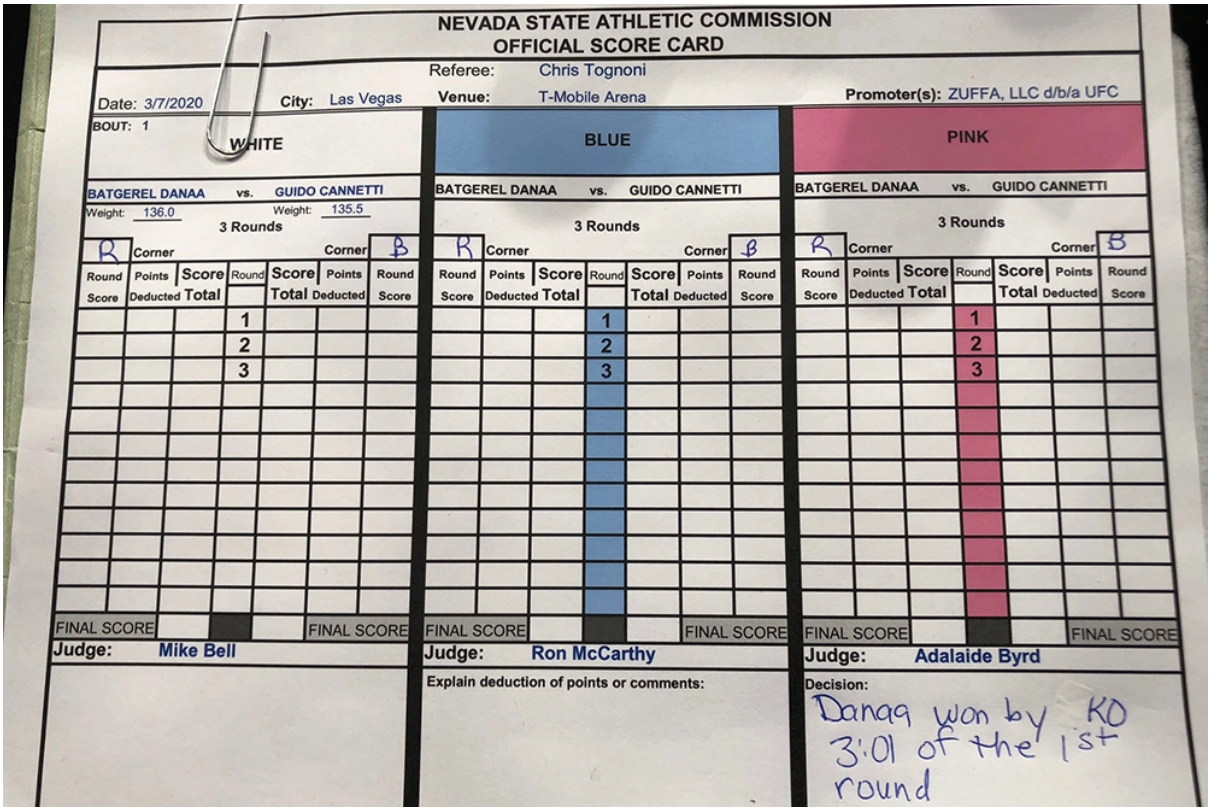
TP3 IA01

Conduite d'expertise d'un SE d'ordre 0+



NEVADA STATE ATHLETIC COMMISSION
OFFICIAL SCORE CARD

Referee: Chris Tognoni
Date: 3/7/2020 City: Las Vegas Venue: T-Mobile Arena Promoter(s): ZUFFA, LLC d/b/a UFC

BOUT: 1

WHITE

BATGEREL DANAA vs. GUIDO CANNETTI
Weight: 136.0 Weight: 135.5
3 Rounds
R Corner Corner B

Round Score	Points Deducted	Score Total	Round	Score Total	Points Deducted	Round Score
			1			
			2			
			3			

FINAL SCORE FINAL SCORE
Judge: Mike Bell

BLUE

BATGEREL DANAA vs. GUIDO CANNETTI
3 Rounds
R Corner Corner B

Round Score	Points Deducted	Score Total	Round	Score Total	Points Deducted	Round Score
			1			
			2			
			3			

FINAL SCORE FINAL SCORE
Judge: Ron McCarthy
Explain deduction of points or comments:

PINK

BATGEREL DANAA vs. GUIDO CANNETTI
3 Rounds
R Corner Corner B

Round Score	Points Deducted	Score Total	Round	Score Total	Points Deducted	Round Score
			1			
			2			
			3			

FINAL SCORE FINAL SCORE
Judge: Adalaide Byrd
Decision:
Danaa won by KO 3:01 of the 1st round

Étudiants : RUBLE Louis
JACAS Nathan

Introduction

L'objectif de ce TP est d'appliquer nos connaissances en systèmes expert au domaine que nous souhaitions, nous avons alors choisi le MMA.
C'est un sport émergent qui est en plein essor. Il combine plusieurs disciplines martiales comme le **kickboxing**, **le karaté**, **la muay thai**, **le jiu-jitsu brésilien** et **le judo**. Les combattants s'affrontent dans un octogone ou dans une cage spécialement conçue pour le MMA.
Les combats sont sous la forme de rounds de 3 à 5 minutes et peuvent se terminer par un KO, une soumission ou un arrêt de l'arbitre. Ce sport est très populaire et a des milliers de fans à travers le monde. Les combattants MMA sont des athlètes très talentueux qui doivent maîtriser plusieurs disciplines martiales pour être compétitifs. Les compétitions sont très spectaculaires et divertissantes, et les combattants sont très respectés pour leur courage et leur détermination.

Dans ce TP, Nathan s'est occupé de la base de connaissance et de règles et Louis s'est occupé de toute la partie algorithmique et de son fonctionnement.

L'utilité d'une IA pour le MMA :

En Octobre 2022, stupeur lors de l'UFC 281, lorsque Sean O' Malley bat Petr Yan sur une décision très controversée, d'où l'utilité d'un système expert pouvant juger un combat de MMA de manière impartiale, basée sur les statistiques du combat.

Sean O'Malley , surpris lorsqu'il apprend qu'il a gagné son combat par décision partagée

L'utilisation de l'intelligence artificielle (IA) dans le MMA peut aider les jurys à prendre des décisions plus impartiales et plus objectives. L'IA peut aider les jurys à évaluer objectivement et plus rapidement les performances des combattants. Elle peut également aider à détecter et à éviter les erreurs humaines et à prendre des décisions plus équitables.

L'IA peut analyser les données des combats et fournir des informations précieuses sur les mouvements des combattants et leurs performances. Cette information peut être utilisée pour aider à évaluer les performances des combattants et à prendre des décisions plus justes et plus impartiales. De plus, l'IA peut également aider à identifier les tactiques et stratégies utilisées par les combattants, ce qui peut aider les jurys à prendre des décisions plus éclairées.

Base de connaissances d'un combat de MMA :

La base de connaissances regroupe des informations sur les combattants et des statistiques pour chaque rounds trouvables sur internet, en voici un exemple :

Exemple :

```
(setq combat '(
(combattant1 'PetrYan) (combattant2 'SeanOmalley)
(Frappes_sign (13 12) (15 16) (17 12))
(Sign_% (13 12) (15 16) (17 12))
(Takedown (2 1) (1 0) (2 0) )
(Temps_CC(1.1 0.2) (1.1 0.2) (1.1 0.2) )
(KO (0 0)(0 0)(0 0))))s
```

Notre base de règle fait le jugement du combat t Round par Round et induit le processus suivant :

1. comparaison des statistiques
2. Avantage dans plusieurs domaines
3. Gain d'un / plusieurs rounds et du combat

Elle est alors ainsi pour l'exemple de 2 rounds :

```
(setq baseDeRegles '(
(R1 (compare 1 'Frappes_sign)(ajouter-fait AvantageTD_C1 avantages) )
(R1_bis (not(compare 1 'Frappes_sign)) (ajouter-fait AvantageTD_C1
avantages))
)
```

L'algorithme choisi :

Nous avons fait le choix d'un algorithme en **chaînage avant**, afin d'exploiter des règles guidés par les faits. C'est-à-dire que lorsque la règle est appliquée, on la désactive juste après.

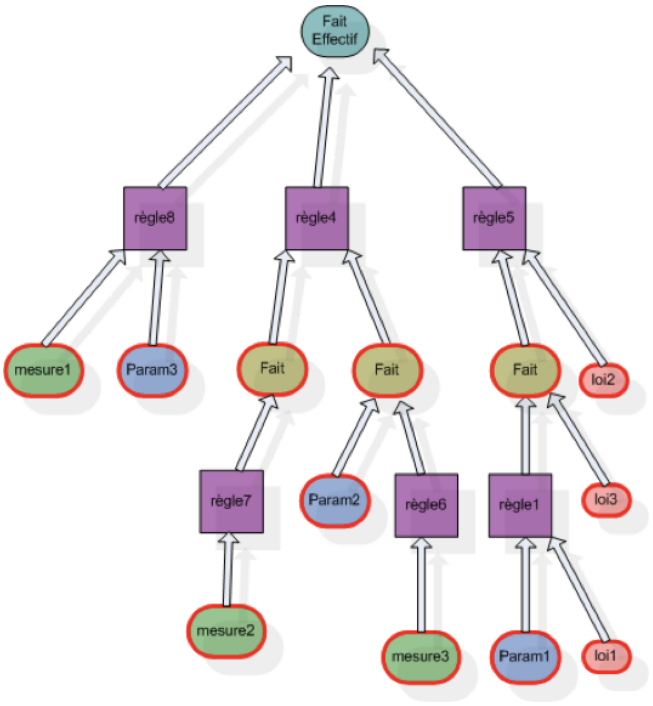
Le moteur d'inférence est lui géré par saturation, c'est à dire qu'il continuera à appliquer les règles disponibles, jusqu'à qu'il n'y en ai plus. En algorithme lisp, cela nous donne cela :

```
(defun chainage-avant (règles faits)
  (while (non-vide règles)
    (let ((règle (premier règles)))
      (if (peut-appliquer règle faits)
          (progn
            (ajouter-fait (conclusion règle) faits)
            (affichage))
            (enlever-règle règles règle))
        (enlever-règle règles règle))))
  faits)
```

Notre algorithme en chaînage est coordonné comme suit à peu de détails près. Vous pourrez retrouver l'ensemble du code en pièce jointe du TP. Nous répétons le chainage avant 3 fois car un combat dure 3 rounds et nous cherchons de savoir qui a été victorieux à chaque round pour avoir la sortie.

Différentes sous-fonctions de l'algorithme :

Les sous-fonctions permettent de factoriser le code en séparant les différentes tâches d'un programme en fonctions distinctes, ce qui rend le code plus lisible et facile à maintenir. En effet, différentes sous fonctions ont été appliquées dans notre algorithme afin de simplifier l'application et la compréhension du code.



- **Peut-on appliquer la règle ou non ?**

```
(defun peut-appliquer (regle) (eval (premisse regle)))
```

Cette fonction est utilisée dans la fonction principale de chaînage avant, afin de vérifier si une règle peut être appliquée.

- **Supprimer une règle**

```
(defun enlever-regle (regle liste) (remove regle liste))
```

Cette fonction permet de supprimer une règle une fois qu'elle a été appliquée sur la base de fait. Cette règle permet de mettre en place le concept de point mort, qui permet de ne pas appliquer la même règle plusieurs fois.

- **Accéder aux prémisses, numéro de la règle, etc..**

```
(defun conclusion-regle (regle) (caddr regle))
```

Cette fonction permet d'accéder facilement aux prémisses, numéro de la règle, etc… C'est une fonction standardisée, dans la même catégorie que les fonctions établies lors du TP1.

- **Accéder à une statistique de la base de faits**

```
(defun get-stats (Round stat)(nth (- Round 1) (cdr(assoc stat
combat))))
```

Cette fonction simplifie la navigation dans la base des faits en permettant d'accéder facilement aux différentes statistiques émises par celle-ci.

- **Définir quel combattant à l'avantage**

```
(defun is-avantage (avantage)
  (subsetp avantage avantages))
```

Cette fonction est utilisée dans la fonction de chaînage avant afin de déterminer quel combattant à l'avantage. Pour cela on prend comme vecteur de comparaison une des caractéristiques de la base de fait.

- **Vérifier si une liste est vide**

```
(defun non-vide (liste)
  (not (null liste)))
```

Cette fonction vérifie simplement si une liste est vide, elle est notamment utilisée pour savoir lorsque toutes les règles ont été traitées.

- **Accéder aux statistiques d'un combattant**

```
((defun get-combattant1 (Round stat)
  (car (get-stats Round stat)))
```

Cette fonction permet aux autres fonctions du programme d'accéder directement aux statistiques précises d'un des deux combattants.

Cette fonction vérifie simplement si une liste est vide, elle est notamment utilisée pour savoir lorsque toutes les règles ont été traitées.

- **Annoncer le gagnant du round et du combat**

```
(defun gain-round (combattant round)
(if (equal combattant 1)
(format t "Le combattant ~@S gagne le round ~@S sur un score de 10-9 ~%" (get-combattant 1) round )
(format t "Le combattant ~@S gagne le round ~@S sur un score de 10-9 ~%" (get-combattant 2) round )))

Cette fonction permet aux autres fonctions du programme d'accéder directement aux statistiques précises d'un des deux combattants.
```

Cette fonction permet d'automatiser l'écriture du message d'annonce de gagnant du round.La liste des sous-fonctions est non exhaustive et vous pourrez retrouver tous les détails dans le code en pièce jointe.

L'algorithme et ses sous-fonctions implantées, nous obtenons alors une bonne sortie :

```
Round  1 !
regle R1 appliquée regle R2 appliquée
regle R3 appliquée
Le combattant 'PETRYAN gagne le round 1 sur un score de 10-9
Round  2 !
regle R1 appliquée
regle R2 appliquée
regle R3 appliquée
Le combattant 'PETRYAN gagne le round 2 sur un score de 10-9
Round  3 !
regle R1_BIS appliquée regle R2_BIS appliquée
regle R3 appliquée
Le combattant 'PETRYAN gagne le round 3 sur un score de 10-9
Le combattant 'PETRYAN gagne le combat !
regle R3_BIS appliquée
Le combattant 'SEANOMALLEY gagne le round 3 sur un score de 10-9
Le combattant 'PETRYAN gagne le combat !
```

Conclusion :

Dans ce TP, nous avons pu formaliser une problématique qui est le souci de Scoring dans les combats de MMA serrés et y appliquer un Système Expert d'ordre 0+, avec un mécanisme de chaînage avant en LISP.

L’avantage de ce Système est qu’en fournissant directement les statistiques du match (même en temps réel), il est possible de rendre compte du supposé victorieux, les règles peuvent aussi être modifiées et interprétables facilement. En effet, en cas de modification des règles, pour par exemple utiliser notre Système expert dans des combats de Box Thaïlandaise, il suffit de changer la base de règle mise en place. Néanmoins, des améliorations sont possibles surtout dans l’édification de la base de règle, en ajoutant par exemple des règles plus précises ou un plus grand nombre de règles. Les règles établies peuvent aussi être biaisées par notre raisonnement, la consultation d’un expert en la matière pour définir le base de règle pourrait-être avenante à la création d’un Système Expert plus équitable.