An Agda Formalisation of the Transitive Closure of Block Matrices (Extended Abstract)

Adam Sandberg Eriksson Patrik Jansson
Chalmers University of Technology, Sweden
{saadam,patrikj}@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.

Categories and Subject Descriptors D.1.1 [*Programming Techniques*]: Applicative (Functional) Programming; F.3.1 [*Logics and Meanings of Programs*]: Specifying and Verifying and Reasoning about Programs

Keywords Dependent types, Linear Algebra

1. Introduction

Bernardy and Jansson (2016) used a recursive block formulation of matrices to certify Valiant's parsing algorithm (Valiant 1975). 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 similar techniques to algorithms that can be described as transitive closures of semi-rings of matrices with inspiration from (Dolan 2013) and (Lehmann 1977).

Development Structure To structure the formal development 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 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 another distinguished 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 Kleene star (reflexive and 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 of w with w^*.

We use two examples of semi-rings with transitive closure: (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 $0_s = \infty$, $1_s = 0$, min plays the role of $+_s$, addition of natural numbers the role of $\cdot_s$ and the closure is 0.

TyDe'16, September 18, 2016, Nara, Japan

http://dx.doi.org/10.1145/2976022.2976025

2. Shapes, Matrices and Closure

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

```
data Shape : Set where
  L : Shape
  B : Shape → Shape → Shape
```

This representation follows the structure of the (block) matrix representation more closely than natural numbers and we can easily compute the corresponding natural number:

```
toNat : Shape → ℕ
toNat L       = 1
toNat (B l r) = toNat l + toNat r
```

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

Matrices are parametrised by the type of elements they contain and indexed by a $Shape$ for each dimension. We use a datatype M with four constructors: One, Row, Col, and Q. The first One lifts an element into a 1-by-1 matrix:

```
data M (a : Set) : (rows cols : Shape) → Set where
  One : a → M a L L
```

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

```
Row : {c₁ c₂ : Shape} →
      M a L c₁ → M a L c₂ → M a L (B c₁ c₂)
Col : {r₁ r₂ : Shape} →
      M a r₁ L → M a r₂ L → M a (B r₁ r₂) L
```

and matrices of other shapes are built from 2×2 smaller matrices

```
Q   : {r₁ r₂ c₁ c₂ : Shape} →
      M a r₁ c₁ → M a r₁ c₂ →
      M a r₂ c₁ → M a r₂ c₂ →
      M a (B r₁ r₂) (B c₁ c₂)
```

This matrix representation allows us to lift a semi-ring to a semi-ring of matrices and allows for intuitive definitions of matrix operations and proofs that the lifted structures satisfy the laws of our semi-rings.

To give a taste of the formal development we include one simple proof and a fragment of the larger development. The proof examples show how we use the Agda standard library's equational reasoning framework to make the proofs easier to write and read, this tool is used heavily throughout the development. To prove that the zero matrix is the right identity of addition we use commutativity of

addition and the proof of the left identity of addition (which itself is a proof by cases on the shapes of the matrix):

```
ident^r_S : (r : Shape) (c : Shape) (x : M s r c) →
            x +_S 0_S r c ≃_S x
ident^r_S r c x =
  let open EqReasoning setoidS
  in begin
    x +_S 0_S r c
  ≈⟨ comm_S r c x (0_S r c) ⟩
    0_S r c +_S x
  ≈⟨ ident^l_S r c x ⟩
    x
  ■
```

The second proof example (in Fig. 1) shows how we use local modules ($lemma_1$) and abbreviations (X) to make the proof terms resemble hand-written proofs.

Transitive Closure Lehmann (1977) presents a definition of the closure on square matrices, $A^* = 1 + A \cdot A^*$: Given a square matrix

$$A = \boxed{\begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array}}$$

the transitive closure of A is defined inductively as

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

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

We have encoded this definition of closure in Agda and implemented a constructive correctness proof using structural induction and equational reasoning. The full development of around 2500 lines of literate Agda code (including this abstract) is available on GitHub (`https://github.com/DSLsofMath/FLABloM`).

Example: Graph Reachability Using this definition of transitive closure of matrices instantiated with the boolean semi-ring defined above we get an implementation of a graph reachability algorithm. If we have a graph (Fig. 2a) and its adjacency matrix (as below) we can find all reachable nodes (Fig. 2b) by computing the transitive closure of the adjacency matrix.

$$\left(\begin{array}{|cc|cc|}\hline 0&0&0&0\\0&0&0&1\\\hline 0&1&0&0\\0&0&0&0\\\hline\end{array}\right)^* = \left(\begin{array}{|cc|cc|}\hline 1&0&0&0\\0&1&0&1\\\hline 0&1&1&1\\0&0&0&1\\\hline\end{array}\right)$$

3. Conclusions

We have presented an algebraic structure useful for (block) matrix computations and implemented and proved correctness of reflexive transitive closure. Compared to (Bernardy and Jansson 2016) our implementation handles arbitrary matrix dimensions but is restricted to semi-rings. Future work would be to extend the proof to cover both arbitrary dimensions and the more general semi-near-ring structure which would allow parallel parsing as an application.

Acknowledgments

This work was partially supported by the projects GRACeFUL (grant agreement No 640954) and CoeGSS (grant agreement No 676547), which have received funding from the European Union's Horizon 2020 research and innovation programme.

```
module lemma_1
  (sh sh_1 : Shape)
  (C C* : M s sh sh)
  (D    : M s sh sh_1)
  (E    : M s sh_1 sh)
  (Δ*   : M s sh_1 sh_1)
  (ih   : C* ≃_S I + C * C*) where
  X = D * Δ* * E * C*
  entire-lem_1 : C* * X ≃_S C * C* * X + X
  entire-lem_1 =
    let open EqReasoning setoidS
    in begin
        C*              * X
      ≈⟨ <> sh sh sh ih (refl_S sh sh) ⟩
        (I + C * C*)  * X
      ≈⟨ distr_S X I (C * C*) ⟩
        I * X + (C * C*) * X
      ≈⟨ <+> sh sh
            (*-identl_S X)
            (*-assoc_S sh sh sh sh C C* X) ⟩
        X +   C * C* * X
      ≈⟨ comm_S sh sh X (C * C* * X) ⟩
        C * C* *  X
              + X
      ■
```

Figure 1: Example lemma from the closure proof. We use a local parametrised module to introduce short parameter names and a local definition of the often used subterm X.

(a) A graph

(b) Reachable nodes

Figure 2: Graph with reachable nodes

References

J.-P. Bernardy and P. Jansson. Certified context-free parsing: A formalisation of Valiant's algorithm in Agda. *Logical Methods in Computer Science*, 12, 2016. doi: 10.2168/LMCS-12(2:6)2016.

S. 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. ISBN 978-1-4503-2326-0. doi: 10.1145/2500365.2500613.

D. J. Lehmann. Algebraic structures for transitive closure. *Theoretical Computer Science*, 4(1):59–76, 1977. ISSN 0304-3975. doi: 10.1016/0304-3975(77)90056-1.

L. G. Valiant. General context-free recognition in less than cubic time. *J. of computer and system sciences*, 10(2):308–314, 1975.