exercice n01:

a) Les sources du probl‘eme d'apprentissage

1. nombre de cycles durant lesquels la performance de l'agent reste sous-optimale pour la tâche de décision donnée.
2. Les ressources de calcul nécessaires durant chaque cycle à l'agent pour réviser sa stratégie et choisir une action.

b) Le modèle d'apprentissage:

modèle d'apprentissage est un cadre formel donnant une mesure des deux sources de complexitè mentionné prècèdament .

c) L'influence des observations, les actions et le feedback sur la difficulté de l'apprentissage

1. L'observation:
   - la dimension peut etre immense, voire infini
   - les valeurs de certains attributs peuvent être imprécises, erronées, ou encore absentes.
   - environnements partiellement observables alors 'une situation incertain
2. L'action: les actions sont des decision soit simple ou complex,
   - l'espace des décisions possède une structure combinatoire; les décisions peuvent prendre la forme d'arbres, de graphes, ou encore d'hypergraphes
   - Les actions simples peuvent avoir un impact sur la difficulté de l'apprentissage selon la manière dont elles influent l'environnement (épisodique/séquentiel)
3. Feedback:Le type de feedback définit le mode d'apprentissage

exercice n02

les composants du problème l'apprentissage de porte logique XOR

la fonction Xor envoi une valeur vraie si les deux entrées ne sont pas egaux et fausse si elles sont egaux

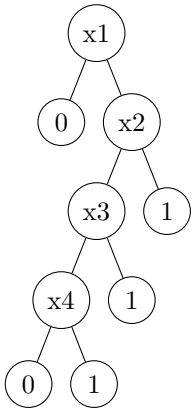
- l'espace des entrées est le couple (a,b)/ (a,b) $\in$ X={(0, 0),(0, 1),(1,0),(1, 1)}
- l'espace des sorties est Y = {true,false}.

* programme Xor en python
  def xor(x,y):
  return bool((x and not y) or (not x and y))

  print(xor(0,0))
  print(xor(0,1))
  print(xor(1,0))
  print(xor(1,1))

exercice n03:

x1 $\vee x2 \wedge x3 \vee x4$



exercice n04:

a) la diff erence entre une requˆete d'appartenance et une requˆete d' equivalence est

1. Une requˆete d'appartenance (MQ) associe 'a une instance x pos ee par l'apprenant la r eponse oui si h(x) = 1, et non sinon.

2. Une requˆete d' equivalence (EQ) associe 'a une hypoth'ese h pos e par l'apprenant la r eponse oui si h = h, et non sinon

b) Si pour toute instance (x1, x2, x3) quelle est l'hypoth'ese (le concept) le plus sp ecifiqu

est h1((x1,x2,x3)) = x1

c) Si h(x1,x2,x3) = x1 x2 et la requˆete est h((0,1,1)) = 0

1. le type de cette requˆete est requˆete d'appartenance (MQ)
2. et si h*((0,1,1)) = 1, requˆete d' equivalence (EQ)
3. Si le contre exemple est (0,1,0) mettre 'a jour h. h((0,1,0)) = 1

exercice05

exercice06

la diff erence entre le mod'ele agnostique et le mod'ele PAC

Dans le mod'ele "agnostique", la distribution D est arbitraire, ce quiimplique qu'il n'existe a priori aucune d ependance fonctionnelle entre une instance x et une d ecision y dans un exemple tir e dans D. En revanche, dans le mod'ele (PAC), nous supposons qu'il existe uned ependance fonctionnelle gouvern ee par une fonction cible

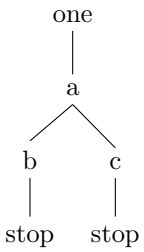
exercice01

1. A[]B = B[]A → Commutativité
2. A[]stop = stop[]A = A → Zéro absorption
3. A[](B[]C) = (A[]B)[]C → Associativité

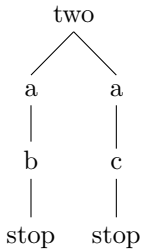
exercice02

exercice03

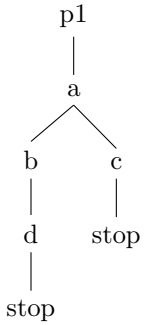
process one [a,b,c] a; (b; stop [] c; stop) endproc



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



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



exercice04

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

AND

specification circuitlogiqueAND [a,b,c] :noexit
type BIT is
sorts BIT
opns 0 (*! constructor *),
1 (*! constructor *) : -¿ BIT
and : BIT ,BIT-¿ BIT
eqns
ofsort BIT
and (0,0) = 0;
and (0,1) = 0;
and (0,1) = 0;
and (1,1) = 1;
endtype
behaviour

gate$_a nd[a, b, c]$
where
processgate$_A ND[a, b]$: *noexit* :=
$a?aa : Bit; b?bb : Bit; c!and(aa, bb);$ *stop*
endproc
endspec

OR

specification circuitlogiqueOR [a,b,c] :noexit
type BIT is
sorts BIT
opns 0 (*! constructor *),
1 (*! constructor *) : -¿ BIT
or : BIT ,BIT-¿ BIT
eqns
ofsort BIT
or (0,0) = 0;
or (0,1) = 1;
or (0,1) = 1;
or (1,1) = 1;
endtype
behaviour
gate$_O R[a, b, c]$
where
processgate$_O R[a, b]$: *noexit* :=
$a?aa : Bit; b?bb : Bit; c!or(aa, bb);$ *stop*
endproc
endspec

XOR

specification circuitlogiqueXOR [a,b,c] :noexit
type BIT is
sorts BIT
opns 0 (*! constructor *),
1 (*! constructor *) : -¿ BIT
xor : BIT ,BIT-¿ BIT
eqns
ofsort BIT
and (0,0) = 0;
and (0,1) = 1;
and (0,1) = 1;
and (1,1) = 0;

endtype
behaviour
gate$_X OR[a, b, c]$
where
process $gate_X Or[a, b] : noexit :=$
$a?aa : Bit; b?bb : Bit; c!xor(aa, bb); stop$
endproc
endspec