Seminar 6 - Sequential Decision Problems

Adrian Perez Keilty

Chalmers University of Technology, Göteborg, Sweden

Exercise 6.1:

Make sure that you fully understand what solving a SDP means.

OK

Exercise 6.2:

Define M, X , Y and next *for the generation dilemma. What could be* Val *and* reward *for this problem?*

If we consider the generation dilemma with probabilities associated to its edges then this would turn into a stochastic system so M should be defined as a probability distribution (SP), otherwise it is a non-deterministic system which requires M to simply be List (its structure is reminiscent of the SDP example "A toy climate problem" described in "*On the Correctness of Monadic Backward Induction*").

(I've tried to translate into Agda some of the Idris code presented in the aforementioned paper, but I've not managed to resolve the type checking issues yet)

X is defined as a dependent pair type in order to ensure that the possible states are consistent with the current state. The first component is the current state and the second component is a list of possible "admissible" states that depend on the current one. Here's an attempt on Agda that I haven't managed to type check yet:

```
data StateGen : Set where
    GU    : StateGen -- good unsafe
    GS    : StateGen -- good safe
    BT    : StateGen -- bad temporary
    B     : StateGen -- bad permanent

-- Admisible states depending on the current state
AdmissibleStateGen : {t : Nat} → StateGen → List StateGen
AdmissibleStateGen {zero} x  = (GU :: [])
AdmissibleStateGen GU   = (GU :: BT :: B :: [])
AdmissibleStateGen GS   = (GS :: [])
AdmissibleStateGen BT   = (GS :: [])
AdmissibleStateGen B    = (B :: [])

-- Set of states as a dependent type on the previous states
postulate Xgen : (t : Nat) → Σ (StateGen) (λ s → AdmissibleStateGen s)
```

Similarly, Y should also be defined to be dependently typed such that only GU allows for a *control* to be taken (a or b). Regarding Val and reward, since the monoid and preorder structure presented for the SDP example "A toy climate problem" (Fig.2) can also be applied in this scenario, we can set $\mathsf{Val} = \mathbb{N}, \oplus = +$ and

```
reward t GU a B = 0
reward t GU a GU = α
reward t GU b BT = β
```

where depending on the interpretation of BT and GU, α and β can vary, e.g, if BT is short term then, reasonably $\beta > \alpha$. The measure should ensure that shorter trajectories that end up in GS always have a better total reward than the ones that are longer (more iterations in GU) but eventually fall into B.

Exercise 6.3:

Explain the (suc t) n - t (suc n) pattern in the definition of PolicySeq.

As I understood from the paper, this pattern is used to follow the backward induction principle: A policy at time step t is prepended to a policy sequence that has been constructed from step $t+n$ to step $t+1$. "(suc t) n" becomes "t (suc n)" simply to increase the length of the policy sequence or vector while maintaining t as the initial step in time.

Exercise 6.4:

A value of type XYSeq t n is like a vector. What is its length? Can n be zero? Why is the first constructor of XYSeq called Last?

The length of a value of type XYSeq t n should be equal to the number of sequences of state-control pairs after n decisions, so the length should still be n. The first constructor is called Last because this corresponds to the case where $n = 0$, hence step t, which is the last step performed in the backward induction algorithm.

Exercise 6.5:

Make sure that you understand the computation of possible trajectories. What are the types of y , mx′ the let-in clauses?

As the third parameter for next, the type of y is Y t x, and the type of mx' is M (X (suc t)).

Exercise 6.6:

Notice that val ps is a vulnerability measure! What are possible and harm here?

According to the above formula and retrieving the early definition

vulnerability $=$ measure $\circ$ fmap harm $\circ$ possible

we should have harm $=$ sumR and possible $=$ trj ps.