

TStreams

A Model of Parallel Computation

HP Cambridge Research Lab

Kath Knobe
Dave Panariti
Alex Nelson
Carl Offner
Dick Flower

Background

◆ Streams

- + very efficient
- limited to FIFO

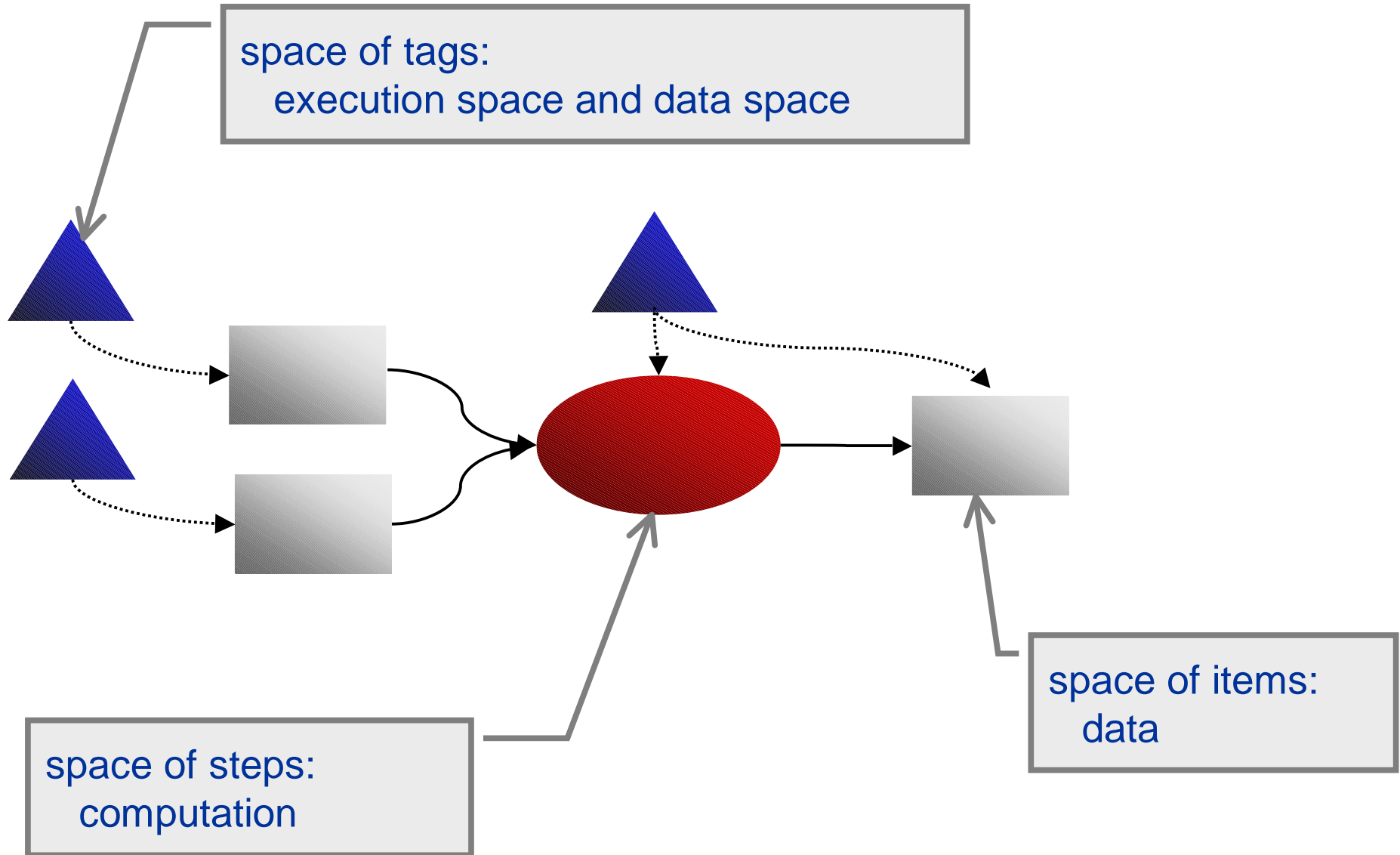
◆ Functional

- + scheduling flexible
- regular accesses not efficient

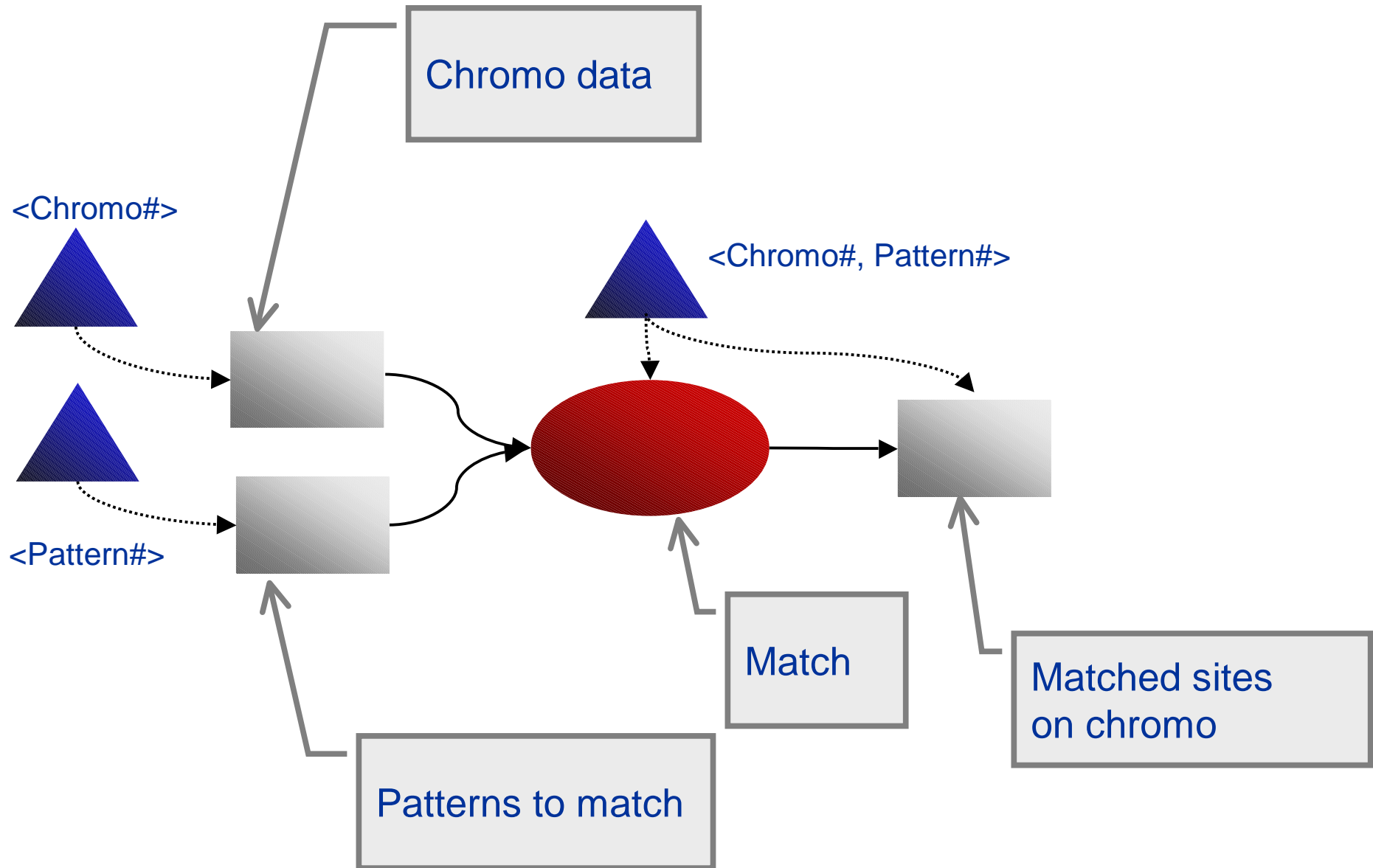
◆ Tuple spaces

- + decoupled producer/consumer
- not functional: scheduling constrained by overwriting

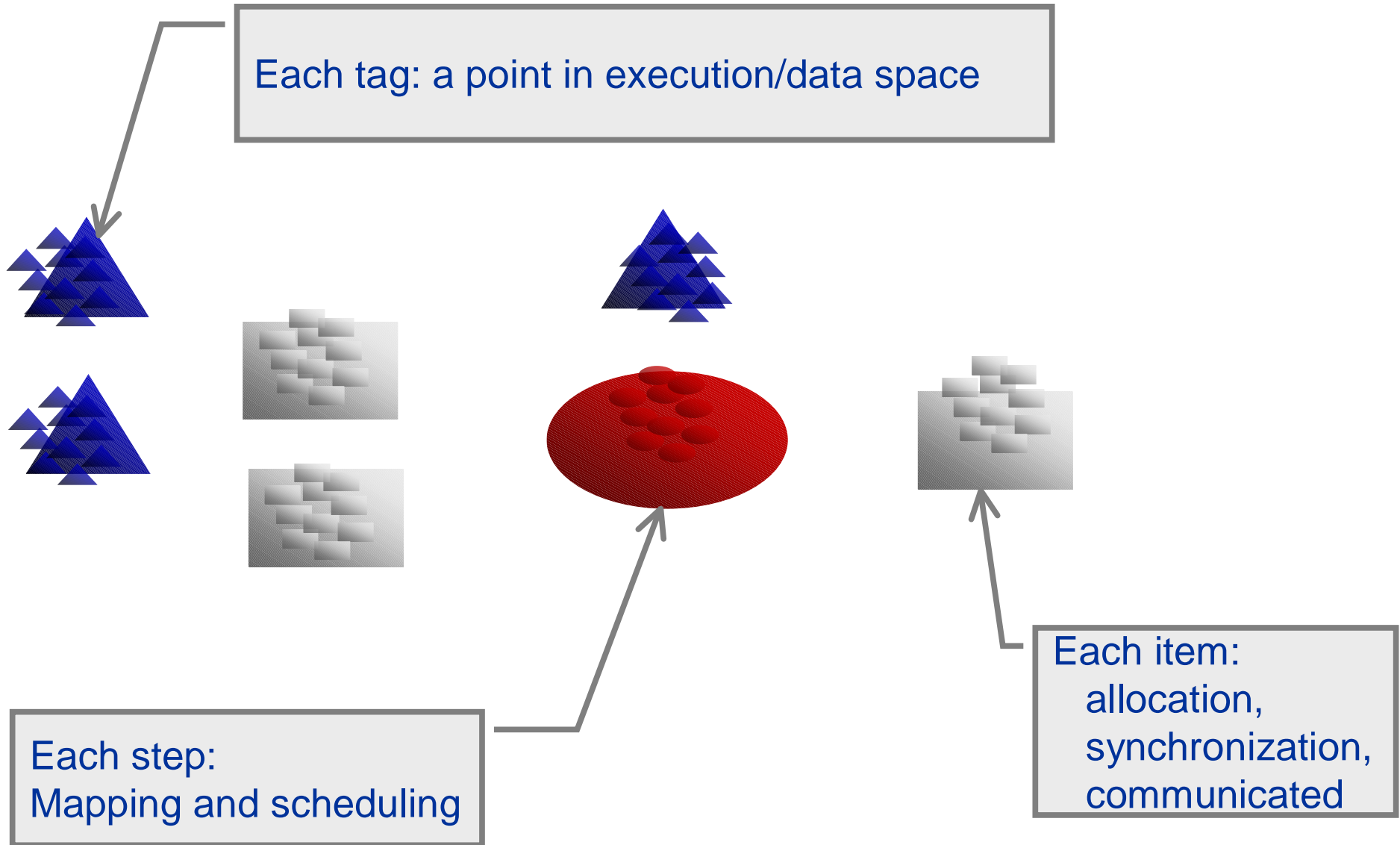
Static Components



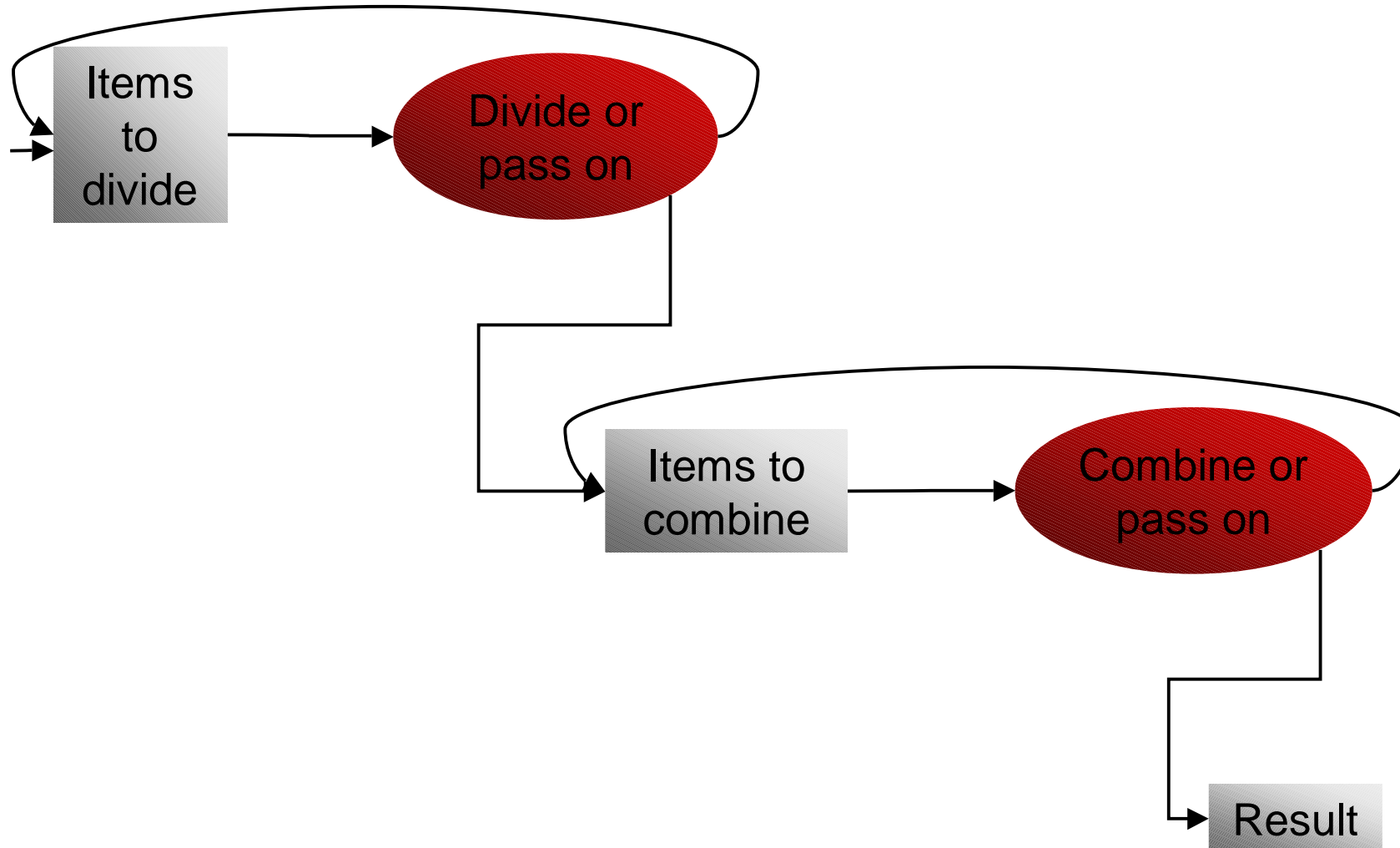
Example 1: Gene Matching



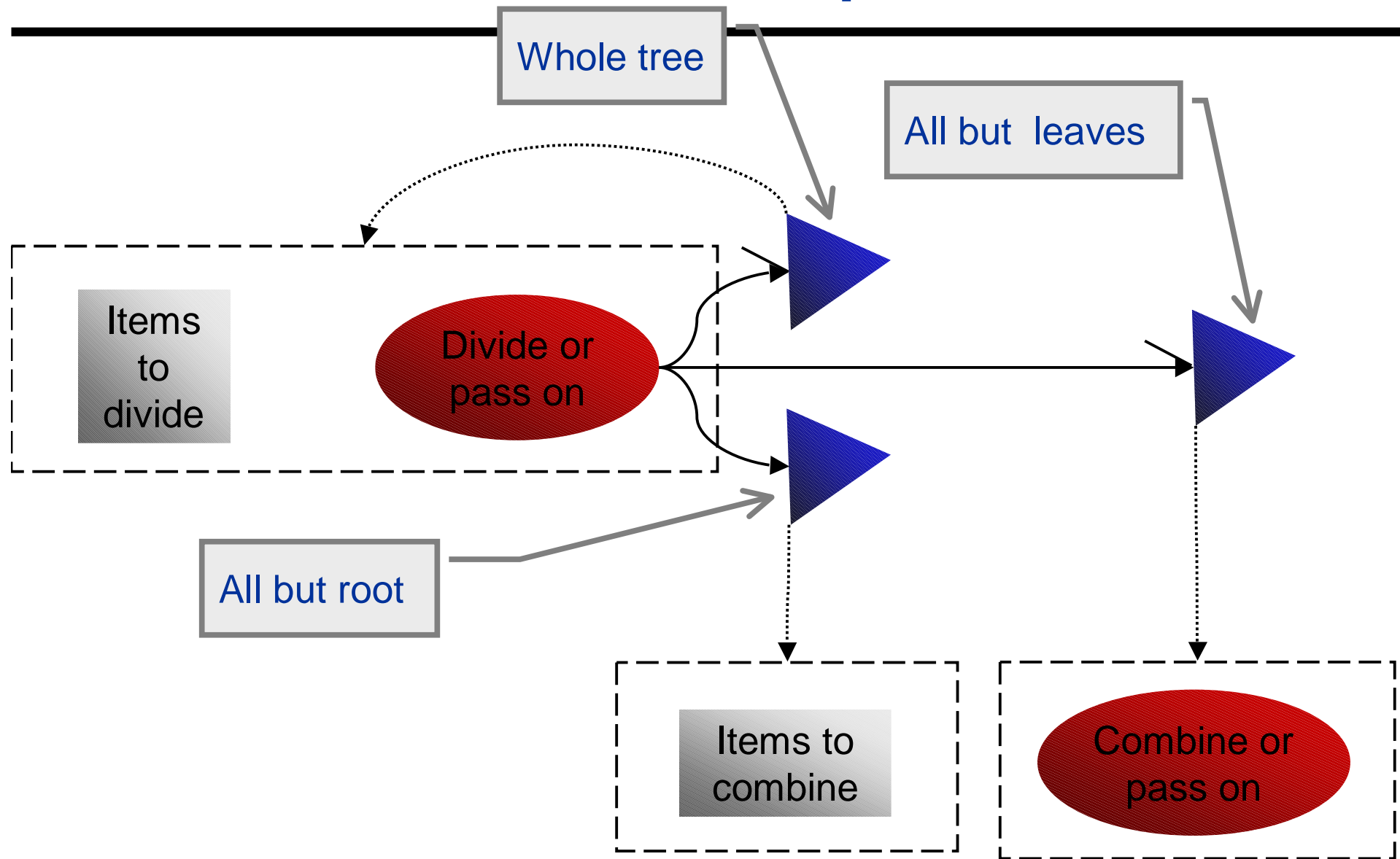
Dynamic Instances



Example 2: Divide & Conquer



Divide & Conquer



Properties

◆ State

- tags in tag spaces, steps executed, produced items

◆ Atomicity

- Tag: in a tag space or not
- Step: executed or not
- Item: produced or not

◆ Implications

- redundant tag production, step execution, item production
- start, drop and restart execution of steps
start, drop and restart production of a tag or an item

◆ Dynamic, continuous, transparent checkpointing

Execution Styles

	grain	map	schedule
data parallel			
task parallel	static	static	static
pipeline parallel			
data flow	static	static	dynamic
reliable	static	dynamic	dynamic
adaptive	In progress	dynamic	dynamic

Current implementation

In progress

The diagram is a 2D matrix with the following data points:

- Row: data parallel
- Row: task parallel (Current implementation)
- Row: pipeline parallel
- Row: data flow
- Row: reliable
- Row: adaptive (In progress)
- Column: grain
- Column: map
- Column: schedule

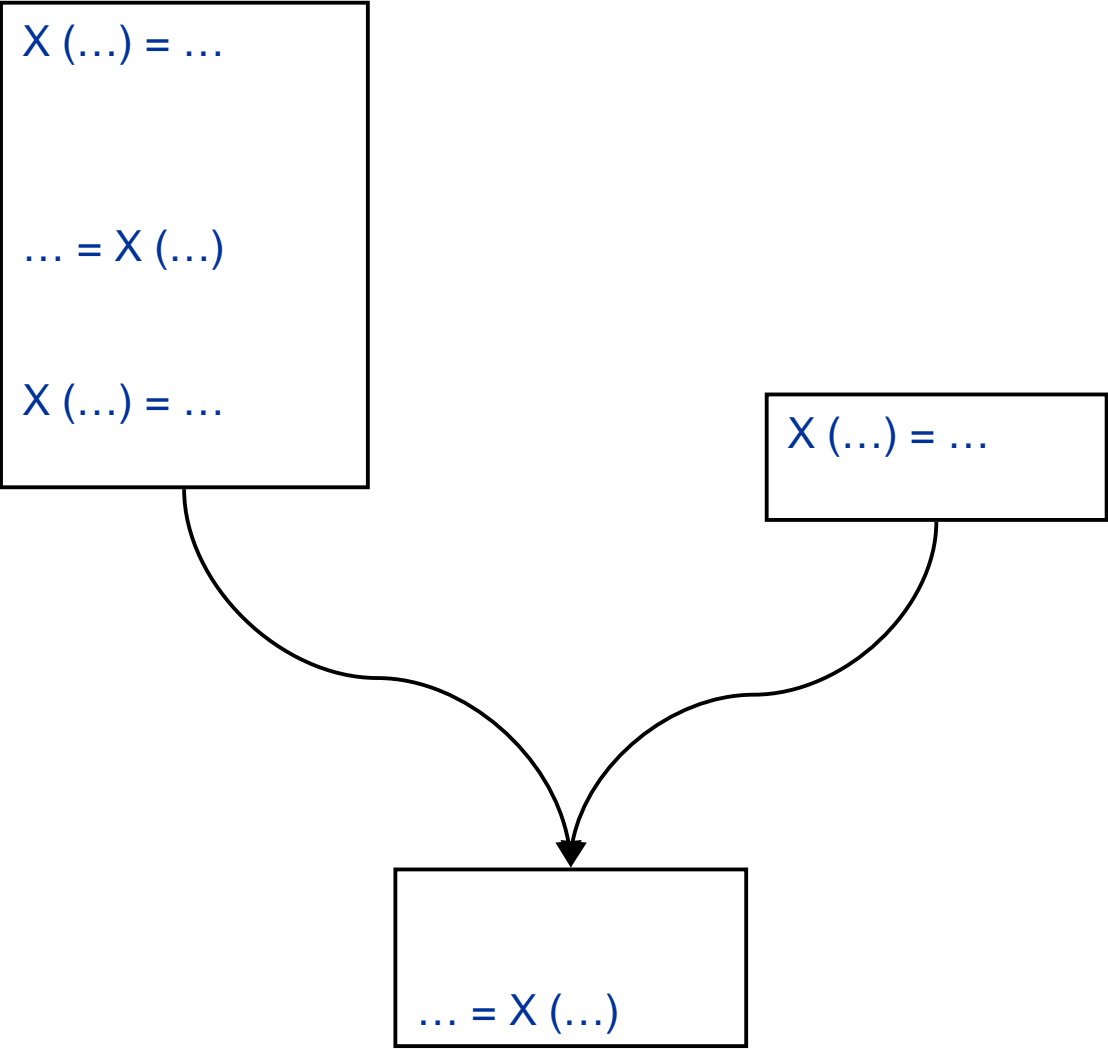
Characteristics for each cell:

- (task parallel, grain): static
- (task parallel, map): static
- (task parallel, schedule): static
- (data flow, grain): static
- (data flow, map): static
- (data flow, schedule): dynamic
- (reliable, grain): static
- (reliable, map): dynamic
- (reliable, schedule): dynamic
- (adaptive, grain): In progress
- (adaptive, map): dynamic
- (adaptive, schedule): dynamic

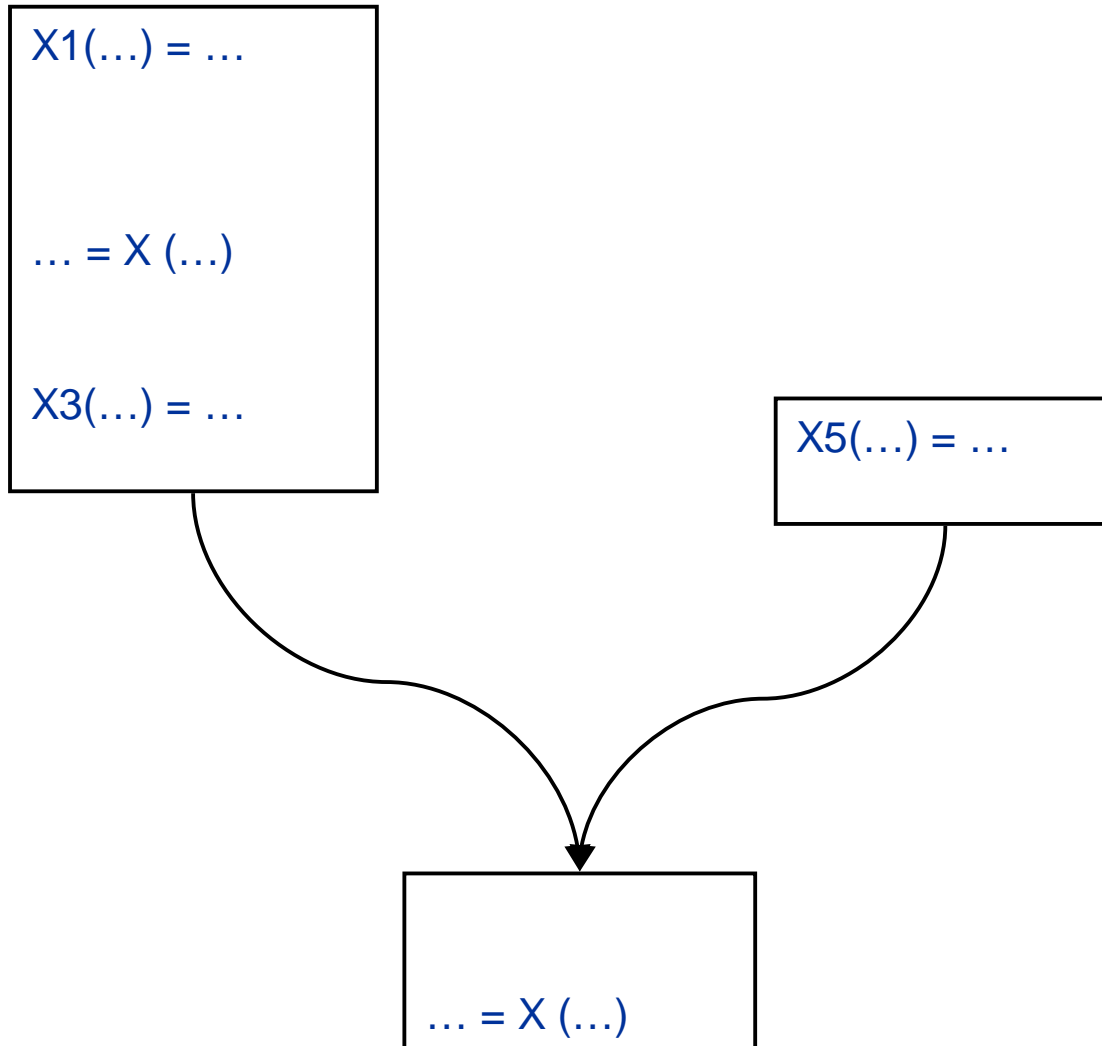
Compiler Algorithms: Transform Normal Programs to TStreams

- ◆ Subspace Analysis (Bill Dally)
- ◆ Array SSA (Vivek Sarkar)
- ◆ DSA (Carl Offner)

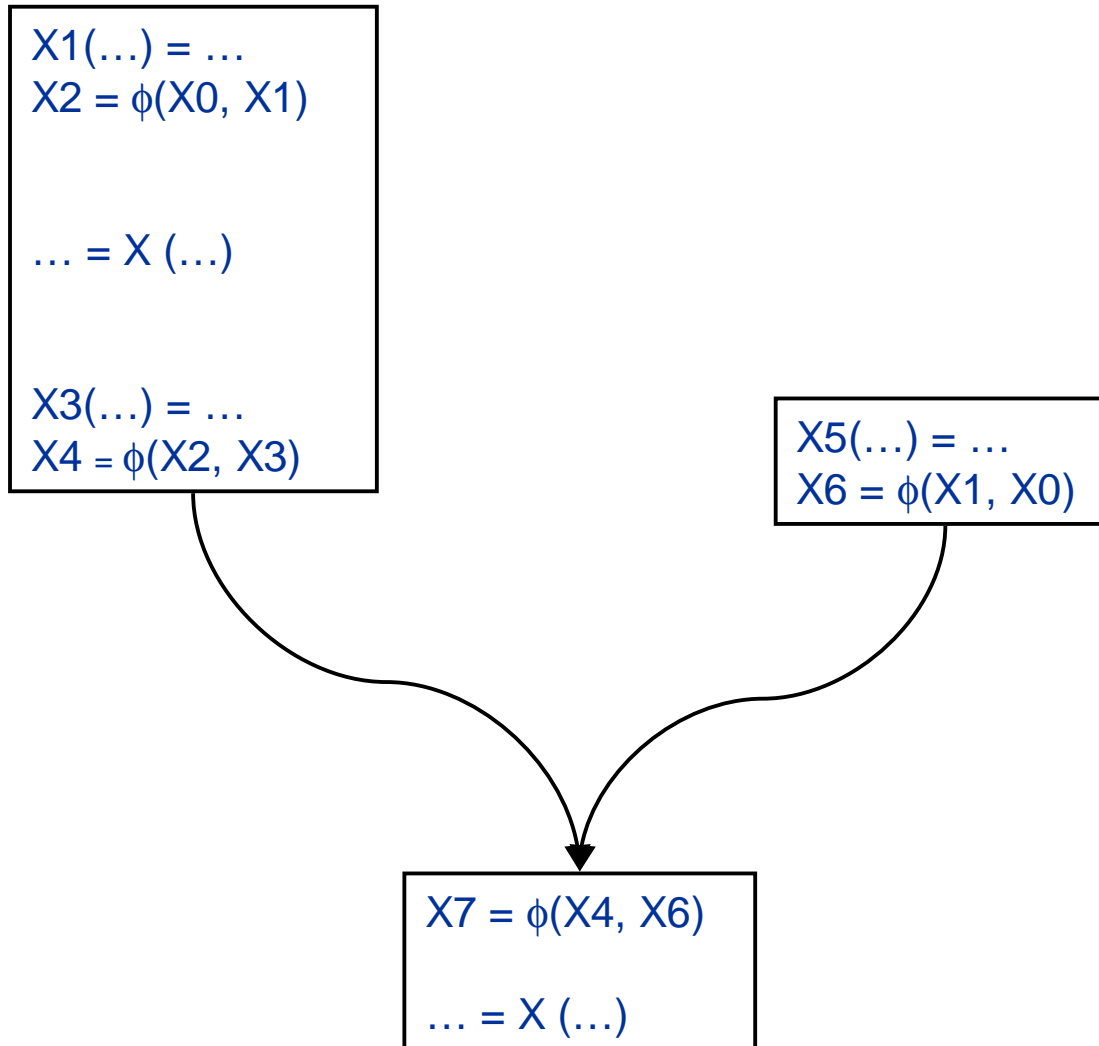
Original Code



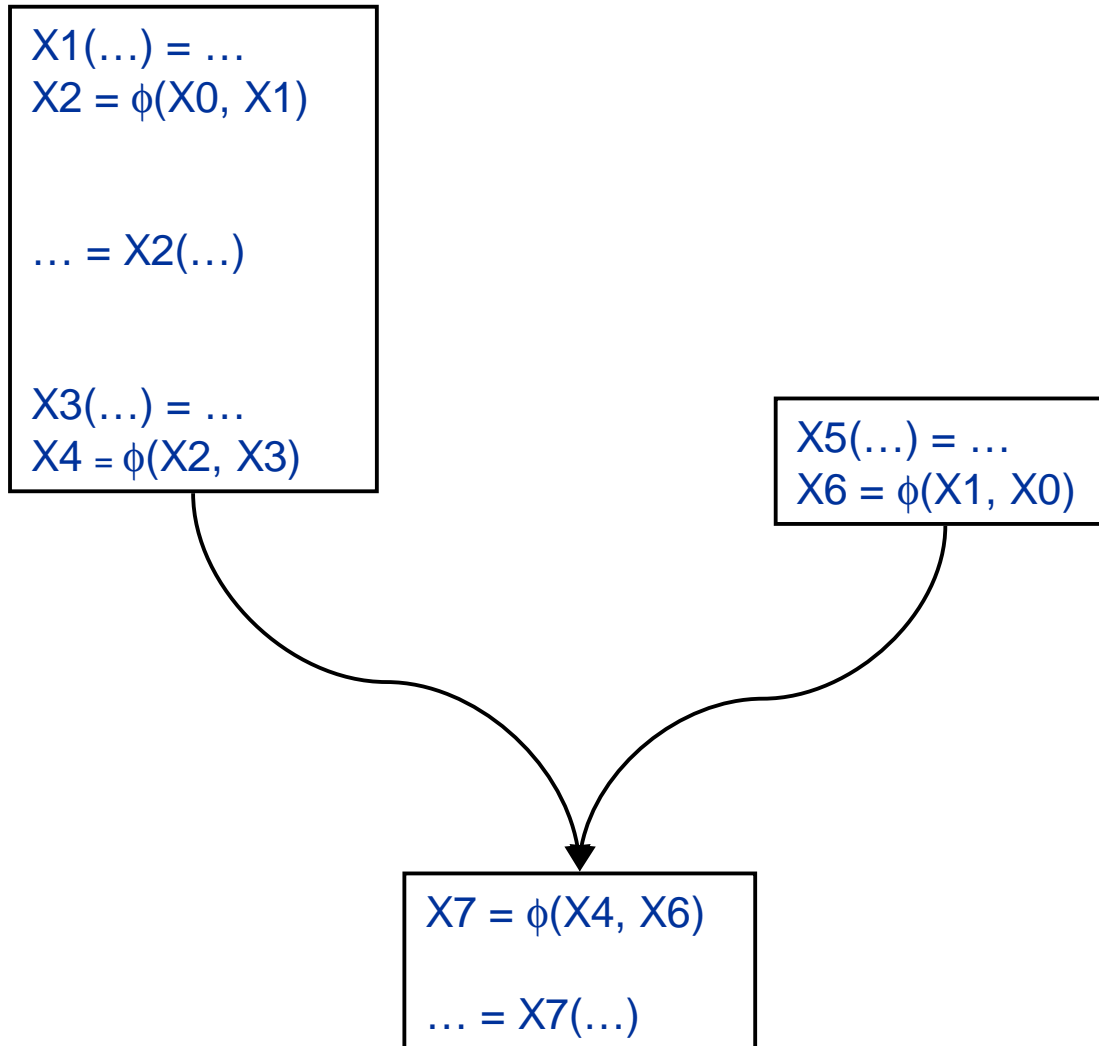
Static Single Assignment



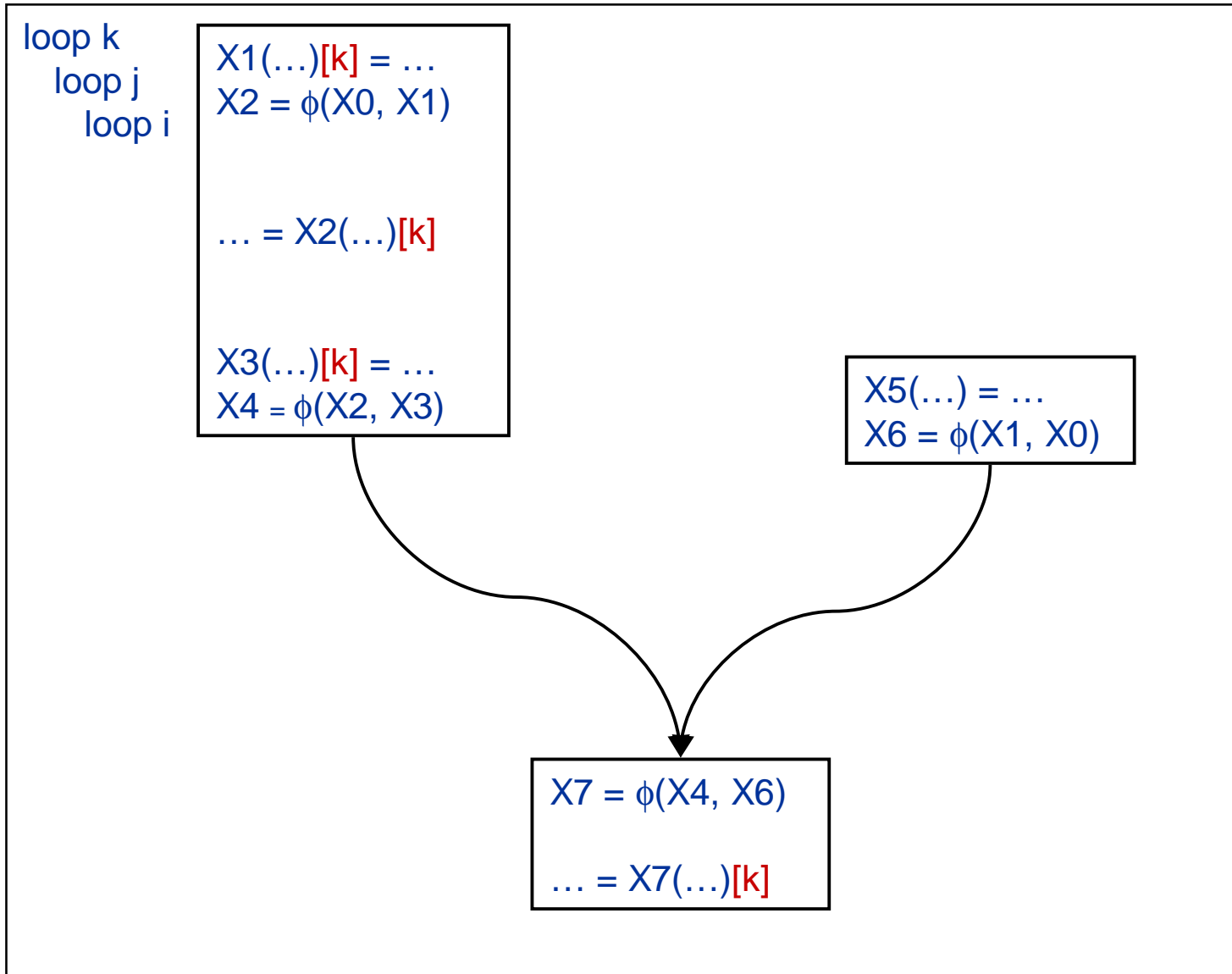
Static Single Assignment



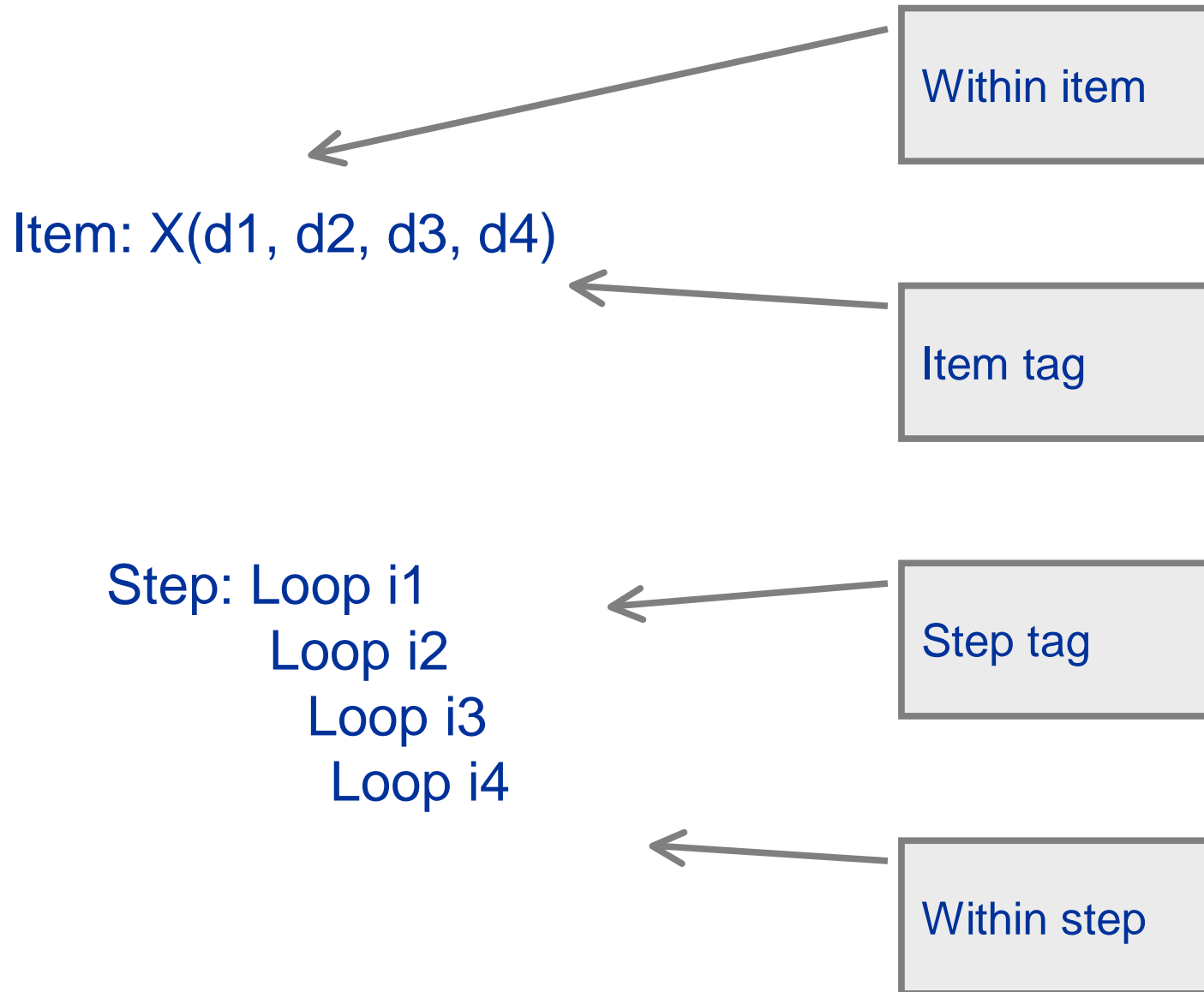
Static Single Assignment



Dynamic Single Assignment



Tags



Summary

- ◆ **Best of streaming languages, functional languages and tuple-spaces**
- ◆ **Versatile execution styles**
- ◆ **Automatic generation**
- ◆ **(Hierarchical)**
- ◆ **(real-time)**

END

Mappings

