2. Les lecteurs et les rédacteurs

Interface	Conditons d'acceptation	Variables d'état	Prédicat
Demander_Lecture DL	pas d'écriture en cours	nbLecteurs : nat (nL)	nR = 0
Terminer_Lecture TL	---		true
Demander_Ecriture DE	pas de lecture ni d'écriture	nbRedacteur : nat (nR)	nL = 0 ^ nR = 0
Terminer_Ecriture TE	---	nbRedAtt : nat	true

Code d'une activité (bon comportements) : `((DL; TL) + (DE; TE))*`

Invariants

inv nR ≤ 1 ^ (nL = 0 v nR = 0)

Variables Conditions

AccèsLecture
AccèsEcriture

Codage

DL

```
si ¬(nR = 0 ^ nbRedAtt = 0) alors
        AccèsLecture.wait
finSi
nL++
{nR = 0 ^ nL >= 1}
AccèsLecture.signal   (réveil en chaine)
```

TL

```
{nR = 0  ^ nL >= 1}
nL--
si nL = 0 alors
        {nL = 0 ^ nR = 0}
        AccèsEcriture.signal
finSi
```

DE

```
si ¬(nL = 0 ^ nR = 0) alors
        nbRedAtt++
        AccèsEcriture.wait
        nbRedAtt--
finSi
{nL = 0 ^ nR = 0}
nR++
```

TE

```
{nL = 0 ^ nR = 1}
nR--
{nL = 0 ^ nR = 0}
si nbRedAtt > 0 alors
        AccèsEcriture.signal
sinon
        AccèsLecture.signal
finSi
```

Stratégie FIFO

Méthodologie pour FIFO :

- suivre méthodologie classique
- mais une seule variable condition (FIFO)
- identifier les bugs
- bidouiller

Interface	Conditons d'acceptation	Variables d'état	Prédicat
DL	pas d'écriture en cours	nbLecteurs : nat (nL)	nR = 0
TL	---		true
DE	pas de lecture ni d'écriture	nbRedacteur : nat (nR)	nL = 0 ˆ nR = 0
TE	---	(nAtt : nat)	true

Invariants

inv nL = 0 v nR = 0 ; nR ≤ 1
inv nAtt > 0 => nL > 0 v nR > 0

Variable Condition

Accès (FIFO)
Sas

Codage

DL

```
si ¬(nR = 0 ^ Accès.empty) alors
        AccèsLecture.wait
finSi
{nR = 0}
nL++
{nR = 0 ^ nL >= 1}
Accès.signal   (*réveil en chaine*)
```

TL

```
{nR = 0  ^ nL >= 1}
nL--
si nL = 0 alors
        {nL = 0 ^ nR = 0}
        Accès.signal
finSi
```

DE

```
si ¬(nL = 0 ^ nR = 0) alors
        Accès.wait
        si nL > 0 alors
                Sas.wait
        finSi
finSi
{nL = 0 ^ nR = 0}
nR++
```

TE

```
{nL = 0 ^ nR = 1}
nR--
{nL = 0 ^ nR = 0}
Accès.signal
finSi
```

Vérification

vérifier que chaque préconditions à `VC.signal` *=> postconditions* `VC.wait`

- 2 => 1 : ✓
- 2 => 4 : ×
- 3 => 1 : ✓
- 3 => 4 : ✓
- 5 => 1 : ✓

```
si Sas.empty alors
        Accès.signal
sinon
        Sas.signal
finSi
```

<- retour

#TD/SC