Array representations for Polly

(Anonymous)

August 21, 2018

Contents

1 Introduction

We mathematically describe the array representations that are used by Fortran and Chapel, two programming languages which support complex array indexing. We motivate the need for LLVM to be able to represent these array representations and their cooresponding indexing operation at a semantic level that is higher than that of `getelementptr`. We showcase existing implementations of such a semantic representation, and would like to propose this as an LLVM standard. Choices for implementation are outlined in this document.

2 Formalization

We first formalize the most general type of array we have encountered - That of Chapel style arrays which are multidimensional, strided, and provide non-zero-based indexing.

We denote the set of all possible arrays (a loosely defined term, to be sure) as *Array*.

$$\begin{aligned}
&\mathit{dim} : \mathit{Array} \to \mathbb{N} \\
&\mathit{dim}(A) = \text{dimensionality of array } A \\
\\
&\mathit{ixbegin}/\mathit{ixend} : \mathit{Array} \times [0, \mathit{dim}(A)] \to \mathbb{Z} \\
&\mathit{ixbegin}/\mathit{ixend}(A, d) = \text{first and last legal indeces allowed along dimension } d \\
\\
&\mathit{stride} : \mathit{Array} \times [0, \mathit{dim}(A)] \to \mathbb{N} \\
&\mathit{stride}(A, d) = \text{stride of an array } A \text{ along dimension } d
\end{aligned}$$

Using the data outlined above, we can now define what a correct indexing function into an array is. An indexing function recieves **logical coordinates** of the array index, which is converted into **physical coordinates**, which is the in-memory representation.

We now try to define a partial function *logicalToPhysical*, which maps logical coordinates to physical coordinates.

First, we define $\mathit{ixset}(A, d)$, the set of valid indeces along a dimension d:

$$\begin{aligned}
&\mathit{ixset} : \mathit{Array} \times [0, \mathit{dim}(A)] \to 2^{\mathbb{N}} \\
&\mathit{ixset}(A, d) = \{\mathit{ixbegin}(A, d) + \alpha \cdot \mathit{stride}(A, d) \mid \alpha \in \mathbb{N}\} \cap (-\infty, \mathit{ixend}(A, d)]
\end{aligned}$$

The *ixset* constructs the set of valid indeces that are allowed along a given dimension. We first give the type of *logicalToPhysical*, and we then gradually build up the full expression.

$$logicalToPhysical : Array \times \prod_{d=1}^{dim(A)} ixset(A, d) \to \mathbb{N}$$

Notice that the logical coordinates live in $\prod_{d=1}^{dim(A)} ixset(A, d)$, while physical coordinates live in $\mathbb{N}$. We wish to view memory as a 1-D sequence of values, with 0 being the "base address" of the array. The logical coordinates allow us to index the "virtual view" of the array that we posess.

3 C-style arrays — no stride, no offset

In the case of C-style arrays, our $ixset(A, i) = \big[0, N(A, i)\big) \subset \mathbb{N}$, where $N(A, i)$ is informally the "size" of array A along dimension i. The word "size" is, however, fraught with danger, since it conflates the size of the "logical indexing space" and the underlying "memory space". In the case of the C language, both of these concepts coincide, but this is not so in other languages such as Chapel.

In the case of a C array, here are the different functions that we have defined:

- Declaration: `T A[`N_1`][`N_2`]`$\ldots$`[`N_d`]`
- Dimension: $dim(A) = d$
- Beginning, ending indeces: $ixbegin(A, \_) = 0$, $ixend(A, d) = N_d - 1$
- Stride: $stride(A, \_) = 1$
- Index set:

$$\begin{aligned} ixset(A, d) &= \{ixbegin(A, d) + \alpha \cdot stride(A, d) \mid \alpha \in \mathbb{N}\} \cap (-\infty, ixend(A, d)] \\ &= \{0 + \alpha \cdot 1 \mid \alpha \in \mathbb{N}\} \cap (-\infty, N_d - 1] \\ &= [0, \infty) \cap (-\infty, N_d - 1] \\ &= [0, N_d - 1] \end{aligned}$$

To calculate the index expression, we create a function, $\#elem(A, d) = |ixset(A, d)|$. $\#elem$ measures the number of elements present along a given dimension. With this ingredient, we can finally re-create the C array indexing function as:

Sid question: How do you actually motivate this definition?

$$
\begin{aligned}
&\mathit{logicalToPhysical} : \mathit{Array} \times \prod_{d=1}^{\mathit{dim}(A)} \mathit{ixset}(A,d) \to \mathbb{N} \\
&\mathit{logicalToPhysical}(A, (ix_1, ix_2, \ldots, ix_{\mathit{dim}(A)})) = \\
&\sum_{d=1}^{\mathit{dim}(A)} ix_d \times \prod_{\mathit{prevd}=1}^{d-1} \#\mathit{elem}(A, \mathit{prevd})
\end{aligned}
$$

With respect to this formula, we understand that ix_1 is the innermost dimension with an increment of 1, and $ix_{\mathit{dim}(A)}$ is the outermost dimension.

4 Fortran-style arrays: only offset

Let us next consider languages which allow us to create arrays with arbitrary starting and ending indeces, such as Fortran.

– Declaration: `T A[`$B_1 : E_1$`][`$B_2 : E_2$`]`$\ldots$`[`$B_d : E_d$`]`

– Dimension: $\mathit{dim}(A) = d$

– Beginning, ending indeces: $\mathit{ixbegin}(A,d) = B_d$, $\mathit{ixend}(A,d) = E_d$

– Stride: $\mathit{stride}(A, \_) = 1$

– Index set:

$$
\begin{aligned}
\mathit{ixset}(A,d) &= \{\mathit{ixbegin}(A,d) + \alpha \cdot \mathit{stride}(A,d) \mid \alpha \in \mathbb{N}\} \cap (-\infty, \mathit{ixend}(A,d)] \\
&= \{B_d + \alpha \cdot 1 \mid \alpha \in \mathbb{N}\} \cap (-\infty, E_d] \\
&= [B_d, \infty) \cap (-\infty, E_d] \\
&= [B_d, E_d]
\end{aligned}
$$

In this case, the value of $\#\mathit{elem}(A,d) = E_d - B_d + 1$.

$$
\begin{aligned}
&\mathit{logicalToPhysical} : \mathit{Array} \times \prod_{d=1}^{\mathit{dim}(A)} \mathit{ixset}(A,d) \to \mathbb{N} \\
&\mathit{logicalToPhysical}(A, (ix_1, ix_2, \ldots, ix_{\mathit{dim}(A)})) = \\
&\sum_{d=1}^{\mathit{dim}(A)} (ix_d - B_d) \times \prod_{\mathit{prevd}=1}^{d-1} \#\mathit{elem}(A, \mathit{prevd})
\end{aligned}
$$

5 Chapel-style arrays: strides and offsets

$$logicalToPhysical : Array \times \prod_{d=1}^{dim(A)} ixset(A, d) \to \mathbb{N}$$

$$logicalToPhysical(A, (ix_1, ix_2, \ldots, ix_{dim(A)})) =$$

$$\sum_{d=1}^{dim(A)} ((ix_d - B_d)/stride(A, i)) \times \prod_{prevd=1}^{d-1} \#elem(A, prevd)$$

6 Reduction of representation

It seems like we require a lot of information for even a single array: that of *dim*, *ixbegin*, *ixend* and *stride*. We first show to eliminate the requirement of a per-dimension *ixbegin* and *ixend*. We change this information into a per-dimension $\#elem$, and a **single** $offset$ **value for an entire array**.

7 Multidimensional representation for LLVM

We have encoded the final representation of arrays in two experimental projects - one being a fork of LLVM + dragonegg + Polly, and the other a fork of LLVM + Chapel + Polly. The first implements this style of indexing in Fortran, which is interpreted by Polly for polyhedral optimisation, and the second project does the same for Chapel.

We argue that this representation will provide more accurate analysis and optimisation opportunities for LLVM, due to the inherent loss in information by the linearization performed by `getelementptr`. For a detailed reference of the loss of information and the (*necessarily incomplete*) heuristics that are used to regain this information, refer to [1] and [2].

7.1 A "canonical" function call

7.2 A new intrinsic

7.3 A new instruction

7.4 Modifying `getelementptr`

8 Caveats

We do not take into account alignment, which is something that must be done for this document to be complete.

9 References: Links

- The Chapel language specification.
- Fortran 2003 standard.

References

[1] Tobias Grosser, Jagannathan Ramanujam, Louis-Noel Pouchet, Ponnuswamy Sadayappan, and Sebastian Pop. Optimistic delinearization of parametrically sized arrays. In *Proceedings of the 29th ACM on International Conference on Supercomputing*, pages 351–360. ACM, 2015.

[2] Vadim Maslov. Delinearization: an efficient way to break multiloop dependence equations. *ACM Sigplan Notices*, 27(7):152–161, 1992.