Sorbonne Université

**RAPPORT PROJET LRC DEMONSTRATEUR POUR LA LOGIQUE ALC EN PROLOG**

Décembre 2023





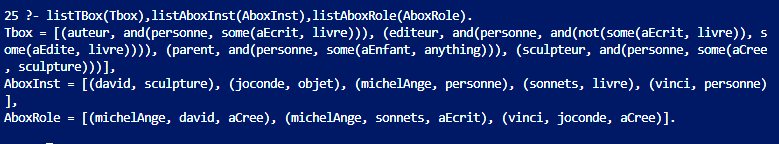
# **Introduction**

Ce projet consiste en la conception d'un démonstrateur en Prolog exploitant l'algorithme des tableaux pour la logique ALC. Structuré en trois parties, chacune revêt une importance particulière pour atteindre les objectifs globaux. La première partie se focalise sur la garantie de la validité syntaxique et sémantique des éléments de la logique ALC. La deuxième partie concerne la saisie des propositions à démontrer, tandis que la troisième et dernière partie implique la construction du démonstrateur. À ce stade, l'algorithme des tableaux sera intégré, permettant au système de générer des preuves et de démontrer la validité des énoncés logiques introduits par l'utilisateur.

# **Implémentation de la première partie**

## **Liste des Concepts, Instances et Rôles**

Nous avons développé des prédicats (***listTBox***, ***listAboxInst***, ***listAboxRole***) pour générer des listes des concepts, instances et rôles de la TBox et de l'ABox respectivement.



## **Vérification Syntaxique et Sémantique**

Pour assurer la validité syntaxique et sémantique, des prédicats tels que ***concept/1***, ***concept/2***, et ***concept/3*** ont été définis pour vérifier la cohérence des concepts, instances et rôles.

Voici un exemple de l’exécution de ***concept*** :

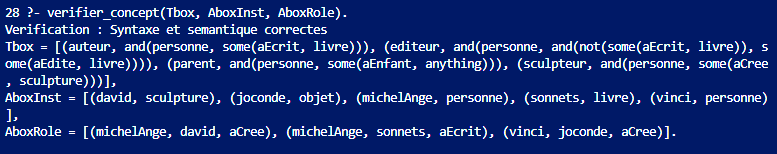


L'expression ***and(auteur, editeur)*** est valide car elle représente un concept qui est la conjonction des concepts "***auteur***" et "***éditeur***" qui sont considérés comme des concepts non atomiques ici.

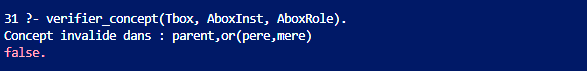
Cependant, l'expression ***some(auteur, editeur)*** n'est pas valide car "auteur" n'est pas un rôle dans le contexte de cet exemple.

La vérification syntaxique et sémantique est réalisée à l'aide du prédicat ***verifier\_concept/3***, qui parcourt les éléments de la TBox et de l'ABox pour s'assurer de leur conformité avec les règles définies. En cas d'incohérence, le système signale les concepts invalides, assurant ainsi une syntaxe et une sémantique correctes.

Voici un exemple de l’exécution de ***verifier\_concept/3***sur la base fournie :



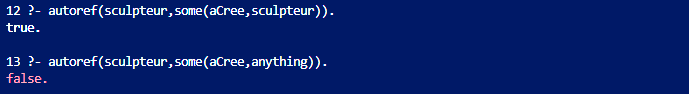
Dans le deuxième exemple, nous avons tenté d'ajouter le concept ***equiv(parent, or(pere, mere)))*** à la T-Box existante. Ce qui a conduit à une erreur, signalant que l'expression ***or(pere, mere)*** n'est pas un concept valide dans le contexte de la T-Box actuelle. En d'autres termes, les concepts "***mere***" et "***pere***" ne sont ni des concepts atomiques ni des concepts non atomiques définis dans la T-Box.



## **Détection de l'Auto-référence**

Le prédicat ***autoref/2***a été élaboré pour traiter divers scénarios d'auto-référence, reconnaissant tant les auto-références directes que les auto-références indirectes. Il identifie l'auto-référence directe lorsqu'un concept apparaît dans sa propre définition (circularité directe) et gère l'auto-référence indirecte en examinant les équivalences entre concepts et en parcourant récursivement les définitions des concepts équivalents pour confirmer l'apparition du même concept. Le prédicat ***pas\_autoref/1***vérifie l'absence d'auto-référence dans la liste des couples (C, E) de la T-Box. Il utilise le prédicat autoref/2 pour chaque couple, assurant ainsi l'absence d'auto-référence dans toute la liste.

Voici quelques exemples de ***autoref/2*** et ***pas\_autoref/1* :**



Dans le premier exemple, la réponse ***true*** indique qu'il y a une auto-référence ici, car ***sculpteur*** est impliqué dans la définition de lui-même via le rôle ***aCree***.

En revanche, dans le deuxième exemple, la réponse ***false*** indique qu'il n'y a pas d'auto-référence lorsque le concept ***sculpteur*** est défini comme créant quelque chose de manière générale.



Cette requête vérifie s'il n'y a pas d'auto-référence dans la liste de T-Box fournie. La réponse ***false*** indique qu'il y a une auto-référence dans cette T-Box, car la définition du concept ***sculpture*** fait référence au concept ***sculpteur***, qui à son tour fait référence à ***sculpture***, créant ainsi une boucle.

Le prédicat ***verifier\_auto\_reference/1*** prend en entrée la T-Box et effectue une vérification complète. S'il ne détecte aucune auto-référence, il affiche un message indiquant que la T-Box n'est pas auto-référente. En revanche, s'il identifie une auto-référence, il signale une erreur et provoque un échec.

Voici un exemple de l’exécution de ***verifier\_auto\_reference/1***:



Dans le cette exemple, nous avons tenté d'ajouter le concept ***equiv(sculpture, and(objet, all(cree\_par, sculpteur)))*** à la T-Box. Cependant, cela a conduit à une détection d'auto-référence, comme indiqué par le message d'erreur "***Erreur : Auto-référence détectée***". Cela signifie que la tentative d'introduction de l'équivalence a créé une référence circulaire, car le concept ***sculpteur*** est impliqué dans la définition du concept ***sculpture***, et vice versa, créant ainsi une boucle dans la hiérarchie des concepts.

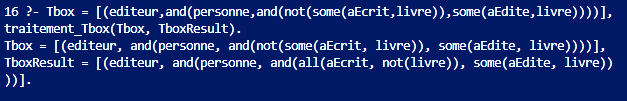
## **Traitement de la T-Box et de l'A-Box**

Le traitement des T-Box et A-Box repose sur le prédicat ***remplace/2***, qui réalise un remplacement récursif des identificateurs de concepts complexes. La T-Box est traitée par ***traitement\_Tbox/2***, où les expressions conceptuelles équivalentes à des concepts complexes sont remplacées par des identificateurs atomiques et mises sous forme normale négative. De même, ***traitement\_Abox/2*** opère sur l'A-Box, substituant les identificateurs complexes par leurs définitions de la T-Box simplifiée, suivies d'une mise sous forme normale négative.

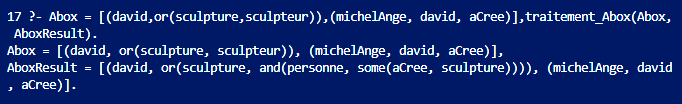
Voici quelques exemples des prédicats ***remplace/2***, ***traitement\_Tbox/2*** et ***traitement\_Abox/2* :**



Le concept ***sculpteur*** dans la disjonction ***or(sculpteur, some(aCree, sculpture))*** est remplacé par sa définition, qui est ***and(personne, some(aCree, sculpture))***.



Dans cet exemple, la T-Box originale est transformée en appliquant le remplacement récursif et en mettant les expressions de concepts dans une Forme Normale Négative.



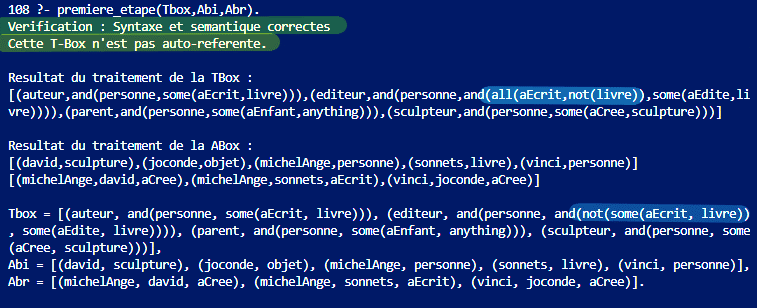
Dans cet exemple, l'A-Box originale est transformée en appliquant le remplacement récursif et la mise en forme normale conjonctive selon les règles définies.

L'orchestration complète est assurée par ***traitement\_Tbox\_Abox/3***, affichant les résultats du traitement.

## **Exécution de la Première Étape**

Le prédicat ***premiere\_etape/3*** exécute la première étape du projet, englobant toutes les vérifications et traitements nécessaires. Il débute par la validation syntaxique et sémantique des éléments, assurée par le prédicat ***verifier\_concept/3***. En suivant, il s'assure de l'intégrité de la T-Box en vérifiant l'absence d'auto-référence à l'aide de ***verifier\_auto\_reference/1***. Enfin, le traitement de la T-Box et de l'A-Box est effectué par le prédicat ***traitement\_Tbox\_Abox/3***, présentant ainsi des résultats prêts pour la suite du projet.

L'exécution de la première étape du projet a produit les résultats suivants :



On observe que suite à la vérification syntaxique, sémantique et la détection d'auto-référence, tous les concepts de la T-Box sont mis sous forme normale négative. Ceci est particulièrement visible dans des expressions telles que ***and(not(some(aEcrit, livre)))***, qui ont été transformées en ***and(all(aEcrit,not(livre)))***.

# **Implémentation de la deuxième partie**

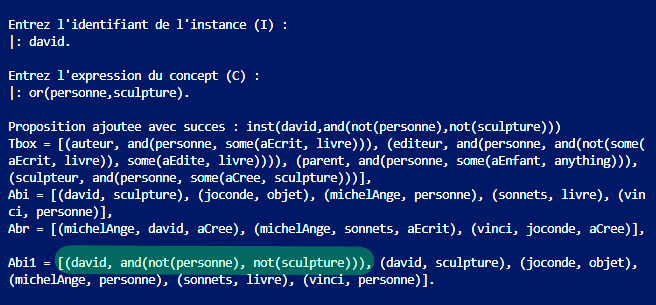
## **Proposition de Type 1 (Appartenance d'Instance à un Concept)**

L'utilisateur commence par choisir le type de proposition (1) et est ensuite invité à entrer l'identifiant d'une instance. Le prédicat ***entrerInstance/1***gère cette étape, vérifiant la validité de l'instance saisie. En cas d'erreur, un message d'erreur est affiché, et l'utilisateur est invité à réessayer.

Après la saisie réussie de l'instance, l'utilisateur est ensuite invité à entrer l'expression d'un concept associé à cette instance. Le prédicat ***entrerConcept/1***prend en charge cette étape, vérifiant la validité de l'expression du concept. Si une erreur est détectée, un message d'erreur est affiché, et l'utilisateur est invité à corriger son entrée.

Une fois l'identifiant de l'instance et l'expression du concept validés, la proposition est considérée comme correcte et ajoutée avec succès à la T-Box. Un message de confirmation est affiché, informant l'utilisateur que la proposition a été ajoutée avec succès.

Voici un exemple de l’exécution de ***acquisition\_prop\_type1/3***sur la base fournie :



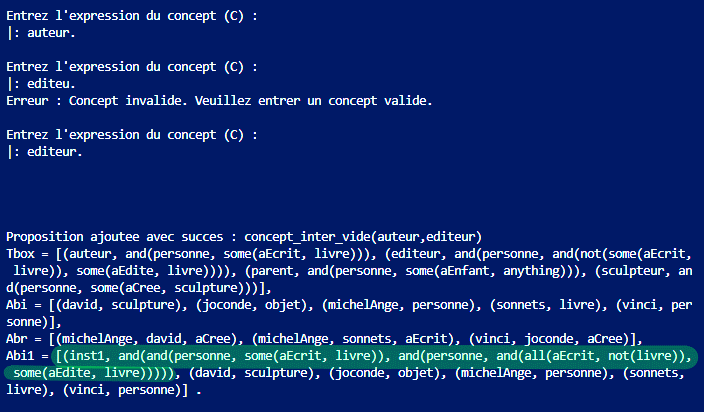
## **Proposition de Type 2 (Intersection Vide)**

L'utilisateur commence par choisir le type de proposition (2) et est ensuite invité à entrer l'expression de deux concepts. Le prédicat ***entrerConcept/1*** gère cette étape, vérifiant la validité de l'expression du concept. En cas d'erreur, un message d'erreur est affiché, et l'utilisateur est invité à réessayer.

Après la saisie réussie des deux concepts, le prédicat ***acquisition\_prop\_type2/3*** effectue le remplacement récursif des concepts, les met sous forme normale négative, et génère un nom pour la nouvelle proposition. Enfin, la proposition est ajoutée avec succès à la T-Box.

Un message de confirmation est affiché, informant l'utilisateur que la proposition a été ajoutée avec succès.

Voici un exemple de l’exécution de ***acquisition\_prop\_type2/3*** sur la base fournie :



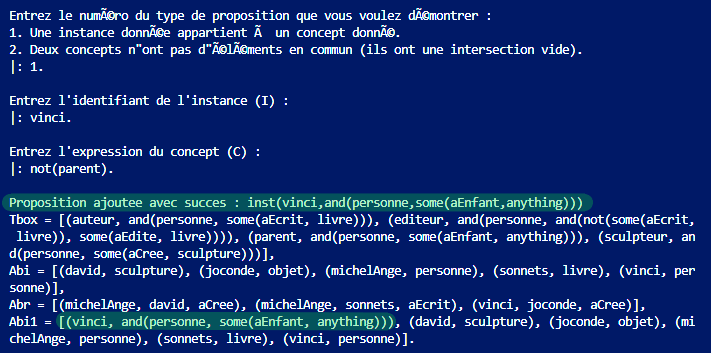
## **Exécution de la Deuxième Étape**

Le prédicat ***deuxieme\_etape/3*** est appelé, recevant la T-Box, l'A-Box d'instances, et l'A-Box de rôles comme arguments. Il fait appel au prédicat ***saisie\_et\_traitement\_prop\_a\_demontrer/3*** pour permettre à l'utilisateur de choisir le type de proposition à démontrer.

Le prédicat ***saisie\_et\_traitement\_prop\_a\_demontrer/3*** guide l'utilisateur en lui demandant de saisir le numéro correspondant au type de proposition qu'il souhaite démontrer. L'utilisateur peut choisir entre deux types de propositions, soit une instance donnée appartient à un concept donné (1), soit deux concepts n'ont pas d'éléments en commun (2).

En fonction de la réponse de l'utilisateur, le prédicat suite/4 appelle le prédicat approprié (***acquisition\_prop\_type1/3*** ou ***acquisition\_prop\_type2/3***) pour traiter et démontrer la proposition. En cas de réponse incorrecte, un message d'erreur est affiché, et l'utilisateur est invité à réessayer.

Voici un exemple de l’exécution de ***deuxieme\_etape/3*** sur la base fournie :

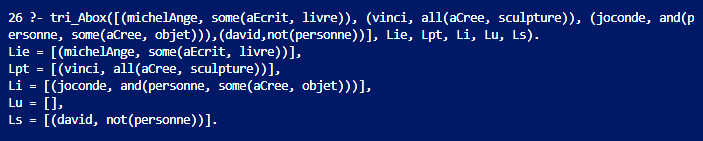


# **Implémentation de la troisième partie**

## **Tri des Assertions de l'ABox**

Le prédicat ***tri\_Abox/6*** est conçu pour trier les assertions de l'A-Box étendue dans différentes listes en fonction de leur type. Il prend en entrée la liste de l'A-Box étendue et renvoie six listes distinctes en fonction du type d'assertion. Le prédicat parcourt chaque assertion de l'A-Box et la place dans la liste appropriée en fonction du type de relation utilisée.

Voici un exemple de l’exécution de ***tri\_Abox/6*** :



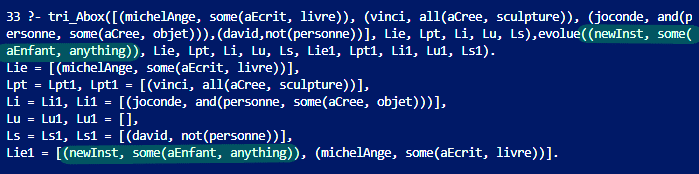
## **Évolution de l'A-Box**

Le prédicat ***evolue/11*** est conçu pour mettre à jour l'état de l'A-Box étendue en ajoutant une nouvelle assertion. Il prend en compte différents types d’assertions.

* Si l'assertion existe déjà dans la liste correspondante, l'état de l'A-Box reste inchangé.
* Si l'assertion n'existe pas dans la liste, elle est ajoutée à la liste appropriée.

Le prédicat ***evolue/11*** prend en entrée l'assertion à ajouter et l'état actuel de l'A-Box, et renvoie l'état mis à jour de l'A-Box. Il est capable de traiter à la fois des assertions individuelles et des listes d'assertions, ce qui le rend particulièrement utile lorsqu'il est nécessaire d'ajouter plusieurs assertions simultanément.

Voici un exemple de l’exécution de ***evolue/11*** :



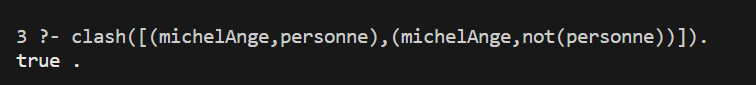
## **Résolution**

Le prédicat ***clash/1*** est conçu pour tester s'il existe un "clash" dans la liste des assertions ‘***Ls*** ‘. Un "clash" survient lorsqu'il y a une paire d'assertions du type ‘***(I, C)*** ’ et ‘***(I, not(C))*** ’ pour un même individu ‘***I*** ‘ et concept ‘***C’***.

Le prédicat ***resolution/6*** représente la phase de résolution du raisonnement. Il fait appel à différents prédicats en fonction de la configuration des listes d'assertions. Si une règle peut être appliquée, le prédicat ***clash/1*** est utilisé pour s'assurer qu'il n'y a pas de "***clash*** " dans la liste ‘***Ls*** ’ avant de poursuivre le raisonnement. Si un "***clash*** " est détecté, le prédicat échoue.

Si aucune règle ne s'applique et toutes les listes sont vides sauf ‘***Ls*** ’, alors aucune des règles ne peut être appliquée, la situation du nœud en cours ne peut plus évoluer. Si dans ce cas, aucun "clash" n'est détecté, la branche en cours est ouverte, ce qui signifie que la ‘***Abr’*** est satisfiable, et l'échec de la résolution est déclaré. On n'a pas pu démontrer la proposition initiale.

Si toutes les branches de l'arbre de résolution sont fermées, ‘***Abr*** ’ est considéré comme insatisfaisable, et l'on peut donc affirmer que la proposition initiale est démontrée.



Ici on voit bien qu’il y a effectivement un clash car “michelAnge” ne peut pas etre a la fois une personne et pas une personne.

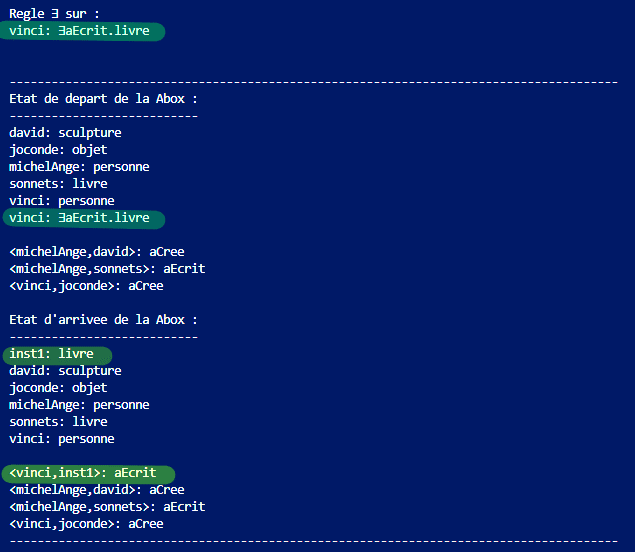
## **Application de la Règle d'Existence**

Le prédicat ***complete\_some/6*** représente l'application de la règle "***il existe***" lors du raisonnement. Il est utilisé pour traiter le cas où une assertion existentielle est présente dans la Abox.

Le prédicat génère un nouvel individu ***Nom*** et ajoute une nouvelle assertion de concept ***(Nom, C)*** à la Abox en utilisant le prédicat ***evolue/11***. Ensuite, il affiche l'évolution de l'A-Box à l'aide du prédicat ***affiche\_evolution\_Abox/12*** et appelle le prédicat ***resolution/6*** pour poursuivre le raisonnement.

La règle "***il existe***" est ainsi appliquée, permettant la création d'un nouvel individu pour satisfaire l'assertion existentielle.

Voici un exemple de l’exécution de ***complete\_some/6*** :



## **Application de la Règle de Conjonction**

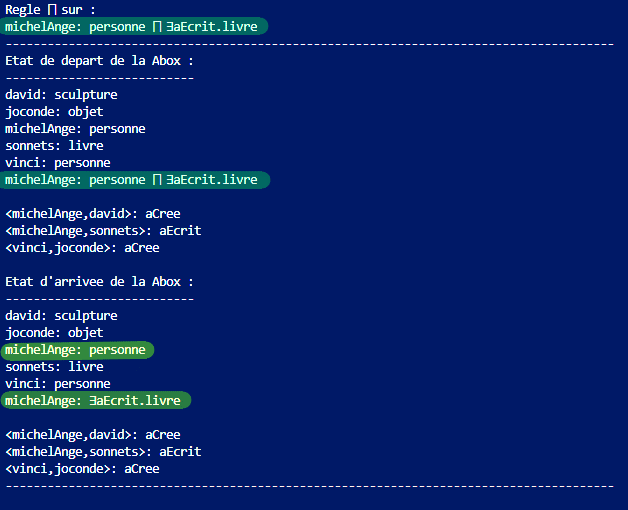
Le prédicat ***transformation\_and/6*** représente l'application de la règle de conjonction "***and***" lors du raisonnement. Il est utilisé pour traiter le cas où une assertion de conjonction est présente dans la Abox.

Le prédicat applique la règle de conjonction en deux étapes :

* Il ajoute la partie gauche de la conjonction ***(I, C1)*** à la Abox en appelant le prédicat ***evolue/11***.
* Il ajoute la partie droite de la conjonction ***(I, C2)*** à la Abox, en utilisant la nouvelle Abox obtenue après l'ajout de la partie gauche.

Ensuite, le prédicat affiche l'évolution de l'A-Box à l'aide du prédicat ***affiche\_evolution\_Abox/12*** et appelle le prédicat ***resolution/6*** pour poursuivre le raisonnement.

Voici un exemple de l'exécution de ***transformation\_and/6*** :



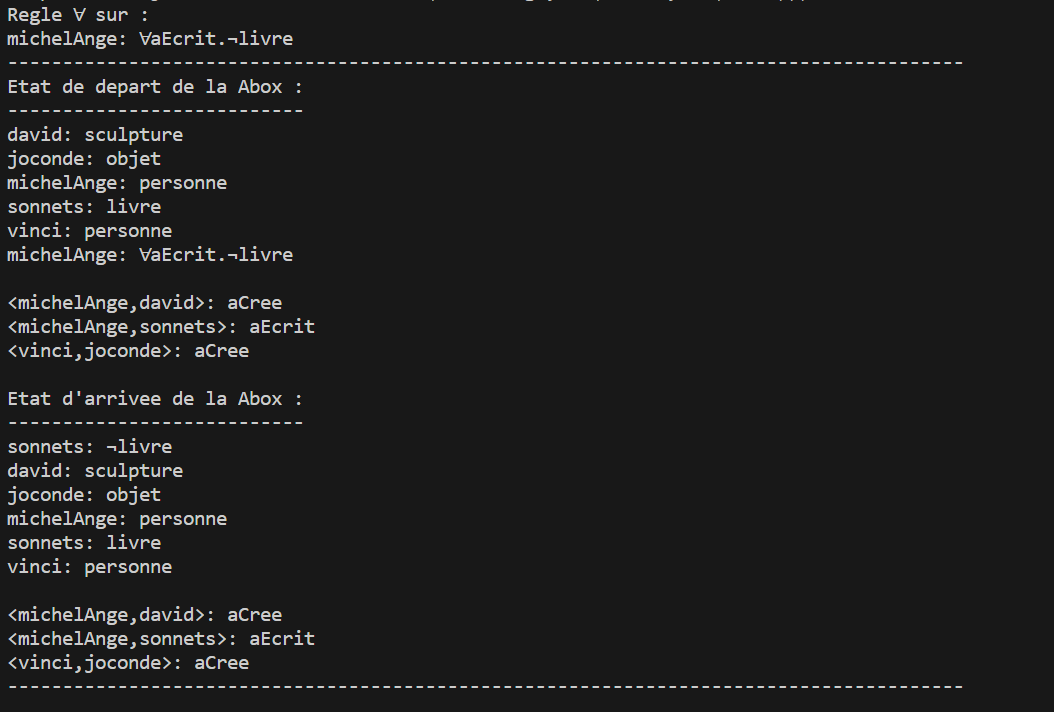
## **Application de la Règle de Déduction**

Le prédicat ***deduction\_all/6*** représente la règle d'inférence universelle (“for all”) lors du raisonnement. Le prédicat traite le cas ou la propriétée P est vraie pour tout élément x d'un ensemble A, alors P est vraie pour tout élément de A.

On cherche d’abord a affirmer que tous les objets en relation ***R*** avec ***I*** doivent également satisfaire le concept ***C.*** Dans le cas ou cela est vrai, on call ***evolue/11*** pour traiter l’assertion.

Ensuite, le prédicat affiche l'évolution de l'A-Box à l'aide du prédicat ***affiche\_evolution\_Abox/12*** et appelle le prédicat ***resolution/6*** pour poursuivre le raisonnement.

Voici un exemple de l'exécution de ***deduction\_all/6*** :



## **Application de la Règle de Disjonction**

Le prédicat ***transformation\_or/6*** représente l'application de la règle de disjonction "***or***" lors du raisonnement. Il est utilisé pour traiter le cas où une assertion de disjonction est présente dans la Abox.

La structure est similaire a la conjonction, car on traite de nouveau les deux concepts C1 et C2 de la disjonction, sauf qu’il faut deux resolutions différentes car on peut trouver une branche ouverte à droite ou a gauche.

Le prédicat affiche aussi l'évolution de l'A-Box à l'aide du prédicat ***affiche\_evolution\_Abox/12*** après chaque évolution.

Voici un exemple de l'exécution de ***transformation\_or/6*** :

