Coordination and safe program execution: problems, solutions and open issues.

Minas Charalambides

October 29, 2014

Motivation

Parallel programming

Too many schedules lead to

- Atomicity violations
- Data races
- Deadlocks

Motivation

Parallel programming

Too many schedules lead to

- Atomicity violations
- Data races
- Deadlocks

Idea: Restrict the set of schedules.

Before

- Locks
- Synchronizers
- Synchronization languages

Recently

- Data-centric synchronization
- Session types

Synchronizers example – Dining Philosophers

```
PickUpConstraint(c1, c2, phil)
{
  atomic( (c1.pick(sender) where sender = phil),
          (c2.pick(sender) where sender = phil) ),
  (c1.pick where sender = phil) stops
}
```

Svend Frølund and Gul Agha: *A language framework for multi-object coordination*. ECOOP 1993

Synchronization language example

```
class Buffer
{
  int size = 0;
  Object[] array = new Object[10];

  void put(Object o) [size < 0]
  {
    array[size++] = obj;
  }

  Object get() [size > 0]
  {
    return array[--size];
  }
}
```

Data-Centric Synchronization

```
class ArrayList {
  int size;
  Object[] entries;

  Object get(int i) {
    synchronized(lock) {
      if (0 <= i && i < this.size) {
        return this.entries[i];
      } else {
        return null;
      }
    }
  }
  void addAll(ArrayList al) {
    synchronized(lock) {
      this.size += al.size;
      // copy elements
    }
  }
}
```

Dolby et al. *A data-centric approach to synchronization.* TOPLAS 2012

Data-Centric Synchronization

```
class ArrayList {
  atomicset(L);
  atomic(L) int size;
  atomic(L) Object[] entries;

  Object get(int i) {
    if (0 <= i && i < this.size) {
      return this.entries[i];
    } else {
      return null;
    }
  }
  void addAll(unitfor(L) ArrayList al) {
    this.size += al.size;
    // copy elements
  }
}
```

Dolby et al. *A data-centric approach to synchronization.* TOPLAS 2012

Converting a Program to Atomic Sets Requires Understanding its Concurrency Structure

- Data-centric synchronization looks beneficial
- But: must *understand* old synchronization to convert it

Conversion Experience of Dolby et al.

- Takes several hours for rather simple programs
- 2 out of 6 programs lack synchronization of some classes
- 2 out of 6 programs accidentally introduced global locks

Our Algorithm

Avoid conversion errors by automatically determining annotations from program traces.

First attempt for automated conversion

Algorithm Strategy and Properties

- Based on set membership for classification
- Analysis done during replay of observation trace
- Observation data scales in size of execution

Evaluation

- Qualitative evaluation shows promising results and room for improvement
- Inferred annotations mostly agree with manual annotations
- Bugs found in manual annotations

P. Dinges, M. Charalambides, G. Agha: *Automated Inference of Atomic Sets for Safe Concurrent Execution*. PASTE workshop, 2013.

Desirable Properties of Atomic Set Inference Algorithms Using Traces

- Take observation *distance* and *scope* into account
- Produce stable annotations that do not vary wildly between different runs
- Observation data bounded by size of codebase
- Produce atomic set estimates on-the-fly
- Improve analysis with longer execution traces

Second attempt: probabilistic inference of atomic sets

Assumptions about Input Programs

- Methods perform semantically meaningful operations
- Data fields that a method accesses are likely connected by invariant

Algorithm Idea

- Observe which pairs of fields a method accesses atomically
  - This is (Bayesian) *evidence* that fields are connected through a semantic invariant
- Store current beliefs for all field pairs in *affinity matrices*

.

P. Dinges, K. Palmskog, G. Agha – work in progress

The Analysis Algorithm Must Support Indirect Field Access and Access Paths

Indirect Access and Distance

- High-level semantic operations use low-level operations
- Example: `get()` might call `getSize()` instead of accessing field `size`
- Propagate observed access to caller's scope
- Quantify directness of access as *distance*

The Analysis Algorithm Must Support Indirect Field Access and Access Paths

Indirect Access and Distance

- High-level semantic operations use low-level operations
- Example: `get()` might call `getSize()` instead of accessing field `size`
- Propagate observed access to caller's scope
- Quantify directness of access as *distance*

Access Paths

- Methods traverse the object graph
- Track *access paths* instead of field names
- Example: `this.urls.size`

Likelihood ratios

Bayesian Inference Variables

H: "f and g are connected through an invariant" [Hypothesis]
e_k: "f, g accessed (non-)atomically with distance d_k" [evidence]

Consider a sequence of observations $e_1, \ldots, e_n$ w.r.t. f and g.
Want to know *probability that H holds given* $e_1, \ldots, e_n$

Likelihood ratios

Bayesian Inference Variables

H: "f and g are connected through an invariant" [Hypothesis]
e_k: "f, g accessed (non-)atomically with distance d_k" [evidence]

Consider a sequence of observations $e_1, \ldots, e_n$ w.r.t. f and g.
Want to know *probability that H holds given $e_1, \ldots, e_n$*

$$\frac{P(H|e_1,\ldots,e_n)}{P(\neg H|e_1,\ldots,e_n)} = \frac{P(e_1,\ldots,e_n|H)}{P(e_1,\ldots,e_n|\neg H)} \times \frac{P(H)}{P(\neg H)}$$

$$\text{updated info} = \text{info from observations} \times \text{original info}$$

$$\text{posterior odds} = \text{likelihood ratio} \times \text{prior odds}$$

$$O(H|e_1,\ldots,e_n) = L(e_1,\ldots,e_n|H) \times O(H)$$

Conditional Independence

If $e_1, \ldots, e_n$ are conditionally independent given H, we can write

$$P(e_1, \ldots, e_n | H) = \prod_{k=1}^{n} P(e_k | H)$$

and similarly for $\neg H$, whereby

$$O(H | e_1, \ldots, e_n) = O(H) \prod_{k=1}^{n} L(e_k | H)$$

Adding one more piece of evidence e_{n+1}, we get

$$O(H | e_1, \ldots, e_n, e_{n+1}) = L(e_{n+1} | H)\, O(H | e_1, \ldots, e_n)$$

Hence, if we have independence, know $O(H)$, and can compute $L(e_k | H)$, we can update odds on-the-fly when observing!

Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

Conditional Independence

- Coarse-grained hypothesis space: $H \cup \neg H$
- With conditional independence, $e_1, \ldots, e_n$ should depend only on hypothesis, not on systematic external influence
- However, we have at least the following external factors:
  - Workload
  - Scheduler

Conditional Independence, continued

- Working assumption: good workload and long executions minimize external influence
- Currently comparing results of instrumented workload with plain workloads
- Given good enough driver programs (test suites) that run long enough, scheduler and driver influence becomes minimal
- It is always safe to place f and g in an atomic set even if no invariant holds

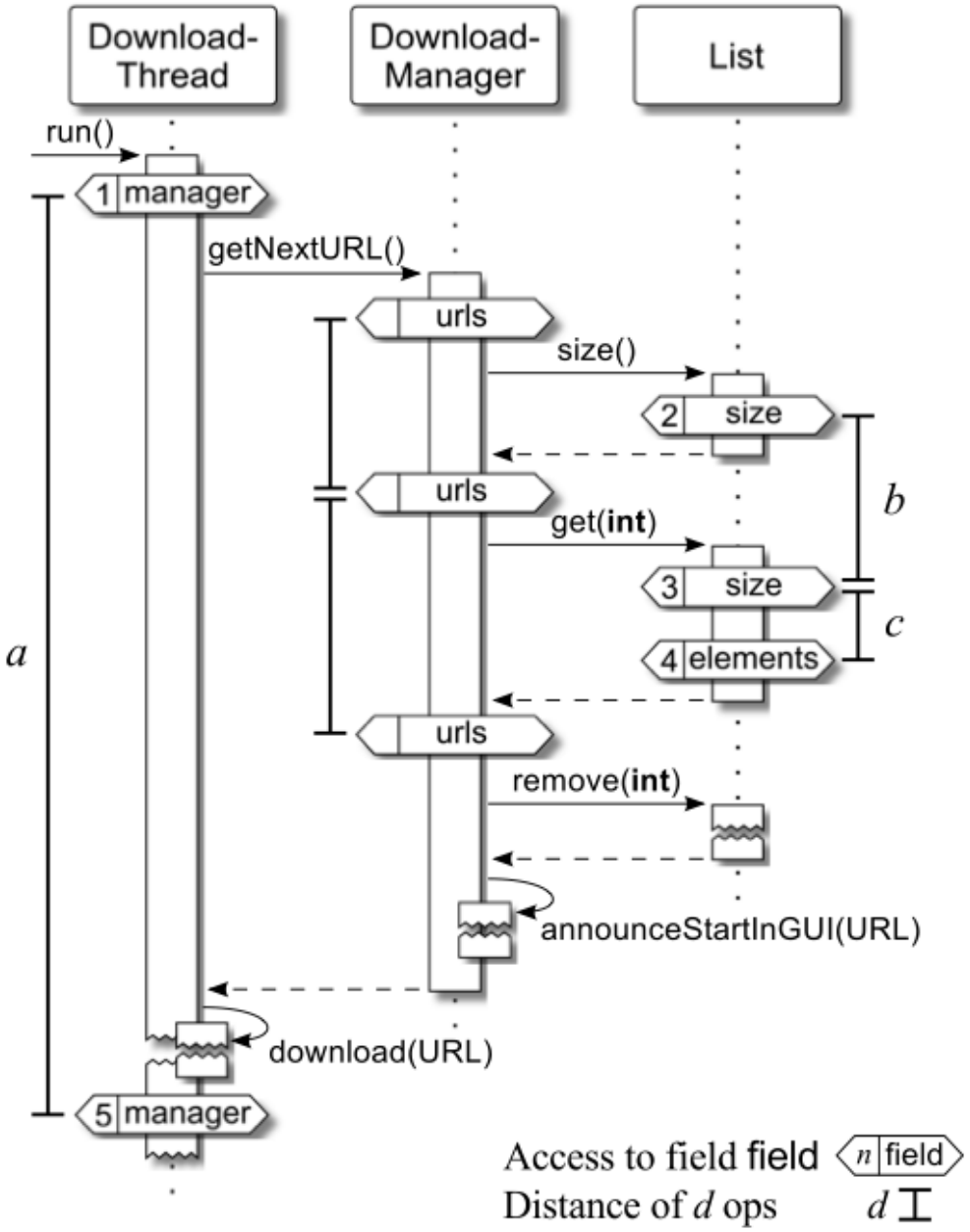
Mapping Observations to Likelihoods

For all f and g, we let

$$O(H) = \frac{P(H)}{P(\neg H)} = 1$$

Then, f and g equally likely *in* as *not in* atomic set.

The effect of access distance

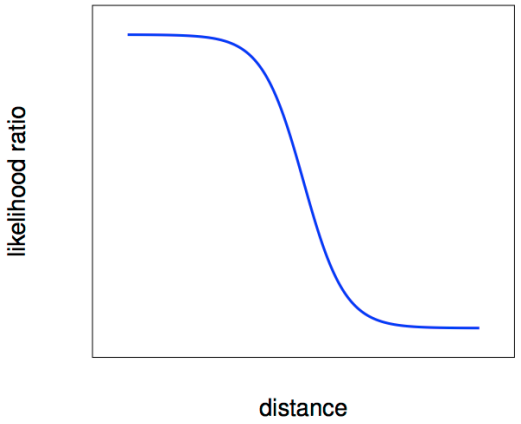


Mapping Observations to Likelihoods

- Given access observation e_k for fields f and g with operation distance d, need to compute $L(e_k|H)$
- $L(e_k|H)$ should increase as d decreases up to some maximum, after which it is flat
- $L(e_k|H)$ should decrease as d increases down to some minimum, after which it is flat

Mapping Observations to Likelihoods

- Given access observation e_k for fields f and g with operation distance d, need to compute $L(e_k|H)$
- $L(e_k|H)$ should increase as d decreases up to some maximum, after which it is flat
- $L(e_k|H)$ should decrease as d increases down to some minimum, after which it is flat
- We use a linear approximation of a *logistic curve*



Parameter Space

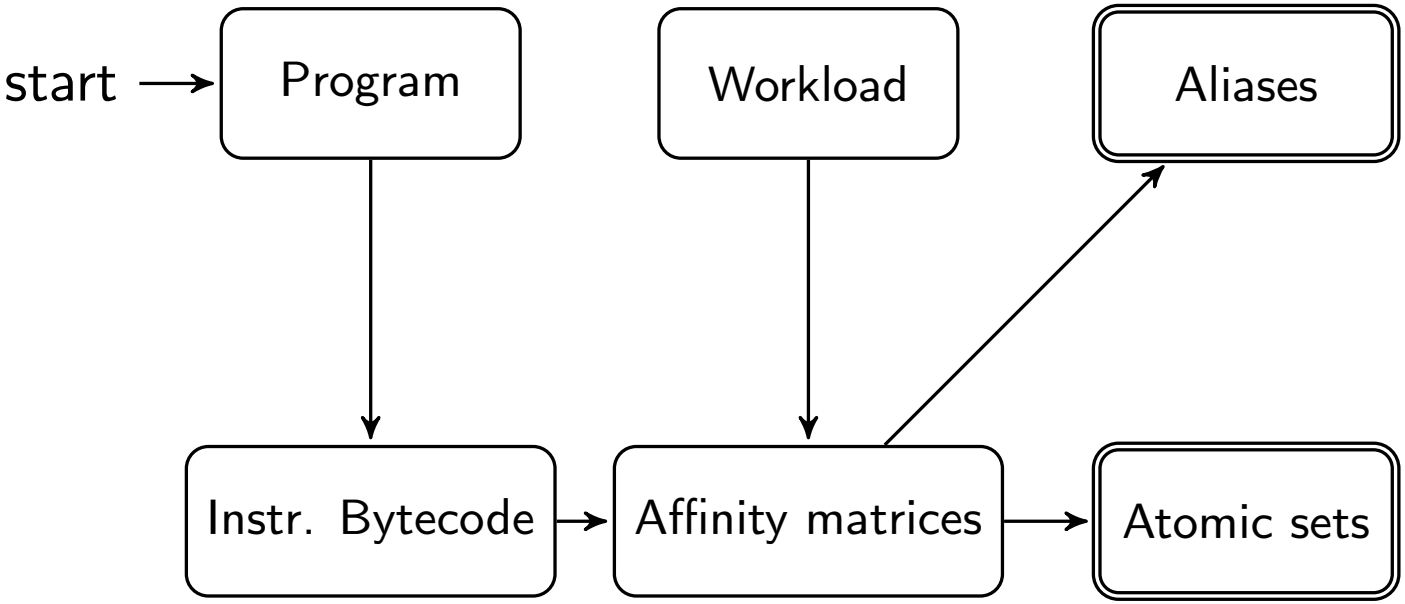
To achieve good results in practice:

- How much should prior odds be revised when observing interleaving access?
- How steep should the logistic curve be?
- How much penalty to distance should, e.g., call and return statements give?

Advantages of On-the-Fly Bayesian Inference

- Likelihoods incorporate scope and distance of observations
- Beliefs can be revised by new evidence, and thus improve with longer executions
- Analysis becomes robust and insensitive to outlier observations
- Size of observation data is in the size of the codebase, not size of execution
- Infers aliases similarly to atomic sets, which is hard to do statically

Implementation Toolchain



Evaluation Strategy: Qualitative Comparison Against Manual Annotations of 6 Programs

Qualitative Evaluation

- Currently comparing inferred annotations against manual annotations for 6 programs
- Manual annotations by AJ developers

Program	*Description*	*kLoC*	*Cls.*
collections	OpenJDK collections	11.1	171
elevator	Elevator simulation	0.3	6
jcurzez1	Console window library	2.7	78
jcurzez2	Console window library	2.8	79
tsp2	Traveling salesman	0.5	6
weblech	Web site crawler	1.3	12

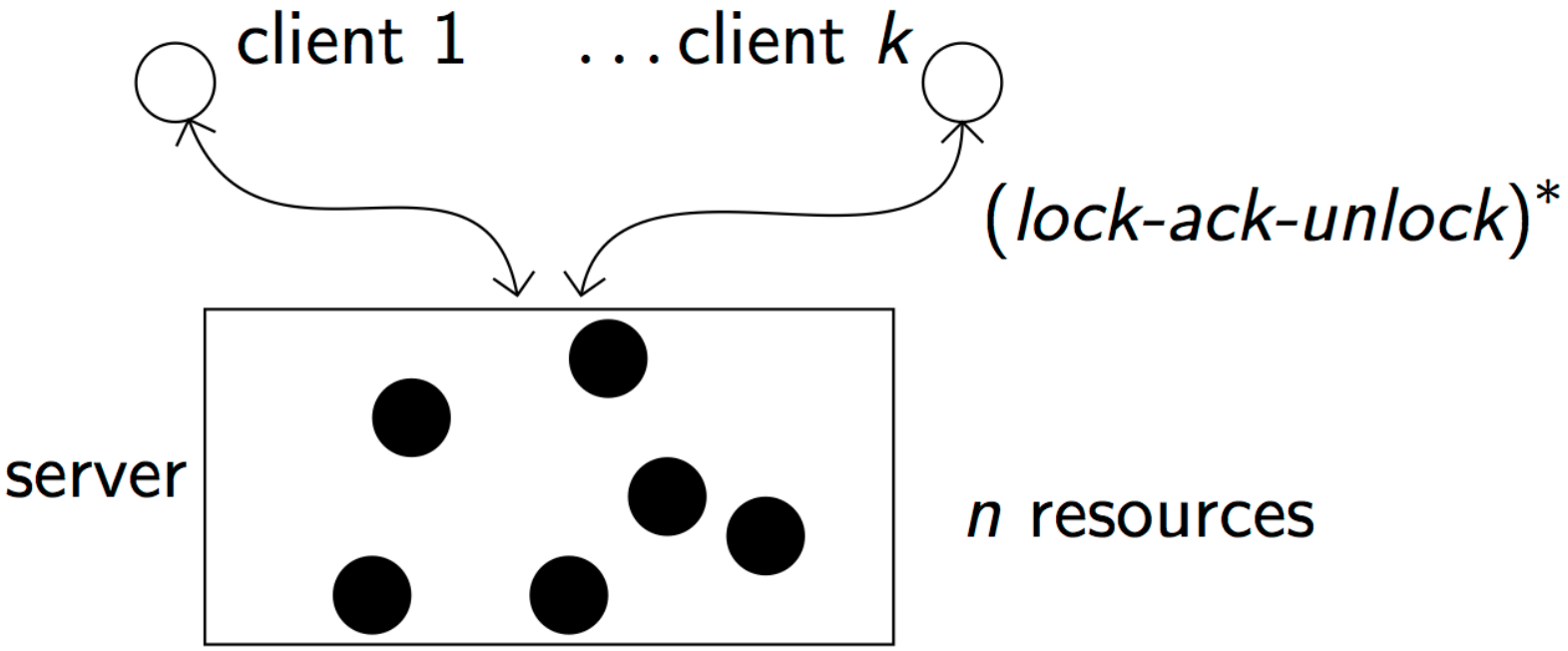
Result: Inferred Annotations are Similar to Manual Annotations

- Inferred annotations mostly agree with manual annotations
- Mistakes in manually annotated programs were revealed in some cases
- Only a few mistakes, due to workload (fuzzer) incompleteness

Session types

- Apply to all parallel programming models
- We focus on Actors
- Determine adherence to communication protocols at compile time

Example - Limited resource sharing



Example - Continued

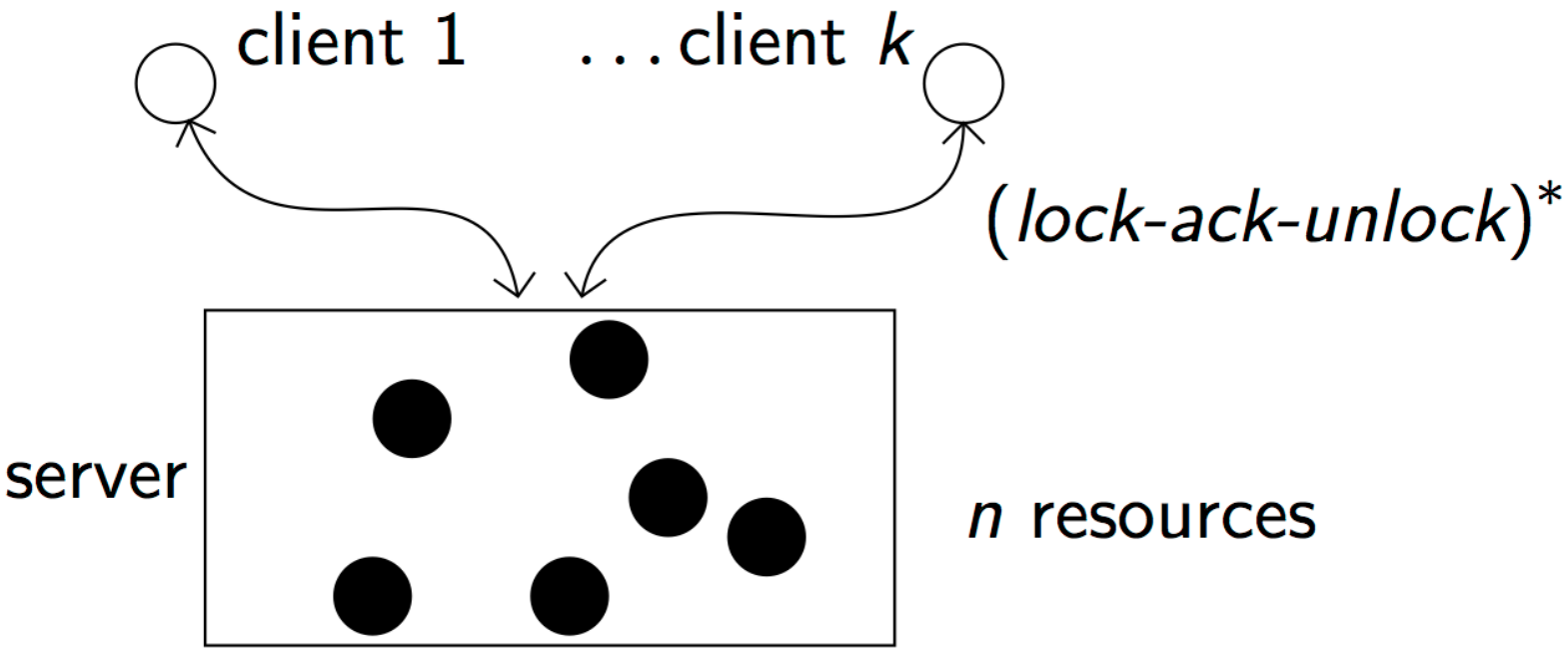
```
// globals
param n
param k
struct lock[n] = { }
struct unlock[n] = { }
struct ack[n] = { }

// server
actor s = {
  spawn (i = 1..n
    message l : lock[i]
    message a : ack[i]
    message u : unlock[i]
    while NotDone {
      select (j = 1..k
        recv(c[j], l);
        send(c[j], a);
        recv(c[j], u); )
    })
```

```
// client
actor c[k] = {
  spawn (i = 1..n
    while NotDone {
      message l : lock[i]
      message a : ack[i]
      message u : unlock[i]
      send(s, l);
      recv(s, a);
      // compute ...
      send(s, u);
    })
```

Parameterization

Example (System-A type)

$$\mathop{\Big\|}_{i=1}^{n} \left(\bigoplus_{j=1}^{k} \left(c_j \xrightarrow{lock_i} s \; ; \; s \xrightarrow{ack_i} c_j \; ; \; c_j \xrightarrow{unlock_i} s \right) \right)^*$$



System-A Global type syntax

G	$::=$	$a \xrightarrow{t} b$	(G-Interaction)	$\mid$	$G\ ;\ G$	(G-Seq)
	$\mid$	$G \oplus G$	(G-Choice)	$\mid$	$\bigoplus_{i=1}^{n} G_i$	(G-Choice)
	$\mid$	$G \parallel G$	(G-Parallel)	$\mid$	$\parallel_{i=1}^{n} G_i$	(G-Parallel-N)
	$\mid$	$G \otimes G$	(G-Shuffle)	$\mid$	$\bigotimes_{i=1}^{n} G_i$	(G-Shuffle-N)
	$\mid$	G^*	(G-KleeneStar)	$\mid$	$\parallel_{*} G$	(G-Parallel-Star)
	$\mid$	(G)	(G-Paren)			

Charalambides et al. *Parameterized Concurrent Multi-Party Session Types.* FOCLASA 2012

Previously

How to infer types?

Previously

Like this!

(Inf-Select-N)

$$\frac{\begin{array}{cc} S \vdash c_1 : cons & S \vdash c_2 : cons \\ \Gamma \vdash R[i/k] : L_i & k \in fv(R) \\ first\,(L_i) = x?y & x = a_i \vee y = m_i \end{array}}{\Gamma \vdash \mathsf{select}(k = c_1..c_2\ R) : (\bigoplus_{i=c_1}^{c_2} L_i)}$$

(Inf-Select)

$$\frac{\begin{array}{cc} \Gamma \vdash R_1 : L_1 & \Gamma \vdash R_2 : L_2 \\ first\,(L_1) = a?m & first\,(L_2) = a'?m' \\ \multicolumn{2}{c}{a \neq a' \vee m \neq m'} \end{array}}{\Gamma \vdash \mathsf{select}(R_1\ R_2) : (L_1 \oplus L_2)}$$

(Inf-Repeat)

$$\frac{\Gamma \vdash R : L \quad S \vdash \mathsf{n} : cons}{\Gamma \vdash \mathsf{repeat}\ n\ R : (L^n)}$$

(Inf-While)

$$\frac{\Gamma \vdash R : L \quad S \vdash \mathsf{exp} : Boolean}{\Gamma \vdash \mathsf{while\ exp}\ R : (L^*)}$$

(Inf-If)

$$\frac{\begin{array}{cc} \Gamma \vdash R_1 : L_1 & \Gamma \vdash R_2 : L_2 \\ \multicolumn{2}{c}{S \vdash \mathsf{exp} : Boolean} \end{array}}{\Gamma \vdash \mathsf{if\ exp}\ R_1\ \mathsf{else}\ R_2 : (L_1 \oplus L_2)}$$

(Inf-Environment)

$$\frac{\begin{array}{c} \Pi = \{a \mid \text{“actor } a = \{R_a\}\text{”} \in P\} \\ \forall a \in \Pi,\ \Gamma \vdash R_a : L_a \end{array}}{\Gamma \vdash P : \Delta \text{ where } \Delta = \{a : L_a \mid a \in \Pi\},}$$

Charalambides et. al, Science of Computer Programming Journal – under review

Previously

- All required information lies in program syntax
- Makes for an awkward programming model

Now

Current work

- More natural programming model
- Actual actor language
- Static analysis infers global types
- Dynamic process creation

Summary

- Parallel programming is hard. Need to restrict schedules.
- From control-centric to data-centric synchronization
- Session types
- Session type inference
- Open problem: Actor creation

Thank you.

Questions?