# Rapport Détaillé - TP Logic Solver

## Introduction

Le TP Logic Solver est un système sophistiqué développé pour l'analyse et la vérification des formules logiques propositionnelles au format CNF (Conjunctive Normal Form), en utilisant la méthode de résolution par réfutation introduite par J.A. Robinson en 1965. Ce projet s'inscrit dans le cadre du module de logique mathématique avancée à l'École nationale Supérieure d'Informatique (ESI).

Le système se compose de deux composants principaux, conçus pour offrir à la fois puissance de calcul et facilité d'utilisation :

* **logic\_solver.exe** : Un moteur de résolution robuste implémentant l'algorithme de résolution par réfutation de Robinson pour déterminer la satisfiabilité des formules logiques.
* **gui\_solver.exe** : Une interface graphique conviviale permettant aux utilisateurs de créer, tester et gérer des formules logiques de manière intuitive et visuelle.

Ce projet représente l'application pratique des concepts théoriques de logique mathématique, permettant de résoudre des problèmes combinatoires et de vérification complexes grâce à la modélisation en logique propositionnelle.

## Fondements Théoriques

### La Résolution par Réfutation

La méthode de résolution par réfutation est une technique de preuve automatique qui vise à démontrer qu'une formule logique est une tautologie en prouvant que sa négation est insatisfaisable. Cette approche repose sur les principes suivants :

1. **Conversion en FNC** : Toute formule logique peut être convertie en Forme Normale Conjonctive (FNC), qui est une conjonction de disjonctions de littéraux.
2. **Principe de Résolution** : Si nous avons deux clauses (C₁ ∨ L) et (C₂ ∨ ¬L), où L est un littéral et C₁, C₂ sont des disjonctions de littéraux, nous pouvons dériver la clause résolvante (C₁ ∨ C₂).
3. **Réfutation** : Pour prouver qu'une formule F est une tautologie, nous ajoutons ¬F à notre ensemble de clauses et nous appliquons la résolution jusqu'à dériver la clause vide (contradiction), ce qui prouve que F est effectivement une tautologie.

### Forme Normale Conjonctive (FNC)

Une formule est en FNC lorsqu'elle est exprimée comme une conjonction de disjonctions de littéraux. Par exemple : (P ∨ ¬Q ∨ R) ∧ (¬P ∨ S) ∧ (T ∨ ¬U)

Où chaque parenthèse représente une clause (disjonction de littéraux) et l'ensemble est connecté par des opérateurs AND (∧).

### Structure des Fichiers CNF

Les formules logiques sont stockées dans des fichiers au format CNF (.cnf), qui suivent une structure spécifique :

1. **Format de Fichier** : Les fichiers CNF sont des fichiers texte simples avec l'extension .cnf.
2. **Structure des Clauses** :
   * Chaque ligne du fichier représente une clause distincte.
   * Les clauses sont automatiquement interprétées comme étant reliées par des opérateurs AND (∧).
   * Dans chaque clause, les littéraux sont séparés par des espaces.
   * La négation est représentée par le symbole ! devant le littéral.
3. **Exemple de Fichier CNF** :

P !Q R

!P S

T !U

Ce fichier représente la formule : (P ∨ ¬Q ∨ R) ∧ (¬P ∨ S) ∧ (T ∨ ¬U)

1. **Conventions de Nommage** :
   * Les variables sont généralement représentées par des lettres majuscules (P, Q, R, etc.).
   * Il est recommandé d'utiliser des noms de fichier significatifs pour faciliter la gestion des formules.
   * Les noms de fichier peuvent contenir des lettres, des chiffres, des tirets bas et des traits d'union.

## Fonctionnalités Détaillées

### 1. Création de Formules

L'interface de création de formules offre une expérience utilisateur guidée :

* **Nommage Intuitif** : Assignez un nom significatif à votre formule sans ajouter l'extension .cnf.
* **Structuration Progressive** : Définissez d'abord le nombre de clauses, puis construisez chaque clause individuellement.
* **Syntaxe Simplifiée** : Utilisez une notation claire avec des espaces comme séparateurs entre littéraux et ! pour la négation.
* **Vérification Automatique** : Le système valide la syntaxe pendant la saisie pour éviter les erreurs.
* **Sauvegarde Instantanée** : Après la saisie complète, la formule est automatiquement enregistrée au format CNF.

**Exemple de procédure de création** :

1. Entrez le nom de la formule : exemple\_resolution
2. Spécifiez le nombre de clauses : 3
3. Saisissez la clause 1 : P !Q R
4. Saisissez la clause 2 : !P S
5. Saisissez la clause 3 : T !U

Un fichier exemple\_resolution.cnf est créé avec le contenu approprié.

### 2. Test de Formules

Le module de test implémente l'algorithme de résolution par réfutation de Robinson :

* **Sélection Visuelle** : Choisissez une formule à tester parmi celles disponibles.
* **Processus de Résolution** : Le moteur logic\_solver.exe exécute l'algorithme de résolution en arrière-plan.
* **Optimisation du Processus** : Le système emploie des techniques comme la propagation des unités et la détection des tautologies pour améliorer l'efficacité.
* **Résultats Binaires** : Une formule est classée comme SATISFIABLE (il existe une interprétation qui la rend vraie) ou UNSATISFIABLE (aucune interprétation ne peut la rendre vraie).
* **Notification Immédiate** : Les résultats sont affichés dans une fenêtre popup claire et concise.

**Étapes de l'algorithme de résolution** :

1. Lecture du fichier CNF et construction des structures de données internes.
2. Application de l'algorithme de résolution par réfutation.
3. Si la clause vide est dérivée, la formule est UNSATISFIABLE.
4. Sinon, lorsque toutes les résolutions possibles ont été effectuées sans dériver la clause vide, la formule est SATISFIABLE.

### 3. Visualisation des Formules

Le module de visualisation permet une gestion efficace des formules créées :

* **Catalogue Complet** : Visualisez l'ensemble des formules .cnf disponibles dans le répertoire.
* **Prévisualisation Formatée** : Lorsqu'une formule est sélectionnée, son contenu est affiché avec une notation logique formelle, remplaçant ! par ¬, et ajoutant les symboles ∨ et ∧ appropriés.
* **Test Direct** : Lancez le test de satisfiabilité directement depuis l'interface de visualisation.
* **Organisation Intuitive** : Les formules sont présentées dans une liste claire et ordonnée.

**Exemple de prévisualisation** : Pour le fichier exemple\_resolution.cnf contenant :

P !Q R

!P S

T !U

L'affichage formaté serait :

(P ∨ ¬Q ∨ R) ∧ (¬P ∨ S) ∧ (T ∨ ¬U)

### 4. Fonctionnalités Avancées

Le système inclut des fonctionnalités supplémentaires pour les utilisateurs avancés :

* **Édition Manuelle** : Les fichiers .cnf peuvent être modifiés directement à l'aide d'un éditeur de texte standard.
* **Module de Conversion** : Un module supplémentaire permet d'obtenir la FNC d'une formule logique générale.
* **Optimisations Algorithmiques** : Le moteur de résolution implémente des optimisations comme la propagation unitaire et l'élimination des tautologies.
* **Application à des Problèmes Concrets** : Le système peut être utilisé pour modéliser et résoudre des problèmes réels de vérification et de combinatoire.

## Architecture du Système

### Composant logic\_solver.exe

Le moteur de résolution logique est le cœur du système :

* **Langage d'Implémentation** : Développé en C pour des performances optimales.
* **Algorithme Principal** : Implémentation complète de la résolution par réfutation de Robinson.
* **Structures de Données** : Utilisation de structures optimisées pour représenter les clauses et les littéraux.
* **Fonctionnalités Clés** :
  + Lecture et analyse des fichiers CNF.
  + Construction des structures internes pour les clauses.
  + Application de l'algorithme de résolution avec optimisations.
  + Détection de la satisfiabilité ou de l'insatisfiabilité.

### Composant gui\_solver.exe

L'interface graphique offre une expérience utilisateur complète :

* **Technologie** : Développé avec Windows API pour une intégration native.
* **Interface Modulaire** : Organisation en menus et sous-menus pour une navigation intuitive.
* **Fonctionnalités Interactives** :
  + Création guidée de formules.
  + Visualisation des formules existantes.
  + Test de satisfiabilité avec affichage des résultats.
  + Prévisualisation formatée des formules.

## Utilisation Pratique

### Installation et Configuration

1. **Prérequis** : Aucune installation supplémentaire n'est nécessaire.
2. **Fichiers Requis** :
   * gui\_solver.exe (interface graphique)
   * logic\_solver.exe (moteur de résolution)
3. **Lancement** : Exécutez gui\_solver.exe pour démarrer l'application.

### Flux de Travail Recommandé

1. **Création de Formule** :
   * Commencez par créer une nouvelle formule via l'option correspondante.
   * Entrez un nom significatif pour votre formule.
   * Définissez le nombre de clauses.
   * Saisissez chaque clause en respectant la syntaxe (littéraux séparés par des espaces, ! pour la négation).
2. **Vérification** :
   * Utilisez l'option "Show Available Formulas" pour prévisualiser votre formule.
   * Vérifiez que la notation formatée correspond à votre intention logique.
3. **Test** :
   * Lancez le test de satisfiabilité via l'option "Test Selected Formula".
   * Analysez le résultat (SATISFIABLE ou UNSATISFIABLE).
4. **Itération** :
   * Si nécessaire, modifiez votre formule (soit en créant une nouvelle version, soit en éditant manuellement le fichier .cnf).
   * Répétez le processus de vérification et de test.

## Bonnes Pratiques

* **Organisation** : Adoptez une convention de nommage cohérente pour vos formules.
* **Documentation** : Gardez une trace des formules créées et de leur signification.
* **Vérification Progressive** : Testez individuellement les clauses avant de construire des formules complexes.
* **Sauvegarde** : Conservez des copies de vos formules importantes.

## Applications Pratiques

Le TP Logic Solver peut être appliqué à divers domaines :

1. **Vérification de Circuits** : Modélisation de circuits logiques pour vérifier leur comportement.
2. **Planification Automatique** : Résolution de problèmes de planification en les traduisant en formules logiques.
3. **Vérification de Logiciels** : Vérification formelle de propriétés logicielles.
4. **Problèmes de Satisfaction de Contraintes** : Résolution de problèmes comme le coloration de graphes, le sudoku, etc.

## Exemples d'Utilisation

**Exemple 1 : Vérification d'une Tautologie**

Pour vérifier si (P ∨ ¬P) est une tautologie :

1. Négation : ¬(P ∨ ¬P) équivalent à (¬P ∧ P)
2. Conversion en CNF :
3. !P

P

1. Application de la résolution : dérive la clause vide (contradiction)
2. Conclusion : La formule originale est une tautologie

**Exemple 2 : Problème de Coloriage**

Pour un graphe à 3 sommets formant un triangle, vérifier s'il peut être colorié avec 2 couleurs :

1. Variables : P\_ic (le sommet i a la couleur c)
2. Contraintes :
   * Chaque sommet a au moins une couleur
   * Les sommets adjacents ont des couleurs différentes
3. Formule CNF :

P\_11 P\_12

P\_21 P\_22

P\_31 P\_32

!P\_11 !P\_21

!P\_11 !P\_31

!P\_21 !P\_31

!P\_12 !P\_22

!P\_12 !P\_32

!P\_22 !P\_32

1. Résultat : UNSATISFIABLE (impossible de colorier un triangle avec 2 couleurs)

## Perspectives d'Amélioration

Le TP Logic Solver pourrait être enrichi par les fonctionnalités suivantes :

1. **Interface de Modélisation** : Outil visuel pour créer des problèmes sans manipulation directe des formules CNF.
2. **Traçage de Résolution** : Visualisation pas à pas du processus de résolution.
3. **Bibliothèque de Problèmes** : Collection de problèmes prédéfinis pour l'apprentissage.
4. **Optimisations Avancées** : Implémentation d'heuristiques supplémentaires pour améliorer les performances.
5. **Exportation de Preuves** : Capacité à générer des preuves formelles de la satisfiabilité ou l'insatisfiabilité.

## Crédits

Ce projet a été développé par :

* LASFER Mohammed Djawed
* MOUSLIM Ali
* HECHEHOUCHE Aboubakeur

Dans le cadre du cours de logique mathématique avancée à l'École nationale Supérieure d'Informatique (ESI), année universitaire 2024-2025, sous la supervision des enseignants :

* Meziani Lila
* Khelifati Si Larbi
* Faical Touka
* Charabi Leila

## Conclusion

Le TP Logic Solver représente une application pratique et complète des concepts théoriques de la logique propositionnelle et de la résolution par réfutation. L'intégration d'une interface graphique conviviale avec un moteur de résolution puissant offre un outil pédagogique et pratique pour l'analyse de formules logiques et la résolution de problèmes combinatoires.

Ce projet démontre comment des concepts théoriques abstraits peuvent être transformés en outils logiciels fonctionnels, permettant aux étudiants de mieux comprendre et d'appliquer les principes de la logique mathématique à des problèmes concrets.

Link to the code source: https://github.com/MOUSLIM11/LogicalSatisfiabilitySolver.git