

# Compte rendu Projet de Connaissances et raisonnement

CONTI Vivien, GAUTIER Alexandre

## Sujet :

Votre mission consiste à capturer, sous une forme logique, les connaissances suffisantes pour répondre à une série de questions relatives au scénario suivant :

*Hier, Jean est allé au supermarché Safeway de North Berkeley et a acheté un kilo de tomates et une livre de viande hachée.*

Commencez par essayer de représenter le contenu de l'énoncé sous la forme d'une série d'assertions. Faites en sorte d'écrire des énoncés ayant une structure logique simple (c'est-à-dire des instructions énonçant que les objets ont certaines propriétés, qu'ils sont liés d'une certaine manière, que tous les objets qui satisfont une propriété en satisfont une autre). Les points suivants peuvent vous aider à démarrer :

- De quelles classes, de quels objets et de quelles relations avez-vous besoin ? Quels sont les parents, frères, etc. ? (Vous aurez notamment besoin d'événements et d'un ordonnancement temporel.)
- Où s'intégreraient-ils dans une hiérarchie plus générale ?
- Quelles sont les contraintes et les relations entre eux ?
- Jusqu'à quel niveau de détail vous faut-il aller pour chacun des différents concepts ?

Pour répondre aux questions ci-après, votre base de connaissances doit contenir des connaissances générales. Vous devrez prendre en compte ce qu'on trouve dans un supermarché, comment les articles choisis sont achetés, ce à quoi ils serviront, etc. Essayez de rendre votre représentation aussi générale que possible. À titre d'exemple simple, ne dites pas : « Les gens achètent de la nourriture chez Safeway », car cela ne vous aidera pas à prendre en compte ceux qui font leurs courses dans un autre supermarché. Par ailleurs, ne transformez pas les questions en réponses. Par exemple, la question (c) est : « Est-ce que Jean a acheté de la viande ? » et non : « Est-ce que Jean a acheté une livre de viande hachée ? ».

Esquissez les enchaînements de raisonnements qui répondraient aux questions. Dans la mesure du possible, utilisez un système de raisonnement logique pour démontrer que votre base de connaissances suffit. Vous écrirez peut-être beaucoup de choses qui ne correspondront que de manière approximative à la réalité. Ne vous en inquiétez pas outre mesure : l'idée est d'extraire le sens commun qui permet de répondre à ces questions. Il est extrêmement difficile d'apporter une réponse absolument complète à cet exercice, probablement même au-delà des capacités actuelles de la représentation des connaissances. Mais vous devriez pouvoir constituer un ensemble cohérent d'axiomes pour les questions limitées posées ici.

## **Choix techniques :**

Nous avons utilisé Prover9 après avoir constaté qu'il était extrêmement rapide comparé au solveur de nltk utilisé en TD de ModLog, mais nous avons quand même décidé de laisser l'implémentation avec nltk disponible au besoin.

Pour reproduire les preuves, il suffit d'ouvrir Prover9 et de charger le fichier prover9\_input dedans (ou de copier-coller le contenu de InputProver9\_assumptions.txt dans la partie *Assumptions*, les *Goals* étant à la fin en commentaires).

## **Classes et objets utilisés :**

Nous avons implémenté différentes assertions pour la base sémantique du problème. Le sens des variables généralement présentes dans ces assertions est renseigné ci-dessous :

- x : Personne
- q ou n : Quantité
- z : Supermarché
- y : Item
- t : Temps
- m : Argent

## **Exemple de prédicats utilisés :**

Personne(x), Nourriture(y), Viande(y), Temps(t), Supermarche(z), Achete(x,y,z,q,t), Masse(q), Item(y), Prepare(z,y), Vend(z,y), ApporteCarteCredit(x), PossedeCarteCredit(x), Adulte(x), A\_Argent(x,m,t), MoinsQue(q1,q2), Rencontre(x1,x2]

## **Toutes les assertions (Base de connaissances et traductions en First Order Logic) :**

Pour modéliser les connaissances, nous avons dû faire certains choix :

- Pour déterminer si Jean était un adulte ou un enfant, nous avons fait le choix de dire que Jean avait une carte de crédit, et que toute personne en possédant une est un adulte.
- Pour déterminer si Jean avait au moins deux tomates, nous avons dit que toute personne possédant un kilo de tomate en possède au moins deux.
- Pour déterminer si Safeway avait préparé les tomates, nous avons dit qu'un supermarché ne prépare aucun produit.
- Pour déterminer si Jean allait manger les tomates, nous avons choisi de dire qu'une personne ayant acheté un produit alimentaire va le manger.

Les assertions sont les suivantes :

%%% DONNEES DIRECTEMENT ISSUES DE L'ENONCE

*%Jean est une personne*

Personne(Jean).

*%Les tomates sont de la nourriture*

Nourriture(Tomate).

*%La viande hachee est de la viande*

Viande(ViandeHachee).

*%Hier est un temps*

Temps(hier).

*%Jean a achete un kilo de tomate hier à Safeway*

Achete(Jean,Tomate,Safeway,UnKilo,hier).

*%Jean a achete une livre de viande hachee hier à Safeway*

Achete(Jean,ViandeHachee,Safeway,UneLivre,hier).

*%Safeway est un supermarche*

Supermarche(Safeway).

%%% Question 1

*%Tout ce qui est de la viande est de la nourriture*

all x (Viande(x) -> Nourriture(x)).

*%Tout ce qui est de la nourriture est un item*

all x (Nourriture(x) -> Item(x)).

*%Jean possède une carte de crédit*

PossedeCarteCredit(Jean).

*%Toute personne qui possède une carte de crédit est un adulte*

all x (Personne(x) & PossedeCarteCredit(x) -> Adulte(x)).

*%Un enfant n'est pas un adulte*

all x (Enfant(x) -> -Adulte(x)).

%%% Question 2

*%UnKilo est un Kilo*

Kilo(UnKilo).

*%Les kilos sont des masses*

all q (Kilo(q) -> Masse(q)).

*%UneLivre est une livre*

Livre(UneLivre).

*%Les livres sont des masses*

all q (Livre(q) -> Masse(q)).

*%PossedeMasse(x, y, q) signifie x possède q masses de y*

*%Toute personne x qui achete un item y dans un supermarché z en masse Q à un moment t possède Q masses de y et apporte un moyen de paiement*

all x all y all z all q all t (Achete(x,y,z,q,t) & Personne(x) & Item(y) & Supermarche(z) & Masse(q) & Temps(t) -> PossedeMasse(x,y,q) & ApporteMoyenPaiement(x)).

*% PossedeAuMoinsQuantite (x, y, q) signifie x possède q unités de y*

*%Toute personne qui possède UnKilo de tomate possède au moins deux tomates*

all x (PossedeMasse(x,Tomate,UnKilo)-> exists n (PossedeAuMoinsQuantite (x, Tomato, n) & PlusQue(n, deux))).

*%Toute personne x ayant n1 quantité de y avec n1 >= n2 en possède aussi n2 quantité*

all x all y all n1 all n2 (PossedeAuMoinsQuantite (x, y, n1) & PlusQue(n1, n2) -> PossedeAuMoinsQuantite (x, y, n2)).

%%% QUESTION 3

*%Toute personne x qui achete de la viande a achete de la viande*

all x all y all z all q all t (Achete(x,y,z,q,t) & Personne(x) & Viande(y) & Supermarche(z) & Masse(q) & Temps(t) -> A\_AcheteViande(x)).

%%% QUESTION 4

*%Toute personne x et toute personne m qui achètent le même item y au même supermarche z au même moment se rencontrent*

all x all y all z all m all t all q1 all q2(Personne(x) & Achete(x,y,z,q1,t) & Personne(m) & Achete(m,y,z,q2,t) & Item(y) & Supermarche(z) & Temps(t)->Rencontre(x,m)).

%%% QUESTION 5

*%Les supermarches ne préparent aucun item*

all z all y(Supermarche(z) & Item(y) -> -Prepare(z,y)).

%%% QUESTION 6

*%Toute personne qui achete de la nourriture en mange*

all x all y(exists z exists q exists t(Personne(x) & Achete(x,y,z,q,t) & Nourriture(y)) -> Mange(x,y)).

%%% QUESTION 7

*%Les déodorants sont des produits non alimentaires*

ProduitNonAlimentaire(Deodorant).

*%Les produits non alimentaires sont des items*

all x(ProduitNonAlimentaire(x)->Item(x)).

*%Tout Supermarche z qui a vendu un item y à un client x à un moment t possède l'item y*

all y all z(exists x exists t exists q(Supermarche(z) & Item(y) & Achete(x, y, z, q, t)) -> Has(x, y)).

*%Tout supermarché possède des déodorants*

all z(Supermarche(z) -> Has(z, Deodorant)).

*%Tout supermarche z qui possède un item y le vend*

all y all z(Supermarche(z) & Item(y) & Has(z, y) -> Vend(z, y)).

%%% QUESTION 8

*%Toute personne qui apporte un moyen de paiement apporte soit une carte de crédit soit du cash*

all x (ApporteMoyenPaiement(x) -> ApporteCarteCredit(x) | ApporteCash(x)).

*%Toute personne qui apporte une carte de crédit en possède une*

all x (ApporteCarteCredit(x) -> PossedeCarteCredit(x)).

*%Toute personne qui apporte du cash a possédé de l'argent en quantité m à un moment t*

all x (ApporteCash(x) -> exists m exists t(A\_Argent(x, m, t))).

%%% QUESTION 9

*%Jean avait une quantite d'argent M1 hier*  
A\_Argent(Jean, M1, hier).

*%aujourd'hui est un temps*  
Temps(aujourd'hui).

*%hier est avant aujourd'hui*  
EstAvant(hier, aujourd'hui).

*%Pour toute quantités m1, m2, si m1 est moins que m2 alors m2 est plus que m1*  
all m1 all m2(MoinsQue(m1, m2) -> PlusQue(m2, m1)).

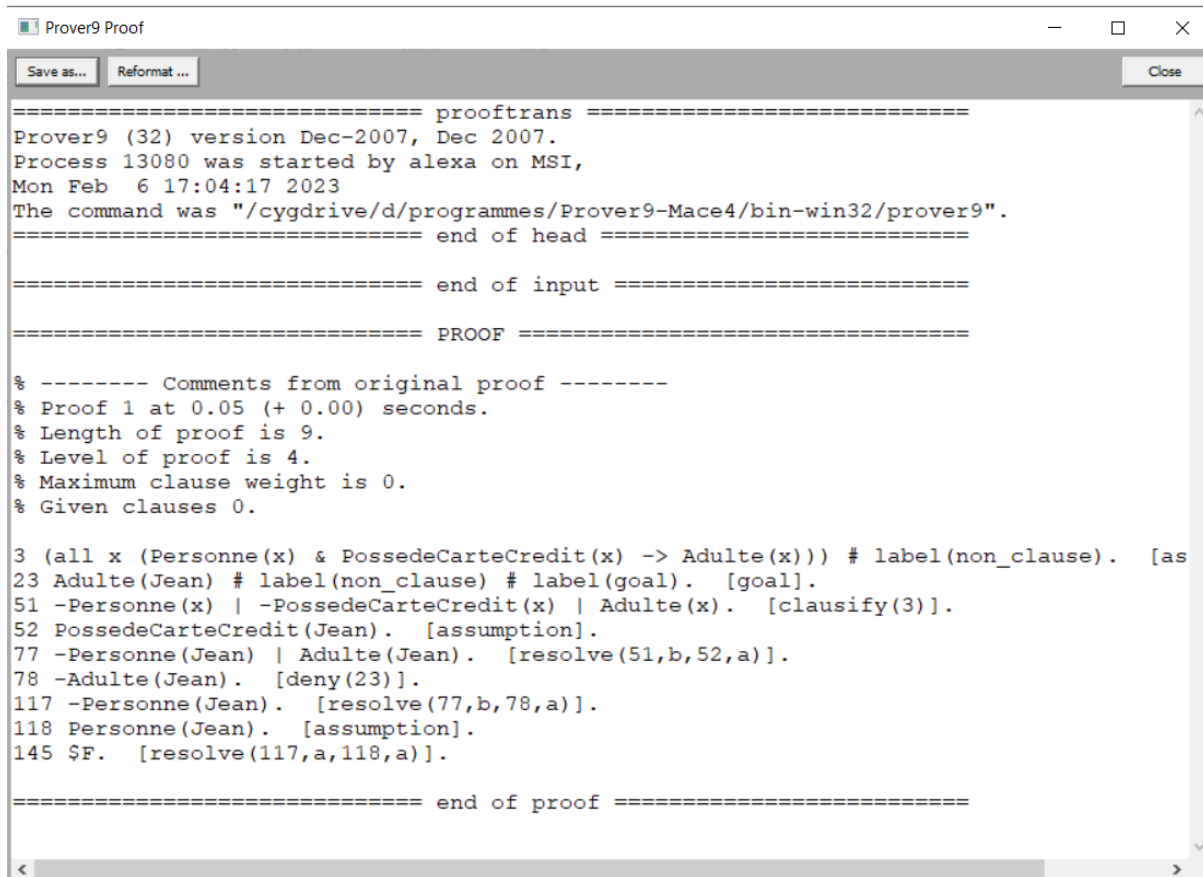
*%Toute personne qui achete un produit à a instant t1, qui possède de l'argent en quantité m1 possède de l'argent en quantité m2 à t2 avec m2 < m1 si t1 est avant t2*  
all x all y all z all q all t1 all t2 all m1 all m2(Achete(x,y,z,q,t1) & A\_Argent(x,m1,t1) & EstAvant(t1, t2) -> A\_Argent(x,m2,t2) & MoinsQue(m2,m1)).

## Questions posées à Prover9 :

a. Jean est-il un enfant ou un adulte ? *[Un adulte]*

Goal :

Adulte(Jean).



```
Prover9 Proof
Save as... Reformat ... Close

===== prooftrans =====
Prover9 (32) version Dec-2007, Dec 2007.
Process 13080 was started by alexa on MSI,
Mon Feb  6 17:04:17 2023
The command was "/cygdrive/d/programmes/Prover9-Mace4/bin-win32/prover9".
===== end of head =====

===== end of input =====

===== PROOF =====

% ----- Comments from original proof -----
% Proof 1 at 0.05 (+ 0.00) seconds.
% Length of proof is 9.
% Level of proof is 4.
% Maximum clause weight is 0.
% Given clauses 0.

3 (all x (Personne(x) & PossedeCarteCredit(x) -> Adulte(x))) # label(non_clause). [as
23 Adulte(Jean) # label(non_clause) # label(goal). [goal].
51 -Personne(x) | -PossedeCarteCredit(x) | Adulte(x). [clausify(3)].
52 PossedeCarteCredit(Jean). [assumption].
77 -Personne(Jean) | Adulte(Jean). [resolve(51,b,52,a)].
78 -Adulte(Jean). [deny(23)].
117 -Personne(Jean). [resolve(77,b,78,a)].
118 Personne(Jean). [assumption].
145 $F. [resolve(117,a,118,a)].

===== end of proof =====
```

b. Jean possède-t-il désormais au moins deux tomates ? *[Oui]*

Goal :

PossedeAuMoinsQuantite(Jean, Tomate, deux).

Marche aussi avec :

PlusQue(deux, un) -> PossedeAuMoinsQuantite(Jean, Tomate, un).

```
Prover9 Proof

Save as... Reformat... Close

The command was "/cygdrive/d/programmes/Prover9-Mace4/bin-win32/prover9".
=====
end of head =====
=====
end of input =====
=====
PROOF =====

% ----- Comments from original proof -----
% Proof 1 at 0.03 (+ 0.00) seconds.
% Length of proof is 36.
% Level of proof is 11.
% Maximum clause weight is 9.
% Given clauses 6.

2 (all x (Nourriture(x) -> Item(x)) # label(non_clause). [assumption].
5 (all q (Kilo(q) -> Masse(q)) # label(non_clause). [assumption].
7 (all x all y all z all t (Achete(x,y,z,q,t) & Personne(x) & Item(y) & Supermarche(z) & Masse(q) & Temps(t) -> PossedeMasse(x,y,q) & ApporteMoyenPaiement(x)) # label(non_clause). [assumption].
9 (all x (PossedeMasse(x,Tomate,UnKilo) -> existe x (PossedeAuMoinsQuantite(x,Tomate,n) & PlusQue(n,deux)))) # label(non_clause). [assumption].
9 (all x all y all n1 all n2 (PossedeAuMoinsQuantite(x,y,n1) & PlusQue(n1,n2) -> PossedeAuMoinsQuantite(x,y,n2)) # label(non_clause). [assumption].
23 PossedeAuMoinsQuantite(Jean,Tomate,deux) # label(non_clause) # label(goal). [goal].
24 -Nourriture(x) | Item(x). [clausify(2)].
28 Nourriture(Tomate). [assumption].
32 -Achete(x,y,z,u,w) | -Personne(x) | -Item(y) | -Supermarche(z) | -Masse(u) | -Temps(w) | PossedeMasse(x,y,u). [clausify(7)].
33 Temps(hier). [assumption].
39 Supermarche(Safeway). [assumption].
43 -Achete(x,y,z,u,hier) | -Personne(x) | -Item(y) | -Supermarche(z) | -Masse(u) | PossedeMasse(x,y,u). [resolve(32,f,39,a)].
44 -Kilo(n) | Masse(n). [clausify(5)].
48 Kilo(UnKilo). [assumption].
50 -Achete(x,y,Safeway,z,hier) | -Personne(x) | -Item(y) | -Masse(z) | PossedeMasse(x,y,z). [resolve(43,d,39,a)].
59 -PossedeMasse(x,Tomate,UnKilo) | PossedeAuMoinsQuantite(x,Tomate,fi(x)). [clausify(8)].
60 -PossedeMasse(x,Tomate,UnKilo) | PlusQue(fi(x),deux). [clausify(8)].
63 -PossedeAuMoinsQuantite(x,y,z) | -PlusQue(x,z,u) | PossedeAuMoinsQuantite(x,y,u). [clausify(9)].
64 -Achete(x,Tomate,Safeway,UnKilo,hier) | -Personne(x) | -Item(Tomate) | -Masse(UnKilo) | PlusQue(fi(x),deux). [resolve(50,e,60,a)].
78 Item(Tomate). [resolve(24,a,28,a)].
84 -Achete(x,Tomate,Safeway,UnKilo,hier) | -Personne(x) | -Item(Tomate) | -Masse(UnKilo) | PossedeAuMoinsQuantite(x,Tomate,fi(x)). [resolve(50,e,59,a)].
98 Masse(UnKilo). [resolve(54,a,56,a)].
103 -Achete(x,Tomate,Safeway,UnKilo,hier) | -Personne(x) | -Masse(UnKilo) | -PossedeAuMoinsQuantite(x,Tomate,fi(x)). [resolve(64,c,78,a)].
104 -Achete(x,Tomate,Safeway,UnKilo,hier) | -Personne(x) | -Masse(UnKilo) | PossedeAuMoinsQuantite(y,z,fi(x)) | PossedeAuMoinsQuantite(y,z,deux). [resolve(86,c,78,a)].
115 Personne(Jean). [assumption].
119 -Achete(x,Tomate,Safeway,UnKilo,hier) | -Personne(x) | PossedeAuMoinsQuantite(x,Tomate,fi(x)). [resolve(102,c,98,a)].
121 -Achete(x,Tomate,Safeway,UnKilo,hier) | -Personne(x) | PossedeAuMoinsQuantite(y,z,fi(x)) | PossedeAuMoinsQuantite(y,z,deux). [resolve(104,c,98,a)].
136 Achete(Jean,Tomate,Safeway,UnKilo,hier). [assumption].
145 -Achete(Jean,Tomate,Safeway,UnKilo,hier) | PossedeAuMoinsQuantite(Jean,Tomate,fi(Jean)). [resolve(119,b,116,a)].
147 -Achete(Jean,Tomate,Safeway,UnKilo,hier) | -PossedeAuMoinsQuantite(x,y,fi(Jean)) | PossedeAuMoinsQuantite(x,y,deux). [resolve(121,b,116,a)].
163 -PossedeAuMoinsQuantite(Jean,Tomate,deux). [deny(23)].
164 PossedeAuMoinsQuantite(Jean,Tomate,fi(Jean)). [resolve(145,a,136,a)].
166 -PossedeAuMoinsQuantite(x,y,fi(Jean)) | PossedeAuMoinsQuantite(x,y,deux). [resolve(147,a,136,a)].
168 PossedeAuMoinsQuantite(Jean,Tomate,deux). [hyper(164,a,165,a)].
169 SF. [resolve(168,a,162,a)].

=====
end of proof =====
```

c. Jean a-t-il acheté de la viande ? *[Oui]*

Goal :

A\_AcheteViande(Jean).

```
Prover9 Proof

Save as... Reformat... Close

Prover9 (32) version Dec-2007, Dec 2007.
Process 14584 was started by alexa on MSI,
Mon Feb 6 17:09:04 2023
The command was "/cygdrive/d/programmes/Prover9-Mace4/bin-win32/prover9".
=====
end of head =====
=====
end of input =====
=====
PROOF =====

% ----- Comments from original proof -----
% Proof 1 at 0.03 (+ 0.01) seconds.
% Length of proof is 20.
% Level of proof is 8.
% Maximum clause weight is 0.
% Given clauses 0.

6 (all q (Livre(q) -> Masse(q)) # label(non_clause). [assumption].
10 (all x all y all z all q all t (Achete(x,y,z,q,t) & Personne(x) & Viande(y) & Supermarche(z) & Masse(q) & Temps(t) -> A_AcheteViande(x)) # label(non_clause). [assumption].
23 A_AcheteViande(Jean) # label(non_clause) # label(goal). [goal].
28 -Achete(x,y,z,u,w) | -Personne(x) | -Viande(y) | -Supermarche(z) | -Masse(u) | -Temps(w) | A_AcheteViande(x). [clausify(10)].
29 Viande(ViandeHachee). [assumption].
33 Temps(hier). [assumption].
37 -Achete(x,ViandeHachee,y,z,u) | -Personne(x) | -Supermarche(y) | -Masse(z) | -Temps(u) | A_AcheteViande(x). [resolve(28,c,29,a)].
39 Supermarche(Safeway). [assumption].
49 -Achete(x,ViandeHachee,y,z,hier) | -Personne(x) | -Supermarche(y) | -Masse(z) | A_AcheteViande(x). [resolve(37,e,33,a)].
56 -Livre(x) | Masse(x). [clausify(6)].
57 Livre(UneLivre). [assumption].
77 -Achete(x,ViandeHachee,Safeway,y,hier) | -Personne(x) | -Masse(y) | A_AcheteViande(x). [resolve(49,c,39,a)].
78 -A_AcheteViande(Jean). [deny(23)].
101 -Achete(Jean,ViandeHachee,Safeway,x,hier) | -Personne(Jean) | -Masse(x). [resolve(77,d,78,a)].
103 Masse(UneLivre). [resolve(56,a,57,a)].
118 Personne(Jean). [assumption].
119 -Achete(Jean,ViandeHachee,Safeway,UneLivre,hier) | -Personne(Jean). [resolve(101,c,103,a)].
140 Achete(Jean,ViandeHachee,Safeway,UneLivre,hier). [assumption].
145 -Achete(Jean,ViandeHachee,Safeway,UneLivre,hier). [resolve(119,b,118,a)].
163 SF. [resolve(145,a,140,a)].

=====
end of proof =====
```

d. Si Marie a acheté des tomates en même temps que Jean, l'a-t-elle vu ? [Oui]

Goal :

Personne(Marie) & exists q(Achete(Marie, Tomate, Safeway, q, hier)) -> Rencontre(Jean,Marie).

```
Prover9 Proof
===== prooftrans =====
Prover9 (32) version Dec-2007, Dec 2007.
Process 13672 was started by alexa on MSI,
Mon Feb 6 17:10:00 2023
The command was "/cygdrive/d/programmes/Prover9-Mace4/bin-win32/prover9".
===== end of head =====

===== end of input =====

===== PROOF =====

% ----- Comments from original proof -----
% Proof 1 at 0.03 (+ 0.00) seconds.
% Length of proof is 21.
% Level of proof is 8.
% Maximum clause weight is 12.
% Given clauses 8.

2 (all x (Nourriture(x) -> Item(x))) # label(non_clause). [assumption].
11 (all x all y all z all m all t all q1 all q2 (Personne(x) & Achete(x,y,z,q1,t) & Personne(m) & Achete(m,y,z,q2,t) & Item(y) & Supermarche(z) & Temps
23 Personne(Marie) & (exists q Achete(Marie,Tomate,Safeway,q,hier)) -> Rencontre(Jean,Marie) # label(non_clause) # label(goal). [goal].
24 -Nourriture(x) | Item(x). [clausify(2)].
25 Nourriture(Tomate). [assumption].
33 Temps(hier). [assumption].
35 -Personne(x) | -Achete(x,y,z,u,w) | -Personne(v5) | -Achete(v5,y,z,v6,w) | -Item(y) | -Supermarche(z) | -Temps(w) | Rencontre(x,v5). [clausify(11)]
39 Supermarche(Safeway). [assumption].
45 -Personne(x) | -Achete(x,y,z,u,hier) | -Personne(w) | -Achete(w,y,z,v5,hier) | -Item(y) | -Supermarche(z) | Rencontre(x,w). [resolve(35,g,33,a)].
77 -Personne(x) | -Achete(x,y,Safeway,z,hier) | -Personne(u) | -Achete(u,y,Safeway,w,hier) | -Item(y) | Rencontre(x,u). [resolve(45,f,39,a)].
78 -Rencontre(Jean,Marie). [deny(23)].
81 Item(Tomate). [resolve(24,a,25,a)].
92 -Personne(Jean) | -Achete(Jean,x,Safeway,y,hier) | -Personne(Marie) | -Achete(Marie,x,Safeway,z,hier) | -Item(x). [resolve(77,f,78,a)].
117 Personne(Jean). [assumption].
118 Achete(Jean,Tomate,Safeway,UnKilo,hier). [assumption].
121 Personne(Marie). [deny(23)].
122 Achete(Marie,Tomate,Safeway,cl,hier). [deny(23)].
124 -Personne(Jean) | -Achete(Jean,Tomate,Safeway,x,hier) | -Personne(Marie) | -Achete(Marie,Tomate,Safeway,y,hier). [resolve(92,e,81,a)].
125 -Achete(Jean,Tomate,Safeway,x,hier) | -Achete(Marie,Tomate,Safeway,y,hier). [copy(124),unit_del(a,117),unit_del(c,121)].
142 -Achete(Marie,Tomate,Safeway,x,hier). [resolve(125,a,118,a)].
143 $. [resolve(142,a,122,a)].

===== end of proof =====
```

e. Les tomates sont-elles produites dans le supermarché ? [Non]

Goal :

Prepare(Safeway, Tomate).



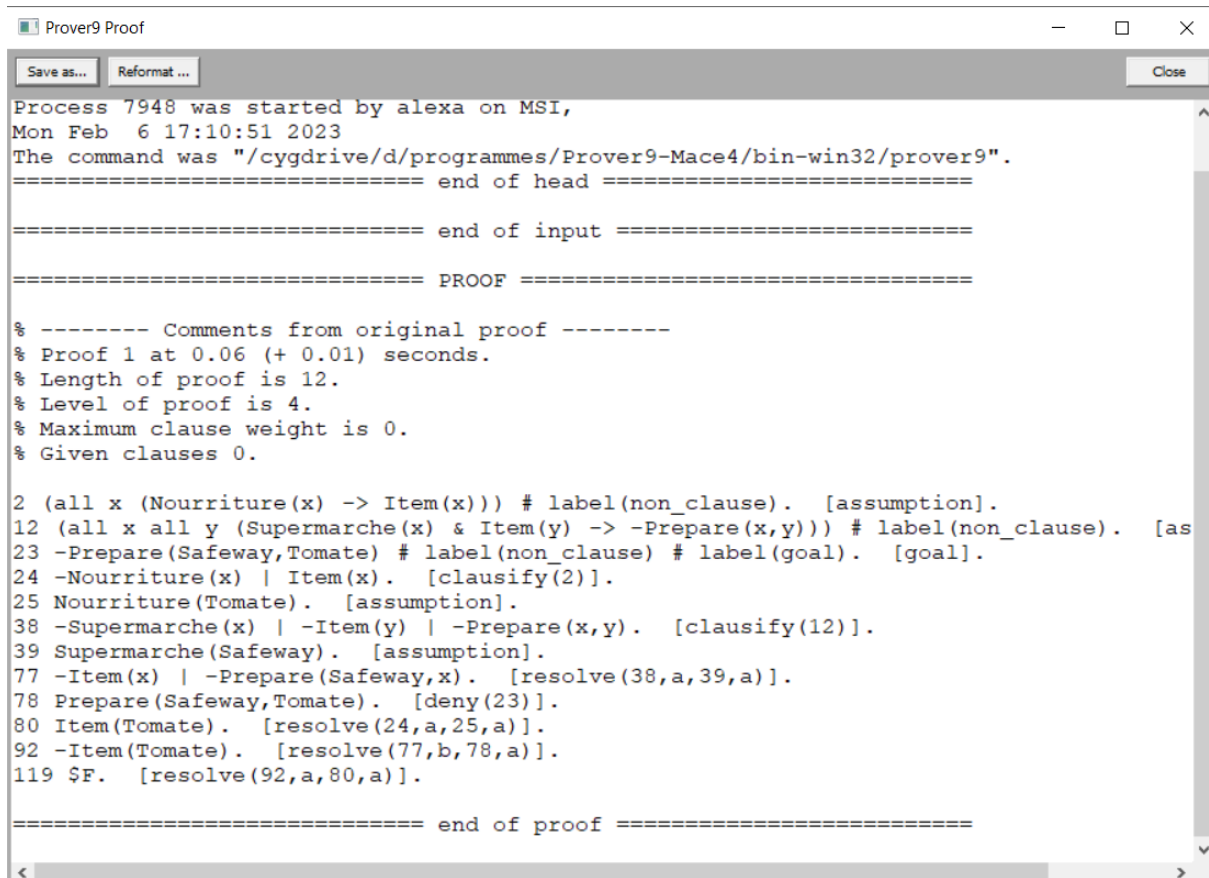
Prover9 Exit: Exhausted

OK



Goal :

-Prepare(Safeway, Tomate).



```
Prover9 Proof
Save as... Reformat ... Close

Process 7948 was started by alexa on MSI,
Mon Feb 6 17:10:51 2023
The command was "/cygdrive/d/programmes/Prover9-Mace4/bin-win32/prover9".
===== end of head =====

===== end of input =====

===== PROOF =====

% ----- Comments from original proof -----
% Proof 1 at 0.06 (+ 0.01) seconds.
% Length of proof is 12.
% Level of proof is 4.
% Maximum clause weight is 0.
% Given clauses 0.

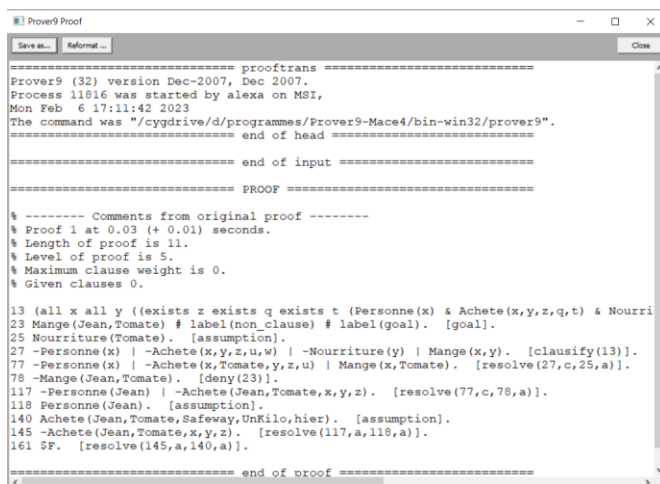
2 (all x (Nourriture(x) -> Item(x))) # label(non_clause). [assumption].
12 (all x all y (Supermarche(x) & Item(y) -> -Prepare(x,y))) # label(non_clause). [as
23 -Prepare(Safeway,Tomate) # label(non_clause) # label(goal). [goal].
24 -Nourriture(x) | Item(x). [clausify(2)].
25 Nourriture(Tomate). [assumption].
38 -Supermarche(x) | -Item(y) | -Prepare(x,y). [clausify(12)].
39 Supermarche(Safeway). [assumption].
77 -Item(x) | -Prepare(Safeway,x). [resolve(38,a,39,a)].
78 Prepare(Safeway,Tomate). [deny(23)].
80 Item(Tomate). [resolve(24,a,25,a)].
92 -Item(Tomate). [resolve(77,b,78,a)].
119 $F. [resolve(92,a,80,a)].

===== end of proof =====
```

f. Que va faire Jean de ses tomates ? [Les manger]

Goal :

Mange(Jean, Tomate).



```
Prover9 Proof
Save as... Reformat ... Close

===== prooftrans =====
Prover9 (32) version Dec-2007, Dec 2007.
Process 11816 was started by alexa on MSI,
Mon Feb 6 17:11:42 2023
The command was "/cygdrive/d/programmes/Prover9-Mace4/bin-win32/prover9".
===== end of head =====

===== end of input =====

===== PROOF =====

% ----- Comments from original proof -----
% Proof 1 at 0.03 (+ 0.01) seconds.
% Length of proof is 11.
% Level of proof is 5.
% Maximum clause weight is 0.
% Given clauses 0.

13 (all x all y ((exists z exists q exists t (Personne(x) & Achete(x,y,z,q,t) & Nourri
23 Mange(Jean,Tomate) # label(non_clause) # label(goal). [goal].
25 Nourriture(Tomate). [assumption].
27 -Personne(x) | -Achete(x,y,z,u,w) | -Nourriture(y) | Mange(x,y). [clausify(13)].
77 -Personne(x) | -Achete(x,Tomate,y,z,u) | Mange(x,Tomate). [resolve(27,c,25,a)].
78 -Mange(Jean,Tomate). [deny(23)].
117 -Personne(Jean) | -Achete(Jean,Tomate,x,y,z). [resolve(77,c,78,a)].
118 Personne(Jean). [assumption].
140 Achete(Jean,Tomate,Safeway,Unkilo,hier). [assumption].
145 -Achete(Jean,Tomate,x,y,z). [resolve(117,a,118,a)].
161 $F. [resolve(145,a,140,a)].

===== end of proof =====
```

g. Est-ce que Safeway vend des déodorants ? [Oui]

Goal:

Vend(Safeway, Deodorant).

Marche aussi avec :

Vend(Safeway, Tomate).



```
Prover9 Proof
Save as... Reformat... Close
===== prooftrans =====
Prover9 (32) version Dec-2007, Dec 2007.
Process 4284 was started by alexa on MSI,
Mon Feb 6 17:12:29 2023
The command was "/cygdrive/d/programmes/Prover9-Mace4/bin-win32/prover9".
===== end of head =====

===== end of input =====

===== PROOF =====

% ----- Comments from original proof -----
% Proof 1 at 0.03 (+ 0.03) seconds.
% Length of proof is 16.
% Level of proof is 5.
% Maximum clause weight is 0.
% Given clauses 0.

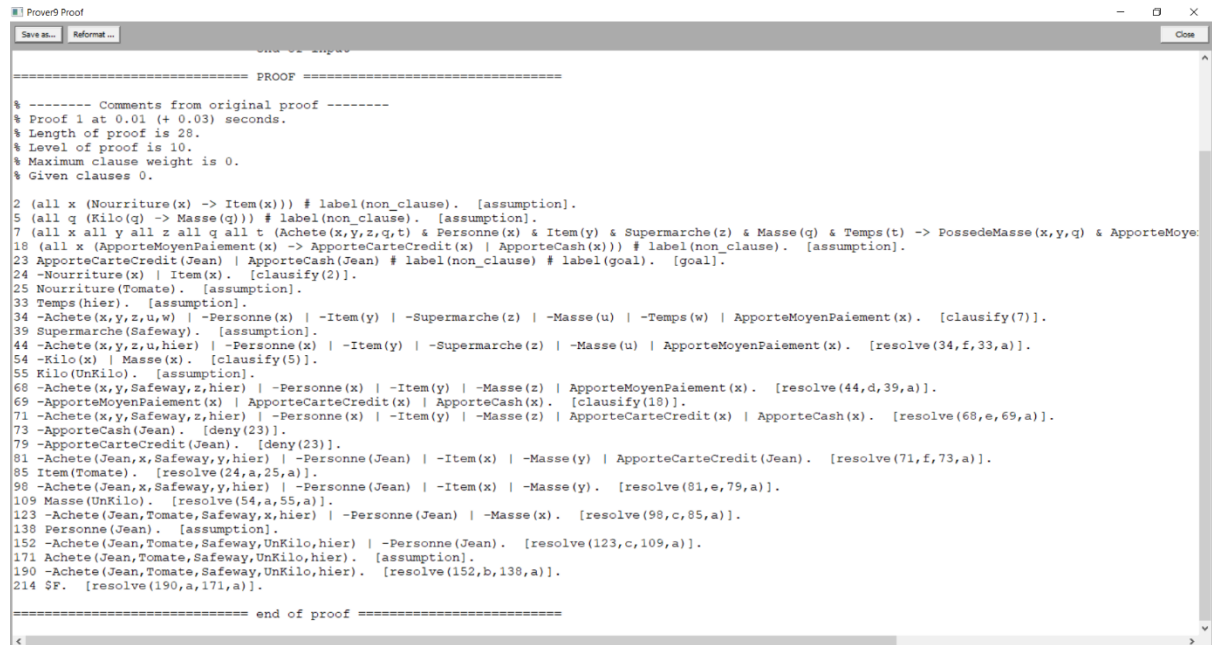
14 (all x (ProduitNonAlimentaire(x) -> Item(x))) # label(non_clause). [assumption].
16 (all x (Supermarche(x) -> Has(x,Deodorant))) # label(non_clause). [assumption].
17 (all x all y (Supermarche(y) & Item(x) & Has(y,x) -> Vend(y,x))) # label(non_clause). [assumption].
23 Vend(Safeway,Deodorant) # label(non_clause) # label(goal). [goal].
39 Supermarche(Safeway). [assumption].
41 ~Supermarche(x) | Has(x,Deodorant). [clausify(16)].
42 ~Supermarche(x) | ~Item(y) | ~Has(x,y) | Vend(x,y). [clausify(17)].
66 ~ProduitNonAlimentaire(x) | Item(x). [clausify(14)].
67 ProduitNonAlimentaire(Deodorant). [assumption].
77 ~Item(x) | ~Has(Safeway,x) | Vend(Safeway,x). [resolve(42,a,39,a)].
78 ~Vend(Safeway,Deodorant). [deny(23)].
89 Item(Deodorant). [resolve(66,a,67,a)].
92 ~Item(Deodorant) | ~Has(Safeway,Deodorant). [resolve(77,c,78,a)].
93 ~Has(Safeway,Deodorant). [resolve(92,a,89,a)].
94 Has(Safeway,Deodorant). [resolve(41,a,39,a)].
117 $F. [resolve(93,a,94,a)].

===== end of proof =====
```

h. Jean a-t-il apporté de l'argent ou une carte de paiement au supermarché ? [Oui]

Goal :

ApporteCarteCredit(Jean) | ApporteCash(Jean).



```
===== PROOF =====
% ----- Comments from original proof -----
% Proof 1 at 0.01 (+ 0.03) seconds.
% Length of proof is 28.
% Level of proof is 10.
% Maximum clause weight is 0.
% Given clauses 0.

2 (all x (Nourriture(x) -> Item(x))) # label(non_clause). [assumption].
5 (all q (Kilo(q) -> Masse(q))) # label(non_clause). [assumption].
7 (all x all y all z all t (Achete(x,y,z,q,t) & Personne(x) & Item(y) & Supermarche(z) & Masse(q) & Temps(t) -> PossedeMasse(x,y,q) & ApporteMoyenPaiement(x) | ApporteCarteCredit(x) | ApporteCash(x))) # label(non_clause). [assumption].
18 (all x (ApporteMoyenPaiement(x) -> ApporteCarteCredit(x) | ApporteCash(x))) # label(non_clause). [assumption].
23 ApporteCarteCredit(Jean) | ApporteCash(Jean) # label(non_clause) # label(goal). [goal].
24 -Nourriture(x) | Item(x). [clausify(2)].
25 Nourriture(Tomate). [assumption].
33 Temps(hier). [assumption].
34 -Achete(x,y,z,u,w) | -Personne(x) | -Item(y) | -Supermarche(z) | -Masse(u) | -Temps(w) | ApporteMoyenPaiement(x). [clausify(7)].
39 Supermarche(Safeway). [assumption].
44 -Achete(x,y,z,u,hier) | -Personne(x) | -Item(y) | -Supermarche(z) | -Masse(u) | ApporteMoyenPaiement(x). [resolve(34,f,33,a)].
54 -Kilo(x) | Masse(x). [clausify(5)].
55 Kilo(UnKilo). [assumption].
68 -Achete(x,y,Safeway,z,hier) | -Personne(x) | -Item(y) | -Masse(z) | ApporteMoyenPaiement(x). [resolve(44,d,39,a)].
69 -ApporteMoyenPaiement(x) | ApporteCarteCredit(x) | ApporteCash(x). [clausify(18)].
71 -Achete(x,y,Safeway,z,hier) | -Personne(x) | -Item(y) | -Masse(z) | ApporteCarteCredit(x) | ApporteCash(x). [resolve(68,e,69,a)].
73 -ApporteCash(Jean). [deny(23)].
79 -ApporteCarteCredit(Jean). [deny(23)].
81 -Achete(Jean,x,Safeway,y,hier) | -Personne(Jean) | -Item(x) | -Masse(y) | ApporteCarteCredit(Jean). [resolve(71,f,73,a)].
85 Item(Tomate). [resolve(24,a,25,a)].
90 -Achete(Jean,x,Safeway,y,hier) | -Personne(Jean) | -Item(x) | -Masse(y). [resolve(81,e,79,a)].
109 Masse(UnKilo). [resolve(54,a,55,a)].
123 -Achete(Jean,Tomate,Safeway,x,hier) | -Personne(Jean) | -Masse(x). [resolve(98,c,85,a)].
138 Personne(Jean). [assumption].
152 -Achete(Jean,Tomate,Safeway,UnKilo,hier) | -Personne(Jean). [resolve(123,c,109,a)].
171 Achete(Jean,Tomate,Safeway,UnKilo,hier). [assumption].
190 -Achete(Jean,Tomate,Safeway,UnKilo,hier). [resolve(152,b,138,a)].
214 $F. [resolve(190,a,171,a)].

===== end of proof =====
```

i. Jean a-t-il moins d'argent en sortant du supermarché ? [Oui]

Goal :

exists M2(A\_Argent(Jean, M2, aujourd'hui) & MoinsQue(M2, M1)).



```
===== prooftrans =====
Prover9 (32) version Dec-2007, Dec 2007.
Process 5008 was started by alexa on MSI,
Mon Feb 6 17:13:38 2023
The command was "/cygdrive/d/programmes/Prover9-Mace4/bin-win32/prover9".
===== end of head =====

===== end of input =====

===== PROOF =====

% ----- Comments from original proof -----
% Proof 1 at 0.00 (+ 0.05) seconds.
% Length of proof is 16.
% Level of proof is 7.
% Maximum clause weight is 8.
% Given clauses 0.

22 (all x all y all z all q all t1 all t2 all m1 all m2 (Achete(x,y,z,q,t1) & A_Argent(x,m1,t1) & EstAvant(t1,t2) -> A_Argent(x,m2,t2) & MoinsQue(m2,m1
23 (exists M2 (A_Argent(Jean,M2,aujourd'hui) & MoinsQue(M2,M1))) # label(non_clause) # label(goal). [goal].
74 -Achete(x,y,z,u,w) | -A_Argent(x,v5,w) | -EstAvant(w,v6) | A_Argent(x,v7,v6). [clausify(22)].
75 EstAvant(hier,aujourd'hui). [assumption].
76 -Achete(x,y,z,u,w) | -A_Argent(x,v5,w) | -EstAvant(w,v6) | MoinsQue(v7,v5). [clausify(22)].
77 -Achete(x,y,z,u,hier) | -A_Argent(x,w,hier) | MoinsQue(v5,w). [resolve(76,c,75,a)].
78 -A_Argent(Jean,x,aujourd'hui) | -MoinsQue(x,M1). [deny(23)].
136 -Achete(x,y,z,u,hier) | -A_Argent(x,w,hier) | A_Argent(x,v5,aujourd'hui). [resolve(74,c,75,a)].
137 Achete(Jean,Tomate,Safeway,UnKilo,hier). [assumption].
139 -Achete(x,y,z,u,hier) | -A_Argent(x,M1,hier) | -A_Argent(Jean,w,aujourd'hui). [resolve(77,c,78,b)].
163 A_Argent(Jean,M1,hier). [assumption].
164 -A_Argent(Jean,x,hier) | A_Argent(Jean,y,aujourd'hui). [resolve(136,a,137,a)].
165 -A_Argent(Jean,M1,hier) | -A_Argent(Jean,x,aujourd'hui). [resolve(139,a,137,a)].
166 -A_Argent(Jean,x,aujourd'hui). [copy(165),unit_del(a,163)].
170 -A_Argent(Jean,x,hier). [back_unit_del(164),unit_del(b,166)].
171 $F. [resolve(170,a,163,a)].

===== end of proof =====
```

## Conclusion :

Notre base de connaissance suffit pour répondre à l'ensemble des questions du corpus, mais pourrait être complétée et enrichie avec des connaissances plus générales, comme l'ensemble des produits disponibles dans un magasin, plusieurs types de magasins, plusieurs enseignes, plusieurs clients...

Nous avons tout de même voulu rester aussi général que possible dans notre modélisation du problème et avons parfois défini des concepts plus larges et non purement nécessaires pour répondre aux questions.