

LRC TME8 - Logique Modale avec LoTREC

Yuxiang ZHANG
Kenan ALSAFADI

Novembre 2025

Table des matières

1	Introduction	2
2	Vérification de modèles dans la logique K	2
2.1	Exécution pas à pas de la stratégie	2
2.2	Règles appliquées dans l'exécution pas à pas	2
2.3	Vérification dans LoTREC du modèle de l'Exercice 1 (TD7)	3
2.3.1	Construction du modèle dans LoTREC	3
2.3.2	Commandes LoTREC pour la vérification des formules	3
2.3.3	Résultats attendus et observations	4
2.4	Création d'un modèle minimal satisfaisant deux contraintes	4
2.4.1	Description du modèle M1	4
2.4.2	Construction dans LoTREC	5
2.4.3	Vérification des contraintes	5
2.4.4	Minimalité du modèle	6
2.5	Ajout du connecteur d'implication	7
2.5.1	Description de l'opérateur	7
2.5.2	Test de l'implication	8
3	Satisfiabilité dans la logique K	9
3.1	Examen de la formule $\Diamond p \wedge \Box \neg p$	9
3.1.1	Procédure	9
3.1.2	Résultats observés	9
3.1.3	Analyse pas à pas	10
3.2	Traitement d'autres formules	10
3.2.1	Formule (a) : $p \wedge \Diamond(q \wedge \Box \neg p)$	10
3.2.2	Formule (b) : $(p \wedge \neg p) \vee \Diamond \Diamond p$	11
3.2.3	Formule (c) : $(p \rightarrow \Diamond(q \vee \neg p)) \vee q$	12
3.3	Utilisation pour montrer la validité	13
3.3.1	Test de la satisfiabilité de la formule originale	13
3.3.2	Test de la satisfiabilité de la négation de la formule	14
3.3.3	Analyse de la contradiction	14
3.3.4	Conclusion	15
4	Bilan et analyse des algorithmes	15
4.1	Algorithmes propositionnels vs modaux : analyse comparative	15
4.2	Différences fondamentales entre tableau modal et tableau propositionnel	16
4.3	Règles spécifiques aux modalités : exemples contrastés	16
4.3.1	Règles de satisfiabilité modale vs propositionnelle	16
4.3.2	Règles de vérification de modèles : double phase	16
4.4	Intégration du connecteur d'implication : une preuve de modularité	17
4.5	Conclusion synthétique	17

1 Introduction

Ce TME a pour but d'utiliser le logiciel LoTREC pour la logique modale minimale K. Nous aborderons deux objectifs distincts : la vérification de modèles (*model checking*) et le test de satisfiabilité. Le rapport présentera les algorithmes mis en œuvre par LoTREC, des exemples détaillés d'utilisation, et les solutions proposées pour étendre la logique.

2 Vérification de modèles dans la logique K

Dans cet exercice, on utilise LoTREC pour vérifier la satisfiabilité d'une formule modale dans un modèle de Kripke. Après avoir cliqué sur *Logic* puis *Predefined Logic*, on choisit *Model-Checking-Monomodal*.

2.1 Exécution pas à pas de la stratégie

La première règle appliquée est *ExampleOfModelAndFormula*. Elle permet de construire le modèle de Kripke M composé de quatre mondes w, u, v, x . Les commandes suivantes y sont exécutées :

```
1 createNewNode w
2 createNewNode u
3 createNewNode v
4 createNewNode x
5
6 link w u R
7 link w v R
8 link v u R
9 link v x R
10
11 add w P
12 add v P
13 add v Q
14 add x Q
15
16 add w isItTrue pos nec or P Q
```

On cherche donc à vérifier en w la formule :

$$M, w \models \Diamond \Box (P \vee Q) ?$$

Après l'exécution de cette règle, LoTREC procède en deux phases :

- **Phase descendante (top-down)** : la formule est décomposée en sous-formules jusqu'à atteindre des éléments atomiques ;
- **Phase montante (bottom-up)** : les valeurs de vérité des sous-formules sont recomposées pour déterminer la valeur de vérité finale de la formule initiale.

Chaque connecteur possède donc deux types de règles : une pour la descente et une pour la remontée. Il est important d'examiner leurs conditions et leurs actions pour comprendre précisément leur fonctionnement.

2.2 Règles appliquées dans l'exécution pas à pas

Lors de l'exécution en mode *step by step*, LoTREC applique les règles suivantes dans cet ordre :

- *ExampleOfModelAndFormula*
- *Pos_Top_Down*
- *Nec_Top_Down*
- *Or_Top_Down*
- *Atom_True_Bottom_Up*
- *Atom_Not_True_Bottom_Up*
- *Or_Not_True_Bottom_Up*
- *Or_Right_True_Bottom_Up*
- *Nec_True_Bottom_Up*

- Nec_Not_True_Bottom_Up
- Pos_True_Bottom_Up

Ces règles illustrent clairement le passage de la décomposition de la formule à la reconstruction de sa valeur de vérité.

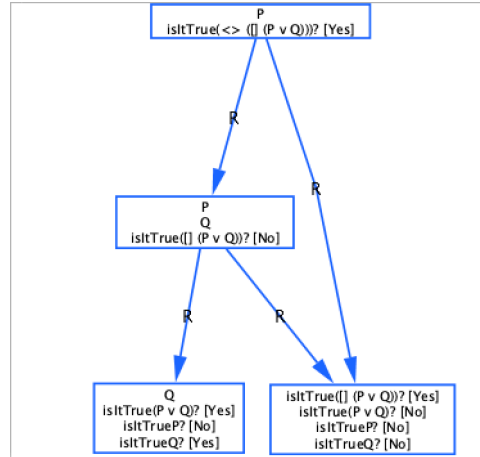


FIGURE 1 – Résultat final de LoTREC : la formule est évaluée à true

2.3 Vérification dans LoTREC du modèle de l'Exercice 1 (TD7)

Nous avons construit dans LoTREC le modèle de Kripke suivant :

$$W = \{w_1, w_2, w_3, w_4\}, \quad R = \{(w_1, w_2), (w_2, w_3), (w_3, w_4), (w_4, w_3), (w_4, w_2)\},$$

et l'interprétation :

$$I(A) = \{w_1, w_3\}, \quad I(B) = \{w_2, w_4\}.$$

2.3.1 Construction du modèle dans LoTREC

```

1 createNewNode w1
2 createNewNode w2
3 createNewNode w3
4 createNewNode w4
5
6 link w1 w2 R
7 link w2 w3 R
8 link w3 w4 R
9 link w4 w3 R
10 link w4 w2 R
11
12 add w1 A
13 add w3 A
14 add w2 B
15 add w4 B

```

2.3.2 Commandes LoTREC pour la vérification des formules

Note importante : À ce stade, le connecteur d'implication `imp` n'a pas encore été défini dans LoTREC. Nous présentons ici l'ensemble des commandes que nous souhaitons tester, y compris celles utilisant l'implication. Cependant, ces dernières n'ont pu être exécutées qu'après avoir défini le connecteur `imp` dans la section 2.5.

```

1 # Formules sans implication (testées immédiatement)
2 add w1 isItTrue B
3 add w1 isItTrue A
4 add w1 isItTrue pos A
5 add w1 isItTrue nec B

```

```

6
7 add w4 isItTrue and nec B nec A
8 add w4 isItTrue and pos B pos A
9 add w4 isItTrue or nec B nec A
10
11 add w1 isItTrue nec pos B
12 add w3 isItTrue pos pos pos A
13 add w3 isItTrue nec nec nec A
14
15 # Formules avec implication (testées APRÈS la section 4)
16 # Ces commandes échoueraient si exécutées avant la définition de 'imp'
17 add w2 isItTrue pos (imp A nec B)
18
19 add w1 isItTrue (imp A pos B)
20 add w2 isItTrue (imp A pos B)
21 add w3 isItTrue (imp A pos B)
22 add w4 isItTrue (imp A pos B)
23
24 add w1 isItTrue nec (imp A pos B)
25 add w2 isItTrue nec (imp A pos B)
26 add w3 isItTrue nec (imp A pos B)
27 add w4 isItTrue nec (imp A pos B)

```

2.3.3 Résultats attendus et observations

Les formules sans implication ont pu être testées immédiatement :

- $M, w_1 \models B$: **faux**
- $M, w_1 \models A$: **vrai**
- $M, w_1 \models \Diamond A$: **faux**
- $M, w_1 \models \Box B$: **vrai**
- $M, w_4 \models \Box B \wedge \Box A$: **faux**
- $M, w_4 \models \Diamond B \wedge \Diamond A$: **vrai**
- $M, w_4 \models \Box B \vee \Box A$: **faux**
- $M, w_1 \models \Box \Diamond B$: **faux**
- $M, w_3 \models \Diamond \Diamond A$: **vrai**
- $M, w_3 \models \Box \Box A$: **faux**

Pour les formules avec implication, nous avons dû attendre d’avoir défini le connecteur `imp` dans la section 2.5. Une fois ce connecteur défini, nous avons pu exécuter les commandes restantes et obtenir les résultats suivants :

- $M, w_2 \models \Diamond(A \rightarrow \Box B)$: **vrai**
- $M \models A \rightarrow \Diamond B$: **vrai** (vrai dans tous les mondes)
- $M \models \Box(A \rightarrow \Diamond B)$: **vrai**

2.4 Création d’un modèle minimal satisfaisant deux contraintes

2.4.1 Description du modèle M1

Nous avons construit un modèle de Kripke M1 minimal composé de deux mondes seulement : w et u . Les relations d’accessibilité et les interprétations sont définies comme suit :

- **Mondes** : $W = \{w, u\}$
- **Relation d’accessibilité** : $R = \{(w, u), (u, u)\}$ (une flèche de w vers u et une boucle sur u)
- **Interprétation** :
 - $I(P) = \{u\}$ (la proposition p est vraie seulement en u)
 - $I(Q) = \{w\}$ (la proposition q est vraie seulement en w)

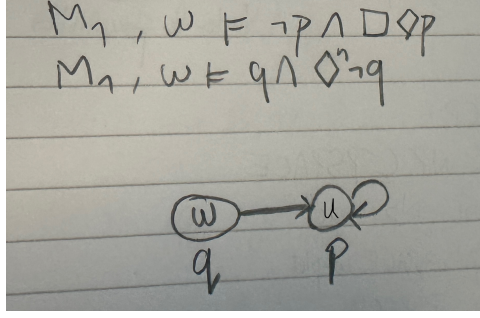


FIGURE 2 – Représentation schématique du modèle M1

2.4.2 Construction dans LoTREC

Le modèle M1 a été implémenté dans LoTREC avec les commandes suivantes :

```

1 createNewNode w
2 createNewNode u
3 link w u R
4 link u u R
5 add w Q
6 add u P

```

2.4.3 Vérification des contraintes

Première contrainte : $\neg p \wedge \Box \Diamond p$ en w Nous vérifions cette formule avec la commande :

```

1 add w isItTrue and not P nec pos P

```

Résultat : La formule est validée (Yes) comme le montre la capture d'écran.

Justification :

- En w , p est faux ($\neg p$ vrai) car P n'est pas dans l'interprétation de w
- Pour $\Box \Diamond p$ en w : le seul monde accessible depuis w est u
- En u , $\Diamond p$ est vrai car u a un successeur (lui-même) où p est vrai
- Donc $M1, w \models \neg p \wedge \Box \Diamond p$

Deuxième contrainte : $q \wedge \Diamond^n \neg q$ pour tout n Nous testons pour différentes valeurs de n :

```

1 # Pour n = 1
2 add w isItTrue and Q pos not Q
3
4 # Pour n = 2
5 add w isItTrue and Q pos pos not Q
6
7 # Pour n = 3
8 add w isItTrue and Q pos pos pos not Q

```

Résultats : Toutes ces formules sont validées (Yes) comme le montre la capture d'écran pour $n = 3$.

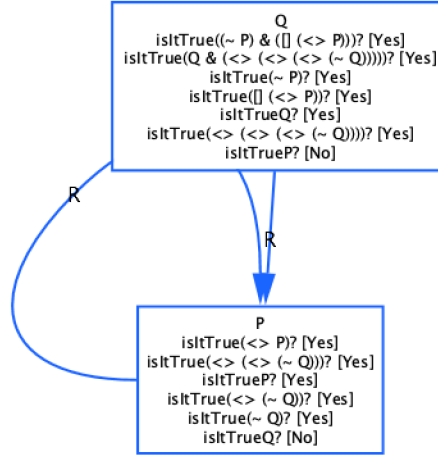


FIGURE 3 – Résultats de vérification dans LoTREC

Analyse des résultats LoTREC : La capture d'écran montre le processus de vérification :

1. `isItTrue((~P) & ([< P]))? [Yes]` : vérification de $\neg p \wedge \Box \Diamond p$
 - `isItTrue(~P)? [Yes]` : $\neg p$ est vrai en w
 - `isItTrue([< P])? [Yes]` : $\Box \Diamond p$ est vrai en w
2. `isItTrue(Q & (< (< (< (~Q))))? [Yes]` : vérification de $q \wedge \Diamond \Diamond \Diamond \neg q$ ($n=3$)
 - `isItTrue(Q)? [Yes]` : q est vrai en w
 - `isItTrue(< (< (< (~Q))))? [Yes]` : $\Diamond \Diamond \Diamond \neg q$ est vrai en w
 - `isItTrue(P)? [No]` : p est faux en w (vérification intermédiaire)
3. `isItTrue(< P)? [Yes]` : vérification de $\Diamond p$ en u
 - `isItTrue(P)? [Yes]` : p est vrai en u (accessible par la boucle)
4. `isItTrue(< (< (~Q)))? [Yes]` : vérification de $\Diamond \Diamond \neg q$ en u
5. `isItTrue(< (~Q))? [Yes]` : vérification de $\Diamond \neg q$ en u
 - `isItTrue(~Q)? [Yes]` : $\neg q$ est vrai en u
 - `isItTrue(Q)? [No]` : q est faux en u

Interprétation :

- Pour $n = 3$ ($\Diamond \Diamond \Diamond \neg q$), le chemin est : $w \rightarrow u \rightarrow u \rightarrow u$
- Chaque étape \Diamond correspond à suivre la relation R
- Le dernier monde (u) satisfait $\neg q$ (car q n'est vrai qu'en w)
- La vérification de $\Diamond p$ en u montre que p est vrai en u (accessible par la boucle)
- La vérification de $\neg p$ en w montre que p est faux en w

Justification pour tout n :

- En w , q est vrai (Q présent)
- Depuis w , on peut atteindre u en 1 pas (wRu)
- En u , q est faux (Q absent)
- Pour $n = 1$: $w \rightarrow u$ (direct, $\Diamond \neg q$ vrai)
- Pour $n > 1$: $w \rightarrow u \rightarrow u \rightarrow \dots \rightarrow u$ (on prend la boucle sur u $n - 1$ fois)
- Puisque u a une boucle (uRu), on peut créer des chemins de longueur arbitraire
- Le dernier monde de tout chemin de longueur n sera u où $\neg q$ est vrai
- Donc $\forall n \geq 1, M1, w \models \Diamond^n \neg q$

2.4.4 Minimalité du modèle

Notre modèle utilise deux mondes, ce qui est minimal sous l'hypothèse que le monde initial possède au moins un successeur. En effet :

1. Satisfaire $\Box \Diamond p$ en w exige qu'un successeur de w rende $\Diamond p$ vrai.

2. Satisfaire $\Diamond^n \neg q$ pour tout n nécessite une boucle permettant des chemins arbitrairement longs.
3. Ces deux contraintes peuvent être satisfaites avec un unique successeur u muni d'une boucle.
4. Un modèle à un seul monde ne satisfait pas simultanément nos contraintes dès lors que l'on impose l'existence d'un successeur pour évaluer $\Box \Diamond p$ de manière non triviale.

2.5 Ajout du connecteur d'implication

2.5.1 Description de l'opérateur

Le connecteur d'implication n'est pas défini par défaut dans la logique prédéfinie *Model-Checking-Monomodal*. Pour l'ajouter, nous suivons les étapes suivantes :

1. Création dans l'onglet *Connectors* Dans l'onglet *Connectors*, nous ajoutons un nouveau connecteur avec les paramètres suivants :

- **Nom** : `imp` (pour implication)
- **Arity** : 2 (connecteur binaire)
- **Display** : `_->_` (affichage dans l'interface graphique)
- **Priority** : 0 (identique aux autres connecteurs, car LoTREC utilise la notation préfixe)
- **Associative** : Non cochée (l'implication n'est pas associative)

2. Ajout des règles dans l'onglet *Rules* Nous devons ajouter quatre règles pour l'implication : une pour la phase descendante et trois pour la phase montante.

```

1 Rule Name: Imp_Top_Down
2
3 Conditions:
4 hasElement w isItTrue imp variable A variable B
5
6 Actions:
7 add w isItTrue not variable A
8 add w isItTrue variable B

```

Cette règle décompose l'implication $A \rightarrow B$ en deux sous-formules : $\neg A$ et B , qui seront évaluées séparément lors de la phase descendante.

```

1 Rule Name: Imp_Left_True_Bottom_Up
2
3 Conditions:
4 hasElement w isItTrue imp variable A variable B
5 isMarkedExpression w isItTrue not variable A Yes
6
7 Actions:
8 markExpressions w isItTrue imp variable A variable B Yes

```

Cette règle valide l'implication lorsque $\neg A$ est vrai (marqué Yes), conformément à la sémantique de l'implication : si l'antécédent est faux, l'implication est vraie.

```

1 Rule Name: Imp_Right_True_Bottom_Up
2
3 Conditions:
4 hasElement w isItTrue imp variable A variable B
5 isMarkedExpression w isItTrue variable B Yes
6
7 Actions:
8 markExpressions w isItTrue imp variable A variable B Yes

```

Cette règle valide l'implication lorsque B est vrai (marqué Yes), ce qui correspond au cas où le conséquent est vrai, rendant l'implication vraie indépendamment de la valeur de A .

```

1 Rule Name: Imp_Not_True_Bottom_Up
2
3 Conditions:

```

```

4 hasElement w isItTrue imp variable A variable B
5 isMarkedExpression w isItTrue not variable A No
6 isMarkedExpression w isItTrue variable B No
7
8 Actions:
9 markExpressions w isItTrue imp variable A variable B No

```

Cette règle invalide l'implication lorsque $\neg A$ est faux (c'est-à-dire A vrai) et B est faux (marqués No), ce qui est le seul cas où l'implication $A \rightarrow B$ est fausse.

3. Ajout à la stratégie dans l'onglet *Strategies* Dans l'onglet *Strategies*, nous ajoutons les nouvelles règles aux stratégies existantes comme suit :

— Dans *Top_Down_Strategy*, nous insérons *Imp_Top_Down* :

```

1 Top_Down_Strategy
2 repeat
3   Not_Top_Down
4   And_Top_Down
5   Or_Top_Down
6   Imp_Top_Down      # Nouvelle règle
7   Nec_Top_Down
8   Pos_Top_Down
9 end

```

Cette position assure que l'implication est décomposée après les connecteurs booléens standards mais avant les modalités.

— Dans *Bottom_Up_Strategy*, nous insérons les trois règles de remontée pour l'implication *Imp_Left_True_Bottom_Up* et *Imp_Right_True_Bottom_Up* et *Imp_Not_True_Bottom_Up* :

```

1 Bottom_Up_Strategy
2 repeat
3   Atom_True_Bottom_Up
4   Atom_Not_True_Bottom_Up
5   Not_True_Bottom_Up
6   Not_Not_True_Bottom_Up
7   And_True_Bottom_Up
8   And_Left_Not_True_Bottom_Up
9   And_Right_Not_True_Bottom_Up
10  Or_Not_True_Bottom_Up
11  Or_Left_True_Bottom_Up
12  Or_Right_True_Bottom_Up
13  Imp_Left_True_Bottom_Up    # Nouvelle règle
14  Imp_Right_True_Bottom_Up  # Nouvelle règle
15  Imp_Not_True_Bottom_Up    # Nouvelle règle
16  Nec_True_Bottom_Up
17  Nec_Not_True_Bottom_Up
18  Pos_Not_True_Bottom_Up
19  Pos_True_Bottom_Up
20 end

```

Cette position garantit que les règles d'implication s'appliquent après que les valeurs de $\neg A$ et B aient été déterminées.

2.5.2 Test de l'implication

Pour tester notre nouvel opérateur, nous utilisons la formule $p \rightarrow \Box(q \vee p)$ dans un modèle de Kripke approprié. Dans notre test, nous avons construit un modèle avec plusieurs mondes et relations d'accessibilité. La capture d'écran montre l'exécution de la vérification.

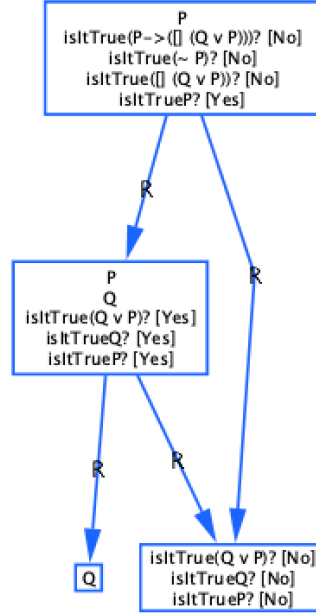


FIGURE 4 – Résultat du test de l'implication dans LoTREC

Analyse des résultats : La capture d'écran montre que la formule $p \rightarrow \Box(q \vee p)$ a été évaluée à No (faux) dans le monde courant. L'exécution pas à pas révèle les étapes suivantes :

- La formule est décomposée en $\neg p$ et $\Box(q \vee p)$.
- $\neg p$ est évalué à No (car p est vrai dans le monde courant).
- $\Box(q \vee p)$ est évalué à No (car il existe un monde accessible où $q \vee p$ est faux).
- Puisque $\neg p$ est faux et $\Box(q \vee p)$ est faux, la règle `Imp_Not_True_Bottom_Up` s'applique et marque l'implication comme No.

Ce résultat est cohérent avec la sémantique de l'implication : p est vrai, mais $\Box(q \vee p)$ est faux, donc $p \rightarrow \Box(q \vee p)$ est faux.

Remarque sur la notation : Dans LoTREC, l'implication est écrite en notation préfixe : `(imp P nec (or Q P))` correspond à $p \rightarrow \Box(q \vee p)$. L'affichage `->` est utilisé uniquement pour la visualisation dans l'interface graphique.

3 Satisfiabilité dans la logique K

Pour tester la satisfiabilité, nous cliquons sur l'onglet *Logic* et choisissons *Predefined Logics* puis *Monomodal-K*.

3.1 Examen de la formule $\Diamond p \wedge \Box \neg p$

3.1.1 Procédure

Nous entrons la formule $\Diamond p \wedge \Box \neg p$ (en notation préfixe : `and pos P nec not P`) dans le cadre des formules et cliquons sur *Satisfiability Check*.

3.1.2 Résultats observés

LoTREC construit un graphe qui montre que la formule est **insatisfiable**. Le graphe contient une contradiction :

- La formule $\Diamond p$ exige l'existence d'un monde accessible où p est vrai
- La formule $\Box \neg p$ exige que dans tous les mondes accessibles, p soit faux
- Ces deux conditions sont incompatibles dans le même monde

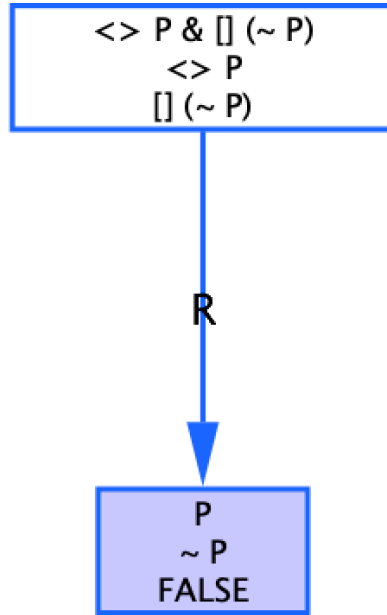


FIGURE 5 – Graphe de satisfiabilité pour $\Diamond p \wedge \Box \neg p$ généré par LoTREC

3.1.3 Analyse pas à pas

En mode pas à pas, nous observons l'application séquentielle des règles dans l'ordre suivant :

1. **Règle And** : La formule $\Diamond p \wedge \Box \neg p$ est décomposée en deux sous-formules distinctes : $\Diamond p$ et $\Box \neg p$.
2. **Règle Pos** : Pour traiter la sous-formule $\Diamond p$, LoTREC crée un nouveau monde accessible depuis le monde courant pour satisfaire p .
3. **Règle Nec** : La sous-formule $\Box \neg p$ exige que dans tous les mondes accessibles depuis le monde courant, $\neg p$ soit vrai, c'est-à-dire que p soit faux. Cette condition s'applique également au nouveau monde créé à l'étape précédente.
4. **Contradiction** : Le nouveau monde doit simultanément satisfaire p (d'après la règle Pos) et $\neg p$ (d'après la règle Nec), ce qui est impossible.
5. **Arrêt (Stop)** : La détection de cette contradiction conduit à la fermeture de la branche et à la conclusion que la formule est insatisfiable.

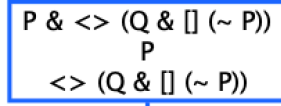
Cette séquence illustre clairement le conflit sémantique au cœur de la formule $\Diamond p \wedge \Box \neg p$: l'existence d'un monde où p est vrai est incompatible avec l'exigence que p soit faux dans tous les mondes accessibles.

3.2 Traitement d'autres formules

3.2.1 Formule (a) : $p \wedge \Diamond(q \wedge \Box \neg p)$

Notation LoTREC : and P pos (and Q nec not P)

Résultat : Satisfiable mais non valide.



R

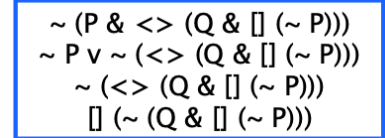
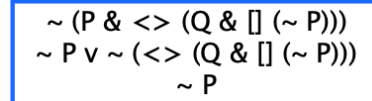
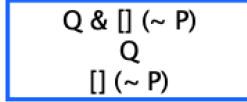


FIGURE 6 – Formule (a) : premodel

FIGURE 7 – Négation de (a) : premodel.1

FIGURE 8 – Négation de (a) : premodel.2

Analyse :

- La formule est satisfiable, comme le montre la figure 6.
- Cependant, sa négation est également satisfiable (figures 7 et 8).
- Notation LoTREC de la négation : not (and P pos (and Q nec not P))
- La satisfiabilité de la négation prouve que la formule originale n'est pas valide
- Modèle construit pour la formule originale :
 - Un monde w où p est vrai
 - Un monde u accessible depuis w où q est vrai et $\Box \neg p$ est vrai
 - Puisque $\Box \neg p$ est vrai en u , tous les mondes accessibles depuis u doivent avoir p faux
 - Ce modèle est cohérent et montre la satisfiabilité

3.2.2 Formule (b) : $(p \wedge \neg p) \vee \Diamond \Diamond \Diamond p$

Notation LoTREC : or (and P not P) pos pos pos P

Résultat : Satisfiable mais non valide.

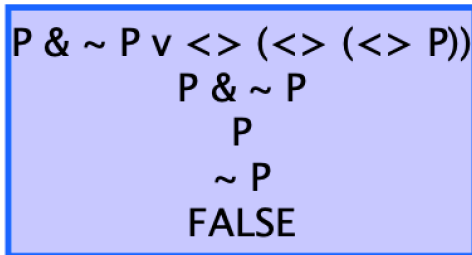


FIGURE 9 – Formule (b) : premodel.1

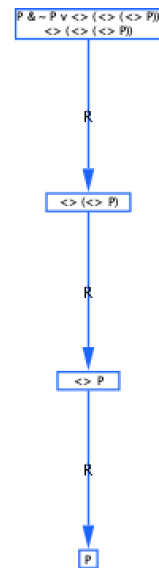


FIGURE 10 – Formule (b) : premodel.2

$$\begin{aligned}
& \sim (P \& \sim P \vee <> (<> (<> P))) \\
& \quad \sim (P \& \sim P) \\
& \quad \sim (<> (<> (<> P))) \\
& \quad \Box (\sim (<> (<> P))) \\
& \quad \quad \sim P \vee \sim (\sim P) \\
& \quad \quad \sim P
\end{aligned}$$

FIGURE 11 – Négation de (b) : premodel.1

$$\begin{aligned}
& \sim (P \& \sim P \vee <> (<> (<> P))) \\
& \quad \sim (P \& \sim P) \\
& \quad \sim (<> (<> (<> P))) \\
& \quad \Box (\sim (<> (<> P))) \\
& \quad \quad \sim P \vee \sim (\sim P) \\
& \quad \quad \sim (\sim P) \\
& \quad \quad P
\end{aligned}$$

FIGURE 12 – Négation de (b) : premodel.2

Analyse :

- La formule est une disjonction : $(p \wedge \neg p) \vee \Diamond\Diamond\Diamond p$
- Comme le montrent les figures 9 et 10, la formule admet deux prémodèles :
 - Le premier prémodèle (figure 9) correspond à la branche $p \wedge \neg p$ et est fermé (contradiction)
 - Le deuxième prémodèle (figure 10) correspond à la branche $\Diamond\Diamond\Diamond p$ et est ouvert
- Puisqu'il existe au moins un prémodèle ouvert, la formule est **satisfiable**.
- Pour déterminer si elle est valide, nous testons sa négation :
 - Notation LoTREC de la négation : not (or (and P not P) pos pos pos P)
 - Les figures 11 et 12 montrent que la négation admet deux prémodèles ouverts
 - La négation est donc également **satisfiable**
- La satisfaisabilité de la négation prouve que la formule originale n'est pas valide

3.2.3 Formule (c) : $(p \rightarrow \Diamond(q \vee \neg p)) \vee q$

Notation LoTREC : or (imp P pos (or Q not P)) Q

Résultat : Satisfiable mais non valide.

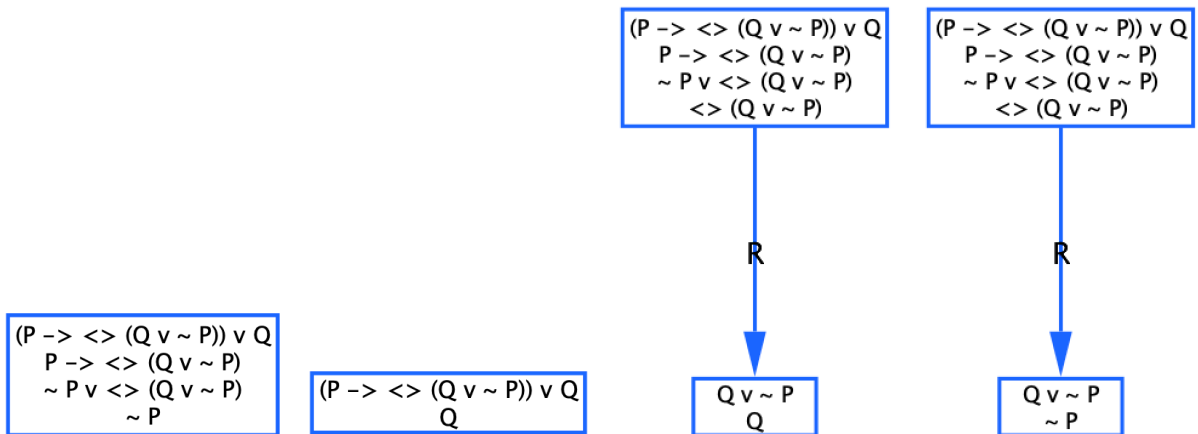


FIGURE 13 – Formule (c) : premodel.1

FIGURE 14 – Formule (c) : premodel.2

FIGURE 15 – Formule (c) : premodel.3.1

FIGURE 16 – Formule (c) : premodel.3.2

$$\begin{array}{c}
\sim ((P \rightarrow \Diamond (Q \vee \sim P)) \vee Q) \\
\sim (P \rightarrow \Diamond (Q \vee \sim P)) \\
\sim Q \\
P \\
\sim (\Diamond (Q \vee \sim P)) \\
\Box (\sim (Q \vee \sim P))
\end{array}$$

FIGURE 17 – Négation de (c) : premodel

Analyse :

- La formule est une disjonction : $(p \rightarrow \Diamond(q \vee \neg p)) \vee q$
- Les figures 13 à 16 montrent différents premodèles ouverts, démontrant que la formule est satisfiable
- Cependant, la négation de la formule (figure 17) est également satisfiable
- Notation LoTREC de la négation : `not (or (imp P pos (or Q not P)) Q)`
- La satisfiabilité de la négation prouve que la formule originale n'est pas valide
- Analyse détaillée :
 - Si q est vrai, la formule est vraie
 - Si q est faux, alors $p \rightarrow \Diamond(q \vee \neg p)$ devient $p \rightarrow \Diamond(\text{false} \vee \neg p)$
 - Qui se simplifie en $p \rightarrow \Diamond \neg p$
 - Cette formule est satisfiable (par exemple, dans un modèle où p est faux, ou où il existe un monde accessible avec $\neg p$)
- Puisque la formule et sa négation sont toutes deux satisfiables, la formule est contingente (ni valide, ni contradiction)

3.3 Utilisation pour montrer la validité

Pour montrer la validité de la formule $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$, nous procédons en deux étapes dans LoTREC.

3.3.1 Test de la satisfiabilité de la formule originale

Nous testons d'abord la satisfiabilité de la formule originale :

Notation LoTREC : `imp (nec (imp P Q)) (imp (nec P) (nec Q))`

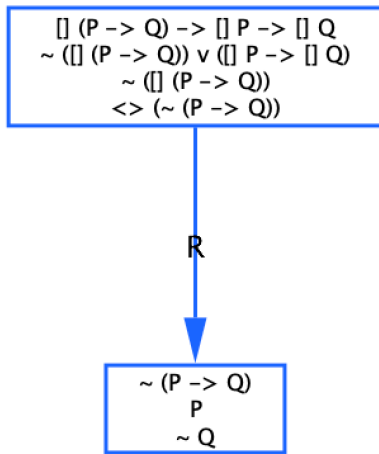


FIGURE 18 – premodel.1

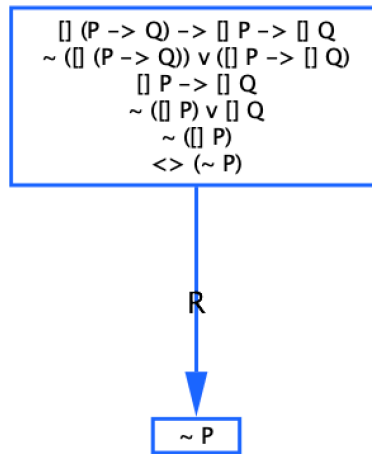


FIGURE 19 – premodel.2.1

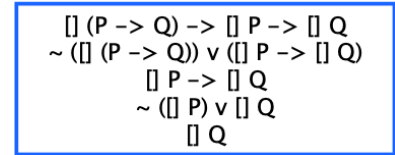


FIGURE 20 – premodel.2.2

Résultat : La formule est **satisfiable**, comme le montrent les trois prémodèles ouverts (figures 18, 19 et 20).

3.3.2 Test de la satisfiabilité de la négation de la formule

Pour déterminer si la formule est valide, nous testons la satisfiabilité de sa négation :

$$\neg[\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)]$$

Notation LoTREC : not (imp (nec (imp P Q)) (imp (nec P) (nec Q)))

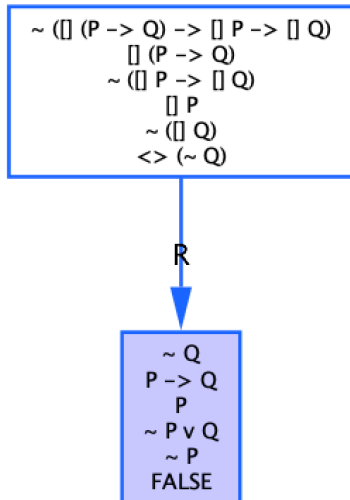


FIGURE 21 – Négation : premodel.1

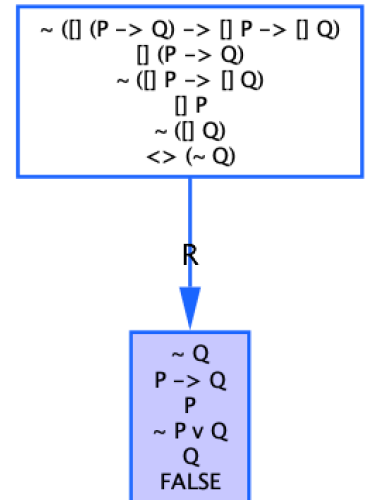


FIGURE 22 – Négation : premodel.2

Résultat : La négation de la formule est **insatisfiable**, car tous les prémodèles sont fermés (figures 21 et 22).

3.3.3 Analyse de la contradiction

La négation de la formule équivaut à :

$$\Box(p \rightarrow q) \wedge \Box p \wedge \neg \Box q$$

La contradiction provient des exigences suivantes :

- $\Box(p \rightarrow q)$: dans tous les mondes accessibles, si p alors q
- $\Box p$: dans tous les mondes accessibles, p est vrai
- $\neg \Box q$: il existe un monde accessible où q est faux

Dans le monde où q est faux, p est vrai (par $\Box p$) et $p \rightarrow q$ est vrai (par $\Box(p \rightarrow q)$), ce qui implique que q doit être vrai. Contradiction avec l'hypothèse que q est faux dans ce monde.

3.3.4 Conclusion

Puisque :

1. La formule originale $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$ est satisfiable (au moins un modèle existe)
2. La négation de la formule est insatisfiable (aucun modèle n'existe)

Nous concluons que la formule $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$ est **valide** dans la logique K. Cela correspond à l'axiome caractéristique de la logique modale K.

4 Bilan et analyse des algorithmes

Ce TME a permis d'expérimenter les algorithmes mis en œuvre par LoTREC pour la logique modale K, tant pour la vérification de modèles que pour le test de satisfiabilité. L'analyse de ces algorithmes révèle des différences fondamentales avec leurs équivalents propositionnels.

4.1 Algorithmes propositionnels vs modaux : analyse comparative

La logique modale introduit une complexité supplémentaire qui se reflète dans la structure même des algorithmes. Le tableau 1 présente une comparaison systématique des stratégies et des règles entre les trois logiques implémentées.

Aspect	Logique propositionnelle (Satisfiabilité)	Logique modale K (Satisfiabilité)	Logique modale K (Vérification de modèles)
Structure algorithmique	CPLStrategy ou CPL_Strategy_With_Cut : boucle unique sur toutes les règles propositionnelles	KStrategy : séquence hiérarchisée CPLStrategy \rightarrow Pos \rightarrow Nec	Model_Checking_Strategy : séquence linéaire ExampleOfModelAndFormula \rightarrow Top_Down_Strategy \rightarrow Bottom_Up_Strategy
Ordre d'application	Priorité aux contradictions (Stop), puis simplification (NotNot), enfin connecteurs	Propositionnel d'abord, modal ensuite. Les négations de modalités (NotNec, NotPos) dans CPLStrategy	Deux phases séparées : décomposition descendante, puis reconstruction ascendante
Règles spécifiques aux modalités	Aucune	<ul style="list-style-type: none"> – Pos : création d'un monde – Nec : propagation à travers R – NotNec/NotPos : élimination des négations 	<ul style="list-style-type: none"> – *_Top_Down : décomposition avec propagation – *_True_Bottom_Up : reconstruction avec quantification
Gestion des mondes	Un seul ensemble de formules	Création dynamique de mondes et gestion des relations R	Parcours d'un modèle fixe prédéfini
Terminaison	Garantie (nombre fini de sous-formules)	Contrôlée par stratégie (évite création infinie)	Garantie (modèle fini)
Exemple de stratégie détaillée	<pre> 1 CPLStrategy: 2 repeat 3 firstRule 4 Stop 5 NotNot 6 And 7 NotOr 8 NotImp 9 NotAnd 10 NotEquiv 11 Imp 12 Equiv 13 Or 14 end 15 end </pre>	<pre> 1 KStrategy: 2 repeat 3 CPLStrategy 4 Pos 5 Nec 6 end </pre>	<pre> 1 Model_Checking_Strategy: 2 ExampleOfModelAndFormula 3 Top_Down_Strategy 4 Bottom_Up_Strategy </pre>

TABLE 1 – Comparaison des algorithmes dans LoTREC

4.2 Différences fondamentales entre tableau modal et tableau propositionnel

L'introduction des modalités \Box et \Diamond transforme radicalement la méthode des tableaux. Le tableau 2 synthétise ces différences essentielles.

Tableau propositionnel	Tableau modal (K)
Structure de données	
Un unique ensemble de formules (la branche) traitée comme une liste plate.	Un graphe de mondes (nœuds), chacun avec son propre ensemble de formules, reliés par des arêtes R .
Règles d'expansion	
Règles purement locales : elles n'affectent que l'ensemble courant. Exemple : $A \wedge B$ génère A et B dans le même monde.	Règles non-locales : – $\Diamond A$: Crée un nouveau monde et une relation R – $\Box A$: Propage A à travers toutes les relations R existantes
Sémantique implémentée	
Connecteurs de vérité de fonctions : la valeur d'une formule ne dépend que de ses composants.	Quantifications sur des mondes accessibles : – \Box : quantification universelle ($\forall u, wRu \Rightarrow u \models A$) – \Diamond : quantification existentielle ($\exists u, wRu \wedge u \models A$)
Gestion de l'accessibilité	
Absente. Pas de notion de relation entre formules.	Centrale. Les conditions <code>isLinked</code> et actions <code>link</code> sont omniprésentes.
Construction du modèle	
Une branche ouverte fournit une valuation (affectation de vérité aux atomes).	Une branche ouverte fournit un modèle de Kripke complet (ensemble W , relation R , interprétation I).
Problème de terminaison	
Terminaison garantie et simple : nombre fini de sous-formules à générer.	Risque de non-terminaison : création potentiellement infinie de mondes (ex : $\Diamond\Box\Diamond\Box\dots p$). Nécessite des techniques de contrôle.
Stratégies de contrôle	
Relativement simples, visent l'efficacité (ex : <code>CPL_Strategy_With_Cut</code> avec règles de coupe).	Complexes et cruciales : hiérarchisées (<code>KStrategy</code>) pour garantir la terminaison et l'efficacité.

TABLE 2 – Différences fondamentales entre les méthodes de tableau

4.3 Règles spécifiques aux modalités : exemples contrastés

4.3.1 Règles de satisfiabilité modale vs propositionnelle

- **En propositionnel** : La règle pour $A \wedge B$ est purement locale : `And` ajoute simplement A et B à l'ensemble courant.
- **En modal K** : Les règles pour les modalités doivent gérer la relation d'accessibilité :

```

1 # Règle Pos (pour A) - Création de monde
2 Conditions: hasElement w pos variable A
3 Actions: createNewNode u ; link w u R ; add u variable A
4
5 # Règle Nec (pour A) - Propagation trans-mondes
6 Conditions: hasElement w nec variable A ; isLinked w u R
7 Actions: add u variable A

```

Ces règles illustrent la dualité création/propagation qui n'existe pas en logique propositionnelle.

4.3.2 Règles de vérification de modèles : double phase

Pour la vérification de modèles, chaque modalité nécessite deux règles distinctes :

```

1 # Phase descendante : décomposition
2 Nec_Top_Down:
3 Conditions: hasElement w isItTrue nec variable A ; isLinked w u R
4 Actions: add u isItTrue variable A
5
6 # Phase montante : reconstruction
7 Nec_True_Bottom_Up:
8 Conditions: hasElement w isItTrue nec variable A
9             isMarkedExpressionInAllChildren w isItTrue variable A R Yes
10 Actions: markExpressions w isItTrue nec variable A Yes

```

Cette double règle traduit opérationnellement la quantification universelle : décomposition vers tous les successeurs, puis reconstruction si tous les successeurs satisfont la formule.

4.4 Intégration du connecteur d'implication : une preuve de modularité

L'ajout de l'implication \rightarrow dans *Model-Checking-Monomodal* a démontré l'architecture modulaire de LoTREC. Cette extension a suivi un processus en trois étapes qui sépare clairement les préoccupations :

1. **Définition syntaxique** (Onglet *Connectors*) : Spécification formelle du connecteur (nom, arité, affichage).
2. **Définition sémantique** (Onglet *Rules*) : Ajout de règles basées sur l'équivalence $A \rightarrow B \equiv \neg A \vee B$, avec des règles distinctes pour chaque phase :
 - Imp_Top_Down pour la décomposition
 - Imp_*)_Bottom_Up pour la reconstruction
3. **Intégration stratégique** (Onglet *Strategies*) : Insertion des nouvelles règles aux positions appropriées dans les stratégies existantes.

Contraste avec une approche procédurale : Dans un algorithme procédural classique, l'ajout d'un nouveau connecteur nécessiterait de modifier le code source de l'algorithme d'évaluation. Avec l'architecture à base de règles de LoTREC, il suffit de déclarer la sémantique du connecteur sous forme de règles, qui sont ensuite interprétées par l'engine existant.

4.5 Conclusion synthétique

LoTREC implémente de manière pédagogique et efficace les fondements du raisonnement automatique pour la logique modale K. Les expériences menées illustrent que le passage des logiques propositionnelles aux logiques modales ne réside pas seulement dans une expressivité accrue, mais surtout dans la nécessité d'algorithmes manipulant explicitement des **structures de graphes (mondes et relations)** et des **quantifications sur ces structures**.

L'analyse des stratégies révèle des choix algorithmiques sophistiqués (comme l'ordre CPLStrategy \rightarrow Pos \rightarrow Nec) qui sont essentiels pour garantir la terminaison et l'efficacité. L'extension réussie avec le connecteur d'implication démontre la modularité et l'extensibilité de l'approche à base de règles adoptée par LoTREC.

Enfin, la comparaison systématique entre les algorithmes propositionnels et modaux met en lumière les défis spécifiques du raisonnement modal : gestion dynamique des mondes, propagation trans-mondes des contraintes, et contrôle de la création potentiellement infinie de nouvelles structures. Ces défis expliquent pourquoi les algorithmes modaux sont à la fois plus complexes et plus riches que leurs équivalents propositionnels.