An exercise in axiomatic semantics

```
Example: double x = x+x = map double [1,2,3] = [2,4,6]
m1: map f[] = []
m2: map f a:as = f(a):(map f as)
i1: inverse [] = []
                                                                          Example: inverse [1,2,3] = [3,2,1]
i2: inverse a:as = (inverse as) ++ [a]
Prove that for all functions f and all lists as.
                                                 inverse (map f as) = map f (inverse as)
inverse (map f []) = inverse []
                                    by m1
                                                          map f (inverse []) = map f []
                                                                                               by i1
                    =[1]
                                          by i1
                                                                                    =[1]
                                                                                                         by m1
inverse (map f a:as) = inverse (f(a):(map f as))
                                                          by m2
                       = (inverse (map f as)) ++ [f(a)])
                                                               bv i2
                        = (map f (inverse as)) ++ [f(a)]
                                                               by inductive hypothesis
                        = map f ((inverse as) ++ [a])
                                                                by lemma1: (map f as) ++ (map f bs) = map f (as ++ bs)
                        = map f (inverse as) ++ (inverse [a])) by lemma2: if len(as) = 1 then inverse as = as
                        = map f (inverse a: as)
                                                               by lemma3: (inverse as) ++ (inverse bs) = inverse (bs ++ as)
```

Exercise 1: Give an inductive definition of the set of lists of natural numbers and prove lemmas 1, 2, and 3 above.

What do we mean by correctness?

- SAFETY: "nothing bed happens"
. If a number is printed then it is a positive prime less than 10'0 - LIVENESS: "something good eventually happens" . All estats looking for a recharge eventually find a charge station BTW: The contlink of sequential programs as multi-threaded ones with 1 thread only But there're serious differences. · testing is hard poor reproducibility heisen bugs localisation · non-détermism: blessing & curse

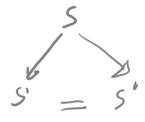
Modelling behaviour

The evolution of a system can be described in terms of state transitions

- states represent the possible configurations the system can be in
- transitions represent the possible evolution from a given configuration.

In its simplest form, such models can be mathematically rendered as binary relations

Sys is deterministic if $\forall s, s', s'' \in S$: $S \Rightarrow S' = S'$



Of course this idea is hardly new and examples can be found in any book on automata or formal languages. Its application to the definition of programming languages can be found in the work of Landin and the Vienna Group [Lan,Oll,Weg].

[Lan] Landin, P.J. (1966) A Lambda-calculus Approach, Advances in Programming and Non-numerical Computation, ed. L. Fox, Chapter 5, pp. 97–154, Pergamon Press.

[OII] Ollengren, A. (1976) Definition of Programming Languages by Interpreting Automata, Academic Press.

[Weg] Wegner, P. (1972) The Vienna Definition Language, ACM Computing Surveys 4(1):5-63.

Examples (Plotkin)

Exercise 2

Complete the definition of the transition relation so that such relation is deterministic for all programs

$$S = (N \cup \overline{Z})^*$$

$$- \lambda_{G} = (N \cup \overline{Z})^*$$

$$(x, w) \in \mathbb{N}$$

Exercise 3
Give a terminal TS for each of the examples above