# Distributed Garbage Collector for Active Objects

## Acyclic and Cyclic Garbage

Guillaume Chazarain

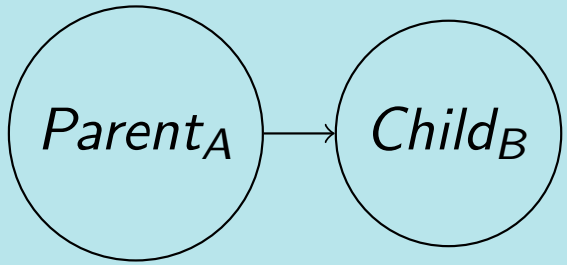
PhD Student, OASIS team

November 29, 2006

# Introduction

## Object graph

Active object A contains a reference to (a stub/proxy of) B




- Heartbeat between an active object and its children
- The heartbeat message expects a reply
- Configurable heartbeat frequency, but currently must be the same for every communicating object

# Principles

## Acyclic DGC

- No heartbeat messages for a certain amount of time implies unreachability
- Content of heartbeat message is discarded

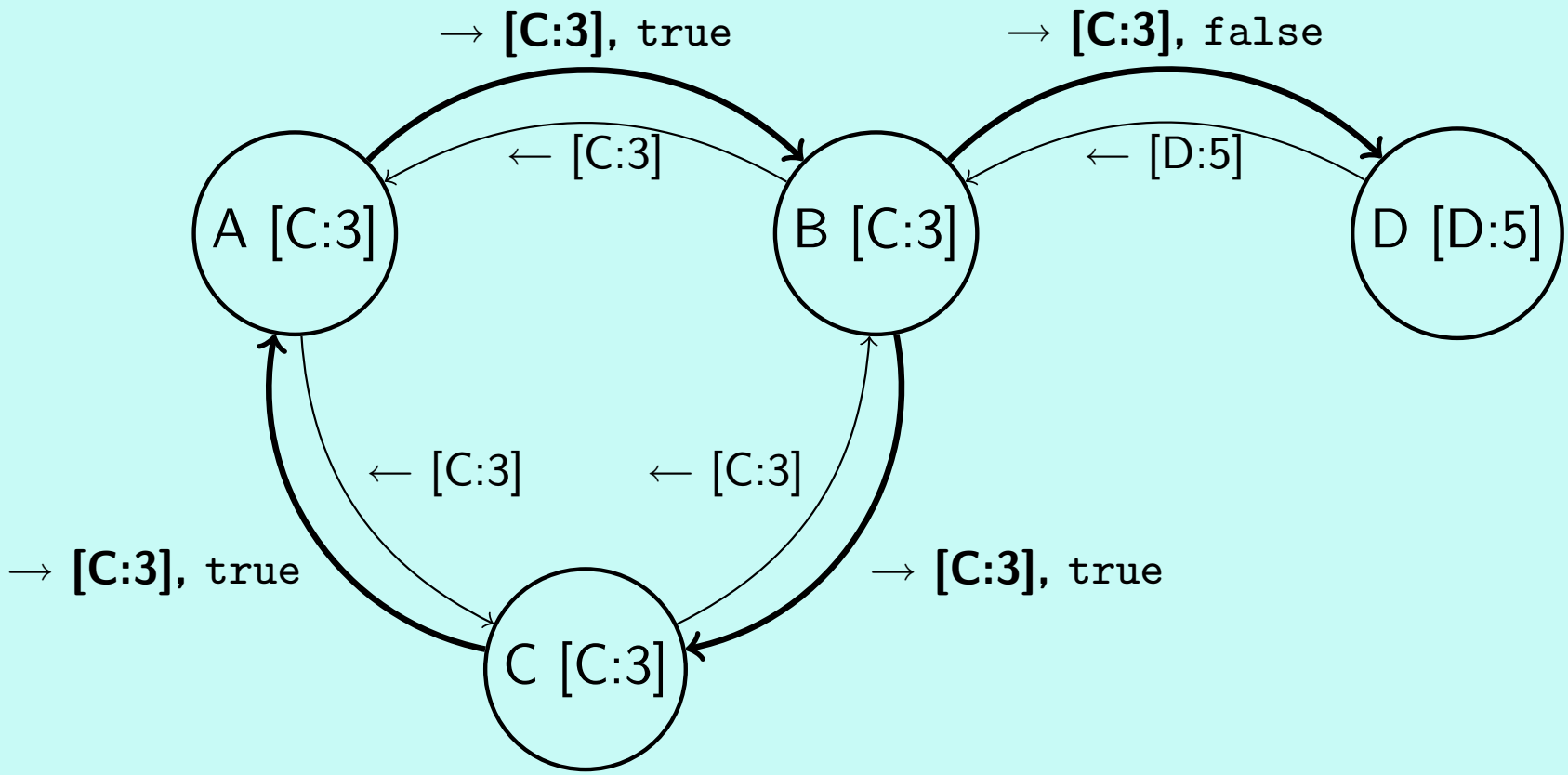
## Cyclic DGC

- Does not try to prove unreachability
- Instead, find a cycle of objects waiting for requests

# Cyclic DGC

- Lamport clock with owner (e.g.: `A:2`) incremented on
  - State change (`Busy` → `Idle`) ⇒ it may be the last activity
  - Loss of a child ⇒ it may be the parent in the spanning tree
  - Loss of a parent ⇒ it may the owner of the last activity
- Propagated through children in the heartbeat message
- Active objects attempt to make a consensus with their parents about the latest activity using the reply to the heartbeat message ⇒ Spanning tree
- If the owner of the latest activity is idle and manages to reach a consensus, it breaks the cycle

# Cyclic DGC Example




Communication in **bold** face is the heartbeat message, with its reply in normal face. The cycle (A-B-C) will make a consensus on C:3, so C will be destroyed, then A, B, D.