Towards Pervasive Incremental Computing
What is Incremental Computation?

A program is incremental if repeating it with a changed input is faster than from-scratch recomputation.
What is Incremental Computation?

4, 2, 6, 3, 1

Sort \( O(N \log N) \)

1, 2, 3, 4, 6
What is Incremental Computation?

Sort $O(N \log N)$

$4,2,6,3,1$ $\rightarrow$ Append $\rightarrow$ $4,2,6,3,1,5$

$1,2,3,4,6$
What is Incremental Computation?

4,2,6,3,1 → Append → 4,2,6,3,1,5

Sort O(N log N)

1,2,3,4,6

Sort O(N log N)

1,2,3,4,5,6
What is Incremental Computation?

4,2,6,3,1 \rightarrow \text{Append} \rightarrow 4,2,6,3,1,5

Sort \quad \text{O(N log N)}

1,2,3,4,6 \rightarrow \text{Insert} \quad \text{O(N)} \rightarrow 1,2,3,4,5,6

Sort \quad \text{O(N log N)}
What is Incremental Computation?

Insert into Heap
\(O(N \log N)\)

4, 2, 6, 3, 1

Using a specialized data structure allows incremental sorting

1, 2, 3, 4, 5, 6

1, 2, 3, 4, 5, 6
What is Incremental Computation?

Computing a Fibonacci number

```
fib(5)
fib(4)  fib(3)
fib(3)  fib(2)  fib(2)  fib(1)  fib(1)
fib(2)  fib(1)  fib(1)  fib(0)  fib(1)  fib(0)
fib(2)  fib(1)  fib(0)  fib(1)  fib(0)
```
What is Incremental Computation?

Computing a Fibonacci number

Employing caches allows incremental functions
Expanding Incremental Computation

Our goal is to provide access to incremental computation for all programs
Expanding Incremental Computation

Our goal is to provide access to incremental computation for all programs

Do these scale?
Are they accurate?

caching

reactive programming

decorators

libraries

invertible functions

incremental data structures

function derivatives

languages
Expanding Incremental Computation

Our goal is to provide access to incremental computation for all programs

Do these scale?
Are they accurate?

caching
decorators
invertible functions
function derivatives

reactive programming
libraries
incremental data structures
languages
Adapton

Build and maintain a partial-order call graph of cached results

On-demand re-evaluation
Minimum update paths
From-scratch consistent
Dynamic call-graph
Adapton

Build and maintain a partial-order call graph of cached results

On-demand re-evaluation
Minimum update paths
From-scratch consistent
Dynamic call-graph
Adapton

Build and maintain a partial-order call graph of cached results

On-demand re-evaluation
Minimum update paths
From-scratch consistent
Dynamic call-graph
Adapton

Build and maintain a partial-order call graph of cached results

On-demand re-evaluation
Minimum update paths
From-scratch consistent
Dynamic call-graph
Adapton

Build and maintain a partial-order call graph of cached results

If Func2 and Func3 are the same function with different parameters, how should the cache and nodes be updated?
Incremental Sequences

3 8 1 4 7 3
Incremental Sequences

Update 5th Element
Incremental Sequences

Insert

Update

10th Element
Incremental Sequences

Use names to identify specific sections

```
<table>
<thead>
<tr>
<th>N1</th>
<th>N2</th>
<th>N3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>N1</th>
<th>N4</th>
<th>N5</th>
<th>N2</th>
<th>N3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>13</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Incremental Sequences

Use names to identify specific sections

But how do you name them, and how do you avoid conflicts?

<table>
<thead>
<tr>
<th>N1</th>
<th>N2</th>
<th>N3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N1</th>
<th>N4</th>
<th>N5</th>
<th>N2</th>
<th>N3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>13</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>
Giraz - Incremental Sequence

Sequence  •  •  •  •  •  •  •  •  •  •
Giraz - Incremental Sequence

![Diagram of incremental sequence with nodes N1, N2, and N3.]
Giraz - Incremental Sequence

Named branches
Primitive arrays
Edit as zipper
Canonical structure
Mimics Adapton structure
Giraz - Incremental Sequence

Names

Sequence

Named branches

Primitive arrays

Edit as zipper

Canonical structure

Mimics Adapton structure
Giraz - Incremental Sequence

Names

Sequence

Interface
- fold_lr
- fold_up
- map

Named branches

Primitive arrays

Edit as zipper

Canonical structure

Mimics Adapton structure
## Giraz - Incremental Sequence

<table>
<thead>
<tr>
<th></th>
<th>speedup over native Rust</th>
<th>Native initial run</th>
<th>Incremental initial run</th>
<th>Incremental update</th>
<th>crossover</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_string</td>
<td>449.0</td>
<td>93.80</td>
<td>95.50</td>
<td>0.21</td>
<td>1</td>
</tr>
<tr>
<td>reverse</td>
<td>22.2</td>
<td>2.01</td>
<td>7.85</td>
<td>0.09</td>
<td>4</td>
</tr>
<tr>
<td>max</td>
<td>57.5</td>
<td>2.84</td>
<td>5.99</td>
<td>0.05</td>
<td>2</td>
</tr>
<tr>
<td>adder</td>
<td>23.9</td>
<td>10.30</td>
<td>91.10</td>
<td>0.43</td>
<td>10</td>
</tr>
<tr>
<td>quickhull</td>
<td>4.2</td>
<td>56.60</td>
<td>213.00</td>
<td>13.50</td>
<td>6</td>
</tr>
</tbody>
</table>

All inputs: 1M, gauges 1k, times in ms
Future work - IODyn

Design a language with implicit incremental features

Additional incremental data structures
  • Tries
  • Graphs

Annotations for expected incremental properties

Type system to enforce proper name usage

Meta-proofs about consistency with from-scratch runs
Significant tools
• data structures
• caching

Adapton dependency graphs

Giraz incremental sequence

Future:
Provide access to incremental features usable in general purpose code