

Modeling and Specification of QLOCK in OTS/CafeOBJ

Lecture Note 05a

CafeOBJ Team for JAIST-FSSV2010

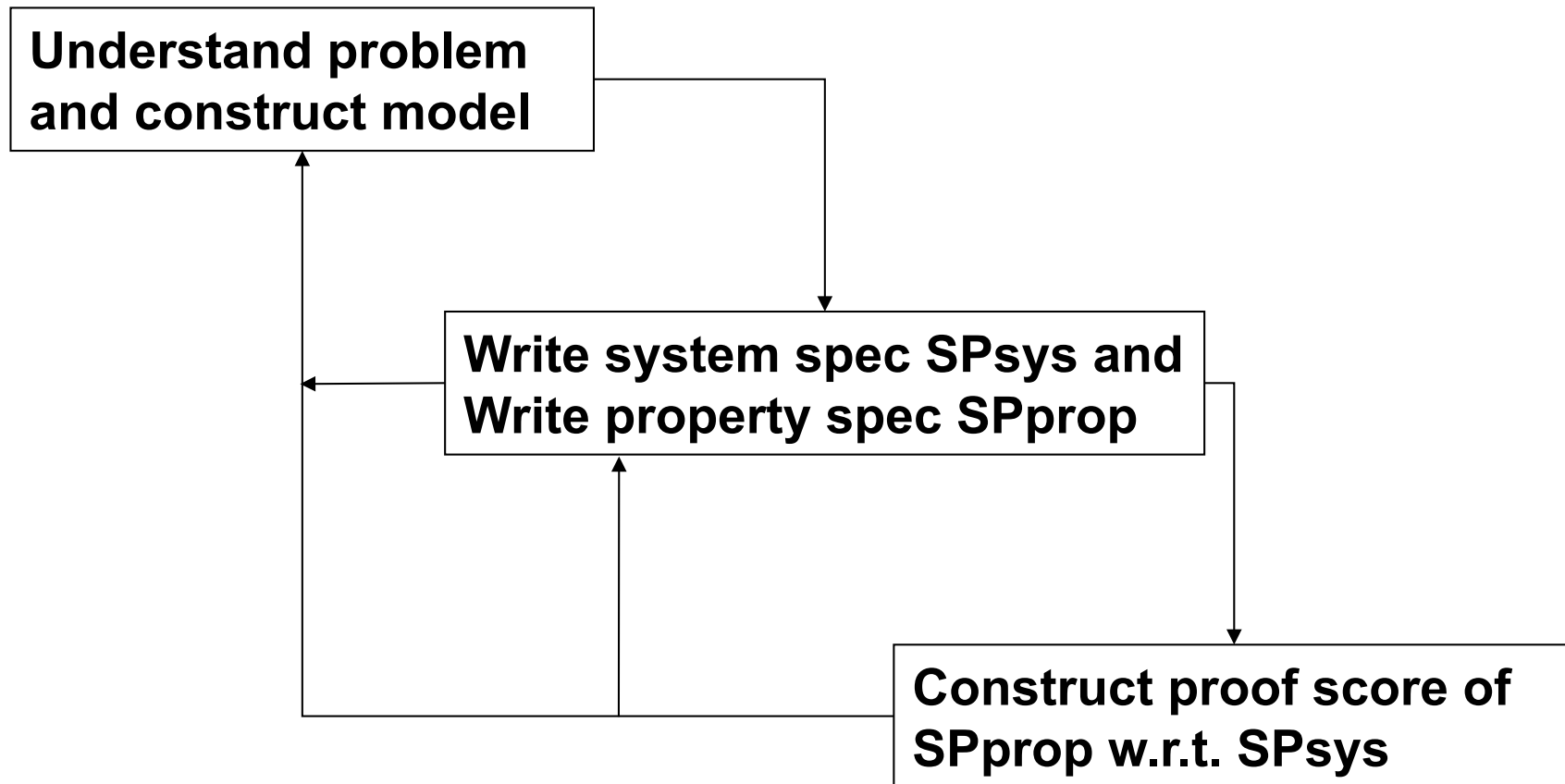
Topics

- **What is QLOCK?**
- **Modeling and Description of QLOCK in OTS**
- **Formal specification of QLOCK in CafeOBJ**
- **Formal specification of mutual exclusion property of QLOCK**

Modeling, Specifying, and Verifying (MSV) in CafeOBJ

1. By understanding a problem to be modeled/ specified, determine **several sorts of objects (entities, data, agents, states) and operations (functions, actions, events) over them** for describing the problem
2. Define the meanings/functions of the operations by declaring **equations over expressions/terms composed of the operations**
3. Write **proof scores** for properties to be verified

MSV with proof scores in CafeOBJ



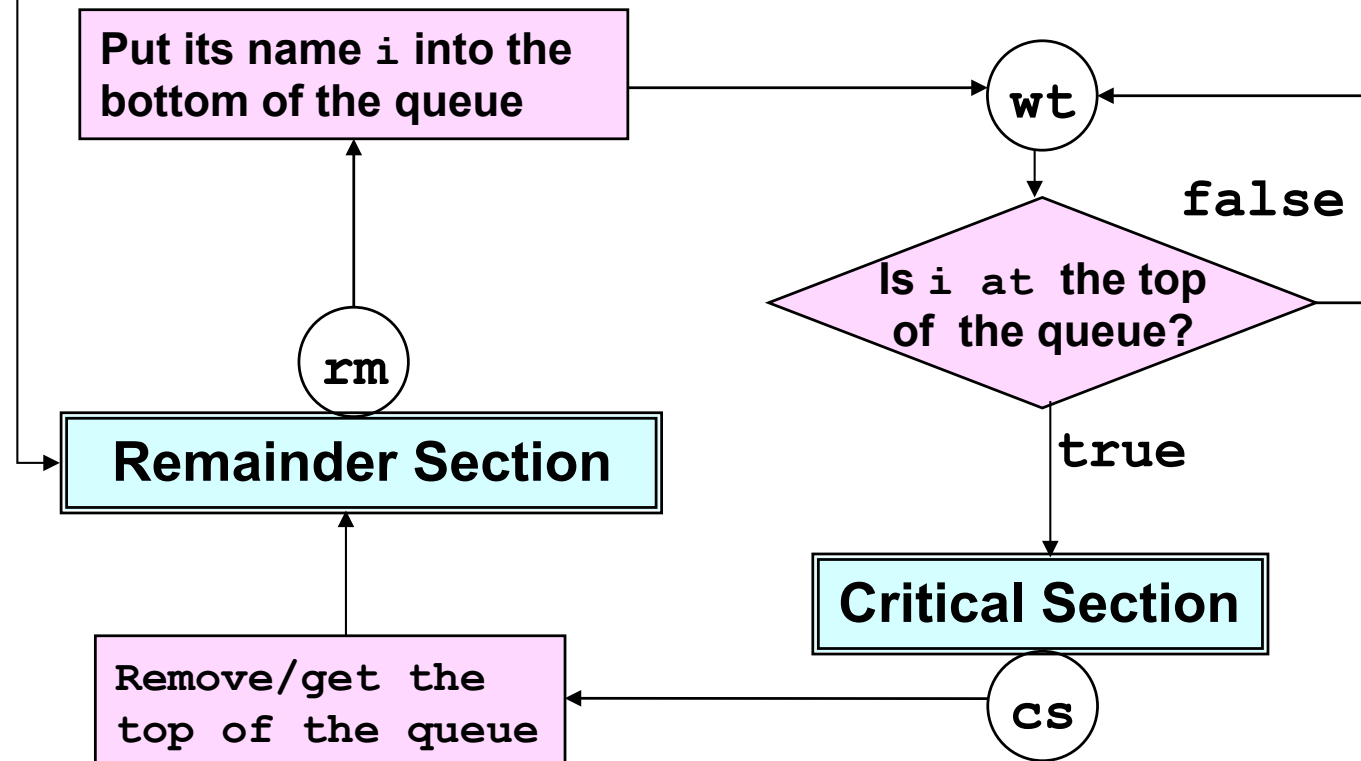
An example: mutual exclusion protocol

Assume that many agents (or processes) are competing for a common equipment, but at any moment of time only one agent can use the equipment. That is, the agents are mutually excluded in using the equipment. A protocol (mechanism or algorithm) which can achieve the mutual exclusion is called “mutual exclusion protocol”.

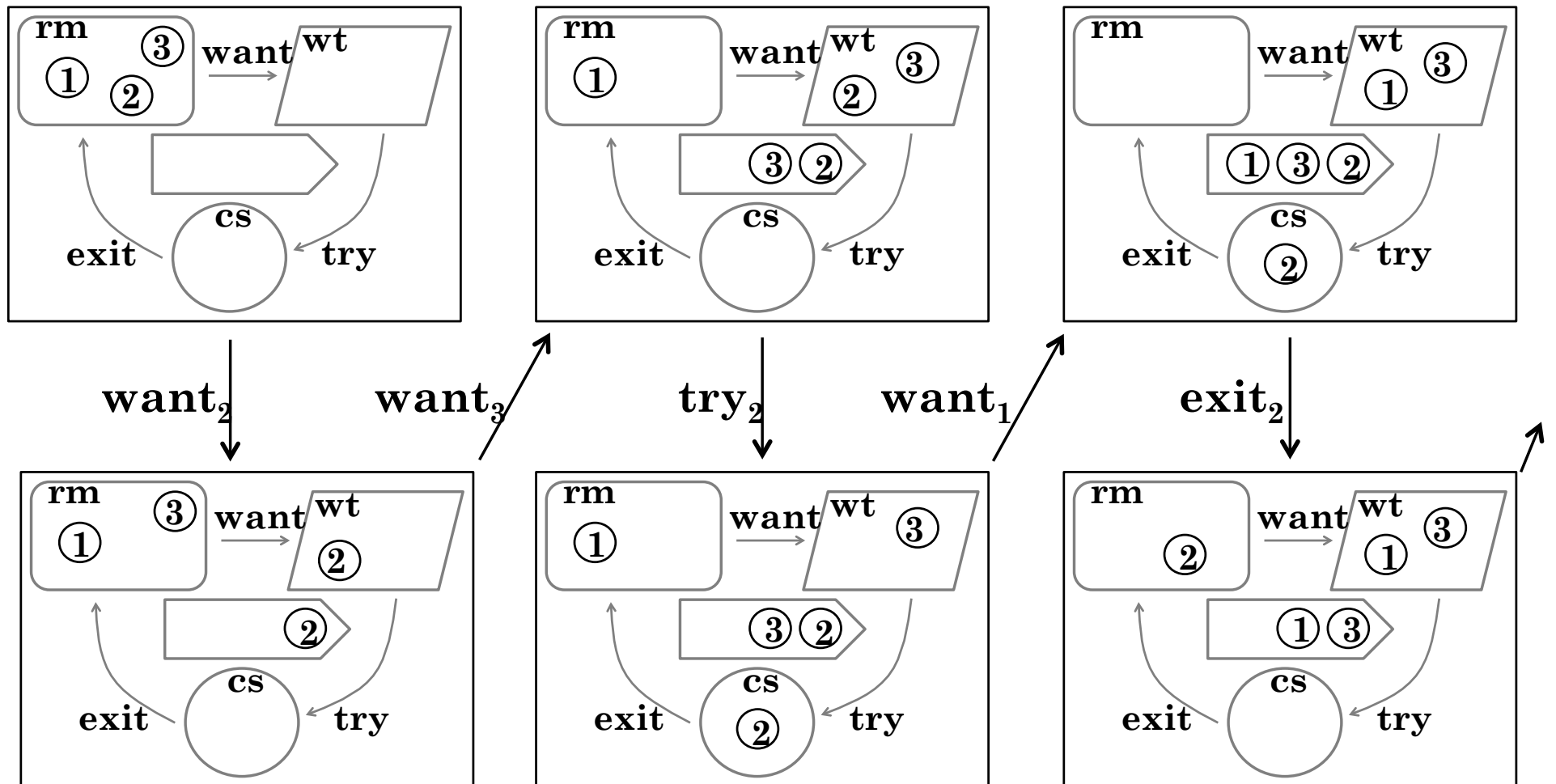
QLOCK (locking with queue): a mutual exclusion protocol

Each agent i is executing:

: atomic action



Some Scenario of Qlock

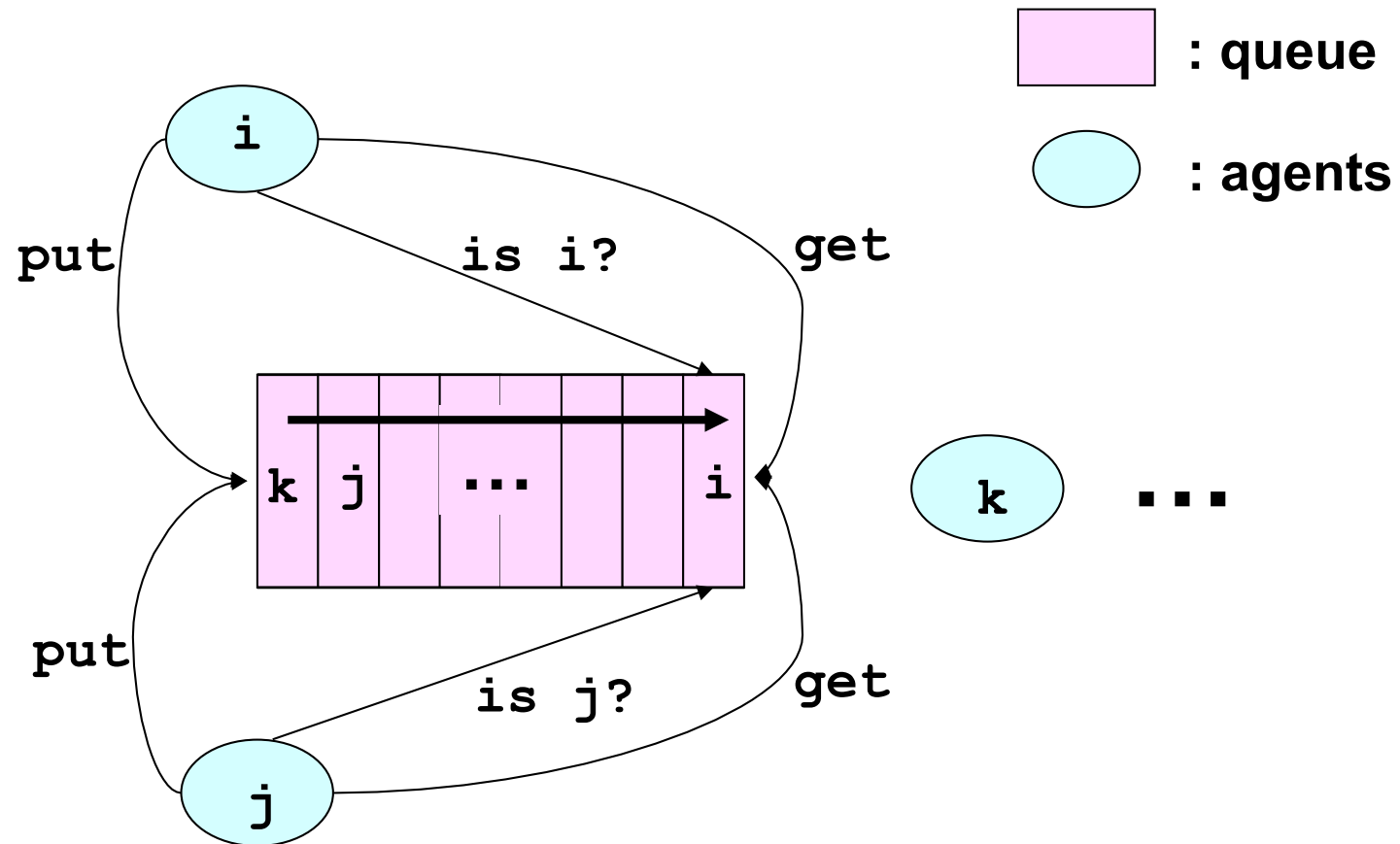


QLOCK: basic assumptions/characteristics

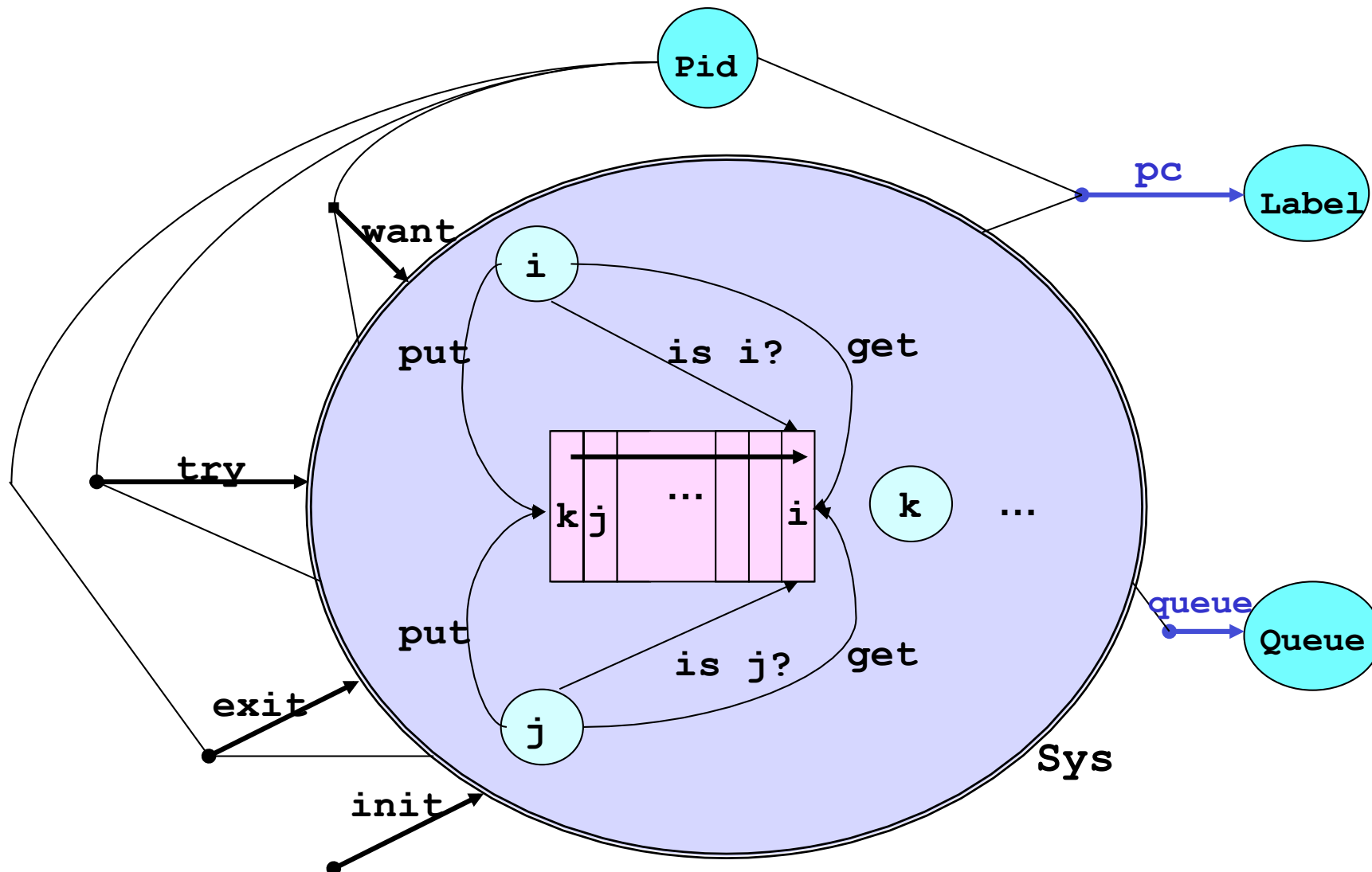
- There is only one queue and all agents/processes share the queue.
- Any basic action on the queue is inseparable (or atomic). That is, when any action is executed on the queue, no other action can be executed until the current action is finished.
- There may be unbounded number of agents.
- In the initial state, every agents are in the remainder section (or at the label rm), and the queue is empty.

The property to be shown is that at most one agent is in the critical section (or at the label cs) at any moment.

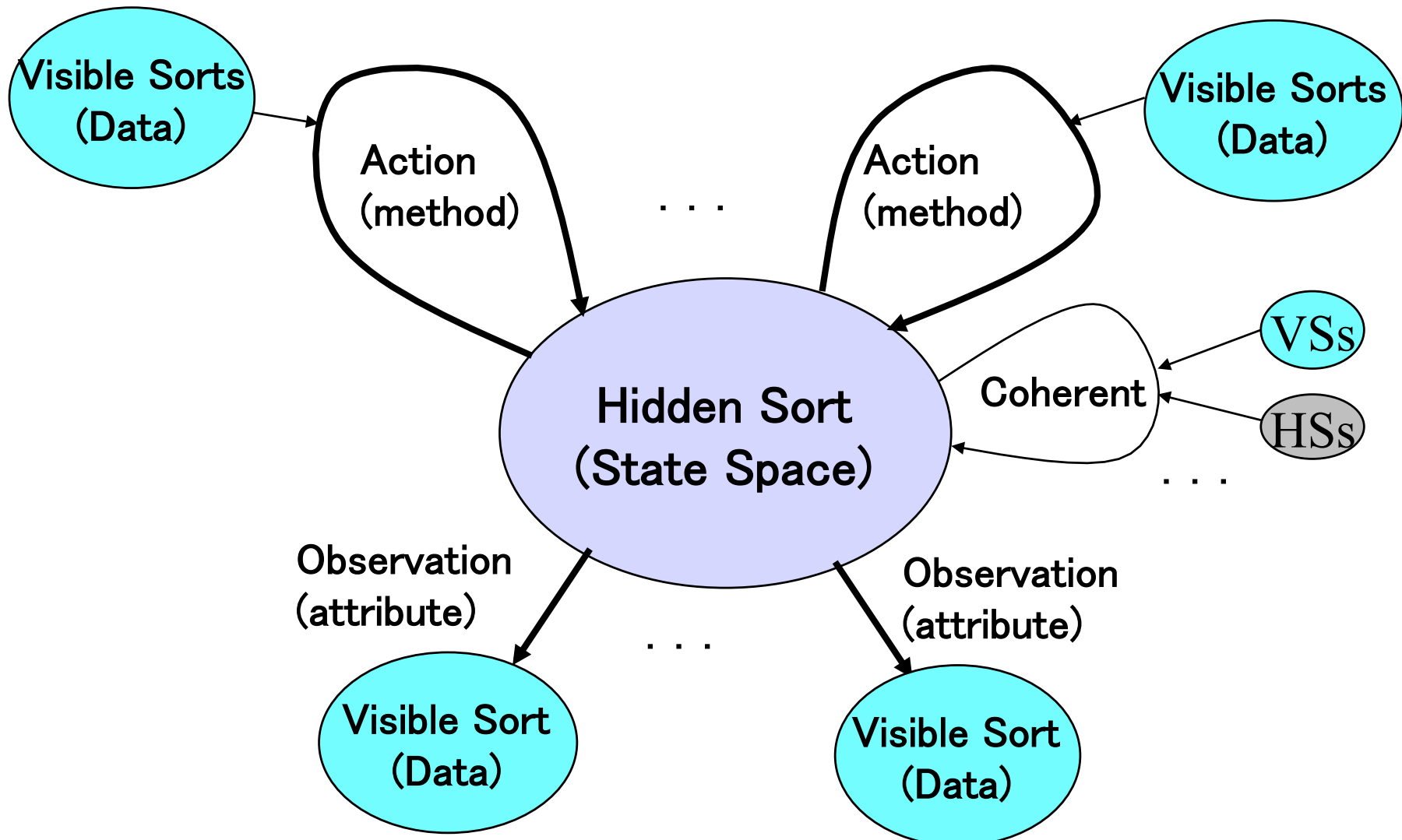
Global (or macro) view of QLOCK



Modeling QLOCK (via Signature Diagram) with OTS (Observational Transition System)



Schematic signature diagram for OTS



Signature for QLOCKwithOTS

- **Sys** is the sort for representing the state space of the system.
- **Pid** is the sort for the set of agent/process names.
- **Label** is the sort for the set of labels; i.e. {rm, wt, cs}.
- **Queue** is the sort for the queues of **Pid**
- **pc** (program counter) is an observer returning a label where each agent resides.
- **queue** is an observer returning the current value of the waiting queue of **Pid**.
- **want** is an action for agent **i** of putting its name/id into the queue.
- **try** is an action for agent **i** of checking whether its name/id is at the top of the queue.
- **exit** is an action for agent **i** of removing/getting its name/id from the top of the queue.

CafeOBJ signature for QLOCKwithOTS

```
-- state space of the system
```

```
[Sys]
```

system sort declaration

```
-- visible sorts for observation
```

```
[Queue Pid Label]
```

visible sort declaration

```
-- observations
```

```
op pc : Sys Pid -> Label
```

```
op queue : Sys -> Queue
```

observation declaration

```
-- any initial state
```

```
op init : -> Sys (constr)
```

```
-- actions
```

```
op want : Sys Pid -> Sys {constr}
```

```
op try : Sys Pid -> Sys {constr}
```

```
op exit : Sys Pid -> Sys {constr}
```

action declaration

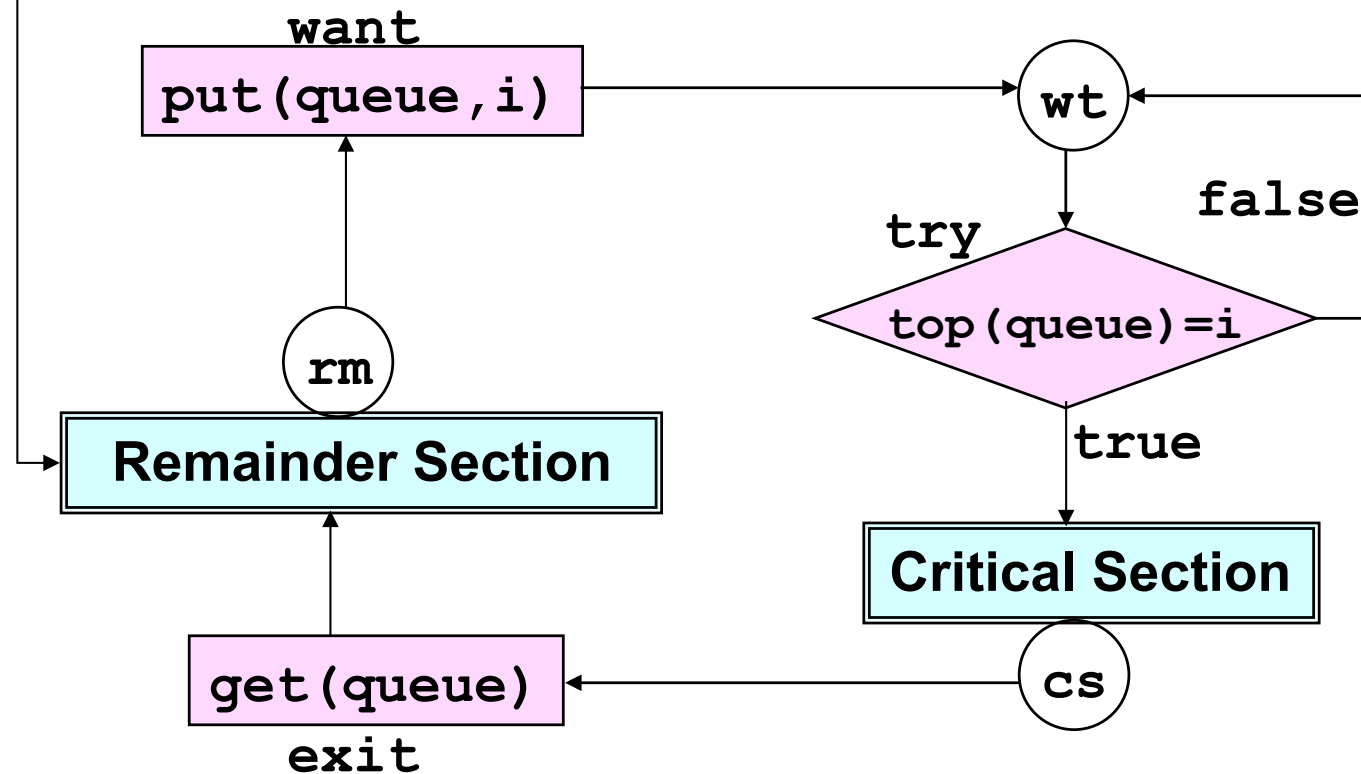
CafeOBJ Code in the file glock.mod (1)

```
mod! LABELconst
mod* LABEL
mod* PID*
mod* TRIV=
mod! QUEUE
```

QLOCK using operators in the CafeOBJ module QUEUE

Each agent i is executing:

 : atomic action



CafeOBJ Code in the file qlock.mod (2)

mod* QLOCK

($_ =^* = _$) is congruent for OTS

The binary relation ($S1 : Sys =^* = S2 : Sys$) is defined to be true iff S1 and S2 have the same observation values.

OTS style of defining the possible changes of the values of observations is characterized by the equations of the form:

$$o(a(s, d), d') = \dots o_1(s, d_1) \dots o_2(s, d_2) \dots o_n(s, d_n) \dots$$

for appropriate data values of $d, d', d_1, d_2, \dots, d_n$.

It can be shown that OTS style guarantees that ($_ =^* = _$) is congruent with respect to all actions.

R_{QLOCK} (set of reachable states) of $\text{OTS}_{\text{QLOCK}}$ (OTS defined by the module QLOCK)

Signature determining $R_{\text{QLOCK}} = \text{Sys}$

```
-- initial state
  op init : -> Sys {constr}
-- actions
  bop want : Sys Pid -> Sys {constr}
  bop try  : Sys Pid -> Sys {constr}
  bop exit : Sys Pid -> Sys {constr}
```

Recursive definition of $R_{\text{QLOCK}} = \text{Sys}$

$$R_{\text{QLOCK}} = \text{Sys} = \{\text{init}\} \cup$$
$$\{\text{want}(s, i) \mid s \in \text{Sys}, i \in \text{Pid}\} \cup$$
$$\{\text{try}(s, i) \mid s \in \text{Sys}, i \in \text{Pid}\} \cup$$
$$\{\text{exit}(s, i) \mid s \in \text{Sys}, i \in \text{Pid}\}$$

Mutual exclusion property as an invariant on $R_{\text{QLOCK}} = \text{Sys}$

```
mod* QLOCK-MEX {
  pr(QLOCK)
  -- declare a predicate to verify to be an invariant
  pred mex : Sys Pid Pid
  -- variables
  var S : Sys .
  vars I J : Pid .
  -- define mex to be the mutual exclusion property
  eq mex(S,I,J)
    = ((pc(S,I) = cs) and (pc(S,J) = cs)) implies I = J) .
}
```

Formulation of proof goal for mutual exclusion property

$\text{QLOCK-MEX} \models \forall s \in \text{Sys} \forall i, j \in \text{Pid}. \text{mex}(s, i, j)$