EPFL Formal Verification Course Exam Solutions for October 2022 Exam

The exam is from 15:15 to 18:00. Do not open the exam until we tell you to.
Place your CAMIPRO card on your desk.
Put all electronic devices in a bag away from bench.
Write using permanent pen (no graphite nor heat-disappearing pen).

Write answers to different problems (1–6) on disjoint sheets of white paper that we supply.
Write your name, SCIPER and question number on top of each sheet you return.
Do not write the solutions that you want us to grade on the sheets with exam questions; please take these printed exam sheets with you after the exam.

Each subquestion is scored independently. Wrong answers do not penalize other parts.

We advise you to first solve questions that you find easier. If you expect you are running out of time on a particular problem, try to convince us that you know the right strategy to solve it.

You are allowed to use every true statement (e.g. theorem, equality) from the lecture slides provided that you clearly repeat it and refer to it as "seen in the lecture" (preferably with slide title or lecture name).

The exam is open book in the sense that you are allowed to take with you any printed material. Please do not take slides that are hand-annotated but only original ones or printed annotations.

The maximal number of points on the exam is 40.

Small reminder: the following are the names of some basic properties of a relation r:

- reflexive (R): $\forall x.\ (x,x) \in r$
- antisymmetric (A): $\forall x. \forall y.\ (x,y) \in r \land (y,x) \in r \to x = y$
- symmetric (S): $\forall x. \forall y.\ (x,y) \in r \to (y,x) \in r$
- transitive (T): $\forall x. \forall y. \forall z.\ (x,y) \in r \land (y,z) \in r \to (x,z) \in r$

Equivalence relation is a relation that is reflexive, symmetric, and transitive.
Partial ordering relation is a relation that is reflexive, antisymmetric, and transitive.

1 Weakest Precondition (3pt)

Let $r \subseteq S \times S$ and $Q \subseteq S$. Give an expression defining weakest precondition $\mathsf{wp}(r, Q)$ using operations of inverse of a relation, $(\_^{-1})$, set difference $(\setminus)$, and image of a relation under a set, $(\_[\_])$. Prove that your expression is correct by expanding the definitions of wp as well as of relational and set operations.

Solution: *We claim that*

$$wp(r, Q) = S \setminus r^{-1}[S \setminus Q]$$

To prove this, we expand both sides and use basic rules of first-order logic, obtaining identical set comprehensions.

$$wp(r, Q) = \{s \in S \mid \forall s' \in S.((s, s') \in r \rightarrow s' \in Q)\}$$

$$\begin{aligned}
S \setminus r^{-1}[S \setminus Q] &= S \setminus \{s \in S \mid \exists s' \in S \setminus Q.(s', s) \in r^{-1}\} \\
&= S \setminus \{s \in S \mid \exists s' \in S \setminus Q.(s, s') \in r\} \\
&= \{s \in S \mid \neg(\exists s' \in S \setminus Q.(s, s') \in r)\} \\
&= \{s \in S \mid \neg(\exists s' \in S.\ (s' \notin Q \land (s, s') \in r))\} \\
&= \{s \in S \mid \forall s' \in S.\ (s' \in Q \lor \neg((s, s') \in r)))\} \\
&= \{s \in S \mid \forall s' \in S.\ ((s, s') \in r \rightarrow s' \in Q)\}
\end{aligned}$$

A less detailed proof would also be acceptable. $\Diamond$

2 Iterating a Relation (7pt)

Let $M = (S, I, r, A)$ be a transition system and $\bar{r} = \{(s, s') \mid (s, a, s') \in r\}$, as usual. Let $\Delta = \{(x, x) \mid x \in S\}$.

Let $\bar{r}^k$ denote the usual composition of relation $\bar{r}$ with itself k times.

Define the sequence of relations r_n, for all non-negative integers n, as follows:

- $r_0 = \Delta \cup \bar{r}$
- $r_{n+1} = r_n \circ r_n$

A) (2pt) Prove that $r_n \subseteq r_{n+1}$ for every n. **Solution:** *Here we remark that $\Delta \subseteq r_n$ for every $n \geq 0$. The proof is by simple induction. In the base case, $\Delta \subseteq \Delta \cup \bar{r} = r_0$. Next, as an inductive hypothesis assume $\Delta \subseteq r_n$. Then*

$$r_{n+1} = r_n \circ r_n \supseteq \Delta \circ \Delta = \Delta$$

by monotonicity of the relation composition $\circ$. Then,

$$r_{n+1} = r_n \circ r_n \supseteq r_n \circ \Delta = r_n$$

$\Diamond$

B) (3pt) Prove that for every n and every k where $0 \leq k \leq 2^n$ we have $\bar{r}^k \subseteq r_n$. **Solution:** *The proof is by induction on n.*

Consider $n = 0$, so $2^n = 1$. we have $k = 0$ or $k = 1$. We have $\bar{r}^0 = \Delta \subseteq r_0$ and $\bar{r}^1 = \bar{r} \subseteq r_0$ by definition of r_0.

Next, suppose the property holds for n and we aim to show it for $n+1$. Let $0 \leq k \leq 2^{n+1}$. If $k \leq 2^n$ then by inductive hypothesis and property A),

$$\bar{r}^k \subseteq r_n \subseteq r_{n+1}$$

as desired. Thus, consider $2^n + 1 \leq k \leq 2^{n+1}$. Let $k_1 = 2^n$ and $k_2 = k - k_1$. We have that $0 \leq k_1 \leq 2^n$ and $0 \leq k_2 = k - k_1 \leq 2^{n+1} - 2^n = 2^n$. Thus

$$\bar{r}^k = \bar{r}^{k_1+k_2} = \bar{r}^{k_1} \circ \bar{r}^{k_2} \subseteq r_n \circ r_n = r_{n+1}$$

again by monotonicity of $\circ$. $\diamondsuit$

C) (2pt) Suppose that S is finite. Find a bound B as a function of $|S|$ such that

$$\mathsf{Reach}(M) \subseteq r_B[I]$$

Aim to find as small bound as possible. **Solution:** *The set of reachable states is the set of final states of traces of the system M. Consider a trace that has a repeated state,*

$$s_0, a_0, s_1, \ldots, s_{i-1}, a_{i-1}, s_i, a_i, s_{i+1}, \ldots, s_{i+k}, a_{i+k}, \ldots, s_n$$

where $s_i = s_{i+k}$. Then the shortened sequence is also a valid trace:

$$s_0, a_0, s_1, \ldots, s_{i-1}, a_{i-1}, s_{i+k}, a_{i+k}, \ldots, s_n$$

which has the same final state s_n. Thus, the set of reachable states is also the set of final states of traces where no state repeats. Such traces can have at most $|S|$ states. Therefore,

$$\textit{Reach(M)} = \left(\bigcup_{k=0}^{|S|} (\bar{r})^k \right) [I]$$

Let $B = \lceil \log_2 |S| \rceil$ so that $|S| \leq 2^B$. For any $k \leq |S| \leq 2^B$ we have $\bar{r}^k \subseteq r_B$. Therefore,

$$\textit{Reach(M)} = \left(\bigcup_{k=0}^{|S|} (\bar{r})^k \right) [I] \subseteq r_B[I]$$

by monotonicity of relation image. $\diamondsuit$

3 Propositional Logic with If (8pt)

Consider the ternary propositional operation $\mathsf{ite(x,y,z)}$ (if x then y else z) defined by the following truth table:

x	y	z	$\mathsf{ite(x,y,z)}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

We consider expressions build only from variables and ite, without constants 0 and 1. We call them pure ite expressions.

A) (1pt) Express $x \wedge y$ as a pure ite expression; **Solution:** $\mathsf{ite(x,y,x)}$ $\Diamond$

B) (1pt) Express $x \vee y$ as a pure ite expression; **Solution:** $\mathsf{ite(x,x,y)}$ $\Diamond$

C) (1pt) Let e denote an ite expression whose set of free variables is $V = \{x_1, \ldots, x_n\}$. Let v be an assignment assigning all variables in V to 0 (false). What is the truth value of e under v? **Solution:** *The truth value of a pure* ite *expression is given by an evaluation function that traverses the expression from the root following a single path until it reaches the leaf of the expression, which can only be a variable. Because all variables are interpreted as 0, the resulting expression is 0. We observe that, analogously, if all variables are interpreted as 1, then the value of the expression is 1.* $\Diamond$

D) (1pt) Given an example of a function $f : \{0,1\}^n \to \{0,1\}$ that is not expressible as a pure ite expression. **Solution:** *From the previous observation $f(0,\ldots,0) = 0$ and $f(1,\ldots,1) = 1$ for every function expressible as a pure* ite *expression. Thus, any function for which either $f(0,\ldots,0) = 1$ or for which $f(1,\ldots,1) = 0$ cannot be expressed using pure* ite *expression. Examples include constant functions: function $f_0(x_1,\ldots,x_n) = 0$ for all x_i, function $f_1(x_1,\ldots,x_n) = 1$ for all x_i, negation $f(x) = \neg x$ because, for example, $f(0) = 1$, and exclusive or $f(x_1,x_2) = x_1 \oplus x_2$ because $f(1,1) = 0$.* $\Diamond$

E) (4pt) List (in the separate sheet) the answer numbers next to all true completions of the statement:

Solution: *The two conditions $f(0,\ldots,0) = 0$ and $f(1,\ldots,1) = 1$ are necessary for f to be expressible as pure* ite *expression, as explained in the solution to C). We now show that it is also sufficient. The idea is that we build a binary tree making case analysis on the truth values $v_1,\ldots,v_n$ of all variables $x_1,\ldots,x_n$. Each branch of a tree is an* ite *node and each leaf is given by a path that follows the values $v_1,\ldots,v_n$. In all branches except the* *ones corresponding to the values $(0,\ldots,0)$ and $(1,\ldots,1)$, there must be a variable v_i that is interpreted as 1 and a variable v_j that is interpreted as 0. If $f(v_1,\ldots,v_n)=1$ then we use as the corresponding leaf for the path $v_1,\ldots,v_n$ the smallest x_i for which $v_i=1$. Otherwise, we put the smallest x_j for which $v_j=0$. Such pure* ite *expression encodes precisely the truth table of f. Remark that for a function of n variables there are 2^n assignments and 2^{2^n} boolean functions. With the restriction on two values of the function, the number of expressible functions is 2^{2^n-2}, or one quarter ($\frac{1}{4}$) of all boolean functions.* $\Diamond$

The functions $f:\{0,1\}^n\to\{0,1\}$ that can be expressed as pure ite expressions using variables $x_1,\ldots,x_n$ are precisely …

1. …the functions that are not constant, i.e., not equal to either the function $f_1(x_1,\ldots,x_n)=1$ nor to the function $f_0(x_1,\ldots,x_n)=0$
2. …all functions if $n\geq 3$; functions expressible using $\wedge,\vee$ in cases $n=1,2,3$
3. …the functions that could be expressed using an expression with only $\wedge$ and $\vee$
4. …the functions f such that $f(0,\ldots,0)=0$
5. …the functions f such that $f(0,\ldots,0)=0$ and $f(1,\ldots,1)=1$
6. …the functions expressible using $\neg(x\wedge y)$ as a binary operation
7. …the functions such that $f(x,\ldots,x)=x$ for every x
8. …the functions such that $f(x,\ldots,x,f(x,\ldots,x))=x$ for every x

For those answers that you chose, explain briefly why they are correct. **Solution:** *Statement 1 is **not** correct because it does not rule out, for example negation.*

*Statement 2 is **not** correct because it does not rule out, for example, a constant function for $n=4$.*

*Statement 3 is **not** correct because it rules out too many functions, including the function $f(x_1,x_2,x_3)=\mathsf{ite}(\mathsf{x}_1,\mathsf{x}_2,\mathsf{x}_3)$. This function is not expressible using $\wedge,\vee$ because it is not monotonic. Indeed, functions expressible using $\wedge,\vee$ are monotonic, which means that*

$$f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n)\leq f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n)$$

for all $0\leq i\leq n$. However, the property

$$\mathsf{ite}(0,\mathsf{x}_2,\mathsf{x}_3)\leq\mathsf{ite}(1,\mathsf{x}_2,\mathsf{x}_3)$$

does not hold as it reduces to $x_3\leq x_2$ and thus fails to hold for $x_3=1$, $x_2=0$ i.e. for

$$\mathsf{ite}(0,0,1)\leq\mathsf{ite}(1,0,1)$$

Statement 4 is not correct because it omits the requirement $f(1,\ldots,1)=1$ and thus would admit function $f(x)=1$ which is not expressible.

Statement 5 *is the correct characterizations and thus one of the answers that should be indicated.*

Statement 6 is not correct because it admits all functions, as $\neg(x \wedge y)$ can express all boolean functions.

Statement 7 *is also correct. It is a universal statement over all x, but x is either 0 or 1, so it is equivalent to the conjunction $f(0,\ldots,0) = 0$ and $f(1,\ldots,1) = 1$.*

Statement 8 is not a correct characterization. uses a universally quantified statement. Note that if the correct characterization holds, then this property holds; the question is whether the property is strong enough. Consider $n = 2$. The case $x = 0$ becomes

$$f(0, f(0,0)) = 0$$

The case $x = 1$ becomes

$$f(1, f(1,1)) = 1$$

If we let f be the exclusive or $\oplus$ (addition modulo 2), then these properties are satisfied. However, exclusive or is not expressible as a pure ite *expression because $f(1,1) = 1 \oplus 1 = 0$.* $\Diamond$

4 Resolution with Congruence (6pt)

Consider a set D with a binary relation $\equiv$ on D and a binary function $\sqcup : D^2 \to D$ which satisfy the following properties:

$$\begin{array}{ll}
\textsf{assoc}: & x \sqcup (y \sqcup z) \equiv (x \sqcup y) \sqcup z \\
\textsf{transE}: & (x \equiv y) \wedge (y \equiv z) \to (x \equiv z) \\
\textsf{sym}: & (x \equiv y) \to (y \equiv x) \\
\textsf{congL}: & (x_1 \equiv x_2) \to (x_1 \sqcup y \equiv x_2 \sqcup y) \\
\textsf{congR}: & (y_1 \equiv y_2) \to (x \sqcup y_1 \equiv x \sqcup y_2)
\end{array}$$

Define another binary relation $\sqsubseteq$ on D by:

$$x \sqsubseteq y \;\;\leftrightarrow\;\; x \sqcup y \equiv y \qquad (\textsf{DefLE})$$

We claim that it follows that $\sqsubseteq$ is a transitive relation, that is:

$$x \sqsubseteq y \wedge y \sqsubseteq z \;\to\; x \sqsubseteq z \qquad (\textsf{TransLE})$$

In other words, we wish to prove that the following holds in pure first-order logic, where all formulas are considered universally quantified:

$$\{\textsf{assoc}, \textsf{transE}, \textsf{sym}, \textsf{congL}, \textsf{congR}, \textsf{DefLE}\} \models \textsf{TransLE} \qquad (*)$$

Note that, when representing the problem in first order logic, $t_1 \equiv t_2$ is just a notation for $E(t_1, t_2)$ where E is some binary predicate symbol, $t_1 \sqsubseteq t_2$ is a notation for $L(t_1, t_2)$ where L is another binary predicate symbol, and $t_1 \sqcup t_2$ is a notation for $f(t_1, t_2)$ where f is some binary function symbol. You can choose to either use notation $\equiv, \sqsubseteq, \sqcup$ or E, L, f, but use one or the other consistently.

Solution: *We first outline why this proof should exist by stating a simple result about orders and associative operations. For this preliminary statement we use the usual mathematical equality; we later expand this into a proof in first-order logic by applying appropriate properties of equality (sym, congL, congR). Let $\sqcup$ be associative and define $\sqsubseteq$ according to DefLE. To prove transitivity (TransLE), assume $x \sqsubseteq y$ and $y \sqsubseteq z$. This means $x \sqcup y = y$ and $y \sqcup z = z$. Thus, using associativity and the definition of $\sqsubseteq$,*

$$x \sqcup z = x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z = y \sqcup z = z$$

which by definition means that $x \sqsubseteq z$. The resolution proof will reflect the proof above, but needs to explicitly invoke, for example, congL and sym to conclude the first equality above. The Stainless code below hints at these steps. $\Diamond$

Our goal is to use resolution for first-order logic to derive a formal proof of $(*)$ by deriving a contradiction.

A) (2pt) To begin the proof, write down a numbered sequence of *clauses* that you will need in your proof, corresponding to assoc, transE, ... (whatever the initial set of clauses should be to prove $(*)$ by contradiction).

1. $\{x \sqcup (y \sqcup z) \equiv (x \sqcup y) \sqcup z\}$ (from assoc)
2. $\{\neg(x \equiv y), \neg(y \equiv z), (x \equiv z)\}$ (from transE)
3. ...

You may need more than one clause to encode some of the formulas.

Solution:

1. *$\{\{x \sqsubseteq y\}, \{y \sqsubseteq z\}, \{\neg(x \sqsubseteq z)\}\}$ (from TransLE)*
2. *$\{\{\neg(x \sqsubseteq y), x \sqcup y \equiv y\}, \{x \sqsubseteq y, \neg(x \sqcup y \equiv y)\}\}$ (from DefLE)*
3. *$\{x \sqcup (y \sqcup z) \equiv (x \sqcup y) \sqcup z\}$ (from assoc)*
4. *$\{\neg(x \equiv y), \neg(y \equiv z), x \equiv z\}$ (from transE)*
5. *$\{\neg(x \equiv y), y \equiv x\}$ (from sym)*
6. *$\{\neg(x_1 \equiv x_2), x_1 \sqcup y \equiv x_2 \sqcup y\}$ (from congL)*
7. *$\{\neg(y_1 \equiv y_2), x \sqcup y_1 \equiv x \sqcup y_2\}$ (from congR)*

$\Diamond$

B) (4pt) Continue to write proof steps where each step follows from previous ones. For each step indicate from which previous steps it follows and by which substitution of variables. The last step should be the empty clause $\emptyset$. To save you time in finding the resolution proof, we provide the following fragment of a Stainless file which we used to check this fact; even if this is not a resolution proof, the invocations of functions in the proof provides hints about the substitutions that you may want to use in your proof.

```
def sym(x: D, y: D) = {...}.ensuring( !(x ≡ y) || y ≡ x )
... // define the other axioms

def transitive(x: D, y: D, z: D) = {
  require(x ⊑ y)
  require(y ⊑ z)
  sym(y ⊔ z, z)
  congR(x, z, y ⊔ z)
  assoc(x, y, z)
  trans(x ⊔ z, x ⊔ (y ⊔ z), (x ⊔ y) ⊔ z)
  congL(x ⊔ y, y, z)
  transE(x ⊔ z, (x ⊔ y) ⊔ z, y ⊔ z)
  transE(x ⊔ z, y ⊔ z, z)
}.ensuring(x ⊑ z)
```

Solution:

8. $\{x \sqsubseteq y\}$ *(by* 1*)*
9. $\{x \sqcup y \equiv y\}$ *(by 8 and* 2 *with* $x := x;\ y := y$*)*
10. $\{y \sqsubseteq z\}$ *(by* 1*)*
11. $\{y \sqcup z \equiv z\}$ *(by 10 and* 2 *with* $x := x;\ y := y$*)*
12. $\{\neg(y \sqcup z \equiv z), z \equiv y \sqcup z\}$ *(by* 5 *with* $x := y \sqcup z;\ y := z$*)*
13. $\{z \equiv y \sqcup z\}$ *(by 11 and 12)*
14. $\{\neg(z \equiv y \sqcup z), x \sqcup z \equiv x \sqcup (y \sqcup z)\}$ *(by* 7 *with* $x := x;\ y_1 := z;\ y_2 := y \sqcup z$*)*
15. $\{x \sqcup z \equiv x \sqcup (y \sqcup z)\}$ *(by 13 and 14)*
16. $\{x \sqcup (y \sqcup z) \equiv (x \sqcup y) \sqcup z\}$ *(by* 3 *with* $x := x;\ y := y;\ z := z$*)*
17. $\{\neg(x \sqcup z \equiv x \sqcup (y \sqcup z)), \neg(x \sqcup (y \sqcup z) \equiv (x \sqcup y) \sqcup z), x \sqcup z \equiv (x \sqcup y) \sqcup z\}$ *(by* 4 *with* $x := x \sqcup z;\ y := x \sqcup (y \sqcup z);\ z := (x \sqcup y) \sqcup z$*)*
18. $\{\neg(x \sqcup (y \sqcup z) \equiv (x \sqcup y) \sqcup z), x \sqcup z \equiv (x \sqcup y) \sqcup z\}$ *(by 15 and 17)*
19. $\{x \sqcup z \equiv (x \sqcup y) \sqcup z\}$ *(by 16 and 18)*
20. $\{\neg(x \sqcup y \equiv y), (x \sqcup y) \sqcup z \equiv y \sqcup z\}$ *(by* 6 *with* $x_1 := x \sqcup y;\ x_2 := y;\ y := z$*)*
21. $\{(x \sqcup y) \sqcup z \equiv y \sqcup z\}$ *(by 9 and 20)*
22. $\{\neg(x \sqcup z \equiv (x \sqcup y) \sqcup z), \neg((x \sqcup y) \sqcup z \equiv y \sqcup z), x \sqcup z \equiv y \sqcup z\}$ *(by* 4 *with* $x := x \sqcup z;\ y := (x \sqcup y) \sqcup z;\ z := y \sqcup z$*)*
23. $\{\neg((x \sqcup y) \sqcup z \equiv y \sqcup z), x \sqcup z \equiv y \sqcup z\}$ *(by 15 and 22)*

24. *$\{x \sqcup z \equiv y \sqcup z\}$ (by 21 and 23)*
25. *$\{\neg(x \sqcup z \equiv y \sqcup z), \neg(y \sqcup z \equiv z), x \sqcup z \equiv z\}$ (by 4 with $x := x \sqcup z;\ y := y \sqcup z;\ z := z$)*
26. *$\{\neg(y \sqcup z \equiv z), x \sqcup z \equiv z\}$ (by 25 and 24)*
27. *$\{x \sqcup z \equiv z\}$ (by 26 and 11)*
28. *$\{\neg x \sqsubseteq z\}$ (by 1)*
29. *$\{x \sqcup z \equiv z\}$ (by 28 and 2 with $x := x;\ y := z$)*
30. *$\{\}$ (by 27 and 29)*

Note that this solution express all the instantiations as individual step for maximum details, but it is perfectly acceptable to do instantiation in resolution steps, which maked the proof much shorter. $\Diamond$

5 Approximating Relations (7pt)

Consider a guarded command language whose meanings are binary relations on the set states U.

Let $E(a_1, \ldots, a_n)$ denote an expression built from some atomic relations $a_1, \ldots, a_n$, as well as diagonal relations

$$\Delta_P = \{(x, x) \mid x \in P\}$$

for various sets $P \subseteq U$. The expression E is built from these relations using union (to model non-deterministic choice, relation composition (to represent sequential composition) and transitive closure (to represent loops).

Let us call a relation $s \subseteq U \times U$ an *effect* if it is reflexive (R) and transitive (T).

A) (2pt) Prove that if s is an effect and $a_i \subseteq s$ for all $1 \leq i \leq n$, then

$$E(a_1, \ldots, a_n) \subseteq s$$

Solution: *The proof is by structural induction on the expression tree of E. The base cases are $a_i \subseteq s$, which is the assumption, and $\Delta_P \subseteq s$, which follows because $(x, x) \in \Delta_P$ implies $(x, x) \in s$ by the assumption that s is reflexive. The first inductive case $E = E_1 \cup E_2$ follows by*

$$E = E_1 \cup E_2 \subseteq s \cup s = s$$

The next inductive case is

$$E = E_1 \circ E_2 \subseteq s \circ s \subseteq s$$

where $s \circ s \subseteq s$ is the transitivity property of s.

Note that transitivity and reflexivity also imply $s^n \subseteq s$ for all $n \geq 0$ by induction on n.

Finally consider the transitive closure operator:

$$E = E_1^* = \bigcup_{i=1}^{\infty} E_1^n \subseteq \bigcup_{i=1}^{\infty} s^n \subseteq s$$

by monotonicity of $\bigcup$. $\Diamond$

B) (2pt) Let $U = \mathbb{Z}^2$ denote pairs of integers, denoted by integer variables x, y. Let s be a specification relation given by the formula:

$$s = \{((x, y), (x', y')) \mid y \geq 0 \to (x' \leq x \land y' \geq 0)\}$$

Show that s is an effect. **Solution:** *To show reflexivity, we let $(x', y') = (x, y)$. The formula defining s becomes*

$$y \geq 0 \to (x \leq x \land y \geq 0)$$

Since $x \leq x$ is true, the formula reduces to $y \geq 0 \to y \geq 0$ which is an instance of the tautology $p \to p$.

For transitivity, we consider three states: $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ and assume $((x_1, y_1), (x_2, y_2)) \in s$, $((x_2, y_2), (x_3, y_3)) \in s$. Thus

$$y_1 \geq 0 \to (x_2 \leq x_1 \land y_2 \geq 0)$$
$$y_2 \geq 0 \to (x_3 \leq x_2 \land y_3 \geq 0)$$

To show $((x_1, y_1), (x_3, y_3)) \in s$, assume $y_1 \geq 0$. Then by first implication above $x_2 \leq x_1$ and $y_2 \geq 0$. From the second implication it then follows $x_3 \leq x_2$ and $y_3 \geq 0$. From the transitivity of $\leq$ we get $x_3 \leq x_1$. Together with $y_3 \geq 0$ this gives us the desired conclusion. Hence

$$y_1 \geq 0 \to (x_3 \leq x_1 \land y_3 \geq 0)$$

so s is transitive. $\Diamond$

C) (3pt) For the effect s in the previous point, prove that $\rho(p) \subseteq s$ where p is the following program (the initial values of variables can be arbitrary):

```
while (y ≥ 0) {
  x = x – y;
  if (x % 2 == 0)
    y = y / 2
  else
    y = 3*y + 1
}
```

Notation $\rho(p)$ denotes the relation corresponding to the program p. **Solution:** *Note that*

$$\rho(p) = (\Delta_{y\geq 0} \circ \rho(body))^* \circ \Delta_{y<0}$$
$$\rho(body) = \rho(x = x - y) \circ (\Delta_{x\%2==0} \circ \rho(y = y/2) \cup \Delta_{x\%2!=0} \circ \rho(y = 3 * y + 1))$$

$\rho(x = x - y) \subseteq s$: if $y \geq 0$, $y' \geq 0$ (since $y = y'$) and $x' = x - y \leq x$.
$\rho(y = y/2) \subseteq s$: if $y \geq 0$, $y' = y/2 \geq 0$. Moreover since $x' = x$, $x' \leq x$.
$\rho(y = 3 * y + 1) \subseteq s$: if $y \geq 0$, $y' = 3 * y + 1 \geq 0$. Moreover since $x' = x$, $x' \leq x$.

Since all the atomic expressions of the programs are subset of s and since s is an effect, by the result of A) $\rho(p) \subseteq s$. $\Diamond$

6 Logic of Partial Functions (9pt)

We wish to use first-order logic for our verification task, which requires proving validity of formulas. We use a first-order language (signature) with a relational symbol E, which we would like to represent equality and satisfy the laws of reflexivity, symmetry, and transitivity, as well as congruence laws (analogous to congL and congR in Problem 4): equal elements are related in the same way by all other relation symbols. We also need a way to represent a *partial* function $\bar{p}(x, y)$ of two arguments: the function can have at most one result, but it can be undefined for certain pairs of elements. Applying $\bar{p}$ when argument is undefined gives undefined result. We will analyze two possibilities for encoding such partial functions in first-order logic, with questions arising in each of them.

Encoding 1. We use a constant b to represent undefined element and a binary function symbol $p(x, y)$ to represent the function $\bar{p}$ (note that we use p to represent the function symbol, whereas $\bar{p}$ represents the function that interprets it). The language of formulas in this part has only symbols $\{E, p, b\}$.

A) (1pt) Write down, as universally quantified first-order logic formulas, the congruence properties of p with respect to E, namely: if in the expression $p(x, y)$ we change x to an E-related element or change y to an E-related element, the new result is E-related to $p(x, y)$.

Solution: *A possible answer is*

$$\forall x.\forall y.\forall z.\ (E(x,z) \rightarrow E(p(x,y), p(z,y)))$$
$$\forall x.\forall y.\forall z.\ (E(y,z) \rightarrow E(p(x,y), p(x,z)))$$

$\Diamond$

B) (1pt) Write down, as a universally quantified first-order logic formula, the property that applying p when one of the arguments is undefined results in an undefined value.

Solution: *A possible answer is*

$$\forall x.\forall y.\ E(p(b,y), b) \wedge E(p(x,b), b)$$

$\Diamond$

C) (1pt) Describe Herbrand universe (the set of ground terms) in this language. Is it finite or infinite?

Solution:

The set of ground terms of this language is of the form

$$GT_{\mathcal{L}} = \{b, p(b,b), p(p(b,b),b)...\}$$

Formally $GT_{\mathcal{L}}^0 = \emptyset \quad GT_{\mathcal{L}}^{i+1} = \{b\} \cup \{p(t_1,t_2) | t_1, t_2 \in GT_{\mathcal{L}}^i\} \quad GT_{\mathcal{L}} = \bigcup_{i=0}^{\infty} GT_{\mathcal{L}}^i$

It is countably infinite. $\Diamond$

Encoding 2. We use a ternary relation symbol $P(x, y, z)$ to represent the fact $\bar{p}(x, y) = z$ when all values are defined. To represent that the function is not defined for a given x and y, relation interpreting P would simply not contain any (interpretation of) z such that $P(x, y, z)$ is true. Let a be a constant denoting an arbitrary element of the universe. The language of formulas in this part has only symbols $\{E, P, a\}$.

D) (1pt) Write down, as universally quantified first-order logic formulas, the congruence properties of P with respect to E, namely: if elements x, y, z are related by P, then elements E-related to them are also related by P, as expected by a relation with properties of equality.

Solution: *A possible answer is*

$$\begin{array}{c}\forall x.\forall y.\forall z.\forall w.\ E(x,w) \rightarrow (P(x,y,z) \leftrightarrow P(w,y,z)) \\ \forall x.\forall y.\forall z.\forall w.\ E(y,w) \rightarrow (P(x,y,z) \leftrightarrow P(x,w,z)) \\ \forall x.\forall y.\forall z.\forall w.\ E(z,w) \rightarrow (P(x,y,z) \leftrightarrow P(x,y,w))\end{array}$$

$\Diamond$

E) (1pt) Write down as a universally quantified formula, the property that P should represent a functional relation modulo E: for given pair of elements x, y, the result of $\bar{p}$, if it exists, is unique up to E.

Solution: *A possible answer is*

$$\forall x.\forall y.\forall z.\forall w.\ (P(x,y,z) \land P(x,y,w)) \rightarrow E(z,w)$$

$\Diamond$

F) (1pt) Consider the result of applying Skolemization of the properties in D) and E). How many new Skolem functions are introduced? Describe the Herbrand universe (the set of ground terms) in this language. Is it finite or infinite?

Solution:

Since there are no existential quantifiers, applying Skolemization does not introduce new function symbols in the language. Therefore, the set of ground terms in this new language is $\{a\}$ $\Diamond$

G) (3pt) Is there a terminating algorithm for checking, given two formulas F_1, F_2 in prenex form with only universal quantifiers using symbols from $\{E, P, a\}$, whether $\mathsf{Ax} \cup \{F_1\} \models F_2$ holds, where Ax are the Skolemized versions of properties introduced in D) and E). If yes, sketch the algorithm. If no, argue why the problem is undecidable.

Solution: *We claim it is sufficient to check* $\mathsf{Ax} \cup \{F_1\} \models F_2$ *on models with* $n+1$ *elements, where* n *is the number of quantifiers in* F_2.

$$\mathsf{Ax} \cup \{F_1\} \models F_2$$

is equivalent to

$$\models (\mathsf{Ax} \wedge F_1) \rightarrow F_2$$

i.e. validity of $(\mathsf{Ax} \wedge F_1) \rightarrow F_2$*. This is equivalent to the satisfiability of* $\neg((\mathsf{Ax} \wedge F_1) \rightarrow F_2)$ *which translates to*

$$\mathsf{Ax} \wedge F_1 \wedge \neg F_2$$

Writing F_2 *as* $\forall x_1, ..., x_n\ F_2'$*, this is again equivalent to*

$$\mathsf{Ax} \wedge F_1 \wedge \exists x_1, ..., x_n\ \neg F_2'$$

We can now skolemize the expression to obtain an equisatisfiable formula:

$$\mathsf{Ax} \wedge F_1 \wedge \neg F_2'[x_1 := c_1, ..., x_n := c_n]$$

This formula has $n + 1$ *constant symbols (*c_1 *to* c_n *and* a*) and no function symbol, so its Herbrand model has* $n + 1$ *elements.*

Hence, to decide whether $\mathsf{Ax} \cup \{F_1\} \models F_2$ *holds, it is sufficient to decide whether* $\mathsf{Ax} \wedge F_1 \wedge \neg F_2'[x_1 : c_1, ..., x_n := c_n]$ *has a model with at most* $n + 1$ *elements (and return the opposite).*

$\Diamond$