```
{-# LANGUAGE FlexibleInstances #-}
{-# LANGUAGE UndecidableInstances #-}
{-# LANGUAGE GeneralizedNewtypeDeriving #-}
module DSLsofMath.W07 where
import DSLsofMath.FunNumInst
type ℝ = Double
```

7 Matrix algebra and linear transformations

Often, especially in engineering textbooks, one encounters the definition: a vector is an $n+1$-tuple of real or complex numbers, arranged as a column:

$$v = \begin{bmatrix} v_0 \\ \vdots \\ v_n \end{bmatrix}$$

Other times, this is supplemented by the definition of a row vector:

$$v = \begin{bmatrix} v_0 & \cdots & v_n \end{bmatrix}$$

The v_is are real or complex numbers, or, more generally, elements of a *field* (analogous to being an instance of *Fractional*). Vectors can be added point-wise and multiplied with scalars, i.e., elements of the field:

$$v + w = \begin{bmatrix} v_0 \\ \vdots \\ v_n \end{bmatrix} + \begin{bmatrix} w_0 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} v_0 + w_0 \\ \vdots \\ v_n + w_n \end{bmatrix}$$

$$s * v = \begin{bmatrix} s * v_0 \\ \vdots \\ s * v_n \end{bmatrix}$$

The scalar s scales all the components of v.

But, as you might have guessed, the layout of the components on paper (in a column or row) is not the most important feature of a vector. In fact, the most important feature of vectors is that they can be *uniquely* expressed as a simple sort of combination of other vectors:

$$v = \begin{bmatrix} v_0 \\ \vdots \\ v_n \end{bmatrix} = v_0 * \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + v_1 * \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} + \cdots + v_n * \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

We denote by

$$e_k = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \leftarrow \text{position } k$$

the vector that is everywhere 0 except at position k, where it is 1, so that

$$v = v_0 * e_0 + ... + v_n * e_n$$

The algebraic structure that captures a set of vectors, with zero, addition, and scaling is called a *vector space*. For every field S of scalars and every set G of indices, the set $Vector\ G = G \to S$ can be given a vector space structure.

There is a temptation to model vectors by lists or tuples, but a more general (and conceptually simpler) way is to view them as *functions* from a set of indices G:

```
type S          = ...   -- the scalars, forming a field (ℝ, or Complex, or ℤ_n, etc.)
type Vector s g = g → s
```

Usually, G is finite, i.e., *Bounded* and *Enumerable* and in the examples so far we have used indices from $G = \{0, \ldots, n\}$. We sometime use *card* G to denote the *cardinality* of the set G, the number of elements ($n + 1$ in this case).

We know from the previous lectures that if S is an instance of *Num*, *Fractional*, etc. then so is $G \to S$, with the pointwise definitions. In particular, the instance declarations for $+$, multiplication, and embedding of constants, give us exactly the structure needed for the vector operations. For example

```
s * v                    = {- s is promoted to a function -}
const s * v              = {- Num instance definition -}
λg → (const s) g * v g   = {- definition of const -}
λg → s * v g
```

The basis vectors are then

$$e\ i : G \to S, e\ i\ g = i\ `is`\ g$$

Implementation:

```
is :: Num s ⇒ Int → Int → s
is a b = if a == b then 1 else 0

e :: Num s ⇒ G → Vector s G
e (G g) = V (λ(G g') → g `is` g')
toL (V v) = [v g | g ← [minBound .. maxBound]]   -- so we can actually see them
```

and every

$$v : G \to S$$

is trivially a linear combination of vectors $e\ i$:

$$v = v\ 0 * e\ 0 + ... + v\ n * e\ n$$

7.1 Functions on vectors

As we have seen in earlier chapters, morphisms between structures are often important. Vector spaces are no different: if we have two vector spaces *Vector* G and *Vector* G' (for the same set of scalars S) we can study functions $f : \mathit{Vector}\ G \to \mathit{Vector}\ G'$:

$$f\ v = f\ (v\ 0 * e\ 0 + ... + v\ n * e\ n)$$

For f to be a "good" function it should translate the operations in *Vector* G into operations in *Vector* G', i.e., should be a homomorphism:

$$f\ v = f\ (v\ 0 * e\ 0 + ... + v\ n * e\ n) = v\ 0 * f\ (e\ 0) + ... + v\ n * f\ (e\ n)$$

But this means that we can determine the values of $f : (G \to S) \to (G' \to S)$ from just the values of $f \circ e : G \to (G' \to S)$, a much "smaller" function. Let $m = f \circ e$. Then

$$f\ v = v\ 0 * m\ 0 + ... + v\ n * m\ n$$

Each of $m\ k$ is a *Vector* G', as is the resulting $f\ v$. We have

$$\begin{aligned}
&f\ v\ g' && = \{\text{- as above -}\}\\
&(v\ 0 * m\ 0 + ... + v\ n * m\ n)\ g' && = \{\text{- } * \text{ and } + \text{ for functions are def. pointwise -}\}\\
&v\ 0 * m\ 0\ g' + ... + v\ n * m\ n\ g' && = \{\text{- using } \mathit{sum}, \text{ and } (*) \text{ commutative -}\}\\
&\mathit{sum}\ [m\ j\ g' * v\ j \mid j \leftarrow [\mathit{minBound}\,..\,\mathit{maxBound}]]
\end{aligned}$$

Implementation:

This is almost the standard vector-matrix multiplication:

$$M = [m\ 0 \mid ... \mid m\ n]$$

The columns of M are the images of the canonical base vectors $e\ i$ through f (or, in other words, the columns of M are $f\ (e\ i)$). Every $m\ k$ has *card* G' rows, and it has become standard to use $M\ i\ j$ to mean the ith element of the jth column, i.e., $m\ j\ i$, so that

$$(M * v)\ i = \mathit{sum}\ [M\ i\ j * v\ j \mid j \leftarrow [0\,..\,n]]$$

We can implement this matrix-vector multiplication as *mulMV*:

```
mulMV :: (Finite g, Num s) ⇒
         (g' → Vector s g) → Vector s g → Vector s g'
mulMV m v = V $ λg' → sumV (m g' * v)
type Matrix s g g' = g' → Vector s g
sumV :: (Finite g, Num s) ⇒ Vector s g → s
sumV (V v) = sum (map v [minBound .. maxBound])
  -- mulMV : Matrix s g g' → (Vector s g → Vector s g')
```

Note that in the terminology of the earlier chapter we can see *Matrix* $g\ g'$ as a type of syntax and the linear transformation (of type *Vector* $s\ g \to$ *Vector* $s\ g'$) as semantics. With this view, *mulMV* is just another $\mathit{eval} :: \mathit{Syntax} \to \mathit{Semantics}$.

Example:

$$(M * e\ k)\ i = \mathit{sum}\ [M\ i\ j * e\ k\ j \mid j \leftarrow [0\,..\,n]] = \mathit{sum}\ [M\ i\ k] = M\ i\ k$$

i.e., $e\ k$ extracts the kth column from M (hence the notation "e" for "extract").

We have seen how a homomorphism f can be fully described by a matrix of scalars, M. Similarly, in the opposite direction, given an arbitrary matrix M, we can define

$$f\ v = M * v$$

and obtain a linear transformation $f = (M*)$. Moreover $((M*) \circ e)\ g\ g' = M\ g'\ g$, i.e., the matrix constructed as above for f is precisely M.

Exercise 7.1: compute $((M*) \circ e)\ g\ g'$.

Therefore, every linear transformation is of the form $(M*)$ and every $(M*)$ is a linear transformation.

Matrix-matrix multiplication is defined in order to ensure that

$$(M' * M) * v = M' * (M * v)$$

that is

$$((M' * M)*) = (M'*) \circ (M*)$$

Exercise 7.2: work this out in detail.

Exercise 7.3: show that matrix-matrix multiplication is associative.

Perhaps the simplest vector space is obtained for $G = ()$, the singleton index set. In this case, the vectors $s : () \to S$ are functions that can take exactly one argument, therefore have exactly one value: $s\ ()$, so they are often identified with S. But, for any $v : G \to S$, we have a function $fv : G \to (() \to S)$, namely

$$fv\ g\ () = v\ g$$

fv is similar to our m function above. The associated matrix is

$$M = [m\ 0 \mid ... \mid m\ n] = [fv\ 0 \mid ... \mid fv\ n]$$

having $n + 1$ columns (the dimension of *Vector* G) and one row (dimension of *Vector* ()). Let w :: *Vector* s G:

$$M * w = w\ 0 * fv\ 0 + ... + w\ n * fv\ n$$

$M * v$ and each of the $fv\ k$ are "almost scalars": functions of type $() \to S$, thus, the only component of $M * w$ is

$$(M * w)\ () = w\ 0 * fv\ 0\ () + ... + w\ n * fv\ n\ () = w\ 0 * v\ 0 + ... + w\ n * v\ n$$

i.e., the scalar product of the vectors v and w.

Remark: I have not discussed the geometrical point of view. For the connection between matrices, linear transformations, and geometry, I warmly recommend binge-watching the "Essence of linear algebra" videos on youtube (start here: `https://www.youtube.com/watch?v=kjBOesZCoqc`).

7.2 Examples of matrix algebra

7.2.1 Polynomials and their derivatives

We have represented polynomials of degree $n+1$ by the list of their coefficients. This is quite similar to standard geometrical vectors represented by $n+1$ coordinates. This suggests that polynomials of degree $n+1$ form a vector space, and we could interpret that as $\{0,...,n\} \to \mathbb{R}$ (or, more generally, $Field\ a \Rightarrow \{0,...,n\} \to a$). The operations $+$ (vector addition) and $*$ (vector scaling) are defined in the same way as they are for functions.

To explain the vector space it is useful to start by defining the canonical base vectors. As for geometrical vectors, they are

$$e\ i : \{0,...,n\} \to Real, e\ i\ j = i\ `is`\ j$$

but how do we interpret them as polynomial functions?

When we represented a polynomial by its list of coefficients, we saw that the polynomial function $\lambda x \to x\hat{}3$ could be represented as $[0,0,0,1]$, where 1 is the coefficient of $x\hat{}3$. Similarly, representing this list of coefficients as a vector (a function from $\{0,...,n\} \to \mathbb{R}$), we get the vector $\lambda j \to$ **if** $j == 3$ **then** 1 **else** 0, which is $\lambda j \to 3\ `is`\ j$ or simply $e\ 3$.

In general, $\lambda x \to x\hat{}i$ is represented by $e\ i$, which is another way of saying that $e\ i$ should be interpreted as $\lambda x \to x\hat{}i$. Any other polynomial function p equals the linear combination of monomials, and can therefore be represented as a linear combination of our base vectors $e\ i$. For example, $p\ x = 2 + x\hat{}3$ is represented by $2 * e\ 0 + e\ 3$.

In general, the evaluator from the vector representation to polynomial functions is as follows:

$$\begin{aligned}
&evalM :: G \to (\mathbb{R} \to \mathbb{R}) \\
&evalM\ (G\ i) = \lambda x \to x\hat{}i \\
&evalP :: Vector\ \mathbb{R}\ G \to (\mathbb{R} \to \mathbb{R}) \\
&evalP\ (V\ v)\ x = sum\ (map\ (\lambda i \to v\ i * evalM\ i\ x)\ [minBound\,..\,maxBound])
\end{aligned}$$

The *derive* function takes polynomials of degree $n+1$ to polynomials of degree n, and since $D\ (f+g) = D\ f + D\ g$ and $D\ (s*f) = s * D\ f$, we expect it to be a linear transformation. What is its associated matrix?

The associated matrix will be

$$M = [D\ (e\ 0), D\ (e\ 1), ..., D\ (e\ n)]$$

where each $D\ (e\ i)$ has length n. Vector $e\ (i+1)$ represents $\lambda x \to x\hat{}(i+1)$, therefore

$$\begin{aligned}
&eval\ (D\ (e\ (i+1))) = \{\text{- } eval \text{ is a homomorphism. Note that the } D \text{ has another type. -}\} \\
&D\ (eval\ (e\ (i+1))) = \{\text{- Def. of } eval \text{ -}\} \\
&D\ (\hat{}(i+1)) \qquad\quad = \{\text{- Def. of } D \text{ for polynomial functions -}\} \\
&\lambda x \to (i+1) * x\hat{}i \ = \{\text{- Def. of } eval \text{ for the base vectors -}\} \\
&eval\ ((i+1) * e\ i)
\end{aligned}$$

i.e.

$$D\ (e\ (i+1))\ j = \textbf{if}\ i == j\ \textbf{then}\ i+1\ \textbf{else}\ 0$$

and

$$D\ (e\ 0) = 0$$

Example: $n + 1 = 3$:

$$M = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Take the polynomial

$$1 + 2 * x + 3 * x\hat{}2$$

as a vector

$$v = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

and we have

$$M * v = \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 1 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 2 \end{bmatrix} \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$$

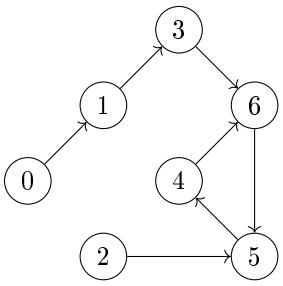
representing the polynomial $2 + 6 * x$.

Exercise 7.4: write the (infinite-dimensional) matrix representing D for power series.

Exercise 7.5: write the matrix associated with integration of polynomials.

7.2.2 Simple deterministic systems (transition systems)

Simple deterministic systems are given by endo-functions[5] on a finite set $f : G \to G$. They can often be conveniently represented as a graph, for example



Here, $G = \{0, ..., 6\}$. A node in the graph represents a state. A transition $i \to j$ means $f\ i = j$. Since f is an endo-function, every node must be the source of exactly one arrow.

We can take as vectors the characteristic functions of subsets of G, i.e., $G \to \{0, 1\}$. $\{0, 1\}$ is not a field w.r.t. the standard arithmetical operations (it is not even closed w.r.t. addition), and the standard trick to avoid this is to extend the type of the functions to $\mathbb{R}$.

The canonical basis vectors are, as usual, $e\ i = \lambda j \to i\text{`}is\text{`}\ j$. Each $e\ i$ is the characteristic function of a singleton set, $\{i\}$. Thus, the inputs to f are canonical vectors.

To make what f does more explicit: if $f\ i == j$ the transition from state i goes to state j.

[5] An *endo-function* is a function from a set X to itself: $f : X \to X$.

To write the matrix associated to f, we have to compute what vector is associated to each canonical base vector vector:

$$M = [f\ (e\ 0), f\ (e\ 1), ..., f\ (e\ n)]$$

Therefore:

$$M = \begin{array}{c} \\ r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Starting with a canonical base vector $e\ i$, we obtain $M * e\ i = e\ (f\ i)$, as we would expect.

The more interesting thing is if we start with something different from a basis vector, say $[0, 0, 1, 0, 1, 0, 0]$ == $e\ 2 + e\ 4$. We obtain $\{f\ 2, f\ 4\} = \{5, 6\}$, the image of $\{2, 4\}$ through f. In a sense, we can say that the two computations were done in parallel. But that is not quite accurate: if start with $\{3, 4\}$, we no longer get the characteristic function of $\{f\ 3, f\ 4\} = \{6\}$, instead, we get a vector that does not represent a characteristic function at all: $[0, 0, 0, 0, 0, 0, 2]$. In general, if we start with an arbitrary vector, we can interpret this as starting with various quantities of some unspecified material in each state, simultaneously. If f were injective, the respective quantities would just gets shifted around, but in our case, we get a more interesting behaviour.

What if we do want to obtain the characteristic function of the image of a subset? In that case, we need to use other operations than the standard arithmetical ones, for example *min* and *max*. The problem is that $(\{0, 1\}, max, min)$ is not a field, and neither is $(\mathbb{R}, max, min)$. This is not a problem if all we want is to compute the evolutions of possible states, but we cannot apply most of the deeper results of linear algebra.

In the example above, we have:

```
newtype G = G Int deriving (Eq, Show)
instance Bounded G where
  minBound = G 0
  maxBound = G 6
instance Enum G where
  toEnum          = G
  fromEnum (G n) = n
instance Num G where
  fromInteger = G ∘ fromInteger
    -- Note that this is just for convenient notation (integer literals),
    -- G should normally not be used with
```

The transition function:

$$f_1\ 0 = 1$$
$$f_1\ 1 = 3$$
$$f_1\ 2 = 5$$
$$f_1\ 3 = 6$$
$$f_1\ 4 = 6$$

$f_1\ 5 = 4$
$f_1\ 6 = 5$

The associated matrix:

$$m_1\ (G\ g') = V\ \$\ \lambda(G\ g) \to g'\ `is`\ f_1\ g$$

Test:

$t1' = mulMV\ m_1\ (e\ 3 + e\ 4)$
$t_1 = toL\ t1'$ -- $[0,0,0,0,0,0,2]$

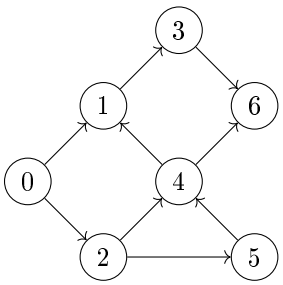
7.2.3 Non-deterministic systems

Another interpretation of the application of M to characteristic functions of a subset is the following: assuming that all I know is that the system is in one of the states of the subset, where can it end up after one step? (this assumes the *max-min* algebra as above).

The general idea for non-deterministic systems, is that the result of applying the step function a number of times from a given starting state is a list of the possible states one could end up in.

In this case, the uncertainty is entirely caused by the fact that we do not know the exact initial state. However, there are cases in which the output of f is not known, even when the input is known. Such situations are modelled by endo-relations: $R : G \to G$, with $g\ R\ g'$ if g' is a potential successor of g. Endo-relations can also be pictured as graphs, but the restriction that every node should be the source of exactly one arrow is lifted. Every node can be the source of one, none, or many arrows.

For example:



Now, starting in 0 we might and up either in 1 or 2 (but not both!). Starting in 6, the system breaks down: there is no successor state.

The matrix associated to R is built in the same fashion: we need to determine what vectors the canonical base vectors are associated with:

$$M = \begin{array}{c} \\ r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Exercise 7.6: start with $e\ 2 + e\ 3$ and iterate a number of times, to get a feeling for the possible evolutions. What do you notice? What is the largest number of steps you can make before the result is the origin vector? Now invert the arrow from 2 to 4 and repeat the exercise. What changes? Can you prove it?

Implementation:

The transition function has type $G \to (G \to Bool)$:

$$
\begin{aligned}
&f_2\ 0\ g = g == 1 \vee g == 2\\
&f_2\ 1\ g = g == 3\\
&f_2\ 2\ g = g == 4 \vee g == 5\\
&f_2\ 3\ g = g == 6\\
&f_2\ 4\ g = g == 1 \vee g == 6\\
&f_2\ 5\ g = g == 4\\
&f_2\ 6\ g = False
\end{aligned}
$$

The associated matrix:

$$m_2\ (G\ g') = V\ \$\ \lambda(G\ g) \to f_2\ g\ g'$$

We need a *Num* instance for *Bool* (not a field!):

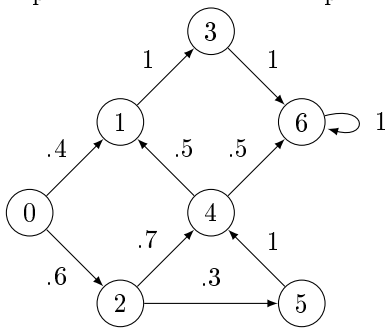
$$
\begin{aligned}
&\textbf{instance}\ Num\ Bool\ \textbf{where}\\
&\quad (+) = (\vee)\\
&\quad (*)\ = (\wedge)\\
&\quad fromInteger\ 0 = False\\
&\quad fromInteger\ 1 = True\\
&\quad negate\ \ \ \ \ \ = \neg\\
&\quad abs\ \ \ \ \ \ \ \ = id\\
&\quad signum\ \ \ \ \ = id
\end{aligned}
$$

Test:

$$
\begin{aligned}
&t2' = mulMV\ m_2\ (e\ 3 + e\ 4)\\
&t_2 = toL\ t2' \quad \text{-- } [False, True, False, False, False, False, True]
\end{aligned}
$$

7.2.4 Stochastic systems

Quite often, we have more information about the transition to possible future states. In particular, we can have *probabilities* of these transitions. For example



One could say that this case is a generalisation of the previous one, in which we can take all probabilities to be equally distributed among the various possibilities. While this is plausible, it

is not entirely correct. For example, we have to introduce a transition from state 6 above. The nodes must be sources of *at least* one arrow.

In the case of the non-deterministic example, the "legitimate" inputs were characteristic functions, i.e., the "vector space" was $G \to \{0,1\}$ (the scare quotes are necessary because, as discussed, the target is not a field). In the case of stochastic systems, the inputs will be *probability distributions* over G, that is, functions $p : G \to [0,1]$ with the property that

$$sum\ [p\ g \mid g \leftarrow [0\,..\,6]] = 1$$

If we know the current probability distributions over states, then we can compute the next one by using the *total probability formula*, normally expressed as

$$p\ a = sum\ [p\ (a \mid b) * p\ b \mid b \leftarrow [0\,..\,6]]$$

This formula in itself would be worth a lecture. For one thing, the notation is extremely suspicious. $(a \mid b)$, which is usually read "a, given b", is clearly not of the same type as a or b, so cannot really be an argument to p. For another, the $p\ a$ we are computing with this formula is not the $p\ a$ which must eventually appear in the products on the right hand side. I do not know how this notation came about: it is neither in Bayes' memoir, nor in Kolmogorov's monograph.

The conditional probability $p\ (a \mid b)$ gives us the probability that the next state is a, given that the current state is b. But this is exactly the information summarised in the graphical representation. Moreover, it is clear that, at least formally, the total probability formula is identical to a matrix-vector multiplication.

As usual, we write the associated matrix by looking at how the canonical base vectors are transformed. In this case, the canonical base vector $e\ i = \lambda j \to i\ `is`\ j$ is the probability distribution *concentrated* in i. This means that the probability to be in state i is 100% and the probability of being anwhere else is 0.

$$M = \begin{array}{c} \\ r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{array} \begin{array}{c} \begin{array}{ccccccc} c_0 & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ .4 & 0 & 0 & 0 & .5 & 0 & 0 \\ .6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .7 & 0 & 0 & 1 & 0 \\ 0 & 0 & .3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & .5 & 0 & 1 \end{pmatrix} \end{array}$$

Exercise 7.7: starting from state 0, how many steps do you need to take before the probability is concentrated in state 6? Reverse again the arrow from 2 to 4. What can you say about the long-term behaviour of the system now?

Exercise 7.8: Implement the example. You will need to define:

The transition function

$f_3 :: G \to (G \to Double)$ -- but we want only $G \to (G \to [0,1])$, the unit interval
$f_3\ g\ g' = undefined$ -- the probability of getting to g' from g

The associated matrix

$m_3 :: G \to (G \to Double)$
$m_3\ g'\ g = undefined$

7.3 Monadic dynamical systems

This section is not part of the intended learning outcomes of the course, but it presents a useful unified view of the three previous sections which could help your understanding.

All the examples of dynamical systems we have seen in the previous section have a similar structure. They work by taking a state (which is one of the generators) and return a structure of possible future states of type G:

- deterministic: there is exactly one possible future states: we take an element of G and return an element of G. The transition function has the type $f : G \to G$, the structure of the target is just G itself.
- non-deterministic: there is a set of possible future states, which we have implemented as a characteristic function $G \to \{0, 1\}$. The transition function has the type $f : G \to (G \to \{0, 1\})$. The structure of the target is the *powerset* of G.
- stochastic: given a state, we compute a probability distribution over possible future states. The transition function has the type $f : G \to (G \to [0, 1])$, the structure of the target is the probability distributions over G.

Therefore:

- deterministic: $f : G \to Id\ G$
- non-deterministic: $f : G \to Powerset\ G$, where $Powerset\ G = G \to \{0, 1\}$
- stochastic: $f : G \to Prob\ G$, where $Prob\ G = G \to [0, 1]$

We have represented the elements of the various structures as vectors. We also had a way of representing, as structures of possible states, those states that were known precisely: these were the canonical base vectors $e\ i$. Due to the nature of matrix-vector multiplication, what we have done was in effect:

$$
\begin{aligned}
& M * v \quad \text{-- v represents the current possible states} \\
= \ & \{\text{- } v \text{ is a linear combination of the base vectors -}\} \\
& M * (v\ 0 * e\ 0 + ... + v\ n * e\ n) \\
= \ & \{\text{- homomorphism -}\} \\
& v\ 0 * (M * e\ 0) + ... + v\ n * (M * e\ n) \\
= \ & \{\text{- } e\ i \text{ represents the perfectly known current state } i\text{, therefore } M * e\ i = f\ i \text{ -}\} \\
& v\ 0 * f\ 0 + ... + v\ n * f\ n
\end{aligned}
$$

So, we apply f to every state, as if we were starting from precisely that state, obtaining the possible future states starting from that state, and then collect all these hypothetical possible future states in some way that takes into account the initial uncertainty (represented by $v\ 0$, ..., $v\ n$) and the nature of the uncertainty (the specific $+$ and $*$).

If you examine the types of the operations involved

$$e : G \to Possible\ G$$

and

$$Possible\ G \to (G \to Possible\ G) \to Possible\ G$$

you see that they are very similar to the monadic operations

$$\begin{aligned}&\mathit{return} : g \to m\ g\\ &(\gg\!=)\ \ : m\ g \to (g \to m\ g') \to m\ g'\end{aligned}$$

which suggests that the representation of possible future states might be monadic. Indeed, that is the case.

Since we implemented all these as matrix-vector multiplications, this raises the question: is there a monad underlying matrix-vector multiplication, such that the above are instances of it (obtained by specialising the scalar type S)?

Exercise: write *Monad* instances for *Id*, *Powerset*, *Prob*.

7.4 The monad of linear algebra

The answer is yes, up to a point. Haskell *Monad*s, just like *Functor*s, require *return* and $\gg\!=$ to be defined for every type. This will not work, in general. Our definition will work for *finite types* only.

$$\begin{aligned}&\textbf{type}\ S = \mathit{Double}\\ &\textbf{newtype}\ \mathit{Vector}\ s\ g = V\ (g \to s)\ \textbf{deriving}\ \mathit{Num}\\ &\mathit{toF}\ (V\ v) \qquad\qquad = v\\ &\\ &\textbf{class}\quad\ (\mathit{Bounded}\ a, \mathit{Enum}\ a, \mathit{Eq}\ a) \Rightarrow \mathit{Finite}\ a\ \textbf{where}\\ &\textbf{instance}\ (\mathit{Bounded}\ a, \mathit{Enum}\ a, \mathit{Eq}\ a) \Rightarrow \mathit{Finite}\ a\ \textbf{where}\\ &\\ &\textbf{class}\ \mathit{FinFunc}\ f\ \textbf{where}\\ &\quad \mathit{func} :: (\mathit{Finite}\ a, \mathit{Finite}\ b) \Rightarrow (a \to b) \to f\ a \to f\ b\\ &\\ &\textbf{instance}\ \mathit{Num}\ s \Rightarrow \mathit{FinFunc}\ (\mathit{Vector}\ s)\ \textbf{where}\\ &\quad \mathit{func} = \mathit{funcV}\\ &\\ &\mathit{funcV} :: (\mathit{Finite}\ g, \mathit{Eq}\ g', \mathit{Num}\ s) \Rightarrow (g \to g') \to \mathit{Vector}\ s\ g \to \mathit{Vector}\ s\ g'\\ &\mathit{funcV}\ f\ (V\ v) = V\ (\lambda g' \to \mathit{sum}\ [v\ g \mid g \leftarrow [\mathit{minBound}\,..\,\mathit{maxBound}], g' == f\ g])\\ &\\ &\textbf{class}\ \mathit{FinMon}\ f\ \textbf{where}\\ &\quad \mathit{embed}\ \ :: \mathit{Finite}\ a \Rightarrow a \to f\ a\\ &\quad \mathit{bind}\ \ \ :: (\mathit{Finite}\ a, \mathit{Finite}\ b) \Rightarrow f\ a \to (a \to f\ b) \to f\ b\\ &\\ &\textbf{instance}\ \mathit{Num}\ s \Rightarrow \mathit{FinMon}\ (\mathit{Vector}\ s)\ \textbf{where}\\ &\quad \mathit{embed}\ a \qquad = V\ (\lambda a' \to \textbf{if}\ a == a'\ \textbf{then}\ 1\ \textbf{else}\ 0)\\ &\quad \mathit{bind}\ (V\ v)\ f = V\ (\lambda g' \to \mathit{sum}\ [\mathit{toF}\ (f\ g)\ g' * v\ g \mid g \leftarrow [\mathit{minBound}\,..\,\mathit{maxBound}]])\end{aligned}$$

A better implementation, using associated types, is in file *Vector.lhs* in the repository.

Exercises:

a. Prove that the functor laws hold, i.e.

$$\begin{aligned}&\mathit{func}\ \mathit{id} \qquad = \mathit{id}\\ &\mathit{func}\ (g \circ f) = \mathit{func}\ g \circ \mathit{func}\ f\end{aligned}$$

b. Prove that the monad laws hold, i.e.

$$\begin{aligned}&\mathit{bind}\ v\ \mathit{return} \qquad = v\\ &\mathit{bind}\ (\mathit{return}\ g)\ f = f\ g\\ &\mathit{bind}\ (\mathit{bind}\ v\ f)\ h = \mathit{bind}\ v\ (\lambda g' \to \mathit{bind}\ (f\ g')\ h)\end{aligned}$$

c. What properties of S have you used to prove these properties? Define a new type class *GoodClass* that accounts for these (and only these) properties.

7.5 Associated code

The scalar product of two vectors is a good building block for matrix multiplication:

$$
\begin{aligned}
&dot :: (Finite\ g, Num\ s) \Rightarrow \\
&\quad (g \to s) \to (g \to s) \to s \\
&dot\ v\ w = sum\ (map\ (v * w)\ [minBound\,..\,maxBound])
\end{aligned}
$$

Note that $v * w :: g \to s$ is using the *FunNumInst*.

Using it we can shorten the definition of *mulMV*

$$
\begin{aligned}
&\quad mulMV\ m\ v\ g' \\
&= \quad \text{-- Earlier definition} \\
&\quad sum\ [m\ g'\ g * v\ g \mid g \leftarrow [minBound\,..\,maxBound]] \\
&= \quad \text{-- replace list comprehension with } map \\
&\quad sum\ (map\ (\lambda g \to m\ g'\ g * v\ g)\ [minBound\,..\,maxBound]) \\
&= \quad \text{-- use } FunNumInst \text{ for } (*) \\
&\quad sum\ (map\ (m\ g' * v)\ [minBound\,..\,maxBound]) \\
&= \quad \text{-- Def. of } dot \\
&\quad dot\ (m\ g')\ v
\end{aligned}
$$

Thus, we can defined matrix-vector multiplication by

$$mulMV\ m\ v\ g' = dot\ (m\ g')\ v$$

We can even go one step further:

$$
\begin{aligned}
&\quad mulMV\ m\ v \\
&= \quad \text{-- Def.} \\
&\quad \lambda g' \to dot\ (m\ g')\ v \\
&= \quad \text{-- } dot \text{ is commutative} \\
&\quad \lambda g' \to dot\ v\ (m\ g') \\
&= \quad \text{-- Def. of } (\circ) \\
&\quad dot\ v \circ m
\end{aligned}
$$

to end up at

$$
\begin{aligned}
&mulMV' :: (Finite\ g, Num\ s) \Rightarrow \\
&\qquad Mat\ s\ g\ g' \to Vec\ s\ g \to Vec\ s\ g' \\
&mulMV'\ m\ v = dot\ v \circ m \\
&\textbf{type}\ Mat\ s\ r\ c = c \to r \to s \\
&\textbf{type}\ Vec\ s\ r = r \to s
\end{aligned}
$$

7.6 Exercises

Search the chapter for tasks marked "Exercise".

Exercise 7.1. Compute $((M*) \circ e)\ g\ g'$.

Exercise 7.2. Matrix-matrix multiplication is defined in order to ensure a homomorphism from $(*)$ to $(\circ)$.

$$\forall\ M.\ \ \forall\ M'.\ \ ((M' * M)*) == (M'*) \circ (M*)$$

or in other words

$$H_2\ ((*), (*), (\circ))$$

Work out the types and expand the definitions to verify that this claim holds. Note that one $(*)$ is matrix-vector multiplication and the other is matrix-matrix multiplication.

Exercise 7.3. Show that matrix-matrix multiplication is associative.

Exercise 7.4. With $G = \mathbb{N}$ for the set of indices, write the (infinite-dimensional) matrix representing D for power series.

Exercise 7.5. Write the matrix I_n associated with integration of polynomials of degree n.

Exercise 7.6. In the context of section 7.2.3: start with $v_0 = e\ 2 + e\ 3$ and iterate $M*$ a number of times, to get a feeling for the possible evolutions. What do you notice? What is the largest number of steps you can make before the result is the origin vector (just zero)?

Now change M to M' by inverting the arrow from 2 to 4 and repeat the exercise. What changes? Can you prove it?

Exercise 7.7. In the context of the example matrix M in section 7.2.4: starting from state 0, how many steps do you need to take before the probability is concentrated in state 6?

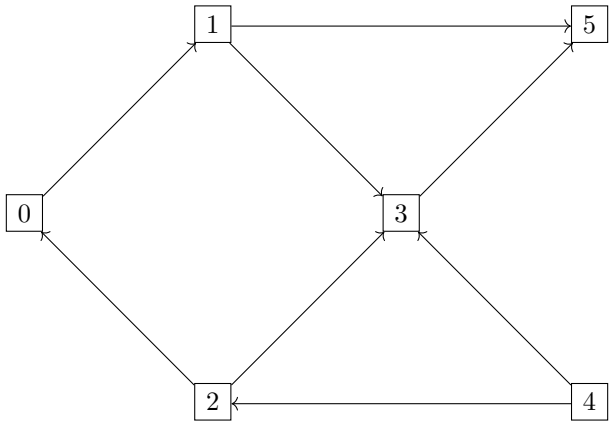
Now change M to M' by inverting the arrow from 2 to 4 and repeat the exercise. What can you say about the long-term behaviour of the system now?

Exercise 7.8. In the context of the example matrix M in section 7.2.4: implement the example. You will need to define the transition function of type $G \to (G \to [0,1])$ returning the probability of getting from g to g', and the associated matrix.

7.6.1 Exercises from old exams

Exercise 7.9. *From exam 2017-03-14*

Consider a non-deterministic system with a transition function $f : G \to [G]$ (for $G = \{0..5\}$) represented in the following graph



The transition matrix can be given the type $m :: G \to (G \to \mathit{Bool})$ and the canonical vectors have type $e\ i :: G \to \mathit{Bool}$ for i in G.

a. (General questions.) What do the canonical vectors represent? What about non-canonical ones? What are the operations on *Bool* used in the matrix-vector multiplication?

b. (Specific questions.) Write the transition matrix m of the system. Compute, using matrix-vector multiplication, the result of three steps of the system starting in state 2.