# Communicating Transactions

C. Spaccasassi (spaccasc@tcd.ie)
TCD team: M. Hennessy, V. Koutavas
MSR supervisors: A. Gordon, N. Benton

Trinity College Dublin, Ireland
Microsoft Research, Cambridge

## Transactions in Distributed Systems

Software Transactional Memory (STM) is a relatively recent mechanism that makes concurrent programming *tractable*, *more modular and composable* [4].
Software Transactions, just like database transactions, are:

- **Atomic**
- **Consistent**
- **Isolated**
- **Durable**


*Communication* is crucial in distributed systems, such as web service mash-ups. The isolation requirement prevents communication – we can't use traditional transactions!

Communicating transactions drop the **I** but retain the **A**, **C** and **D**. This makes distributed check-pointing and automated error recovery easy.

## The Construct

Communicating Transactions introduce **two** new expressions:

$P$ is the process run at the start of transaction $k$

notifies the system that transaction $k$ can commit

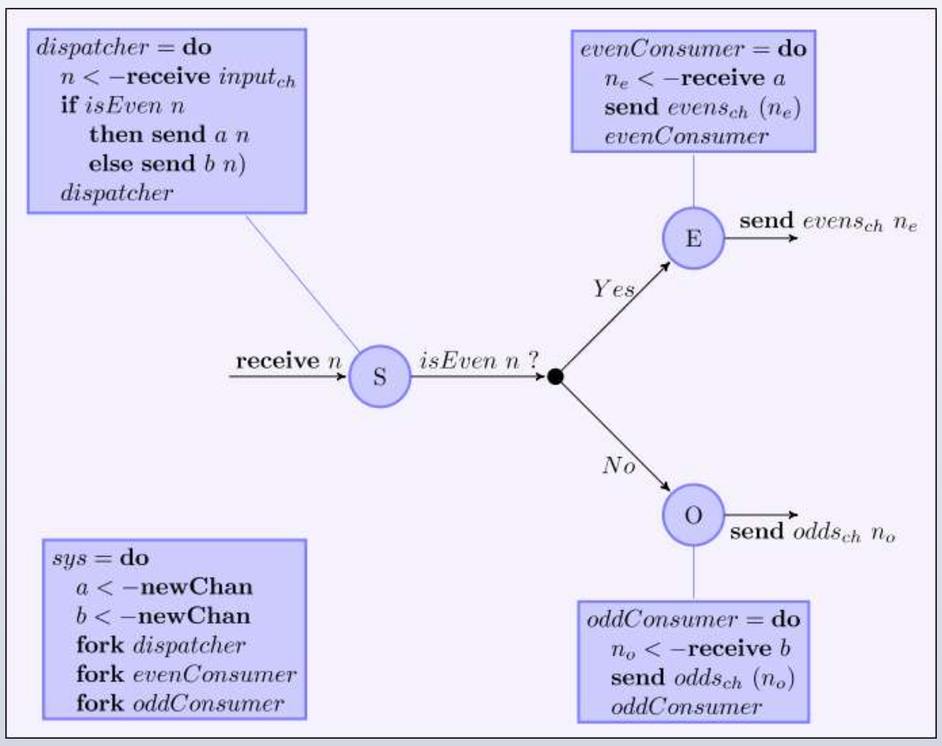
$$[\![ P \quad \triangleright_k \quad Q ]\!] \qquad \mathbf{co}\ k$$

In case of problems, $P$ is rolled-back: its effects on the environment are reversed, and process $Q$ is run instead

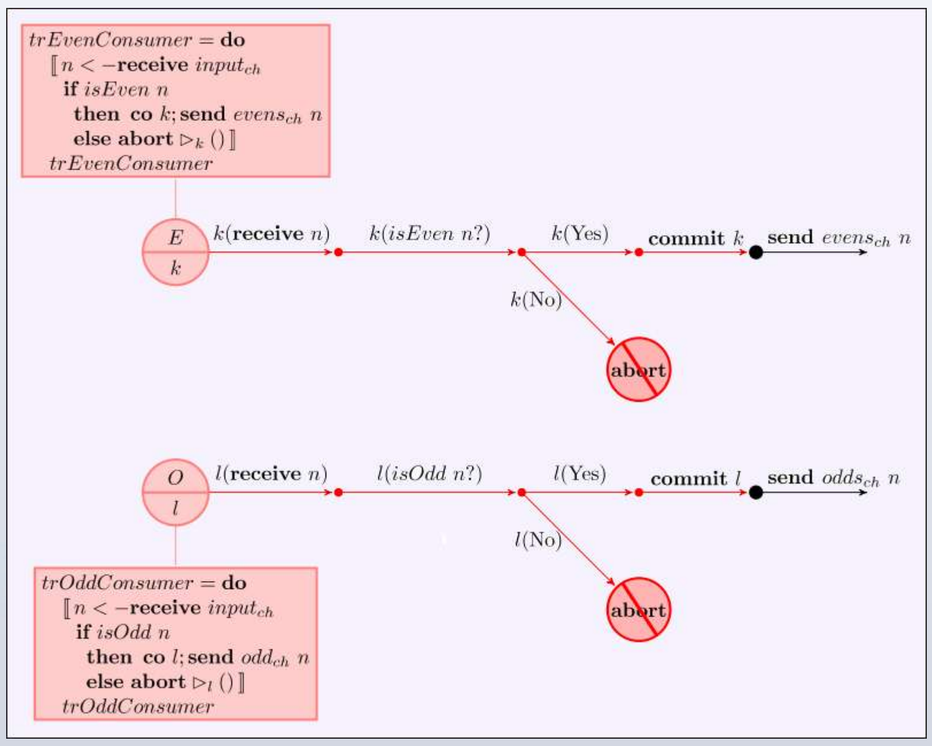

The system manages communicating transactions through ***signals***:

→ **Commit:** finalizes the transaction.
→ **Abort:** undoes all actions performed by P and their effect on the environment since the transaction started.
→ **Embed:** lets a process join the transaction. Communication is restricted to processes in the same transaction!

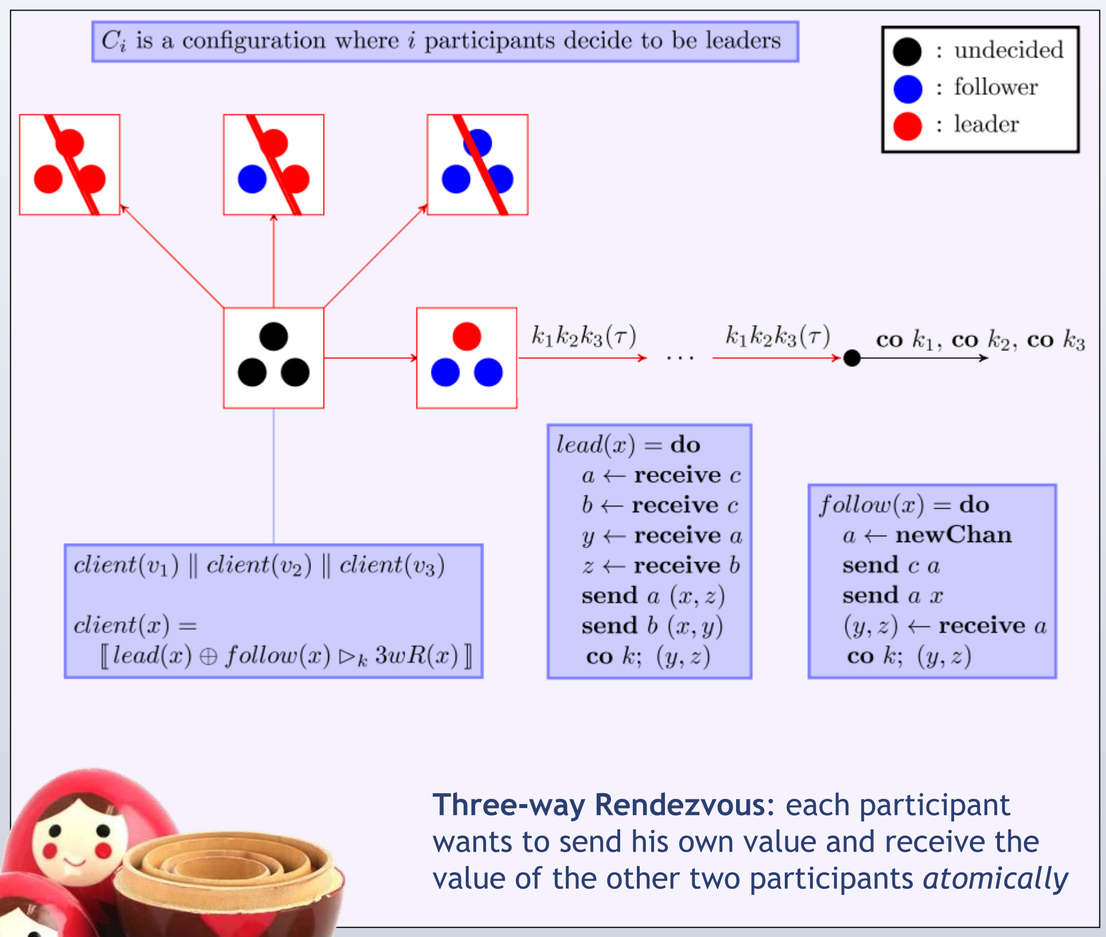
## Are We Defining The Same System?




≈




## Leader Election and Nesting




**Three-way Rendezvous:** each participant wants to send his own value and receive the value of the other two participants *atomically*

## Research Questions

- What equivalences can we devise for systems using communicating transactions?
- Can Communicating Transactions be implemented efficiently?
- Are there any useful programming patterns?
- Can we leverage Haskell's type system to create a CT monad?
- How would CT interact with Software Transactional Memory?

## Current Status

- Developed TCML, a transactional version of Concurrent ML, which provides a simple setting to study communicating transactions
- Implementing an interpreter for TCML
- Working on efficient transaction scheduling
- Investigating declarative, Prolog-like patterns for concurrent programming


## References and Contact

[1] de Vries, Koutavas, Hennessy: **Communicating Transactions,** CONCUR 2010
[2] de Vries, Koutavas, Hennessy: **Liveness of Communicating Transactions,** APLAS 2010
[3] Donnelly, Fluet: **Transactional Events**, Journal of Functional Programming 2008
[4] T. Harris, S. Marlow, S. P. Jones, M. Herlihy: **Composable Memory Transactions**, PPoPP'05
[5] Lesani, Palsberg: **Communicating memory transactions**, PPoPP 2011

My contact info: spaccasc@scss.tcd.ie
Carlo Spaccasassi, FMG, Computer Science Dept. Trinity College Dublin 2, Ireland