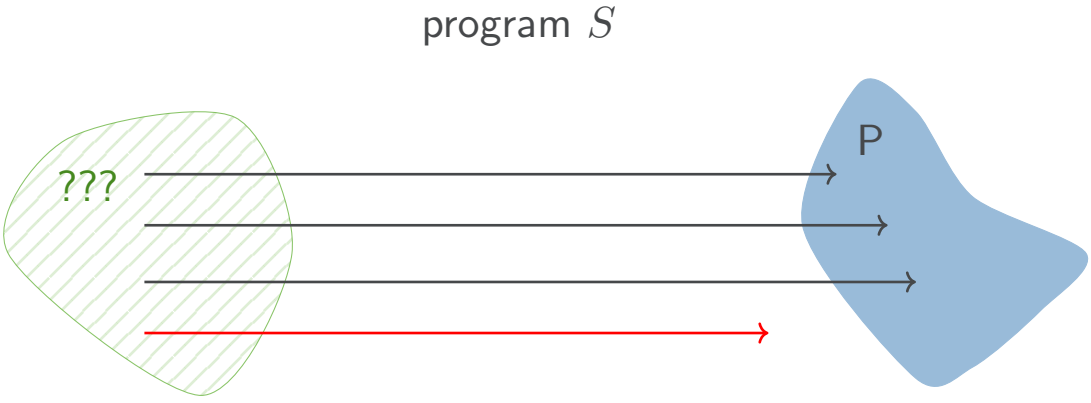
```
// ?

while (fuel > 100) {
    fuel := fuel − 10
};
if (engineTemp > 900){
    engineOn := False
}
else {
    engineOn := True
};
fuelPressure := fuelPressure + 50
```

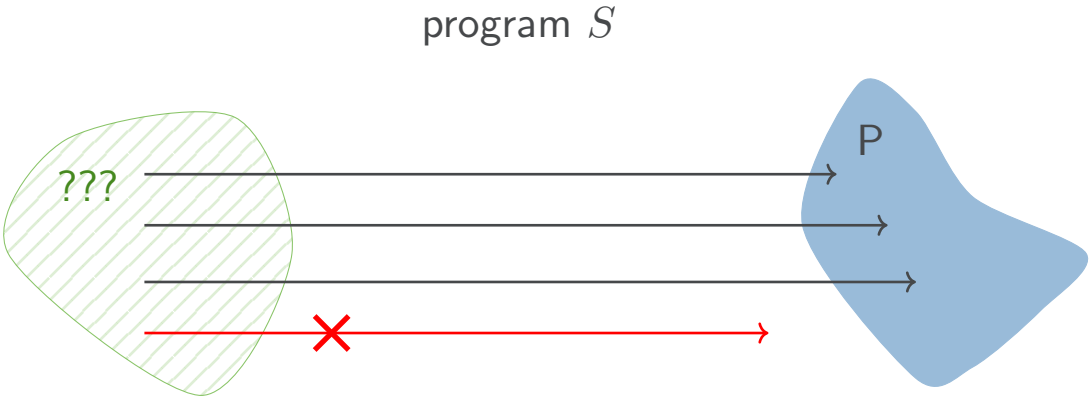


Postcondition:

```
// [fuelPressure > 500 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```


program S
???
P


program S
???
P


program S
???
P

Recall example program for rocket landing

```
fuelPressure := fuelPressure + 50
```

Recall example program for rocket landing

```
fuelPressure := fuelPressure + 50
// [fuelPressure > 500 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

Recall example program for rocket landing

What property needs to hold before program execution to ensure the postcondition?

```
fuelPressure := fuelPressure + 50
```

```
// [fuelPressure > 500 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

Recall example program for rocket landing

What property needs to hold before program execution to ensure the postcondition?

$$// \ [\mathit{fuelPressure} > 450 \land \mathit{engineOn} = \mathit{True} \land 50 \leq \mathit{fuel} \leq 100]$$
$$// \ [\mathit{fuelPressure} + 50 > 500 \land \mathit{engineOn} = \mathit{True} \land 50 \leq \mathit{fuel} \leq 100]$$

$$\mathit{fuelPressure} := \mathit{fuelPressure} + 50$$

$$// \ [\mathit{fuelPressure} > 500 \land \mathit{engineOn} = \mathit{True} \land 50 \leq \mathit{fuel} \leq 100]$$

```
if (engineTemp > 900){

    engineOn := False

}
else {

    engineOn := True

};
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

```
if (engineTemp > 900){

    engineOn := False
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
}
else {

    engineOn := True
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
};
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

```
if (engineTemp > 900){

    engineOn := False
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
}
else {
// [fuelPressure > 450 ∧ True = True ∧ 50 ≤ fuel ≤ 100]
    engineOn := True
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
};
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

```
if (engineTemp > 900){
// [fuelPressure > 450 ∧ False = True ∧ 50 ≤ fuel ≤ 100]
    engineOn := False
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
}
else {
// [fuelPressure > 450 ∧ True = True ∧ 50 ≤ fuel ≤ 100]
    engineOn := True
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
};
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

```
// [engineTemp ≤ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel ≤ 100]
// [(engineTemp > 900 ∧ False) ∨
   (engineTemp ≯ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel ≤ 100)]
if (engineTemp > 900){
// [fuelPressure > 450 ∧ False = True ∧ 50 ≤ fuel ≤ 100]
    engineOn := False
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
}
else {
// [fuelPressure > 450 ∧ True = True ∧ 50 ≤ fuel ≤ 100]
    engineOn := True
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
};
// [fuelPressure > 450 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

Weakest Preconditions

program S	$\texttt{wp}[\![S]\!](P)$
$x \coloneqq E$	$P[x/E]$
$\texttt{if}\ (\varphi)\ \{\ S_1\ \}\ \texttt{else}\ \{\ S_2\ \}$	$(\varphi \wedge \texttt{wp}[\![S_1]\!](P)) \vee (\neg\varphi \wedge \texttt{wp}[\![S_2]\!](P))$
$S_1\ ;\ S_2$	$\texttt{wp}[\![S_1]\!](\texttt{wp}[\![S_2]\!](P))$
$\texttt{while}\ (\varphi)\ \{S\}$	?

Loops

```
while (fuel > 100) {
    fuel := fuel − 10
}
```

```
// [engineTemp ≤ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel ≤ 100]
```

Loops

```
while (fuel > 100) {
    fuel := fuel − 10
}
```

```
// [50 ≤ fuel ≤ 100]
```

```
// [engineTemp ≤ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel ≤ 100]
```

Loops

```
while (fuel > 100) {
    fuel := fuel − 10
}
```

$// \ [50 \leq fuel \leq 100]$

$// \ [engineTemp \leq 900 \land fuelPressure > 450 \land 50 \leq fuel \leq 100]$

Loops

```
// [50 ≤ fuel]

while (fuel > 100) {
    fuel := fuel − 10
}

// [50 ≤ fuel ≤ 100]

// [engineTemp ≤ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel ≤ 100]
```

Loops

```
// [50 ≤ fuel]

while (fuel > 100) {
    fuel := fuel − 10
}

// [50 ≤ fuel ≤ 100]

// [engineTemp ≤ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel ≤ 100]
```

How can we compute it?

Loops

```
// [50 ≤ fuel]

while (fuel > 100) {
    fuel := fuel − 10
}

// [50 ≤ fuel ≤ 100]

// [engineTemp ≤ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel ≤ 100]
```

How can we compute it?

Idea: Unroll loop backwards,

$\Phi(X) = [50 \leq fuel \leq 100] \vee ([fuel > 100] \wedge \mathtt{wp}[\![fuel := fuel - 10]\!](X))$

Loops

```
// [50 ≤ fuel]

while (fuel > 100) {
    fuel := fuel − 10
}

// [50 ≤ fuel ≤ 100]

// [engineTemp ≤ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel ≤ 100]
```

How can we compute it?
Idea: Unroll loop backwards,
$\Phi(X) = [50 \leq fuel \leq 100] \vee ([fuel > 100] \wedge \texttt{wp}[\![fuel := fuel - 10]\!](X))$
generally: $\Phi(X) = (\neg\varphi \wedge P) \vee (\varphi \wedge \texttt{wp}[\![S]\!](X))$

$\Phi^n(X) \equiv$ wp of n-times executing the loop on the empty set of states

$$\Phi(X) = [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel \coloneqq fuel - 10]\!](X))]$$

$$\Phi(X) = [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel \coloneqq fuel - 10]\!](X))]$$

$$\begin{aligned}\Phi^1(\mathsf{False}) =& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel \coloneqq fuel - 10]\!](\mathsf{False}))] \\ =& [50 \leq fuel \leq 100]\end{aligned}$$

$$\Phi(X) = [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!](X))]$$

$$\begin{aligned}
\Phi^1(\mathsf{False}) =& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!](\mathsf{False}))] \\
=& [50 \leq fuel \leq 100] \\
\Phi^2(\mathsf{False}) =& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!]([50 \leq fuel \leq 100]))] \\
=& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge [60 \leq fuel \leq 110])] \\
=& [50 \leq fuel \leq 110]
\end{aligned}$$

$$\Phi(X) = [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel := fuel - 10]\!](X))]$$

$$\begin{aligned}
\Phi^1(\mathsf{False}) =& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel := fuel - 10]\!](\mathsf{False}))] \\
=& [50 \leq fuel \leq 100] \\
\Phi^2(\mathsf{False}) =& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel := fuel - 10]\!]([50 \leq fuel \leq 100]))] \\
=& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge [60 \leq fuel \leq 110])] \\
=& [50 \leq fuel \leq 110] \\
\Phi^3(\mathsf{False}) =& [50 \leq fuel \leq 110 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel := fuel - 10]\!]([50 \leq fuel \leq 110]))] \\
=& [50 \leq fuel \leq 120] \\
& \ldots
\end{aligned}$$

$$\Phi(X) = [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!](X))]$$

$$\begin{aligned}
\Phi^1(\mathsf{False}) &= [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!](\mathsf{False}))] \\
&= [50 \leq fuel \leq 100] \\
\Phi^2(\mathsf{False}) &= [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!]([50 \leq fuel \leq 100]))] \\
&= [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge [60 \leq fuel \leq 110])] \\
&= [50 \leq fuel \leq 110] \\
\Phi^3(\mathsf{False}) &= [50 \leq fuel \leq 110 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!]([50 \leq fuel \leq 110]))] \\
&= [50 \leq fuel \leq 120] \\
&\ldots \\
\Phi^n(\mathsf{False}) &= [50 \leq fuel \leq 100 + 10(n-1)]
\end{aligned}$$

$$\Phi(X) = [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel := fuel - 10]\!](X))]$$

$$\begin{aligned}
\Phi^1(\mathsf{False}) &= [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel := fuel - 10]\!](\mathsf{False}))] \\
&= [50 \leq fuel \leq 100] \\
\Phi^2(\mathsf{False}) &= [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel := fuel - 10]\!]([50 \leq fuel \leq 100]))] \\
&= [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge [60 \leq fuel \leq 110])] \\
&= [50 \leq fuel \leq 110] \\
\Phi^3(\mathsf{False}) &= [50 \leq fuel \leq 110 \vee (fuel > 100 \wedge \mathtt{wp}[\![fuel := fuel - 10]\!]([50 \leq fuel \leq 110]))] \\
&= [50 \leq fuel \leq 120] \\
&\ldots \\
\Phi^n(\mathsf{False}) &= [50 \leq fuel \leq 100 + 10(n-1)]
\end{aligned}$$

$$\sup_n \Phi^n(\mathsf{False}) = [50 \leq fuel] = \mathrm{lfp}\, X.\Phi(X)$$

$$\Phi(X) = [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!](X))]$$

$$\begin{aligned}
\Phi^1(\mathsf{False}) =& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!](\mathsf{False}))] \\
=& [50 \leq fuel \leq 100] \\
\Phi^2(\mathsf{False}) =& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!]([50 \leq fuel \leq 100]))] \\
=& [50 \leq fuel \leq 100 \vee (fuel > 100 \wedge [60 \leq fuel \leq 110])] \\
=& [50 \leq fuel \leq 110] \\
\Phi^3(\mathsf{False}) =& [50 \leq fuel \leq 110 \vee (fuel > 100 \wedge \texttt{wp}[\![fuel := fuel - 10]\!]([50 \leq fuel \leq 110]))] \\
=& [50 \leq fuel \leq 120] \\
& \ldots \\
\Phi^n(\mathsf{False}) =& [50 \leq fuel \leq 100 + 10(n-1)]
\end{aligned}$$

$$\sup_n \Phi^n(\mathsf{False}) = [50 \leq fuel] = \mathrm{lfp}\, X.\Phi(X)$$

$$\texttt{wp}[\![\texttt{while } (\varphi)\ \{S\}]\!](P) = \mathrm{lfp}\, X.(\varphi \wedge \texttt{wp}[\![S]\!](X)) \vee (\neg\varphi \wedge P)$$

Weakest Preconditions

program S	$\texttt{wp}[\![S]\!](P)$
$x \coloneqq E$	$P[x/E]$
$\texttt{if}\ (\varphi)\ \{\ S_1\ \}\ \texttt{else}\ \{\ S_2\ \}$	$(\varphi \wedge \texttt{wp}[\![S_1]\!](P)) \vee (\neg\varphi \wedge \texttt{wp}[\![S_2]\!](P))$
$S_1\ ;\ S_2$	$\texttt{wp}[\![S_1]\!](\texttt{wp}[\![S_2]\!](P))$
$\texttt{while}\ (\varphi)\ \{S\}$	$\mathrm{lfp}\, X.(\varphi \wedge \texttt{wp}[\![S]\!](X)) \vee (\neg\varphi \wedge P)$

Caesar

- Tool for deductive verification of (probabilistic) programs by RWTH Aachen University
- Given a program S, a precondition Q and a postcondition P, compute $\texttt{wp}[\![S]\!](P)$ and check if it implies a given precondition Q, i.e. if Q is a valid precondition

Caesar

- Tool for deductive verification of (probabilistic) programs by RWTH Aachen University
- Given a program S, a precondition Q and a postcondition P, compute $\texttt{wp}[\![S]\!](P)$ and check if it implies a given precondition Q, i.e. if Q is a valid precondition

We want to use it to verify that our computations are correct.

1. Install Caesar
   https://www.caesarverifier.org (easiest using VSCode)

Caesar

- Tool for deductive verification of (probabilistic) programs by RWTH Aachen University
- Given a program S, a precondition Q and a postcondition P, compute $\texttt{wp}[\![S]\!](P)$ and check if it implies a given precondition Q, i.e. if Q is a valid precondition

We want to use it to verify that our computations are correct.

1. Install Caesar
   https://www.caesarverifier.org (easiest using VSCode)
2. Open rocket_landing_bool.heyvl
   https://rwth-aachen.sciebo.de/s/aKYC3EBGy7TPoPX

Caesar

caesar > rocket_landing_bool.heyvl

```
1  coproc rocket(init_fuel: UInt, init_engineTemp: UInt, init_fuelPressure: UInt) -> (fuel:
   UInt, engineTemp: UInt, fuelPressure: UInt, engineOn: Bool)
2      pre [50<=init_fuel && init_fuelPressure > 450 && init_engineTemp <=900]
3      post [fuelPressure > 500 && engineOn == true && 50 <= fuel && fuel <= 100]
4  {
5      fuel = init_fuel
6      engineTemp = init_engineTemp
7      fuelPressure = init_fuelPressure
8      @invariant([50<=fuel && fuelPressure > 450 &&  engineTemp <=900])
9      while fuel > 100 {
10         fuel = fuel-10
11     }
12     if engineTemp > 900 {
13             engineOn = false
14     } else {
15         engineOn = true
16     }
17     fuelPressure = fuelPressure + 50
18 }
```

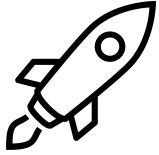
Caesar

caesar > rocket_landing_bool.heyvl

```
coproc rocket(init_fuel: UInt, init_engineTemp: UInt, init_fuelPressure: UInt) -> (fuel: UInt, engineTemp: UInt,
fuelPressure: UInt, engineOn: Bool)
    pre [50<=init_fuel && init_fuelPressure > 450 && init_engineTemp <=900]
    post [fuelPressure > 500 && engineOn == true && 50 <= fuel && fuel <= 100]
{
    ▷ [((50 <= init_fuel) && ((init_fuelPressure > 450) && (init_engineTemp <= 900)))]
    fuel = init_fuel
    engineTemp = init_engineTemp
    fuelPressure = init_fuelPressure
    ▷ [((50 <= fuel) && ((fuelPressure > 450) && (engineTemp <= 900)))]
    @invariant([50<=fuel && fuelPressure > 450 &&  engineTemp <=900])
    while fuel > 100 {
        ▷ [((50 <= (fuel − 10)) && ((fuelPressure > 450) && (engineTemp <= 900)))]
        fuel = fuel−10
        ▷ [((50 <= fuel) && ((fuelPressure > 450) && (engineTemp <= 900)))]
    }
    ▷ ite((engineTemp > 900), [((((fuelPressure + 50) > 500) && ((false == true) && ((50 <= fuel) && (fuel <= 100))))], [(
    if engineTemp > 900 {
            ▷ [((((fuelPressure + 50) > 500) && ((false == true) && ((50 <= fuel) && (fuel <= 100))))]
            engineOn = false
            ▷ [((((fuelPressure + 50) > 500) && ((engineOn == true) && ((50 <= fuel) && (fuel <= 100))))]
        } else {
            ▷ [((((fuelPressure + 50) > 500) && ((true == true) && ((50 <= fuel) && (fuel <= 100))))]
            engineOn = true
            ▷ [((((fuelPressure + 50) > 500) && ((engineOn == true) && ((50 <= fuel) && (fuel <= 100))))]
        }
    ▷ [((((fuelPressure + 50) > 500) && ((engineOn == true) && ((50 <= fuel) && (fuel <= 100))))]
    fuelPressure = fuelPressure + 50
    ▷ [(((fuelPressure > 500) && ((engineOn == true) && ((50 <= fuel) && (fuel <= 100))))]
}
```

Program for Rocket Landing

```
// [fuelPressure > 450 ∧ engineTemp <= 900 ∧ 50 ≤ fuel]

while (fuel > 100) {
    fuel := fuel − 10
};
if (engineTemp > 900){
    engineOn := False
}
else {
    engineOn := True
};
fuelPressure := fuelPressure + 50

// [fuelPressure > 500 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

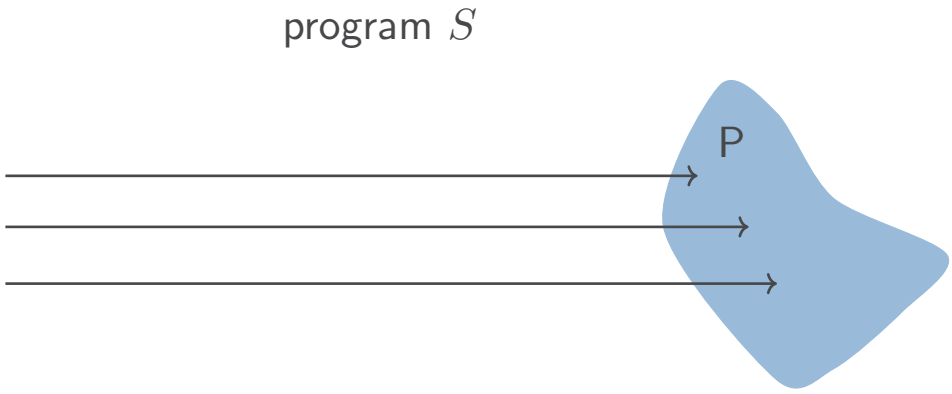


Program for Rocket Landing

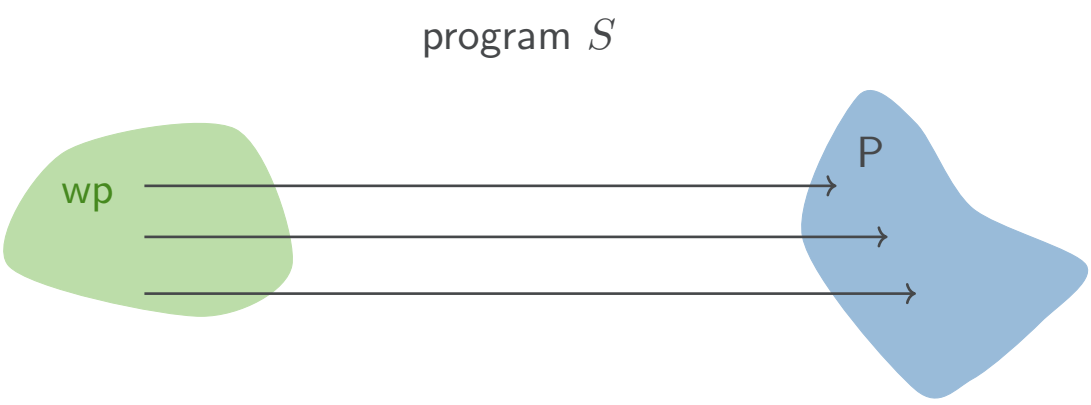
```
// [fuelPressure > 450 ∧ engineTemp <= 900 ∧ 50 ≤ fuel]

while (fuel > 100) {
    fuel := fuel − 10
};
if (engineTemp > 900){
    engineOn := False
}
else {
    engineOn := True
};
fuelPressure := fuelPressure + 50

// [fuelPressure > 500 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

Try variations,
e.g. which pre
holds always?

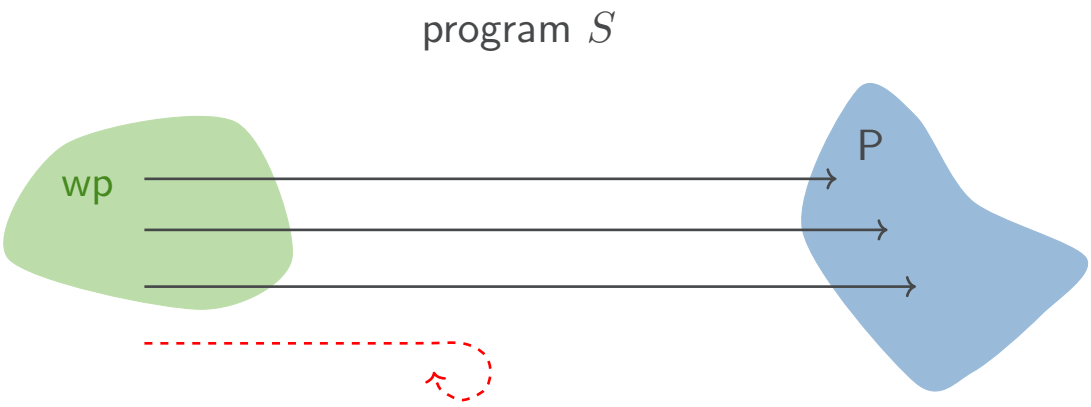
Diverging Loops



Diverging Loops

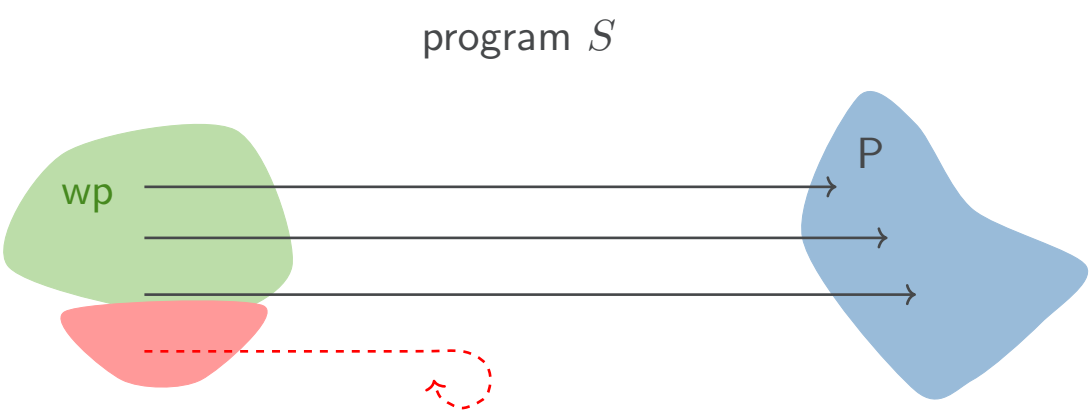


Diverging Loops



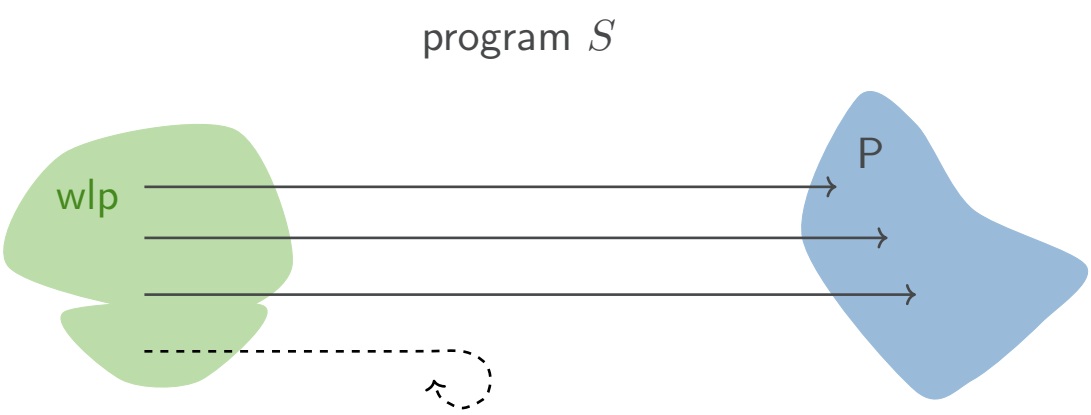
What about diverging loops?

Diverging Loops



What about diverging loops? currently those initial states are not included

Diverging Loops



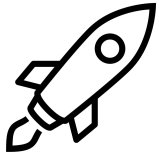
What about diverging loops? currently those initial states are not included

There is a version where they are included: weakest *liberal* preconditions
$\texttt{wlp}[\![\texttt{while}\ (\varphi)\ \{S\}]\!](P) = \mathrm{gfp}\, X.(\varphi \wedge \texttt{wlp}[\![S]\!](X)) \vee (\neg\varphi \wedge P) \vee (\varphi \wedge X)$

Add Probabilities

Why? E.g. to make the system error-tolerant for sensor issues

```
engineTemp := realTemp [0.95] engineTemp := falseTemp;
```

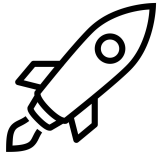


```
if (engineTemp > 900){
    engineOn := False
}
else {
    engineOn := True
};
```

Add Probabilities

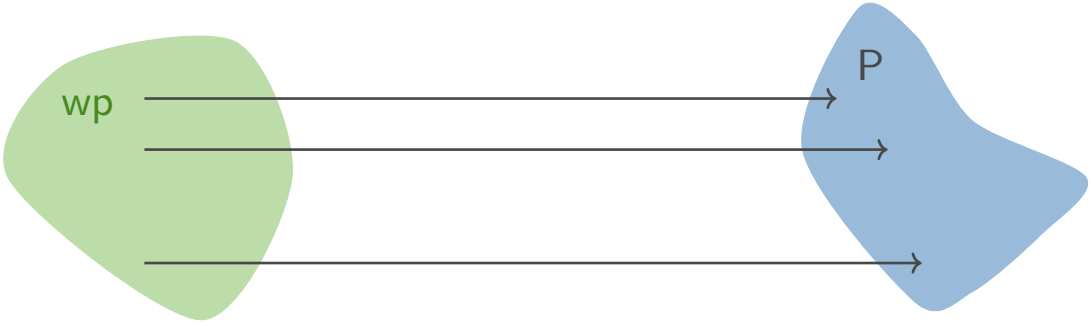
Why? E.g. to make the system error-tolerant for sensor issues

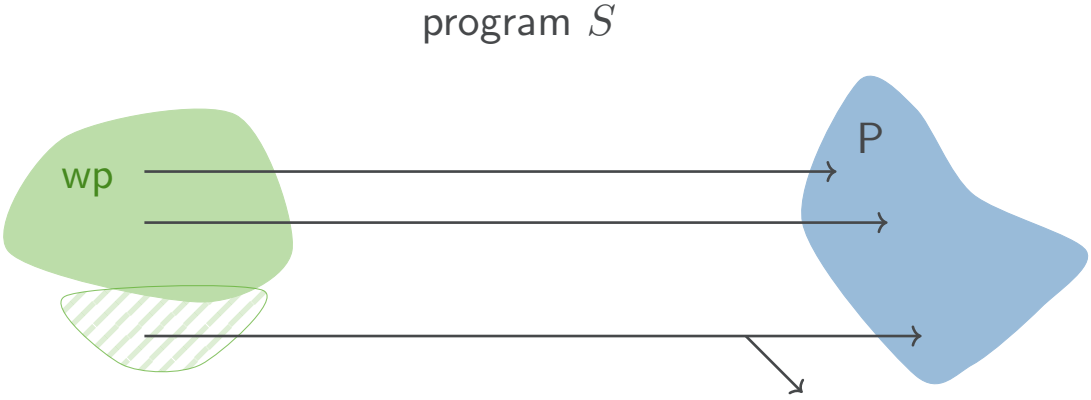
```
// ?

engineTemp := realTemp [0.95] engineTemp := falseTemp;

// [engineTemp ≤ 900]

if (engineTemp > 900){
    engineOn := False
}
else {
    engineOn := True
};

// [engineOn = True]
```

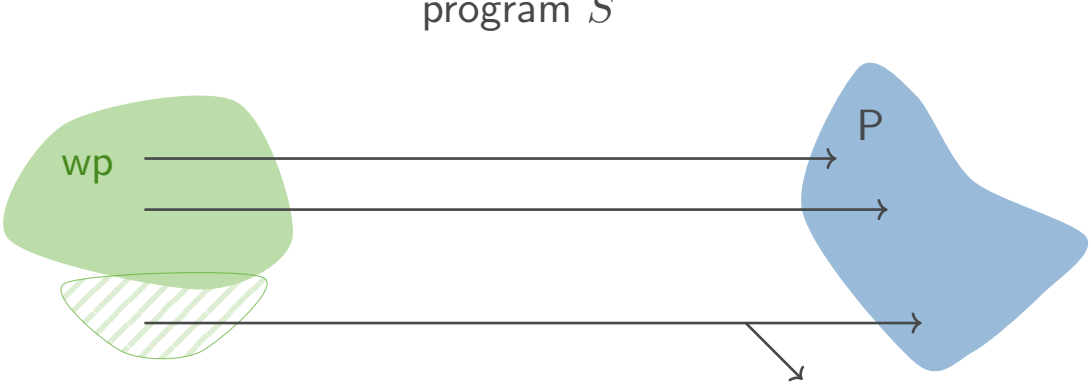


Weakest Preconditions

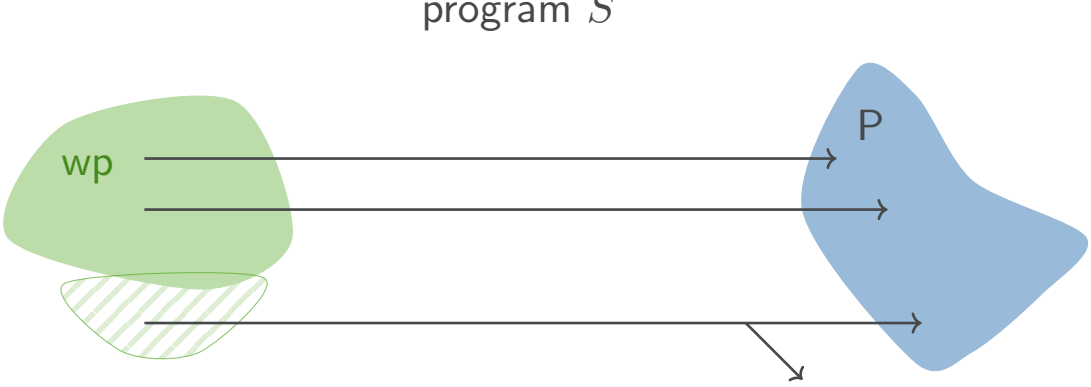
program S	$\texttt{wp}[\![S]\!](P)$
$x \coloneqq E$	$P[x/E]$
$\texttt{if}\ (\varphi)\ \{\ S_1\ \}\ \texttt{else}\ \{\ S_2\ \}$	$(\varphi \wedge \texttt{wp}[\![S_1]\!](P)) \vee (\neg\varphi \wedge \texttt{wp}[\![S_2]\!](P))$
$S_1\ ;\ S_2$	$\texttt{wp}[\![S_1]\!](\texttt{wp}[\![S_2]\!](P))$
$\texttt{while}\ (\varphi)\ \{S\}$	$\mathrm{lfp}\, X.(\varphi \wedge \texttt{wp}[\![S]\!](X)) \vee (\neg\varphi \wedge P)$
$S_1\ [p]\ S_2$	?


program S
wp
P


program S
wp
P



What about probabilistic branching? Only accept states that satisfy P with probability at least 0.5?



What about probabilistic branching? Only accept states that satisfy P with probability at least 0.5?

Change type of predicates to $[0, 1]$-valued functions

Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to [0, 1]$
$f(\sigma) = f([\![S]\!](\sigma))$

Quantitative Predicates

Predicate before:

$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$

$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:

$f : \Sigma \to [0, 1]$

$f(\sigma) = f([\![S]\!](\sigma))$

Probability that f holds in σ

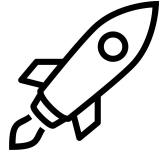
Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to [0, 1]$
$f(\sigma) = f([\![S]\!](\sigma))$

Probability that f holds in σ

```
engineTemp := realTemp [0.95] engineTemp := falseTemp
// [engineTemp ≤ 900]
```



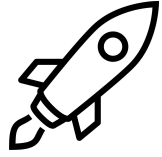
Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to [0, 1]$
$f(\sigma) = f([\![S]\!](\sigma))$

Probability that f holds in σ

```
// 0.95 [realTemp ≤ 900] + 0.05 [falseTemp ≤ 900]
engineTemp := realTemp [0.95] engineTemp := falseTemp
// [engineTemp ≤ 900]
```



Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to [0, 1]$
$f(\sigma) = f([\![S]\!](\sigma))$

Probability that f holds in σ

```
// 0.95 [realTemp ≤ 900] + 0.05 [falseTemp ≤ 900]
engineTemp := realTemp [0.95] engineTemp := falseTemp
// [engineTemp ≤ 900]
```

f = 0.95 [realTemp ≤ 900] + 0.05 [falseTemp ≤ 900]
f(realTemp = 1000, falseTemp = 800) = 0.95 · 0 + 0.05 · 1 = 0.05
f(realTemp = 800, falseTemp = 1000) = 0.95 · 1 + 0.05 · 0 = 0.95
f(realTemp = 950, falseTemp = 1000) = 0.95 · 0 + 0.05 · 0 = 0

```
if (φ){

    S1
}
else {

    S2
};
// P
```

```
if (φ){
// wp⟦S1⟧(P)
    S1
}
else {
// wp⟦S2⟧(P)
    S2
};
// P
```

// $\varphi \land \texttt{wp}[\![S_1]\!](P) \lor \neg\varphi \land \texttt{wp}[\![S_2]\!](P)$

if $(\varphi)\{$

// $\texttt{wp}[\![S_1]\!](P)$

S_1

$\}$
else $\{$

// $\texttt{wp}[\![S_2]\!](P)$

S_2

$\};$

// P

// $[\varphi] \cdot \mathtt{wp}[\![S_1]\!](P) + [\neg\varphi] \cdot \mathtt{wp}[\![S_2]\!](P)$

~~// $\varphi \wedge \mathtt{wp}[\![S_1]\!](P) \vee \neg\varphi \wedge \mathtt{wp}[\![S_2]\!](P)$~~

if $(\varphi)\{$

// $\mathtt{wp}[\![S_1]\!](P)$

S_1

$\}$
else $\{$

// $\mathtt{wp}[\![S_2]\!](P)$

S_2

$\}$;

// P

Weakest Preconditions

program S	$\texttt{wp}[\![S]\!](P)$	$\texttt{wp}[\![S]\!](f)$
$x \coloneqq E$	$P[x/E]$	$f[x/E]$
$\texttt{if}\ (\varphi)\ \{\, S_1 \,\}\ \texttt{else}\ \{\, S_2 \,\}$	$(\varphi \wedge \texttt{wp}[\![S_1]\!](P)) \vee (\neg\varphi \wedge \texttt{wp}[\![S_2]\!](P))$	$([\varphi]\texttt{wp}[\![S_1]\!](f)) + ([\neg\varphi]\texttt{wp}[\![S_2]\!](f))$
$S_1 \mathbin{;} S_2$	$\texttt{wp}[\![S_1]\!](\texttt{wp}[\![S_2]\!](P))$	$\texttt{wp}[\![S_1]\!](\texttt{wp}[\![S_2]\!](f))$
$\texttt{while}\ (\varphi)\ \{S\}$	$\mathrm{lfp}\, X.(\varphi \wedge \texttt{wp}[\![S]\!](X)) \vee (\neg\varphi \wedge P)$	$\mathrm{lfp}\, X.([\varphi]\texttt{wp}[\![S]\!](X)) + ([\neg\varphi]f)$
$S_1\ [p]\ S_2$	?	$p \cdot \texttt{wp}[\![S_1]\!](f) + (1-p) \cdot \texttt{wp}[\![S_2]\!](f)$

Caesar

Open rocket_landing_bounded.heyvl

https://rwth-aachen.sciebo.de/s/aKYC3EBGy7TPoPX

caesar > rocket_landing_bounded.heyvl

```
1  coproc rocket(init_fuel: UInt, init_fuelPressure: UInt, init_realTemp: UInt, init_falseTemp: UInt) -> (fuel: UInt,
   engineTemp: UInt, fuelPressure: UInt, engineOn: Bool, realTemp: Uint, falseTemp: Uint)
2      pre  0.95 * [init_realTemp <= 900 && 50<=init_fuel && init_fuelPressure > 450] + 0.05 * [init_falseTemp <= 900 &&
       50<=init_fuel && init_fuelPressure > 450]
3      post [fuelPressure > 500 && engineOn == true && 50 <= fuel && fuel <= 100]
4  {
5      fuel = init_fuel
6      realTemp = init_realTemp
7      falseTemp = init_falseTemp
8      fuelPressure = init_fuelPressure
9
10     var coin: Bool = flip(0.95)
11     if coin {
12         engineTemp = realTemp
13     }
14     else {
15         engineTemp = falseTemp
16     }
17     @invariant([50<=fuel && fuelPressure > 450&&  engineTemp <=900])
18     while fuel > 100 {
19         fuel = fuel-10
20     }
21     if engineTemp > 900 {
22             engineOn = false
23         } else {
24             engineOn = true
25         }
26     fuelPressure = fuelPressure + 50
27 }
```

Program for Rocket Landing

```
// 0.95 [realTemp ≤ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel]
+ 0.05 [falseTemp ≤ 900 ∧ fuelPressure > 450 ∧ 50 ≤ fuel]
```

```
engineTemp := realTemp [0.95] engineTemp := falseTemp;
while (fuel > 100) {
    fuel := fuel − 10
};
if (engineTemp > 900){
    engineOn := False
}
else {
    engineOn := True
};
fuelPressure := fuelPressure + 50
```

```
// [fuelPressure > 500 ∧ engineOn = True ∧ 50 ≤ fuel ≤ 100]
```

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
engineTemp := realTemp [0.95] engineTemp := falseTemp
// engineTemp
```

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
// 0.95 · realTemp + 0.05 · falseTemp
engineTemp ≔ realTemp [0.95] engineTemp ≔ falseTemp
// engineTemp
```

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
// 0.95 · realTemp + 0.05 · falseTemp
engineTemp := realTemp [0.95] engineTemp := falseTemp
// engineTemp
```

f = engineTemp

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
// 0.95 · realTemp + 0.05 · falseTemp
engineTemp ≔ realTemp [0.95] engineTemp ≔ falseTemp
// engineTemp
```

f = engineTemp
f(realTemp = 1000, falseTemp = 800) = 0.95 · 1000 + 0.05 · 800 = 990

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
// 0.95 · realTemp + 0.05 · falseTemp
engineTemp ≔ realTemp [0.95] engineTemp ≔ falseTemp
// engineTemp
```

f = engineTemp
f(realTemp = 1000, falseTemp = 800) = $0.95 \cdot 1000 + 0.05 \cdot 800 = 990$
f(realTemp = 800, falseTemp = 1000) = 810

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
// 0.95 · realTemp + 0.05 · falseTemp
engineTemp ≔ realTemp [0.95] engineTemp ≔ falseTemp
// engineTemp
```

f = engineTemp
f(realTemp = 1000, falseTemp = 800) = $0.95 \cdot 1000 + 0.05 \cdot 800 = 990$
f(realTemp = 800, falseTemp = 1000) = 810
f(realTemp = 950, falseTemp = 1000) = 952.5

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
// 0.95 · 1.2 · engineTemp
engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0
// engineTemp
```

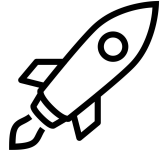
Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
// 1.14 · engineTemp
// 0.95 · 1.2 · engineTemp
engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0
// engineTemp
```



Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
// 1.14 · engineTemp
// 0.95 · 1.2 · engineTemp
engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0
// engineTemp
```

f = engineTemp

f(engineTemp = 1000) = 1.14 · 1000 = 1140

Unbounded Quantitative Predicates

Predicate before:
$P : \Sigma \to \{\texttt{true}, \texttt{false}\}$
$P(\sigma) = \texttt{true}$ iff $P([\![S]\!](\sigma)) = \texttt{true}$

Predicate now:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Expected value of f in σ

```
// 1.14 · engineTemp
// 0.95 · 1.2 · engineTemp
engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0
// engineTemp
```

f = engineTemp
f(engineTemp = 800) = 1.14 · 800 = 912

f(engineTemp = 1000) = 1.14 · 1000 = 1140
f(engineTemp = 950) = 1083

Unbounded Quantitative Predicates

```
while (fuel > 100) {

    fuel := fuel − 10;

    engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0

}
```

Unbounded Quantitative Predicates

```
while (fuel > 100) {

    fuel := fuel − 10;

    engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0

}
// engineTemp
```

Unbounded Quantitative Predicates

```
while (fuel > 100) {

    fuel := fuel − 10;

    engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0
// [fuel ≤ 100] · engineTemp + [fuel > 100] · 1.14^⌈(fuel−100)/10⌉ · engineTemp
}
// engineTemp
```

Unbounded Quantitative Predicates

```
while (fuel > 100) {

    fuel := fuel − 10;
// [fuel ≤ 100] · 1.14 · engineTemp
    + [fuel > 100] · 1.14^⌈(fuel +10-100)/10⌉ · engineTemp
    engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0
// [fuel ≤ 100] · engineTemp + [fuel > 100] · 1.14^⌈(fuel−100)/10⌉ · engineTemp
}
// engineTemp
```

Unbounded Quantitative Predicates

```
while (fuel > 100) {
// [fuel ≤ 110] · 1.14 · engineTemp
    + [fuel > 110] · 1.14^⌈(fuel−100)/10⌉ · engineTemp
    fuel := fuel − 10;
// [fuel ≤ 100] · 1.14 · engineTemp
    + [fuel > 100] · 1.14^⌈(fuel +10-100)/10⌉ · engineTemp
    engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0
// [fuel ≤ 100] · engineTemp + [fuel > 100] · 1.14^⌈(fuel−100)/10⌉ · engineTemp
}
// engineTemp
```

Unbounded Quantitative Predicates

$// \ [\text{fuel} \leq 100] \cdot \text{engineTemp} + [\text{fuel} > 100] \cdot 1.14^{\lceil \frac{\text{fuel}-100}{10} \rceil} \cdot \text{engineTemp}$

```
while (fuel > 100) {
```

$// \ [\text{fuel} \leq 110] \cdot 1.14 \cdot \text{engineTemp} + [\text{fuel} > 110] \cdot 1.14^{\lceil \frac{\text{fuel}-100}{10} \rceil} \cdot \text{engineTemp}$

```
    fuel := fuel − 10;
```

$// \ [\text{fuel} \leq 100] \cdot 1.14 \cdot \text{engineTemp} + [\text{fuel} > 100] \cdot 1.14^{\lceil \frac{\text{fuel}+10-100}{10} \rceil} \cdot \text{engineTemp}$

```
    engineTemp := 1.2 · engineTemp [0.95] engineTemp := 0
```

$// \ [\text{fuel} \leq 100] \cdot \text{engineTemp} + [\text{fuel} > 100] \cdot 1.14^{\lceil \frac{\text{fuel}-100}{10} \rceil} \cdot \text{engineTemp}$

```
}
```

$// \ \text{engineTemp}$

How to handle loops?

```
while (φ) {
    body
}
```

```
if (φ) {
    body;

    while (φ) {
        body
    }

}
```

How to handle loops?

```
while (φ) {
    body
}
// post
```

```
if (φ) {
    body;

    while (φ) {
        body
    }

}
```

How to handle loops?

```
// wp[[while]](post)
while (φ) {
    body
}
// post
```

```
if (φ) {
    body;
    while (φ) {
        body
    }
}
```

How to handle loops?

```
// wp[[while]](post)
while (φ) {
    body
}
// post
```

```
if (φ) {
    body;
    while (φ) {
        body
    }
}
// post
```

How to handle loops?

```
// wp[[while]](post)
while (φ) {
    body
}
// post
```

```
if (φ) {
    body;
    while (φ) {
        body
    }
    // post
}
// post
```

How to handle loops?

```
// wp[[while]](post)
while (φ) {
    body
}
// post
```

```
if (φ) {
    body;
// wp[[while]](post) = X
    while (φ) {
        body
    }
// post
}
// post
```

How to handle loops?

```
// wp[[while]](post)
while (φ) {
    body
}
// post
```

```
// wp[[while]](post)
if (φ) {
    body;
// wp[[while]](post) = X
    while (φ) {
        body
    }
// post
}
// post
```

How to handle loops?

```
// wp⟦while⟧(post)
while (φ) {
    body
}
// post
```

wp⟦while⟧(post)
= wp⟦if (φ) {body; while}⟧(post)

```
// wp⟦while⟧(post)
if (φ) {
    body;
// wp⟦while⟧(post) = X
    while (φ) {
        body
    }
// post
}
// post
```

How to handle loops?

```
// wp⟦while⟧(post)
while (φ) {
    body
}
// post
```

$$
\begin{aligned}
&\texttt{wp}[\![\text{while}]\!](\text{post}) \\
&= \texttt{wp}[\![\text{if } (\varphi) \ \{\text{body; while}\}]\!](\text{post}) \\
&= [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](X) + [\neg\varphi] \cdot \text{post},
\end{aligned}
$$

```
// wp⟦while⟧(post)
if (φ) {
    body;
// wp⟦while⟧(post) = X
    while (φ) {
        body
    }
// post
}
// post
```

How to handle loops?

```
// wp⟦while⟧(post)
while (φ) {
    body
}
// post
```

$$
\begin{aligned}
&\texttt{wp}[\![\text{while}]\!](\text{post}) \\
&= \texttt{wp}[\![\text{if } (\varphi) \ \{\text{body; while}\}]\!](\text{post}) \\
&= [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post}, \\
&= \operatorname{lfp} X.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post}
\end{aligned}
$$

```
// wp⟦while⟧(post)
if (φ) {
    body;
// wp⟦while⟧(post) = X
    while (φ) {
        body
    }
// post
}
// post
```

Finding fixpoints is hard

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \operatorname{lfp} \mathsf{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post} = \operatorname{lfp} X.\ \Phi_{\text{post}}(X)$$

Reminder: $\mathsf{X}, \mathsf{Y} \in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

- $\mathbb{R}^{\infty}_{\geq 0}$ is partially ordered (reflexive, transitive, antisymmetric)

Finding fixpoints is hard

undecidable

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \operatorname{lfp} \mathsf{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post} = \operatorname{lfp} X.\ \Phi_{\text{post}}(X)$$

Reminder: $\mathsf{X}, \mathsf{Y} \in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

- $\mathbb{R}^{\infty}_{\geq 0}$ is partially ordered (reflexive, transitive, antisymmetric)
- least element $\bot = 0$ exists

Finding fixpoints is hard

undecidable

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \text{lfp}\,\mathsf{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post} = \text{lfp}\, X.\ \Phi_{\text{post}}(X)$$

Reminder: $\mathsf{X}, \mathsf{Y} \in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

- $\mathbb{R}^{\infty}_{\geq 0}$ is partially ordered (reflexive, transitive, antisymmetric)
- least element $\bot = 0$ exists
- for every ascending ω-chain $p_0 \leq p_1 \leq \ldots$ the supremum $\bigsqcup_{i \in \mathbb{N}} p_i \in \mathbb{R}^{\infty}_{\geq 0}$ exists

Finding fixpoints is hard

undecidable

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \text{lfp}\,\text{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\text{X}) + [\neg\varphi] \cdot \text{post} = \text{lfp}\, X.\ \Phi_{\text{post}}(X)$$

Reminder: X, Y $\in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

- $\mathbb{R}^{\infty}_{\geq 0}$ is partially ordered (reflexive, transitive, antisymmetric)
- least element $\bot = 0$ exists
- for every ascending ω-chain $p_0 \leq p_1 \leq \ldots$ the supremum $\bigsqcup_{i \in \mathbb{N}} p_i \in \mathbb{R}^{\infty}_{\geq 0}$ exists

$\Rightarrow$ the set $\mathbb{R}^{\infty}_{\geq 0}$ is *ω-complete partially ordered*

Finding fixpoints is hard

undecidable

$$\texttt{wp}[\![\texttt{while}]\!](\text{post}) = \text{lfp}\,\text{X}.\ [\varphi] \cdot \texttt{wp}[\![\texttt{body}]\!](\text{X}) + [\neg\varphi] \cdot \text{post} = \text{lfp}\, X.\ \Phi_{\text{post}}(X)$$

Reminder: X, Y $\in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

- $\mathbb{R}^{\infty}_{\geq 0}$ is partially ordered (reflexive, transitive, antisymmetric)
- least element $\bot = 0$ exists
- for every ascending ω-chain $p_0 \leq p_1 \leq \ldots$ the supremum $\bigsqcup_{i \in \mathbb{N}} p_i \in \mathbb{R}^{\infty}_{\geq 0}$ exists

$\Rightarrow$ the set $\mathbb{R}^{\infty}_{\geq 0}$ is *ω-complete partially ordered*

$\Rightarrow$ transfers to set of expectations $\{f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$

Finding fixpoints is hard

undecidable

$$\texttt{wp}[\![\texttt{while}]\!](\text{post}) = \text{lfp}\,\mathsf{X}.\ [\varphi] \cdot \texttt{wp}[\![\texttt{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post} = \text{lfp}\, X.\ \Phi_{\text{post}}(X)$$

Reminder: X, Y $\in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

- $\mathbb{R}^{\infty}_{\geq 0}$ is partially ordered (reflexive, transitive, antisymmetric)
- least element $\bot = 0$ exists
- for every ascending ω-chain $p_0 \leq p_1 \leq \ldots$ the supremum $\bigsqcup_{i \in \mathbb{N}} p_i \in \mathbb{R}^{\infty}_{\geq 0}$ exists

$\Rightarrow$ the set $\mathbb{R}^{\infty}_{\geq 0}$ is *ω-complete partially ordered*

$\Rightarrow$ transfers to set of expectations $\{f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$

- *monotone* since $\forall X, Y : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$

$$X \leq Y \implies \Phi(X) \leq \Phi(Y)$$

Finding fixpoints is hard

undecidable

$$\texttt{wp}[\![\texttt{while}]\!](\text{post}) = \text{lfp}\,\text{X}.\ [\varphi] \cdot \texttt{wp}[\![\texttt{body}]\!](\text{X}) + [\neg\varphi] \cdot \text{post} = \text{lfp}\, X.\ \Phi_{\text{post}}(X)$$

Reminder: X, Y $\in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

- $\mathbb{R}^{\infty}_{\geq 0}$ is partially ordered (reflexive, transitive, antisymmetric)
- least element $\bot = 0$ exists
- for every ascending ω-chain $p_0 \leq p_1 \leq \ldots$ the supremum $\bigsqcup_{i \in \mathbb{N}} p_i \in \mathbb{R}^{\infty}_{\geq 0}$ exists

$\Rightarrow$ the set $\mathbb{R}^{\infty}_{\geq 0}$ is *ω-complete partially ordered*

$\Rightarrow$ transfers to set of expectations $\{f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$

- *monotone* since $\forall X, Y : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$

$$X \leq Y \implies \Phi(X) \leq \Phi(Y)$$

- *ω-continuous* since for every ascending ω-chain $X_0 \leq X_1 \leq \ldots \subseteq \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$

$$\Phi(\bigsqcup_{n \in \mathbb{N}} X_n) = \bigsqcup_{n \in \mathbb{N}} \Phi(X_n)$$

Finding fixpoints is hard

$$\mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi] \cdot \mathtt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X)$$

Reminder: $\mathsf{X}, \mathsf{Y} \in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

Kleene's Fixed Point Theorem

Set of expectations is ω-complete partially ordered and Φ_{post} is ω-continuous implies

$$\mathrm{lfp}\,\mathsf{X}.\ \Phi_{\mathsf{post}}(\mathsf{X}) = \bigsqcup_{n \in \mathbb{N}} \Phi^n_{\mathsf{post}}(\bot).$$

Finding fixpoints is hard

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \text{lfp}\,\text{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\text{X}) + [\neg\varphi] \cdot \text{post} = \text{lfp}\,X.\ \Phi_{\text{post}}(X) = \bigsqcup_{n\in\mathbb{N}} \Phi^n_{\text{post}}(\bot)$$

```
i := 20; fuel := 100;

while (i > 0) {

    i := i - 1;
    fuel := fuel + 10;

}
// fuel
```

Finding fixpoints is hard

$$\texttt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi] \cdot \texttt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X) = \bigsqcup_{n \in \mathbb{N}} \Phi^n_{\mathsf{post}}(\bot)$$

```
i := 20; fuel := 100;

while (i > 0) {

    i := i - 1;
    fuel := fuel + 10;
    // X
}
// fuel
```

Finding fixpoints is hard

$$\mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi]\cdot \mathtt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi]\cdot \mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X) = \bigsqcup_{n\in\mathbb{N}} \Phi^n_{\mathsf{post}}(\bot)$$

```
i := 20; fuel := 100;

while (i > 0) {
    // X[fuel/fuel+10, i/i-1]
    i := i - 1;
    fuel := fuel + 10;
    // X
}
// fuel
```

Finding fixpoints is hard

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \operatorname{lfp} \mathsf{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post} = \operatorname{lfp} X.\ \Phi_{\text{post}}(X) = \bigsqcup_{n \in \mathbb{N}} \Phi^n_{\text{post}}(\bot)$$

```
i := 20; fuel := 100;

while (i > 0) {
    // X[fuel/fuel+10, i/i-1]
    i := i - 1;
    fuel := fuel + 10;
    // X
}
// fuel
```

$$\Phi_{\text{fuel}}(\mathsf{X}) = [\mathsf{i} > 0] \cdot \mathsf{X}\ [\text{fuel}/\text{fuel}+10,\ \mathsf{i}/\mathsf{i}\text{-}1] + [\mathsf{i} = 0] \cdot \text{fuel} \qquad \bot = 0 : \sigma \mapsto 0$$

Finding fixpoints is hard

$$\mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi]\cdot\mathtt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi]\cdot\mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X) = \bigsqcup_{n\in\mathbb{N}} \Phi^n_{\mathsf{post}}(\bot)$$

```
i := 20; fuel := 100;

while (i > 0) {
    // X[fuel/fuel+10, i/i-1]
    i := i - 1;
    fuel := fuel + 10;
    // X
}
// fuel
```

$$\Phi_{\mathsf{fuel}}(\mathsf{X}) = [\mathsf{i} > 0]\cdot \mathsf{X}\ [\mathsf{fuel}/\mathsf{fuel}+10,\ \mathsf{i}/\mathsf{i}\text{-}1] + [\mathsf{i} = 0]\cdot \mathsf{fuel} \qquad \bot = 0 : \sigma \mapsto 0$$

$$\Phi_{\mathsf{fuel}}(0) = [\mathsf{i} > 0]\cdot 0 + [\mathsf{i} = 0]\cdot \mathsf{fuel}$$

Finding fixpoints is hard

$$\mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi]\cdot\mathtt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi]\cdot\mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X) = \bigsqcup_{n\in\mathbb{N}} \Phi^n_{\mathsf{post}}(\bot)$$

```
i := 20; fuel := 100;

while (i > 0) {
    // X[fuel/fuel+10, i/i-1]
    i := i - 1;
    fuel := fuel + 10;
    // X
}
// fuel
```

$$\Phi_{\mathsf{fuel}}(\mathsf{X}) = [\mathsf{i} > 0]\cdot \mathsf{X}\ [\mathsf{fuel}/\mathsf{fuel}+10,\ \mathsf{i}/\mathsf{i}\text{-}1] + [\mathsf{i} = 0]\cdot \mathsf{fuel} \qquad \bot = 0 : \sigma \mapsto 0$$

$$\Phi_{\mathsf{fuel}}(0) = [\mathsf{i} > 0]\cdot 0 + [\mathsf{i} = 0]\cdot \mathsf{fuel}$$

$$\Phi^2_{\mathsf{fuel}}(0) = [\mathsf{i} > 1]\cdot 0 + [\mathsf{i} = 1]\cdot (\mathsf{fuel} + 10) + [\mathsf{i} = 0]\cdot \mathsf{fuel}$$

Finding fixpoints is hard

$$\mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi]\cdot \mathtt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi]\cdot \mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X) = \bigsqcup_{n\in\mathbb{N}} \Phi^n_{\mathsf{post}}(\bot)$$

```
i := 20; fuel := 100;

while (i > 0) {
    // X[fuel/fuel+10, i/i-1]
    i := i - 1;
    fuel := fuel + 10;
    // X
}
// fuel
```

$$\Phi_{\mathsf{fuel}}(\mathsf{X}) = [\mathsf{i} > 0]\cdot \mathsf{X}\ [\mathsf{fuel}/\mathsf{fuel}{+}10,\ \mathsf{i}/\mathsf{i}\text{-}1] + [\mathsf{i} = 0]\cdot \mathsf{fuel} \qquad \bot = 0 : \sigma \mapsto 0$$

$$\Phi_{\mathsf{fuel}}(0) = [\mathsf{i} > 0]\cdot 0 + [\mathsf{i} = 0]\cdot \mathsf{fuel}$$

$$\Phi^2_{\mathsf{fuel}}(0) = [\mathsf{i} > 1]\cdot 0 + [\mathsf{i} = 1]\cdot (\mathsf{fuel} + 10) + [\mathsf{i} = 0]\cdot \mathsf{fuel}$$

$$\Phi^3_{\mathsf{fuel}}(0) = [\mathsf{i} > 2]\cdot 0 + [\mathsf{i} = 2]\cdot (\mathsf{fuel} + 20) + [\mathsf{i} = 1]\cdot (\mathsf{fuel} + 10) + [\mathsf{i} = 0]\cdot \mathsf{fuel}$$

Finding fixpoints is hard

$$\mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi]\cdot \mathtt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi]\cdot \mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X) = \bigsqcup_{n\in\mathbb{N}} \Phi^n_{\mathsf{post}}(\bot)$$

```
i := 20; fuel := 100;

while (i > 0) {
    // X[fuel/fuel+10, i/i-1]
    i := i - 1;
    fuel := fuel + 10;
    // X
}
// fuel
```

$$\Phi_{\mathsf{fuel}}(\mathsf{X}) = [\mathsf{i} > 0]\cdot \mathsf{X}\ [\mathsf{fuel}/\mathsf{fuel}+10,\ \mathsf{i}/\mathsf{i}\text{-}1] + [\mathsf{i} = 0]\cdot \mathsf{fuel} \qquad \bot = 0 : \sigma \mapsto 0$$

$$\Phi_{\mathsf{fuel}}(0) = [\mathsf{i} > 0]\cdot 0 + [\mathsf{i} = 0]\cdot \mathsf{fuel}$$

$$\Phi^2_{\mathsf{fuel}}(0) = [\mathsf{i} > 1]\cdot 0 + [\mathsf{i} = 1]\cdot (\mathsf{fuel}+10) + [\mathsf{i} = 0]\cdot \mathsf{fuel}$$

$$\Phi^3_{\mathsf{fuel}}(0) = [\mathsf{i} > 2]\cdot 0 + [\mathsf{i} = 2]\cdot (\mathsf{fuel}+20) + [\mathsf{i} = 1]\cdot (\mathsf{fuel}+10) + [\mathsf{i} = 0]\cdot \mathsf{fuel}$$

$$\Phi_{\mathsf{fuel}}(0) \le \Phi^2_{\mathsf{fuel}}(0) \le \Phi^3_{\mathsf{fuel}}(0) \le \dots$$

Finding fixpoints is hard

$$\mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi]\cdot\mathtt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi]\cdot\mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X) = \bigsqcup_{n\in\mathbb{N}} \Phi^n_{\mathsf{post}}(\bot)$$

```
i := 20; fuel := 100;
// fuel + i · 10
while (i > 0) {
    // X[fuel/fuel+10, i/i−1]
    i := i − 1;
    fuel := fuel + 10;
    // X
}
// fuel
```

$$\Phi_{\mathsf{fuel}}(\mathsf{X}) = [\mathsf{i} > 0]\cdot \mathsf{X}\ [\mathsf{fuel}/\mathsf{fuel}{+}10,\ \mathsf{i}/\mathsf{i}\text{-}1] + [\mathsf{i} = 0]\cdot\mathsf{fuel} \qquad \bot = 0 : \sigma \mapsto 0$$

$$\Phi_{\mathsf{fuel}}(0) = [\mathsf{i} > 0]\cdot 0 + [\mathsf{i} = 0]\cdot\mathsf{fuel}$$

$$\Phi^2_{\mathsf{fuel}}(0) = [\mathsf{i} > 1]\cdot 0 + [\mathsf{i} = 1]\cdot(\mathsf{fuel}+10) + [\mathsf{i} = 0]\cdot\mathsf{fuel}$$

$$\Phi^3_{\mathsf{fuel}}(0) = [\mathsf{i} > 2]\cdot 0 + [\mathsf{i} = 2]\cdot(\mathsf{fuel}+20) + [\mathsf{i} = 1]\cdot(\mathsf{fuel}+10) + [\mathsf{i} = 0]\cdot\mathsf{fuel}$$

$$\Phi_{\mathsf{fuel}}(0) \le \Phi^2_{\mathsf{fuel}}(0) \le \Phi^3_{\mathsf{fuel}}(0) \le \dots$$

$$\bigsqcup_{n\in\mathbb{N}} \Phi^n_{\mathsf{fuel}}(0) = \mathsf{fuel} + \mathsf{i}\cdot 10$$

Finding fixpoints is hard

$$\mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi]\cdot\mathtt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi]\cdot\mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X) = \bigsqcup_{n\in\mathbb{N}} \Phi^n_{\mathsf{post}}(\bot)$$

```
// 300
i := 20; fuel := 100;
// fuel + i · 10
while (i > 0) {
    // X[fuel/fuel+10, i/i−1]
    i := i − 1;
    fuel := fuel + 10;
    // X
}
// fuel
```

$$\Phi_{\mathsf{fuel}}(\mathsf{X}) = [\mathsf{i} > 0]\cdot \mathsf{X}\ [\mathsf{fuel}/\mathsf{fuel}{+}10,\ \mathsf{i}/\mathsf{i}\text{-}1] + [\mathsf{i} = 0]\cdot\mathsf{fuel} \qquad \bot = 0 : \sigma \mapsto 0$$

$$\Phi_{\mathsf{fuel}}(0) = [\mathsf{i} > 0]\cdot 0 + [\mathsf{i} = 0]\cdot\mathsf{fuel}$$

$$\Phi^2_{\mathsf{fuel}}(0) = [\mathsf{i} > 1]\cdot 0 + [\mathsf{i} = 1]\cdot(\mathsf{fuel}+10) + [\mathsf{i} = 0]\cdot\mathsf{fuel}$$

$$\Phi^3_{\mathsf{fuel}}(0) = [\mathsf{i} > 2]\cdot 0 + [\mathsf{i} = 2]\cdot(\mathsf{fuel}+20) + [\mathsf{i} = 1]\cdot(\mathsf{fuel}+10) + [\mathsf{i} = 0]\cdot\mathsf{fuel}$$

$$\Phi_{\mathsf{fuel}}(0) \le \Phi^2_{\mathsf{fuel}}(0) \le \Phi^3_{\mathsf{fuel}}(0) \le \dots$$

$$\bigsqcup_{n\in\mathbb{N}} \Phi^n_{\mathsf{fuel}}(0) = \mathsf{fuel} + \mathsf{i}\cdot 10$$

Proving upper bounds is easier

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \operatorname{lfp} \mathsf{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post} = \operatorname{lfp} X.\ \Phi_{\text{post}}(X)$$

Reminder: $\mathsf{X}, \mathsf{Y} \in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

Proving upper bounds is easier

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \operatorname{lfp} \mathsf{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post} = \operatorname{lfp} X.\ \Phi_{\text{post}}(X)$$

Reminder: $\mathsf{X}, \mathsf{Y} \in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

Park Induction

Set of expectations is ω-complete partially ordered and Φ_{post} is ω-continuous implies

$$\Phi_{\text{post}}(\mathsf{I}) \leq \mathsf{I} \implies \texttt{wp}[\![\text{while}]\!](\text{post}) \leq \mathsf{I}.$$

Proving upper bounds is easier

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \operatorname{lfp} \mathsf{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \text{post} = \operatorname{lfp} X.\ \Phi_{\text{post}}(X)$$

Reminder: $\mathsf{X}, \mathsf{Y} \in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

Park Induction

Set of expectations is ω-complete partially ordered and Φ_{post} is ω-continuous implies

$$\Phi_{\text{post}}(\mathsf{I}) \leq \mathsf{I} \implies \texttt{wp}[\![\text{while}]\!](\text{post}) \leq \mathsf{I}.$$

$$\Phi_{\text{fuel}}(\mathsf{X}) = [\mathsf{i} > 0] \cdot \mathsf{X}\ [\text{fuel/fuel+10, i/i-1}] + [\mathsf{i} = 0] \cdot \text{fuel}$$

Show that $\texttt{wp}[\![\text{while}]\!](\text{post}) \leq 2 \cdot \text{fuel} + 30 \cdot \mathsf{i}$:

Proving upper bounds is easier

$$\texttt{wp}[\![\text{while}]\!](\text{post}) = \text{lfp}\,\text{X}.\ [\varphi] \cdot \texttt{wp}[\![\text{body}]\!](\text{X}) + [\neg\varphi] \cdot \text{post} = \text{lfp}\,X.\ \Phi_{\text{post}}(X)$$

Reminder: X, Y $\in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

Park Induction

Set of expectations is ω-complete partially ordered and Φ_{post} is ω-continuous implies

$$\Phi_{\text{post}}(\text{I}) \leq \text{I} \implies \texttt{wp}[\![\text{while}]\!](\text{post}) \leq \text{I}.$$

$$\Phi_{\text{fuel}}(\text{X}) = [\text{i} > 0] \cdot \text{X}\ [\text{fuel/fuel+10, i/i-1}] + [\text{i} = 0] \cdot \text{fuel}$$

Show that $\texttt{wp}[\![\text{while}]\!](\text{post}) \leq 2 \cdot \text{fuel} + 30 \cdot \text{i}$:

$$\Phi_{\text{fuel}}(2 \cdot \text{fuel} + 30 \cdot \text{i}) = [\text{i} > 0] \cdot (2 \cdot \text{fuel} + 25 \cdot \text{i} - 10) + [\text{i} = 0] \cdot \text{fuel} \leq 2 \cdot \text{fuel} + 30 \cdot \text{i}$$ ✓

Proving upper bounds is easier

$$\mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) = \mathrm{lfp}\,\mathsf{X}.\ [\varphi] \cdot \mathtt{wp}[\![\mathsf{body}]\!](\mathsf{X}) + [\neg\varphi] \cdot \mathsf{post} = \mathrm{lfp}\,X.\ \Phi_{\mathsf{post}}(X)$$

Reminder: $\mathsf{X}, \mathsf{Y} \in \{\Sigma \to \mathbb{R}^{\infty}_{\geq 0}\}$ are expectations

Park Induction

Set of expectations is ω-complete partially ordered and Φ_{post} is ω-continuous implies

$$\Phi_{\mathsf{post}}(\mathsf{I}) \leq \mathsf{I} \implies \mathtt{wp}[\![\mathsf{while}]\!](\mathsf{post}) \leq \mathsf{I}.$$

- there exist many more proof rules like κ-induction, ω-induction, guard strengthening, ...
- some consider upper bounds, others refute lower bounds

Caesar

Open rocket_landing_bounded.heyvl

https://rwth-aachen.sciebo.de/s/aKYC3EBGy7TPoPX

caesar > rocket_landing_bounded.heyvl

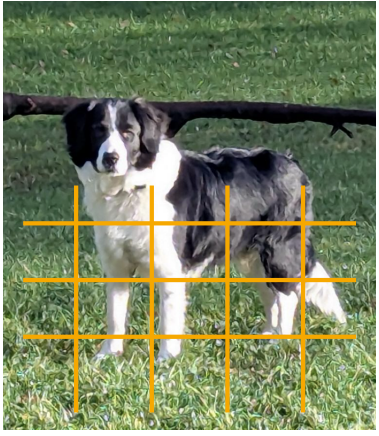
```
1  coproc rocket(init_fuel: UInt, init_fuelPressure: UInt, init_realTemp: UInt, init_falseTemp: UInt) -> (fuel: UInt,
   engineTemp: UInt, fuelPressure: UInt, engineOn: Bool, realTemp: Uint, falseTemp: Uint)
2      pre  0.95 * [init_realTemp <= 900 && 50<=init_fuel && init_fuelPressure > 450] + 0.05 * [init_falseTemp <= 900 &&
       50<=init_fuel && init_fuelPressure > 450]
3      post [fuelPressure > 500 && engineOn == true && 50 <= fuel && fuel <= 100]
4  {
5      fuel = init_fuel
6      realTemp = init_realTemp
7      falseTemp = init_falseTemp
8      fuelPressure = init_fuelPressure
9
10     var coin: Bool = flip(0.95)
11     if coin {
12         engineTemp = realTemp
13     }
14     else {
15         engineTemp = falseTemp
16     }
17     @invariant([50<=fuel && fuelPressure > 450&&  engineTemp <=900])
18     while fuel > 100 {
19         fuel = fuel-10
20     }
21     if engineTemp > 900 {
22             engineOn = false
23         } else {
24             engineOn = true
25         }
26     fuelPressure = fuelPressure + 50
27 }
```

Break until 16:45?

Online Optimization

Short-Sighted Cow*

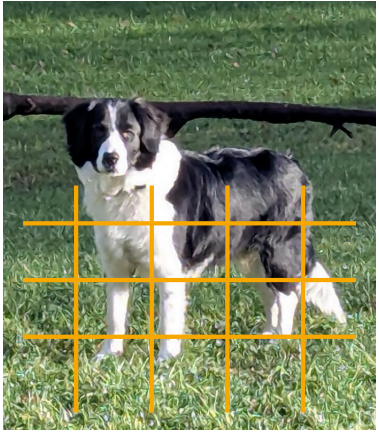
*Dog



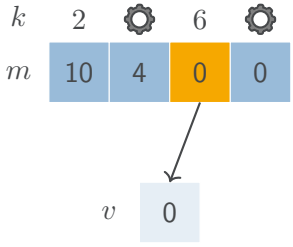
Online Optimization

Short-Sighted Cow*

*Dog



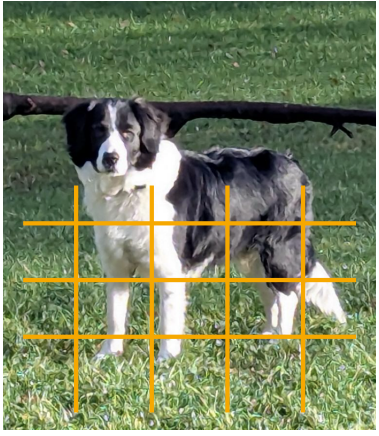
Lock-Freedom



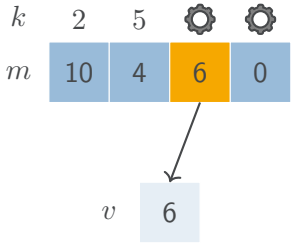
Online Optimization

Short-Sighted Cow*

*Dog



Lock-Freedom

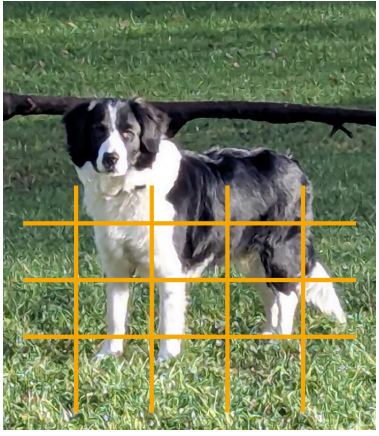


Ski-Rental Problem

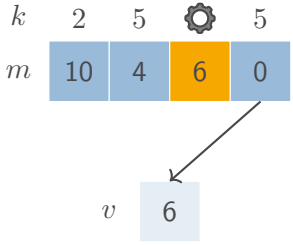
Online Optimization

Short-Sighted Cow*

*Dog



Lock-Freedom

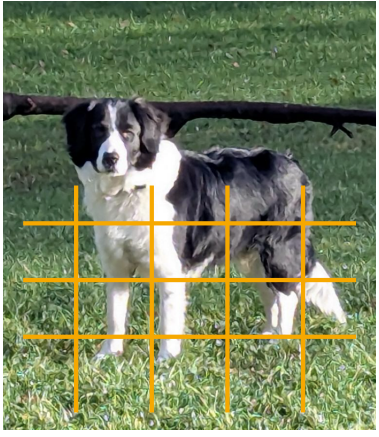


Ski-Rental Problem

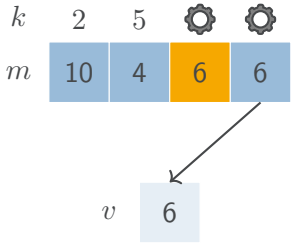
Online Optimization

Short-Sighted Cow*

*Dog



Lock-Freedom

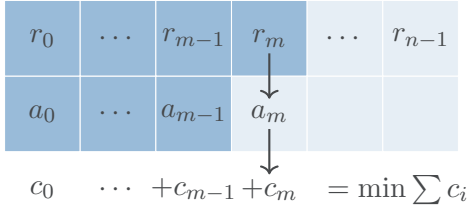


Ski-Rental Problem

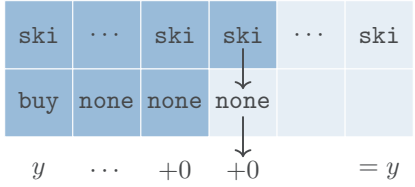
Online Optimization

An online algorithm is one that receives a sequence of requests and, performs an immediate action in response to each request. Each sequence of requests and corresponding actions has an associated cost.

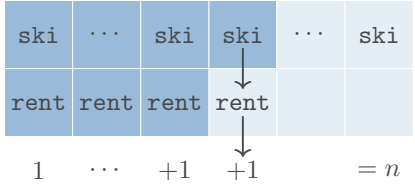
Richard M. Karp, 1992



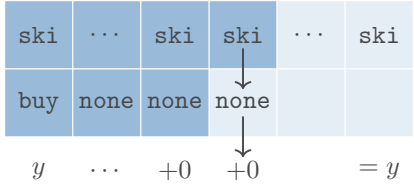
Ski-Rental Problem

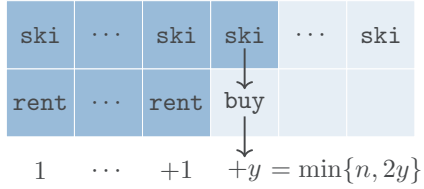


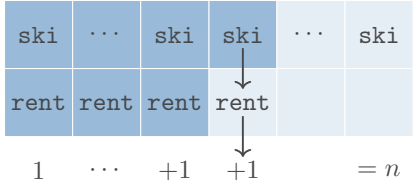
Ski-Rental Problem



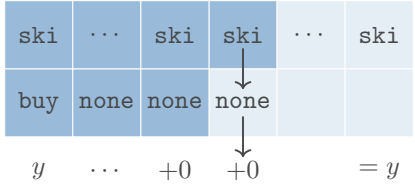
Ski-Rental Problem

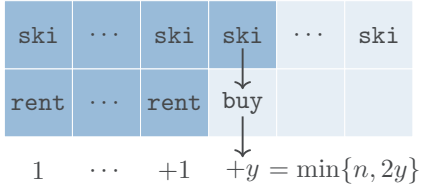


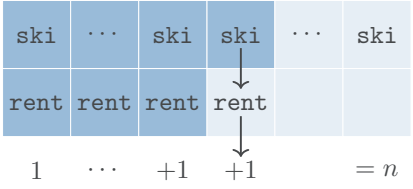


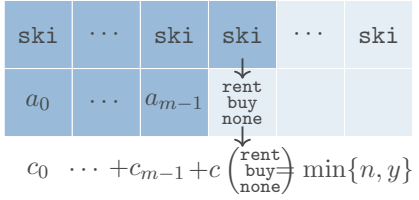


Ski-Rental Problem









Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions
  $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$,
  $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

$\mathsf{Off} = \texttt{while}\ (m < n)\ \{$

$m \coloneqq m + 1\}$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

$$
\begin{aligned}
\mathsf{Off} = \texttt{while}\ (m < n)\ \{ & \\
& \texttt{if}\ (\texttt{true}) \to \{ \quad \} \\
& \oplus\ (\texttt{true}) \to \{ \quad \} \\
& \oplus\ (\texttt{true}) \to \{ \quad \} \\
& m \coloneqq m + 1 \}
\end{aligned}
$$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

$$
\begin{aligned}
\mathsf{Off} = \texttt{while}\ (m < n)\ \{ & \\
& \texttt{if}\ (\texttt{true}) \to \{\ \odot 1\ \fatsemi\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot y\ \fatsemi\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\ \fatsemi\ \} \\
& m \coloneqq m + 1 \}
\end{aligned}
$$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

$$
\begin{aligned}
\mathsf{Off} = \texttt{while}\ (m < n)\ \{ & \\
& \texttt{if}\ (\texttt{true}) \to \{\ \odot 1 \mathbin{;} \texttt{skip}\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot y \mathbin{;}\ s \coloneqq 1\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \texttt{skip}\ \} \\
& m \coloneqq m + 1 \}
\end{aligned}
$$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

$$
\begin{aligned}
\mathsf{Off} = \texttt{while}\ (m < n)\ \{ & \\
& \texttt{if}\ (\texttt{true}) \to \{\ \odot 1 \mathbin{;} \texttt{skip}\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot y \mathbin{;}\ s \coloneqq 1\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \texttt{skip}\ \} \\
& m \coloneqq m + 1 \}
\end{aligned}
$$

Next Action f^m

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

Next Action f^m

- specifies algorithm $(\texttt{rent}, \ldots, \texttt{rent}, \texttt{buy}, \texttt{none}, \ldots)$

$$
\begin{aligned}
\mathrm{Off} = \texttt{while}\ (m < n)\ \{ & \\
& \texttt{if}\ (\texttt{true}) \to \{\ \odot 1 \fatsemi \texttt{skip}\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot y \fatsemi s \coloneqq 1\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \fatsemi \texttt{skip}\ \} \\
& m \coloneqq m + 1 \}
\end{aligned}
$$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

Next Action f^m

- specifies algorithm $(\texttt{rent}, \ldots, \texttt{rent}, \texttt{buy}, \texttt{none}, \ldots)$

$$\begin{aligned}
\mathsf{Off} = \texttt{while}\ &(m < n)\ \{ \\
&\texttt{if}\ (\texttt{true}) \to \{\ \odot 1 \mathbin{;} \texttt{skip}\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot y \mathbin{;}\ s \coloneqq 1\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \texttt{skip}\ \} \\
&m \coloneqq m + 1\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{On} = \texttt{while}\ &(m < n)\ \{ \\
&\\
&\\
&\\
&\\
&m \coloneqq m + 1\}
\end{aligned}$$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

Next Action f^m

- specifies algorithm $(\texttt{rent}, \ldots, \texttt{rent}, \texttt{buy}, \texttt{none}, \ldots)$

$$
\begin{aligned}
\mathsf{Off} = \texttt{while}\ &(m < n)\ \{ \\
&\texttt{if}\ (\texttt{true}) \to \{\ \odot 1 \mathbin{;} \texttt{skip}\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot y \mathbin{;}\ s \coloneqq 1\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \texttt{skip}\ \} \\
&m \coloneqq m + 1\}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{On} = \texttt{while}\ &(m < n)\ \{ \\
&\texttt{if}\ (\qquad) \to \{\qquad\qquad\qquad\qquad\qquad\qquad\ \} \\
&\oplus\ (\qquad) \to \{\qquad\qquad\qquad\qquad\qquad\qquad\ \} \\
&\oplus\ (\qquad) \to \{\qquad\qquad\qquad\qquad\qquad\qquad\ \} \\
&m \coloneqq m + 1\}
\end{aligned}
$$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

Next Action f^m

- specifies algorithm $(\texttt{rent}, \ldots, \texttt{rent}, \texttt{buy}, \texttt{none}, \ldots)$

$$
\begin{aligned}
\mathsf{Off} = \texttt{while}\ (m < n)\ \{ & \\
& \texttt{if}\ (\texttt{true}) \to \{\ \odot 1 \mathbin{;} \texttt{skip}\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot y \mathbin{;}\ s \coloneqq 1\ \} \\
& \oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \texttt{skip}\ \} \\
& m \coloneqq m + 1 \}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{On} = \texttt{while}\ (m < n)\ \{ & \\
& \texttt{if}\ (\quad\quad) \to \{\ \odot 1 \mathbin{;} \quad\quad\quad\quad\quad\quad\ \} \\
& \oplus\ (\quad\quad) \to \{\ \odot y \mathbin{;} \quad\quad\quad\quad\quad\quad\ \} \\
& \oplus\ (\quad\quad) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \quad\quad\ \} \\
& m \coloneqq m + 1 \}
\end{aligned}
$$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

Next Action f^m

- specifies algorithm $(\texttt{rent}, \ldots, \texttt{rent}, \texttt{buy}, \texttt{none}, \ldots)$

$$
\begin{aligned}
\mathsf{Off} = \texttt{while}\ &(m < n)\ \{ \\
&\texttt{if}\ (\texttt{true}) \to \{\ \odot 1 \fatsemi \texttt{skip}\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot y \fatsemi s \coloneqq 1\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \fatsemi \texttt{skip}\ \} \\
&m \coloneqq m + 1\}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{On} = \texttt{while}\ &(m < n)\ \{ \\
&\texttt{if}\ (\quad) \to \{\ \odot 1 \fatsemi \texttt{skip} \fatsemi \quad\} \\
&\oplus\ (\quad) \to \{\ \odot y \fatsemi s \coloneqq 1 \fatsemi \quad\} \\
&\oplus\ (\quad) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \fatsemi \texttt{skip}\ \} \\
&m \coloneqq m + 1\}
\end{aligned}
$$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

Next Action f^m

- specifies algorithm $(\texttt{rent}, \ldots, \texttt{rent}, \texttt{buy}, \texttt{none}, \ldots)$

$$
\begin{aligned}
\mathsf{Off} = \texttt{while}\ (m < n)\ \{ \\
&\texttt{if}\ (\texttt{true}) \to \{\ \odot 1 \mathbin{;} \texttt{skip}\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot y \mathbin{;}\ s \coloneqq 1\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \texttt{skip}\ \} \\
&m \coloneqq m + 1\}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{On} = \texttt{while}\ (m < n)\ \{ \\
&\texttt{if}\ (\quad\quad) \to \{\ \odot 1 \mathbin{;} \texttt{skip} \mathbin{;}\ c \coloneqq c + 1\ \} \\
&\oplus\ (\quad\quad) \to \{\ \odot y \mathbin{;}\ s \coloneqq 1 \mathbin{;}\ c \coloneqq c + 1\ \} \\
&\oplus\ (\quad\quad) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \texttt{skip}\ \} \\
&m \coloneqq m + 1\}
\end{aligned}
$$

Ski-Rental Problem Translated

Optimization Problem

- requests $\{\texttt{ski}\} = \mathcal{R}$
- actions $\{\texttt{rent}, \texttt{buy}, \texttt{none}\} = \mathcal{A}$
- costs $c_{\texttt{rent}} = 1$, $c_{\texttt{buy}} = y$, $c_{\texttt{none}} = [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty$

Next Action f^m

- specifies algorithm $(\texttt{rent}, \ldots, \texttt{rent}, \texttt{buy}, \texttt{none}, \ldots)$

$$
\begin{aligned}
\mathsf{Off} = \texttt{while}\ (m < n)\ \{ \\
&\texttt{if}\ (\texttt{true}) \to \{\ \odot 1 \mathbin{;} \texttt{skip}\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot y \mathbin{;}\ s \coloneqq 1\ \} \\
&\oplus\ (\texttt{true}) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \texttt{skip}\ \} \\
&m \coloneqq m + 1\}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{On} = \texttt{while}\ (m < n)\ \{ \\
&\texttt{if}\ (c < y) \to \{\ \odot 1 \mathbin{;} \texttt{skip} \mathbin{;}\ c \coloneqq c + 1\ \} \\
&\oplus\ (c = y) \to \{\ \odot y \mathbin{;}\ s \coloneqq 1 \mathbin{;}\ c \coloneqq c + 1\ \} \\
&\oplus\ (c > y) \to \{\ \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \mathbin{;} \texttt{skip}\ \} \\
&m \coloneqq m + 1\}
\end{aligned}
$$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbf{1})$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbf{1})$
- commutative monoid $(W, \oplus, \mathbf{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

- monoid $(\mathbb{N}^\infty, +, 0)$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

- monoid $(\mathbb{N}^\infty, +, 0)$
- commutative monoid $(\mathbb{N}^\infty, \min, \infty)$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

- monoid $(\mathbb{N}^\infty, +, 0)$
- commutative monoid $(\mathbb{N}^\infty, \min, \infty)$
- scalar operation $+$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

- monoid $(\mathbb{N}^\infty, +, 0)$
- commutative monoid $(\mathbb{N}^\infty, \min, \infty)$
- scalar operation $+$

$\to \mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

- monoid $(\mathbb{N}^\infty, +, 0)$
- commutative monoid $(\mathbb{N}^\infty, \min, \infty)$
- scalar operation $+$

$\to$ $\mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$

Expectation Module

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

- monoid $(\mathbb{N}^\infty, +, 0)$
- commutative monoid $(\mathbb{N}^\infty, \min, \infty)$
- scalar operation $+$

$\to \mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$

Expectation Module

- monoid $\mathcal{M}_{\text{prob}} = ([0, 1], \cdot, 1)$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

- monoid $(\mathbb{N}^\infty, +, 0)$
- commutative monoid $(\mathbb{N}^\infty, \min, \infty)$
- scalar operation $+$

$\to \mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$

Expectation Module

- monoid $\mathcal{M}_{\text{prob}} = ([0,1], \cdot, 1)$
- commutative monoid $(\mathbb{R}^\infty_{\geq 0}, +, 0)$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

- monoid $(\mathbb{N}^\infty, +, 0)$
- commutative monoid $(\mathbb{N}^\infty, \min, \infty)$
- scalar operation $+$

$\to \mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$

Expectation Module

- monoid $\mathcal{M}_{\text{prob}} = ([0,1], \cdot, 1)$
- commutative monoid $(\mathbb{R}^\infty_{\geq 0}, +, 0)$
- scalar multiplication $\cdot$

Weighted Programming wGCL

Monoid-Module $\mathcal{W}$

- monoid $\mathcal{M} = (M, \odot, \mathbb{1})$
- commutative monoid $(W, \oplus, \mathbb{0})$
- scalar operation $\otimes : M \times W \to W$

Natural Order $x \preccurlyeq z \iff \exists y : x \oplus y = z$

- partial order with least element $\bot \in \mathcal{W}$, supremum of all ω-ascending chains $\bigsqcup_{i \in \mathbb{N}} x_i \in \mathcal{W}$ exists
  $\implies$ *ω-complete partial order*
- addition $\oplus$ and scalar multiplication $\otimes$ are ω-continuous
  $\implies$ *ω-continuous monoid-module*

Tropical Semiring

- monoid $(\mathbb{N}^\infty, +, 0)$
- commutative monoid $(\mathbb{N}^\infty, \min, \infty)$
- scalar operation $+$

$\to \mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$

Expectation Module

- monoid $\mathcal{M}_{\text{prob}} = ([0,1], \cdot, 1)$
- commutative monoid $(\mathbb{R}^\infty_{\geq 0}, +, 0)$
- scalar multiplication $\cdot$
- module $\mathcal{W}_{\text{prob}} = (\mathbb{R}^\infty_{\geq 0}, +, 0, \cdot)$ over $\mathcal{M}_{\text{prob}}$

Unbounded Quantitative Predicates

Predicate before:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Predicate now:
$w : \Sigma \to \mathcal{W}$
$w(\sigma) = w([\![S]\!](\sigma))$

$$
\begin{array}{rcl}
C & \to & x \coloneqq E \quad \mid \quad C_1 \mathbin{;} C_2 \quad \mid \quad \otimes v \\
& \mid & \texttt{if}\ (\varphi)\ \{\, C_1 \,\}\ \texttt{else}\ \{\, C_2 \,\} \quad \mid \quad \texttt{while}\ (\varphi)\ \{C_1\} \quad \mid \quad \{\, C_1 \,\} \oplus \{\, C_2 \,\},
\end{array}
$$

Unbounded Quantitative Predicates

Predicate before:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Predicate now:
$w : \Sigma \to \mathcal{W}$
$w(\sigma) = w([\![S]\!](\sigma))$

Weighting via w in σ

$$C \;\to\; x \coloneqq E \;\mid\; C_1 \mathbin{;} C_2 \;\mid\; \otimes v$$
$$\mid\; \texttt{if}\,(\varphi)\,\{\,C_1\,\}\,\texttt{else}\,\{\,C_2\,\} \;\mid\; \texttt{while}\,(\varphi)\,\{C_1\} \;\mid\; \{\,C_1\,\} \oplus \{\,C_2\,\}\,,$$

$$\otimes 1 \mathbin{;}\; s \coloneqq 1$$
$$\otimes\big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big)$$
$$\{\,\otimes 1\,\} \oplus \{\,\otimes y \mathbin{;}\; s \coloneqq 1\,\}$$

Unbounded Quantitative Predicates

Predicate before:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Predicate now:
$w : \Sigma \to \mathcal{W}$
$w(\sigma) = w([\![S]\!](\sigma))$

Weighting via w in σ

$$C \;\to\; x \coloneqq E \;\mid\; C_1 \fatsemi C_2 \;\mid\; \otimes v$$
$$\mid\; \texttt{if}\,(\varphi)\,\{\,C_1\,\}\,\texttt{else}\,\{\,C_2\,\} \;\mid\; \texttt{while}\,(\varphi)\,\{C_1\} \;\mid\; \{\,C_1\,\} \oplus \{\,C_2\,\}\,,$$

$$\otimes 1 \fatsemi s \coloneqq 1$$

$$\{\,\otimes 1\,\} \oplus \{\,\otimes y \fatsemi s \coloneqq 1\,\}$$

Unbounded Quantitative Predicates

Predicate before:
$f : \Sigma \to \mathbb{R}^{\infty}_{\geq 0}$
$f(\sigma) = f([\![S]\!](\sigma))$

Predicate now:
$w : \Sigma \to \mathcal{W}$
$w(\sigma) = w([\![S]\!](\sigma))$

Weighting via w in σ

$$C \;\to\; x \coloneqq E \;\mid\; C_1 \fatsemi C_2 \;\mid\; \otimes v$$
$$\mid\; \texttt{if}\,(\varphi)\,\{\,C_1\,\}\,\texttt{else}\,\{\,C_2\,\} \;\mid\; \texttt{while}\,(\varphi)\,\{C_1\} \;\mid\; \{\,C_1\,\} \oplus \{\,C_2\,\}\,,$$

$$\otimes 1 \fatsemi s \coloneqq 1$$
$$\otimes\big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big)$$
$$\{\,\otimes 1\,\} \oplus \{\,\otimes y \fatsemi s \coloneqq 1\,\}$$

Weakest Preweightings

program S	$\texttt{wp}[\![S]\!](f)$	program S	$\texttt{wp}[\![S]\!](w)$
$x \coloneqq E$	$f[x/E]$	$x \coloneqq E$	$w[x/E]$
		$\otimes\, v$	$\otimes\, v$
$\texttt{if}\ (\varphi)\ \{\, S_1 \,\}\ \texttt{else}\ \{S_2\}$	$[\varphi]\texttt{wp}[\![S_1]\!](f) + [\neg\varphi]\texttt{wp}[\![S_2]\!](f)$	$\texttt{if}\ (\varphi)\ \{\, S_1 \,\}\ \texttt{else}\ \{S_2\}$	$[\varphi]\texttt{wp}[\![S_1]\!](w) \oplus [\neg\varphi]\texttt{wp}[\![S_2]\!](w)$
$S_1 \fatsemi S_2$	$\texttt{wp}[\![S_1]\!](\texttt{wp}[\![S_2]\!](f))$	$S_1 \fatsemi S_2$	$\texttt{wp}[\![S_1]\!](\texttt{wp}[\![S_2]\!](f))$
$\texttt{while}\ (\varphi)\ \{S\}$	$\operatorname{lfp} X.[\varphi]\texttt{wp}[\![S]\!](X) + [\neg\varphi]f$	$\texttt{while}\ (\varphi)\ \{S\}$	$\operatorname{lfp} X.[\varphi]\texttt{wp}[\![S]\!](X) \oplus [\neg\varphi]w$
$S_1\ [p]\ S_2$	$p \cdot \texttt{wp}[\![S_1]\!](f) + (1-p) \cdot \texttt{wp}[\![S_2]\!](f)$	$\{\, S_1 \,\} \oplus \{\, S_2 \,\}$	$\texttt{wp}[\![S_1]\!](w) \oplus \texttt{wp}[\![S_2]\!](w)$

Weakest Preweightings Semantics

Consider tropical semiring $\mathbb{S}_{\mathsf{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$: $= (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$

Weakest Preweightings Semantics

Consider tropical semiring $\mathbb{S}_{\mathsf{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$: $= (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$

$$\otimes 1 \fatsemi s \coloneqq 1 \qquad \mathtt{wp}[\![\otimes 1 \fatsemi s \coloneqq 1]\!](0) = 1 \qquad \mathtt{wp}[\![\ldots]\!]([s = 1] \cdot 1) = 2$$

Weakest Preweightings Semantics

Consider tropical semiring $\mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$: $= (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$

$$\otimes 1 \fatsemi s \coloneqq 1 \qquad \texttt{wp}[\![\otimes 1 \fatsemi s \coloneqq 1]\!](0) = 1 \qquad \texttt{wp}[\![\ldots]\!]([s = 1] \cdot 1) = 2$$

$$\otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \texttt{wp}[\![\ldots]\!](0) = 0 \text{ or } \infty$$

Weakest Preweightings Semantics

Consider tropical semiring $\mathbb{S}_{\mathsf{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$: $= (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$

$$\otimes 1 \fatsemi s \coloneqq 1 \qquad \mathtt{wp}[\![\otimes 1 \fatsemi s \coloneqq 1]\!](0) = 1 \qquad \mathtt{wp}[\![\ldots]\!]([s = 1] \cdot 1) = 2$$

$$\otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0 \text{ or } \infty$$

$$s \coloneqq 1 \fatsemi \otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0$$

Weakest Preweightings Semantics

Consider tropical semiring $\mathbb{S}_{\mathsf{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$: $= (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$

$$\otimes 1 \fatsemi s \coloneqq 1 \qquad \mathtt{wp}[\![\otimes 1 \fatsemi s \coloneqq 1]\!](0) = 1 \qquad \mathtt{wp}[\![\ldots]\!]([s = 1] \cdot 1) = 2$$

$$\otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0 \text{ or } \infty$$

$$s \coloneqq 1 \fatsemi \otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0$$

$$\{ \otimes 1 \} \oplus \{ \otimes y \fatsemi s \coloneqq 1 \} \qquad \mathtt{wp}[\![\ldots]\!](0) = 1 \oplus y$$

Weakest Preweightings Semantics

Consider tropical semiring $\mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$: $= (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$

$$\otimes 1 \fatsemi s \coloneqq 1 \qquad \mathtt{wp}[\![\otimes 1 \fatsemi s \coloneqq 1]\!](0) = 1 \qquad \mathtt{wp}[\![\ldots]\!]([s = 1] \cdot 1) = 2$$

$$\otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0 \text{ or } \infty$$

$$s \coloneqq 1 \fatsemi \otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0$$

$$\{ \otimes 1 \} \oplus \{ \otimes y \fatsemi s \coloneqq 1 \} \qquad \mathtt{wp}[\![\ldots]\!](0) = 1 \oplus y$$

- parametrized over module $\mathcal{W}$ over monoid $\mathcal{M}$

Weakest Preweightings Semantics

Consider tropical semiring $\mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$: $= (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$

$$\otimes 1 \fatsemi s \coloneqq 1 \qquad \mathtt{wp}[\![\otimes 1 \fatsemi s \coloneqq 1]\!](0) = 1 \qquad \mathtt{wp}[\![\ldots]\!]([s = 1] \cdot 1) = 2$$

$$\otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0 \text{ or } \infty$$

$$s \coloneqq 1 \fatsemi \otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0$$

$$\{ \otimes 1 \} \oplus \{ \otimes y \fatsemi s \coloneqq 1 \} \qquad \mathtt{wp}[\![\ldots]\!](0) = 1 \oplus y$$

- parametrized over module $\mathcal{W}$ over monoid $\mathcal{M}$
- undecidable in general (loops are hard)

Weakest Preweightings Semantics

Consider tropical semiring $\mathbb{S}_{\text{trop}} = (\mathbb{N}^\infty, \min, +, \infty, 0)$: $= (S, \oplus, \odot, \mathbb{0}, \mathbb{1})$

$$\otimes 1 \fatsemi s \coloneqq 1 \qquad \mathtt{wp}[\![\otimes 1 \fatsemi s \coloneqq 1]\!](0) = 1 \qquad \mathtt{wp}[\![\ldots]\!]([s = 1] \cdot 1) = 2$$

$$\otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0 \text{ or } \infty$$

$$s \coloneqq 1 \fatsemi \otimes \big([s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\big) \qquad \mathtt{wp}[\![\ldots]\!](0) = 0$$

$$\{ \otimes 1 \} \oplus \{ \otimes y \fatsemi s \coloneqq 1 \} \qquad \mathtt{wp}[\![\ldots]\!](0) = 1 \oplus y$$

- parametrized over module $\mathcal{W}$ over monoid $\mathcal{M}$
- undecidable in general (loops are hard)
- semi-automatic verification tools (like Caesar) exist for some instances of weighted programming

Lets Talk About Loops

$$\texttt{wp}[\![\texttt{while}\ (\varphi)\ \{S\}]\!](w) = \mathrm{lfp}\, X.\ [\varphi] \cdot \texttt{wp}[\![S]\!](X) \oplus [\neg\varphi] \cdot w$$

Kleene's Fixed Point Theorem

Set of *weightings* is ω-complete partially ordered and Φ_{post} is ω-continuous implies

$$\mathrm{lfp}\, \Phi_{\mathsf{post}} = \bigsqcup_{n \in \mathbb{N}} \Phi_{\mathsf{post}}^{n}(\bot).$$

Park Induction

Set of *weightings* is ω-complete partially ordered and Φ_{post} is ω-continuous implies

$$\Phi_{\mathsf{post}}(\mathsf{I}) \leq \mathsf{I} \implies \mathrm{lfp}\, \Phi_{\mathsf{post}} \leq I.$$

Weakest Preweightings of Translations

$$
\begin{aligned}
&\mathsf{Off} = \texttt{while}\ (m < n)\ \{\\
&\quad \texttt{if}\ (\texttt{true}) \to \{\, \odot 1\ \texttt{;}\ \texttt{skip} \,\}\\
&\quad \oplus\ (\texttt{true}) \to \{\, \odot y\ \texttt{;}\ s \coloneqq 1 \,\}\\
&\quad \oplus\ (\texttt{true}) \to \{\, \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \,\}\\
&\quad m \coloneqq m + 1\\
&\}.
\end{aligned}
$$

$$
\begin{aligned}
&\mathsf{On} = \texttt{while}\ (m < n)\ \{\\
&\quad \texttt{if}\ (c < y) \to \{\, \odot 1\ \texttt{;}\ \texttt{skip}\ \texttt{;}\ c \coloneqq c + 1 \,\}\\
&\quad \oplus\ (c = y) \to \{\, \odot y\ \texttt{;}\ s \coloneqq 1\ \texttt{;}\ c \coloneqq c + 1 \,\}\\
&\quad \oplus\ (c > y) \to \{\, \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty \,\}\\
&\quad m \coloneqq m + 1\\
&\}.
\end{aligned}
$$

Weakest Preweightings of Translations

$$\mathsf{Off} = \texttt{while}\ (m < n)\ \{\ \texttt{if}\ (\texttt{true}) \to \{\ \odot 1\ \texttt{;}\ \texttt{skip}\ \} \oplus (\texttt{true}) \to \{\ \odot y\ \texttt{;}\ s \coloneqq 1\ \} \oplus (\texttt{true}) \to \{\ \odot[s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\ \}\ m \coloneqq m + 1\ \}.$$

$$\texttt{wp}[\![\mathsf{Off}]\!](0) = n \oplus y = \min\{n, y\}$$

$$\mathsf{On} = \texttt{while}\ (m < n)\ \{\ \texttt{if}\ (c < y) \to \{\ \odot 1\ \texttt{;}\ \texttt{skip}\ \texttt{;}\ c \coloneqq c + 1\ \} \oplus (c = y) \to \{\ \odot y\ \texttt{;}\ s \coloneqq 1\ \texttt{;}\ c \coloneqq c + 1\ \} \oplus (c > y) \to \{\ \odot[s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\ \}\ m \coloneqq m + 1\ \}.$$

$$\texttt{wp}[\![\mathsf{On}]\!](0) = n \oplus 2y = \min\{n, 2y\}$$

Weakest Preweightings of Translations

$$\mathsf{Off} = \texttt{while}\ (m < n)\ \{$$
$$\quad \texttt{if}\ (\texttt{true}) \to \{\, \odot 1\ \texttt{;}\ \texttt{skip}\,\}$$
$$\quad \oplus\ (\texttt{true}) \to \{\, \odot y\ \texttt{;}\ s \coloneqq 1\,\}$$
$$\quad \oplus\ (\texttt{true}) \to \{\, \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\,\}$$
$$\quad m \coloneqq m + 1$$
$$\}.$$

$$\mathsf{On} = \texttt{while}\ (m < n)\ \{$$
$$\quad \texttt{if}\ (c < y) \to \{\, \odot 1\ \texttt{;}\ \texttt{skip}\ \texttt{;}\ c \coloneqq c + 1\,\}$$
$$\quad \oplus\ (c = y) \to \{\, \odot y\ \texttt{;}\ s \coloneqq 1\ \texttt{;}\ c \coloneqq c + 1\,\}$$
$$\quad \oplus\ (c > y) \to \{\, \odot [s = 1] \cdot 0 \oplus [s = 0] \cdot \infty\,\}$$
$$\quad m \coloneqq m + 1$$
$$\}.$$

$\mathtt{wp}[\![\mathsf{Off}]\!](0) = n \oplus y = \min\{n, y\}$

$\mathtt{wp}[\![\mathsf{On}]\!](0) = n \oplus 2y = \min\{n, 2y\}$

$$D = \sup_{R \in \mathcal{R}^n} \frac{\sigma_R(n \oplus 2y)}{\sigma_R(n \oplus y)} = \sup_{n>0,\ y>1} \frac{\min\{n, 2y\}}{\min\{n, y\}} = 2$$

Using Weakest Preweightings for Competitive Ratio

Online Optimization:

- for each optimization problem manual proofs by hand

Using Weakest Preweightings for Competitive Ratio

Online Optimization:

- for each optimization problem manual proofs by hand

Translation into Weighted Programming:

Using Weakest Preweightings for Competitive Ratio

Online Optimization:

- for each optimization problem manual proofs by hand

Translation into Weighted Programming:

- uniform framework

Using Weakest Preweightings for Competitive Ratio

Online Optimization:

- for each optimization problem manual proofs by hand

Translation into Weighted Programming:

- uniform framework
- semi-automatic verification via tools

Using Weakest Preweightings for Competitive Ratio

Online Optimization:

- for each optimization problem manual proofs by hand

Translation into Weighted Programming:

- uniform framework
- semi-automatic verification via tools
- further research in this area applies directly

Using Weakest Preweightings for Competitive Ratio

Online Optimization:

- for each optimization problem manual proofs by hand

Translation into Weighted Programming:

- uniform framework
- semi-automatic verification via tools
- further research in this area applies directly
- partial observability

Using Weakest Preweightings for Competitive Ratio

Online Optimization:

- for each optimization problem manual proofs by hand

Translation into Weighted Programming:

- uniform framework
- semi-automatic verification via tools
- further research in this area applies directly
- partial observability

problem-specific proofs by hand

$$\downarrow$$

$$D := \sup_{R \in \mathcal{R}^n} \frac{c(\mathsf{On}(R))}{c(\mathsf{Off}(R))} = \sup_{R \in \mathcal{R}^n} \frac{\mathtt{wp}[\![\mathtt{On}]\!](0)(\sigma_R)}{\mathtt{wp}[\![\mathtt{Off}]\!](0)(\sigma_R)}$$

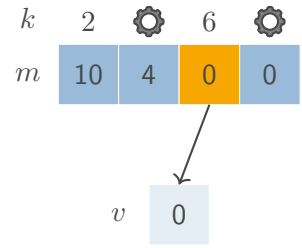
$$\uparrow$$

framework for semi-automatic verification

Proving Lock-Freedom


```
while (true) {
    ⨁_{j=1}^{N} { i := j };
    if (m[i] = v) {
        ⊗S ; v := k ; m[i] := v ;
    } else {
        ⊗F ; m[i] := v ; k := ⚙ ;
    }
}
```

S is success, F is failure.



Formal Languages Module

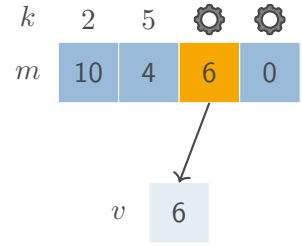
- alphabet $\Gamma = \{S, F\}$
- monoid $(\Gamma, \cdot, \epsilon)$, commutative monoid $(\Gamma^\omega, \cup, \emptyset)$
- scalar operation $\otimes : (g, G) \mapsto \{g \cdot g' \mid g' \in G\}$

The algorithm is *lock-free* if there is always progress. No progress is $F^\omega = FFF\cdots$.

Proving Lock-Freedom

```
while (true) {
    ⨁_{j=1}^{N} { i := j };
    if (m[i] = v) {
        ⊗S ; v := k ; m[i] := v ;
    } else {
        ⊗F ; m[i] := v ; k := ⚙ ;
    }
}
```

S is success, F is failure.



Formal Languages Module

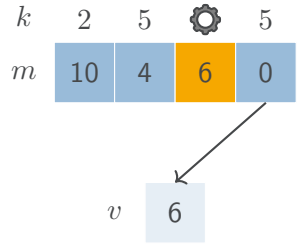
- alphabet $\Gamma = \{S, F\}$
- monoid $(\Gamma, \cdot, \epsilon)$, commutative monoid $(\Gamma^\omega, \cup, \emptyset)$
- scalar operation $\otimes : (g, G) \mapsto \{g \cdot g' \mid g' \in G\}$

The algorithm is *lock-free* if there is always progress. No progress is $F^\omega = FFF\cdots$.

Proving Lock-Freedom


```
while (true) {
    ⨁_{j=1}^{N} { i := j };
    if (m[i] = v) {
        ⊗S ; v := k ; m[i] := v ;
    } else {
        ⊗F ; m[i] := v ; k := ⚙ ;
    }
}
```

S is success, F is failure.



Formal Languages Module

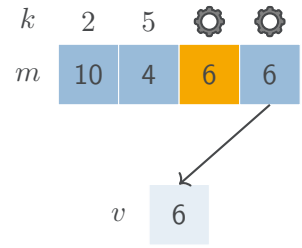
- alphabet $\Gamma = \{S, F\}$
- monoid $(\Gamma, \cdot, \epsilon)$, commutative monoid $(\Gamma^\omega, \cup, \emptyset)$
- scalar operation $\otimes : (g, G) \mapsto \{g \cdot g' \mid g' \in G\}$

The algorithm is *lock-free* if there is always progress. No progress is $F^\omega = FFF\cdots$.

Proving Lock-Freedom


```
while (true) {
    ⨁_{j=1}^{N} { i := j };
    if (m[i] = v) {
        ⊗S ; v := k ; m[i] := v ;
    } else {
        ⊗F ; m[i] := v ; k := ⚙ ;
    }
}
```

S is success, F is failure.



Formal Languages Module

- alphabet $\Gamma = \{S, F\}$
- monoid $(\Gamma, \cdot, \epsilon)$, commutative monoid $(\Gamma^\omega, \cup, \emptyset)$
- scalar operation $\otimes : (g, G) \mapsto \{g \cdot g' \mid g' \in G\}$

The algorithm is *lock-free* if there is always progress. No progress is $F^\omega = FFF\cdots$.

Proving Lock-Freedom


```
while (true) {
    ⨁_{j=1}^{N} { i := j };
    if (m[i] = v) {
        ⊗ S ; v := k ; m[i] := v ;
    } else {
        ⊗ F ; m[i] := v ; k := ⚙ ;
    }
}
```

$$\Phi_{\{\epsilon\}}(X) = \bigoplus_{j=1}^{N} \big([m[i] = v] \cdot S \otimes X[m[i]/v, v/k] \oplus [m[i] \neq v] \cdot F \otimes X[m[i]/v]\big)$$

$$\Phi_{\{\epsilon\}}(\Gamma^\omega) = \bigoplus_{j=1}^{N} \big([m[i] = v] \cdot \Gamma^\omega \oplus [m[i] \neq v] \cdot \Gamma^\omega\big)$$

$$\Phi^2_{\{\epsilon\}}(\Gamma^\omega) = \bigoplus_{j=1}^{N} \big([m[i] = v] \cdot \Gamma^\omega \oplus [m[i] \neq v] \cdot \Gamma^\omega\big)$$

S is success, F is failure.

The algorithm is *lock-free* if there is always progress. No progress is $F^\omega = FFF\cdots$.


Measurement
Entanglement
Schrödinger's Cat
Quantum Computing
Superposition
Qubit
Quantum state
Circuit
Bloch sphere

Qubit

Bit:
classical computer
value: 0 or 1

Qubit

Bit:
classical computer
value: 0 or 1

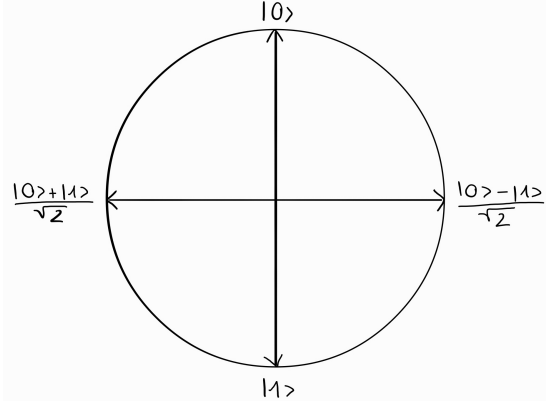
Qubit:
quantum computer
value: $|0\rangle$ or $|1\rangle$ or mixture of them
normalized vector in C^2

Qubit

Bit:
classical computer
value: 0 or 1

Qubit:
quantum computer
value: $|0\rangle$ or $|1\rangle$ or mixture of them
normalized vector in C^2

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

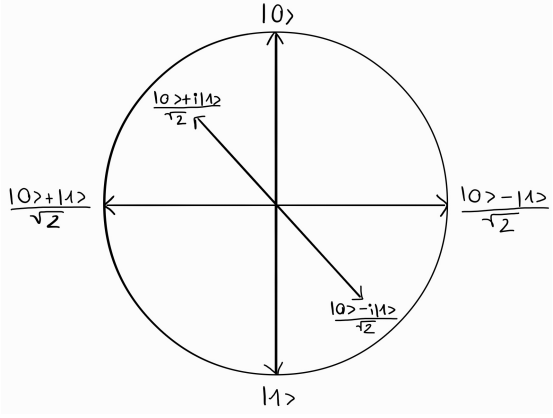


Qubit

Bit:
classical computer
value: 0 or 1

Qubit:
quantum computer
value: $|0\rangle$ or $|1\rangle$ or mixture of them
normalized vector in C^2

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$



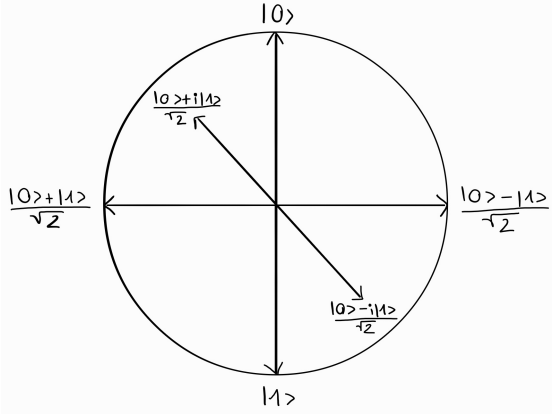
Qubit

Bit:
classical computer
value: 0 or 1

Qubit:
quantum computer
value: $|0\rangle$ or $|1\rangle$ or mixture of them
normalized vector in C^2

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

general: $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ with $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$



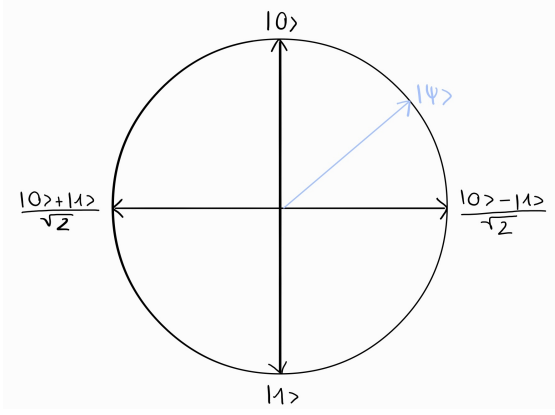
Qubit

Bit:
classical computer
value: 0 or 1

Qubit:
quantum computer
value: $|0\rangle$ or $|1\rangle$ or mixture of them
normalized vector in C^2

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

general: $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ with $\alpha, \beta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 = 1$



Measurements

$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle$ is in a **superposition** between $|0\rangle$ and $|1\rangle$ ($\approx$ linear combination)

We cannot look at a state and get values α, β.

Measurements

$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle$ is in a **superposition** between $|0\rangle$ and $|1\rangle$ ($\approx$ linear combination)

We cannot look at a state and get values α, β.

Solution: **Measurements**

- qubit changes to $|0\rangle$ with probability $|\alpha|^2$
- qubit changes to $|1\rangle$ with probability $|\beta|^2$

Measurements

$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle$ is in a **superposition** between $|0\rangle$ and $|1\rangle$ ($\approx$ linear combination)

We cannot look at a state and get values α, β.

Solution: **Measurements**

- qubit changes to $|0\rangle$ with probability $|\alpha|^2$
- qubit changes to $|1\rangle$ with probability $|\beta|^2$

Note: destroys previous state, states cannot be copied

Measurements

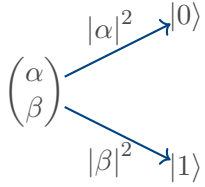
$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle$ is in a **superposition** between $|0\rangle$ and $|1\rangle$ ($\approx$ linear combination)

We cannot look at a state and get values α, β.

Solution: **Measurements**

- qubit changes to $|0\rangle$ with probability $|\alpha|^2$
- qubit changes to $|1\rangle$ with probability $|\beta|^2$

Note: destroys previous state, states cannot be copied



Measurements

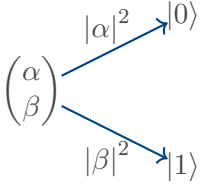
$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle$ is in a **superposition** between $|0\rangle$ and $|1\rangle$ ($\approx$ linear combination)

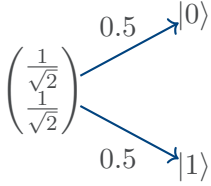
We cannot look at a state and get values α, β.

Solution: **Measurements**

- qubit changes to $|0\rangle$ with probability $|\alpha|^2$
- qubit changes to $|1\rangle$ with probability $|\beta|^2$

Note: destroys previous state, states cannot be copied





Measurements

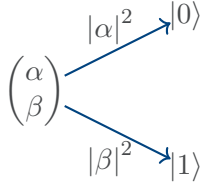
$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle$ is in a **superposition** between $|0\rangle$ and $|1\rangle$ ($\approx$ linear combination)

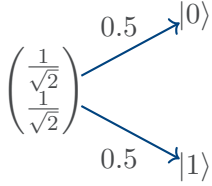
We cannot look at a state and get values α, β.

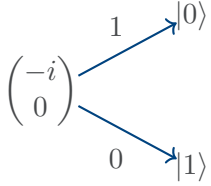
Solution: **Measurements**

- qubit changes to $|0\rangle$ with probability $|\alpha|^2$
- qubit changes to $|1\rangle$ with probability $|\beta|^2$

Note: destroys previous state, states cannot be copied







Measurements

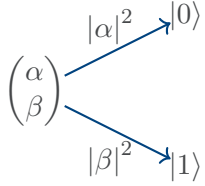
$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle$ is in a **superposition** between $|0\rangle$ and $|1\rangle$ ($\approx$ linear combination)

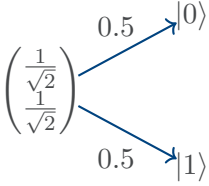
We cannot look at a state and get values α, β.

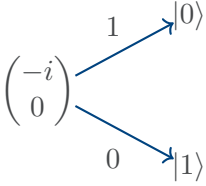
Solution: **Measurements**

- qubit changes to $|0\rangle$ with probability $|\alpha|^2$
- qubit changes to $|1\rangle$ with probability $|\beta|^2$

Note: destroys previous state, states cannot be copied







Schrödinger's Cat: A cat in a box is alive and dead at the same time and as soon as we open the box and look inside, it is either alive or dead ($\approx$ measurement)

Unitaries

How can we change states?

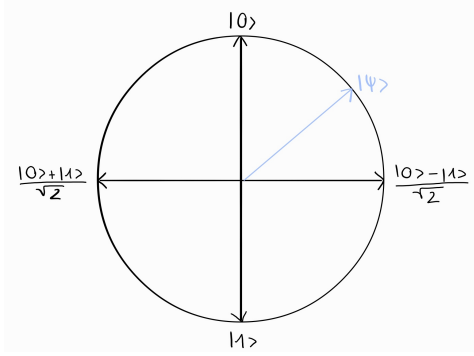
- state = vector
- (linear) changes = apply matrix (here unitary; i.e. easy reversible)

Unitaries

How can we change states?

- state = vector
- (linear) changes = apply matrix (here unitary; i.e. easy reversible)

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

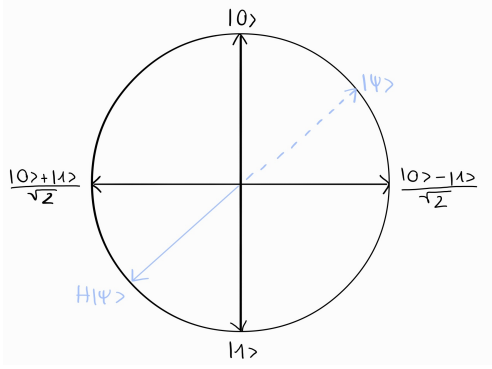


Unitaries

How can we change states?

- state = vector
- (linear) changes = apply matrix (here unitary; i.e. easy reversible)

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

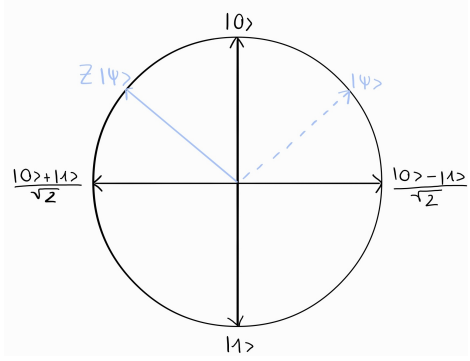


Unitaries

How can we change states?

- state = vector
- (linear) changes = apply matrix (here unitary; i.e. easy reversible)

$$H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \; Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \; \ldots$$



Multiple Quibts

Combine vectors using tensor product to describe state of bigger system: $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$

Multiple Quibts

Combine vectors using tensor product to describe state of bigger system: $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$

This is also written as $|01\rangle = |0\rangle \otimes |1\rangle$, $|11\rangle = |1\rangle \otimes |1\rangle$, ...

Multiple Quibts

Combine vectors using tensor product to describe state of bigger system: $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$

This is also written as $|01\rangle = |0\rangle \otimes |1\rangle$, $|11\rangle = |1\rangle \otimes |1\rangle$, ...
If we want to apply an unitary to one of them, e.g. H to second qubit, apply $I \otimes H$ to combined state.

Multiple Quibts

Combine vectors using tensor product to describe state of bigger system: $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$

This is also written as $|01\rangle = |0\rangle \otimes |1\rangle$, $|11\rangle = |1\rangle \otimes |1\rangle$, ...
If we want to apply an unitary to one of them, e.g. H to second qubit, apply $I \otimes H$ to combined state.

There are also unitaries that act on both qubits together, e.g. $CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Multiple Quibts

Combine vectors using tensor product to describe state of bigger system: $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$

This is also written as $|01\rangle = |0\rangle \otimes |1\rangle$, $|11\rangle = |1\rangle \otimes |1\rangle$, ...
If we want to apply an unitary to one of them, e.g. H to second qubit, apply $I \otimes H$ to combined state.

There are also unitaries that act on both qubits together, e.g. $CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Intuition: $CNOT\,|xy\rangle = |x, y \oplus x\rangle$ for $x, y \in \{0, 1\}$, i.e. toggle second qubit if first one is $|1\rangle$

Entanglement

Definition: Two qubits in state $\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$ that cannot be written in form $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix}$.

Entanglement

Definition: Two qubits in state $\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$ that cannot be written in form $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix}$.

Example: $\beta_{00} = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$

Entanglement

Definition: Two qubits in state $\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$ that cannot be written in form $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix}$.

Example: $\beta_{00} = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$

How can we get such a state?

$CNOT(H \otimes I)\,|00\rangle$

Entanglement

Definition: Two qubits in state $\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$ that cannot be written in form $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix}$.

Nicer way to write such a sequence: **circuit**

Example: $\beta_{00} = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$

How can we get such a state?
$CNOT(H \otimes I)\,|00\rangle$

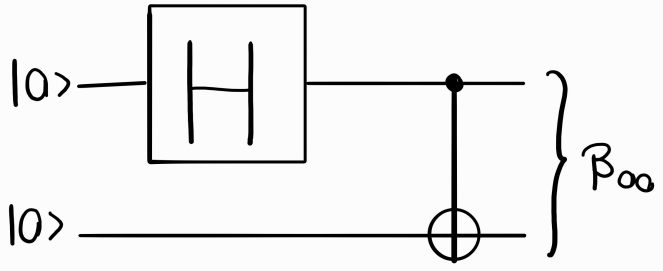
Entanglement

Definition: Two qubits in state $\begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{pmatrix}$ that cannot be written in form $\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix}$.

Example: $\beta_{00} = \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{pmatrix}$

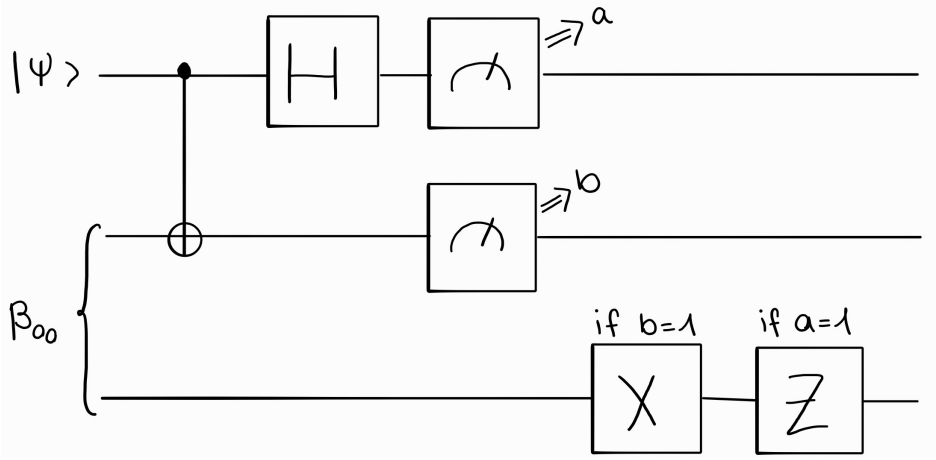
How can we get such a state?
$CNOT(H \otimes I)\,|00\rangle$

Nicer way to write such a sequence: **circuit**



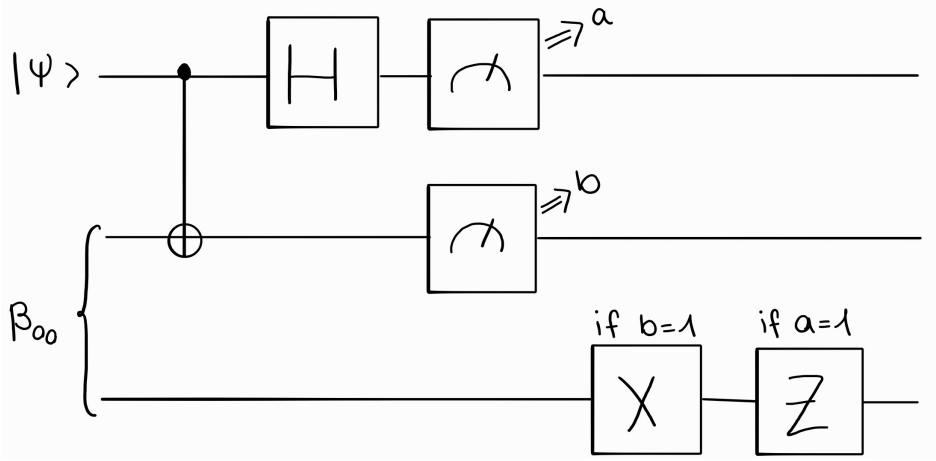
Teleportation Circuit

But we are computer scientists and want to use if-branches, loops, ...



Teleportation Circuit

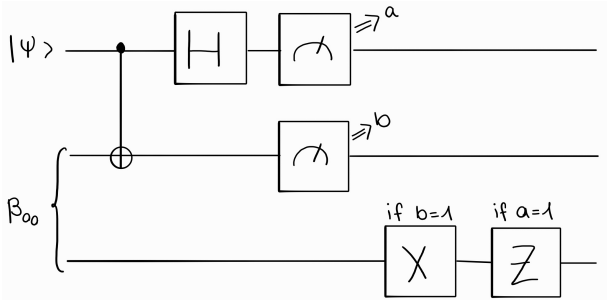
But we are computer scientists and want to use if-branches, loops, ...



not a nice way to enter this into a computer, we want a programming language

Teleportation Program

$q_1 q_2 \coloneqq CNOT(q_1 q_2)$;
$q_1 \coloneqq H q_1$;
if (measure $q_2 = 1$){
    $q_3 \coloneqq X q_3$
}
else {
    skip
};
if (measure $q_1 = 1$){
    $q_3 \coloneqq Z q_3$
}
else {
    skip
}



Programming Language

program S	
skip	skip
$q \coloneqq \lvert 0\rangle$	init
$q \coloneqq Uq$	apply unitary
`if` (measure q) $\{ S_1 \}$ `else` $\{ S_2 \}$	measurement with branching
$S_1 \mathbin{;} S_2$	concatenation
`while` (measure q) $\{S\}$	while loop with induced measurement

Verification

WP similar to probabilistic case [DP06, Yin12]

- Predicate: 1- bounded observable (special kind of matrix)
- Expected value: $\langle\psi| P |\psi\rangle \in [0,1]$ where $\langle\psi| = \overline{|\psi\rangle}^T$
  "The probability that P holds in state $|\psi\rangle$"

Verification

WP similar to probabilistic case [DP06, Yin12]

- Predicate: 1- bounded observable (special kind of matrix)
- Expected value: $\langle\psi| P |\psi\rangle \in [0,1]$ where $\langle\psi| = \overline{|\psi\rangle}^T$
  "The probability that P holds in state $|\psi\rangle$"

Example: $P = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ means "the qubit has value $|0\rangle$"

- state $|0\rangle$: $\langle 0| P |0\rangle = 1$
- state $|1\rangle$: $\langle 1| P |1\rangle = 0$
- state $|+\rangle$: $\langle +| P |+\rangle = 0.5$

Verification

WP similar to probabilistic case [DP06, Yin12]

- Predicate: 1- bounded observable (special kind of matrix)
- Expected value: $\langle\psi| P |\psi\rangle \in [0,1]$ where $\langle\psi| = \overline{|\psi\rangle}^T$
  "The probability that P holds in state $|\psi\rangle$"

Example: $P = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ means "the qubit has value $|0\rangle$"

- state $|0\rangle$: $\langle 0| P |0\rangle = 1$
- state $|1\rangle$: $\langle 1| P |1\rangle = 0$
- state $|+\rangle$: $\langle +| P |+\rangle = 0.5$

With $P = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ is $\langle\psi| P |\psi\rangle = 1$ for all states $|\psi\rangle$.

Verification

WP similar to probabilistic case [DP06, Yin12]

- Predicate: 1- bounded observable (special kind of matrix)
- Expected value: $\langle\psi| P |\psi\rangle \in [0,1]$ where $\langle\psi| = \overline{|\psi\rangle}^T$
  "The probability that P holds in state $|\psi\rangle$"

Example: $P = |0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ means "the qubit has value $|0\rangle$"

- state $|0\rangle$: $\langle 0| P |0\rangle = 1$
- state $|1\rangle$: $\langle 1| P |1\rangle = 0$
- state $|+\rangle$: $\langle +| P |+\rangle = 0.5$

With $P = I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ is $\langle\psi| P |\psi\rangle = 1$ for all states $|\psi\rangle$.

With $P = \mathbf{0} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$ is $\langle\psi| P |\psi\rangle = 0$ for all states $|\psi\rangle$.

Weakest Precondition

program S	$\texttt{wp}[\![S]\!](P)$
skip	P
$q := \|0\rangle$	$\|0\rangle\langle 0\| P \|0\rangle\langle 0\| + \|1\rangle\langle 0\| P \|0\rangle\langle 1\|$
$q := Uq$	$U^\dagger P U$
$\texttt{if}\ (\text{measure}\ q)\ \{ S_1 \}\ \texttt{else}\ \{ S_2 \}$	$\|0\rangle\langle 0\| \texttt{wp}[\![S_1]\!](P) \|0\rangle\langle 0\| + \|1\rangle\langle 1\| \texttt{wp}[\![S_2]\!](P) \|1\rangle\langle 1\|$
$S_1 \fatsemi S_2$	$\texttt{wp}[\![S_1]\!](\texttt{wp}[\![S_2]\!](P))$
$\texttt{while}\ (\text{measure}\ q)\ \{S\}$	$\bigvee_{n=0}^{\infty} P_n$ with $P_0 = \mathbf{0}$
	$P_{n+1} = \|0\rangle\langle 0\| P \|0\rangle\langle 0\| + \|1\rangle\langle 1\| \texttt{wp}[\![S]\!](P_n) \|1\rangle\langle 1\|$

$q_2 \coloneqq |0\rangle\,;$
$q_3 \coloneqq |0\rangle\,;$
$q_2 \coloneqq Hq_2\,;$
$q_2q_3 \coloneqq CNOT(q_2q_3)\,;$
$q_1q_2 \coloneqq CNOT(q_1q_2)\,;$
$q_1 \coloneqq Hq_1\,;$
if (measure $q_2 = 1$){
    $q_3 \coloneqq Xq_3$
};
if (measure $q_1 = 1$){
    $q_3 \coloneqq Zq_3$
}

$q_2 \coloneqq |0\rangle\,;$
$q_3 \coloneqq |0\rangle\,;$
$q_2 \coloneqq Hq_2\,;$
$q_2q_3 \coloneqq CNOT(q_2q_3)\,;$

β_{00}

$q_1q_2 \coloneqq CNOT(q_1q_2)\,;$
$q_1 \coloneqq Hq_1\,;$
if (measure $q_2 = 1$){
  $q_3 \coloneqq Xq_3$
};
if (measure $q_1 = 1$){
  $q_3 \coloneqq Zq_3$
}

teleportation

$q_2 \coloneqq |0\rangle\,;$
$q_3 \coloneqq |0\rangle\,;$
$q_2 \coloneqq Hq_2\,;$
$q_2q_3 \coloneqq CNOT(q_2q_3)\,;$

β_{00}

$q_1q_2 \coloneqq CNOT(q_1q_2)\,;$
$q_1 \coloneqq Hq_1\,;$
if (measure $q_2 = 1$){
  $q_3 \coloneqq Xq_3$
};
if (measure $q_1 = 1$){
  $q_3 \coloneqq Zq_3$
}

teleportation

// $I \otimes I \otimes |\psi\rangle\langle\psi|$

// $|\psi\rangle\langle\psi| \otimes \mathsf{I} \otimes \mathsf{I}$

$q_2 \coloneqq |0\rangle$;
$q_3 \coloneqq |0\rangle$;
$q_2 \coloneqq H q_2$;
$q_2 q_3 \coloneqq CNOT(q_2 q_3)$;

β_{00}

$q_1 q_2 \coloneqq CNOT(q_1 q_2)$;
$q_1 \coloneqq H q_1$;
if (measure $q_2 = 1$){
  $q_3 \coloneqq X q_3$
};
if (measure $q_1 = 1$){
  $q_3 \coloneqq Z q_3$
}

teleportation

// $\mathsf{I} \otimes \mathsf{I} \otimes |\psi\rangle\langle\psi|$

Extensions & Current Work - Quantum

Extensions:

- Weakest **liberal** preconditions (to include non-terminating runs) [Yin12]
- Classical variables [FY21]
- Conditioning [GUK25]
- ...

Work in progress:

- Translate quantum programs to probabilistic programs to use existing tools for automated verification (especially Caesar)
- Extend observables to unbounded operators, i.e. move from predicates to expectations as done in the probabilistic case

Thanks! Any questions?

References I

Ellie D'hondt and Prakash Panangaden.
Quantum weakest preconditions.
page 429–451, 2006.

Yuan Feng and Mingsheng Ying.
Quantum hoare logic with classical variables.
ACM Transactions on Quantum Computing, 2(4), December 2021.

Christina Gehnen, Dominique Unruh, and Joost-Pieter Katoen.
Bayesian Inference in Quantum Programs.
In *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*, volume 334 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 157:1–157:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

Mingsheng Ying.
Floyd–hoare logic for quantum programs.
ACM Trans. Program. Lang. Syst., 2012.