Sparse Abstract Machine: Sparse Tensor Algebra as Dataflow Graphs

Olivia Hsu

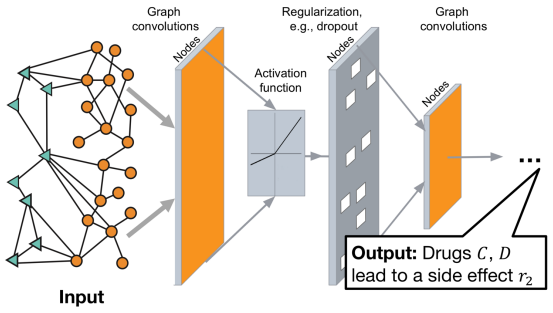
In collaboration with Max Strange, Jaeyeon Won, Ritvik Sharma, Kunle Olukotun, Joel Emer, Mark Horowitz, and Fredrik Kjolstad

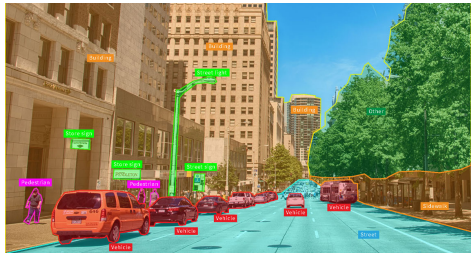
August 25th, 2022

Motivation for dataflow backends that support sparsity

Motivation for dataflow backends that support sparsity

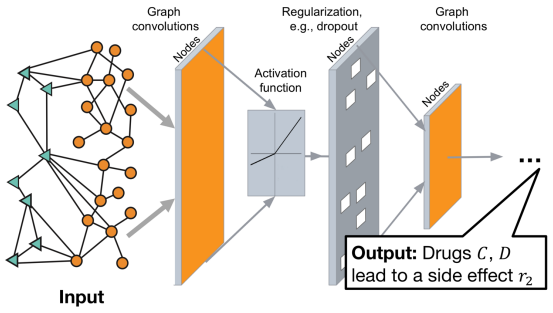
Machine Learning

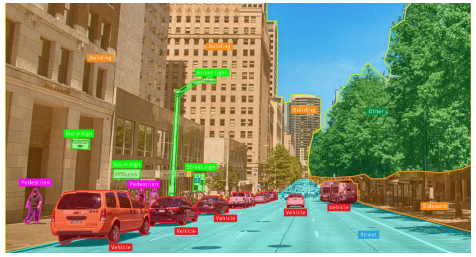




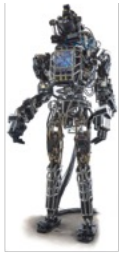
Motivation for dataflow backends that support sparsity

Machine Learning

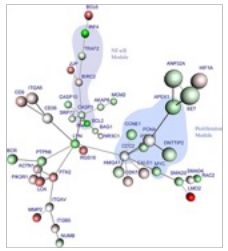




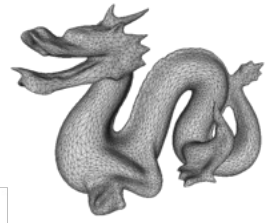
Scientific Computing



Robotics



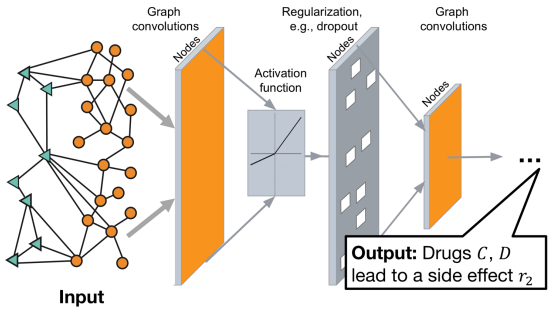
Computational Biology

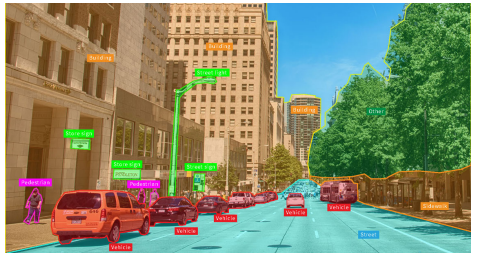


Simulations

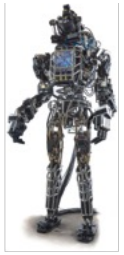
Motivation for dataflow backends that support sparsity

Machine Learning

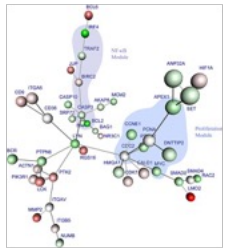




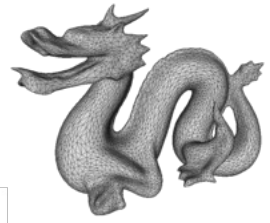
Scientific Computing



Robotics



Computational Biology

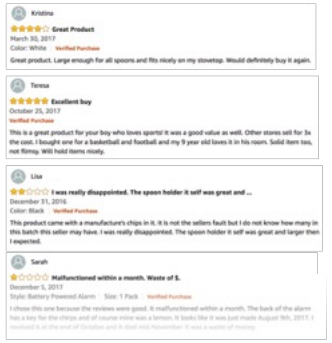


Simulations

Data Analytics

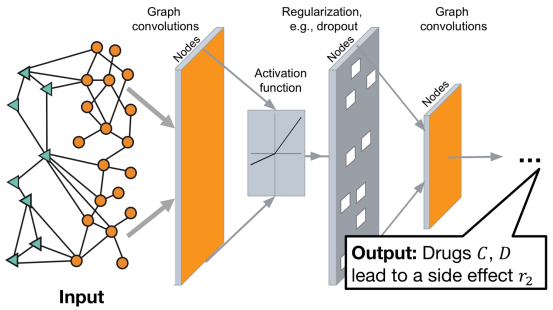
Movies

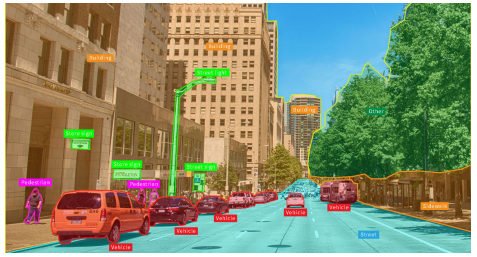
Social Networks



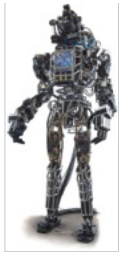
Motivation for dataflow backends that support sparsity

Machine Learning

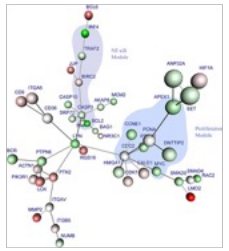




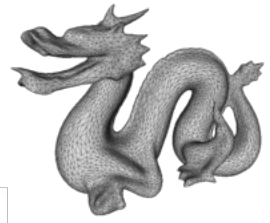
Scientific Computing



Robotics



Computational Biology

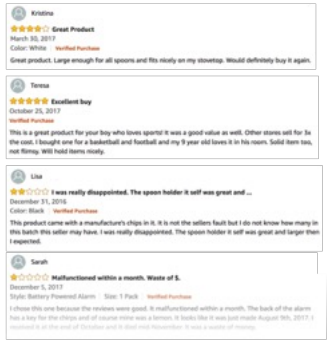


Simulations

Data Analytics

Movies

Social Networks



NVIDIA

arm

Benefits of a compiler that targets dataflow accelerators

Benefits of a compiler that targets dataflow accelerators

Long Tail of Expressions (algorithm)

Benefits of a compiler that targets dataflow accelerators

Long Tail of Expressions (algorithm) × Varying Compression Structures (format)

Benefits of a compiler that targets dataflow accelerators

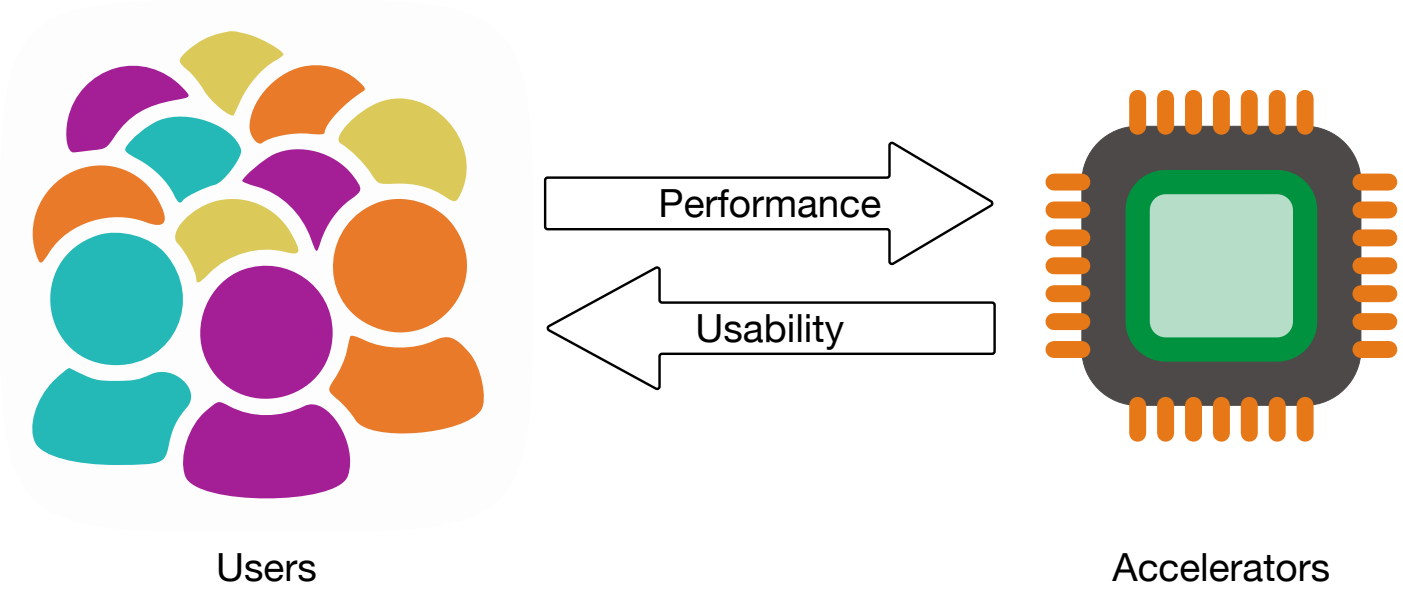
Long Tail of Expressions (algorithm) × Varying Compression Structures (format) × More Performant Backends (platform)

Benefits of a compiler that targets dataflow accelerators

Long Tail of Expressions (algorithm) × Varying Compression Structures (format) × More Performant Backends (platform) × Backend-Specific Transformations (schedules)

Benefits of a compiler that targets dataflow accelerators

Long Tail of Expressions (algorithm) × Varying Compression Structures (format) × More Performant Backends (platform) × Backend-Specific Transformations (schedules)



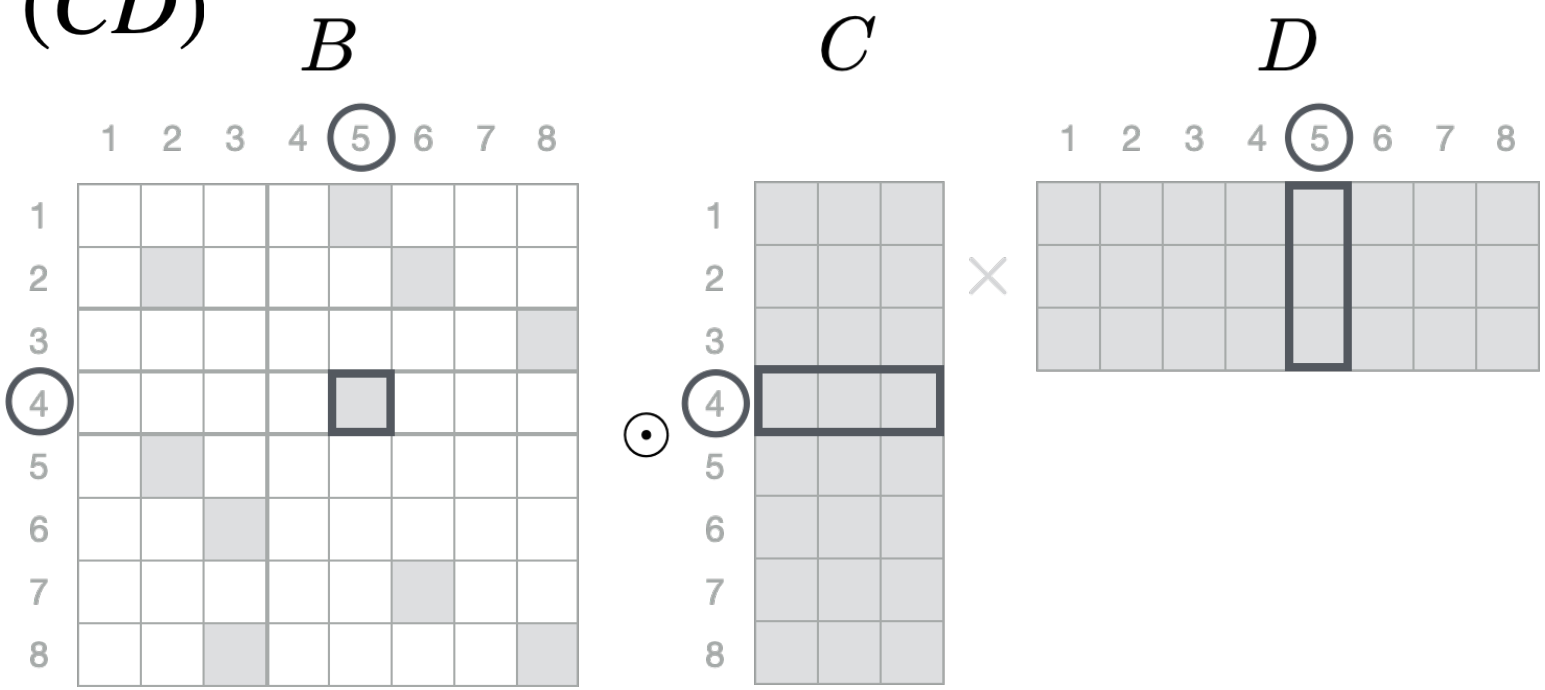
Fusion over factorization in sparse applications

Fusion over factorization in sparse applications

$$A = B \odot (CD)$$

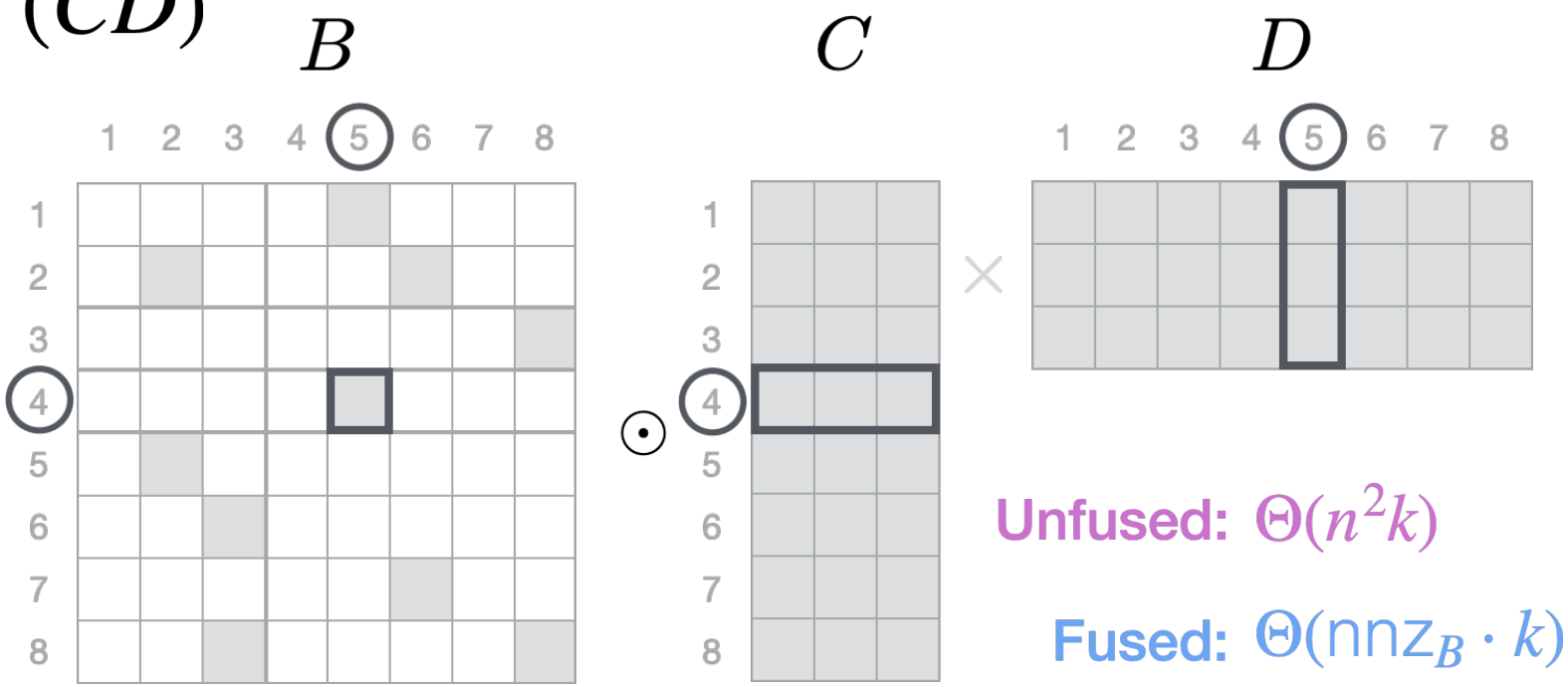
Fusion over factorization in sparse applications

$$A = B \odot (CD)$$



Fusion over factorization in sparse applications

$$A = B \odot (CD)$$



Unfused: $\Theta(n^2k)$

Fused: $\Theta(\text{nnz}_B \cdot k)$

Goal of the Sparse Abstract Machine (SAM)

Goal of the Sparse Abstract Machine (SAM)

Create a ***logical abstraction*** that can represent ***sparse*** applications for ***dataflow*** hardware

Goal of the Sparse Abstract Machine (SAM)

Create a ***logical abstraction*** that can represent ***sparse*** applications for ***dataflow*** hardware

Also an ***intermediate representation (IR)*** for automatic compilation

Goal of the Sparse Abstract Machine (SAM)

Create a ***logical abstraction*** that can represent ***sparse*** applications for ***dataflow*** hardware

Also an ***intermediate representation (IR)*** **for automatic compilation**

Describe the ***entire*** space of sparse tensor algebra expressions

Goal of the Sparse Abstract Machine (SAM)

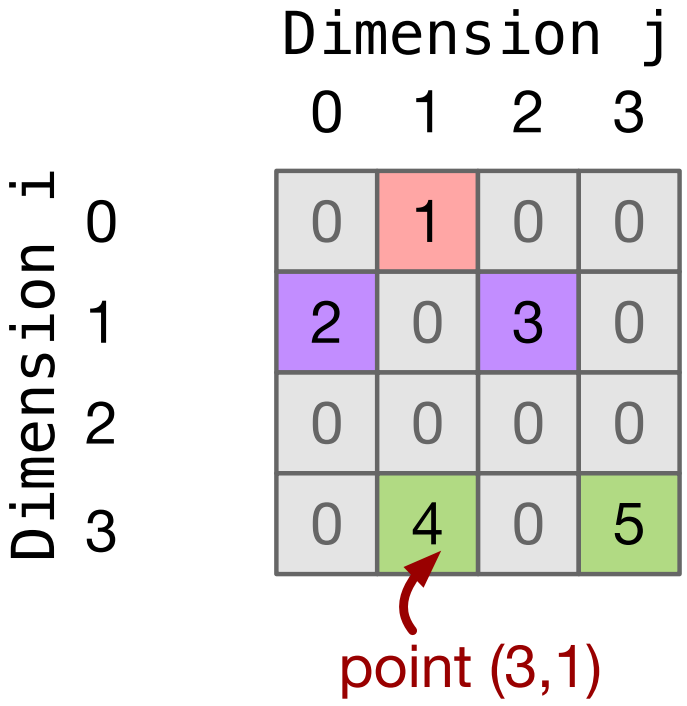
Create a ***logical abstraction*** that can represent ***sparse*** applications for ***dataflow*** hardware

Also an ***intermediate representation (IR)*** **for automatic compilation**

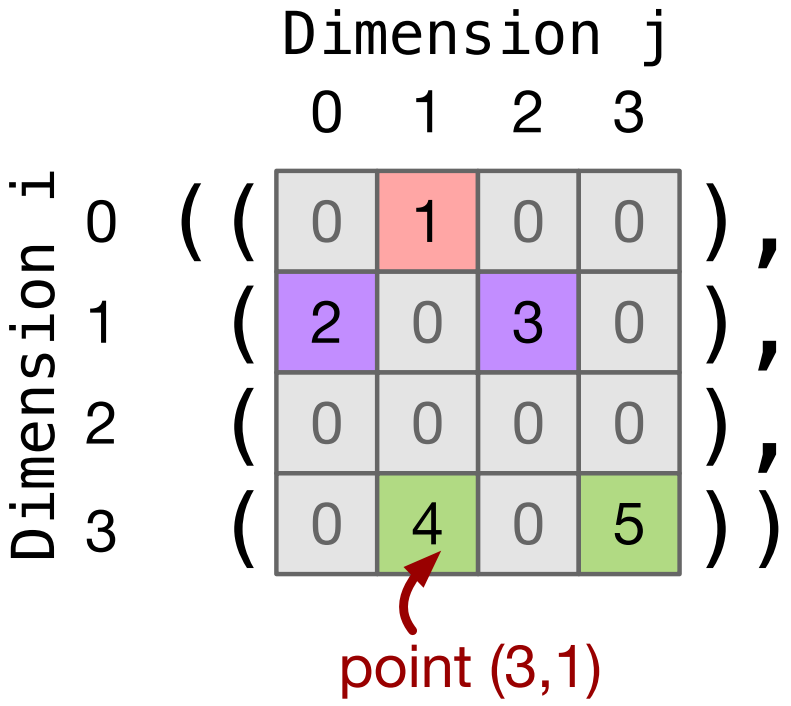
Describe the ***entire*** space of sparse tensor algebra expressions

Develop and generate ***efficient hardware implementations,*** namely the Onyx CGRA

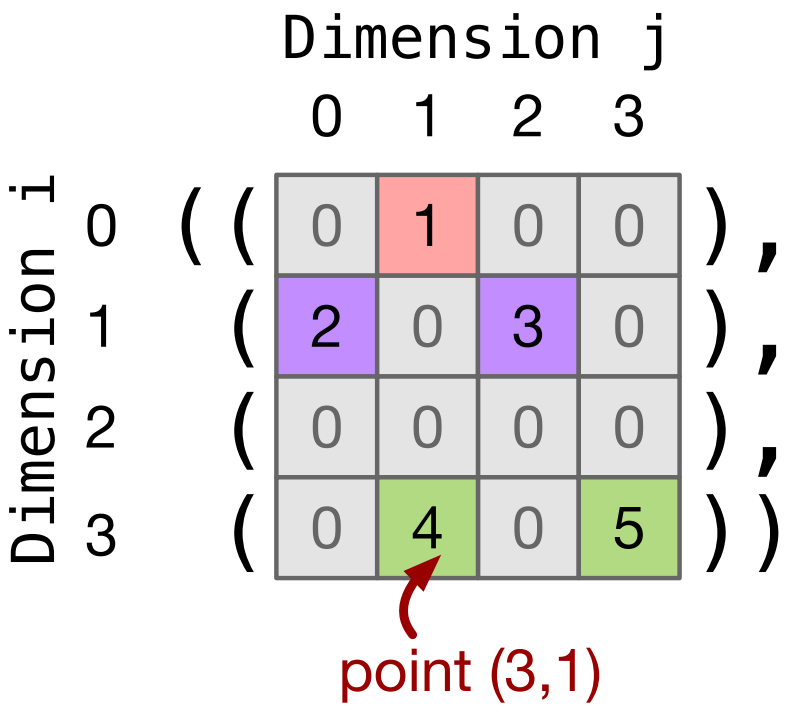
Abstract tensor data model (fiber trees)

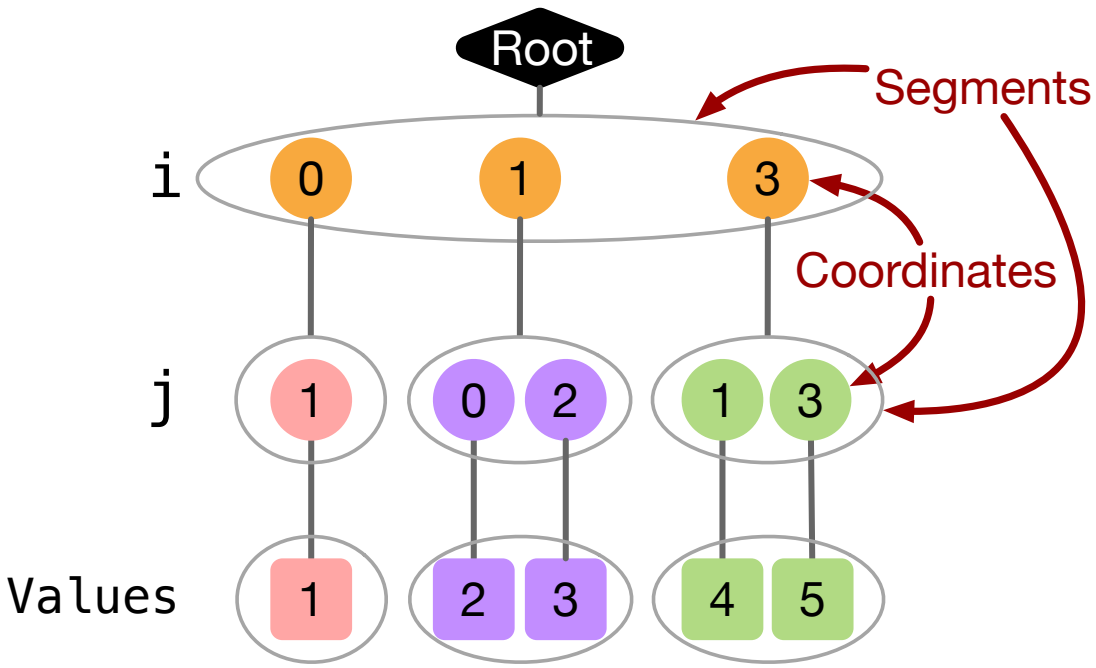


Abstract tensor data model (fiber trees)

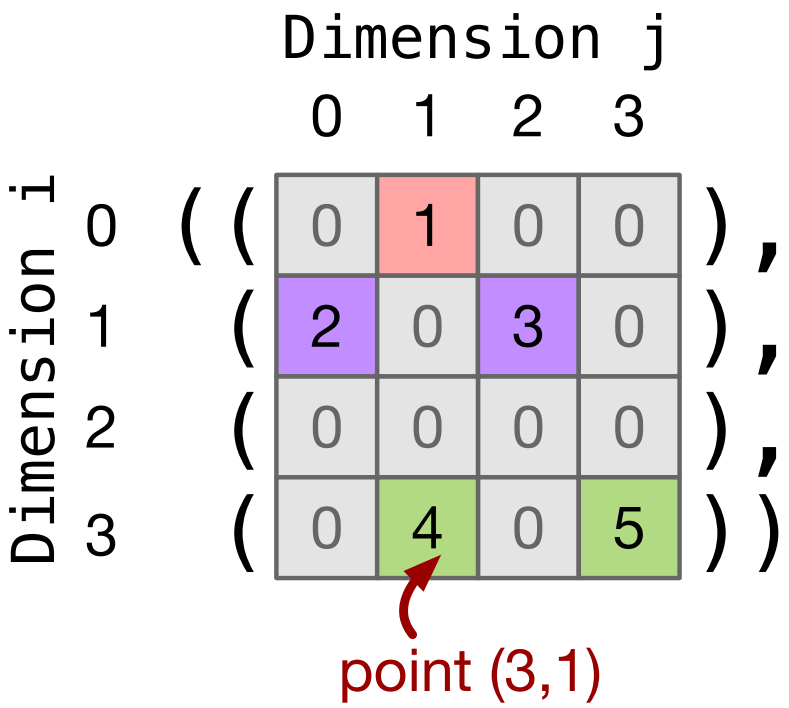


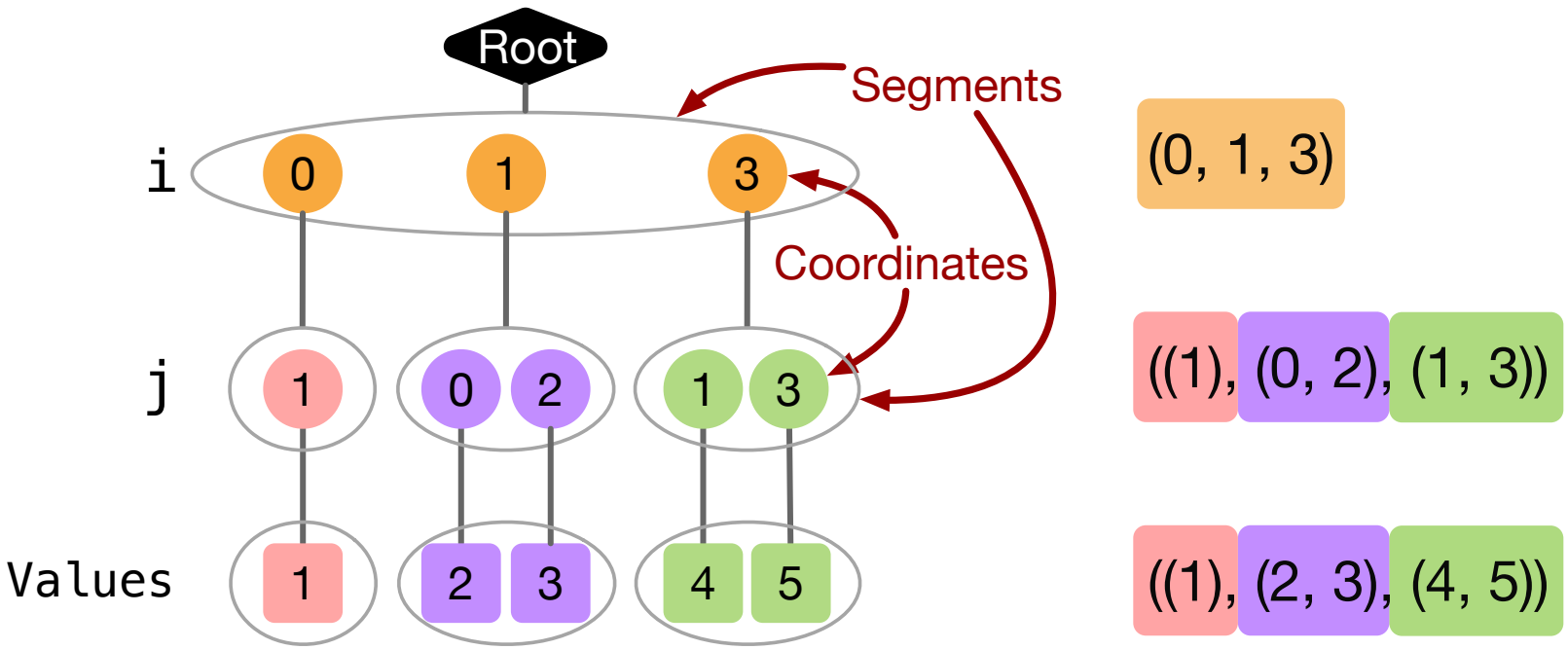
Abstract tensor data model (fiber trees)





Abstract tensor data model (fiber trees)





Tensors in SAM

(0, 1, 3)

((1), (0, 2), (1, 3))

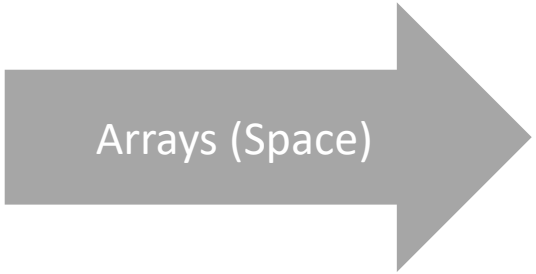
((1), (2, 3), (4, 5))

Tensors in SAM

(0, 1, 3)

((1), (0, 2), (1, 3))

((1), (2, 3), (4, 5))



Tensors in SAM

(0, 1, 3)

((1), (0, 2), (1, 3))

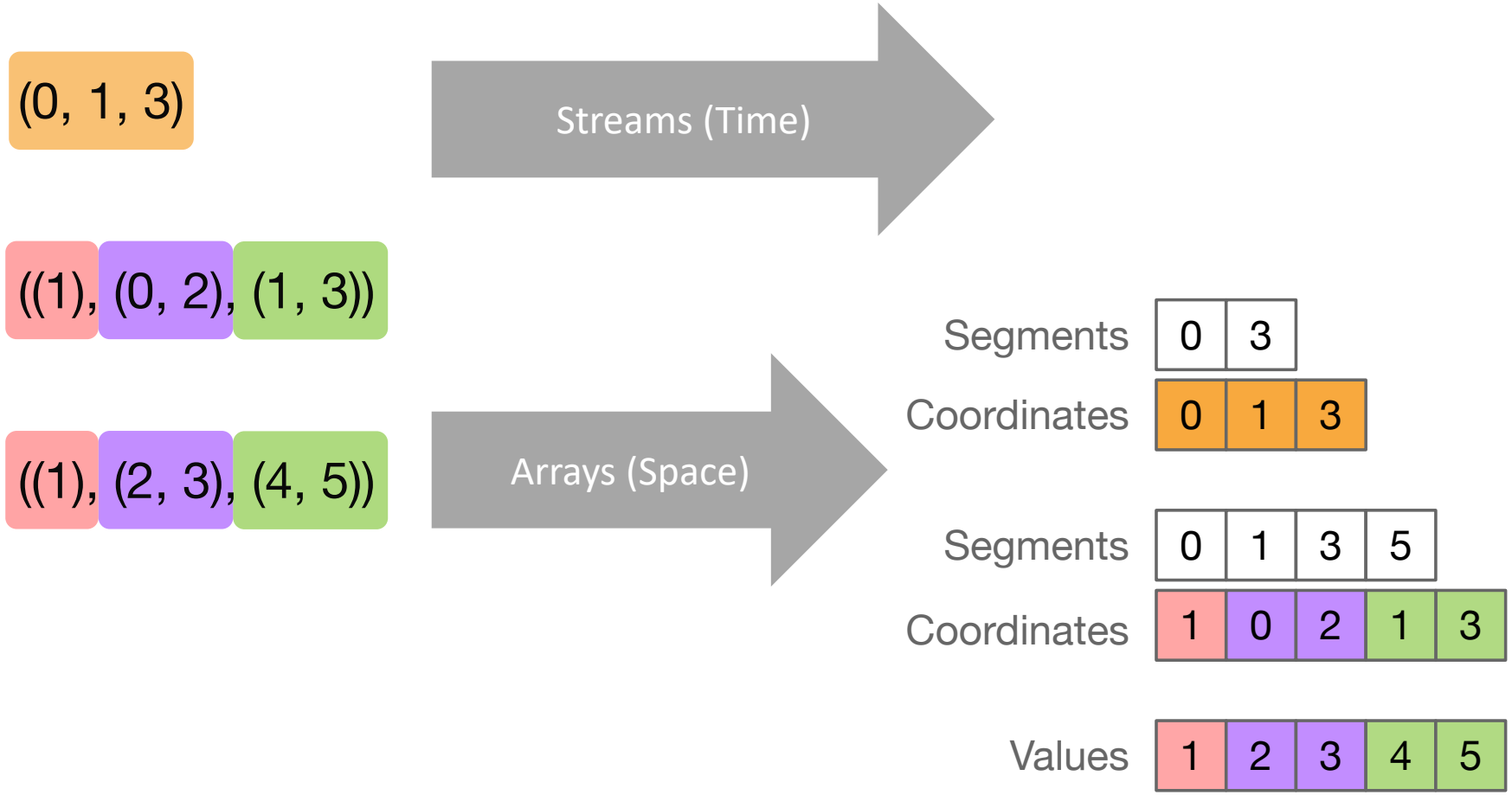
((1), (2, 3), (4, 5))

Arrays (Space)

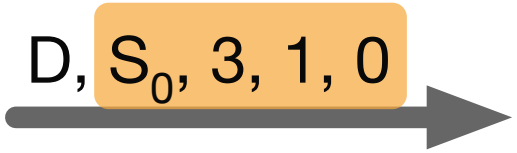
Segments	0	3	
Coordinates	0	1	3

Segments	0	1	3	5
Coordinates	1	0	2	1	3
Values	1	2	3	4	5

Tensors in SAM



Tensors in SAM



(0, 1, 3)

Streams (Time)

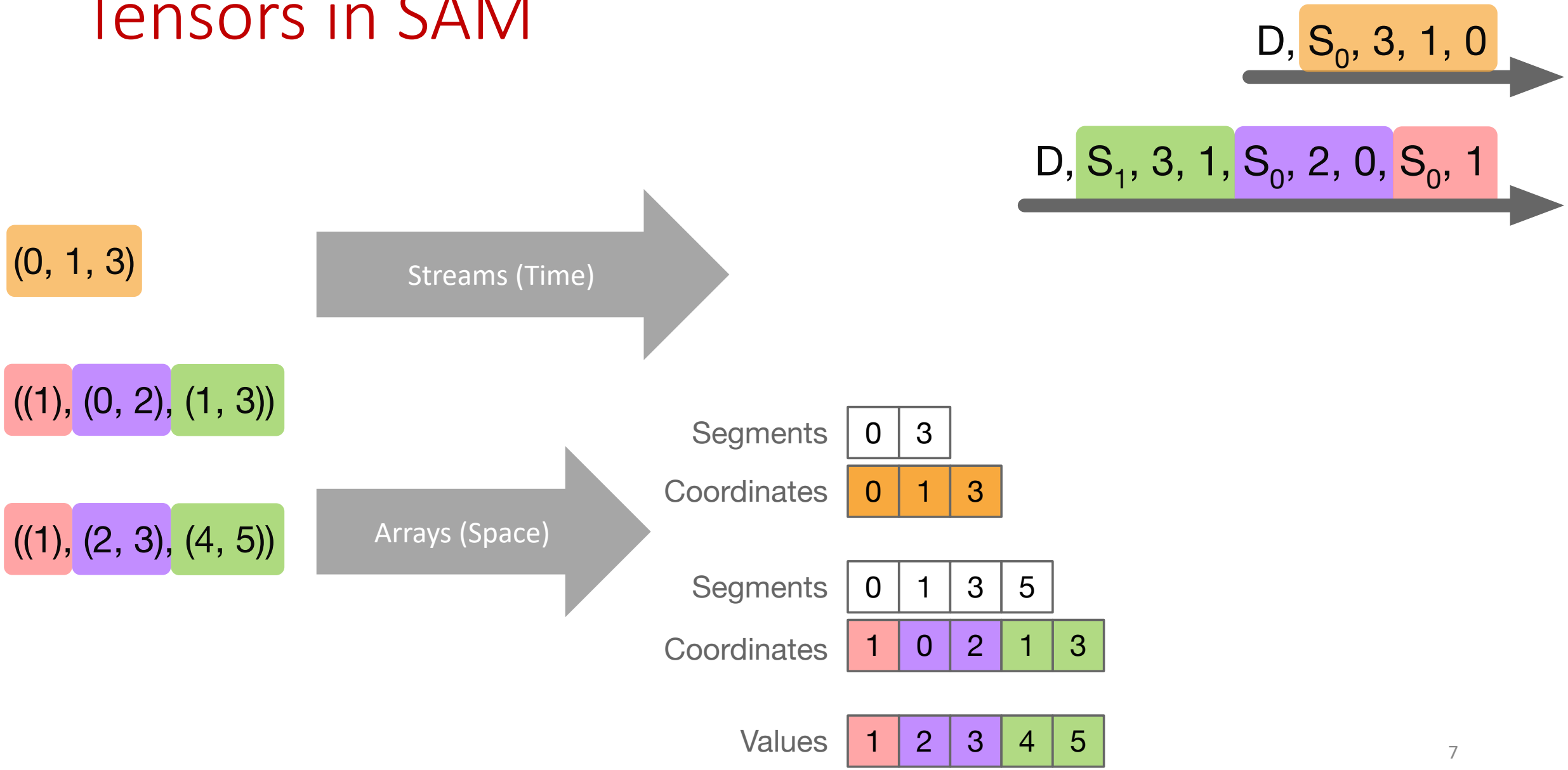
((1), (0, 2), (1, 3))

((1), (2, 3), (4, 5))

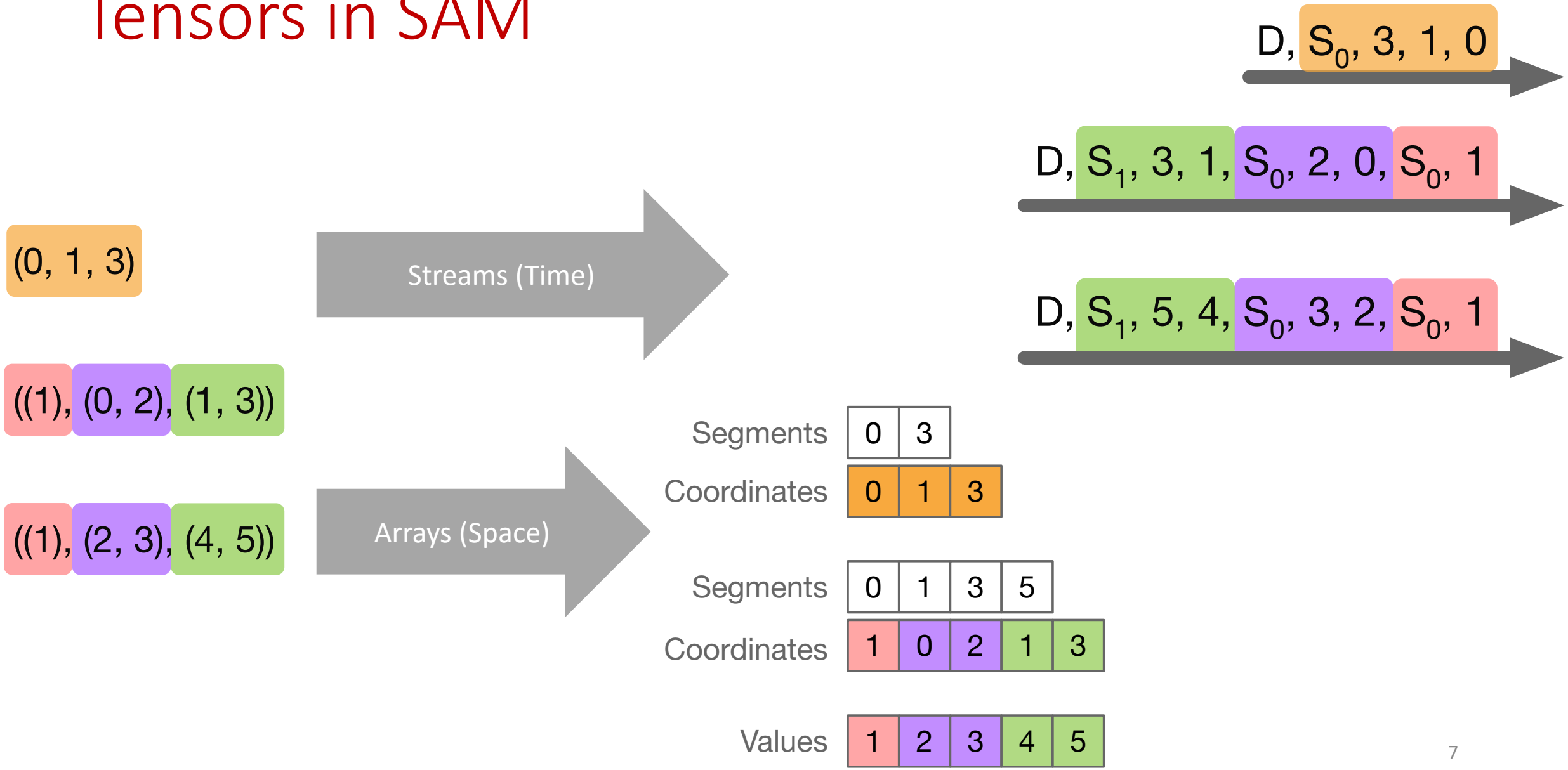
Arrays (Space)

Segments	0	3			
Coordinates	0	1	3		
Segments	0	1	3	5	
Coordinates	1	0	2	1	3
Values	1	2	3	4	5

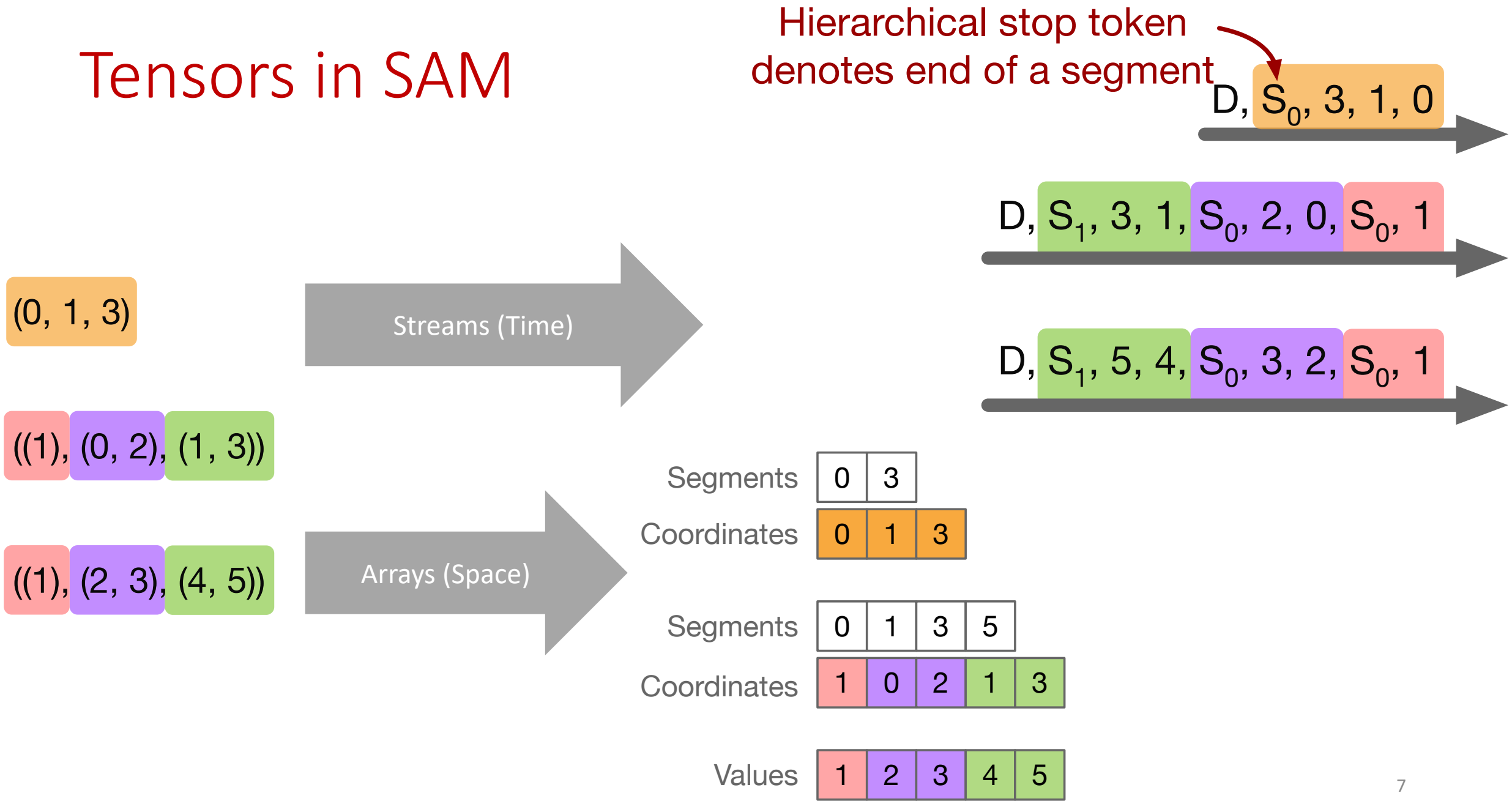
Tensors in SAM



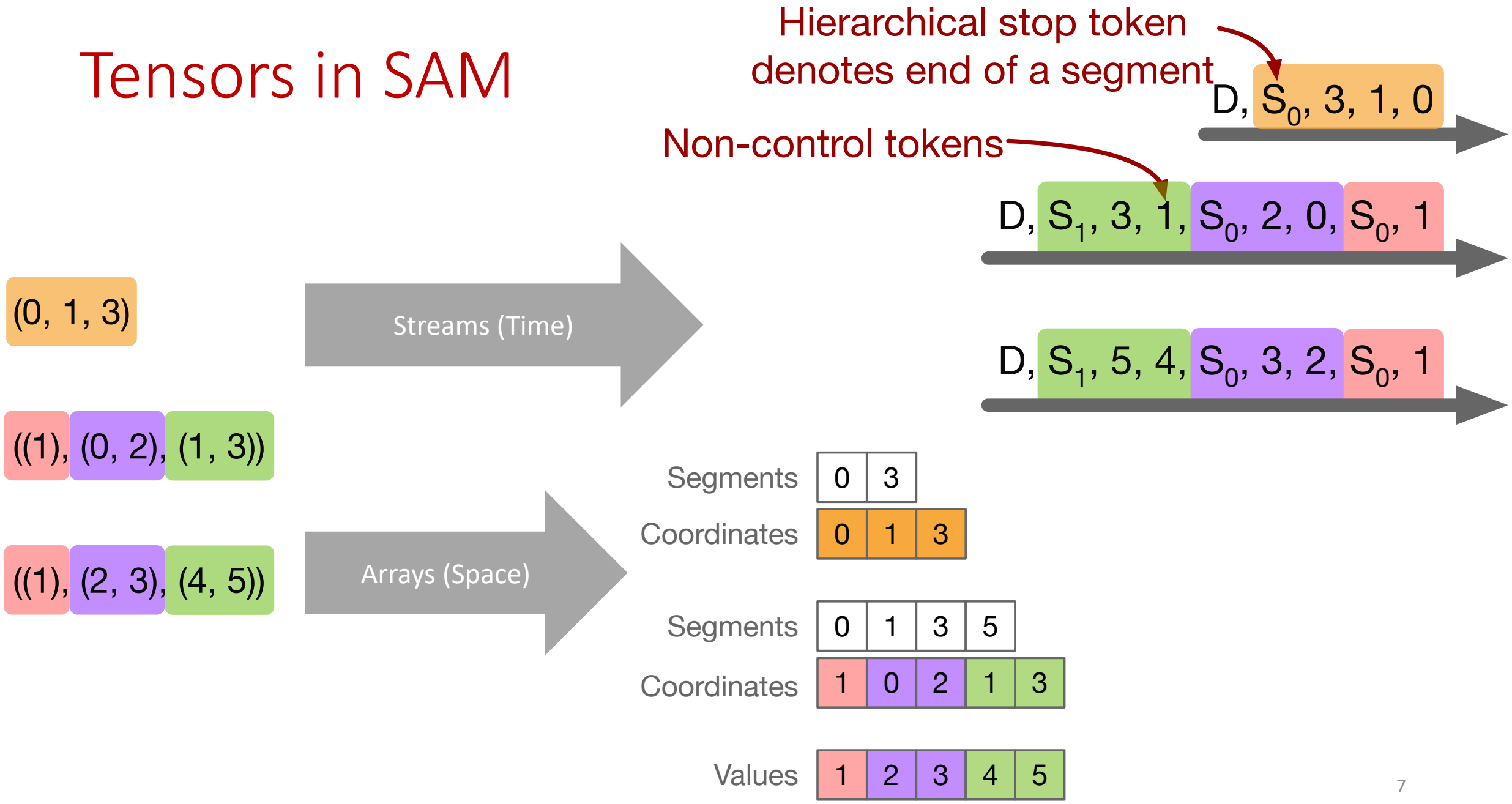
Tensors in SAM



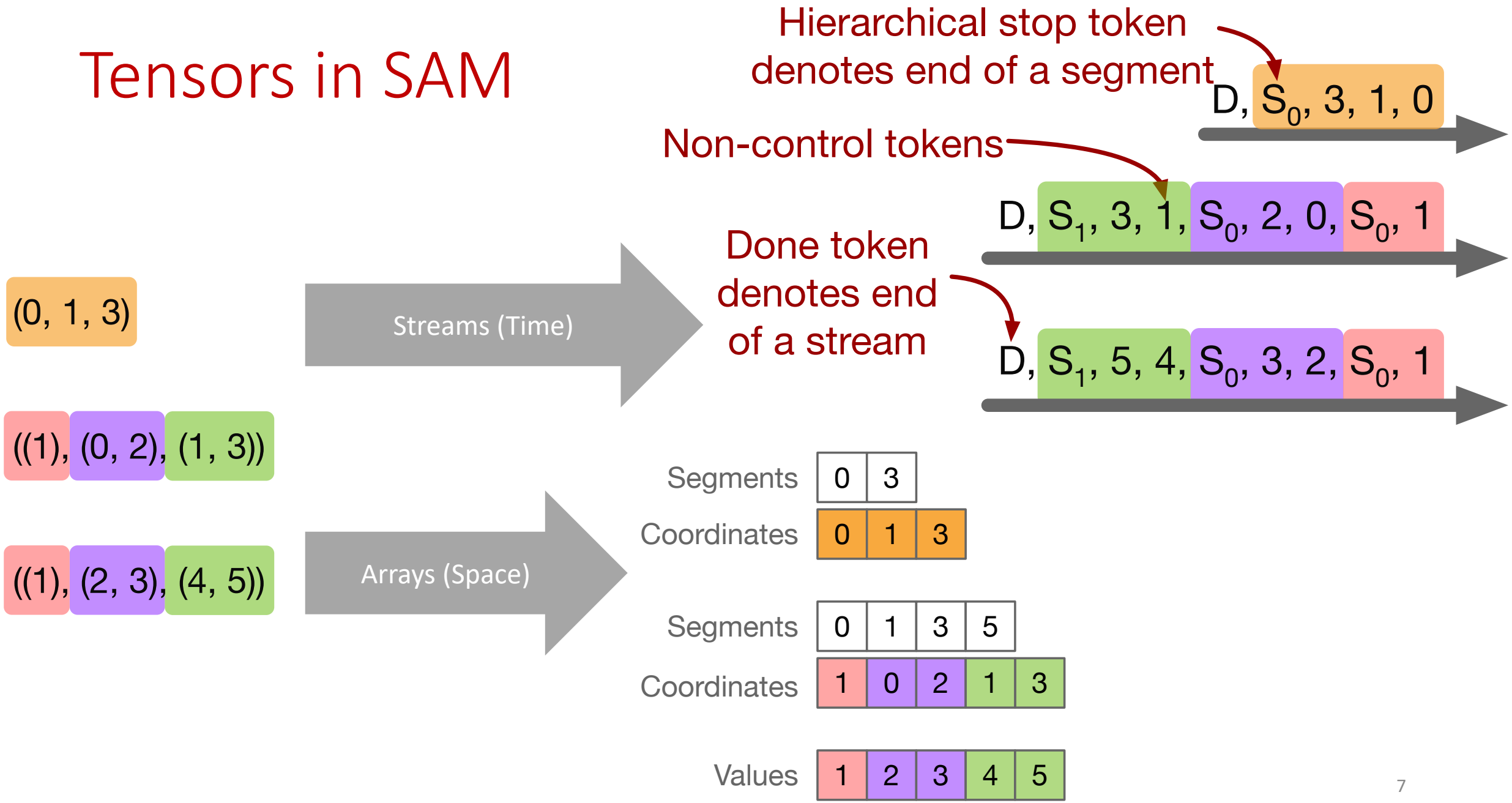
Tensors in SAM



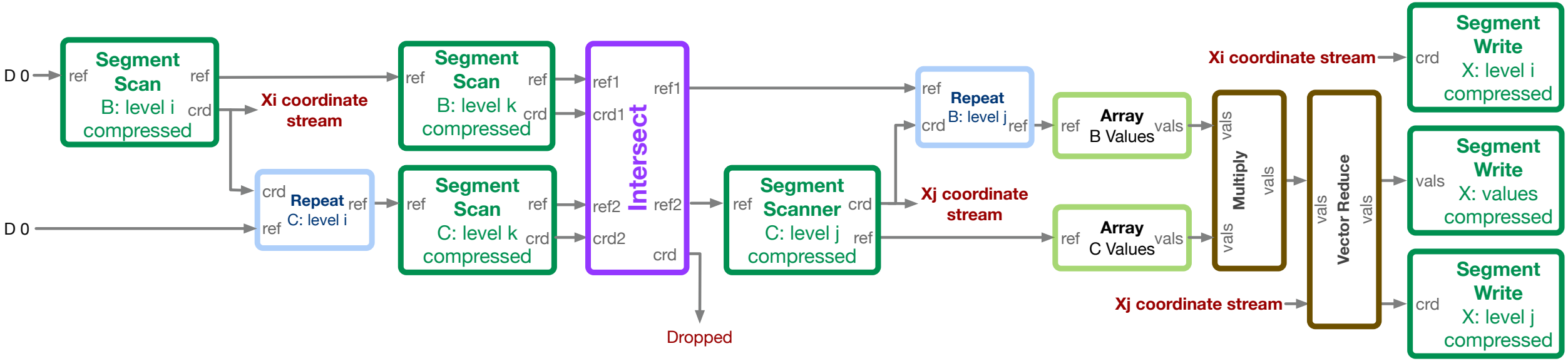
Tensors in SAM



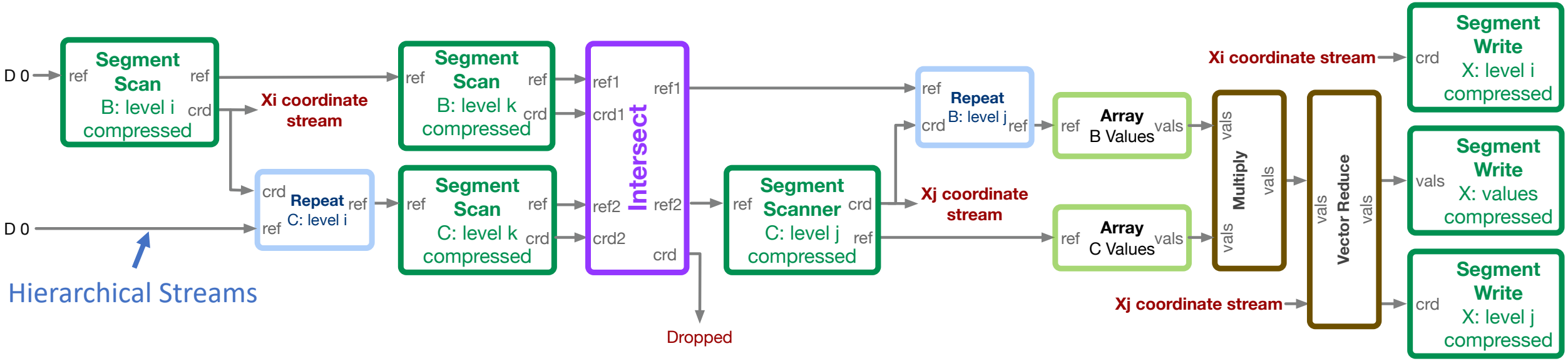
Tensors in SAM



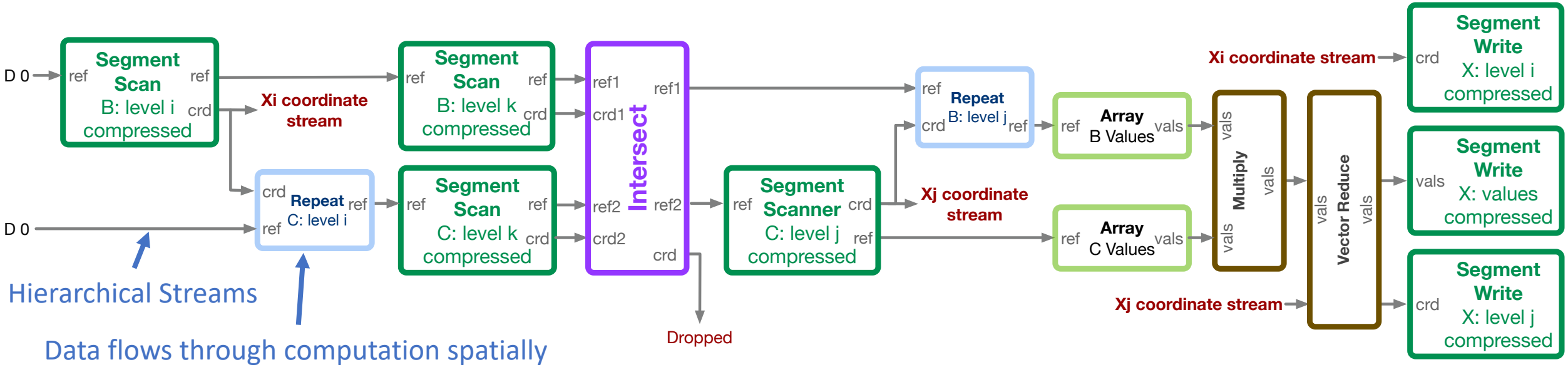
Sparse-matrix sparse-matrix multiplication in SAM



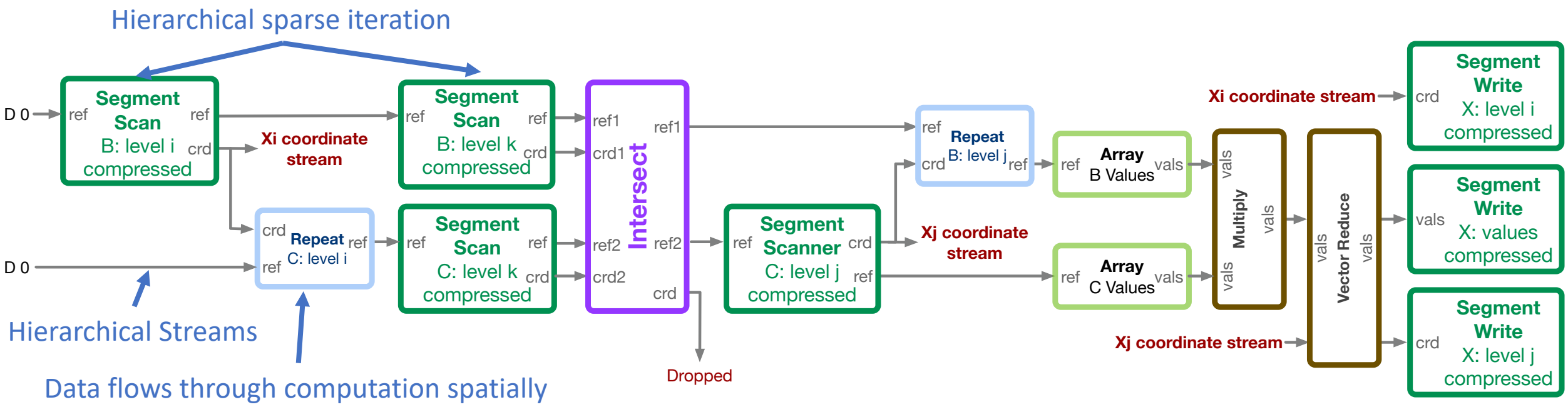
Sparse-matrix sparse-matrix multiplication in SAM



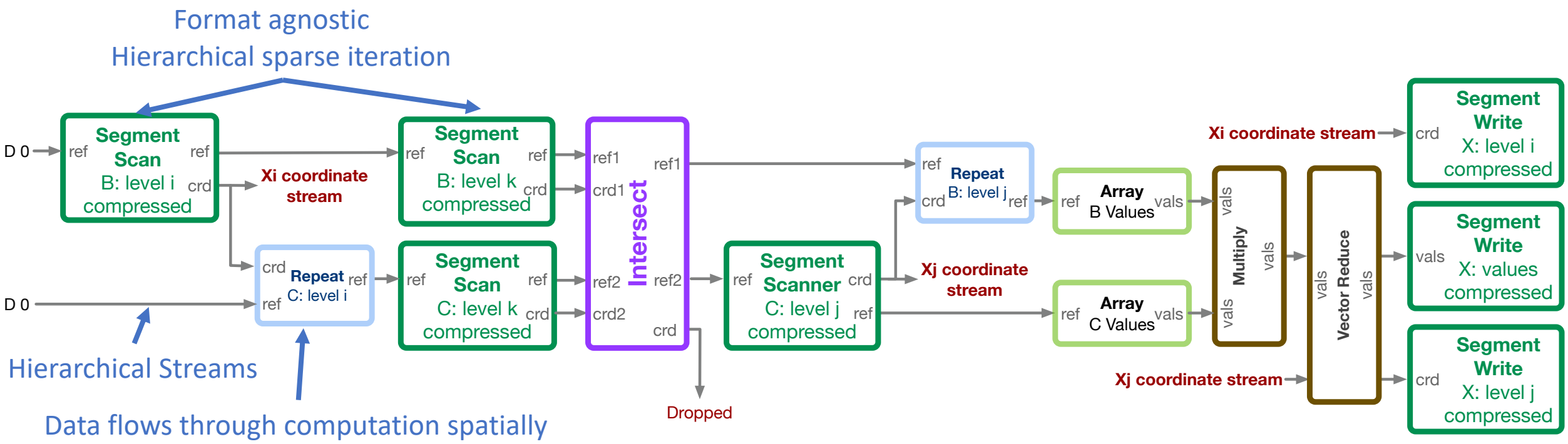
Sparse-matrix sparse-matrix multiplication in SAM



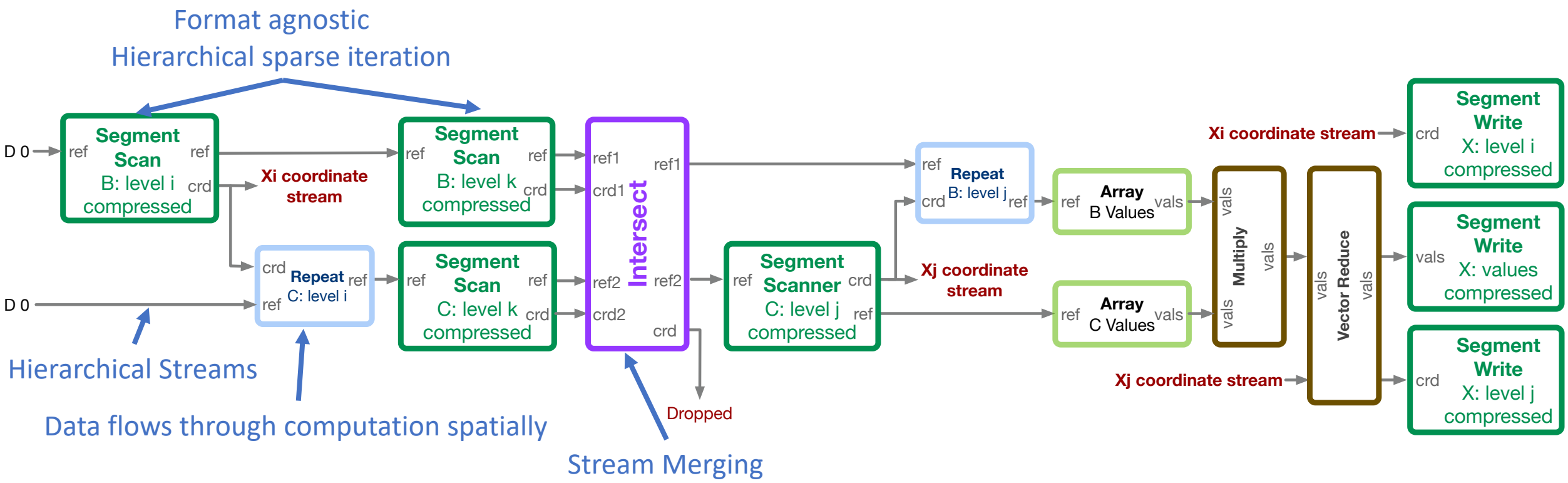
Sparse-matrix sparse-matrix multiplication in SAM



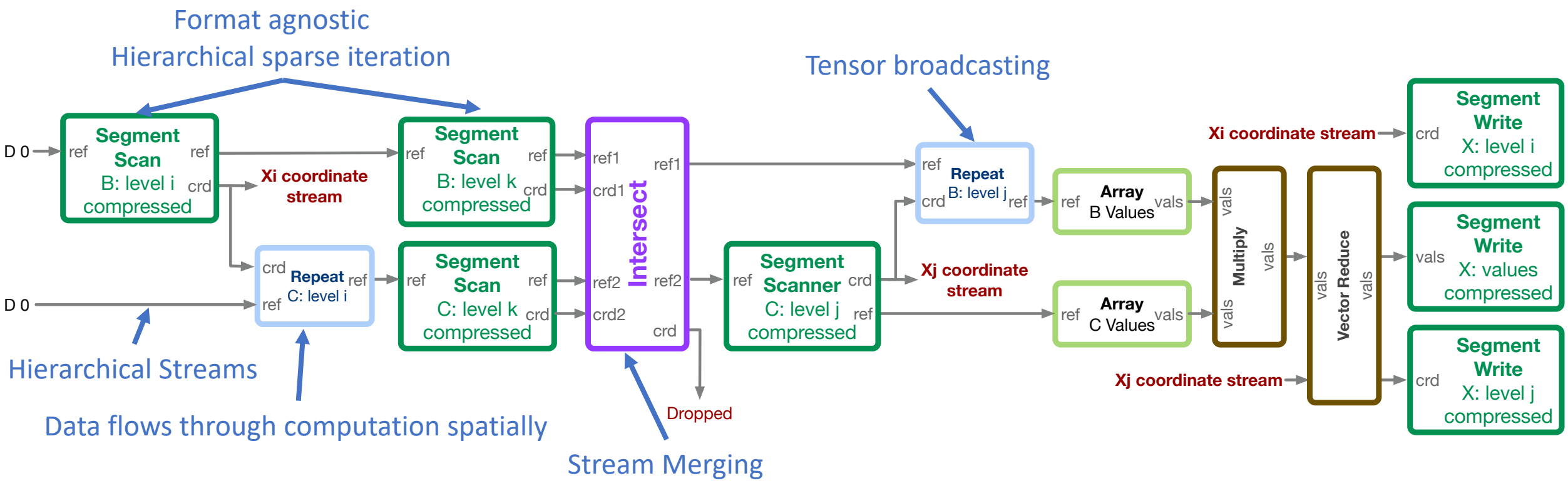
Sparse-matrix sparse-matrix multiplication in SAM



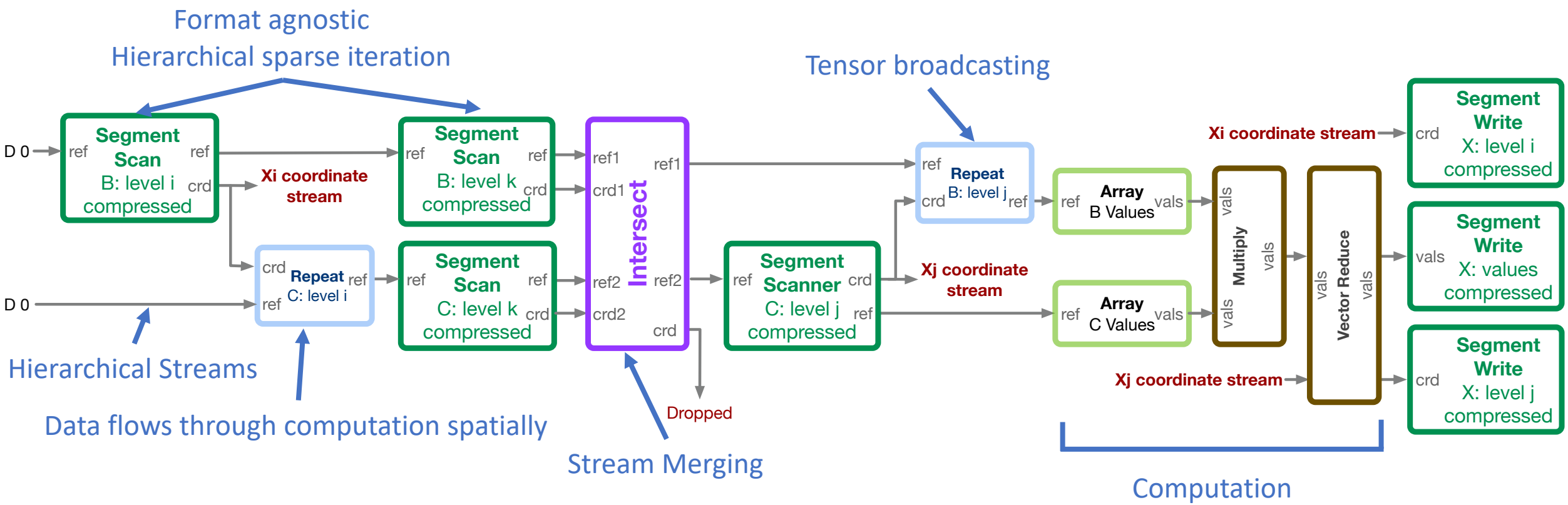
Sparse-matrix sparse-matrix multiplication in SAM



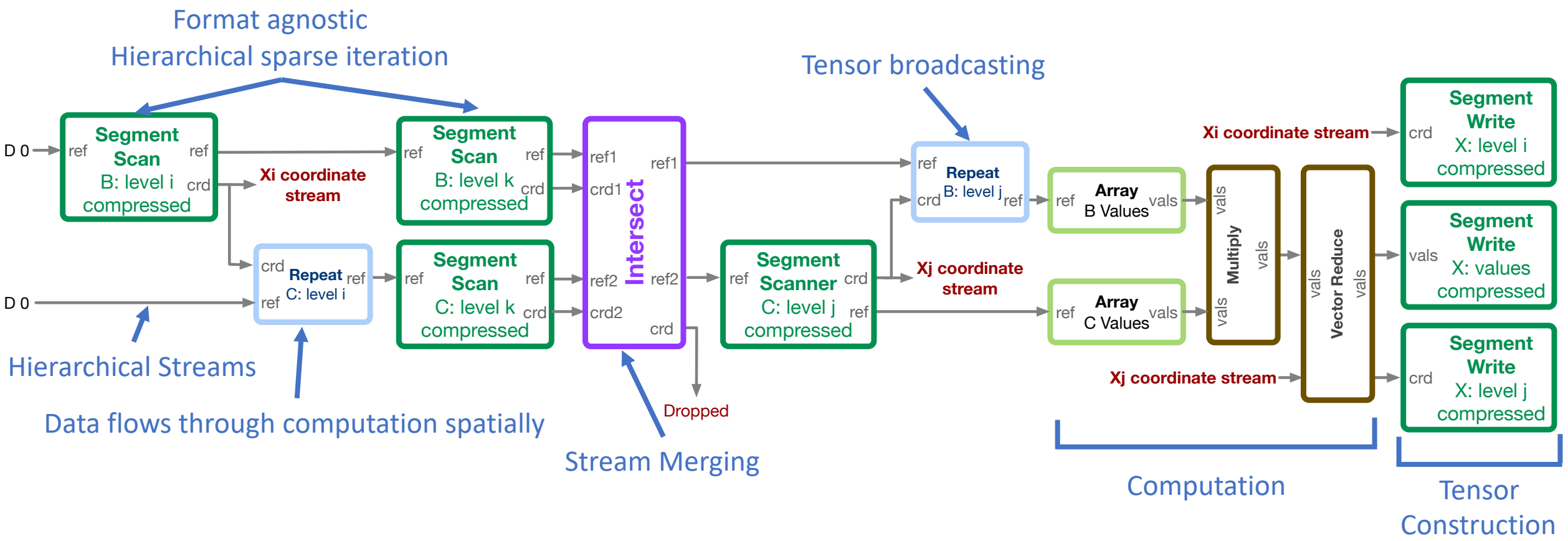
Sparse-matrix sparse-matrix multiplication in SAM



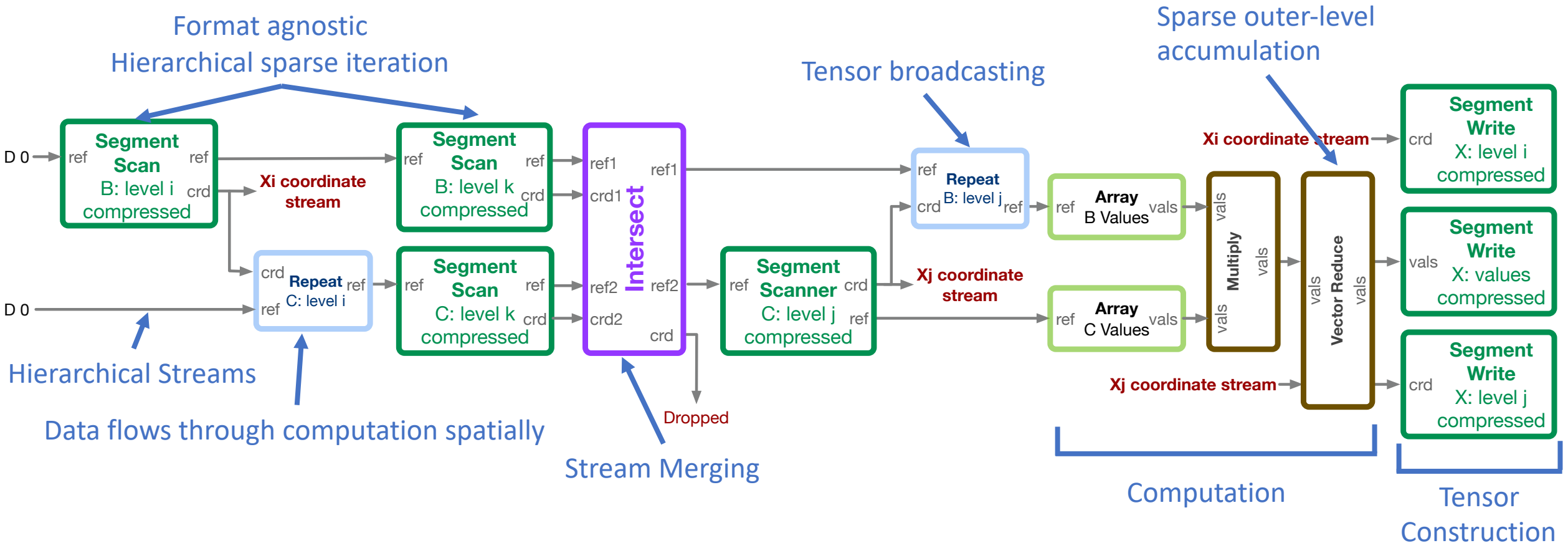
Sparse-matrix sparse-matrix multiplication in SAM



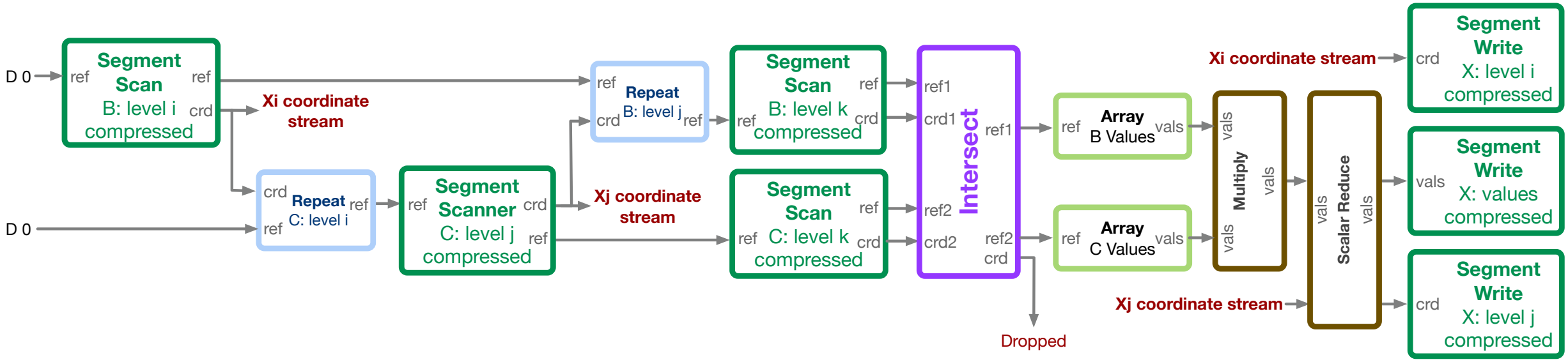
Sparse-matrix sparse-matrix multiplication in SAM



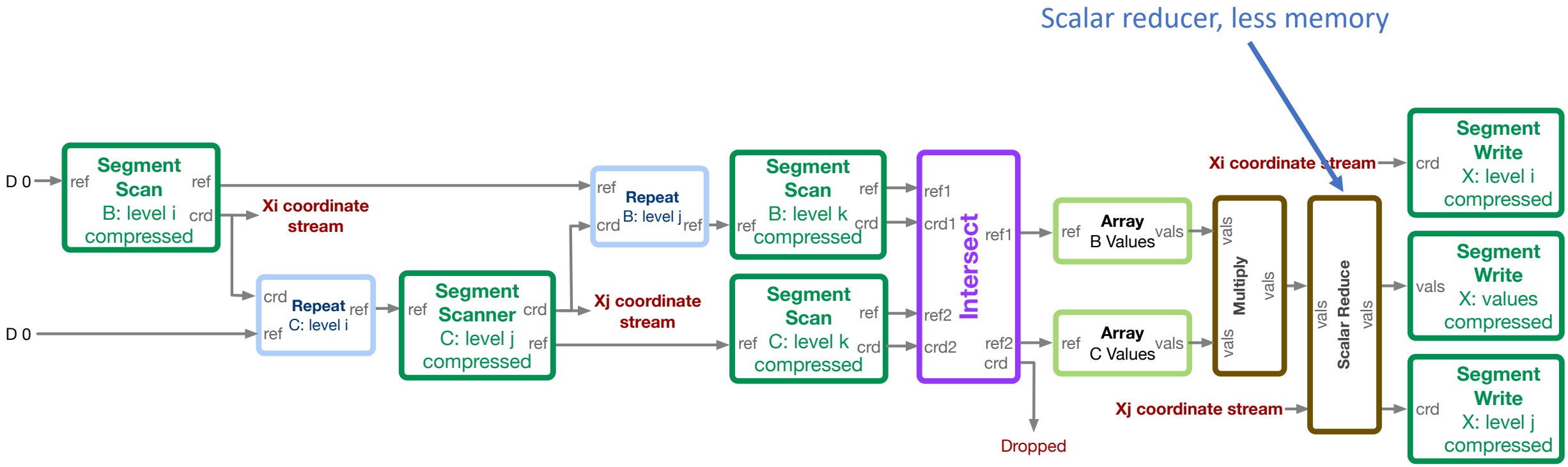
Sparse-matrix sparse-matrix multiplication in SAM



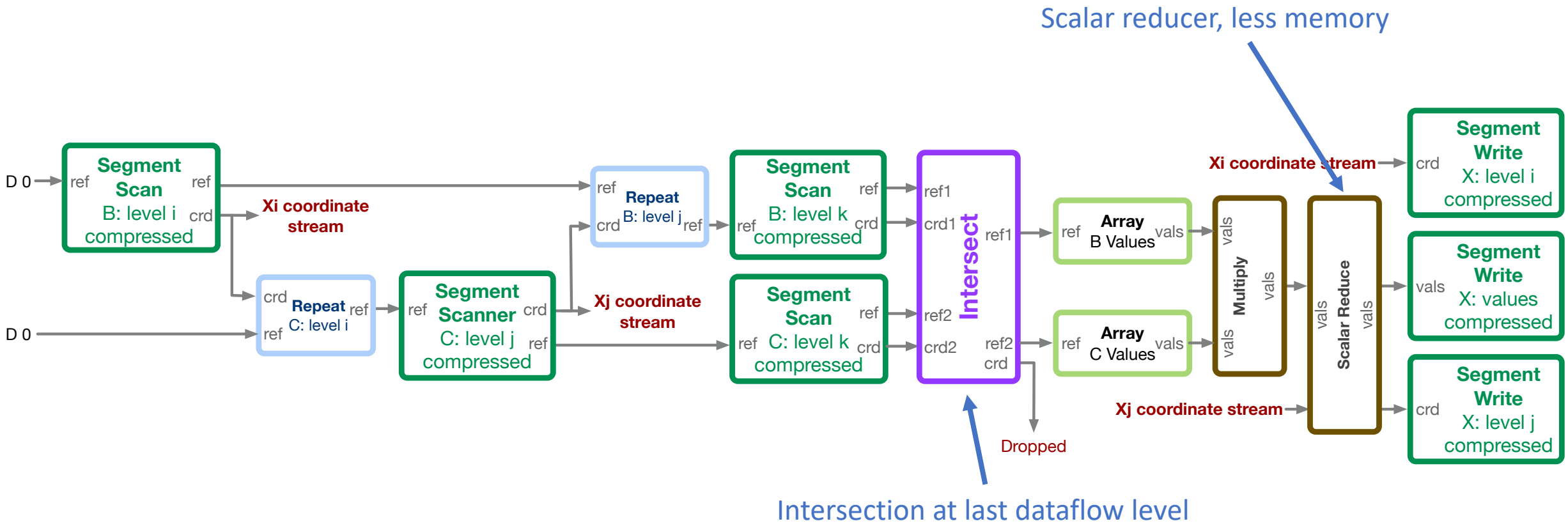
Inner-product algorithm SAM graph



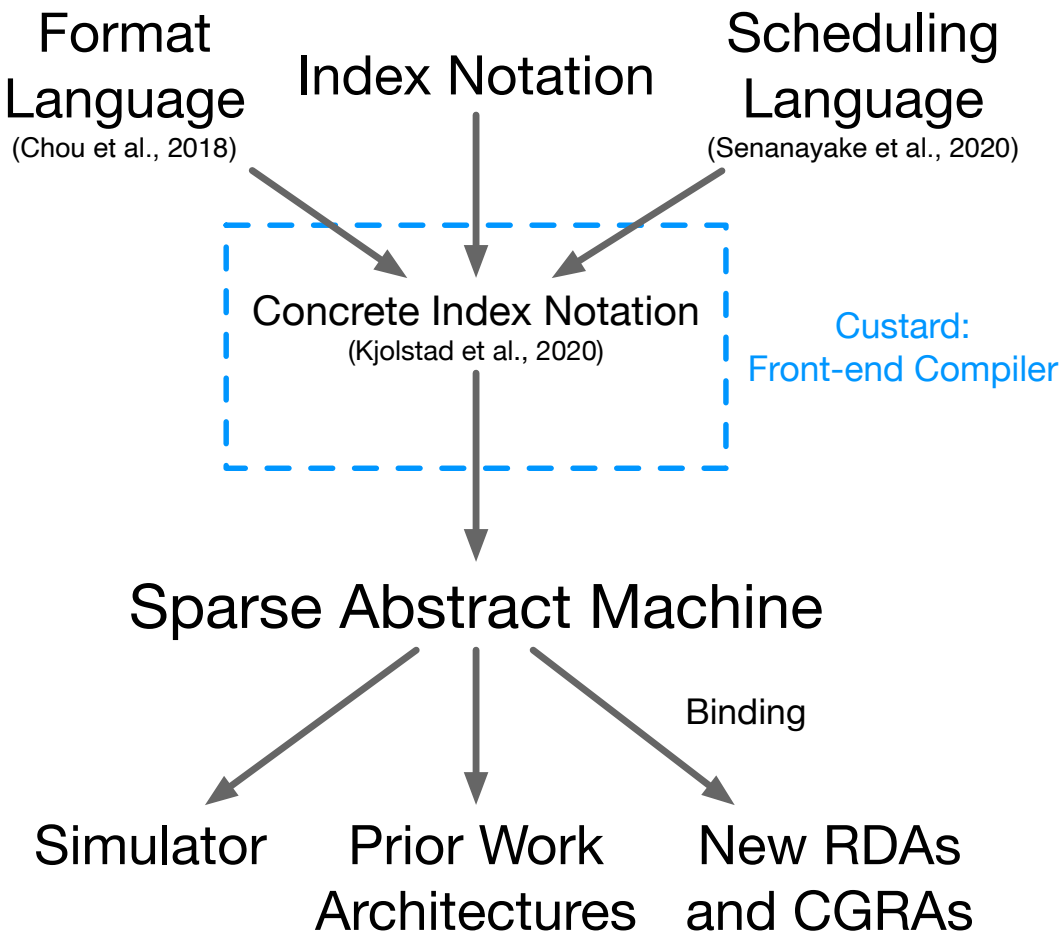
Inner-product algorithm SAM graph



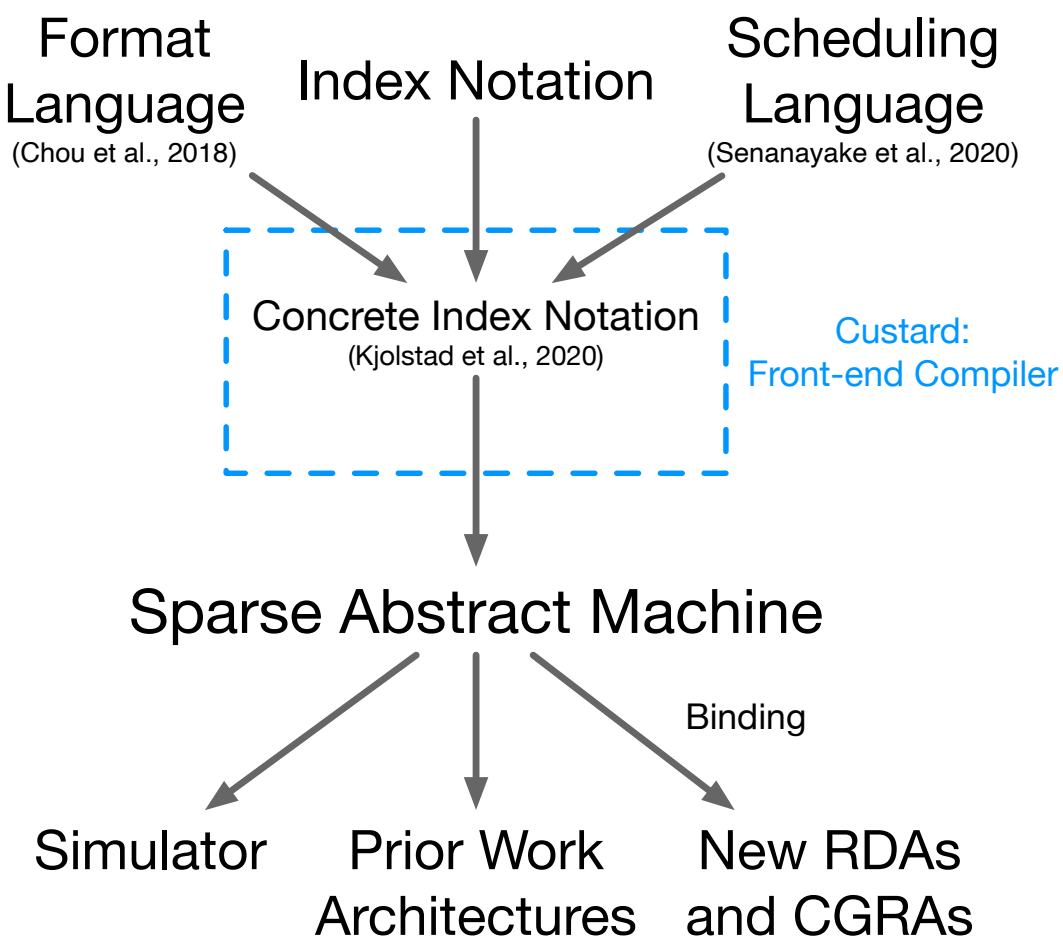
Inner-product algorithm SAM graph

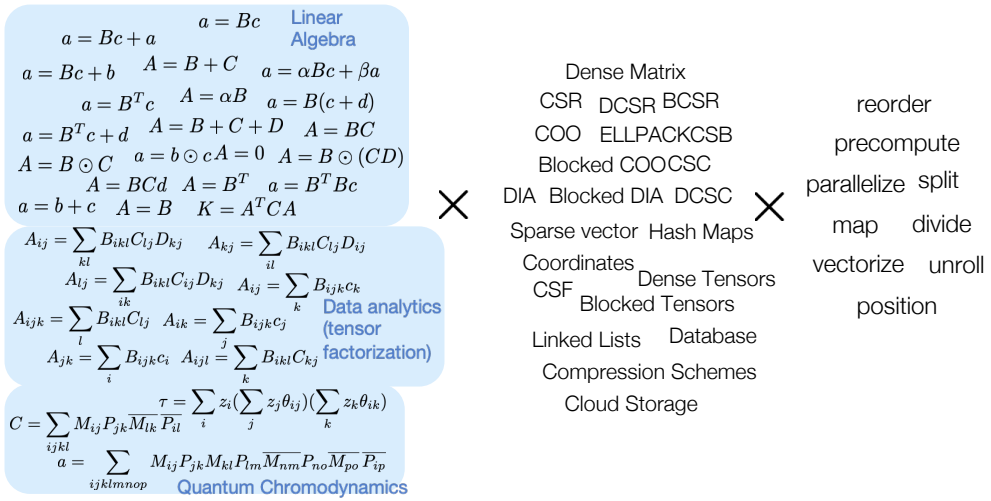


System overview

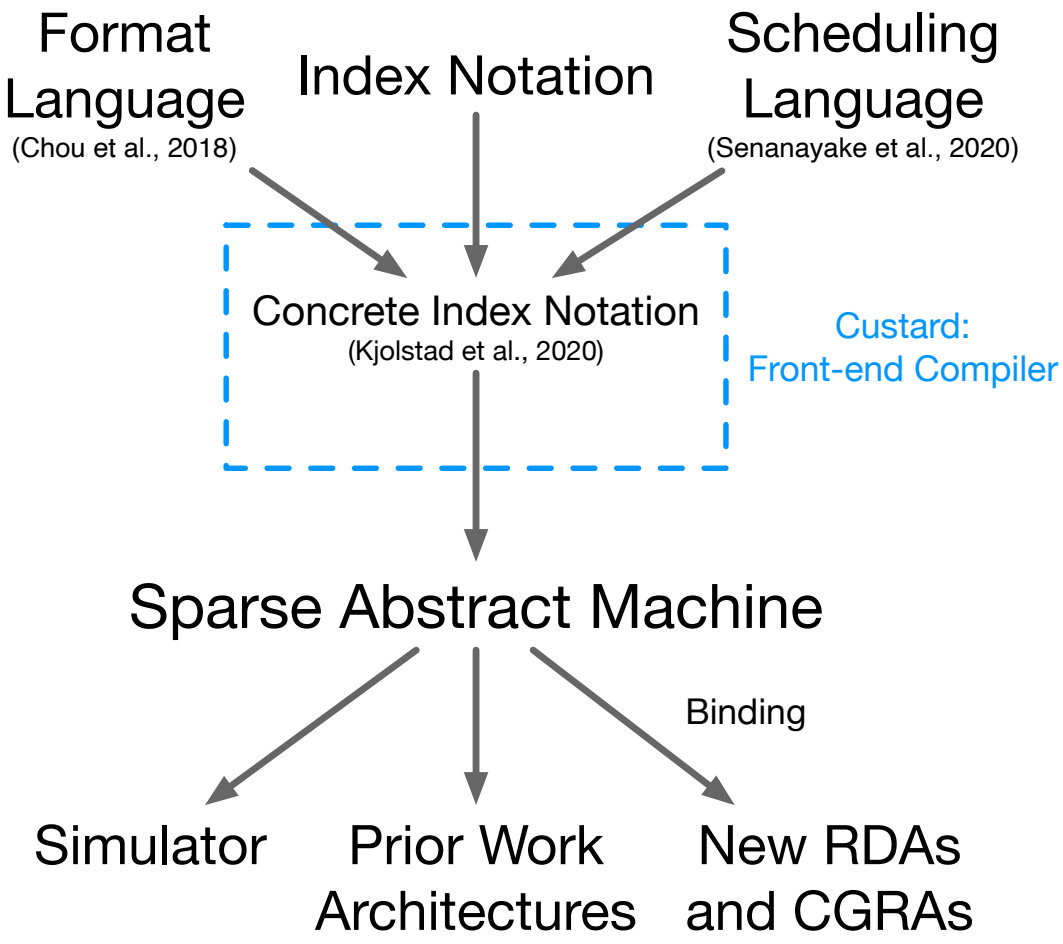


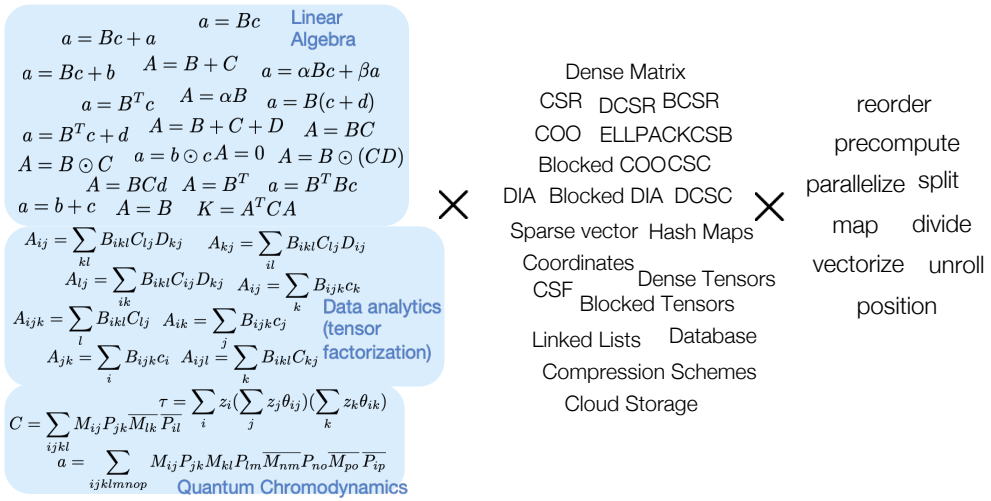
System overview

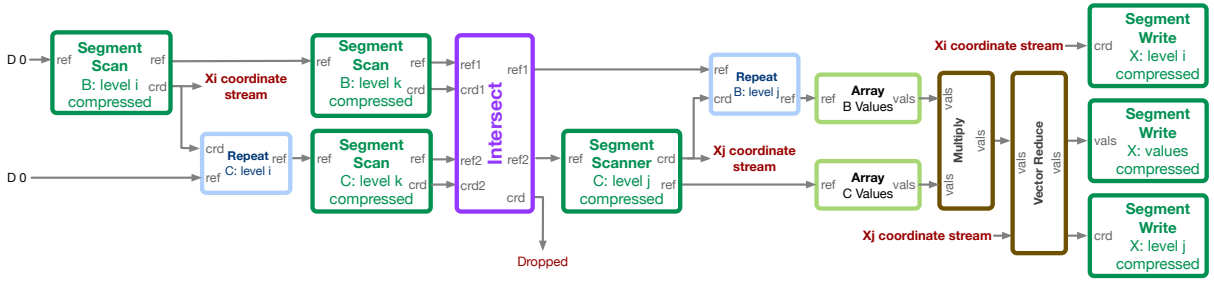




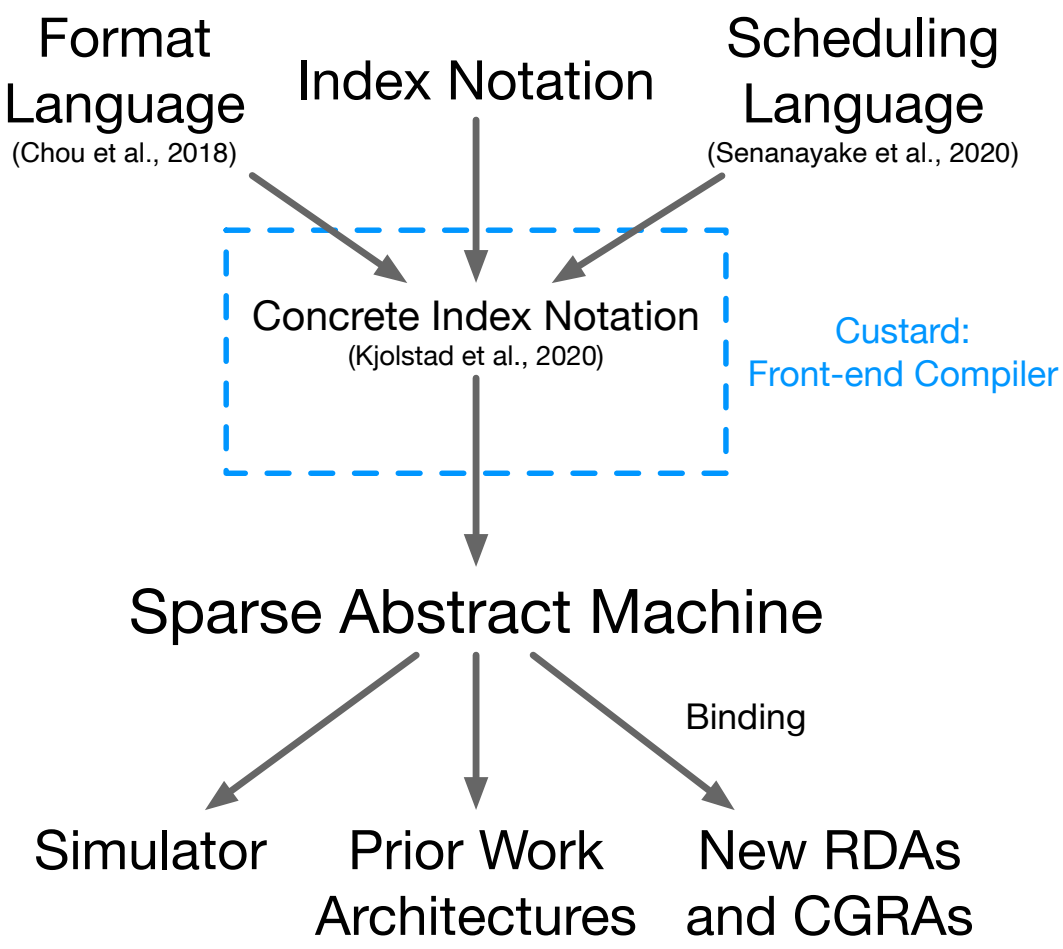
System overview

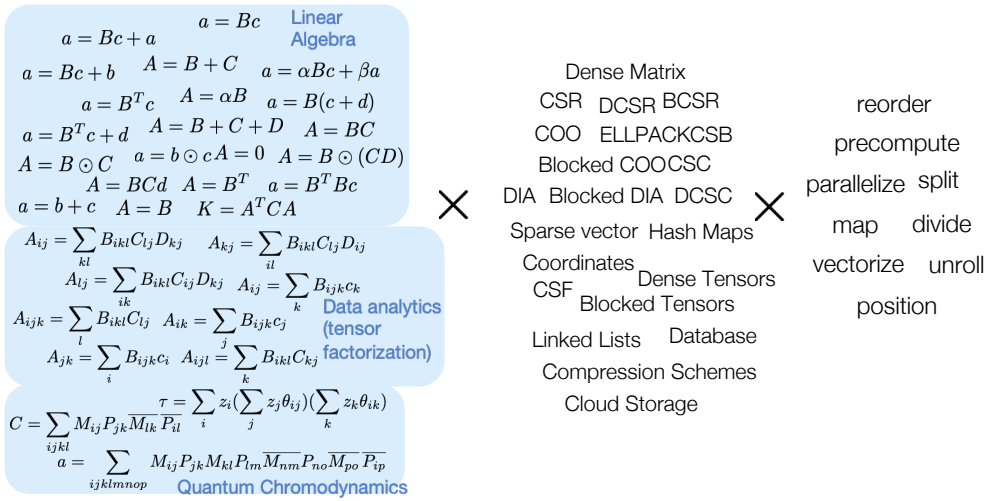


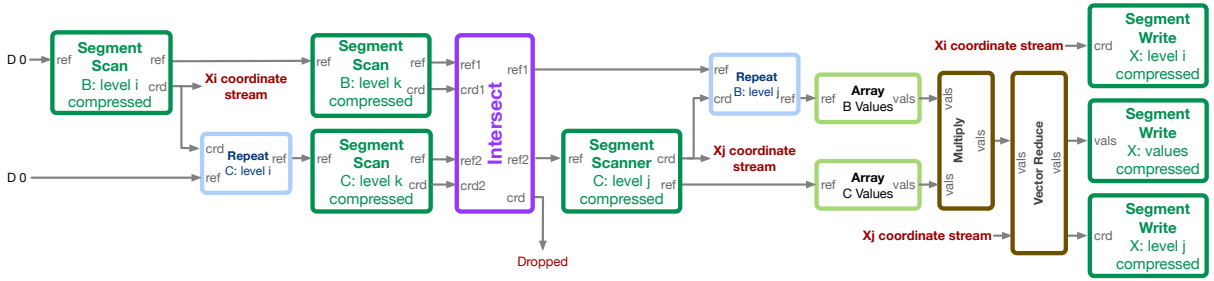




System overview







Compiler flow

Expression

$$X_{ij} = B_{ik} * C_{kj}$$

Format Language

```
B=({sparse,sparse},
   {mode0,mode1})
C=({sparse, sparse},
   {mode1, mode0})
```

Schedule

```
reorder(i, k, j)
```

Compiler flow

Expression

$$X_{ij} = B_{ik} * C_{kj}$$

Format Language

```
B=({sparse,sparse},
   {mode0,mode1})
C=({sparse, sparse},
   {mode1, mode0})
```

Schedule

```
reorder(i, k, j)
```

$$\forall_i \forall_k \forall_j X_{ij} = \Sigma_k (B_{ik} * C_{kj})$$

Compiler flow

Expression

$$X_{ij} = B_{ik} * C_{kj}$$

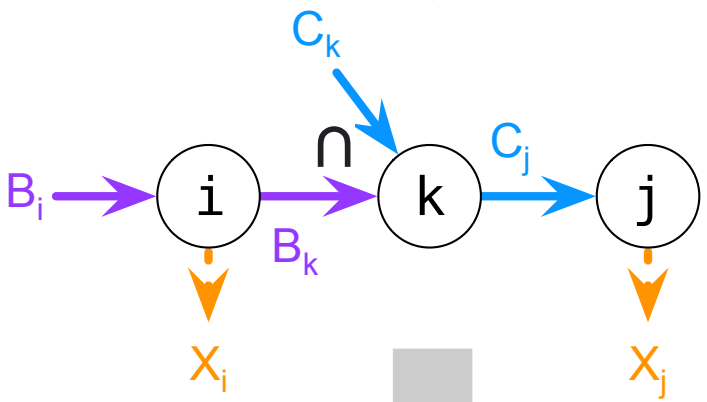
Format Language

```
B=({sparse,sparse},
   {mode0,mode1})
C=({sparse, sparse},
   {mode1, mode0})
```

Schedule

```
reorder(i, k, j)
```

$$\forall_i \forall_k \forall_j X_{ij} = \Sigma_k (B_{ik} * C_{kj})$$



Compiler flow

Expression

$$X_{ij} = B_{ik} * C_{kj}$$

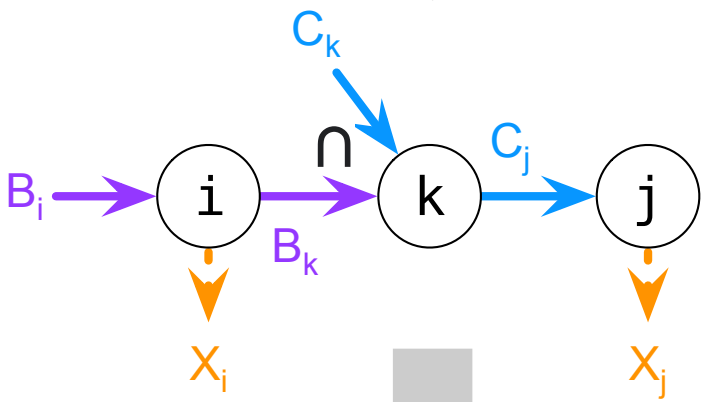
Format Language

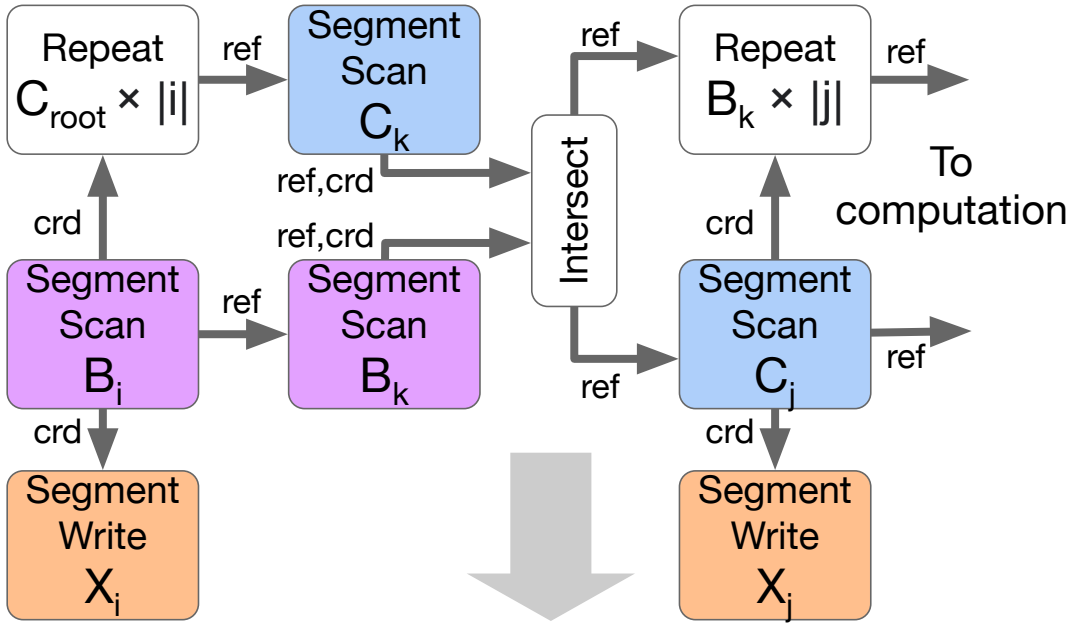
```
B=({sparse,sparse},
   {mode0,mode1})
C=({sparse, sparse},
   {mode1, mode0})
```

Schedule

```
reorder(i, k, j)
```

$$\forall_i \forall_k \forall_j X_{ij} = \Sigma_k (B_{ik} * C_{kj})$$





Lower to Example Implementation

Optimization extensions

Optimization extensions

- Tensor Locating
  - Coordinate skipping

Optimization extensions

- Tensor Locating
  - Coordinate skipping
- Parallelization
  - Bit vectors as streams (instead of coordinates)
  - Parallelize/Serialize

Optimization extensions

D, **S**, 12, 11, 9, 3, 2, 0 →

- Tensor Locating
  - Coordinate skipping
- Parallelization
  - Bit vectors as streams (instead of coordinates)
  - Parallelize/Serialize

Optimization extensions

- Tensor Locating
  - Coordinate skipping
- Parallelization
  - Bit vectors as streams (instead of coordinates)
  - Parallelize/Serialize

D, **S**, 12, 11, 9, 3, 2, 0 →

Assuming 4-bit elements

D, **S**, 1000, 0101, 0000, 1011 →

Optimization extensions

- Tensor Locating
  - Coordinate skipping
- Parallelization
  - Bit vectors as streams (instead of coordinates)
  - Parallelize/Serialize
- Splitting and Flattening

D, **S**, 12, 11, 9, 3, 2, 0 →

Assuming 4-bit elements

D, **S**, 1000, 0101, 0000, 1011 →

Optimization extensions

- Tensor Locating
  - Coordinate skipping

D, S, 12, 11, 9, 3, 2, 0 →

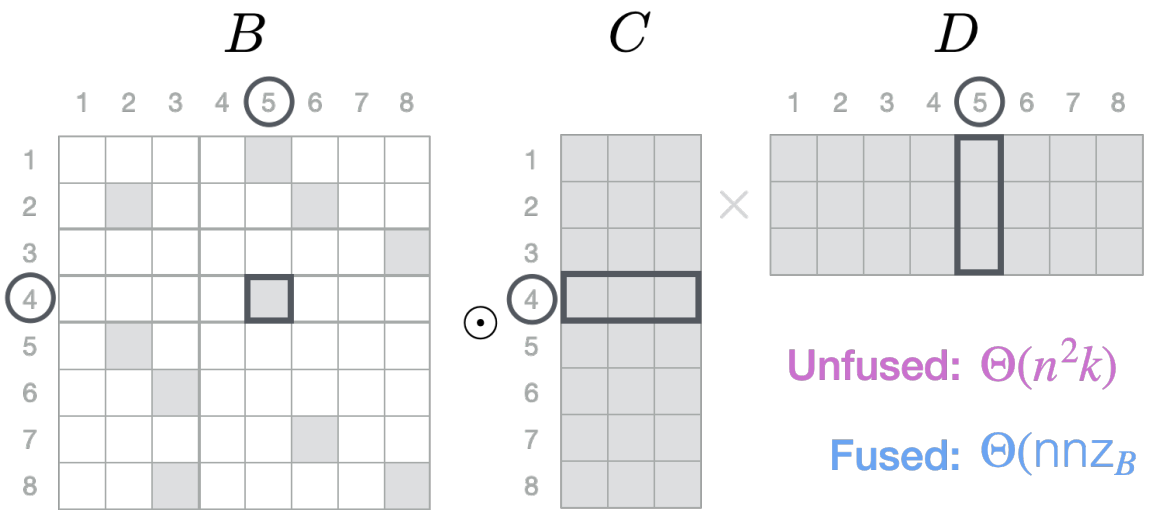
Assuming 4-bit elements

D, S, 1000, 0101, 0000, 1011 →

- Parallelization
  - Bit vectors as streams (instead of coordinates)
  - Parallelize/Serialize
- Splitting and Flattening

Fusion is key

$$A = B \odot (CD)$$

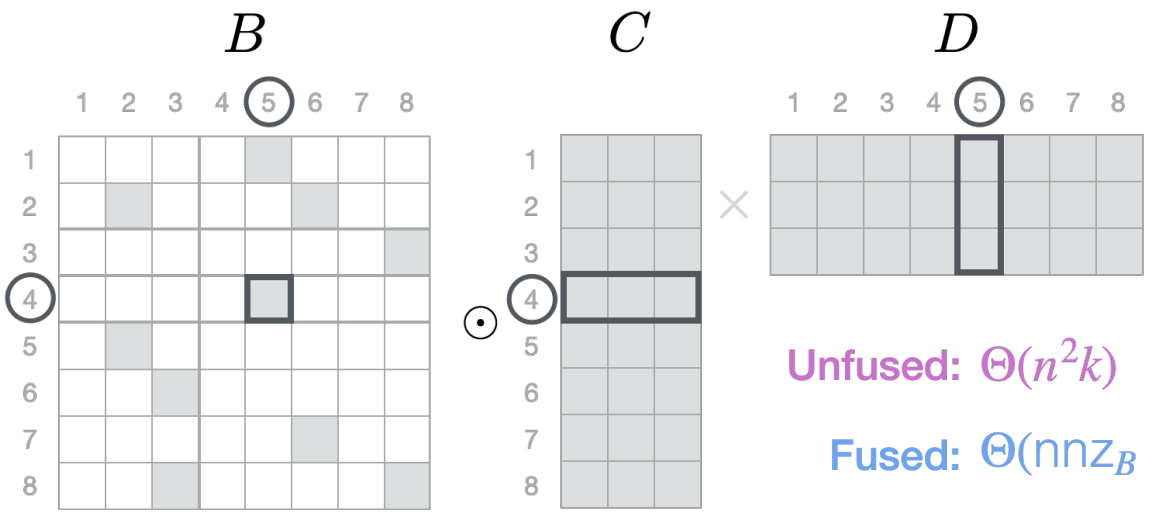


Unfused: $\Theta(n^2 k)$

Fused: $\Theta(\text{nnz}_B \cdot k)$

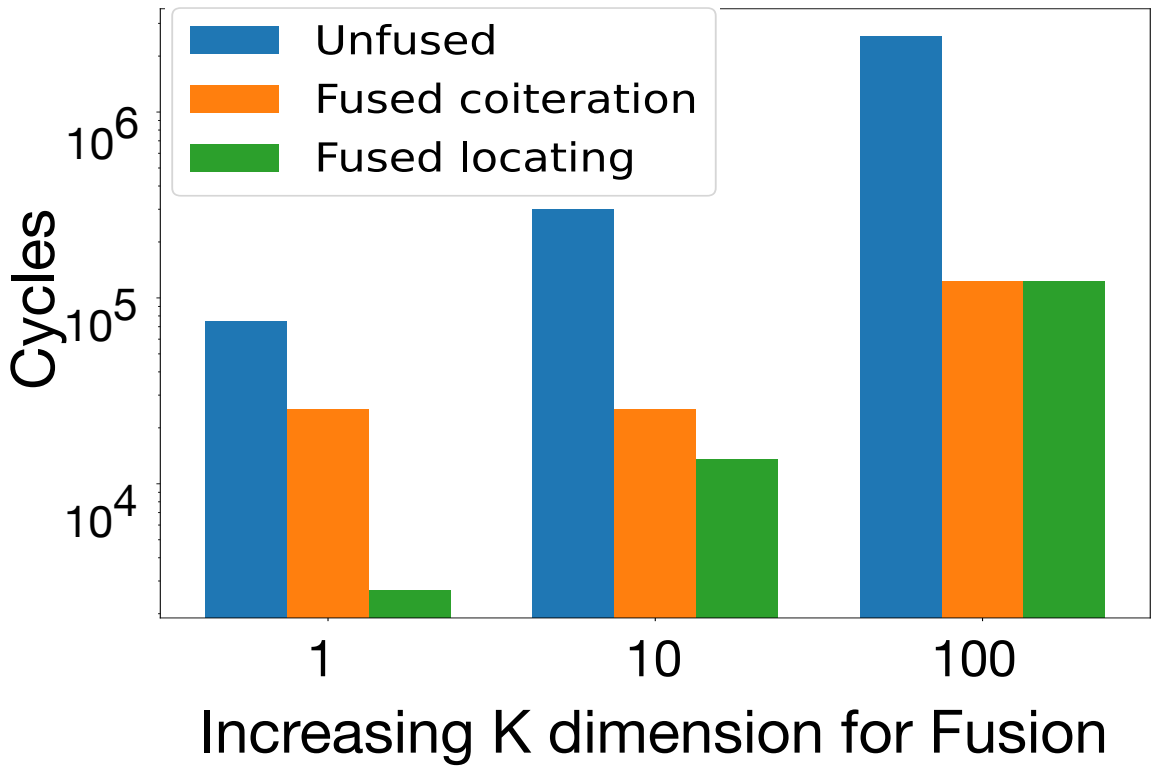
Fusion is key

$$A = B \odot (CD)$$

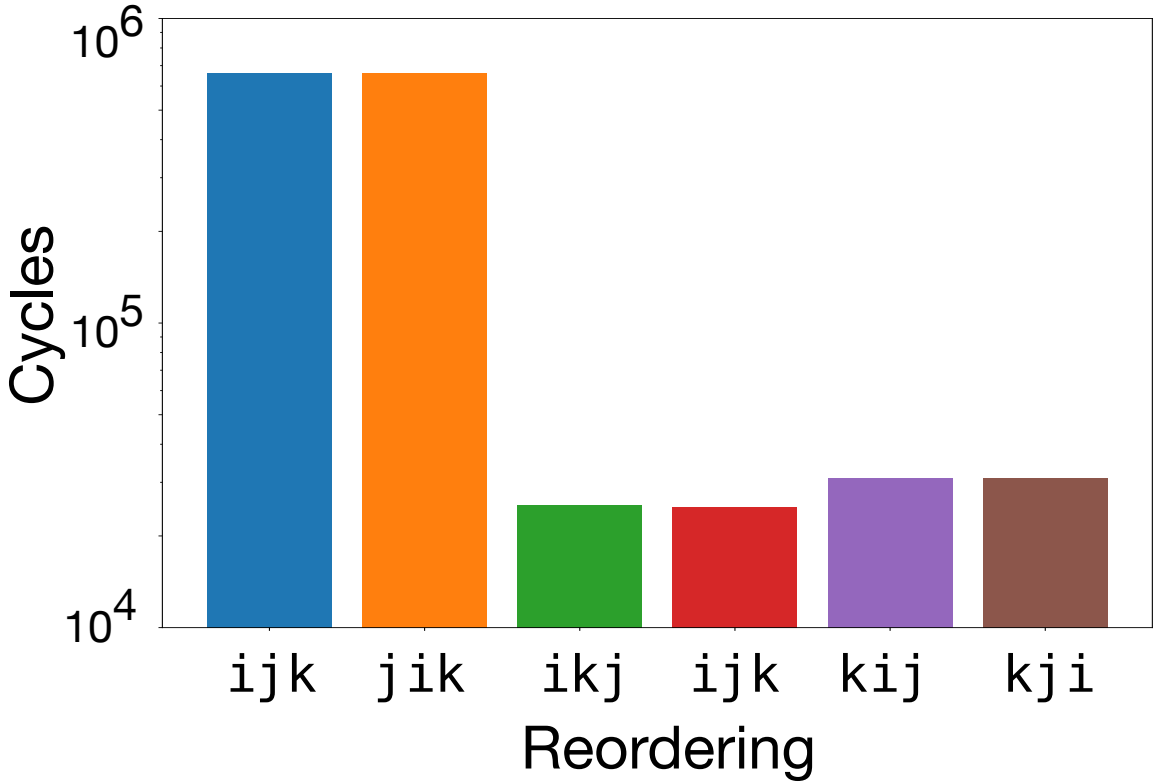


Unfused: $\Theta(n^2k)$

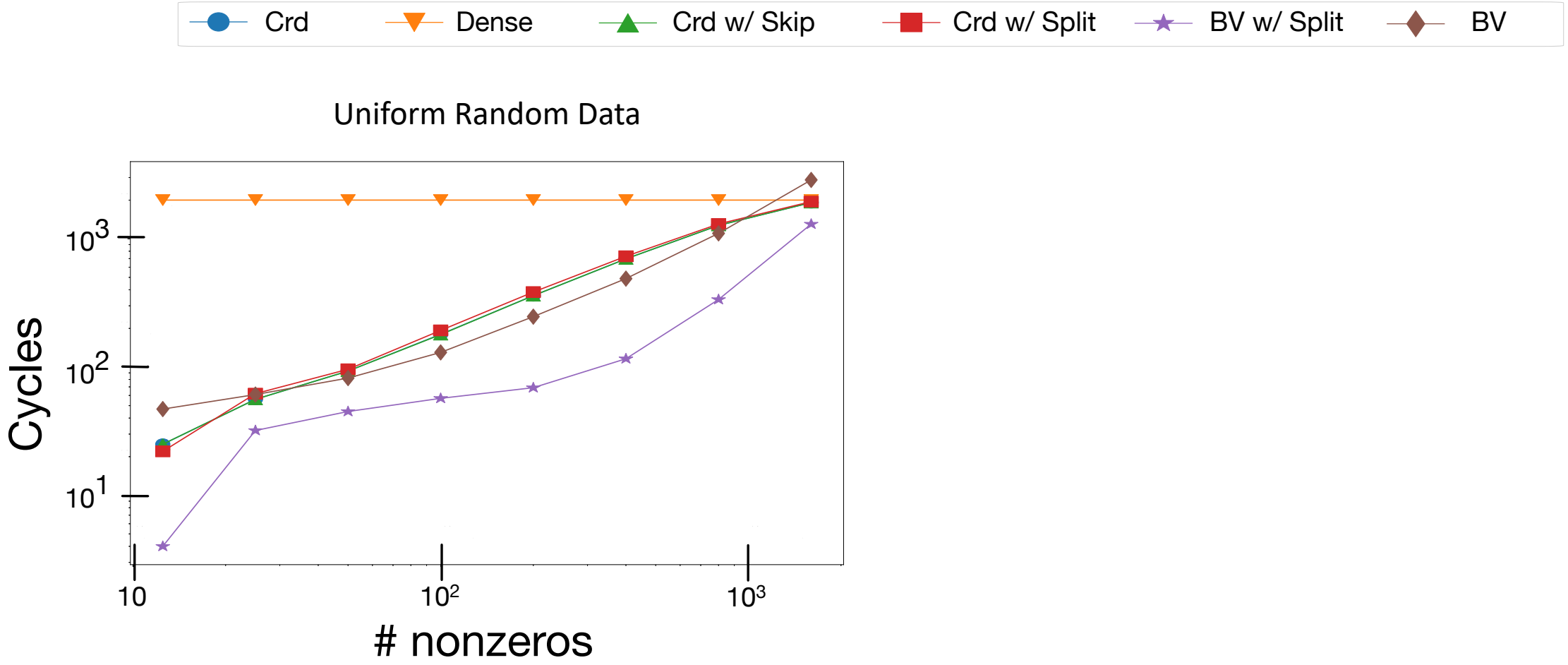
Fused: $\Theta(\text{nnz}_B \cdot k)$



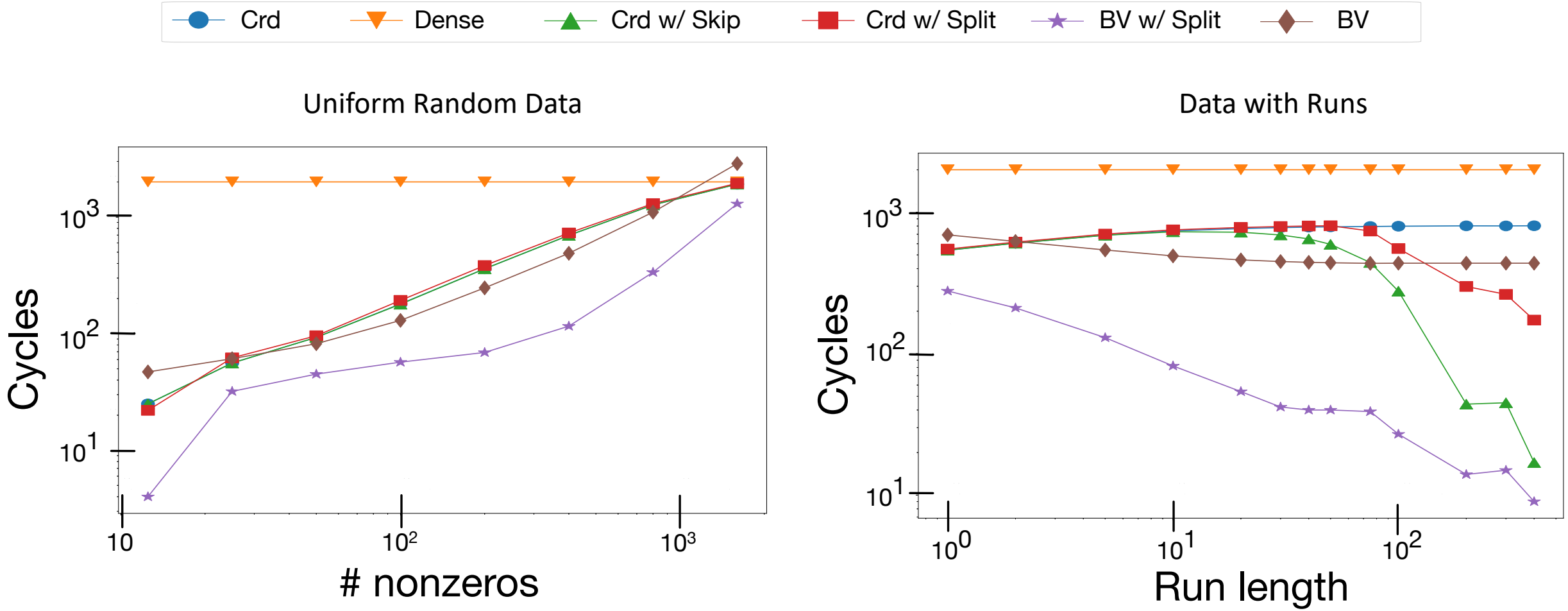
Dataflow matters for asymptotic behavior too



Optimizations depend on the data



Optimizations depend on the data



Impact and Future Work

Impact and Future Work

- Targets both fixed-function and reconfigurable dataflow accelerators

Impact and Future Work

- Targets both fixed-function and reconfigurable dataflow accelerators
- Unified abstraction for apples-to-apples comparison of sparse techniques and designs

Impact and Future Work

- Targets both fixed-function and reconfigurable dataflow accelerators
- Unified abstraction for apples-to-apples comparison of sparse techniques and designs
- End-to-end compilation and hardware generation

Impact and Future Work

- Targets both fixed-function and reconfigurable dataflow accelerators
- Unified abstraction for apples-to-apples comparison of sparse techniques and designs
- End-to-end compilation and hardware generation

- Generating the sparse design of Onyx with SAM

Impact and Future Work

- Targets both fixed-function and reconfigurable dataflow accelerators
- Unified abstraction for apples-to-apples comparison of sparse techniques and designs
- End-to-end compilation and hardware generation

- Generating the sparse design of Onyx with SAM
- Outer-level blocking and tiling for hardware memory systems

Impact and Future Work

- Targets both fixed-function and reconfigurable dataflow accelerators
- Unified abstraction for apples-to-apples comparison of sparse techniques and designs
- End-to-end compilation and hardware generation

- Generating the sparse design of Onyx with SAM
- Outer-level blocking and tiling for hardware memory systems
- Front-end compiler and simulator improvements

Impact and Future Work

- Targets both fixed-function and reconfigurable dataflow accelerators
- Unified abstraction for apples-to-apples comparison of sparse techniques and designs
- End-to-end compilation and hardware generation

- Generating the sparse design of Onyx with SAM
- Outer-level blocking and tiling for hardware memory systems
- Front-end compiler and simulator improvements
- SAM abstraction for sparse machine learning models