Mini-Exercise 1

Mini-Solution 1.1

Evaluate $A \cap \text{elems}(\text{trace}) = \emptyset$
in environment $\{A \mapsto \{2, 4, 6\}, \text{trace} \mapsto \{1, 2, 1, 3\}\}$ where

- $\text{elems}$ takes a list and returns the set of values in that list

Solution

\[
\begin{align*}
A \cap \text{elems}(\text{trace}) &= \emptyset \\
&= " \text{lookup } A \text{ and } \text{trace} \text{ in environment} " \\
\{2, 4, 6\} \cap \text{elems}((1, 2, 1, 3)) &= \emptyset \\
&= " \text{behaviour of } \text{elems} " \\
\{2, 4, 6\} \cap \{1, 2, 3\} &= \emptyset \\
&= " \text{behaviour of } \cap (\text{rhs}) " \\
\{2\} &= \emptyset \\
&= " \text{clearly not equal} " \\
&= \text{False}
\end{align*}
\]

Mini-Solution 1.2

Is $\exists n : \mathbb{N} \cdot n \geq 3 \land \text{prime}(n) \Rightarrow \text{prime}(n - 1)$ true or false? Justify your answer.

Solution

- True.
- It succeeds for $n = 3$

\[
\begin{align*}
3 \geq 3 \land \text{prime}(3) &\Rightarrow \text{prime}(3 - 1) \\
3 \geq 3 \land \text{prime}(3) &\Rightarrow \text{prime}(2) \\
\text{True} \land \text{True} &\Rightarrow \text{prime}(2) \\
\text{True} &\Rightarrow \text{True} \\
\text{True} &
\end{align*}
\]

- The overall predicate succeeds because it only has to be true for one $n : \mathbb{N}$

Mini-Solution 1.3

Write one paragraph about a software failure (not yours) that you found most vexing.

Solution

Don't get me started! . . .
Mini-Exercise 2

Mini-Solution 2.1

For the following predicate, state for every variable occurrence if it is free, binding or bound:

\[(\forall a \bullet (\forall b \bullet (\exists c \bullet a \land c \lor b) \land (c \lor \neg a)) \lor b)\]

Solution

• using colour:

\[(\forall a \bullet (\forall b \bullet (\exists c \bullet a \land c \lor b) \land (c \lor \neg a)) \lor b)\]

• using labelling:

\[(\forall a \bullet (\forall b \bullet (\exists c \bullet a \land c \lor b) \land (c \lor \neg a)) \lor b)\]

\[
\begin{array}{cccccccc}
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & F & F \\
B & B & B & B & B & F & B & R \\
D & D & D & N & N & E & N & E \\
G & G & G & D & D & E & D & E \\
\end{array}
\]

Issues raised by Mini-exercise 2.1

• There was some misunderstanding of the scope of a quantifier.
  It extends as far to the right as possible
  It ends at an enclosing right-bracket

• Consider:

  \(\forall b \bullet c \land (\exists c \bullet (b \Rightarrow c) \land (\forall d \bullet c \lor e \land b))\)

  – The starting \(\forall b\) covers everything
  – The \(\exists c\) covers up to inside the last bracket
  – The \(\forall d\) covers up to inside the 2nd-last bracket

• Now consider

\[(\forall b \bullet c) \land (\exists c \bullet (b \Rightarrow c) \land (\forall d \bullet c \lor e)) \land b\]

  – The starting \(\forall b\) covers to just after the 1st \(c\)
  – The \(\exists c\) covers to inside the 2nd bracket after \(e\)
  – The \(\forall d\) covers to inside the 1st bracket after \(e\)
Mini-Solution 2.2

Perform the following (simultaneous, double) substitution:

$$(\exists t, t' \cdot t' - t = tr' - tr)[t \triangleright tr', t' \triangleright tr' / tr, tr']$$

Here $tr$ and $tr'$ are different variables.

Solution

$$(\exists t, t' \cdot t' - t = tr' - tr)[t \triangleright tr', t' \triangleright tr' / tr, tr']$$

=  

“free $t$, $t'$ will be captured, so need to $\alpha$-rename $t$, $t'$”

$$(\exists t, t' \cdot t' - t = tr' - tr)\{t \mapsto s\}\{t' \mapsto s'\}$$

=  

“do $\alpha$-rename”

$$(\exists s, s' \cdot s' - s = tr' - tr)[t \triangleright tr', t' \triangleright tr' / tr, tr']$$

=  

“do free substitution”

$$(\exists s, s' \cdot s' - s = (t' \triangleright tr') - (t \triangleright tr))$$

Issues raised by Mini-exercise 2.2

- we can’t immediately do the substitution, or we risk name-capture of $s$. We need to do $\alpha$-renaming first.

- In general a simultaneous substitution $[E, F/x, y]$ cannot be done in several stages, i.e. $[E/x]$ followed by $[F/y]$

Consider:

$$(x + y)[y^2/x][k/y] = (x + y)[y^2, k/x, y] = (y^2 + y)[k/y] = k^2 + k \neq y^2 + k$$

The two things are equivalent only when $y \notin E$, which was the case in this exercise.
Mini-Solution 2.3

The proof of ⟨∃-trading⟩ has no justifications—fill them in.

Solution

\[ \exists x | R \bullet P \]

\[ = \text{“ ⟨gen-deMorgan⟩ ”} \]

\[ \neg (\forall x | R \bullet \neg P) \]

\[ = \text{“ ⟨∀-trading⟩ ”} \]

\[ \neg (\forall x \bullet R \Rightarrow \neg P) \]

\[ = \text{“ ⟨⇒-def⟩ ”} \]

\[ \neg (\forall x \bullet \neg (R \lor P)) \]

\[ = \text{“ ⟨deMorgan⟩ ”} \]

\[ \neg (\forall x \bullet \neg (R \land P)) \]

\[ = \text{“ ⟨gen-deMorgan⟩ ”} \]

\[ \exists x \bullet R \land P \]

Strictly speaking, the last step above is incorrect, as ⟨gen-deMorgan⟩ applies to predicates with ranges.

\[ \neg (\forall x \bullet \neg (R \land P)) \]

\[ = \text{“ ⟨Λ-unit⟩ ”} \]

\[ \neg (\forall x \bullet \text{true} \land \neg (R \land P)) \]

\[ = \text{“ ⟨∀-trading⟩ ”} \]

\[ \neg (\forall x | \text{true} \bullet \neg (R \land P)) \]

\[ = \text{“ ⟨gen-deMorgan⟩ ”} \]

\[ \exists x | \text{true} \bullet R \land P \]

\[ = \text{“ ⟨∀-trading⟩ ”} \]

\[ \exists x \bullet \text{true} \land R \land P \]

\[ = \text{“ ⟨Λ-unit⟩ ”} \]

\[ \exists x \bullet R \land P \]
Mini-Exercise 3

Mini-Solution 3.1

Expand out and simplify the predicate definition of the following program fragment

\[ f := f \cdot x; \ x := x - 1 \]

\[ f := f \cdot x; \ x := x - 1 \]

\[ f' = f \cdot x \wedge x' = x; \ x := x - 1 \]

\[ f' = f \cdot x \wedge x' = x; \ x' := x - 1 \wedge f' = f \]

\[ \exists f_m, x_m \cdot \left( f' = f \cdot x \wedge x' = x \right) \left[ f_m, x_m / f, x \right] \]

\[ \wedge \left( x' := x - 1 \wedge f' = f \right) \left[ f_m, x_m / f, x \right] \]

\[ \exists f_m, x_m \cdot f = f \cdot x \wedge x := x - 1 \wedge f' = f \]

\[ x := e; \ x := f \]

\[ x' = e \wedge \nu' = \nu; \ x' = f \wedge \nu' = \nu \]

\[ x' = e \wedge \nu' = \nu; \ x := f \wedge \nu' = \nu \]

\[ \exists x_m, \nu_m \cdot (x' = e \wedge \nu' = \nu) \left[ x_m, \nu_m / x', \nu' \right] \]

\[ \wedge \left( x' = f \wedge \nu' = \nu \right) \left[ x_m, \nu_m / x, \nu \right] \]

\[ \exists x_m, \nu_m \cdot x_m = e \wedge \nu_m = \nu \wedge x' = f \left[ x_m, \nu_m / x, \nu \right] \wedge \nu' = \nu_m \]

\[ x' = f \left[ e, \nu / x, \nu \right] \wedge \nu' = \nu \]

\[ x := f \left[ e / x \right] \]
Mini-Exercise 4

0.0.1 Mini-Solution 4.1

\[ S \sqsubseteq P \lor Q \equiv (S \sqsubseteq P) \land (S \sqsubseteq Q) \]  \hspace{1cm} \text{\small \textcircled{\text{-prog-alt}}} \\

Solution  Goal: \[ S \sqsubseteq P \lor Q \equiv (S \sqsubseteq P) \land (S \sqsubseteq Q) \]

Strategy: reduce rhs to lhs

\[
(S \sqsubseteq P) \land (S \sqsubseteq Q) \\
= \text{" \small \textcircled{\text{-def} \text{"} \}} \\
[P \Rightarrow S] \land [Q \Rightarrow S] \\
= \text{" \small \textcircled{\text{[]-split} \text{"} \}} \\
[P \Rightarrow S \land Q \Rightarrow S] \\
= \text{" \small \textcircled{\text{ante-\lor-distr} \text{"} \}} \\
[P \lor Q \Rightarrow S] \\
= \text{" \small \textcircled{\text{-def} \text{"} \}} \\
S \sqsubseteq P \lor Q
\]
0.0.2 Mini-Solution 4.2

Show that \( x := -y \); \( y := -x \) refines \( x \vdash [x' = -y] \)

**Solution**  

Goal: \( x \vdash [x' = -y] \) \( \sqsubseteq \) \( x := -y \); \( y := -x \)  

Strategy: reduce to true  

\[
x \vdash [x' = -y] \quad \sqsubseteq \quad x := -y \; ; \; y := -x
\]

\[
\begin{align*}
x &\vdash [x' = -y] \quad \sqsubseteq \quad x := -y \; ; \; y := -x \\
&= \quad \text{`sim-:=merge\()\} \}
\end{align*}
\]

\[
x \vdash [x' = -y] \quad \sqsubseteq \quad x, y := -y, -x[-y/x]
\]

\[
\begin{align*}
x &\vdash [x' = -y] \quad \sqsubseteq \quad x, y := -y, -(-y)
&= \quad \text{`substitution\} \}
\end{align*}
\]

\[
x \vdash [x' = -y] \quad \sqsubseteq \quad x' = -y \land y' = -(y)
&= \quad \text{`sim-:=def\} \}
\end{align*}
\]

\[
x' = -y \land y' = y \quad \sqsubseteq \quad x' = -y \land y' = y
\]

\[
\begin{align*}
x' &\vdash -y \land y' = y \quad \sqsubseteq \quad x' = -y \land y' = y
&= \quad \text{`frame-def\} \}
\end{align*}
\]

\[
\begin{align*}
&\sqsubseteq \quad x' = -y \land y' = y
&= \quad \text{`\sqsubseteq-refl\} \}
\end{align*}
\]

true

Issues raised by Mini-exercise 4.2

- When expanding \( x := -y \); \( y := -x \) many people obtained:

\[
x' = -y \land y' = \nu \quad ; \quad y' = -x \land \nu' = \nu
\]

The problem here is that \( \nu \) in the first assignment contains \( y \), among possible other variables, whilst in the second assignment it contains \( x \) with other variables (and similarly for \( \nu' \)). The correct expansion would be:

\[
x' = -y \land y' = y \land \nu' = \nu \quad ; \quad y' = -x \land x' = x \land \nu' = \nu
\]

—or, even more pedantically:

\[
x' = -y \land \nu_1 = \nu_1 \quad ; \quad y' = -x \land \nu_2 = \nu_2
\]

\[
A = \{x, x', \nu_1, \nu_1'\} = \{y, y', \nu_2, \nu_2'\}
\]
Mini-Exercise 5

Q5 Given

\[
\begin{align*}
    n, f, x & : \mathbb{N} \\
    \text{fac}(0) & = 1 \\
    \text{fac}(n) & = n \cdot \text{fac}(n-1), \quad n > 0 \\
\end{align*}
\]

\[
\begin{align*}
    F\text{Spec} & \triangleq f, x : [f' = \text{fac}(n)] \\
    F\text{Prog} & \triangleq f, x := 1, 2; (x \leq n) \ast f, x := f \ast x, x + 1
\end{align*}
\]

5.1 State clearly the proofs that need to be done.

5.2 Determine a suitable invariant.

5.3 Prove that statement (one of the proofs in the answer to 5.1) that \( F\text{Spec} \) is refined appropriately.

0.0.3 Mini-Solution 5

Mini-Solution 5.1, the proofs

Given appropriate invariant \( \text{finv} \), we need to

1. Prove \( \bar{v} : [S] \sqsubseteq \bar{v} : [\text{pre} \Rightarrow \text{inv} \land \neg \text{c}] \), i.e.:

\[
f, x : [f' = \text{fac}(n)] \sqsubseteq f, x : [\text{true} \Rightarrow \text{finv} \land \neg (x' \leq n')]
\]

2. Prove \( \bar{v} : [\text{pre} \Rightarrow \text{inv}'] \sqsubseteq \text{init} \), i.e.:

\[
f, x : [\text{true} \Rightarrow \text{finv}'] \sqsubseteq f, x := 1, 2
\]

3. Prove \( \bar{v} : [\text{c} \land \text{inv} \Rightarrow \text{finv}'] \sqsubseteq P \), i.e.:

\[
f, x : [x \leq n \land \text{finv} \Rightarrow \text{finv}'] \sqsubseteq f, x := f \ast x, x + 1
\]

Mini-Solution 5.2, The invariant

Looking at loop execution (again!)

<table>
<thead>
<tr>
<th>( x )</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>( k )</th>
<th>( n )</th>
<th>( n + 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f )</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>\text{fac}(k-1)</td>
<td>\text{fac}(n-1)</td>
<td>\text{fac}(n)</td>
</tr>
</tbody>
</table>

We need to get \( \text{fac}(n) \) into the picture

We can see that \( \text{fac}(n) = \text{fac}(x-1) \ast x \ast (x+1) \ast \cdots \ast n \), so let’s introduce a shorthand, with \( F(x, n) = x \ast (x+1) \ast \cdots \ast n \), satisfying:

\[
\begin{align*}
    F(\ell, u) & \triangleq 1, \quad \ell > u \\
    F(x, x) & \triangleq x \\
    F(\ell, u) & \triangleq \ell \ast F(\ell + 1, u), \quad \ell < u
\end{align*}
\]

Now, given \( f = \text{fac}(x-1) \), our invariant is now

\[
\begin{align*}
    \text{finv} & \triangleq f \ast F(x, n) = \text{fac}(n)
\end{align*}
\]
Mini-Solution 5.3, *FSpec ok*

*FSpec ⊑ f, x : [finv' ∧ x' > n']*

\[
f, x : [f' = fac(n)] ⊑ f, x : [f' * F(x', n') = fac(n') ∧ x' > n']
\]

= “⟨⟨ frame-def ⟩⟩(twice) ”

\[
f' = fac(n) ∧ n' = n
\]

= “defn. *F*, given x' > n' ”

\[
f' = fac(n) ∧ n' = n
\]

= “arithmetic”

\[
f' = fac(n) ∧ n' = n
\]

= “⟨⟨ prog-strengthen ⟩⟩”

\[
true
\]
Mini-Exercise 6

Q6.0 Install SAOITHÍN (v0.90α4)

Q6.1 Use it to prove conjectures 1–10 (1 – 4 done in class, figure out 5–10)

Paper Submission .txt files generated (per proof).

Electronic Submission .txt files generated (per proof), plus final version of GS3001.teoric
Proof of ⟨01 TRUE⟩

True

Reduce to True.

True

□

Proof of ⟨02 ≡-refl⟩

\[ P \equiv P \]

Reduce to True.

\[ P \equiv P \]
\[ \equiv \text{“GS3BA31}$2$≡-Unit:REqv”} \]

True

□
Proof of \( \langle \langle 03 \neg \equiv \text{Swap} \rangle \rangle \)

\[(\neg P \equiv Q) \equiv (P \equiv \neg Q)\]

Reduce to True.

\[
(\neg P \equiv Q) \equiv (P \equiv \neg Q)
\]

\[
\equiv \quad \text{" GS3BA31$0$\neg \equiv \text{Distr:REqv } @1 "}
\]

\[
\neg (P \equiv Q) \equiv (P \equiv \neg Q)
\]

\[
\equiv \quad \text{" GS3BA31$0$\equiv \text{Comm:LEqv } @2 "}
\]

\[
\neg (P \equiv Q) \equiv (\neg Q \equiv P)
\]

\[
\equiv \quad \text{" GS3BA31$0$\neg \equiv \text{Distr:REqv } @2 "}
\]

\[
\neg (P \equiv Q) \equiv (\neg (Q \equiv P))
\]

\[
\equiv \quad \text{" GS3BA31$0$\equiv \text{Comm:LEqv } @2.1 "}
\]

\[
\neg (P \equiv Q) \equiv (\neg (P \equiv Q))
\]

\[
\equiv \quad \text{" GS3BA31$0$\equiv \text{Unit:REqv } "}
\]

\[True\]

\[
\Box
\]

Proof of \( \langle \langle 04 \neg \text{-Invol} \rangle \rangle \)

\[-\neg P \equiv P\]

Reduce to True.

\[
\neg \neg P \equiv P
\]

\[
\equiv \quad \text{" GS3BA31$3$\neg \equiv \text{-Swap:LEqv } "}
\]

\[
\neg P \equiv \neg P
\]

\[
\equiv \quad \text{" GS3BA31$3$\equiv \text{-Unit:REqv } "}
\]

\[True\]

\[
\Box
\]