Interacting Process Classes

P.S. Thiagarajan

National University of Singapore

Joint with: Ankit Goel, Abhik Roychoudhury

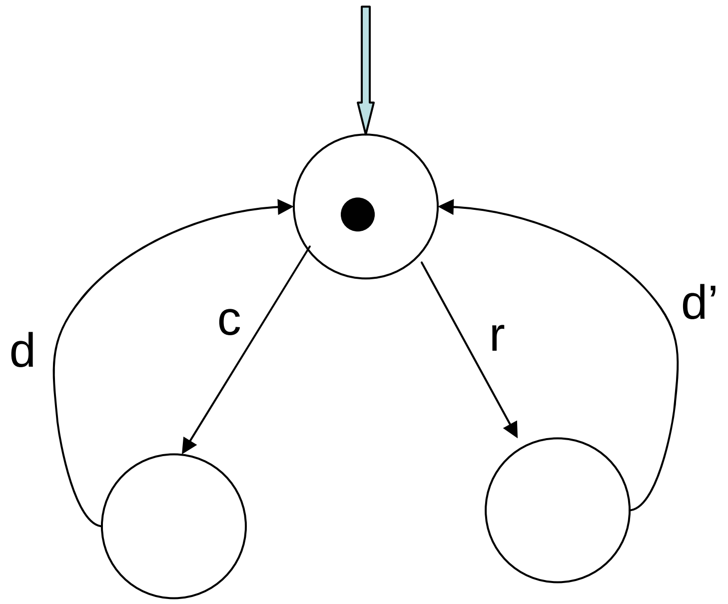
Outline

- Many reactive systems consist of ***classes of active objects*** interacting with each other.
- Active objects:
  - Phones, trains, airplanes, …
- Similar behaviors:
  - Take part in the same sequences of ***transactions***.
  - Play same ***roles*** in these transactions.

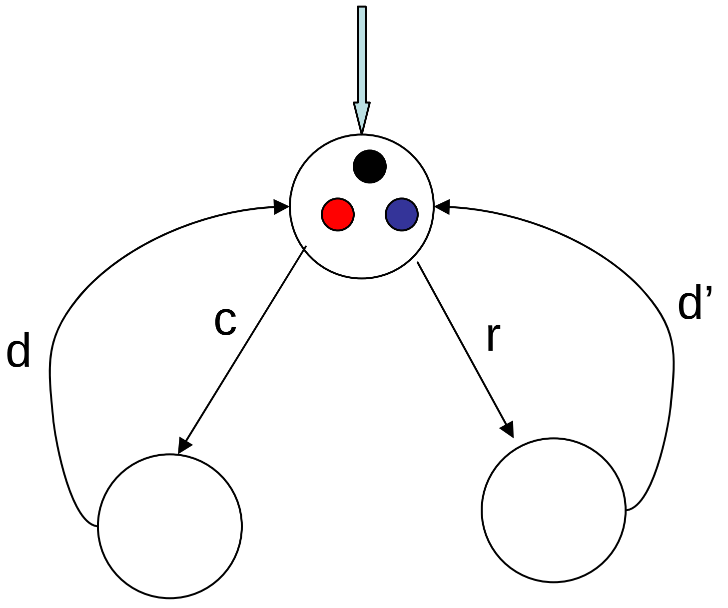
Outline

- A modeling technique using familiar notations.
- An efficient symbolic simulation technique.
  - ***Don't*** maintain a name space.
    - ***Thousands of objects in a class.***
  - ***Don't*** fix the number of objects in a class.

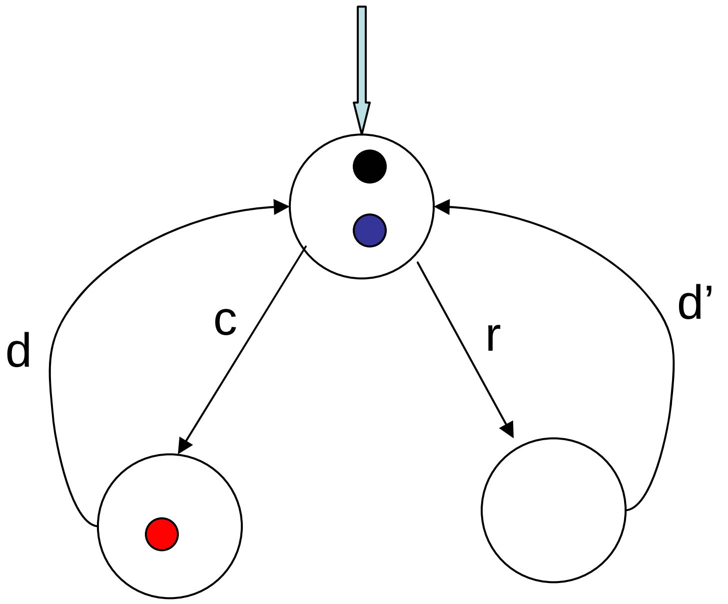
A Process



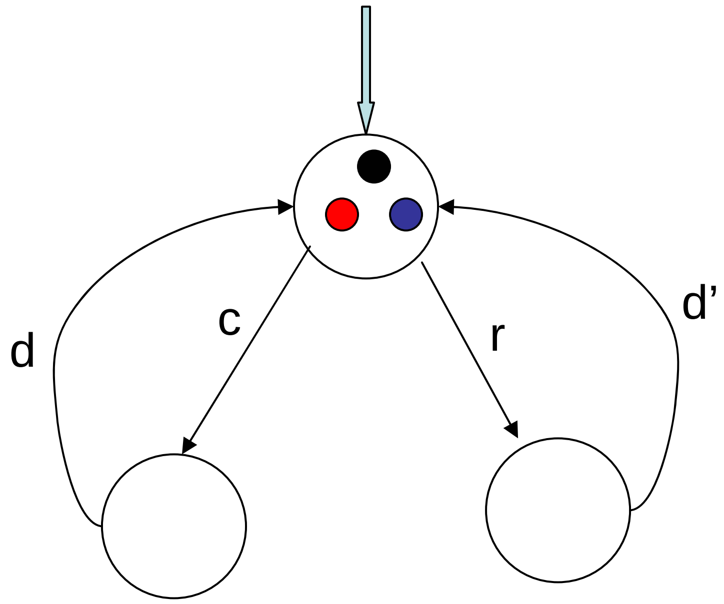
A Process Class (Multiple Instances)



A Process Class (Multiple Instances)

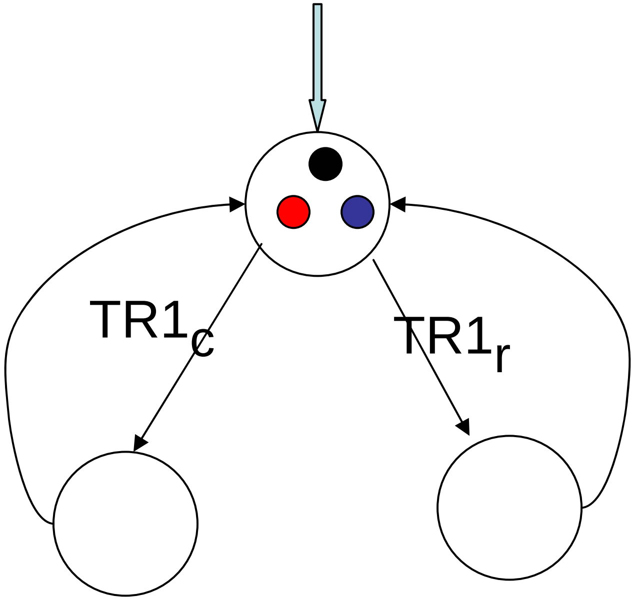


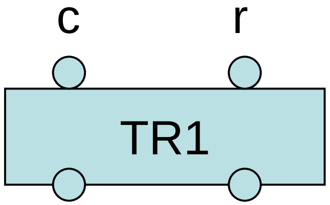
A Process Class (Multiple Instances)



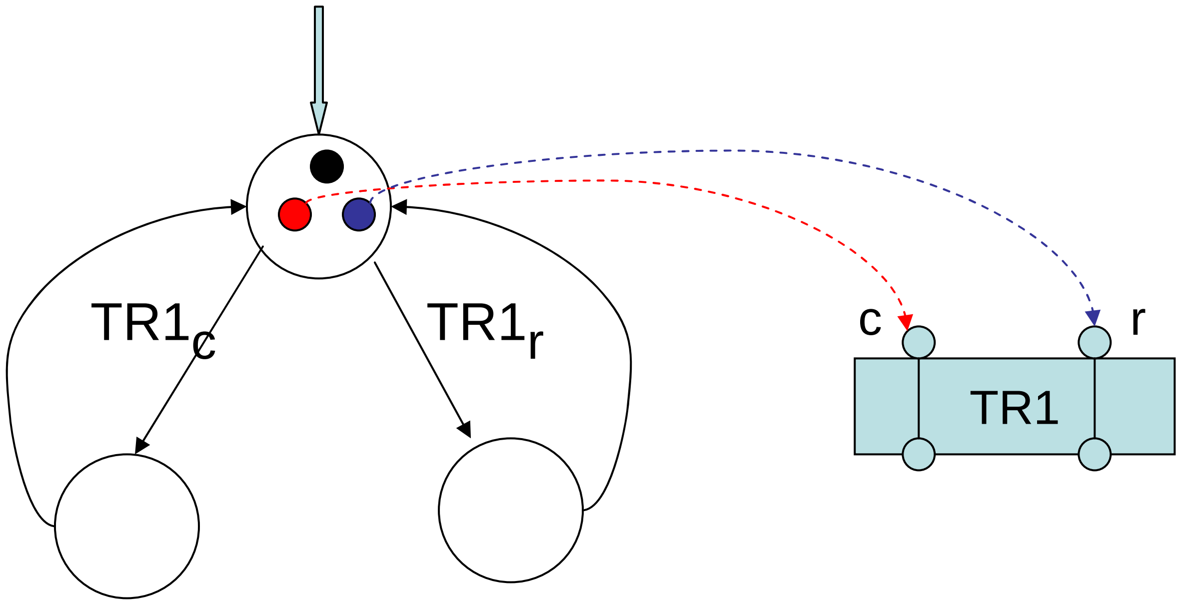
But the actions c, r, d, d’ can represent transactions between different objects of the same class.

A Process Class (Multiple Instances)

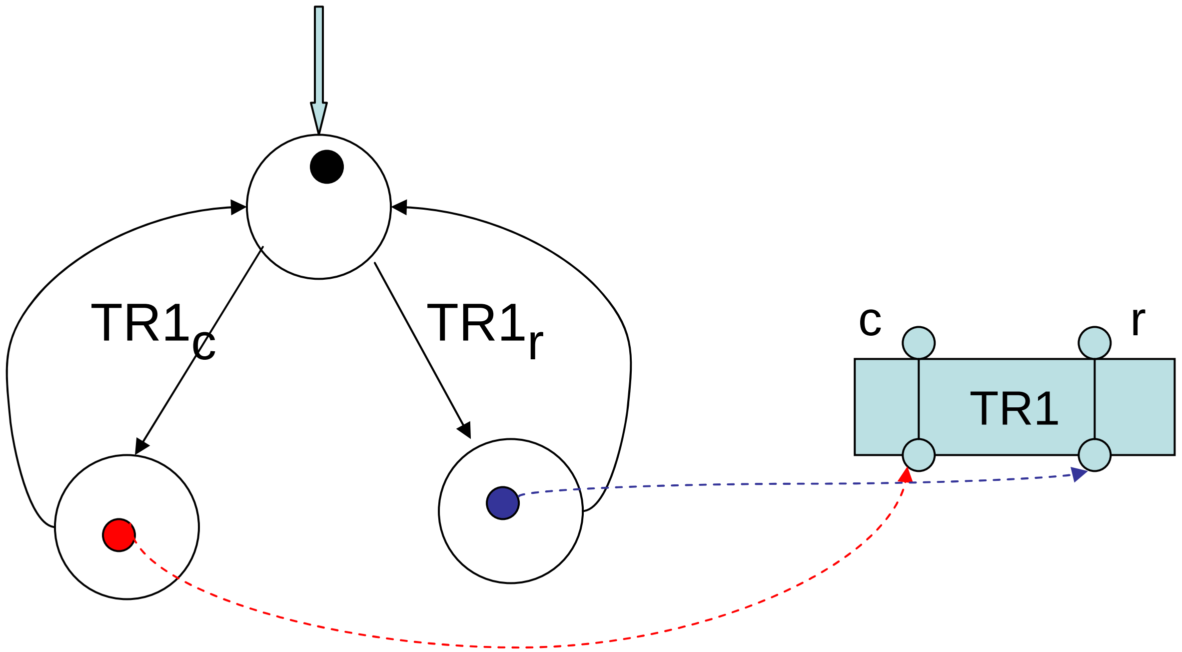




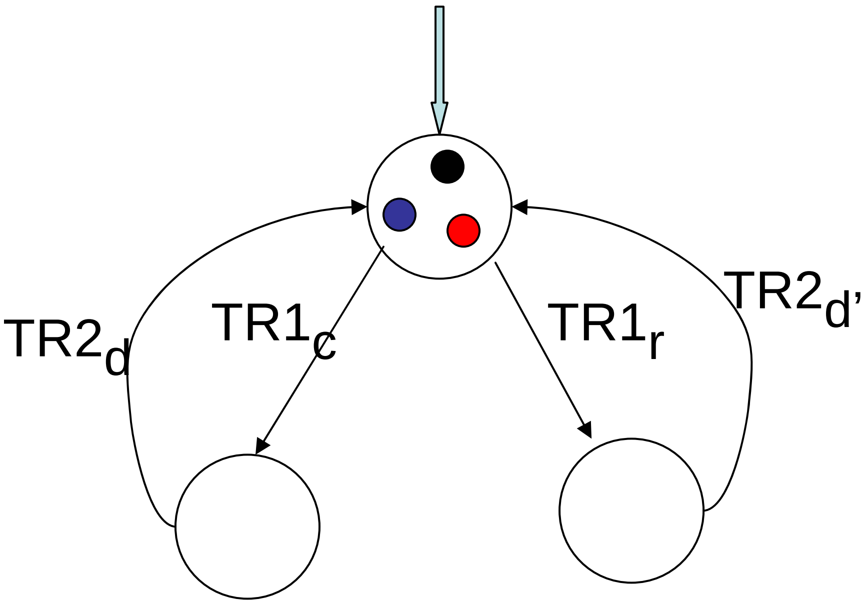
A Process Class (Multiple Instances)

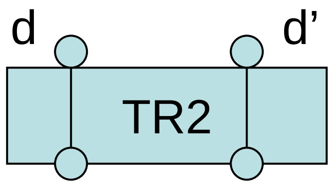


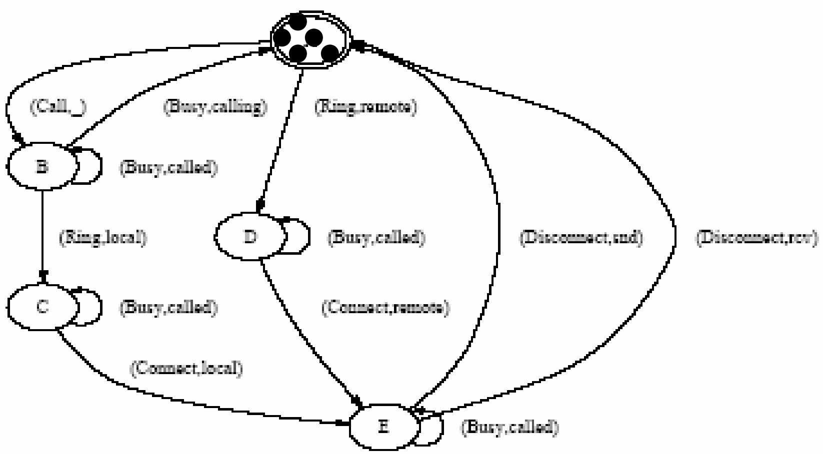
A Process Class (Multiple Instances)

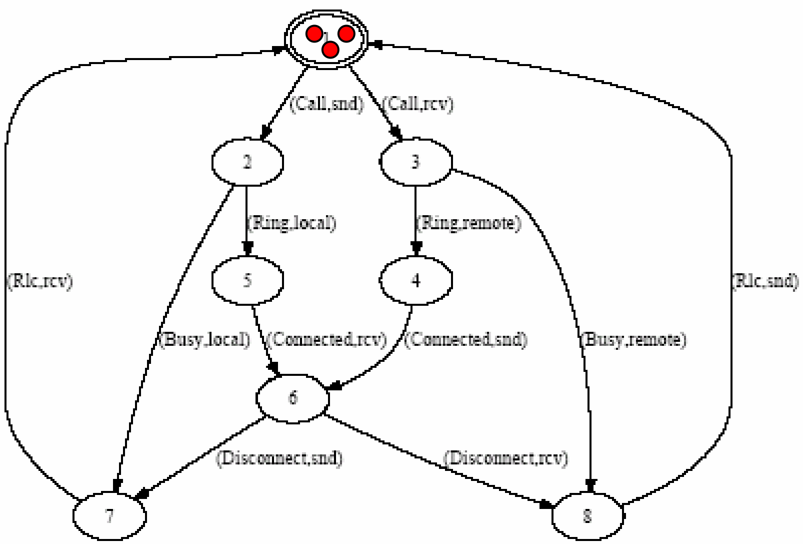


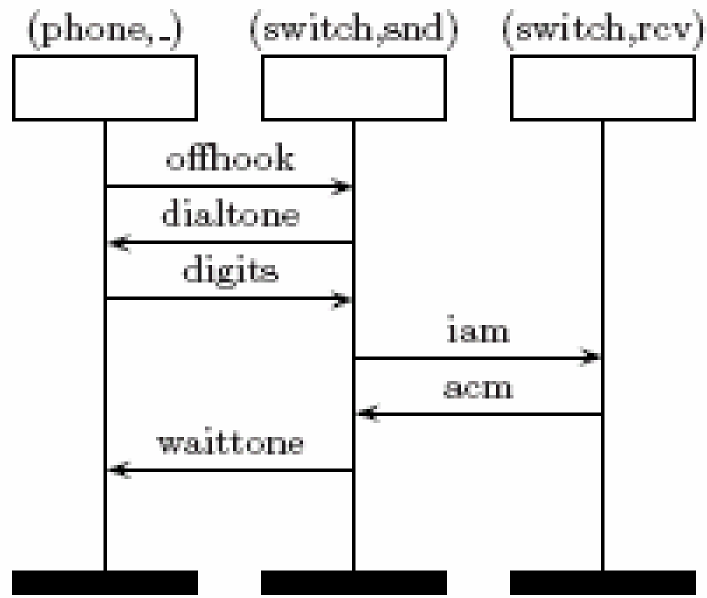
A Process Class (Multiple Instances)









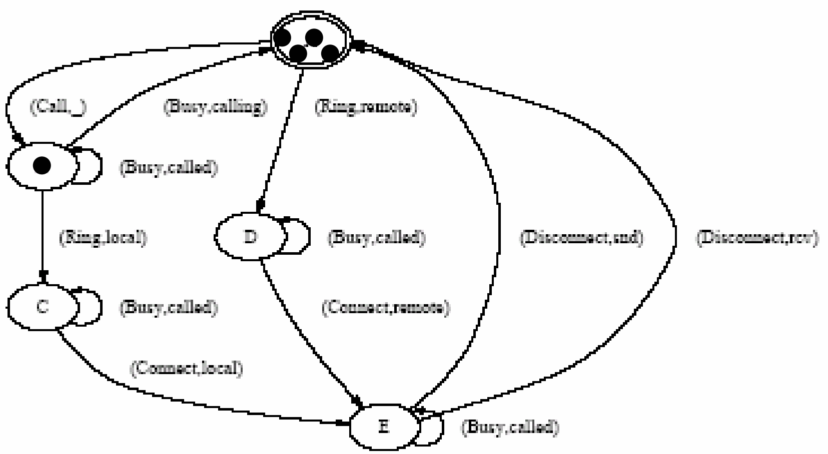


Call

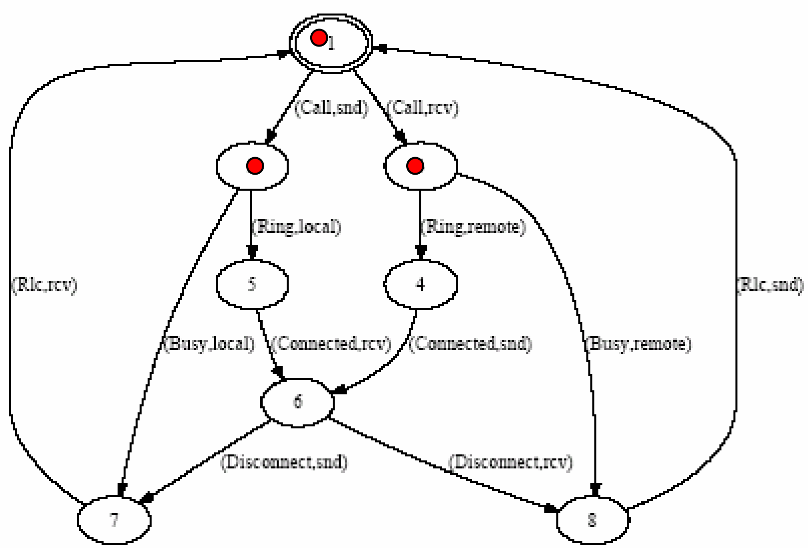
MSCs::

Depict Two way flow of information.

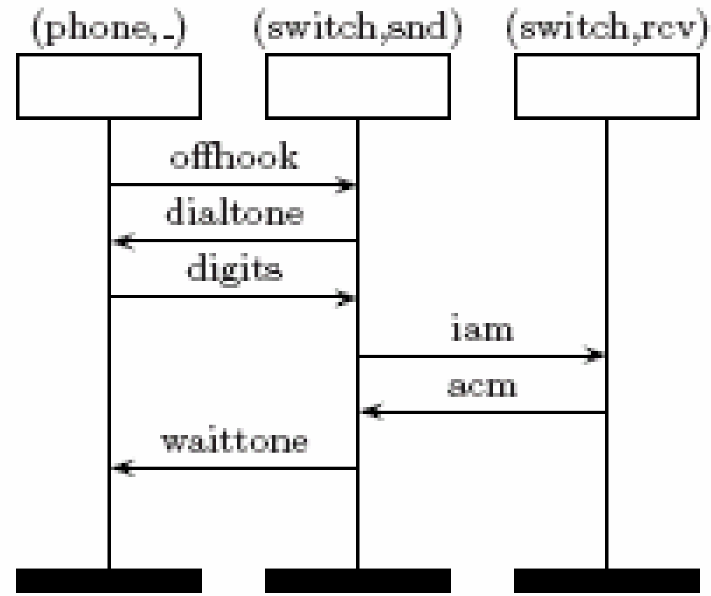
Define ***roles.***



(Call,_)
(Busy,calling)
(Ring,remote)
(Busy,called)
(Ring,local)
D
(Busy,called)
(Disconnect,snd)
(Disconnect,rcv)
C
(Busy,called)
(Connect,remote)
(Connect,local)
E
(Busy,called)



1
(Call,snd)
(Call,rcv)
(Ring,local)
(Ring,remote)
(Rlc,rcv)
5
4
(Rlc,snd)
(Busy,local)
(Connected,rcv)
(Connected,snd)
(Busy,remote)
6
(Disconnect,snd)
(Disconnect,rcv)
7
8

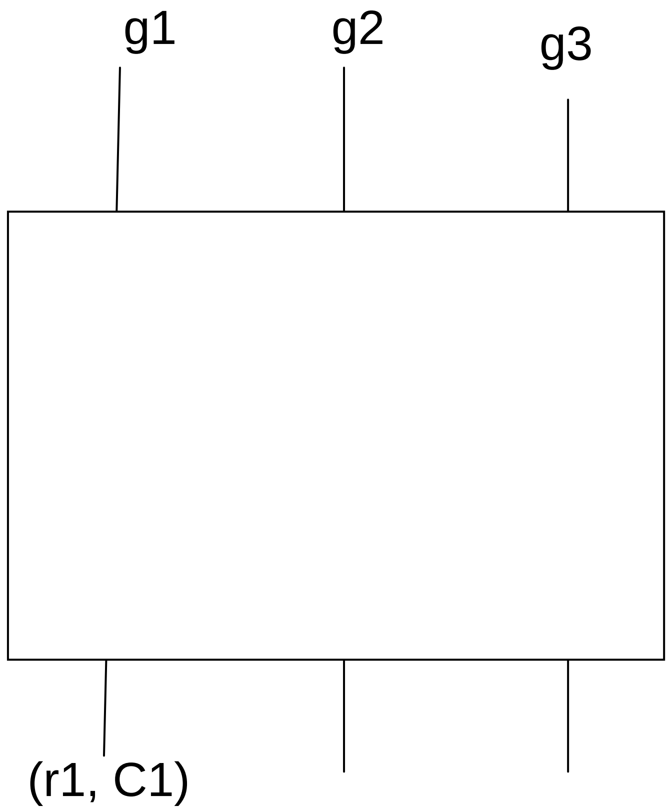


(phone,_)
(switch,snd)
(switch,rcv)
offhook
dialtone
digits
iam
acm
waittone

Call

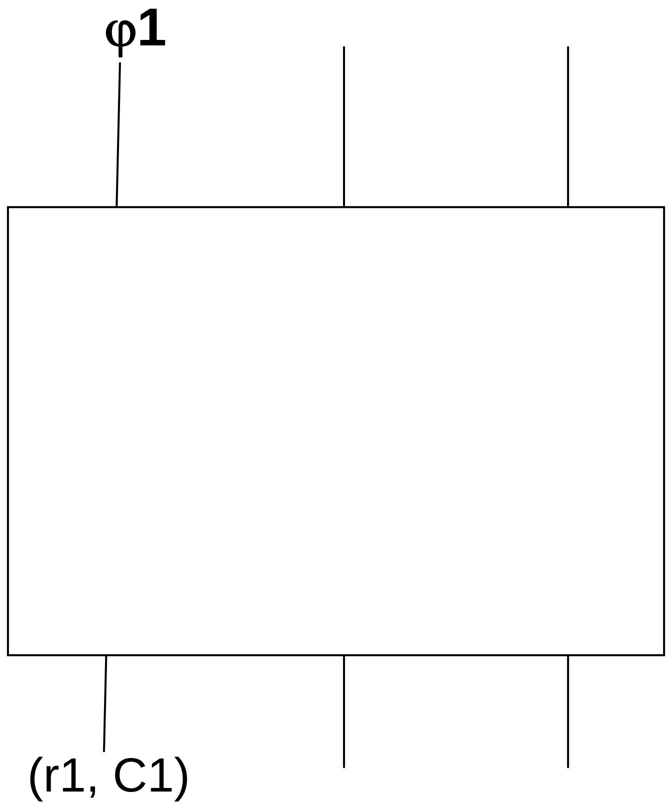
Interacting Process Classes

- Multiple process Classes
- Transactions:
  - Can involve multiple objects
    - Belonging to the same class
    - Belonging to different classes
  - Will have guards
    - Histories of the participating objects
    - States of the participating objects
      - Values of the variables of the objects.
    - Static and dynamic associations.

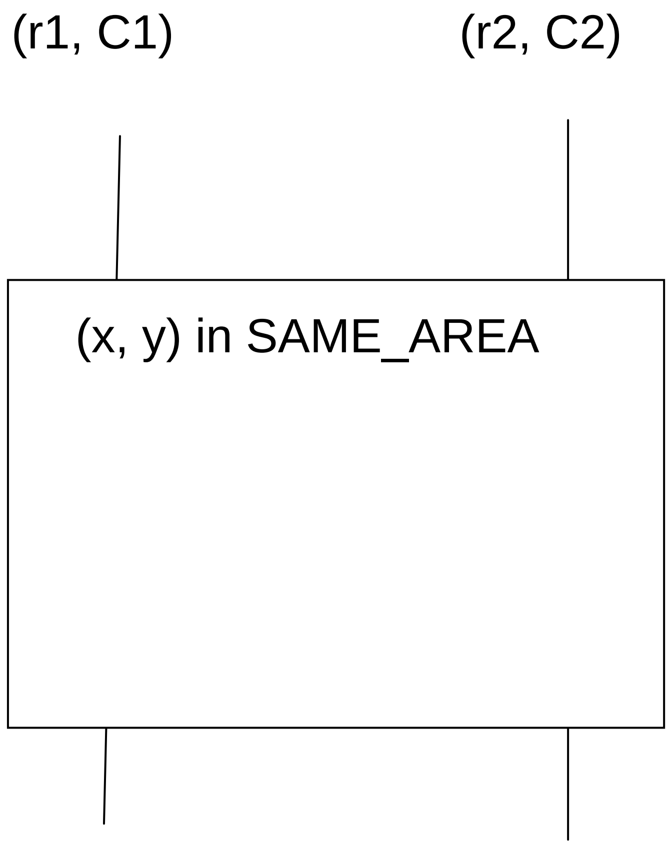


g1:

a regular expression over the local “actions” [(TR, role)] of the transition system of C1.



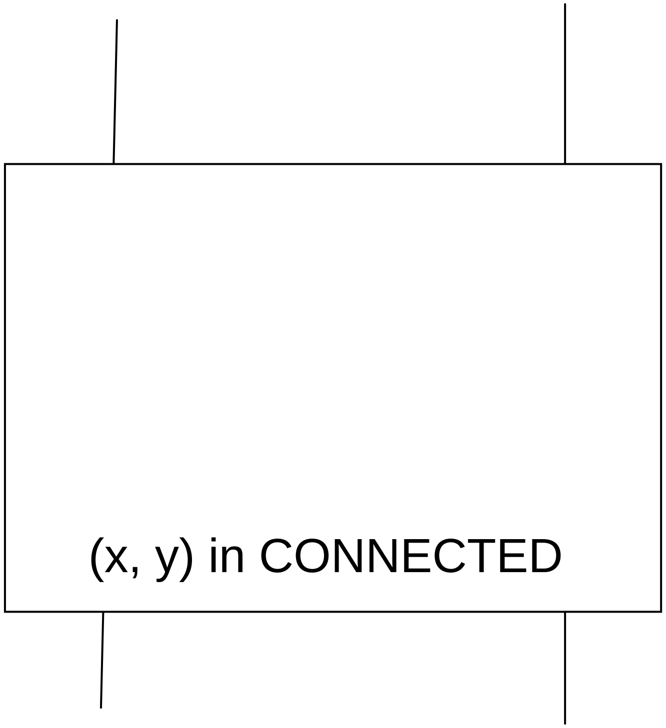
φ1: A boolean predicate over the values of the variables associated with the object in C1 chosen to play the role r1.



Local Call

Static associations capture the structural constraints.

Relations with ***fixed extensions.***

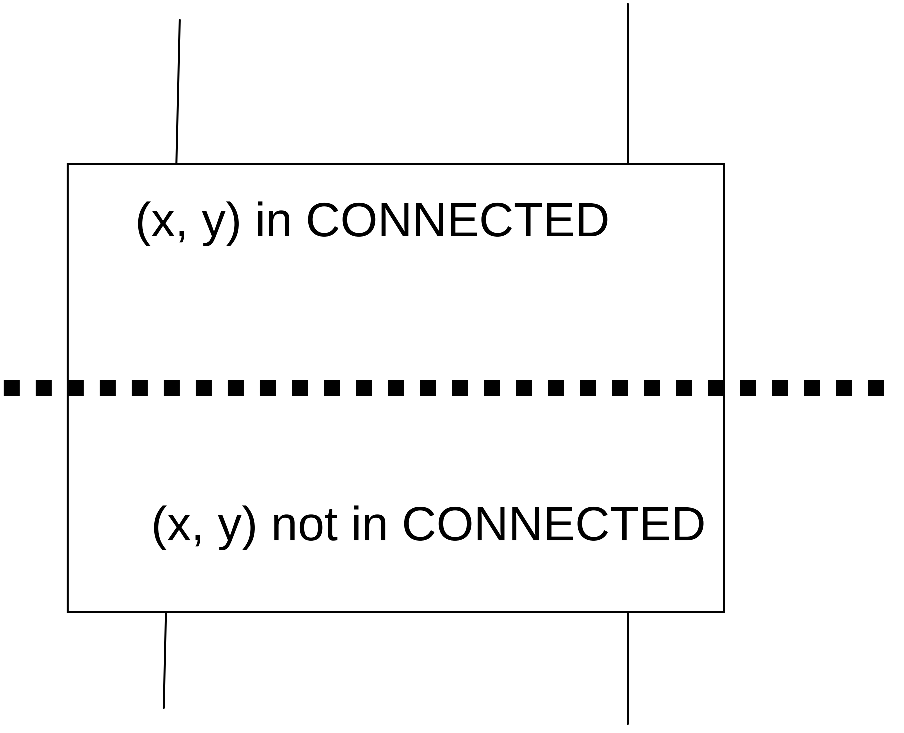


Local Call

Dynamic Associations:

Established across classes.

Relations with ***changing extensions***.



Disconnect

Symbolic Simulation

- ***Do not maintain name spaces.***
- Group the objects of a class into **behavioral subclasses**.
- Track only **the number of objects** in a behavioral subclass.
- When a transaction executes these counts will be updated
  - Behavioral subclasses get split and merged.

Symbolic simulation

- This is an (over) approximation.
- There may be spurious symbolic runs with no corresponding concrete runs.
- But one can check –not efficiently!- whether a symbolic run corresponds to a concrete run.

Current Status

- Drastically cuts simulation (resource) time/memory requirements for realistic controllers
  - CTAS weather update controller
  - Rail Shuttle system
  - Rail car system
  - Telephone switch network
- Presented at ICSE’06.

This is all very well in practice

but

What about the theory?

Research Issues

- Abstraction-based verification methods.
- What is a good **first order temporal logic** for this language?

For this workshop……

- Components ***classes***.
- Clear separation of ***computations*** and ***communications***.
- Not specific to synchronous/asynchronous.
- No timing features yet.
  - No clear separation of system/environment.