Ministère de l'Enseignement Supérieur et de Recherche Scientifique
UNIVERSITE KASDI MERBAH OUARGLA
Faculté des Nouvelles Technologies de l'Information et de la Communication
Département d'Informatique et des Technologies de l'Information

Mémoire de Master Professionnel

Domaine : Informatique et Technologie de l'Information

Filière : Informatique

Spécialité : Informatique Fondamental

Présenté par : Legmairi Sonia

Thème

Conception et Implémentation d'un Outil de Preuve basé sur la Méthode des Tableaux Sémantiques

Soutenue le :11/10/2020

Devant le jury composé de :

Dr .TOUMI Chahrazed	Université d'Ouargla	Président du jury
Dr .Chama Wafa	Université d'Ouargla	Superviseur
Dr. MAARAF Nadia	Université d'Ouargla.	Membre du jury

Année universitaire : 2019/ 2020

الاهداء

الى من أفضلها على نفسي ، ولم لا فلقد ضحت من أجلي ولم تدخر جهدا في سبيل راحتي، وتيسرت دروبي بطهر دعواتها،

الى من احتوت وطمأنت وتفانت في العطاء وشجعت،

الى من تستحق كل نجاحاتي الى أعظم امرأة في حياتي "أمي "

الى من تحملت يداه قسوة الحياة لننعم برغد العيش ،

الى من سعى من اجل راحتي ونجاحي، وكان اول شخص تلقيت منه التشجيع والدعم من أجل مواصلتي الدراسة

الى ذلك الرجل العظيم "أبي"

الى جدران روحي الأربعة ، الى الذين كل ما جار علينا الزمان جادوا ، الى إخوتي" عبد الفتاح ، عرفات ، محمد شعيب ، يوسف"

الى من شدت أزري كلما وهنت ، صديقتي ورفيقة دربي ، وأغلى ما أملك

أختي " رميصاء "

الى كل أفراد عائلتي من صغيرها وصولا لكبيرها

الى من ساهم في تعليمي ولو حرف واحد من معلمي الابتدائي الى الاساتذة والدكاترة الجامعيين .

الى من زرع في حب الاطلاع والإيمان بان لا شيء مستحيل اذا سعينا بجد واجتهاد واخص بالذكر زوجي المستقبلي "كحلة محمد الحبيب" ،

وأستاذتي المشرفة " وفاء شـــامة ".

الى من استواه موضوعنا وأحب الاطلاع على محتوى مذكرتنا المتواضعة .

Remerciements

Avant toute chose, nous remercions «ALLAH », l'omnipotent, pour nos avoir donné la force, la patience et le courage pour mener ce travail à son terme.

Je tiens à exprimer ma reconnaissance à ma directrice de mémoire, Madame CHAMA WAFA. Je la remercie de m'avoir encadrée, orientée, aidée et conseillée.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions.

Je remercie mes très chers parents, MILOUDE et FARIDA, qui ont toujours été là pour moi.

Résumé

De nos jours, la technologie joue un rôle important dans tous les domaines. L'utilisation de l'outil informatique pour la mise en œuvre d'un système de preuve pour les mathématiciens et les informaticiens, résout des problèmes de complexité, d'efficacité, et de correction.

Dans ce mémoire, nous proposons d'automatiser l'une des méthodes de décision, la méthode des tableaux sémantiques pour la logique propositionnelle en utilisant le langage Java. Notre outil permet non seulement la preuve des formules mais également fournit tous les interprétations vraies.

Mots clés : logique propositionnelle, formule, méthodes de preuve, méthode des tableaux sémantiques, Java.

Abstract

Nowadays, technology plays an important role in all fields. The use of the computer science to implement a proof system for mathematicians and computer scientists solves complexity problems, efficiency, and correctness.

In this thesis, we propose to automate one of the decision methods, semantic tables' method for propositional logic using Java language. Our tool allows not only formulas' proof, but also provides all true interpretations.

Keywords: propositional logic, formula, proof methods, method of semantic tables, Java.

ملخص

في الوقت الحاضر، تلعب التكنولوجيا دورًا مهمًا في جميع المجالات. فإن استخدام أداة تكنولوجيا المعلومات لتنفيذ نظام إثبات لعلماء الرياضيات و الكمبيوتر، يحل مشاكل التعقيد، الكفاءة وكذا الصحة.

في هذه المذكرة ، نقترح تحويل اوتوماتيكي لإحدى طرق إتخاذ القرار ، وهي طريقة الجداول الدلالية للمنطق الافتراضي باستخدام لغة الجافا. لا تسمح أداتنا بإثبات الصيغ فحسب، بل توفر أيضًا جميع التفسيرات الحقيقية.

الكلمات المفتاحية : المنطق الإفتراضي ، المعادلة ، طرق الإثبات ، طريقة الجداول الدلالية ، جافا

Table des matières

Liste des figures

Liste des tables

Introduction Générale

Introduction Générale

La logique au sens philosophique du terme, remonte à la période grecque. Le premier qui a enseigné la logique est le philosophe « Aristote ». Par la suite, ces livres ont été traduits en Arabe, par le savant musulman « Ibn Sina Farabi » dans la période Abbasside [1].

On peut définir la logique comme : « la science qui étudie les règles générales du raisonnement correct »

Donc, le but d'étudier la logique est :

- Raisonner correctement
- Résoudre les problèmes complexes
- Trouver les solutions rapidement

La logique mathématique est apparue vers la fin du 19 e siècle, Russel et Frege se sont parmi les fondateurs de cette science. A cette époque il y avait beaucoup de problèmes en mathématiques ; énoncés non encore démontrés, paradoxes, etc. Ces leaders ont fondé les bases de la logique mathématique afin de résoudre ces problèmes en représentant les énoncés mathématiques sous forme de formules ensuite les démontrer avec des méthodes de raisonnement rigoureuses [2].

L'utilisation de la logique mathématique n'est pas limitée dans le domaine théorique pur, mais elle a fortement contribué à la naissance des premiers ordinateurs. La binarité de la valeur de vérité est la base de tous les circuits électroniques qui composent l'ordinateur. Les bases de la logique mathématique ont beaucoup contribué dans les applications de l'intelligence artificielle.

La logique classique est le premier langage qui sert à formuler tous les raisonnements mathématique appelée simplement logique au début. Avec la révolution, d'autres systèmes logiques formels sont apparues. Notamment pour la logique intuitionniste, qui a suscité l'adjonction de l'adjectif classique au terme logique.

La logique classique est caractérisée par des postulats qui la fondent et la différencient de la logique intuitionniste, exprimés dans le formalisme du calcul des propositions ou du calcul des prédicats [3].

Il existe d'autres logiques non-classiques tels que :

- La logique modale
- La logique temporelle
- La logique de description
- La logique trivalente

- La logique floue
- La logique probabiliste

Le calcul des propositions est l'un des langages formels privilégiés de la logique mathématique pour la formulation de ses concepts en systèmes formels, en raison de son applicabilité aux fondements des mathématiques et de la richesse de ses propriétés relevant de la théorie de la démonstration.

L'un des moyens de preuve les plus importants en logique la méthode des tableaux sémantique. Cette méthode est un algorithme de résolution du problème de la décision pour le calcul des propositions, ainsi est une méthode de preuve pour la logique du premier ordre. La méthode des tableaux peut également déterminer la satisfiabilité des ensembles finis de formules de diverses logiques. C'est la méthode de preuve la plus populaire pour les logiques modales.

Sur la base de ce qui précède, nous avons proposé dans notre sujet d'utiliser la méthode des tableaux sémantiques afin de prouver toutes les propriétés en logique propositionnelle. Egalement, de concevoir un programme Java qui l'utilise pour expliquer les propriétés d'une formule simple, complexe et aussi d'un ensemble de formules.

Dans le premier chapitre, nous allons étudier le langage de la logique propositionnelle avec ses composants. Après, dans le deuxième chapitre, nous allons voir toutes les parties de la sémantique du calcul propositionnel. Ensuite, dans le troisième chapitre, nous allons parler de la méthode des tableaux sémantiques et comment l'exploiter. Enfin, dans le dernier chapitre, nous allons donner quelques exemples d'utilisation de notre outil qu'été implémenté en utilisant le langage Java avec les résultats obtenus.

Chapitre 1 : *Syntaxe de la Logique Propositionnelle*

1. Introduction

Toute logique est définie par une syntaxe, une sémantique et un système de preuve. Ce chapitre explique en détail avec des exemples les concepts de la syntaxe propositionnelle ; sa structure linguistique, comment elle est écrite et les caractéristiques structurelles les plus importantes.

2. Logique Propositionnelle

Le calcul des propositions ou calcul propositionnel est une théorie logique ayant pour objet l'étude des relations logiques entre « propositions » et la définition des lois formelles selon lesquelles, au moyen de connecteurs logiques, les propositions se coordonnent et s'enchaînent pour produire des raisonnements valides.

En logique mathématique, le calcul des propositions est la première étape dans la définition de la logique et du raisonnement. Il définit les règles de déduction qui relient les propositions entre elles, sans examiner le contenu. Ainsi, il est la première étape dans la construction du calcul des prédicats, qui lui s'intéresse au contenu des propositions et qui est une formalisation achevée du raisonnement mathématique. Le calcul des propositions est parfois appelé logique des propositions, logique propositionnelle ou calcul des énoncés, et parfois théorie des fonctions de vérité. [3]

3. Syntaxe

En générale, la syntaxe est un langage qui définit les règles de construction des formules propositionnelles. Une formule de cette logique est construite à l'aide de connecteurs logiques, de variables propositionnelles et des constantes vrai (parfois représentée par "1" ou "⊤" ou "V") et faux (parfois représentée par "0" ou "⊥" ou "F") [4].

4. Proposition

Une proposition est un énoncé auquel on peut attribuer une valeur de vérité *vrai* ou *faux* [5].

On écrit tout court P, Q afin de désigner une proposition

Exemple : L'enseignent explique le cours ⇒ une proposition

L'enseignent est-il arrivé ? Non proposition, nous ne pouvons pas attribuer une valeur de vérité.

5. Langage propositionnel

Le langage propositionnel L se compose de l'ensemble des alphabets suivants :

- Les constantes
- Les atomes (variables propositionnelles)
- Les connecteurs logiques {↔, →, ∧, ∨, ¬}
- Les parenthèses (et).

[6] ,[7]

5.1. Constantes

⊤ et ⊥ (1 et 0 ou V et F) représentent respectivement le vrai et le faux.

5.2. Atomes

Nous appelons atomes ou variables propositionnelles ou propositions élémentaires des énoncés dont nous ne connaissons pas (et ne cherchons pas à connaître) la structure interne, et qui gardent leur identité tout au long du calcul propositionnel qui nous occupe.

- L'ensemble des variables propositionnelles est noté v(L).
- On désigne généralement des propositions élémentaires par des lettres de l'alphabet majuscule (habituellement A, B, C…). [7]

5.3. Connecteurs logiques

A partir d'une, deux ou plusieurs propositions on peut créer de nouvelles propositions à l'aide de *connecteurs logiques*. Nous allons définir les règles pour les cinq connecteurs 'non', 'et', 'ou', 'si …alors' et 'si et seulement si' comme suit. [6]

5.3.1. Négation

La négation d'une proposition est une proposition qui est vraie si celle-ci est fausse et vice-versa.

Notation : ¬

Quelques traductions usuelles :

- "non"
- "il est faux que"

- "ne pas".

5.3.2. Conjonction

La conjonction de deux propositions est une proposition qui est vraie si les deux propositions sont simultanément vraies. Elle est fausse dès que l'une au moins des deux propositions est fausse.

Notation : ∧

Quelques traductions usuelles :

- "et"
- "mais"
- "quoique"
- "bien que".

5.3.3. Disjonction

La disjonction de deux propositions est une proposition qui est vraie dès que l'une au moins des deux propositions est vraie. Elle est fausse si les deux propositions sont simultanément fausses.

Notation : ∨

Quelques traductions usuelles :

- " ou "
- "à moins que".

5.3.4. Implication

Si P et Q sont deux propositions, alors l'implication "si *P* alors *Q*" est une proposition qui est vraie si *P* est faux, ou bien si *P* et *Q* sont simultanément vrais. Cette implication est fausse uniquement si l'antécédent P est vrai et le *conséquent Q* faux.

Notation : →

Quelques traductions usuelles :

- "si …alors"
- "implique"
- "a pour conséquence"
- " donc"

L'implication *P*→Q sera aussi traduite par "*P* seulement si *Q*", car la proposition "*P* seulement si *Q"* n'est faux que dans le cas ou *P* est vrai et *Q* est faux, et elle a donc la même table de vérité que P→Q.

5.3.5. Equivalence

Si *P* et *Q* sont des propositions, alors l'équivalence "*P* si et seulement si *Q*" est une proposition qui signifie *(P si Q) et (Q seulement si P).*

La valeur de vérité de P↔Q est la valeur de vérité de (P→Q) ∧ (*Q* →P). L'équivalence 'P si et seulement si *Q'* est donc vraie uniquement si *P* et *Q* ont la même valeur de vérité.

Notation : ↔

Quelques traductions usuelles :

- "si et seulement si"
- "équivaut à"
- "équivaut à dire"
- "revient à dire"

La négation est un connecteur *unaire* (elle n'a qu'un argument), alors que les autres connecteurs sont *binaires* (elles ont deux arguments). La conjonction, la disjonction et l'équivalence sont commutatives dans le sens que les propositions *P* ∧*Q* et *Q* ∧ *P* (respectivement P∨Q et Q∨P, ou P↔Q et Q↔P) ont la même table de vérité.

L'implication n'est pas commutative ; les propositions *P*→*Q* et *Q*→*P* n'ont pas la même table de vérité.

5.4.Parenthèses

- On omet toujours les parenthèses les plus à l'extérieur. En général, l'omission de parenthèses peut être source d'ambiguïtés. [8]

Ex : ¬P∧Q peut correspondre à (¬P) ∧Q et à ¬ (P ∧Q)

- Cependant, on peut souvent omettre les parenthèses en donnant des priorités aux connecteurs, dans l'ordre suivant de la plus forte à la plus faible ; les opérateurs sur une même ligne ont la même priorité :

1. ¬
2. ∧, ∨
3. →, ↔

Ex : ¬ P ∧ Q ∨ P↔ Q ∧ P correspond à (((¬ P) ∧Q) ∨ P) ↔ (Q ∧ P)

Attention : Il n'existe pas de convention universelle pour la priorité en logique. Elle peut varier d'un livre à l'autre et d'un outil à l'autre. Nous allons utiliser cette priorité dans notre travail.[8]

6. Formules Propositionnelles

6.1. Formules atomiques

Une formule est atomique si c'est ⊤, ⊥ ou si c'est une variable propositionnelle [7].

6.2. Formules bien formées

L'ensemble des formules de la logique propositionnelle (ou formules bien formées) est le plus petit ensemble de mots construits sur l'alphabet tel que : [8]

- Si A est une formule atomique alors A est une formule
- (¬A) est une formule si A est une formule
- (A ∧B) est une formule si A et B sont des formules
- (A ∨B) est une formule si A et B sont des formules
- (A→B) est une formule si A et B sont des formules
- (A ↔ B) est une formule si A et B sont des formules

Exemple

NON FORMULES	**FORMULES BIEN FORMEES**
P ∧	P
P ¬Q	(¬P)
(Q→ (P ∨Q)	(Q → (P ∨ Q))

Table 1 : *Exemple de formules bien formée*

L'ensemble des sous formules d'une formule φ est le plus petit ensemble tel que :

- φ est une sous-formule de φ.
- Si (¬B) est une sous-formule de φ alors B est une sous formule de φ.
- Si B ∧C est une sous-formule de φ alors B et C sont des sous-formules de φ.

- Si B v C est une sous-formule de φ alors B et C sont des sous-formules de φ.
- Si B →C'est une sous-formule de φ alors B et C sont des sous-formules φ.
- Si B↔C est une sous-formule de φ alors B et C sont des sous-formules de φ.

L'endroit où une sous-formule apparaît est son occurrence. [8]

Exemple

Soit la formule φ = ((P v Q) ∧¬P).

Sous formules de φ = {P, Q, ¬P, P v Q, P v Q ∧¬ P}

7. Arbres de construction

Il est important de noter que toute formule bien formée a un et un seul arbre de construction ou arbre de décomposition (c'est une conséquence du fait que le langage de la logique est syntaxiquement non ambigu) ; et que de surcroît apparaissent dans cet arbre exactement toutes ses sous-formules.[9]

Processus de décomposition

- On découpe au niveau du connecteur principal d'une formule.
- On enlève les parenthèses les plus externes de chacun des arguments du connecteur principal.
- On répète l'opération de décomposition pour chacun des arguments jusqu'à arriver aux atomes.

Exemple

((¬ (P ∨Q) →¬¬¬Q) ↔ R) peut être décomposée de la manière représentée à la figure 1.

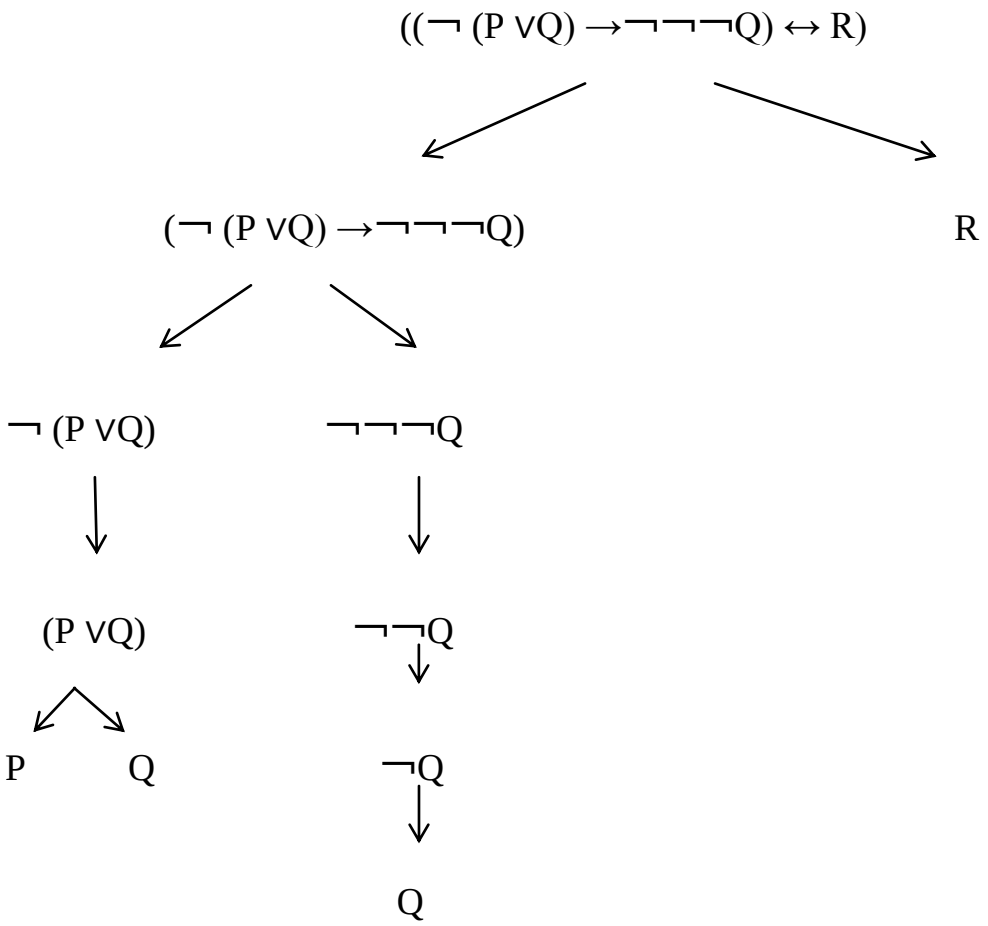


Figure 1 : *Exemple d'arbre de construction.*

La formule précédente est bien formée :

Elle a été construite avec le connecteur ↔, et les deux sous-formules ¬(P∨Q) → ¬¬¬Q d'une part et R d'autre part.

Le connecteur ↔ est appelé connecteur principal de la formule. La formulation des règles de syntaxe garantit que toute formule à un et un seul signe principal. On voit que pour garantir que la formule initiale est bien formée, il faut maintenant vérifier que les deux premiers fils dans l'arbre correspondent sont des formules bien formées. Il faut donc réitérer le processus, jusqu'à aboutir à des propositions élémentaires comme R.

8. Règles Simplificatrices

Les règles simplificatrices de la logique propositionnelle sont analogues à celles de l'arithmétique. Elles sont destinées à abréger l'écriture des formules et à faciliter la lecture.
Voici d'abord deux règles d'un emploi habituel :[10]

- Omission des parenthèses extérieures :

  On écrit : P ∧ Q au lieu de (P ∧ Q)

  Q ↔ ¬ (Q → ¬ Q) au lieu de (Q ↔¬ (Q → ¬Q)).

- Utilisation de l'associativité des opérateurs ∧ et ∨ :

  On écrit P ∨ Q ∨ R au lieu de (P ∨ Q) ∨ R ou P ∨(Q ∨ R).

Certains ouvrages utilisent en outre les règles suivantes :

- Groupement à gauche des opérateurs non associatifs :

  On écrit parfois P → Q → R au lieu de (P → Q) → R.

- Priorité des opérateurs :

  En arithmétique, on écrit souvent A + B ∗ C pour A + (B ∗ C) ; de même on applique les règles de précédence (priorité) entre connecteurs comme suit :

  - ¬
  - ∨, ∧ avec associativité à gauche
  - →, ↔ avec associativité à gauche

Exemple

- A↔¬B∧C→¬A∨D∧B
- A↔ (¬B) ∧C → (¬A) ∨D∧B …...….¬.
- A↔ ((¬B) ∧C) → (¬A) ∨ D ∧ B ……. ∨, ∧ avec associativité à gauche.
- A↔ ((¬B) ∧C) → ((¬A) ∨ D) ∧B ……. ∨, ∧ avec associativité à gauche
- A↔ ((¬B) ∧ C) → (((¬A) ∨D) ∧B) …….. ∨, ∧ avec associativité à gauche.
- (A ↔ ((¬B) ∧C)) → (((¬A) ∨ D) ∧ B) ……. →, ↔ avec associativité à gauche.
- ((A↔ ((¬B) ∧ C)) → (((¬A) ∨ D) ∧ B)) ……. →, ↔ avec associativité à gauche.

9. Notation polonaise

Les logiciens polonais ont proposé des notations permettant de se passer complètement des parenthèses. La notation polonaise directe, ou préfixée, consiste à écrire l'opérateur avant ses opérandes ; la notation polonaise inverse, ou postfixée, consiste à écrire l'opérateur après l'ordre de parcours des nœuds dans l'arbre de dérivation. Dans notre travail nous avons utilisé la notation préfixée.

Exemple

Notation infixée	((P ⇒ Q) ↔ (¬Q ⇒ ¬P))	(P ⇒ (Q ↔ ¬(Q⇒ ¬P)))
Notation simplifiée	P ⇒ Q ↔ (¬Q ⇒ ¬P)	P ⇒ (Q ↔ ¬(Q ⇒ ¬P))
Notation préfixée	↔⇒ P Q ⇒ ¬ Q ¬ P	⇒ P ↔ Q ¬ ⇒ Q ¬ P
Notation postfixée	P Q ⇒ Q ¬ P ¬ ⇒↔	P Q Q P ¬ ⇒ ¬ ↔⇒

Table 2 : *Les Notations Polonaises*

Les calculatrices de poche Hewlett-Packard proposent ou imposent l'usage de la notation postfixée, ce qui a contribué à rendre cette notion familière aux non-logiciens. [10]

10. Conclusion

Ce chapitre a pour but de représenter la syntaxe de la logique propositionnelle. Nous avons parlé en détail de tous les symboles qui composent le langage propositionnel, les constantes, les atomes, les connecteurs logiques et les délimiteurs. Puis, nous avons présenté les types des formules bien formées quelques soit les formules atomiques, formules composées et les sous-formules. Enfin, nous avons expliqué le processus d'obtenir un arbre de construction et comment utiliser les règles de simplification et même comment parcourir un arbre afin d'aboutir la notation polonaise.

Nous concluons que la logique propositionnelle est la base de tous les autres types de logique en termes de syntaxe.

Dans le chapitre suivant, nous présentons la sémantique de la logique propositionnelle et comment assigner une valeur de vérité à une formule.

Chapitre 2 : *Sémantique de la Logique Propositionnelle*

1. Introduction

Ce chapitre détermine le cœur de toute logique, la sémantique formelle qui concerne l'aspect sens et signification.

En général, la sémantique offre une assignation des valeurs de vérité aux constructions linguistiques qu'offre le langage déjà étudié dans le chapitre précédent.

La sémantique de la logique propositionnelle étude le sens des unités linguistiques et de leurs combinaisons. En d'autres termes, elle détermine la valeur de vérité d'un énoncé, ou d'une formule.

Nous parlons de l'interprétation d'une formule qu'il s'agit plus concrètement d'affecter une valeur vraie ou fausse à chacune des variables propositionnelles qui la compose. Pour une formule à n variables, il y a 2^n valuations possibles.

2. Valuation

On appelle valuation, ou L-Modèle, d'un ensemble de variables propositionnelles v(L), une fonction v de v(L) dans {0, 1}
(v : v(L) → {0, 1}).

- v (¬X) = ¬ v(X)
- v(X k Y) = v(X) k v(Y) tel que : k ∈ {∧, ∨, ↔, →}.[11]

3. Interprétation

3.1. Interprétation d'une formule

Une valuation appliquée à une formule appelée interprétation, est une fonction L de {φ1, φ2, ..., φn} dans {0,1}. L(φi) est la valeur de vérité (v ou f) attribuée à φi. La signification d'une formule dépend des valeurs de vérité des variables qu'elle contient et de la sémantique des connecteurs (vérifonctionnalité). Il peut être étendue à l'ensemble des formules par :[2]

- L(φ) = v(φ) si φ est une variable
- Si φ= ¬φ1 alors L(φ) = ¬ L(φ1)
- Si φ= φ1 k φ2 alors L(φ) = L(φ1) k L(φ2) avec k ∈ {∧, ∨, ↔, →}

3.2. Interprétation des connecteurs

La proposition P	Sa négation ¬P
0	1
1	0

Table 3: *Interprétation de la négation*

P	Q	P∧Q	P∨Q	P→Q	P↔Q
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	0
1	1	1	1	1	1

Table 4 : *Interprétation des connecteurs*

4. Modèle d'une formule

Une valuation v des variables d'une formule φ est dite modèle si L(φ)= 1 par cette valuation.

On appelle modèle v d'un ensemble de formules F = {φ1, φ2, ..., φk} une interprétation qui rend vraie chaque formule φ1, φ2, ..., φk . [3]

L(φ1)= L(φ2)=….= L(φk)=1 ⇒ v ⊨ F

5. Table de vérité réduite

La connaissance de la valeur de vérité d'une sous-formule peut permettre de déterminer la valeur globale de la formule (voir table 5).[12]

Par exemple si v(B) = 1, on sait sans connaître v(A) que L(A →B) = 1

On peut donc éviter d'énumérer tous les cas (cependant il devient moins évident que tous les cas sont énumérés et cela doit être vérifié).

A	B	C	A∧B	**(A∧B)→C**	B→C	**A→(B→C)**
?	?	1	?	**1**	1	**1**
0	?	0	0	**1**	?	**1**
1	0	0	0	**1**	1	**1**
1	1	0	1	**0**	0	**0**

Table 5: *Exemple d'une table de vérité réduite*

Dans la littérature, il existe une méthode s'appelle Diagramme de Quine qui vise à déterminer la table de vérité réduite. Pour plus d'information voir [4]

6. Formes Normales d'une formule propositionnelle

Dans l'exemple précédent (table 5), nous avons vu qu'il existe des possibilités pour exprimer une formule dans une forme équivalente. En fait, pour toute formule φ il y a une infinité de formules équivalentes :

φ, φ∧φ, φ∧φ∧φ, etc. D'autres pour lesquelles l'équivalence peut être beaucoup moins évidente.

Vraiment, nous n'avons pas besoin de toute cette multitude de possibilités d'exprimer la même chose. Généralement, quand les informaticiens travaillent avec des expressions symboliques, ils définissent souvent un tel standard, appelé une « forme normale ». Dans cette section, nous allons étudier quelques formes normales des formules propositionnelles.

6.1. Forme Normale de Négation

Une formule est en forme normale de négation si [5] :

- elle est construite avec les connecteurs ¬, ∧ et ∨ seulement ;
- elle ne contient pas d'applications de l'opérateur ¬ sauf des applications devant les variables propositionnelles.

Exemple

- ¬x ∧¬y ∧ z et (¬x ∨ y) ∧ (¬z ∨ x) sont en forme normale de négation
- ¬(x∧y) et ¬¬y ne le sont pas

Les auteurs de [5] donnent un ensemble de règles de réécriture pour transformer une formule en une forme normale de négation équivalente :

1) ¬¬X → X
2) ¬(X ∧Y) → (¬X ∨¬Y)
3) ¬(X ∨Y) → (¬X ∧¬Y)

Exemple : ¬(X∧(Y∨¬Z))

- se transforme en ¬X∨¬(Y∨¬Z) par la règle (2)
- se transforme en ¬X∨(¬Y∧¬¬Z) par la règle (3)
- se transforme en ¬X∨(¬Y∧Z) par la règle (1)

6.2.Forme Normale Disjonctive

Une formule est en forme disjonctive normale si elle est [7] :

- soit la constante False
- soit une clause conjonctive
- soit une disjonction d'aux moins deux clauses conjonctives

On écrit parfois plus brièvement DNF, c'est l'abréviation des mots en anglais « disjonctive normal forme ».

- Par une conjonction d'aux moins deux littéraux, nous entendons une formule de la forme.

  (L1 ∧(L2 ∧(L3 ∧...)...)) où chacun des Li est un littéral.

- Et par une disjonction d'aux moins deux clauses conjonctives une formule de la forme

  (C1 ∨(C2 ∨(C3 ∨...)...) où chacun des Ci est une formule conjonctive.

Pour transformer une formule en une formule normale disjonctive équivalente, nous commençons d'abord à la mettre en forme normale de négation, puis nous transformons la

formule obtenue en forme normale de disjonction à l'aide du système de réécriture suivant : [7]

4) $X \wedge (Y \vee Z) \rightarrow (X \wedge Y) \vee (X \wedge Z)$
5) $(X \vee Y) \wedge Z \rightarrow (X \wedge Z) \vee (Y \wedge Z)$
6) $(X \wedge Y) \wedge Z \rightarrow X \wedge Y \wedge Z$
7) $(X \vee Y) \vee Z \rightarrow X \vee Y \vee Z$

Par exemple : $X1 \wedge (\neg(Y1 \vee Y2) \wedge \neg\neg(Z1 \vee Z2))$

- Se transforme en $X1 \wedge (\neg(Y1 \vee Y2) \wedge (Z\,1 \vee Z\,2))$ par règle 1
- Se transforme en $X1 \wedge ((\neg Y1 \wedge \neg Y2) \wedge (Z\,1 \vee Z\,2))$ par règle 3
- Se transforme en $X1 \wedge (((\neg Y1 \wedge \neg Y2) \wedge Z\,1) \vee ((\neg Y1 \wedge \neg Y2) \wedge Z\,2))$ par règle 4
- Se transforme en $X1 \wedge ((\neg Y1 \wedge \neg Y2 \wedge Z\,1) \vee (\neg Y1 \wedge \neg Y2 \wedge Z\,2))$ par règle 6
- Se transforme en $(X1 \wedge \neg Y1 \wedge \neg Y2 \wedge Z\,1) \vee (X1 \wedge \neg Y1 \wedge \neg Y2 \wedge Z\,2)$ par règle 5

6.3.Forme Normale Conjonctive

La forme conjonctive normale est définie de façon analogue à la forme disjonctive normale :

Une formule est en forme conjonctive normale (abrégé CNF) si elle est la conjonction de zéro ou plus clauses disjonctives.

- Par une disjonction d'aux moins deux littéraux, nous entendons une formule de la forme :

$(L1 \vee (L2 \vee (L3 \vee ...)...))$ où chacun des Li est un littéral.

- et par une conjonction d'aux moins deux clauses disjonctives, une formule de la forme :

$(C1 \wedge (C2 \wedge (C3 \wedge ...)...)$ où chacun des Ci est une formule disjonctive.

Exemple $(X \vee Y) \wedge (\neg Z \vee Y \vee \neg X)$ est en forme normale conjonctive. [7]

Remarque importante

Il y a des formules qui sont à la fois en forme disjonctive normale et en forme conjonctive normale.

Exemple : $(X \vee \neg Y \vee Z)$

Nous pouvons voir comme une *seule clause disjonctive* et donc une formule en forme conjonctive normale; ou comme une *disjonction de trois clauses conjonctives* qui chacune consiste en un seul littéral, et donc comme une formule en forme disjonctive normale.

La procédure est analogue à celle de la mise en forme disjonctive normale, sauf que les deux premières règles de transformation (4) et (5) sont à remplacer par : [7]

8) $X \vee (Y \wedge Z) \rightarrow (X \vee Y) \wedge (X \vee Z)$

9) $(X \wedge Y) \vee Z \rightarrow (X \vee Z) \wedge (Y \vee Z)$

7. Validité et Consistance

7.1. Formule Satisfiable (Consistante)

Une formule satisfiable ou sémantiquement consistante est une formule vraie dans au moins une interprétation. $\exists\, v / v \models \varphi$. [8]

Exemple

- X
- $X \rightarrow Y$

7.2. Formule Valide

Une formule valide, ou tautologie, est une formule φ satisfiable dans toute interprétation, quelles que soient les valeurs de vérité des atomes qui la composent.

$\forall\, v / v \models \varphi$, on la note : $\models \varphi$. [8]

Exemple

- $X \vee \neg X \vee Y$
- $X \rightarrow X$

7.3. Formule Insatisfaite (Inconsistante)

Une formule insatisfaite, ou sémantiquement inconsistante, ou encore antilogie, est une formule fausse dans toute interprétation. $\forall\, v / L(\varphi)=0$. [8]

Exemple

- $X \wedge \neg X$
- $(X \vee \neg X) \rightarrow (X \wedge \neg X)$

7.4.Formule Invalide

Une formule invalide est fausse dans au moins une interprétation. $\exists$ v / L(φ)=0. [8]

Exemple

- X
- $X \rightarrow Y$

7.5.Formule Contingente

Une formule contingente est vraie dans certaines interprétations et fausse dans d'autres. [9]

8. Équivalence des formules bien formées

Deux formules sont équivalentes quand elles ont une même valeur dans toutes interprétation (notation : $A \equiv B$).

Soient A, B et C trois formules bien formées. [10]

1. Par commutativité :
   - $\models [(A \wedge B) \equiv (B \wedge A)]$
   - $\models [(A \vee B) \equiv (B \vee A)]$
   - $\models [(A \equiv B) \equiv (B \equiv A)]$
2. Par associativité :
   - $\models [((A \wedge B) \wedge C) \equiv ((A \wedge (B \wedge C))]$
   - $\models [((A \vee B) \vee C) \equiv ((A \vee (B \vee C))]$
   - $\models [((A \equiv B) \equiv C) \equiv ((A \equiv (B \equiv C))]$
3. Par distributivité
   - $\models [((A \wedge B) \vee C) \equiv ((A \vee C) \wedge (B \vee C))]$
   - $\models [((A \vee B) \wedge C) \equiv ((A \wedge C) \vee (B \wedge C))]$
4. Par absorption :
   - $\models [A \wedge (A \vee B)) \equiv A]$
   - $\models (A \vee (A \wedge B)) \equiv A]$

5. Par idempotence
   - $\models [(A \wedge A) \equiv A]$
   - $\models [(A \vee A) \equiv A]$
6. Par les lois de Morgan :
   - $\models [\neg(A \wedge B) \equiv (\neg A \vee \neg B)]$
   - $\models [\neg(A \vee B) \equiv (\neg A \wedge \neg B)]$
7. Par implication :
   - $\models [(A \rightarrow B) \equiv (\neg A \vee B)]$
   - $\models [(A \rightarrow B) \equiv \neg(A \wedge \neg B)]$
8. Par équivalence :
   - $\models [(A \equiv B) \equiv ((A \rightarrow B) \wedge (B \rightarrow A))]$
   - $\models [(A \equiv B) \equiv ((A \wedge B) \vee (\neg A \wedge \neg B))]$
9. Par contraposition :
   - $\models [(A \rightarrow B) \equiv (\neg B \rightarrow \neg A)]$
10. Par auto distributivité :
    - $\models [(A \rightarrow (B \rightarrow C)) \equiv ((A \rightarrow B) \rightarrow (A \rightarrow C))]$
11. Par import-export :
    - $\models [(A \rightarrow (B \rightarrow C)) \equiv (B \rightarrow (A \rightarrow C))]$
12. Par transitivité
    - $\models [((A \rightarrow B) \wedge (B \rightarrow C)) \equiv (A \rightarrow C)]$

9. Compatibilité

9.1.Formule compatible

Une formule φ est compatible si et seulement si φ a au moins un modèle (φ est satisfaite).

- φ est compatible $\Leftrightarrow \exists v / \ v \models \varphi$
- φ est incompatible $\Leftrightarrow \forall v \ L(\varphi)=0 \Leftrightarrow$ φ antilogie. [5]

9.2. Compatibilité d'un ensemble de formules

Un ensemble de formules est compatible si et seulement si chaque formule est satisfaite par le même modèle. [5]

Soient Σ un ensemble et φ une formule :

- Σ est compatible $\Leftrightarrow \exists v \ \forall \varphi \in \Sigma \quad v|=\varphi$
- Σ est incompatible $\Leftrightarrow \ \forall v \ \exists \varphi \in \Sigma \quad L(\varphi)=0$

Exemple

L'ensemble $\Sigma= \{A\rightarrow B, A\wedge \neg C, \ C \vee B\}$ est compatible pour v(A)=1, v(B)=1, v(C)=0

10. Conséquence logique (conséquence tautologique)

10.1. Conséquence d'une formule

Soient deux formules φ1 et φ2. Nous dirons que φ1 est la conséquence logique de φ2 (notée φ2 |= φ1) si tout modèle de φ2 est un modèle de φ1.

$\varphi 2 \models \varphi 1 \leftrightarrow \forall v \ (L(\varphi 2) =1 \rightarrow L(\varphi 1)=1)$. [10]

10.2. Conséquence d'un ensemble de formules

On dit que φ est une conséquence valide de l'ensemble Σ={ A,B,...} si tout modèle de Σ est un modèle de φ (en d'autre termes, dans la table de vérité on trouve que φ est vraie à chaque fois que A,B,... sont vraies). [10]

Notée : A,B,... |= φ.

Pour vérifier que φ est une conséquence valide de l'ensemble Σ, il faut :

- Vérifier si Σ est compatible si oui
- Trouver tous les modèles de Σ,
- Pour chaque modèle v de Σ, vérifier que v |= φ,

Exemple

- A , B |= A
- A , B |= A ∧B
- $A , A \rightarrow B \models B$

11. Conclusion

Dans ce chapitre, nous avons entamé la partie la plus importante et la plus délicate de cette étude. Il s'agit d'examiner toutes les parties de la sémantique *propositionnelle.* Réellement, nous trouvons énormément de problèmes qui peuvent être traités à travers les formules de la logique des propositions, néanmoins quand le nombre de variables est important ou des fois inconnu, l'utilisation des tables de vérité devient coûteuse en espace et en temps. Donc, nous devons chercher une méthode formelle qui ne fait pas appel aux tables de vérité. Dans le chapitre suivant, nous essayerons de clarifier la méthode des tableaux sémantique.

Chapitre 3 : *Méthode des Tableaux Sémantiques*

1. Introduction

Après la présentation de la syntaxe et de la sémantique de la logique propositionnelle, dans ce chapitre nous allons étudier l'une des méthodes qui est considéré comme la meilleure, la plus simple et la plus rapide des systèmes de preuve, les tableaux sémantiques. Nous allons essayer de la définir, de l'expliquer et de mentionner les algorithmes utilisés pour l'adapter.

2. Définitions

2.1. Littéral

On appelle littéral une variable propositionnelle (atome) ou la négation d'une variable propositionnelle. [13]

2.2. Complémentaire

- Si A est un atome alors {A, ¬A} est une paire de littéraux complémentaires.
- Si A est une formule logique alors {A, ¬A} est une paire de formules complémentaires.

Où : A est le complément de ¬A et ¬A est le complément de A. [13]

3. Présentation de la méthode des tableaux

Les tables de vérité [12], permettent de décider, à propos de toute proposition, si celle-ci est satisfiable, une tautologie, une contradiction, une antilogie, etc.

Le principe de cette méthode est très simple. Toute formule contient un nombre fini d'atomes et admet donc un nombre fini d'interprétations. En conséquence, on peut déterminer la valeur de vérité de la formule pour toutes ses interprétations. On présente souvent le résultat sous forme d'une table de vérité, appelée aussi tableau matriciel (voir la table 6).

On voit immédiatement que la méthode est très inefficace ; si la formule à analyser contient n atomes distincts, elle admet 2^n interprétations. L'algorithme est donc exponentiel (en temps et espace) en fonction du nombre de propositions intervenant dans la formule.

Exemple

Soit $\varphi = \neg(A \rightarrow B) \wedge (A \rightarrow C)$

A	B	C	A→B	¬(A→B)	A→C	φ
0	0	0	1	0	1	**0**
0	0	1	1	0	1	**0**
0	1	0	1	0	1	**0**
0	1	1	1	0	1	**0**
1	0	0	0	1	0	**0**
1	0	1	0	1	1	**1**
1	1	0	1	0	0	**0**
1	1	1	1	0	1	**0**

Table 6 : *Table de vérité de φ*

On conclue que la formule *φ est satisfaite***, v6 |= φ / v6(A,B,C)= (1,0,1)**

On peut améliorer la méthode des tables de vérité en utilisant diverses simplifications. La plus importante consiste à ne pas attendre la fin de la construction de la table pour tirer une conclusion.

La méthode des tableaux sémantique [14] consiste en la recherche systématique d'un modèle d'une formule φ donnée, ou un anti-modèle. Le fait d'imposer une valeur de vérité à une formule peut déterminer univoquement la valeur des composants de la formule, ou au contraire laisser plusieurs choix possibles. La recherche systématique d'un modèle conduit à la construction progressive d'une structure arborescente particulière, appelée tableau sémantique.

4. Construction du Tableau

Nous utilisons comme exemple explicatif la même formule traitée précédemment :

$$\varphi = \neg(A \rightarrow B) \wedge (A \rightarrow C)$$

On essaie d'appliquer l'idée de la recherche systématique d'un modèle :

- ✓ La formule φ est une conjonction, tout modèle de φ sera un modèle de ses deux composants. Donc un modèle de l'ensemble :

$$\{\neg(A \rightarrow B), (A \rightarrow C)\}$$

- ✓ La négation d'une implication sera vrai si et seulement si l'antécédent est vrai et le conséquent faux (¬(A → B) ≡A∧¬B). L'analyse de φ est donc réduite à celle de l'ensemble :

$$\{A, \neg B, A \rightarrow C\}$$

- ✓ L'implication sera vraie si en falsifiant l'antécédent ou en vérifiant le conséquent (A → C ≡¬A∨C). On obtient deux possibilités (non exclusives). Les modelés de φ sont donc, d'une part, ceux de l'ensemble :

$$\{A, \neg B, \neg A\}$$

Et, d'autre part, ceux de l'ensemble :

$$\{A, \neg B, C\}$$

La décomposition de la formule φ est achevée parce que les deux ensembles ne comportent que des littéraux. En effet, un ensemble de littéraux est consistant si et seulement s'il ne contient pas deux littéraux complémentaires.

On voit que l'ensemble { A, ¬B, ¬A } est inconsistant et que l'ensemble {A,¬B, C} est consistant.

Le modèle de la formule φ est : v(A)=1, v(B)=0, v(C)=1

Ce processus est organisé sous forme d'un arbre, appelé *tableau sémantique*. Les tableaux sémantiques corresponds à la formule φ sont représentés par la figure 2.

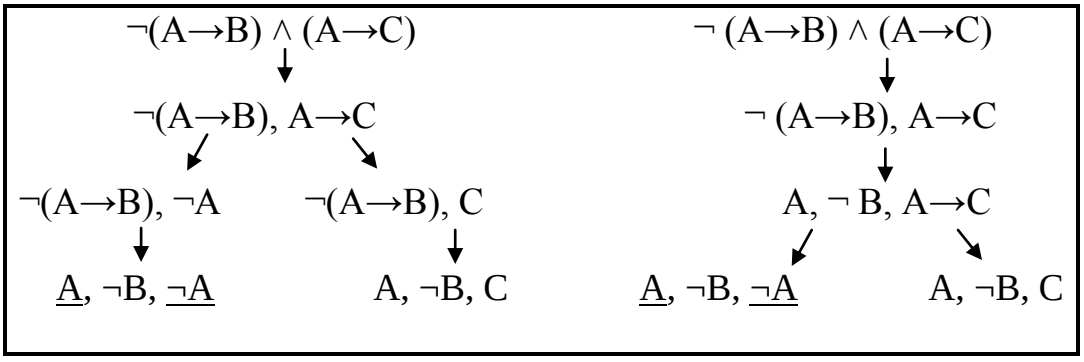


Figure 2 : *Tableaux sémantiques possibles*

D'après l'exemple une formule peut avoir plusieurs tableaux sémantiques différents suivant l'ordre d'application des règles de construction. Mais, tous conduisent à la même conclusion.

5. Règles de décomposition

La preuve par tableau prend la forme d'un arbre dont les nœuds sont étiquetés par un ensemble de formules logiques. Dans le cadre de la logique propositionnelle, cet arbre peut être vu comme une mise sous forme normale disjonctive : les feuilles de l'arbre représentent des conjonctions de sous-formules atomiques à satisfaire, tandis que l'arbre lui-même représente la disjonction de ces conjonctions. Une branche peut être vue comme une suite d'implications, des feuilles vers la racine. [14]

En effet, la construction des tableaux sémantiques est basée sur la partition des formules en trois catégories :

- Les littéraux
- Les formules conjonctives
- Les formules disjonctives

Exemple

- La formule $\neg(X \rightarrow Y)$ est conjonctive car elle est équivalente à $X \wedge \neg Y$
- La formule $X \rightarrow Y$ est disjonctive car elle est équivalente à $\neg X \vee Y$
- La formule $X \leftrightarrow Y$ est conjonctive et remplacée par $(X \rightarrow Y) \wedge (Y \rightarrow X)$
  ou disjonctive et remplacée par $(X \wedge Y) \vee (\neg X \wedge \neg Y)$

La construction est basée sur deux types de règles de décomposition de formules : les règles de prolongation (type-α) concernent les formules conjonctives et les règles de ramification (type-β) pour les formules disjonctives.

5.1. Règle de prolongation

La règle de prolongation (appelée également α-règle) d'une formule propositionnelle FP est équivalente à la conjonction de deux sous-formules α1 et α2 ou à une simplification α. La règle signifie que pour satisfaire une FP il faut satisfaire α1 et α2. La règle est représentée par la table 7. [16]

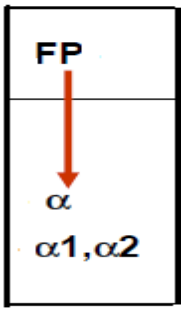


Table 7 : *Forme de α-règles*

Les règles de prolongation pour la logique propositionnelle sont données dans la table 8.

FP	¬¬φ	φ1∧φ2	¬(φ1∨φ2)	¬(φ1→φ2)	φ1↔ φ2
α α1,α2	φ	φ1, φ2	¬φ1, ¬φ2	φ1, ¬φ2	φ1→φ2, φ2→φ1

Table 8 : *Règles de prolongation pour la logique propositionnelle*

5.2.Règle de ramification

La règle de ramification (appelée également β-règle) d'une formule propositionnelle FP est équivalente à la disjonction de deux sous-formules β1 et β2. La règle est notée FP = β1 ∨ β2 et signifie que pour satisfaire FP, il faut satisfaire β1 ou β2. La règle est représentée par la table 9. [16]

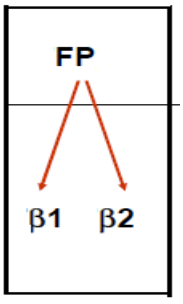


Table 9 : *Forme de β-Règle*

Les règles de ramification pour la logique propositionnelle sont données dans la table 10.

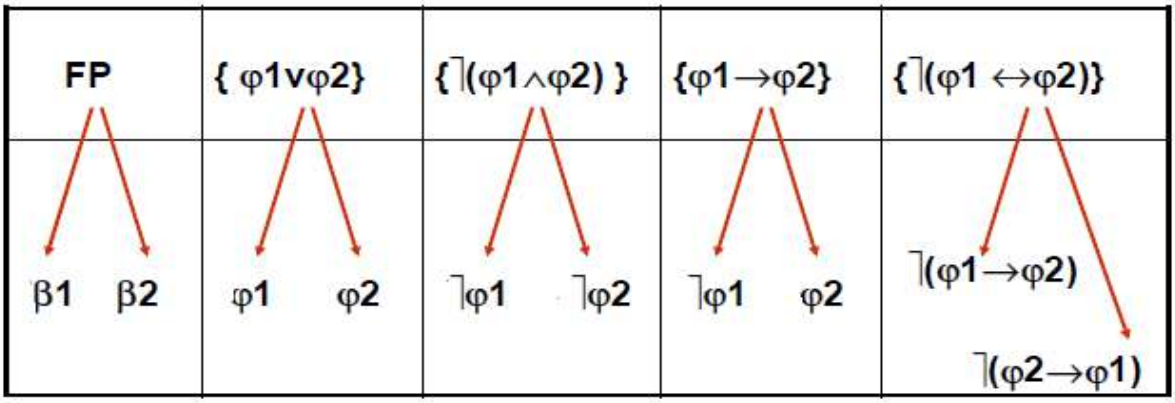


Table 10 : *Règles de ramification pour la logique propositionnelle*

Exemple

Soit la formule propositionnelle suivante :

$$\varphi = \neg ((\neg A \wedge B) \vee (A \vee \neg B))$$

Le tableau sémantique de la formule φ est donné par la figure 3. Cette formule est antilogie, elle n'a aucun modèle.

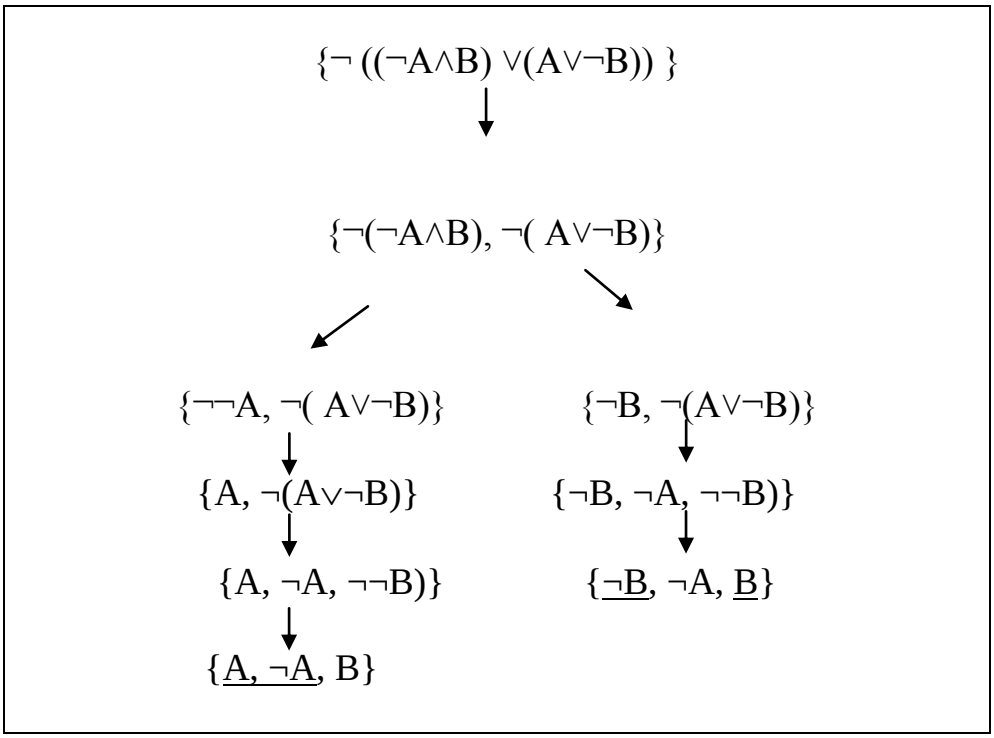


Figure 3 : *Exemple de Tableau sémantique*

6. Algorithme des Tableaux Sémantiques

6.1. Algorithme

- Ci-dessous quelques notations utilisées dans l'algorithme de construction [15]:
  - Un arbre représentant le tableau sémantique d'une formule φ sera noté $\mathbf{T_{\varphi}}$.
  - **l, n, m** représentent des nœuds d'un arbre.
  - **Lab(l)** dénote l'étiquette du nœud l, c'est un ensemble de formules.
  - **Status(l)** est le statut du nœud l, ce statut est *vide* pour tout nœud qui n'est pas une feuille et Status(l) ϵ {ouvert, fermé} si l est une feuille.
  - **Q, R** dénotent des ensembles de nœuds.
- Algorithme :
  - Initialisation : créer un nœud l puis R : = {l} ; Lab(l) : = {φ} ;
  - Itérer tant que $R \neq \emptyset$: choisir $l \in R$:
    - *Si* Lab(l) est un ensemble de littéraux *alors* :

      Si Lab(l) contient des littéraux complémentaires :

      Status (l) : = fermé et R : = R \ {l}.

      Si Lab(l) ne contient pas des littéraux complémentaires :

      Status(l) : = ouvert et R : = R \ {l}.
    - *Si* Lab (l) n'est pas un ensemble de littéraux alors : choisir $\varphi 1 \epsilon$ Lab(l) qui n'est pas un littéral et :

      Si $\varphi 1$ est une α-formule alors créer un fils m pour l et :

      R : = R ∪{m} \ {l} ;

      Lab(m) := (Lab(l)\ $\{\varphi 1\}$) ∪ $\{\alpha 1, \alpha 2\}$

      Si $\varphi 1$ est une β-formule alors créer deux fils, m et n, pour l et :

      R : =R ∪{m, n} \ {l} ;

      Lab(m) := (Lab(l)\ $\{\varphi 1\}$) ∪ $\{\beta 1\}$

      Lab(n) := (Lab(l)\ $\{\varphi 1\}$) ∪ $\{\beta 2\}$
  - Terminaison : la construction est achevée quand toutes les feuilles sont marquées fermées /ouvertes.

6.2.Propriétés algorithmique

L'algorithme des tableaux sémantiques est l'une des procédures de décision pour la satisfiabilité de formules. Le tableau retourne un arbre qui contient quelques feuilles ouvertes si et seulement si la formule traitée est satisfiable. [17]

Cet algorithme est classé dans la catégorie décision puisqu'il vérifie les trois propriétés suivantes :

- Terminaison : la largeur (nombre de règles applicables par nœud) et la profondeur (avec la stratégie de bloc King) de l'arbre de tableaux sont bornées. Donc, l'algorithme termine pour toutes les formules de la logique propositionnelle
- Correction : à partir d'un arbre de tableaux d'une formule satisfiable on peut construire un modèle pour cette formule. Si l'algorithme construit un tableau qui ferme pour une formule alors cette formule est non satisfaisable.
- Complétude : pour toute formule non satisfaisable, l'algorithme construit un tableau fermé.

Pour savoir les démonstrations de ces trois propriétés, nous conseillions de lire les références [15] et [14].

7. Tableaux Sémantiques en Pratique

7.1.Tableau complet

Un tableau sémantique est dit complet si toutes ses feuilles sont fermées ou ouvertes [13].

7.2.Tableau fermé / ouvert

Un tableau complet est dit fermé si toutes ses feuilles sont fermées, sinon (c'est-à-dire le tableau contient des feuilles ouvertes) le tableau est dit ouvert. [13]

7.3.Preuve de la satisfiabilité

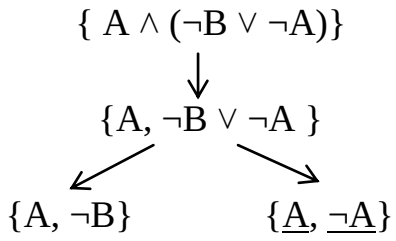
Pour prouver qu'une formule propositionnelle est satisfaite on cherche systématiquement un modèle. Le problème de preuve de satisfiabilité a été réduit à un problème de satisfiabilité d'un ensemble de littéraux. [15]

Donc : ***Si T_φ est ouvert alors φ est consistante.***

Si T_φ est fermé alors φ est inconsistante.

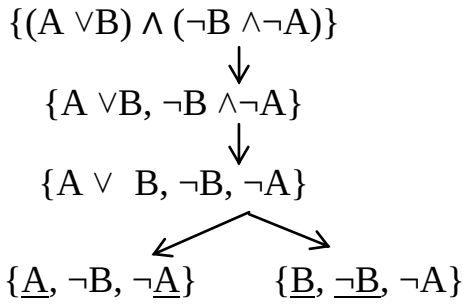
Exemple

1. $\varphi = A \wedge (\neg B \vee \neg A)$



T_φ est ouvert $\Rightarrow$ v(A,B)=(1,0) $\Rightarrow$ A ∧ (¬B ∨¬A) est satisfaite

2. $\varphi = (A \vee B) \wedge (\neg B \wedge \neg A)$



T_φ est fermé $\Rightarrow$ (A ∨B) ∧ (¬B ∧¬A) est non satisfaite.

7.4.Preuve de la validité

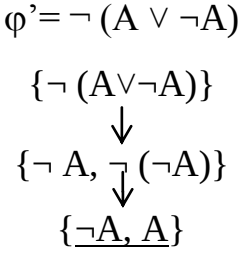
Pour prouver qu'une formule est valide (ou tautologie) revient à prouver l'inconsistance de sa négation. φ valide ⇨ ¬ φ est non satisfaite [16]

Donc : ***Si $T_{\neg\varphi}$ est fermé alors φ est valide.***

Si $T_{\neg\varphi}$ est ouvert alors φ est non valide.

Exemple

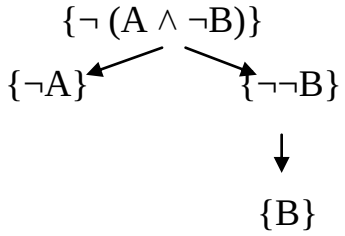
1. $\varphi = A \vee \neg A$



$T_{\varphi'}$ est fermé $\Rightarrow$ φ' est non satisfaite $\Rightarrow$ A ∨ ¬A est valide

2. φ= (A ∧ ¬B)

φ'= ¬ (A ∧ ¬B)



$T_{\varphi'}$ est ouvert ⇒ φ' est satisfaite ⇒ (A ∧ ¬B) est invalide

7.5.Preuve de la Compatibilité d'un ensemble de formules

Prouver que Σ = {φ1, …, φn} est compatible revient à prouver que la formule φ = φ1 ∧ φ2 ∧…. ∧ φn est satisfaite. [16]

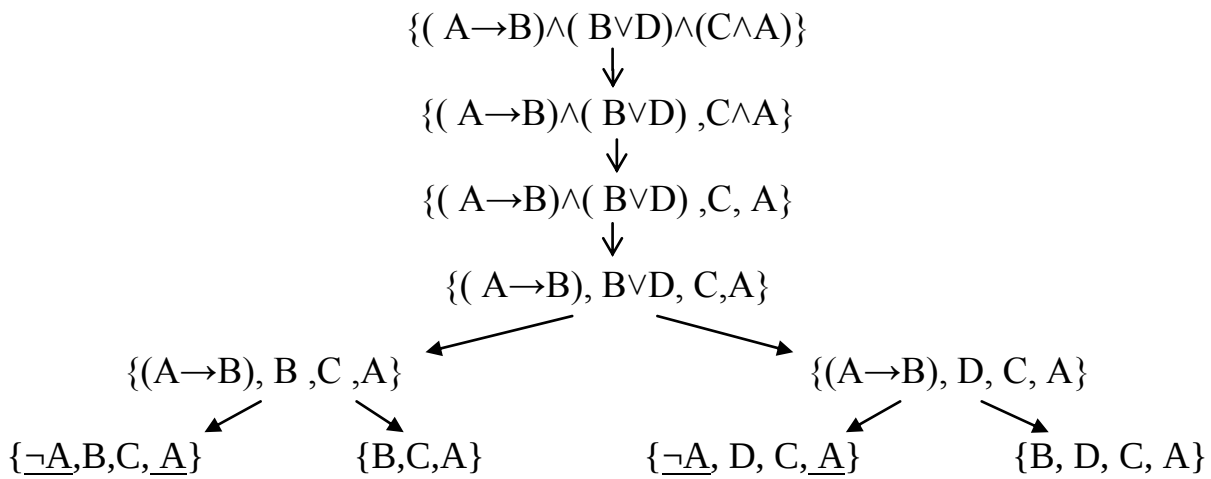
Donc : ***Si T_φ est ouvert alors Σ est compatible.***

Si T_φ est fermé alors Σ est incompatible.

Exemple

Σ ={A→B, B∨D, C∧A} est compatible ?

φ = (A→B)∧(B∨D)∧(C∧A)



T_φ est ouvert ⇒ φ est satisfaite ⇒ Σ est compatible.

7.6.Preuve de la Conséquence

Prouver que φ1, …., φn ⊨ φ est une conséquence logique revient à prouver que la formule FP ≡ φ1∧…∧φn → φ est tautologie. [15,16]

Donc : ***Si $T_{\neg FP}$ est fermé alors la conséquence existe.***

Si $T_{\neg FP}$ est ouvert alors la conséquence n'existe pas.

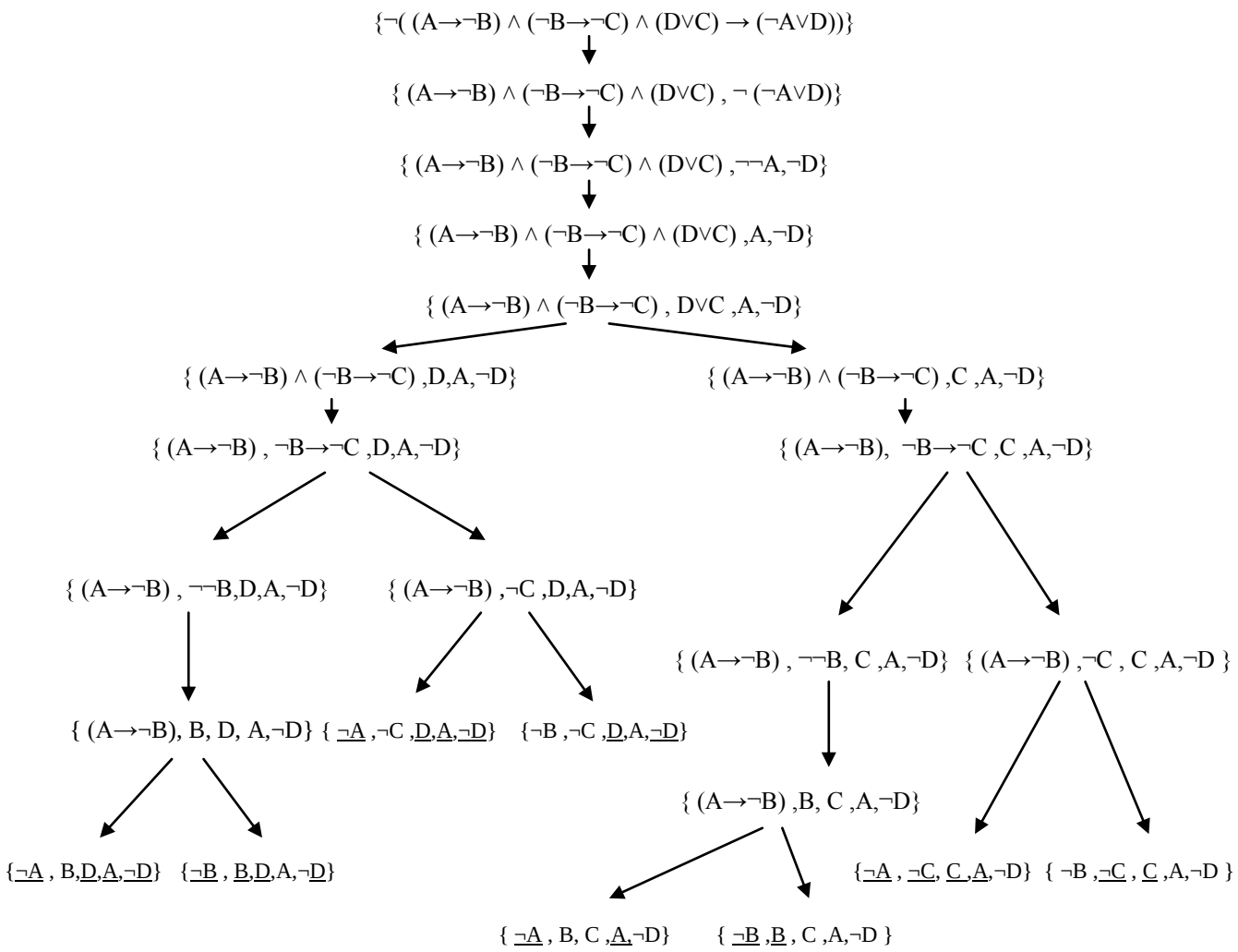
Exemple

A→¬B, ¬B→¬C, D∨C ⊨ ¬A∨D ??

FP = (A→¬B) ∧ (¬B→¬C) ∧ (D∨C) → (¬A∨D) est valide ?

¬FP =¬((A→¬B) ∧ (¬B→¬C) ∧ (D∨C) → (¬A∨D)) est non satisfaite ?

La conséquence logique existe ⇨FP valide ⇨ ¬FP non satisfaite



$T_{\neg FP}$ est fermé ⇒ FP est tautologie ⇒ A→¬B, ¬B→¬C, D∨C ⊨ ¬A∨D

8. Conclusion

Ce présent chapitre introduit la méthode la plus facile de preuve les tableaux sémantiques. Cette méthode consiste à réduire la complexité, économiser l'espace mémoire et diminuer le temps de réponse. Nous avons expliqué ses propriétés ainsi que les algorithmes utilisés pour prouver et raisonner sur les formules propositionnelles. Le prochain chapitre, détaille les tableaux en pratique en montrant quelques exemples expérimentaux traités dans notre outil.

Chapitre 4 : *Réalisation et Implémentation*

1. Introduction

Dans ce chapitre, nous allons tenter d'identifier les objectifs et les exigences de notre mémoire fin d'étude. Au premier lieu, nous identifions et expliquons la plupart du matériel et des logiciels mis à notre disposition pour réaliser notre application. Nous montrons également l'interface utilisateur du système tout en présentant les résultats de notre implémentation pour plusieurs exemples différents.

2. Les objectifs

Notre système est capable de lire les formules bien formées qui sont écrites en logique propositionnelle afin d'extraire un ensemble de propriétés (la satisfiabilité, la validité, la compatibilité, la conséquence logique). Notre outil raisonne sur les formules en dessinant l'arbre sémantique et en donnant tous les modèles disponibles à travers la méthode des tableaux sémantiques.

3. Logiciels et matériels

Au début, nous allons présenter l'environnement de développement et le langage de programmation que nous avons utilisé pour programmer notre application. Ensuite, nous allons décrire le matériel employé pour la réalisation.

3.1. Logiciels

Nous avons utilisé le langage Java à travers le logiciel Netbeans IDE.

3.1.1. Netbeans

Netbeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL (Common Development and Distribution License) et GPLv2. En plus de Java, Netbeans permet la prise en charge native de divers langages tels le C, le C++, le JavaScript, le XML, le Groovy, le PHP et le HTML, ou d'autres (dont Python et Ruby) par l'ajout de greffons. Il offre toutes les facilités d'un IDE moderne (éditeur avec coloration syntaxique, projets multi-langage, refactorisation, éditeur graphique d'interfaces et de pages Web). [18]

3.1.2. Qu'est-ce que Java !

Java est un langage de programmation orienté objet et un environnement d'exécution, développé par Sun Microsystems. Il fut présent officiellement en 1995. Le Java était à la base un

langage pour Internet, pour pouvoir rendre plus dynamiques les pages (tout comme le JavaScript aujourd'hui). Java est aujourd'hui officiellement supporte par Sun, mais certaines entreprises comme IBM font beaucoup pour Java [19].

Le langage Java avait beaucoup d'avantages, on peut citer quelques-unes [20] :

- Portabilité excellente.
- Langage puissant.
- Langage orienté objet.
- Langage de haut niveau.
- JDK très riche.
- Nombreuses librairies.
- Très grande productivité.
- Des applications plus sûres et plus simples.
- Nombreuses implémentations, JVM et compilateurs, libres ou non.
- IDE de très bonne qualité et libre : Eclipse et Netbeans par exemple.
- Supporté par de nombreuses entreprises telles que Sun ou encore IBM et des projets comme Apache.

3.2.Matériel

Le système est implémenté sur Laptop HP avec les spécifications suivantes :

- CPU : intel(R) Core(TM) i3-3110M CPU @ 2.50GHz.
- RAM : 4 GB.
- Système d'exploitation : Windows 10

4. Implémentation

Maintenant, nous allons expliquer les différentes interfaces de notre système à l'aide des exemples pour illustrer les propriétés (satisfiabilité, validité, compatibilité, conséquence logique) d'une formule en présentant les résultats obtenus pour chaque exemple.

Dans cette section, nous essayons de fournir quelques captures d'écran de notre application :

4.1.Interface principale

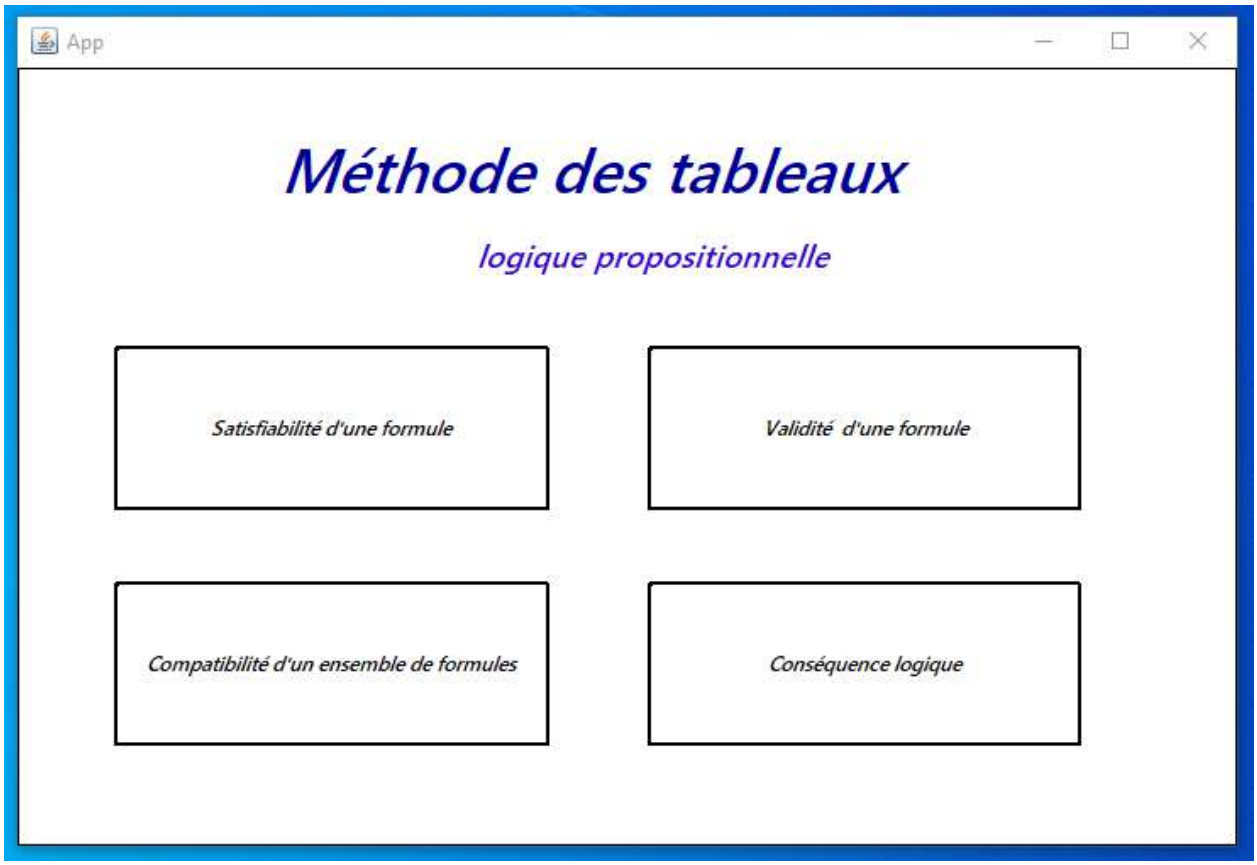


Figure 4 : *Interface principale*

Dans la figure précédente, l'interface principale de l'application est affichée, ce qui vous permet de choisir l'une des propriétés d'une formule pour la prouver par la méthode des tableaux sémantiques.

4.2. Exemples d'utilisation

Nous expliquons notre application à travers l'implémentation de quelques exemples et l'affichage des résultats obtenus.

4.2.1. La satisfiabilité

Afin de prouver la propriété de la satisfiabilité, l'utilisateur doit choisir « Satisfiabilité d'une formule » (voir figure 4), l'interface ci-dessous sera apparaître.

Dans la zone « Formule » l'utilisateur entre une formule propositionnelle, soit :

$$\phi = (A \rightarrow \neg B) \wedge \neg C \rightarrow D \vee (B \leftrightarrow \neg D)$$

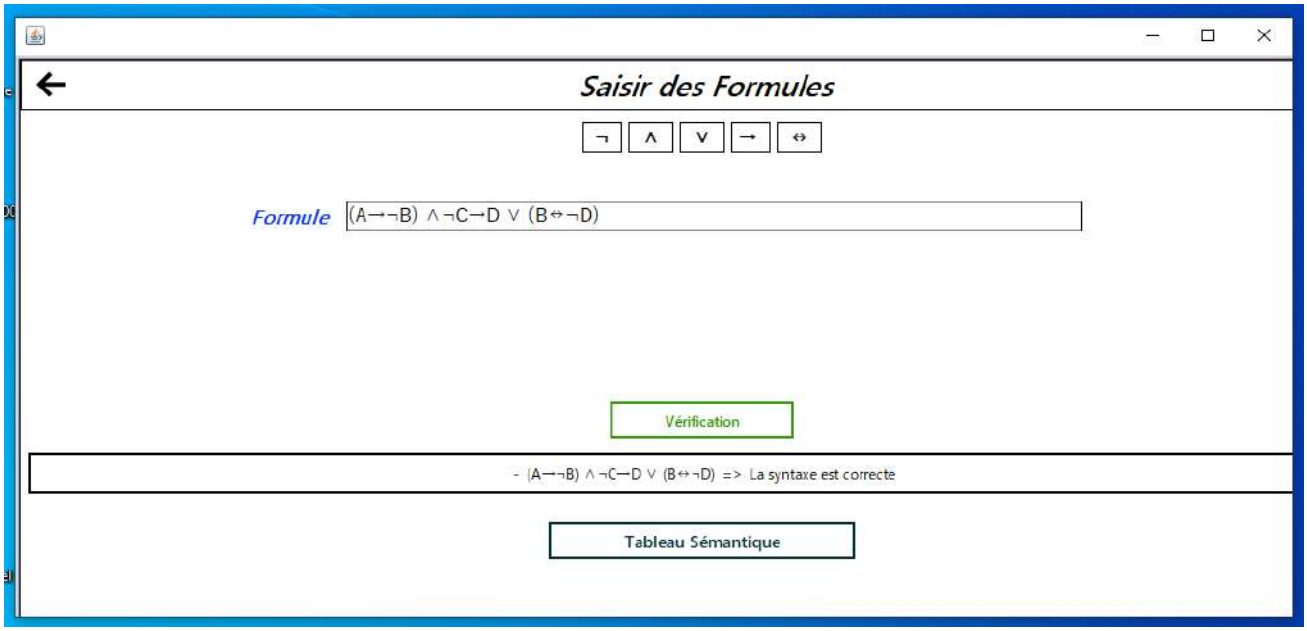


Figure 5 : *Interface de la satisfiabilité*

Si la formule ϕ est bien formée, l'interface de la figure 5 certifie sa correction, et fournit un bouton « tableau Sémantique » pour la traiter.

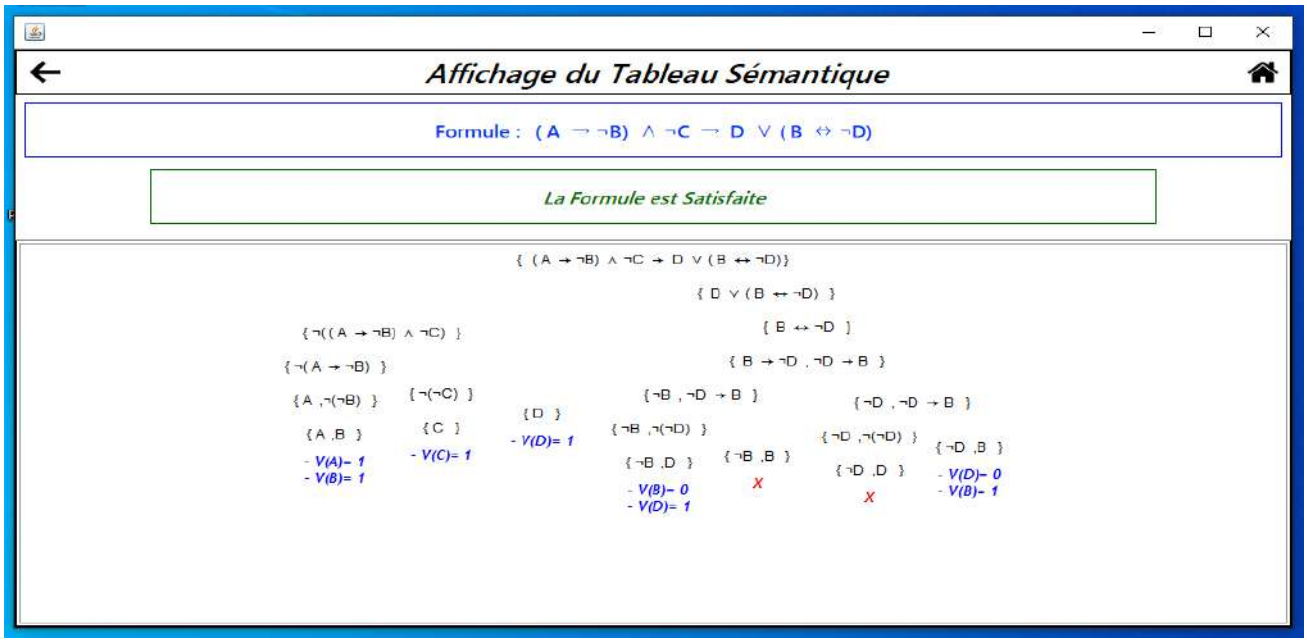


Figure 6 : *Interface de l'arbre de la satisfiabilité*

Notre application traite bien la simplification des formules, en d'autres termes elle tient compte à la priorité entre les connecteurs. Le tableau sémantique de la formule précédente est montré par la figure 6.

La lecture de cet arbre permet de nous connaitre tous les modèles possibles de ϕ. Par exemple la feuille qui se située à droite démontre les valeurs de vérité des deux variables B et D, ce qui permettre d'étudier les possibilités des restes. En effet, la feuille présente quatre modèles :

v1/ v1(A, B, C, D) = (0,**1**,0,**0**).

v2/ v2(A, B, C, D) = (0,**1**,1,**0**).

v3/ v3(A, B, C, D) = (1,**1**,0,**0**).

v4/ v4(A, B, C, D) = (1,**1**,1,**0**).

Dès que l'outil trouve un modèle, il déclare que la formule est satisfaite (voir la définition de la satisfiabilité).

4.2.2. La validité

Nous choisissons la même formule :

$$\phi = (A \rightarrow \neg B) \wedge \neg C \rightarrow D \vee (B \leftrightarrow \neg D)$$

Dans la section précédente, nous avons dit qu'à partir de l'arbre d'une formule, nous pouvons extraire tous les modèles. Donc, il est clair que :

Si on trouve 2 à la puissance le nombre de variables de ϕ modèles, on peut conclure que ϕ est valide. Mais notre but essentiel est d'appliquer la méthode des tableaux sémantiques et non pas de calculer le nombre des modèles.

Pour ϕ, il est très clair qu'elle est non valide, puisque on a au moins une feuille qui comporte deux littéraux complémentaires (voir figure 6). Notre application traite la propriété de la validité comme suit :

- ϕ' : =¬ ϕ
- Algorithme des tableaux sémantiques pour ϕ'
- Si ϕ' non satisfaite ($T_{\phi'}$ est fermé) alors ϕ valide
  Sinon ϕ non valide

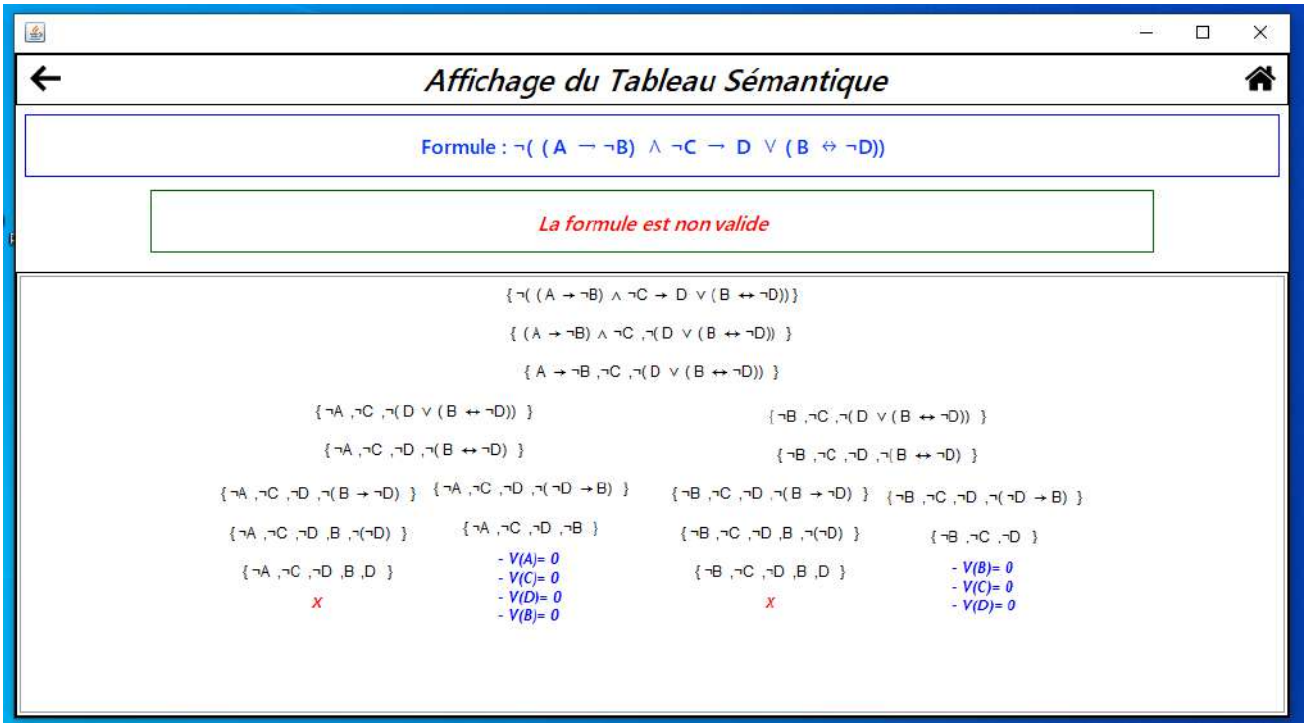


Figure 7 : Interface de traitement de la validité

L'interface de la figure 7 contient un tableau ouvert ce qui conduit à la non validité de la formule ϕ.

4.2.3. La compatibilité

Afin de prouver la compatibilité d'un ensemble de formules, notre application fournit une option d'ajout permettant la saisie de *n* formules (nous ne savons pas le nombre de formules traitées). Pour chaque formule saisie nous vérifions qu'elle est bien formée, si oui l'outil affiche un bouton « Tableaux Sémantiques » (voir la figure 8).

Nous essayons d'étudier le problème :

Peut-on accepter quelqu'un dans une université qu'a énoncé le règlement suivant pour l'inscription des étudiants ???

1. Tout inscrit non algérien a une chambre dans la cité universitaire
2. Tout inscrit à une carte d'identité ou n'a pas une chambre dans la cité universitaire
3. Tout inscrit marié n'a pas de bourse
4. Un inscrit à une bourse si et seulement s'il est algérien
5. Tout inscrit qui a une carte d'identité est algérien et marié.

6. Tout inscrit algérien a une carte d'identité.

La question posée revient à vérifier la compatibilité des conditions d'inscription ?

Formulation des conditions *:*

Variables propositionnelles :

A : x est algérien

C : x a une chambre universitaire

D : x a une carte d'identité

B : x a une bourse

M : x marié

Formules propositionnelles :

$\phi 1 = \neg A \rightarrow C$

$\phi 2 = D \vee \neg C$

$\phi 3 = M \rightarrow \neg B$

$\phi 4 = B \leftrightarrow A$

$\phi 5 = D \rightarrow A \wedge M$

$\phi 6 = A \rightarrow D$

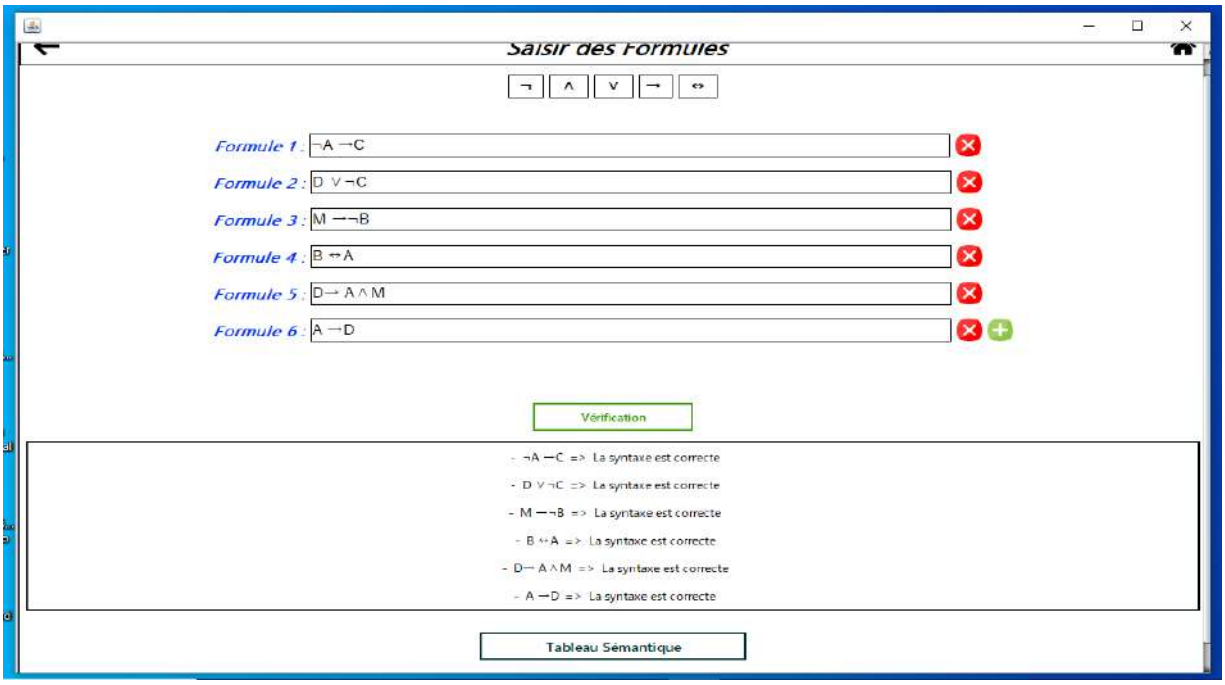


Figure 8 : *Interface de la compatibilité*

Le traitement de cet exemple se fait comme suit :

Si $\Sigma = \{\phi1, \phi2, \phi3, \phi4, \phi5, \phi6\}$ alors $\phi = \phi1 \wedge \phi2 \wedge \phi3 \wedge \phi4 \wedge \phi5 \wedge \phi6$

Donc si ϕ est consistante (T_ϕ est ouvert) alors Σ est compatible

Sinon Σ est incompatible

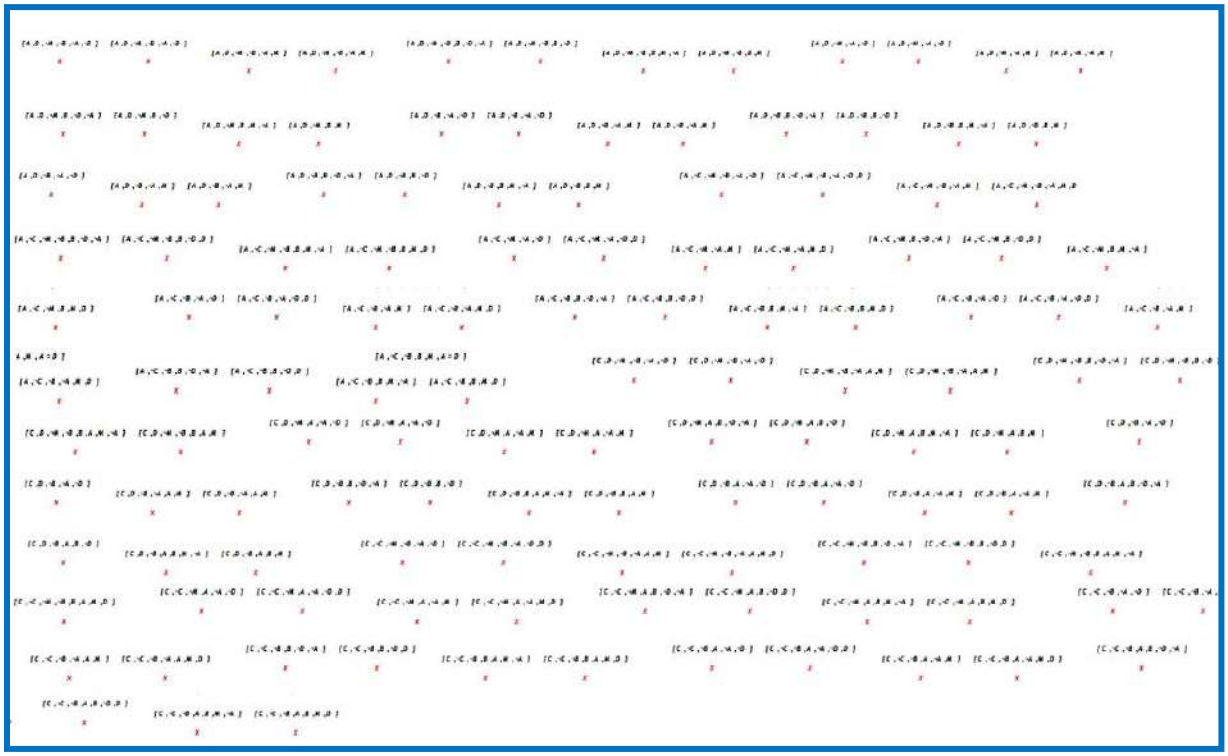


***Figure 9* :** Exemple de la compatibilité dans notre application

La figure 9 montre un tableau fermé donc aucune personne ne peut être inscrite à cette université. Nous savons que la figure est non visible, mais notre but est non pas de la lire mais de voir les nombreuses feuilles résultantes environ 128 feuilles. Le calcule manuellement pour raisonner tels problèmes est trop difficile pour ne pas dire impossible.

4.2.4. Conséquence Logique

Nous essayons de prouver la conséquence suivante :

$A \wedge \neg B, B \leftrightarrow C, \neg A \rightarrow D \models A$

Notre application permet de saisir *n* formules dans la partie gauche et seulement une dans la partie droite (voir la figure 10). Après la vérification syntaxique un bouton est apparait « tableau Sémantique » qui alloue l'étude de la conséquence.

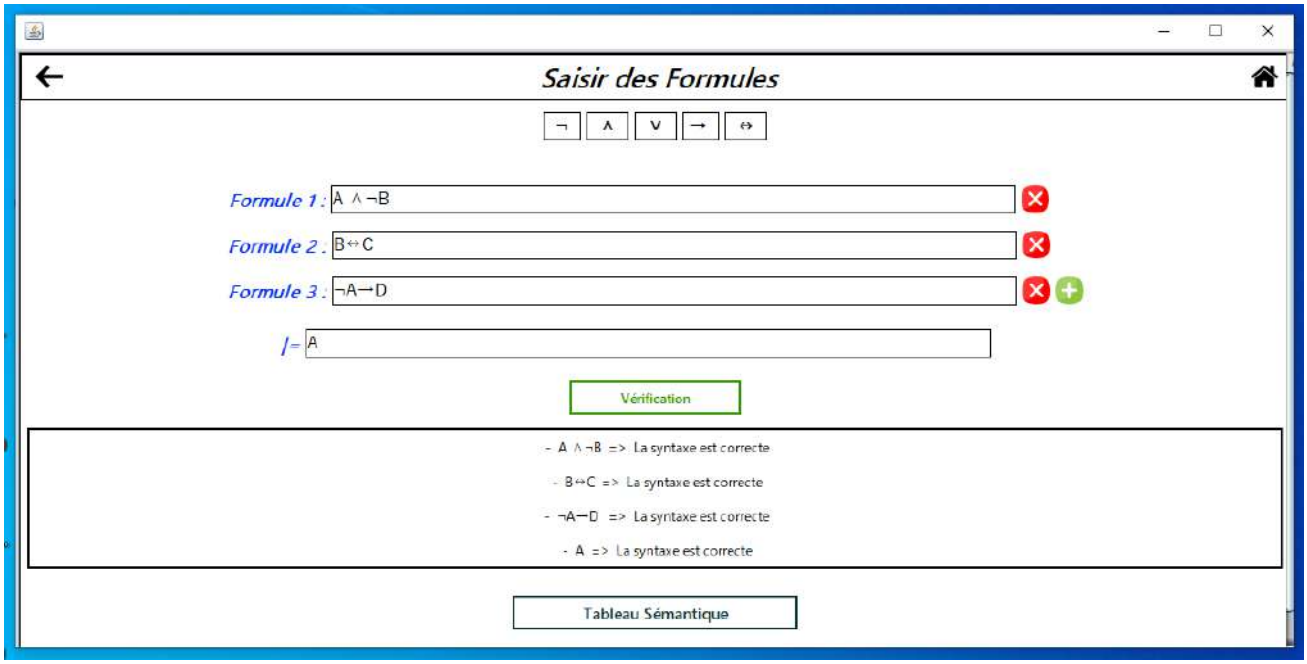


Figure 10 : Interface conséquence logique

L'étude de la conséquence revient à étudier la validité de l'implication (voir chapitre 3). Donc :

- $\phi' := \phi 1 \wedge \ldots \wedge \phi n \rightarrow \phi$
- Algorithme des tableaux sémantiques pour $\neg\phi'$
- Si $\neg\phi'$ est non satisfaite ($T_{\neg\phi'}$ est fermé) alors la conséquence existe

  Sinon non conséquence logique

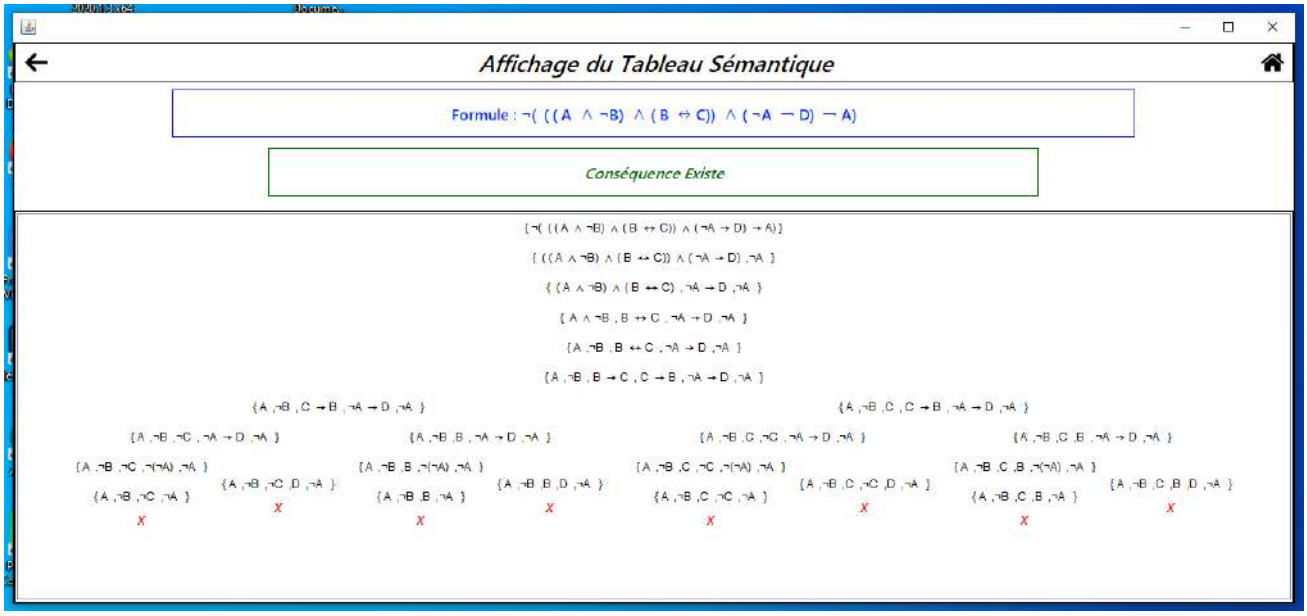


Figure 11 : Traitement de la conséquence logique

La figure précédente 11 affiche l'interface du tableau sémantique obtienne de la conséquence A ∧¬B, B↔C, ¬A→D ⊨ A. Le tableau est fermé cela permet de prouver la conséquence.

5. Conclusion

Dans ce chapitre, nous avons défini les objectifs et les exigences de notre travail, identifié et expliqué la plupart des matériels et logiciels dont nous disposons pour implémenter notre application. Également, nous avons expliqué les interfaces utilisateur du système tout en présentant les résultats de notre implémentation pour plusieurs exemples différents

Conclusion Générale

Conclusion Générale

Dans ce mémoire, nous nous somme intéressées au domaine logique plus particulièrement la logique propositionnelle. En d'autres termes, nous avons étudié l'une des systèmes de preuve, la méthode des tableaux sémantiques qui consiste en la recherche systématique d'un modèle d'une formule donnée, ou évidemment un anti-modèle. Cette recherche crée la simplicité, l'efficacité, et même l'exactitude.

Face aux difficultés rencontrées dans l'application pratique de la méthode des tableaux, nous avons proposé de l'automatiser à l'aide de l'outil informatique. Notre proposition vise à simplifier les tâches de raisonnement logique. Donc, nous avons implémenté un outil permet de résoudre des problèmes logiques complexes dans les plus brefs délais d'une manière simple et très facile. Notre outil présenté gagne beaucoup de puissance du fait qu'il repose sur des concepts formels de la logique propositionnelle et son système de preuve.

A travers notre outil nous essayons de traiter toutes les propriétés des formules propositionnelles tels que la satisfaction, la validité, la compatibilité et la conséquence quelque soit entre deux formules ou avec un ensemble de formules. Nous avons montré tous ce que nous avons dits dans le dernier chapitre avec quelques exemples complexes et difficiles à étudier manuellement.

Notre travail développé constitue une étape qui pourrait contribuer au bénéfice des étudiants, professeurs et chercheurs spécialisés dans les domaines mathématique et informatique en réduisant les efforts et le temps qu'ils y consacrent. Vraiment, nous aimons travailler sur ce sujet.

Malgré les efforts que nous avons déployés pour accomplir cet humble travail et malgré les perceptions que nous avons acquises, nous pensons que notre contribution n'est que le début d'un long chemin. Le travail que nous avons effectué pourrait être amélioré, complété et suivi de plusieurs façons, notamment :

- Utiliser l'algorithme de manière optimale, en d'autres termes arrêter le processus de calcul dès que on trouve un modèle (le cas de la satisfiabilité) ou si on trouve une feuille fermée qui comporte deux littéraux complémentaires (le cas de la validité).
- Fournir une table de vérité complète contenant toutes les interprétations acquises par la méthode.
- Utiliser cette méthode dans d'autres types de logique

Références

[1] BENKADDOUR Fatima Zohra, INTRODUCTION A LA LOGIQUE MATHEMATIQUE, Département des sciences exactes E.N.S d'Oran, Algérie

[2] BENFERHAT Salem, Logique propositionnelle, Centre de Recherche en Informatique de Lens, Université d'Artois, France

[3] Frappier Marc, Logique et mathématiques discrètes-MAT115, Université de Sherbrooke, Québec, Canada, 2020.

[4] Chill Ralph , Logique et théorie des ensembles , Laboratoire de Mathématiques et Applications de Metz, Université Paul Verlaine, France, 2007.

[5] Treinen Ralf, Outils Logiques, Université Paris Diderot - Paris 7, France, 2012.

[6] Belardinelli Francesco, Cours 2 : La Logique (Classique) Propositionnelle Syntaxe, Université d'Evry, Paris, France, 2018.

[7] Stéphane Devismes, Pascal Lafourcade, Michel Lévy. Informatique théorique : Logique et démonstration automatique, Introduction à la logique propositionnelle et à la logique du premier ordre. Ellipses, 2012.

[8] Robert Cori and Daniel Lascar. Logique Mathématique Cours et exercices tome 1 Calcul propositionnel, Algèbres de Boole, calcul des prédicats. Axiomes. Masson, 1993

[9] Damien Nouvel, Logique des propositions, université François Rabelais Tours, France.

[10] P. Gibomont , Logiques pour l'intelligence artificielle ,2006-2007

[11] Séverine Fratani, Luigi Santocanale, Logique et Calculabilité, Aix-Marseille Université, France, 2014

[12] Stéphane Devismes, Pascal Lafourcade, Michel Lévy, Logique et démonstration automatique : Une introduction à la logique propositionnelle et à la logique du premier ordre, université Grenoble Alpes, France, 2018.

[13] Lila BELKACEMI, Mémoire de Magister, Formalisation de la logique temporelle dans Coq, Université MAHMED BOUGARA-BOUMERDES, Algérie, 2013.

[14] Pascal Gribomont, Logique , Université de Liège, 2006.

[15] Emmanuel Filiot, Logique pour l'Informatique, Université libre de bruxelles, 2011.

[16] Zeghib Nadia, Logique propositionnelle, université Constantine 2, 2015

[17] Nhan LE THANH, Introduction à la Logique de Description (résumé), Ecole doctorale STIC, Université Nice Sophia Antipolis.

[18] Gauthier Picard, Laurent Vercouter , ”Initiation à la programmation orientée-objet avec le langage Java”, école nationale supérieure des mines.

[19] F. De Leo, ”pour quoi utilise java”. https://info-rital.developpez.com/tutoriel/java/pourquoi/ , 11 décembre 2005.

[20] Rémi Forax. Gilles Roussel, E. Duris, Java et internet - Concepts et programmation (Français) Broché – 29 juillet 2017