Probabilistic Programming

Lecture #5: Weakest Preconditions

Joost-Pieter Katoen

RWTH Lecture Series on Probabilistic Programming 2018

Overview

Overview

Approaches to semantics

- Operational semantics: (developed by Plotkin)
  - The meaning of a program in terms of how it executes on an abstract machine.
  - Useful for modelling the execution behaviour of a program.
- Axiomatic semantics: (developed by Floyd and Hoare)
  - Provides correctness assertions for each program construct.
  - Useful for verifying that a program's computed results are correct with respect to the specification.
- Denotational semantics: (developed by Strachey and Scott)
  - Provides a mapping of language constructs onto mathematical objects.
  - Useful for obtaining an abstract insight into the working of a program.

Today: denotational semantics of Dijkstra's GCL in terms of weakest preconditions. No probabilities yet.

Next lecture: how to extent preconditions to the probabilistic setting.

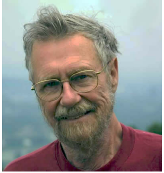
Code-level reasoning

Proving properties of programs: not by executing them,
but by reasoning at the syntax level of programs.

Compositionality: determine the correctness of composed program P
by reasoning about its parts in isolation and
then obtain P's correctness result by combining those parts' analyses.

Overview

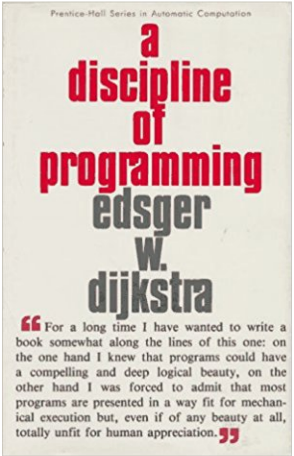
Dijkstra's guarded command language

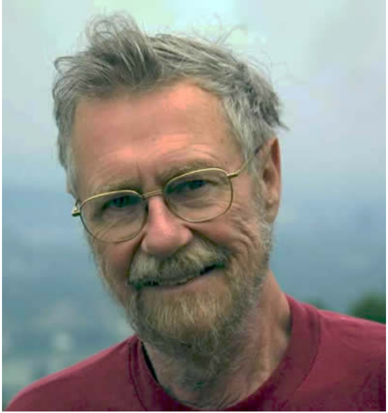


- `skip` — empty statement
- `diverge` — divergence
- `x := E` — assignment
- `prog1 ; prog2` — sequential composition
- `if (G) prog1 else prog2` — choice
- `prog1 [] prog2` — non-deterministic choice
- `while (G) prog` — iteration

For simplicity: we omit non-deterministic choice.

A discipline of programming





Some preliminaries

- Variable valuation $s : Vars \rightarrow \mathbb{Q}$ maps each program variable onto a value (here: rational numbers)

- Let $\mathbb{S}$ denote the set of variable valuations.

- Let $[\![E]\!]$ denote the valuation of expression E

- The indicator function of guard G is denoted by $[G]$:

$$[G](s) = \begin{cases} 1 & \text{if } s \models G \\ 0 & \text{if } s \not\models G \end{cases}$$

These are also known as Iverson brackets.

Predicate transformers

Predicates

A predicate F maps program states onto Booleans, i.e., $F : \mathbb{S} \rightarrow \mathbb{B}$.

Let $\mathbb{P}$ denote the set of all predicates and $F \sqsubseteq G$ if and only if $F \Rightarrow G$.

$(\mathbb{P}, \sqsubseteq)$ is a complete lattice.

Proof.

Predicate F equals $\{ s \in \mathbb{S} \mid s \models F \}$. Thus $P = 2^{\mathbb{S}}$. Partial order $\sqsubseteq$ equals $\subseteq$. □

Predicate transformer

A predicate transformer is a total function between predicates.

Examples

- Predicates are sets of variable valuations.
- Program statements can be viewed as predicate transformers
- One is interested in preconditions that are least restrictive

Overview

Weakest preconditions

Weakest precondition

For program P and $E, F \in \mathbb{P}$, the predicate transformer $wp(P, \cdot) : \mathbb{P} \rightarrow \mathbb{P}$ is defined by $wp(P, F) = E$ if and only if when P starts in an initial state satisfying E it holds:

1. the execution of P terminates in a state satisfying F, and
2. for any $H \in \mathbb{P}$ such that P terminates in a state satisfying F, $H \Rightarrow E$.

$wp(P, F)$ is called the weakest precondition on the initial state of P such that P terminates in a final state satisfying the postcondition F.

Weakest preconditions correspond to so-called total correctness.

Examples.

$F = (a \cdot b = 0)$

$P = \underbrace{a \mathrel{+}:= 1}_{a = a+1} ; \; b \mathrel{-}:= 1$

$a = a+1 \; ; \; b = b-1$

$wp(P, F) = a = -1 \lor b = 1$

$P = \text{if } (x \geq 0) \; \{ y := z+1 \} \text{ else } \{ y := z+2 \}$

$F = (y \geq 0)$

$wp(P,F) = (x \geq 0 \land z+1 \geq 0)$

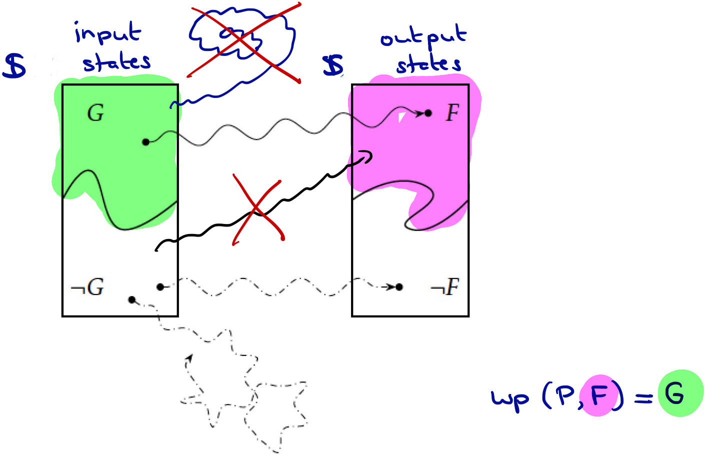
$\lor \; (x < 0 \land z+2 \geq 0)$

$P = \text{while } (x \neq 0) \; \{ x \mathrel{-}:= 1 \}$

$F = \text{true}$

$wp(P, F) = x \geq 0$

Weakest precondition G w.r.t. postcondition F



This holds for every deterministic program.

Weakest preconditions

Consider the program P and postcondition $F \in \mathbb{P}$.

Then $wp(P, F) = E$ means:

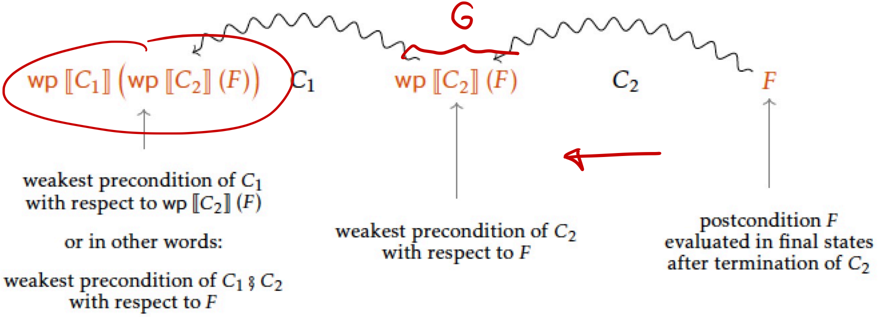
1. From any state $s \models E$, the program P terminates in some state $t \models F$.
2. From any state $s \not\models E$, it holds:
   2.1 either P terminates in a state $t \not\models F$
   2.2 or P does not terminate at all.

Backward reasoning

$P = C_1 ; C_2$

postcondition $F \in \mathbb{P}$

$wp(C_1, wp(C_2, F))$



Weakest preconditions reason in a backward manner about programs.

Predicate transformer semantics of Dijkstra's GCL

program P

Syntax

```
skip
diverge
x := E
P ; Q
if (G) P else Q
while (G)P
```

2x+z

= if (G) P; while (G) P else skip

$\underbrace{x^2 + y > 0}_{F}$

$wp(P, F)$

F

false

$F[x := [\![E]\!]]$

$wp(P, wp(Q, F))$

$(G \wedge wp(P,F)) \vee (\neg G \wedge wp(Q,F))$

$(2x+z)^2 + y > 0$

$F\ [x := 2x+z]$

Predicate transformer semantics of Dijkstra's GCL

Syntax	Semantics $wp(P, F)$
`skip`	F
`diverge`	$false$
`x := E`	$F[x := E]$
`P ; Q`	$wp(P, wp(Q, F))$
`if (G) P else Q`	$(G \wedge wp(P, F)) \vee (\neg G \wedge wp(Q, F))$
wp(`while (G)P`, F)	$\mathrm{lfp}\, X.\ ((G \wedge wp(P, X)) \vee (\neg G \wedge F))$

"skip"

$= \Phi(X)$

Predicate transformer semantics of Dijkstra's GCL

Syntax	Semantics $wp(P, F)$
`skip`	F
`diverge`	$false$
`x := E`	$F[x := E]$ ←
`P ; Q`	$wp(P, wp(Q, F))$
`if (G) P else Q`	$(G \wedge wp(P, F)) \vee (\neg G \wedge wp(Q, F))$
`while (G)P`	$\text{lfp}\, X.\ \underbrace{((G \wedge wp(P, X)) \vee (\neg G \wedge F))}_{\Phi(X)}$

lfp is the least fixed point wrt. the ordering $\sqsubseteq$ = $\Rightarrow$ on the set $\mathbb{P}$ of predicates.

$(\mathbb{P}, \sqsubseteq)$
↳ $\Rightarrow$

Loop-free examples

a. $wp(x := 1,\ x = 1) \stackrel{\text{def}}{=} x = 1\,(x := 1) = 1 = 1 = \text{true}$

b. $wp(x +:= 1,\ x = 3) = x = 3\,(x +:= 1) = x + 1 = 3$

$\iff x = 2$

c. $wp(\underbrace{x +:= 1 ; y -:= 1}_{P},\ \underbrace{x \leq y}_{F})$

$= wp(x +:= 1,\ \underbrace{wp(y -:= 1,\ x \leq y)}_{x \leq y - 1})$

$= x + 1 \leq y - 1$

Loops

$$wp(\text{while } (G)\{\, P \,\}, F) \;=\; \text{lfp}\, X.\, \underbrace{((G \wedge wp(P, X)) \vee (\neg G \wedge F))}_{\Phi(X)}$$

Scott continuity of Φ

The function $\Phi : \mathbb{P} \rightarrow \mathbb{P}$ (defined as above) is continuous on $(\mathbb{P}, \sqsubseteq)$.

Proof.

By structural induction on the program P. □

smallest element in $(\mathbb{P}, \sqsubseteq)$

Corollary

By Kleene's fixpoint theorem, it follows $\text{lfp}\ \Phi = \sup_{n \in \mathbb{N}} \Phi^n(\text{false})$.

$\Phi^n(\text{false})$ denotes the wp of running while $(G)\{\, P \,\}$ exactly n times starting from the empty set of states.

A loopy program example

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

What is the weakest pre-condition on x
such that on termination x is non-negative?

$x \geqslant 0$

while (x>0) { x-- } $F = x \geq 0$

$$\Phi(X) = (x > 0 \wedge wp(x--, X)) \vee (x \leq 0 \wedge x \geq 0)$$

lfp $\Phi(X)$

$$\Phi^0(\text{false}) = \text{false}$$

$$\Phi^1(\text{false}) = x > 0 \wedge \underbrace{wp(x--, \text{false})}_{\text{false}} \vee x = 0$$

$$= x = 0$$

$$\Phi^2(\text{false}) = \Phi(x=0)$$

$$= (x > 0 \wedge \underbrace{wp(x--, x=0)}_{\substack{x-1=0 \\ x=1}}) \vee x = 0$$

$$= x = 1 \vee x = 0$$

$$\Phi^k(\text{false}) = x = k-1 \vee x = k-2 \vee \dots \vee x = 0$$

$$\sup_{n \in \mathbb{N}} \Phi^n(\text{false}) = x \geq 0$$

Approximating while-loops

Let:

$$\begin{aligned} \text{while}^{0}(G)\{\, P \,\}) &= \text{diverge} \\ \text{while}^{n+1}(G)\{\, P \,\}) &= \text{if } (G) \text{ then } P; \text{while}^{n}(G)\{\, P \,\}) \text{ else skip} \end{aligned}$$

Approximating while-loops

Let:

$$\begin{aligned}\text{while}^{0}(G)\{\,P\,\}) &= \text{diverge}\\ \text{while}^{n+1}(G)\{\,P\,\}) &= \text{if } (G) \text{ then } P; \text{while}^{n}(G)\{\,P\,\}) \text{ else skip}\end{aligned}$$

Let $\Phi(X) = ((G \wedge wp(P, X)) \vee (\neg G \wedge F))$. Then for all $n \in \mathbb{N}$ it holds:

$$\Phi^{n}(\text{false}) = wp(\text{while}^{n}(G)\{\,P\,\}, F)$$

Proof.

By induction on n using the inductive definition of wp. □

Overview

$P = \text{if } (y>0)\ \{x:=5\}\ \text{else}\ \{x:=2\};$
$y:=x-3;\ \text{skip}$

$F = y^2>2$

$= y>0$

$(y>0 \wedge \text{true}) \vee (y \leq 0 \wedge \text{false})$

$\text{if } (y>0)\ \{$

$(5-3)^2>2 \quad \text{true}$

$x:=5$

$(x-3)^2>2$

$\}\ \text{else}\ \{$

$(2-3)^2>2 = \text{false}$

$x:=2$

$(x-3)^2>2$

$\}\ (x-3)^2>2$

$y:=x-3$

$y^2>2$

$\text{skip} \quad y^2>2$

Weakest liberal preconditions

Weakest liberal precondition

For program P and $E, F \in \mathbb{P}$, the predicate transformer $wlp(P, \cdot) : \mathbb{P} \rightarrow \mathbb{P}$ is defined by $wlp(P, F) = E$ if and only if when P starts in an initial state satisfying E it holds:

partial correctness

$wp(P,F) = E$ — if initial state $\models E$, P terminates in F

→ total correctness

Weakest liberal preconditions

Weakest liberal precondition

For program P and $E, F \in \mathbb{P}$, the predicate transformer $wlp(P, \cdot) : \mathbb{P} \rightarrow \mathbb{P}$ is defined by $wlp(P, F) = E$ if and only if when P starts in an initial state satisfying E it holds:

1. either P diverges or P terminates in a state satisfying F, and

as for $wp(P, F)$

Weakest liberal preconditions

Weakest liberal precondition

For program P and $E, F \in \mathbb{P}$, the predicate transformer $wlp(P, \cdot) : \mathbb{P} \to \mathbb{P}$ is defined by $wlp(P, F) = E$ if and only if when P starts in an initial state satisfying E it holds:

1. either P diverges or P terminates in a state satisfying F, and
2. for any $H \in \mathbb{P}$ such that P either diverges or terminates in a state satisfying F, $H \Rightarrow E$.

E is the weakest predicate satisfying 1.

Weakest liberal preconditions

Weakest liberal precondition

For program P and $E, F \in \mathbb{P}$, the predicate transformer $wlp(P, \cdot) : \mathbb{P} \to \mathbb{P}$ is defined by $wlp(P, F) = E$ if and only if when P starts in an initial state satisfying E it holds:

1. either P diverges or P terminates in a state satisfying F, and
2. for any $H \in \mathbb{P}$ such that P either diverges or terminates in a state satisfying F, $H \Rightarrow E$.

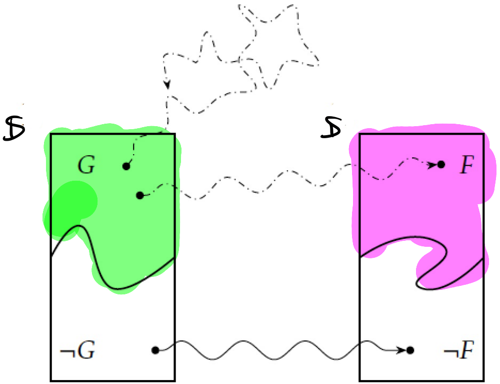
$wlp(P, F)$ is called the weakest liberal precondition on the initial state of P such that P either diverges or terminates in a final state satisfying the postcondition F.

Weakest liberal preconditions correspond to so-called partial correctness: a program is either correct, or diverges. $wlp(x{+}{:=}1, x>0)$

Examples. $wlp(\text{diverge}, F) = \text{true}$ $= wlp(x{+}{:=}1, x>0)$

Weakest **liberal** precondition G w.r.t. F

wlp (P, F) = G



This holds for every deterministic program.

Weakest liberal preconditions

Consider the program P and postcondition $F \in \mathbb{P}$.

Then $wlp(P, F) \;=\; E$ means:

1. From any state $s \models E$,
   1.1 either the program P terminates in a state $t \models F$
   1.2 or does not terminate at all.
2. From any state $s \not\models E$, the program P terminates in a state $t \not\models F$

Weakest liberal preconditions for Dijkstra's GCL

Syntax	Semantics $wlp(P, F)$
`skip`	F
`diverge`	$true$
`x := E`	$F[x := E]$
`P ; Q`	$wlp(P, wlp(Q, F))$
`if (G) P else Q`	$(G \wedge wlp(P, F)) \vee (\neg G \wedge wlp(Q, F))$
`while (G)P`	$\text{gfp } X. \, ((G \wedge wlp(P, X)) \vee (\neg G \wedge F))$

gfp is the greatest fixed point wrt. the ordering $\sqsubseteq \, = \, \Rightarrow$ on the set $\mathbb{P}$ of predicates.

Loops

$$wlp(\text{while } (G)\{ P \}, F) \;=\; \text{gfp}\, X.\; \underbrace{((G \wedge wlp(P, X)) \vee (\neg G \wedge F))}_{\Phi(X)}$$

Scott continuity of Φ

The function $\Phi : \mathbb{P} \to \mathbb{P}$ (defined as above) is continuous on $(\mathbb{P}, \sqsubseteq)$.

Corollary

By Kleene's fixpoint theorem, it follows $\text{gfp}\ \Phi = \inf_{n \in \mathbb{N}} \Phi^n(\text{true})$.

$\Phi^n(\text{true})$ denotes the wp of running while $(G)\{ P \}$ exactly n times starting from the entire set $\mathbb{S}$ of states.

Elementary properties of Dijkstra's wp and wlp

- Monotonicity: $F \Rightarrow G$ implies $wp(P, F) \Rightarrow wp(P, G)$

Elementary properties of Dijkstra's wp and wlp

- Monotonicity: $F \Rightarrow G$ implies $wp(P, F) \Rightarrow wp(P, G)$
- Duality: $wlp(P, F) = wp(P, F) \vee \neg wp(P, true)$

Elementary properties of Dijkstra's wp and wlp

- Monotonicity: $F \Rightarrow G$ implies $wp(P, F) \Rightarrow wp(P, G)$
- Duality: $wlp(P, F) = wp(P, F) \vee \neg wp(P, true)$
- Strictness: $wp(P, false) = false$ and $wlp(P, true) = true$

Elementary properties of Dijkstra's wp and wlp

- Monotonicity: $F \Rightarrow G$ implies $wp(P, F) \Rightarrow wp(P, G)$
- Duality: $wlp(P, F) = wp(P, F) \vee \neg wp(P, true)$
- Strictness: $wp(P, false) = false$ and $wlp(P, true) = true$
- Distribution $wp(P, F \vee G) = wp(P, F) \vee wp(P, G)$

$wp(P, true)$ = weakest precondition under which P terminates

$wp(P, F) \neq wlp(P, F) \Rightarrow P$ may diverge