Andrey Tyukin

Exercise 7.1 (Compositions of strictly monotone functions) Let O be some set of *outcomes*, $\preceq$ a total, reflexive and transitive relation (*preference* relation). Let $u : O \times O \to \mathbb{R}$ *represent* the relation $\preceq$, that is, for all $x, y \in O$ it holds:

$$x \preceq y \Leftrightarrow u(x) \leq u(y).$$

Let $v : \mathbb{R} \to \mathbb{R}$ be *strictly* monotonously increasing. We claim that $v \circ u$ also represents $\preceq$.

Lemma: If $v : \mathbb{R} \to \mathbb{R}$ is strictly monotonous, that is:

$$a < b \Rightarrow v(a) < v(b),$$

then it holds:

$$a \leq b \Leftrightarrow v(a) \leq v(b).$$

Proof of the lemma: Suppose $a \leq b$. Then either $a = b$ or $a < b$. The first case clearly implies $v(a) = v(b)$, because v is a function, not some non-deterministic algorithm. The second case implies by definition $v(a) < v(b)$. In both cases we obtain $v(a) \leq v(b)$.

Now suppose $v(a) \leq v(b)$. If $v(a) = v(b)$, then $a = b$, because v is injective. If $v(a) < v(b)$, then $b \leq a$ cannot hold, because otherwise we would have $v(b) \leq v(a)$. Therefore, it must hold $a < b$. In both cases we obtain $a \leq b$.

Both directions together show the lemma. ■

Now, the proof of the main claim that $(v \circ u)$ also represents $\preceq$ is easy:

Proof: For all $x, y \in O$ it holds:

$$x \preceq y \Leftrightarrow u(x) \leq u(y) \Leftrightarrow v(u(x)) \leq v(u(y)).$$

■

Remark: Injectivity of v is really necessary, because otherwise we get only

$$x \preceq y \Leftrightarrow u(x) \leq u(y) \Rightarrow v(u(x)) \leq v(u(y)).$$

with $v = 0$ as counterexample for the opposite implication.

Exercise 7.2 (Lex) Define the lexicographical order on $I^2 := [0, 1]^2$ as follows:

$$x \leq_{lex} y :\Leftrightarrow x_1 < x_y \vee (x_1 = y_2 \wedge x_2 \leq y_2).$$

a) We claim that $\leq$ is a total, reflexive and transitive relation.

Proof: Let $x, y \in I^2$ be two arbitrary points. Since the usual relation $\leq$ is total on I, there are three possible cases: $x_1 < y_1$, $x_1 = y_1$ or $x_1 > y_1$. The first and last cases are analogous, so we consider only the first two. In the first case it holds $x \leq_{lex} y$. In the second case, at least one of the both inequalities $x_2 \leq y_2$ or $y_2 \leq x_2$ must hold, therefore it's either $x \leq_{lex} y$ or $y \leq_{lex} x$ respectively. In every case, either $x \leq_{lex} y$ or $y \leq_{lex} x$ holds, so $\leq_{lex}$ is total.

Obviously, $x_1 = x_1$ and $x_2 \leq x_2$, therefore $x \leq_{lex} x$, so $\leq_{lex}$ is reflexive.

Now let $z \in I^2$ be a third point. We want to show that from $x \leq_{lex} y \leq_{lex} z$ it follows that $x \leq_{lex} z$. The proof can again be done by a trivial case-analysis. However, since it's tedious and error-prone, it seems to be easier and safer to check all the cases by brute-force, for example using the following little C program:

```
#include <stdio.h>

int lex(int a1, int a2, int b1, int b2) {
  return (a1 < b1) || (a1 == b1 && a2 <= b2);
}

int main(void) {
  int allImplicationsHold = 1;
  int x1; for (x1 = 0; x1 < 3; x1 ++) {
  int x2; for (x2 = 0; x2 < 3; x2 ++) {
  int y1; for (y1 = 0; y1 < 3; y1 ++) {
  int y2; for (y2 = 0; y2 < 3; y2 ++) {
  int z1; for (z1 = 0; z1 < 3; z1 ++) {
  int z2; for (z2 = 0; z2 < 3; z2 ++) {
    int xLy = lex(x1, x2, y1, y2);
    int yLz = lex(y1, y2, z1, z2);
    int xLz = lex(x1, x2, z1, z2);
    int implication = !(xLy && yLz) || xLz;
    allImplicationsHold = allImplicationsHold && implication;
    printf("%d %d %d %d\n", xLy, yLz, xLz, implication);
  }
  }
  }
  }
  }
  }
  printf("All implications hold: %d ", allImplicationsHold);
}
```

This program generates all possible cases for all possible relations of x_1, x_2, ..., z_2 and checks that $x \leq_{lex} z$ holds whenever $x \leq_{lex} y$ and $y \leq_{lex} z$ hold. The output is a truth table with $3^6 = 729$ lines, together with a last line that tells us whether the implication always holds.

```
1 1 1 1
...
1 0 0 1
1 0 1 1
1 0 1 1
1 1 1 1
1 1 1 1
...
1 0 0 1
1 0 0 1
1 0 0 1
1 0 1 1
1 0 1 1
1 0 1 1
1 1 1 1
...
1 0 0 1
```

```
1 1 1 1
All implications hold: 1
```

The result is 1, which means *true*, so the computer-assisted case analysis proves our claim. ■

b) We want to show that $\leq_{lex}$ on I^2 is not representable by any $\mathbb{R}$-valued function.

Suppose for the sake of contradiction that there is some $u : I^2 \to \mathbb{R}$ which represents $\leq_{lex}$. For all $x \in I$ it must hold: $u(x, 0) < u(x, 1)$, that is $u(x, 1) - u(x, 0)$ is positive. Subdivide the unit interval as follows:

$$X_n := \left\{ x \in I \;\middle|\; u(x,1) - u(x,0) > \frac{1}{n} \right\}.$$

Since I is uncountable, there must exists some $N \in \mathbb{N}$ such that X_N is also uncountable, because otherwise the unit interval I would be countable as a countable union of countable sets. Let $M := \lceil u(1,1) \rceil$ be the ceiled maximum value of the function u, and $m := \lfloor u(0,0) \rfloor$ the floored minimum value of the function u. Let $K := 2(M-m)N + 1$. Since X_N contains infinitely many elements, we can find K distinct elements $x_1, \ldots, x_K$ in X_N. Without loss of generality, we can assume that the finite sequence $(x_i)_{i=1}^K$ is sorted in increasing order. It holds:

$$u(x_i, 1) + \frac{1}{N} < u(x_{i+1}, 0) + \frac{1}{N} < u(x_{i+1}, 1),$$

applying this $K-1$ times to the whole sequence $(x_i)_i$ yields:

$$u(x_K, 1) > u(x_1, 1) + \frac{2(M-m)N}{N} = u(x_1, 1) + 2(M-m),$$

and in particular

$$u(1,1) \geq u(x_K, 1) > u(x_1, 1) + 2(M-m) \geq u(0,0) + 2(M-m).$$

Now we obtain the situation where the difference between $u(0,0)$ and $u(1,1)$ is at least twice as big as the difference between the smallest and the largest value of u. This is a contradiction. Therefore there can not be a representing function for $\leq_{lex}$. ■

Exercise 7.3 (Harasanyi-Game) This seems to be just a "term-unification"/"definition-understanding" task, so we just show how the concepts in the text match the symbols in the definition of a Harasanyi-game.

The set of players is $N = \{\text{buyer}, \text{seller}\}$. The sets of types are

$$\begin{aligned} T_{\text{buyer}} &= \{\text{wantCar}\} \\ T_{\text{seller}} &= \{\text{offerGoodCar}, \text{offerBadCar}\}. \end{aligned}$$

The type-dependent action sets are:

$$\begin{aligned} A_{\text{buyer}}\,(\text{wantCar}) &= \{\text{buy}, \neg\text{buy}\} \\ A_{\text{seller}}(\text{offerGoodCar}) = A_{\text{seller}}(\text{offerBadCar}) &= \{\text{sell}, \neg\text{sell}\}. \end{aligned}$$

The probability distribution $\mathbb{P} \in \Delta(T)$ is described by a single constant $q \in [0,1]$, because

$$T = \{(\mathrm{wantCar}, \mathrm{offerGoodCar}), (\mathrm{wantCar}, \mathrm{offerBadCar})\}$$

contains just two elements, so it holds:

$$\begin{aligned} \mathbb{P}\left[\{(\mathrm{wantCar}, \mathrm{offerGoodCar})\}\right] &= q \\ \mathbb{P}\left[\{(\mathrm{wantCar}, \mathrm{offerBadCar})\}\right] &= 1 - q. \end{aligned}$$

The type dependent utility functions u_1, u_2 are given by the matrices (we use the convention that u_1 and u_2 are written in single cell):

$$\begin{aligned} u_{1,2}(\mathrm{wantCar}, \mathrm{offerGoodCar}) &= \begin{bmatrix} & \mathrm{sell} & \neg\mathrm{sell} \\ \mathrm{buy} & 6-c, c & 0,5 \\ \neg\mathrm{buy} & 0,5 & 0,5 \end{bmatrix} \\ u_{1,2}(\mathrm{wantCar}, \mathrm{offerBadCar}) &= \begin{bmatrix} & \mathrm{sell} & \neg\mathrm{sell} \\ \mathrm{buy} & 4-c, c & 0,0 \\ \neg\mathrm{buy} & 0,0 & 0,0 \end{bmatrix}. \end{aligned}$$

Here c stands for the cost of the car.