GIND5439
Systèmes Intelligents

Chapitre 2: Systèmes à base de règles

Contenu du chapitre

- Définition de connaissances
- Représentation des connaissances
- Les règles
- Chaînage avant et arrière
- Résolution de conflits
- Avantages et désavantages

Connaissances

- Les connaissances sont une compréhension théorique ou pratique d'un domaine particulier.
- Les connaissances représentent la somme de ce qui est connu.
- Ceux qui possèdent des connaissances sont considérés comme des experts.

Connaissances

- On a déjà vu comment classifier les experts et leur expertise.
- Cependant, on doit être capable de représenter cette expertise sous forme textuelle ou graphique.
- Une méthode bien connue est l'utilisation de règles.

Représentation des connaissances

- Un langage de programmation est une façon de représenter des connaissances.
- Connaissances procédurales:
  - « comment »
  - Connaissances sur comment accomplir une tâche
- Connaissances déclaratives:
  - « qu'est-ce que c'est »
  - La capacité de formuler ou décrire quelque chose

Représentation des connaissances

- On utilise des règles:
  - Une structure IF – THEN qui relie de l'information ou des faits de la partie IF avec l'action dans la partie THEN.
    - Ex: IF « lumière est verte » THEN « action est go »
    - Ex: IF « lumière est rouge » THEN « action est stop »
- IF « condition est vraie »
  THEN « faire quelque action ».

Représentation des connaissances

- Une règle est constituée de deux parties:
  - 1. Antécédent: la partie IF
  - 2. Conséquence: la partie THEN
- IF « antécédent » THEN « conséquence »
- Une règle est **déclenchée** lorsque l'antécédent est vrai et que la partie conséquence est exécutée.

Représentation des connaissances

- Une règle peut avoir plusieurs antécédents:
  - Conjonction AND
  - Disjonction OR
  - Ou une combinaison des deux
- Une conséquence peut avoir plusieurs parties:
  - IF « antécédent » THEN « conséquence 1 »
« conséquence 2 »
« conséquence 3 » …
- Il est préférable de ne pas mêler des conjonctions et disjonctions dans la même règle.

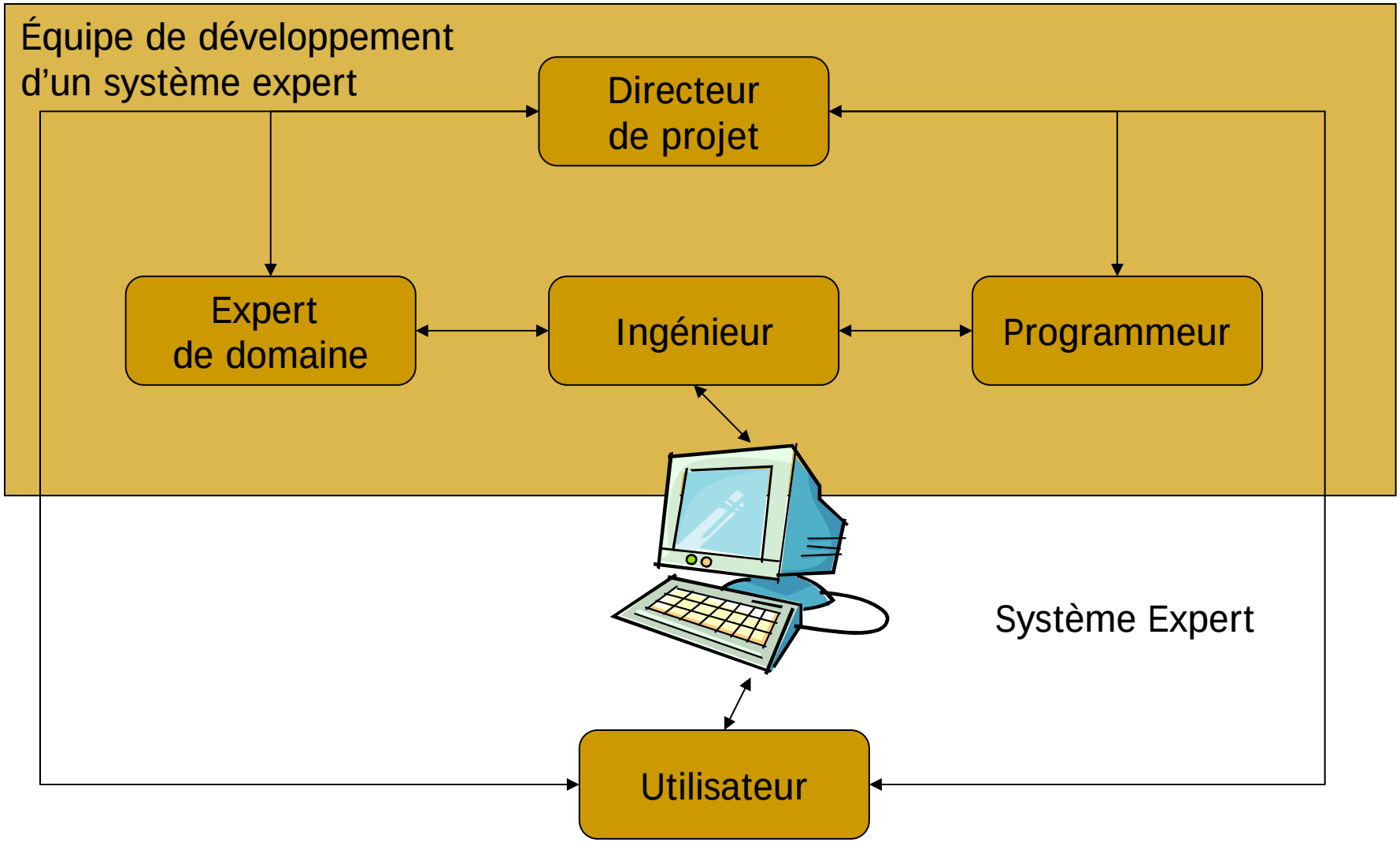
Représentation des connaissances

- Un antécédent est constitué de:
  - Un objet (objet linguistique)
  - Une valeur
  - Ex: l'objet linguistique « lumière » peut prendre la valeur « rouge » ou « verte »
- L'objet et sa valeur sont liés par un opérateur
  - Identifie l'objet et assigne la valeur
  - Ex: is, are, is not, are not (est, n'est pas, etc…)
  - Ex: opérateurs mathématiques: < , >, =, …
- Les conséquences ont la même structure que les antécédents.

Représentation des connaissances

- Les règles peuvent représenter:
  - Des relations
  - Des recommandations
  - Des directives
  - Des stratégies
  - Des heuristiques
    - Heuristique: Méthode de recherche empirique ayant recours aux essais et erreurs pour la résolution de problèmes. [Office de la langue française, 2000]

Équipe de développement d'un SE



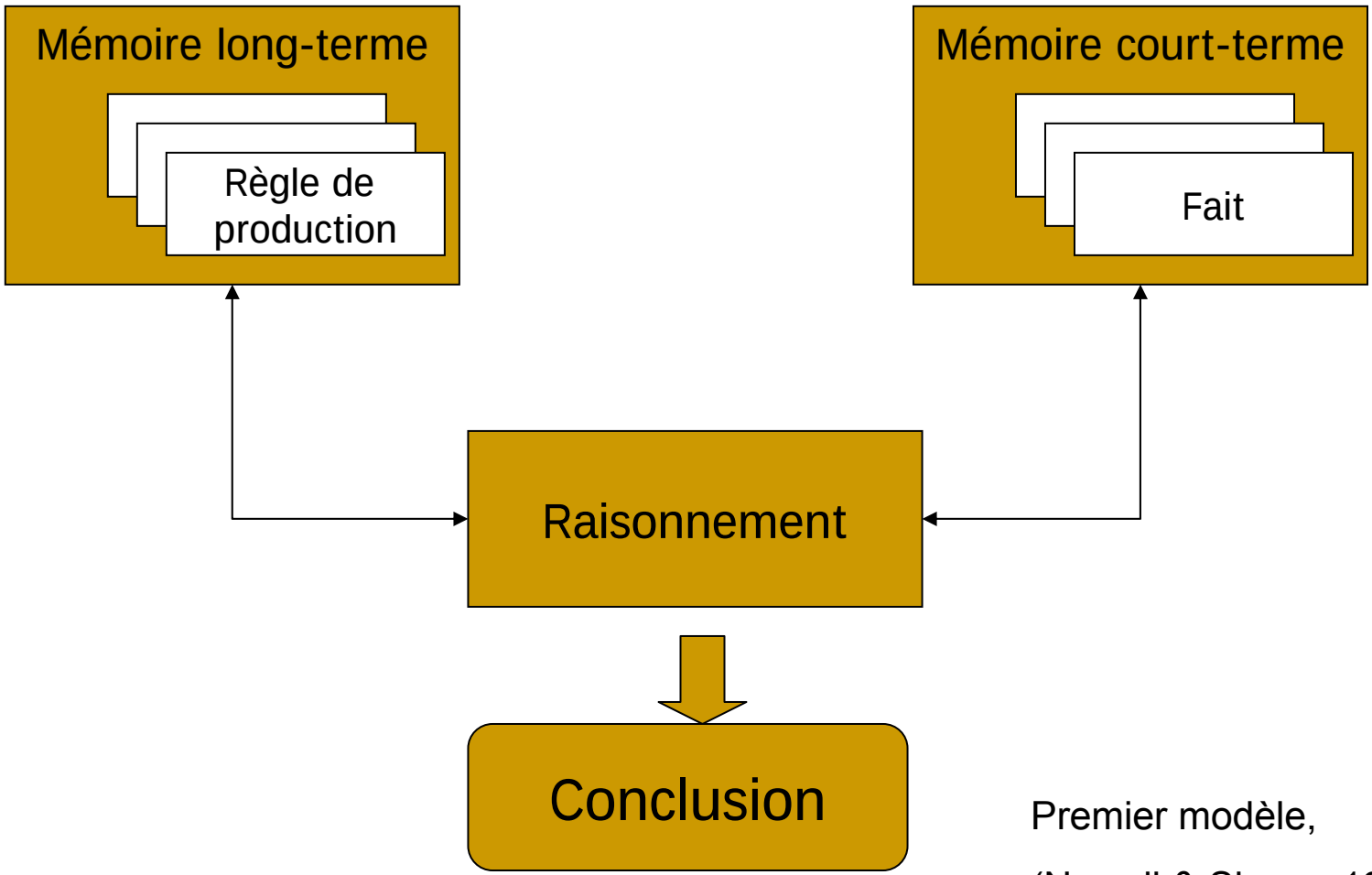
Équipe de développement d'un SE

- Expert de domaine
  - Personne ayant des connaissances et compétences capable de résoudre des problèmes dans un domaine spécifique.
  - L'expert doit être capable de communiquer ses connaissances, doit vouloir participer au développement du système expert, et doit avoir le temps pour le faire.
- Ingénieur de connaissances
  - Capable de concevoir, construire et tester un système expert
  - L'ingénieur choisit les tâches appropriés du système expert.
  - Il (ou elle) questionne l'expert pour trouver comment solutionner le problème.
  - L'ingénieur est responsable de choisir la méthode de solution (logique floue, règles, algorithme génétique, etc…).

Équipe de développement d'un SE

- Programmeur
  - Développe des structures de représentation des connaissances et données.
  - Doit avoir des connaissances dans des langages de programmation symboliques, et dans des langages conventionnels.
- Directeur de projet
  - La personne en charge du projet.
- Utilisateur (« end-user »)
  - Le système expert doit satisfaire aux exigences de tous les utilisateurs.

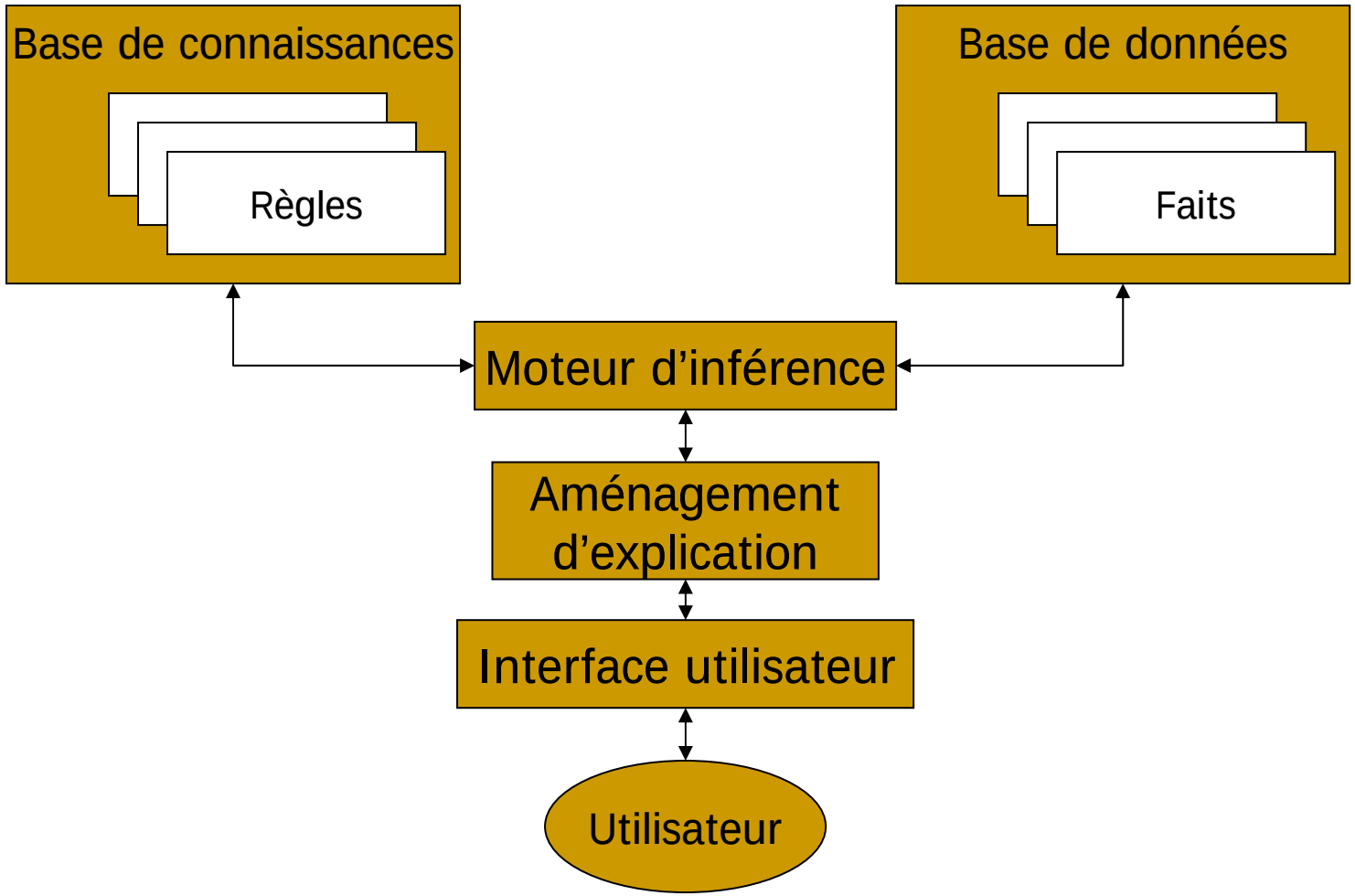
Structure d'un système expert



Premier modèle,

(Newell & Simon, 1972)

Composantes d'un système expert à base de règles



Composantes d'un système expert à base de règles

- Base de connaissances
  - Contient les connaissances utiles au problème
  - On appelle aussi la « base des règles »
- Base de données (*database*)
  - Contient les faits ou données à vérifier contre les conditions
  - Représente l'état actuel du monde
- Moteur d'inférence
  - Interprète les règles
  - Fait le raisonnement pour atteindre une solution
  - Lien entre les connaissances et les données

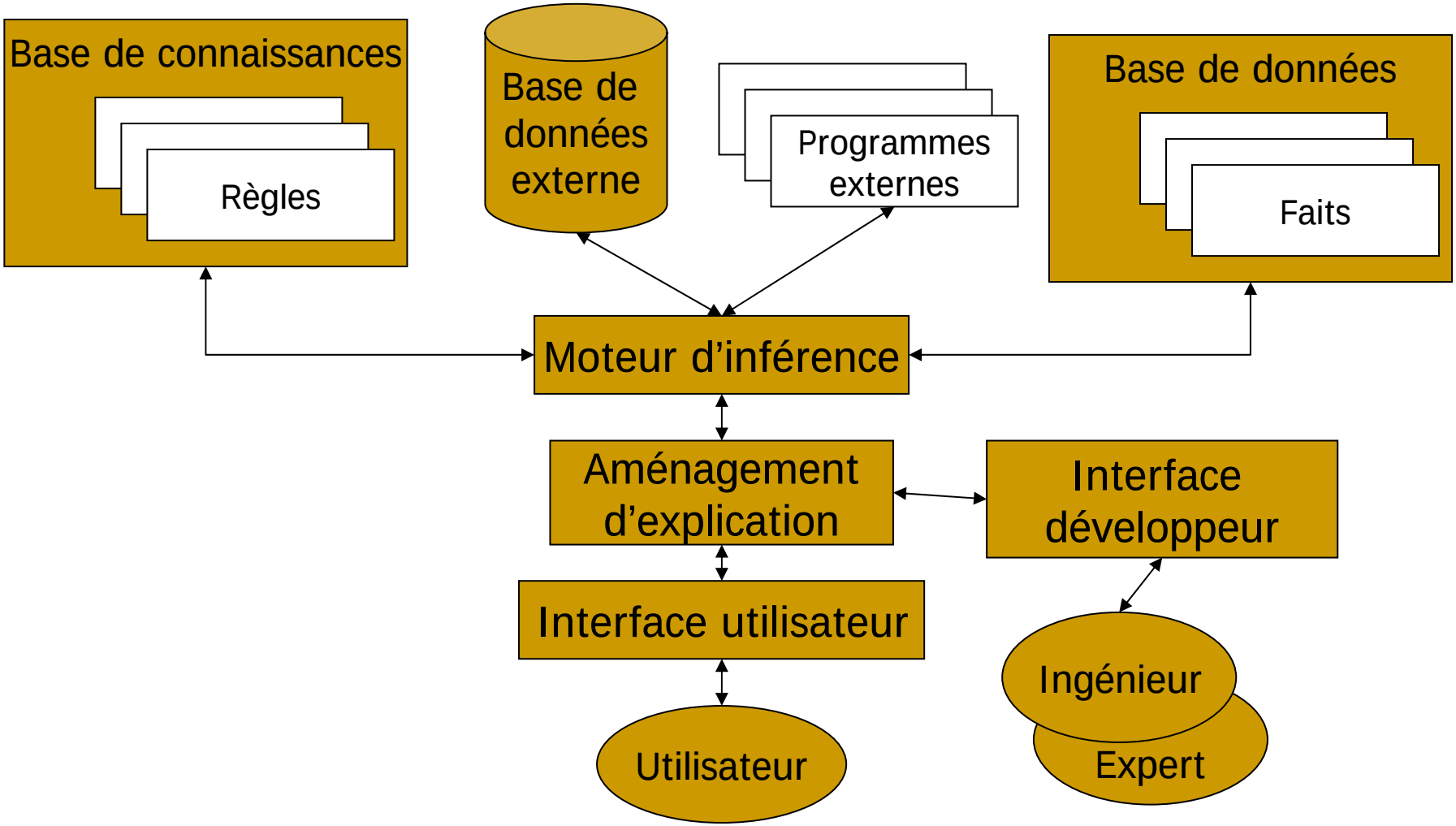
Composantes d'un système expert à base de règles

- Aménagement d'explication
  - Explique le raisonnement et justifie la solution proposée
- Interface utilisateur

Composantes additionnelles

- Interface Externe
  - Données externes, fichiers, programmes dans des langages conventionnels
- Interface du développeur
  - Éditeur de la base de connaissances, outils de débogage, interface entrée/sortie.

Système expert



Caractéristiques d'un système expert

- Construit pour performer à un niveau d'expert humain dans un domaine *précis, spécialisé*.
  - Performance de haute qualité
  - Solutions rapides
  - Utilisation de heuristiques pour guider le raisonnement
- Capable d'explications
  - Le système est capable de réviser son raisonnement et expliquer des décisions
- Utilise un raisonnement symbolique
- Peut fonctionner avec des données incomplètes
- Peut faire des erreurs!
- Connaissances séparées du processus

Comparaisons

Expert Humain	Système Expert	Programme conventionnel
Utilise des connaissances sous forme de heuristique pour résoudre des problèmes dans un domaine précis	Traite des connaissances exprimées comme règles, et raisonnement symbolique pour résoudre des problèmes dans un domaine précis	Traite des données et utilise des algorithmes pour résoudre des problème numériques généraux
Connaissances compilées dans le cerveau	Connaissances et traitement complètement séparés	Aucune séparation des connaissances et structures de contrôle
Capable d'expliquer un raisonnement et fournir des détails	Peut suivre les règles déclenchées et expliquer les conclusions	Aucune explication
Utilise un raisonnement inexacte, peut fonctionner avec des données incomplètes	Permet un raisonnement inexacte, peut fonctionner avec des données incomplètes	Fonctionne seulement avec des données complètes
Peut faire des erreurs quand l'information est incomplète	Peut faire des erreurs quand l'information est incomplète	Aucune solution lorsque les données sont incomplètes
Qualité augmente avec la pratique. Lent, inefficace et dispendieux	Qualité augmente en ajoutant des règles ou en ajustant. Changement sont faciles.	Qualité augmente en modifiant le code. Changements difficiles.

Terminologie

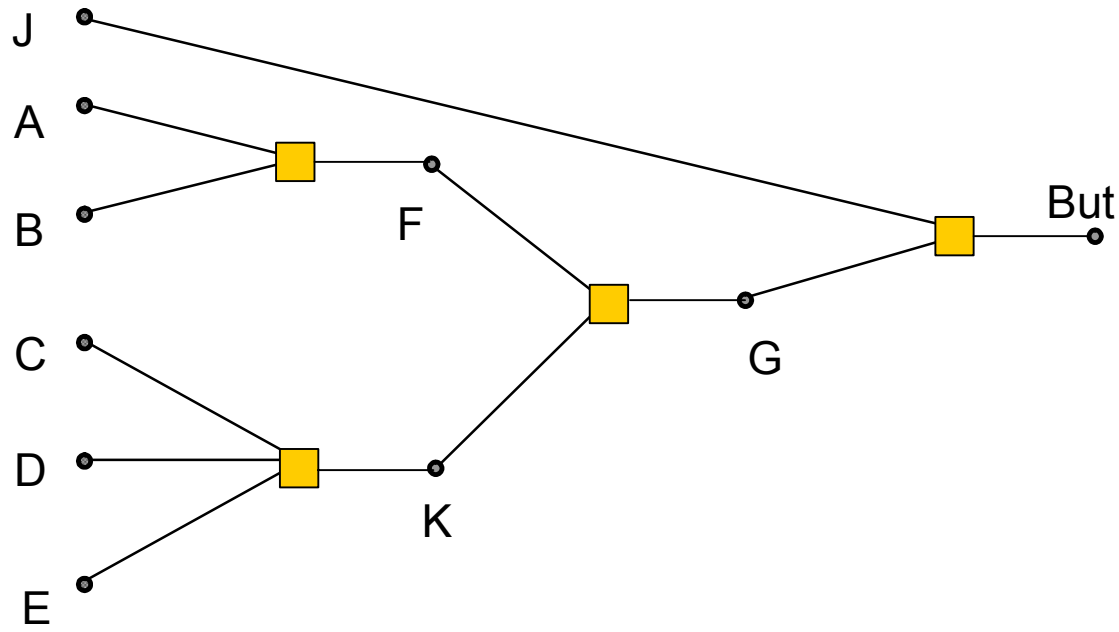
- Une règle est **activée** lorsque l'antécédent est VRAI.
- Une règle est **déclenchée** lorsque la conséquence se produit.
- Si une règle n'est pas déclenchée:
  - L'antécédent est faux
  - La règle n'a pas été choisie

Techniques d'inférence

- Moteur d'inférence
  - Compare les règles dans la base de connaissances avec les faits dans la base de données.
  - Lorsqu'une condition égale un fait, la règle est déclenchée et l'action est exécutée.
  - Une action peut modifier la base de données en ajoutant de nouveaux faits.
- Chaîne d'inférence
  - Indique comment le système expert applique des règles pour atteindre une conclusion.

Chaînage de règles

IF A and B THEN F
IF C and D
and E THEN K
IF F and K THEN G
IF J and G THEN But



On peut faire un chaînage avant vers le but ou commencer au but et essayer de le prouver : chaînage arrière.

Chaînage

- La différence entre le chaînage avant et le chaînage arrière:
  - Supposons qu'on veut prendre l'avion de Moncton à Athènes. Il n'y a pas de vol direct.
  - On peut soit commencer en cherchant tous les vols qui partent de Moncton et noter leur destination. Continuer ce processus jusqu'à ce qu'on trouve un vol qui atterrisse à Athènes. C'est le chaînage avant.
  - On peut commencer d'Athènes et chercher tous les vols qui y arrivent. On trouve la ville de départ de tous ces vols. On continue le processus jusqu'à ce qu'on trouve une ville dont le point de départ est Moncton. C'est le chaînage arrière (on commence du but).

Chaînage avant

- Raisonnement dirigé par les données
  - On commence par des données connues et on procède de l'avant avec ces données
- Seul la règle la plus élevée est déclenchée
- Les règles déclenchées ajoutent des nouvelles données dans la base de données
- Une règle peut seulement être activée une fois
- Le cycle s'arrête lorsqu'il n'y a plus de règles à déclencher.

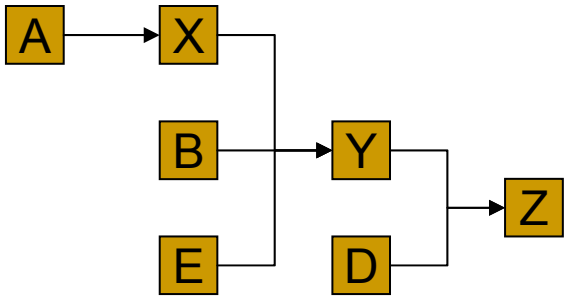
Chaînage avant

- Comment fonctionne-t-il?
  - En cycles
  - Les faits dans la mémoire sont mises à jour avec de l'information du cycle précédent.
  - Les règles sont examinées et toutes les règles dont l'antécédent est vrai sont déclenchées
  - Une collection de règles déclenchées est appelé *l'ensemble de conflits*.
  - Le conflit doit être résolu puisque seulement une règle peut être déclenchée.

Chaînage avant: exemple

- Règle 1: IF Y AND D THEN Z
- Règle 2: IF X AND B AND E THEN Y
- Règle 3: IF A THEN X
- Règle 4: IF C THEN L
- Règle 5: IF L AND M THEN N
- La base de données comprend initialement les faits A, B, C, D et E

Chaînage avant: exemple



R1: Y & D → Z
R2: X & B & E → Y
R3: A → X
R4: C → L
R5: L & M → N

Données: A, B, C, D, E

Cycle 1: on passe les règles jusqu'à ce qu'on en trouve une dont l'antécédent est vrai. R3 et R4 sont donc activées. X et L sont maintenant dans la base de données.

Cycle 2: On repasse les règles qui n'ont pas été déclenchées au cycle précédent. R2 est donc déclenchée, puisque X est dans la base de données. Y est maintenant dans la base de données.

Cycle 3: On repasse les règles qui n'ont pas été activées. R1 est déclenchée, et alors le but, Z, est activé.

Chaînage avant

- On récolte de l'information, puis inférence
- Plusieurs règles peuvent être exécutées qui n'ont rien à faire avec le but.
- N'est pas nécessairement efficace
- L'utilisateur n'est jamais demandé pour de l'information additionnelle.

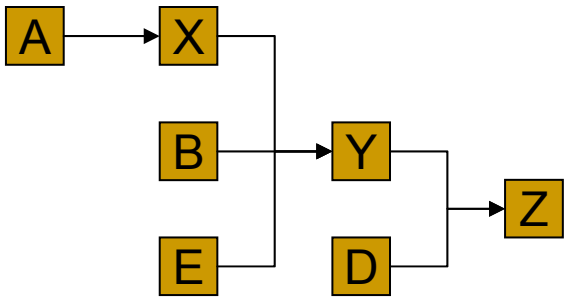
Chaînage arrière

- Raisonnement dirigé par le but
- Le système a un but, et le moteur d'inférence essaie de trouver des preuves pour le prouver.
- On cherche dans la base de connaissances pour trouver les règles qui mènent au but.
  - Des règles qui ont le but dans leur partie « action »
- Si la condition d'une telle règle est dans la base de données, la règle est déclenchée et le but est prouvé.

Chaînage arrière

- Comment fonctionne-t-il?
  - En cycles
  - Avec une pile (« *stack* »)
  - On cherche des sous-buts pour prouver des conditions
  - On continue la pile jusqu'à ce qu'on trouve aucune règle qui peut prouver un sous-but.
  - Plus efficace lorsqu'on veut inférer un certain fait
  - L'utilisateur peut être demandé de donner des informations additionnelles

Chaînage arrière: exemple



R1: Y & D → Z
R2: X & B & E → Y
R3: A → X
R4: C → L
R5: L & M → N

Données: A, B, C, D, E

Cycle 1: on cherche une règle où Z est dans l'action. On le trouve (R1); le fait D est dans la base de données, mais il faut trouver Y.

Cycle 2: on cherche une règle où Y est dans l'action. On le trouve (R2); les faits E et B sont dans la base de données, mais pas X.

Cycle 3: on cherche une règle où X est dans l'action. On le trouve (R3); le fait A est dans la base de données.

Cycle 4: On cherche A dans la base de données, et donc R3 est activée: X est dans la base.

Cycle 5: On cherche à déclencher R2; X, B et E sont dans la base de données, alors Y est activé.

Cycle 6: On cherche à déclencher R1; Y et D sont dans la base; alors Z est prouvé.

Chaînage arrière

- Remarquez que 3 règles ont été déclenchée dans le chaînage arrière, versus 4 règles dans le chaînage avant pour le même exemple.
- Des systèmes experts réels ont des centaines de règles; il faut éviter le plus possible de déclencher des règles de surplus afin d'accélérer le système.

Chaînage arrière vs chaînage avant

- Un raisonnement basé sur les données est approprié lorsqu'il existe plusieurs buts acceptables, peu de faits et un seul état initial.
  - Les faits requis sont disponibles
  - Il est difficile de formuler un but à vérifier
- L'inférence à base de but est approprié lorsque:
  - Les données sont acquises pendant le processus
  - Il y a plusieurs règles à appliquer.

Règles de résolution de conflits

- Utilise la première règle dont la condition est satisfaite
  - L'ordre des règles est importante.
- On peut assigner des priorités aux règles et utiliser celle avec la plus haute priorité.
  - Mais il faut décider de la priorité.
- Utiliser les règles les plus spécifiques
  - Celle avec le plus de détails ou contraintes
- Utiliser la règle qui utilise la donnée la plus récente
- Choisir une règle aléatoirement
- Construire plusieurs base de données et déclencher les règles en parallèle
- Chercher pour la règle la plus appropriée.

Métaconnaissance

- Des connaissances à propos des connaissances
- Connaissances à propos de l'utilisation et contrôle du domaine des connaissances
- Représenté par des méta-règles
- Une méta-règle détermine une stratégie pour l'utilisation de règles spécifiques à une tâche.
- L'ingénieur donne ces règles:
  - Ex: règles fournies par un expert ont plus de priorité que celle fournies par des non-experts.
  - Ex: règles qui indiquent la méningite ont plus de priorité que celle qui indiquent un rhume.

Avantages des systèmes à base de règles

- Représentation naturelle des connaissances
- Structure uniforme
- Séparation des connaissances du traitement
- Peut opérer avec des connaissances incomplètes ou incertaines
  - Facteur de certitude
  - Représente l'incertitude par des chiffres (cf 0.1, ou cf 0.9)
  - Établissement d'un niveau de confiance.

Problèmes

- Relations parfois opaques entre les règles, surtout quand il y en a beaucoup.
  - Difficile de comprendre l'interaction entre les différentes règles.
- Stratégie de recherche inefficace.
  - Le système passe au travers de toutes les règles; pour des systèmes complexes, ceci peut être très lent.
- Incapacité d'apprendre.
  - Un expert humain sait quand « briser les règles »; un système expert ne peut pas le faire. L'ingénieur doit entrer des nouvelles règles ou en modifier; le système ne peut pas le faire de lui-même.

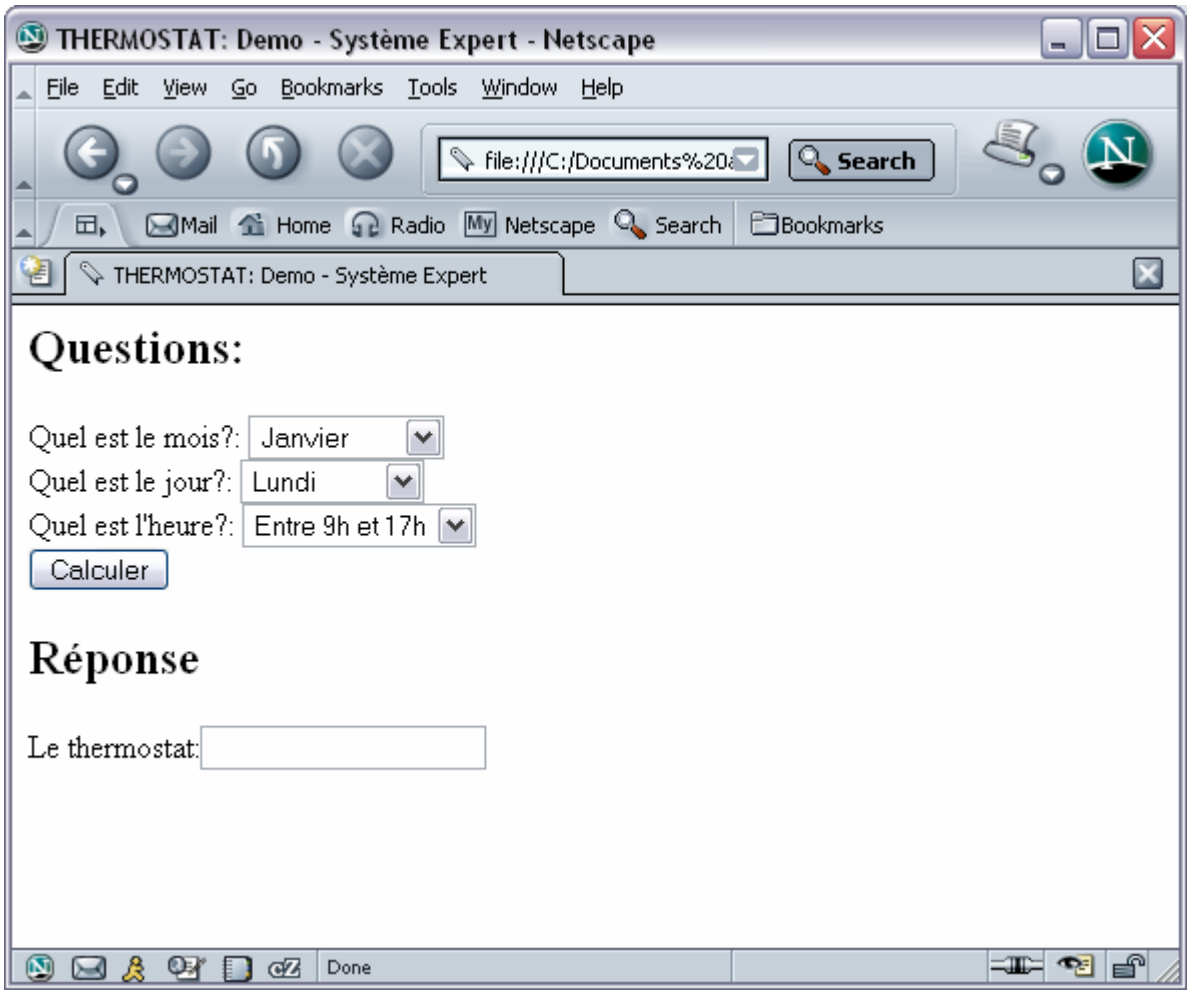
Exemple: THERMOSTAT

- On considère ici un système pour régler automatiquement la température dans un édifice de bureaux.
- Le système a 3 entrées {mois, jour, heure} et 4 variables {saison, aujourd'hui, opération, thermostat} dont une est la sortie {thermostat}.
- Les entrées peuvent prendre les valeurs suivantes:
  - Mois: {janvier, février, mars, avril, mai, juin, juillet, août, septembre, octobre, novembre, décembre}
  - Jour: {lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche}
  - Heure: {entre 9h et 17h, avant 9h, après 17h}

Exemple: THERMOSTAT

- Le but est de déterminer la saison, le jour et l'heure afin de régler la température.
- Les variables prennent les valeurs suivantes:
  - Saison: {été, automne, hiver, printemps}
  - Aujourd'hui: {jour de travail, weekend}
  - Opération:{pendant les heures de travail, pas pendant les heures de travail}
- Il faut 17 règles pour déterminer le réglage du thermostat. (p.41 – 43 du manuel)

Exemple: THERMOSTAT



Implantation HTML du système expert THERMOSTAT

Exemple: THERMOSTAT

- Évidemment, l'implantation HTML du système expert THERMOSTAT est assez simple. Pour un système réel, l'interface utilisateur serait bien plus élégant.
- L'implantation ici ne nécessite qu'une connaissance de base du HTML et du Javascript.

Exemple: THERMOSTAT

```
<body>
<h2>Questions:</h2>

<form name="frmES" action="">
Quel est le mois?: <select
name="selMois">
<option value="Janvier">Janvier
<option value="Février">Février
<option value="Mars">Mars
<option value="Avril">Avril
<option value="Mai">Mai
<option value="Juin">Juin
<option value="Juillet">Juillet
<option value="Août">Août
<option value="Septembre">Septembre
<option value="Octobre">Octobre
<option value="Novembre">Novembre
<option value="Décembre">Décembre
</select>
<br>
Quel est le jour?: <select name="selJour">
<option value="Lundi">Lundi
<option value="Mardi">Mardi
<option value="Mercredi">Mercredi
<option value="Jeudi">Jeudi
<option value="Vendredi">Vendredi
<option value="Samedi">Samedi
<option value="Dimanche">Dimanche
</select>
<br>
Quel est l'heure?: <select name="selHeure">
<option value="entre9et5">Entre 9h et 17h
<option value="avant9">Avant 9h
<option value="après5">Après 5h
</select>
<br>
<input type="button" value="Calculer" onClick="SE();">
</form>
<h2>Réponse</h2>
<form name="frmReponse">
Le thermostat:<input type="text" name="edtReponse">
</form>
</body>
```

Code HTML pour l'interface

Exemple: THERMOSTAT

Code javascript du système expert

```
<script language="JavaScript" type="text/javascript">

function SE(){

 var aujourdhui, operation, saison, thermostat;
 Form = document.frmES;
 mois = Form.selMois.options[Form.selMois.selectedIndex].value;
 jour = Form.selJour.options[Form.selJour.selectedIndex].value;
 heure = Form.selHeure.options[Form.selHeure.selectedIndex].value;

 //***** Règles ********

 if (jour == "Lundi" | jour == "Mardi" | jour == "Mercredi" | jour == "Jeudi" | jour == "Vendredi") { aujourdhui = "jour_de_travail"}
 if (jour == "Samedi" | jour == "Dimanche") { aujourdhui = "weekend" }
 if (aujourdhui == "jour_de_travail" & heure == "entre9et5") {operation = "heure_de_travail"}
 if (aujourdhui == "jour_de_travail" & heure == "avant9") {operation = "pas_heure_de_travail"}
 if (aujourdhui == "jour_de_travail" & heure == "après5") {operation = "pas_heure_de_travail"}
 if (aujourdhui == "weekend") {operation = "pas_heure_de_travail"}
 if (mois == "Janvier" | mois == "Février" | mois == "Mars") {saison = "hiver"}
 if (mois == "Avril" | mois == "Mai" | mois == "Juin") {saison = "printemps"}
 if (mois == "Juillet" | mois == "Août" | mois == "Septembre") {saison = "ete"}
 if (mois == "Octobre" | mois == "Novembre" | mois == "Décembre") {saison = "automne"}
 if (saison == "printemps" & operation == "heure_de_travail") {thermostat = "20 degrés"}
 if (saison == "printemps" & operation == "pas_heure_de_travail") {thermostat = "15 degrés"}
 if (saison == "ete" & operation == "heure_de_travail") {thermostat = "24 degrés"}
 if (saison == "ete" & operation == "pas_heure_de_travail") {thermostat = "27 degrés"}
 if (saison == "automne" & operation == "heure_de_travail") {thermostat = "20 degrés"}
 if (saison == "automne" & operation == "pas_heure_de_travail") {thermostat = "16 degrés"}
 if (saison == "hiver" & operation == "heure_de_travail") {thermostat = "18 degrés"}
 if (saison == "hiver" & operation == "pas_heure_de_travail") {thermostat = "14 degrés"}

 document.frmReponse.edtReponse.value = thermostat;
}
</script>
```