Overview

1. What is probabilistic programming?
2. Probabilistic Guarded Command Language
3. Weakest preconditions
4. Conditioning
5. Expected runtime analysis
6. Analysing Bayesian networks
Dijkstra’s guarded command language

- **skip**
- **diverge**
- **x := E**
- **prog1 ; prog2**
- **if (G) prog1 else prog2**
- **prog1 [] prog2**
- **while (G) prog**

- empty statement
- divergence
- assignment
- sequential composition
- choice
- non-deterministic choice
- iteration
Weakest preconditions

Weakest precondition

[Dijkstra 1975]

An predicate $F$ maps program states onto Booleans, i.e., $F : \Sigma \rightarrow \mathbb{B}$. Let $F \leq G$ if and only if $F \implies G$.

A predicate transformer is a total function between two predicates.

The predicate transformer $wp(P, F)$ for program $P$ and postcondition $F$ yields the "weakest" precondition $E$ on the initial state of $P$ ensuring that the execution of $P$ terminates in a final state satisfying $F$.

Weakest preconditions correspond to so-called total correctness.

A weakest liberal precondition $wlp(P, F)$ yields the weakest predicate for which $P$ either does not terminate or establishes $F$. It does not ensure termination and corresponds to partial correctness.
Weakest precondition $G$ w.r.t. postcondition $F$

This holds for deterministic programs. (A non-deterministic program starting in $\neg G$ may also terminate in a state satisfying $F$.)
Weakest liberal precondition $G$ w.r.t. $F$

This holds for deterministic programs. A nondeterministic program starting in $\neg G$ may also terminate in a state satisfying $F$ or diverge.
## Weakest preconditions

**Predicate transformer semantics of Dijkstra’s GCL**

### Syntax

- **skip**
- **diverge**
- **x := E**
- **P1 ; P2**
- **if (G)P1 else P2**
- **P1 [] P2**
- **while (G)P**

### Semantics $wp(P, F)$

- $F$
- $false$
- $F[x := E]$
- $wp(P_1, wp(P_2, F))$
- $(G \land wp(P_1, F)) \lor (\neg G \land wp(P_2, F))$
- $wp(P_1, F) \land wp(P_2, F)$
- $lfp \ X. ((G \land wp(P, X)) \lor (\neg G \land F))$

$lfp$ is the least fixed point wrt. the ordering $\Rightarrow$ on predicates.

$\Phi(x)$

wlp-semantics differs from wp-semantics only for **while** and **diverge**. How?
A loopy program example

```java
while (x > 0) {
    x--;  
}
```

What is the weakest pre-condition on \( x \) such that on termination \( x \) is non-negative?
\[
\text{while } (x > 0) \{ x \leftarrow 3 \} \quad F = x \geq 0
\]

\[
\Phi(x) = (x > 0 \land \wp (x \leftarrow 3, x)) \lor (x \leq 0 \land x > 0)
\]

if \( \Phi(x) \) computation:

0. \( \Phi^0(false) = false \)

1. \( \Phi^1(false) = x > 0 \land \underbrace{\wp (x \leftarrow 3, false)}_{false} \land \underbrace{x = 0}_{false} \)

\[
= x = 0
\]

2. \( \Phi^2(false) = \Phi (x = 0) \)

\[
= (x > 0 \land \underbrace{\wp (x \leftarrow 3, x = 0)}_{x - 1 = 0 \land x = 1} \land x = 0) \land \underbrace{x = 1}_{true} \land \underbrace{x = 0}_{false} \]

\[
= x = 1 \lor x = 0
\]

\[
\Phi^k(false) = x = k - 1 \lor x = k - 2 \lor \ldots \lor x = 0
\]

\[
\sup_{n \in \mathbb{N}} \Phi^n(false) = x \geq 0
\]
Elementary properties of Dijkstra’s \( \text{wp} \)

- **Continuity:** \( \text{wp}(P, F) \) is continuous, that is for every chain:
  \[
  F = F_0 \Rightarrow F_1 \Rightarrow F_2 \Rightarrow \ldots \quad \text{wp}(P, \text{sup } F) = \text{sup } \text{wp}(P, F)
  \]

- **Monotonicity:** \( F \Rightarrow G \) implies \( \text{wp}(P, F) \Rightarrow \text{wp}(P, G) \)

- **Duality:** \( \text{wlp}(P, F) = \text{wp}(P, F) \lor \neg \text{wp}(P, \text{true}) \)

- **Strictness:** \( \text{wp}(P, \text{false}) = \text{false} \) and \( \text{wlp}(P, \text{true}) = \text{true} \)

- **Distribution**: \( \text{wp}(P, F \lor G) = \text{wp}(P, F) \lor \text{wp}(P, G) \)

  \[
  \text{wp}(P, \text{true}) = \text{weakest condition under which } P \text{ terminates}
  \]

\(^6\)For deterministic \( P \).
Probalistic GCL

- **skip**
- **diverge**
- **x := E**
- **observe (G)**
- **prog1 ; prog2**
- **if (G) prog1 else prog2**
- **prog1 [p] prog2**
- **while (G) prog**

Conditioning will be treated later. For the moment: no conditioning.
Examples: Intuition

1. Let program \( P \) be:

\[
x := 5 \quad \text{[4/5]} \quad x := 10
\]

The expected value of \( x \) on \( P \)'s termination is: \( \frac{4}{5} \cdot 5 + \frac{1}{5} \cdot 10 = 6 \)
Examples: Intuition

1. Let program $P$ be:
   \[ x := 5 \ [4/5] \ x := 10 \]
   The expected value of $x$ on $P$’s termination is: \( \frac{4}{5} \cdot 5 + \frac{1}{5} \cdot 10 = 6 \)

2. Let program $Q$ be:
   \[ x := x+5 \ [4/5] \ x := 10 \]
   The expected value of $x$ on $Q$’s termination is: \( \frac{4}{5} \cdot (x+5) + \frac{1}{5} \cdot 10 = \frac{4x}{5} + 6 \)
Examples: Intuition

1. Let program $P$ be:

\[
x := 5 \quad \text{[4/5]} \quad x := 10
\]

The expected value of $x$ on $P$’s termination is: $\frac{4}{5} \cdot 5 + \frac{1}{5} \cdot 10 = 6$

2. Let program $Q$ be:

\[
x := x+5 \quad \text{[4/5]} \quad x := 10
\]

The expected value of $x$ on $Q$’s termination is: $\frac{4}{5} \cdot (x+5) + \frac{1}{5} \cdot 10 = \frac{4x}{5} + 6$

3. The probability that $x = 10$ on $Q$’s termination is:

\[
\frac{4}{5} \cdot [x+5 = 10] + \frac{1}{5} \cdot 1 = \frac{4 \cdot [x = 5] + 1}{5}
\]
Weakest pre-expectations \[\text{[Kozen 1981, McIver & Morgan 2004]}\]

An expectation\(^7\) maps states onto \(\mathbb{R}_{\geq 0} \cup \{\infty\}\). It is the quantitative analogue of a predicate. Let \(f \leq g\) iff \(f(s) \leq g(s)\), for every state \(s\).

An expectation transformer is a total function between two expectations.

The transformer \(\text{wp}(P, f)\) yields the least expectation \(e\) on \(P\)'s initial state ensuring that \(P\) terminates with expectation \(f\).

Annotation \(\{e\} P \{f\}\) holds for total correctness iff \(e \leq \text{wp}(P, f)\).

Weakest liberal pre-expectation \(\text{wlp}(P, f) = \text{"wp}(P, f) + \text{Pr}[P \text{ diverges}]\)".

\(^7\) Expectations in probability theory. Think of it as a random variable.
### Expectation transformer semantics of pGCL

**Syntax**

- `skip`
- `diverge`
- `x := E`
- `P1 ; P2`
- `if (G)P1 else P2`
- `P1 [p] P2`
- `while (G)P`

**Semantics** \( wp(P, f) \)

- `f`
- `0`
- \( f[x := E] \)
- \( wp(P_1, wp(P_2, f)) \)
- \( [G] \cdot wp(P_1, f) + [\neg G] \cdot wp(P_2, f) \)
- \( p \cdot wp(P_1, f) + (1-p) \cdot wp(P_2, f) \)
- \( lfp \ X. ([G] \cdot wp(P, X) + [\neg G] \cdot f) \)

\( lfp \) is the least fixed point operator wrt. the ordering \( \leq \) on expectations.

wlp-semantics differs from wp-semantics only for `while` and `diverge`.

\[ \triangleleft \text{ defined over } \{ f \mid f: \Sigma \rightarrow [0, 1] \} \]
Weakest preconditions

Examples

1. Let program $P$ be:

   $x := 5 \ [4/5] \ x := 10$

   For $f = x$, we have:

   $wp(P, x) = \frac{4}{5} \cdot wp(x := 5, x) + \frac{1}{5} \cdot wp(x := 10, x) = \frac{4}{5} \cdot 5 + \frac{1}{5} \cdot 10 = 6$
Examples

1. Let program $P$ be:
   \[
   x := 5 \quad [4/5] \quad x := 10
   \]
   For $f = x$, we have
   \[
   wp(P, x) = \frac{4}{5} \cdot wp(x := 5, x) + \frac{1}{5} \cdot wp(x := 10, x) = \frac{4}{5} \cdot 5 + \frac{1}{5} \cdot 10 = 6
   \]

2. Let program $P'$ be:
   \[
   x := x + 5 \quad [4/5] \quad x := 10
   \]
   For $f = x$, we have:
   \[
   wp(P', x) = \frac{4}{5} \cdot wp(x + := 5, x) + \frac{1}{5} \cdot wp(x := 10, x) = \frac{4}{5} \cdot (x + 5) + \frac{1}{5} \cdot 10 = \frac{4x}{5} + 6
   \]
Weakest preconditions

Examples

1. Let program $P$ be:

\[ x := 5 \quad [4/5] \quad x := 10 \]

For $f = x$, we have

\[ wp(P, x) = \frac{4}{5} \cdot wp(x := 5, x) + \frac{1}{5} \cdot wp(x := 10, x) = \frac{4}{5} \cdot 5 + \frac{1}{5} \cdot 10 = 6 \]

2. Let program $P'$ be:

\[ x := x+5 \quad [4/5] \quad x := 10 \]

For $f = x$, we have:

\[ wp(P', x) = \frac{4}{5} \cdot wp(x +:= 5, x) + \frac{1}{5} \cdot wp(x := 10, x) = \frac{4}{5} \cdot (x+5) + \frac{1}{5} \cdot 10 = \frac{4x}{5} + 6 \]

3. For program $P'$ (again) and $f = [x = 10]$, we have:

\[ wp(P', [x=10]) = \frac{4}{5} \cdot wp(x := x+5, [x=10]) + \frac{1}{5} \cdot wp(x := 10, [x=10]) \]

\[ = \frac{4}{5} \cdot [x+5 = 10] + \frac{1}{5} \cdot [10 = 10] \]

\[ = \frac{4[x=5]+1}{5} \]

\[ = 1 \]
Continuation passing

\[ \text{wp } [C_1] \left( \text{wp } [C_2] \left( F \right) \right) \]

- weakest precondition of \( C_1 \) with respect to \( \text{wp } [C_2] \left( F \right) \)
- or in other words:
  - weakest precondition of \( C_1 \) with respect to \( F \)

\[ \text{wp } [C_2] \left( F \right) \]

- weakest precondition of \( C_2 \) with respect to \( F \)

\[ F \]

- postcondition \( F \) evaluated in final states after termination of \( C_2 \)
Weakest preconditions

\[
x := 0 \quad [1/2] \quad x := 1; \quad // \text{command } c1
\]
\[
y := 0 \quad [1/3] \quad y := 1; \quad // \text{command } c2
\]

Take postexpectation \( f = [x=y] \)

\[
\frac{1}{6} + \frac{2}{6}
\]
\[
\left[ \frac{1}{2} \cdot \left( \frac{1}{3} \ [0=0] + \frac{2}{3} \ [0=1] \right) \right.
\]
\[
+ \ \frac{1}{2} \left( \frac{1}{3} \ [1=0] + \frac{2}{3} \ [1=1] \right)
\]
\[
\frac{1}{2} \ x := 0 \quad + \quad \frac{1}{2} \ g(x := 1)
\]
\[
x := 0 \quad [\frac{1}{2}] \quad x := 1
\]
\[
\frac{1}{3} \ [x=0] + \frac{2}{3} \ [x=1] \quad \leftarrow \text{call this } g
\]
\[
y := 0 \quad [\frac{1}{3}] \quad y := 1
\]
\[
[x=y]
\]
Loops

\[
wp(\text{while } (G)\{ P \}, f) = \text{lfp } X. (\underbrace{[G] \cdot wp(P, X)}_{\Phi_f(X)} + [\neg G] \cdot f)
\]

Scott continuity of \( \Phi \)

The expectation transformer \( \Phi_f \) (defined as above) is continuous on expectations ordered by \( \leq \).

Corollary

By Kleene’s fixpoint theorem, it follows \( \text{lfp } \Phi_f = \sup_{n \in \mathbb{N}} \Phi_f^n(0) \).

\( \Phi_f^n(0) \) is the expected value over the final states of running while \((G)\{ P \}\) exactly \( n \) times when starting with the constant expectation \( 0 \).
A simple loopy program example

\[
x := 0;
\text{while} (c) \{
\quad \{ c := 0 \} [0.5] \{ x++ \}
\}
\]

What is the expected value of \(x\) on termination?
\[ \wp ( \text{while} \ (c) \ \{ c:=0 \ [ \frac{1}{2} ] \ x++ \}) \]

\[ \Phi^\circ(x) = \left[ c=1 \right] \ wp ( c:=0 \ [ \frac{1}{2} ] \ x++ , X ) + [c=1] \cdot x \]

\[ = \ldots \text{calculate} \ldots \]

\[ = \left[ c=1 \right] \left( \frac{1}{2} \cdot X ( c:=0 ) + \frac{1}{2} X ( x:=x+1 ) \right) + [c=1] \cdot x \]

Iterating:
\[ \Phi^0 (0) = 0 \]
\[ \Phi^1 (0) = [c=1] \cdot x \]
\[ \Phi^2 (0) = ( [c=1] \cdot x ) \]
\[ = \left[ c=1 \right] \left( \frac{1}{2} \cdot [c=1] \cdot x ( c:=0 ) + \frac{1}{2} [c=1] \cdot x ( x++ ) \right) + [c=1] \cdot x \]
\[ = \left[ c=1 \right] \left( \frac{1}{2} \cdot x + \frac{1}{2} [c=1] ( x+1 ) \right) + [c=1] \cdot x \]
\[ = \left[ c=1 \right] \cdot \frac{1}{2} \cdot x + [c=1] \cdot x \]
\[ \Phi^3 (0) = \left( \left[ c=1 \right] \frac{1}{2} \cdot x + [c=1] \cdot x \right) \]
\[ = \left[ c=1 \right] \cdot \left( \frac{1}{2} \cdot x + \frac{1}{4} ( x+1 ) \right) + [c=1] \cdot x \]
\[ \Phi^n (0) = \left[ c=1 \right] \cdot \sum_{0 \leq i < n} \left( \frac{1}{2} \right)^i ( x+i-1 ) + [c=1] \cdot x \]
\[
wp ( \text{while } (c) \{ c := 0 \ [\forall u \ x := x + 3 \}, x) \\
\sum_{i=1}^{8} \left( \frac{1}{2} \right)^i (x + i - 1) + [c \neq 1] \cdot x
\]

\[
wp (x := 0 ; \text{loop})
\]

\[
\sum_{i=1}^{8} \left( \frac{1}{2} \right)^i (i-1)
\]

know: \[
\sum_{i=1}^{N} p^i(i-1) = \frac{p^2}{(1-p)^2} \quad \text{for } |p| < 1
\]

\[
[c=1] \cdot 1
\]
**Elementary properties of probabilistic wp**

- **Continuity**: \( wp(P, f) \) is continuous, that is
  
  for every chain: \( F = f_0 \leq f_1 \leq f_2 \leq \ldots \) \( wp(P, \sup F) = \sup wp(P, F) \)

- **Monotonicity**: \( f \leq g \) implies \( wp(P, f) \leq wp(P, g) \)

- **Linearity**: \( wp(P, r \cdot f + g) = r \cdot wp(P, f) + wp(P, g) \) for every \( r \in \mathbb{R}_{\geq 0} \)

- **Duality**: \( wlp(P, f) = wp(P, f) + (1 - wp(P, 1)) \)

- **Strictness**: \( wp(P, 0) = 0 \) and \( wlp(P, 1) = 1 \)

\[
wp(P, 1) = \text{termination probability of program } P
\]
Backward compatibility

The wp-semantics of pGCL is a **conservative extension** of Dijkstra's wp-semantics. For any **ordinary** GCL program $P$ and predicate $F \in \mathbb{P}$:

$$\underbrace{wp(P, [F])}_{\text{pGCL}} = \left[ \underbrace{wp(P, F)}_{\text{Dijkstra}} \right]$$
Weakest pre-expectations = expected rewards

For every pGCL program \( P \), input \( s \) and expectation \( f \):

\[
wp(P, f)(s) = \text{ER}^{\lbrack P \rbrack}(\langle P, s \rangle, \text{sink})
\]

The \( wp(P, f) \) for input \( s \) equals the expected reward to reach terminal state \( \text{sink} \) in MC \( \lbrack P \rbrack \) where rewards \( r \) are defined by: \( r(\langle \downarrow, s' \rangle) = f(s') \) and 0 otherwise.
Weakest pre-expectations = expected rewards

For every pGCL program $P$, input $s$ and expectation $f$:

$$wp(P, f)(s) = ER^{[P]}(\langle P, s \rangle, \Diamond \text{sink})$$

The $wp(P, f)$ for input $s$ equals the expected reward to reach terminal state $\text{sink}$ in $\text{MC \[ P \]}$ where rewards $r$ are defined by: $r(\langle \downarrow, s' \rangle) = f(s')$ and 0 otherwise.

For finite-state programs, wp-reasoning can be done with model checkers such as PRISM and Storm (www.stormchecker.org).
Probabilistic model checking

Program $P$

finite-state Markov chain $[P]$

Result $\checkmark$ $\times$

Quantitative results

Counter-example

System requirements

Probabilistic temporal logic specification

$P_{<0.1} [\Diamond \text{fail}]$

stormchecker.org

storm
Results Probabilistic Model Checker Competition 2019

http://qcomp.org/competition/2019/
The lost passenger ticket puzzle

- Passengers are queueing to board a fully-booked airplane

- All – except only the first – passengers have their boarding pass

- The first (pass-less) passenger randomly picks a seat

- All other passengers proceed as:
  - take your seat, if the seat on boarding pass is free
  - otherwise, randomly pick a seat

Q: how likely does the last passenger get the seat on her boarding pass?
The lost passenger ticket problem

```
E := 1000;
roll{
    1/E   :: b := true;
    1-1/E :: b := false;
}
E := E-1;

while (E > 1){
    if (!b) {
        roll {
            1/E   :: roll{
                1/E   :: b := true;
                1-1/E :: b := false;
            }
            1-1/E :: skip}
        E := E-1;
    }
    return b
```

<table>
<thead>
<tr>
<th>E</th>
<th>time (in s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.1</td>
</tr>
<tr>
<td>1,000</td>
<td>0.1</td>
</tr>
<tr>
<td>10,000</td>
<td>0.2</td>
</tr>
<tr>
<td>1,000,000</td>
<td>6.4</td>
</tr>
<tr>
<td>10,000,000</td>
<td>66.8</td>
</tr>
</tbody>
</table>
Weakest preconditions

**Induction for upper bounds: Park’s lemma**

\[
wp(\text{while } (G)\{P\}, f) = \text{lfp } X. (\underbrace{[G] \cdot wp(P, X)}_{\Phi_f(X)} + \underbrace{[\neg G] \cdot f)}
\]

**Upper bounds on weakest pre-expectations**

For while\((G)\{P\}\) and expectations \(f\) and \(I\) we have:

\[
\Phi_f(I) \leq I \quad \text{implies} \quad wp(\text{while}(G)\{P\}, f) \leq I
\]

Every wp-superinvariant of a loop for \(f\) is an upper bound to the wp of the loop (and \(f\)).
Example

\[
\text{while } (c = 1) \{ x++ \ [\frac{1}{2}] \ c := 0 \} \quad f = x
\]

\[
\Phi_f(x) = [c \neq 1] \cdot x + [c = 1] \left( \frac{1}{2} \times (x++) + \frac{1}{2} \times (c := 0) \right)
\]

\[
\text{let } I = x + [c = 1]
\]

Induction: \[
\Phi(I) = [c \neq 1] \cdot x + [c = 1] \cdot \frac{1}{2} \left( x+1 + [c = 1] + x \right)
\]
\[
= [c \neq 1] \cdot x + [c = 1] \cdot \frac{1}{2} \left( 2x+1 \right)
\]
\[
= [c \neq 1] \cdot x + \frac{1}{2} [c = 1] + [c = 1] \cdot x
\]
\[
= x + \frac{1}{2} [c = 1]. \quad \text{Thus } \Phi(I) \leq I \quad \text{and } I \text{ is a superinvariant}.
\]
Conditioning

\[ P(A|B) = \frac{P(B|A)P(A)}{P(B)} \]
Extending wp-semantics of \textit{pGCL}

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics $wp(P, f)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>$f$</td>
</tr>
<tr>
<td>diverge</td>
<td>0</td>
</tr>
<tr>
<td>$x := E$</td>
<td>$f[x := E]$</td>
</tr>
<tr>
<td>observe $(G)$</td>
<td>$[G] \cdot f$</td>
</tr>
<tr>
<td>$P_1 ; P_2$</td>
<td>$wp(P_1, wp(P_2, f))$</td>
</tr>
<tr>
<td>if $(G)P_1$ else $P_2$</td>
<td>$[G] \cdot wp(P_1, f) + [\neg G] \cdot wp(P_2, f)$</td>
</tr>
<tr>
<td>$P_1 [p] P_2$</td>
<td>$p \cdot wp(P_1, f) + (1-p) \cdot wp(P_2, f)$</td>
</tr>
<tr>
<td>while $(G)P$</td>
<td>lfp $X$. $( [G] \cdot wp(P, X) + [\neg G] \cdot f )$</td>
</tr>
</tbody>
</table>

\textit{lfp} is the least fixed point operator wrt. the ordering $\leq$ on expectations.

\textit{wlp-semantics} differs from \textit{wp-semantics} only for \textbf{while} and \textbf{diverge}.
The piranha puzzle

\[
f_1 := \text{gf \ [0.5] } f_1 := \text{pir}; \\
f_2 := \text{pir}; \\
s := f_1 \ [0.5] s := f_2; \\
\text{observe } (s = \text{pir})\
\]

What is the probability that the original fish in the bowl was a piranha?

Consider the expected reward of successful termination without violating any observation

\[
\mathbb{E}^{[P]}(s_0, \Diamond \langle \text{sink} \rangle \mid \neg \Diamond \langle \ell \rangle) = \frac{1 \cdot 1/2 + 0 \cdot 1/4}{1 - 1/4} = \frac{1/2}{3/4} = \frac{2}{3}.
\]
The piranha program – a wp perspective

\[
\begin{align*}
\text{f1} &:= \text{gf} [0.5] \ 	ext{f1} := \text{pir}; \\
\text{f2} &:= \text{pir}; \\
\text{s} &:= \text{f1} [0.5] \ 	ext{s} := \text{f2}; \\
\text{observe} \ (s = \text{pir})
\end{align*}
\]

What is the probability that the original fish in the bowl was a piranha?

\[
\mathbb{E}(f1 = \text{pir} \mid \text{“feasible” run}) = \frac{1\cdot1/2 + 0\cdot1/4}{1 - 1/4} = \frac{1/2}{3/4} = \frac{2}{3}.
\]

Let \(cwp(P, f) = \frac{wp(P, f)}{wlp(P, 1)}\). In fact \(cwp(P, f) = (wp(P, f), wlp(P, 1))\).

Note: \(wlp(P, 1) = 1 - Pr[P \text{ violates an observation}]\). This includes diverging runs.
Q: What is the probability that $x = 1$ on termination?

Consider the two programs:

- $x := 1$ [0.5] diverge
- $x := 1$ [0.5] observe($false$)

For the left program this is $\frac{1}{2}$; for the right one this is $1$.

$$cwp(P_{\text{left}}, ([x=1], 1)) = (\frac{1}{2}, 1) \quad cwp(P_{\text{right}}, (\cdot)) = (\frac{1}{2}, \frac{1}{2})$$
Divergence matters

diverge [0.5] {
  x := 0 [0.5] x := 1;
  y := 0 [0.5] y := 1;
  observe (x = 0 || y = 0)
}

What is the probability that $y = 0$ on termination?

We:

$\text{wp}(P, f)\text{wp}(P, 1) = \frac{2}{7}$

In contrast:

$\text{wp}(P, f)\text{wp}(P, 1) = \frac{2}{3}$

Then:

observe $G$ while (!G) skip

Warning: This is a silly example. Typically divergence comes from loops.

Joost-Pieter Katoen
Principles of Probabilistic Programming
Divergence matters

\[
\text{\texttt{diverge \, [0.5] \{}} \\
\quad \text{\texttt{x := 0 \, [0.5] \, x := 1;}} \\
\quad \text{\texttt{y := 0 \, [0.5] \, y := 1;}} \\
\quad \text{\texttt{observe (x = 0 \lor y = 0)}} \\
\text{\texttt{\}}} 
\]

Q: What is the probability that \( y = 0 \) on termination?

We:\ \frac{wp(P, f)}{wlp(P, 1)} = \frac{2}{7}

In contrast:\ \frac{wp(P, f)}{wp(P, 1)} = \frac{2}{3}

Then:

\text{\texttt{observe \, (G) \equiv \, while(!G) \, skip}}

Warning: This is a silly example. Typically divergence comes from loops.
Observations inside loops

- Certain divergence
- \((wp(P_{left}, f), wlp(P_{left}, 1)) = (0, 1)\)
- Conditional \(wp = 0\)

- Divergence with probability zero
- \((wp(P_{right}, f), wlp(P_{right}, 1)) = (0, 0)\)
- Conditional \(wp = undefined\)
Observations inside loops

```
int x := 1;
while (x = 1) {
    x := 1
}
```

- Certain divergence
- \((wp(P_{left}, f), wlp(P_{left}, 1)) = (0, 1)\)
- Conditional \(wp = 0\)

```
int x := 1;
while (x = 1) {
    x := 1 [0.5] x := 0;
    observe (x = 1)
}
```

- Divergence with probability zero
- \((wp(P_{right}, f), wlp(P_{right}, 1)) = (0, 0)\)
- Conditional \(wp = \text{undefined}\)

We do distinguish these programs.
Contextual equivalence?

\[ P : \{ x := 0 \} \frac{1}{2} \{ x := 1 \}; \text{observe}(x = 1) \]
\[ Q : \{ x := 0; \text{observe}(x = 1) \}\frac{1}{2} \{ x := 1; \text{observe}(x = 1) \} \]

Of course

\[
\frac{\text{wp}(P, [x = 1])}{\text{wlp}(P, 1)} = \frac{\text{wp}(Q, [x = 1])}{\text{wlp}(Q, 1)} = \frac{1/2}{1/2} = 1
\]

\[ \llbracket P \rrbracket = \llbracket Q \rrbracket \]

note: \( \text{wlp}(P, \cdot) = \text{wp}(P, \cdot) \)
and the same holds for \( Q \)
Contextual equivalence?

\[ P : \{x := 0\} [1/2] \{x := 1\}; \text{observe}(x = 1) \]
\[ Q : \{x := 0; \text{observe}(x = 1)\} [1/2] \{x := 1; \text{observe}(x = 1)\} \]

Of course
\[
\frac{\text{wp}(P, [x = 1])}{\text{wlp}(P, 1)} = \frac{\text{wp}(Q, [x = 1])}{\text{wlp}(Q, 1)} = \frac{1/2}{1/2} = 1
\]

but we cannot decompose
\[
\frac{\text{wp}(Q, [x = 1])}{\text{wlp}(Q, 1)} \neq 0.5 \frac{\text{wp}(Q_1, [x = 1])}{\text{wlp}(Q_1, 1)} + 0.5 \frac{\text{wp}(Q_2, [x = 1])}{\text{wlp}(Q_2, 1)}
\]

This all motivates that we deal with pairs rather than fractions.
Backward compatibility

We have seen earlier:

Mclver’s wp-semantics is a conservative extension of Dijkstra’s wp-semantics.

For any ordinary (aka: GCL) program $P$ and predicate $F$:

$$\left[ wp(P, [F]) \right] = wp(P, F)$$

Mclver  Dijkstra
Backward compatibility

We have seen earlier:

Mclver’s wp-semantics is a **conservative extension** of Dijkstra’s wp-semantics.

For any **ordinary** (aka: GCL) program $P$ and predicate $F$:

$$\left[ wp(P, [F]) \right]_{\text{Mclver}} = wp(P, F)_{\text{Dijkstra}}$$

The cwp-semantics is a **conservative extension** of Mclver’s wp-semantics.

For any **observe-free** pGCL program $P$ and expectation $f$:

$$cwp(P, (f, 1)) = (f', g') \text{ implies } \frac{f'}{g'} = wp(P, f)$$

and similar for liberal pre-expectations.
**Conditional $wp = \text{conditional expected rewards}$**

For program $P$, input $s$ and expectation $f$:

\[
\frac{wp(P, f)(s)}{wl\rho(P, \mathbf{1})(s)} = ER^{\parallel P\parallel}(\langle P, s \rangle, \langle \Diamond(sink) \mid \neg \Diamond(\mathbf{f}) \rangle)
\]

The ratio of $wp(P, f)$ over $wl\rho(P, \mathbf{1})$ for input $s$ equals\(^8\) the conditional expected reward to reach a successful terminal state $\langle sink \rangle$ while satisfying all observations in $P$’s MC when starting with $s$. (The rewards in MC $\parallel P\parallel$ are defined as before.)

---

\(^8\)Either both sides are equal or both sides are undefined.
**Program transformation to remove conditioning**

- **Idea:** `restart` an infeasible run until all `observe`-statements are passed

- For program variable \( x \) use auxiliary variable \( sx \)
  - store initial value of \( x \) into \( sx \)
  - on each new loop-iteration restore \( x \) to \( sx \)

- Use auxiliary variable `flag` to signal observation violation:

  ```
  flag := true; while(flag) { flag := false; mprog }
  ```

- Change `prog` into `mprog` by:
  - `observe(G)` \( \rightarrow \) `flag := !G || flag`
  - `diverge` \( \rightarrow \) `if(!flag) diverge`
  - `while(G) prog` \( \rightarrow \) `while(G && !flag) prog`
Resulting program

sx1, ..., sxn := x1, ..., xn; flag := true;
while (flag) {
    flag := false;
    x1, ..., xn := sx1, ..., sxn;
    modprog
}

In machine learning, this is known as rejection sampling.
Removal of conditioning

the transformation in action:

\[
\begin{align*}
\text{sx, sy := x, y; flag := true;}
\text{while (flag)} \{ \\
\quad x, y := \text{sx, sy; flag := false;}
\quad x := 0 \ [p] \ x := 1;
\quad y := 0 \ [p] \ y := 1;
\quad \text{flag := (x = y)}
\}
\end{align*}
\]

a simple data-flow analysis yields:

\[
\begin{align*}
\text{repeat} \{ \\
\quad x := 0 \ [p] \ x := 1;
\quad y := 0 \ [p] \ y := 1
\} \ \text{until} (x \neq y)
\end{align*}
\]
Removal of conditioning

Correctness of transformation

For program $P$ that has at least one feasible run, transformed program $\hat{P}$, and expectation $f$:

\[ cwp(P, (f, 1)) = wp(\hat{P}, f) \]
A dual program transformation

repeat
  a0 := 0 [0.5] a0 := 1;
  a1 := 0 [0.5] a1 := 1;
  a2 := 0 [0.5] a2 := 1;
  i := 4*a2 + 2*a1 + a0 + 1
until (1 <= i <= 6)

Loop-by-observe replacement if there is “no data flow” between loop iterations
Independent and identically distributed loops

**iid-Loop**

Loop `while (G)P` is iid if and only if for any expectation `f`:

\[
wp(P, [G] \cdot wp(P, f)) = wp(P, [G]) \cdot wp(P, f)
\]

Event that `G` holds after `P` is independent of the expected value of `f` after `P`.

**Correctness of transformation**

For iid-loop `repeat P until (G)` and expectations `f, g` we have:

\[
cwp\left(repeat P until (G),(f,g)\right) = cwp(P ; observe (G),(f,g))
\]

Loop-free programs are easier to reason about — no loop invariants.
Take-home messages

- Expectations are the quantitative analogue of predicates
- Conditioning involves considering program divergence
- Weakest preconditions for programs with observes correspond to conditional expected rewards
- Conditioning is equivalent to a loop
- Conditioning can be “pushed” backwards through the program

Next lecture: Termination.