Modal Types for Mobile Code

(Thesis Proposal)

Tom Murphy VII

Thesis Statement

Modal type systems provide an elegant and practical way to control spatially distributed resources in mobile programs.

Plan for Thesis Project

Design a new programming language, ML5

· · · based on a modal type system

Implement it

· · · a certifying compiler and runtime

Build an application in the language

· · · to demonstrate its practicality, effectiveness

Plan for this Talk

Show the problem with local resources

· · · in the context of the ConCert project

Sketch the tool I'll use, modal logic

· · · a logic for reasoning about place

Present the modally-typed ML5

· · · some typing rules, some examples

The Proof is in the Proposal

The proposal document contains all of the detail.

http://tom7.org/proposal/

Distributed Computing in ConCert

Trustless Grid computing

(Chang *et al.* 2002)

Distributed Computing in ConCert

Trustless Grid computing

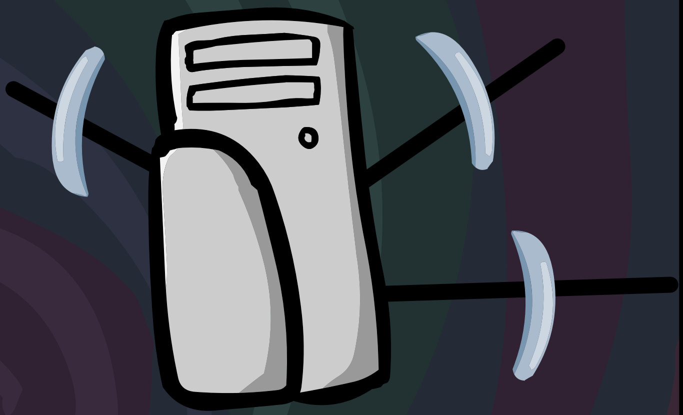
Distributed Computing in ConCert

Trustless Grid computing

Data and code
move around
between hosts.

Distributed Computing in ConCert

Trustless Grid computing

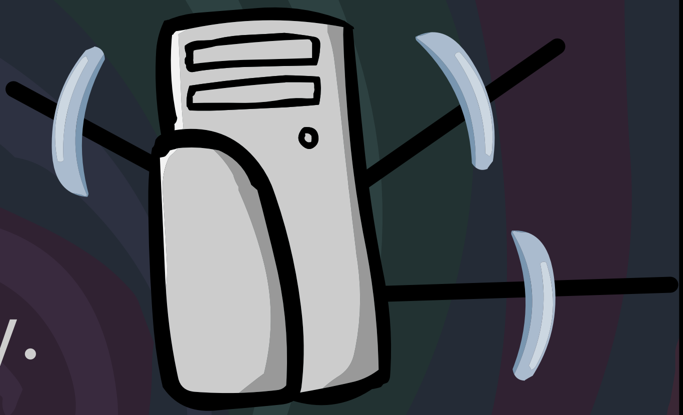


Distributed Computing in ConCert

Trustless Grid computing

Hosts needn't trust one another.

They verify that code and data are "safe" according to some policy.



Problems with ConCert

Only allows sharing of CPU resources

· · · because hosts are assumed to be uniform

This excludes many distributed apps

· · · many rely on localized resources, like sensing instruments, storage arrays, etc.

Problems with ConCert

Programming with resources is tricky.

· · · let's look at some example pitfalls.

Grid/ML

Grid/ML is our ML-like language for writing ConCert applications.

Grid/ML

Grid/ML is our ML-like language for writing ConCert applications.

I start the Grid/ML program on my computer (the "client")

Grid/ML

Grid/ML is our ML-like language for writing ConCert applications.

It can run computations on the Grid and get back results.

Grid/ML

$$\mathrm{run\_on\_grid} : (\mathrm{unit} \to \alpha) \to \alpha$$

Run code of arbitrary type on the grid, and return the result.

Grid/ML

run_on_grid : (unit → α) → α

Run code of arbitrary type on the grid, and return the result.

run_on_grid (fn () => factorize *n*)

Grid/ML pitfalls

The client can perform I/O to communicate with the user.

But code that runs on the Grid cannot.

Grid/ML pitfalls

The client can perform I/O to communicate with the user.

But code that runs on the Grid cannot.

```
run_on_grid (fn () => openfile "result.dat")
```

No!

(runtime failure)

"But I'd never write that program!"

Can be more subtle:

```
fun crack_rsa factorer key =
   run_on_grid
      (fn () =>
         factorer (RSA.getmodulus key))
```

"But I'd never write that program!"

```
fun crack_rsa factorer key =
    run_on_grid
        (fn () =>
          factorer (RSA.getmodulus key))
```

this argument might use a database
of primes from disk?

"But I'd never write that program!"

```
fun crack_rsa factorer key =
   run_on_grid
       (fn () =>
         factorer (RSA.getmodulus key))
```

this free reference might consult a local keyring or keyring server (or do so in a later version without changing the module's interface)

"But I'd never write that program!"

```
fun crack_rsa factorer key =
   run_on_grid
      (fn () =>
        factorer (RSA.getmodulus key))
```

might modify some state within key,
but we made a copy of key

"But I'd never write that program!"

```
(* XXX don't screw up *)
fun crack_rsa factorer key =
   run_on_grid
       (fn () =>
          factorer (RSA.getmodulus key))
```

Resource *use*, not *reference*

```
let val l : (string x file) list = ...
    fun gsort cmp l =
      run_on_grid
          (fn () => ...
              let val (a, b) = split l
              in merge (gsort cmp a)
                       (gsort cmp b)
              end)
in
  gsort (fn ((s1, _), (s2, _)) => s1 ≤ s2) l
end
```

Not just Grid/ML

These problems come up in many distributed languages.

Not just Grid/ML

These problems come up in many distributed languages.

e.g. Java RMI:

Failure at send-time if not serializable

· · · but this excludes the previous (correct) program

Not just Grid/ML

These problems come up in many distributed languages.

e.g. Java RMI:

Failure at send-time if not serializable

· · · but this excludes the previous (correct) program

Silent copying of RPC arguments

· · · even if they are stateful

Not just Grid/ML

These problems come up in many distributed languages.

e.g. Java RMI:

Failure at send-time if not serializable

· · · but this excludes the previous (correct) program

Silent copying of RPC arguments

· · · even if they are stateful

Don't screw up

Design goals

ML5 should...

Design goals

ML5 should...

Support an arbitrary set of places

· · · (not just client and non-client)

Design goals

ML5 should...

Support an arbitrary set of places
· · · (not just client and non-client)

Statically check that resources will be used in the correct place

Design goals

ML5 should...

Support an arbitrary set of places

- · · · (not just client and non-client)

Statically check that resources will be used in the correct place

Not burden functional programs

- · · · *i.e.* ones that do not use local resources

Solution

Associate a place with each value

· · · and therefore each bound variable

Solution

Associate a place with each value

· · · and therefore each bound variable

Only allow values to be used in that place

· · · by tracking where exps will be evaluated

Solution

Associate a place with each value

· · · and therefore each bound variable

Only allow values to be used in that place

· · · by tracking where exps will be evaluated

Allow some values to be used globally

· · · for code that does not use local resources

Solution

The ML5 type system comes from **modal logic**.

(Lewis 1918)

Modal Logic

Modal logic is a family of logics with the ability to reason about truth from multiple perspectives.

We'll call these perspectives worlds.

Modal Logic

Rather than judgments of the form

$$A\ \mathsf{true},\ B\ \mathsf{true},\ \ldots\ \vdash\ C\ \mathsf{true}$$

(Standard propositional logic, the basis for ML.)

Modal Logic

Rather than judgments of the form

$$A \text{ true}, B \text{ true}, \ldots \;\vdash\; C \text{ true}$$

We have truth indexed by worlds w:

$$A \text{ true} @ w_1, B \text{ true} @ w_2, \ldots \;\vdash\; C \text{ true} @ w_3$$

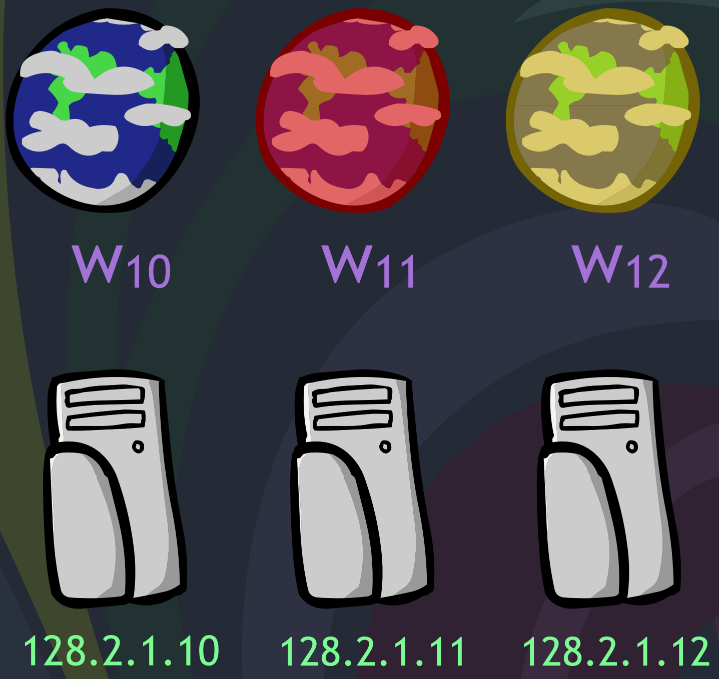
(Simpson 1994)

Modal Logic

A true @ w_1, B true @ w_2, w_4 exists, … $\vdash$ C true @ w_3

Worlds can be drawn from some set of known constants, or be hypothetical as here.

Computational Modal Logic



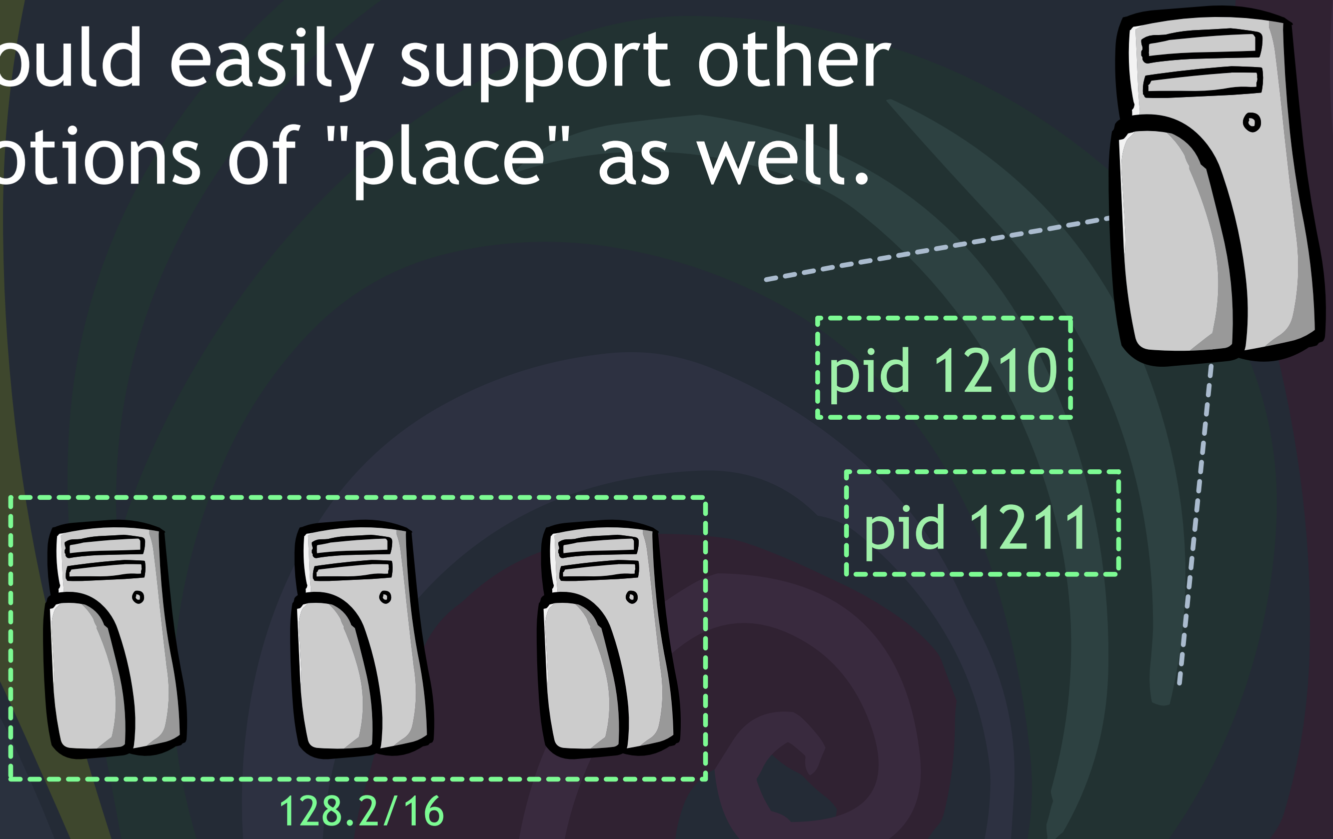
Using the Curry-Howard isomorphism, the worlds in the logic become the hosts in the programming language.

(Jia *et al.* 2004)

(Murphy *et al.* 2004)

Computational Modal Logic

Could easily support other notions of "place" as well.



ML5

ML5's typing judgment is thus indexed by worlds:

$$\Gamma \vdash M : A \mathbin{@} w$$

ML5

$$\Gamma \vdash M : A \mathbin{@} w$$

The expression M is okay to evaluate at world w.

ML5

$$\Gamma \vdash M : A \mathbin{@} w$$

The expression M is okay to evaluate at world w.

It will evaluate to a value of type A which can be used at w.

ML5

Variables are also adorned with their worlds:

$$\Gamma, x{:}B@w_1, y{:}C@w_2 \vdash M : A @ w$$

ML5

Variables are also adorned with their worlds:

$$\Gamma, x{:}B@w_1, y{:}C@w_2 \vdash M : A @ w$$

So they can only be used in the correct place:

$$\frac{}{\Gamma, x{:}A@w, \Gamma' \vdash x : A @ w}$$

ML5

The world on a variable indicates *where it makes sense*, not *where it is*.

ML5

The world on a variable indicates *where it makes sense*, not *where it is*.

ML5 programs may abstractly manipulate terms that only make sense elsewhere

- · · · store them in data structures
- · · · ship them to other hosts, etc.

ML5

The world on a variable indicates *where it makes sense*, not *where it is*.

ML5 programs may abstractly manipulate terms that only make sense elsewhere

· · · store them in data structures

· · · ship them to other hosts, etc.

They just may not consume them.

Computing with worlds

Dynamically, we need values with which to refer to worlds: addresses.

$$\frac{}{\Gamma \vdash \overline{w_1} : w_1 \text{ addr } @ \; w_2}$$

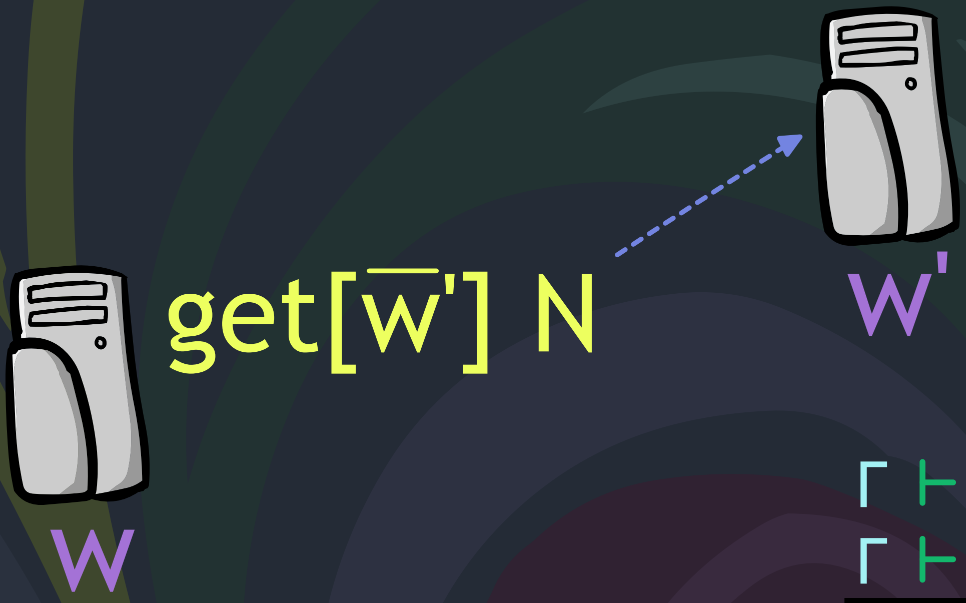
$$\frac{}{\Gamma \vdash \text{localhost}() : w \text{ addr } @ \; w}$$

Computing with worlds

We can use an address by transferring control to that world in order to evaluate an expression.

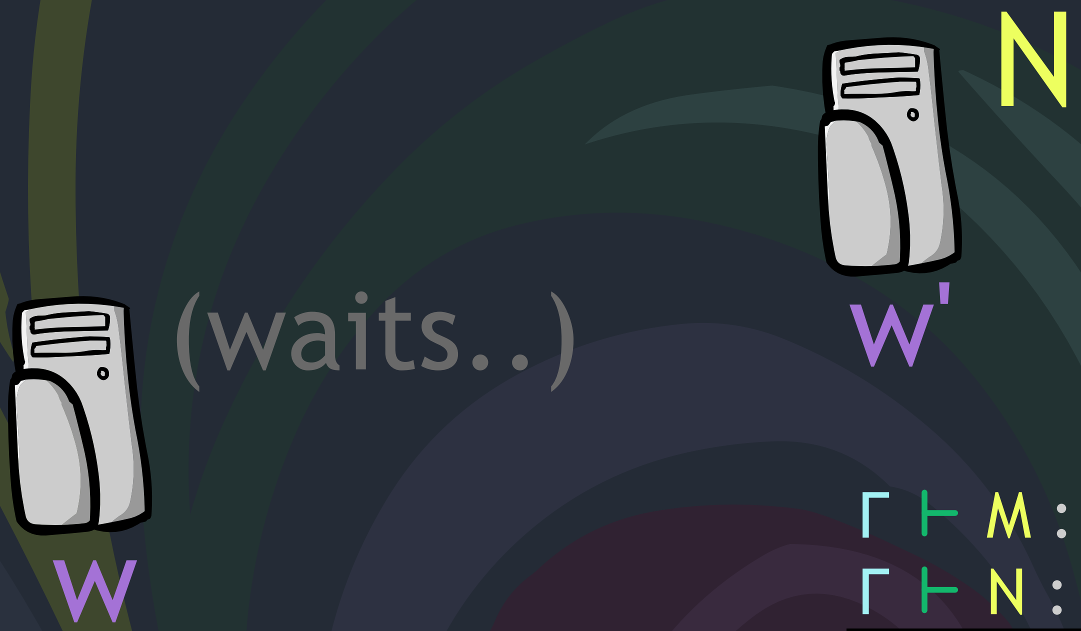
$$\frac{\begin{array}{c}\Gamma \vdash M : w'\ \mathrm{addr}\ @\ w \\ \Gamma \vdash N : A\ @\ w' \\ \text{(one more condition)}\end{array}}{\Gamma \vdash \mathrm{get}[M]\ N : A\ @\ w}$$

Computing with worlds



$$\frac{\begin{array}{l}\Gamma \vdash M : w' \text{ addr} @ w \\ \Gamma \vdash N : A @ w' \\ \text{(one more condition)}\end{array}}{\Gamma \vdash \text{get}[M]\ N : A @ w}$$

Computing with worlds

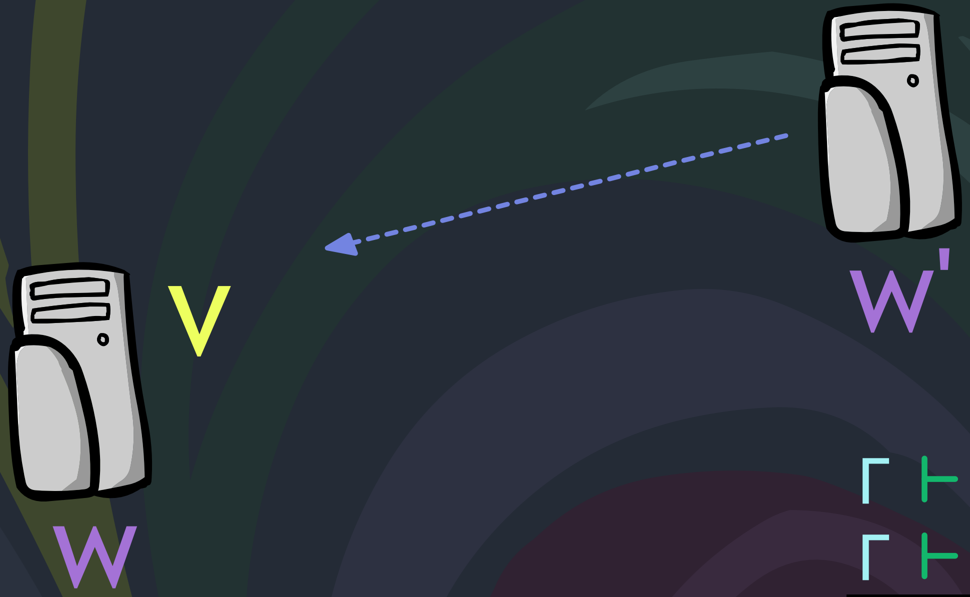


$$\frac{\Gamma \vdash M : w'\ \mathrm{addr}\ @\ w \qquad \Gamma \vdash N : A\ @\ w' \qquad \text{(one more condition)}}{\Gamma \vdash \mathrm{get}[M]\ N : A\ @\ w}$$

Computing with worlds

$$\frac{\begin{array}{l}\Gamma \vdash M : w'\ \mathrm{addr} @ w \\ \Gamma \vdash N : A @ w' \\ \text{(one more condition)}\end{array}}{\Gamma \vdash \mathrm{get}[M]\ N : A @ w}$$

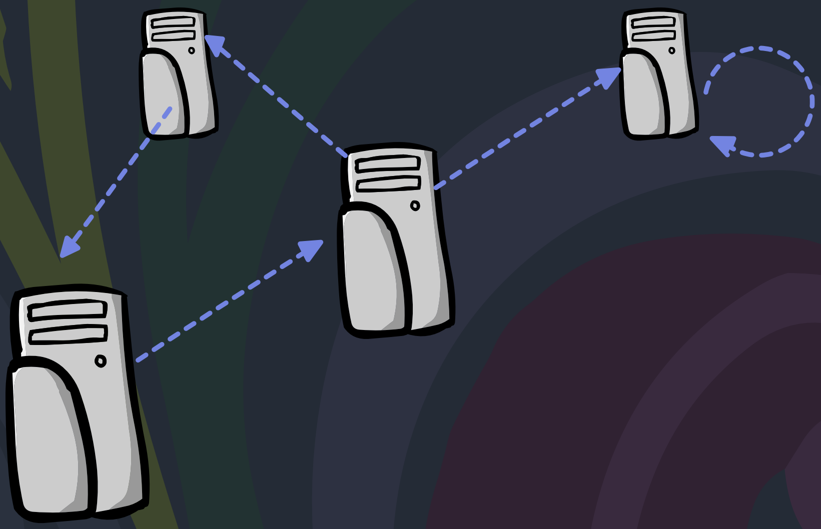
Computing with worlds



$$\frac{\Gamma \vdash M : w' \text{ addr } @ \ w \qquad \Gamma \vdash N : A \ @ \ w' \qquad \text{(one more condition)}}{\Gamma \vdash \text{get}[M]\ N : A \ @ \ w}$$

Computing with worlds

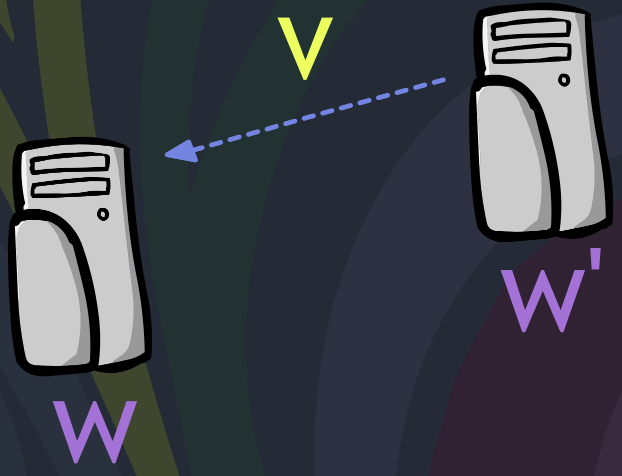
ML5 programs make their way around the network by these nested **get** expressions (only).



$$\frac{\begin{array}{c}\Gamma \vdash M : w' \ \mathrm{addr} \ @ \ w \\ \Gamma \vdash N : A \ @ \ w' \\ \text{(one more condition)}\end{array}}{\Gamma \vdash \mathrm{get}[M]\ N : A \ @ \ w}$$

Computing with worlds

Suspicious: v came from N : A @ w', but now has type A @ w. What if v refers to resources local to w'?



$$\frac{\begin{array}{l}\Gamma \vdash M : w'\ \mathrm{addr} @ w \\ \Gamma \vdash N : A @ w' \\ \text{(one more condition)}\end{array}}{\Gamma \vdash \mathrm{get}[M]\ N : A @ w}$$

Computing with worlds

Restrict get to "mobile" types.

$$\frac{\Gamma \vdash M : w' \text{ addr } @ \, w \qquad \Gamma \vdash N : A \, @ \, w' \qquad A \text{ mobile}}{\Gamma \vdash \text{get}[M] \, N : A \, @ \, w}$$

Computing with worlds

A type is mobile if its values can never reference local resources.

$$\frac{}{w \text{ addr mobile}} \qquad \frac{}{\text{int mobile}}$$

$$\frac{A \text{ mobile} \quad B \text{ mobile}}{A \text{ x } B \text{ mobile}}$$

Computing with worlds

A type is mobile if its values can never reference local resources.

$$\cancel{\dfrac{}{\text{file mobile}}}$$

Files *are* references to local resources!

$$\cancel{\dfrac{}{A \rightarrow B \text{ mobile}}}$$

Functions may access local resources when applied.

Modalities

The mobile judgment concerns types. We also care about the portability of specific values.

Modalities

Not all functions are portable, but some are:

```
(fn x => write(fd, x))
  (fn x => x + 1)
```

Modalities

The type □A classifies computations that are portable, even if A is not mobile.

```
   box w'. (fn x => x + 1)
    : □(int -> int) @ w
```

Modalities

$$\frac{\Gamma, w' \text{ world} \vdash M : A\ @\ w'}{\Gamma \vdash \text{box } w'.M : \Box A\ @\ w}$$

Modalities

To see that M does not use any local resources, check that it is well typed at a world about which nothing is known, w'.

$$\frac{\Gamma,\ w' \text{ world} \vdash M : A \mathbin{@} w'}{\Gamma \vdash \text{box } w'.M : \Box A \mathbin{@} w}$$

Modalities

We can open a box to evaluate its contents.

$$\frac{\Gamma \vdash M : \Box A \mathbin{@} w}{\Gamma \vdash \text{unbox}\ M : A \mathbin{@} w}$$

Modalities

$\Box A$ is mobile no matter what A is.

$$\frac{}{\Box A \text{ mobile}}$$

Modalities

Two other modalities of interest:

$\Diamond A$ — A value of type A that makes sense at *some* abstract world.

Modalities

Two other modalities of interest:

$\Diamond A$ A value of type A that makes sense at *some* abstract world.

A at w A value of type A that makes sense at the world w.

Modalities

$$\frac{\Gamma \vdash v : A \mathbin{@} w'}{\Gamma \vdash \mathrm{hold}_{w'}\, v : A \ \mathrm{at}\ w' \mathbin{@} w}$$

$$\frac{\begin{array}{l}\Gamma \vdash M : A \ \mathrm{at}\ w' \mathbin{@} w \\ \Gamma, x{:}A \mathbin{@} w' \vdash N : C \mathbin{@} w\end{array}}{\Gamma \vdash \mathrm{leta}\ x = M\ \mathrm{in}\ N : C \mathbin{@} w}$$

Modalities

File-sorting example:

val l : (string x file at w_{home}) list

Examples

$\Box A$ at $w' \rightarrow w'$ addr $\rightarrow A$ @ w

```
fun f b a =
  leta x = b
  in unbox (get[a] x)
  end
```

easy!

Examples

$\Box A \rightarrow \Box\Box A \quad @\ w$

```
fun f (b : □A) =
  let val a : w addr @ w = localhost()
  in box w'.(get[a] b)
  end
```

Examples

$\Box A \rightarrow \Box\Box A \quad @ \; w$

```
fun f (b : □A) =
  let val a : w addr @ w = localhost()
  in box w'.(get[a] b)
  end
```

No! Ill-typed!

The address a can only be used at w, but we try to use it at w'.

Valid values

ML5 also supports "valid" values.

$$\Gamma, \boxed{u{\sim}A} \vdash M : B \mathbin{@} w$$

Valid values

ML5 also supports "valid" values.

$$\dfrac{\Gamma, u\sim A \vdash M : B \;@\; w}{\Gamma, u\sim A, \Gamma' \vdash u : A \;@\; w}$$

The valid variable u can be used at any world.

Making valid values

Two ways to introduce valid variables:

$$\frac{\begin{array}{l}\Gamma \vdash M : A \mathbin{@} w \qquad\qquad A \text{ mobile} \\ \Gamma, u{\sim}A \vdash N : C \mathbin{@} w\end{array}}{\Gamma \vdash \text{putm } u = M \text{ in } N : C \mathbin{@} w}$$

$$\frac{\begin{array}{l}\Gamma, w' \text{ world} \vdash v : A \mathbin{@} w' \\ \Gamma, u{\sim}A \vdash N : C \mathbin{@} w\end{array}}{\Gamma \vdash \text{putv } u = w'.v \text{ in } N : C \mathbin{@} w}$$

Making valid values

(both are **shorthands** for a use of a fourth modality, OA. Details in the proposal.)

(Park 2005)

Example revisited

$\Box A \to \Box\Box A$ @ w

```
fun f (b : □A) =
  let val a : w addr @ w = localhost()
  in box w'.(get[a] b)
  end
```

Ill-typed!

Example revisited

$\Box A \to \Box\Box A$ @ w

```
fun f (b : □A) =
  putm u ~ w addr = localhost()
  in box w'.(get[u] b)
  end
```

(addresses are mobile)

Example: libraries

Validity allows us to share common library code without boxing and explicitly moving it around.

```
putv map =
   w'.(fn f =>
         let fun go nil = nil
                | go (h::t) = f h :: go t
         in go
         end)
in ...
end
```

Implementation

A major part of the thesis work will be the implementation of ML5.

Compiler

Translate source programs into runnable code

Runtime

Run the program on the network of hosts

Compiler

Type-directed, certifying compiler

- catch **compiler bugs**
- output suitable for **trustless grid**

Server OK

Type-directed, certifying compiler

- catch compiler bugs
- output suitable for trustless grid

Interesting because of worlds, validity

- modal types for low level languages
- elaboration, CPS, closure conversion in the proposal
- formalized some phases in Twelf

Runtime

Form and maintain the network

· · · can reuse parts of ConCert here

Runtime

Form and maintain the network

· · · can reuse parts of ConCert here

Verify certificiates

· · · handled by TALT

Runtime

Form and maintain the network

· · · can reuse parts of ConCert here

Verify certificiates

· · · handled by TALT

Distributed garbage collection

· · · hard; will do something really simple

Runtime

Form and maintain the network

· · · can reuse parts of ConCert here

Verify certificiates

· · · handled by TALT

Distributed garbage collection

· · · hard; will do something really simple

Bind to local resources

· · · when code/data arrive

Application

Should be realistic (even useful)

Application

Should be realistic (even useful)

Should be able to compare it to other systems

Application

Should be realistic (even useful)

Should be able to compare it to other systems

No fancy concurrency / fault tolerance

· · · but I may need simple support for these

Application

Should be realistic (even useful)

Should be able to compare it to other systems

No fancy concurrency / fault tolerance

. . . but I may need simple support for these

Should rely on use of local resources

Application Ideas?

Scientific computing

· · · localized resources: instruments, storage

Application Ideas?

Scientific computing

· · · localized resources: instruments, storage

Multiplayer game, collaborative workspace

· · · localized resources: keyboard, screen

Application Ideas?

Scientific computing

· · · localized resources: instruments, storage

Multiplayer game, collaborative workspace

· · · localized resources: keyboard, screen

Distributed network measurements

· · · easy; short-running

Application Ideas?

Scientific computing

· · · localized resources: instruments, storage

Multiplayer game, collaborative workspace

· · · localized resources: keyboard, screen

Distributed network measurements

· · · easy; short-running

Distributed filesystem

· · · arbitrarily hard (or easy); lots to compare

Application Ideas?

· · · your idea here?

Summary

Modal type systems provide an elegant and practical way to control spatially distributed resources in mobile programs.

Summary

A novel **programming language** whose **type system** supports a concept of *place*.

Summary

A novel **programming language** whose **type system** supports a concept of *place*.

Based on modal logic

· · · support localized resources
along with global code/data

Summary

A novel **programming language** whose **type system** supports a concept of *place*.

Based on modal logic

· · · support localized resources
along with global code/data

Thesis plan: design, implement, apply

· · · components of both theory and practice

Thanks!

Questions?