Regions

- A region is a (typed) collection

- Regions are the cross product of
  - An index space
  - A field space

Example 9

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>true</td>
</tr>
<tr>
<td>8</td>
<td>true</td>
</tr>
<tr>
<td>9</td>
<td>false</td>
</tr>
</tbody>
</table>

Discussion

- Regions are the way to organize large data collections in Regent

- Can have any number of fields

- Built-in support for 1D, 2D and 3D index spaces
Privileges
- A task that takes region arguments must
  - Declare its privileges on the region
  - Reads, Writes, Reduces
- The task may only perform operations for which it has privileges
  - Including any subtasks it calls
- Example 10

Legion Spy
- A tool for showing ordering dependencies
  - Very useful for figuring out why things are not running in parallel
- Workflow
  - Use Legion Prof to find idle time (white space)
  - Use Legion Spy to examine tasks that are delayed
    - What are they waiting for?!
- Example 11

Partitioning
- To enable parallelism on a region, partition it into smaller pieces
  - And then run a task on each piece
- Steps:
  - Color elements of the region
  - Partition the region, creating one subregion for each color

Partitioning Example
<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>false</td>
</tr>
<tr>
<td>1</td>
<td>false</td>
</tr>
<tr>
<td>2</td>
<td>false</td>
</tr>
<tr>
<td>3</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>false</td>
</tr>
<tr>
<td>5</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>true</td>
</tr>
<tr>
<td>7</td>
<td>true</td>
</tr>
<tr>
<td>8</td>
<td>true</td>
</tr>
<tr>
<td>9</td>
<td>false</td>
</tr>
</tbody>
</table>
**Partitioning Example**

- Example 12
  - Partitioning does not create copies
    - It names subsets of the data
  - Partitioning does not remove the parent
    - It still exists and can be used
  - Regions and partitions are first-class values
    - Can be created, destroyed, stored in data structures, passed to and returned from tasks

**More Discussion**

- The same data can be partitioned multiple ways
  - Again, these are just names for subsets
- Subregions can themselves be partitioned

**Region Trees**
Dependence Analysis

- Regent uses tasks region declarations to compute which tasks can run in parallel
  - What region is being accessed
    - Does it overlap with another region that is in use?
  - What field is being accessed
    - If a task is using an overlapping region, is it using the same field?
  - What are the privileges?
    - If two tasks are accessing the same field, are they both reading or both reducing?

Coherence

- Coherence is a fourth dimension of information for dependence analysis
  - How are other tasks allowed to use the region?

- For today, all coherence is exclusive
  - A task always has exclusive access to region arguments
  - The default (no need to declare)

A Crucial Fact

- Regent analyzes sibling tasks
  - Tasks launched directly by the same parent task

- Theorem: Analyzing dependencies between sibling tasks is sufficient to guarantee sequential semantics

- Never check for dependencies otherwise
  - Crucial to the overall design of Regent

Consequences

- Dependence analysis is a source of runtime overhead

- Can be reduced by reducing the number of sibling tasks
  - Group some tasks into subtasks

- But beware
  - This may also reduce the available parallelism

- Example 14
Example 14

- Note that passing a region to a task does not mean the data is copied to where that task runs
  - C.f., launcher task must name the parent region for type checking reasons
- If the task doesn’t touch a region/field, that data doesn’t need to move

Fills

- A better way to initialize regions is to use fill operations
  \[
  \text{fill}(\text{region.field}, \text{value})
  \]
- Example 15

Multiple Partitions

- Different views onto the same data
- Again, can have multiple views in use at the same time
- Regent will figure out the data dependencies
  - Example 16 & 17

Discussion
Aliased Partitions

Example 18

- Equal partitions
- Aliased partitions

Summary

- Significant Regent applications have interesting region trees
  - Multiple views
  - Aliased partitions
  - Multiple levels of nesting

- And complex task dependencies
  - Subregions, fields, privileges, coherence

- Regions express locality
  - Data that will be used together
  - An example of a "local address space" design
    - Tasks can only access their region arguments