

# Combining Timing, Localities and Migration in a Process Calculus

Andrew Hughes

<http://www.dcs.shef.ac.uk/~andrew>

Department of Computer Science  
University of Sheffield

BCTCS - 05/04/2006

# Outline

- 1 Introduction
- 2 The Starting Point: A Calculus with Global Synchronization
- 3 Localities: The First Step to Mobility
- 4 Migration Gives Mobility
- 5 Further Thoughts and Conclusions

# Aim

- Combine discrete time and mobility to gain a calculus with:
  - **Global synchronization**
  - **Localities**
  - **Migration**
- Two routes: we take that of adding mobility to a calculus with global synchronization.
- Be as *conservative* as possible.

# Motivation

- Masters project developed semantics for the Cashew-S web-service orchestration language [Norton, Foster and Hughes, 2005]
- Used the *Calculus for Synchrony and Encapsulation* (**CaSE**) [Norton, Lüttgen and Mendler, 2005], a conservative extension of CCS
- **Idea:** Would be interesting to extend CaSE with mobility

# Hennessy's Temporal Process Language (TPL)

- CCS with the addition of a single clock.
- Time, but not as we naively know it.
- Primary motivation is **synchronization**
- Exhibits a phenomenon known as **maximal progress**
- The clock ticks after all  $\tau$  actions.

# Timeouts

## Example

$$\lfloor E \rfloor \sigma(F)$$

- $E$  and  $F$  are processes and  $\sigma$  is a clock.
- $F$  acts if  $E$  times out on the clock  $\sigma$

# Scaling Synchronization

## Example

$$a.0 \mid \bar{a}.0$$

- Easy to do *local synchronization* in CCS – one sender, one receiver.
- But what about with an arbitrary number ( $n$ ) of processes?
- Can be done, but *not compositionally*

# The Problem

## Example

$$\bar{a}_1.\bar{a}_2.E \mid a_1.F \mid a_2.G$$

- We can model the case with two receivers fine. . .

# The Problem

## Example

$$\bar{a}_1.\bar{a}_2.\bar{a}_3.E \mid a_1.F \mid a_2.G \mid a_3.H$$

- But further composition requires rebuilding the semantics

# The Solution

## Example

$$\mu X. [\bar{o}.X] \sigma(P) \mid o.E \mid o.F \mid o.G$$

- Recursive output with the clock signal effectively the base case.
- Clock will tick when no more synchronizations can occur.

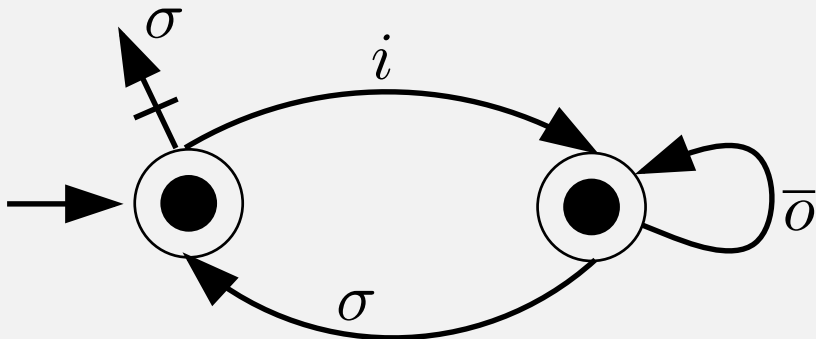
# Hierarchies

- Multiple clocks
- Arranged in hierarchies
- Accomplished via *clock hiding*

## Example

$$((E|F)/\sigma)|G$$

## Broadcast



- $\mu X.i.(\mu Y.[\bar{\sigma}. Y]\sigma(X)) + \Delta$
- Insistence is provided by the *timelock* operator,  $\Delta$

# What are Localities?

- Localities *group* a set of composed processes.
- Multitude of uses – common one is *distribution*
- Nested localities echo *clock hiding*
- We combine the two.

# A Slight Syntax Change

## Example

$$((E|F)/\sigma)|G$$

## Example

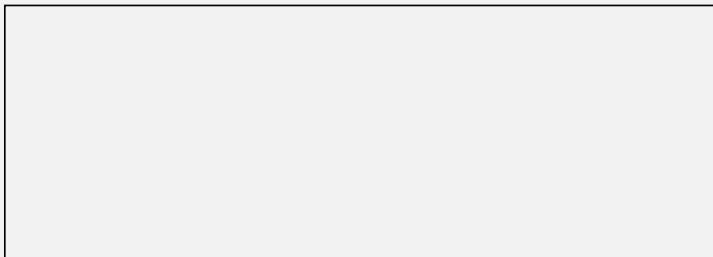
$$(I[E|F]_{\{\sigma\}})|G$$

- Consider
  - *uniqueness* of locality names
  - *structure* – one top-level locality or more?

## Merging in Ideas from Ambient-based Calculi

- We allow our localities to be moved.
- Adopt ambient-like capabilities:
  - *in n*
  - *out n*
- Expand on this to increase granularity

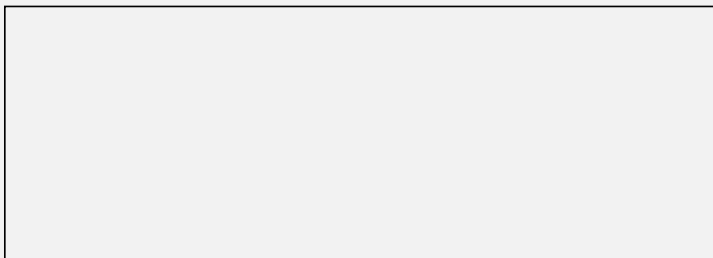
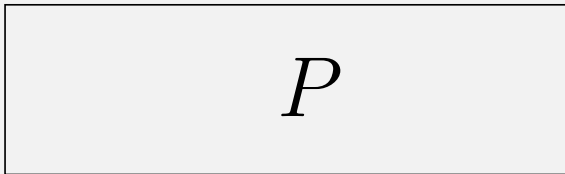
# An Example

 $n$  $m$  $in\ n.out\ n.P$

# An Example

 $n$  $m$  $out\ n.P$

# An Example

 $n$  $m$ 

# Communication Between Ambients

- Two choices:
  - 1  $open\ n$  – dissolve  $n$  from the parent
  - 2 or add the following Seal calculus primitives, as does Boxed Ambients [Bugliesi, Castagna and Crafa, 2001]:
    - $a^n$  (to child  $n$ )
    - $a^\uparrow$  (to parent)
    - Or generalize to just  $a^n$ , where  $n$  is an arbitrary locality
- Depends on the use of the model

# Using the New Calculus

- Lots of uses we can think of. . .
- Because lots of complex systems with *componentisation* and *dynamic elements*
- Hopefully feed some of this back into the Cashew project
- Useful test base

## Our Case Study: Biology

- Lots of cases of moving elements with internal synchronization
- Ambients already used in this context
- P-systems similar and imply a clock
- Interesting area to look into

## Further Points

- Mobility via value passing
- Give clocks a value for broadcast
- Typing of processes (given names, processes, clocks and localities)

# Conclusions

- Started with CaSE
- Added localities and migration
- Applications specifically web service composition and biology
- Lots of possibilities to take it even further. . .

Introduction

The Starting Point: A Calculus with Global Synchronization

Localities: The First Step to Mobility

Migration Gives Mobility

Further Thoughts and Conclusions

The End

Thanks for listening.



NORTON, B., FOSTER, S., AND HUGHES, A.

A compositional operational semantics for OWL-S.

In *Proceedings of the 2nd International Workshop on Web Services and Formal Methods (WS-FM 2005)* (September 2005), no. 3670 in LNCS, Springer-Verlag, pp. 303–317.



NORTON, B., LÜTTGEN, G., AND MENDLER, M.

A compositional semantic theory for synchronous component-based design.

In *Proceedings of the 14th International Conference on Concurrency Theory (CONCUR '03)* (2003), no. 2761 in LNCS, Springer-Verlag.



BUGLIESLI, M., CASTAGNA, G., AND CRAFA, S.

Boxed ambients.

In *(TACS '01)* (2001), vol. 2215 of LNCS, pp. 38–63.