TME 7

KADHI Youssef 3680916 et UNG Richard 3680881

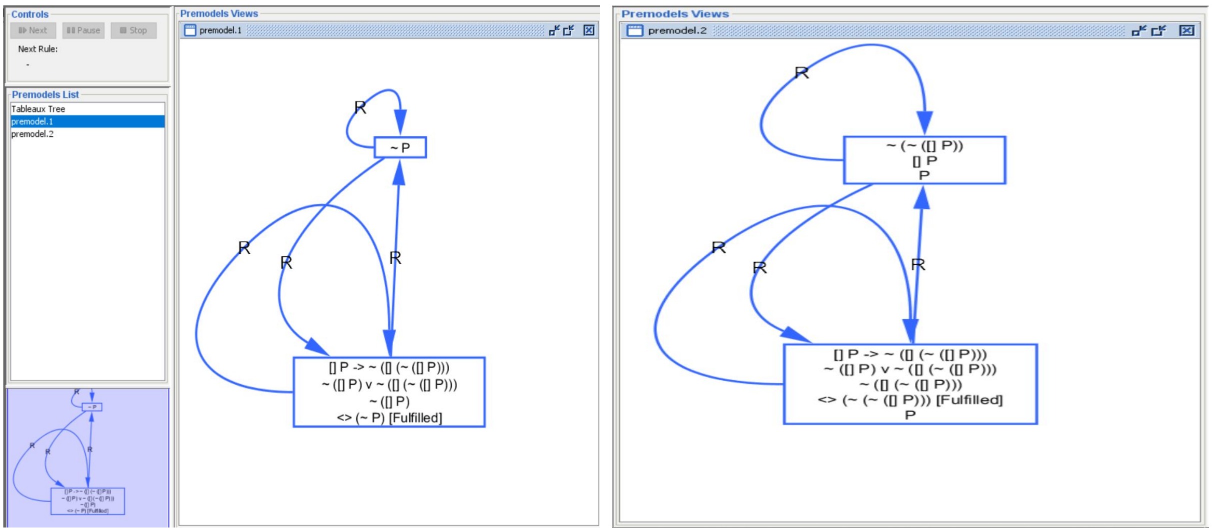
Exercice 1

1. Vérifions si les formules suivantes sont satisfiables, insatisfiables, ou valides dans S5.

    - $Kp \rightarrow \neg K \neg Kp$
    - $Kp \wedge KK\neg p$
    - $Kp \rightarrow KK\neg p$

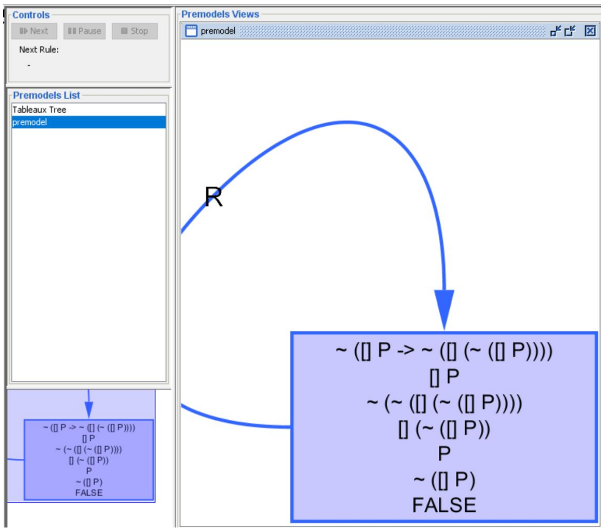
- La formule $Kp \rightarrow \neg K \neg Kp$ se réécrit sous forme préfixe : imp nec P not nec not nec P.

Avec l'aide de LoTREC, on peut voir que cette formule nous donne une arborescence avec deux feuilles ouvertes. Donc on peut en déduire qu'elle est satisfiable.



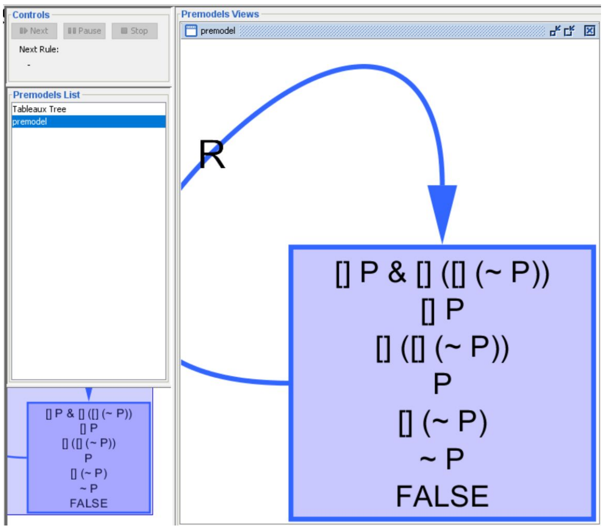
Vérifions maintenant si $Kp \rightarrow \neg K \neg Kp$ est valide. Pour cela, on prend la négation de la formule, et on obtient sous forme préfixe : not imp nec P not nec not nec P.

Avec l'aide de LoTREC, on peut voir que la négation nous donne une arborescence avec une unique feuille fermée. La négation de la formule $Kp \rightarrow \neg K \neg Kp$ est insatisfiable. Donc on peut en déduire que $Kp \rightarrow \neg K \neg Kp$ est valide.



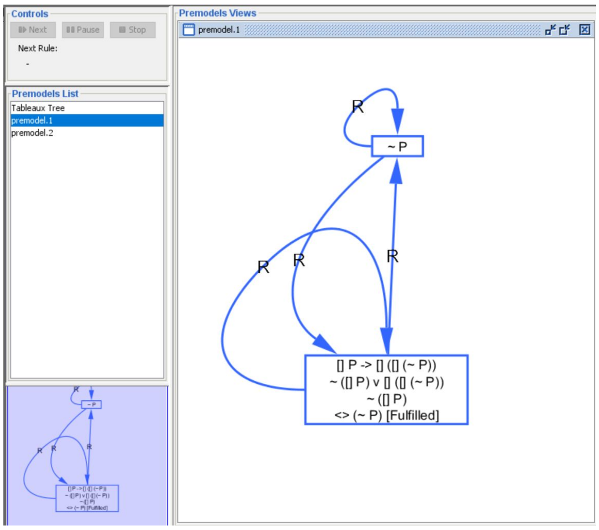
- La formule Kp ∧ KK¬p se réécrit sous forme préfixe : and nec P nec nec not P.

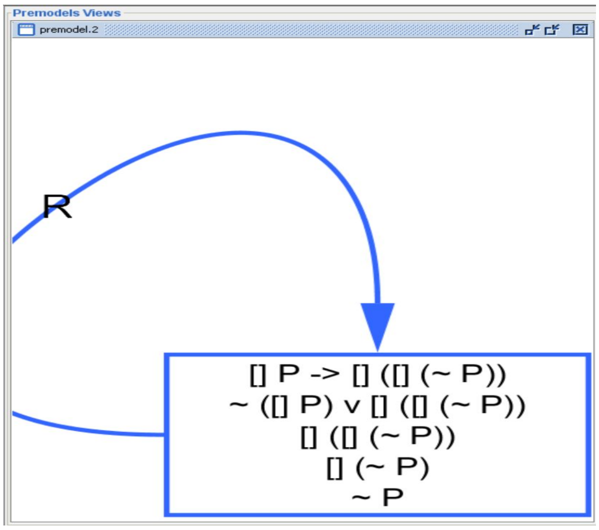
Avec l'aide de LoTREC, on peut voir que cette formule nous donne une arborescence avec une unique feuille fermée. Donc on peut en déduire qu'elle est insatisfiable.



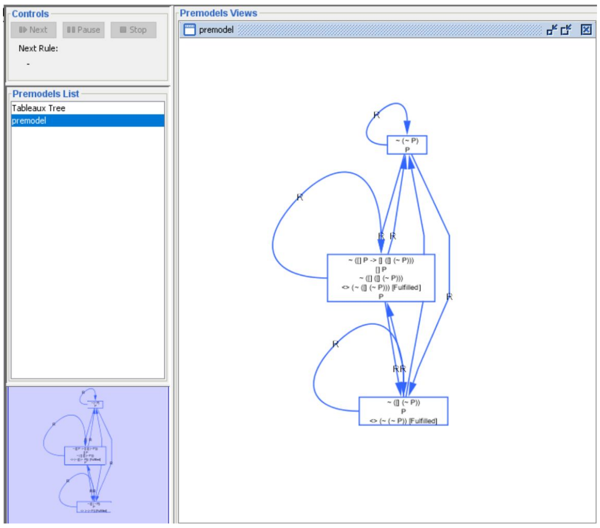
- La formule Kp → KK¬p se réécrit sous forme préfixe : imp nec P nec nec not P.

Avec l'aide de LoTREC, on peut voir que cette formule nous donne une arborescence avec deux feuilles ouvertes. Donc on peut en déduire qu'elle est satisfiable.





Vérifions maintenant si Kp → KK¬p est valide. Pour cela, on prend la négation de la formule, et on obtient sous forme préfixe : not imp nec P nec nec not P.
Avec l'aide de LoTREC, on peut voir que la négation nous donne une arborescence avec une unique feuille ouverte. La négation de la formule Kp → ¬KKp est satisfiable. Donc on peut en déduire que Kp → ¬KKp est non valide.



2. Examinons les règles de la stratégie de la logique de type S5 une à une par rapport à la stratégie du TME 6 (Monomodal-K).

Règles comprises dans les deux logiques :

- Stop : cette règle vérifie si une feuille possède à la fois une variable propositionnelle et sa négation. Si c'est le cas, cette règle ferme la feuille.
- NotNot: Cette règle indique que la double négation d'une proposition est logiquement équivalente à la proposition.

- And: Cette règle vérifie l’opérateur AND, c’est une formule de type **α**. Les deux termes qu’elle a manipulés sont des conséquences logiques de cette formule. Par conséquent, on les ajoute toutes deux sur la branche.
- NotOr: Cette règle vérifie l’opérateur NOR, c’est une formule de type **α**. Les deux termes qu’elle a manipulés sont des conséquences logiques de cette formule. Par conséquent, on les ajoute toutes deux sur la branche.
- NotImp: Cette règle indique la négation de l’implication, c’est une formule de type **α**. Les deux termes qu’elle a manipulés sont des conséquences logiques de cette formule. Par conséquent, on les ajoute toutes deux sur la branche.
- NotAnd: Cette règle vérifie l’opérateur NAND, c’est une formule de type **β**. Si cette formule est vraie alors au moins l’un des deux termes est faux. Par conséquent, on crée deux nouvelles branches et on ajoute un des termes sur une des branches et l’autre terme sur l'autre branche.
- NotEquiv: Cette règle indique la négation de l’équivalence, c’est une formule de type **β**. Si cette formule est vraie alors au moins l’une des deux implications est fausse. Par conséquent, on crée deux nouvelles branches et on ajoute une implication sur une branche et l’autre implication sur la seconde branche.
- Imp: Cette règle vérifie l’implication, c’est une formule de type **β**. Si cette formule est vraie alors soit l’opérande de gauche est faux, soit l’opérande de droite vrai. Par conséquent, on crée deux nouvelles branches et on ajoute un des opérandes sur une des branches et l’autre opérande sur l'autre branche.
- Equiv: Cette règle vérifie l’équivalence, c’est une formule de type **α**. Les deux implications qu’elle a manipulés sont des conséquences logiques de cette formule. Par conséquent, on les ajoute toutes deux sur la branche.
- Or: Cette règle vérifie l’opérateur OR, c’est une formule de type **β**. Si cette formule est vraie alors au moins l’un des deux termes est vrai. Par conséquent, on crée deux nouvelles branches et on ajoute un des termes sur une des branches et l’autre terme sur l'autre branche.
- NotNec: Cette règle indique la négation de la nécessité d’une formule. Cela revient à la possibilité de la négation de cette formule.
- NotPos: Cette règle indique la négation de la possibilité d’une formule. Cela revient à la nécessité de la négation de cette formule.
- Pos: Cette règle vérifie la possibilité d’une formule. Elle crée un nouveau monde et le relie avec celui dans lequel elle se trouve actuellement par une relation R puis ajoute la formule sur le nouveau monde.
- Nec: Cette règle vérifie la nécessité d’une formule. Elle ajoute la formule à tous les mondes liés par une relation R depuis le monde sur lequel la règle est appliquée.

Règles comprises uniquement dans la logique S5:

- ReflexiveArcs: Cette règle définit la réflexivité de la relation R dans un monde.
- SymmetricArcs: Cette règle définit la symétrie de la relation R d’un monde à un autre. s'il existe une relation R d’un monde w à un monde u, alors il existe une relation R du monde u au monde w.
- TransitiveArcs: Cette règle définit la transitivité de la relation R. S'il existe une relation R d’un monde w à un monde u et une relation R d’un monde u à un monde v alors il existe une relation R du monde w au monde v.

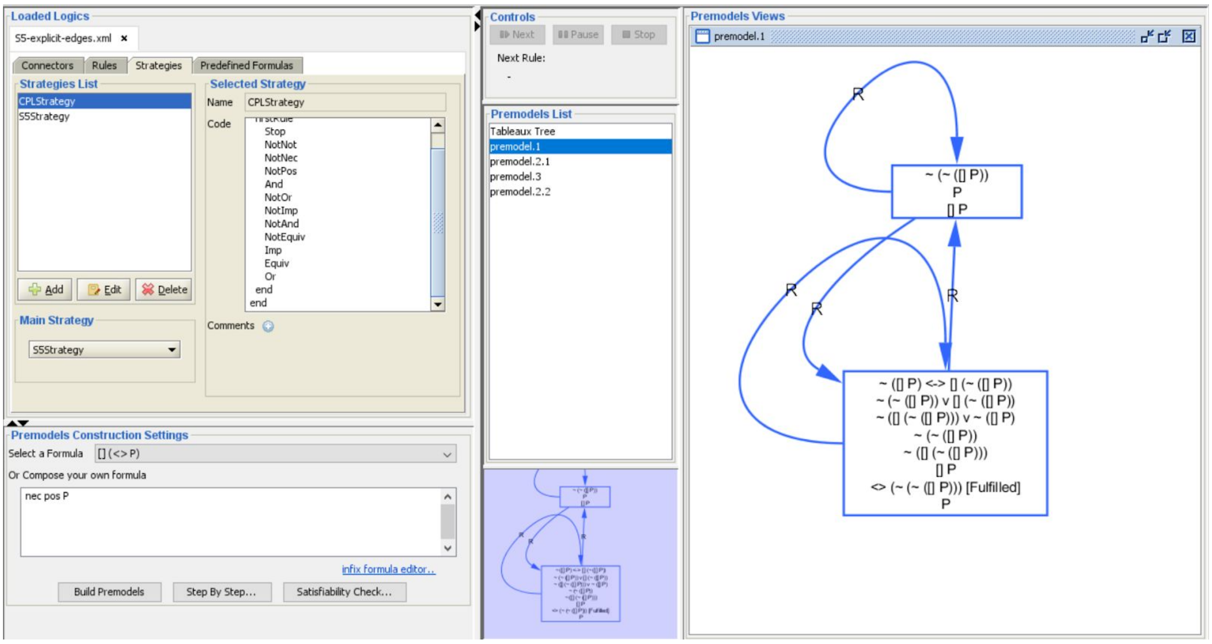
- MarkFulfilledPos: Cette règle définit la vérification d’une formule pour la possibilité. Elle cherche dans tous les mondes accessibles si une formule est vérifiée et lui ajoute Fulfilled.

On remarque que R est une relation d’équivalence dans la logique S5 mais pas dans celle monomodal K. La logique S5 et la logique K emploient la méthode des tableaux auxquels on ajoute les règles de nécessité et de possibilité. En plus de ça, la logique S5 fait en sorte d’appliquer la relation d’équivalence de R ainsi qu’une règle de vérification parmi les mondes accessibles grâce à R.

Exercice 2

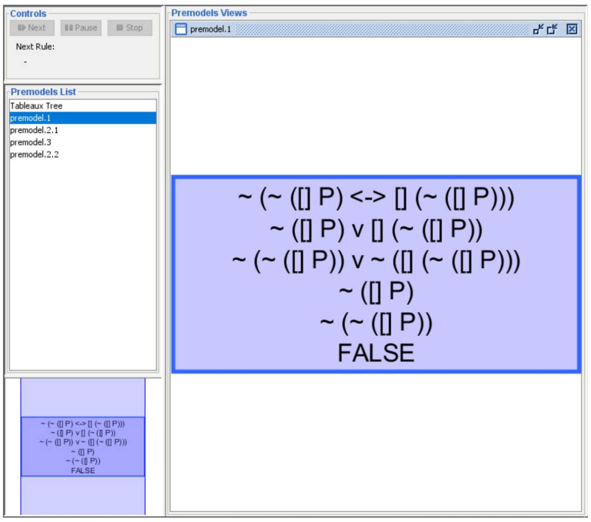
1. La formule ¬Kp ⇔ K¬Kp se réécrit sous forme préfixe : equiv not nec P nec not nec P.

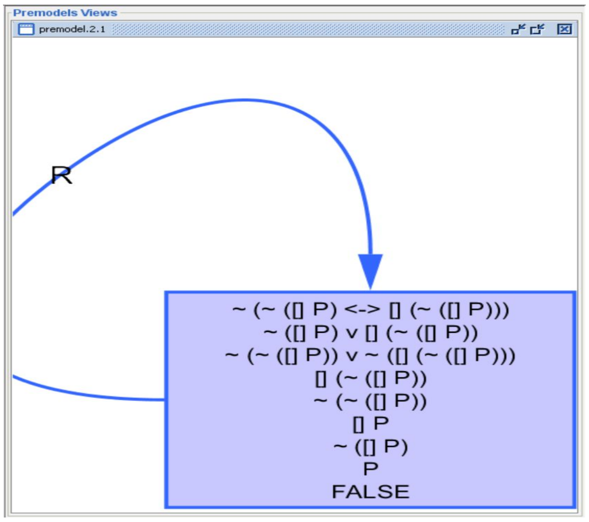
Dans le système S5, on peut voir que cette formule nous donne une arborescence avec quatre feuilles dont au moins une est ouverte. Donc on peut en déduire qu’elle est satisfiable. Mais ce n’est pas la question.

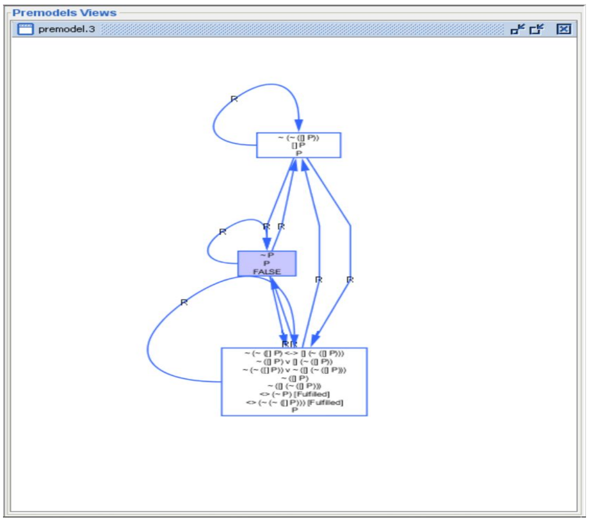


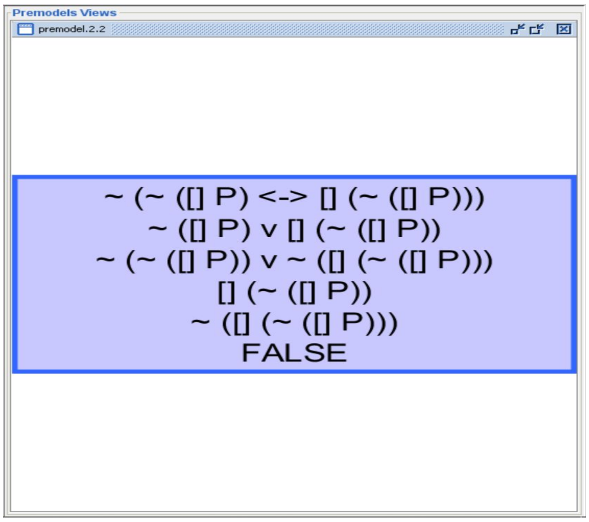
Vérifions si ¬Kp ⇔ K¬Kp est valide. Pour cela, on prend la négation de la formule, et on obtient sous forme préfixe : not equiv not nec P nec not nec P.

Avec l’aide de LoTREC, on peut voir que la négation nous donne une arborescence avec quatre feuilles fermées. La négation de la formule ¬Kp ⇔ K¬Kp est insatisfiable. Donc on peut en déduire que ¬Kp ⇔ K¬Kp est valide.





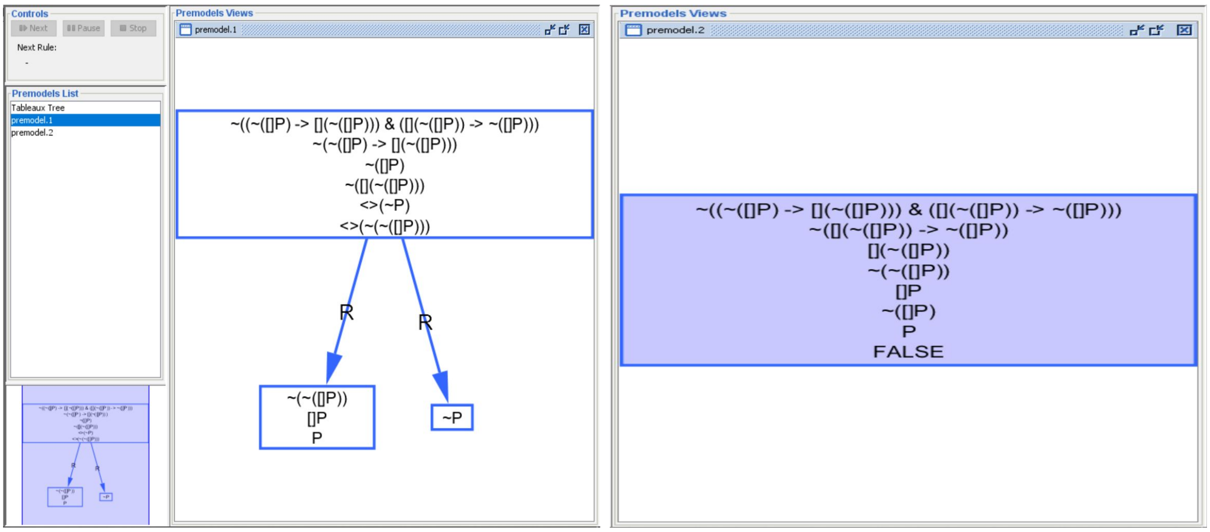




2. La formule ¬Kp ⇔ K¬Kp se réécrit sous forme préfixe sans équivalence: and imp not nec P nec not nec P imp nec not nec P not nec P.

Le système S4 ne reconnaît pas le connecteur equiv. Vérifions si ¬Kp ⇔ K¬Kp est valide dans S4. Pour cela, on prend la négation de la formule, et on obtient sous forme préfixe : and imp not nec P nec not nec P imp nec not nec P not nec P.

Avec l'aide de LoTREC, on peut voir que la négation nous donne une arborescence avec deux feuilles dont l'une est ouverte. La négation de la formule ¬Kp ⇔ K¬Kp est donc satisfiable. Donc on peut en déduire que ¬Kp ⇔ K¬Kp n'est pas valide dans le système S4.



Les deux systèmes possèdent des axiomes en commun. L'axiome K (connaissance), l'axiome T (réflexivité) et l'axiome 4 (transitivité). Le système S5 possède l'axiome 5 (euclidiannité) en plus des axiomes que possède le système S4.

Regardons la feuille ouverte ci-dessus. On peut voir un monde (w_1) qui possède 2 relations R, l'une vers un monde (w_2) qui possède Kp et p, et l'autre vers un monde (w_3) qui possède ¬p. Si on supposait que l'on possède l'axiome 5 dans le système S4, on aurait eu par euclidiannité une relation R entre les mondes w_2 et w_3. À ce moment, Kp aurait utilisé cette relation entre les mondes w_2 et w_3 pour ajouter p au monde w_3 et là on aurait eu notre feuille fermée nous permettant de dire que cette formule est insatisfiable et donc que ¬Kp ⇔ K¬Kp est valide.

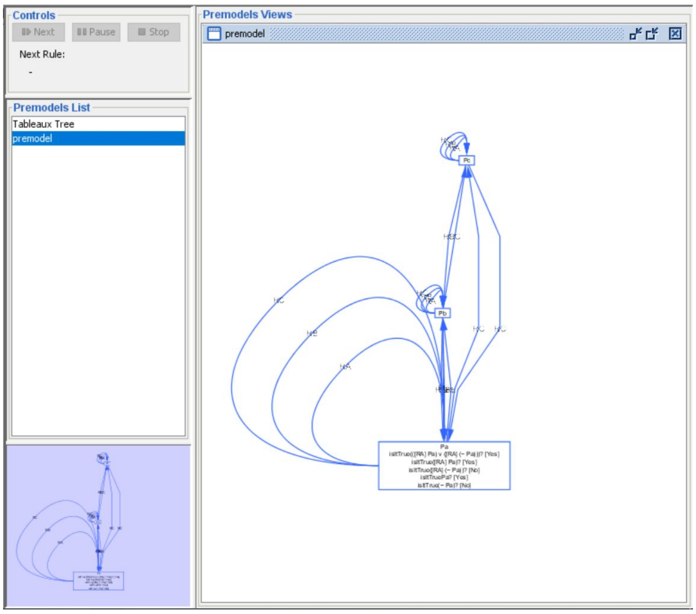
Sans l'axiome 5, ¬Kp → K¬Kp n'est pas toujours vrai donc l'axiome 5 est nécessaire pour prouver cette équivalence.

Exercice 3

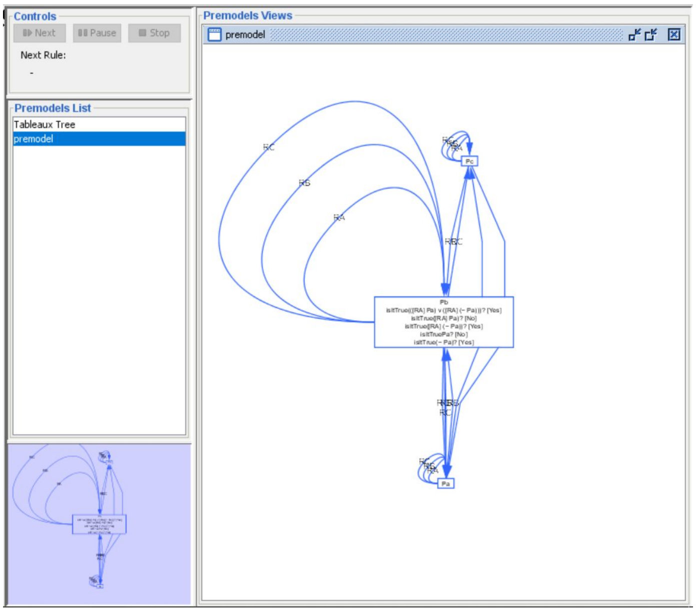
1. Pour cela on considère le modèle M donné par la spécification suivante :

- $W = \{w_1, w_2, w_3\}$
- $RA = \{(w_1, w_1), (w_2, w_2), (w_3, w_3)\}$
- $RB = \{(w_1, w_1), (w_2, w_2), (w_3, w_3), (w_1, w_2), (w_2, w_1)\}$
- $RC = \{(w_1, w_1), (w_2, w_2), (w_3, w_3), (w_1, w_2), (w_2, w_1), (w_1, w_3), (w_3, w_1), (w_2, w_3), (w_3, w_2)\}$
- π est défini par π(Pa) = $\{w_1\}$, π(Pb) = $\{w_2\}$ et π(Pc) = $\{w_3\}$.

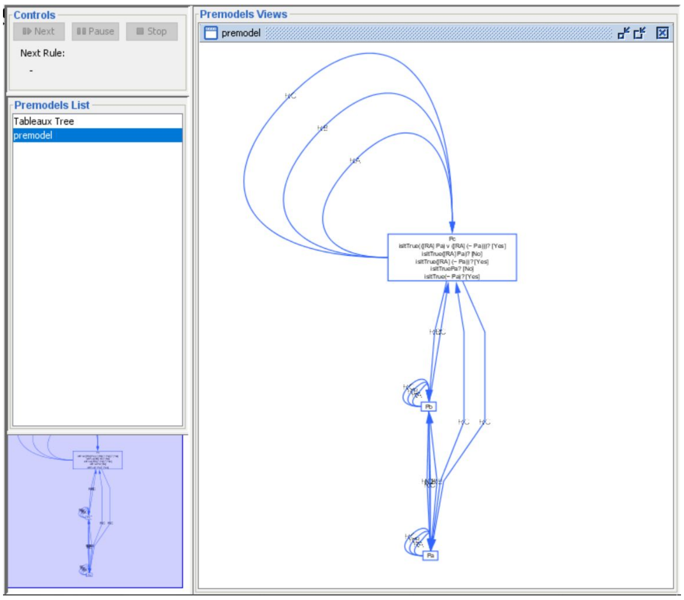
2. Vérifions les formules de l'exercice.
   a. Que faut-il vérifier pour montrer que A peut toujours savoir si elle a le papillon sur la tête ?
      i. $w_1 \models (K_{RA\ Pa}) \vee (K_{RA \neg Pa})$ donne: add w_1 isItTrue or nec RA Pa nec RA not Pa.



ii. w_2 |= $(K_{RA\ Pa}) \vee (K_{RA \neg Pa})$ donne: add w_2 isItTrue or nec RA Pa nec RA not Pa.



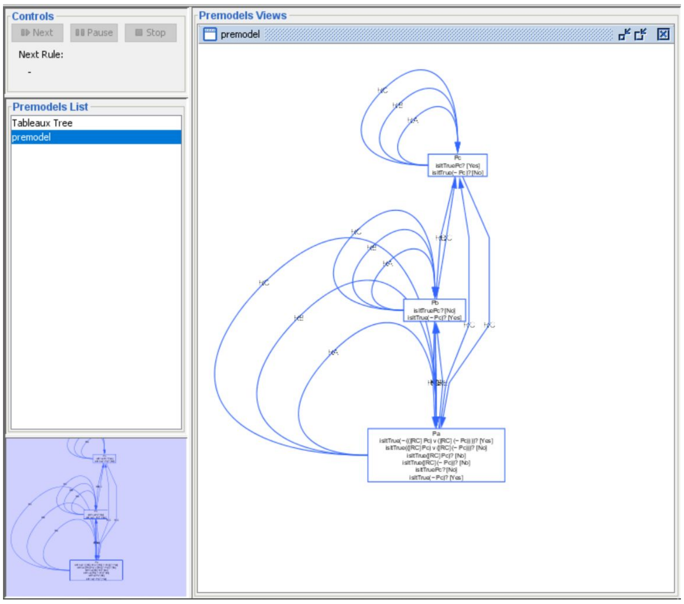
iii. w_3 |= $(K_{RA\ Pa}) \vee (K_{RA \neg Pa})$ donne: add w_3 isItTrue or nec RA Pa nec RA not Pa.



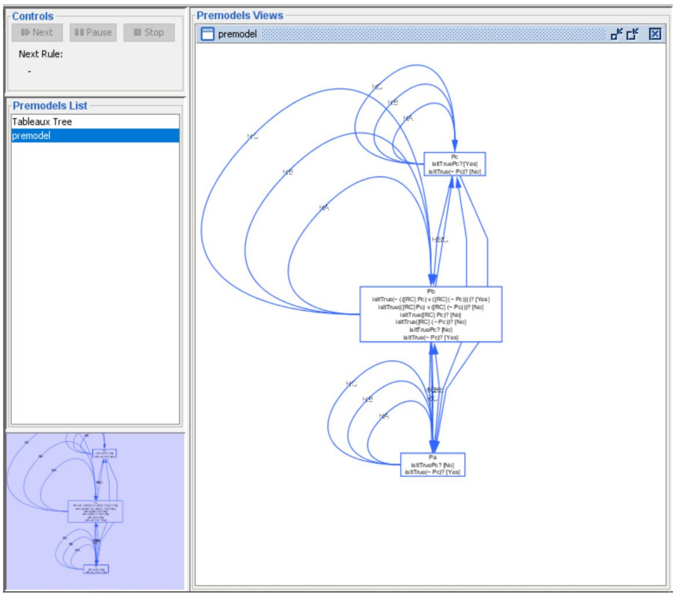
On trouve que M |= ($K_{RA\ Pa}$) V ($K_{RA\neg Pa}$) est vrai, quelque soit le monde appartenant à M . Donc A peut toujours savoir si elle a le papillon sur la tête.

b. Même question pour montrer que C ne peut jamais savoir si elle a le papillon sur la tête ?

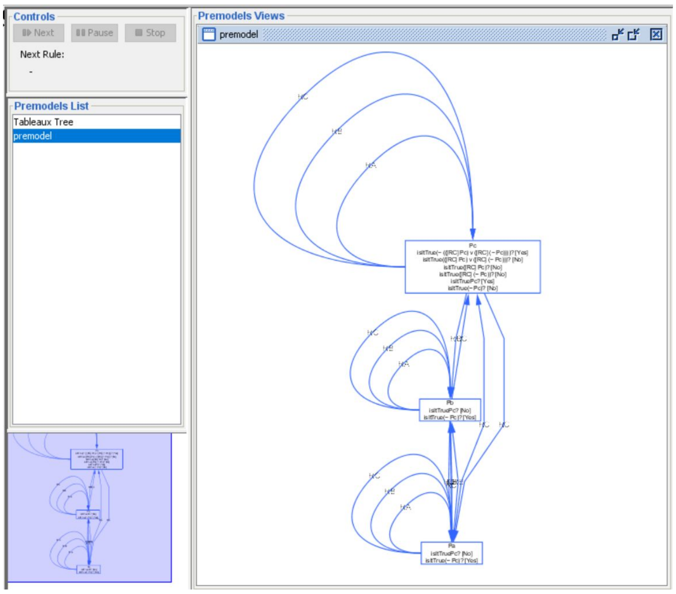
i. w_1 |= ¬(($K_{RC\ Pc}$) V ($K_{RC\neg Pc}$)) donne: add w_1 isItTrue not or nec RC Pc nec RC not Pc.



ii. w_2 |= ¬(($K_{RC\ Pc}$) V ($K_{RC\neg Pc}$)) donne: add w_2 isItTrue not or nec RC Pc nec RC not Pc.



iii. $w_3 \models \neg((K_{RC\ Pc}) \vee (K_{RC \neg Pc}))$ donne: add w_3 isItTrue not or nec RC Pc nec RC not Pc.

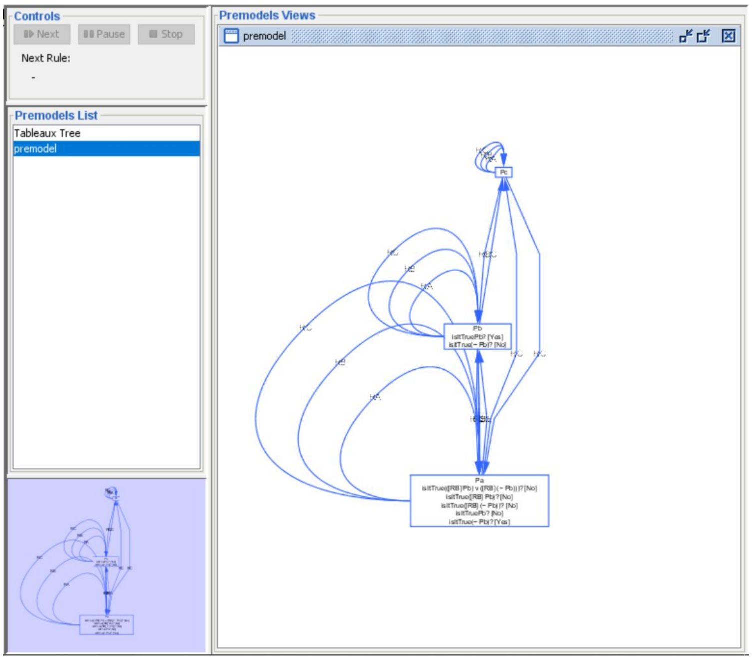


On trouve que $M \models \neg((K_{RC\ Pc}) \vee (K_{RC\neg Pc}))$ est vrai quelque soit le monde appartenant à M. Donc C ne peut jamais savoir si elle a le papillon sur la tête .

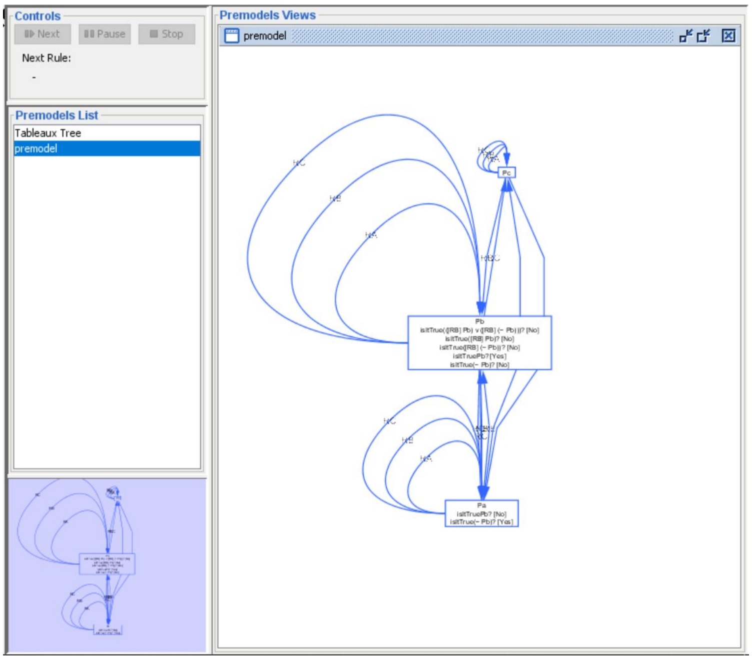
3. Vérifions maintenant quelques formules.

   a. Montrer que B peut savoir si elle a un papillon sur la tête.

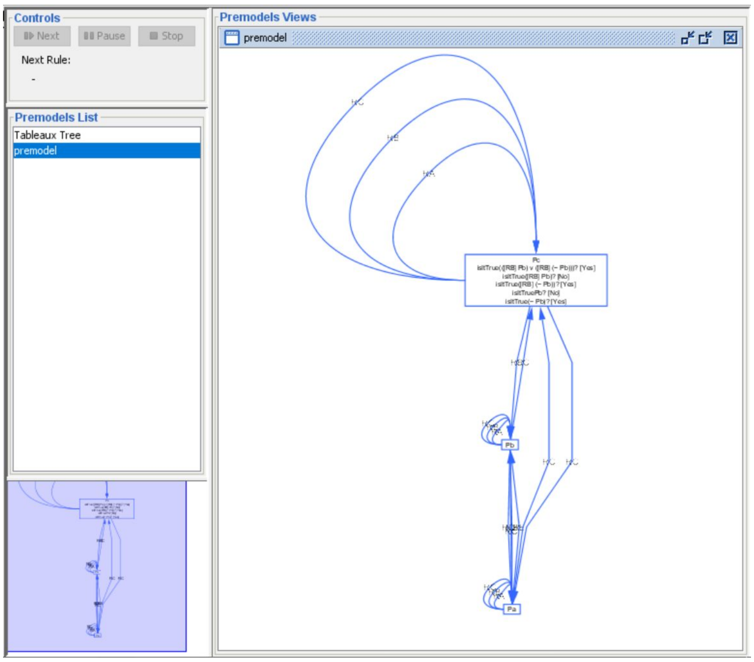
      i. $w_1 \models (K_{RB\,Pb}) \vee (K_{RB\neg Pb})$ donne: add w_1 isItTrue or nec RB Pb nec RB not Pb.



      ii. $w_2 \models (K_{RB\,Pb}) \vee (K_{RB\neg Pb})$ donne: add w_2 isItTrue or nec RB Pb nec RB not Pb.



iii. $w_3 \models (K_{RB\ Pb}) \vee (K_{RB\neg Pb})$ donne: add w_3 isItTrue or nec RB Pb nec RB not Pb.

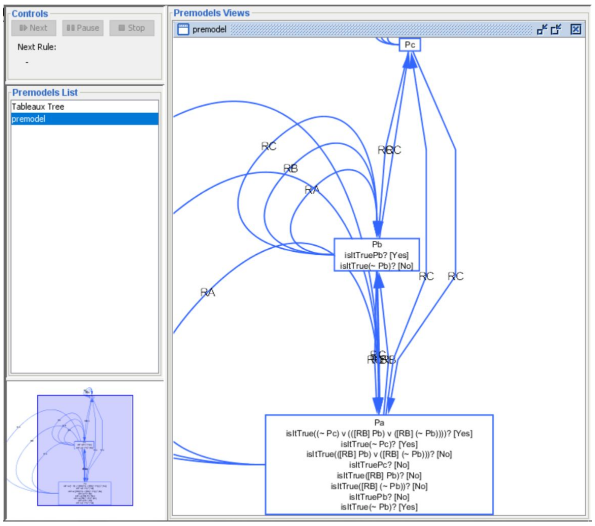


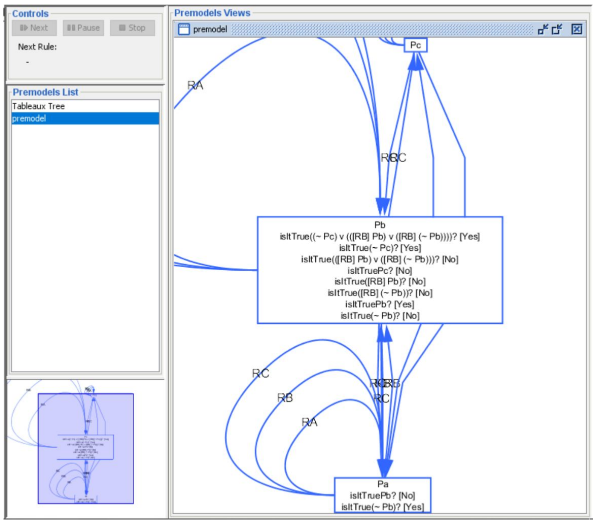
On trouve que $M \models \neg((K_{RC\ Pc}) \vee (K_{RC\neg Pc}))$ est vrai uniquement pour le monde w_3.
Pour cette question, il suffit que la formule marche pour l'un des trois mondes pour pouvoir dire que B peut savoir si elle a un papillon sur la tête.

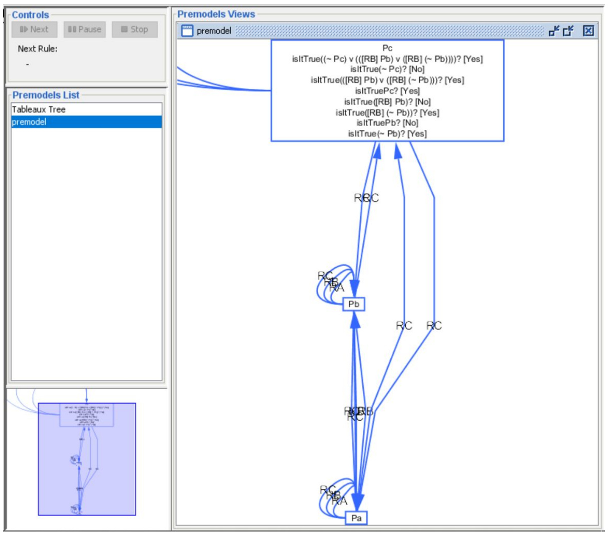
b. Observer le résultat : Pouvez-vous en déduire que si C a le papillon sur la tête, alors B sait si elle a le papillon sur la tête ?

Oui, on le peut. Comme on peut le voir, le seul cas parmi les 3 photos précédentes qui est vrai est le cas où le papillon est sur la tête (dans le monde w3). D'après la consigne, la personne B peut voir ceux qui a sur la tête de la personne C. Donc il est normal de dire que si C a le papillon sur la tête, alors B sait si elle a le papillon sur la tête. On peut le montrer aussi à l'aide de LoTREC.

Cette question se traduit sous la forme $M \models Pc \rightarrow (K_{RB\ Pb}) \vee (K_{RB\neg Pb})$ ce qui donne : add w_i isItTrue or not Pc or nec RB Pb nec RB not Pb.
On observe que le résultat est vrai quelque soit le monde appartenant à M. Donc si C a le papillon sur la tête, alors B sait si elle a le papillon sur la tête



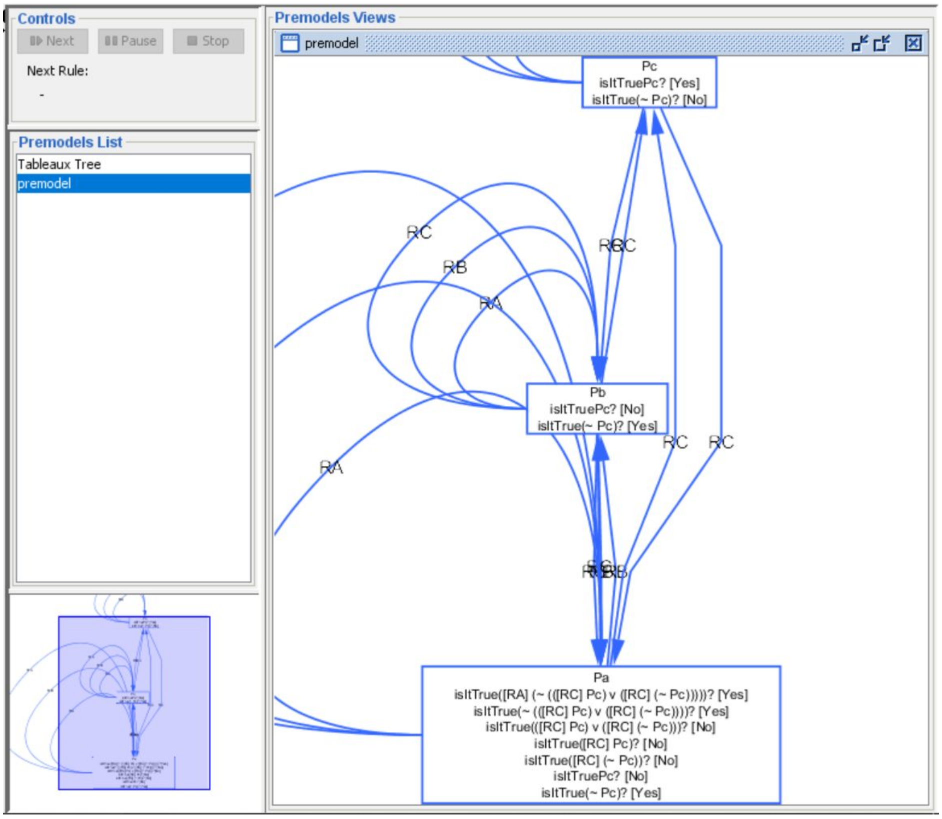




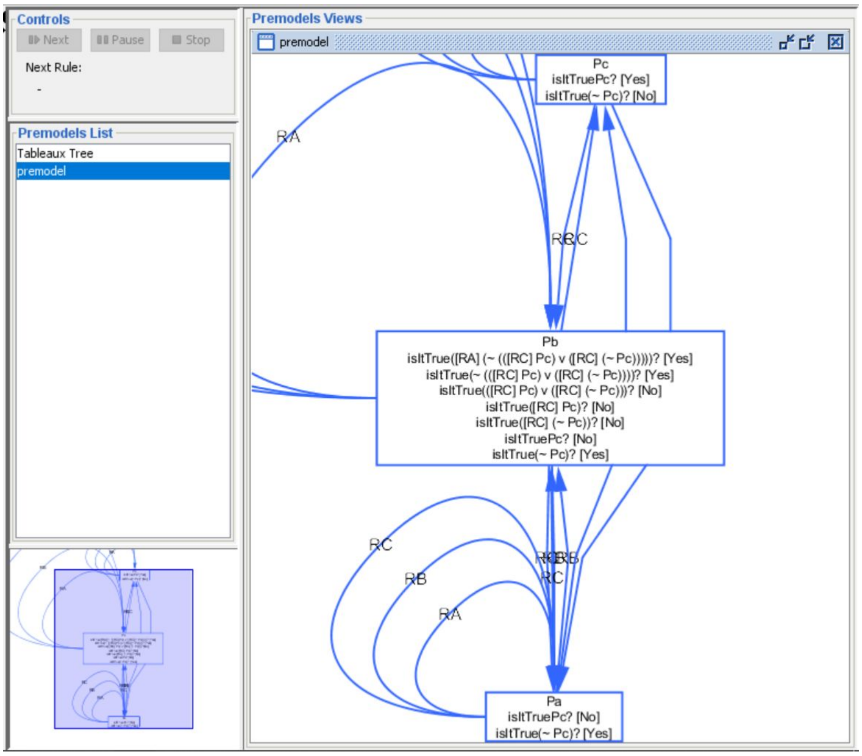
4. . Est-il vrai que A sait que C ne sait pas si elle a un papillon sur la tête ?

Cela revient à montrer que $M \models K_{RA}\neg((K_{RC\ Pc}) \vee (K_{RC\neg Pc}))$ soit vraie.

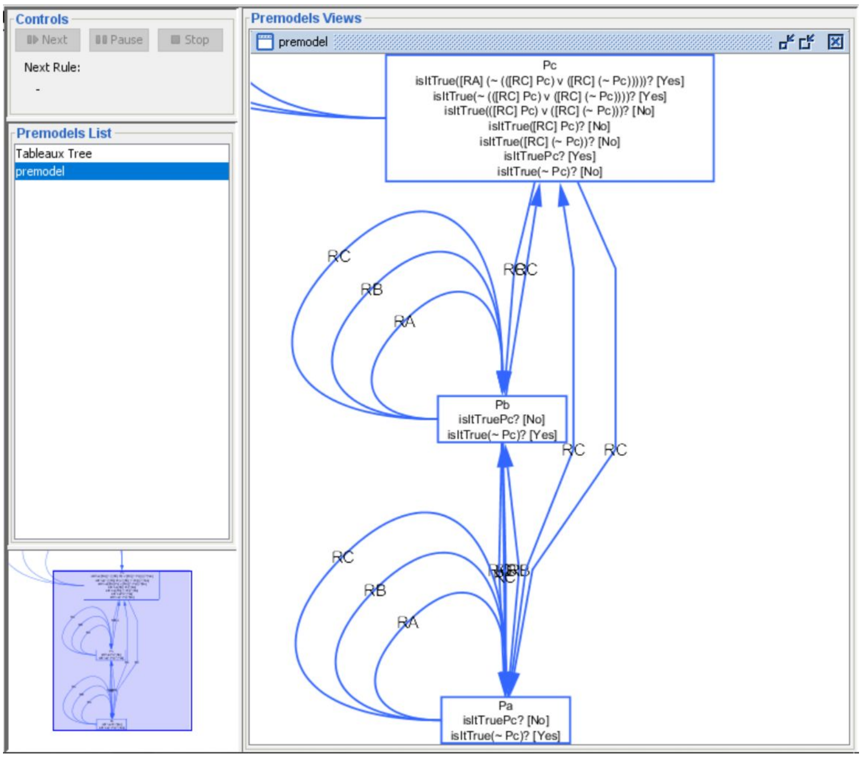
a. $w_1 \models K_{RA}\neg((K_{RC\ Pc}) \vee (K_{RC\neg Pc}))$ donne: add w_1 isItTrue nec RA not or nec RC Pc nec RC not Pc.



b. w_2 |= $K_{RA}\neg((K_{RC\ Pc}) \vee (K_{RC\neg Pc}))$ donne: add w_2 isItTrue nec RA not or nec RC Pc nec RC not Pc.



c. w_3 |= $K_{RA}\neg((K_{RC\ Pc}) \vee (K_{RC\neg Pc}))$ donne: add w_3 isItTrue nec RA not or nec RC Pc nec RC not Pc.

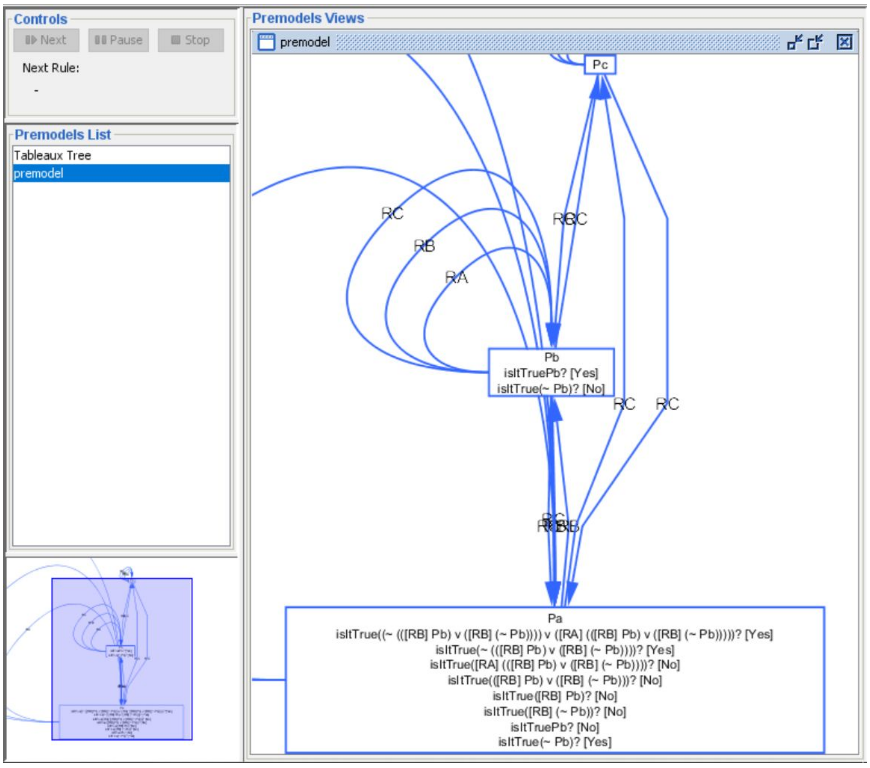


On trouve que M |= $K_{RA}\neg((K_{RC\ Pc}) \vee (K_{RC\neg Pc}))$ est vraie quelque soit le monde appartenant à M. Donc A sait que C ne sait pas si elle a un papillon sur la tête.

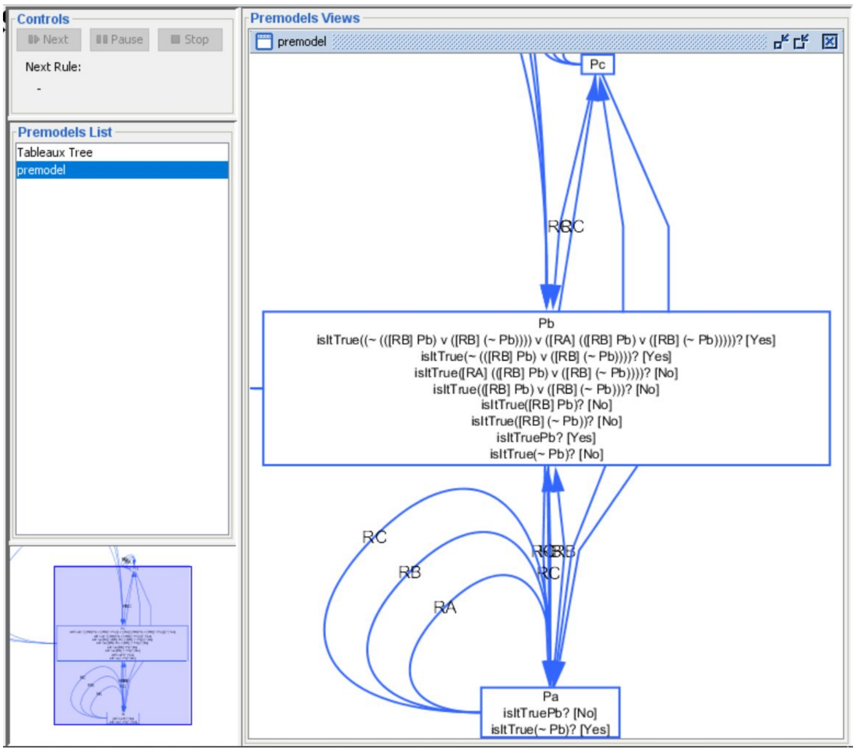
5. Est-il vrai que quand B sait si elle a un papillon sur la tête, alors A sait qu'elle (B) le sait ?

Cela revient à montrer que M |= $((K_{RB\ Pb}) \vee (K_{RB\neg Pb})) \rightarrow K_{RA}((K_{RB\ Pb}) \vee (K_{RB\neg Pb}))$ soit vraie.

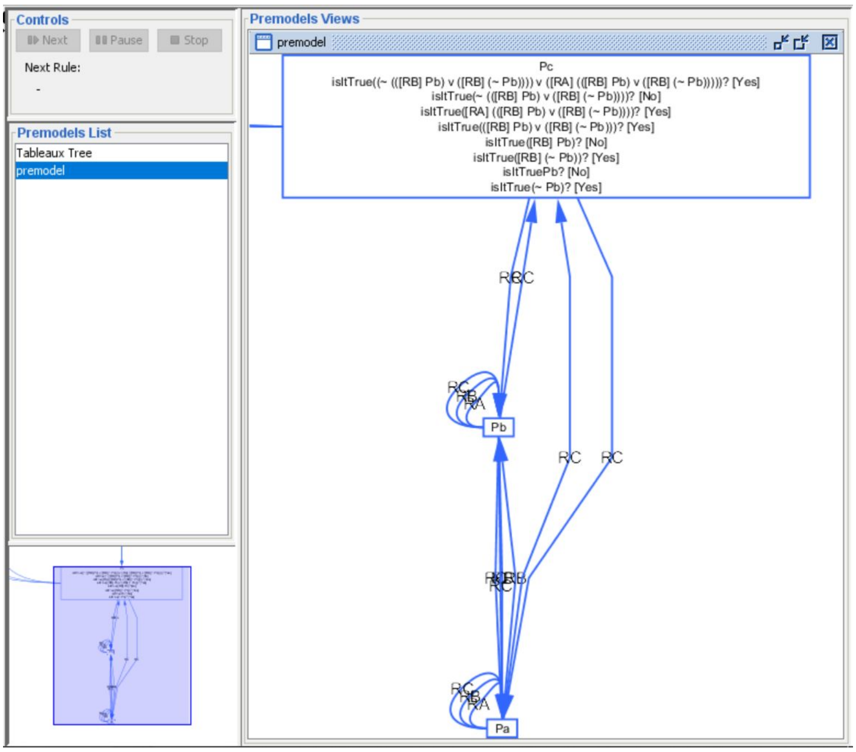
a. w_1 |= $((K_{RB\ Pb}) \vee (K_{RB\neg Pb})) \rightarrow K_{RA}((K_{RB\ Pb}) \vee (K_{RB\neg Pb}))$ donne: add w_1 isItTrue or not (or nec RB Pb nec RB not Pb) nec RA (or nec RB Pb nec RB not Pb).



b. $w_2 \models ((K_{RB\ Pb}) \vee (K_{RB\neg Pb})) \rightarrow K_{RA}((K_{RB\ Pb}) \vee (K_{RB\neg Pb}))$ donne: add w_1 isItTrue or not (or nec RB Pb nec RB not Pb) nec RA (or nec RB Pb nec RB not Pb).



c. w_3 |= $((K_{RB\ Pb}) \lor (K_{RB\neg Pb})) \rightarrow K_{RA}((K_{RB\ Pb}) \lor (K_{RB\neg Pb}))$ donne: add w_1 isItTrue or not (or nec RB Pb nec RB not Pb) nec RA (or nec RB Pb nec RB not Pb).



On trouve que M |= $((K_{RB\ Pb}) \vee (K_{RB\neg Pb})) \rightarrow K_{RA}((K_{RB\ Pb}) \vee (K_{RB\neg Pb}))$ est vraie quelque soit le monde appartenant à M . Donc vrai que quand B sait si elle a un papillon sur la tête, alors A sait qu'elle (B) le sait.

Exercice 4

1. Pour cela on considère le modèle M donné par la spécification suivante dans la logique S5:

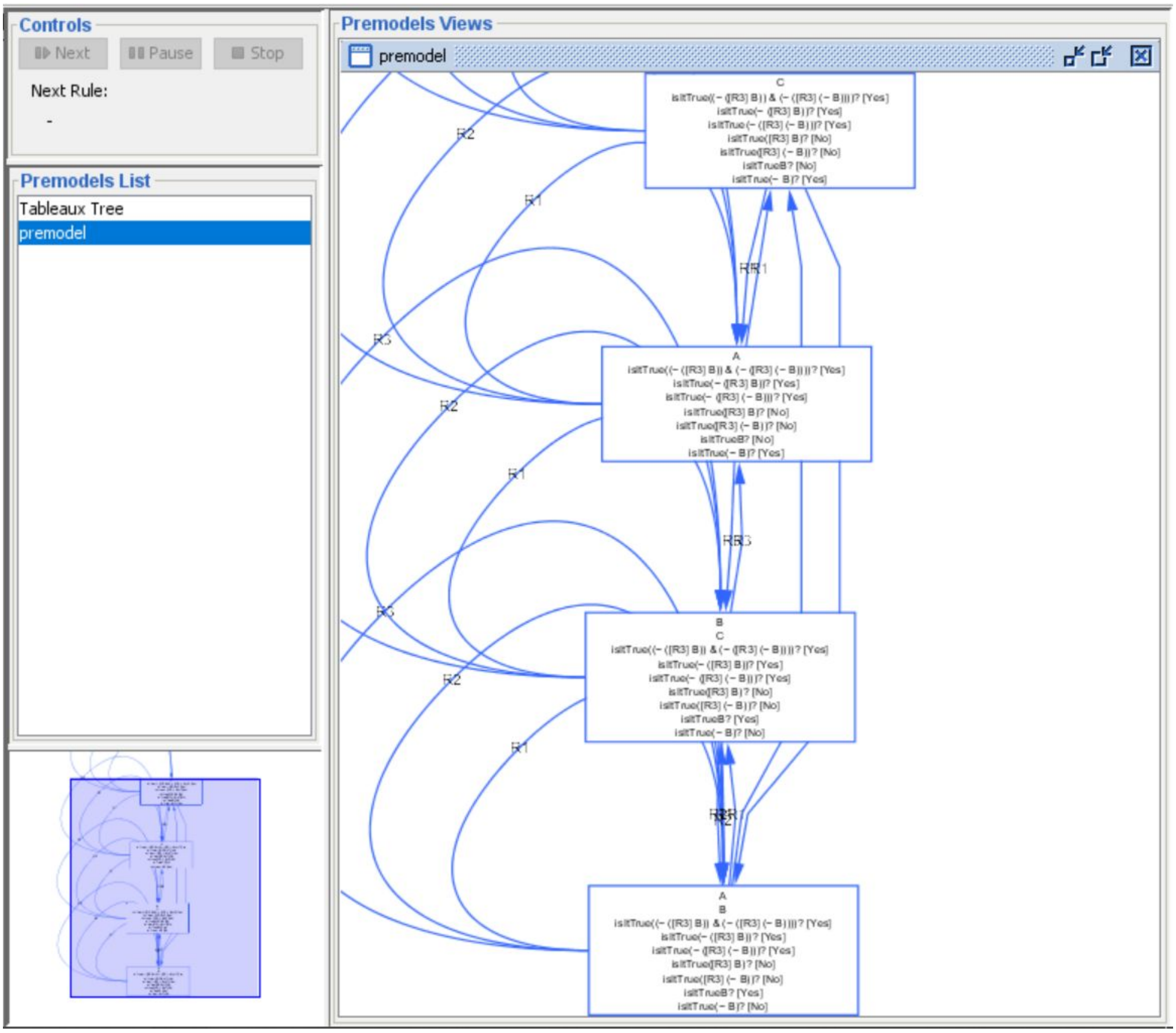
- $W = \{w_1, w_2, w_3, w_4\}$
- $R1 = \{(w_1, w_1), (w_2, w_2), (w_3, w_3), (w_4, w_4), (w_1, w_2), (w_2, w_1), (w_3, w_4), (w_4, w_3)\}$
- $R2 = \{(w_1, w_1), (w_2, w_2), (w_3, w_3), (w_4, w_4), (w_1, w_2), (w_2, w_1)\}$
- $R3 = \{(w_1, w_1), (w_2, w_2), (w_3, w_3), (w_4, w_4), (w_1, w_3), (w_3, w_1), (w_2, w_4), (w_4, w_2)\}$
- π est défini par π(a) = $\{w_2, w_3\}$, π(b) = $\{w_1, w_2\}$ et π(c) = $\{w_1, w_4\}$.

   a. Montrons que M |= $\neg K_3 b \wedge \neg K_3 \neg b$ soit valide. La formule $\neg K_3 b \wedge \neg K_3 \neg b$ donne sous forme préfixe : and not nec R3 B not nec R3 not B.

En rentrant :

- add w_1 isItTrue and not nec R3 B not nec R3 not B
- add w_2 isItTrue and not nec R3 B not nec R3 not B
- add w_3 isItTrue and not nec R3 B not nec R3 not B
- add w_4 isItTrue and not nec R3 B not nec R3 not B

On voit que le résultat est vrai dans tous les mondes de M . Donc que M |= $\neg K_3 b \wedge \neg K_3 \neg b$ est valide.

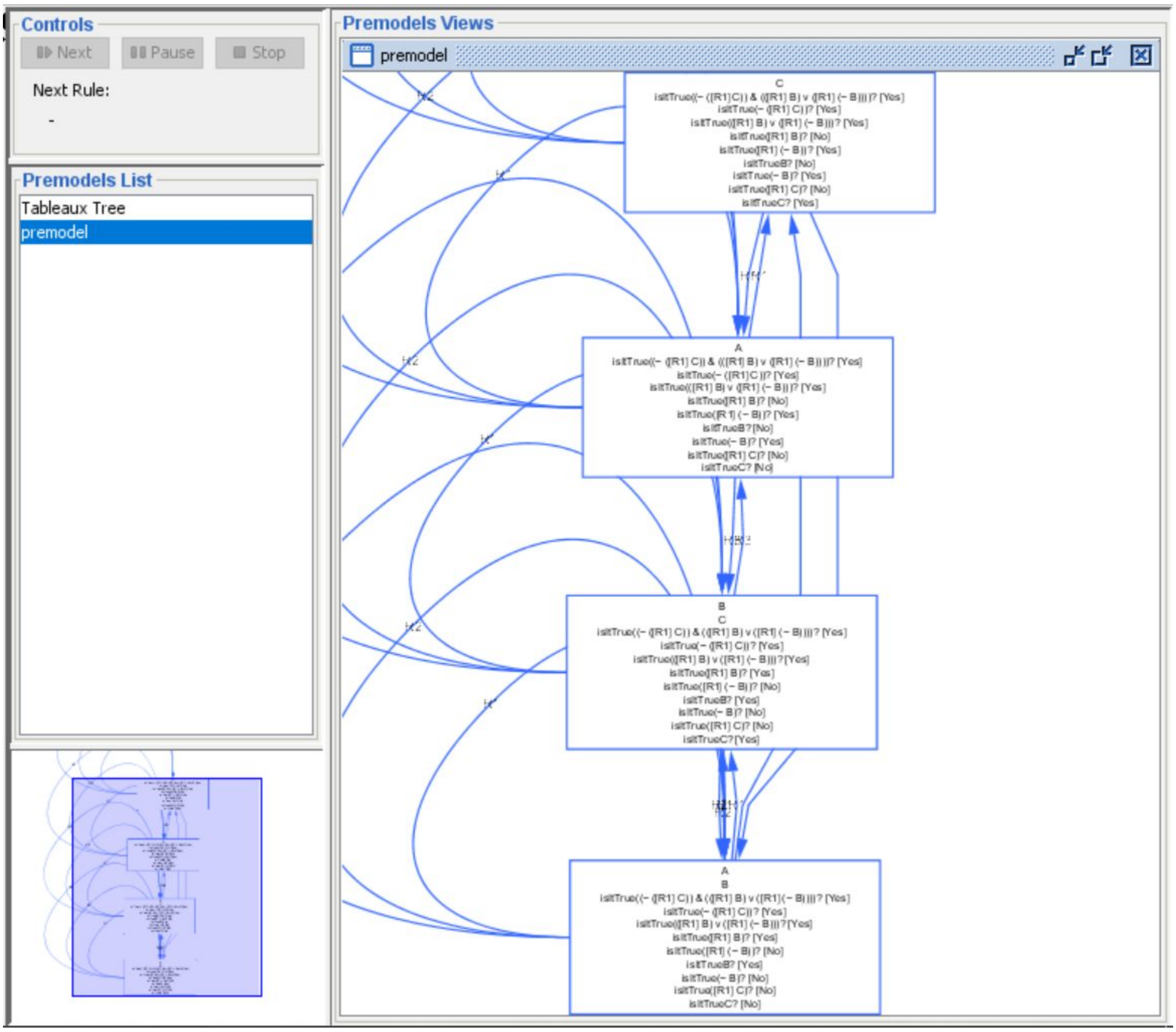


b. Montrons que $M \models \neg K_1 c \wedge (K_1 b \vee K_1 \neg b)$ soit valide. La formule $\neg K_1 c \wedge (K_1 b \vee K_1 \neg b)$ donne sous forme préfixe : and not nec R1 C or (nec R1 B nec R1 not B).

En rentrant :

- add w_1 isItTrue and not nec R1 C or (nec R1 B nec R1 not B)
- add w_2 isItTrue and not nec R1 C or (nec R1 B nec R1 not B)
- add w_3 isItTrue and not nec R1 C or (nec R1 B nec R1 not B)
- add w_4 isItTrue and not nec R1 C or (nec R1 B nec R1 not B)

On voit que le résultat est vrai dans tous les mondes de M. Donc que $M \models \neg K_1 c \wedge (K_1 b \vee K_1 \neg b)$ est valide.

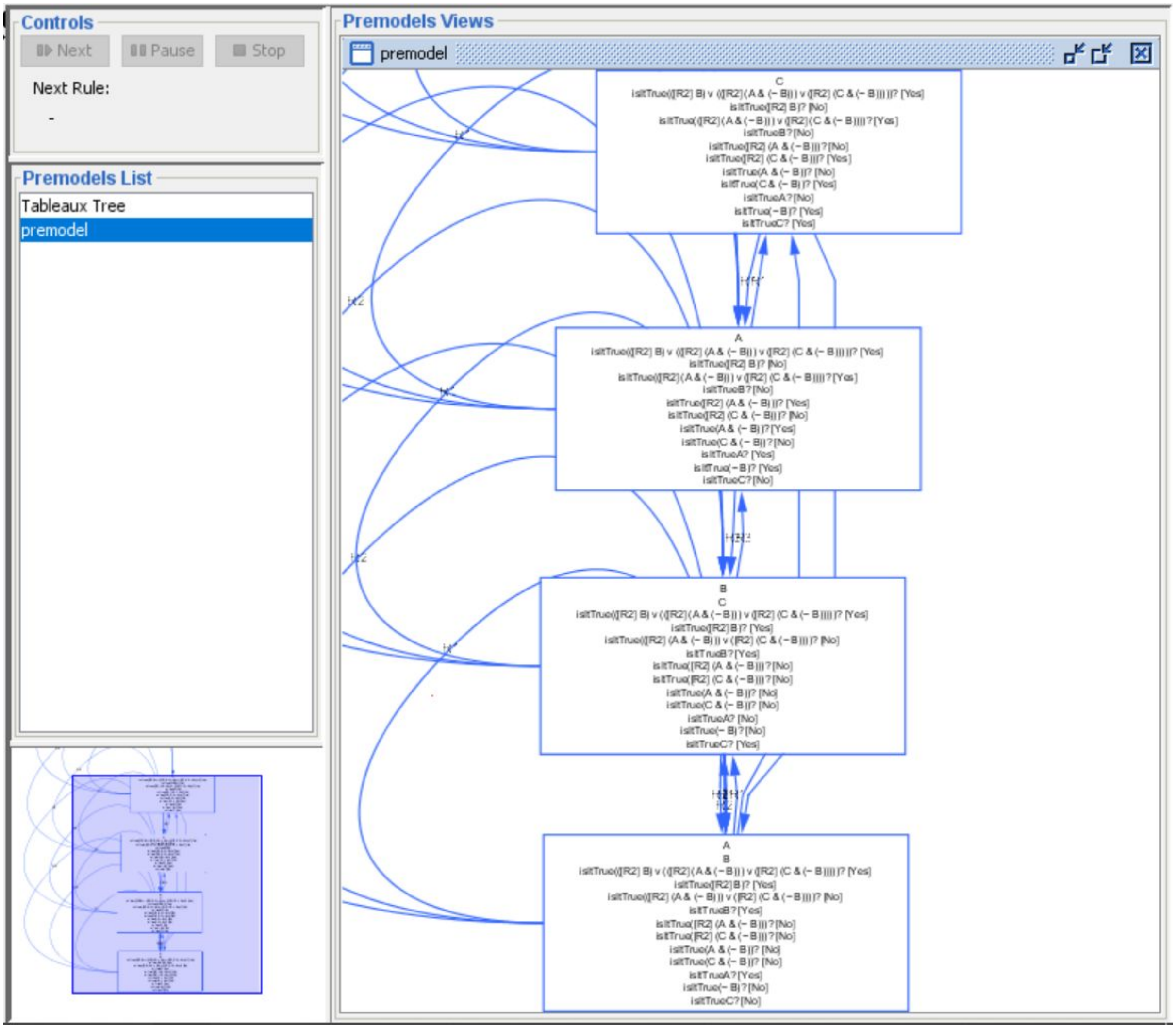


c. $M \models K_2b \lor K_2(a \land \neg b) \lor K_2(c \land \neg b)$ soit valide. La formule $K_2b \lor K_2(a \land \neg b) \lor K_2(c \land \neg b)$ donne sous forme préfixe : or nec R2 B or nec R2 (and A not B) nec R2 (and C not B).

En rentrant :

- add w_1 isItTrue or nec R2 B or nec R2 (and A not B) nec R2 (and C not B)
- add w_2 isItTrue or nec R2 B or nec R2 (and A not B) nec R2 (and C not B)
- add w_3 isItTrue or nec R2 B or nec R2 (and A not B) nec R2 (and C not B)
- add w_4 isItTrue or nec R2 B or nec R2 (and A not B) nec R2 (and C not B)

On voit que le résultat est vrai dans tous les mondes de M . Donc que $M \models K_2b \lor K_2(a \land \neg b) \lor K_2(c \land \neg b)$ est valide.


Controls
Next
Pause
Stop
Next Rule:
-
Premodels List
Tableaux Tree
premodel
Premodels Views
premodel