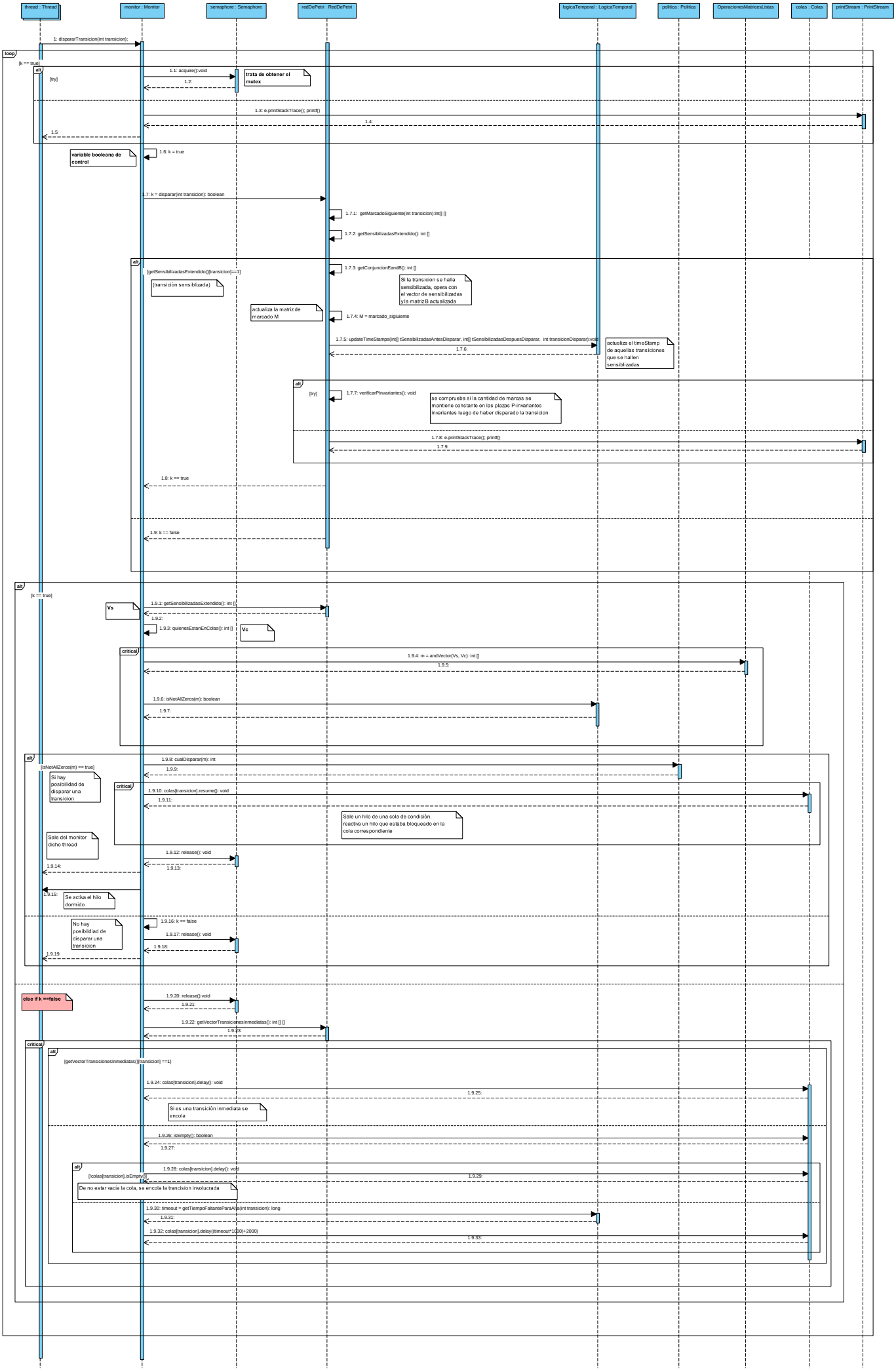
thread : Thread
monitor : Monitor
semaphore : Semaphore
redDePetri : RedDePetri
logicaTemporal : LogicaTemporal
politica : Politica
OperacionesMatricesListas
colas : Colas
printStream : PrintStream
1: dispararTransicion(int transicion);
loop
[k == true]
alt
[try]
1.1: acquire():void
1.2:
trata de obtener el mutex
1.3: e.printStackTrace(); printf()
1.4:
1.5:
variable booleana de control
1.6: k = true
1.7: k = disparar(int transicion): boolean
1.7.1: getMarcadoSiguiente(int transicion):int[] []
1.7.2: getSensibilizadasExtendido(): int []
alt
[getSensibilizadasExtendido()[transicion]==1]
(transición sensibilizada)
1.7.3: getConjuncionEandB(): int []
Si la transicion se halla sensibilizada, opera con el vector de sensibilizadas y la matriz B actualizada
actualiza la matriz de marcado M
1.7.4: M = marcado_sigiuiente
1.7.5: updateTimeStamps(int[] tSensibilizadasAntesDisparar, int[] tSensibilizadasDespuesDisparar, int transicionDisparar):void
1.7.6:
actualiza el timeStamp de aquellas transiciones que se hallen sensibilizadas
alt
[try]
1.7.7: verificarPInvariantes(): void
se comprueba si la cantidad de marcas se mantiene constante en las plazas P-invariantes invariantes luego de haber disparado la transicion
1.7.8: e.printStackTrace(): printf()
1.7.9:
1.8: k == true
1.9: k == false
alt
[k == true]
Vs
1.9.1: getSensibilizadasExtendido(): int []
1.9.2:
1.9.3: quienesEstanEnColas(): int []
Vc
critical
1.9.4: m = andVector(Vs, Vc): int []
1.9.5:
1.9.6: isNotAllZeros(m): boolean
1.9.7:
alt
[isNotAllZeros(m) == true]
Si hay posibilidad de disparar una transicion
1.9.8: cualDispararr(m): int
1.9.9:
critical
1.9.10: colas[transicion].resume(): void
1.9.11:
Sale un hilo de una cola de condición, reactiva un hilo que estaba bloqueado en la cola correspondiente
Sale del monitor dicho thread
1.9.12: release(): void
1.9.13:
1.9.14:
1.9.15:
Se activa el hilo dormido
No hay posibilidad de disparar una transicion
1.9.16: k == false
1.9.17: release(): void
1.9.18:
1.9.19:
else if k ==false
1.9.20: release():void
1.9.21:
1.9.22: getVectorTransicionesInmediatas(): int [] []
1.9.23:
critical
alt
[getVectorTransicionesInmediatas()[transicion] ==1]
1.9.24: colas[transicion].delay(): void
1.9.25:
Si es una transición inmediata se encola
1.9.26: isEmpty(): boolean
1.9.27:
alt
[!colas[transicion].isEmpty()]
1.9.28: colas[transicion].delay(): void
1.9.29:
De no estar vacía la cola, se encola la trancision involucrada
1.9.30: timeout = getTiempoFaltanteParaAlja(int transicion): long
1.9.31:
1.9.32: colas[transicion].delay((timeout*1000)+2000)
1.9.33: