Foundations II

Sanjay Rajopadhye

Contents

1 Introduction & Motivation

These notes are a followup to the first set of foundation notes. They cover the "manipulations" or transformations of systems of affine recurrence equations, and also some of the basic material about the closure properties of polyhedra. We first recap the ldefinitions from th eprevious set of notes.

2 Recurrence Equations

In what follows, $\mathcal{Z}$ denotes the set of integers, and $\mathcal{N}$ the set of natural numbers.

Definition 1 *A* ***Recurrence Equation*** *defining a function (variable) X at all points, z, in a domain, D, is an equation of the form*

$$X[z] = D^X \quad : \quad g(\ldots X[f(z)] \ldots) \tag{1}$$

where

- *z is an n-dimensional* ***index variable****.*

- *X is an "n-dimensional"* ***data variable****. There a couple of equivalent alternative ways to view X. It can be thought of as an n-dimensional array whose values at all $z \in D^X$ are implicitly defined by the equation; it may also be seen as a function of n integer arguments.*

- *$f(z)$ is a* ***dependency function*** *(also called an* ***index*** *or* ***access*** *function), $f : \mathcal{Z}^n \to \mathcal{Z}^n$;*

- *the "$\ldots$" indicate that g may have other arguments, each with the same syntax;*

- *g is a strict, single-valued function; it is often written implicitly as an* expression *involving operands of the form $X[f(z)]$ combined with basic operators and parentheses. Note that for analysis purposes, g is considered atomic (i.e., executing in a single step) unless it has a* reduction *(as defined later). If it has a reduction it may or may not be considered atomic, depending on the assumptions of the machine model used for the analysis.*

- *D^X is a set of points in $\mathcal{Z}^n$ and is called the* ***domain*** *of the equation. Domains are often polyhedral index spaces, parameterized with one or more, say s size parameters. The parameters are viewed as an s-dimensional vector p.*

A variable may be defined by more than one equation. In this case, we use the syntax shown below:

$$X[z] = \begin{cases} & \vdots \\ D_i \ : & g_i(\ldots X[f(z)] \ldots) \\ & \vdots \end{cases} \tag{2}$$

Each line is called a ***case****, and the domain of X is the union of the (disjoint) domains of all the cases, $D^X = \bigcup_i D_i$.*

Definition 2 *A recurrence equation (1) as defined above, is called an* ***Affine Recurrence Equation*** *(*ARE*) if every dependence function is of the form, $f(z) = Az + Bp + a$, where A (respectively B) is a constant $n \times n$ (respectively, $n \times l$) matrix and a is a constant n-vector. It is said to be a* ***Uniform Recurrence Equation*** *(*URE*) if it is of the form, $f(z) = z + a$, where a is a constant n-dimensional vector, called the dependence vector.* URE*s are a proper subset of* ARE*s, where A is the identity matrix and $B = 0$.*

Definition 3 *A **system** of recurrence equations (*SRE*) is a set of m such equations, defining the data variables $X_1 \dots X_m$. Each variable, X_i is of dimension n_i, and since the equations may now be mutually recursive, the dependence functions f must now have the appropriate type.*

Problem Think of the above definitions of recurrence equations as an informal syntax of an equational "programming" language. Before reading any further, please try to describe some of the examples of Section 1 as SREs. Explain what difficulties you encounter.

Reductions

The key difficulty you must have encountered is that we have no syntax for the *reduction* operations: associative and commutative operators like addition, multiplication, max, etc., applied to a collection of values.

We will now introduce a simple, yet powerful syntax for this. We simply allow the function g to have the form, $\texttt{reduce}(\texttt{op}, f', \texttt{expr})$. Here,

- `op` is an associative and commutative operator;
- `expr` is an expression (it is most convenient to assume that the expression is just a new variable, Y, and to assume that there is an equation $Y = \texttt{expr}$ defined over an appropriate domain D^Y);
- f' is a many-to-one mapping from indices to indices, usually it maps $\mathcal{Z}^n$ to $\mathcal{Z}^{n-k}$ (where the `expr` is n-dimensional).

Consider a reduction equation as follows.

$$X(z) = \texttt{reduce}(\texttt{op}, f', Y)$$

Its semantics can be explained as follows. X is defined over a domain D^X which is the image of D^Y by the function f' (this implies D^X is $n-k$-dimensional). Because f' is many-to-one, each $z \in D^X$ is the image of many points $z' \in D^Y$. The reduce expression states that the value of X at any point z is obtained by applying `op` to the values of Y at all the z' that are mapped by f' to z (this is a mouthful; please read each word carefully to make sure you understand what this says).

With this explanation, we can now write an SRE for the forward substitution example:

$$x[i] \;=\; \begin{cases} i = 1 & : \; b[i] \\ i > 1 & : \; b[i] - \texttt{reduce}(+, (i,j \rightarrow i), T[i,j]) \end{cases} \tag{3}$$

$$T[i,j] \;=\; \{i,j \mid 1 \leq j < i \leq n\} : L_{i,j}x_j \tag{4}$$

Taxonomy of Recurrence Equations

As we have seen above, recurrence equations may be classified along many aspects:

- single or system;
- class of dependence functions: arbitrary, affine or uniform;
- parameterized domains or single domain;
- class of domains over which they are defined.

3 "Manipulating" Equations

We now describe what we can do with equations, in a more formal manner, as well as the tools that such manipulation requires.

Domains, Polyhedra and Representations

The (data) variables defined in an SRE may also be viewed as multidimensional array variables (stored in some memory locations if the SRE is viewed as a recursively evaluated program), or alternatively as mappings from tuples of indices to values. Hence, the domains over which the SREs are defined play a crucial role in our analysis.

Note that these domains consist of integer-valued points, and the conventional mathematical notions of polyhedra are typically over the reals or rationals. This sometimes introduces subtle problems. For example, the image of a rational polyhedron by an affine function is a always a rational polyhedron, but this closure property does not hold for integer polyhedra: e.g. what is the image of the polyhedron, $\{i,j \mid i = 2j\}$ by the function $(i,j \rightarrow i)$?).

We have already seen instances of domains in our informal examples, where we simply used set-theoretic notation to describe them: $\{z \in \mathcal{Z}^n \mid P(z)\}$, where P is some predicate (Boolean function of z, and possibly the size parameters). The syntax of domains is

$$\{i_1, \ldots i_n \mid c_1, \ldots c_m\}$$

where each c_i is a single *constraint* (Boolean predicate) on the indices, and the comma is assumed to mean conjunction (logical and).

We have also seen situations (e.g. branches of a "case") where we have not *completely* specified the entire domain but have given only the pertinent condition allowing us to distinguish between the cases under consideration. In general, such sloppiness acceptable if the meaning is clear from the context.

Affine functions An *index function* from $\mathcal{Z}^m$ to $\mathcal{Z}^n$ is written as $(i_1, \ldots i_m \rightarrow e_1, \ldots e_n)$, where the i's are *index names* and the e's are *expressions* involving the i's (and possibly, size parameters). When the e's are linear or affine functions, the index function is often written in matricial form, $(z \rightarrow Az + a)$ or $(z \rightarrow Az + Bp + a)$. Here, A is an $n \times m$ matrix, a is an n-vector, and B is an $n \times s$ matrix. Another useful class of index functions are *piecewise affine* functions which are written with the same "case-like" syntax used for SREs.

Polyhedra

Definition 4 *A* polyhedron *is a set of the form:*

$$P = \{x \in \mathcal{Z}^n \mid Qx \geq q\}$$

where Q is an $m \times n$ integer matrix, and q is an integer m-vector.

A parametric family of polyhedra corresponds to the case where the rhs of each constraint is an affine function of the s size parameters p, i.e., $P = \{z \in \mathcal{Z}^n \mid Qz \geq q - Bp\}$, or equivalently

$$\left\{ \begin{pmatrix} z \\ p \end{pmatrix} \in \mathcal{Z}^{n+s} \mid \begin{bmatrix} Q & B \end{bmatrix} \begin{pmatrix} z \\ p \end{pmatrix} \geq q \right\}$$

Thus, a parameterized family of polyhedra is just a single higher-dimensional polyhedron. Note that the converse view—that *any* single $n + s$ dimensional polyhedron is equivalent to a family, parameterized by s parameters, of n-dimensional polyhedra—is valid for rational polyhedra, but not for integer polyhedra. For example, the projection of an integral polyhedron on one of its axes is not necessarily an integral polyhedron.

Polyhedra also admit an equivalent dual definition in terms of *generators* as defined below.

Definition 5 *A linear combination of given a set of vectors, $v_1, \ldots v_k$, is the sum, $\sum_{i=1}^{k} a_i v_i$, for a set of* constant *coefficients $a_1, \ldots a_k$.*

A positive combination *is a linear combination where the coefficients are all non-negative, i.e., $a_i \geq 0$ for $i = 1 \ldots k$.*

A convex combination *is a positive combination with the additional constraint that the coefficients add up to unity, i.e., $\sum_{i=1}^{k} a_i = 1$*

A polytope, i.e., a bounded polyhedron is uniquely specified as the set of positive combinations of a finite number of points called its vertices (the columns of G below).

$$\{z \in \mathcal{Z}^n \mid z = Ga;\ a_i \geq 0; \sum_i a_i = 1\}$$

If a polyhedron is unbounded, it may have *rays* and *lines.*

Definition 6 *ρ is called a* ray *of a polyhedron, P iff, $\forall z \in P, z + k\rho \in P$ for any $k \geq 0$. A ray may be viewed as a direction along which the polyhedron "extends unboundedly." If a ρ and $-\rho$ are both rays of a polyhedron, they constitute a* line*.*

In general, a polyhedron can be defined in the generator form as a convex combination of its vertices, a positive combination of its rays and a linear combination of its lines, as follows:

$$P = \{z \in \mathcal{Z}^n \mid z = Va + Rb + Lc;\ a_i, b_i \geq 0; \sum_i a_i = 1\}$$

Example 1: Consider $\{i, j, n \mid 0 \leq j \leq i; j \leq n; 1 \leq n\}$ as a 3-dimensional polyhedron. It has two vertices: $[0, 0, 1]$ and $[1, 1, 1]$. Its rays are the vectors in the set $\{[1, 0, 0], [0, 0, 1], [0, 1, 1]\}$. The same polyhedron, viewed as a 2-dimensional polyhedron (parameterized by n) has two vertices, $\{[0, 0], [n, n]\}$ and a ray, $[1, 0]$.

For a single polyhedron, the vertices, rays and lines are constants. For a parametric family polyhedra the vertices, rays and lines are all piece-wise affine functions of the parameters.

Example 2: Consider $\mathcal{P}_1 = \{i, j, n, m \mid 0 \leq j \leq i \leq n; j \leq m; 1 \leq n, m\}$ as a 2-dimensional polyhedron (with parameters n and m). Depending on the relative values of its parameters, it is either a triangle (if $n \leq m$) or a trapezium (otherwise). It does not have rays, and we can see that the set of its vertices is a piece-wise affine function of its parameters:

4 Operations on Domains

If we review the rules of transformation (CoB) of equations and also consider the different ways we may need to manipulate, SREs, we realize that domain should be an abstract data type (ADT) that supports the following operations:

- Intersection
- Union
- Preimage by the class of dependence functions
- Image by the class CoB transformation functions

In addition, we must ensure that

- The CoB Transformation functions must belong to the class of Dependence functions
- Dependence functions are closed under composition

If we do this, then our equations will be "closed" under the kinds of manipulations we seek. We will see how polyhedra support these operations. It turns out that for some operations, the constraint representation is useful, and for others, the generator representation leads to easier manipulation.

Intersection: For arbitrary domains (i.e., sets of integer vectors) the set-theoretic notion of intersection carries through. For polyhedral domains, the constraint representation of polyhedra is obviously suitable, we simply put all the constraints together (and simplify to remove redundant constraints). Let $P_1 = \{x \in \mathcal{Z}^n \mid Q_1 x \geq q_1\}$, and $P_2 = \{x \in \mathcal{Z}^n \mid Q_2 x \geq q_2\}$. Then

$$P_1 \cap P_2 = \left\{ x \in \mathcal{Z}^n \ \mid \ \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} x \geq \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} \right\}$$

Union: For arbitrary domains, the set-theoretic definition holds. For polyhedral domains, the first thing to note is that we do not have closure (the union of two polyhedra in not necessarily a polyhedron) but we can define the convex hull (the smallest polyhedron that contains the union). For computing this, the generator representation is more suitable, and is easiest to see for polytopes (no rays and lines).

The convex hull of the union of P_1 defined by vertices $v_1 \dots v_k$ and P_2 defined by vertices $v'_1 \dots v'_l$ is a polyhedron with vertices belonging to a subset of $v_1 \dots v_k, v'_1 \dots v'_l$. We simply put all the vertices together (and simplify to remove points that can be expressed as the linear combination of the others; there are standard convex hull algorithms).

Image and Preimage: For any set $S \subseteq \mathcal{Z}^n$ and functions $f_1 : \mathcal{Z}^n \to \mathcal{Z}^m$, and $f_2 : \mathcal{Z}^{m'} \to \mathcal{Z}^n$, the image of S by f_1, denoted by $\mathrm{Image}(S, f_1)$, or simply $f_1(S)$ is the set of points obtained by applying f_1 to all the points in S, i.e., $f_1(S) = \{x \in \mathcal{Z}^m \mid x = f_1(z), z \in S\}$. The preimage of S by f_2 denoted by $\mathrm{PreImage}(S, f_2)$, or simply $f_2^{-1}(S)$ is the set of points that are mapped by f_2 to points in S, i.e., $f_2^{-1}(S) = \{x \in \mathcal{Z}^{m'} \mid f_2(x) \in S\}$.

Note that rational polyhedra are closed under image by an affine function, but not integer polyhedra, but again, we can find the convex hull of the image. The generator representation is useful for finding the image and the constraint representation is useful for finding the preimage.

$$\begin{aligned}
\mathcal{A}(z) &\equiv \mathcal{Z}^n \to \mathcal{Z}^m \ : \ z \mapsto Az + a && (5)\\
\mathcal{P} &\equiv \text{polyhedron} \subseteq \mathcal{Z}^m && (6)\\
\mathcal{A}^{-1}(\mathcal{P}) &\equiv \mathrm{PreImage}(\mathcal{P}, \mathcal{A}) && (7)\\
&\equiv \{z \in \mathcal{Z}^n \mid \mathcal{A}(z) \in \mathcal{P}\} && (8)\\
\text{by definition} &= \{z \in \mathcal{Z}^n \mid Az + a \in \mathcal{P}\} && (9)\\
\text{constraint repr.} &= \{z \in \mathcal{Z}^n \mid Q(Az + a) \geq q\} && (10)\\
&= \{z \in \mathcal{Z}^n \mid QAz \geq q - Qa\} && (11)
\end{aligned}$$

thus the preimage is a polyhedron with constraints $\langle QA, q - Qa \rangle$.

$$\begin{aligned}
\mathcal{A}(z) &\equiv \mathcal{Z}^n \to \mathcal{Z}^m \ : \ z \mapsto Az + a && (12)\\
\mathcal{P} &\equiv \text{polyhedron} \subseteq \mathcal{Z}^n && (13)\\
\mathcal{A}(\mathcal{P}) &\equiv \mathrm{Image}(\mathcal{P}, \mathcal{A}) && (14)\\
&\equiv \{\mathcal{A}(z) \in \mathcal{Z}^m \mid z \in \mathcal{P}\} && (15)
\end{aligned}$$

The image is a polyhedron with vertices $\langle AV + a \rangle$ and rays $\langle AR \rangle$. Practical hint: it is also ***preimage*** by $\mathcal{A}^{-1}$.

Change of Basis (CoB)

The most important manipulation that we can perform on an SRE is called a *change of basis* of its variable(s) (also called a *reindexing transformation* or *space-time mapping*). The transformation, $\mathcal{T}$ must admit a *left inverse* for all points in the domain of the variable. When applied to the variable X of an SRE defined as follows:

$$X[z] \quad = \quad \begin{cases} & \vdots \\ D_i^X & : \quad g_i(\ldots Y[f(z)]\ldots) \\ & \vdots \end{cases} \tag{16}$$

the SRE obtained by applying the following rules is provably equivalent to the original:

- Replace each D_i^X by $\mathcal{T}(D_i^X)$, the image of D_i^X by $\mathcal{T}$.
- On the right hand side (rhs) of the equation for X, replace each dependency f by $f \circ \mathcal{T}^{-1}$, the composition[1] of f and $\mathcal{T}^{-1}$.
- In all occurrences $X[g(z)]$ on the rhs of *any* equation, replace the dependency g by $\mathcal{T} \circ g$.

The occurrences of X on the rhs of the equation for X itself constitute a special case where the last two rules are *both* applicable, and we replace the dependency f by $\mathcal{T} \circ f \circ \mathcal{T}^{-1}$.

[1]Recall that function composition is right associative, i.e., $(g \circ h)(z) = g(h(z))$.