Microsoft Research

Each year Microsoft Research hosts hundreds of influential speakers from around the world including leading scientists, renowned experts in technology, book authors, and leading academics, and makes videos of these lectures freely available.

2016 © Microsoft Corporation. All rights reserved.
#if ABSOLUTE
    position = new;
#else RELATIVE
    position += new;
#endif

Why don't we have both?
Applications of variational programming

Eric Walkingshaw
Oregon State University
Original motivation:
highly configurable software systems
Original motivation: highly configurable software systems
Original motivation: highly configurable software systems
Original motivation: highly configurable software systems
Original motivation: highly configurable software systems

1. configure

2. validate

3. deploy

program analysis
Original motivation:
highly configurable software systems
Original motivation: highly configurable software systems

1. configure

2. validate

3. deploy

program analysis

This process finds errors too late!

... but way too many configs to check them all
Solution: variational analyses
Solution: variational analyses
Solution: variational analyses

class buffer {
    int buff = 0;
    int get() {
        return buff;
    }
    void set(int x) {
        buff = x;
    }
}

program analysis

program

variational analysis

variational analysis
Solution: variational analyses

Our work:
- languages
- theory
- data structures
- algorithms
to do this correctly and efficiently
Solution: variational analyses

Our work:
- languages
- theory
- data structures
- algorithms
to do this correctly and efficiently

“variational programming”
Example: variational typing

lambda calculus -> Hindley-Milner -> types

variational lambda calculus

variational typing -> variational types
Example: variational typing

- Lambda calculus → Hindley-Milner → types
- Variational lambda calculus → variational typing → variational types
- Variational programming inside!
Variational programming by example
Variational programming by example

\[ A(2,3) + 4 \]
Variational programming by example

\[ A\{2,3\} + 4 \]
\[ \Rightarrow A\{6,7\} \]
Variational programming by example

\[ A(2,3) + 4 \]
\[ \Rightarrow A(6,7) \]

\[ A(2,3) + A(10,20) \]
Variational programming by example

\[ A(2,3) + 4 \Rightarrow A(6,7) \]

\[ A(2,3) + A(10,20) \Rightarrow A(12,23) \]
Variational programming by example

\[
A\{2,3\} + 4 \\
\Rightarrow A\{6,7\}
\]

\[
A\{2,3\} + A\{10,20\} \\
\Rightarrow A\{12,23\}
\]

\[
A\{2,3\} + B\{10,20\}
\]
Variational programming by example

\[ A\{2,3\} + 4 \Rightarrow A\{6,7\} \]

\[ A\{2,3\} + A\{10,20\} \Rightarrow A\{12,23\} \]

\[ A\{2,3\} + B\{10,20\} \Rightarrow A\{B\{12,22\},B\{13,23\}\} \]
Variational programming by example

\[ A(2,3) + 4 \rightarrow A(6,7) \]

\[ A(2,3) + A(10,20) \rightarrow A(12,23) \]

\[ A(2,3) + B(10,20) \rightarrow A(\{12,22\}, B(13,23)) \]
Variational programming by example

\[ A\{2,3\} + 4 \rightarrow A\{6,7\} \]

\[ A\{2,3\} + A\{10,20\} \rightarrow A\{12,23\} \]

\[ A\{2,3\} + B\{10,20\} \rightarrow A\{B\{12,22\},B\{13,23\}\} \]
Variational programming by example

\[ A(2,3) + 4 \Rightarrow A(6,7) \]
\[ A(2,3) + A(10,20) \Rightarrow A(12,23) \]
\[ A(2,3) + B(10,20) \Rightarrow A(B(12,22), B(13,23)) \]
Variational programming by example

\[ A(2,3) + 4 \Rightarrow A(6,7) \]
\[ A(2,3) + A(10,20) \Rightarrow A(12,23) \]
\[ A(2,3) + B(10,20) \Rightarrow A(B(12,22), B(13,23)) \]
Variational programming by example

\[ A\{2,3\} + 4 \rightarrow A\{6,7\} \]

\[ A\{2,3\} + A\{10,20\} \rightarrow A\{12,23\} \]

\[ A\{2,3\} + B\{10,20\} \rightarrow A\{B\{12,22\},B\{13,23\}\} \]

\[ A(\text{true},3) : A(\text{Bool},\text{Int}) \]

\[ A(\text{succ},\text{even}) : \text{Int} \rightarrow A(\text{Int},\text{Bool}) \equiv A(\text{Int} \rightarrow \text{Int},\text{Int} \rightarrow \text{Bool}) \]
Variational programming by example

A(not, even) A(True, 3)
Variational programming by example

\[ A(\text{not, even}) \ A(\text{True, 3}) \]

\[ \mapsto \text{False} \]
Variational programming by example

\[ A(\text{not, even}) \ A(\text{True, 3}) \]

⇒ False

≡ \[ A(\text{False, False}) \]
Variational programming by example

\[ A(\text{not,even}) \ A(\text{True,3}) \]

\[ \Rightarrow \text{False} \]

\[ \equiv A(\text{False,False}) \]

\[ A(\text{succ,even}) \ B(2,4) \]
Variational programming by example

\[
A(\text{not}, \text{even}) \ A(\text{True}, 3) \\
\Rightarrow \text{False} \\
\equiv A(\text{False}, \text{False})
\]

\[
A(\text{succ}, \text{even}) \ B(2, 4) \\
\Rightarrow A(B(3, 5), \text{True})
\]
Variational programming by example

\[ A(\text{not, even}) \ A(\text{True, 3}) \quad \Rightarrow \quad \text{False} \]
\[ \equiv \ A(\text{False, False}) \]

\[ A(\text{succ, even}) \ B(2, 4) \quad \Rightarrow \quad A(B(3, 5), \text{True}) \]

\[ \text{vsum} \ (A(2, 3) + B(10, 20)) \]
Variational programming by example

\[ A(\text{not, even}) A(\text{True, 3}) \Rightarrow \text{False} \]
\[ \equiv A(\text{False, False}) \]

\[ vsum (A(2, 3) + B(10, 20)) \Rightarrow 70 \]

\[ A(\text{succ, even}) B(2, 4) \Rightarrow A(B(3, 5), \text{True}) \]
Variational programming by example

\[
A(\text{not, even}) \ A(\text{True, 3}) \quad \Rightarrow \quad \text{False} \\
\equiv \ A(\text{False, False})
\]

\[
A(\text{succ, even}) \ B(2, 4) \quad \Rightarrow \quad A(B(3, 5), \text{True})
\]

\[
v\text{sum} \ (A(2, 3) + B(10, 20)) \quad \Rightarrow \quad 70
\]

\[
v\text{max} \ (A(2, 3) + B(10, 20))
\]
Variational programming by example

A(not,even) A(True,3)
→ False
≡ A(False,False)

vsum (A(2,3) + B{10,20})
→ 70

A(succ,even) B{2,4}
→ A(B{3,5},True)

vmax (A(2,3) + B{10,20})
→ 23 @ [A.R,B.R]
Challenge #1
exponential variation space
Challenge #1
exponential variation space

Share as much as possible
- split late and join early
- clever data structures
Challenge #1
exponential variation space

Share as much as possible
• split late and join early
• clever data structures

Pick the right domains
Challenge #2

the variation... it gets... everywhere
Challenge #2

the variation... it gets... everywhere

Variation in structured data
Challenge #2

the variation... it gets... everywhere

Variation in structured data

A\{[1,2,3], [1,2,4,5]\}
Challenge #2

the variation... it gets... everywhere

Variation in structured data

\[ A[[1,2,3],[1,2,4,5]] \equiv [1,2,A(3,4),A(5,\_)] \]
Challenge #2

the variation... it gets... everywhere

Variation in structured data

A([[1,2,3],[1,2,4,5]])
≡ [1,2,A(3,4),A(5,_)_]

... need variational data structures
Challenge #2

the variation... it gets... everywhere

Variation in structured data

\[ A\{[1,2,3],[1,2,4,5]\} \equiv [1,2,A\{3,4\},A\{5,_\}] \]

... need variational data structures

How to handle side effects?
Variational typing

lambda calculus → Hindley-Milner → types

lambda calculus + choices → variational typing → types + choices
Variational typing

lambda calculus → Hindley-Milner → types

lambda calculus + choices → variational typing → types + choices

Lots of sharing
Variational typing

Hindley-Milner

Lots of sharing
Scales roughly linearly w/ size of program

Lambda calculus

Lambda calculus + choices

Variational typing

Types

Types + choices
Variational typing

lambda calculus → Hindley-Milner → types

Lots of sharing

Scales roughly linearly w/ size of program

lambda calculus + choices → variational typing → types + choices

Scales to Linux kernel (Kästner et al.)
Only good for product line analyses?
Only good for product line analyses?

What makes a good nail?
- lots of variation
- lots of sharing
Only good for product line analyses?

What makes a good nail?
- lots of variation
- lots of sharing

Idea: use variation to explore hypothetical scenarios ...

Why don’t we have both?
Error location

fold f z [] = [z]
fold f z (h:t) = fold f (f z h) t

flip f x y = f y x

reverse = fold (flip (:)) []
palindrome xs = reverse xs == xs
Error location

fold f z [] = [z]
fold f z (h:t) = fold f (f z h) t

flip f x y = f y x

reverse = fold (flip (:)) []

palindrome xs = reverse xs == xs

- Occurs check: cannot construct the infinite type: a ~ [a]
  Expected type: [[a]]
  Actual type: [a]
- In the second argument of `(==)`, namely `xs`
  In the expression: reverse xs == xs
Error location

fold f z [] = [z]
fold f z (h:t) = fold f (f z h) t
flip f x y = f y x
reverse = fold (flip (:)) []
palindrome xs = reverse xs == xs

error is in base case of fold
... but fold type checks!

use of fold also type checks!

error finally detected

- Occurs check: cannot construct the infinite type: a ~ [a]
  Expected type: [[a]]
  Actual type: [a]
- In the second argument of ‘(==)’, namely ‘xs’
  In the expression: reverse xs == xs
Counterfactual typing

Problem: locating the cause of a type error is hard
  - type inference commits too early
  - a successfully inferred type could be wrong!
Error location

fold f z [] = [z]
fold f z (h:t) = fold f (f z h) t
flip f x y = f y x
reverse = fold (flip (:)) []
palindrome xs = reverse xs == xs

error is in base case of fold
... but fold type checks!

use of fold also type checks!
error finally detected

- Occurs check: cannot construct the infinite type: a ~ [a]
  Expected type: [[a]]
  Actual type: [a]
- In the second argument of `(==)`, namely `xs`
  In the expression: `reverse xs == xs`
Counterfactual typing

**Problem:** locating the cause of a type error is hard

- type inference commits too early
- a successfully inferred type could be wrong!
Counterfactual typing

**Problem:** locating the cause of a type error is hard
- type inference commits too early
- a successfully inferred type could be wrong!

**Solution:**
Counterfactual typing

Problem: locating the cause of a type error is hard
  - type inference commits too early
  - a successfully inferred type could be wrong!

Solution:

1. Error-tolerant variational type inference
   where for every subexpression

   \[ e : T \rightarrow e : d(T, a) \quad d \& a \text{ are fresh} \]
Counterfactual typing

Problem: locating the cause of a type error is hard
- type inference commits too early
- a successfully inferred type could be wrong!

Solution:

1. Error-tolerant variational type inference where for every subexpression
   \[ e : T \rightarrow e : d(T, a) \quad d \& a \text{ are fresh} \]

2. Search output variational type for
   - non-error type
   - as few right selections as possible
Migrating gradual types

```python
def f(mode: bool, x):
    if mode:
        return even(x)
    else:
        return not(x)
```

Gradual typing
mix static and dynamic types in the same program
Migrating gradual types

```python
def f(mode: bool, x):
    if mode:
        return even(x)
    else:
        return not(x)
```

Gradual typing
mix static and dynamic types
in the same program

Migration challenges:

- (adding/removing annotations)

- mutually exclusive annotations
Migrating gradual types

```python
def f(mode: bool, x):
    if mode:
        return even(x)
    else:
        return not(x)
```

Gradual typing
mix static and dynamic types
in the same program

Migration challenges:

- mutually exclusive annotations
- local type-safety maxima
- potential for extreme performance degradation

(adding/removing annotations)
Migrational typing

**Problem:** migrating gradual types is perilous and exploration by trial-and-error is infeasible

**Solution:**

1. Every parameter is initially $d(a, ?)$
   - $d$ & $a$ are fresh
   - $?$ is the dynamic type

2. Variational gradual type inference + cost analysis
   - output = summary of all possible migrations

3. Filter/search variational output
   - most static = fewest right selections
   - cheapest = lowest cost
Variational programming take aways

Key ideas
- make variation *explicit* in programs and data, then compute with it!
- share as much as possible between variants

Can efficiently explore lots of hypothetical scenarios
Chapel Comes of Age: Productive Parallelism at Scale

Brad Chamberlain, Chapel Team, Cray Inc.
PNW PLSE Workshop
May 14, 2018
Chapel: Niche or Quiche?

Brad Chamberlain, Chapel Team, Cray Inc.
PNW PLSE Workshop
May 14, 2018
What is Chapel?

Chapel: A productive parallel programming language

- portable & scalable
- open-source & collaborative
What is Chapel?

Chapel: A productive parallel programming language
- portable & scalable
- open-source & collaborative

Goals:
- Support general parallel programming
  - “any parallel algorithm on any parallel hardware”
- Make parallel programming at scale far more productive
Chapel and Productivity

Chapel aims to be as...

...programmable as Python
...fast as Fortran
...scalable as MPI
...portable as C
...flexible as C++
...fun as [your favorite programming language]
CLBG Cross-Language Summary
(Oct 2017 standings, zoomed in)

- Execution Time (normalized to fastest entry)
- Compressed Code Size (normalized to smallest entry)

Languages analyzed:
- C
- C++
- C#
- Clojure
- Rust
- Python
- Java
- Scala
- OCaml
- F#
- Swift
- Go
- Haskell
- Typescript
- Javascript
- and more
CLBG Cross-Language Summary
(Oct 2017 standings, zoomed in)

Execution Time (normalized to fastest entry)

Compressed Code Size (normalized to smallest entry)

Languages:
- TypeScript
- Javascript
- OCaml
- Haskell
- F#
- Scala
- Swift
- Pascal
- Java
- C#
- Go
- Chapel
- Rust
- Fortran
- C++
- C

CRAY

Copyright 2018 Cray Inc.
CLBG Cross-Language Summary (Oct 2017 standings)

- Execution Time (normalized to fastest entry)
- Compressed Code Size (normalized to smallest entry)

Languages and their respective positions on the graph:
- Chapel
- CSharpCore
- Dart
- Erlang
- F# (F
core)
- FSharp
- Go
- Groovy
- Hack
- Java
- JRuby
- Lua
- Node
- OCaml
- Python
- Perl
- PHP
- Rust
- Scala
- Swift
- TypeScript
- VB
- YARV
- Mean-smallest
- Mean-fastest

Key: smaller, faster
CLBG: Qualitative Code Comparisons

Can also browse program source code (but this requires actual thought!):

```chapel
proc main()
{
    printColorEquations();
    const group1 = [i in 1..popSize] new Chamonees(i, ([i-1]%4)+Color);
    const group2 = [i in 1..popSize2] new Chamonees(i, colors[i%4]);
    coalition {
        holdMeetings(group1, n);
        holdMeetings(group2, n);
    }
    print(group1);
    print(group2);
    for c in group1 do delete c;
    for c in group2 do delete c;
}
// Print the results of getNewColor() for all color pairs.
//
proc printColorEquations()
{
    for c1 in Color do
        for c2 in Color do
            writeln(c1, " ", c2, " -> ", getNewColor(c1, c2));
}
// Hold meetings among the population by creating a shared meeting
// place, and then creating per-chamonees tasks to have meetings.
//
proc holdMeetings(population, numMeetings)
{
    const place = new MeetingPlace(numMeetings);
    foreach c in population do
        c.havemeetings(place, population);
    delete place;
}
```

```c
void get_affinity(int ia_mp, cpu_set_t* affinity1, cpu_set_t* affinity2)
{
    cpu_set_t active_cpus;
    FILE* f;
    char buf [2048];
    char const* pos;
    int cpu_id;
    int physical_id;
    int core_id;
    int cpu_cores;
    int apic_id;
    size_t cpu_count;
    size_t i;

    char const* processor_str = "processor";
    size_t processor_str_len = strlen(processor_str);
    char const* physical_id_str = "physical_id";
    size_t physical_id_str_len = strlen(physical_id_str);
    char const* core_id_str = "core_id";
    size_t core_id_str_len = strlen(core_id_str);
    char const* cpu_cores_str = "cpu_cores";
    size_t cpu_cores_str_len = strlen(cpu_cores_str);

    CPU_ZERO(&active_cpus);
    sched_getaffinity(0, sizeof(active_cpus), &active_cpus);
    cpu_count = 0;
    for (i = 0; i < CPU_SETSIZE; i++)
    {
        if (CPU_ISSET(i, &active_cpus))
        {
            cpu_count += 1;
        }
    }
    if (cpu_count == 1)
    {
        ia_mp[0] = 0;
        return;
    }
    ia_mp[0] = 1;
    CPU_ZERO(affinity1);
}
```
CLBG: Qualitative Code Comparisons

Can also browse program source code (but this requires actual thought!):

```c
cobegin {
    holdMeetings(group1, n);
    holdMeetings(group2, n);
}
```

```c
proc holdMeetings(population, numMeetings) {
    const place = new MeetingPlace(numMeetings);
    coforall c in population do // create
        c.haveMeetings(place, population);
    delete place;
}
```
Can also browse program source code *(but this requires actual thought!)*:

```c
void get_affinity(int* is_smp, cpu_set_t* affinity1, cpu_set_t* affinity2) {
  cpu_set_t active_cpus;
  FILE* f;
  char buf [2048];
  char const* core_id_str = "core id";
  size_t core_id_str_len = strlen(core_id_str);
  char const* cpu_cores_str = "cpu cores";
  size_t cpu_cores_str_len = strlen(cpu_cores_str);
  CPU_ZERO(&active_cpus);
  sched_getaffinity(0, sizeof(active_cpus), &active_cpus);
  cpu_count = 0;
  for (i = 0; i != CPU_SETSIZE; i += 1) {
    if (CPU_ISSET(i, &active_cpus)) {
      cpu_count += 1;
    }
  }
  if (cpu_count == 1) {
    is_smp[0] = 0;
    return;
  }
}
```

Excerpt from 1210 gz Chapel entry

```c
excerpt from 2863 gz C gcc entry
```

```
```
Excerpt from PNW PLSE Review

“Chapel has been around for quite a while, and it still seems like a niche language…”
Chapel: “A Niche Language”?

Chapel is arguably niche in that it...

...was originally designed for HPC
Chapel: “A Niche Language”?

Chapel is arguably niche in that it...
...was originally designed for HPC

Yet, Chapel’s chief concerns aren’t HPC-specific:
- performance
Chapel: “A Niche Language”?

Chapel is arguably niche in that it...
...was originally designed for HPC

Yet, Chapel’s chief concerns aren’t HPC-specific:
- performance
- programmability (cf. Python)
Chapel: “A Niche Language”?

Chapel is arguably niche in that it...

...was originally designed for HPC

Yet, Chapel’s chief concerns aren’t HPC-specific:

- performance
- programmability (cf. Python)
- parallelism (cf. multicore)
Chapel: “A Niche Language”?  

Chapel is arguably niche in that it...  
...was originally designed for HPC

Yet, Chapel’s chief concerns aren’t HPC-specific:  
- performance  
- programmability (cf. Python)  
- parallelism (cf. multicore)  
- distributed memory (cf. cloud computing)
Chapel: “A Niche Language”?

Chapel is arguably niche in that it...
...was originally designed for HPC
...has only a modest-sized community (so far)
Chapel: “A Niche Language”?

Chapel is arguably niche in that it...
...was originally designed for HPC
...has only a modest-sized community (so far)

Yet, we’ve historically discouraged its use in production...
Chapel: A Quiche Language!

The outsider’s impression:

Why aren’t more people using this delectable language?
Chapel: A Quiche Language!

The outsider’s impression:

Why aren’t more people using this delectable language?
Chapel: A Quiche Language!

The outsider’s impression:

Why aren’t more people using this delectable language?

The reality, for most of Chapel’s history:
Chapel: A Quiche Language!

The outsider’s impression: Why aren’t more people using this delectable language?

Though recently, it’s more like:
Chapel: “Been Around for Quite Awhile”

Chapel’s Infancy: DARPA HPCS (2003–2012)

- Research focus: ~6-7 FTEs
  - distinguish locality from parallelism
  - seamlessly mix data- and task-parallelism
  - support user-defined distributed arrays, parallel iterators
Chapel: “Been Around for Quite Awhile”

Chapel’s Infancy: DARPA HPCS (2003–2012)
- Research focus: ~6-7 FTEs
  - distinguish locality from parallelism
  - seamlessly mix data- and task-parallelism
  - support user-defined distributed arrays, parallel iterators

Chapel’s Adolescence: “the five-year push” (2013–2018)
- Development focus: ~13-14 FTEs
  - performance and scalability
  - ecosystem: documentation, libraries, tools, …
  - base language fixes: OOP features, error-handling, strings, …
Chapel: “Been Around for Quite Awhile”

Chapel’s Infancy: DARPA HPCS (2003–2012)
- Research focus: ~6-7 FTEs
  - distinguish locality from parallelism
  - seamlessly mix data- and task-parallelism
  - support user-defined distributed arrays, parallel iterators

Chapel’s Adolescence: “the five-year push” (2013–2018)
- Development focus: ~13-14 FTEs
  - performance and scalability
  - ecosystem: documentation, libraries, tools, …
  - base language fixes: OOP features, error-handling, strings, …
Chapel Ecosystem: Then vs. Now
Documentation: Then

After HPCS:
- a PDF language specification
Documentation: Then

After HPCS:
- a PDF language specification
- a Quick Reference sheet
Documentation: Then

After HPCS:
- a PDF language specification
- a Quick Reference sheet
- a number of READMEs
Documentation: Then

After HPCS:
- a PDF language specification
- a Quick Reference sheet
- a number of READMEs
- ~22 primer examples
Documentation: Now

Now: 200+ modern, hyperlinked, web-based documentation pages
Libraries: Then

After HPCS: ~25 library modules

- documented via source comments, if at all:
Libraries: Now

Now: ~60 library modules

- web-documented, many user-contributed
Tools: Then

After HPCS:

- highlighting modes for emacs and vim
- chpldoc: documentation tool (rough draft)
Now:

- highlighting modes for emacs, vim, atom, ...
- chpldoc: documentation tool
- mason: package manager
- c2chapel: interoperability aid
- chpltags: helps search Chapel code
- bash tab completion: command-line help
- chplvis: performance visualizer / debugger
Chapel Performance: Then vs. Now
Performance Focus Areas during 5-year push

- Cleaner, simpler generated code
- NUMA sensitivity within multi-socket nodes
- Best-use of RDMA and NIC memory registration
- Reduced overheads in tasks, memory, communication
- Bulk transfer optimizations
- ...and much more...
STREAM Triad Performance: Chapel Then

STREAM Performance (GB/s)

Locales (x 36 cores / locale)

Chapel 1.7

better
STREAM Triad Performance: Chapel Then vs. Now

STREAM Performance (GB/s)

- Chapel 1.17
- Chapel 1.7

Locales (x 36 cores / locale)

GB/s

0 500 1000 1500 2000 2500 3000

Copyright 2018 Cray Inc.
STREAM Triad Performance: Chapel Now vs. ref

STREAM Performance (GB/s)

- Reference
- Chapel 1.17

Locales (x 36 cores / locale)

GB/s

0 5000 10000 15000 20000 25000 30000

better
PRK Stencil Performance: Then

PRK Stencil Performance (GFLOPs/s)

Chapel 1.7

Locales (x 36 cores / locale)

GFLOPs/s

better
PRK Stencil Performance: Chapel Now vs. ref

PRK Stencil Performance (GFLOPs/s)

Locales (x 36 cores / locale)
HPC Patterns: Chapel Now vs. reference

LCALS: Chapel 1.17 vs. reference

HPCC RA

RA Performance: Chapel 1.17 vs. reference

STREAM Triad

PRK Stencil

PRK Stencil Performance: Chapel 1.17 vs. reference

Nightly performance tickers online at: https://chapel-lang.org/perf-nightly.html
HPC Patterns: Chapel Now vs. reference

- Local loop kernels
- Global Random Updates
- Embarrassing/Pleasing Parallelism
- Bucket-Exchange Pattern
- Stencil Boundary Exchanges

Nightly performance tickers online at: https://chapel-lang.org/perf-nightly.html
HPCC Random Access Kernel: MPI

```c
/* Performed updates to main table. The scalar equivalent is: */

/* for (i++) if (Ran < 0) */
/* Ran = (Ran + 1) % (i + 1); */
/* Table[Ran] & (TABSIZE - 1) = Ran; */

MPI_Irecv(localsendbuffer, localsendBufferSize, tparams, dtype64, MPI_ANY_SRC, MPI_ANY_TAG, MPI_COMM_WORLD, &recv);
while (!recv) {
    /* receive messages */
    do {
        MPI_Test(&isrecv, &have_done, &status);
        if (have_done) {
            if (status.MPI_TAG == UPDATE_TAG) {
                MPI_Get_count(&isrecv, &recvUpdates);
                bufferSize = 0;
                for (j = 0; j < recvUpdates; j++) {
                    immsg = LocalRecBuffer[bufferSize];
                    localOffset = (immsg & (tparams->TableSize - 1)) - tparams.GlobalStartMyProc;
                    if (recvUpdates == 0) {
                        immsg = 0;
                    } else {
                        immsg = LocalRecBuffer[bufferSize];
                        localOffset = (immsg & (tparams->TableSize - 1)) - tparams.GlobalStartMyProc;
                    }
                    MPI_Table[localOffset] = immsg;
                } else if (status.MPI_TAG == FINISHED_TAG) {
                    NumberReceiving = 0;
                } else {
                    MPI_Abort(MPI_COMM_WORLD, -1);
                }
            } while (have_done && NumberReceiving > 0);
            if (pendingUpdates > 0) {
                if (Ran = (Ran + 1) % (164Int + 1); MPI_GlobalOffset = Ran & (tparams->TableSize - 1);) {
                    NumberReceiving = 0;
                    if (GlobalOffset < tparams->Top) {
                        while (GlobalOffset < tparams->Top) {
                            if (GlobalOffset < tparams->Top) {
                                NumberReceiving = 0;
                            } else {
                                NumberReceiving = 0;
                            }
                        }
                    } else {
                        NumberReceiving = 0;
                    }
                } else if (status.MPI_TAG == FINISHED_TAG) {
                    /* we got a done message. Thanks for playing... */
                } else if (MPI_Abort(MPI_COMM_WORLD, -1)) {
                    MPI_Irecv(localsendbuffer, localsendBufferSize, tparams, dtype64, MPI_ANY_SRC, MPI_ANY_TAG, MPI_COMM_WORLD, &recv);
                } while (have_done && NumberReceiving > 0);
            }
        } else {
            MPI_Irecv(localsendbuffer, localsendBufferSize, tparams, dtype64, MPI_ANY_SRC, MPI_ANY_TAG, MPI_COMM_WORLD, &recv);
        } else {
            MPI_Test(&isrecv, &have_done, &status);
            if (have_done) {
                if (status.MPI_TAG == UPDATE_TAG) {
                    MPI_Get_count(&isrecv, &recvUpdates);
                    bufferSize = 0;
                    for (j = 0; j < recvUpdates; j++) {
                        immsg = LocalRecBuffer[bufferSize];
                        localOffset = (immsg & (tparams->TableSize - 1)) - tparams.GlobalStartMyProc;
                        MPI_Table[localOffset] = immsg;
                    } else if (status.MPI_TAG == FINISHED_TAG) {
                        pendingUpdates = 0;
                    } else {
                        MPI_Abort(MPI_COMM_WORLD, -1);
                    }
                } else if (recvUpdates == 0) {
                    immsg = 0;
                } else {
                    immsg = LocalRecBuffer[bufferSize];
                    localOffset = (immsg & (tparams->TableSize - 1)) - tparams.GlobalStartMyProc;
                    MPI_Table[localOffset] = immsg;
                } else if (status.MPI_TAG == FINISHED_TAG) {
                    pendingUpdates = 0;
                } else if (MPI_Abort(MPI_COMM_WORLD, -1)) {
                    MPI_Irecv(localsendbuffer, localsendBufferSize, tparams, dtype64, MPI_ANY_SRC, MPI_ANY_TAG, MPI_COMM_WORLD, &recv);
                } while (have_done && NumberReceiving > 0);
            }
        }
    }
}
```
HPCC Random Access Kernel: MPI

/* Perform updates to main table. The scalar equivalent is:

for (HG = WURDATE; HG != NULL; HG = HG->next)
    Ran = (Ran << 1) + ((HG->Ran < 9) ? POLY : 0);
    Table[Ran & (TABLESIZE - 1)] = HG->Ran;
*/

/* 4-probe message */

} else if (init_targ == STRAND_REQ)
    HCC_targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);

} else if (init_targ == STRING_VAR)
    HCC_Targ精湛er(strand, request, response);
/* Perform updates to main table. The scalar equivalent is:
 */

for (i=0; i<NUPDATE; i++) {
    Ran = (Ran << 1) ^ (((s64Int) Ran < 0) ? POLY : 0);
    Table[Ran & (TABSIZE-1)] ^= Ran;
}
HPCC Random Access Kernel: MPI

Chapel Kernel

```chapel
forall (_, r) in zip(Updates, RAStream()) do
    T[r & indexMask] ^= r;
```

MPI Comment

```c
/* Perform updates to main table. The scalar equivalent is:
   */
   for (i=0; i<NUPDATE; i++) {
       Ran = (Ran << 1) ^ ((s64Int) Ran < 0) ? POLY : 0);
       Table[Ran & (TABSIZE-1)] ^= Ran;
   }
   */
```
RA Performance: Chapel Now vs. reference

RA Performance (GUPS)

- Reference
- Chapel 1.17

Locales (x 36 cores / locale)

0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8

GUPS

16 32 64 128 256

better
What’s Next?

**CHIUW 2018**: The 5th annual Chapel Implementers and Users Workshop

- Vancouver BC, Friday May 25th
**What’s Next?**

**CHIUW 2018:** The 5th annual Chapel Implementers and Users Workshop
- Vancouver BC, Friday May 25th

**Chapel’s college years:** plans for 2018-2021
- Further Performance and Scalability Improvements
- Libfabric/OFI Support
- GPU Support
- Cloud Support
- Chapel AI
The Chapel Team at Cray (May 2018)

13 full-time employees + ~2 summer interns
Chapel Community Partners

Haverford College  AMD  The George Washington University
Washington, DC  Western Washington University

東京大学  The University of Arizona  Rice University

Berkeley Lab  Lawrence Livermore National Laboratory  Sandia National Laboratories

Department of Defense  United States of America  Yale

(and several others...)

https://chapel-lang.org/collaborations.html
Summary

- Chapel’s made huge progress over the past five years
- Ready for use in production*
- Open to collaborations
  - Plenty of research questions remain
Musical Ornaments

John Leo

Halfaya Research

May 14, 2018
Outline

- A look toward the future
- Music Tools
- Equivalences and Ornaments

Sources:
- https://github.com/halfaya/MusicTools
Eventually all the arbitrary programming languages are going to be just swept away with the oceans, and we will have the permanence of constructive, intuitionistic type theory as the master theory of computation—without doubt, in my mind, no question. So, from my point of view—this is a personal statement—working in anything else is a waste of time.

CMU Homotopy Type Theory lecture 1, 52:56–53:20.
What will programming look like in 50 years?

- Convergence of math and computer science
- Functional Programming, Algebra of Programming
- Dependent Types or a successor (Cubical?)
- Who does the programming?
How do we get there from here?

- Add dependent types to an industrial-strength language (Haskell)
- Make a dependently typed language (Agda, Idris) practical to use
- Learn how to program using dependent types
- Many theoretical and practical advances are still needed
Euterpea

The Haskell School of Music

— From Signals to Symphonies —

Paul Hudak

Yale University
Department of Computer Science
Music Tools

- Collection of composable tools for synthesis and analysis of music
- Originally written in Haskell
- Converted to Agda, using Haskell for MIDI interface
- Explore programming using dependent types in a circumscribed yet rich domain
- Use math, including transport of equivalences (from HoTT) and Ornaments
Look vs Time (1997)
Look vs Time (1997)
open import MidiEvent
open import Util

tempo : N
tempo = 84

melodyChannel : Channel-1
melodyChannel = # 0

melodyInstrument : InstrumentNumber-1
melodyInstrument = # 8 -- celesta

melodyNotes : List Note
melodyNotes =
  note (8th 3) (c 3) ::
  note (8th 5) (d 3) ::
  note (8th 3) (c 3) ::
  note (8th 5) (d 3) ::
  note (8th 1) (g 3) ::
  note (8th 1) (f 3) ::
  note (8th 1) (e 3) ::
  note (8th 5) (d 3) ::
  note (8th 1) (g 3) ::
  note (8th 1) (f 3) ::
  note (8th 1) (e 3) ::
  note (8th 5) (d 3) ::
  note (8th 1) (a 3) ::
Music Representation à la Euterpea

```
data Pitch : Set where
  pitch : ℤ → Pitch

data Duration : Set where
  duration : ℕ → Duration

data Note : Set where
  note : Duration → Pitch → Note
  rest : Duration → Note

data Music : Set where
  note : Note → Music
  _∷_: : Music → Music → Music -- sequential
  _∥∥_: : Music → Music → Music -- parallel
```
Equivalent Representations of Pitch

```haskell
data Pitch : Set where
  pitch : ℤ → Pitch

chromaticScaleSize : ℕ
chromaticScaleSize = 12

data RelativePitch : Set where
  relativePitch : Fin chromaticScaleSize → RelativePitch

data Octave : Set where
  octave : ℤ → Octave

PitchOctave : Set
PitchOctave = RelativePitch × Octave
```
Equivalences

- Define an equivalence between \texttt{Pitch} and \texttt{PitchOctave}
- Using HoTT techniques, automatically lift this equivalence to functions defined using \texttt{Pitch}
- See \textit{Equivalences for Free!} (Tabareau, Tanter, Sozeau)
- Challenge: Defining base equivalences. Can this be automated?
Ornaments

data Music a = ... 
  | Modify Control (Music a)

data Control = ... 
  | Phrase [PhraseAttribute]

data PhraseAttribute = ... 
  | Orn Ornament

data Ornament =
  Trill | Mordent | InvMordent | DoubleMordent | Turn | TrilledTurn | ShortTrill ...

Functions on a base Music structure can be automatically lifted to operate on Music ornamented with additional information.

See works on Ornaments by McBride, Dagand and others.

Challenge: Shallow embedding of ornaments in Agda.
Conclusion

- Music is a good domain in which to explore practical application of dependent types
- Using math can be more work at first, but should be a big win in the long term
- Figure out how to minimize the work and maximize the reward