Corrigé-type.

QC

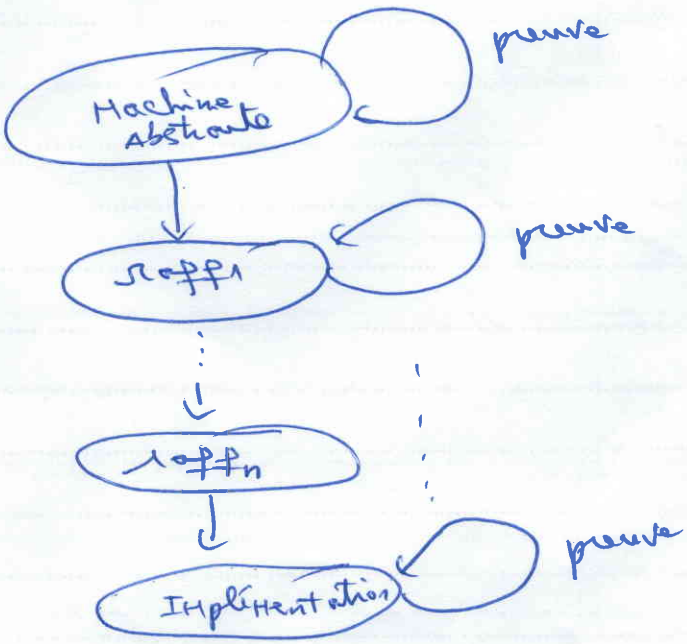
①

Approche	principe	Exemple
- Algébrique	- Définit une algèbre d'opérations sur un type abstrait de données. Sémantique : paradigme fonctionnel	Monde (Module fonctionnel
- basé état	- Définit l'ens des états admissibles sémantique : Pré/Post-Condition	B
- basé transition	- Définit l'ens des transitions admissibles sémantique : logique de réécriture	Monde (module système)
- basé historique	- Définit l'ens des instants temporels admissibles dans le futur sémantique = logique temporelle	LTL.

② principe de la méthode B.

→ raff. successifs

→ correction par construction.



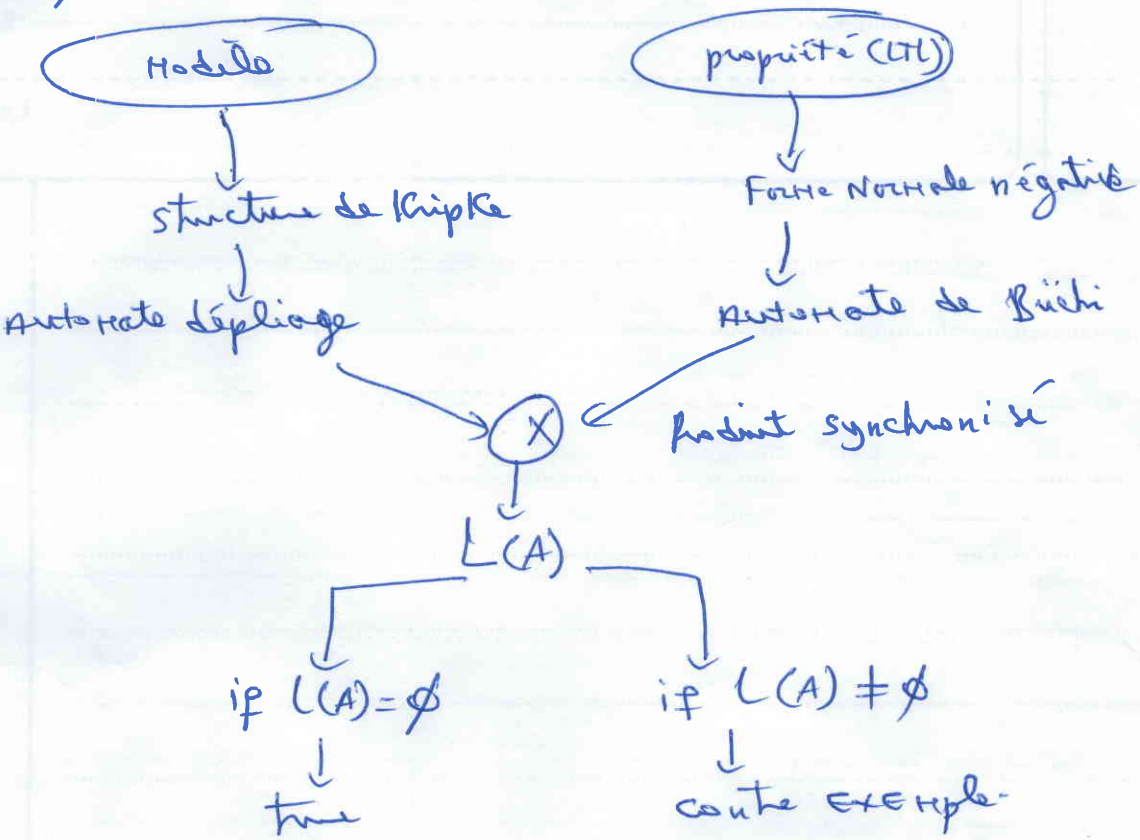
(3) trois avantages du langage MAUDE :

1- Simplicité
2- expressivité
3- bonne performances.

(4) - trois techniques de vérif. formelle :

1- Model-checking
2- preuve de theorèmes
3- bi-simulation.

(5) - principe de l'approche basé état de model-checking :



(6) Problème : Explosion combinatoire

Solution : Vérification à la volée
ou vérification symbolique.

EXO1 :

fmod demain is

protecting NAT .

Sort date .

Sorts jour mois an .

Subsorts jour mois an < Nat .

op Nbjour : mois an ⟶ jour .

op _)_)_ : jour mois an ⟶ date [ctor] .

op lendemain : date ⟶ date .

Ceq Nbjour (y,8) = 31 if y=1 ∨ y=3 ∨ y=5 ∨ y=7 ∨ y=8
∨ y=10 ∨ u=12 .

Ceq Nbjour (y,8) = 30 if y=4 ∨ y=6 ∨ y=9 ∨ y=11 .

Ceq Nbjour (y,8) = 29 if y=2 ∧ (y modulo 400 = 0)
∨ ((y modulo 4 = 0) ∧ (y modulo 100 =/= 0)) .

Ceq Nbjour (y,8) = 28 if y=2 ∧ (y modulo 400 =/= 0)
∧ ((y modulo 4 =/= 0) ∨ (y modulo 100 = 0)) .

Ceq lendemain (31,12,8) = (1,1,SZ) .

Ceq lendemain (u,y,8) = (1,Sy,8) if Nbjour (y,8) - u = 0 .

Ceq lendemain (u,y,8) = (Su,y,8) if Nbjour (y,8) - u > 0 .

endfm.

Ex 02

```
omod ASCenCam is
  protecting NAT .
  sort etatPorte .
  ops ouverte fermee : ---> etatporte [ctor] .
  msg appel : Oid Nat ---> Msg .
  class Asc | app: Nat , etage : Nat , Destination : Nat ,
              Porte : etatPorte , charge : Nat .
  var A : Oid .
  vars X y z c : Nat .
  var et : etatPorte .
```

rl [appel] : appel(A,X) < A : Asc | app : X , etage : y , Porte : et , charge : c > .

$\Rightarrow$ < A : Asc | app : X , etage : y , Porte : fermee , charge : c >

crl [dep_bas] : < A : Asc | app : X , etage : y , Porte : fermee , charge : c >

$\Rightarrow$ < A : Asc | app : X , etage : y - 1 , Porte : fermee , charge : c >

if y > X .

crl [=appel] : < A : Asc | app : X , etage : y , Porte : fermee , charge : c >

$\Rightarrow$ < A : Asc | app : X , etage : y , Porte : ouvert , charge : 0 >

if y = X .

crl [entre] : < A : Asc | etage : y , Porte : ouvert , charge : 0 >

$\Rightarrow$ < A : Asc | etage : y , Porte : fermee , charge : 1 >

Appel (A,z) .

crl [destin haut] : Appel(A,z) < A : Asc | etage : y , Porte : fermee , charge : 1 , destination : z >

$\Rightarrow$ < A : Asc | etage : y+1 , Porte-fermee , charge : 1 , destination : z > if z > y .

cnL [=destina]: $\langle A: Asc \mid etage: y, Porte: ferme, charge: 1, destination: z\rangle$

$\Rightarrow \langle A: Asc \mid etage: y, Porte: ouvert, charge: 1, destination: z\rangle$ if $z = y$.

nL [sortir]: $\langle A: Asc \mid etage: y, Porte: ouvert, charge: 1, destination: t\rangle$

$\Rightarrow \langle A: Asc \mid etage: y, Porte: ouvert, charge: 0\rangle$.

cnL [dep-haut]: $\langle A: Asc \mid app: x, etage: y, Porte: ferme, charge: c\rangle$

$\Rightarrow \langle A: Asc \mid app: x, etage: y+1, Porte: ferme, charge: c\rangle$

if $y < x$.

cnL [destin bas]: Appel(A, z) $\langle A: Asc \mid etage: y, Porte: ferme, charge: 1, destination: z\rangle$

$\Rightarrow \langle A: Asc \mid etage: y-1, Porte: ferme, charge: 1, destination: z\rangle$ if $z < y$.

end).