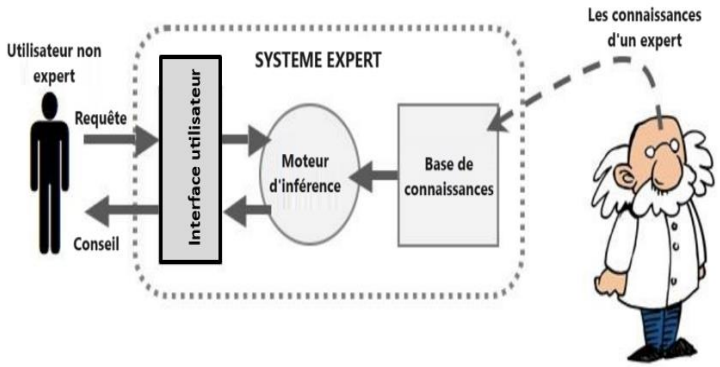
Systèmes basés sur des règles et autres systèmes experts

Dr. NSENGE MPIA HERITIER, Ph.D

Précédemment

- Représentation des règles
  - Antécédents et conséquences
  - Formes et relations des règles
  - Valeurs par défaut et exceptions
  - Les bases de l'inférence et de la correspondance
  - Arbres vers règles et vice versa
- Systèmes experts : Quoi, pourquoi, quand ?
  - L'IA pour résoudre des problèmes dans des domaines spécialisés
  - Composants de base
- Systèmes experts basés sur des règles
  - De loin les plus courants et les plus faciles à développer
  - Systèmes de production (et règles)

```
If x = 1 and y = 1 then class = a
If z = 1 and w = 1 then class = a
Otherwise class = b
```



Plan de la leçon

- Base de connaissances
- Moteur d'inférence
- Autres composants
- Autres types de systèmes experts
- Avantages et inconvénients
- Shells

Systèmes basés sur des règles : Base de connaissances

Rappel : Principes de base des règles

- Représentation des connaissances sous forme de "règles", c'est-à-dire
  - Expressions IF-THEN:

    If <condition (s)> **then** <conclusion>
  - Expressions Condition-Action:

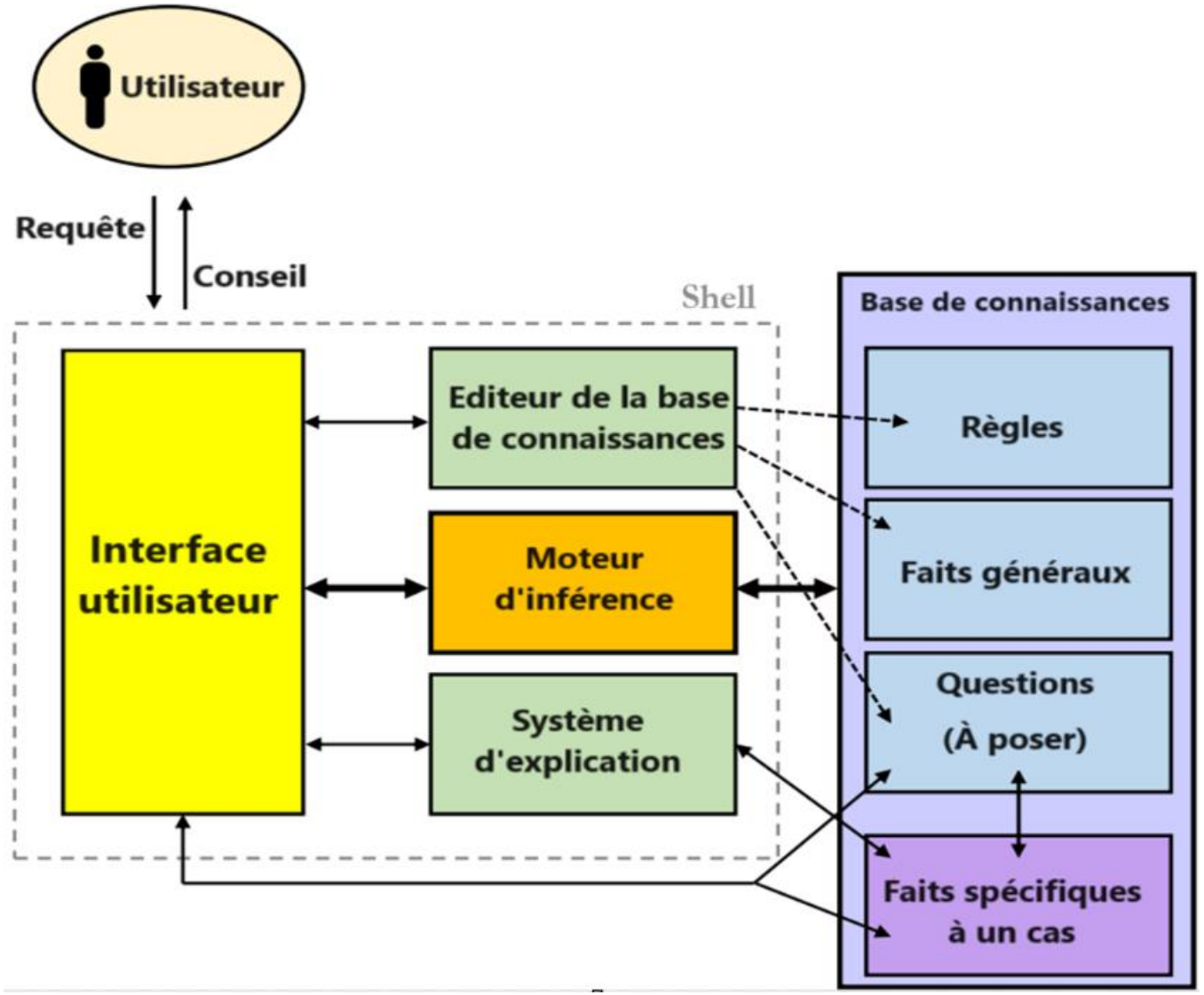
    If <condition (s)> **then** <action>

- Implications effectives de la logique propositionnelle ou du premier ordre
- Exemple simple :
  - SI le feu de circulation est rouge ET que vous vous êtes arrêté, ALORS vous pouvez tourner à droite
- Exemple biomédical :
  - **SI** le patient présente des niveaux élevés de ferritine dans son sang
    ET que le patient présente la mutation Cys282→Tyr dans le gène HFE,
    ALORS, le patient est atteint d'hémochromatose*.

Proposition 1: Vrai
Proposition 2: Vrai
Proposition 3: Vrai

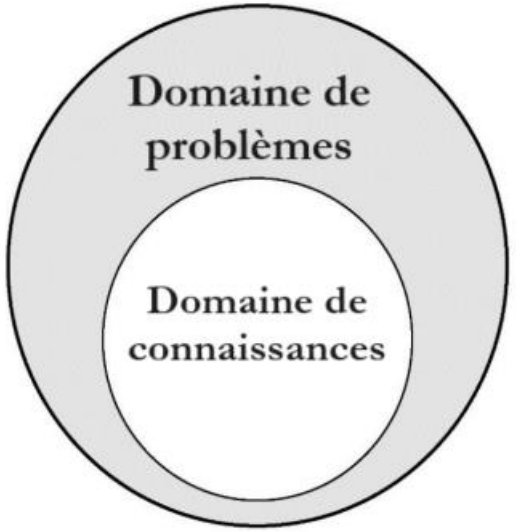
- Condition :
  - Aussi connue comme prémisse, antécédent, variables indépendantes
- Conclusion :
  - Aussi connue comme action, résultat, conséquence, but, variable dépendante

Rappel: Schéma général d'un système expert



Domaine de problèmes vs domaine des connaissances

- Nous commencerons notre discussion sur la base de connaissances en identifiant d'abord la différence entre un problème et un domaine de connaissances.
- En règle générale, les connaissances d'un expert sont spécifiques à un domaine de problèmes.
  - Par exemple : la médecine, la finance, la science ou l'ingénierie.
- Les connaissances de l'expert en matière de résolution de problèmes spécifiques sont appelées domaine de connaissances.
- Pour clarifier, le domaine de problèmes est toujours un surensemble du domaine des connaissances.

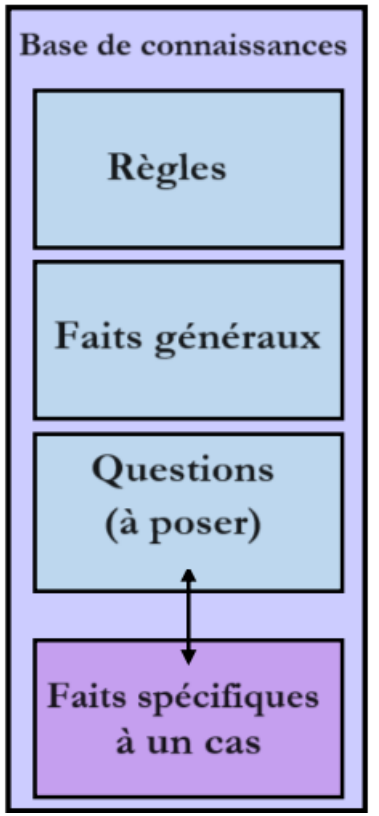


Base de connaissances

- Contient l'expertise que le système peut déployer
- Aussi connue comme: Mémoire de production (dans les systèmes de production)
- Développée de manière unique pour une tâche d'expertise spécifique
- Existe indépendamment du shell du système expert
- Comprend la mémoire à court et à long terme du système expert
  - À long terme :
    - Faits/données généraux et règles "qui ne changent pas".
  - À court terme (c'est-à-dire la mémoire de travail) :
    - Faits spécifiques à un cas ajouté à la base de connaissances par le moteur d'inférence en réponse à une requête

Éléments de la base de connaissances

- Illustration de l'image ci-contre:
  - Mémoire à long terme (cases intérieures en bleu)
  - Mémoire à court terme (case en rose)
- Base de règles :
  - Règles d'implication capturant la connaissance du domaine
- Base de faits :
  - Faits généraux qui ne changent pas
  - Peut faire référence à une base de données d'informations clés ou de valeurs variables.
- Base de questions :
  - Les questions rédigées en langage naturel peuvent être spécifiées ici :
    - Demande à l'utilisateur des faits spécifiques à un cas
    - Le moteur d'inférence utilise les réponses aux questions pour déterminer si une règle donnée est activée
- Faits spécifiques à un cas :
  - Faits fournis par l'utilisateur pour une requête donnée
  - Peuvent être fournis sous forme de réponses à des questions (lorsque peu de faits sont nécessaires)
  - Peuvent être chargés en masse (lorsque de nombreux faits uniques uniques nécessaires/disponibles)



Base de règles

- Si l'on se penche un peu plus sur la base de règles, on s'aperçoit qu'il s'agit généralement d'un ensemble de règles non ordonnées qui ne se répètent pas.
- En fonction de la complexité de la tâche, les bases de règles pour les problèmes du monde réel peuvent contenir de quelques centaines à quelques milliers de règles.
- Si l'on considère que nous n'utilisons que les règles IF, AND, THEN, comment gérer les situations OR ?
- Par exemple :
  - si nous voulons dire que IF A OR B OR C THEN D.
  - Dans le contexte d'un système expert standard basé sur des règles, il suffirait de décomposer cette expression en règles individuelles :
    - IF A THEN D
    - IF B THEN D
    - IF C THEN D.

Base de règles (Cont.)

- Voici ci-dessous un exemple de ce à quoi une règle pourrait ressembler dans la base de règles d'un système expert:

  IF fils_de($fils, $pere)

  THEN pere_de($pere, $fils)

- La règle ci-dessus ressemble beaucoup à ce que nous avons vu dans la logique du premier ordre ou la logique des prédicats.
- Dans ce cas, le fils est le fils du père, alors le père est le père du fils.
- Cette règle simple démontre simplement que la relation père-fils peut être interprétée dans les deux sens.
- Cela semble trivial, mais c'est le genre de choses que vous devez mettre en œuvre dans votre base de règles afin de gérer différents types de situations.

Base de faits

- Examinons de plus près la base de faits.
- Il s'agit généralement d'un ensemble non ordonné de faits qui donnent des valeurs à des propositions ou à des énoncés de prédicats.
- Les faits doivent suivre la syntaxe des règles afin que le moteur d'inférence puisse faire correspondre les faits à appliquer aux règles appropriées.
- En reprenant notre exemple de la règle :

  IF fils_de($fils, $pere)
  THEN pere_de($pere, $fils)

- voici quelques faits que nous pourrions avoir dans notre base de faits:

  fils_de(Mpia, Mbol)

  fils_de(Masamba, Dokolo)

- Remarquez que le fils_de avec deux entrées correspond à la syntaxe de la partie IF de la règle ci-dessus.
- Ces faits indiquent donc que Mpia est le fils de Mbol et que Masamba est le fils de Dokolo.

Base des questions

- C’est un ensemble non ordonné de questions appliquées par le système expert lorsque cela est nécessaire pour donner au système un aspect plus humain à l'interaction avec l'utilisateur.
- Les questions renvoient les valeurs suivantes saisies par l'utilisateur :
  - Oui/Non
  - Vrai/Faux
  - Choix multiple (choisissez une option ou toutes celles qui s'appliquent)
  - Saisie d'une valeur (Int, float, texte)
- Exemple de question :
  - qui_est_le_pere($reponse)

    Qui est votre père ?

    $reponse = selectionnez_1

    1 : Mbol

    2 : Mpia

    3 : Masamba

    4 : Dokolo

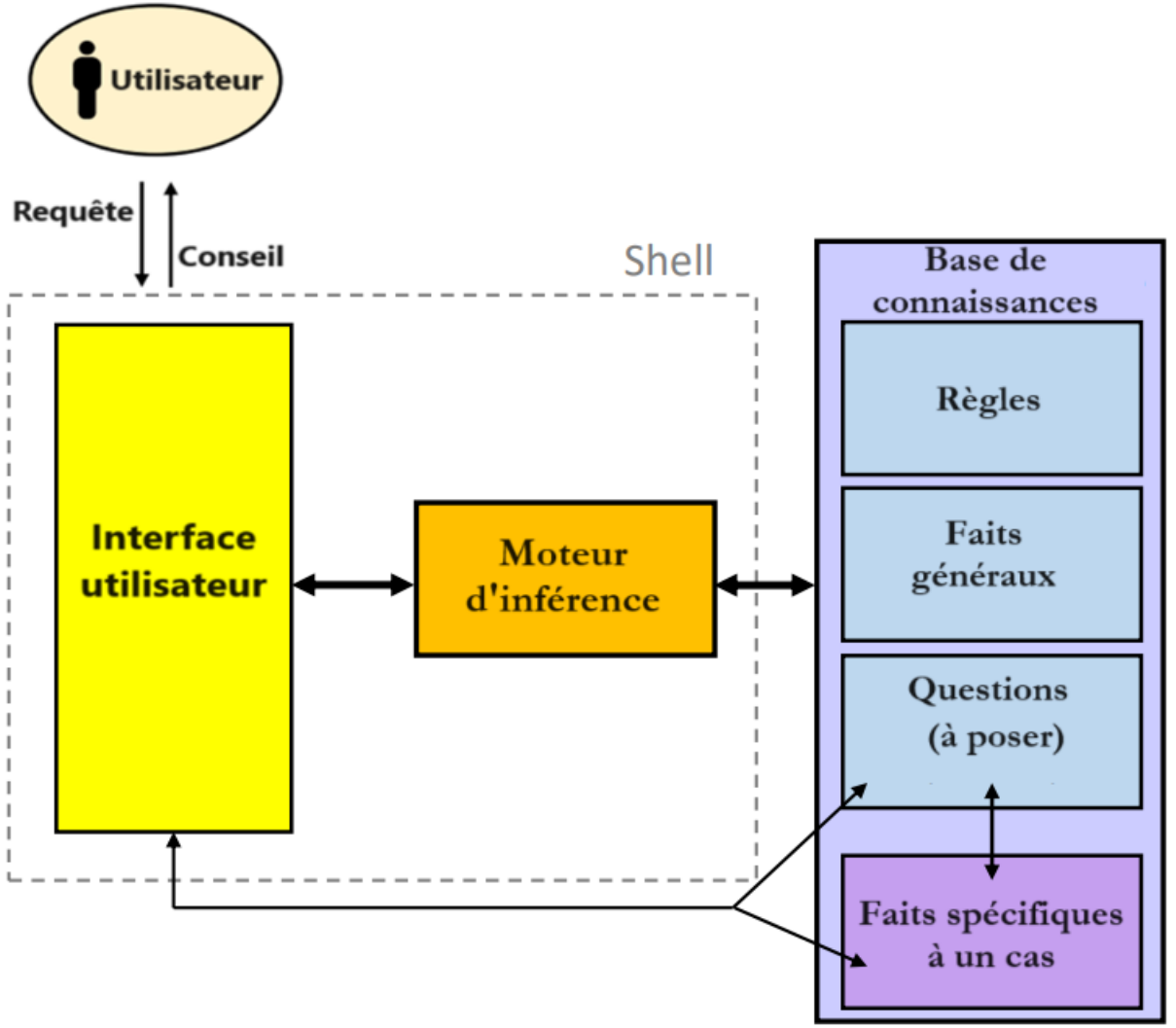
    5 : Je ne sais pas

L’utilisateur va voir ce texte

Moteur d’inférence

Schéma d'un système expert : Moteur d'inférence

- Le moteur d'inférence joue évidemment un rôle central dans le fonctionnement d'un système expert.
- Notez que le moteur d'inférence interagit avec la base de connaissances ainsi qu'avec l'interface utilisateur.



Moteur d'inférence

- Aussi connu comme: Interprète de règles
- C’est la "partie pensante" ou "cerveau" du système
  - Il contrôle la manière dont les règles IF:THEN sont appliquées aux faits.
  - Permet l'acquisition d'informations supplémentaires de la part de l'utilisateur.
    - Invite l’utilisateur à fournir des informations supplémentaires via une interface en langage naturel (éventuellement par le biais de questions)
  - Peut être utilisé pour:
    - parvenir à une hypothèse/solution
    - Affiner une hypothèse/solution
    - Résoudre un conflit entre des hypothèses/solutions actuellement concurrentes
  - Associe des règles à des faits pour générer de nouveaux faits
    - Les nouveaux faits représentent des conclusions sur l'état du domaine compte tenu des observations.
  - Etant donné que nous nous basons sur des systèmes experts basés sur des règles, le moteur d’inférence appliquera toujours soit un chaînage avant ou chaînage arrière comme stratégie d’inférence.

Rappel : Inférence de règles

- Rappelez-vous de notre dernière leçon où nous avons discuté brièvement de l'inférence de règles.

- Il existe généralement deux tâches principales qui impliquent deux régimes de contrôle spécifiques:
  - L'approche de chaînage avant qui est basée sur les données.
  - L'approche par chaînage arrière qui est orientée vers les buts.

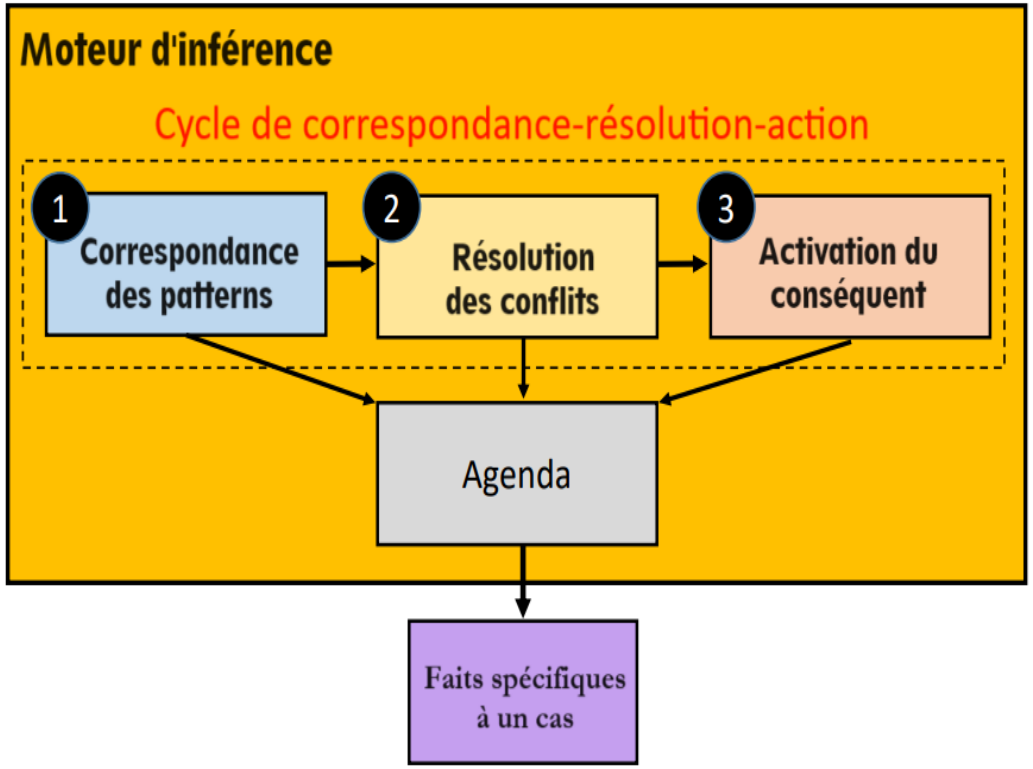
- L'un est-il meilleur que l'autre ?
  - Cela dépend de l'application

En savoir plus sur le chaînage

Chaînage avant	Chaînage arrière
Raisonnement basé sur des données	Raisonnement basé sur des buts
Commence par les faits de la base de connaissances	Commence par un but à atteindre défini par l'utilisateur
Les règles d'inférence sont appliquées en faisant correspondre les FAITS à la partie IF des règles de la base de connaissances	Les règles d'inférence sont appliquées en faisant correspondre l'objectif à la partie THEN des règles de la base de la base de connaissances
Résulte en ce que les nouveaux faits sont ajoutés à la base de connaissances	Résulte en ce que l'on prouve des sous buts
Les règles sont exécutées automatiquement lorsque le chaînage avant est activé pour affirmer de nouveaux faits	Les règles sont directement utilisées pour prouver un but

Composants du moteur d'inférence

- Nous allons maintenant examiner de plus près les composants du moteur d'inférence lui-même.
- En règle générale, le moteur d'inférence suit ce que nous appelons un cycle de correspondance-résolution-action.
- Cycle de correspondance-résolution-action :
  - **Aussi connu comme**: cycle reconnaître agir.
  - Ce cycle *décrit l'exécution des règles par un moteur d'inférence*.
  - Voici une illustration de ce cycle qui comprend :
    - (1) d'abord la correspondance de patterns,
    - (2) ensuite la résolution des conflits,
    - (3) enfin l'activation du conséquent.
    - Les étapes de ce qu'il faut faire ensuite sont stockées dans un agenda et, selon le type d'inférence que nous faisons, le moteur d'inférence peut produire de nouveaux faits spécifiques au cas qui sont ajoutés à la base de connaissances.



Rappel : Correspondance de patterns et de règles

- Rappelez-vous que lors de notre dernière leçon, nous avons brièvement abordé l'idée de la correspondance des patterns ou des règles.
- Cela correspond à l'image illustrée ci-dessus (composant du moteur d'inférence).
- La correspondance détermine les connaissances applicables dans une situation donnée (c'est la partie "recherche" du système).
  - Les règles sont activées ou déclenchées en fonction du type d'inférence ou d'enchaînement utilisé.
    - Chaînage avant :
      - faire correspondre les antécédents des règles à la base de données.
    - Chaînage arrière :
      - faire correspondre les conséquences de la règle aux bases de faits et de règles.
- Toutes les règles correspondantes sont ajoutées à l'agenda comme nous venons de le mentionner.
  - Il s'agit d'une liste de règles classées par ordre de priorité, créée par le moteur d'inférence.

Liaison des variables

- Liaison
  - Un élément important de ce processus de mise en correspondance est le concept de ***liaison des variables***.
  - Ce concept est similaire à celui de la ***liaison logique***.
  - Ici, les variables et les règles acquièrent des valeurs au cours du processus de mise en correspondance.
  - Les antécédents et les conséquences peuvent contenir des variables.
  - Pour l'instant, nous allons supposer que les variables vont ressembler à ($x)
  - En gros, un signe de dollar et un nom de variable.
- Exemple de règle :

  IF fils_de($fils, $pere)
  THEN pere_de($pere, $fils)
- Exemple de faits :

  fils_de(Mpia, Mbol)

  fille_de(Nguele, Mbol)
- Maintenant, pour l'exemple ci-dessus, supposons que nous fassions un chaînage avant.
  - Nous allons donc rechercher tous les faits qui correspondent à l'un des IFs de notre base de règles.
  - Dans ce cas, nous avons fils_de(Mpia, Mbol) qui correspond à if fils_de($son, $pere).
  - Pour que ce fait corresponde correctement à la première partie (IF) de la règle ci-dessus, nous devons lier quelques variables.
  - Ainsi, la valeur Mpia est liée à $fils et Mbol est lié à $pere.
  - Maintenant, lorsque le conséquent de cette règle est activé, la partie "THEN" de la règle acceptera également les valeurs des variables liées.
  - Cela signifie que nous pouvons ajouter un nouveau fait à notre base de connaissances qui devrait être pere_de(Mbol, Mpia). Et comme il s'agit d'un **nouveau fait temporaire**, il est ajouté à nos faits spécifiques, autrement dit, notre mémoire de travail.

Liaison des variables (Cont.)

- Une fois qu'une variable est liée, elle est remplacée par sa liaison chaque fois qu'elle apparaît dans le même pattern ou dans les patterns traités ultérieurement

- Chaque fois que les variables d'un pattern sont remplacées par leurs liaisons, on dit que le pattern est instancié.
  - Vous vous souvenez de la logique ?

- Dans le chaînage avant, les liaisons persistent généralement en tant que nouveaux faits.

- Dans le chaînage arrière, les liaisons peuvent être temporaires car nous testons différentes hypothèses et essayons de prouver, par le chaînage arrière, une chaîne de valeurs de vérité.

Résolution des conflits

- Au cours de la dernière étape de la mise en correspondance, ce processus peut souvent produire plus d'une règle applicable ou de mise en correspondance qui est ajoutée dans l'agenda.
- La question est alors de savoir *laquelle nous examinons en premier ou si nous ne voulons examiner qu'une seule des règles de correspondance*.
- Par exemple :
  - nous ne voulons peut-être prescrire qu'un seul médicament, même si plusieurs médicaments seraient appropriés.
  - En supposant que nous voulons qu'une seule action soit déclenchée/activée.
    - On dit que ces instances de règles de correspondance sont en conflit.
    - Pour cela, nous devons utiliser une sorte de "résolution de conflit" pour n'en choisir qu'une seule à déclencher (Supposons que nous sommes toujours dans le chaînage avant).
- Une façon de résoudre les conflits est de sélectionner la règle ayant la plus haute priorité dans l'agenda.
  - Mais il existe différentes façons d'attribuer une priorité
  - et différentes stratégies peuvent être utilisées pour résoudre ces conflits.

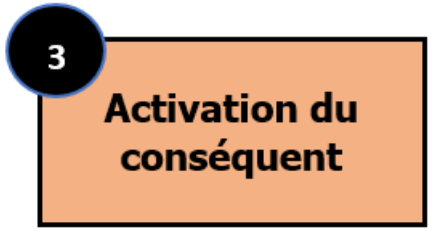
Stratégies de résolution des conflits

- La première stratégie est l'ordre ou l'ordonnancement du fichier source.
  - L'ordre consiste à *choisir la première règle qui correspond dans la base de règles en fonction de l'ordre des règles dans le fichier*.
  - Cependant, cette méthode n'est pas toujours idéale et est même très pénible pour les très grands ensembles de règles, car il faut maintenant se préoccuper de l'ordre de toutes les règles dans le fichier.
- La seconde est la spécificité.
  - Ici, nous choisissons la règle la plus spécifique.
  - Par plus spécifique, nous entendons que si les conditions d'une règle sont un sous-ensemble de celles d'une deuxième règle, nous choisissons la deuxième règle.
  - Exemple :
    - (IF A AND B THEN C) vs. (IF A THEN C) → choisir la première règle parce qu'elle est plus spécifique.
- Ordre lexical ou alphabétique ou dictionnaire.
  - Pour de nombreux problèmes, cette stratégie est quelque peu arbitraire, mais il peut être utile dans certains domaines.

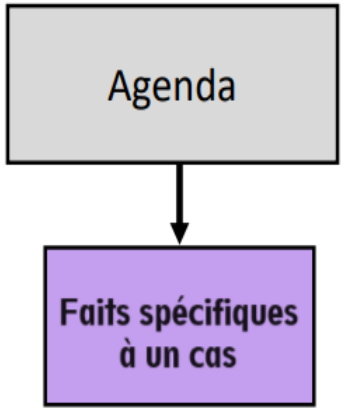
Stratégies de résolution des conflits (Cont.)

- L'importance (c'est-à-dire la saillance) :
  - Choisir la règle la plus prioritaire
  - Les règles sont définies à l'aide d'un *score de priorité*
  - Généralement cette stratégie est déconseillée, car il peut être très difficile de la définir avec précision dans les grandes bases de règles
  - Mais si on veut l'utiliser, elle peut être contrôlée par des "méta-règles".
    - Exemple :
      - IF le patient a des douleurs, THEN prescrire des analgésiques (priorité 10)
      - IF le patient a des douleurs thoraciques, THEN traiter la maladie cardiaque (priorité 100).
        - Si les deux conditions ci-dessus de ces règles sont remplies, c'est la deuxième règle qui est appliquée, car elle a une priorité plus élevée.
- Récence :
  - Choisir la règle la plus récemment ajoutée ou modifiée
  - ou dont les antécédents ont été ajoutés ou modifiés le plus récemment
  - Nécessite des "étiquettes temporelles".
- Refactorisation:
  - Ne pas permettre à la même instance contraignante d'une règle de se déclencher à nouveau

Activation du conséquent



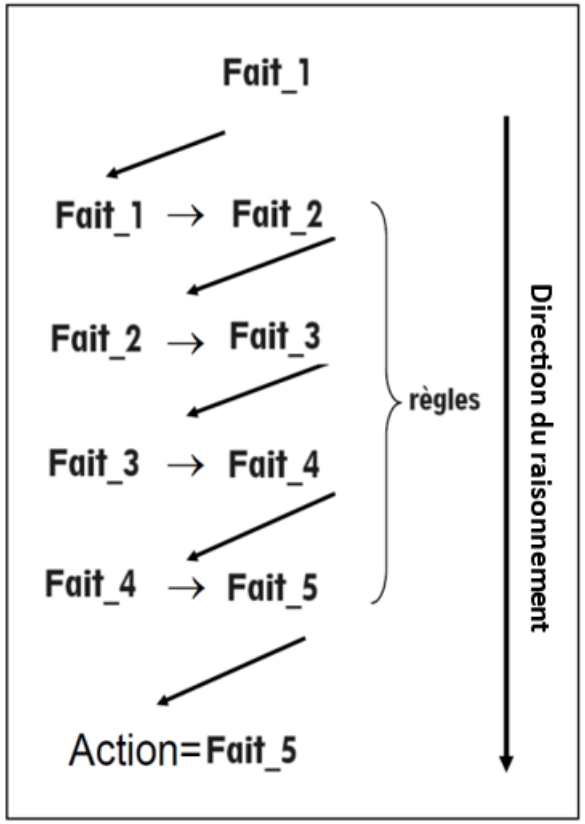
- La dernière partie du moteur d'inférence est l'activation du conséquent.
- Une fois qu'une règle de l'agenda est sélectionnée, l'activation du conséquent consiste à activer ou à déclencher la règle.
  - Si nous procédons à un chaînage avant, nous exécutons la conséquence d'une règle applicable.
  - Cela se traduit par l'ajout d'un nouveau fait à nos faits spécifiques.
- Une fois que cela est fait, nous pouvons retirer la règle de l'agenda puisque nous n'avons plus à la traiter.



- Le cycle complet du moteur d'inférence se termine lorsqu'il n'y a plus de règles dans l'agenda ou lorsqu'il n'y a plus de règles dans l'agenda.

Chaînage avant avec des règles

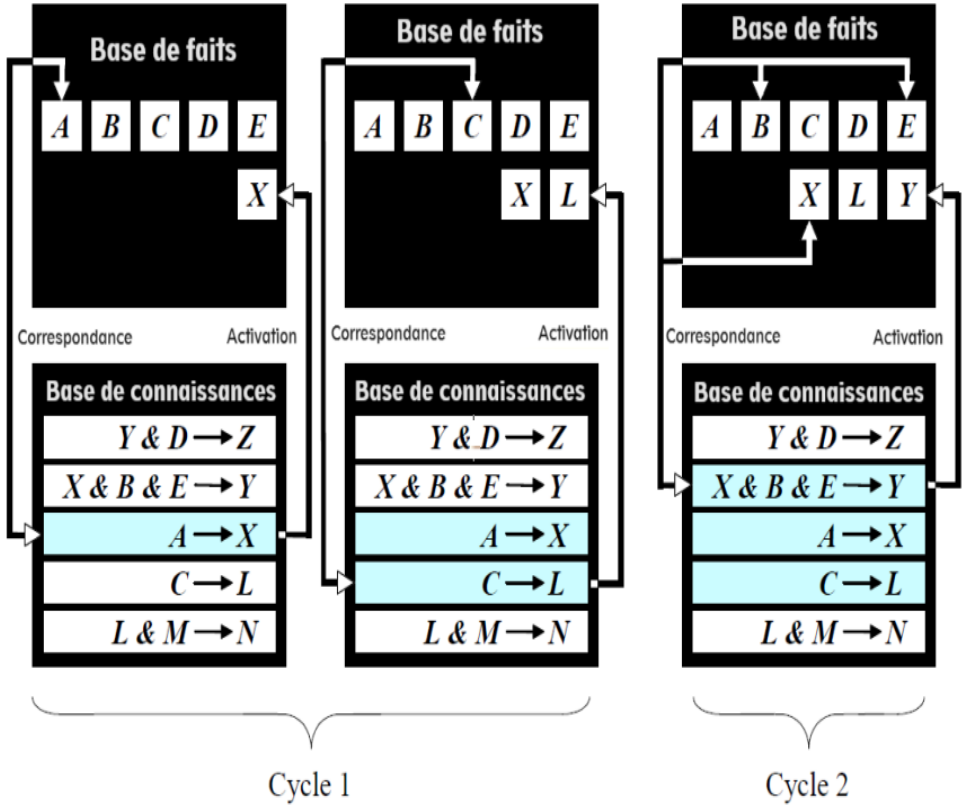
- Revenons à présent sur le fonctionnement du chaînage avant dans le contexte de ces systèmes basés sur des règles.
- Tout cela devrait nous sembler très familier, si l'on se réfère à notre leçon sur le chaînage à partir de la logique.
- En ce qui concerne le chaînage avant:
  - répétez ce qui suit :
    - Appliquer toutes les règles aux faits actuels
    - et chaque déclenchement de règle peut ajouter de nouveaux faits.
  - Procédez ainsi jusqu'à ce qu'aucun nouveau fait ne soit ajouté.
  - Ainsi, dans l'exemple de l'image en haut, nous avons un fait (Fait 1) en haut et nous avons un ensemble de règles dans notre base de connaissances.
  - Le fait 1 satisfait notre première règle, ce qui crée un nouveau fait qui est ajouté à notre base de connaissances.
  - Ce nouveau fait (Fait 2) permet à la règle suivante d'être déclenchée en ajoutant le fait 3.
  - Le fait 3 permet de déclencher le fait 4.
  - Le fait 4 permet au fait 5 d'être déclenché
  - et peut-être que le fait 5 est notre objectif ou une sorte de décision ou d'action qui va avoir lieu.



Guide du chaînage avant

- Voyons maintenant un exemple un peu plus complexe d'inférence par chaînage avant.
- Disons que nous commençons avec le premier composant du cycle 1 représentant une base de faits où A, B, C, D et E sont tous des faits.
- Ces faits peuvent être codés en dur dans le système ou avoir été fournis par l'utilisateur au début de la requête.
- Nous allons maintenant examiner notre premier fait (A) et voir s'il active l'une des règles.
- Dans ce cas, A active ou déclenche X.
- X est donc ajouté (dans la boîte noire à côté de E) à notre base de faits temporaire.
- Nous voyons ensuite que B (dans le deuxième élément du cycle 1) peut déclencher des règles, mais comme vous pouvez le voir dans la base de connaissances, il ne déclenche rien de lui-même.
- Nous examinons ensuite C et, dans ce cas, C déclenche L.
- À ce stade, L est donc ajouté à notre base de faits.
- C'est ainsi que se termine le premier cycle de notre approche de chaînage avant.
- On peut considérer qu'un cycle *consiste à essayer d'ajouter autant de choses que possible sur la base de notre ensemble de faits de départ*.
- Ainsi, à la fin de ce cycle, nous disposons de tous nos faits initiaux, mais aussi de X et de L.
- Nous vérifions maintenant notre base de connaissances pour voir si de nouvelles règles peuvent être déclenchées.
- Dans ce cas, nous connaissons maintenant X, L et E grâce à l'ajout de X lors du dernier cycle.
- C'est donc à ce moment-là que Y est déclenché et ajouté à notre base de faits.
- Cependant, Y vient juste d'être ajouté donc nous ne pouvons pas encore déclencher la règle Y & D->Z et nous ne connaissons pas L et M de la règle L & M->N.
- Voilà qui met fin au deuxième cycle. On va maintenant passer au troisième cycle

L'objectif est de prouver Z

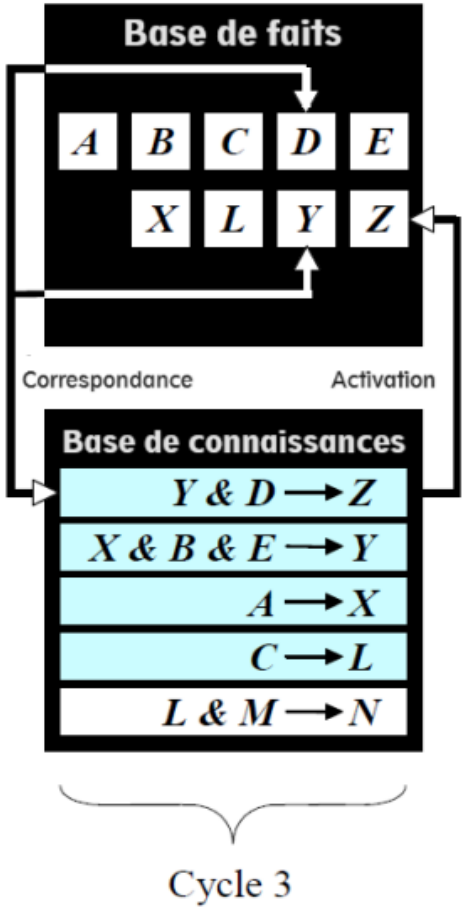


Guide du chaînage avant (Cont.)

- Dans notre troisième cycle, nous savons maintenant comment A, B, C, D, E, X, L et Y.
- À ce stade, nous pouvons maintenant déclencher la règle Y & D -> Z qui nous permet d'inclure Z dans notre base de faits.
- À ce stade, Z peut être retourné à l'utilisateur, soit comme *l'objectif que nous essayons de prouver*, soit comme une sorte de recommandation, et c'est donc Z qui est renvoyé à l'utilisateur.

- Remarquez que nous n'avons pas utilisé toutes les règles de notre base de connaissances.
  - Dans ce cas, nous n'avons jamais eu le fait M et la règle L & M -> N ne peut donc pas être déclenchée.

L'objectif est de prouver Z



Exemple de chaîne d'inférence

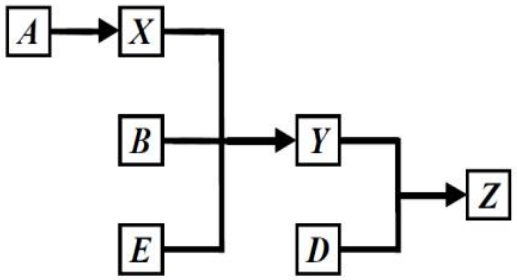
- Nous pouvons illustrer ce processus de chaînage avant par ce que nous appelons une *chaîne d'inférence*.
- Dans cet exemple, nous supposons que A, B et E sont des faits.
- Et nous avons les règles ci-contre.
- A, B et E sont des faits dans la chaîne et nous pouvons voir comment un fait (A) conduit à l'ajout d'un nouveau fait (X).
- Ainsi, par exemple, la règle 3 dit que si A est vrai, alors X est vrai.
- Nous avons donc A, X, B et E comme faits.
- Pour montrer que Y est vrai, nous devons montrer que vous avez X, B et E.
- C'est pourquoi ils constituent le deuxième niveau de la chaîne dessinée, car ils nous permettent d'ajouter Y dans la base de faits.
- Tout ce qu'il nous faut pour obtenir Z, c'est montrer que D est également vrai.

- Peut-être avons-nous conçu ce modèle comme système expert de manière à ce qu'il demande à l'utilisateur, à l'avant-dernier niveau de la chaîne, si D est vrai ou non, afin de déterminer si nous pouvons conclure Z ou si D n'est pas vrai et que nous ne pouvons pas conclure Z.

Règle 1: IF *Y* is true
AND *D* is true
THEN *Z* is true

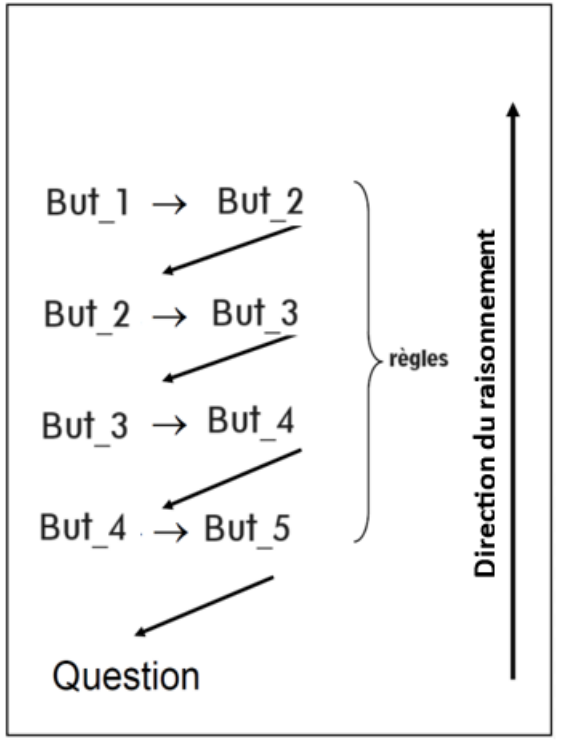
Règle 2: IF *X* is true
AND *B* is true
AND *E* is true
THEN *Y* is true

Règle 3: IF *A* is true
THEN *X* is true



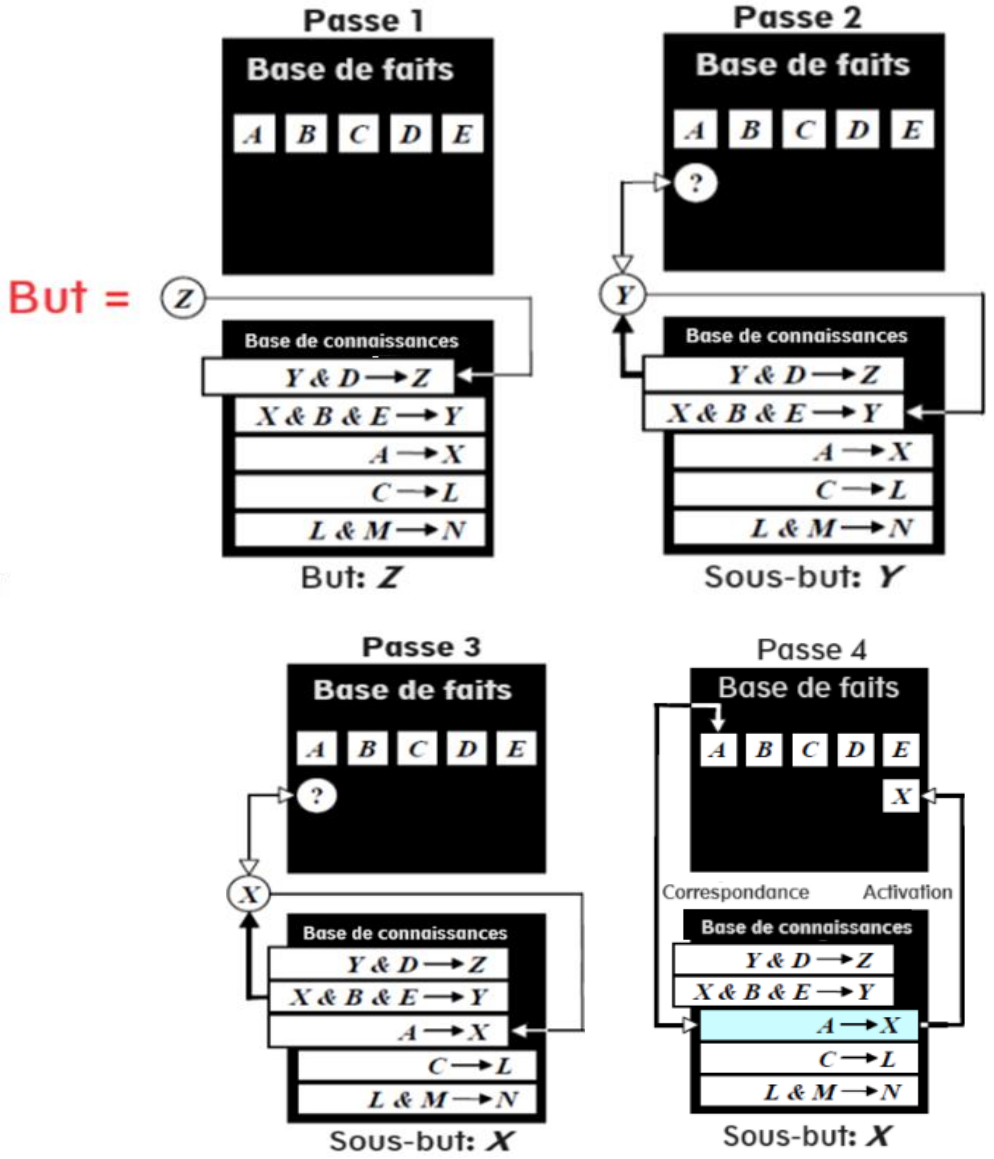
Chaînage arrière avec des règles

- Passons maintenant à des exemples similaires de chaînage arrière.
- Voici la série d'étapes utilisées dans le chaînage arrière fondé sur des règles :
  - Répéter :
    - voir si le résultat se trouve dans la base de fait.
      - Dans l'exemple de l'image ci-contre, le but_5 est la conséquence que nous recherchons.
      - Nous vérifions donc s'il est déjà dans la base de faits ou non ;
    - Si ce n'est pas dans la base de faits:
      - trouver des règles qui ont ce conséquent cible.
      - Ainsi, dans le cas présent, la règle but_4 -> but_5 inclut but_5 et en est le conséquent ;
        - c'est donc ce que nous essayons de prouver
    - de telles règles "s'empilent".
      - Toutes les règles qui ont un conséquent correspondant s'empilent (en d'autres termes, leurs antécédents ou la partie gauche de la règle devient le nouveau conséquent cible (but_4 devient ainsi le sous-but).
      - Nous devons donc maintenant trouver une règle qui corresponde à ce sous-but, que nous trouvons dans but_3 -> but_4.
        - En fait, but_4 fait partie de son conséquent ou nous pourrions également voir si but_4 était déjà dans la base de faits.
  - Faire jusqu'à ce qu'un ensemble connecté de sous-buts soit satisfait par des faits qui sont dans la base de connaissances ou la base de faits.
    - Dans ce cas, vous pouvez voir que le chaînage arrière fonctionne jusqu'à ce que nous trouvions la règle but_1-> but_2 où but_1 se trouve dans notre base de faits.
    - Puisqu'il se trouve dans notre base de faits, la véracité se propage jusqu'à notre question initiale.
    - Bien sûr, il peut arriver qu'il n'y ait pas de sous-buts connectés qui puissent être satisfaits par les faits.



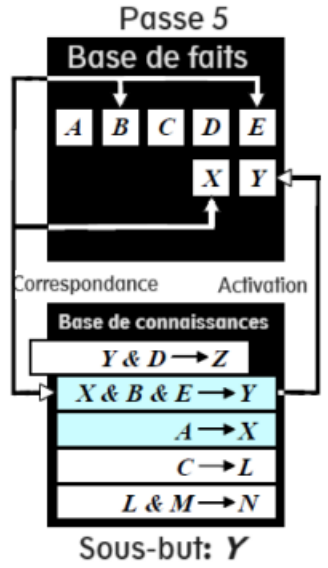
Guide du chaînage arrière

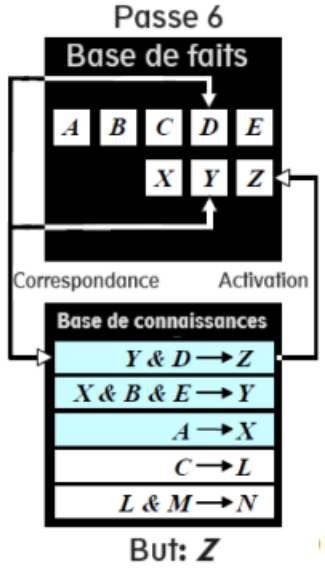
- Ici, nous allons faire une démonstration un peu plus compliquée en utilisant le chaînage arrière.
- Dans ce cas, le ***but*** que nous essayons de prouver est Z.
- Nous avons une base de connaissances avec des règles et notre base de faits.
  - Ces faits peuvent être fournis d'emblée par l'utilisateur ou codés en dur dans la base de connaissances.
- En partant de Z, nous trouvons une règle dans notre base de connaissances (Y & D -> Z) dont la conséquence est Z.
  - À ce stade, nous savons que pour prouver Z, nous devons également prouver Y et D.
  - Nous commencerons par Y.
    - Nous allons vérifier si Y se trouve déjà dans notre base de connaissances, ce qui n'est pas le cas.
    - Et comme ce n'est pas le cas, nous allons chercher une règle dont la conséquence est Y.
  - Donc dans la passe 2, la règle X & B & E -> Y est quelque chose que nous devons également prouver.
  - Continuons sur cette voie pour voir si nous pouvons prouver Y.
  - Pour prouver Y, nous devons prouver X, B et E.
    - B et E font déjà partie de notre base de faits.
    - La seule chose qu'il nous reste à prouver est donc X.
  - Nous devons donc trouver une règle (A->X) dont la conséquence est X.
  - Puisque A est un fait, nous pouvons ajouter X à notre base de faits.
  - Donc maintenant notre base de faits inclut X, B, et E au passe 4.



Guide du chaînage arrière (Cont.)

- A ce stade (passe 5), toutes les conditions de la règle X & B & E -> Y pour prouver Y sont vraies puisqu'elles sont toutes dans notre base de faits.
- Nous pouvons donc ajouter Y à notre base de faits.
- À la passe 6, en revenant à la première règle (Y & D ->Z) dont nous nous sommes occupés, nous savons maintenant que Y est vrai et que D est vrai.
- Nous avons donc prouvé notre but Z. Dans ce cas, nous avons un **exemple réussi** de chaînage arrière prouvant la requête.





But ***Z*** *prouvé!*

Échec de la requête

- Lorsqu'une requête échoue, on peut interpréter l'échec comme une négation.
- En d'autres termes, l'échec de la requête suggère qu'elle n'est pas vraie.
- Cependant, l'échec de la requête peut également signifier que la base de données était incomplète ou mal gérée.
- Il faut donc garder cela à l'esprit:
  - Dans certaines circonstances, il existe une possibilité de boucle infinie lorsqu'on tente de récupérer des informations manquantes.
  - Dans ce cas, il est nécessaire d'appliquer des critères de terminaison qui détectent explicitement ce phénomène et permettent au système d'échouer de manière élégante.
    - En d'autres termes, lorsque vous construisez un système expert, il peut y avoir des situations où vous devez faire attention à ne pas laisser une boucle infinie se produire.

Chaînage mixte

- Certains systèmes experts peuvent appliquer un chaînage avant et arrière simultanément

- La stratégie consiste pour le système à enchaîner dans un sens, puis de passer à l'autre sens, de manière à ce que
  - le diagnostic soit trouvé avec une efficacité maximale
  - le comportement du système soit perçu comme humain.

Efficacité du moteur d'inférence

- En ce qui concerne l'efficacité d'un moteur d'inférence, on peut se demander comment réduire le coût du processus de mise en ***correspondance***.
- L'une des solutions consiste à utiliser ce que l'on appelle:
  - les algorithmes de Markov:
    - Ici les règles les plus prioritaires sont appliquées en premier
    - Ceci permet une exécution plus efficace des systèmes de production
    - Mais cette approche n'est pas encore assez efficace pour les systèmes experts avec de grands ensembles de règles
  - Sauvegarder les informations sur les correspondances intermédiaires (algorithme RETE) :
    - L'idée ici est de faire en sorte que la correspondance ne soit pas répété à l'infini
      - On partage les informations sur les correspondances intermédiaires entre les règles
      - On recalcule les informations intermédiaires en cas de changement
      - Cela nécessite une mémoire supplémentaire pour les informations sur les correspondances intermédiaires
      - S'adapte bien aux grands ensembles de règles
  - Recalculer les correspondances pour les règles affectées par les modifications (algo TREAT) :
    - Vérifier les modifications par rapport aux règles de l'ensemble de règles conflictuelles
    - Moins de mémoire que Rete
    - Ne s'adapte pas aussi bien aux grands ensembles de règles.

  - Faire un usage intensif du hachage (mappage entre la mémoire et les tests/règles)

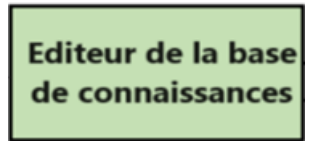
Autres composants du système expert

Interface utilisateur (UI)

- Le moyen d'interaction entre un utilisateur à la recherche d'une solution et un système expert
  - Permet d'écrire les faits/règles
  - Permet d'interroger le système
  - Permet de répondre à des questions
  - Se voir présenter une ou des solutions et des explications
- L'interface utilisateur est directement ou indirectement connectée à toutes les parties du système expert
- L'interface utilisateur peut être:
  - Question/réponse simple
  - Il est basé sur des menus, par exemple une interface utilisateur graphique (IUG)
  - Prise en charge du traitement du langage naturel (NLP)
    - Pour qu'elle puisse être facilement utilisée par une personne connaissant bien le domaine d'application, mais pas nécessairement les systèmes d'intelligence artificielle.

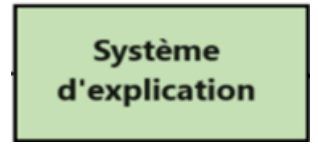
Éditeur de la base de connaissances

- **Aussi connu comme**: dispositif d'acquisition de connaissances
- Il fournit en principe à l'utilisateur un moyen automatique d'introduire des connaissances dans le système, sans qu'il soit nécessaire de disposer d'une expertise explicite en matière de codage.
- en fin de compte, il facilite:
  - l'intégration de nouvelles connaissances
  - l'édition des connaissances existantes



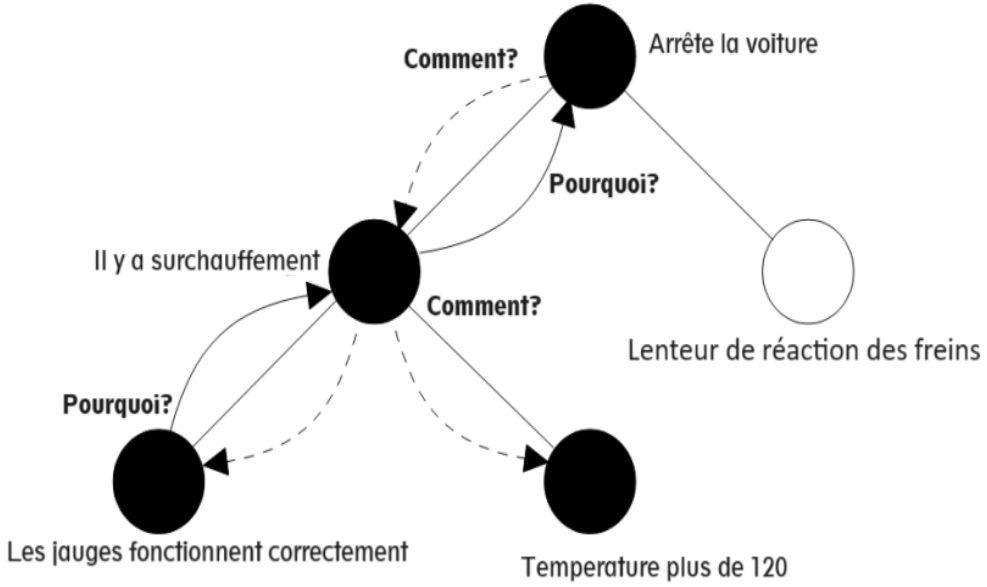
Système d'explication

- Aussi connu comme: Justificateur
- Il explique à l'utilisateur le raisonnement du système expert
  - par exemple, comment une conclusion particulière a été obtenue (chaîne de raisonnement)
    - Cela peut mettre en évidence les règles qui ont été activées et les faits qui ont été déduits en cours de route.
  - Ils peuvent également être utilisés pour répondre à la question suivante : pourquoi n'est-on pas parvenu à une autre conclusion ?
  - ou pourquoi un fait spécifique est nécessaire si le système le demande à l'utilisateur ?
  - Ils peuvent également être utilisés pour répondre à la question de savoir pourquoi un fait n'a pas été utilisé ou pourquoi il n'a pas fait partie d'une inférence.
- Un système d'explication est évidemment important dans des situations telles que celle où un *médecin doit comprendre le raisonnement d'une recommandation*.
- La composante "système d'explication" du système expert constitue un avantage majeur des systèmes experts par rapport aux systèmes basés sur les réseaux neuronaux :
  - Malgré le succès notable des systèmes de réseaux neuronaux dans de nombreux travaux actuels, les systèmes experts resteront probablement pendant longtemps la technique d'IA de prédilection:
    - Des applications critiques telles que le diagnostic médical
    - Lorsque, pour des raisons juridiques, une défense verbale d'une décision doit être disponible sur demande.



Exemple de système d'explication

- Voici un exemple approximatif de la manière dont un système d'explication pourrait être mis en place à l'aide d'un *système de production de contrôle automobile*.
- Dans le nœud noir au-dessus, nous avons l'action "arrêter la voiture" et cette décision pourrait être prise sur la base d'informations relatives à la surchauffe et à la lenteur de réaction des freins.
- De plus, le fait qu'il y ait une surchauffe peut provenir d'une combinaison entre le fait de savoir que la jauge de chaleur de la voiture fonctionne correctement et que la température qu'elle indique est supérieure à 120.
- Un utilisateur pourrait demander pourquoi nous avons pris la décision d'arrêter la voiture ?
- Il peut répondre à cette question en revenant au fait qu'il y a une surchauffe.
- La question "comment" demande comment nous avons pris la décision d'arrêter la voiture.
  - Là encore, nous pouvons revenir à la partie précédente de la chaîne et identifier que le comment provient du fait qu'il y a une surchauffe.
- Nous pouvons suivre cette chaîne jusqu'aux faits initiaux qui forment la base de la chaîne, ce qui permet aux utilisateurs d'obtenir un ensemble très riche d'explications sur la manière dont une décision a pu être prise.



Autres types de systèmes experts

Autres types des systèmes experts

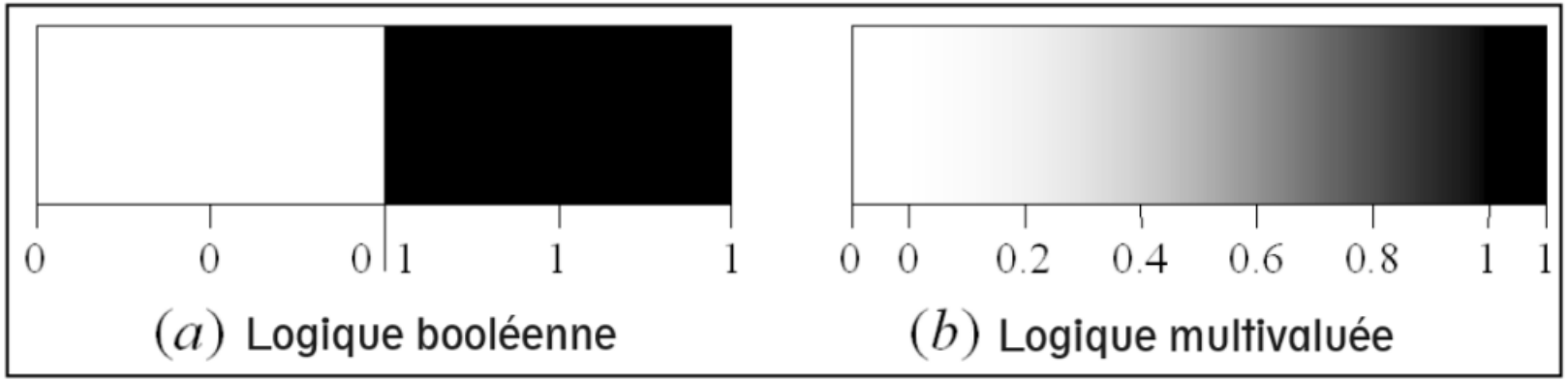
- Nous voulons vous illustrer quelques autres types des systèmes experts, juste pour que vous sachiez ce qui existe.
- Voici un tableau très complexe, que je vais détailler, qui illustre certains des principaux types de systèmes experts existants.
- Dans la deuxième colonne, nous avons le système expert basé sur des règles que nous venons d'aborder

- Fuzzy système expert
  - Applique la logique floue avec des règles
- Système expert basé sur le cadre
  - Utilise la représentation du cadre
- Système expert hybride
  - Système expert neuronal
    - Règles + réseaux neuronaux
  - Système expert neuronal flou
    - Règles floues + réseaux neuronaux

Paramètres	Système expert basé sur les règles	Système expert flou	Système expert basé sur les cadres	Système expert neuronal	Système expert neuro-flou
Représentation des connaissances	Règles IF-THEN dans la base de connaissances	Sur la base du degré d'appartenance à l'aide de la logique floue	Les connaissances dans les cadres en utilisant une structure hiérarchique	Règles IF-THEN dans la base de connaissances neuronale	Dans une variable linguistique et en utilisant les règles IF-THEN dans une structure floue
Capacité d'apprentissage	Ne peut pas apprendre seul et mettre à jour la base de connaissances existante	Manque de capacité à apprendre de l'expérience	Ne peut pas apprendre et s'adapter à un nouvel environnement	Le réseau neuronal peut apprendre, mais l'apprentissage est un processus de boîte noire pour l'utilisateur	Possède une capacité d'apprentissage grâce au réseau neuronal qui en est l'un des composants
Tolérance d'incertitude	Difficulté de mesurer l'incertitude	Le raisonnement probabiliste permet de faire face à l'incertitude	Impossible en raison de la structure des connaissances	Raisonnement approximatif	Utilisation de la valeur script et du raisonnement probabiliste
Tolérance d'imprécision	Très faible, nécessite des informations précises	Élevé, car la logique floue peut gérer l'imprécision	Des données très faibles et imprécises peuvent conduire à des résultats erronés	Le réseau neuronal peut traiter des données imprécises	Très élevé en raison de la combinaison du réseau neuronal et de la logique floue
Facilité d'explication	Oui	Oui, l'utilisation de variables linguistiques	Oui, bonne explication du résultat	Oui, en raison de la composante basée sur des règles	Oui, très efficace
Moteur d'inférence	Traite les règles et déduit la conclusion	Règles de traitement utilisant la fuzzification et la défuzzification	Recherche des buts à l'aide de méthodes et de démons	Réseau neuronal à trois couches combiné à l'extraction de règles	Processus d'inférence floue utilisant la fuzzification et la défuzzification
Mise à jour des connaissances	Difficulté d'ajouter de nouvelles règles	Difficile d'introduire de nouvelles variables linguistiques dans la structure existante	Pas possible	Oui, avec de nouvelles connaissances, les composants du réseau neuronal peuvent apprendre	Oui, de nouvelles variables linguistiques peuvent être ajoutées à la structure de connaissance existante
Maintenabilité	Moyennement difficile	Très difficile	Facile	Facile	Difficile à cause de la composante floue
Adaptabilité	Non	Non	Non	Oui	Oui
Temps de calcul	Très élevé en raison du traitement de chaque règle	Le temps de traitement est réduit par rapport au système expert à base de règles	Les connaissances stockées dans les cadres peuvent être traitées très rapidement	L'apprentissage d'un réseau neuronal prend du temps, mais une fois que les connaissances sont stockées dans les neurones, le temps de traitement des règles est rapide	L'apprentissage d'un réseau neuronal prend du temps, mais le temps de traitement est considérablement réduit
Structure des connaissances	Adhoc, ne peut pas comprendre la dépendance logique des règles	Assez peu structuré	Hautement structuré	Structuré mais stocké dans les neurones	Moyennement structuré

La logique floue en bref

- Logique qui décrit le flou
  - c'est-à-dire qui décrit l'imprécision
- Les décisions sont basées sur un "degré d'appartenance" plutôt que sur un critère booléen, oui/non.
  - Ici toutes les choses sont décrites sur une "échelle mobile".
  - qui suggère la possibilité qu'une énoncé donné soit vrai ou faux
- Ainsi, dans l'image illustrée ci-bas, au lieu d'un oui ou d'un non booléen, nous avons une zone d'échelle quantitative grise (b) indiquant le degré d'appartenance à un groupe ou à un autre.



Ensembles flous

- Dans l'expression $\mu A(X)$, notre terme flou est A et A peut avoir un degré d'appartenance X qui peut varier de 0 à 1 ;
  - où le terme A est 1 si X est totalement dans A et A est 0 si X n'est pas du tout dans A.
  - Et plus communément, A est entre 0 et 1 si X est partiellement dans A.
- Une façon de capturer cette idée est de penser à l'appartenance à des groupes définis comme étant Court, Moyen ou Elancé
  - Nous pourrions dire que toute personne de plus de 200 centimètres est clairement une personne qui est considérée comme élancée. En d'autres termes, elle fait totalement partie du groupe A, ou nous dirons ici que A est élancé.
  - Cependant, au fur et à mesure que l'on descend en taille, il est moins certain que l'on puisse qualifier quelqu'un d'élancé ou de moyen,
  - mais à 170 centimètres, certains pourraient considérer cette personne comme élancée tandis que d'autres pourraient la considérer comme moyennement élancée.
  - Il s'agit donc d'essayer de saisir le degré d'appartenance à un groupe ou à un autre sur la base de la probabilité de certitude d'appartenir à un groupe ou à un autre.

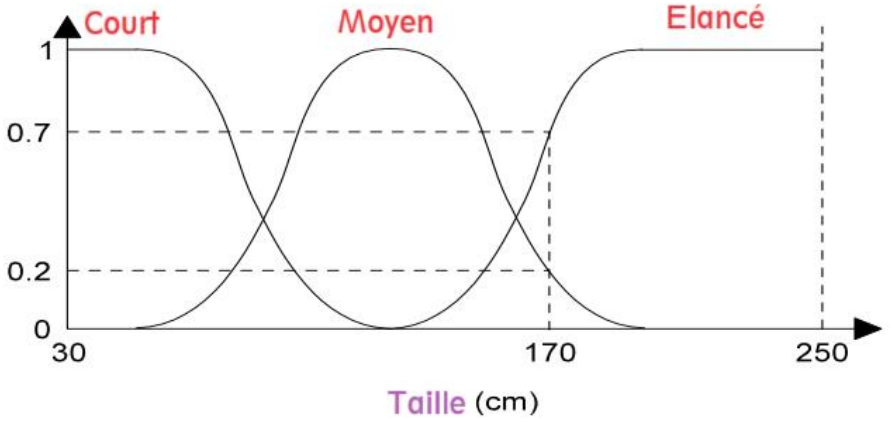
$\mu A(x) : \quad X \longrightarrow [0,1],$

Où

$\mu A(x) = 1$ si X est totalement dans A

$\mu A(x) = 0$ si X n'est pas dans A

$0 < \mu A(x) < 1$ si X est partiellement dans A



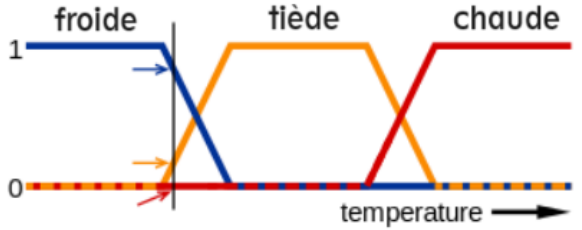
Règles floues

- Règle classique IF:THEN (logique binaire)
  - Règle 1
    - IF température > 30
    - THEN vitesse du ventilateur est de 5
  - Règle 2
    - IF température < 10
    - THEN vitesse du ventilateur est de 1
- Règle IF:THEN (logique floue)
  - Règle 1
    - IF température est élevée
    - THEN vitesse du ventilateur rapide
  - Règle 2
    - IF température est basse
    - THEN vitesse du ventilateur est faible

  - Il faut noter que les règles floues sont définies de façon qualitative plutôt que quantitative précise

Inférence floue

- En ce qui concerne l'inférence floue, il existe une correspondance entre une entrée/input donnée et une sortie/output en utilisant la théorie des ensembles flous.
- Cela se fait en quatre étapes.
  1. La fuzzification.
     - Ici, étant donné certaines entrées nettes, nous déterminons le degré d'appartenance de ces entrées à chaque ensemble flou.
     - Ainsi, une entrée nette peut être la température exacte (c'est-à-dire froide, tiède, chaude), mais une température donnée peut appartenir à un degré différent (comme dans l'image ci-contre) à différents ensembles.
       - Peut-être en fonction des différents points de vue des gens sur ce qui est froid, chaud ou tiède.
  2. Évaluation des règles.
     - Ici, nous prenons les entrées fuzzifiées et les appliquons aux antécédents.
  3. Agrégation des résultats de la règle.
     - Nous prenons les fonctions d'appartenance de toutes les conséquences des règles et les combinons en un seul ensemble flou.
       - Cela permet d'unifier les résultats de toutes les règles.
  4. Défuzzification.
     - Ici, nous transformons la sortie de la recherche en un nombre ou une valeur précise.



- Notamment, lorsqu'il s'agit de systèmes de règles floues ou booléennes,
  - un système booléen qui utilise AND utiliserait MINIMUM dans un système flou.
  - un système booléen qui utilise un OU deviendrait MAX dans un système flou
  - et NOT X dans un système booléen deviendrait 1-X dans un système flou.

Booléen	Flou
AND(x,y)	MIN(x,y)
OR(x,y)	MAX(x,y)
NOT(x)	1 – x

Système expert flou

- Quand pouvons-nous utiliser un système expert flou ?
- Lorsque nous voulons *exprimer des connaissances d'expert* qui utilisent des mots vagues et peu clairs tels que "légèrement surchargé", "fortement réduit", "modérément difficile", "pas si vieux", "très grand" , etc.

- Les systèmes flous sont
  - faciles à développer et à déboguer
  - faciles à comprendre
  - faciles et peu coûteux à entretenir

Systèmes experts basés sur des cadres

- Les cadres nous fournissent un moyen structuré de représenter la connaissance sous forme de hiérarchie (héritage),
- mais que devons-nous faire lorsque nous voulons valider et manipuler les connaissances ?
  - La réponse à cette question réside dans les méthodes et les daemon
- Une méthode est une procédure associée à un attribut de cadre qui est exécutée chaque fois qu'elle est demandé
  - La méthode est représentée par une série de commandes similaires à une macro dans Microsoft Excel
- En général, les daemons ont une structure IF-THEN.
  - Il est exécuté chaque fois qu'un attribut de l'instruction IF du daemon change de valeur.
- Inférence :
  - Les règles ont pour rôle d'évaluer les informations contenues dans les cadres.
  - Les objectifs sont établis soit dans une méthode, soit dans un daemon.

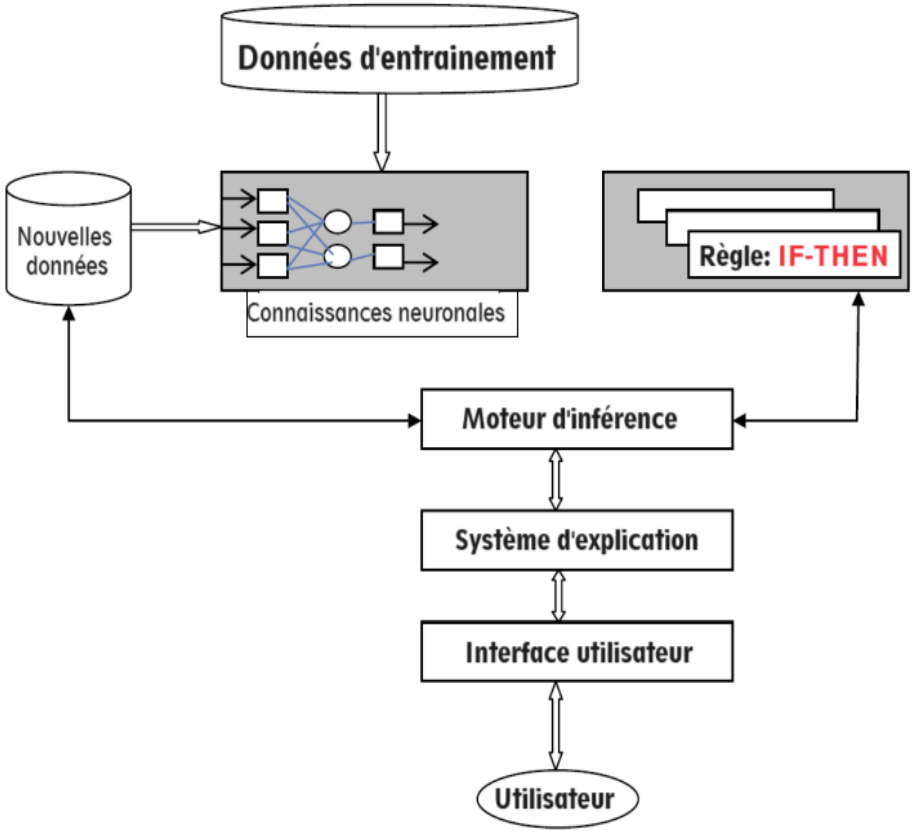
Systèmes hybrides

- Il existe de nombreuses façons créatives de créer des systèmes experts hybrides.
  - Ils peuvent faire appel à n'importe quelle combinaison de représentations ou intégrer des méthodes de déduction et d'induction.
  - Cependant, on ne peut pas s'attendre à ce que toutes les combinaisons fonctionnent.
- Considérons:
  - un instant que nous disposons d'un système expert qui a été mis en place pour stocker et utiliser l'expertise humaine existante.
    - Puisqu'il est basé sur l'expertise humaine, il peut bien sûr être défectueux.
  - Et si nous pouvions mettre à jour notre système grâce à une expérience continue ?
    - C'est là que l'apprentissage automatique (apprentissage inductif) pourrait jouer un rôle dans les systèmes experts en trainant (formant) un décideur à l'aide d'exemples plutôt qu'en le préprogrammant.
    - Par conséquent, les méthodes d'apprentissage automatique ont potentiellement la capacité d'obtenir de meilleurs résultats que l'expertise humaine.
- L'une des façons de combiner l'apprentissage automatique et les systèmes experts est donc d'hybrider les systèmes experts avec des systèmes sous-symboliques, en d'autres termes, des éléments tels que les réseaux neuronaux.
  - Dans ce cas, les réseaux neuronaux pourraient être utilisés pour prétraiter les données et les transmettre ensuite à des règles,
  - ou ils pourraient être utilisés pour apprendre de meilleures règles

Systèmes hybrides : Système expert neuronal

- Il s'agit d'un exemple de système hybride, plus précisément d'un système expert à réseau neuronal ou d'un système basé sur un réseau neuronal.
  - Nous disposons de données d'apprentissage,
  - nous avons formé un réseau neuronal et,
  - à droite de l'image et dans le reste du flux de l'image, nous avons notre système expert.
- Toutefois, dans ce cas, les règles peuvent être extraites des poids entre les nœuds d'un réseau neuronal.
- L'utilisation d'un réseau neuronal peut permettre au système de mieux traiter les données bruyantes et complètes grâce à sa capacité de généralisation.
- Étant donné que nous nous éloignons à présent du raisonnement déductif, nous ne pouvons réellement dire qu'un système comme celui-ci est une approximation du raisonnement, car nous nous inspirons à la fois des méthodes de raisonnement par déduction et par induction.

Structure basique d'un système expert neuronal



Systèmes hybrides : Système expert neuro-flou

- Alors que les *réseaux neuronaux sont des structures informatiques de bas niveau* qui donnent de bons résultats lorsqu'ils traitent des données brutes, la logique floue traite le raisonnement à un niveau plus élevé, en utilisant des informations linguistiques acquises auprès d'experts du domaine
- Par conséquent, les réseaux neuronaux deviennent plus transparents, tandis que les systèmes flous deviennent capables d'apprendre grâce à leur intégration.
- Chaque couche du système neuro-flou est associée à une étape particulière du processus d'inférence floue.

Avantages et inconvénients des systèmes experts

Autres avantages des systèmes experts

- Représentation naturelle des connaissances :
  - en d'autres termes, vous obtenez des règles IF:THEN lisibles par l'homme qui constituent la représentation
    - Elles ont une structure uniforme
    - Le système lui-même est fondamentalement auto-documenté
    - et chaque règle est un élément de connaissance indépendant
- Selon le système expert que vous utilisez, ils peuvent traiter des connaissances incomplètes et incertaines :
  - Nous reviendrons sur les connaissances incertaines dans la leçon ultérieure.
- Les systèmes experts peuvent être développés et mis à jour par étapes
- Ils facilitent le transfert de connaissances vers des lieux éloignés
- En outre, les systèmes experts fournissent des réponses cohérentes pour les décisions, les processus et les tâches répétitifs.
- Ils sont capables de combiner plusieurs intelligences humaines expertes grâce à la collaboration de leur développement.
- Ils permettent de centraliser le processus de prise de décision.

Inconvénients des systèmes experts

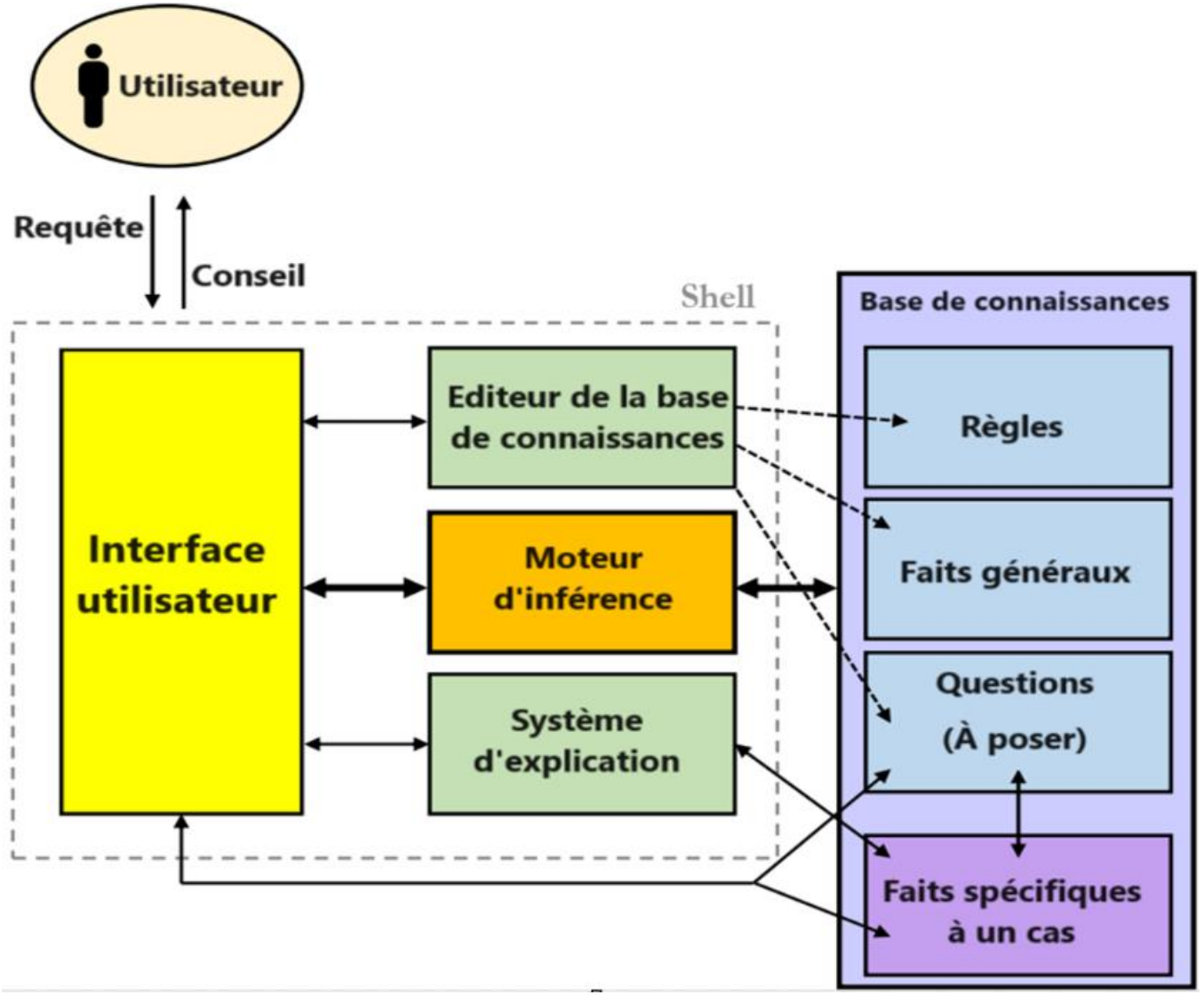
- Peut ne pas arriver à une conclusion ou ne pas savoir si une conclusion est possible
- Ne fonctionnent bien que dans un domaine étroit de connaissances
- En général, incapables d'apprendre ou d'être créatifs
  - Les méthodes fondées sur les connaissances ne s'auto-modifient pas ou ne s'améliorent pas (risqué).
- La recherche de règles peut être longue et lente
  - Peut être inadaptée aux applications en temps réel
- Relations opaques entre les règles :
  - Si les règles individuelles peuvent être interprétées, les interactions logiques entre une grande base de règles peuvent être opaques.
    - Absence de structure hiérarchique
  - Il peut être difficile de comprendre comment les règles individuelles servent la stratégie globale.
- Généralement, les règles sont élaborées sur la base de l'expérience passée des experts, de leur instinct et de leurs intuitions, ainsi que d'une approche fondée sur les essais et les erreurs.
  - Peut donc faire des suggestions incorrectes

Inconvénients des systèmes experts (Cont.)

- Impossibilité de raisonner sur des situations connexes ou nouvelles
- L'acquisition de connaissances peut constituer un goulot d'étranglement pour la mise en œuvre
- Manque de confiance
  - Les utilisateurs peuvent ne pas vouloir laisser les décisions critiques aux machines
- Pas de connaissances "de bon sens".
  - Tout doit être explicitement programmé
- Manque de flexibilité et de capacité à s'adapter à des environnements changeants
- Les systèmes experts ne conviennent pas à tous les types de domaines et de tâches
  - Ils ne sont pas utiles ou préférables lorsque ...
    - Des algorithmes conventionnels efficaces sont connus
    - Le principal défi est le calcul, pas la connaissance
    - Les connaissances ne peuvent pas être capturées efficacement ou utilisées efficacement

Shells

Rappel des composants du système expert



Shell de système expert

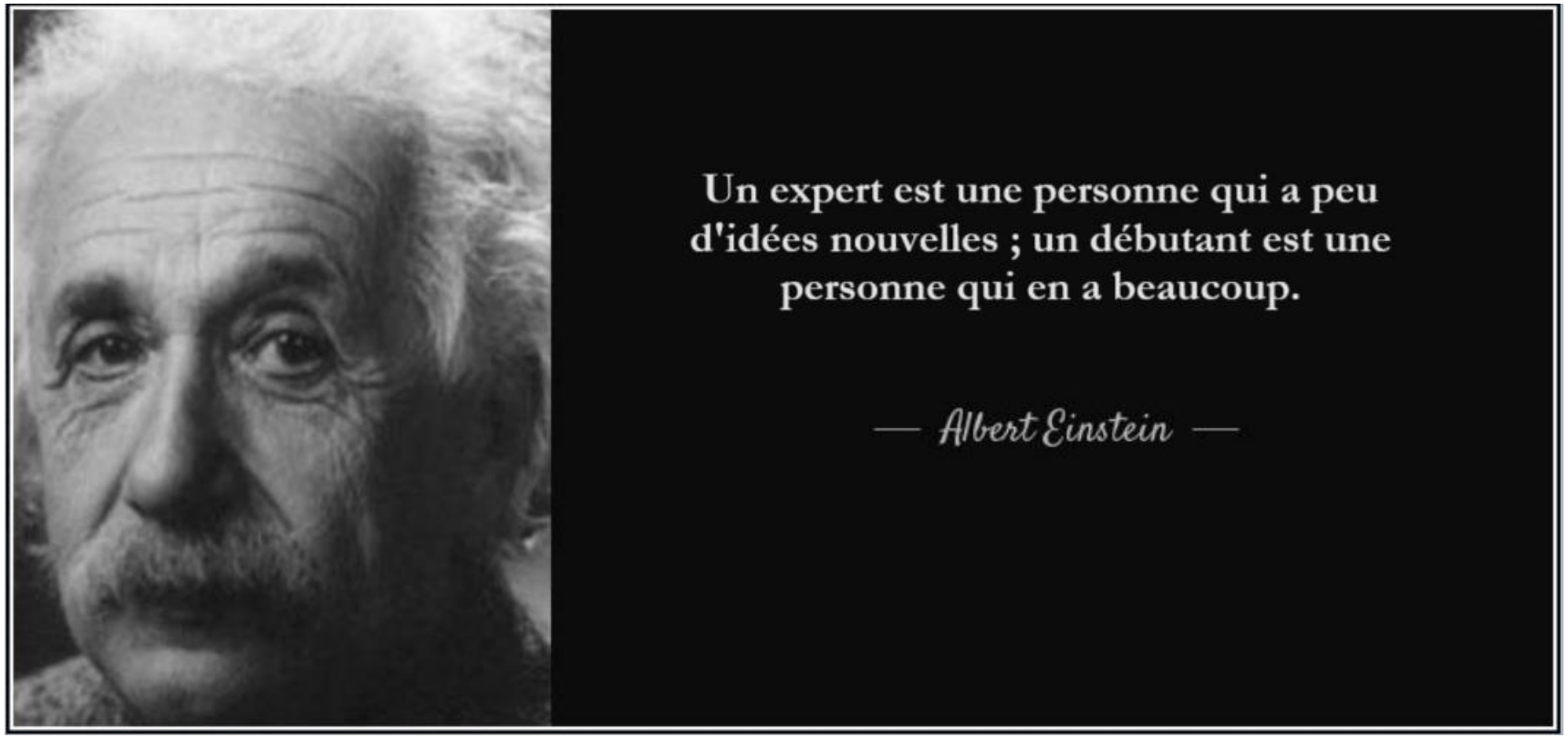
- Shell est la partie "réutilisable" et "recyclable" d'un système expert.
  - Nous pouvons la considérer comme un outil/cadre/environnement de développement d'un système expert.
  - Lorsqu'il dispose d'un shell, le développeur n'a plus qu'à générer la base de connaissances nécessaire au développement d'un système expert fonctionnel.
    - C'est quelque chose que nous allons pratiquer dans ce cours.
- Le shell simplifie le processus de création d'un système expert de manière plus rapide, moins coûteuse et plus efficace.
- La principale mise en garde est que vous devez utiliser une représentation avec laquelle le moteur d'inférence du shell peut raisonner ;
  - dans la plupart des cas, il s'agit de règles.
- Notamment:
  - différents shells peuvent inclure différents composants,
  - peuvent utiliser différents langages de codage des connaissances pour les programmer,
  - ainsi que différentes syntaxes pour la mise en œuvre des règles, des faits, etc.

Synthèse sur le shell

- Les bons shells offrent également des possibilités de communication avec des sources externes y compris
  - les systèmes de gestion de base de données
  - les feuilles de calcul
  - les packages graphiques.
- Fonctionnalités non essentielles mais extrêmement utiles
  - Éditeur de base de connaissances
  - Interface utilisateur
  - Système d'explication
- Exemples des shells:
  - Experta, CLIPS, PyCLIPS, JESS, EMYCIN, Babylon, Exsys, Vidwan, Knowledge Pro, K-Vision, Age, KAS, Leonardo, Xi Plus, Savoir & XpertRule
- Dans ce cours, nous apprendrons à utiliser Experta
  - Il est entièrement en Python 3x
  - Utilise la programmation logique
  - N'a pas :
    - Editeur de base de connaissances
    - Interface utilisateur
    - Système d'explication

Sommaire de la leçon

- Base de connaissances
  - Règles, faits, questions, faits spécifiques à un cas
- Moteur d'inférence
  - Cycle Correspondance-Résolution-Action
  - Chaînage avant
  - Chaînage arrière
- Autres composants
  - Interface utilisateur
  - Éditeur de la base de connaissances
  - Système d'explication
- Autres types de systèmes experts
  - Flou, basé sur des cadres, neuronal, neuro-flou
- Avantages et inconvénients
  - Nombreux
- Shell
  - Mise en œuvre plus facile et plus rapide des systèmes experts


Un expert est une personne qui a peu d'idées nouvelles ; un débutant est une personne qui en a beaucoup.
— Albert Einstein —

Travail pratique 3: Rappel

- **Objectifs :**
  - Représentation des connaissances, systèmes experts et Experta
  - Pratiquer la représentation des cadres
  - Apprendre à installer et à utiliser Experta
  - Pratiquer la représentation des règles
  - Construire un système expert très simple pour le diagnostic
  - Apprendre à programmer des questions
  - Comprendre les chaînages avant et arrière dans les systèmes experts basés sur des règles

- Les résumés des travaux de mi-parcours sont à rendre