A language approach to accelerating dynamic programming problems on the GPU

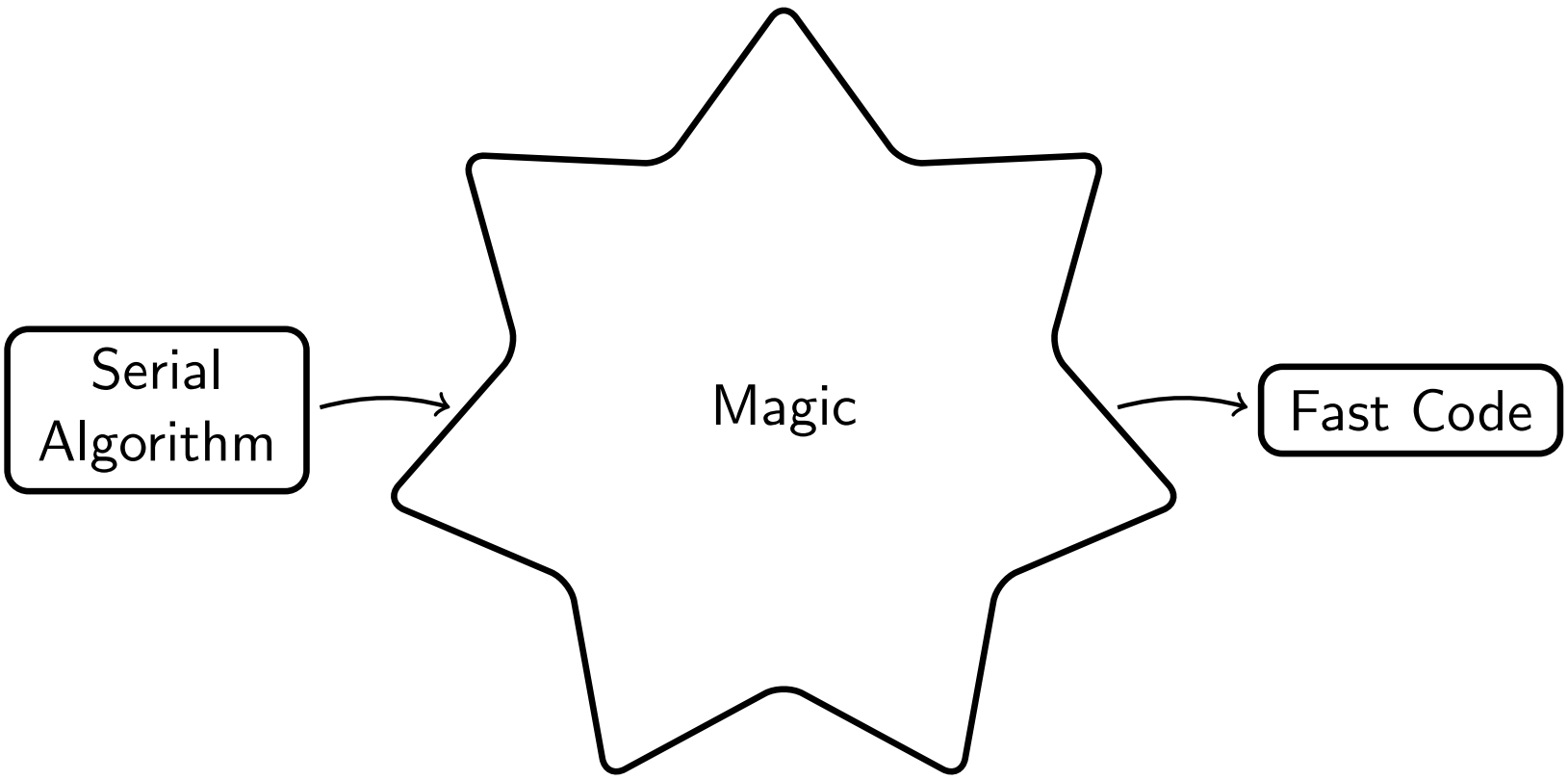
Luke Cartey

Programming Tools Group, Computing Laboratory, University of Oxford

14th December 2010

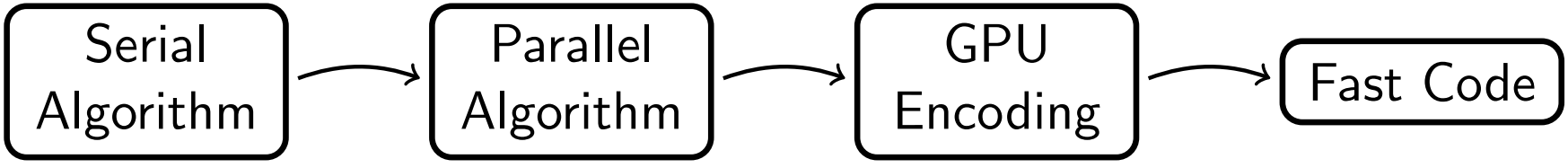
Mapping problems is (can be?) hard

What the user wants:



Mapping problems is (can be?) hard

What the user gets:

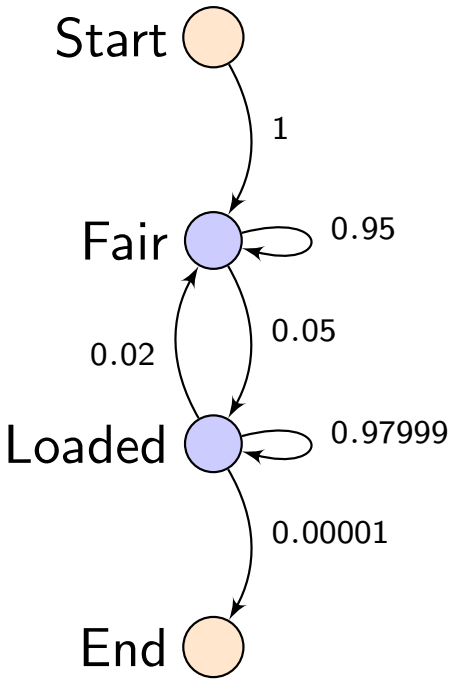


Difficult Questions?

- Which parts of the program will we parallelise?
- Can we appeal to data-parallelism? Problem-level parallelism?
- Size of the block? Size of the grid?
- Shared memory? Cached memory? Registers?

Can we answer these questions in a meaningful way if we focus on domains?

Bioinformatics as a domain



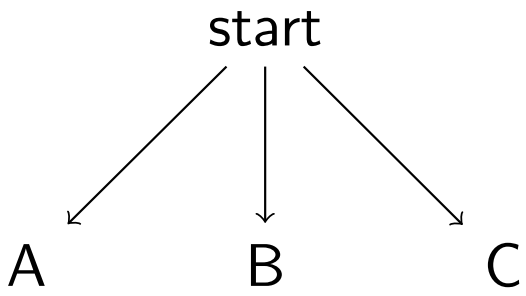
HMMingbird

- A Hidden Markov Model compiler
- Targets NVIDIA GPUs, using C for CUDA.
- Describes a fixed set of algorithms using templates.

Roadmap

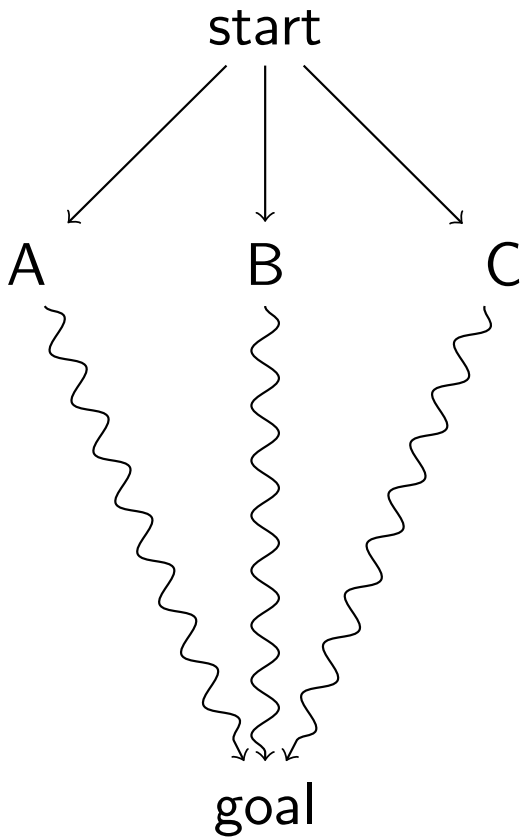
- Data Definition Layer - HMMingbird
- Algorithm Definition Layer - Functional language for optimisation problems

Optimisation Problems



Choices

Optimisation Problems



Choices

Optimal Substructure

Edit distance

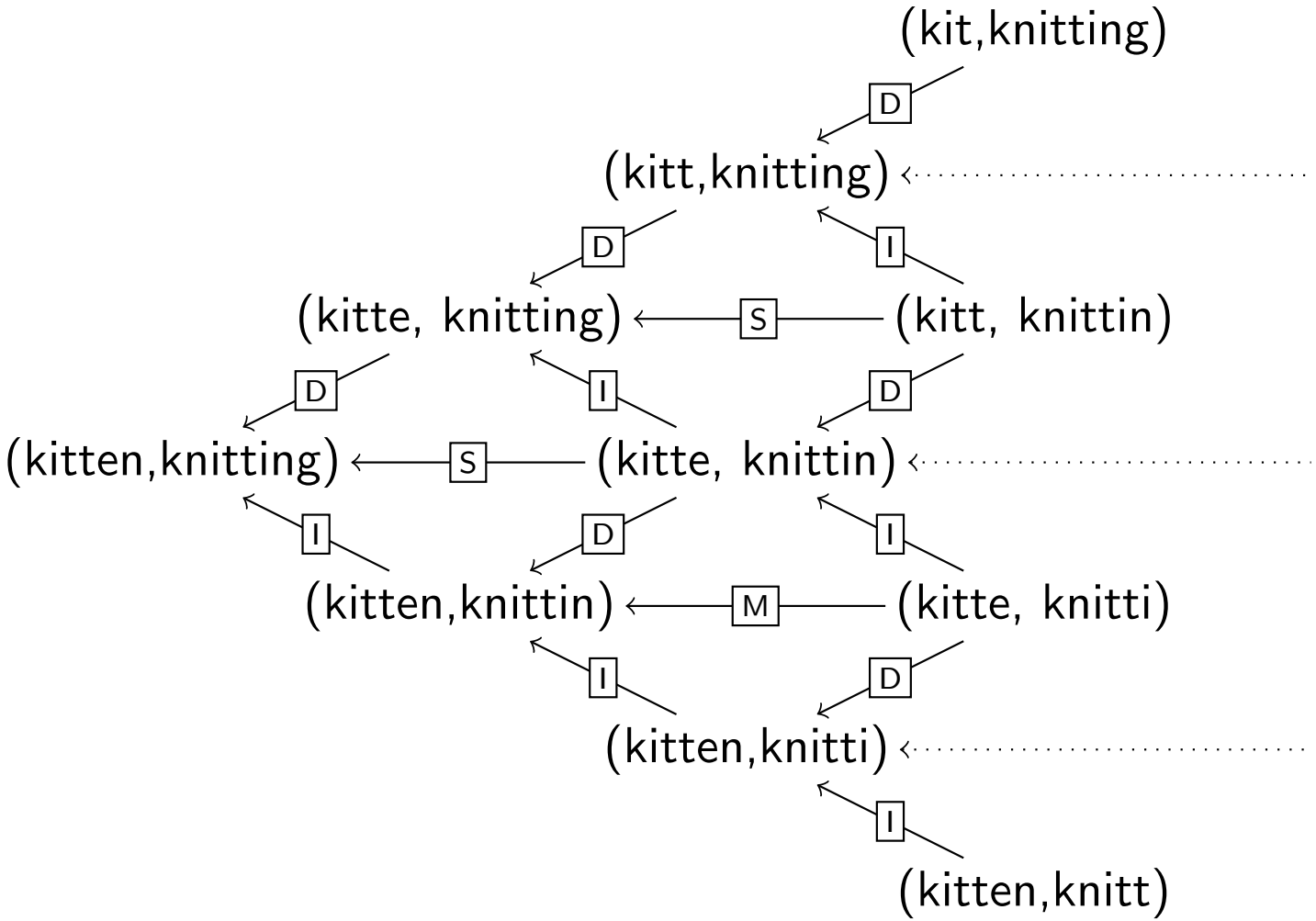
Minimum operations required to transform one string to another.
e.g

$$\text{kitten} \implies \text{kitten}\mathbf{g}\ (\text{I})$$

$$\text{kitteng} \implies \text{kitt}\mathbf{i}\text{ng}\ (\text{S})$$

$$\text{kitting} \implies \text{k}\mathbf{n}\text{itting}\ (\text{I})$$

Graph problem



Recursion

$$f(x,y) = \begin{cases} x & \text{if } y = 0 \\ y & \text{if } x = 0 \\ f(x-1,y-1) & \text{if } s[x] = t[y] \\ min(f(x-1,y), & \\ \quad f(x,y-1), & \text{otherwise} \\ \quad f(x-1,y-1)) + 1 & \end{cases}$$

Recursion in language

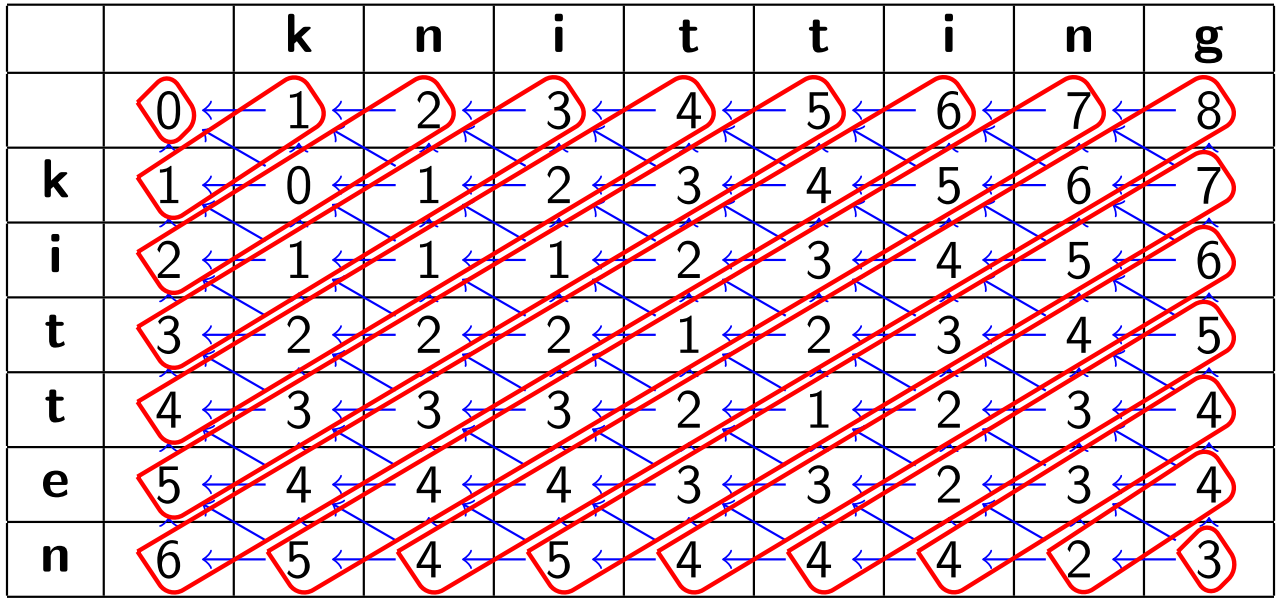
```
f(String s[x],String t[y]) = if x = 0 then y
          else if y = 0 then x
          else if s[x - 1] = t[y - 1] then f(x - 1,y - 1)
          else min(f(x - 1, y),f(x, y - 1), f(x - 1, y- 1)) + 1

f("kitten", "knitting")
```

Partitions

		k	**n**	**i**	**t**	**t**	**i**	**n**	**g**
	0	1	2	3	4	5	6	7	8
k	1	0	1	2	3	4	5	6	7
i	2	1	1	1	2	3	4	5	6
t	3	2	2	2	1	2	3	4	5
t	4	3	3	3	2	1	2	3	4
e	5	4	4	4	3	3	2	3	4
n	6	5	4	5	4	4	4	2	3

Partitions


k n i t t i n g
0 1 2 3 4 5 6 7 8
k 1 0 1 2 3 4 5 6 7
i 2 1 1 1 2 3 4 5 6
t 3 2 2 2 1 2 3 4 5
t 4 3 3 3 2 1 2 3 4
e 5 4 4 4 3 3 2 3 4
n 6 5 4 5 4 4 4 2 3

Partition function

Describe each partition based on the parameters e.g

$$P_f(x, y) = x + y$$

Proving suitability of a partition function

Consider each recursive call in turn using descent functions.

$$f(d_x(x,y), d_y(x,y))$$

e.g

$$f(\ x-1\ ,\ \ y-1\ )$$

Verify the recursive equation satisfies the partition function:

$$\forall_{x,y} P_f(x,y) > P_f(d_x(x,y), d_y(x,y))$$

e.g

$$\forall_{x,y} x + y > (x-1) + (y-1)$$

More analysis

Termination

Partition base case is when $P_f(x, y) = 0$:

The base case of the recursion needs to satisfy this.

Out of bounds check

We must also check the recursion never jumps out of bounds.

GPU Mapping

- Data-parallel/Problem-parallel
- Block size, grid size
- Sliding window optimisations

Questions solved?

Have we answered these?

- Which parts of the program will we parallelise?
- Can we appeal to data-parallelism? Problem-level parallelism?
- Size of the block? Size of the grid?
- Shared memory? Cached memory? Registers?

Further work

Tie work into higher-level domains for Bioinformatics
(e.g Hidden Markov Models)
Permit more sophisticated recursions