FLABloM: Functional linear algebra with block matrices

Adam Sandberg Eriksson

Chalmers University of Technology, Sweden
saadam@chalmers.se

Abstract

We define a block based matrix representation in Agda and lift various algebraic structures (semi-near-rings, semi-rings and closed semi-rings) to matrices in order to verify algorithms that can be implemented using the closure operation in a semi-ring.

Introduction

In [1] Bernardy and Jansson used a nice formulation of matrices to certify Valiant's parsing algorithm. Their matrix formulation was restricted to matrices of size $2^n \times 2^n$ and this work extends the matrix formulation to allow for all sizes of matrices and applies the techniques to other algorithms that can be described as closed semi-near-rings with inspiration from [2] and [3].

We define a hierarchy of ring structures as Agda records. A semi-near-ring for some type s needs an equivalence relation $\simeq_s$, a distinguished element 0_s and operations addition $+_s$ and multiplication $\cdot_s$. Our semi-near-ring requires proofs that

- 0_s and $+_s$ form a commutative monoid (i.e. $+_s$ commutes and 0_s is the left and right identity of $+_s$),
- 0_s is the left and right zero of $\cdot_s$,
- $+_s$ is idempotent ($\forall\ x \to x\ +_s\ x\ \simeq_s\ x$) and
- $\cdot_s$ distributes over $+_s$.

For the semi-ring we extend the semi-near-ring with an element 1_s and proofs that $\cdot_s$ is associative and that 1_s is the left and right identity of $\cdot_s$.

Finally we extend the semi-ring with an operation *closure* that computes the transitive closure of an element of the semi-ring (c is the closure of w if $c\ \simeq_s\ 1_s\ +_s\ w\ \cdot_s\ c$ holds), we denote the closure with *.

We use two examples of closed semi-rings: (1) the booleans with disjunction as addition, conjunction as multiplication and the closure being *true*; and (2) the natural numbers ($\mathbb{N}$) extended with an element ∞, we let $0_s\ =\ \infty$, $1_s\ =\ \mathit{0}$, *min* plays the role of $+_s$, addition of natural numbers the role of $\cdot_s$ and the closure is 0.

Matrices

To represent the dimensions of matrices we use a datatype of non-empty binary trees:

$$\begin{aligned}&\textbf{data}\ \mathit{Shape}\ :\ \mathit{Set}\ \textbf{where}\\&\quad L\ :\ \mathit{Shape}\\&\quad B\ :\ (s_1\ s_2\ :\ \mathit{Shape}) \to \mathit{Shape}\end{aligned}$$

This representation follows the structure of the matrix representation closer than natural numbers and we can easily compute the corresponding natural number:

$$\mathit{toNat}\ :\ \mathit{Shape} \to \mathbb{N};\ \mathit{toNat}\ L\ =\ \mathit{1};\ \mathit{toNat}\ (B\ l\ r)\ =\ \mathit{toNat}\ l + \mathit{toNat}\ r$$

while the other direction is slightly more complicated because we want a somewhat balanced tree and we have no translation of 0.

Matrices are parametrised by the type of elements they contain and indexed by a *Shape* for each dimension. 1-by-1 matrices lift the element into a matrix

$$
\begin{array}{l}
\textbf{data}\ M\ (a\ :\ \mathit{Set})\ :\ (\mathit{rows}\ \mathit{cols}\ :\ \mathit{Shape}) \to \mathit{Set}\ \textbf{where} \\
\quad \mathit{One}\ :\ a \to M\ a\ L\ L
\end{array}
$$

Row and column matrices are built from smaller matrices which are either 1-by-1 matrices or further row respectively column matrices

$$
\begin{array}{ll}
\mathit{Row}\ : & \{c_1\ c_2\ :\ \mathit{Shape}\} \to \\
 & M\ a\ L\ c_1 \to M\ a\ L\ c_2 \to \\
 & M\ a\ L\ (B\ c_1\ c_2) \\
\mathit{Col}\ : & \{r_1\ r_2\ :\ \mathit{Shape}\} \to \\
 & M\ a\ r_1\ L \to \\
 & M\ a\ r_2\ L \to \\
 & M\ a\ (B\ r_1\ r_2)\ L
\end{array}
$$

and matrices of other shapes are built from 4 smaller matrices, like $X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$ where $X_{11}, X_{12}, X_{21}, X_{22}$ are again matrices.

$$
\begin{array}{ll}
Q\ : & \{r_1\ r_2\ c_1\ c_2\ :\ \mathit{Shape}\} \to \\
 & M\ a\ r_1\ c_1 \to M\ a\ r_1\ c_2 \to \\
 & M\ a\ r_2\ c_1 \to M\ a\ r_2\ c_2 \to \\
 & M\ a\ (B\ r_1\ r_2)\ (B\ c_1\ c_2)
\end{array}
$$

This matrix representation allows for simple formulations of matrix addition, multiplication, and as we will see also the transitive closure of a matrix.

Transitive closure of matrices In [3] Lehmann presents a definition of the closure on square matrices, $A^* = 1 + A \cdot A^*$: Given

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

the transitive closure of A is defined inductively as

$$A^* = \begin{bmatrix} A_{11}^* + A_{11}^* \cdot A_{12} \cdot \Delta^* \cdot A_{21} \cdot A_{11}^* & A_{11}^* \cdot A_{12} \cdot \Delta^* \\ \Delta^* \cdot A_{21} \cdot A_{11}^* & \Delta^* \end{bmatrix}$$

where $\Delta = A_{22} + A_{21} \cdot A_{11}^* \cdot A_{12}$ and the base case being the 1-by-1 matrix where we use the transitive closure of the element of the matrix:

$$[s]^* = [s^*]$$

Acknowledgements Many thanks to Patrik Jansson for for the inspiration for this project and guidance in implementation.

References

[1] Jean-Philippe Bernardy and Patrik Jansson. Certified context-free parsing: A formalisation of Valiant's algorithm in Agda. *Logical Methods in Computer Science*, 2016. Accepted 2015-12-22 for publication in LMCS.

[2] Stephen Dolan. Fun with semirings: A functional pearl on the abuse of linear algebra. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming*, ICFP '13, pages 101–110, New York, NY, USA, 2013. ACM.

[3] Daniel J. Lehmann. Algebraic structures for transitive closure. *Theoretical Computer Science*, 4(1):59–76, 1977.