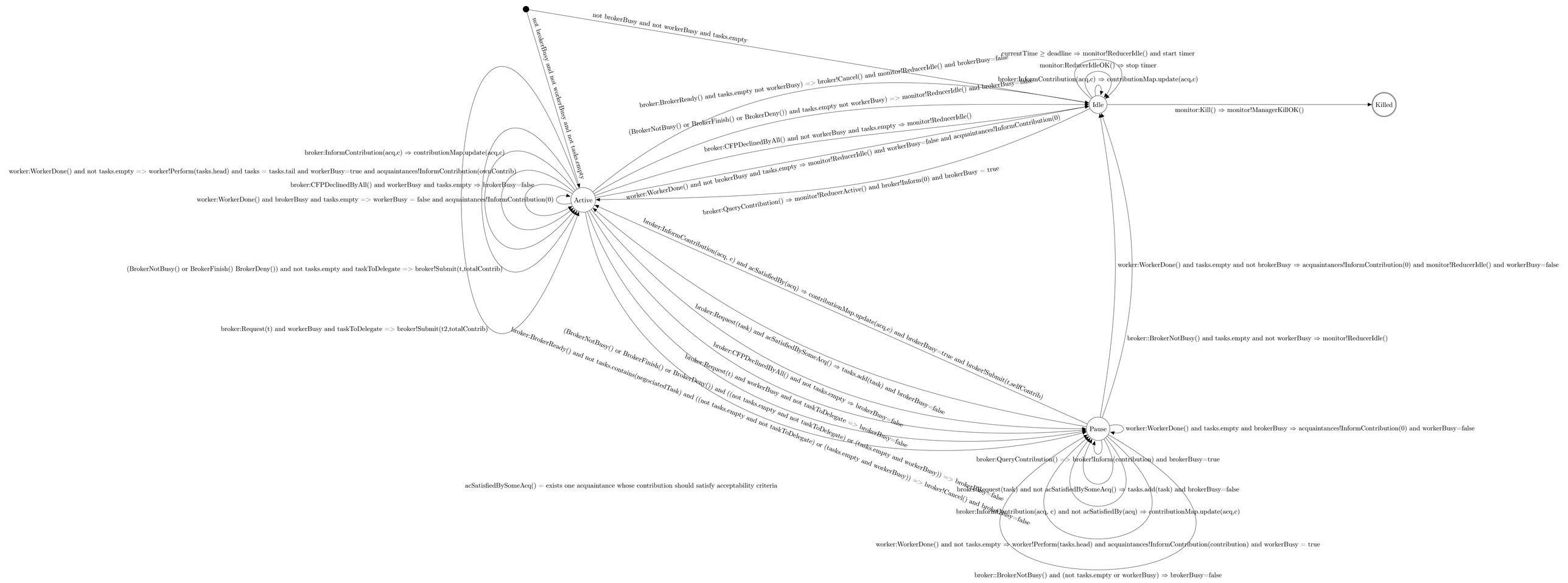
not brokerBusy and not workerBusy and tasks.empty
not brokerBusy and not workerBusy and not tasks.empty
currentTime ≥ deadline ⇒ monitor!ReducerIdle() and start timer
monitor:ReducerIdleOK() ⇒ stop timer
broker:InformContribution(acq,c) ⇒ contributionMap.update(acq,c)
broker:BrokerReady() and tasks.empty not workerBusy) => broker!Cancel() and monitor!ReducerIdle() and brokerBusy=false
(BrokerNotBusy() or BrokerFinish() or BrokerDeny()) and tasks.empty not workerBusy) => monitor!ReducerIdle() and brokerBusy=false
broker:CFPDeclinedByAll() and not workerBusy and tasks.empty ⇒ monitor!ReducerIdle()
worker:WorkerDone() and not brokerBusy and tasks.empty ⇒ monitor!ReducerIdle() and workerBusy=false and acquaintances!InformContribution(0)
broker:QueryContribution() ⇒ monitor!ReducerActive() and broker!Inform(0) and brokerBusy = true
Idle
Killed
monitor:Kill() ⇒ monitor!ManagerKillOK()
broker:InformContribution(acq,c) ⇒ contributionMap.update(acq,c)
worker:WorkerDone() and not tasks.empty => worker!Perform(tasks.head) and tasks = tasks.tail and workerBusy=true and acquaintances!InformContribution(owuContrib)
broker:CFPDeclinedByAll() and workerBusy and tasks.empty ⇒ brokerBusy=false
worker:WorkerDone() and brokerBusy and tasks.empty => workerBusy = false and acquaintances!InformContribution(0)
Active
(BrokerNotBusy() or BrokerFinish() BrokerDeny()) and not tasks.empty and taskToDelegate => broker!Submit(t,totalContrib)
broker:Request(t) and workerBusy and taskToDelegate => broker!Submit(t2,totalContrib)
broker:InformContribution(acq, c) and acSatisfiedBy(acq) ⇒ contributionMap.update(acq,c) and brokerBusy=true and broker!Submit(t,selfContrib)
broker:Request(task) and acSatisfiedBySomeAcq() ⇒ tasks.add(task) and brokerBusy=false
broker:CFPDeclinedByAll() and not tasks.empty ⇒ brokerBusy=false
broker:Request(t) and workerBusy and not taskToDelegate => brokerBusy=false
(BrokerNotBusy() or BrokerFinish() or BrokerDeny()) and ((not tasks.empty and not taskToDelegate) or (tasks.empty and workerBusy)) => brokerBusy=false
broker:BrokerReady() and not tasks.contains(negociatedTask) and ((not tasks.empty and not taskToDelegate) or (tasks.empty and workerBusy)) => broker!Cancel() and brokerBusy=false
worker:WorkerDone() and tasks.empty and not brokerBusy ⇒ acquaintances!InformContribution(0) and monitor!ReducerIdle() and workerBusy=false
broker::BrokerNotBusy() and tasks.empty and not workerBusy ⇒ monitor!ReducerIdle()
Pause
worker:WorkerDone() and tasks.empty and brokerBusy ⇒ acquaintances!InformContribution(0) and workerBusy=false
broker:QueryContribution() => broker!Inform(contribution) and brokerBusy=true
acSatisfiedBySomeAcq() = exists one acquaintance whose contribution should satisfy acceptability criteria
broker:Request(task) and not acSatisfiedBySomeAcq() ⇒ tasks.add(task) and brokerBusy=false
broker:InformContribution(acq, c) and not acSatisfiedBy(acq) ⇒ contributionMap.update(acq,c)
worker:WorkerDone() and not tasks.empty ⇒ worker!Perform(tasks.head) and acquaintances!InformContribution(contribution) and workerBusy = true
broker::BrokerNotBusy() and (not tasks.empty or workerBusy) ⇒ brokerBusy=false