

Universite de M'sila
Département math et informatique
Master 2 IA
logique et algebre de processus

Gherabi Amira

November 24, 2021

exercice n01: la demonstration des trois lois pour l'operateur \parallel

1. $A \parallel B = B \parallel A \rightarrow A$ Commutativité

* on a $A \parallel B \xrightarrow{a} A$ et $B \parallel A \xrightarrow{a} A$ alors $A \parallel B = B \parallel A$

2. $A \parallel \text{stop} = \text{stop} \parallel A = A \rightarrow A$ Zéro absorption

* on a $A \parallel \text{stop} \xrightarrow{a} A$ et $\text{stop} \parallel A \xrightarrow{a} A$ alors $\text{stop} \parallel A = A \parallel \text{stop} = A$

3. $A \parallel (B \parallel C) = (A \parallel B) \parallel C \rightarrow$ Associativité

on a $B \parallel C \xrightarrow{a} B$ (1)

on a $A \parallel B \xrightarrow{a} B$ (2)

depuis (1) et (2) on trouve que : $A \parallel (B \parallel C) \xrightarrow{a} B$ (A)

et $(A \parallel B) \parallel C \xrightarrow{a} B$ (B)

depuis (A) et (B) on trouve que : $A \parallel (B \parallel C) = (A \parallel B) \parallel C$

exercice n02: la demonstration des trois lois pour l'operateur \parallel

1. $A \parallel B = B \parallel A \rightarrow A$ Commutativité

* on a $A \parallel B \xrightarrow{a} A$ et $B \parallel A \xrightarrow{a} A$ alors $A \parallel B = B \parallel A$

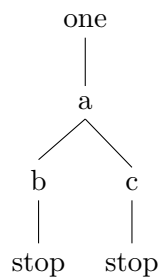
2. $A \parallel \text{stop} = \text{stop} \parallel A = A \rightarrow A$ Zéro absorption

* on a $A \parallel \text{stop} \xrightarrow{a} \text{stop}$ et $\text{stop} \parallel A \xrightarrow{a} \text{stop}$ alors $\text{stop} \parallel A = A \parallel \text{stop} = \text{stop}$

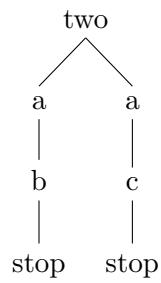
3. $A \parallel (B \parallel C) = (A \parallel B) \parallel C \rightarrow$ Associativité

exercice03

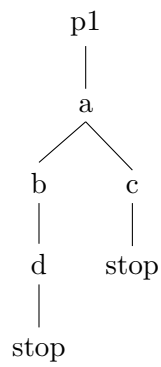
process one [a,b,c] a; (b; stop \parallel c; stop) endproc



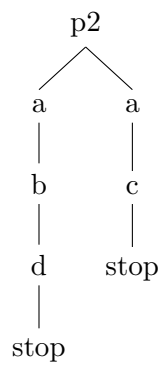
process two [a,b,c] a; b; stop [] a; c; stop **Endproc**



P1 := a; (b; d; stop [] c; stop)



P2 := a;b;d;stop[]a;c;stop



exercice04

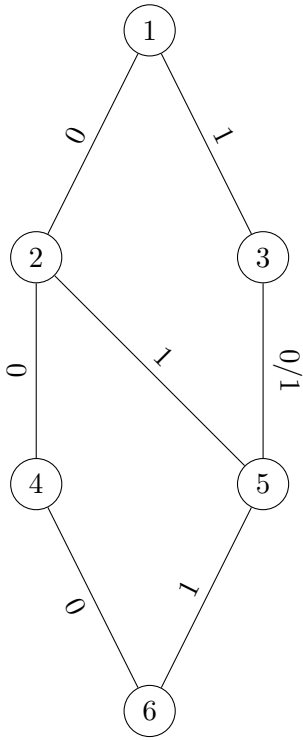
Les spécifications lotos pour les circuits logiques: or, and et xor

la spécification lotos pour le circuit logique OR

```

spec OR [a,b,c] :noexit
type BIT is
sorts BIT
opns 0 (*! constructor *),
1 (*! constructor *) :  $\rightarrow BIT$ 
or :  $BIT, BIT \rightarrow BIT$ 
eqns
ofsortBIT
or(0,0) = 0;
or(0,1) = 1;
or(1,0) = 1;
or(1,1) = 1;
endtype
behaviour
gateOR[a, b, c]
where
process gate – OR[a, b] : noexit :=
a ?aa : Bit; b ?bb : Bit; c !or(aa, bb); stop
endproc
endspec

```

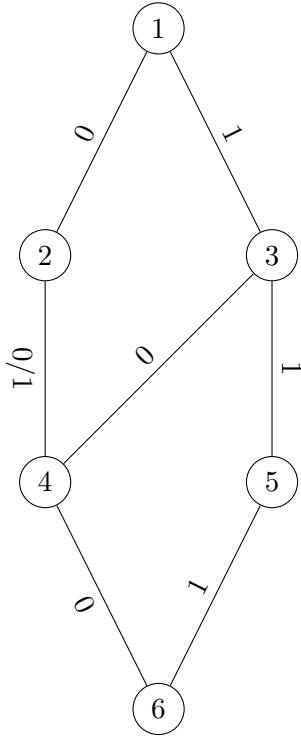


la spécification lotos pour le circuit logique AND

```

spec AND [a,b,c] :noexit
type BIT is
sorts BIT
opns 0 (*! constructor *),
1 (*! constructor *) :  $\rightarrow BIT$ 
and :  $BIT, BIT \rightarrow BIT$ 
eqns
ofsort BIT
and(0,0) = 0;
and(0,1) = 0;
and(0,1) = 0;
and(1,1) = 1;
endtype
behaviour
gateand[a,b,c]
where
processgate – AND[a,b] : noexit :=
a?aa : Bit; andb?bb : Bit; andc!and(aa,bb); andstop
endproc
endspec

```



la spécification lotos pour le circuit logique XOR

```

spec XOR [a,b,c] :noexit
type BIT is
sorts BIT
opns 0 (*! constructor *),
1 (*! constructor *) :  $\rightarrow$  BIT
xor : BIT, BIT  $\rightarrow$  BIT
eqns
ofsort BIT
xor(0,0) = 0;
xor(0,1) = 1;
xor(1,0) = 1;
xor(1,1) = 0;
endtype
behaviour
gate - XOR[a,b,c]
where
processgateXOr[a,b] : noexit :=
a?aa : Bit; andb?bb : Bit; andc!xor(aa,bb); stop
endproc
endspec

```

