Solutions to selected exercises in complement of the book

Principles of Abstract Interpretation

(MIT Press, 2021)

Patrick Cousot

New York University

July 22, 2020

Solution to exercise 3.40 Consider $P = \{0\}$. We have $P \subseteq \{z \mid z \leqslant 0\} = \gamma_{\pm}(\{z \mid z \leqslant 0\})$ and $P \subseteq \{z \mid z \geqslant 0\} = \gamma_{\pm}(\{z \mid z \geqslant 0\})$ and so, by def. of a Galois connection, $\alpha_{\pm}(P) \sqsubseteq_{\pm} \{z \mid z \leqslant\}$ and $\alpha_{\pm}(P) \sqsubseteq_{\pm} \{z \mid z \geqslant 0\}$. The only element with this property in $\langle \mathbb{P}^{\pm}, \sqsubseteq_{\pm} \rangle$ is $\varnothing$ so we must have $\alpha_{\pm}(P) = \varnothing$ so $\alpha_{\pm}(\{0\}) \sqsubseteq_{\pm} \varnothing$ by reflexivity. By def. of a Galois connection, it follows that $\{0\} \subseteq \gamma_{\pm}(\varnothing) = \varnothing$, a contradiction. □

1 Solutions to selected exercises of chapter 4

Solution to exercise 4.4 An `if (B)` $\mathtt{S}_t$ `else` $\mathtt{S}_f$ can be replaced with $\big((\mathtt{B}) \uparrow (\mathtt{B}) \text{ is } \neg(\mathtt{B})\big)$

$$\texttt{while (B) \{ } \mathtt{S}_t \texttt{ break ; \} while ((B) } \uparrow \texttt{ (B)) \{ } \mathtt{S}_f \texttt{ break ; \}}.$$ □

Solution to exercise 4.7

$$\begin{array}{ll}
\mathsf{after}[\![\mathtt{P}_{15}]\!] = \mathsf{after}[\![\mathtt{Sl}_{14}]\!] = \ell_7 & \mathsf{after}[\![\mathtt{S}_9]\!] = \ell_2 \\
\mathsf{after}[\![\mathtt{Sl}_{12}]\!] = \mathsf{at}[\![\mathtt{S}_{13}]\!] = \ell_6 & \mathsf{after}[\![\mathtt{Sl}_6]\!] = \mathsf{after}[\![\mathtt{S}_9]\!] = \ell_2 \\
\mathsf{after}[\![\mathtt{S}_{13}]\!] = \mathsf{after}[\![\mathtt{Sl}_{14}]\!] = \ell_7 & \mathsf{after}[\![\mathtt{Sl}_4]\!] = \mathsf{at}[\![\mathtt{S}_5]\!] = \ell_4 \\
\mathsf{after}[\![\mathtt{Sl}_{10}]\!] = \mathsf{at}[\![\mathtt{S}_{11}]\!] = \ell_2 & \mathsf{after}[\![\mathtt{S}_5]\!] = \mathsf{after}[\![\mathtt{Sl}_6]\!] = \ell_2 \\
\mathsf{after}[\![\mathtt{S}_{11}]\!] = \mathsf{after}[\![\mathtt{Sl}_{12}]\!] = \ell_6 & \mathsf{after}[\![\mathtt{Sl}_1]\!] = \mathsf{at}[\![\mathtt{S}_2]\!] = \ell_3 \\
\mathsf{after}[\![\mathtt{Sl}_7]\!] = \mathsf{at}[\![\mathtt{S}_8]\!] = \ell_1 & \mathsf{after}[\![\mathtt{S}_2]\!] = \mathsf{after}[\![\mathtt{Sl}_4]\!] = \ell_4 \\
\mathsf{after}[\![\mathtt{S}_8]\!] = \mathsf{after}[\![\mathtt{Sl}_{10}]\!] = \ell_2 & \mathsf{after}[\![\mathtt{S}_3]\!] = \mathsf{after}[\![\mathtt{S}_5]\!] = \ell_2
\end{array}$$

So after$[\![\texttt{S}]\!]$ is the label where execution goes on when `S` terminates without a **`break ;`**. □

Solution to exercise 4.12

$$\begin{aligned}
&\text{break-to}[\![\texttt{P}_{15}]\!] = \text{break-to}[\![\texttt{Sl}_{14}]\!] = \ell_7\\
&\text{break-to}[\![\texttt{Sl}_{12}]\!] = \text{break-to}[\![\texttt{S}_{13}]\!] = \text{break-to}[\![\texttt{Sl}_{14}]\!] = \ell_7\\
&\text{break-to}[\![\texttt{Sl}_{10}]\!] = \text{break-to}[\![\texttt{S}_{11}]\!] = \text{break-to}[\![\texttt{Sl}_{12}]\!] = \ell_7\\
&\text{break-to}[\![\texttt{Sl}_{7}]\!] = \text{break-to}[\![\texttt{S}_{8}]\!] = \text{break-to}[\![\texttt{Sl}_{10}]\!] = \ell_7\\
&\text{break-to}[\![\texttt{S}_{9}]\!] = \text{after}[\![\texttt{S}_{11}]\!] = \ell_6\\
&\text{break-to}[\![\texttt{Sl}_{6}]\!] = \text{break-to}[\![\texttt{S}_{9}]\!] = \ell_6\\
&\text{break-to}[\![\texttt{Sl}_{4}]\!] = \text{break-to}[\![\texttt{S}_{5}]\!] = \text{break-to}[\![\texttt{Sl}_{6}]\!] = \ell_6\\
&\text{break-to}[\![\texttt{Sl}_{1}]\!] = \text{break-to}[\![\texttt{S}_{2}]\!] = \text{break-to}[\![\texttt{Sl}_{4}]\!] = \ell_6\\
&\text{break-to}[\![\texttt{S}_{3}]\!] = \text{break-to}[\![\texttt{S}_{5}]\!] = \ell_6
\end{aligned}$$

so a `break` before the `while` loop would terminate the program at ℓ_7 while a `break` inside the `while` loop (like ℓ_5**`break ;`**) terminates this loop at ℓ_6. □

2 Solutions to selected exercises of chapter 5

Solution to exercise 5.1 Given $\texttt{a} \in \mathcal{A}$ and the empty string ϵ, a regular expression has syntax $R ::= \texttt{a} \mid \epsilon \mid R? \mid R_1 \mid R_2 \mid R^+ \mid R^*$, and semantics $\mathcal{S}[\![\texttt{a}]\!] \triangleq \{\texttt{a}\}$, $\mathcal{S}[\![\epsilon]\!] \triangleq \{\epsilon\}$, $\mathcal{S}[\![R?]\!] \triangleq \mathcal{S}[\![R]\!] \cup \{\epsilon\}$, $\mathcal{S}[\![R_1 \mid R_2]\!] \triangleq \mathcal{S}[\![R_1]\!] \cup \mathcal{S}[\![R_2]\!]$, $\mathcal{S}[\![R^0]\!] \triangleq \{\epsilon\}$, $\mathcal{S}[\![R^1]\!] \triangleq \mathcal{S}[\![R]\!]$, $\mathcal{S}[\![R^{n+1}]\!] \triangleq \mathcal{S}[\![R]\!]\mathcal{S}[\![R^n]\!]$, $\mathcal{S}[\![R^+]\!] \triangleq \bigcup_{n>0} \mathcal{S}[\![R^n]\!]$, $\mathcal{S}[\![R^*]\!] \triangleq \bigcup_{n\geqslant 0} \mathcal{S}[\![R^n]\!]$ and language concatenation $\mathcal{L}_1\mathcal{L}_2 \triangleq \{\sigma_1\sigma_2 \mid \sigma_1 \in \mathcal{L}_1 \wedge \sigma_2 \in \mathcal{L}_2\}$ is the concatenation of strings in the languages. □

Solution to exercise 5.4

```
aterm:
  | NUM
  | IDENT
  | MINUS aterm
  | LPAREN aexpr RPAREN

aexpr:
  | aterm MINUS aexpr
  | aterm

bterm:
  | aexpr LT aexpr
```

```
  | LPAREN bexpr RPAREN

bexpr:
  | bterm NAND bexpr
  | bterm
```

□

Solution to exercise 5.5

```
stmt:
  | IDENT ASSIGN aexpr SEMICOLON
  | SEMICOLON
  | IF LPAREN bexpr RPAREN stmt
  | IF LPAREN bexpr RPAREN thenstmt ELSE stmt
  | WHILE LPAREN bexpr RPAREN stmt
  | BREAK SEMICOLON
  | LBRACKET stmtlist RBRACKET

thenstmt:
  | IDENT ASSIGN aexpr SEMICOLON
  | SEMICOLON
  | IF LPAREN bexpr RPAREN thenstmt ELSE thenstmt
  | WHILE LPAREN bexpr RPAREN thenstmt
  | BREAK SEMICOLON
  | LBRACKET stmtlist RBRACKET
```

□

Solution to exercise 5.10

```
(* File main.ml *)

open AbstractSyntax

let rec calculate_aexpr a r = match a with
| Num i -> i
| Var v -> if List.mem_assoc v r then List.assoc v r
           else failwith ("uninitialized variable:" ^ v)
| Minus (a1, a2) -> (calculate_aexpr a1 r) - (calculate_aexpr a2 r)

let rec calculate_node s r  = match s with
    | Prog sl       -> calculate_nodelist sl r
    | Assign (v, a) -> let va = calculate_aexpr a r in ((v, va) :: r, va)
    | Stmtlist sl   -> calculate_nodelist sl r
    | _             -> failwith "invalid program"
```

```
and calculate_nodelist sl r = match sl with
    | []       -> failwith "invalid program"
    | [s]      -> calculate_node s r
    | s :: sl' -> let (r', va) = calculate_nodelist sl' r in
                      calculate_node s r';; (* nodes in inverse order *)

let lexbuf = Lexing.from_channel stdin in
  try
    let (r, va) = calculate_node (Parser.prog Lexer.token lexbuf) [] in
       print_int va; print_newline ()
  with
  | Lexer.Error msg ->
      Printf.fprintf stderr "%s%!" msg
  | Parser.Error ->
      Printf.fprintf stderr
          "At offset %d: syntax error.\n%!" (Lexing.lexeme_start lexbuf)
```

□

Solution to exercise 5.11

```
(* File interpreter.ml *)

open AbstractSyntax

let bot = 0
and neg = 1
and zero = 2
and pos = 3
and negz = 4
and nzero = 5
and posz = 6
and top = 7

let print_sign s = match s with
| 0   -> print_string "_|_"
| 1   -> print_string "<0"
| 2  -> print_string "=0"
| 3   -> print_string ">0"
| 4  -> print_string "<=0"
| 5 -> print_string "=/=0"
| 6  -> print_string ">=0"
| 7   -> print_string "T"
| _ -> failwith "incorrect sign"
```

```
let minus_sign = Array.make 8 (Array.make 8 bot);;

Array.set minus_sign bot   [|bot;bot;bot;  bot;bot; bot;  bot; bot|];;
Array.set minus_sign neg   [|bot;top;neg;  neg;top; top;  neg; top|];;
Array.set minus_sign zero  [|bot;pos;zero; neg;posz;nzero;negz;top|];;
Array.set minus_sign pos   [|bot;pos;pos;  top;pos; top;  top; top|];;
Array.set minus_sign negz  [|bot;top;negz; neg;top; top;  negz;top|];;
Array.set minus_sign nzero [|bot;top;nzero;top;top; top;  top; top|];;
Array.set minus_sign posz  [|bot;pos;posz; top;posz;top;  top; top|];;
Array.set minus_sign top   [|bot;top;top;  top;top; top;  top; top|];;

let rec analyze_aexpr a r = match a with
| Num i -> if i < 0 then neg
           else if i = 0 then zero
           else pos
| Var v -> if List.mem_assoc v r then List.assoc v r else
           failwith ("uninitialized variable:" ^ v)
| Minus (a1, a2) -> let s1 = (analyze_aexpr a1 r)
                    and s2 = (analyze_aexpr a2 r) in
                       Array.get (Array.get minus_sign s1) s2


let rec analyze_node s r  = match s with
    | Prog sl       -> analyze_nodelist sl r
    | Assign (v, a) -> let va = analyze_aexpr a r in ((v, va) :: r, va)
    | Stmtlist sl   -> analyze_nodelist sl r
    | _             -> failwith "invalid program"
and analyze_nodelist sl r = match sl with
    | []       -> failwith "invalid program"
    | [s]      -> analyze_node s r
    | s :: sl' -> let (r', va) = analyze_nodelist sl' r in
                     analyze_node s r';; (* nodes in inverse order *)

let lexbuf = Lexing.from_channel stdin in
  try
    let (r, va) = analyze_node (Parser.prog Lexer.token lexbuf) [] in
       print_sign va; print_newline ()
  with
  | Lexer.Error msg ->
      Printf.fprintf stderr "%s%!" msg
  | Parser.Error ->
      Printf.fprintf stderr "At offset %d: syntax error.\n%!"
                                          (Lexing.lexeme_start lexbuf)
```

□

3 Solutions to selected exercises of chapter 9

Solution to exercise 9.13 Assume we have such an algorithm `correct(P, f)` which always terminates and returns true if and only if `P`$(n) = f(n)$ for all integers n for which $f(n)$ is well-defined ($n \in \mathsf{dom}(f)$). We can even fix f *e.g.* $f(n) = n^3$.

Then the following algorithm would always terminate and return true if and only if `P` terminates on input `i`

```
let terminate(p, i) =
  let t(n) = p(i); return f(n) in
    correct(t, f);
```

`correct(t, f)` is true if and only if t$(n) = f(n)$ for all integers n for which $f(n)$ is well-defined, if and only if `P` terminates on input `i`, which is undecidable. □

4 Solutions to selected exercises of chapter 11

Solution to exercise 11.11

$$
\begin{aligned}
& R^*(P) \supseteq Q \\
\Leftrightarrow\ & \forall y \in Q\ .\ \forall x \in P\ .\ \langle x,\ y\rangle \in R && \text{〈def. } \supseteq \text{ and } R^* \text{〉} \\
\Leftrightarrow\ & \forall x \in P\ .\ \forall y \in Q\ .\ \langle x,\ y\rangle \in R && \text{〈def. } \forall \text{〉} \\
\Leftrightarrow\ & P \subseteq R^\dagger(Q) && \text{〈def. } \subseteq \text{ and } R^\dagger \text{〉}
\end{aligned}
$$
□

Solution to exercise 11.13 For all $w \in \Sigma^*$, $L_1, L_2 \in \wp(\Sigma^*)$, we have $L_1 \subseteq w^{-1}L_2$ if and only if $(x \in L_1 \Rightarrow wx \in L_2)$ if and only if $wL_1 \subseteq L_2$ so $wL_1 \subseteq L_2 \Leftrightarrow L_1 \subseteq w^{-1}L_2$. Moreover $w^{-1}(wL) = L$ for all $L \in \wp(\Sigma^*)$. Therefore $\langle \wp(\Sigma^*),\ \subseteq\rangle \underset{\alpha_{\overleftarrow{w}}}{\overset{\gamma_{\overleftarrow{w}}}{\leftrightarrows}} \langle \wp(\Sigma^*),\ \subseteq\rangle$ where $\alpha_{\overleftarrow{w}}(L) = wL$ and $\gamma_{\overleftarrow{w}}(L) = w^{-1}L$. Similarly, $L_1 w \subseteq L_2 \Leftrightarrow L_1 \subseteq L_2 w^{-1}$ so $\langle \wp(\Sigma^*),\ \subseteq\rangle \underset{\alpha_{\overrightarrow{w}}}{\overset{\gamma_{\overrightarrow{w}}}{\leftrightarrows}} \langle \wp(\Sigma^*),\ \subseteq\rangle$ where $\alpha_{\overrightarrow{w}}(L) = Lw$ and $\gamma_{\overrightarrow{w}}(L) = Lw^{-1}$. M □

Solution to exercise 11.14 We have $\langle \wp(\mathbb{R} \to \mathbb{R}),\ \subseteq\rangle \underset{\alpha_O}{\overset{\gamma_O}{\leftrightarrows}} \langle \mathbb{R} \to \mathbb{R}_*,\ \dot{\leqslant}\rangle$ where

$$
\begin{aligned}
\alpha_O(P) &\triangleq \dot{min}\{|g| \in \mathbb{R} \to \mathbb{R} \mid \forall f \in P\ .\ \exists x_0, c \in \mathbb{R}\ .\ \forall x \geqslant x_0\ .\ |f(x)| \leqslant c|g(x)|\} \\
\gamma_O(g) &\triangleq \{f \in \mathbb{R} \to \mathbb{R} \mid \exists x_0, c \in \mathbb{R}\ .\ \forall x \geqslant x_0\ .\ |f(x)| \leqslant c|g(x)|\}
\end{aligned}
$$

and $|g|$, $\dot{\leqslant}$, and $\dot{min}$ are pointwise.

$\alpha_O(P) \mathrel{\dot\subseteq} |h|$

$\Leftrightarrow$ $\dot{min}\{g \in \mathbb{R} \to \mathbb{R} \mid \forall f \in P \,.\, \exists x_0, c \in \mathbb{R} \,.\, \forall x \geqslant x_0 \,.\, |f(x)| \leqslant c|g(x)|\} \mathrel{\dot\leqslant} |h|$ ⟅def. α⟆

$\Leftrightarrow$ $\exists g \in \mathbb{R} \to \mathbb{R} \,.\, \forall f \in P \,.\, \exists x_0, c \in \mathbb{R} \,.\, \forall x \geqslant x_0 \,.\, |f(x)| \leqslant c|g(x)| \wedge |g(x)| \leqslant |h(x)|$
⟅def. $\dot{min}$ and $\mathrel{\dot\leqslant}$⟆

$\Rightarrow$ $\exists g \in \mathbb{R} \to \mathbb{R} \,.\, \forall f \in P \,.\, \exists x_0, c \in \mathbb{R} \,.\, \forall x \geqslant x_0 \,.\, |f(x)| \leqslant c|h(x)|$ ⟅$\leqslant$ transitive⟆

$\Leftrightarrow$ $\forall f \in P \,.\, \exists x_0, c \in \mathbb{R} \,.\, \forall x \geqslant x_0 \,.\, |f(x)| \leqslant c|h(x)|$ ⟅def. $\exists$⟆

$\Leftrightarrow$ $P \subseteq \{f \mid \exists x_0, c \in \mathbb{R} \,.\, \forall x \geqslant x_0 \,.\, |f(x)| \leqslant c|h(x)|\}$ ⟅def. $\subseteq$⟆

$\Leftrightarrow$ $P \subseteq \gamma_O(h)$ ⟅def. γ_O.⟆

Conversely,

$\forall f \in P \,.\, \exists x_0, c \in \mathbb{R} \,.\, \forall x \geqslant x_0 \,.\, |f(x)| \leqslant c|h(x)|$

$\Rightarrow$ $\exists g \in \mathbb{R} \to \mathbb{R} \,.\, \forall f \in P \,.\, \exists x_0, c \in \mathbb{R} \,.\, \forall x \geqslant x_0 \,.\, |f(x)| \leqslant c|g(x)| \wedge |g(x)| \leqslant |h(x)|$
⟅taking $g = h$⟆ □

Solution to exercise 11.15 A property of a distribution is an element of $\wp(\mathbb{V} \to [0,1])$. Define $\alpha_{\mathsf{E}} \in \wp(\mathbb{V} \to [0,1]) \to \wp(\mathbb{V})$ by $\alpha_{\mathsf{E}}(\mathcal{P}) \triangleq \{\mathsf{E}(X) \mid P_X \in \mathcal{P}\}$. This is an homomorphic/partitioning abstraction of Exercise 11.6 so a Galois connection. In statistics one is often interested in properties of a given distribution P_X. Then $\alpha_{\mathsf{E}}(\{P_X\}) = \{\mathsf{E}(X)\}$ which is identifies with $\mathsf{E}(X)$. The concretization is a set of distributions so the best guesses prediction based on the expectation is valid for any of them, which can be imprecise for skewed distributions with mean far from the median. □

Solution to exercise 11.19

$\alpha \fatsemi \sqsubseteq = \leqslant \fatsemi \gamma^{-1}$

$\Leftrightarrow$ $\forall P, Q : (\langle P,\, Q\rangle \in \alpha \fatsemi \sqsubseteq) \Leftrightarrow (\langle P,\, Q\rangle \in \leqslant \fatsemi \gamma^{-1})$ ⟅def. equality of relations⟆

$\Leftrightarrow$ $\forall P, Q : (\exists R : \langle P,\, R\rangle \in \alpha \wedge \langle R,\, Q\rangle \in \sqsubseteq) \Leftrightarrow (\exists R' : \langle P,\, R'\rangle \in \leqslant \wedge \langle R',\, Q\rangle \in \gamma^{-1})$
⟅def. composition of relations $r_1 \fatsemi r_2 \triangleq \{\langle x,\, z\rangle \mid \exists y : \langle x,\, y\rangle \in r_1 \wedge \langle y,\, z\rangle \in r_2\}$⟆

$\Leftrightarrow$ $\forall P, Q : (\exists R : \langle P,\, R\rangle \in \alpha \wedge \langle R,\, Q\rangle \in \sqsubseteq) \Leftrightarrow (\exists R' : \langle P,\, R'\rangle \in \leqslant \wedge \langle Q,\, R'\rangle \in \gamma)$
⟅def. inverse of relations⟆

$\Leftrightarrow$ $\forall P, Q : (\exists R : \langle P,\, R\rangle \in \alpha \wedge R \sqsubseteq Q) \Leftrightarrow (\exists R' : P \leqslant R' \wedge \langle Q,\, R'\rangle \in \gamma)$
⟅def. order relations⟆

$\Leftrightarrow$ $\forall P, Q : (\exists R : R = \alpha(P) \wedge R \sqsubseteq Q) \Leftrightarrow (\exists R' : P \leqslant R' \wedge R' = \gamma(Q))$ ⟅α and γ are functions⟆

$\Leftrightarrow$ $\forall P, Q : (\alpha(P) \sqsubseteq Q) \Leftrightarrow (P \leqslant \gamma(Q))$ ⟅simplification⟆

$\Leftrightarrow$ $\langle C, \leqslant\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \sqsubseteq\rangle$ ⟅by (11.1)⟆ □

Solution to exercise 11.21 For all $f \in \mathcal{D} \xrightarrow{\nearrow} \mathcal{D}$ and $y \in \mathcal{D}$,

$$
\begin{array}{lll}
& \alpha_p(f) \sqsubseteq y & \\
\Leftrightarrow & f(p) \sqsubseteq y & \text{(def. } \alpha_p\text{)} \\
\Leftrightarrow & \forall x \sqsubseteq p \,.\, f(x) \sqsubseteq y & \text{(} f \text{ increasing and } \sqsubseteq \text{ reflexive and transitive)} \\
\Leftrightarrow & \forall x \,.\, f(x) \sqsubseteq (\!| x \sqsubseteq p \mathbin{?} y \mathbin{:} \top |\!) & \text{(def. conditional and supremum } \top\text{)} \\
\Leftrightarrow & \forall x \,.\, f(x) \sqsubseteq \gamma_p(y)(x) & \text{(by defining } \gamma_p(y)(x) \triangleq (\!| x \sqsubseteq p \mathbin{?} y \mathbin{:} \top |\!)\text{)} \\
\Leftrightarrow & f \mathrel{\dot{\sqsubseteq}} \gamma_p(y) & \text{(pointwise)} \quad \square
\end{array}
$$

Solution to exercise 11.22

$$
\begin{array}{lll}
& \alpha_h(X) \mathrel{\dot{\subseteq}} Y & \\
\Leftrightarrow & \forall a \in A \,.\, \alpha_h(X)\, a \subseteq Y(a) & \text{(pointwise def. } \dot{\subseteq}\text{)} \\
\Leftrightarrow & \forall a \in A \,.\, \{f(a)x \mid x \in X\} \subseteq Y(a) & \text{(def. } \alpha_h\text{)} \\
\Leftrightarrow & \forall a \in A \,.\, \forall x \in X \,.\, f(a)x \in Y(a) & \text{(def. } \subseteq\text{)} \\
\Leftrightarrow & \forall x \in X \,.\, \forall a \in A \,.\, f(a)x \in Y(a) & \text{(def. } \forall\text{)} \\
\Leftrightarrow & X \subseteq \{x \mid \forall a \in A \,.\, f(a)x \in Y(a)\} & \text{(def. } \subseteq\text{)} \\
\Leftrightarrow & X \subseteq \gamma_h(Y) & \text{(by defining } \gamma_h(Y) \triangleq \{x \mid \forall a \in A \,.\, f(a)x \in Y(a)\}\text{)} \quad \square
\end{array}
$$

Solution to exercise 11.30 If $x \in X$ then $x \sqsubseteq_1 \sqcup_1 X$ by def. lub so $f(x) \sqsubseteq_2 f(\sqcup_1 X)$ since f is increasing, proving that $f(\sqcup_1 X)$ is an upper bound of $\{f(x) \mid x \in X\}$ hence $\sqcup_2\{f(x) \mid x \in X\} \sqsubseteq_2 f(\sqcup_1 X)$ by def. existing lub. $\square$

Solution to exercise 11.36 By $\alpha \in \mathbb{N} \to \{\bullet\}$, we have $\forall n \in \mathbb{N} \,.\, \alpha(n) = \bullet$. By $\gamma \in \{\bullet\} \to \mathbb{N}$, we have $\gamma(\bullet) = n$ for some $n \in \mathbb{N}$. Then $\gamma(\alpha(n+1)) = n \not\leqslant n+1$, in contradiction with $\gamma \circ \alpha$ is extensive in Exercise 11.34. A fix is to consider $\mathbb{N} \cup \{\infty\}$ with $\gamma(\bullet) = \infty$. $\square$

Solution to exercise 11.39 γ does not preserves meets. $\square$

Solution to exercise 11.40 γ preserves finite meets but not infinite ones. $\square$

Solution to exercise 11.43 Define $\alpha_y(z) = x \times y$ and $\gamma_y(x) = x \div y$. Then $\forall x, y, z \in \mathbb{N} \,.\, z \times y \leqslant x \Leftrightarrow z \leqslant x \div y$ implies $\alpha_y(z) \leqslant x \Leftrightarrow z \leqslant \gamma_y(x)$ *i.e.* $\langle \mathbb{N}, \leqslant \rangle \underset{\alpha_y}{\overset{\gamma_y}{\leftrightarrows}} \langle \mathbb{N}, \leqslant \rangle$ which, by Lemma 11.41, implies $x \div y = \max\{z \mid x \times y \leqslant x\}$. $\square$

Solution to exercise 11.47

$\gamma(a)$

$= \max\{c \in C \mid c \sqsubseteq \gamma(a)\}$ ⟅The max exists and is $\gamma(a)$ by reflexivity⟆

$= \max\{c \in C \mid \alpha(c) \preccurlyeq a\}$ ⟅$\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$⟆

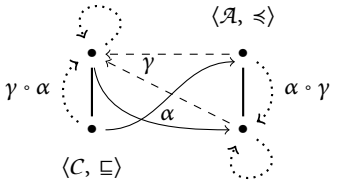
$= \max\{c \in C \mid \alpha(c) \in \downarrow a\}$ ⟅def. $\downarrow a \triangleq \{x \in \mathcal{A} \mid x \preccurlyeq a\}$⟆

$= \max \alpha^{-1}(\downarrow a)$ ⟅def. $\alpha^{-1}(\downarrow a) \triangleq \{c \in C \mid \alpha(c) \in \downarrow a\}$⟆

$\max \alpha^{-1}(\downarrow a)$ is the lub of $\alpha^{-1}(\downarrow a)$. The dual is $\alpha(c) = \min \gamma^{-1}(\uparrow a)$. □

Solution to exercise 11.52 $\gamma(P)x \triangleq \bigsqcap\{y \in P \mid x \sqsubseteq y\}$. □

Solution to exercise 11.57 Not necessarily, here is a counter-example (α is not increasing).



□

Solution to exercise 11.63 Let us prove $\langle \wp(\mathcal{P}), \subseteq\rangle \underset{\downarrow}{\overset{\uparrow}{\leftrightarrows}} \langle \wp(\mathcal{P}), \supseteq\rangle$.

$\downarrow(X) \supseteq Y$

$\Leftrightarrow \forall y \in Y \,.\, y \in \downarrow(X)$ ⟅def. $\supseteq$⟆

$\Leftrightarrow \forall y \in Y \,.\, \forall x \in X \,.\, y \sqsubseteq x$ ⟅def. $\downarrow$ and $\in$⟆

$\Leftrightarrow \forall x \in X \,.\, \forall y \in Y \,.\, y \sqsubseteq x$ ⟅def. $\forall$⟆

$\Leftrightarrow X \subseteq \{x \in \mathcal{P} \mid \forall y \in Y \,.\, y \sqsubseteq x\}$ ⟅def. $\forall$⟆

$\Leftrightarrow X \subseteq \uparrow(Y)$ ⟅def. $\uparrow$⟆

By Exercise 11.55, $\uparrow \circ \downarrow$ is a closure operator. Therefore $\langle \uparrow \circ \downarrow(\wp(\mathcal{P})), \supseteq\rangle$ is a complete lattice by Exercise 11.62. We have $\dot{\uparrow}(x) \triangleq \uparrow(\{x\}) = \{z \in \mathcal{P} \mid \forall y \in \{x\} \,.\, y \sqsubseteq z\} = \{z \in \mathcal{P} \mid x \sqsubseteq z\}$. By the Galois connection and Exercise 11.44, $\dot{\uparrow}(x) = \uparrow(\{x\}) = \uparrow \circ \downarrow \circ \uparrow(\{x\}) = \uparrow \circ \downarrow(\dot{\uparrow}(x))$, proving that $\dot{\uparrow} \in \mathcal{P} \to \uparrow \circ \downarrow(\wp(\mathcal{P}))$. Moreover, if $x \sqsubseteq y$ then $\{z \mid x \sqsubseteq z\} \supseteq \{z \mid y \sqsubseteq z\}$ by transitivity, and so, $\dot{\uparrow}(x) \supseteq \dot{\uparrow}(y)$. Conversely $\dot{\uparrow}(x) \supseteq \dot{\uparrow}(y)$ implies $\{z \mid x \sqsubseteq z\} \supseteq \{z \mid y \sqsubseteq z\}$ and so, by reflexivity and def. $\supseteq$, $y \in \{z \mid x \sqsubseteq z\}$, proving that $x \sqsubseteq y$. It follows that $\dot{\uparrow}$ is an order-embedding of $\langle \mathcal{P}, \sqsubseteq\rangle$ into $\langle \uparrow \circ \downarrow(\wp(\mathcal{P})), \supseteq\rangle$ such that $\forall x, y \in \mathcal{P} \,.\, x \sqsubseteq y \Leftrightarrow \dot{\uparrow}(x) \supseteq \dot{\uparrow}(y)$. So $(x = y) \Leftrightarrow (x \sqsubseteq y \wedge y \sqsubseteq x) \Leftrightarrow (\dot{\uparrow}(x) \supseteq \dot{\uparrow}(y) \wedge \dot{\uparrow}(y) \supseteq \dot{\uparrow}(x)) \Leftrightarrow (\dot{\uparrow}(x) = \dot{\uparrow}(y))$. By contraposition in Section 2.4.1, $(x \neq y) \Leftrightarrow (\dot{\uparrow}(x) \neq \dot{\uparrow}(y))$ proving that $\dot{\uparrow}$ is bijective so distinct elements of $\mathcal{P}$ are be mapped to distinct elements of $\uparrow \circ \downarrow(\wp(\mathcal{P}))$. If $x, y \in \mathcal{P}$ are not comparable then $\dot{\uparrow}(x)$ and $\dot{\uparrow}(y)$ are not comparable since otherwise $\dot{\uparrow}(x) \supseteq \dot{\uparrow}(y)$ would imply $x \sqsubseteq y$, a contradiction, and inversely.

Otherwise $x \sqsubseteq y$ are comparable and then $x \sqsubseteq y \Leftrightarrow \dot{\wr}(x) \supseteq \dot{\wr}(y)$ implies that they have the same ordering in $\langle \uparrow \circ \downarrow(\wp(\mathcal{P})), \supseteq \rangle$. Let $\dot{\wr}^{-1}$ be the inverse of the bijection $\dot{\wr} \in \mathcal{P} \rightarrowtail\!\!\!\!\twoheadrightarrow \uparrow \circ \downarrow(\wp(\mathcal{P}))$. We have $X \supseteq Y$ implies $\dot{\wr} \circ \dot{\wr}^{-1}(X) \supseteq \dot{\wr} \circ \dot{\wr}^{-1}(Y)$ implies $\dot{\wr}^{-1}(X) \sqsubseteq \dot{\wr}^{-1}(Y)$ by the embedding, proving that $\dot{\wr}^{-1}$ is decreasing. If $x \in \mathcal{P}$ and $Y \in \uparrow \circ \downarrow(\wp(\mathcal{P}))$ then $\dot{\wr} x \supseteq Y \Leftrightarrow \dot{\wr}^{-1} \circ \dot{\wr} x \sqsubseteq \dot{\wr}^{-1}(Y) \Leftrightarrow x \sqsubseteq \dot{\wr}^{-1}(Y)$, proving $\langle \mathcal{P}, \sqsubseteq \rangle \underset{\dot{\wr}}{\overset{\dot{\wr}^{-1}}{\leftrightarrows}} \langle \uparrow \circ \downarrow(\wp(\mathcal{P})), \supseteq \rangle$.

The proof by MacNeille [**MacNeille-completion-37**] uses the order embedding of x into cuts $\langle \{y \mid y \sqsubseteq x\}, \{z \mid x \sqsubseteq z\} \rangle$ generalizing the cuts used by Dedekind [**Dedekind1892**] to construct the real numbers from the rational numbers, hence the name "Dedekind–MacNeille completion". □

Solution to exercise 11.66 An hint is to use Lemma 11.37 for α_a. □

5 Solutions to selected exercises of chapter 12

Solution to exercise 12.10 Take $\mathcal{P} = \{f, g\}$ where $f(x) = 0$ and $g(x) = 1$. Then $T = \alpha_{\mathcal{F}}(\mathcal{P})$ is $T(P) = [\![P = \varnothing \mathbin{?} \varnothing \mathbin{:} \{0, 1\}]\!]$ so $\gamma_{\mathcal{F}}(T) = \{f \mid \forall x \,.\, f(x) \in \{0, 1\}\}$ which is different from $\{f, g\}$. □

Solution to exercise 12.16

$$
\begin{aligned}
& \mathsf{pre}[R]Q \cap \widetilde{\mathsf{pre}}[R]Q \\
= \;& \mathsf{pre}[R]\mathbb{Q} \cap \widetilde{\mathsf{pre}}[R]Q && \text{(Exercise 12.15)} \\
= \;& \{x \in \mathbb{P} \mid \exists y \in \mathbb{Q} \,.\, \langle x,\ y \rangle \in R\} \cap \widetilde{\mathsf{pre}}[R]Q && \text{(def. (12.11) of pre)} \\
= \;& \mathbb{P} \cap \widetilde{\mathsf{pre}}[R]Q && \text{($R \in \wp(\mathbb{P} \times \mathbb{Q})$ and R is total)} \\
= \;& \widetilde{\mathsf{pre}}[R]Q && \text{(def. (12.12) of } \widetilde{\mathsf{pre}} \text{ so that } \widetilde{\mathsf{pre}}[R]Q \subseteq \mathbb{P}) \quad \square
\end{aligned}
$$

Solution to exercise 12.17

$$
\begin{aligned}
& \widetilde{\mathsf{pre}}[R]Q \\
= \;& \mathsf{pre}[R]Q \cap \widetilde{\mathsf{pre}}[R]Q && \text{(Exercise 12.16)} \\
= \;& \{x \in \mathbb{P} \mid \exists y \in Q \,.\, \langle x,\ y \rangle \in R \wedge \forall y' \in \mathbb{Q} \,.\, \langle x,\ y' \rangle \in R \Rightarrow y' \in Q\} \\
& && \text{(def. (12.11) of pre and } \widetilde{\mathsf{pre}}\text{, def. } \cap) \\
= \;& \{x \in \mathbb{P} \mid \exists y \in Q \,.\, \langle x,\ y \rangle \in R\} \\
& && \text{($(\subseteq)$ } A \wedge B \Rightarrow A\text{, } (\supseteq) \text{ if } \langle x,\ y' \rangle \in R \text{ then } y' = y \text{ since } R \text{ is deterministic so } y \in Q\text{,} \\
& && \quad \text{and antisymmetry)} \\
= \;& \widetilde{\mathsf{pre}}[R]Q && \text{(def. (12.11) of pre)}
\end{aligned}
$$

— $\mathsf{pre}[R] = \mathsf{post}[R^{-1}] = \widetilde{\mathsf{post}}[R^{-1}] = \widetilde{\mathsf{pre}}[R]$ by (12.11), (12.2), (12.3), and (12.12). □

Solution to exercise 12.24

— $\mathsf{post}[R]P \subseteq Q$

$\Leftrightarrow$ $\{y \in \mathbb{Q} \mid \exists x \in P \,.\, \langle x,\ y\rangle \in R\} \subseteq Q$ ⟅def. (12.2) of post⟆

$\Leftrightarrow$ $\forall y \in \mathbb{Q} \,.\, (\exists x \in P \,.\, \langle x,\ y\rangle \in R) \Rightarrow y \in Q$ ⟅def. $\subseteq$⟆

$\Leftrightarrow$ $\forall y \in \mathbb{Q} \,.\, \forall x \in P \,.\, \langle x,\ y\rangle \in R \Rightarrow y \in Q$ ⟅def. $\subseteq$⟆

$\Leftrightarrow$ $\forall x \in P \,.\, \forall y \in \mathbb{Q} \,.\, \langle x,\ y\rangle \in R \Rightarrow y \in Q$ ⟅def. $\forall$⟆

$\Leftrightarrow$ $P \subseteq \{x \in \mathbb{P} \mid \forall y \in \mathbb{Q} \,.\, \langle x,\ y\rangle \in R \Rightarrow y \in Q\}$ ⟅def. $\subseteq$⟆

$\Leftrightarrow$ $P \subseteq \widetilde{\mathsf{pre}}[R]Q$ ⟅def. (12.12) of $\widetilde{\mathsf{pre}}$⟆

— By Lemma 11.37, it follows that $\mathsf{post}[R] \in \wp(\mathbb{P}) \longrightarrow\!\!\!\!\!\sqcup\ \wp(\mathbb{Q})$.

— $\mathsf{post}[R] \mathrel{\dot\subseteq} T$

$\Leftrightarrow$ $\{y \in \mathbb{Q} \mid \exists x \in P \,.\, \langle x,\ y\rangle \in R\} \subseteq T(P)$

⟅pointwise def. $\dot\subseteq$ and (12.2) of post⟆

$\Leftrightarrow$ $\forall y \in \mathbb{Q} \,.\, (\exists x \in P \,.\, \langle x,\ y\rangle \in R) \Rightarrow y \in T(P)$ ⟅def. $\subseteq$⟆

$\Leftrightarrow$ $\forall x \in \mathbb{P} \,.\, \forall y \in \mathbb{Q} \,.\, (x \in P \Rightarrow (\langle x,\ y\rangle \in R) \Rightarrow y \in T(P))$ ⟅def. $\Rightarrow$ and $\forall$⟆

$\Leftrightarrow$ $\forall \langle x,\ y\rangle \in \mathbb{P} \times \mathbb{Q} \,.\, (\langle x,\ y\rangle \in R) \Rightarrow y \in T(\{x\})$

⟅($\Rightarrow$) for $P = \{x\}$ so $x \in P$ is true;
($\Leftarrow$) if $x \in P$ and $\langle x,\ y\rangle \in R$ then $y \in T(\{x\}) \subseteq T(P)$ since T preserves joins so is increasing hence $y \in T(P)$. ⟆

$\Leftrightarrow$ $R \subseteq \{\langle x,\ y\rangle \in \mathbb{P} \times \mathbb{Q} \mid y \in T(\{x\})\}$ ⟅def. $\subseteq$⟆

$\Leftrightarrow$ $R \subseteq \mathsf{post}^{-1}[T]$ ⟅def. (12.6) of post^{-1}⟆

— We have $\langle \wp(\mathbb{P}),\ \subseteq\rangle \underset{\mathsf{post}[R]}{\overset{\widetilde{\mathsf{pre}}[R]}{\leftrightarrows}} \langle \wp(\mathbb{Q}),\ \subseteq\rangle$, in particular $\langle \wp(\mathbb{P}),\ \subseteq\rangle \underset{\mathsf{post}[R^{-1}]}{\overset{\widetilde{\mathsf{pre}}[R^{-1}]}{\leftrightarrows}} \langle \wp(\mathbb{Q}),\ \subseteq\rangle$ when R is R^{-1} so $\langle \wp(\mathbb{P}),\ \subseteq\rangle \underset{\mathsf{pre}[R]}{\overset{\widetilde{\mathsf{post}}[R]}{\leftrightarrows}} \langle \wp(\mathbb{Q}),\ \subseteq\rangle$ by (12.11) and (12.12). Another proof would use the conjugate as in Section **11.9.2**.

— $\mathsf{pre}[R] \mathrel{\dot\subseteq} T$

$\Leftrightarrow$ $\mathsf{post}[R^{-1}] \mathrel{\dot\subseteq} T$ ⟅def. (12.11) of pre⟆

$\Leftrightarrow$ $R^{-1} \subseteq \mathsf{post}^{-1}[T]$ ⟅by $\langle \wp(\mathbb{Q} \times \mathbb{P}),\ \subseteq\rangle \underset{\mathsf{post}}{\overset{\mathsf{post}^{-1}}{\leftrightarrows}} \langle \wp(\mathbb{Q}) \longrightarrow\!\!\!\!\!\sqcup\ \wp(\mathbb{P}),\ \mathrel{\dot\subseteq}\rangle$⟆

$\Leftrightarrow$ $R \subseteq (\mathsf{post}^{-1}[T])^{-1}$ ⟅$^{-1}$ is increasing and its own inverse⟆

$R \subseteq \mathsf{pre}^{-1}[T]$ ⟅def. (12.21) of pre^{-1}⟆ □

Solution to exercise 12.25 Let $P \in \wp(\mathbb{T}^+)$ and $Q \in \wp(\mathbb{T}^{+\infty})$. We have

$$
\begin{aligned}
& \overrightarrow{\mathsf{post}}[\mathcal{S}]P \subseteq Q \\
\Leftrightarrow\ & \{\pi_2 \in \mathbb{T}^{+\infty} \mid \exists \pi_0, \pi_1 \,.\, \pi_0 \in P \wedge \pi_1 \frown \pi_2 \in \mathcal{S}(\pi_0)\} \subseteq Q && \text{⟅def. } \overrightarrow{\mathsf{post}}[\mathcal{S}]\text{⟆} \\
\Leftrightarrow\ & \forall \pi_0, \pi_1, \pi_2 \,.\, (\pi_0 \in P \wedge \pi_1 \frown \pi_2 \in \mathcal{S}(\pi_0)) \Rightarrow (\pi_2 \in Q) && \text{⟅def. } \subseteq\text{⟆} \\
\Leftrightarrow\ & P \subseteq \{\pi_0 \mid \forall \pi_1, \pi_2 \,.\, (\pi_1 \frown \pi_2 \in \mathcal{S}(\pi_0)) \Rightarrow (\pi_2 \in Q)\} && \text{⟅def. } \subseteq\text{⟆} \\
\Leftrightarrow\ & P \subseteq \neg\{\pi_0 \mid \exists \pi_1, \pi_2 \,.\, (\pi_1 \frown \pi_2 \in \mathcal{S}(\pi_0)) \wedge (\pi_2 \in \neg Q)\} && \text{⟅def. complement⟆} \\
\Leftrightarrow\ & P \subseteq \neg\overrightarrow{\mathsf{pre}}[\mathcal{S}](\neg Q) && \text{⟅def. } \overrightarrow{\mathsf{pre}}[\mathcal{S}]\text{⟆} \\
\Leftrightarrow\ & P \subseteq \widetilde{\overrightarrow{\mathsf{pre}}}[\mathcal{S}]Q && \text{⟅def. } \widetilde{\overrightarrow{\mathsf{pre}}}[\mathcal{S}]\text{⟆}
\end{aligned}
$$

The second Galois connection is the conjugate of the first one, see Section **11.9.2**. □

Solution to exercise 12.27 An execution starting with an initial environment in P, will have the following behaviors (a) $\mathsf{post}[\mathsf{S}]P \subseteq Q$, (b) $\mathsf{post}[\mathsf{S}]P \subseteq \neg Q$, (c) $\mathsf{post}[\mathsf{S}]P \subseteq \{\bot\}$, (ab) $\mathsf{post}[\mathsf{S}]P \subseteq \mathbb{Q} \setminus \{\bot\} \wedge \mathsf{post}[\mathsf{S}]P \not\subseteq Q \wedge \mathsf{post}[\mathsf{S}]P \not\subseteq \neg Q$, (ac) $\mathsf{post}[\mathsf{S}]P \subseteq Q \cup \{\bot\}$, (bc) $\mathsf{post}[\mathsf{S}]P \subseteq \neg Q \cup \{\bot\}$, (abc) $\mathsf{post}[\mathsf{S}]P \not\subseteq Q \wedge \mathsf{post}[\mathsf{S}]P \not\subseteq \neg Q \wedge \mathsf{post}[\mathsf{S}]P \not\subseteq \{\bot\}$. □

6 Solutions to selected exercises of chapter 13

Solution to exercise 13.2 The smallest topology on $\mathcal{X}$ is $\{\varnothing, \mathcal{X}\}$ and the largest is $\wp(\mathcal{X})$. □

Solution to exercise 13.3 $\wp(\mathcal{X})$ is the only topology that makes every subset of $\mathcal{X}$ both an open and closed set. □

7 Solutions to selected exercises of chapter 15

Solution to exercise 15.1 $f(x) = x + 1$ has no fixpoint on $\mathbb{Z}$, $f(x) = -x$ has one fixpoint (0), and $f(x) = x$ has infinitely many fixpoints. □

Solution to exercise 15.5 The complete lattice is $\langle [0, 1], \leqslant \rangle$. The least fixpoint is 0. □

Solution to exercise 15.33 We have $\mathsf{lfp}^{\subseteq} f \subseteq \mathsf{gfp}^{\subseteq} f$ so $\neg\mathsf{gfp}^{\subseteq} f \subseteq \neg\mathsf{lfp}^{\subseteq} f$ and, by Theorem 15.32, $\neg\mathsf{gfp}^{\subseteq} f = \mathsf{lfp}^{\subseteq} \tilde{f}$ so $\mathsf{lfp}^{\subseteq} \tilde{f} \subseteq \neg\mathsf{lfp}^{\subseteq} f$. It follows that $\mathsf{lfp}^{\subseteq} f \cap \mathsf{lfp}^{\subseteq} \tilde{f} \subseteq \mathsf{lfp}^{\subseteq} f \cap \neg\mathsf{lfp}^{\subseteq} f = \varnothing$. □

Solution to exercise 15.34 If $\mathsf{lfp}^{\subseteq} f$ is the unique fixpoint of f then, by Theorem 15.32, $\mathsf{lfp}^{\subseteq} f = \mathsf{gfp}^{\subseteq} f = \neg\mathsf{lfp}^{\subseteq} \tilde{f}$ so $\mathsf{lfp}^{\subseteq} f \cup \mathsf{lfp}^{\subseteq} \tilde{f} = \mathsf{lfp}^{\subseteq} f \cup \neg\mathsf{lfp}^{\subseteq} f = S$.

Conversely, assume that $\mathsf{lfp}^{\subseteq} f \cup \mathsf{lfp}^{\subseteq} \tilde{f} = S$ and f has at least two distinct fixpoints so that by Tarski fixpoint Theorem 15.6, $\mathsf{lfp}^{\subseteq} f \subsetneq \mathsf{gfp}^{\subseteq} f$. By Theorem 15.32, $\mathsf{lfp}^{\subseteq} f \subsetneq \neg\mathsf{lfp}^{\subseteq} \tilde{f}$, in contradiction with $\mathsf{lfp}^{\subseteq} f \cup \mathsf{lfp}^{\subseteq} \tilde{f} = S$. □

8 Solutions to selected exercises of chapter 16

Solution to exercise 16.14 The language L defined by the context-free grammar $X ::= X\ X \mid a$ can be specified by the deductive system with axiom $a \in L$ and inference rule $\frac{\sigma_1 \in L, \sigma_2 \in L}{\sigma_1\sigma_2 \in L}$. The corresponding fixpoint definition is $L = \mathsf{lfp}^{\subseteq} F$ where $F(X) = \{a\} \cup \{\sigma_1\sigma_2 \mid \sigma_1, \sigma_2 \in L\} = \{a^n \mid n \geqslant 1\}$. □

9 Solutions to selected exercises of chapter 18

Solution to exercise 18.5

$$
\begin{aligned}
& \overline{f}'(\overline{x}) \\
= & \bigvee_{i=1}^{n} \alpha(f(x^i)) && \text{⦃decomposition hypothesis } \gamma(\overline{x}) = \textstyle\bigsqcup_{i=1}^{n} x^i\text{⦄} \\
= & \alpha(\bigsqcup_{i=1}^{n} f(x^i)) && \text{⦃the lower adjoint of a Galois connection preserves existing joins⦄} \\
= & \alpha(f(\bigsqcup_{i=1}^{n} x^i)) && \text{⦃} f \in C\, joinmorphismto C \text{ preserves existing arbitrary joins⦄} \\
= & \alpha \circ f \circ \gamma(\overline{x}) && \text{⦃decomposition hypothesis } \gamma(\overline{x}) = \textstyle\bigsqcup_{i=1}^{n} x^i \text{ and def. function composition} \\
& \circ\text{⦄} \\
\preceq & \overline{f}(\overline{x}) && \text{⦃hypothesis } \alpha \circ f \circ \gamma \mathrel{\dot{\preceq}} \overline{f}.\text{⦄} \quad \square
\end{aligned}
$$

Solution to exercise 18.13 The proof that $\alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} \Leftrightarrow f \circ \gamma \sqsubseteq \gamma \circ \overline{f}$ does not use the fact that f and $\overline{f}$ are increasing.

$$
\begin{aligned}
& \alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} \\
\Rightarrow\ & f \circ \gamma \sqsubseteq \gamma \circ \overline{f} && \text{⦃def. Galois connection⦄} \\
\Rightarrow\ & \alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \alpha \circ \gamma \circ \overline{f} && \text{⦃Galois connection so } \alpha \text{ is increasing⦄} \\
\Rightarrow\ & \alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} && \text{⦃Galois connection so } \alpha \circ \gamma \mathrel{\dot{\preccurlyeq}} \mathbb{1}_{\mathcal{A}}\text{⦄}
\end{aligned}
$$

$\alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} \Leftrightarrow \alpha \circ f \mathrel{\dot{\preccurlyeq}} \overline{f} \circ \alpha$ holds if $\overline{f}$ is increasing or the Galois connection is a retraction so $\alpha \circ \gamma = \mathbb{1}_{\mathcal{A}}$.

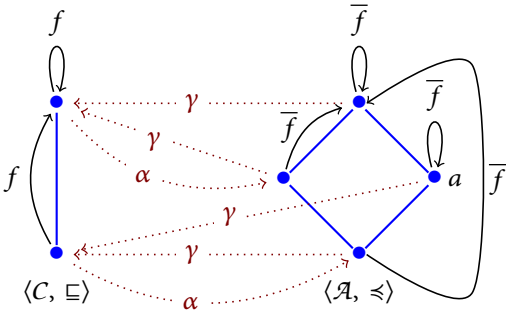
$$
\begin{aligned}
& \alpha \circ f \mathrel{\dot{\preccurlyeq}} \overline{f} \circ \alpha \\
\Rightarrow\ & \alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} \circ \alpha \circ \gamma && \text{⦃function application⦄} \\
\Rightarrow\ & \alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} \\
& && \text{⦃if } \alpha \circ \gamma = \mathbb{1}_{\mathcal{A}} \text{ or } \overline{f} \text{ increasing since } \alpha \circ \gamma \mathrel{\dot{\preccurlyeq}} \mathbb{1}_{\mathcal{A}} \text{ by the Galois connection⦄}
\end{aligned}
$$

$\Rightarrow\ \alpha \circ f \circ \gamma \circ \alpha \mathrel{\dot\preccurlyeq} \overline{f} \circ \alpha$ ⟅function application⟆

$\Rightarrow\ \alpha \circ f \mathrel{\dot\preccurlyeq} \overline{f} \circ \alpha$

⟅by $\mathbb{1}_C \sqsubseteq \gamma \circ \alpha$ and α increasing in a Galois connection and f increasing⟆

This may not hold when $\overline{f}$ is not increasing, as shown by the following counter-example where $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$, $f \in C \xrightarrow{\nearrow} C$, $\alpha \circ f \mathrel{\dot\preccurlyeq} \overline{f} \circ \alpha$ but $\alpha \circ f \circ \gamma(a) \not\preccurlyeq \overline{f}(a)$.



□

Solution to exercise 18.15 If $\overline{f}(y) \preccurlyeq y$ then $f(\gamma(y)) \sqsubseteq \gamma(\overline{f}(y)) \sqsubseteq \gamma(y)$ by semi-commutation $f \circ \gamma \mathrel{\dot\sqsubseteq} \gamma \circ \overline{f}$ and γ increasing. So $\gamma(y) \in \{x \mid f(x) \sqsubseteq x\}$ proving, by Tarski's fixpoint Theorem 15.6 and def. lub, that $\mathsf{lfp}^{\sqsubseteq}\, f = \sqcup\{x \mid f(x) \sqsubseteq x\} \sqsubseteq \gamma(y)$. This hold for any fixpoint y of $\overline{f}$, if any, by reflexivity. □

Solution to exercise 18.30 $\langle D \xrightarrow{\sqcup} D, \mathrel{\dot\sqsubseteq}\rangle$ and $\langle D, \sqsubseteq\rangle$ are complete lattices, $\mathcal{F} \in (D \xrightarrow{\sqcup} D) \xrightarrow{\sqcup} (D \xrightarrow{\sqcup} D)$ is $\mathrel{\dot\sqsubseteq}$-increasing. We have $\langle D \xrightarrow{\sqcup} D, \mathrel{\dot\sqsubseteq}\rangle \underset{\alpha_x}{\overset{\gamma_x}{\leftrightarrows}} \langle D, \sqsubseteq\rangle$ by Exercise 11.38 since lubs exist in a complete lattice and α_x preserves arbitrary joins:

$$
\begin{array}{ll}
& \alpha_x(\dot{\bigsqcup_i} f_i) \\
= & \mathcal{F}(\dot{\bigsqcup_i} f_i)x \qquad \text{⟅def. } \alpha_x\text{⟆} \\
= & (\dot{\bigsqcup_i} \mathcal{F}(f_i))x \qquad \text{⟅}\mathcal{F}\text{ preserves joins⟆} \\
= & \bigsqcup_i (\mathcal{F}(f_i)x) \qquad \text{⟅pointwise def. } \dot{\sqcup}\text{⟆} \\
= & \bigsqcup_i \alpha_x(f_i) \qquad \text{⟅def. } \alpha_x\text{⟆}
\end{array}
$$

$F(x) \in D \xrightarrow{\sqcup} D$ is $\sqsubseteq$-increasing and we have the commutation property $\alpha_x \circ \mathcal{F} = F(x) \circ \alpha_x$. By Theorem 18.21, it follows that $\mathsf{lfp}^{\sqsubseteq}\, F(x) = \alpha_x(\mathsf{lfp}^{\mathrel{\dot\sqsubseteq}}\, \mathcal{F}) = \mathcal{F}(\mathsf{lfp}^{\mathrel{\dot\sqsubseteq}}\, \mathcal{F})x = (\mathsf{lfp}^{\mathrel{\dot\sqsubseteq}}\, \mathcal{F})x$ for all $x \in D$ so $\mathsf{lfp}^{\mathrel{\dot\sqsubseteq}}\, \mathcal{F} = x \in D \mapsto \mathsf{lfp}^{\sqsubseteq}\, F(x)$. □

Solution to exercise 18.40 By $\gamma(\mathbf{0}) = \bot$, we have $\gamma(\overline{f}^0(\mathbf{0})) = f^0(\bot)$. By recurrence using $f \circ \gamma = \gamma \circ \overline{f}$, we have $\forall n \in \mathbb{N} \,.\, \gamma(\overline{f}^n(\mathbf{0})) = f^n(\bot)$. Because $\mathbf{0}$ is the infimum and $\overline{f}$ is increasing, the abstract iterates $\langle \overline{f}^n(\mathbf{0}),\ n \in \mathbb{N}\rangle$ form an increasing chain. By the ascending chain condition $\exists \ell \in \mathbb{N} \,.\, \forall n \geqslant \ell \,.\, \overline{f}^n(\mathbf{0}) = \overline{f}^\ell(\mathbf{0}) = \mathsf{lfp}^{\preccurlyeq}\, \overline{f}$. It follows, by Theorem 15.26 and γ increasing, that $\mathsf{lfp}^{\sqsubseteq}\, f = \bigsqcup_{n\in\mathbb{N}} f^n(\bot) = \bigsqcup_{n\in\mathbb{N}} \gamma(\overline{f}^n(\mathbf{0})) = \bigsqcup_{n\in\mathbb{N}} \gamma(\overline{f}^\ell(\mathbf{0})) = \gamma(\overline{f}^\ell(\mathbf{0})) = \gamma(\mathsf{lfp}^{\preccurlyeq}\, \overline{f}))$. □

10 Solutions to selected exercises of chapter 19

Solution to exercise 19.9 We have $\langle \wp(\mathbb{E}\mathtt{v} \times \mathbb{E}\mathtt{v}),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(\mathbb{E}\mathtt{v}),\ \subseteq\rangle$ with $\alpha(R) \triangleq \{\rho \mid \exists \rho_0 \in \mathbb{E}\mathtt{v} \,.\, \langle \rho_0,\ \rho\rangle \in R\}$ and $\gamma(r) \triangleq \{\langle \rho_0,\ \rho\rangle \mid \rho_0 \in \mathbb{E}\mathtt{v} \wedge \rho \in r\}$. By point wise extension in Exercise 11.20, it follows that $\langle \mathbb{L} \to \wp(\mathbb{E}\mathtt{v} \times \mathbb{E}\mathtt{v}),\ \dot{\subseteq}\rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle \mathbb{L} \to \wp(\mathbb{E}\mathtt{v}),\ \dot{\subseteq}\rangle$. It follows, by Theorem 11.75, that $\langle \wp(\mathbb{E}\mathtt{v} \times \mathbb{E}\mathtt{v}) \xrightarrow{\nearrow} (\mathbb{L} \to \wp(\mathbb{E}\mathtt{v} \times \mathbb{E}\mathtt{v})),\ \ddot{\subseteq}\rangle \underset{\vec{\alpha}}{\overset{\vec{\gamma}}{\leftrightarrows}} \langle \wp(\mathbb{E}\mathtt{v}) \xrightarrow{\nearrow} (\mathbb{L} \to \wp(\mathbb{E}\mathtt{v})),\ \ddot{\subseteq}\rangle$ where $\vec{\alpha} \triangleq \mathcal{S} \mapsto \dot{\alpha} \circ \mathcal{S} \circ \gamma$ and $\vec{\gamma} \triangleq \overline{\mathcal{S}} \mapsto \dot{\gamma} \circ \overline{\mathcal{S}} \circ \alpha$. Moreover, $\mathcal{S}^{\vec{\mathsf{r}}}[\![\mathtt{S}]\!] = \vec{\alpha}(\mathcal{S}^{\overline{\mathsf{R}}}[\![\mathtt{S}]\!])$. □

Solution to exercise 19.27 No, because of iteration. A counter-example is provided by Example 19.1 □

11 Solutions to selected exercises of chapter 21

Solution to exercise 21.20

- *Abstract invariant of an iteration statement* $\mathtt{S} ::= \mathtt{while}^{\,\ell}\ \mathtt{(B)}\ \mathtt{S}_b$

$$
\begin{array}{lcl}
\widehat{\mathcal{S}}^{\natural}[\![\mathsf{S}]\!]\,\mathcal{R}_0\,\ell' & = & \mathsf{let}\ \overline{\mathcal{F}}^{\natural}[\![\mathtt{while}^{\ell}\ \mathtt{(B)}\ \mathsf{S}_b]\!] \quad \in \quad \mathbb{P}^{\natural} \stackrel{\nearrow}{\longrightarrow} (\mathbb{P}^{\natural} \stackrel{\nearrow}{\longrightarrow} \mathbb{P}^{\natural}) \\
 & & \quad \overline{\mathcal{F}}^{\natural}[\![\mathtt{while}^{\ell}\ \mathtt{(B)}\ \mathsf{S}_b]\!]\,\mathcal{R}_0\,X \quad = \quad \mathcal{R}_0 \sqcup^{\natural} \widehat{\mathcal{S}}^{\natural}[\![\mathsf{S}_b]\!]\,(\mathsf{test}^{\natural}[\![\mathsf{B}]\!]X)\,\ell \\
 & & \mathsf{and}\ I \quad = \quad \mathsf{lfp}^{\sqsubseteq^{\natural}}\,\overline{\mathcal{F}}^{\natural}[\![\mathtt{while}^{\ell}\ \mathtt{(B)}\ \mathsf{S}_b]\!]\,\mathcal{R}_0\ \mathsf{in} \\
 & & \quad \{\!|\ \ell' = \ell\ ?\ I \\
 & & \quad |\!|\ \ell' \in \mathsf{in}[\![\mathsf{S}_b]\!] \setminus \{\ell\}\ ?\ \widehat{\mathcal{S}}^{\natural}[\![\mathsf{S}_b]\!]\,(\mathsf{test}^{\natural}[\![\mathsf{B}]\!]\,I)\,\ell' \\
 & & \quad |\!|\ \ell' = \mathsf{after}[\![\mathsf{S}]\!]\ ?\ \overline{\mathsf{test}}^{\natural}[\![\mathsf{B}]\!]\,I \sqcup^{\natural} \bigsqcup_{\ell'' \in \mathsf{breaks\text{-}of}[\![\mathsf{S}_b]\!]}^{\natural} \widehat{\mathcal{S}}^{\natural}[\![\mathsf{S}_b]\!]\,(\mathsf{test}^{\natural}[\![\mathsf{B}]\!]\,I)\,\ell'' \\
 & & \quad :\ \varnothing\ |\!\}
\end{array}
$$

The proof is similar to that of Exercise 19.18 with $I = (\mathsf{lfp}^{\sqsubseteq^{\natural}}\,\mathcal{F}^{\natural}[\![\mathtt{while}^{\ell}\ \mathtt{(B)}\ \mathsf{S}_b]\!]\,\mathcal{R}_0)\,\ell$. □

12 Solutions to selected exercises of chapter 22

13 Solutions to selected exercises of chapter 23

Solution to exercise 23.16 In OCaml,

```
type abstract_property = BOT | INT of int | TOP

let leq a1 a2 = match (a1,a2) with
| (BOT,_) -> true
| (_,BOT) -> false
| (_,TOP) -> true
| (TOP,_) -> false
| (INT v1, INT v2) -> (v1=v2)

let join a1 a2 = match (a1,a2) with
| (BOT,a2) -> a2
| (a1,BOT) -> a1
| (_,TOP) -> TOP
| (TOP,_) -> TOP
| (INT v1, INT v2) -> if (v1=v2) then INT v1 else TOP

let test_x_gt i a = match a with
| BOT -> BOT
| INT v -> if (v>i) then a else BOT
| TOP -> TOP
```

```
let negtest_x_gt i a = match a with
| BOT -> BOT
| INT v -> if (v<=i) then a else BOT
| TOP -> TOP

let assign_incr_x i a = match a with
| BOT -> BOT
| INT v -> INT (v+i)
| TOP -> TOP

let eqns r0  (xl1, xl2, xl3, xl4, xl5) =
  (join r0 (negtest_x_gt 9 xl3),
   test_x_gt 0 xl1,
   assign_incr_x 1 xl2,
   test_x_gt 9 xl3,
   join (test_x_gt 0 xl1) xl4)

let pbot = (BOT, BOT, BOT, BOT, BOT)

let pleq (a1, a2, a3, a4, a5) (a'1, a'2, a'3, a'4, a'5) = (leq a1 a'1)
   && (leq a2 a'2) && (leq a3 a'3) && (leq a4 a'4) && (leq a5 a'5)

let rec lfp a f leq = if leq (f a) a then a else lfp (f a) f leq

lfp pbot (eqns (INT 0)) pleq;; (* = (INT 0, BOT, BOT, BOT, BOT) *)
lfp pbot (eqns (INT 1)) pleq;; (* = (TOP, TOP, TOP, TOP, TOP) *)
```
□

14 Solutions to selected exercises of chapter 24

Solution to exercise 24.5 Apply Tarski's theorem to the complete lattice $\langle \{x \in L \mid a \sqsubseteq x\}, \sqsubseteq \rangle$.
□

Solution to exercise 24.7 $\langle \mathcal{X} \to \mathcal{L}, \dot{\sqsubseteq}, \dot{\bot}, \dot{\top}, \dot{\sqcap}, \dot{\sqcup} \rangle$ is a complete lattice for the pointwise partial order $\dot{\sqsubseteq}$. By Tarski fixpoint Theorem 15.6, $\mathsf{lfp}^{\dot{\sqsubseteq}}\, F = \dot{\bigsqcap}\{f \mid F(f) \mathrel{\dot{\sqsubseteq}} f\}$ so for all $x \in \mathcal{X}$, $(\mathsf{lfp}^{\dot{\sqsubseteq}}\, F)x = \bigsqcap\{f(x) \mid F(f)x \sqsubseteq f(x)\}$. Following the proof of Theorem 24.1, $(\mathsf{lfp}^{\dot{\sqsubseteq}}\, F)x \sqsubseteq P(x)$ if and only if $\exists I \in \mathcal{X} \to \mathcal{L}\,.\, F(I)x \sqsubseteq I(x) \wedge I(x) \sqsubseteq P(x)$. □

Solution to exercise 24.8 We apply Theorem 24.1 with $P = f_!$. $F_!$ is increasing on the complete lattice $\langle \wp(\mathbb{Z} \times \mathbb{Z}), \subseteq \rangle$. Let $I = f_!$ so that (b) holds. Moreover $F_!(I) = F_!(f_!) = \{\langle 0,\ 1\rangle\} \cup \{\langle n,\ n \times y\rangle \mid \langle n-1,\ y\rangle \in f_!\} = \{\langle 0,\ 1\rangle\} \cup \{\langle n,\ n \times (n-1)!\rangle \mid n - 1 \geqslant \mathbb{N}\} = f_! = I$, proving (b) by reflexivity of $\subseteq$. We conclude that $\mathsf{lfp}^{\sqsubseteq}\, F \subseteq P = f_!$. □

Solution to exercise 24.13 Define $\mathcal{P} = \{r \in \wp(\mathbb{Z} \times \mathbb{Z}) \mid \forall n \in \mathbb{N} \,.\, \exists y \in \mathbb{Z} \,.\, \langle n,\ y\rangle \in r\}$. We use Theorem 24.11 to prove that $\mathsf{lfp}^{\subseteq} F_! \in \mathcal{P}$. Let us define $\forall i \in \mathbb{N} \,.\, Q_i \triangleq \{r \in \wp(\mathbb{Z} \times \mathbb{Z}) \mid \forall n \in [0, i[\,.\, \exists y \in \mathbb{Z} \,.\, \langle n,\ y\rangle \in r \wedge \forall n \geqslant i \,.\, \forall y \in \mathbb{Z} \,.\, \langle n,\ y\rangle \notin r\}$ and $Q \triangleq \bigcup_{i\in\mathbb{N}} Q_i$.

– We have $\varnothing \in \{\varnothing\} = Q_0 \subseteq Q$, proving (24.11.a).
– Assume that $i \in \mathbb{N}$ and $r \in Q_i$. We have

$$\begin{aligned}
&F_!(r)\\
&= \{\langle 0,\ 1\rangle\} \cup \{\langle n,\ n \times y\rangle \mid \langle n-1,\ y\rangle \in r\} && \text{(def. } F_! \text{ in Exercise 2.14)}\\
&\subseteq \{\langle 0,\ 1\rangle\} \cup \{\langle n,\ n \times y\rangle \mid \langle n-1,\ y\rangle \in r \wedge \forall k \in [0, i[\,.\, \exists z \in \mathbb{Z} \,.\, \langle k,\ z\rangle \in r\} && (r \in Q_i)\\
&\subseteq \{\langle 0,\ 1\rangle\} \cup \{\langle k,\ z\rangle \mid k \in [1, i] \wedge z \in \mathbb{Z}\} \in Q_{i+1} \subseteq Q && \text{(def. } Q_{i+1})
\end{aligned}$$

proving $F_!(r) \in Q_{i+1}$.
– If $r \in Q$ then $r \in Q_i$ for some $i \in \mathbb{N}$. Then $F_!(r) \in Q_{i+1} \subseteq Q$ proving (24.11.b).
– Let $r_0 \subseteq r_1 \subseteq \ldots \subseteq r_i \subseteq \ldots$ be a $F_!$-maximal increasing chain of elements of Q. There are two cases.

Either the chain is stationary at some rank i such that $r_0 \subsetneq r_1 \subsetneq \ldots \subsetneq r_i = r_{i+1} = r_{i+2} = \ldots$. Then $r_i \in Q_j$ for some $j \in \mathbb{N}$. So $r_{i+1} = r_i \in Q_j$ and $r_{i+1} = F_!(r_i) \in Q_{j+1}$, a contradiction since $Q_j \cap Q_{j+1} = \varnothing$.

Since the chain is assumed to be a $F_!$-maximal increasing chain, the only other possibility is that it is strictly increasing $r_0 \subsetneq r_1 \subsetneq \ldots \subsetneq r_n \subsetneq \ldots$. Then, by def. Q, we have $r_0 \in Q_{j_0}$, $r_1 \in Q_{j_1}$, …, $r_n \in Q_{j_n}$, $r_{n+1} \in Q_{j_{n+1}}$, …. Since the chain $r_0 \subsetneq r_1 \subsetneq \ldots \subsetneq r_n \subsetneq \ldots$ is strictly increasing, we have $j_0 < j_1 < \ldots < j_n < j_{n+1} < \ldots$ so $j_{n+1} > n+1$. Since $r_{n+1} \in Q_{j_{n+1}}$, $\exists y \in \mathbb{Z} \,.\, \langle n,\ y\rangle \in r_{n+1}$.

To prove that $\bigcup_{i\in\mathbb{N}} r_i \in \mathcal{P}$, assume by reductio ad absurdum, that $\exists n \in \mathbb{N} \,.\, \forall y \in \mathbb{Z} \,.\, \langle n,\ y\rangle \notin \bigcup_{i\in\mathbb{N}} r_i$ so $\exists n \in \mathbb{N} \,.\, \forall y \in \mathbb{Z} \,.\, \forall i \in \mathbb{N} \,.\, \langle n,\ y\rangle \notin r_i$. In particular $\forall y \in \mathbb{Z} \,.\, \langle n,\ y\rangle \notin r_{n+1}$, a contradiction.

We have proved (24.11.c) hence $\mathsf{lfp}^{\subseteq} F_! \in \mathcal{P}$, that is $\forall n \in \mathbb{N} \,.\, \exists y \in \mathbb{Z} \,.\, \langle n,\ y\rangle \in \mathsf{lfp}^{\subseteq} F_!$.

Another way of proving termination for a positive parameter is to observe that it strictly decreases on recursive call and remains positive which can be done only a finite number of times since $\langle \mathbb{N},\ <\rangle$ is well-founded [**DBLP:conf/popl/CousotC12**]. □

Solution to exercise 24.15 $\langle L, \sqsubseteq, \bot, \sqcup\rangle$ is a complete lattice so $\langle (L \to L), \dot{\sqsubseteq}, \dot{\bot}, \dot{\sqcup}\rangle$ is a complete lattice, pointwise. The Galois connection $\langle (L \to L), \dot{\sqsubseteq}\rangle \underset{\vec{F}}{\overset{\vec{F}}{\leftrightarrows}} \langle (L \to L), \dot{\sqsubseteq}\rangle$ implies that $\vec{F}$ preserves existing lubs by Lemma 11.37 so is upper-continuous proving that $\mathsf{lfp}^{\sqsubseteq} \vec{F}$ exists by Scott's iterative fixpoint Theorem 15.26. By duality, $\mathsf{gfp}^{\sqsubseteq} \vec{F}$ does exist.

Let us proof by recurrence on $n \in \mathbb{N}$ that $\vec{F}^n(X) \mathrel{\dot{\sqsubseteq}} Y \Leftrightarrow X \mathrel{\dot{\sqsubseteq}} \vec{F}^n(Y)$.

- for the basis $\vec{F}^0(X) = X \mathrel{\dot{\sqsubseteq}} Y \Leftrightarrow X \mathrel{\dot{\sqsubseteq}} Y = \vec{F}^0(Y)$;
- for the induction step,

$$\begin{aligned}
&\vec{F}^{n+1}(X) \mathrel{\dot{\sqsubseteq}} Y\\
&\Leftrightarrow \vec{F}(\vec{F}^n(X)) \mathrel{\dot{\sqsubseteq}} Y && \text{(def. iterates)}
\end{aligned}$$

$\Leftrightarrow \vec{F}^n(X) \mathrel{\dot\sqsubseteq} \bar{F}(Y)$ ⟅Galois connection hypothesis⟆

$\Leftrightarrow X \mathrel{\dot\sqsubseteq} \bar{F}^n(\bar{F}(Y))$ ⟅recurrence hypothesis⟆

$\Leftrightarrow X \mathrel{\dot\sqsubseteq} \bar{F}^{n+1}(Y)$ ⟅def. iterates⟆

It follows that

$(\mathsf{lfp}^{\sqsubseteq}\, \vec{F})(X) \sqsubseteq Y$

$\Leftrightarrow (\dot{\bigsqcup_{n\in\mathbb{N}}} \vec{F}^n(\bot))(X) \sqsubseteq Y$ ⟅Scott's iterative fixpoint Theorem 15.26⟆

$\Leftrightarrow \bigsqcup_{n\in\mathbb{N}} (\vec{F}^n(\bot)(X)) \sqsubseteq Y$ ⟅pointwise def. $\dot\bigsqcup$⟆

$\Leftrightarrow \forall n \in \mathbb{N} \,.\, (\vec{F}^n(\bot)(X)) \sqsubseteq Y$ ⟅def. lub $\bigsqcup$⟆

$\Leftrightarrow \forall n \in \mathbb{N} \,.\, X \sqsubseteq \bar{F}^n(\bot)(Y)$ ⟅$\forall n \in \mathbb{N} \,.\, \langle (L \to L), \dot\sqsubseteq\rangle \underset{\vec{F}^n}{\overset{\bar{F}^n}{\leftrightarrows}} \langle (L \to L), \dot\sqsubseteq\rangle$⟆

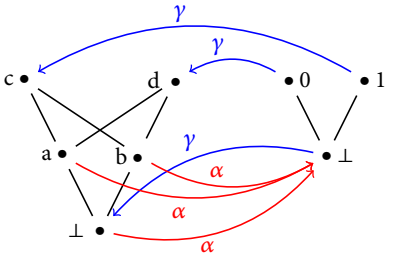
$\Leftrightarrow X \sqsubseteq \bigsqcap_{n\in\mathbb{N}} \bar{F}^n(\bot)(Y)$ ⟅def. glp $\bigsqcap$⟆

$\Leftrightarrow X \sqsubseteq (\dot{\bigsqcap_{n\in\mathbb{N}}} \bar{F}^n(\bot))(Y)$ ⟅pointwise def. $\dot\bigsqcap$⟆

$\Leftrightarrow X \sqsubseteq (\mathsf{gfp}^{\sqsubseteq}\, \bar{F})(Y)$ ⟅dual of Scott's iterative fixpoint Theorem 15.26⟆ □

15 Solutions to selected exercises of chapter 27

Solution to exercise 27.14 In the example below, c and d have no greatest lower bound.



We have $a \sqsubseteq c = \gamma(1)$, $a \sqsubseteq d = \gamma(0)$, $a \not\sqsubseteq \bot = \gamma(\bot)$ so $\alpha(a) = 0 \sqcap 1 = \bot$ but $a \not\sqsubseteq b = \gamma(\bot)$ so $a \not\sqsubseteq \gamma \circ \alpha(a)$ proving by Exercise 11.34.(3) that α is not the adjoint of γ. □

16 Solutions to selected exercises of chapter 28

Solution to exercise 28.2

$$
\begin{aligned}
& \dot\alpha_{\dot x}(P) \mathrel{\dot\subseteq} \overline{P} \\
\Leftrightarrow\ & \forall x \in \mathbb{V} \,.\, \dot\alpha_{\dot x}(P)x \subseteq \overline{P}(x) && \text{⟅pointwise def, } \dot\subseteq\text{⟆} \\
\Leftrightarrow\ & \forall x \in \mathbb{V} \,.\, \{\rho(x) \mid \rho \in P\} \subseteq \overline{P}(x) && \text{⟅def. (28.1) of } \dot\alpha_{\dot x}\text{⟆} \\
\Leftrightarrow\ & \forall x \in \mathbb{V} \,.\, \forall \rho \in P \,.\, \rho(x) \in \overline{P}(x) && \text{⟅def. } \subseteq\text{⟆} \\
\Leftrightarrow\ & \forall \rho \in P \,.\, \forall x \in \mathbb{V} \,.\, \rho(x) \in \overline{P}(x) && \text{⟅def. } \forall\text{⟆} \\
\Leftrightarrow\ & P \subseteq \{\rho \in \mathbb{E}_\forall \mid \forall x \in \mathbb{V} \,.\, \rho(x) \in \overline{P}(x)\} && \text{⟅def. } \subseteq\text{⟆} \\
\Leftrightarrow\ & P \subseteq \dot\gamma_{\dot x}(\overline{P}) && \text{⟅def. (28.1) of } \dot\gamma_{\dot x}\text{⟆} \quad \square
\end{aligned}
$$

17 Solutions to selected exercises of chapter 29

Solution to exercise 29.4 Assuming that (28.38) is implemented by `basic_test b` and `notbasic_test b`, the local iterations are the following.

```
let rec iterate f a  =
   let x = (narrow a (f a)) in
      if (leq a x) then a else iterate f x
let test b r = iterate (basic_test b) r
let nottest b r = iterate (notbasic_test b) r
```

□

18 Solutions to selected exercises of chapter 31

Solution to exercise 31.16

```
let abs n = if n=min_int then failwith "abs: overflow"
            else if n<0 then -n else n
type abstractProperty = BOT | CC of (int * int) (* c+mZ, $0$ <= c < m *)
let normalize c m = if m=0 then (c,m) else (c mod (abs m), abs(m))
let meet a1 a2 = match (a1,a2) with
    | (BOT,_) -> BOT
    | (_,BOT) -> BOT
    | (CC (c1,m1),CC (c2,m2)) ->
      let g = gcd m1 m2 in
         let (_,r) = divide (abs (c1-c2)) g in
            if r=0 then
               let m = lcm m1 m2 in
                  let (_,c) = divide c1 m in
                     CC (normalize c m)
            else BOT
# meet (CC (0,3)) (CC (3,7));;      #  meet (CC (0,3)) (CC (1,3));;
- : abstractProperty = CC (0, 21)   - : abstractProperty = BOT
```

□

19 Solutions to selected exercises of chapter 38

Solution to exercise 38.3 If $\vec{x} \in P$ then $\vec{x} = \overrightarrow{x_0} + \mathbf{B}\vec{a}$ for some coefficients $\vec{a}$. So $\vec{x} = \overrightarrow{x_0} + (\mathbf{B}, \mathbf{B}', \vec{x_0}' - \vec{x_0}) \begin{bmatrix} \vec{a} \\ \vec{0} \\ 0 \end{bmatrix}$ proving $\vec{x} \in P \vec{\sqcup} P'$ (or $\vec{x} = \overrightarrow{x_0} + (\mathbf{B}, \mathbf{B}') \begin{bmatrix} \vec{a} \\ \vec{0} \end{bmatrix}$ when $\vec{x_0}' = \vec{x}_0$). Similarly, if $\vec{x} \in P'$ then $\vec{x} = \overrightarrow{x_0}' + \mathbf{B}'\vec{a}'$ for some coefficients $\vec{a}'$. So $\vec{x} = \overrightarrow{x_0} + (\mathbf{B}, \mathbf{B}', \vec{x_0}' - \vec{x_0}) \begin{bmatrix} \vec{0} \\ \vec{a}' \\ 1 \end{bmatrix}$ proving $\vec{x} \in P \vec{\sqcup} P'$ (or $\vec{x} = \overrightarrow{x_0}' + (\mathbf{B}, \mathbf{B}') \begin{bmatrix} \vec{0} \\ \vec{a}' \end{bmatrix}$ when $\vec{x_0}' = \vec{x}_0$). So $P \cup P' \subseteq P \vec{\sqcup} P'$.

Let Q with system of generators $\langle \overrightarrow{x_Q}, \mathbf{B}_Q \rangle$ be another upper bound of P and P'. Let us prove that $P \vec{\sqcup} P' \vec{\sqsubseteq} Q$ *i.e.* $P \vec{\sqcup} P' \subseteq Q$.

If $\vec{x} \in P \subseteq Q$ then $\exists \vec{a}\ .\ \vec{x} = \overrightarrow{x_0} + \mathbf{B}\vec{a}$ so $\exists \vec{b}$ such that $\vec{x} = \overrightarrow{x_Q} + \mathbf{B}_Q\vec{b}$. Similarly, if $\vec{x}' \in P' \subseteq Q$ then $\exists \vec{a}'\ .\ \vec{x}' = \overrightarrow{x_0}' + \mathbf{B}'\vec{a}'$ so $\exists \vec{b}'$ such that $\vec{x} = \overrightarrow{x_Q} + \mathbf{B}_Q\vec{b}'$. Therefore if $\vec{x} \in P \vec{\sqcup} P'$ then $\exists \vec{a}, \vec{a}', c\ .$ $\vec{x} = \overrightarrow{x_0} + (\mathbf{B}, \mathbf{B}', \vec{x_0}' - \vec{x_0}) \begin{bmatrix} \vec{a} \\ \vec{a}' \\ c \end{bmatrix} = \overrightarrow{x_0} + \mathbf{B}\vec{a} + \mathbf{B}'\vec{a}' + c(\vec{x_0}' - \vec{x_0}) = (\overrightarrow{x_Q} + \mathbf{B}_Q\vec{b}) + (\overrightarrow{x_Q} - \overrightarrow{x_0}' + \mathbf{B}_Q\vec{b}') + c(\vec{x_0}' - \vec{x_0}) = 2\overrightarrow{x_Q} + (c-1)\vec{x_0}' - c\vec{x}_0 + \mathbf{B}_Q(2\vec{b})$, which is a point of the affine space Q with the origin translated by $\overrightarrow{x_Q} + (c-1)\vec{x_0}' - c\vec{x}_0$, which, by def. of an affine space in Section 37.6.1 is the affine space Q itself. The reasoning is the same when $\vec{x_0}' = \vec{x}_0$. So $P \vec{\sqcup} P' \subseteq Q$. It follows, by the Galois connection (38.2) and Lemma 11.37, that for all $P, Q \in \wp(\mathbb{F}^m)\ .\ \alpha_{\mathbb{A}}(P \cup Q) = \alpha_{\mathbb{A}}(P) \vec{\sqcup} \alpha_{\mathbb{A}}(Q)$. □

20 Solutions to selected exercises of chapter 41

Solution to exercise 41.11 By choosing $v \neq \rho(\mathsf{y})$, we have $\rho(\mathsf{y}) = \mathscr{A}[\![\mathsf{A}]\!]\,\rho \neq \mathscr{A}[\![\mathsf{A}]\!]\,\rho[\mathsf{y} \leftarrow v] = v$. So $\mathsf{use}[\![\mathsf{x = y}]\!]\,\rho \triangleq \{\mathsf{y} \mid \rho(\mathsf{x}) \neq \rho(\mathsf{y})\}$ since when $\rho(\mathsf{x}) = \rho(\mathsf{y})$ the assignment can be skipped. □

21 Solutions to selected exercises of chapter 44

Solution to exercise 44.15 Very informally,

$(\mathsf{R}_1 \mid \mathsf{R}_2)^*$

$\eqsim (\mathsf{R}_1 \mid \mathsf{R}_2)(\mathsf{R}_1 \mid \mathsf{R}_2) \ldots (\mathsf{R}_1 \mid \mathsf{R}_2)$ ⟅def. *, 0 or more times⟆

$\eqsim \mathsf{R}_1^*\mathsf{R}_2^*\mathsf{R}_1^*\mathsf{R}_2^* \ldots \mathsf{R}_1^*\mathsf{R}_2^*$ ⟅grouping/ungrouping consecutive R_1's and R_2's using ε if necessary⟆

$\eqsim (\mathsf{R}_1{}^*\mathsf{R}_2{}^*)^*$ ⟅def. *, 0 or more times⟆ □

Solution to exercise 44.17

Proof of Lemma 44.16 The proof is by structural induction. The base cases for ε and $\mathtt{L} : \mathtt{B}$ are trivial. The cases $\mathsf{R}_1 \mid \mathsf{R}_2$ and (R) are an easy induction. Otherwise,

– $\mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_1\mathsf{R}_2)]\!]$

$= \mathcal{S}^r[\![\bigmid_{i=1}^{n_1}\bigmid_{j=1}^{n_2} \mathsf{R}_1^i\mathsf{R}_2^j]\!]$ ⟅def. dnf⟆

$= \bigcup_{i=1}^{n_1}\bigcup_{j=1}^{n_2} \mathcal{S}^r[\![\mathsf{R}_1^i\mathsf{R}_2^j]\!]$ ⟅def. $\mathcal{S}^r$⟆

$= \bigcup_{i=1}^{n_1}\bigcup_{j=1}^{n_2} \mathcal{S}^r[\![\mathsf{R}_1^i]\!] \circledcirc \mathcal{S}^r[\![\mathsf{R}_2^j]\!]$ ⟅def. $\mathcal{S}^r$⟆

$= \bigcup_{i=1}^{n_1} \mathcal{S}^r[\![\mathsf{R}_1^i]\!] \bullet \bigcup_{j=1}^{n_2} \mathcal{S}^r[\![\mathsf{R}_2^j]\!]$ ⟅def. $\bullet$⟆

$= \mathcal{S}^r[\![\bigmid_{i=1}^{n_1} \mathsf{R}_1^i]\!] \bullet \mathcal{S}^r[\![\bigmid_{j=1}^{n_2} \mathsf{R}_2^j]\!]$ ⟅def. $\mathcal{S}^r$⟆

$= \mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_1)]\!] \bullet \mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_2)]\!]$ ⟅def. dnf where $\mathsf{R}_k^1 \mid \ldots \mid \mathsf{R}_k^{n_k} = \mathsf{dnf}(\mathsf{R}_k), k = 1, 2$⟆

$= \mathcal{S}^r[\![\mathsf{R}_1]\!] \bullet \mathcal{S}^r[\![\mathsf{R}_2]\!]$ ⟅ind. hyp.⟆

$= \mathcal{S}^r[\![\mathsf{R}_1\mathsf{R}_2]\!]$ ⟅def. $\mathcal{S}^r$⟆

– $\mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}^*)]\!]$

$= \mathcal{S}^r[\![((\mathsf{R}^1)^* \ldots (\mathsf{R}^n)^*)^*]\!]$ ⟅def. dnf where ⟆

$= \mathcal{S}^r[\![(\mathsf{R}^1 \mid \ldots \mid \mathsf{R}^n)^*]\!]$ ⟅Exercise 44.15⟆

$= \bigcup_{k\geqslant 0} \mathcal{S}^r[\![(\mathsf{R}^1 \mid \ldots \mid \mathsf{R}^n)]\!]^k$ ⟅def. $\mathcal{S}^r$⟆

$= \bigcup_{k\geqslant 0} \mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R})]\!]^k$ ⟅$\mathsf{R}^1 \mid \ldots \mid \mathsf{R}^n = \mathsf{dnf}(\mathsf{R})$⟆

$= \bigcup_{k\geqslant 0} \mathcal{S}^r[\![\mathsf{R}]\!]^k$ ⟅ind. hyp.⟆

$= \mathcal{S}^r[\![\mathsf{R}^*]\!]$ ⟅def. $\mathcal{S}^r$⟆

The proof for R^+ is similar. □

Solution to exercise 44.20

Proof of Lemma 44.19 The proof that $\mathsf{R}' \in \mathbb{R}^{+}$ is $\mathsf{|}$-free is by structural on R, observing that the definition (44.18) of fstnxt involves no alternative $\mathsf{|}$. The proof that $\mathsf{R} \eqsim \mathsf{L} : \mathsf{B} \bullet \mathsf{R}'$ that is $\mathbb{S}^{r}[\![\mathsf{R}]\!] = \mathbb{S}^{r}[\![\mathsf{L} : \mathsf{B} \bullet \mathsf{R}']\!]$ is by structural on R.

- Let us first prove that $\ni$ is the neutral element of $\bullet$.

  $\mathbb{S}^{r}[\![\mathsf{R} \bullet \varepsilon]\!]$
  $= \{\langle \underline{\varrho},\ \pi \cdot \pi' \rangle \mid \langle \underline{\varrho},\ \pi \rangle \in \mathbb{S}^{r}[\![\mathsf{R}]\!] \wedge \langle \underline{\varrho},\ \pi' \rangle \in \mathbb{S}^{r}[\![\varepsilon]\!]\}$ ⟅(44.7)⟆
  $= \{\langle \underline{\varrho},\ \pi \cdot \ni \rangle \mid \langle \underline{\varrho},\ \pi \rangle \in \mathbb{S}^{r}[\![\mathsf{R}]\!]\}$ ⟅since $\pi' = \ni$ by (44.7)⟆
  $= \mathbb{S}^{r}[\![\mathsf{R}]\!]$ ⟅def. concatenation $\cdot$ and $\in$⟆

  Similarly $\varepsilon \bullet \mathsf{R} \cdot \mathsf{R}$ and this extends to all $\mathsf{R}' \in \mathbb{R}_{\varepsilon}$.

- It follows that Lemma 44.19 holds for $\mathsf{fstnxt}(\mathsf{L} : \mathsf{B})$ and $\mathsf{fstnxt}(\mathsf{R}_1\mathsf{R}_2)$ when $\mathsf{R}_1 \in \mathbb{R}_{\varepsilon}$.

- For $\mathsf{fstnxt}(\mathsf{R}_1\mathsf{R}_2)$ when $\mathsf{R}_1 \notin \mathbb{R}_{\varepsilon}$, there are two cases.

  – Either $\mathsf{R}_1^{n} \in \mathbb{R}_{\varepsilon}$ and then

    $\mathsf{R}_1^{f} \bullet \mathsf{R}_2$
    $\eqsim \mathsf{R}_1^{f} \bullet \mathsf{R}_1^{n} \bullet \mathsf{R}_2$ ⟅since $\mathsf{R}_1^{n} \in \mathbb{R}_{\varepsilon}$ so $\mathsf{R}_1^{f} \bullet \mathsf{R}_1^{n} \eqsim \mathsf{R}_1^{f}$⟆
    $\eqsim \mathsf{R}_1 \bullet \mathsf{R}_2$ ⟅by ind. hyp. since $\langle \mathsf{R}_1^{f},\ \mathsf{R}_1^{n} \rangle = \mathsf{fstnxt}(\mathsf{R}_1)$, *Q.E.D.*⟆

  – Otherwise $\mathsf{R}_1^{n} \notin \mathbb{R}_{\varepsilon}$ and then

    $\mathsf{R}_1^{f} \bullet \mathsf{R}_1^{n} \bullet \mathsf{R}_2$
    $\eqsim \mathsf{R}_1 \bullet \mathsf{R}_2$ ⟅by ind. hyp. since $\langle \mathsf{R}_1^{f},\ \mathsf{R}_1^{n} \rangle = \mathsf{fstnxt}(\mathsf{R}_1)$, *Q.E.D.*⟆

- For $\mathsf{fstnxt}(\mathsf{R}^{+})$, let $\langle \mathsf{R}^{f},\ \mathsf{R}^{n} \rangle = \mathsf{fstnxt}(\mathsf{R})$. There are two cases.

  – Either $\mathsf{R}^{n} \in \mathbb{R}_{\varepsilon}$ and then

    $\mathsf{R}^{f} \bullet \mathsf{R}^{*}$
    $\eqsim \mathsf{R}^{f} \bullet \mathsf{R}^{n} \bullet \mathsf{R}^{*}$ ⟅since $\mathbb{S}^{r}[\![\mathsf{R}^{n}]\!] = \mathbb{S}^{r}[\![\varepsilon]\!]$⟆
    $\eqsim \mathsf{R} \bullet \mathsf{R}^{*}$ ⟅ind. hyp. since $\langle \mathsf{R}^{f},\ \mathsf{R}^{n} \rangle = \mathsf{fstnxt}(\mathsf{R})$⟆
    $\eqsim \mathsf{R}^{+}$ ⟅def. (44.7) of $\mathbb{S}^{r}[\![\mathsf{R}^{*}]\!]$ and $\mathbb{S}^{r}[\![\mathsf{R}^{+}]\!]$⟆

  – Otherwise $\mathsf{R}^{n} \notin \mathbb{R}_{\varepsilon}$ and then $\mathsf{R}^{f} \bullet \mathsf{R}^{n} \bullet \mathsf{R}^{*} \eqsim \mathsf{R}$, as shown above.

- The last case for $\mathsf{fstnxt}((\mathsf{R}))$ follows by structural induction from $\mathbb{S}^{r}[\![(\mathsf{R})]\!] \triangleq \mathbb{S}^{r}[\![\mathsf{R}]\!]$.

□

Solution to exercise 44.33 Define $\gamma_{\mathscr{M}^\dagger\langle \underline{\varrho}, \mathsf{R}\rangle}(M) \triangleq \{\pi \mid \forall \mathsf{R}' \in \mathbb{R} \,.\, (\langle \mathrm{tt}, \mathsf{R}'\rangle = \mathscr{M}^t\langle \rho, \mathsf{R}\rangle(\pi)) \Rightarrow (\pi \in M)\}$. □

Solution to exercise 44.53

— Let us first prove that $X \mapsto \vec{\tau} \frown X$ preserves arbitrary joins. If $\vec{\tau}$ is $\varnothing$, this is $\varnothing$ whichever is X. Because $\tau \in \wp(\mathbb{S} \times \mathbb{S})$, we cannot have $\tau = \ni$. Otherwise, if X is empty then $\vec{\tau} \frown \varnothing = \varnothing$. For $\Delta = \varnothing$, $\vec{\tau} \frown \bigcup_{i\in\varnothing} X_i = \vec{\tau} \frown \varnothing = \varnothing = \bigcup_{i\in\varnothing} \vec{\tau} \frown X_i$. Otherwise, assuming $\Delta \neq \varnothing$, we have

$$
\begin{aligned}
&\vec{\tau} \frown \Big(\bigcup_{i\in\Delta} X_i\Big) \\
&= \{\vec{\tau} \mid \ni \in \bigcup_{i\in\Delta} X_i\} \cup \{\sigma\sigma'\pi \mid \langle\sigma, \sigma'\rangle \in \tau \wedge \sigma'\pi \in \bigcup_{i\in\Delta} X_i\} && \text{(def. } \frown \text{ and } \vec{\tau}\text{)} \\
&= \bigcup_{i\in\Delta} \{\vec{\tau} \mid \ni \in X_i\} \cup \bigcup_{i\in\Delta} \{\sigma\sigma' \frown \sigma'\pi \mid \langle\sigma, \sigma'\rangle \in \tau \wedge \sigma'\pi \in X_i\} && \text{(def. } \bigcup \text{ and } \frown\text{)} \\
&= \bigcup_{i\in\Delta} \big(\{\vec{\tau} \mid \ni \in X_i\} \cup \{\sigma\sigma' \frown \sigma'\pi \mid \langle\sigma, \sigma'\rangle \in \tau \wedge \sigma'\pi \in X_i\}\big) && \text{(def. } \bigcup\text{)} \\
&= \bigcup_{i\in\Delta} (\vec{\tau} \frown X_i) && \text{(def. } \frown \text{ and } \vec{\tau}\text{)}
\end{aligned}
$$

It follows that $X \mapsto \mathbb{S}^1 \cup \vec{\tau} \frown X$ preserves non-empty joins.

$$
\begin{aligned}
&\mathbb{S}^1 \cup \Big(\vec{\tau} \frown \bigcup_{i\in\Delta} X_i\Big) \\
&= \mathbb{S}^1 \cup \bigcup_{i\in\Delta} (\vec{\tau} \frown X_i) && \text{(as shown above)} \\
&= \bigcup_{i\in\Delta} (\mathbb{S}^1 \cup \vec{\tau} \frown X_i) && \text{(} \bigcup \text{ associative)}
\end{aligned}
$$

It does not preserve empty joins since $\mathbb{S}^1 \cup \vec{\tau} \frown \bigcup_{i\in\varnothing} X_i = \mathbb{S}^1 \cup \vec{\tau} \frown \varnothing = \mathbb{S}^1 \neq \varnothing = \bigcup_{i\in\varnothing} (\mathbb{S}^1 \cup \vec{\tau} \frown X_i)$.

— By recurrence on n.

– for $n = 0$,

$$
\begin{aligned}
&X^0 \\
&= \varnothing && \text{(def. iterates from } \varnothing\text{)} \\
&= \bigcup_{0} \varnothing && \text{(def. } \bigcup\text{)} \\
&= \bigcup_{i=1} \mathscr{S}^i_{\mathsf{t}}[\![\tau]\!] && \text{(def. } \textstyle\bigcup_{i=1}^{j} x_i = \varnothing \text{ when } j < i\text{)}
\end{aligned}
$$

– for $n = 1$,

$$X^1$$

$= \mathbb{S}^1 \cup \vec{\tau} \frown X^0$ ⟅def. iterates⟆

$= \mathbb{S}^1$ ⟅$X^0 = \varnothing$ and def. $\frown$⟆

$= \boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!]$ ⟅$\boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!] = \mathbb{S}^1 \triangleq \{\pi \in \mathbb{S}^1 \mid \pi_0 = \iota_0\}$⟆

$= \bigcup_{i=1}^{1} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!]$ ⟅def. $\bigcup_{i=1}^{j} x_i = x_1$ with $j = i$⟆

— for the induction, assume that $X^n = \bigcup_{i=1}^{n} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!]$, by ind. hyp. Then

X^{n+1}

$= \mathbb{S}^1 \cup \vec{\tau} \frown X^n$ ⟅def. iterates⟆

$= \mathbb{S}^1 \cup \vec{\tau} \frown (\bigcup_{i=1}^{n} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!])$ ⟅ind. hyp.⟆

$= \mathbb{S}^1 \cup \bigcup_{i=1}^{n} (\vec{\tau} \frown \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!])$ ⟅$X \mapsto \vec{\tau} \frown X$ preserves arbitrary joins, as shown above⟆

$= \boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!] \cup \bigcup_{i=1}^{n} (\boldsymbol{\mathcal{S}}^{i+1}_{\mathrm{t}}[\![\tau]\!])$ ⟅$\mathbb{S}^1 = \boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!]$ and $\boldsymbol{\mathcal{S}}^{i+1}_{\mathrm{t}}[\![\tau]\!] = \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!] \frown \vec{\tau}$⟆

$= \boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!] \cup \bigcup_{j=2}^{n+1} (\boldsymbol{\mathcal{S}}^j_{\mathrm{t}}[\![\tau]\!])$ ⟅letting $j = i + 1$⟆

$= \bigcup_{j=1}^{n+1} (\boldsymbol{\mathcal{S}}^j_{\mathrm{t}}[\![\tau]\!])$ ⟅incorporating the term $j = 1$⟆ □

— Let us apply Scott iterative fixpoint Theorem 15.26.

$\vec{X}^{\infty} \triangleq \bigcup_{n \in \mathbb{N}} \vec{X}^n$ ⟅def. iterates $\vec{X}^n$ of $X \mapsto \mathbb{S}^1 \cup X \frown \vec{\tau}$ from $\varnothing$⟆

$= \bigcup_{n \in \mathbb{N}} \bigcup_{i=1}^{n} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!]$ ⟅$X^n = \bigcup_{i=1}^{n} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!]$, as shown above⟆

$= \bigcup_{n \in \mathbb{N}} \boldsymbol{\mathcal{S}}^n_{\mathrm{t}}[\![\tau]\!]$ ⟅def. $\bigcup$⟆

$= \boldsymbol{\mathcal{S}}_{\mathrm{t}}[\![\tau]\!]$ ⟅def. $\boldsymbol{\mathcal{S}}_{\mathrm{t}}[\![\tau]\!]$ in (44.54)⟆

which is $\mathsf{lfp}^{\subseteq}\, X \mapsto \mathbb{S}^1 \cup \vec{\tau} \frown X$ by Scott iterative fixpoint Theorem 15.26 knowing that $X \mapsto \mathbb{S}^1 \cup \vec{\tau} \frown X$ is preserves non-empty joins and therefore is continuous and $\langle \mathbb{S}^*, \subseteq \rangle$ is a complete lattice hence a CPO. □

Solution to exercise 44.55 This is a join homomorphic/partitioning abstraction of Exercise 11.8. □

Solution to exercise 44.60 We have $\alpha^{\mathcal{T}}(\varnothing)\langle\sigma,\ \Sigma\rangle = \mathtt{tt}$ and $\langle\sigma,\ \Sigma\rangle \mapsto \mathtt{tt}$ is the infimum for $\dot{\Leftarrow}$. Otherwise, for $\Delta \neq \varnothing$,

$$
\begin{aligned}
& \alpha^{\mathcal{T}}((\bigcup_{i\in\Delta} X_i)\langle\sigma,\ \Sigma\rangle \\
\Leftrightarrow\ & (\{\pi \in \bigcup_{i\in\Delta} X_i \mid \pi_0 = \sigma\} \subseteq \alpha^{\mathbb{T}}(\{P \in \boldsymbol{\mathcal{S}}[\![T]\!] \mid P_0 = \Sigma\})) \Leftarrow b && \text{(def. (44.62) of } \alpha^{\mathcal{T}}\text{)} \\
\Leftrightarrow\ & (\bigcup_{i\in\Delta}\{\pi \in X_i \mid \pi_0 = \sigma\} \subseteq \alpha^{\mathbb{T}}(\{P \in \boldsymbol{\mathcal{S}}[\![T]\!] \mid P_0 = \Sigma\})) \Leftarrow b && \text{(def. } \bigcup\text{)} \\
\Leftrightarrow\ & \bigwedge_{i\in\Delta}(\{\pi \in X_i \mid \pi_0 = \sigma\} \subseteq \alpha^{\mathbb{T}}(\{P \in \boldsymbol{\mathcal{S}}[\![T]\!] \mid P_0 = \Sigma\}) \Leftarrow b) && \text{(def. } \subseteq\text{)} \\
\Leftrightarrow\ & \bigwedge_{i\in\Delta} \alpha^{\mathcal{T}}(X_i)\langle\sigma,\ \Sigma\rangle && \text{(def. (44.62) of } \alpha^{\mathcal{T}}\text{)}
\end{aligned}
$$

proving $X \mapsto \alpha^{\mathcal{T}}(X)\langle\sigma,\ \Sigma\rangle$ preserves arbitrary joins in the complete lattice $\langle\mathbb{B},\ \Leftarrow,\ \mathtt{tt},\ \mathtt{ff},\ \bigwedge,\ \bigvee\rangle$, hence by Exercise 11.38, $\forall\sigma \in \mathbb{S}\ .\ \forall\Sigma \in \wp(\mathbb{S})\ .\ \langle\wp(\mathbb{S})^*,\ \subseteq\rangle \underset{X\mapsto\alpha^{\mathcal{T}}(X)\langle\sigma,\ \Sigma\rangle}{\overset{Y\mapsto\gamma^{\mathcal{T}}(Y)\langle\sigma,\ \Sigma\rangle}{\leftrightarrows}} \langle\mathbb{B},\ \Leftarrow\rangle$. The pointwise extension $\langle(\mathbb{S}\times\wp(\mathbb{S})) \to \wp(\mathbb{S})^*,\ \subseteq\rangle \underset{\alpha^{\mathcal{T}}}{\overset{\gamma^{\mathcal{T}}}{\leftrightarrows}} \langle(\mathbb{S}\times\wp(\mathbb{S})) \to \mathbb{B},\ \dot{\Leftarrow}\rangle$ follows by Exercise 11.20. □

Solution to exercise 44.63

$$
\begin{aligned}
& \alpha^{\mathcal{T}}(\mathbb{S}^1 \cup \vec{\mathtt{t}} \mathbin{\tilde{\cdot}} X)\langle\sigma,\ \Sigma\rangle \\
=\ & \alpha^{\mathcal{T}}(\mathbb{S}^1)\langle\sigma,\ \Sigma\rangle \wedge \alpha^{\mathcal{T}}(\vec{\mathtt{t}} \mathbin{\tilde{\cdot}} X)\langle\sigma,\ \Sigma\rangle && (X \mapsto \alpha^{\mathcal{T}}(X)\langle\sigma,\ \Sigma\rangle \text{ preserves joins}) \quad \text{(A)}
\end{aligned}
$$

The first term of (A) is

$$
\begin{aligned}
& \alpha^{\mathcal{T}}(\mathbb{S}^1)\langle\sigma,\ \Sigma\rangle \\
=\ & \{\pi \in \mathbb{S}^1 \mid \pi_0 = \sigma\} \subseteq \alpha^{\mathbb{T}}(\{P \in \boldsymbol{\mathcal{S}}[\![T]\!] \mid P_0 = \Sigma\}) && \text{(def. (44.62) of } \alpha^{\mathcal{T}}\text{)} \\
=\ & \{\pi \in \mathbb{S}^1 \mid \pi_0 = \sigma\} \subseteq \bigcup\{\alpha^{\mathbb{T}}(P) \mid P \in \{P \in \boldsymbol{\mathcal{S}}[\![T]\!] \mid P_0 = \Sigma\}\} && \text{(def. (44.58) of } \alpha^{\mathbb{T}}\text{)} \\
=\ & \{\pi \in \mathbb{S}^1 \mid \pi_0 = \sigma\} \subseteq \bigcup\{\alpha^{\mathbb{T}}(P) \mid P \in \boldsymbol{\mathcal{S}}[\![T]\!] \wedge P_0 = \Sigma\} && \text{(def. } \in\text{)} \\
=\ & \{\pi \in \mathbb{S}^1 \mid \pi_0 = \sigma\} \subseteq \{\pi \in \mathbb{S}^n \mid n \in \mathbb{N}^+ \wedge \exists P \in \boldsymbol{\mathcal{S}}[\![T]\!]\ .\ P_0 = \Sigma \wedge \forall i \in [0,n[\ .\ \pi_i \in P_i\} \\
& && \text{(def. (44.57) of } \alpha^{\mathbb{T}}(P) \triangleq \textstyle\bigcup_{n\in\mathbb{N}^+}\{\pi \in \mathbb{S}^n \mid \forall i \in [0,n[\ .\ \pi_i \in P_i\}\text{)} \\
=\ & \{\pi \in \mathbb{S}^1 \mid \pi_0 = \sigma\} \subseteq \{\pi \in \mathbb{S}^1 \mid \exists P \in \boldsymbol{\mathcal{S}}[\![T]\!]\ .\ P_0 = \Sigma \wedge \pi_0 \in P_0\} \\
& && (\subseteq \text{ so that the traces must have the same length}) \\
=\ & \exists P \in \boldsymbol{\mathcal{S}}[\![T]\!]\ .\ \sigma \in P_0 = \Sigma && (\Rightarrow \text{ and } \subseteq) \\
=\ & \exists P \in \{P \in \wp(\mathbb{S})^\infty \mid \forall i \in \mathbb{N}\ .\ \langle P_i,\ P_{i+1}\rangle \in T\}\ .\ \sigma \in P_0 = \Sigma && \text{(def. (44.56) of } \boldsymbol{\mathcal{S}}[\![T]\!]\text{)} \\
=\ & \sigma \in \Sigma && (T \text{ is total and } \mathbb{S} \text{ not empty})
\end{aligned}
$$

The second term of (A) is

$\alpha^{\mathcal{T}}(\vec{\tau} \frown X)\langle \sigma,\ \Sigma\rangle$

$= \alpha^{\mathcal{T}}(\{\sigma'\sigma''\pi \mid \langle \sigma',\ \sigma''\rangle \in \tau \wedge \sigma''\pi \in X\})\langle \sigma,\ \Sigma\rangle$ ⟅def. $\vec{\tau}$ and $\frown$ in Exercise 44.53⟆

$= \{\pi \in \{\sigma'\sigma''\pi \mid \langle \sigma',\ \sigma''\rangle \in \tau \wedge \sigma''\pi \in X\} \mid \pi_0 = \sigma\} \subseteq \alpha^{\mathbb{T}}(\{P \in \mathcal{S}[\![T]\!] \mid P_0 = \Sigma\})$ ⟅def. (44.62) of $\alpha^{\mathcal{T}}$⟆

$= \{\sigma'\sigma''\pi \mid \sigma' = \sigma \wedge \langle \sigma',\ \sigma''\rangle \in \tau \wedge \sigma''\pi \in X\} \subseteq \alpha^{\mathbb{T}}(\{P \in \mathcal{S}[\![T]\!] \mid P_0 = \Sigma\})$ ⟅def. $\in$⟆

$= \{\sigma\sigma''\pi \mid \langle \sigma,\ \sigma''\rangle \in \tau \wedge \sigma''\pi \in X\} \subseteq \{\pi' \in \mathbb{S}^n \mid n \in \mathbb{N}^+ \wedge \exists P \in \mathcal{S}[\![T]\!] \,.\, P_0 = \Sigma \wedge \forall i \in [0, n[\,.\, \pi'_i \in P_i\}$

⟅def. (44.57) of $\alpha^{\mathbb{T}}(P) \triangleq \bigcup_{n\in\mathbb{N}^+}\{\pi \in \mathbb{S}^n \mid \forall i \in [0, n[\,.\, \pi_i \in P_i\}$⟆

$= \bigwedge_{\langle \sigma,\ \sigma''\rangle \in \tau} \forall \sigma''\pi \in X \,.\, \exists P \in \mathcal{S}[\![T]\!] \,.\, P_0 = \Sigma \wedge \sigma \in P_0 \wedge \sigma'' \in P_1 \wedge \forall i \in [0, |\pi|[\,.\, \pi_i \in P_{i+1}$

⟅def. $\subseteq$ where $\pi' = \sigma\sigma''\pi$⟆

$= \bigwedge_{\langle \sigma,\ \sigma''\rangle \in \tau} \forall \sigma''\pi \in X \,.\, \exists \Sigma'', P' \,.\, \langle \Sigma,\ \Sigma''\rangle \in T \wedge \Sigma''P' \in \mathcal{S}[\![T]\!] \wedge \sigma \in \Sigma \wedge \sigma'' \in \Sigma'' \wedge \forall i \in [0, |\pi|[\,.\, \pi_i \in P'_i$

⟅by def. (44.56) of $\mathcal{S}[\![T]\!]$, $P \in \mathcal{S}[\![T]\!]$ if and only if $\exists \Sigma', \Sigma'', P' \,.\, \langle \Sigma',\ \Sigma''\rangle \in T \wedge \Sigma''P' \in \mathcal{S}[\![T]\!] \wedge P = \Sigma''P'$, so $\Sigma' = \Sigma$ since $P_0 = \Sigma$ and $P_{i+1} = P'_i$⟆

$= \bigwedge_{\langle \sigma,\ \sigma''\rangle \in \tau} \bigvee_{\langle \Sigma,\ \Sigma''\rangle \in T} \sigma \in \Sigma \wedge \sigma'' \in \Sigma'' \wedge \big(X \subseteq \{\sigma''\pi \mid \sigma'' \in \Sigma'' \wedge \exists P' \,.\, \Sigma''P' \in \mathcal{S}[\![T]\!] \wedge \forall i \in [0, |\pi|[\,.\, \pi_i \in P'_i\}$

⟅def. $\bigcup$ and $\subseteq$⟆

$= \bigwedge_{\langle \sigma,\ \sigma''\rangle \in \tau} \bigvee_{\langle \Sigma,\ \Sigma''\rangle \in T} \sigma \in \Sigma \wedge \sigma'' \in \Sigma'' \wedge \big(X \subseteq \{\pi' \mid (\pi'_0 = \sigma'') \Rightarrow (\pi'_0 \in \Sigma'' \wedge \exists P \,.\, \pi'_0 \in P_0 = \Sigma'' \wedge P \in \mathcal{S}[\![T]\!] \wedge \forall i \in [0, |\pi'| - 1[\,.\, \pi'_i \in P_{i+1})\}$

⟅letting $\pi' = \sigma''\pi$ and $P = \Sigma''P'$⟆

$= \bigwedge_{\langle \sigma,\ \sigma''\rangle \in \tau} \bigvee_{\langle \Sigma,\ \Sigma''\rangle \in T} \sigma \in \Sigma \wedge \sigma'' \in \Sigma'' \wedge \big(X \subseteq \{\pi' \mid (\pi'_0 = \sigma'') \Rightarrow (\exists P \in \mathcal{S}[\![T]\!] \,.\, P_0 = \Sigma'' \wedge \forall i \in [0, |\pi'| - 1[\,.\, \pi'_i \in P_{i+1})\}$

⟅including $\pi'_0 \in P_0$ in $\forall i \in [0, |\pi'| - 1[\,.\, \pi'_i \in P_{i+1}$⟆

$= \bigwedge_{\langle \sigma,\ \sigma''\rangle \in \tau} \bigvee_{\langle \Sigma,\ \Sigma''\rangle \in T} \sigma \in \Sigma \wedge \sigma'' \in \Sigma'' \wedge \alpha^{\mathcal{T}}(X)\langle \sigma'',\ \Sigma''\rangle$

since

$\alpha^{\mathcal{T}}(X)\langle \sigma'',\ \Sigma''\rangle$

$= \{\pi \in X \mid \pi_0 = \sigma''\} \subseteq \alpha^{\mathbb{T}}(\{P \in \mathcal{S}[\![T]\!] \mid P_0 = \Sigma''\}$ ⟅(44.62)⟆

$= \{\pi \in X \mid \pi_0 = \sigma''\} \subseteq \{\pi \in X \mid \pi_0 = \sigma''\} \subseteq \bigcup\{\alpha^{\mathbb{T}}(P) \mid P \in \{P \in \mathcal{S}[\![T]\!] \mid P_0 = \Sigma''\}\}$

⟅def. (44.58) of $\alpha^{\mathbb{T}}$⟆

$= \{\pi \in X \mid \pi_0 = \sigma''\} \subseteq \bigcup\{ \bigcup_{n\in\mathbb{N}^+} \{\pi \in \mathbb{S}^n \mid \forall i \in [0,n[\ .\ \pi_i \in P_i\} \mid P \in \{P \in \boldsymbol{\mathcal{S}}[\![T]\!] \mid P_0 = \Sigma''\}\}$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ ⟅def (44.57) of $\alpha^{\mathbb{T}}$⟆

$= \{\pi \in X \mid \pi_0 = \sigma''\} \subseteq \{\pi \in \mathbb{S}^* \mid \exists P \in \boldsymbol{\mathcal{S}}[\![T]\!]\ .\ P_0 = \Sigma'' \wedge \forall i \in [0,|\pi|[\ .\ \pi_i \in P_i\}$ ⟅def. ∈ and ∪⟆

$= X \subseteq \{\pi \in \mathbb{S}^* \mid (\pi_0 = \sigma'') \Rightarrow (\exists P \in \boldsymbol{\mathcal{S}}[\![T]\!]\ .\ P_0 = \Sigma'' \wedge \forall i \in [0,|\pi|[\ .\ \pi_i \in P_i)\}$ ⟅def. ⇒⟆

(44.65) follows by grouping the two terms of (A) together, renaming, and factorizing the condition $\sigma \in \Sigma$.

— We have $\alpha^{\mathcal{T}}(\langle \sigma,\ \Sigma\rangle \mapsto \varnothing) = \langle \sigma,\ \Sigma\rangle \mapsto \mathsf{ff}$ and commutation, as shown above, so by the exact fixpoint abstraction Theorem 18.21 in a complete lattice, we have

mc

$\triangleq \alpha^{\mathcal{T}}(\boldsymbol{\mathcal{S}}_{\mathsf{t}}[\![\tau]\!])$ ⟅def. (44.64) of mc⟆

$= \alpha^{\mathcal{T}}(\mathsf{lfp}^{\subseteq}\, X \mapsto \mathbb{S}^1 \cup \vec{\tau} \mathbin{\vec{\frown}} X)$ ⟅Exercise 44.53⟆

$= \mathsf{lfp}^{\dot{\Leftarrow}}\, X \mapsto \langle \sigma,\ \Sigma\rangle \mapsto \big((\sigma \in \Sigma) \wedge \bigwedge_{\langle \sigma,\sigma'\rangle\in\tau} \bigvee_{\langle \Sigma,\Sigma'\rangle\in T} X(\sigma',\Sigma')\big)$ ⟅Exercise 44.60 and Theorem 18.21⟆

$= \mathsf{gfp}^{\dot{\Rightarrow}}\, X \mapsto \langle \sigma,\ \Sigma\rangle \mapsto \big((\sigma \in \Sigma) \wedge \bigwedge_{\langle \sigma,\sigma'\rangle\in\tau} \bigvee_{\langle \Sigma,\Sigma'\rangle\in T} X(\sigma',\Sigma')\big)$ ⟅order-duality⟆ □

22 Solutions to selected exercises of chapter 47

Solution to exercise 47.10 $\mathsf{x} \not\rightsquigarrow^{\ell_1} \mathsf{y}$, $\mathsf{x} \not\rightsquigarrow^{\ell_2} \mathsf{y}$, and $\mathsf{x} \rightsquigarrow^{\ell_3} \mathsf{y}$. □

Solution to exercise 47.12 The sequence of values of x at ℓ is 0, 1, 2, …17, … 42, … for P_0 (so x does not depend on its initial value x_0 at ℓ) and x_0, $x_0 + 1$, $x_0 + 2$,…$x_0 + 17$…$x_0 + 42$ …for P_u (so x depends on its initial value x_0 at ℓ since changing x_0 will change the first element of this sequence). □

Solution to exercise 47.13 The sequence of values y at ℓ_0 is a finite prefix of the infinite sequence y_0, $y_0 + 1$, $y_0 + 2$, … which length depends on the initial value x_0 of x. This is a timing channel and there is no dependency, even if y and x_0 are related at ℓ_0. □

Solution to exercise 47.14 If the initial value x_0 of x at ℓ_0 is positive then the infinite sequence of values of y at ℓ_5 is $1 \cdot 2 \cdot 3 \cdot \ldots$ while it is $1 \cdot 1 \cdot 2 \cdot 2 \cdot 3 \cdot 3 \cdot \ldots$ when the initial value x_0 of x at

ℓ_0 is strictly negative. They have a common prefix but differ at position 2 so y depends upon the initial value of x at ℓ_5.

The situation is different at ℓ_4, since in both cases the sequence of values of y is $0 \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdot \ldots$ so y does not depend upon the initial value of x at ℓ_4.

With the iteration condition `i < 5`, the sequence of values taken by y at ℓ_4 is $0 \cdot 1 \cdot 2 \cdot 3 \cdot 4$ when the initial value x_0 of x at ℓ_0 is positive whereas it is $0 \cdot 1 \cdot 2$ when x_0 is strictly negative. These sequences do not involve differences on values stored in variable y but differences on their lengths linked to the rate of termination. There is a timing channel but not a dependency. □

Solution to exercise 47.17 Let the future of a semantic property $\mathcal{P} \in \wp(\wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty}))$ be $\alpha^\ell_{\mathrm{f}}[\![\mathsf{y}]\!](\mathcal{P}) \triangleq \{\mathsf{seqval}[\![\mathsf{y}]\!]^\ell(\pi_0, \pi) \mid \langle \pi_0, \pi\rangle \in \mathcal{P}\}$. Then $\langle \wp(\wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty})), \subseteq\rangle \underset{\alpha^\ell_{\mathrm{f}}[\![\mathsf{y}]\!]}{\overset{\gamma^\ell_{\mathrm{f}}[\![\mathsf{y}]\!]}{\leftrightarrows}} \langle \wp(\mathbb{V}^{+\infty}), \subseteq\rangle$. □

Solution to exercise 47.20 There exist initialization traces π_0 and π'_0 with same initial values of variables but for x (*i.e.* $\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}\} \,.\, \varrho(\pi_0)\mathsf{z} = \varrho(\pi'_0)\mathsf{z} \wedge \varrho(\pi_0)\mathsf{x} \neq \varrho(\pi'_0)\mathsf{x}$) and continuations π_1 and π'_1 (*i.e.* $\langle \pi_0, \pi_1\rangle, \langle \pi'_0, \pi'_1\rangle \in \Pi$) sharing a common prefix (*i.e.* ω_0 in (47.18)), *where all non-deterministic choices are identical*, but have different values ($v \neq v'$ in (47.18)) of y at ℓ after that common prefix (*i.e.* ω_0). For example, $\mathsf{x} \rightsquigarrow \mathsf{z}$ in `input y; z = y + x ;`. □

Solution to exercise 47.31 In S, x = y = 1 at ℓ_2 so x and y depend on no other variable. For S′ changing the initial value of x *e.g.* from 2 to 3 will change the value of x and y at ℓ_2 so both depend upon the initial value of x. □

Solution to exercise 47.43

Proof of (47.42)

$$
\begin{aligned}
&\alpha^{\mathrm{d}}(\{\mathcal{S}^{+\infty}[\![\mathsf{S}]\!]\})\,(\ell)\\
&= \alpha^{\mathrm{d}}(\{\mathcal{S}^{*}[\![\mathsf{S}]\!]\})\,\ell && \text{(Lemma 47.23)}\\
&= \{\langle \mathsf{x}', \mathsf{y}\rangle \mid \mathcal{S}^{*}[\![\mathsf{S}]\!] \in \mathcal{D}_{\mathsf{diff}}(\ell)\langle \mathsf{x}', \mathsf{y}\rangle\} && \text{(def. (47.25) of } \alpha^{\mathrm{d}}\text{)}\\
&= \{\langle \mathsf{x}', \mathsf{y}\rangle \mid \exists \langle \pi_0, \pi_1\rangle, \langle \pi'_0, \pi'_1\rangle \in \mathcal{S}^{*}[\![\mathsf{S}]\!] \,.\, (\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \varrho(\pi_0)\mathsf{z} = \varrho(\pi'_0)\mathsf{z}) \wedge\\
&\quad \mathsf{diff}(\mathsf{seqval}[\![\mathsf{y}]\!](\ell)(\pi_0, \pi_1), \mathsf{seqval}[\![\mathsf{y}]\!](\ell)(\pi'_0, \pi'_1))\} && \text{(def. (47.19) of } \mathcal{D}_{\mathsf{diff}}(\ell)\langle \mathsf{x}', \mathsf{y}\rangle\text{)}\\
&= \{\langle \mathsf{x}', \mathsf{y}\rangle \mid \exists \langle \pi_0, \pi_1\rangle, \langle \pi'_0, \pi'_1\rangle \in \mathcal{S}^{*}[\![\mathsf{S}]\!] \,.\, (\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \varrho(\pi_0)\mathsf{z} = \varrho(\pi'_0)\mathsf{z}) \wedge \mathsf{diff}(\mathfrak{d}, \mathfrak{d})\}\\
&\qquad \text{(def. } \mathcal{S}^{*}[\![\mathsf{S}]\!] \text{ so that if } \langle \pi, \pi'\rangle \in \mathcal{S}^{*}[\![\mathsf{S}]\!] \text{ then } \pi \text{ ends } \mathsf{at}[\![\mathsf{S}]\!] \text{ and } \pi' \text{ contains only labels of}\\
&\qquad \mathsf{labx}[\![\mathsf{S}]\!] \text{ so that, def. (47.16) of } \mathsf{seqval}[\![\mathsf{y}]\!], \mathsf{seqval}[\![\mathsf{y}]\!](\ell)(\pi_0, \pi_1) = \mathsf{seqval}[\![\mathsf{y}]\!](\ell)(\pi'_0, \pi'_1) = \mathfrak{d}\text{)}\\
&= \varnothing && \text{(def. (47.18) of } \mathsf{diff}(\omega, \omega') \text{ which implies } \omega \neq \mathfrak{d} \text{ and } \omega' \neq \mathfrak{d}\text{)}
\end{aligned}
$$
□

Solution to exercise 47.46 We can define $\widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\mathtt{1}]\!] \triangleq \varnothing$, $\widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\mathtt{x}]\!] \triangleq \{\mathtt{x}\}$, and $\widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\mathtt{A}_1 - \mathtt{A}_2]\!] \triangleq \{\mathtt{y} \in \mathbb{vars}[\![\mathtt{A}_1]\!] \cup \mathbb{vars}[\![\mathtt{A}_2]\!] \mid \mathtt{A}_1 \neq \mathtt{A}_2\}$. This handles the case $\widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\mathtt{x - x}]\!] = \varnothing$ while $\mathbb{vars}[\![\mathtt{x - x}]\!] = \{\mathtt{x}\}$. Even more precision can be achieved by considering reachable environments only (see Remark 47.39). For example, using a constant propagation, an interval, or a zone/octagon analysis, $\mathtt{y} \in \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\mathtt{A}_1 - \mathtt{A}_2]\!]$ only if this analysis cannot prove that $\mathtt{A}_1 - \mathtt{A}_2$ is constant. This can be implemented by a reduced product. □

Solution to exercise 47.61 $\widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\mathtt{Sl}]\!]\, \ell_2 = \{\langle \mathtt{x}, \mathtt{y}\rangle\} \cup \{\langle \mathtt{z}, \mathtt{z}\rangle \mid \mathtt{z} \in \mathbb{V} \setminus \{\mathtt{y}\}\}$. This proves that $\mathtt{y}$ at ℓ_2 does not depend on its initial value at ℓ_0 but not that $\mathtt{y}$ at ℓ_2 does not depend on $\mathtt{x}$ at ℓ_0 (which would require to take values of variables into account *e.g.* by a linear equality analysis of Chapter 38). □

Solution to exercise 47.66

$$
\begin{array}{ll}
& \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\mathtt{S}_b]\!]\, \ell_0 \\
= & \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\{\ell_1\ \mathtt{y = z\ ;} \ell_2\ \mathtt{z = x\ ;}\}]\!]\, \ell_0 & \text{(def. } \mathtt{S}_b\text{)} \\
= & \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ell_1\ \mathtt{y = z\ ;} \ell_2\ \mathtt{z = x\ ;}]\!]\, \ell_0 & \text{(compound statement (47.57))} \\
= & \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ell_1\ \mathtt{y = z\ ;}]\!]\, \ell_2 \fatsemi \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ell_2\ \mathtt{z = x\ ;}]\!]\, \ell_0 & \text{((47.60.b) where } \mathtt{Sl}' = \ell_1\ \mathtt{y = z\ ;}\text{)} \\
= & \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ \epsilon\ ]\!]\, \ell_1 \fatsemi \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ell_1\ \mathtt{y = z\ ;}]\!]\, \ell_2 \fatsemi \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ell_2\ \mathtt{z = x\ ;}]\!]\, \ell_0 & \text{((47.60.b) where } \mathtt{Sl}' = \epsilon\ \ell_1\text{)} \\
= & \mathbb{1}_{\mathbb{V}} \fatsemi \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ell_1\ \mathtt{y = z\ ;}]\!]\, \ell_2 \fatsemi \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ell_2\ \mathtt{z = x\ ;}]\!]\, \ell_0 & \text{((47.54))} \\
= & \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ell_1\ \mathtt{y = z\ ;}]\!]\, \ell_2 \fatsemi \widehat{\widehat{\mathcal{S}}}^{\exists}_{\text{diff}}[\![\ell_2\ \mathtt{z = x\ ;}]\!]\, \ell_0 & (\mathbb{1}_{\mathbb{V}} \text{ neutral element of } \fatsemi) \\
= & \{\langle \mathtt{z}, \mathtt{y}\rangle, \langle \mathtt{z}, \mathtt{z}\rangle, \langle \mathtt{x}, \mathtt{x}\rangle\} \fatsemi \{\langle \mathtt{x}, \mathtt{z}\rangle, \langle \mathtt{x}, \mathtt{x}\rangle, \langle \mathtt{y}, \mathtt{y}\rangle\} & \text{((47.44))} \\
= & \{\langle \mathtt{x}, \mathtt{x}\rangle, \langle \mathtt{x}, \mathtt{z}\rangle, \langle \mathtt{z}, \mathtt{y}\rangle\} & \text{(def. } \fatsemi\text{)}\ \square
\end{array}
$$

23 Solutions to selected exercises of chapter 48

Solution to exercise 48.60 — The proof is by structural induction on $\boldsymbol{\tau}'$.

- If $\boldsymbol{\tau}' = \alpha \in \mathbb{V}_{\mathbb{t}}$ then $\{\langle \alpha, \boldsymbol{\tau}\rangle\}(\boldsymbol{\tau}') = \{\langle \alpha, \boldsymbol{\tau}\rangle\}(\alpha) = \boldsymbol{\tau}$ by def. function application. On the other hand, $\boldsymbol{\tau}[\alpha \leftarrow \boldsymbol{\tau}'] = \boldsymbol{\tau}[\alpha \leftarrow \alpha] = \boldsymbol{\tau}$ by (48.5);
- If $\alpha \neq \boldsymbol{\tau}' = \beta \in \mathbb{V}_{\mathbb{t}}$ then $\{\langle \alpha, \boldsymbol{\tau}\rangle\}(\boldsymbol{\tau}') = \{\langle \alpha, \boldsymbol{\tau}\rangle\}(\beta) = \beta$ by (48.30) and $\alpha \notin \mathsf{dom}(\{\langle \alpha, \boldsymbol{\tau}\rangle\}) = \{\alpha\}$. This is equal to $\boldsymbol{\tau}'[\alpha \leftarrow \boldsymbol{\tau}] = \beta[\alpha \leftarrow \boldsymbol{\tau}] = \beta$, by (48.5);
- Otherwise, $\boldsymbol{\tau}' = f(\boldsymbol{\tau}'_1, \ldots, \boldsymbol{\tau}'_n)$ so that, by (48.30), ind. hyp., and (48.5), we have $\{\langle \alpha, \boldsymbol{\tau}\rangle\}(\boldsymbol{\tau}') = \{\langle \alpha, \boldsymbol{\tau}\rangle\}(f(\boldsymbol{\tau}'_1, \ldots, \boldsymbol{\tau}'_n)) = f(\{\langle \alpha, \boldsymbol{\tau}\rangle\}(\boldsymbol{\tau}'_1), \ldots, \{\langle \alpha, \boldsymbol{\tau}\rangle\}(\boldsymbol{\tau}'_n)) = f(\boldsymbol{\tau}'_1[\alpha \leftarrow \boldsymbol{\tau}], \ldots, \boldsymbol{\tau}'_n[\alpha \leftarrow \boldsymbol{\tau}]) = f(\boldsymbol{\tau}'_1, \ldots, \boldsymbol{\tau}'_n)[\alpha \leftarrow \boldsymbol{\tau}] = \boldsymbol{\tau}'[\alpha \leftarrow \boldsymbol{\tau}]$. □

24 Solutions to selected exercises of chapter 49

Solution to exercise 49.2

$$
\begin{array}{rcll}
\mathbb{U}^0 & \triangleq & \mathbb{B} \cup \mathbb{Z} \cup \{\texttt{nil}\} & \text{booleans, integers, and null list} \\
\mathbb{U}^{n+1} & \triangleq & \mathbb{U}^n & \\
 & & \cup\, \{\langle x,\ y\rangle \mid x, y \in \mathbb{U}^n\} & \text{pairs} \\
 & & \cup\, \{x \texttt{ :: } y \mid x, y \in \mathbb{U}^n\} & \text{lists} \\
\mathbb{U} & \triangleq & \bigcup_{n\in\mathbb{N}} \mathbb{U}^n \cup \{\Omega^\sigma, \Omega^\delta\} & \text{values} \quad \square
\end{array}
$$

Solution to exercise 49.6

```
(* syntax of dynamic types *)

type dtype =
    Dbool
  | Dint
  | Dnil
  | Dpair of dtype * dtype
  | Dlist of dtype
  | Derr

(* equivalent up to Nil for lists *)

let rec equivalent dt1 dt2 =
  match dt1, dt2 with
  | Dlist dt, Dlist dt' ->
      equivalent dt dt'
  | Dpair (dt1, dt2), Dpair (dt3, dt4) ->
      (equivalent dt1 dt3) && (equivalent dt2 dt4)
  | Dlist dt, Dnil -> true
  | Dnil, Dlist dt -> true
  | _, _ -> dt1 = dt2

(* values *)

type value =
        Vbool of bool
  | Vint of int
  | Vnil
  | Vpair of value * value
  | Vlist of value * value
  | Vderr
  | Vserr

(* dynamic type of values *)
```

```
let rec dtypeof v =
  match v with
  | Vbool b -> Dbool
  | Vint i  -> Dint
  | Vnil -> Dnil
  | Vpair (v1,v2) ->
      let dt1 = dtypeof(v1) and dt2 = dtypeof(v2) in
        if (dt1 = Derr) || (dt2 = Derr) then Derr
        else Dpair (dt1, dt2)
  | Vlist (h,t) ->
      (match dtypeof h, dtypeof t with
       | Derr, Derr -> Derr
       | dh, Dnil -> Dlist dh
       | dh, Dlist dt ->
           if (equivalent dh dt) then Dlist dh
           else Derr
       | _, _ -> Derr)
  | Vderr -> Derr
  | Vserr -> Derr

# dtypeof (Vlist (Vnil, Vnil));;
- : dtype = Dlist Dnil
# dtypeof (Vlist (Vpair (Vint 1, Vlist (Vint 1, Vnil)), Vnil));;
- : dtype = Dlist (Dpair (Dint, Dlist Dint))
```

□

Solution to exercise 49.10

```
(* syntax of expressions *)

type program_variable = string
type expression =
    One
  | Var of program_variable
  | Minus of expression * expression
  | Nil
  | Pair of expression * expression
  | Cons of expression * expression
  | Hd of expression
  | Tl of expression
  | Less of expression * expression
  | Isnil of expression
  | Nand of expression * expression

(* environments *)

type environment = (program_variable * value) list

let rec valueof r x =
   match r with
     [] -> Vserr
```

```
   | (y, v) :: t ->
         if (y = x) then v
         else valueof t x

(* evaluation of expressions *)

let rec eval e r =
  match e with
    One ->  Vint 1
  | Var x -> valueof r x
  | Minus (e1, e2) ->
        (match (eval e1 r, eval e2 r) with
         | Vserr, _ -> Vserr
         | _, Vserr -> Vserr
         | _, Vderr -> Vderr
         | Vderr, _ -> Vderr
         | Vint i1, Vint i2 -> Vint (i1 - i2)
         | _, _ -> Vserr)
  | Nil -> Vnil
  | Pair (e1, e2) ->
        (match (eval e1 r, eval e2 r) with
         | Vserr, _ -> Vserr
         | _, Vserr -> Vserr
         | _, Vderr -> Vderr
         | Vderr, _ -> Vderr
         | v1, v2 -> Vpair (v1, v2))
  | Cons (e1, e2) ->
        (match (eval e1 r, eval e2 r) with
         | Vserr, _ -> Vserr
         | _, Vserr -> Vserr
         | _, Vderr -> Vderr
         | Vderr, _ -> Vderr
         | v1, v2 ->
             let l = Vlist (v1, v2) in
               if (dtypeof l) <> Derr then l
               else Vserr)
  | Hd e1 -> let v1 = eval e1 r in
               (match dtypeof v1 with
                | Dlist dh ->
                     (match v1 with
                      | Vnil -> Vderr
                      | Vlist (h,t) -> h
                      | _ -> Vserr)
                | _ -> Vserr)
 | Tl e1 -> let v1 = eval e1 r in
               (match dtypeof v1 with
                | Dlist dh ->
                     (match v1 with
                      | Vnil -> Vderr
                      | Vlist (h,t) -> t
                      | _ -> Vserr)
                | _ -> Vserr)
```

```
  | Less (e1, e2) ->
        (match (eval e1 r, eval e2 r) with
         | Vserr, _ -> Vserr
         | _, Vserr -> Vserr
         | _, Vderr -> Vderr
         | Vderr, _ -> Vderr
         | Vint i1, Vint i2 -> Vbool (i1 < i2)
         | _, _ -> Vserr)
  | Isnil e1 ->
        (match (eval e1 r) with
         | Vserr -> Vserr
         | Vderr -> Vderr
         | Vnil  -> (Vbool true)
         | v1  -> (match dtypeof v1 with
                   | Dlist dh -> (Vbool false)
                   | _ -> Vserr))
  | Nand (e1, e2) ->
        (match (eval e1 r, eval e2 r) with
         | Vserr, _ -> Vserr
         | _, Vserr -> Vserr
         | _, Vderr -> Vderr
         | Vderr, _ -> Vderr
         | Vbool i1, Vbool i2 -> Vbool (not (i1 && i2))
         | _, _ -> Vserr)

# eval (Cons ((Pair (One, (Cons (One, Nil)))), Nil)) [];;
- : value = Vlist (Vpair (Vint 1, Vlist (Vint 1, Vnil)), Vnil)
# eval (Cons (Nil, (Var "x"))) [("x", (Vint 1))];;
- : value = Vserr
```

□

Solution to exercise 49.9

$$\mathscr{A}[\![\mathsf{x}]\!]\rho \quad \triangleq \quad \mathsf{let}\, v = \rho(\mathsf{x})\ \mathsf{in}\ (\!|\, \tau^{\delta}(v) = \mathit{err}\ ?\ \Omega^{\delta}\ :\ v \,|\!)$$

This is a dynamic error since initial values or inputs must be checked at runtime. □

Solution to exercise 49.35

```
type monotype =
    Mbool
  | Mint
  | Mpair of monotype * monotype
  | Mlist of monotype

(* type environment mapping program variables to monotypes *)

type menvironment = (program_variable * monotype) list
```

```
(* check g|-e:m i.e. in type environment g, expression e has monotype m *)

let rec mcheck g e m =
  match e with
  | One -> m = Mint
  | Var x ->
      (try (List.assoc x g) = m
       with Not_found -> false)
  | Minus (e1, e2) ->
      (mcheck g e1 Mint) && (mcheck g e2 Mint) && (m = Mint)
  | Nil ->
      (match m with
       | Mlist _ -> true
       | _ -> false)
  | Pair (e1, e2) ->
      (match m with
       | Mpair (m1, m2) -> (mcheck g e1 m1) && (mcheck g e2 m2)
       | _ -> false)
  | Cons (e1, e2) ->
      (match m with
       | Mlist m' -> (mcheck g e1 m') && (mcheck g e2 m)
       | _ -> false)
  | Hd e1 ->
      (match m with
       | Mlist m' -> (mcheck g e1 m')
       | _ -> false)
  | Tl e1 ->
      (match m with
       | Mlist m' -> (mcheck g e1 m)
       | _ -> false)
  | Less (e1, e2) ->
      (mcheck g e1 Mint) && (mcheck g e2 Mint) && (m = Mbool)
  | Isnil e1 ->
      (match m with
       | Mlist m' -> true
       | _ -> false)
  | Nand (e1, e2) ->
      (mcheck g e1 Mbool) && (mcheck g e2 Mbool) && (m = Mbool)

# mcheck [("x", Mint)] (Cons ((Var "x"), Nil)) (Mlist Mint) ;;
- : bool = true
```

□

Solution to exercise 49.42

```
(* monotypes with variables *)

type type_variable = string
type monotypevar =
  | Tvar of type_variable
  | Tbool
```

```
  | Tint
  | Tpair of monotypevar * monotypevar
  | Tlist of monotypevar

(* occurrence of a variable in a type with variables *)

let rec occurrence alpha tv =
  match tv with
  | Tvar beta -> alpha = beta
  | Tbool -> false
  | Tint -> false
  | Tpair (tv1, tv2) -> occurrence alpha tv1 || occurrence alpha tv2
  | Tlist tv1 -> occurrence alpha tv1

(* Substitutions *)

type substitution = (type_variable * monotypevar) list

let identity : substitution = []

(* application of a substitution to a monotype with variables *)

let rec apply (s:substitution) (tv:monotypevar) =
  match tv with
  | Tvar alpha -> (try List.assoc alpha s
                   with Not_found -> Tvar alpha)
  | Tbool -> tv
  | Tint -> tv
  | Tpair (tv1, tv2) -> Tpair (apply s tv1, apply s tv2)
  | Tlist tv1 -> Tlist (apply s tv1)

(* composition of substitutions *)

let rec domain (s:substitution) =
  match s with
    [] -> []
  | (x, tv) :: tl -> x :: domain tl

let rec union l1 l2 =
  match l1 with
    [] -> l2
  | hd :: tl -> union tl (if List.mem hd l2 then l2 else hd :: l2)

let rec apply_s2_to (s1:substitution) (s2:substitution) : substitution =
   match s1 with
   | [] -> []
   | (a, tv) :: s1' ->
        if (apply s2 tv) = (Tvar a)
        then apply_s2_to s1' s2
        else (a, (apply s2 tv)) :: (apply_s2_to s1' s2)

let rec remove (s2:substitution) d : substitution =
```

```
   match s2 with
   | [] -> []
   | (a, tv) :: s2' ->
        if List.mem a d
        then remove s2' d
        else (a, tv) :: (remove s2' d)

let compose (s1:substitution) (s2:substitution) : substitution =
   List.append (apply_s2_to s1 s2)  (remove s2 (domain s1))

(* systems of equations *)

type equations = (monotypevar * monotypevar) list

(* is alpha free in tv? *)

let rec free_in alpha (tv:monotypevar) =
   match tv with
  | Tvar beta -> (alpha = beta)
  | Tbool -> false
  | Tint -> false
  | Tpair (tv1, tv2) -> (free_in alpha tv1) || (free_in alpha tv2)
  | Tlist tv1 -> free_in alpha tv1

(* is alpha in the range of the equations eqns? *)

let rec in_range alpha eqns =
  match eqns with
  | [] -> false
  | (tv, tv') :: eqns' -> (free_in alpha tv') || (in_range alpha eqns')

(* is tv not a type variable? *)

let not_var tv =
  match tv with
  | (Tvar x) -> false
  | _ -> true

(* apply a substitution to a system of equations *)

let apply_subst_to_eqns (s:substitution) (eqns:equations) : equations =
  List.map (fun (tv1, tv2) -> (apply s tv1, apply s tv2)) eqns

exception NotTypable

(* apply the transformation rule first in eqnsR to equations {eqnsL, eqnsR} *)
(* and report a change if any.                                              *)

let rec apply_rule change (eqnsL:equations) (eqnsR:equations) : bool * equations * equations =
  match eqnsR with
    [] -> (change, eqnsL, [])
  | (tv, tv') :: eqnsR' when (tv = tv') -> (true, eqnsL, eqnsR')
```

```
  | (Tvar alpha, tv) :: eqnsR' when (occurrence alpha tv) -> raise NotTypable
  | (Tvar alpha, tv) :: eqnsR' when (in_range alpha eqnsL) || (in_range alpha eqnsR') ->
    (true, (List.append (apply_subst_to_eqns [(alpha, tv)] eqnsL) [(Tvar alpha, tv)]), (apply_subst_to_eqns [(alpha, tv)]
  | (tv, Tvar beta) :: eqnsR' when (not_var tv) -> (true, eqnsL, ((Tvar beta, tv) :: eqnsR'))
  | (Tbool, Tbool) :: eqnsR' -> (true, eqnsL, eqnsR')
  | (Tbool, tv) :: eqnsR' when (not_var tv) -> raise NotTypable
  | (tv, Tbool) :: eqnsR' when (not_var tv)  -> raise NotTypable
  | (Tint, Tint) :: eqnsR' -> (true, eqnsL, eqnsR')
  | (Tint, tv) :: eqnsR' when (not_var tv)  -> raise NotTypable
  | (tv, Tint) :: eqnsR' when (not_var tv)  -> raise NotTypable
  | (Tpair (tv1, tv2), Tpair (tv1', tv2')) :: eqnsR' ->
      (true, eqnsL, ((tv1, tv1') :: (tv2, tv2') :: eqnsR'))
  | (Tpair (tv1, tv2), tv) :: eqnsR' when (not_var tv)  -> raise NotTypable
  | (tv, Tpair (tv1', tv2')) :: eqnsR' when (not_var tv)  -> raise NotTypable
  | (Tlist tv1, Tlist tv1') :: eqnsR' ->
      (true, eqnsL, ((tv1, tv1') :: eqnsR'))
  | (Tlist tv1, tv) :: eqnsR' when (not_var tv)  -> raise NotTypable
  | (tv, Tlist tv1') :: eqnsR' when (not_var tv)  -> raise NotTypable
  | (tv, tv') :: eqnsR' -> apply_rule change (List.append eqnsL [(tv, tv')]) eqnsR'

(* transform solved equations into a substitution *)

let rec subst_of_eqns (eqns:equations) : substitution =
   match eqns with
   | [] -> []
   | ((Tvar x),tv)::eqns' -> (x,tv)::(subst_of_eqns eqns')
   | _ -> failwith "equations not solved"

(* most general unifier: apply the rule to the equations until no change *)

let rec mgu (eqns:equations) =
  let (change, eqnsL, eqnsR) = (apply_rule false [] eqns) in
    if change then (mgu (List.append eqnsL eqnsR))
    else (subst_of_eqns eqnsL)

# mgu [(Tvar "a", Tvar "b"); (Tvar "b", Tvar "c"); (Tvar "c", Tvar "a")];;
- : substitution = [("a", Tvar "c"); ("b", Tvar "c")]
# mgu [(Tlist (Tpair (Tint, Tvar "a")), (Tlist (Tvar "a")))];;
Exception: NotTypable.
```

□

Solution to exercise 49.43

```
(* type environment *)

type type_env = (program_variable * monotypevar) list

(* apply a substitution to a type environment *)

let apply_env (s:substitution) (env:type_env) : type_env =
  List.map (fun (x, tv) -> (x, apply s tv)) env
```

```
(* merge environments with different variables *)

let rec merge (env1:type_env) (env2:type_env) : type_env =
   match env1 with
   | [] -> env2
   | (v, tv) :: env1' ->
       if List.mem_assoc v env2
       then (v, tv) :: (List.remove_assoc v env2)
       else (v, tv) :: (merge env1' env2)

(* most general unifier of type environments *)

let rec mgu_env (env1:type_env) (env2:type_env) : substitution =
   match env1 with
   | [] -> identity
   | (v, tv) :: env1' ->
       try let tv' = List.assoc v env2 in
         let s =  mgu [(tv, tv')] in
           compose (mgu_env env1' (List.remove_assoc v env2)) s
       with Not_found -> (mgu_env env1' env2)

(* fresh variables *)
let next_var = ref 0

let fresh () =
  incr next_var;
  (Tvar ("a" ^ string_of_int !next_var))

let rec infer e =
  match e with
  | One -> ([], Tint)
  | Var x ->  let a = fresh () in ([(x, a)], a)
  | Minus (e1, e2) ->
      let (env1, tv1) = infer e1 and (env2, tv2) = infer e2 in
        let s = (compose (mgu_env env1 env2) (mgu [(tv1,tv2); (tv2,Tint)])) in
          (apply_env s (merge env1 env2), Tint)
  | Nil -> let a = fresh () in ([], Tlist a)
  | Pair (e1, e2) ->
      let (env1, tv1) = infer e1 and (env2, tv2) = infer e2 in
        let a = fresh () and b = fresh () in
           let s = (compose (mgu_env env1 env2) (mgu [((Tpair (tv1,b)),(Tpair (a,tv2)))])) in
              (apply_env s (merge env1 env2), (Tpair (apply s tv1, apply s tv2)))
  | Cons (e1, e2) ->
      let (env1, tv1) = infer e1 and (env2, tv2) = infer e2 in
         let s = (compose (mgu_env env1 env2) (mgu [(Tlist tv1, tv2)])) in
           (apply_env s (merge env1 env2), Tlist (apply s tv1))
  | Hd e1 ->
      let (env1, tv1) = infer e1 in
         let a = fresh () in
            let s = mgu [(tv1,Tlist a)] in
              (apply_env s env1, apply s a)
```

```
  | Tl e1 ->
      let (env1, tv1) = infer e1 in
         let a = fresh () in
            let s = mgu [(tv1,Tlist a)] in
              (apply_env s env1, apply s tv1)
  | Less (e1, e2) ->
      let (env1, tv1) = infer e1 and (env2, tv2) = infer e2 in
        let s = (compose (mgu_env env1 env2) (mgu [(tv1,tv2); (tv2,Tint)])) in
          (apply_env s (merge env1 env2), Tbool)
  | Isnil e1 ->
      let (env1, tv1) = infer e1 in
         let a = fresh () in
            let s = mgu [(tv1,Tlist a)] in
              (apply_env s env1, Tbool)
  | Nand  (e1, e2) ->
      let (env1, tv1) = infer e1 and (env2, tv2) = infer e2 in
        let s = (compose (mgu_env env1 env2) (mgu [(tv1,tv2); (tv2,Tbool)])) in
          (apply_env s (merge env1 env2), Tbool)

# infer (Cons ((Var "x"),(Var "y")));;
- : type_env * monotypevar =
([("x", Tvar "a1"); ("y", Tlist (Tvar "a1"))], Tlist (Tvar "a1"))
# infer (Isnil (Var "x"));;
- : type_env * monotypevar = ([("x", Tlist (Tvar "a4"))], Tbool)
```

□

25 Solutions to selected exercises of chapter 50

Solution to exercise 50.34

```
l1: {n:_|_} n = (0 - 10);
while l2: (n != 0) {n:[0, oo]}
   l3: {n:[1, oo]} n = (n - 1);
l4: {n:[0, oo]}
```

The specification only requires the program to terminate without any constraint on the final value of n. This is only possible if $n \geqslant 0$ within and on entrance of the loop. But n is strictly negative at ℓ_2. So the program will never terminate. This is shown by $\bot$ at ℓ_1 *i.e.* the only way for the program to terminate is to never execute it! □

26 Solutions to selected exercises of chapter 51

Solution to exercise 51.6

```
*** Labelled program:
```

```
l1: x = y;
l2:
*** programspec:
l2:{ x:[42,42] }
*** Result of the backward-forward extremal static analysis:
    <{x:T; y:[42, 42]},
     [l1: {x:_|_; y:_|_}; l2: {x:[42, 42]; y:[42, 42]}]>
*** Result of the forward-backward extremal static analysis:
    <{x:T; y:[42, 42]},
     [l1: {x:_|_; y:_|_}; l2: {x:[42, 42]; y:T}]>
```
□

Solution to exercise 51.13 The backward-forward static analysis with iterated extremal reduction of Section 51.3 starts with a backward analysis from the specification `l4: { x:[0,20]; y:[10, 30]}` and yields

```
l1: {x:[-oo+10, oo-10]; y:[-oo+10, oo-10]} x = (x - x);
l2: {x:[10, 20]; y:[-oo+10, oo-10]} y = (y - y);
if l3: (x == y){x:[10, 20]; y:[10, 20]}
   l4: {x:[0, 20]; y:[10, 30]} ;
l5: {x:_|_; y:_|_}
```

which is imprecise. Then the forward analysis from `r0 = {x:[-oo+10, oo-10]; y:[-oo+10, oo-10]}` returns

```
l1: {x:[-oo+10, oo-10]; y:[-oo+10, oo-10]} x = (x - x);
l2: {x:T; y:[-oo+10, oo-10]} y = (y - y);
if l3: (x == y){x:T; y:T}
   l4: {x:T; y:T} ;
```

The next backward analysis shows that the extremal reduction has converged and the intersection with the specifications yields (51.11).

The backward-forward static analysis with iterated intermediate reduction starts with the same backward analysis. However, the next forward analysis is

```
l1: {x:[-oo+10, oo-10]; y:[-oo+10, oo-10]} x = (x - x);
l2: {x:[10, 20]; y:[-oo+10, oo-10]} y = (y - y);
if l3: (x == y){x:[10, 20]; y:[10, 20]}
   l4: {x:[10, 20]; y:[10, 20]} ;
l5: {x:_|_; y:_|_}
```

because of the intersection with the backward analysis at `l2` (so that we get `{x:[10, 20]; y:[10, 20]}` instead of `{x:T; y:T}`). The next backward analysis shows that the intermediate reduction has converged and the intersection with the specifications yields (51.12). □