Solutions to Selected Exercises
in Complement of the Book

Principles of Abstract Interpretation

MIT Press

Patrick Cousot

New York University

January 5, 2021

Contents

1 Solutions to Selected Exercises of Chapter 4

Solution to Exercise 4.3

```
$ cat iterate1.c
#include <stdio.h>
#define tt 1
int main () {
   int x = 0;
   x = x + 1;
   while (tt) {
     x = x + 1;
     if (x > 2) break;
   } ;
   printf("x = %d\n", x);
}
```

```
$ gcc iterate1.c
$ ./a.out
x = 3
$
```

□

Solution to Exercise 4.4 An `if (B)` S_t `else` S_f can be replaced with $\big(($`B`$) \uparrow ($`B`$)$ is $\neg($`B`$)\big)$

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

□

Solution to Exercise 4.7

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

Hence $\mathsf{after}[\![\mathtt{S}]\!]$ is the label at which execution goes on when `S` terminates without a **`break ;`**. □

Solution to Exercise 4.12

$$\begin{array}{l}
\mathsf{break\text{-}to}[\![\mathtt{P}_{15}]\!] = \mathsf{break\text{-}to}[\![\mathtt{Sl}_{14}]\!] = \ell_7 \\
\mathsf{break\text{-}to}[\![\mathtt{Sl}_{12}]\!] = \mathsf{break\text{-}to}[\![\mathtt{S}_{13}]\!] = \mathsf{break\text{-}to}[\![\mathtt{Sl}_{14}]\!] = \ell_7 \\
\mathsf{break\text{-}to}[\![\mathtt{Sl}_{10}]\!] = \mathsf{break\text{-}to}[\![\mathtt{S}_{11}]\!] = \mathsf{break\text{-}to}[\![\mathtt{Sl}_{12}]\!] = \ell_7 \\
\mathsf{break\text{-}to}[\![\mathtt{Sl}_7]\!] = \mathsf{break\text{-}to}[\![\mathtt{S}_8]\!] = \mathsf{break\text{-}to}[\![\mathtt{Sl}_{10}]\!] = \ell_7 \\
\mathsf{break\text{-}to}[\![\mathtt{S}_9]\!] = \mathsf{after}[\![\mathtt{S}_{11}]\!] = \ell_6 \\
\mathsf{break\text{-}to}[\![\mathtt{Sl}_6]\!] = \mathsf{break\text{-}to}[\![\mathtt{S}_9]\!] = \ell_6 \\
\mathsf{break\text{-}to}[\![\mathtt{Sl}_4]\!] = \mathsf{break\text{-}to}[\![\mathtt{S}_5]\!] = \mathsf{break\text{-}to}[\![\mathtt{Sl}_6]\!] = \ell_6 \\
\mathsf{break\text{-}to}[\![\mathtt{Sl}_1]\!] = \mathsf{break\text{-}to}[\![\mathtt{S}_2]\!] = \mathsf{break\text{-}to}[\![\mathtt{Sl}_4]\!] = \ell_6 \\
\mathsf{break\text{-}to}[\![\mathtt{S}_3]\!] = \mathsf{break\text{-}to}[\![\mathtt{S}_5]\!] = \ell_6
\end{array}$$

so a `break` before the `while` loop would terminate the program at ℓ_7 whereas 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 $\mathtt{a} \in \mathcal{A}$ and the empty string ϵ, a regular expression has syntax $R ::= \mathtt{a} \mid \epsilon \mid R? \mid R_1 \mid R_2 \mid R^+ \mid R^*$, and semantics $\mathcal{S}[\![\mathtt{a}]\!] \triangleq \{\mathtt{a}\}$, $\mathcal{S}[\![\epsilon]\!] \triangleq \{\epsilon\}$, $\mathcal{S}[\![R?]\!] \triangleq \mathcal{S}[\![R]\!] \cup \{\epsilon\}$, $\mathcal{S}[\![R_1 \mid R_2]\!] \triangleq \mathcal{S}[\![R_1]\!] \cup \mathcal{S}[\![R_2]\!]$, $\mathcal{S}[\![R^0]\!] \triangleq \{\epsilon\}$, $\mathcal{S}[\![R^1]\!] \triangleq \mathcal{S}[\![R]\!]$, $\mathcal{S}[\![R^{n+1}]\!] \triangleq \mathcal{S}[\![R]\!]\mathcal{S}[\![R^n]\!]$, $\mathcal{S}[\![R^+]\!] \triangleq \bigcup_{n>0} \mathcal{S}[\![R^n]\!]$, $\mathcal{S}[\![R^*]\!] \triangleq \bigcup_{n\geqslant 0} \mathcal{S}[\![R^n]\!]$ 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 9.3 By reductio ad absurdum, let `P` be a program and `X` be a variable not in `P`. Define `P'`

```
var X : int = 0;
P;
X := 1 / X;
```

`P` terminates if and only if `P'` has a runtime error (division by 0, because `P` does not use or modify `X`). So if the absence of runtime error were decidable then termination would be decidable, which is a contradiction. □

Solution to Exercise 9.4 Sign analysis in section 3.2 is undecidable because otherwise, given a program `P`, consider a fresh variable `x` not in `P` and the derived program $\texttt{P}' = \texttt{P;x = 1 ;}$. $\texttt{P}'$ assigns a strictly positive value to the variable `x` different from the initial value 0 of `x` if and only if `P` terminates. Therefore, if the sign problem were decidable, termination would also be decidable, which is a contradiction. □

Solution to Exercise 9.13 Assume that we have an algorithm `correct(P, f)` that always terminates and returns true if and only if $\texttt{P}(n) = f(n)$ 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 $\mathrm{t}(n) = f(n)$ 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 11.1 Define $\gamma_n(x) = x + n$ so that $\alpha_n(x) \leqslant y \Leftrightarrow x - n \leqslant y \Leftrightarrow x \leqslant y + n \Leftrightarrow \gamma_n(x) \leqslant y$. Moreover, α_n is bijective. □

Solution to Exercise 11.11

$$
\begin{array}{ll}
\quad R^*(P) \supseteq Q & \\
\Leftrightarrow \forall y \in Q \,.\, \forall x \in P \,.\, \langle x,\ y\rangle \in R & \text{(definition of } \supseteq \text{ and } R^*\text{)} \\
\Leftrightarrow \forall x \in P \,.\, \forall y \in Q \,.\, \langle x,\ y\rangle \in R & \text{(definition of } \forall\text{)} \\
\Leftrightarrow P \subseteq R^\dagger(Q) & \text{(definition of } \subseteq \text{ and } R^\dagger\text{)}
\end{array}
$$

□

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 $\langle\wp(\Sigma^*),\ \subseteq\rangle \xleftrightarrows[\alpha_{\overleftarrow{w}}]{\gamma_{\overleftarrow{w}}} \langle\wp(\Sigma^*),\ \subseteq\rangle$ 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 $\langle\wp(\Sigma^*),\ \subseteq\rangle \xleftrightarrows[\alpha_{\overrightarrow{w}}]{\gamma_{\overrightarrow{w}}} \langle\wp(\Sigma^*),\ \subseteq\rangle$ where $\alpha_{\overrightarrow{w}}(L) = Lw$ and $\gamma_{\overrightarrow{w}}(L) = Lw^{-1}$. M □

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 the homomorphic/partitioning abstraction of exercise 11.6 and 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 identified with $\mathsf{E}(X)$. The concretization is a set of distributions, so the best-guess 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

$$
\begin{array}{l}
\quad \alpha \,\fatsemi\, \sqsubseteq\, = \,\leqslant \,\fatsemi\, \gamma^{-1} \\
\Leftrightarrow \forall P, Q : (\langle P,\ Q\rangle \in \alpha \,\fatsemi\, \sqsubseteq) \Leftrightarrow (\langle P,\ Q\rangle \in \leqslant \,\fatsemi\, \gamma^{-1}) \hfill \text{(def. equality of relations)} \\
\Leftrightarrow \forall P, Q : (\exists R : \langle P,\ R\rangle \in \alpha \wedge \langle R,\ Q\rangle \in \sqsubseteq) \Leftrightarrow (\exists R' : \langle P,\ R'\rangle \in \leqslant \wedge \langle R',\ Q\rangle \in \gamma^{-1}) \\
\hfill \text{(def. composition of relations } r_1 \fatsemi r_2 \triangleq \{\langle x,\ z\rangle \mid \exists y : \langle x,\ y\rangle \in r_1 \wedge \langle y,\ z\rangle \in r_2\}\text{)}
\end{array}
$$

$\Leftrightarrow\ \forall P, Q : (\exists R : \langle P,\ R\rangle \in \alpha \wedge \langle R,\ Q\rangle \in \sqsubseteq) \Leftrightarrow (\exists R' : \langle P,\ R'\rangle \in \leqslant \wedge \langle Q,\ R'\rangle \in \gamma)$

$\wr$def. inverse of relations$\wr$

$\Leftrightarrow\ \forall P, Q : (\exists R : \langle P,\ R\rangle \in \alpha \wedge R \sqsubseteq Q) \Leftrightarrow (\exists R' : P \leqslant R' \wedge \langle Q,\ R'\rangle \in \gamma)$

$\wr$def. order relations$\wr$

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

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

$\Leftrightarrow\ \langle C,\ \leqslant\rangle \xleftarrow[\alpha]{\gamma} \langle A,\ \sqsubseteq\rangle$ $\wr$by (11.1)$\wr$

$\square$

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

$$
\begin{aligned}
& \alpha_p(f) \sqsubseteq y \\
\Leftrightarrow\ & f(p) \sqsubseteq y && \wr\text{definition of } \alpha_p\wr \\
\Leftrightarrow\ & \forall x \sqsubseteq p\ .\ f(x) \sqsubseteq y && \wr f \text{ increasing and } \sqsubseteq \text{ reflexive and transitive}\wr \\
\Leftrightarrow\ & \forall x\ .\ f(x) \sqsubseteq [\![x \sqsubseteq p \mathrel{?} y \mathrel{\text{⁏}} \top]\!] && \wr\text{def. conditional and supremum } \top\wr \\
\Leftrightarrow\ & \forall x\ .\ f(x) \sqsubseteq \gamma_p(y)(x) && \wr\text{by defining } \gamma_p(y)(x) \triangleq [\![x \sqsubseteq p \mathrel{?} y \mathrel{\text{⁏}} \top]\!]\wr \\
\Leftrightarrow\ & f \mathrel{\dot\sqsubseteq} \gamma_p(y) && \wr\text{pointwise}\wr
\end{aligned}
$$

$\square$

Solution to Exercise 11.22

$$
\begin{aligned}
& \alpha_h(X) \mathrel{\dot\subseteq} Y \\
\Leftrightarrow\ & \forall a \in A\ .\ \alpha_h(X)\, a \subseteq Y(a) && \wr\text{pointwise definition of } \dot\subseteq\wr \\
\Leftrightarrow\ & \forall a \in A\ .\ \{f(a)x \mid x \in X\} \subseteq Y(a) && \wr\text{definition of } \alpha_h\wr \\
\Leftrightarrow\ & \forall a \in A\ .\ \forall x \in X\ .\ f(a)x \in Y(a) && \wr\text{definition of } \subseteq\wr \\
\Leftrightarrow\ & \forall x \in X\ .\ \forall a \in A\ .\ f(a)x \in Y(a) && \wr\text{definition of } \forall\wr \\
\Leftrightarrow\ & X \subseteq \{x \mid \forall a \in A\ .\ f(a)x \in Y(a)\} && \wr\text{definition of } \subseteq\wr \\
\Leftrightarrow\ & X \subseteq \gamma_h(Y) && \wr\text{by defining } \gamma_h(Y) \triangleq \{x \mid \forall a \in A\ .\ f(a)x \in Y(a)\}\wr
\end{aligned}
$$

$\square$

Solution to Exercise 11.30 If $x \in X$ then $x \sqsubseteq_1 \sqcup_1 X$ by definition of the lub so $f(x) \sqsubseteq_2 f(\sqcup_1 X)$ because 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 definition of an 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 to $\gamma \circ \alpha$ is extensive in exercise 11.34. A fix is to consider $\mathbb{N} \cup \{\infty\}$ with $\gamma(\bullet) = \infty$. □

Solution to Exercise 11.39 γ does not preserve meets. □

Solution to Exercise 11.40 γ preserves finite meets but not infinite ones. □

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 $x \div y = \max\{z \mid x \times y \leqslant x\}$. □

Solution to Exercise 11.47

$$
\begin{array}{ll}
 & \gamma(a) \\
= & \max\{c \in C \mid c \sqsubseteq \gamma(a)\} \hfill \text{≀The max exists and is } \gamma(a) \text{ by reflexivity≀} \\
= & \max\{c \in C \mid \alpha(c) \preccurlyeq a\} \hfill \text{≀}\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle\text{≀} \\
= & \max\{c \in C \mid \alpha(c) \in \downarrow a\} \hfill \text{≀definition of } \downarrow a \triangleq \{x \in \mathcal{A} \mid x \preccurlyeq a\}\text{≀} \\
= & \max \alpha^{-1}(\downarrow a) \hfill \text{≀definition of } \alpha^{-1}(\downarrow a) \triangleq \{c \in C \mid \alpha(c) \in \downarrow a\}\text{≀}
\end{array}
$$

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

Solution to Exercise 11.49 — If α is surjective then $\forall \overline{P} \in \mathcal{A} : \exists P \in C : \alpha(P) = \overline{P}$. Therefore if $\gamma(\overline{P}) = \gamma(\overline{P}')$ then $\gamma(\alpha(P)) = \gamma(\alpha(P'))$ for some $P, P' \in \mathcal{A}$ such that $\overline{P} = \alpha(P)$ and $\overline{P}' = \alpha(P')$. By reflexivity, $\gamma(\alpha(P)) \leqslant \gamma(\alpha(P'))$ hence $P \leqslant \gamma(\alpha(P'))$ because $\gamma \circ \alpha$ is extensive. By (11.1), this implies $\alpha(P) \sqsubseteq \alpha(P')$ that is $\overline{P} \sqsubseteq \overline{P}'$. Exchanging $\overline{P}$ and $\overline{P}'$ in the previous proof, we get $\overline{P}' \sqsubseteq \overline{P}$ and so $\overline{P} = \overline{P}'$ by antisymmetry, proving γ to be injective.

— By exercise 11.44, we have $\gamma \circ \alpha \circ \gamma(\overline{P}) = \gamma(\overline{P})$ for all $\overline{P} \in A$ so if γ is injective then $\alpha \circ \gamma(\overline{P}) = \overline{P}$.

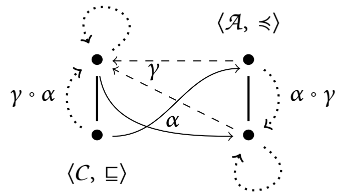
— If $\alpha(P) = Q$ then $\alpha(P) \sqsubseteq Q$ so $P \leqslant \gamma(Q)$, proving $\gamma(Q)$ to be greater that all elements of $\{P \in C \mid \alpha(P) = Q\}$. Moreover, $\alpha \circ \gamma$ is the identity on $\mathcal{A}$ so $\gamma(Q) \in \{P \in C \mid \alpha(P) = Q\}$, proving $\gamma(Q)$ to be the maximum of the elements of $\{P \in C \mid \alpha(P) = Q\}$.

— Finally, if $\forall Q \in A : \gamma(Q) = \max\{P \in C \mid \alpha(P) = Q\}$ then given any $Q \in A$, $\gamma(Q) \in \{P \in C \mid \alpha(P) = Q\}$ so $\alpha(\gamma(Q)) = Q$, proving α to be surjective.

— An isomorphism between C and $\mathcal{A}$ is not necessarily increasing. □

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

Solution to Exercise 11.58 Not necessarily — here is a counterexample (α is not increasing).



□

Solution to Exercise 11.64 Let us prove $\langle \wp(\mathcal{P}),\ \subseteq\rangle \xrightleftharpoons[\downarrow]{\uparrow} \langle \wp(\mathcal{P}),\ \supseteq\rangle$.

$$
\begin{array}{ll}
\quad \downarrow(X) \supseteq Y & \\
\Leftrightarrow\ \forall y \in Y\ .\ y \in \downarrow(X) & \wr\text{definition of } \supseteq\wr \\
\Leftrightarrow\ \forall y \in Y\ .\ \forall x \in X\ .\ y \sqsubseteq x & \wr\text{definition of } \downarrow \text{ and } \in\wr \\
\Leftrightarrow\ \forall x \in X\ .\ \forall y \in Y\ .\ y \sqsubseteq x & \wr\text{definition of } \forall\wr \\
\Leftrightarrow\ X \subseteq \{x \in \mathcal{P} \mid \forall y \in Y\ .\ y \sqsubseteq x\} & \wr\text{definition of } \forall\wr \\
\Leftrightarrow\ X \subseteq \uparrow(Y) & \wr\text{definition of } \uparrow\wr
\end{array}
$$

By exercise 11.56, $\uparrow \circ \downarrow$ is a closure operator. Therefore $\langle \uparrow \circ \downarrow(\wp(\mathcal{P})),\ \supseteq\rangle$ is a complete lattice by exercise 11.63. 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 definition of $\supseteq$, $y \in \{z \mid x \sqsubseteq z\}$, 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 because 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 $x \sqsubseteq y \Leftrightarrow \dot{\uparrow}(x) \supseteq \dot{\uparrow}(y)$ implies that they have the same ordering in $\langle \uparrow \circ \downarrow(\wp(\mathcal{P})),\ \supseteq\rangle$. Let $\dot{\uparrow}^{-1}$ be the inverse of the bijection $\dot{\uparrow} \in \mathcal{P} \rightarrowtail\!\!\!\!\twoheadrightarrow \uparrow \circ \downarrow(\wp(\mathcal{P}))$. We have $X \supseteq Y$ implies $\dot{\uparrow} \circ \dot{\uparrow}^{-1}(X) \supseteq \dot{\uparrow} \circ \dot{\uparrow}^{-1}(Y)$ implies $\dot{\uparrow}^{-1}(X) \sqsubseteq \dot{\uparrow}^{-1}(Y)$ by the embedding, proving that $\dot{\uparrow}^{-1}$ is decreasing. If $x \in \mathcal{P}$ and $Y \in \uparrow \circ \downarrow(\wp(\mathcal{P}))$ then $\dot{\uparrow} x \supseteq Y \Leftrightarrow \dot{\uparrow}^{-1} \circ \dot{\uparrow} x \sqsubseteq \dot{\uparrow}^{-1}(Y) \Leftrightarrow x \sqsubseteq \dot{\uparrow}^{-1}(Y)$, proving $\langle \mathcal{P},\ \sqsubseteq\rangle \xrightleftharpoons[\dot{\uparrow}]{\dot{\uparrow}^{-1}} \langle \uparrow \circ \downarrow(\wp(\mathcal{P})),\ \supseteq\rangle$.

The proof by MacNeille [3, Theorem 11.9] 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 [1] to construct the real numbers from the rational numbers, hence the name *Dedekind–-MacNeille completion*. □

Solution to Exercise 11.67 An hint is to use lemma 11.37 for α_a. □

Solution to Exercise 11.72 See theorem 11.71. □

5 Solutions to Selected Exercises of Chapter 12

Solution to Exercise 12.24

— $\mathsf{post}[R]P \subseteq Q$

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

$\Leftrightarrow \forall y \in \mathbb{Q} \,.\, (\exists x \in P \,.\, \langle x,\ y\rangle \in R) \Rightarrow y \in Q$ $\quad$ ≀definition of ⊆≀

$\Leftrightarrow \forall y \in \mathbb{Q} \,.\, \forall x \in P \,.\, \langle x,\ y\rangle \in R \Rightarrow y \in Q$ $\quad$ ≀definition of ⊆≀

$\Leftrightarrow \forall x \in P \,.\, \forall y \in \mathbb{Q} \,.\, \langle x,\ y\rangle \in R \Rightarrow y \in Q$ $\quad$ ≀definition of ∀≀

$\Leftrightarrow P \subseteq \{x \in \mathbb{P} \mid \forall y \in \mathbb{Q} \,.\, \langle x,\ y\rangle \in R \Rightarrow y \in Q\}$ $\quad$ ≀definition of ⊆≀

$\Leftrightarrow P \subseteq \widetilde{\mathsf{pre}}[R]Q$ $\quad$ ≀definition (12.12) of $\widetilde{\mathsf{pre}}$≀

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

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

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

≀pointwise definition of $\dot{\subseteq}$ and (12.2) of post≀

$\Leftrightarrow \forall y \in \mathbb{Q} \,.\, (\exists x \in P \,.\, \langle x,\ y\rangle \in R) \Rightarrow y \in T(P)$ $\quad$ ≀definition of ⊆≀

$\Leftrightarrow \forall x \in \mathbb{P} \,.\, \forall y \in \mathbb{Q} \,.\, (x \in P \Rightarrow (\langle x,\ y\rangle \in R) \Rightarrow y \in T(P))$ $\quad$ ≀definition of ⇒ and ∀≀

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

≀(⇒) for $P = \{x\}$ so $x \in P$ is true;
(⇐) 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 $y \in T(P)$. ≀

$\Leftrightarrow R \subseteq \{\langle x,\ y\rangle \in \mathbb{P} \times \mathbb{Q} \mid y \in T(\{x\})\}$ $\quad$ ≀definition of ⊆≀

$\Leftrightarrow R \subseteq \mathsf{post}^{-1}[T]$ $\quad$ ≀definition (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.

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

$\Leftrightarrow \mathsf{post}[R^{-1}] \mathrel{\dot{\subseteq}} T$ $\quad$ ≀definition (12.11) of pre≀

$$\Leftrightarrow\ R^{-1} \subseteq \mathsf{post}^{-1}[T] \qquad \wr \text{by } \langle \wp(\mathbb{Q}\times\mathbb{P}),\ \subseteq\rangle \xrightleftharpoons[\mathsf{post}]{\mathsf{post}^{-1}} \langle \wp(\mathbb{Q}) \xrightarrow{\sqcup} \wp(\mathbb{P}),\ \dot{\subseteq}\rangle \wr$$

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

$$R \subseteq \mathsf{pre}^{-1}[T] \qquad \wr \text{definition (12.21) of } \mathsf{pre}^{-1} \wr$$

□

Solution to Exercise 12.27 An execution starting with an initial environment in P, will have the following behaviors (a) $\mathsf{post}[\mathtt{S}]P \subseteq Q$, (b) $\mathsf{post}[\mathtt{S}]P \subseteq \neg Q$, (c) $\mathsf{post}[\mathtt{S}]P \subseteq \{\bot\}$, (ab) $\mathsf{post}[\mathtt{S}]P \subseteq \mathbb{Q} \setminus \{\bot\} \wedge \mathsf{post}[\mathtt{S}]P \nsubseteq Q \wedge \mathsf{post}[\mathtt{S}]P \nsubseteq \neg Q$, (ac) $\mathsf{post}[\mathtt{S}]P \subseteq Q \cup \{\bot\}$, (bc) $\mathsf{post}[\mathtt{S}]P \subseteq \neg Q \cup \{\bot\}$, (abc) $\mathsf{post}[\mathtt{S}]P \nsubseteq Q \wedge \mathsf{post}[\mathtt{S}]P \nsubseteq \neg Q \wedge \mathsf{post}[\mathtt{S}]P \nsubseteq \{\bot\}$. □

6 Solutions to Selected Exercises of Chapter 13

Solution to Exercise 13.2 The smallest topology on $\mathcal{X}$ is $\{\emptyset, \mathcal{X}\}$ 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 18

Solution to Exercise 18.5

$$\overline{f}'(\overline{x})$$

$$= \bigvee_{i=1}^{n} \alpha(f(x^i)) \qquad \wr \text{decomposition hypothesis } \gamma(\overline{x}) = \textstyle\bigsqcup_{i=1}^{n} x^i \wr$$

$$= \alpha(\bigsqcup_{i=1}^{n} f(x^i)) \qquad \wr \text{the lower adjoint of a Galois connection preserves existing joins} \wr$$

$$= \alpha(f(\bigsqcup_{i=1}^{n} x^i)) \qquad \wr f \in C\, joinmorphismtoC \text{ preserves existing arbitrary joins} \wr$$

$$= \alpha \circ f \circ \gamma(\overline{x}) \qquad \wr \text{decomposition hypothesis } \gamma(\overline{x}) = \textstyle\bigsqcup_{i=1}^{n} x^i \text{ and definition of function composition } \circ \wr$$

$$\preceq \overline{f}(\overline{x}) \qquad \wr \text{hypothesis } \alpha \circ f \circ \gamma \mathrel{\dot{\preceq}} \overline{f}. \wr$$

□

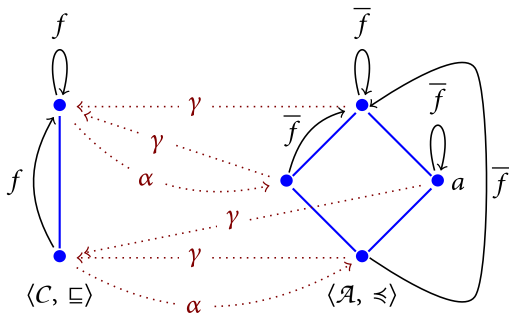
Solution to Exercise 18.14 The proof that $\alpha \circ f \circ \gamma \mathrel{\dot{\preceq}} \overline{f} \Leftrightarrow f \circ \gamma \sqsubseteq \gamma \circ \overline{f}$ does not use the fact that f and $\overline{f}$ are increasing.

$$\begin{aligned}
&\alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} \\
\Rightarrow\ & f \circ \gamma \sqsubseteq \gamma \circ \overline{f} && \text{(definition of Galois connections)} \\
\Rightarrow\ & \alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \alpha \circ \gamma \circ \overline{f} && \text{(Galois connection so } \alpha \text{ is increasing)} \\
\Rightarrow\ & \alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} && \text{(Galois connection so } \alpha \circ \gamma \mathrel{\dot{\preccurlyeq}} \mathbb{1}_{\mathcal{A}}\text{)}
\end{aligned}$$

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

$$\begin{aligned}
&\alpha \circ f \mathrel{\dot{\preccurlyeq}} \overline{f} \circ \alpha \\
\Rightarrow\ & \alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} \circ \alpha \circ \gamma && \text{(function application)} \\
\Rightarrow\ & \alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f} \\
& && \text{(if } \alpha \circ \gamma = \mathbb{1}_{\mathcal{A}} \text{ or } \overline{f} \text{ increasing because } \alpha \circ \gamma \mathrel{\dot{\preccurlyeq}} \mathbb{1}_{\mathcal{A}} \text{ by the Galois connection)} \\
\Rightarrow\ & \alpha \circ f \circ \gamma \circ \alpha \mathrel{\dot{\preccurlyeq}} \overline{f} \circ \alpha && \text{(function application)} \\
\Rightarrow\ & \alpha \circ f \mathrel{\dot{\preccurlyeq}} \overline{f} \circ \alpha \\
& && \text{(by } \mathbb{1}_{C} \sqsubseteq \gamma \circ \alpha \text{ and } \alpha \text{ increasing in a Galois connection and } f \text{ increasing)}
\end{aligned}$$

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



□

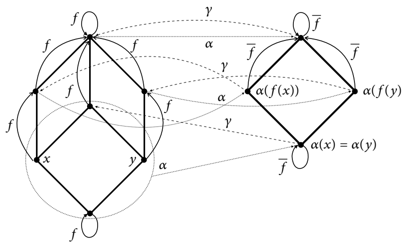
Solution to Exercise 18.16 If $\overline{f}(y) \preccurlyeq y$ then $f(\gamma(y)) \sqsubseteq \gamma(\overline{f}(y)) \sqsubseteq \gamma(y)$ by semicommutation $f \circ \gamma \mathrel{\dot{\sqsubseteq}} \gamma \circ \overline{f}$ and γ increasing. So $\gamma(y) \in \{x \mid f(x) \sqsubseteq x\}$ proving, by Tarski's fixpoint theorem 15.6 and definition of the 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 (a) Assume $\alpha \circ f = \overline{f} \circ \alpha$. Then $\alpha \circ f = \overline{f} \circ \alpha$ then $\alpha \circ f = \alpha \circ f \circ \gamma \circ \alpha$ and so if $\alpha(x) = \alpha(y)$ then $\alpha \circ f(x) = \alpha \circ f \circ \gamma \circ \alpha(x) = \alpha \circ f \circ \gamma \circ \alpha(y) = \alpha \circ f(y)$, proving (18.31). Conversely, by the dual of exercise 11.44, $\forall x \in C \mathrel{.} \alpha(x) = \alpha(\gamma \circ \alpha(x))$ so (18.31) implies that $f(\alpha(x)) = \alpha(f(\gamma \circ \alpha(x))) = \overline{f}(\alpha(x))$.

(b) Assume that $\bigvee_{i\in\Delta} \overline{x}_i$ and $\bigsqcup_{i\in\Delta} \gamma(\overline{x}_i)$ do exist in the posets $\mathcal{A}$ and C. Then

$$
\begin{aligned}
&\overline{f}(\bigvee_{i\in\Delta} x_i)\\
=\;& \alpha \circ f \circ \gamma(\bigvee_{i\in\Delta} x_i) && \wr\text{definition of } \overline{f}\,\wr\\
=\;& \alpha \circ f(\bigsqcup_{i\in\Delta} \gamma(x_i))\\
& \wr\text{By lemma 11.37, } \alpha \text{ preserves existing lubs and by exercise 11.49, } \alpha \circ \gamma = \mathbb{1}_{\mathcal{A}}\\
& \text{so } \alpha(\textstyle\bigsqcup_{i\in\Delta} \gamma(x_i)) = \bigvee_{i\in\Delta} \alpha \circ \gamma(x_i) = \bigvee_{i\in\Delta} x_i = \alpha(\gamma(\bigvee_{i\in\Delta} x_i)) \text{ and so, by (18.31),}\\
& \alpha(f(\textstyle\bigsqcup_{i\in\Delta} \gamma(x_i))) = \alpha(f(\gamma(\bigvee_{i\in\Delta} x_i)))\wr\\
=\;& \alpha \circ \bigsqcup_{i\in\Delta} f(\gamma(x_i)) && \wr\text{by hypothesis, } f \text{ preserves existing lubs}\wr\\
=\;& \bigvee_{i\in\Delta} \alpha \circ f \circ \gamma(x_i) && \wr\text{by lemma 11.37, } \alpha \text{ preserves existing lubs}\wr\\
=\;& \bigvee_{i\in\Delta} \overline{f}(x_i) && \wr\text{definition of } \overline{f}\,\wr
\end{aligned}
$$

(c) Here is a counterexample.



□

Solution to Exercise 18.32 $\langle D \xrightarrow{\sqcup} D, \dot{\sqsubseteq}\rangle$ and $\langle D, \sqsubseteq\rangle$ are complete lattices, $\mathcal{F} \in (D \xrightarrow{\sqcup} D) \longrightarrow (D \xrightarrow{\sqcup} D)$ is $\dot{\sqsubseteq}$-increasing. We have $\langle D \xrightarrow{\sqcup} D, \dot{\sqsubseteq}\rangle \xleftrightarrows[\alpha_x]{\gamma_x} \langle D, \sqsubseteq\rangle$ by exercise 11.38 because lubs exist in a complete lattice and α_x preserves arbitrary joins:

$$
\begin{aligned}
&\alpha_x(\dot{\bigsqcup_i} f_i)\\
=\;& \mathcal{F}(\dot{\bigsqcup_i} f_i)x && \wr\text{definition of } \alpha_x\wr\\
=\;& (\dot{\bigsqcup_i} \mathcal{F}(f_i))x && \wr\mathcal{F} \text{ preserves joins}\wr\\
=\;& \bigsqcup_i (\mathcal{F}(f_i)x) && \wr\text{pointwise definition of } \dot{\bigsqcup}\wr\\
=\;& \bigsqcup_i \alpha_x(f_i) && \wr\text{definition of xs}\alpha_x\wr
\end{aligned}
$$

$F(x) \in D \xrightarrow{\sqcup} D$ is $\sqsubseteq$-increasing and we have the commutation property $\alpha_x \circ \mathcal{F} = F(x) \circ \alpha_x$. By theorem 18.22, it follows that $\mathsf{lfp}^{\sqsubseteq} F(x) = \alpha_x(\mathsf{lfp}^{\dot\sqsubseteq} \mathcal{F}) = \mathcal{F}(\mathsf{lfp}^{\dot\sqsubseteq} \mathcal{F})x = (\mathsf{lfp}^{\dot\sqsubseteq} \mathcal{F})x$ for all $x \in D$ so $\mathsf{lfp}^{\dot\sqsubseteq} \mathcal{F} = x \in D \mapsto \mathsf{lfp}^{\sqsubseteq} F(x)$. $\square$

Solution to Exercise 18.42 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 $\mathsf{lfp}^{\sqsubseteq} f = \bigsqcup_{n\in\mathbb{N}} f^{n}(\bot) = \bigsqcup_{n\in\mathbb{N}} \gamma(\overline{f}^{n}(\mathbf{0})) = \bigsqcup_{n\in\mathbb{N}} \gamma(\overline{f}^{\ell}(\mathbf{0})) = \gamma(\overline{f}^{\ell}(\mathbf{0})) = \gamma(\mathsf{lfp}^{\preccurlyeq} \overline{f}))$. $\square$

8 Solutions to Selected Exercises of Chapter 19

Solution to Exercise 19.9 We have $\langle \wp(\mathbb{E}\mathrm{v}\times\mathbb{E}\mathrm{v}), \subseteq\rangle \xleftrightarrow[\alpha]{\gamma} \langle \wp(\mathbb{E}\mathrm{v}), \subseteq\rangle$ with $\alpha(R) \triangleq \{\rho \mid \exists \rho_0 \in \mathbb{E}\mathrm{v} \;.\; \langle \rho_0,\ \rho\rangle \in R\}$ and $\gamma(r) \triangleq \{\langle \rho_0,\ \rho\rangle \mid \rho_0 \in \mathbb{E}\mathrm{v} \wedge \rho \in r\}$. By pointwise extension in exercise 11.20, it follows that $\langle \mathbb{L} \to \wp(\mathbb{E}\mathrm{v}\times\mathbb{E}\mathrm{v}), \dot\subseteq\rangle \xleftrightarrow[\dot\alpha]{\dot\gamma} \langle \mathbb{L} \to \wp(\mathbb{E}\mathrm{v}), \dot\subseteq\rangle$. It follows, by theorem 11.77, that $\langle \wp(\mathbb{E}\mathrm{v}\times \mathbb{E}\mathrm{v}) \xrightarrow{\nearrow} (\mathbb{L} \to \wp(\mathbb{E}\mathrm{v}\times\mathbb{E}\mathrm{v})), \ddot\subseteq\rangle \xleftrightarrow[\vec\alpha]{\vec\gamma} \langle \wp(\mathbb{E}\mathrm{v}) \xrightarrow{\nearrow} (\mathbb{L} \to \wp(\mathbb{E}\mathrm{v})), \ddot\subseteq\rangle$ 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, $\mathcal{S}^{\vec{r}}[\![\mathsf{S}]\!] = \vec\alpha(\mathcal{S}^{\vec{R}}[\![\mathsf{S}]\!])$. $\square$

Solution to Exercise 19.27 No, because of iteration. A counterexample is provided by example 19.1 $\square$

9 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) *)
```
□

10 Solutions to Selected Exercises of Chapter 24

Solution to Exercise 24.16 $\langle L, \sqsubseteq, \bot, \sqcup\rangle$ is a complete lattice so $\langle (L \to L), \dot{\sqsubseteq}, \dot{\bot}, \dot{\sqcup}\rangle$ is a complete lattice, pointwise. The Galois connection $\langle (L \to L), \dot{\sqsubseteq}\rangle \underset{\overleftarrow{F}}{\overset{\vec{F}}{\leftrightarrows}} \langle (L \to L), \dot{\sqsubseteq}\rangle$ 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–Kleene's iterative fixpoint theorem 15.26. By duality, $\mathsf{gfp}^{\sqsubseteq}\, \overleftarrow{F}$ does exist.

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

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

$$
\begin{array}{lll}
& \vec{F}^{n+1}(X) \dot{\sqsubseteq} Y & \\
\Leftrightarrow & \vec{F}(\vec{F}^n(X)) \dot{\sqsubseteq} Y & \wr\text{definition of the iterates}\wr \\
\Leftrightarrow & \vec{F}^n(X) \dot{\sqsubseteq} \overleftarrow{F}(Y) & \wr\text{Galois connection hypothesis}\wr \\
\Leftrightarrow & X \dot{\sqsubseteq} \overleftarrow{F}^n(\overleftarrow{F}(Y)) & \wr\text{recurrence hypothesis}\wr \\
\Leftrightarrow & X \dot{\sqsubseteq} \overleftarrow{F}^{n+1}(Y) & \wr\text{definition of the iterates}\wr
\end{array}
$$

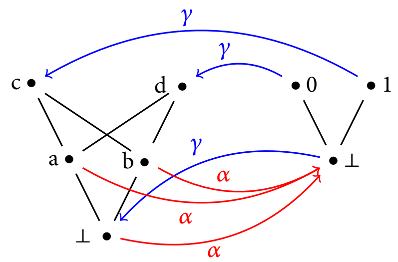
It follows that

$$
\begin{array}{lll}
& (\mathsf{lfp}^{\dot{\sqsubseteq}}\, \vec{F})(X) \sqsubseteq Y & \\
\Leftrightarrow & (\dot{\bigsqcup_{n\in\mathbb{N}}} \vec{F}^n(\bot))(X) \sqsubseteq Y & \wr\text{Scott–Kleene's iterative fixpoint theorem 15.26}\wr \\
\Leftrightarrow & \bigsqcup_{n\in\mathbb{N}}(\vec{F}^n(\bot)(X)) \sqsubseteq Y & \wr\text{pointwise definition of } \dot{\bigsqcup}\wr \\
\Leftrightarrow & \forall n \in \mathbb{N} \,.\, (\vec{F}^n(\bot)(X)) \sqsubseteq Y & \wr\text{definition of the lub } \bigsqcup\wr \\
\Leftrightarrow & \forall n \in \mathbb{N} \,.\, X \sqsubseteq \overleftarrow{F}^n(\bot)(Y) & \wr\forall n \in \mathbb{N} \,.\, \langle (L \to L),\, \dot{\sqsubseteq}\rangle \underset{\vec{F}^n}{\overset{\overleftarrow{F}^n}{\leftrightarrows}} \langle (L \to L),\, \dot{\sqsubseteq}\rangle\wr \\
\Leftrightarrow & X \sqsubseteq \bigsqcap_{n\in\mathbb{N}} \overleftarrow{F}^n(\bot)(Y) & \wr\text{definition of the glb } \bigsqcap\wr \\
\Leftrightarrow & X \sqsubseteq (\dot{\bigsqcap_{n\in\mathbb{N}}} \overleftarrow{F}^n(\bot))(Y) & \wr\text{pointwise definition of } \dot{\bigsqcap}\wr \\
\Leftrightarrow & X \sqsubseteq (\mathsf{gfp}^{\dot{\sqsubseteq}}\, \overleftarrow{F})(Y) & \wr\text{dual of Scott–Kleene's iterative fixpoint theorem 15.26}\wr
\end{array}
$$

□

11 Solutions to Selected Exercises of Chapter 27

Solution to Exercise 27.14 In the following example, 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. $\square$

12 Solutions to Selected Exercises of Chapter 28

Solution to Exercise 28.36

— $\dot{\alpha}_{\vec{\times}}(\mathsf{test}^{\vec{r}}[\![\mathsf{A}_1 < \mathsf{A}_2]\!](\mathcal{R}_0))$

$= \dot{\alpha}_{\vec{\times}}(\{\rho \in \mathcal{R}_0 \mid \boldsymbol{\mathscr{B}}[\![\mathsf{A}_1 < \mathsf{A}_2]\!]\,\rho = \mathtt{tt}\})$ $\quad$ ≀definition (19.16) of $\mathsf{test}^{\vec{r}}[\![\mathsf{B}]\!]$∫

$= \dot{\alpha}_{\vec{\times}}(\{\rho \in \mathcal{R}_0 \mid \boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho < \boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho\})$ $\quad$ ≀definition (3.4) of $\boldsymbol{\mathscr{B}}[\![\mathsf{A}_1 < \mathsf{A}_2]\!]$∫

$\dot{\subseteq}\ \dot{\alpha}_{\vec{\times}}(\{\rho \in \mathcal{R}_0 \mid \exists v_2 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \,.\, \boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho < v_2\} \cap \{\rho \in \mathcal{R}_0 \mid \exists v_1 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \,.\, v_1 < \boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho\})$

≀because $\{\rho \in S \mid f(\rho) < g(\rho)\} \subseteq \{\rho \in S \mid \exists v_1 \in \{f(\rho) \mid \rho \in S\} \,.\, v_1 < g(\rho)\}$ and similarly for g, and $\dot{\alpha}_{\vec{\times}}$ increasing∫

$= \dot{\alpha}_{\vec{\times}}(\{\rho \in \mathcal{R}_0 \mid \boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\rho \in \{v_1 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \mid \exists v_2 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \,.\, v_1 < v_2\}\} \cap \{\rho \in \mathcal{R}_0 \mid \boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\rho \in \{v_2 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \mid \exists v_1 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \,.\, v_1 < v_2\}\})$
≀definition of $\in$, letting $v1 = \boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho$ and $v_2 = \boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\rho$∫

$\dot{\subseteq}\ \dot{\alpha}_{\vec{\times}}(\boldsymbol{\mathscr{A}}^{\underline{\times}_1}[\![\mathsf{A}_1]\!]\ \{v_1 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \mid \exists v_2 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \,.\, v_1 < v_2\}\ \mathcal{R}_0)\ \dot{\cap}\ \dot{\alpha}_{\vec{\times}}(\boldsymbol{\mathscr{A}}^{\underline{\times}_1}[\![\mathsf{A}_2]\!]\ \{v_2 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \mid \exists v_1 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \,.\, v_1 < v_2\}\ \mathcal{R}_0)$

≀$\dot{\alpha}_{\vec{\times}}$ is $\dot{\subseteq}$-increasing by Galois connection of exercise 28.2, and $\dot{\cap}$ is $\dot{\subseteq}$ increasing.∫

$= \mathsf{let}\ \langle \chi_1, \chi_2 \rangle = \langle \{v_1 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \mid \exists v_2 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \,.\, v_1 < v_2\}, \{v_2 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \mid \exists v_1 \in \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho \mid \rho \in \mathcal{R}_0\} \,.\, v_1 < v_2\}\rangle\ \mathsf{in}$

$\quad \dot{\alpha}_{\vec{\times}}(\boldsymbol{\mathscr{A}}^{\underline{\times}_1}[\![\mathsf{A}_1]\!]\ \chi_1\ \mathcal{R}_0)\ \dot{\cap}\ \dot{\alpha}_{\vec{\times}}(\boldsymbol{\mathscr{A}}^{\underline{\times}_1}[\![\mathsf{A}_2]\!]\ \chi_2\ \mathcal{R}_0)$ $\quad$ ≀definition of let∫

$= \mathsf{let}\ \langle \chi_1, \chi_2 \rangle = \otimes^{\vec{\times}}\ \langle \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_1]\!]\,\rho \mid \rho \in \mathcal{R}_0\}, \{\boldsymbol{\mathscr{A}}[\![\mathsf{A}_2]\!]\,\rho \mid \rho \in \mathcal{R}_0\}\rangle\ \mathsf{in}$

$\quad \dot{\alpha}_{\vec{\times}}(\boldsymbol{\mathscr{A}}^{\underline{\times}_1}[\![\mathsf{A}_1]\!]\ \chi_1\ \mathcal{R}_0)\ \dot{\cap}\ \dot{\alpha}_{\vec{\times}}(\boldsymbol{\mathscr{A}}^{\underline{\times}_1}[\![\mathsf{A}_2]\!]\ \chi_2\ \mathcal{R}_0)$ $\quad$ ≀definition $\otimes^{\vec{\times}}$∫

$\dot{\subseteq}\ \mathsf{let}\ \langle \chi_1, \chi_2 \rangle = \otimes^{\vec{\times}}\ \langle \boldsymbol{\mathscr{A}}^{\times}[\![\mathsf{A}_1]\!]\dot{\alpha}_{\vec{\times}}(\mathcal{R}_0), \boldsymbol{\mathscr{A}}^{\times}[\![\mathsf{A}_2]\!]\dot{\alpha}_{\vec{\times}}(\mathcal{R}_0)\rangle\ \mathsf{in}$

$\quad \boldsymbol{\mathscr{A}}^{\underline{\times}_1}[\![\mathsf{A}_1]\!]\ \chi_1\ \dot{\alpha}_{\vec{\times}}(\mathcal{R}_0)\ \dot{\cap}\ \boldsymbol{\mathscr{A}}^{\underline{\times}_1}[\![\mathsf{A}_2]\!]\ \chi_2\ \dot{\alpha}_{\vec{\times}}(\mathcal{R}_0)$ $\quad$ ≀induction hypothesis∫

$= \mathsf{test}^{\vec{\times}}[\![\mathsf{A}_1 < \mathsf{A}_2]\!]\dot{\alpha}_{\vec{\times}}(\mathcal{R}_0)$ $\quad$ ≀definition (28.34) of $\mathsf{test}^{\vec{\times}}[\![\mathsf{A}_1 < \mathsf{A}_2]\!]$∫

□

13 Solutions to Selected Exercises of Chapter 33

Solution to Exercise 33.4

```
let neq (lx,hx) (ly,hy) =
  if (lx=hx)&&(lx=ly)&&(ly=hy) then (infimum,infimum) (* equal constants not != *)
   else if (lx=hx)&&(hx=ly)&&(ly<hy) then ((lx,hx),(ly+1,hy))
   else if (lx=hx)&&(lx=hy)&&(ly<hy) then ((lx,hx),(ly,hy-1))
   else if (lx<hx)&&(hx=ly)&&(ly=hy) then ((lx,hx-1),(ly,hy))
   else if (lx<hx)&&(hy=lx)&&(ly=hy) then ((lx+1,hx),(ly,hy))
   else ((lx,hx),(ly,hy))
```

□

Solution to Exercise 33.10 In order to simulate the precondition at ℓ_2 and observe the postcondition at ℓ_5, we analyze the following program:

```
if l1: (n < 1){i:T; n:T}
   {
      l2: {i:T; n:[-oo, 0]} i = n;
      while l3: (i != 1) {i:[-oo, 0]; n:[-oo, 0]}
         l4: {i:[-oo, 0]; n:[-oo, 0]} i = (i - 1);
      l5: {i:_|_; n:_|_} ;
   }
l6: {i:T; n:[1, oo]}
```

which shows that if initially $\mathtt{n} < 1$ at ℓ_2 then the program does not terminate at ℓ_5. □

Solution to Exercise 36.11

— $\overline{\mathbb{P}}_1 \stackrel{\frown}{\leqslant} \overline{\mathbb{P}}_2$

$\Rightarrow \forall \overline{P}_2 \in \overline{\mathbb{P}}_2 \,.\, \exists \overline{P}_1 \in \overline{\mathbb{P}}_1 \,.\, \gamma_1(\overline{P}_1) = \gamma_2(\overline{P}_2)$ $\qquad$ ⦃definition of $\stackrel{\frown}{\leqslant}$⦄

$\Rightarrow \forall P_2 \in \mathbb{P} \,.\, \exists \overline{P}_1 \in \overline{\mathbb{P}}_1 \,.\, \gamma_1(\overline{P}_1) = \gamma_2(\alpha_2(P_2))$ $\qquad$ ⦃because $\alpha_2(P_2) \in \overline{\mathbb{P}}_2$⦄

$\Rightarrow \forall P_2 \in \mathbb{P} \,.\, \exists \overline{P}_1 \in \overline{\mathbb{P}}_1 \,.\, \gamma_1 \circ \alpha_1 \circ \gamma_1(\overline{P}_1) = \gamma_2 \circ \alpha_2(P_2)$

⦃$\gamma_1 \circ \alpha_1 \circ \gamma_1 = \gamma_1$ in Galois connection and definition of $\circ$⦄

$\Rightarrow \forall P_2 \in \mathbb{P} \,.\, \exists P_1 \in \mathbb{P} \,.\, \gamma_1 \circ \alpha_1(P_1) = \gamma_2 \circ \alpha_2(P_2)$ $\qquad$ ⦃taking $P_1 = \gamma_1(\overline{P}_1)$⦄

$\Rightarrow \gamma_2 \circ \alpha_2(\mathbb{P}) \subseteq \gamma_1 \circ \alpha_1(\mathbb{P})$ $\qquad$ ⦃definition of $\subseteq$⦄

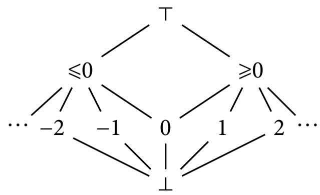
— Conversely, for all $\overline{P}_2 \in \overline{\mathbb{P}}_2$ then $\gamma_2(\overline{P}_2) \in \mathbb{P}$ so

$$\begin{aligned}
&\quad \exists P_1 \in \mathbb{P} \,.\, \gamma_1 \circ \alpha_1(P_1) = \gamma_2 \circ \alpha_2(\gamma_2(\overline{P}_2)) = \gamma_2(\overline{P}_2) && \text{(hyp. and } \gamma_2 \circ \alpha_2 \circ \gamma_2 = \gamma_2 \text{ in GC)}\\
&\Rightarrow \exists \overline{P}_1 \in \overline{\mathbb{P}}_1 \,.\, \gamma_1(\overline{P}_1) = \gamma_2(\overline{P}_2) && \text{(choosing } \overline{P}_1 = \alpha_1(P_1)\text{)}\\
&\Rightarrow \overline{\mathbb{P}}_1 \mathrel{\widehat{\leqslant}} \overline{\mathbb{P}}_2 && \text{(definition of } \widehat{\leqslant}\text{)}
\end{aligned}$$

□

14 Solutions to Selected Exercises of Chapter 36

Solution to Exercise 36.5



□

Solution to Exercise 36.25

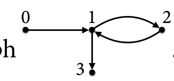
Define $\min(c + m\mathbb{Z}, a) \triangleq a + ((c - a) \pmod{|m|})$ when $m \neq 0$ to be the least element of $c + m\mathbb{Z}$ greater that or equal to a. We have $a \leqslant \min(c + m\mathbb{Z}, a) \leqslant a + |m|$ and $\min(c + m\mathbb{Z}, a) = a + (c - a) - |m|((c - a) \pmod{|m|}) = c - |m|((c - a) \pmod{|m|}) \in c + m\mathbb{Z}$. Similarly, let $\max(c + m\mathbb{Z}, a) \triangleq a - ((a - c) \pmod{|m|})$ when $m \neq 0$ be the greatest element of $c + m\mathbb{Z}$ less than of equal to a. By convention $\min(c + m\mathbb{Z}, -\infty) = -\infty$ and $\max(c + m\mathbb{Z}, \infty) \triangleq \infty$.

The reduction is as follows [2, proposition 6.1].

$$\begin{aligned}
\rho(\varnothing, i) &\triangleq \langle \varnothing,\ \varnothing \rangle\\
\rho(c, \varnothing) &\triangleq \langle \varnothing,\ \varnothing \rangle\\
\rho(c + 0\mathbb{Z}, i) &\triangleq \langle \varnothing,\ \varnothing \rangle && \text{when } c \notin i\\
\rho(c + 0\mathbb{Z}, i) &\triangleq \langle c + 0\mathbb{Z},\ [c, c] \rangle && \text{when } c \in i\\
\rho(c + m\mathbb{Z}, [\ell, h]) &\triangleq \langle \varnothing,\ \varnothing \rangle && \text{when } \min(c + m\mathbb{Z}, \ell) > \max(c + m\mathbb{Z}, h)\\
\rho(c + m\mathbb{Z}, [\ell, h]) &\triangleq \langle \min(c + m\mathbb{Z}, \ell) + 0\mathbb{Z},\ [\min(c + m\mathbb{Z}, \ell), \min(c + m\mathbb{Z}, \ell)] \rangle\\
& && \text{when } \min(c + m\mathbb{Z}, \ell) = \max(c + m\mathbb{Z}, h)\\
\rho(c + m\mathbb{Z}, [\ell, h]) &\triangleq \langle c + m\mathbb{Z},\ [\min(c + m\mathbb{Z}, \ell), \max(c + m\mathbb{Z}, h)] \rangle\\
& && \text{when } \min(c + m\mathbb{Z}, \ell) < \max(c + m\mathbb{Z}, h)
\end{aligned}$$

□

15 Solutions to Selected Exercises of Chapter 39

Solution to Exercise 39.29 Consider the following graph 

. Initially, $12 \in \widehat{\mathcal{F}}^0_{\pi}(1, 2)$, $13 \in \widehat{\mathcal{F}}^0_{\pi}(1, 3)$ and $21 \in \widehat{\mathcal{F}}^0_{\pi}(2, 1)$. The next iterate is identical because there is no path through

0. The next iterate through $1 \notin \{2,3\}$ adds $21 \odot 13 = 213 \in \widehat{\mathscr{F}}^2_\pi(2,3)$. The next iterate through $2 \notin \{1,3\}$ adds $12 \odot 213 = 1213 \in \widehat{\mathscr{F}}^3_\pi(1,3)$ which is not elementary and so does not belong to $\mathsf{p}^\ni(1,3)$. $\square$

16 Solutions to Selected Exercises of Chapter 44

Solution to Exercise 44.20

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

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

$$\begin{aligned}
&\mathcal{S}^{\mathsf{r}}[\![\mathsf{R} \bullet \varepsilon]\!] \\
&= \{\langle \underline{\varrho},\ \pi \cdot \pi' \rangle \mid \langle \underline{\varrho},\ \pi \rangle \in \mathcal{S}^{\mathsf{r}}[\![\mathsf{R}]\!] \wedge \langle \underline{\varrho},\ \pi' \rangle \in \mathcal{S}^{\mathsf{r}}[\![\varepsilon]\!]\} && \wr(44.7)\int \\
&= \{\langle \underline{\varrho},\ \pi \cdot \ni \rangle \mid \langle \underline{\varrho},\ \pi \rangle \in \mathcal{S}^{\mathsf{r}}[\![\mathsf{R}]\!]\} && \wr\text{because } \pi' = \ni \text{ by } (44.7)\int \\
&= \mathcal{S}^{\mathsf{r}}[\![\mathsf{R}]\!] && \wr\text{definition of concatenation} \cdot \text{and} \in\int
\end{aligned}$$

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

$$\begin{aligned}
&\mathsf{R}_1^f \bullet \mathsf{R}_2 \\
&\simeq \mathsf{R}_1^f \bullet \mathsf{R}_1^n \bullet \mathsf{R}_2 && \wr\text{because } \mathsf{R}_1^n \in \mathbb{R}_\varepsilon \text{ so } \mathsf{R}_1^f \bullet \mathsf{R}_1^n \simeq \mathsf{R}_1^f\int \\
&\simeq \mathsf{R}_1 \bullet \mathsf{R}_2 && \wr\text{by induction hypothesis because } \langle \mathsf{R}_1^f,\ \mathsf{R}_1^n \rangle = \mathsf{fstnxt}(\mathsf{R}_1),\ \textit{Q.E.D.}\int
\end{aligned}$$

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

$$\begin{aligned}
&\mathsf{R}_1^f \bullet \mathsf{R}_1^n \bullet \mathsf{R}_2 \\
&\simeq \mathsf{R}_1 \bullet \mathsf{R}_2 && \wr\text{by induction hypothesis because } \langle \mathsf{R}_1^f,\ \mathsf{R}_1^n \rangle = \mathsf{fstnxt}(\mathsf{R}_1),\ \textit{Q.E.D.}\int
\end{aligned}$$

- 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

$$\begin{aligned}
&\mathsf{R}^f \bullet \mathsf{R}^* \\
&\simeq \mathsf{R}^f \bullet \mathsf{R}^n \bullet \mathsf{R}^* && \wr\text{because } \mathcal{S}^{\mathsf{r}}[\![\mathsf{R}^n]\!] = \mathcal{S}^{\mathsf{r}}[\![\varepsilon]\!]\int \\
&\simeq \mathsf{R} \bullet \mathsf{R}^* && \wr\text{induction hypothesis because } \langle \mathsf{R}^f,\ \mathsf{R}^n \rangle = \mathsf{fstnxt}(\mathsf{R})\int \\
&\simeq \mathsf{R}^+ && \wr\text{definition } (44.7) \text{ of } \mathcal{S}^{\mathsf{r}}[\![\mathsf{R}^*]\!] \text{ and } \mathcal{S}^{\mathsf{r}}[\![\mathsf{R}^+]\!]\int
\end{aligned}$$

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

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

□

Solution to Exercise 44.53

— Let us first prove that $X \mapsto \vec{\tau} \mathbin{\frown} X$ 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 $\vec{\tau} \mathbin{\frown} \varnothing = \varnothing$. For $\Delta = \varnothing$, $\vec{\tau} \mathbin{\frown} \bigcup_{i\in\varnothing} X_i = \vec{\tau} \mathbin{\frown} \varnothing = \varnothing = \bigcup_{i\in\varnothing} \vec{\tau} \mathbin{\frown} X_i$. Otherwise, assuming $\Delta \neq \varnothing$, we have

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

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

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

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

— By recurrence on n.

– for $n = 0$,

$$
\begin{aligned}
& X^0 \\
= {} & \varnothing && \text{(definition of iterates from } \varnothing\text{)} \\
= {} & \bigcup_{0} \varnothing && \text{(definition of } \textstyle\bigcup\text{)} \\
= {} & \bigcup_{i=1} \boldsymbol{\mathcal{S}}^{i}_{\mathsf{t}}[\![\tau]\!] && \text{(definition of } \textstyle\bigcup_{i=1}^{j} x_i = \varnothing \text{ when } j < i\text{)}
\end{aligned}
$$

– for $n = 1$,

$$
\begin{aligned}
& X^1 \\
= \ & \mathbb{S}^1 \cup \vec{\tau} \mathbin{\frown} X^0 && \text{⟅definition of the iterates⟆} \\
= \ & \mathbb{S}^1 && \text{⟅}X^0 = \varnothing \text{ and definition of } \mathbin{\frown}\text{⟆} \\
= \ & \boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!] && \text{⟅}\boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!] = \mathbb{S}^1 \triangleq \{\pi \in \mathbb{S}^1 \mid \pi_0 = \iota_0\}\text{⟆} \\
= \ & \bigcup_{i=1}^{1} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!] && \text{⟅definition of } \textstyle\bigcup_{i=1}^{j} x_i = x_1 \text{ with } j = i\text{⟆}
\end{aligned}
$$

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

$$
\begin{aligned}
& X^{n+1} \\
= \ & \mathbb{S}^1 \cup \vec{\tau} \mathbin{\frown} X^n && \text{⟅definition of the iterates⟆} \\
= \ & \mathbb{S}^1 \cup \vec{\tau} \mathbin{\frown} (\bigcup_{i=1}^{n} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!]) && \text{⟅induction hypothesis⟆} \\
= \ & \mathbb{S}^1 \cup \bigcup_{i=1}^{n} (\vec{\tau} \mathbin{\frown} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!]) && \text{⟅}X \mapsto \vec{\tau} \mathbin{\frown} X \text{ preserves arbitrary joins, as shown previously⟆} \\
= \ & \boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!] \cup \bigcup_{i=1}^{n} (\boldsymbol{\mathcal{S}}^{i+1}_{\mathrm{t}}[\![\tau]\!]) && \text{⟅}\mathbb{S}^1 = \boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!] \text{ and } \boldsymbol{\mathcal{S}}^{i+1}_{\mathrm{t}}[\![\tau]\!] = \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!] \mathbin{\frown} \vec{\tau}\text{⟆} \\
= \ & \boldsymbol{\mathcal{S}}^1_{\mathrm{t}}[\![\tau]\!] \cup \bigcup_{j=2}^{n+1} (\boldsymbol{\mathcal{S}}^j_{\mathrm{t}}[\![\tau]\!]) && \text{⟅letting } j = i + 1\text{⟆} \\
= \ & \bigcup_{j=1}^{n+1} (\boldsymbol{\mathcal{S}}^j_{\mathrm{t}}[\![\tau]\!]) && \text{⟅incorporating the term } j = 1\text{⟆}
\end{aligned}
$$

— Let us apply Scott–Kleene's iterative fixpoint theorem 15.26.

$$
\begin{aligned}
\vec{X}^\infty \triangleq \ & \bigcup_{n\in\mathbb{N}} \vec{X}^n && \text{⟅definition of the iterates } \vec{X}^n \text{ of } X \mapsto \mathbb{S}^1 \cup X \mathbin{\frown} \vec{\tau} \text{ from } \varnothing\text{⟆} \\
= \ & \bigcup_{n\in\mathbb{N}} \bigcup_{i=1}^{n} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!] && \text{⟅}X^n = \textstyle\bigcup_{i=1}^{n} \boldsymbol{\mathcal{S}}^i_{\mathrm{t}}[\![\tau]\!]\text{, as shown previously⟆} \\
= \ & \bigcup_{n\in\mathbb{N}} \boldsymbol{\mathcal{S}}^n_{\mathrm{t}}[\![\tau]\!] && \text{⟅definition of } \textstyle\bigcup\text{⟆} \\
= \ & \boldsymbol{\mathcal{S}}_{\mathrm{t}}[\![\tau]\!] && \text{⟅definition of } \boldsymbol{\mathcal{S}}_{\mathrm{t}}[\![\tau]\!] \text{ in (44.54)⟆}
\end{aligned}
$$

which is $\mathsf{lfp}^{\subseteq} X \mapsto \mathbb{S}^1 \cup \vec{\tau} \mathbin{\frown} X$ by Scott–Kleene's iterative fixpoint theorem 15.26 knowing that $X \mapsto \mathbb{S}^1 \cup \vec{\tau} \mathbin{\frown} X$ is preserves nonempty joins and therefore is continuous and $\langle \mathbb{S}^*, \subseteq\rangle$ is a complete lattice hence a CPO. □

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

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

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. $\square$

Solution to Exercise 44.63

— $\alpha^{\mathcal{T}}(\mathbb{S}^1 \cup \vec{\tau} \,\widehat{\ }\, X)\langle\sigma,\ \Sigma\rangle$

$= \alpha^{\mathcal{T}}(\mathbb{S}^1)\langle\sigma,\ \Sigma\rangle \wedge \alpha^{\mathcal{T}}(\vec{\tau} \,\widehat{\ }\, X)\langle\sigma,\ \Sigma\rangle$ (X ↦ $\alpha^{\mathcal{T}}(X)\langle\sigma,\ \Sigma\rangle$ preserves joins) (A)

The first term of (A) is

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

The second term of (A) is

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

$= \alpha^{\mathcal{T}}(\{\sigma'\sigma''\pi \mid \langle\sigma',\ \sigma''\rangle \in \tau \wedge \sigma''\pi \in X\})\langle\sigma,\ \Sigma\rangle$ $\wr$definitions of $\vec{\tau}$ and $\frown$ in exercise 44.53$\int$

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

$= \{\sigma'\sigma''\pi \mid \sigma' = \sigma \wedge \langle\sigma',\ \sigma''\rangle \in \tau\wedge\sigma''\pi \in X\} \subseteq \alpha^{\mathbb{T}}(\{P \in \boldsymbol{\mathcal{S}}[\![T]\!] \mid P_0 = \Sigma\})$ $\wr$definition of $\in$$\int$

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

$\wr$definition (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\}$$\int$

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

$\wr$definition of $\subseteq$ where $\pi' = \sigma\sigma''\pi$$\int$

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

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

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

$\wr$definitions of $\bigcup$ and $\subseteq$$\int$

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

$\wr$letting $\pi' = \sigma''\pi$ and $P = \Sigma''P'$$\int$

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

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

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

because

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

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

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

$\wr$definition (44.58) of $\alpha^{\mathbb{T}}$$\int$

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

$\wr$ def (44.57) of $\alpha^{\mathbb{T}}$ $\int$

$= \{\pi \in X \mid \pi_0 = \sigma''\} \subseteq \{\pi \in \mathbb{S}^* \mid \exists P \in \boldsymbol{\mathcal{S}}[\![T]\!]\;.\;P_0 = \Sigma'' \wedge \forall i \in [0,|\pi|[\;.\;\pi_i \in P_i\}$ $\wr$ definitions of $\in$ and $\cup$ $\int$

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

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

— We have $\alpha^{\mathcal{T}}(\langle\sigma,\ \Sigma\rangle \mapsto \emptyset) = \langle\sigma,\ \Sigma\rangle \mapsto \mathit{ff}$ and commutation, as shown previously, so by the exact fixpoint abstraction theorem 18.22 in a complete lattice, we have

mc

$\triangleq \alpha^{\mathcal{T}}(\boldsymbol{\mathcal{S}}_{\mathsf{t}}[\![\tau]\!])$ $\wr$ definition (44.64) of mc $\int$

$= \alpha^{\mathcal{T}}(\mathsf{lfp}^{\subseteq}\, X \mapsto \mathbb{S}^1 \cup \vec{\tau} \mathbin{\hat{\,}} X)$ $\wr$ exercise 44.53 $\int$

$= \mathsf{lfp}^{\Leftarrow}\, X \mapsto \langle\sigma,\ \Sigma\rangle \mapsto \big((\sigma \in \Sigma) \wedge \bigwedge_{\langle\sigma,\sigma'\rangle\in\tau} \bigvee_{\langle\Sigma,\Sigma'\rangle\in T} X(\sigma',\Sigma')\big)$ $\wr$ exercise 44.60 and theorem 18.22 $\int$

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

□

17 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 47.14 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, because 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 47.43

Proof of (47.42)

$\alpha^{\mathrm{d}}(\{\mathcal{S}^{+\infty}[\![\mathtt{S}]\!]\})(\ell)$

$= \alpha^{\mathrm{d}}(\{\mathcal{S}^{*}[\![\mathtt{S}]\!]\})\,\ell$ ⟅lemma 47.23⟆

$= \{\langle \mathtt{x}',\ \mathtt{y}\rangle \mid \mathcal{S}^{*}[\![\mathtt{S}]\!] \in \mathcal{D}(\ell)\langle \mathtt{x}',\ \mathtt{y}\rangle\}$ ⟅definition (47.25) of α^{d}⟆

$= \{\langle \mathtt{x}',\ \mathtt{y}\rangle \mid \exists\langle\pi_0,\ \pi_1\rangle, \langle\pi_0',\ \pi_1'\rangle \in \mathcal{S}^{*}[\![\mathtt{S}]\!]\ .\ (\forall \mathtt{z} \in \mathbb{V} \setminus \{\mathtt{x}'\}\ .\ \varrho(\pi_0)\mathtt{z} = \varrho(\pi_0')\mathtt{z}) \wedge \mathsf{diff}(\mathsf{seqval}[\![\mathtt{y}]\!](\ell)(\pi_0, \pi_1), \mathsf{seqval}[\![\mathtt{y}]\!](\ell)(\pi_0', \pi_1'))\}$ ⟅definition (47.19) of $\mathcal{D}(\ell)\langle \mathtt{x}',\ \mathtt{y}\rangle$⟆

$= \{\langle \mathtt{x}',\ \mathtt{y}\rangle \mid \exists\langle\pi_0,\ \pi_1\rangle, \langle\pi_0',\ \pi_1'\rangle \in \mathcal{S}^{*}[\![\mathtt{S}]\!]\ .\ (\forall \mathtt{z} \in \mathbb{V} \setminus \{\mathtt{x}'\}\ .\ \varrho(\pi_0)\mathtt{z} = \varrho(\pi_0')\mathtt{z}) \wedge \mathsf{diff}(\ni, \ni)\}$

⟅definition of $\mathcal{S}^{*}[\![\mathtt{S}]\!]$ so that if $\langle\pi,\ \pi'\rangle \in \mathcal{S}^{*}[\![\mathtt{S}]\!]$ then π ends $\mathsf{at}[\![\mathtt{S}]\!]$ and π' contains only labels of $\mathsf{labx}[\![\mathtt{S}]\!]$ so that, definition (47.16) of $\mathsf{seqval}[\![\mathtt{y}]\!]$, $\mathsf{seqval}[\![\mathtt{y}]\!](\ell)(\pi_0, \pi_1) = \mathsf{seqval}[\![\mathtt{y}]\!](\ell)(\pi_0', \pi_1') = \ni$⟆

$= \emptyset$ ⟅definition (47.18) of $\mathsf{diff}(\omega, \omega')$ which implies $\omega \neq \ni$ and $\omega' \neq \ni$⟆ □

□

Solution to Exercise 47.61 $\widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\mathtt{Sl}]\!]\,\ell_2 = \{\langle \mathtt{x},\ \mathtt{y}\rangle\} \cup \{\langle \mathtt{z},\ \mathtt{z}\rangle \mid \mathtt{z} \in \mathbb{V} \setminus \{\mathtt{y}\}\}$. 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, for example, by a linear equality analysis of chapter 38). □

Solution to Exercise 47.66

$\widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\mathtt{S}_b]\!]\,\ell_0$

$= \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\{\ell_1\ \mathtt{y = z\ ;}\ell_2\ \mathtt{z = x\ ;}\}]\!]\,\ell_0$ ⟅definition of $\mathtt{S}_b$⟆

$= \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\ell_1\ \mathtt{y = z\ ;}\ell_2\ \mathtt{z = x\ ;}]\!]\,\ell_0$ ⟅compound statement (47.57)⟆

$= \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\ell_1\ \mathtt{y = z\ ;}]\!]\,\ell_2 \mathbin{⨾} \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\ell_2\ \mathtt{z = x\ ;}]\!]\,\ell_0$ ⟅(47.60.b) where $\mathtt{Sl}' = \ell_1\ \mathtt{y = z\ ;}$⟆

$= \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\,\epsilon\,]\!]\,\ell_1 \mathbin{⨾} \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\ell_1\ \mathtt{y = z\ ;}]\!]\,\ell_2 \mathbin{⨾} \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\ell_2\ \mathtt{z = x\ ;}]\!]\,\ell_0$ ⟅(47.60.b) where $\mathtt{Sl}' = \epsilon\ \ell_1$⟆

$= \mathbb{1}_{\mathbb{V}} \mathbin{⨾} \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\ell_1\ \mathtt{y = z\ ;}]\!]\,\ell_2 \mathbin{⨾} \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\ell_2\ \mathtt{z = x\ ;}]\!]\,\ell_0$ ⟅(47.54)⟆

$= \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\ell_1\ \mathtt{y = z\ ;}]\!]\,\ell_2 \mathbin{⨾} \widehat{\overline{\mathcal{S}}}^{\exists}_{\mathsf{diff}}[\![\ell_2\ \mathtt{z = x\ ;}]\!]\,\ell_0$ ⟅$\mathbb{1}_{\mathbb{V}}$ neutral element of $⨾$⟆

$= \{\langle \mathtt{z},\ \mathtt{y}\rangle, \langle \mathtt{z},\ \mathtt{z}\rangle, \langle \mathtt{x},\ \mathtt{x}\rangle\} \mathbin{⨾} \{\langle \mathtt{x},\ \mathtt{z}\rangle, \langle \mathtt{x},\ \mathtt{x}\rangle, \langle \mathtt{y},\ \mathtt{y}\rangle\}$ ⟅(47.44)⟆

$= \{\langle \mathtt{x},\ \mathtt{x}\rangle, \langle \mathtt{x},\ \mathtt{z}\rangle, \langle \mathtt{z},\ \mathtt{y}\rangle\}$ ⟅definition of $⨾$⟆

□

18 Solutions to Selected Exercises of Chapter 48

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

- If $\boldsymbol{\tau}' = \alpha \in \mathbb{V}_{\mathfrak{t}}$ then $\{\langle\alpha,\ \boldsymbol{\tau}\rangle\}(\boldsymbol{\tau}') = \{\langle\alpha,\ \boldsymbol{\tau}\rangle\}(\alpha) = \boldsymbol{\tau}$ by definition of function application. On the other hand, $\boldsymbol{\tau}[\alpha \leftarrow \boldsymbol{\tau}'] = \boldsymbol{\tau}[\alpha \leftarrow \alpha] = \boldsymbol{\tau}$ by (48.5);
- If $\alpha \neq \boldsymbol{\tau}' = \beta \in \mathbb{V}_{\mathfrak{t}}$ 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), induction hypothesis, and (48.5), we have $\{\langle\alpha,\ \boldsymbol{\tau}\rangle\}(\boldsymbol{\tau}') = \{\langle\alpha,\ \boldsymbol{\tau}\rangle\}(f(\boldsymbol{\tau}'_1, \ldots, \boldsymbol{\tau}'_n)) = f(\{\langle\alpha,\ \boldsymbol{\tau}\rangle\}(\boldsymbol{\tau}'_1), \ldots, \{\langle\alpha,\ \boldsymbol{\tau}\rangle\}(\boldsymbol{\tau}'_n)) = f(\boldsymbol{\tau}'_1[\alpha \leftarrow \boldsymbol{\tau}], \ldots, \boldsymbol{\tau}'_n[\alpha \leftarrow \boldsymbol{\tau}]) = f(\boldsymbol{\tau}'_1, \ldots, \boldsymbol{\tau}'_n)[\alpha \leftarrow \boldsymbol{\tau}] = \boldsymbol{\tau}'[\alpha \leftarrow \boldsymbol{\tau}]$.

□

19 Solutions to Selected Exercises of Chapter 49

Solution to Exercise 49.2

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

□

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

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

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

Solution to Exercise 49.33 `hd([])` is a definite dynamic error so can be rejected $\dfrac{\boldsymbol{\Gamma} \vdash \mathtt{S} : \boldsymbol{\mu}\ \mathtt{list},\ \mathtt{S} \neq [\,]}{\boldsymbol{\Gamma} \vdash \mathsf{hd}(\mathtt{S}) : \boldsymbol{\mu}}$.
Of course, this refinement is endless because $(\mathtt{hd} \mid \mathtt{tl})^{*}([\,])$ also definitely yield a dynamic error. More generally, a static analysis would be useful. □

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)
```

□

20 Solutions to Selected Exercises of Chapter 50

Solution to Exercise 50.37 The result of the static analysis

```
while l1: (n > 0) {n:_|_}
   l2: {n:_|_} n = (n - 1);
l3: {n:_|_}
```

states that the invariance specification is unsatisfiable (no execution can reach a program point ℓ in a state satisfying $\mathcal{R}_f(\ell)$). $\square$

Solution to Exercise 50.53 We could define $\mathcal{S}^{\tilde{\overleftarrow{\varrho}}}[\![\mathsf{S}]\!] \triangleq \varnothing$ for noncompilable programs. Then $\forall \pi \,.\, \mathcal{S}[\![\mathsf{S}]\!]\,\pi = \varnothing$ implies $\mathcal{S}^{\tilde{\overleftarrow{\varrho}}}[\![\mathsf{S}]\!]\,\mathcal{R}_f = \mathbb{E}\mathbb{v}^{\varrho} \not\subseteq \mathcal{S}^{\overleftarrow{\varrho}}[\![\mathsf{S}]\!]\,\mathcal{R}_f = \varnothing$. However, for the semantics of chapter 6, “Structural Deductive Stateless Prefix Trace Semantics,” and chapter 7, “Maximal Trace Semantics,” we always have $\forall \pi \,.\, \mathcal{S}[\![\mathsf{S}]\!]\,\pi \neq \varnothing$ and so $\widehat{\mathcal{S}}^{\tilde{\overleftarrow{\varrho}}}[\![\mathsf{S}]\!]\,\mathcal{R}_f \subseteq \widehat{\mathcal{S}}^{\overleftarrow{\varrho}}[\![\mathsf{S}]\!]\,\mathcal{R}_f$. $\square$

21 Solutions to Selected Exercises of Chapter 51

Solution to Exercise 51.6

```
*** Labeled 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}]>
```

$\square$

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). □

22 Bibliography

[1] Richard Dedekind. *Stetigkeit und irrationale Zahlen.* Braunschweig : F. Vieweg, 1892 (11).

[2] Philippe Granger. "Static Analysis of Arithmetical Congruences." *International Journal of Computer Mathematics.* 30.3 & 4 (1989), pp. 165–190 (382, 384, 21).

[3] Holbrook M. MacNeille. "Partially Ordered Sets." *Trans. Amer. Math. Soc.* 42.3 (1937), pp. 416–460 (11).