RÉALISATION

4.1 Introduction

L'implémentation est la phase la plus importante après celle de la conception. Cette phase consiste à transformer le modèle conceptuel établi précédemment en des composants logiciels formant SEGM.
Dans ce chapitre, nous allons commencer par la description de l'environnement de travail puis donner une description des différentes interfaces graphiques de l'application développée. Les résultats obtenus à partir de l'implémentation du SEGM sur des exercises de la géométrie dans le plan sont présentés dans la fin de ce chapitre.

4.2 Environnement de développement

L'environnement de travail est constitué par deux parties nommées environnement matériel et environnement logiciel.

4.2.1 Environnement matériel

Le développement de l'environnement matériel est caractérisé par :
1. Système d'exploitation : Windows 10 Professionnel.
2. Processeur : Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz 2.19
3. Mémoire installée (RAM) : 8.00 GB.
4. System type : 64-bit operating system, x64-based processor.

4.2.2 Environnement logiciel

NetBeans 8.2 :
Netbeans est un environnement de développement intégré(EDI), placé en open source par Sun en juin 2000. En plus de Java, NetBeans permet également de supporter différents autres langages, comme

Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, éditeur graphique d'interfaces et de pages Web . . .). Conçu en Java, NetBeans est disponible sous Windows, Linux, Mac OS X etc ou sous une version indépendante des systèmes d'exploitation (requérant une machine virtuelle Java). Un environnement Java Développement Kit (JDK) est requis pour les développements en Java.
NetBeans constitue par ailleurs une plate forme qui permet le développement d'applications spécifiques (bibliothèque Swing (Java)). L'IDE Netbeans s'appuie sur cette plate forme Ainsi L'IDE Netbeans s'enrichit à l'aide de plugins.

4.2.3 Le langage de programmation

JAVA :
C'est un langage de programmation orienté objet, développé par Sun Microsystems en 1995. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris).
Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels. Enfin, ce langage peut être utilisé sur internet pour des petites applications intégrées à la page web (applet) ou encore comme langage serveur (jsp).
Enfin, nous rappelons que le Java, étant un langage de programmation orienté objet utilisable sur divers systèmes d'exploitation, est un langage assez robuste, portable et à hautes performances.

4.3 Présentation des interfaces graphiques de l'application

SEGM compose de trois interfaces graphique qui sont :

4.3.1 Interface principale

Au lancement de l'application, l'interface principale du SEGM se présentera (Figure4.1) tel que :

(1) Expert : pour accéder à l'interface expert (Figure4.2).
(2) Utilisateur : pour accéder à l'interface utilisateur (Figure4.7).

FIGURE 4.1 – Interface principale

4.3.2 Interface expert

L'interface expert englobée menue des boutons depuis ces boutons l'expert peut faire la gestion du SEGM (Figure4.2) telle que :
1- Home : ce bouton pour retour à la page principale (Figure4.1).

FIGURE 4.2 – Interface expert

2- Bases des connaissances : ce bouton pour ouvrir une interface secondaire à partir de laquelle nous pouvons faire les trois opérations suivantes sur la base de connaissances : créer, supprimer et ouvrir une base (Figure4.3) tel que :

① créer nouvelle base : pour créer une nouvelle base de connaissances qui contient trois fichiers (base de règles, base de prédicats, liste des contradictions)
② pour supprimer une base de connaissances
③ pour séléctionner une base de connaissances parmi les bases éxistants

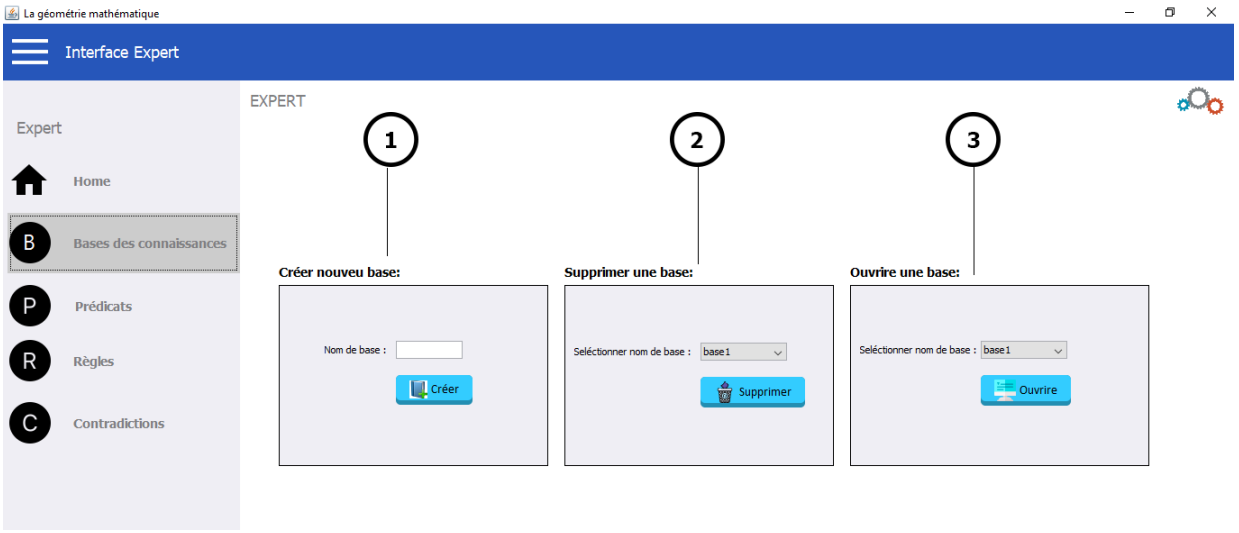

FIGURE 4.3 – Fenêtre base de connaissances

3- Prédicat : ce bouton pour ouvrir une interface secondaire à partir de laquelle nous pouvons faire les deux opérations suivantes : ajouter et supprimer prédicat (Figure4.4) tel que :

① pour ajouter un prédicat à la base de prédicats
② espace pour afficher l'ensemble des prédicats
③ pour supprimer un prédicat de la base sélectionnée

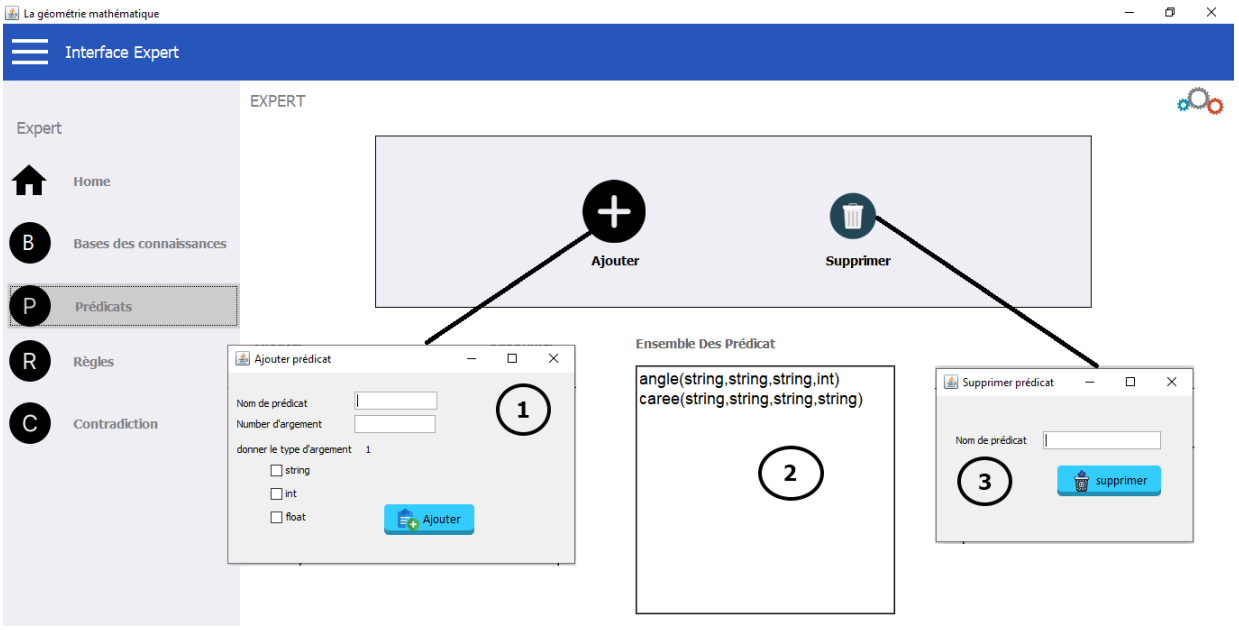

FIGURE 4.4 – Fenêtre prédicat

4- Règles : ce bouton pour ouvrir une interface secondaire à partir de laquelle nous pouvons faire les trois opérations suivantes : ajouter, supprimer et modifier une règle (Figure4.5) tel que :

① pour ajouter une règle à la base sélectionnée
② pour supprimer une règle de la base sélectionnée
③ pour modifier une règle de la base sélectionnée
④ espace pour afficher l'ensemble des règles de la base sélectionnée

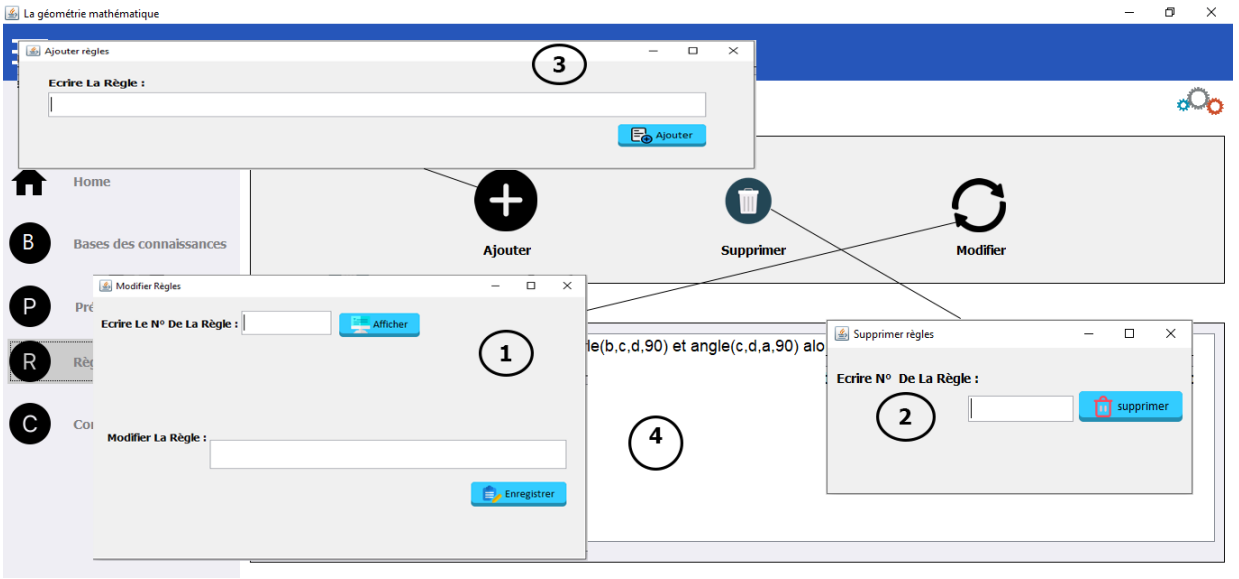

FIGURE 4.5 – Fenêtre règle

5- Contradiction : ce bouton pour ouvrir une interface secondaire à partir de laquelle nous pouvons faire les deux opérations suivantes : ajouter et supprimer une contradiction (Figure4.6) tel que :

① pour ajouter une contradiction à la liste des contradictions de la base sélectionnée
② espace pour afficher l'ensemble des contradictions
③ pour supprimer une contradiction de la base sélectionnée

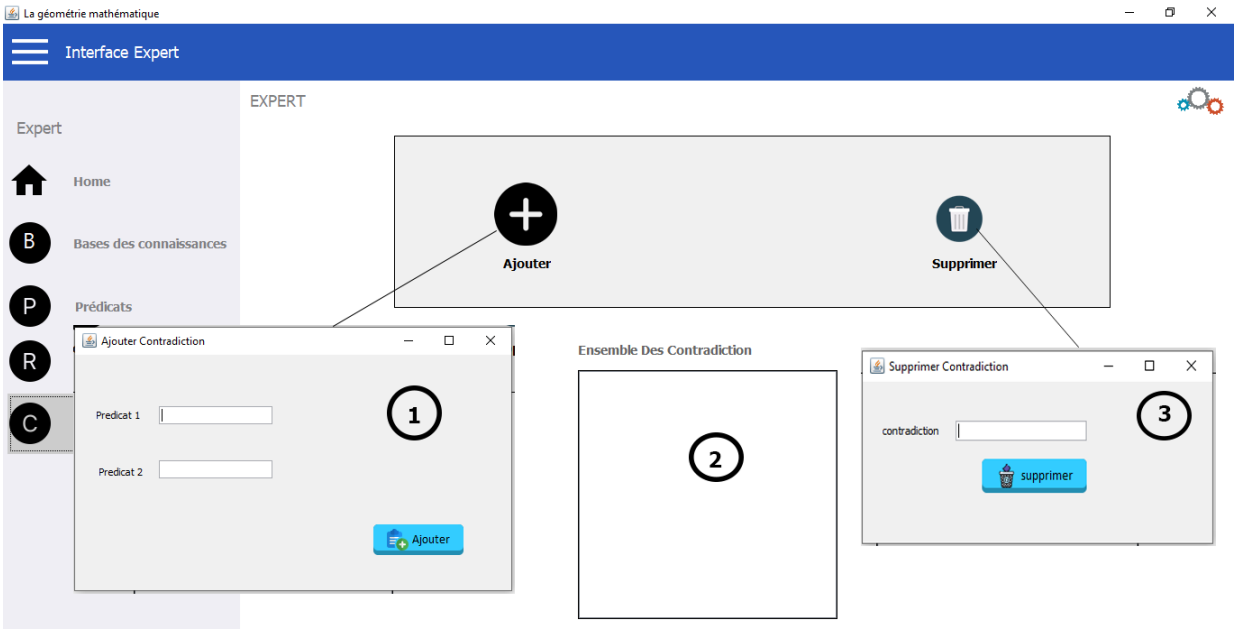

FIGURE 4.6 – Fenêtre contradiction

4.3.3 Interface utilisateur

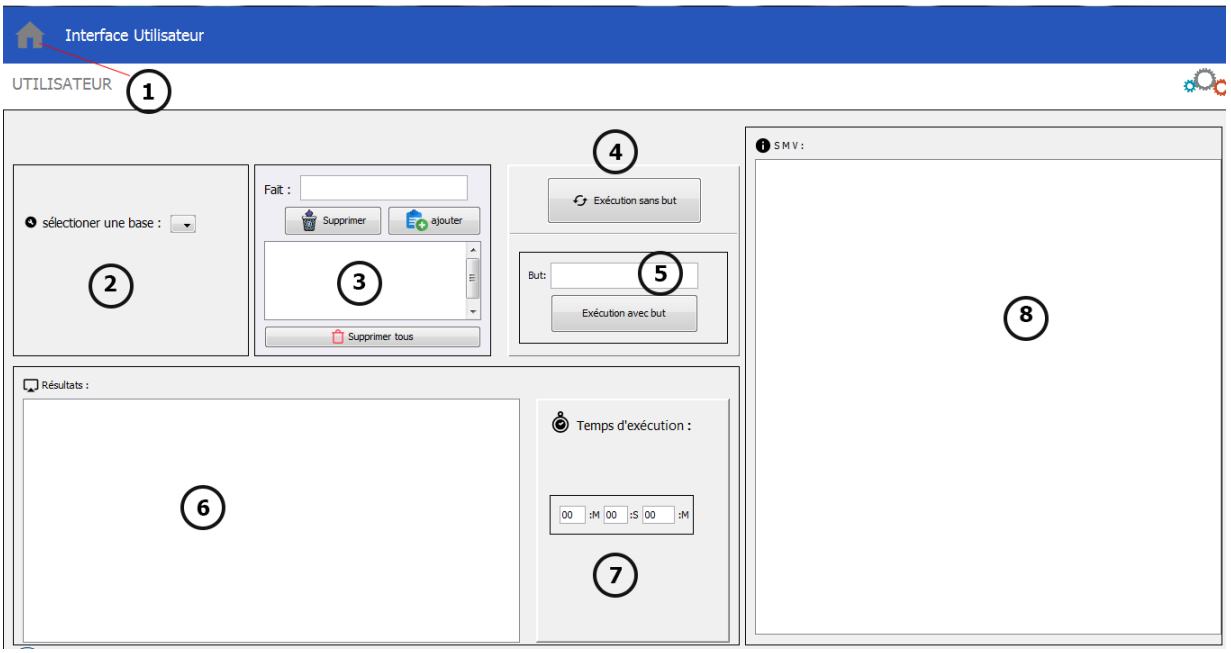

FIGURE 4.7 – Interface utilisateur

(1) Home : ce bouton pour retour à la page principale.
(2) Cette tâche pour sélectionner une base de connaissances.
(3) Dans cette tâche nous pouvons faire les deux opérations suivantes : ajouter et supprimer un fait.
(4) Bouton pour lancer l'exécution (sans but).
(5) Bouton pour lancer l'exécution (avec but).
(6) Espace pour afficher le résultat.
(7) Chronométre pour calculer le temps de réponse.
(8) Pour garder la trace de SMV.

4.4 Exemples de manipulation

Dans cette partie, nous allons présenter les différentes phases de la réalisation de notre projet en mentionnant des imprimes écrans de notre application.

4.4.1 Analyseur syntaxique

Le bouton Ajouter : vérifie que les règles ont été correctement tapées, c'est-à-dire qu'elles réspectent la grammaire proposée dans le chapitre précédent.
Erreur 1 : le nom de prédicat incorrect (Figure4.8).

FIGURE 4.8 – Exemple de détection d'erreur syntaxique

Erreur 2 : la parenthèse de fermeture est absente (Figure4.9).

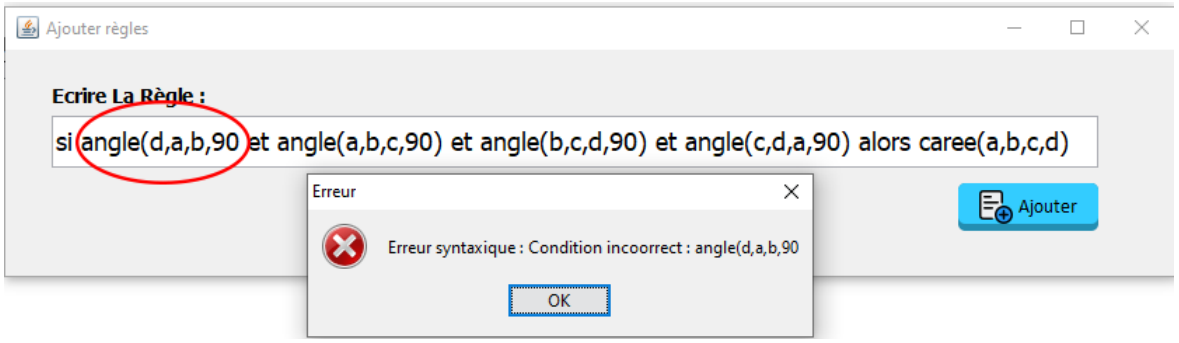

FIGURE 4.9 – Exemple de détection d'erreur syntaxique

Erreur 3 : absence de «et» entre les deux prédicats (Figure4.10).

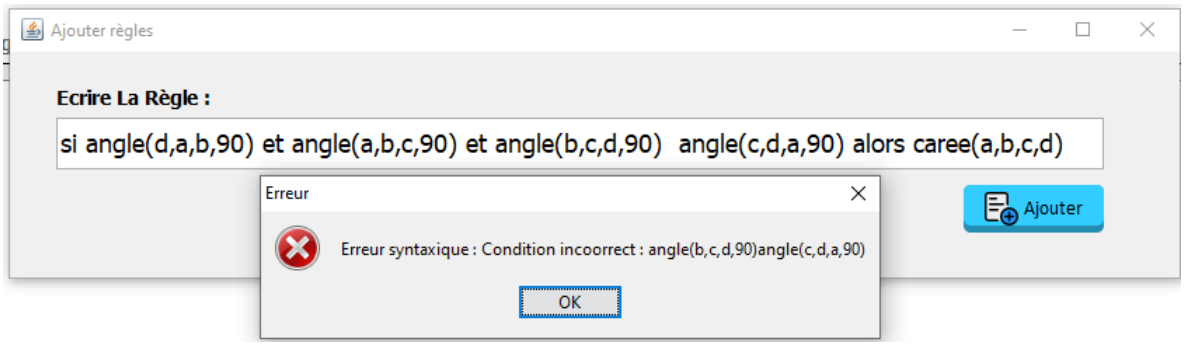

FIGURE 4.10 – Exemple de détection d'erreur syntaxique

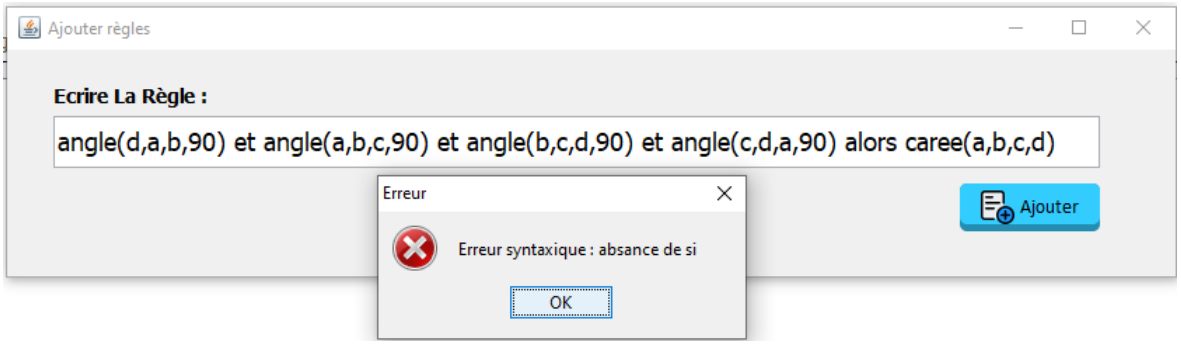

FIGURE 4.11 – Exemple de détection d'erreur syntaxique

Erreur 4 : absence de «si» au début de la règle (Figure4.11).
Erreur 5 : absence de nom de prédicat (Figure4.12).

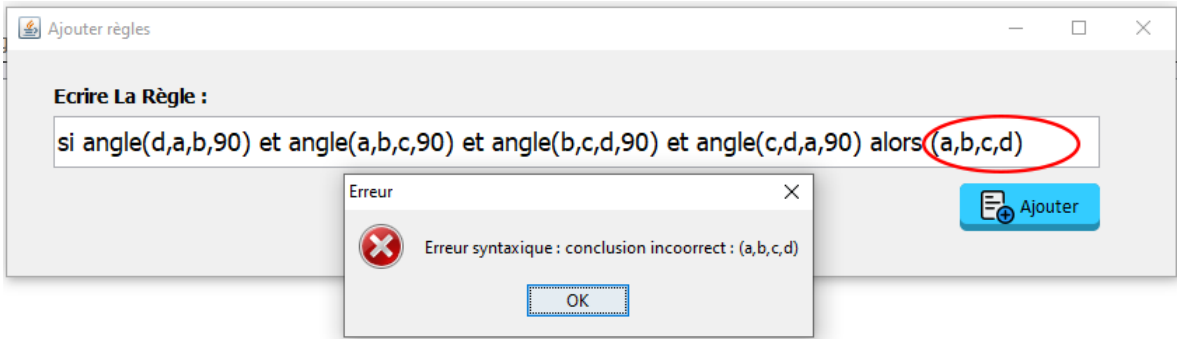

FIGURE 4.12 – Exemple de détection d'erreur syntaxique

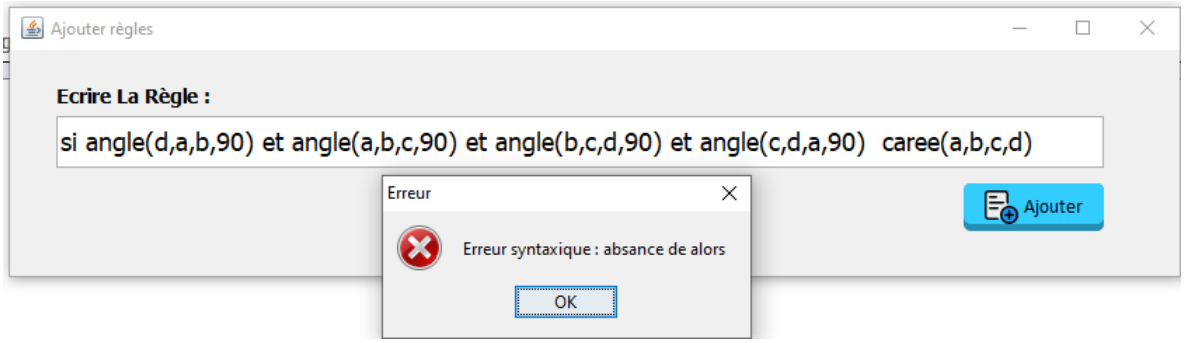

FIGURE 4.13 – Exemple de détection d'erreur syntaxique

Erreur 6 : absence de «alors» (Figure4.13).
Erreur 7 : Le nombre de paramètres est différent (Figure4.14).

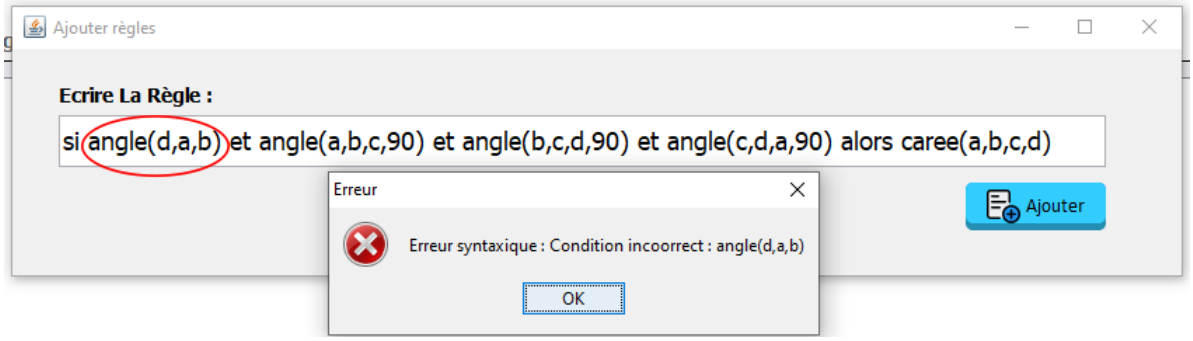

FIGURE 4.14 – Exemple de détection d'erreur syntaxique

4.4.2 Analyseur sémantique

L'analyseur sémantique vérifie les propriétés suivantes : nom, nombre des paramètres et les types des paramètres de chaque fait.

Erreur 1 : pas de prédicat avec cette nom dans le dictionnaire (Figure4.15).

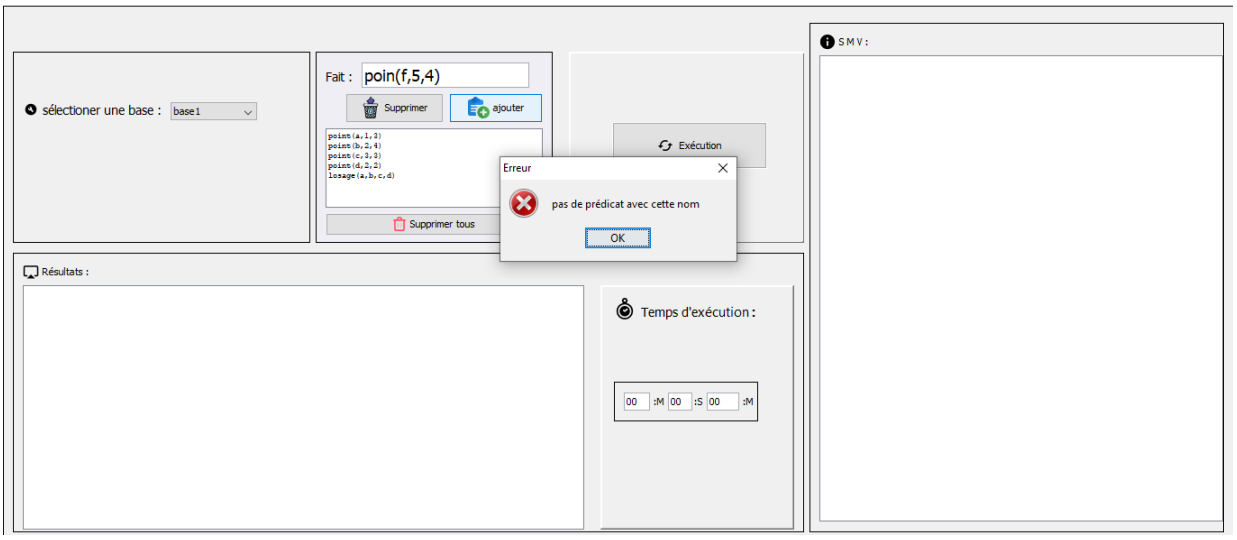

FIGURE 4.15 – Exemple de détection d'erreur sémantique

Erreur 2 : Le nombre d'arguments de prédicat point égale 3 pas 4 point(arg1,arg2,arg3)

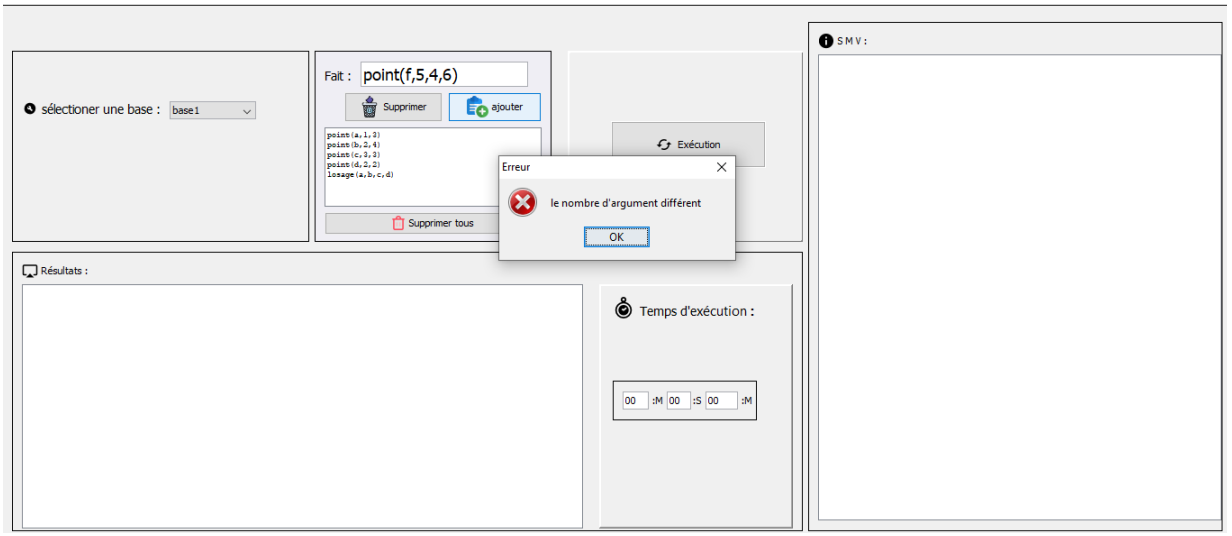

FIGURE 4.16 – Exemple de détection d'erreur sémantique

(Figure4.16).

Erreur 3 : Les types d'argument différents, le type d'argument 3 est float n'est pas string (Figure4.17).

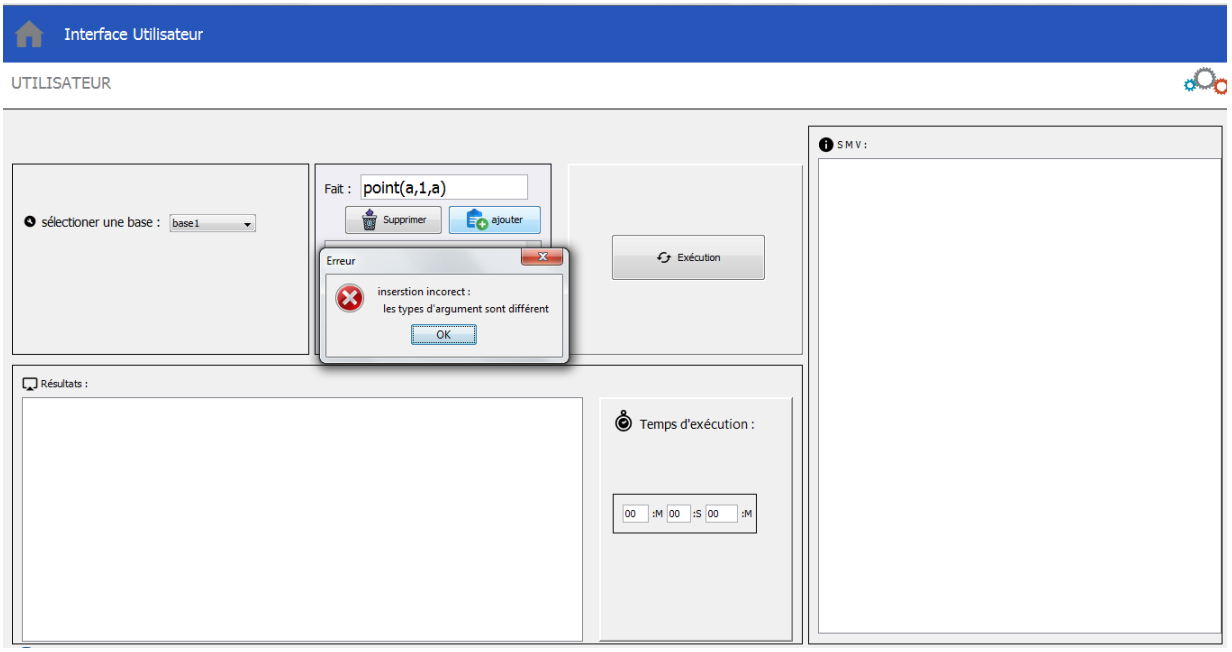

FIGURE 4.17 – Exemple de détection d'erreur sémantique

Liste des suggestions :

Nous utilisons un dictionnaire qui propose des prédicats similaires à celui en cours de saisie.

Exemple :

Lorsque vous tapez un mot comme "t", les prédicat similaires nous apparaissent. Si nous n'avons aucune suggestion, cela signifie qu'il n'existe pas, pour ajouter clique sur un choix parmi les prédicat proposé (Figure4.18).

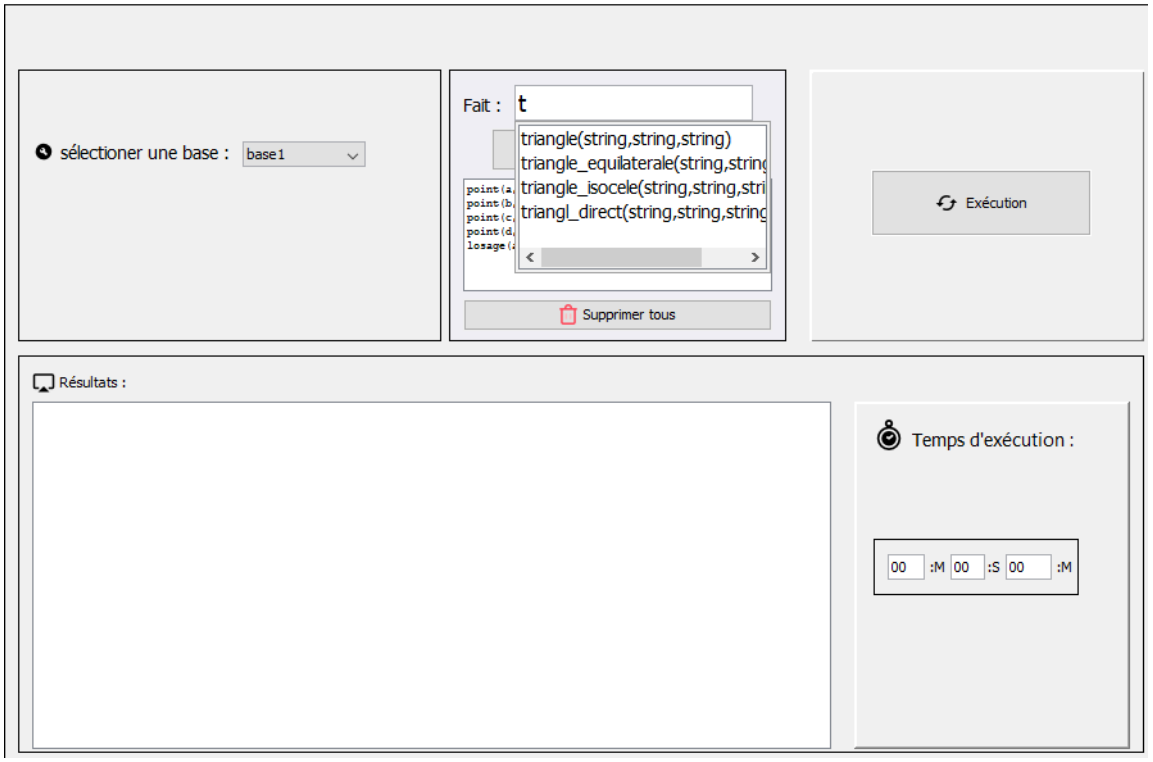

FIGURE 4.18 – Exemple de manipulation de dictionnaire

4.4.3 Contrôle de cohérence

Dans cette partie nous allons vérifier les deux types de cohérence.

4.4.3.1 Contrôles locaux

Dans cette étape nous allons vérifier trois types des règles
type 1 : A et A alors B (Figure4.19).

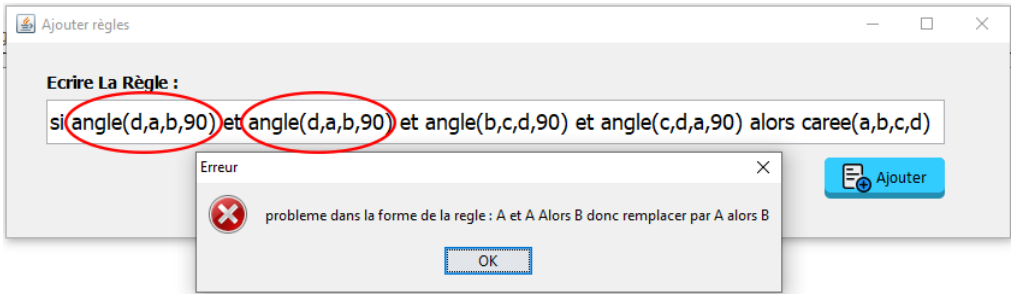

FIGURE 4.19 – Exemple de contrôle local

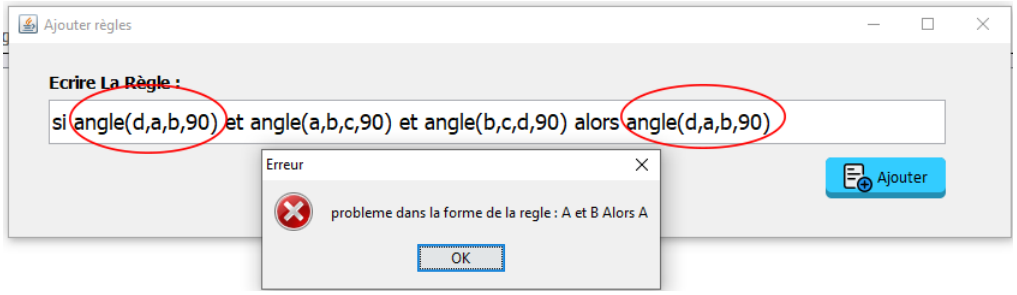

FIGURE 4.20 – Exemple de contrôle local

type 2 : A et B alors A (Figure4.20).
type 3 : A et non A (Figure4.21).

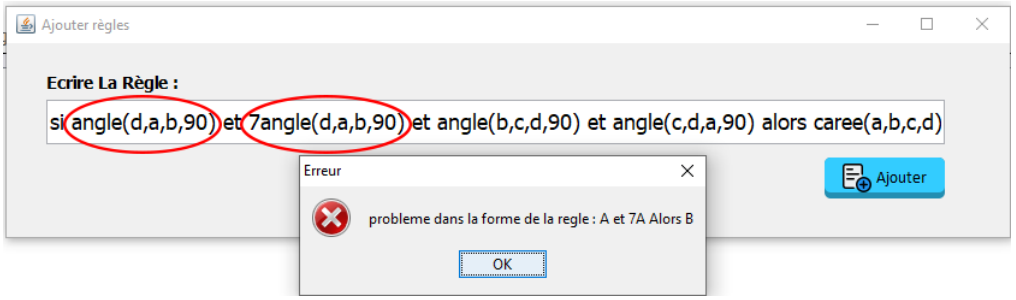

FIGURE 4.21 – Exemple de contrôle local

4.4.3.2 Contrôles globaux

dans cette étape nous allons vérifier trois conflits possibles

1. Contrôle de redondance : pour éviter la répétition des règles dans la base de connaissances (Figure4.22).

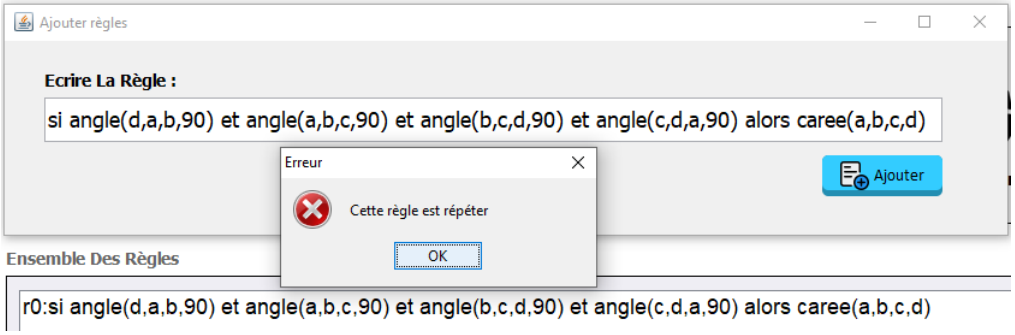

FIGURE 4.22 – Exemple de contrôle de redondance

2. Contrôle d'inclusion :
   - la nouvelle règle inclus dans la règle r0 alors remplacer la règle r0 par nouvelle

règle (Figure4.23).

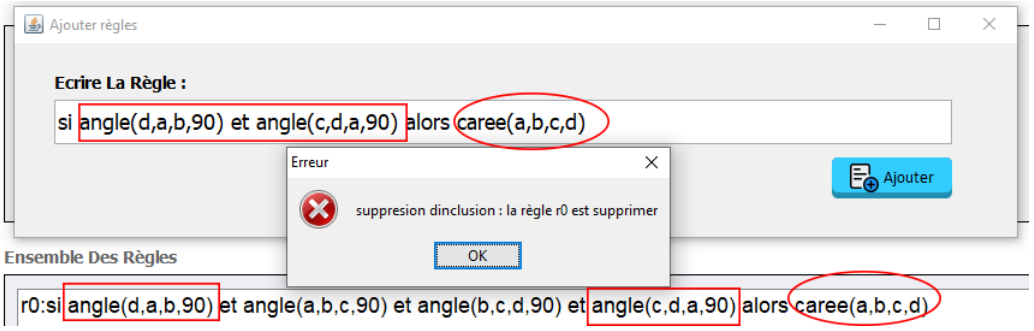

FIGURE 4.23 – Exemple de contrôle d'inclusion

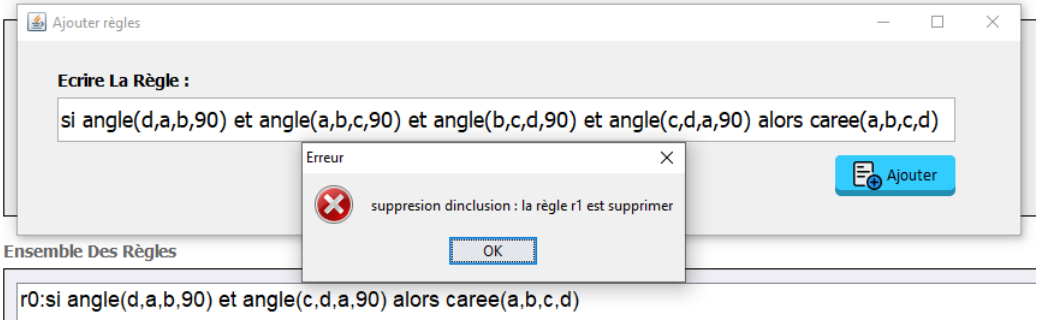

FIGURE 4.24 – Exemple de contrôle d'inclusion

- la règle r0 inclus dans la nouvelle règle alors supprimer la nouvelle règle (Figure4.24).

3. Format des cycles : s'il y a un cycle n'ajoute pas la nouvelle règle (Figure4.25).

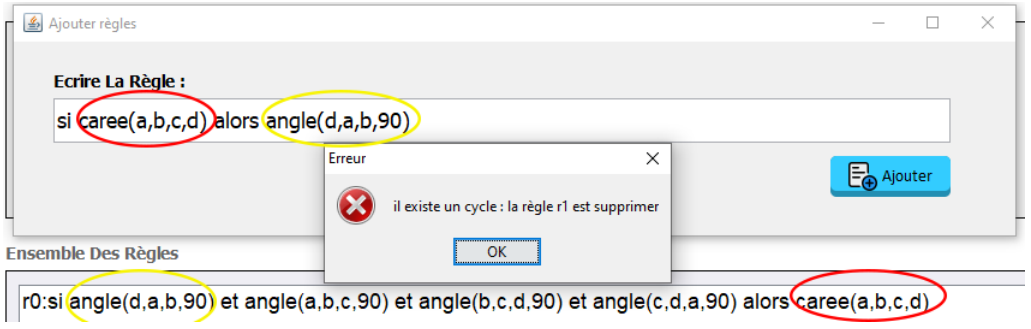

FIGURE 4.25 – Format des cycles

4.4.4 Chaînage Avant

- Chaînage avant sans but : Voici la réponse du SEGM dans un exemple d'exécution (Figure4.26) :

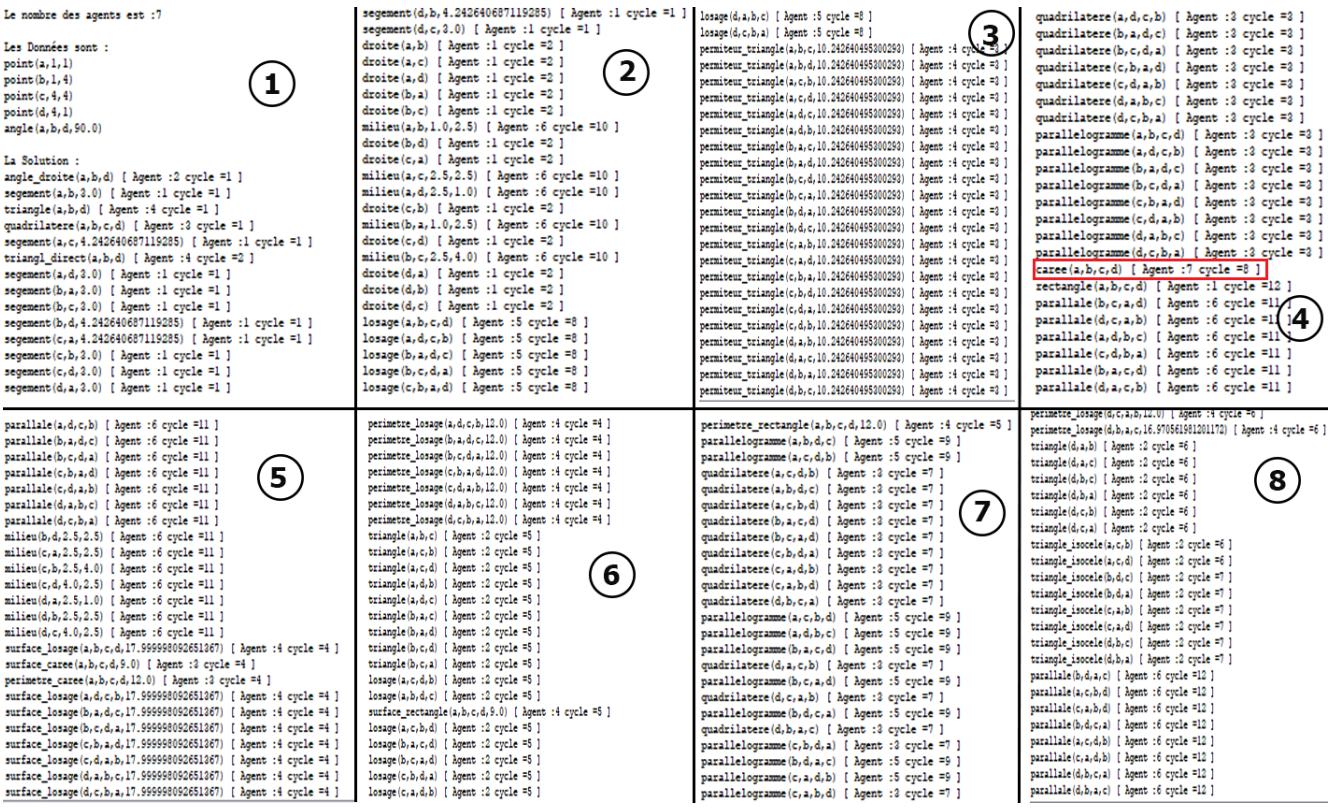

FIGURE 4.26 – Manipulation de chaînage avant sans but

- Chaînage avant avec but : Voici la réponse du SEGM dans un exemple d'exécution (Figure4.27) :

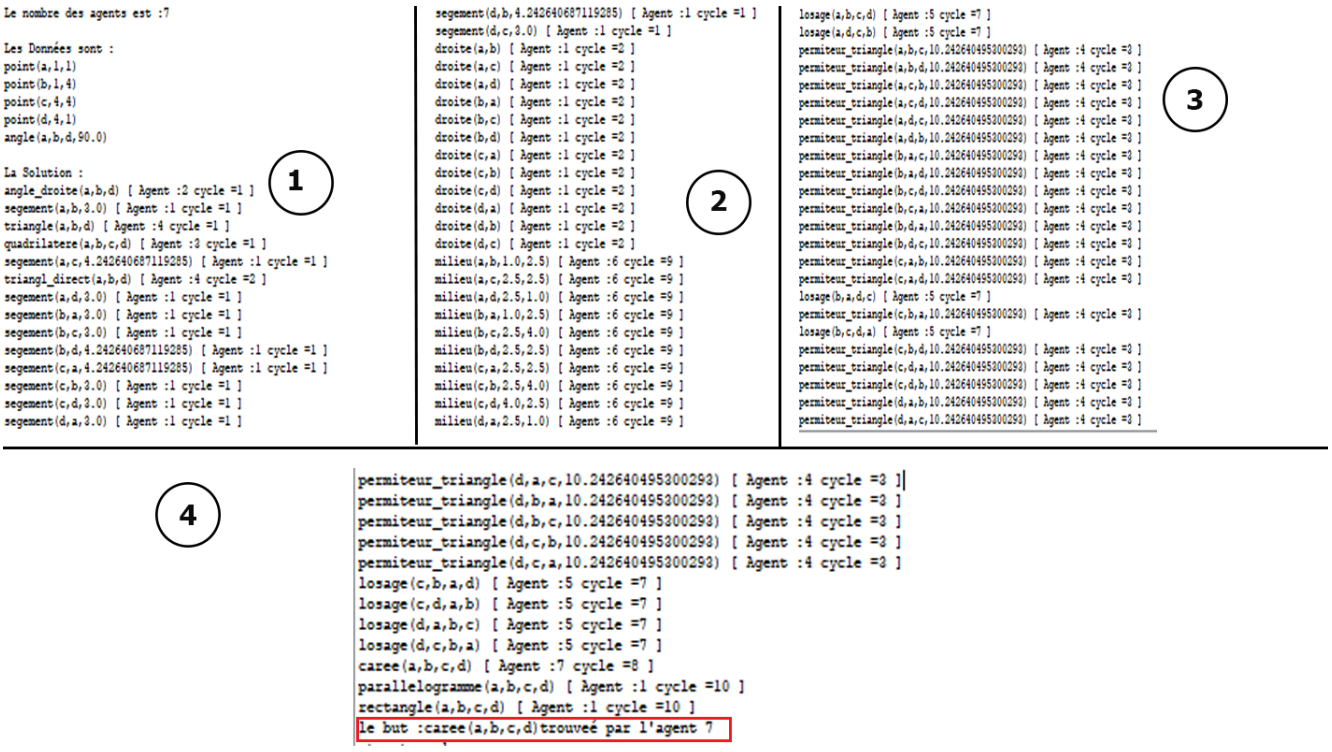

FIGURE 4.27 – Manipulation de chaînage avant avec but

4.4.5 Système de maintien de vérité (SMV)

Dans cette partie nous avons exécuté SEGM avec les données suivantes : angle(a,b,c,90), angle(c,a,b,40), angle(b,c,a,50), triangle-équilatérale(a,b,c), triangle-équilatérale(a,d,c) (Figure 4.28).
Tel qu'il existe une contradiction entre triangle-équilatérale et triangle-rectangle la Figure 4.29 présente la réaction du SEGM dans ce cas.
D'après la Figure 4.29 SEGM détecte la contradiction entre triangle-équilatérale et triangle-rectangle et lance une propagation, le résultat est visualisé dans la Figure 4.30.
Interprétation du résultat de la Figure 4.30 **:**
Dans la base de règles de SEGM on a la règle suivante :
si angle(A,B,C,V) et angle(B,C,A,U) et angle(C,A,B,W) et V==90 alors triangle-rectangle(A,B,C)
Donc après raisonnement avec chaînage avant triangle-rectangle(a,b,c) sera ajouté à la base de connaissances avec un état IN car toutes les hypothèses sont dans l'état IN, mais tant que la contradiction est détectée l'état de triangle-équilatérale(a,b,c) a été changé vers OUT.

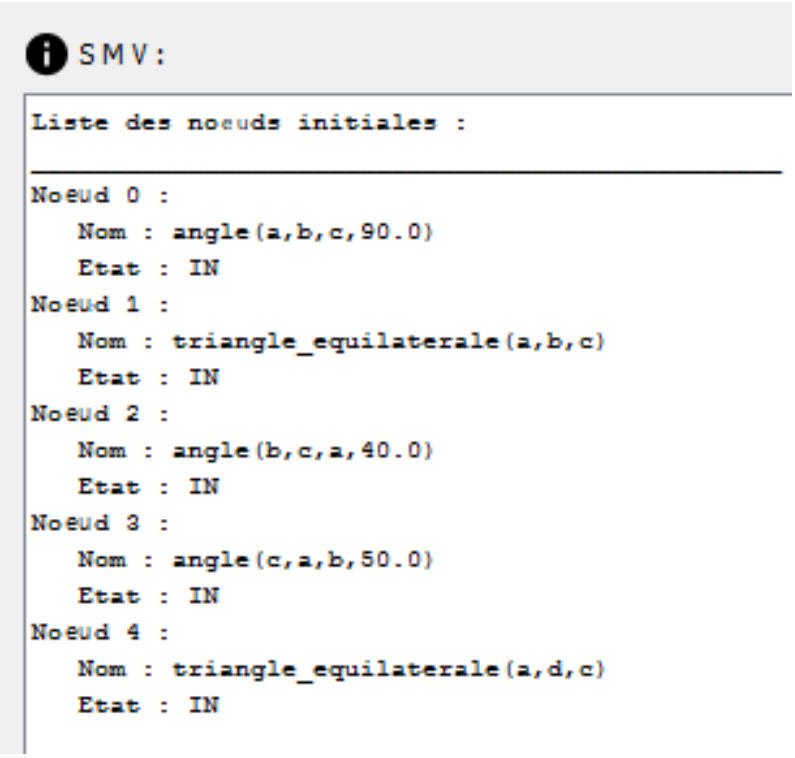

FIGURE 4.28 – Liste des noeuds initiales

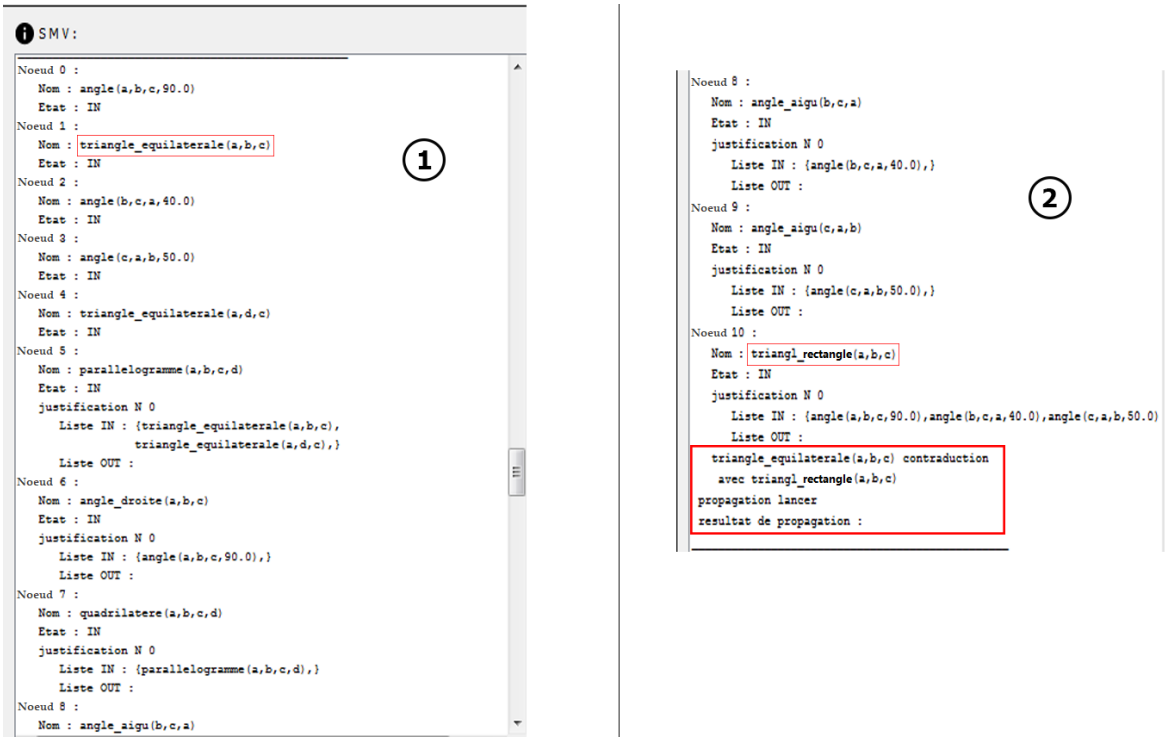

FIGURE 4.29 – Détection de contradiction

FIGURE 4.30 – Résultat de propagation

4.4.6 Raisonnement centralisé VS Raisonnement distribué

Nous avons calculé le temps de réponse du SEGM sur les mêmes données dans les deux cas suivants :
1- Avec un raisonnement centralisé (mono agent) (Figure4.31).
2- En utilisant le raisonnement distribué (SMA plusieurs agents raisonnent sur le problème posé) (Figure4.31).

Le tableau 4.1 représente d'autres exemples de comparaison entre raisonnement centralisé et le raisonnement distribué
La figure 4.32 représente les excercices

Temps d'exécution / Exercices	Mono-agent	Multi-agents
1	0_m :00_s :546_{ms}	0_m :00_s :344_{ms}
2	0_m :20_s :823_{ms}	0_m :05_s :390_{ms}
3	4_m :09_s :659_{ms}	0_m :57_s :930_{ms}
4	5_m :44_s :948_{ms}	2_m :13_s :300_{ms}

TABLE 4.1 – Exemples d'exécution Mono-agent VS Multi-agents

Discussion des résultats :
Les résultats (en termes de temps d'exécution) obtenus selon le raisonnement distribué

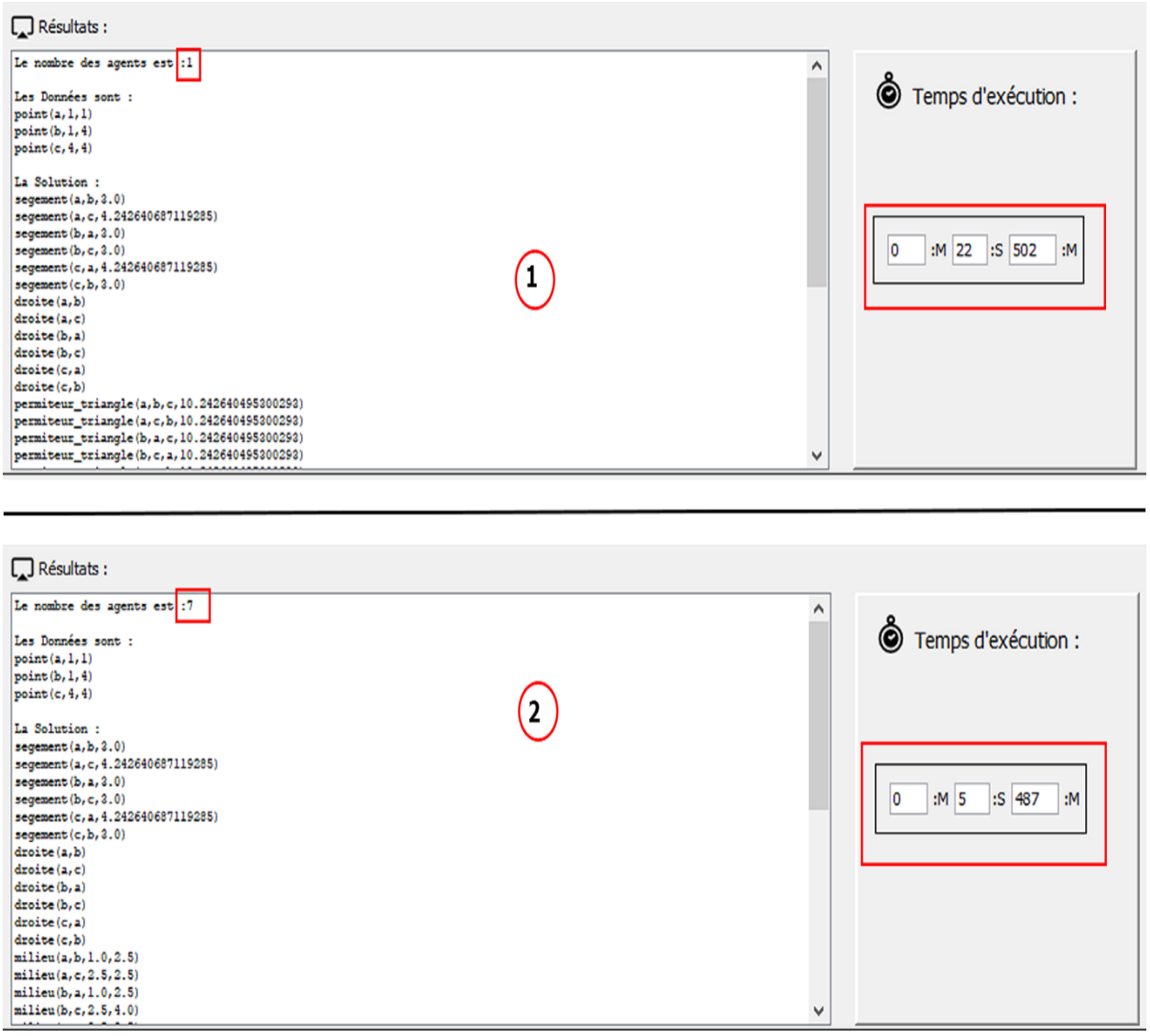

FIGURE 4.31 – Temps de réponse avec raisonnement centralisé et distribué

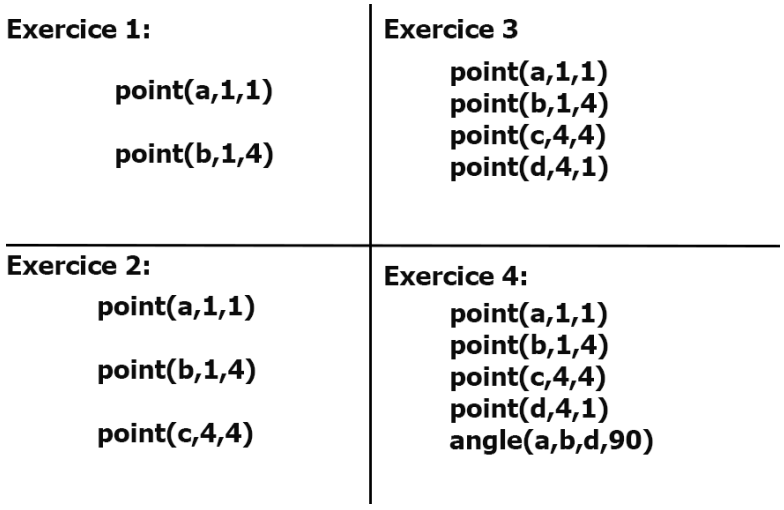

FIGURE 4.32 – Exercices

sont meilleurs par rapport à raisonnement centralisé.
- Pour le premier exercice nous avons remarqué que le temps d'exécution est proche dans les deux cas (raisonnement centralisé, raisonnement distribué) parce que cet exercice ne nécessite pas de grands calculs.
- Pour le dernier exercice nous avons remarqué que le temps d'exécution du SEGM dans le cas multi-agents est grand par rapports au deux exemples précédents parce que CPU de notre PC (Personal computer) atteint le maximaux de sa performance

4.5 Conclusion

La phase de réalisation est une étape très importante dans le cycle de vie de l'application, car à la fin de cette phase nous avons vu le résultat de toutes les étapes précédent. Au cours de ce chapitre, nous avons présenté l'environnement matériel et logiciel ainsi que les différents algorithmes utilisés. Enfin, nous avons testé et présenté les différentes interfaces offertes par notre application(SEGM).