SU LRC - TME 9 M1 AI2D
Logique Épistémique avec LoTREC
Année 2025-2026

Yuxiang ZHANG
Kenan ALSAFADI

Decembre 2025

Résumé

Ce TME vise à explorer les fondements de la logique épistémique S5 à travers l'utilisation pratique du logiciel LoTREC. Notre étude s'articule autour de quatre axes principaux : (1) la vérification systématique de formules épistémiques dans S5, permettant d'identifier leur statut (satisfiable, insatisfiable, ou valide) ; (2) l'analyse comparative entre les systèmes S5 et S4, mettant en lumière l'impact de la propriété de symétrie sur la validité de l'équivalence d'introspection négative ; (3) la modélisation multiagents d'un problème épistémique classique (les trois femmes sur l'escalier) et la vérification de propriétés de connaissance complexes ; (4) la construction et l'étiquetage d'un modèle de Kripke pour trois agents avec validation de formules spécifiques. Ce rapport présente une approche méthodique combinant expérimentation avec LoTREC et analyse théorique, offrant ainsi une compréhension approfondie des mécanismes de raisonnement épistémique et de leur implémentation formelle.

Table des matières

1 Introduction

La logique épistémique, en formalisant les notions de connaissance et de croyance, constitue un outil essentiel pour l'étude des systèmes multiagents, de la théorie des jeux, et de l'intelligence artificielle distribuée. Ce travail dirigé (TME) a pour objectif d'approfondir la compréhension de la logique épistémique S5, considérée comme le système standard modélisant une connaissance parfaite (véridique, introspective et complètement introspective), en utilisant l'environnement de preuve LoTREC (Logic Tableau Research, Education and Communication).

Nous structurons notre étude en quatre exercices progressifs. Le premier exercice se concentre sur la vérification de formules dans S5, nous permettant de distinguer les concepts de satisfiabilité, insatisfiabilité et validité, tout en analysant les stratégies de preuve spécifiques à cette logique. Le deuxième exercice met en évidence une différence fondamentale entre S5 et S4 en étudiant la validité de l'équivalence $\neg Kp \Leftrightarrow K\neg Kp$, illustrant ainsi le rôle crucial de la propriété de symétrie (introspection négative). Le troisième exercice étend notre investigation au cadre multiagents en modélisant et analysant le problème classique des trois femmes sur l'escalier, ce qui nous permet de vérifier des propriétés épistémiques complexes impliquant plusieurs agents. Enfin, le quatrième exercice, issu d'un examen, nous confronte à la tâche inverse : construire et étiqueter un modèle de Kripke pour trois agents de manière à satisfaire un ensemble de formules épistémiques données.

À travers cette démarche expérimentale avec LoTREC, couplée à une analyse théorique rigoureuse, ce rapport vise non seulement à valider des propriétés formelles, mais aussi à développer une intuition opérationnelle des mécanismes de raisonnement épistémique et de leur implémentation dans un outil de preuve assistée.

2 Exercice 1 : Vérification de formules dans S5

Dans cet exercice, nous choisissons la logique S5, appelée `S5-explicit-edges` dans LoTREC. Cette logique implémente les propriétés caractéristiques des cadres de Kripke S5.

2.1 Axiomes et propriétés de S5

Le système S5 (également noté KT45) est caractérisé par les axiomes suivants :

— **(K)** : $\vdash K(F \to G) \to (KF \to KG)$ (distribution de la connaissance)

— **(T)** : $\vdash KF \to F$ (réflexivité, véracité de la connaissance)

— **(4)** : $\vdash KF \to KKF$ (introspection positive, transitivité)

— **(5)** : $\vdash \neg KF \to K\neg KF$ (introspection négative, euclidianité)

Ces axiomes correspondent aux propriétés suivantes de la relation d'accessibilité R dans les modèles de Kripke :

— **Réflexivité** : $\forall w, wRw$ (chaque monde est accessible depuis lui-même)

— **Transitivité** : si uRv et vRw alors uRw

— **Euclidianité** : si uRv et uRw alors vRw

— **Symétrie** : si uRv alors vRu (conséquence de la réflexivité et de l'euclidianité)

2.2 Stratégies dans S5-explicit-edges

LoTREC implémente une stratégie spécifique pour S5, appelée `S5Strategy`, qui garantit que tous les modèles construits satisfont les propriétés S5.

2.2.1 Stratégie S5Strategy

```
S5Strategy
repeat
    CPLStrategy
    MarkFulfilledPos
    Pos
    ReflexiveArcs
    SymmetricArcs
    TransitiveArcs
    Nec
end
```

Listing 1 – Stratégie S5Strategy dans LoTREC

2.2.2 Stratégie CPLStrategy

```
CPLStrategy
repeat
    firstRule
    Stop
    NotNot
    NotNec
    NotPos
    And
    NotOr
    NotImp
    NotAnd
    NotEquiv
    Imp
    Equiv
    Or
end
```

Listing 2 – Stratégie CPLStrategy pour la partie propositionnelle

2.3 Vérification de satisfiabilité et validité

Nous vérifions si les formules suivantes sont satisfiables, insatisfiables, ou valides dans S5 :

1. $Kp \rightarrow \neg K \neg Kp$
2. $Kp \wedge KK\neg p$
3. $Kp \rightarrow KK\neg p$

2.3.1 Formule 1 : $Kp \rightarrow \neg K \neg Kp$

Notation LoTREC : `imp nec P not nec not nec P`

Test de satisfiabilité : Étapes d'exécution :

— Next Rule : Imp

— Next Rule : Or (Créer deux premodel : premodel.1 et premodel.2)

— On traite d'abord `premodel.1` :

  — Next Rule : NotNec

  — Next Rule : Pos

  — Next Rule : ReflexiveArcs

  — Next Rule : SymmetricArcs

  — Next Rule : TransitiveArcs

  — Next Rule : MarkFulfilledPos

  — Next Rule : SymmetricArcs

— Maintenant on traite `premodel.2` :

  — Next Rule : NotNec

  — Next Rule : Pos

  — Next Rule : ReflexiveArcs

  — Next Rule : SymmetricArcs

  — Next Rule : TransitiveArcs

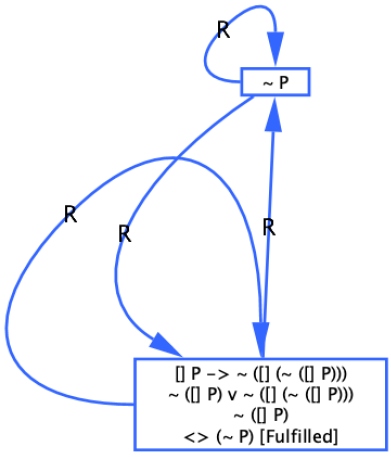
  — Next Rule : NotNot

  — Next Rule : MarkFulfilledPos

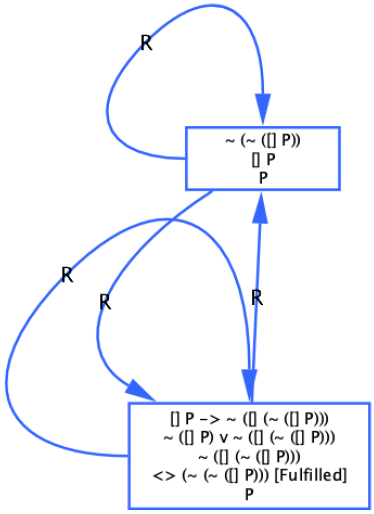
  — Next Rule : SymmetricArcs

  — Next Rule : Nec

La formule génère deux prémodèles, tous deux ouverts (Figure 1). La formule est donc **satisfiable**.



(a) Prémodel.1 (ouvert)



(b) Prémodel.2 (ouvert)

FIGURE 1 – Deux prémodèles ouverts pour $Kp \rightarrow \neg K \neg Kp$

Test de validité : Pour vérifier si la formule est valide, on teste sa négation :

$$\neg(Kp \rightarrow \neg K \neg Kp)$$

Entrée dans LoTREC : `not imp nec P not nec not nec P`
Étapes d'exécution :

— Next Rule : NotImp
— Next Rule : NotNot
— Next Rule : ReflexiveArcs
— Next Rule : SymmetricArcs
— Next Rule : TransitiveArcs
— Next Rule : Nec
— Next Rule : Stop

La négation de la formule génère un prémodèle fermé (Figure 2). La négation est donc **insatisfiable**, ce qui signifie que la formule originale est **valide** dans S5.

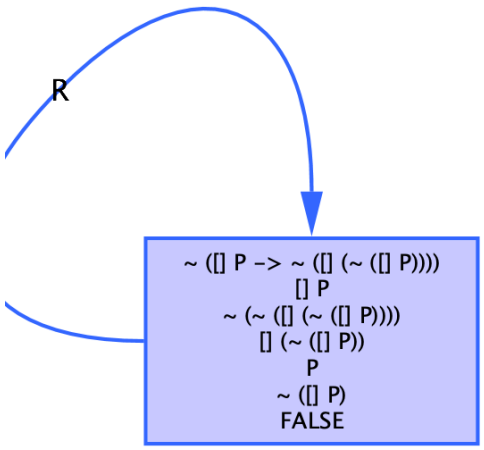


FIGURE 2 – Prémodèle fermé pour $\neg(Kp \rightarrow \neg K \neg Kp)$

2.3.2 Formule 2 : $Kp \wedge KK\neg p$

Notation LoTREC : `and nec P nec nec not P`

Test de satisfiabilité : Étapes d'exécution :

— Next Rule : And
— Next Rule : ReflexiveArcs
— Next Rule : SymmetricArcs
— Next Rule : TransitiveArcs
— Next Rule : Nec
— Next Rule : Nec
— Next Rule : Stop

La formule génère un seul prémodèle qui est fermé (Figure 3). La formule est donc **insatisfiable** dans S5.

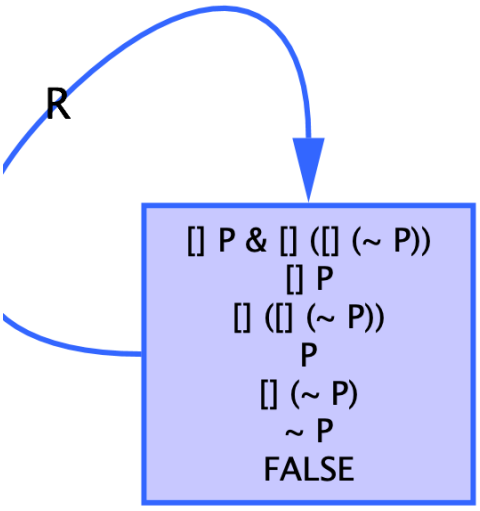


FIGURE 3 – Prémodèle fermé pour $Kp \wedge KK\neg p$

2.3.3 Formule 3 : $Kp \rightarrow KK\neg p$

Notation LoTREC : `imp nec P nec nec not P`

Test de satisfiabilité : Étapes d'exécution :

— Next Rule : Imp
— Next Rule : Or (Créer deux premodel)
— On traite d'abord `premodel.1` :
  — Next Rule : NotNec
  — Next Rule : Pos
  — Next Rule : ReflexiveArcs
  — Next Rule : SymmetricArcs
  — Next Rule : TransitiveArcs
  — Next Rule : MarkFulfilledPos

— Next Rule : SymmetricArcs

— Maintenant on traite `premodel.2` :

— Next Rule : ReflexiveArcs

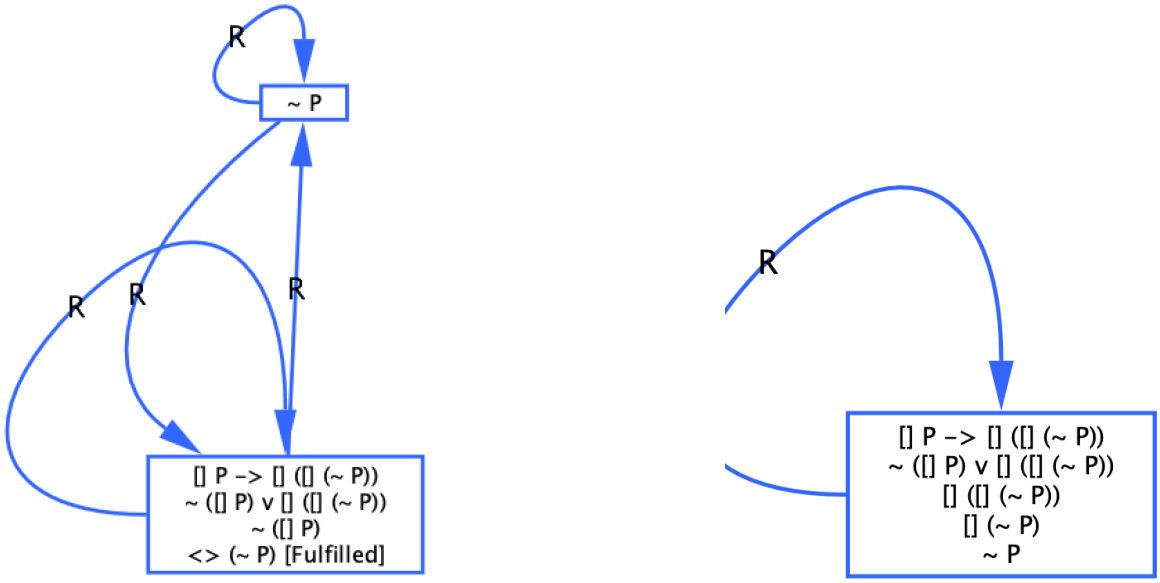
— Next Rule : SymmetricArcs

— Next Rule : TransitiveArcs

— Next Rule : Nec

— Next Rule : Nec

La formule génère deux prémodèles ouverts (Figure 4). La formule est donc **satisfiable**.



(a) Prémodel.1 (ouvert) (b) Prémodel.2 (ouvert)

FIGURE 4 – Deux prémodèles ouverts pour $Kp \rightarrow KK\neg p$

Test de validité : Pour vérifier si la formule est valide, on teste sa négation :

$$\neg(Kp \rightarrow KK\neg p)$$

Entrée dans LoTREC : `not imp nec P nec nec not P`
Étapes d'exécution :

— Next Rule : NotImp

— Next Rule : NotNec

— Next Rule : Pos

— Next Rule : ReflexiveArcs

— Next Rule : SymmetricArcs

— Next Rule : TransitiveArcs

— Next Rule : Nec

— Next Rule : NotNec

— Next Rule : MarkFulfilledPos

— Next Rule : Pos

— Next Rule : ReflexiveArcs
— Next Rule : SymmetricArcs
— Next Rule : TransitiveArcs
— Next Rule : Nec
— Next Rule : NotNot
— Next Rule : MarkFulfilledPos
— Next Rule : SymmetricArcs
— Next Rule : TransitiveArcs

La négation de la formule génère un prémodèle ouvert (Figure 5). La négation est donc **satisfiable**, ce qui signifie que la formule originale n'est pas valide. Elle est seulement **satisfiable** dans S5.

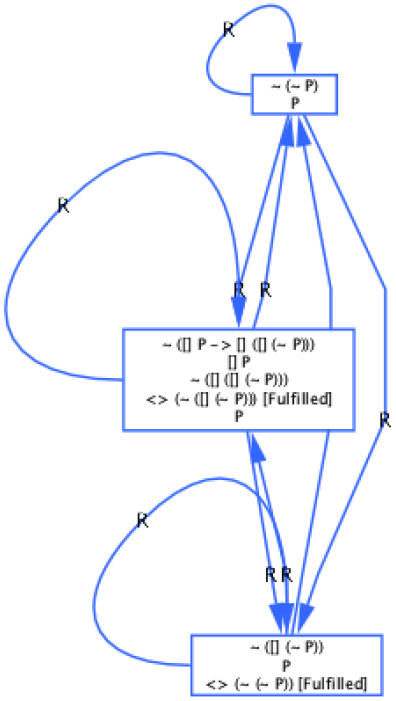


FIGURE 5 – Prémodèle ouvert pour $\neg(Kp \rightarrow KK\neg p)$

2.4 Analyse comparative des stratégies

Une comparaison entre les stratégies de la logique K (TME précédent) et de la logique S5 révèle des différences fondamentales reflétant les propriétés spécifiques des cadres de Kripke.

Logique K (Monomodal-K)	**Logique S5 (S5-explicit-edges)**
Propriétés de R	**Propriétés de R**
Aucune contrainte (relation quelconque)	Réflexive, symétrique et transitive
Structure stratégique	**Structure stratégique**
`KStrategy: repeat{CPLStrategy; Pos; Nec}`	`S5Strategy: repeat{CPLStrategy; MarkFulfilledPos; Pos; ReflexiveArcs; SymmetricArcs; TransitiveArcs; Nec}`
Règles spécifiques	**Règles spécifiques**
`Pos, Nec`	`ReflexiveArcs, SymmetricArcs, TransitiveArcs, MarkFulfilledPos`
Optimisation	**Optimisation**
Pas de règles de fermeture optimisées	`MarkFulfilledPos` pour éviter les mondes inutiles

TABLE 1 – Comparaison des stratégies LoTREC pour K et S5

Observations clés :

1. **Modularité** : Les deux logiques utilisent la même `CPLStrategy` pour la partie propositionnelle.
2. **Propriétés relationnelles** : S5 impose explicitement les propriétés réflexive, symétrique et transitive via des règles dédiées.
3. **Optimisation** : La règle `MarkFulfilledPos` en S5 minimise la création de nouveaux mondes, cruciale pour contrôler l'explosion combinatoire due à la symétrie et transitivité.
4. **Construction des modèles** : En S5, tous les modèles construits sont automatiquement équipés d'une relation d'équivalence.

3 Exercice 2 : Validité de $\neg Kp \Leftrightarrow K\neg Kp$

Cet exercice examine la validité de l'équivalence $\neg Kp \Leftrightarrow K\neg Kp$ dans les systèmes S5 et S4.

3.1 Validité dans S5

3.1.1 Test de la formule originale

En notation LoTREC : `equiv not nec P nec not nec P`
La formule génère quatre prémodèles (Figure 6) :

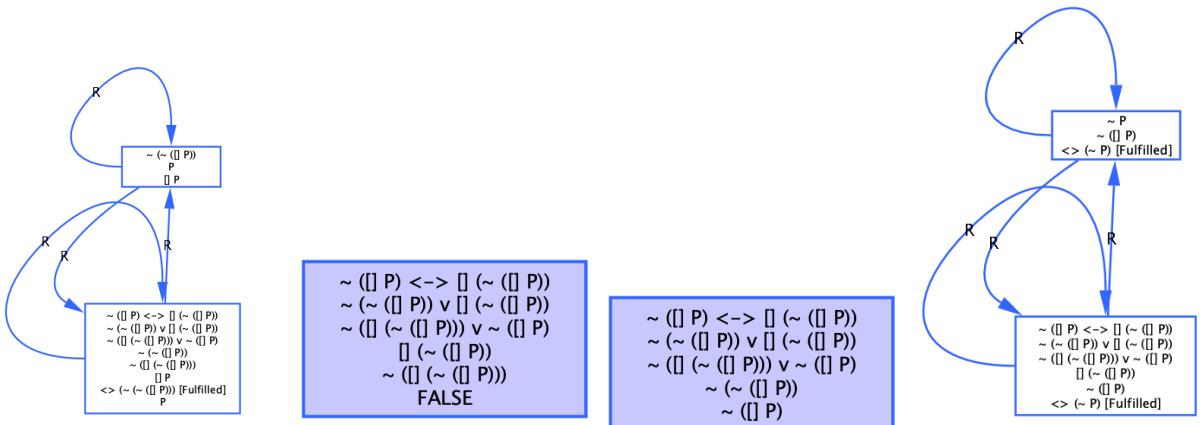
— **premodel.1** : ouvert

— **premodel.2.1** : fermé

— **premodel.3** : fermé

— **premodel.2.2** : ouvert

La présence de prémodèles ouverts indique que la formule est **satisfiable** dans S5.



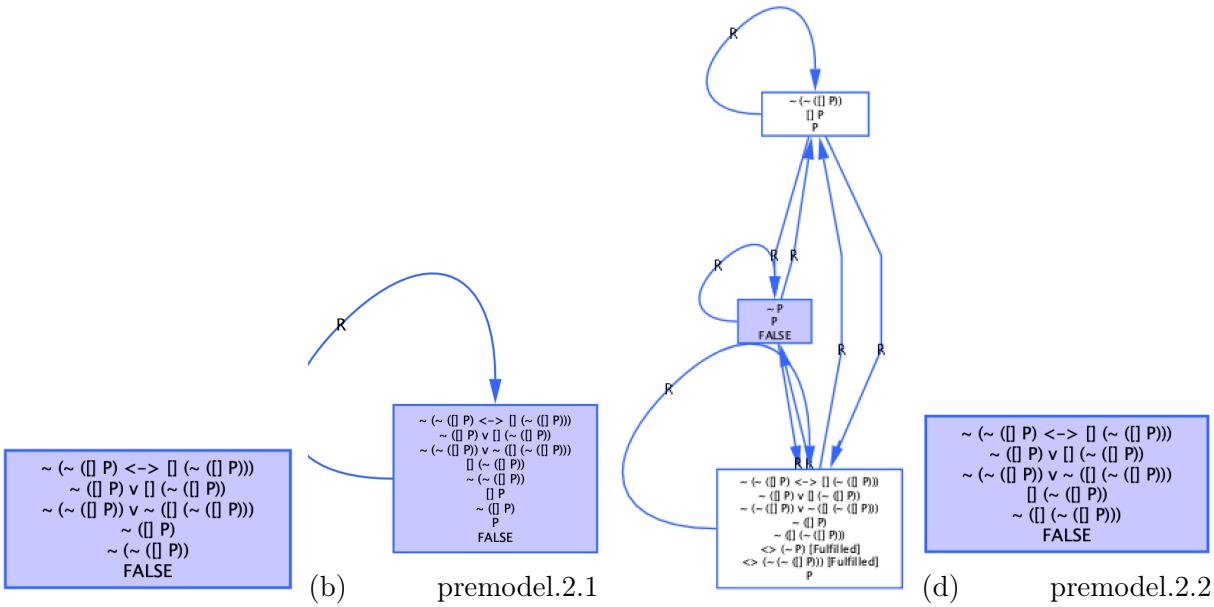
(a) premodel.1 (ouvert) (b) premodel.2.1 (fermé) (c) premodel.3 (fermé) (d) premodel.2.2 (ouvert)

FIGURE 6 – Prémodèles pour $\neg Kp \Leftrightarrow K\neg Kp$ dans S5

3.1.2 Test de la négation

En notation LoTREC : `not equiv not nec P nec not nec P`

La négation génère quatre prémodèles, tous fermés (Figure 7). La négation est donc **insatisfiable**.



(a) premodel.1 (fermé) (b) premodel.2.1 (fermé) (c) premodel.3 (fermé) (d) premodel.2.2 (fermé)

FIGURE 7 – Prémodèles pour $\neg(\neg Kp \Leftrightarrow K\neg Kp)$ dans S5

3.1.3 Conclusion pour S5

Puisque :

1. La formule $\neg Kp \Leftrightarrow K\neg Kp$ est satisfiable dans S5
2. Sa négation est insatisfiable dans S5

l'équivalence est **valide** dans le système S5 (KT45).

3.2 Non-validité dans S4

3.2.1 Adaptation pour S4-Explicit-R

Dans S4-Explicit-R, le connecteur `equiv` n'est pas défini. Nous réécrivons donc l'équivalence en utilisant :

$$A \Leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$$

et

$$A \rightarrow B \equiv \neg A \vee B$$

L'équivalence devient :

$$(Kp \vee K\neg Kp) \wedge (\neg K\neg Kp \vee \neg Kp)$$

En notation LoTREC :

```
and (or (nec P) (nec (not nec P))) (or (not (nec (not nec P))) (not
    (nec P)))
```

3.2.2 Test dans S4

La formule réécrite génère quatre prémodèles dans S4 (Figure 8) :

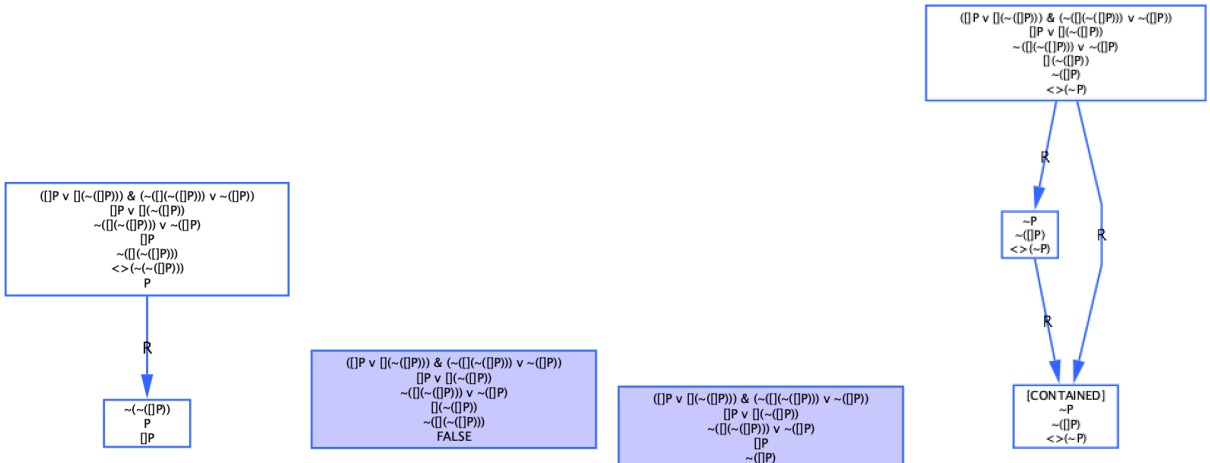
— **premodel.1** : ouvert

— **premodel.2.1** : fermé

— **premodel.3** : fermé

— **premodel.2.2** : ouvert

La formule est donc **satisfiable** dans S4.



(a) premodel.1 (ouvert) (b) premodel.2.1 (fermé) (c) premodel.3 (fermé) (d) premodel.2.2 (ouvert)

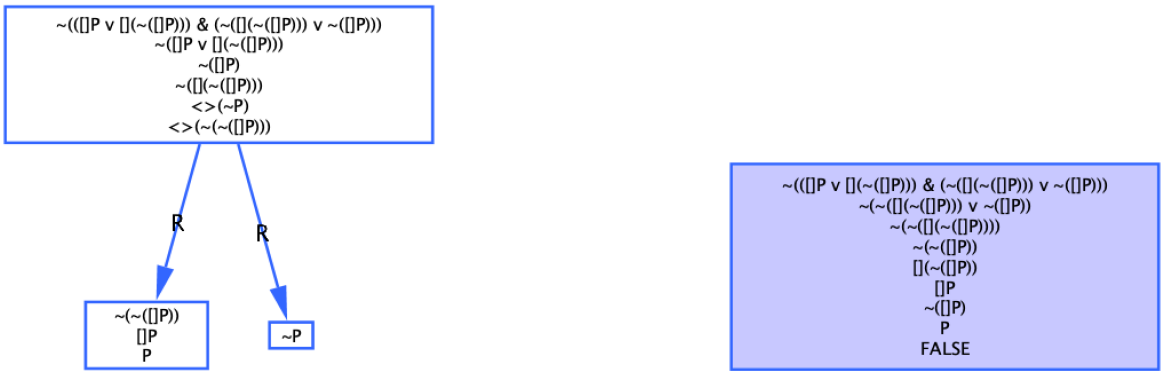
FIGURE 8 – Prémodèles pour $\neg Kp \Leftrightarrow K\neg Kp$ dans S4

3.2.3 Test de la négation dans S4

La négation de la formule réécrite génère deux prémodèles (Figure 9) :

— **premodel.1** : ouvert

— **premodel.2** : fermé
La négation est donc également **satisfiable** dans S4.



(a) premodel.1 (ouvert) (b) premodel.2 (fermé)

FIGURE 9 – Prémodèles pour $\neg(\neg Kp \Leftrightarrow K\neg Kp)$ dans S4

3.2.4 Conclusion pour S4

Puisque :

1. La formule $\neg Kp \Leftrightarrow K\neg Kp$ est satisfiable dans S4
2. Sa négation est également satisfiable dans S4

l'équivalence est **contingente** dans S4 : vraie dans certains modèles, fausse dans d'autres. Elle n'est donc **pas valide** dans le système S4 (KT4).

3.3 Analyse comparative S4 vs S5

3.3.1 Contre-modèle spécifique à S4

Le contre-modèle suivant explique pourquoi l'équivalence échoue dans S4 mais pas dans S5 :

$$\mathcal{M} = \langle W, R, I \rangle$$

avec :

— $W = \{w, u\}$
— $R = \{(w, w), (w, u), (u, u)\}$ (réflexive et transitive, mais **non symétrique**)
— $I(p) = \{u\}$ (p vrai seulement en u)
Évaluation :
— En w : $\neg Kp$ vrai (car p faux en w)
— En w : $K\neg Kp$ faux (car en u, $\neg Kp$ faux puisque Kp vrai en u)
Donc en w : $\neg Kp$ vrai mais $K\neg Kp$ faux, l'équivalence est fausse.

3.3.2 Propriétés relationnelles

La différence fondamentale entre S4 et S5 réside dans les propriétés de R :

— **S4 (KT4)** : R réflexive et transitive
— **S5 (KT45)** : R réflexive, transitive et symétrique (relation d'équivalence)

3.3.3 Interprétation épistémique

L'équivalence $\neg Kp \Leftrightarrow K\neg Kp$ exprime l'**introspection négative** : "ne pas savoir p" équivaut à "savoir qu'on ne sait pas p".

— En S5 (avec symétrie) : valide → connaissance parfaite avec pleine introspection

— En S4 (sans symétrie) : non valide → permet des modèles où un agent ignore p sans savoir qu'il l'ignore

4 Exercice 3 : Logique multiagents et l'exercice des trois femmes sur l'escalier

Dans cet exercice, nous passons à un cadre multiagents qui nécessite l'utilisation simultanée de plusieurs modalités. Nous utilisons la logique prédéfinie `Model-Checking-Multimodal` dans LoTREC, qui permet de spécifier à quelle relation fait référence l'emploi d'une modalité donnée. Par exemple, `nec R1 P` utilise la relation R1 et correspond à $K_1 p$, tandis que `nec R2 P` utilise la relation R2 et correspond à $K_2 p$. Les noms des relations doivent obligatoirement commencer par une majuscule.

Description du problème

L'exercice des trois femmes sur l'escalier présente la situation suivante :

— Trois femmes (A, B, C) sont sur différentes marches d'un escalier :
  — A sur la plus haute
  — B sur la moyenne
  — C sur la plus basse

— Les conditions de visibilité sont :
  — A voit le sommet de la tête de B et C
  — B voit le sommet de la tête de C
  — C ne voit rien

— Une quatrième femme passe et annonce qu'une d'entre elles a un papillon sur la tête.

4.1 Question 1 : Construction du modèle de Kripke

Nous avons utilisé deux méthodes pour construire la structure de Kripke correspondante dans LoTREC.

4.1.1 Méthode 1 : Édition directe de la règle ExampleOfModelAndFormula

La première méthode consiste à éditer directement la règle `ExampleOfModelAndFormula` dans l'interface de LoTREC. Voici le code que nous avons utilisé :

```
createNewNode wA
createNewNode wB
createNewNode wC

link wA wA R1
```

```
link wB wB R1
link wC wC R1

link wA wA R2
link wA wB R2
link wB wA R2
link wB wB R2
link wC wC R2

link wA wA R3
link wA wB R3
link wA wC R3
link wB wA R3
link wB wB R3
link wB wC R3
link wC wA R3
link wC wB R3
link wC wC R3

add wA A
add wB B
add wC C
```

Listing 3 – Code LoTREC pour construire le modèle des trois femmes

Explication du code :

— **Mondes** : wA, wB, wC représentent respectivement les mondes où le papillon est sur A, B, C.

— **Relations** :

  — R1 (pour A) : seules les boucles réflexives (A peut voir B et C, donc distingue tous les mondes).

  — R2 (pour B) : boucles réflexives + connexions entre wA et wB (B ne peut voir que C, donc ne distingue pas wA et wB).

  — R3 (pour C) : connexions complètes (C ne voit rien, donc ne distingue aucun monde).

— **Valuation** : attribution des propositions atomiques A, B, C dans chaque monde.

4.1.2 Méthode 2 : Modification du fichier XML

La deuxième méthode consiste à modifier directement le fichier XML de la logique `Model-Checking-Multimodal` (disponible sur GitHub : https://github.com/bilals/lotrec/blob/master/src/lotrec/logics/Model-Checking-Multimodal.xml). Nous avons créé une nouvelle règle `ExampleOfModelAndFormula` :

```
<rule>
    <rule-name>ExampleOfModelAndFormula</rule-name>

    <!-- Cr ation des mondes possibles -->
    <action>
```

```
        <action-name>createNewNode</action-name>
        <parameter>wA</parameter>
    </action>
    <action>
        <action-name>createNewNode</action-name>
        <parameter>wB</parameter>
    </action>
    <action>
        <action-name>createNewNode</action-name>
        <parameter>wC</parameter>
    </action>

    <!-- Relation R1 (Agent A) : A peut voir B et C, donc distingue
        tous les mondes -->
    <action>
        <action-name>link</action-name>
        <parameter>wA</parameter>
        <parameter>wA</parameter>
        <parameter>R1</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wB</parameter>
        <parameter>wB</parameter>
        <parameter>R1</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wC</parameter>
        <parameter>wC</parameter>
        <parameter>R1</parameter>
    </action>

    <!-- Relation R2 (Agent B) : B ne peut voir que C, donc ne
       distingue pas wA et wB -->
    <action>
        <action-name>link</action-name>
        <parameter>wA</parameter>
        <parameter>wA</parameter>
        <parameter>R2</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wA</parameter>
        <parameter>wB</parameter>
        <parameter>R2</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wB</parameter>
        <parameter>wA</parameter>
```

```
        <parameter>R2</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wB</parameter>
        <parameter>wB</parameter>
        <parameter>R2</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wC</parameter>
        <parameter>wC</parameter>
        <parameter>R2</parameter>
    </action>

    <!-- Relation R3 (Agent C) : C ne voit rien, donc ne distingue
       aucun monde -->
    <action>
        <action-name>link</action-name>
        <parameter>wA</parameter>
        <parameter>wA</parameter>
        <parameter>R3</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wA</parameter>
        <parameter>wB</parameter>
        <parameter>R3</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wA</parameter>
        <parameter>wC</parameter>
        <parameter>R3</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wB</parameter>
        <parameter>wA</parameter>
        <parameter>R3</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wB</parameter>
        <parameter>wB</parameter>
        <parameter>R3</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wB</parameter>
        <parameter>wC</parameter>
```

```
        <parameter>R3</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wC</parameter>
        <parameter>wA</parameter>
        <parameter>R3</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wC</parameter>
        <parameter>wB</parameter>
        <parameter>R3</parameter>
    </action>
    <action>
        <action-name>link</action-name>
        <parameter>wC</parameter>
        <parameter>wC</parameter>
        <parameter>R3</parameter>
    </action>

    <!-- Valuation des propositions atomiques -->
    <action>
        <action-name>add</action-name>
        <parameter>wA</parameter>
        <parameter>A</parameter>
    </action>
    <action>
        <action-name>add</action-name>
        <parameter>wB</parameter>
        <parameter>B</parameter>
    </action>
    <action>
        <action-name>add</action-name>
        <parameter>wC</parameter>
        <parameter>C</parameter>
    </action>
</rule>
```

Listing 4 – Extrait du fichier XML modifié

Comparaison des deux méthodes :

— **Méthode 1** : Plus directe et immédiate, mais nécessite de refaire la construction à chaque utilisation.

— **Méthode 2** : Plus permanente et réutilisable, mais nécessite des connaissances XML et peut présenter des problèmes de compatibilité selon la version de LoTREC.

4.1.3 Résultat : Modèle de Kripke obtenu

Après exécution de l'une des deux méthodes, LoTREC génère le modèle de Kripke représenté dans la Figure 10.

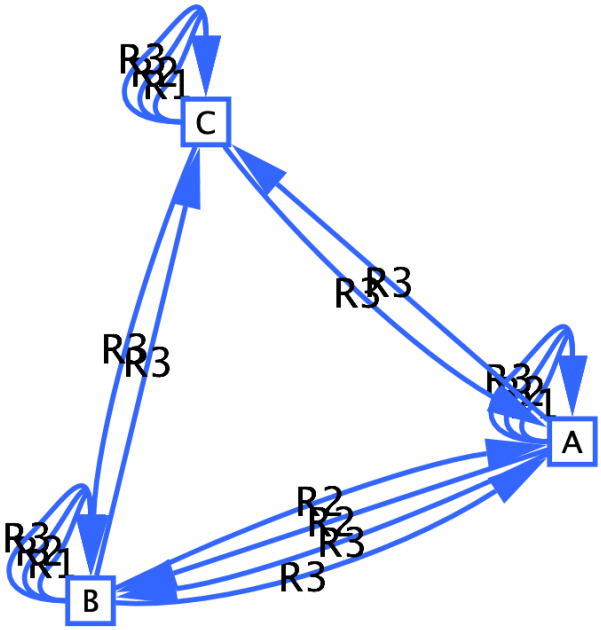


FIGURE 10 – Modèle de Kripke pour l'exercice des trois femmes sur l'escalier

Analyse du modèle : Le modèle représente fidèlement la situation cognitive des trois femmes :

- **Mondes** :
  - wA : papillon sur A (A vrai, B et C faux)
  - wB : papillon sur B (B vrai, A et C faux)
  - wC : papillon sur C (C vrai, A et B faux)
- **Relations d'accessibilité** :
  - R1 (A) : chaque monde n'est accessible qu'à lui-même → A distingue parfaitement les trois situations.
  - R2 (B) : wA et wB sont mutuellement accessibles, wC n'est accessible qu'à lui-même → B ne peut pas distinguer si le papillon est sur A ou sur B.
  - R3 (C) : tous les mondes sont mutuellement accessibles → C ne peut distinguer aucune situation.

4.2 Question 2 : Vérification des formules de l'exercice

Selon les indications du professeur, nous devons vérifier les formules suivantes :

4.2.1 A sait si elle a le papillon sur la tête ?

Pour montrer que A sait si elle a le papillon sur la tête, nous devons vérifier :

$$M \models K_A^{si} A \quad \Leftrightarrow \quad M \models K_A A \vee K_A \neg A$$

Cette formule doit être vraie dans tous les mondes wA, wB et wC.

En notation LoTREC, cela correspond à vérifier dans chaque monde :

```
or (nec R1 A) (nec R1 (not A))
```

On ajoute les 3 actions suivantes dans la règle `ExampleOfModelAndFormula`.

```
add wA isItTrue or (nec R1 A) (nec R1 (not A))
add wB isItTrue or (nec R1 A) (nec R1 (not A))
add wC isItTrue or (nec R1 A) (nec R1 (not A))
```

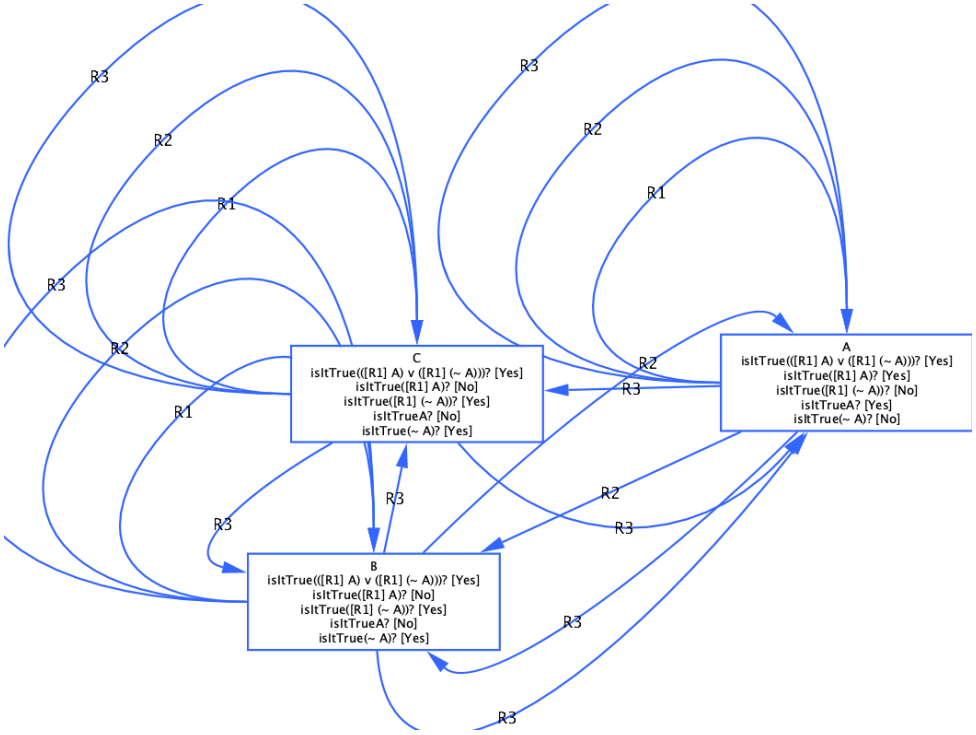


FIGURE 11 – Résultat de la vérification pour A : $K_A A \vee K_A \neg A$ dans tous les mondes

Résultat : Comme le montre la Figure 11, dans les trois mondes wA, wB et wC, la formule `isItTrue(([R1] A) v ([R1] (~A)))` est marquée comme **Yes**. Cela signifie que $M \models K_A A \vee K_A \neg A$ est vrai dans tous les mondes.

Par conséquent, **A sait si elle a le papillon sur la tête**.

4.2.2 C ne sait pas si elle a le papillon sur la tête ?

Pour montrer que C ne sait pas si elle a le papillon sur la tête, nous devons vérifier :

$$M \models \neg K_C^{si} C \quad \Leftrightarrow \quad M \models \neg(K_C C \vee K_C \neg C) \quad \Leftrightarrow \quad M \models \neg K_C C \wedge \neg K_C \neg C$$

Cette formule doit également être vraie dans tous les mondes wA, wB et wC.

En notation LoTREC, cela correspond à vérifier dans chaque monde :

```
and (not nec R3 C) (not nec R3 (not C))
```

On ajoute les 3 actions suivantes dans la règle `ExampleOfModelAndFormula`.

```
add wA isItTrue and (not nec R3 C) (not nec R3 (not C))
add wB isItTrue and (not nec R3 C) (not nec R3 (not C))
add wC isItTrue and (not nec R3 C) (not nec R3 (not C))
```

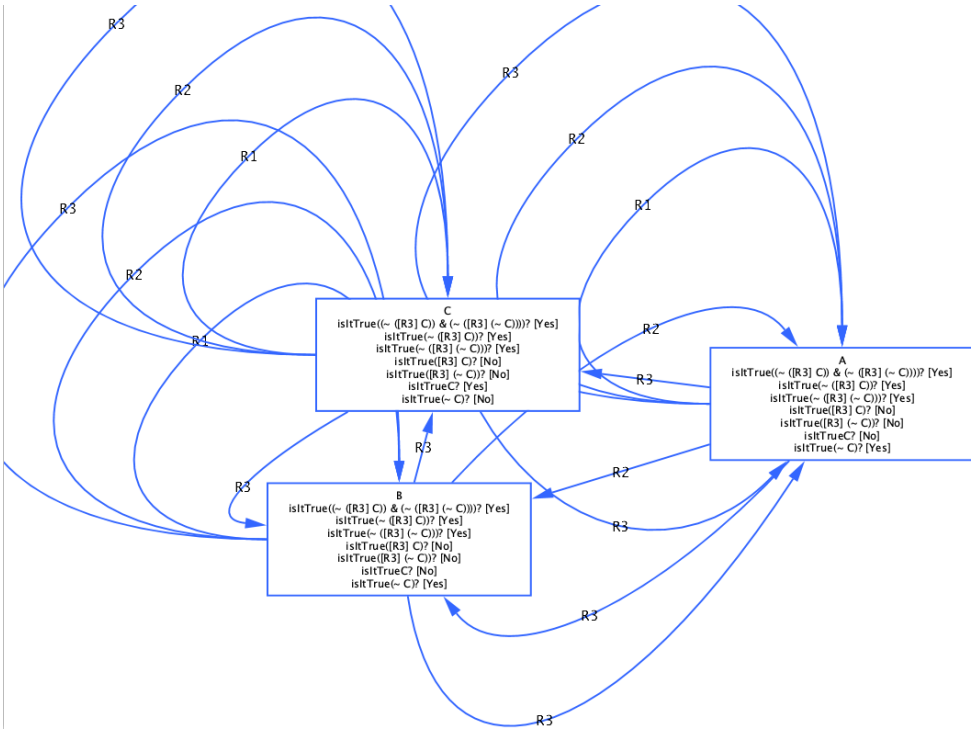


FIGURE 12 – Résultat de la vérification pour C : $\neg K_C C \wedge \neg K_C \neg C$ dans tous les mondes

Résultat : Comme le montre la Figure 12, dans les trois mondes wA, wB et wC, la formule `isItTrue((~([R3]C)) & (~([R3] (~C))))` est marquée comme **Yes**. Cela signifie que $M \models \neg K_C C \wedge \neg K_C \neg C$ est vrai dans tous les mondes.

Par conséquent, **C ne sait pas si elle a le papillon sur la tête**.

4.3 Question 3 & 4 & 5 : Vérification des propriétés supplémentaires

4.3.1 B peut-elle savoir si elle a un papillon ?

Pour vérifier que B peut savoir si elle a un papillon sur la tête, nous devons vérifier la formule :

$$M \models K_B B \vee K_B \neg B$$

Cette formule exprime que B sait soit qu'elle a le papillon ($K_B B$), soit qu'elle ne l'a pas ($K_B \neg B$).

En notation LoTREC, cela correspond à :

```
or (nec R2 B) (nec R2 (not B))
```

Nous ajoutons les vérifications dans les trois mondes :

```
add wA isItTrue or (nec R2 B) (nec R2 (not B))
add wB isItTrue or (nec R2 B) (nec R2 (not B))
add wC isItTrue or (nec R2 B) (nec R2 (not B))
```

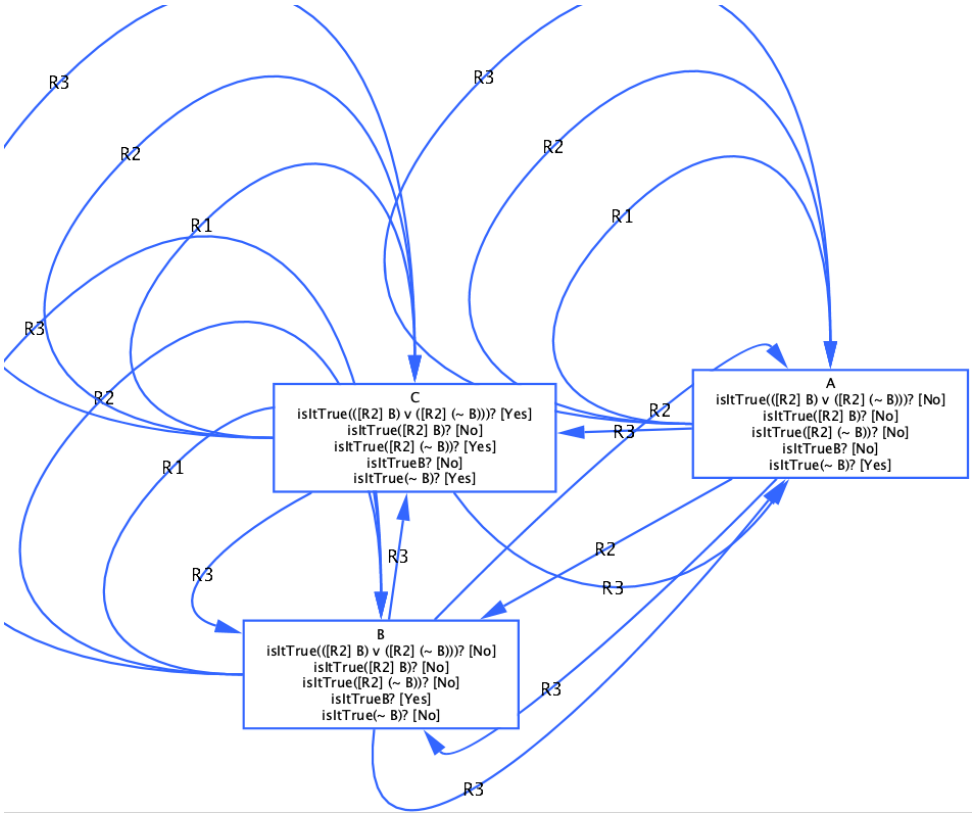


FIGURE 13 – Vérification de $K_B B \vee K_B \neg B$ dans les trois mondes

Résultat : Comme le montre la Figure 13, la formule `isItTrue(([R2] B) v ([R2] (~B)))` donne les résultats suivants :

— En wA : **No** (B ne sait pas si elle a le papillon)

— En wB : **No** (B ne sait pas si elle a le papillon)

— En wC : **Yes** (B sait qu'elle n'a pas le papillon)

Cela s'explique par les relations d'accessibilité de B (R2) :

— En wA et wB : Ces deux mondes sont mutuellement accessibles pour B, donc B ne peut pas les distinguer. Comme B est vrai en wB mais faux en wA, B ne sait pas si B est vrai ou faux.

— En wC : Seul wC est accessible depuis wC (boucle réflexive), donc B sait que C est vrai (papillon sur C) et en déduit qu'elle n'a pas le papillon ($K_B \neg B$).

4.3.2 Si C a le papillon, alors B sait si elle a le papillon ?

Pour vérifier cette implication, nous devons vérifier la formule :

$$M \models C \rightarrow (K_B B \vee K_B \neg B)$$

En notation LoTREC :

```
imp C (or (nec R2 B) (nec R2 (not B)))
```

Note : Le connecteur d'implication `imp` n'était pas défini par défaut dans la logique `Model-Checking-Multimodal`. En suivant la même approche que dans le TME8, nous avons ajouté manuellement ce connecteur dans notre fichier XML, en définissant :

— Un connecteur binaire `imp` avec affichage `_->_`

— Quatre règles de déduction : `Imp_Top_Down`, `Imp_Left_True_Bottom_Up`, `Imp_Right_True_Bottom_U` et `Imp_Not_True_Bottom_Up`

— L'insertion de ces règles dans les stratégies `Top_Down` et `Bottom_Up`

Nous vérifions cette formule dans tous les mondes :

```
add wA isItTrue imp C (or (nec R2 B) (nec R2 (not B)))
add wB isItTrue imp C (or (nec R2 B) (nec R2 (not B)))
add wC isItTrue imp C (or (nec R2 B) (nec R2 (not B)))
```

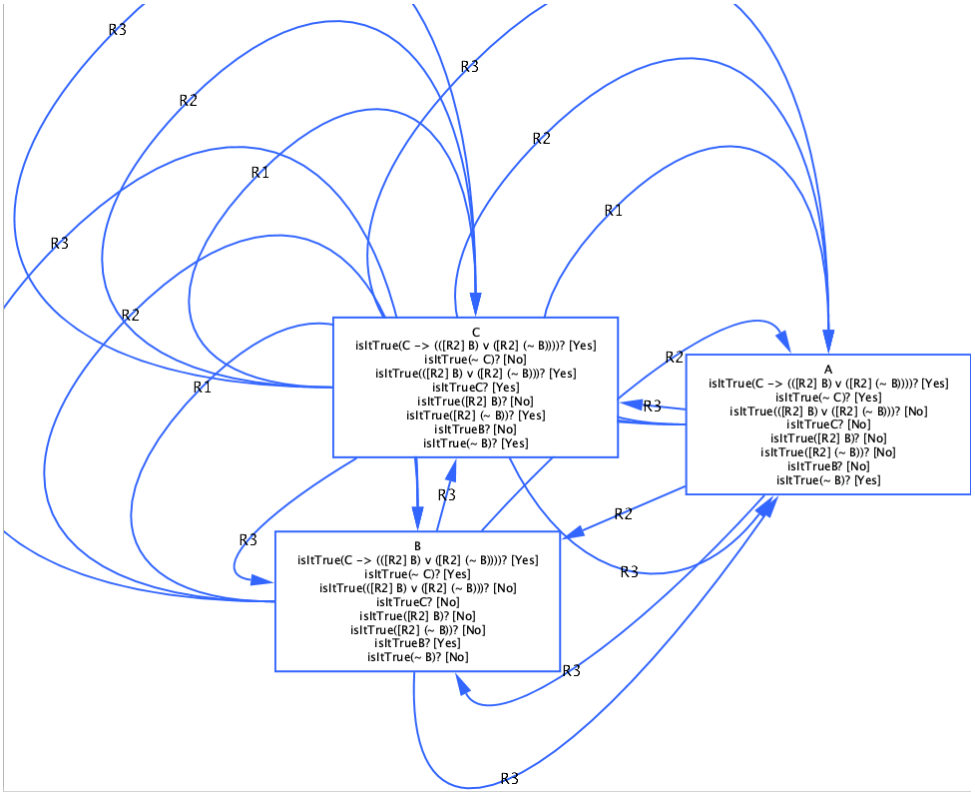


FIGURE 14 – Vérification de $C \rightarrow (K_B B \vee K_B \neg B)$ dans les trois mondes

Résultat : Cette formule est **Yes** dans tous les mondes :

— En wA et wB : C est faux, donc l'implication est vraie.

— En wC : C est vrai, et nous avons déjà vu que $K_B B \vee K_B \neg B$ est vrai en wC.

Conclusion : Nous pouvons en déduire que **si C a le papillon, alors B sait si elle a le papillon sur la tête**. Plus précisément, quand C a le papillon, B sait qu'elle ne l'a pas.

4.3.3 A sait-elle que C ne sait pas si elle a un papillon ?

Pour vérifier si A sait que C ne sait pas si elle a un papillon, nous devons vérifier :

$$M \models K_A(\neg K_C C \wedge \neg K_C \neg C)$$

En notation LoTREC :

```
nec R1 (and (not nec R3 C) (not nec R3 (not C)))
```

Nous vérifions cette formule dans tous les mondes :

```
add wA isItTrue nec R1 (and (not nec R3 C) (not nec R3 (not C)))
add wB isItTrue nec R1 (and (not nec R3 C) (not nec R3 (not C)))
add wC isItTrue nec R1 (and (not nec R3 C) (not nec R3 (not C)))
```

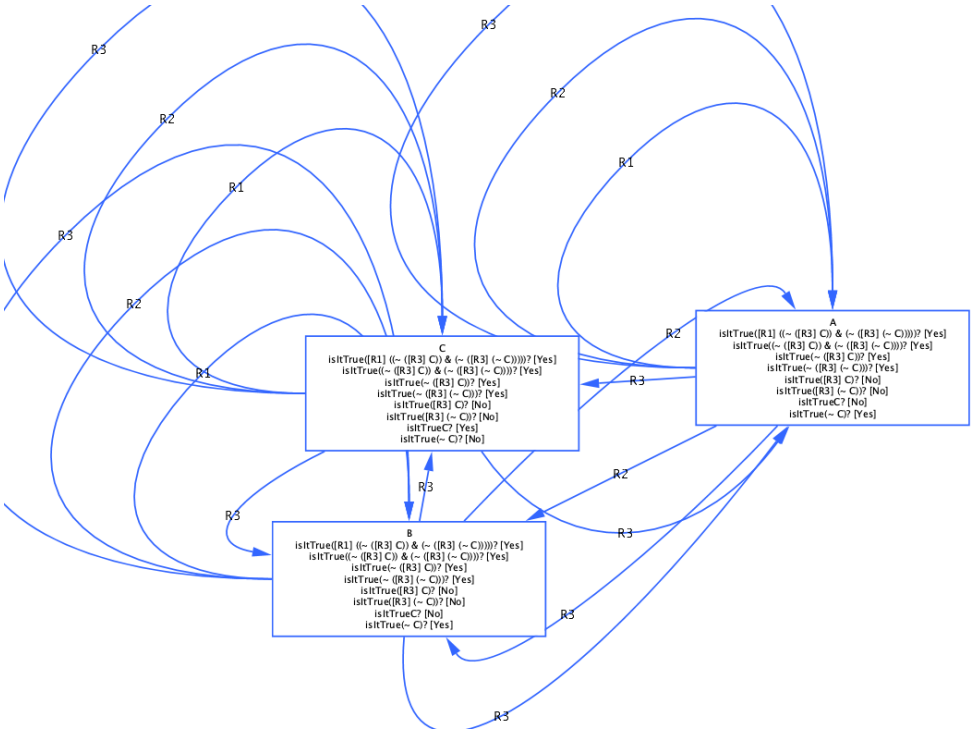


FIGURE 15 – Vérification de $K_A(\neg K_C C \wedge \neg K_C \neg C)$ dans les trois mondes

Résultat : Comme le montre la Figure 15, la formule est **Yes** dans tous les mondes.

Explication :

— Nous avons déjà montré que $\neg K_C C \wedge \neg K_C \neg C$ est vrai dans tous les mondes (C ne sait jamais si elle a le papillon).

— La relation R1 de A n'a que des boucles réflexives, donc A a une connaissance parfaite : dans chaque monde, A connaît la vérité de toutes les formules.

— Par conséquent, A sait que C ne sait pas si elle a le papillon, et ce dans tous les mondes.

4.3.4 Quand B sait si elle a un papillon, A sait-elle que B le sait ?

Pour vérifier cette propriété, nous devons vérifier l'implication :

$$M \models (K_B B \vee K_B \neg B) \rightarrow K_A(K_B B \vee K_B \neg B)$$

En notation LoTREC :

```
imp (or (nec R2 B) (nec R2 (not B))) (nec R1 (or (nec R2 B) (nec R2
    (not B))))
```

Nous vérifions cette formule dans tous les mondes :

```
add wA isItTrue imp (or (nec R2 B) (nec R2 (not B))) (nec R1 (or (
   nec R2 B) (nec R2 (not B))))
add wB isItTrue imp (or (nec R2 B) (nec R2 (not B))) (nec R1 (or (
   nec R2 B) (nec R2 (not B))))
add wC isItTrue imp (or (nec R2 B) (nec R2 (not B))) (nec R1 (or (
   nec R2 B) (nec R2 (not B))))
```

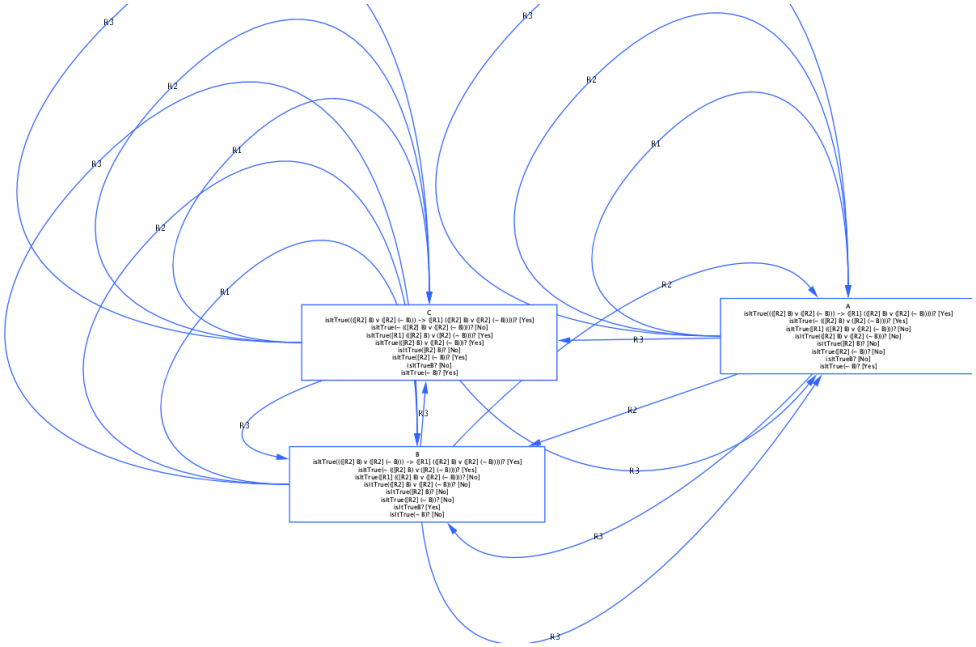


FIGURE 16 – Vérification de $(K_B B \vee K_B \neg B) \rightarrow K_A(K_B B \vee K_B \neg B)$ dans les trois mondes

Résultat : Comme le montre la Figure 16, cette formule est **Yes** dans tous les mondes.

Explication :

— En wA et wB : $K_B B \vee K_B \neg B$ est faux, donc l'implication est vraie.

— En wC : $K_B B \vee K_B \neg B$ est vrai. Comme A a une connaissance parfaite (R1 n'a que des boucles réflexives), A sait tout ce qui est vrai dans le monde actuel. Donc $K_A(K_B B \vee K_B \neg B)$ est vrai en wC.

— Ainsi, l'implication est vraie dans tous les mondes.

Conclusion : Il est vrai que **quand B sait si elle a un papillon sur la tête, alors A sait que B le sait**.

4.4 Résumé des résultats

Propriété	wA	wB	wC
$K_A A \vee K_A \neg A$	Yes	Yes	Yes
$\neg K_C C \wedge \neg K_C \neg C$	Yes	Yes	Yes
$K_B B \vee K_B \neg B$	No	No	Yes
$C \rightarrow (K_B B \vee K_B \neg B)$	Yes	Yes	Yes
$K_A(\neg K_C C \wedge \neg K_C \neg C)$	Yes	Yes	Yes
$(K_B B \vee K_B \neg B) \rightarrow K_A(K_B B \vee K_B \neg B)$	Yes	Yes	Yes

TABLE 2 – Résultats des vérifications pour l'exercice des trois femmes sur l'escalier

5 Exercice 4 : Modèle de Kripke et logique épistémique S5

Cet exercice est basé sur un examen de 2ème session de 2015. Nous considérons le modèle de Kripke M suivant avec quatre mondes et trois agents (1, 2 et 3).

5.1 Description du modèle

Le modèle M est composé de quatre mondes avec les valuations suivantes :

— $w_1 = \{b, c\}$ (b et c vrais, a faux)

— $w_2 = \{a, b\}$ (a et b vrais, c faux)

— $w_3 = \{a\}$ (a vrai, b et c faux)

— $w_4 = \{c\}$ (c vrai, a et b faux)

Les relations d'accessibilité (non-étiquetées initialement) sont les suivantes :

— w_1 est connecté à w_2 et w_3

— w_2 est connecté à w_1 et w_4

— w_3 est connecté à w_1 et w_4

— w_4 est connecté à w_2 et w_3

5.2 Étiquetage des relations par les agents

On suppose qu'il y a trois agents (1, 2 et 3) et que les relations de réflexivité avec ces trois agents sont présentes mais pas explicitement représentées. Nous proposons l'étiquetage suivant :

— Relation $w_1 - w_2$: agents 1 et 2

— Relation $w_1 - w_3$: agent 3

— Relation $w_2 - w_4$: agent 3

— Relation $w_3 - w_4$: agent 1

Chaque agent a également des boucles réflexives (implicites) sur tous les mondes, conformément à S5.

5.3 Construction du modèle dans LoTREC

Nous avons implémenté ce modèle dans LoTREC en utilisant la logique `Model-Checking-Multimodal`. Voici les commandes LoTREC pour construire le modèle :

```
createNewNode w1
createNewNode w2
createNewNode w3
createNewNode w4

# Valuation des propositions (A, B, C en majuscules)
add w1 B
add w1 C
add w2 A
add w2 B
add w3 A
add w4 C

# Relation R1 (Agent 1)
link w1 w1 R1
link w2 w2 R1
link w3 w3 R1
link w4 w4 R1
link w1 w2 R1
link w2 w1 R1
link w3 w4 R1
link w4 w3 R1

# Relation R2 (Agent 2)
link w1 w1 R2
link w2 w2 R2
link w3 w3 R2
link w4 w4 R2
link w1 w2 R2
link w2 w1 R2

# Relation R3 (Agent 3)
link w1 w1 R3
link w2 w2 R3
link w3 w3 R3
link w4 w4 R3
link w1 w3 R3
link w3 w1 R3
link w2 w4 R3
link w4 w2 R3
```

Listing 5 – Commandes LoTREC pour construire le modèle de l'Exercice 4

5.3.1 Modèle obtenu

Après exécution des commandes de construction, LoTREC génère le modèle de Kripke représenté dans la Figure 17.

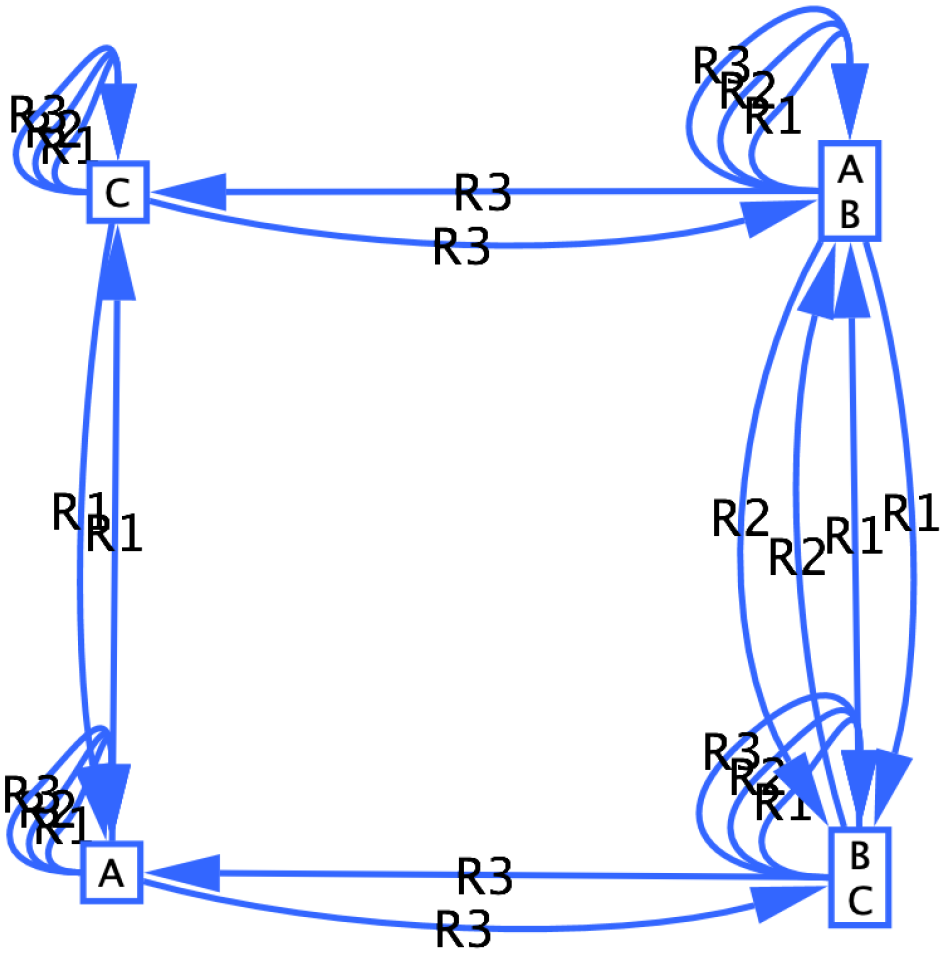


FIGURE 17 – Modèle de Kripke pour l'Exercice 4 avec étiquettes d'agents

5.4 Vérification pas à pas des formules

Nous allons maintenant vérifier les trois formules une par une en ajoutant les commandes correspondantes.

5.4.1 Vérification de la formule (a) : $\neg K_3 b \wedge \neg K_3 \neg b$

Nous ajoutons les commandes suivantes pour vérifier la formule (a) dans tous les mondes :

```
# Verification formule (a)
add w1 isItTrue and (not nec R3 B) (not nec R3 (not B))
add w2 isItTrue and (not nec R3 B) (not nec R3 (not B))
add w3 isItTrue and (not nec R3 B) (not nec R3 (not B))
add w4 isItTrue and (not nec R3 B) (not nec R3 (not B))
```

Listing 6 – Vérification de la formule (a) dans LoTREC

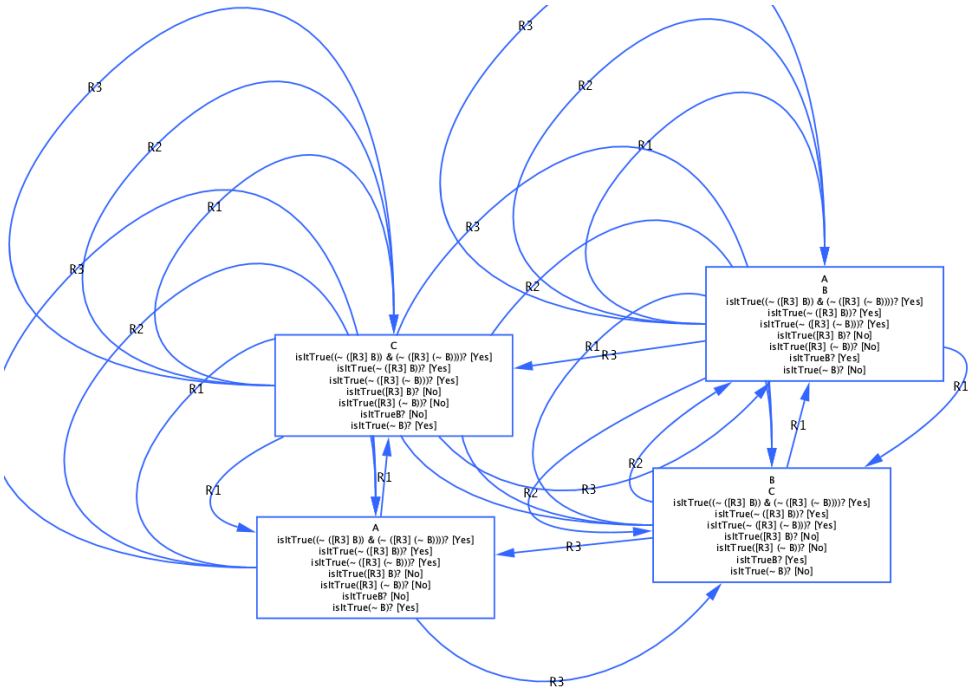


FIGURE 18 – Résultat de la vérification de la formule (a) dans tous les mondes

Résultat : Comme le montre la Figure 18, la formule `isItTrue and (not nec R3 B) (not nec R3 (not B))` est marquée comme **Yes** dans tous les mondes w1, w2, w3 et w4.

5.4.2 Vérification de la formule (b) : $\neg K_1 c \wedge (K_1 b \vee K_1 \neg b)$

Nous ajoutons les commandes suivantes pour vérifier la formule (b) dans tous les mondes :

```
# Verification formule (b)
add w1 isItTrue and (not nec R1 C) (or (nec R1 B) (nec R1 (not B)))
add w2 isItTrue and (not nec R1 C) (or (nec R1 B) (nec R1 (not B)))
add w3 isItTrue and (not nec R1 C) (or (nec R1 B) (nec R1 (not B)))
add w4 isItTrue and (not nec R1 C) (or (nec R1 B) (nec R1 (not B)))
```

Listing 7 – Vérification de la formule (b) dans LoTREC

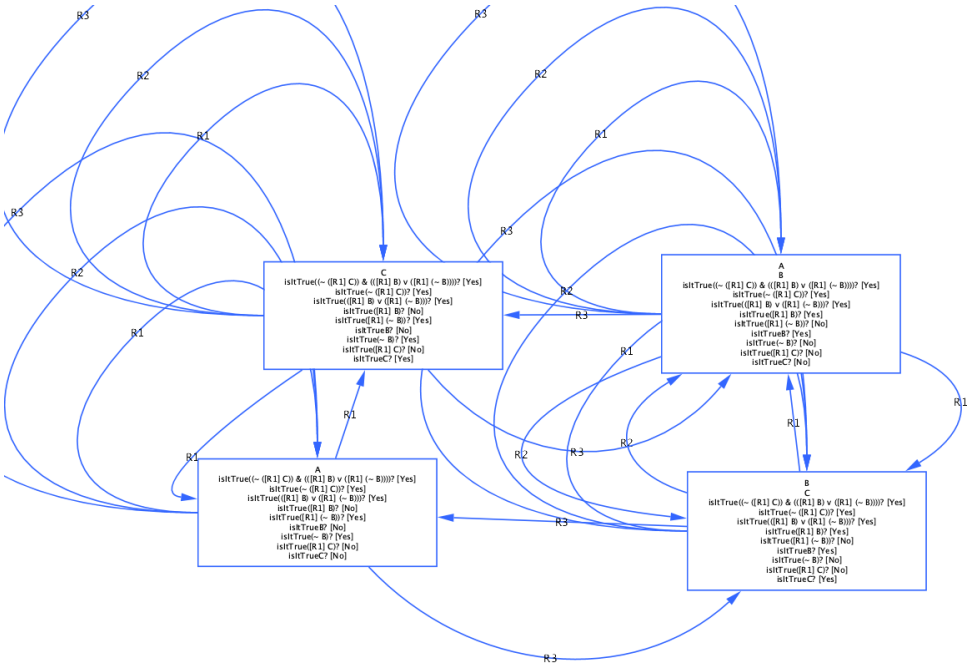


FIGURE 19 – Résultat de la vérification de la formule (b) dans tous les mondes

Résultat : Comme le montre la Figure 19, la formule `isItTrue and (not nec R1 C) (or (nec R1 B) (nec R1 (not B)))` est marquée comme **Yes** dans tous les mondes w1, w2, w3 et w4.

5.4.3 Vérification de la formule (c) : $K_2 b \vee K_2(a \wedge \neg b) \vee K_2(c \wedge \neg b)$

Nous ajoutons les commandes suivantes pour vérifier la formule (c) dans tous les mondes :

```
# Verification formule (c)
add w1 isItTrue or (nec R2 B) (or (nec R2 (and A (not B))) (nec R2
   (and C (not B))))
add w2 isItTrue or (nec R2 B) (or (nec R2 (and A (not B))) (nec R2
   (and C (not B))))
add w3 isItTrue or (nec R2 B) (or (nec R2 (and A (not B))) (nec R2
   (and C (not B))))
add w4 isItTrue or (nec R2 B) (or (nec R2 (and A (not B))) (nec R2
   (and C (not B))))
```

Listing 8 – Vérification de la formule (c) dans LoTREC

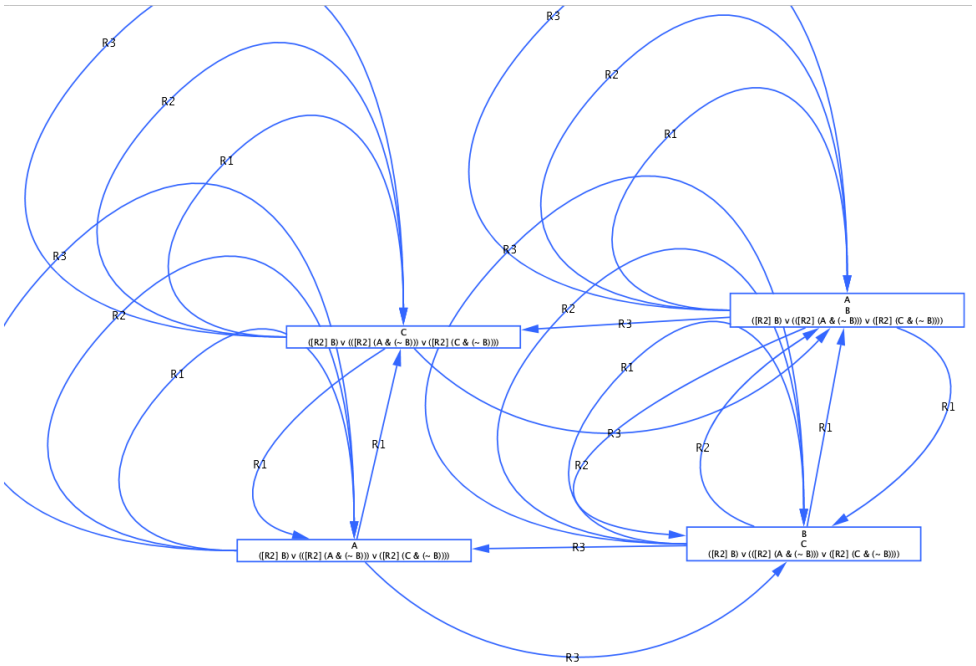


FIGURE 20 – Résultat de la vérification de la formule (c) dans tous les mondes

Résultat : Comme le montre la Figure 20, la formule `isItTrue or (nec R2 B) (or (nec R2 (and A (not B))) (nec R2 (and C (not B))))` est marquée comme **Yes** dans tous les mondes w1, w2, w3 et w4.

5.5 Analyse des résultats

5.5.1 Formule (a) : $M \models \neg K_3 b \wedge \neg K_3 \neg b$

Cette formule exprime que l'agent 3 ne sait pas si b est vrai et ne sait pas si b est faux.

Justification :

— Pour R3, les classes d'équivalence sont $\{w_1, w_3\}$ et $\{w_2, w_4\}$

— Dans $\{w_1, w_3\}$: $w_1 \models b$ mais $w_3 \models \neg b$

— Dans $\{w_2, w_4\}$: $w_2 \models b$ mais $w_4 \models \neg b$

Ainsi, dans chaque classe d'équivalence, il y a à la fois un monde où b est vrai et un monde où b est faux. L'agent 3 ne peut donc pas distinguer si b est vrai ou faux.

5.5.2 Formule (b) : $M \models \neg K_1 c \wedge (K_1 b \vee K_1 \neg b)$

Cette formule exprime que l'agent 1 ne sait pas si c est vrai, mais sait soit que b est vrai, soit que b est faux.

Justification :

— Pour R1, les classes d'équivalence sont $\{w_1, w_2\}$ et $\{w_3, w_4\}$

— Pour $\neg K_1 c$:

  — Dans $\{w_1, w_2\}$: $w_1 \models c$ mais $w_2 \models \neg c$

  — Dans $\{w_3, w_4\}$: $w_4 \models c$ mais $w_3 \models \neg c$

  Donc l'agent 1 ne sait pas si c est vrai.

— Pour $K_1 b \vee K_1 \neg b$:

— Dans $\{w_1, w_2\}$: tous les mondes satisfont b, donc $K_1 b$ est vrai

— Dans $\{w_3, w_4\}$: tous les mondes satisfont $\neg b$, donc $K_1 \neg b$ est vrai

5.5.3 Formule (c) : $M \models K_2 b \vee K_2(a \wedge \neg b) \vee K_2(c \wedge \neg b)$

Cette formule exprime que l'agent 2 sait soit que b est vrai, soit que $a \wedge \neg b$ est vrai, soit que $c \wedge \neg b$ est vrai.

Justification :

— Pour R2, les classes d'équivalence sont $\{w_1, w_2\}$, $\{w_3\}$, $\{w_4\}$

— Classe $\{w_1, w_2\}$: tous les mondes satisfont b, donc $K_2 b$ est vrai

— Classe $\{w_3\}$: $w_3 \models a \wedge \neg b$, donc $K_2(a \wedge \neg b)$ est vrai

— Classe $\{w_4\}$: $w_4 \models c \wedge \neg b$, donc $K_2(c \wedge \neg b)$ est vrai

6 Conclusion

Ce TME nous a permis d'explorer en profondeur les concepts de la logique épistémique à travers l'utilisation pratique de l'outil LoTREC. L'étude de la logique S5 a mis en évidence les propriétés fondamentales de la connaissance parfaite, caractérisée par la réflexivité, la transitivité et la symétrie des relations d'accessibilité, qui assurent respectivement la véracité, l'introspection positive et l'introspection négative des croyances épistémiques.

La vérification systématique de formules dans S5, puis la comparaison avec S4, nous a montré l'importance cruciale de la symétrie (ou propriété euclidienne) pour valider le principe d'introspection négative $\neg Kp \Leftrightarrow K\neg Kp$. Cette analyse comparative illustre comment les propriétés algébriques des cadres de Kripke se traduisent directement en propriétés épistémiques des agents.

Dans le cadre multiagents, la modélisation de l'exercice des trois femmes sur l'escalier a démontré la puissance expressive des modèles de Kripke pour représenter des situations complexes de connaissance et d'ignorance. La construction pas à pas du modèle et la vérification des formules nous ont permis de valider les raisonnements épistémiques des agents et d'identifier précisément les conditions dans lesquelles chaque agent acquiert des connaissances.

L'implémentation dans LoTREC a particulièrement mis en lumière :

— La modularité des stratégies de preuve, avec une séparation claire entre le traitement propositionnel et les règles spécifiques à chaque logique modale

— L'importance des optimisations comme `MarkFulfilledPos` pour contrôler l'explosion combinatoire dans les logiques avec symétrie

— La flexibilité du système pour gérer plusieurs relations d'accessibilité simultanément, essentielle pour les applications multiagents

Ce travail pratique a également souligné certaines limites et défis :

— La nécessité d'adapter les connecteurs logiques selon les logiques prédéfinies (comme l'absence de `equiv` en S4-Explicit-R)

— L'importance de bien comprendre la sémantique des modèles pour interpréter correctement les résultats des vérifications

— La complexité croissante de la construction manuelle des modèles pour des problèmes multiagents plus élaborés

En conclusion, ce TME a renforcé notre compréhension théorique des logiques épistémiques tout en développant nos compétences pratiques avec LoTREC. L'articulation entre théorie (propriétés des cadres de Kripke) et pratique (implémentation et vérification) s'est avérée essentielle pour maîtriser les subtilités du raisonnement épistémique. Ces compétences sont fondamentales pour aborder des problèmes plus complexes en intelligence artificielle distribuée, en théorie des jeux épistémiques, ou dans toute application nécessitant une modélisation formelle des connaissances et croyances d'agents rationnels.