Synching up

A few models of "side-by-side" computing

L.G. Meredith
Biosimilarity LLC

Agenda

- Terminology and notation
- A few models
- Process calculi
- Actors
- Threads
- Comparisons
- Conclusions

L.G. Meredith
Biosimilarity LLC

Terminology and Notation

- Model -- provides clear, concise and *effective* definition of a computation
  - lambda calculus
  - Turing machines
  - rewrite systems
  - Petri nets

L.G. Meredith
Biosimilarity LLC

Terminology and Notation

- Parallel -- most tightly coupled form of side-by-side computation; everything fails together
- Concurrent -- a little less tightly coupled; some computations can fail and others succeed
- Distributed -- least tightly coupled; introduces macro time and space phenomena -- e.g. clock drift

L.G. Meredith
Biosimilarity LLC

A few models

- The process calculi
- Actors
- Data flow
- Concurrent constraint programming
- Threads
- Software transactional memory
- Petri nets

L.G. Meredith
Biosimilarity LLC

Process calculi

Encouraged by the success of typed functional languages (like ML progenitor of Scala and F#) base on the typed lambda calculus Milner and Hoare were looking for algebraic models of concurrency for reasoning, verification and ultimately execution

operational semantics

denotational semantics

- CCS, CSP

both assumed fixed communication topology

- π-calculus, join calculus, blue calculus

allowed dynamic communication topology

- ambient calculus

arguably the first model to directly address distribution

L.G. Meredith
Biosimilarity LLC

Example: the rho calculus

a simpler variant of π-calculus -- yet higher order

These are the ingredients of a complete specification of a DSL: Syntax, Structural Equvalence and Reduction Rules

Syntax	Structural Equivalence	Reduction Rules
P,Q ::= 0		
| x?(y1, ..., yn)P	P | 0 = P	x?(y1, ..., yn)P | x!(Q1, ..., Qn)
| x!(Q1, ..., Qn)	P | Q = Q | P	-> P{ @<Q1>/y1, ..., @<Qn>/yn }
| P|Q		
| *x	P = Q => @<P> = @<Q>	P -> P' => P | Q -> P' | Q
x,y ::= @<P>	@<*@<P>> = @<P>	

Like the lambda calculus -- full specification of the language fits on a page!

L.G. Meredith
Biosimilarity LLC

Example: the rho calculus

a simpler variant of π-calculus -- yet higher order

Like the lambda calculus -- has the potential to be typed!

Syntax	Structural Equivalence	Reduction Rules
P,Q ::= 0		
| x?(y1, ..., yn)P	P | 0 = P	x?(y1, ..., yn)P | x!(Q1, ..., Qn)
| x!(Q1, ..., Qn)	P | Q = Q | P	-> P{ @<Q1>/y1, ..., @<Qn>/yn }
| P|Q		
| *x	P = Q => @<P> = @<Q>	P -> P' => P | Q -> P' | Q
x,y ::= @<P>	@<*@<P>> = @<P>	

L.G. Meredith
Biosimilarity LLC

Actors

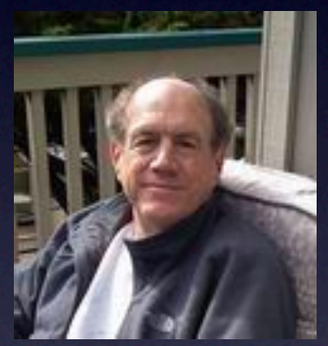
- What is an actor?
- How does that relate to what's out there today?

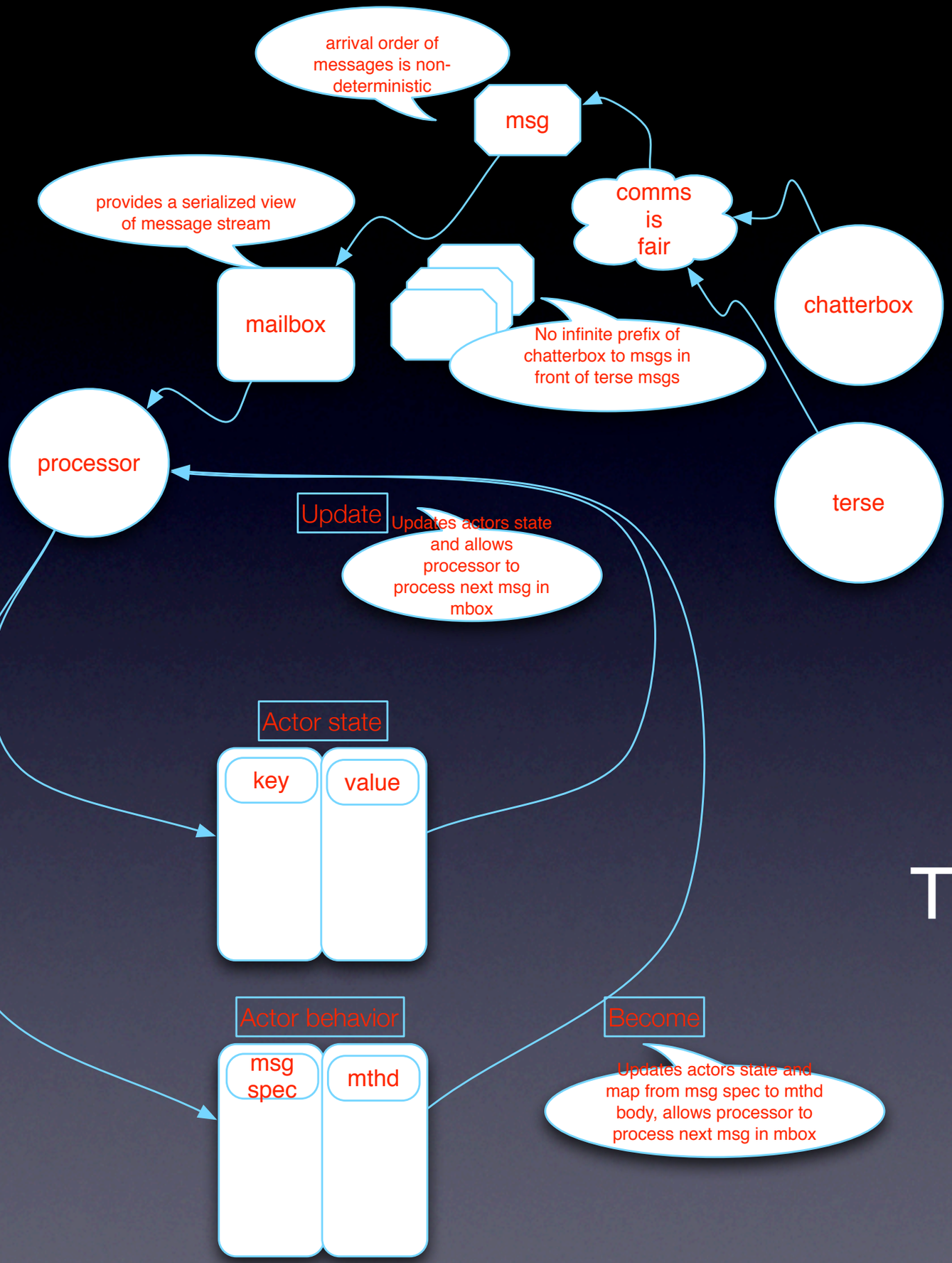
That's a great question! We're still waiting for a specification of what the actual computational model is from the various modern libraries.

The older language-based proposals -- ABCL, ABLC/R, Rosette, etc -- were pretty clear

L.G. Meredith
Biosimilarity LLC

The actor model





Data flow

- Not what steps, but what data
- Combinators are boxes with typed ports
- Natural segue to concurrent constraint programming
- Scala concurrent collections + delimited continuations could provide a substrate for dataflow model

L.G. Meredith
Biosimilarity LLC

Threads

- Synchronization
  - Locks
  - Monitors
- Sharing
  - Potentially everything is shared!
  - Mutate at your own risk

This is an old story, now. We don't need to be reminded of the pain...

L.G. Meredith
Biosimilarity LLC

Monads

```
for(
   event1 <- evntStrm1( pattern1 );
   ...
   eventN <- evntStrmN( patternN );
   if ( condWithPossibleBacktracking( event, ..., eventN ) )
) {
    handle( event1, ..., eventN )
}
```

L.G. Meredith
Biosimilarity LLC

Monads

Generalizes

```
select E1 ... En from DataSource1 ...
DataSourceN
where cond
return handle( E1, ..., En )
```

L.G. Meredith
Biosimilarity LLC

Monads

- LINQ
- Map/reduce
  - Hadoop
  - ...
- RX frameworks

Hiding inside these we find monadic structure -- which can be accessed by a polymorphic form of SELECT-FROM-WHERE

L.G. Meredith
Biosimilarity LLC

Comparison

	Process calculi	Actors	Data flow	Threads
Parallel	★	★	★	★
Concurrent	★	★		★
Distributed	★	★		
Programming	★	★	★	★
Reasoning	★		★	
Composes	★	★	★	
Transactional	★	★	★	★

Meaning plays well with transactions

Arguably, STM mitigates this...

L.G. Meredith
Biosimilarity LLC

Comparison

	Process calculi	Monads
Parallel	★	★
Concurrent	★	★
Distributed	★	★
Programming	★	★
Reasoning	★	★
Composes	★	★
Transactional	★	★

L.G. Meredith
Biosimilarity LLC

Conclusions

- In the world of *sequential* computation typed functional languages are winning out as providing a model of computation that scales in terms of
  - complexity management
  - cost management
  - performance

L.G. Meredith
Biosimilarity LLC

Conclusions

- In the world of side-by-side computation we are still waiting for a model or family of models of computation that scales in terms of
  - complexity management
  - cost management
  - performance

L.G. Meredith
Biosimilarity LLC

Conclusions

- No one size fits all answer
- Know your domain
- When locality is clear use it!
- Types are coming soon!

L.G. Meredith
Biosimilarity LLC