Modélisation de la Résolution de Problèmes

Raisonnement - Rappels

Raisonnement : « Suite d'opérations par lesquelles on conclut qu'une proposition implique la vérité d'une autre proposition » (*)

Historiquement : Nécessité de séparer l'évidence intuitive et démonstration rationnelle. Une démonstration est une succession d'étapes dont l'enchaînement ne doit laisser aucune place au doute.

→ Proposition $_k$ → Proposition $_{k+1}$ → ...

Dans la mesure où le processus repose sur des postulats (axiomes, propositions non démontrées), la vérité des conclusions repose sur l'acceptation de ces axiomes et des hypothèses initiales et sur la rigueur de l'enchaînement des étapes.

Raisonner n'est donc plus convaincre de la vérité des conclusions, mais garantir, dans une axiomatique donnée, la cohérence entre hypothèses et conclusions.

{ Axiomes } + { Propositions } $_k$ → Proposition $_{k+1}$

(*) *« proposition » est à prendre au sens large d'énoncé exprimé dans un langage (formel ou non)*

Par souci d'économie et pour éviter de repartir des axiomes, on 'démontre' (on produit) puis ré-utilise des théorèmes (propositions génériques non initiales).

Axiomes → ... → Théorème → ... → Théorème
{ Théorèmes, Axiomes } + { Propositions } $_k$ → Proposition $_{k+1}$

Dans tout domaine existent des propositions productives (de type 'loi expérimentale') et/ou définitionnelles. On les regroupe sous le terme de Contrainte.

{ Théorèmes, Axiomes, Contraintes } + { Propositions } $_k$ → Proposition $_{k+1}$

… et dans tout problème, il y a des propositions factuelles et/ou des buts.

Exemples

théorème :

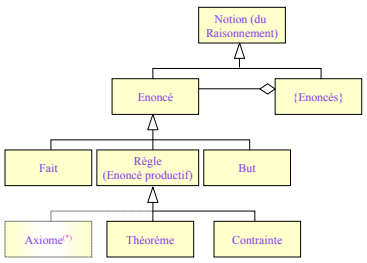
contraintes (lois, définitions) :

faits :

buts :

Un raisonnement est un enchaînement d'étapes de 'production' de nouveaux énoncés (hormis les Axiomes) à partir d'une sélection d'énoncés déjà produits.

{ Enoncés } → Enoncé



Raisonnement Formel

Ramener tout *Raisonnement* à un *Calcul* consiste à assimiler :

- toute donnée à un ensemble d'énoncés exprimés dans un langage formel L , et
- toute étape du raisonnement à la production d'un nouvel énoncé par utilisation d'une règle de ré-écriture.

Un formalisme (un langage) L *d'expression des énoncés*
+
Un formalisme (un langage) L' *d'expression des règles de ré-écriture*

Si L est d'ordre n, L' est d'ordre n+1

Rappel de quelques règles classiques de ré-écriture de formules pour L ~ logique d'ordre 1 :

règles de substitution

$\forall x \in \mathcal{E}\ p(x) + a \in \mathcal{E} \rightarrow p(a)$ $\qquad$ $p(a) + a \in \mathcal{E} \rightarrow \exists x \in \mathcal{E} \mid p(x)$

propagation de faits

$p(a) + \forall x \in \mathcal{E}\ p(x) \Rightarrow q(x) \rightarrow q(a)$ $\qquad$ $\neg q(a) + \forall x \in \mathcal{E}\ p(x) \Rightarrow q(x) \rightarrow \neg p(a)$

et moins classiques :

transfert de buts

$q(a)? + \forall x \in \mathcal{E}\ p(x) \Rightarrow q(x) \rightarrow p(a)?$

génération de but (raisonnement hypothético-déductif) :

$q(a) + \forall x \in \mathcal{E}\ p(x) \Rightarrow q(x) \rightarrow p(a)?$

Raisonnement & Résolution de problèmes

Tout raisonnement s'inscrit dans une dynamique de résolution de problèmes.
Il met en jeu :

des connaissances spécifiques au problème : faits & buts (BdD)
+
des connaissances spécifiques au domaine : contraintes (lois & définitions)
+
des connaissances multi-domaines : axiomes & théorèmes

langage L

+
des règles de production de formules en L indépendantes des pb et domaines
+
un contrôle de la résolution

meta langage L'

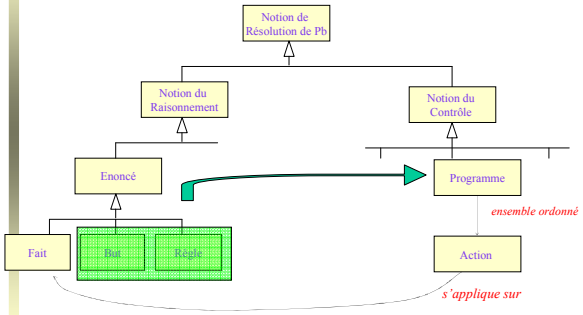
Définition du contrôle dans les systèmes intelligents :

« En résolvant le problème du contrôle, un système décide quel problème il va tenter de résoudre, quelle connaissance il va utiliser et quelle méthode de résolution de problèmes et stratégies il va appliquer. Il décide comment il doit évaluer des solutions concurrentes, comment il doit savoir quand un problème est résolu et dans quelles circonstances il doit distraire son attention d'un sous-problème en cours de résolution »

B. Hayes-Roth "*A Blackboard Architecture for Control*"
Artificial Intelligence, 26, pp 252-321 1985

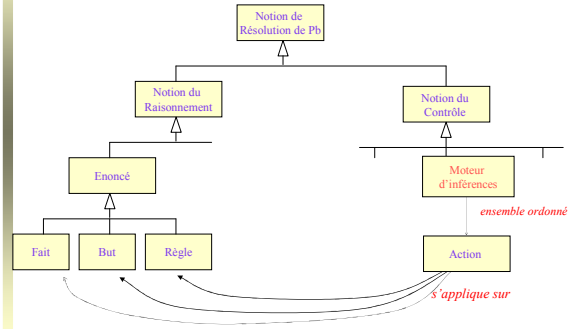
Contrôle - Expression procédurale

Les buts, contraintes, axiomes et théorèmes sont inclus dans la tâche de résolution



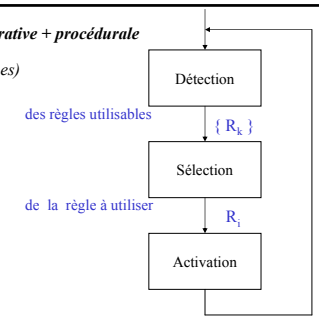
Contrôle - Expression déclarative + procédurale

Les buts, contraintes, axiomes et théorèmes sont extérieurs à la tâche de résolution



Contrôle - Expression déclarative + procédurale

(Programme = Moteur d'Inférences)



Marche avant :
R_k :: Règles utilisant les faits connus
(= règles dont toutes les conditions sont remplies)

Marche arrière :
R_k :: Règles concluant sur un des buts courants

Marche avant :

(rr3) t : règle
{t.condition} ∈ {fait} → t.détectéeMAv
/si les conditions d'une règle sont remplies, elle est détectée en Marche Avant

(rr4) t : règle
t.détectéeMAv + t.utilisée$_k$ → {fait}$_{k+1}$:= {fait}$_k$ ∪ { t.conclusion}
/si une règle est utilisée en MAv, ses conclusions deviennent des faits

---

Marche arrière :

(rr1) t : règle, b : but
b ∈ {but} + b ∈ {t.conclusion} → t.détectéeMAr
/si une règle conclut sur un des buts visés, elle est détectée pour aider à prouver ce but

(rr2) t : règle
t.détectéeMAr + t.sélectionnée$_k$ → {but}$_{k+1}$:= {but}$_k$ ∪ {t.condition}
/si une règle utilisable en Mar est sélectionnée, l'ensemble de ses conditions deviennent de nouveaux buts

(rr5) {but}$_k$ + {fait}$_{k+1}$ → {but}$_{k+1}$:= {but}$_k$ - ({but}$_k$ ∩ {fait}$_{k+1}$)
/les buts déjà prouvés sont éliminés des buts à prouver

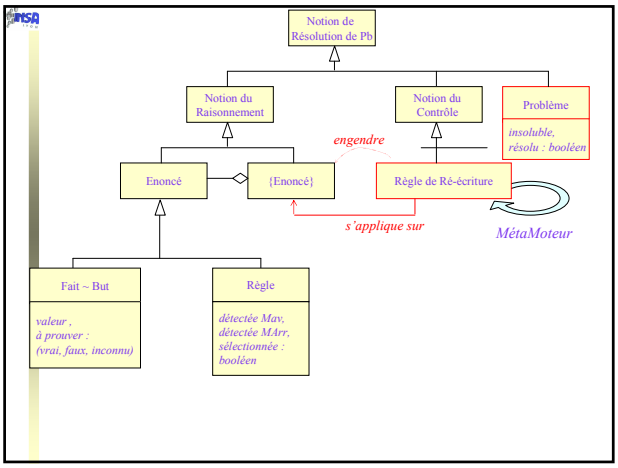
(rr6) b1, b2 : but
b1, b2 ∈ {but} + contradictoires(b1, b2) → problème.insoluble

(rr7) f1, f2 : fait
f1, f2 ∈ {fait} + contradictoires(f1, f2) → problème.insoluble

(rr8) card ({but}) = 0 → problème.résolu

(rr9) t : règle
card ({t.détectéeMAr}) = 0 + card ({but}) > 0 → ¬ problème.résolu

Expression déclarative du moteur d'inférences.
problème et de règle de ré-écriture sont des notions de résolution.

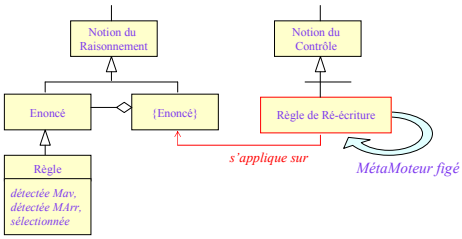


Mais
On ne dit pas ici comment choisir parmi l'ensemble { règle.détectéeMAv } ou parmi l'ensemble { règle. détectéeMAr } i.e. comment passer de détectée à sélectionnée

Contrôle implicite déterministe

exemple :

l'ensemble des buts à prouver est géré en *lifo* (*fifo*)
l'ensemble des règles détectées est géré en *fifo* (*lifo*)
la propagation des nouveaux faits en profondeur d'abord *lifo* (largeur d'abord *fifo*)



Contrôle explicite et déclaratif :
des règles gèrent l'emploi des règles de réécriture

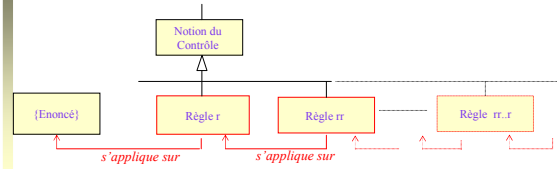
exemple :

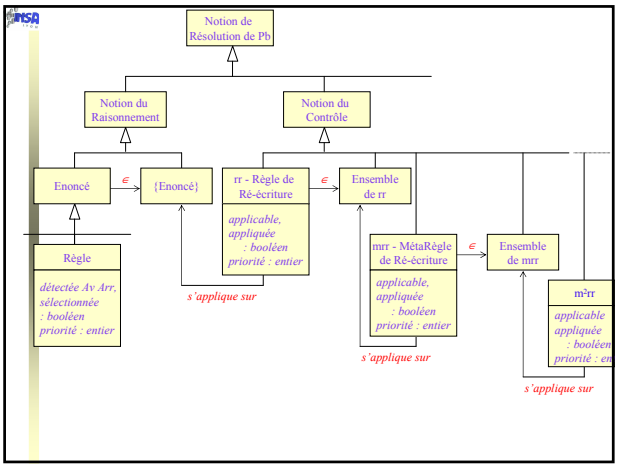
k1 : *« Contre 3SA entamer dans une longue »* k1 → r_1
k2 : *« Contre 3SA entamer dans une majeure »* k2 → r_2

k3 : *« Préférer une entame dans une majeure à une entame dans une longue »*

k3 → rr_1

rr_1 : r_1 .détectée + r_2 .détectée → r_2 .sélectionnée





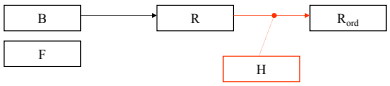
Heuristiques :

Connaissances de résolution empiriques,
décrivant des préférences intuitives,
pas nécessairement optimales

→ ***Méta-heuristiques***

Opérationnalisation
nécessite une adaptation au contexte :
une formalisation, c.a.d. la définition de métriques
et de processus d'évaluation rapide associés

Soit B = {but} R = {règle utilisable} F = {fait}
On a $B \cap F = \emptyset$



En Marche Arrière,
si R = ∅ alors B est non prouvable,
sinon, se pose la question de l'ordonnancement des r dans R

On définit la couverture K du but b par l'ensemble des règles utilisables concluant sur b :
$K(b) = \{r\} \mid b \in \{r.conclusion\}$
la couverture d'une disjonction de buts $B = \{b1 \vee b2 \vee .. \vee bn\}$ par :
$K(B) = \cup K(bi) \mid bi \in B$
et la couverture d'une conjonction de buts $B = \{b1 \wedge b2 \wedge .. \wedge bn\}$ par :

$$K(B) = \begin{cases} \emptyset \text{ si } \exists bi \mid K(bi) = \emptyset \\ \cup K(bi) \mid bi \in B \text{ sinon} \end{cases}$$

Méta-heuristique h10 : *« Lorsque ça doit rater, autant le savoir le plus vite possible » donne « Dans le cas d'une conjonction de buts, commencer par les buts ayant le moins de possibilités de preuves »*

se traduit par (rr10) : Ordonner {b1 ∧ b2 ∧ .. ∧ bn} par ordre croissant de card(K(bi))

Exemple

Avec B = {b1 , b2 , b3}

r1 : … ⇒ b1 ∧ b2 ∧ b3
r2 : … ⇒ b1 ∧ b4
r3 : … ⇒ b1
r4 : … ⇒ b2 ∧ b3
r5 : … ⇒ b1 ∧ b2 ∧ b4
r6 : … ⇒ b4

K(b1) = {r1, r2, r3, r5}
K(b2) = {r1, r4, r5}
K(b3) = {r1, r4, r6}

h10 (rr10) B → B = { }

Méta-heuristique h10' : *« Aller au plus simple » donne « Dans le cas d'une disjonction de buts, commencer par les plus faciles, ceux ayant le plus de possibilités de preuves»*

se traduit par (rr10') : Ordonner {b1∨ b2 ∨ .. ∨ bn} par ordre décroissant de card(K(bi))

Exemple

Avec B = {b1 ∨ b4}

r1 : … ⇒ b1 ∧ b2 ∧ b3
r2 : … ⇒ b1 ∧ b4
r3 : … ⇒ b1
r4 : … ⇒ b2 ∧ b3
r5 : … ⇒ b1 ∧ b2 ∧ b3
r6 : … ⇒ b4

K(b1) = {r1, r2, r3, r5}
K(b4) = {r2, r6}

K(b1 ∨ b4) = {r1, r2, r3, r5, r6}

h10' (rr10') B → B = (d'abord b1 et puis sinon b4)

h10'(rr10') {r1, r2, r3, r5, r6} → { }

On définit l'apport d'une règle r par le nombre de ses conclusions :

$A(r) = card(\{r.conclusion\})$

l'apport relatif de r vis à vis d'une conjonction de buts B par la proportion des buts de B prouvés par r : $A_r(r, B) = card(\{r.conclusion\} \cap B) / card(B)$

la dispersion de r vis à vis d'une conjonction de buts B par le nb de buts prouvés ne faisant par partie de B : $D(r, B) = card(\{r.conclusion\} - B)$

et la dispersion relative d'une règle r vis à vis de l'ensemble de buts B :

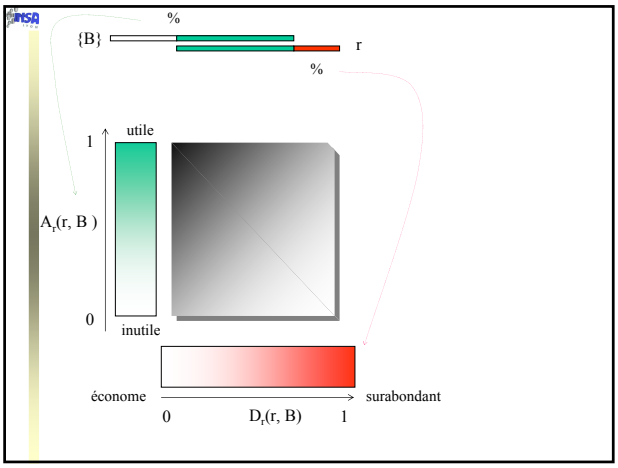
$D_r(r, B) = D(r, B) / A(r)$

Exemple

Avec B = {b1 ∧ b2 ∧ b3} et

r1 : … ⇒ b1 ∧ b2 ∧ b3
r2 : … ⇒ b1 ∧ b4
r3 : … ⇒ b1
r4 : … ⇒ b2 ∧ b3
r5 : … ⇒ b1 ∧ b2 ∧ b3 ∧ b4
r6 : … ⇒ b4

	A(r)	$A_r(r, B)$	D(r, B)	$D_r(r, B)$
r1				
r2				
r3				
r4				
r5				
r6				



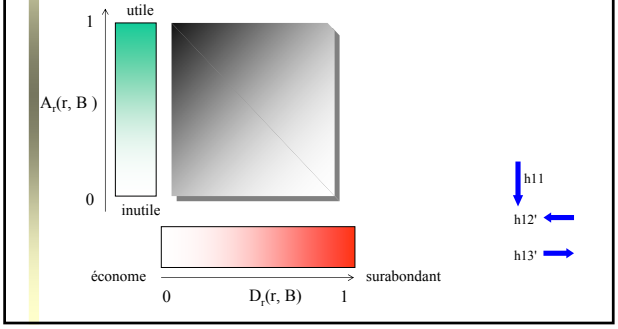
Méta-heuristique économique h11 : *« préférer les règles permettant de remplir le maximum des buts à atteindre»* se traduit par
(rr11) : R = { r } | r.utilisable → R | r par ordre décroissant de A_r (r, B)

Méta-heuristique anticipative h12 = *« tout ce qui est fait n'est plus à faire»* donne :
(rr12) : R = { r } | r.utilisable → R | r par ordre décroissant de D(r, B)
ou encore
(rr12') : R = { r } | r.utilisable → R | r par ordre décroissant de D_r(r, B)

Méta-heuristique du paresseux h13 = *« ne pas en faire plus que nécessaire»* donne :
(rr13) : R = { r } | r.utilisable → R | r par ordre croissant de D(r, B)
ou encore
(rr13') : R = { r } | r.utilisable → R | r par ordre croissant de D_r(r, B)

h11 (rr11) {r1, r2, r3, r4, r5} → { }
h12 (rr12) {r1, r2, r3, r4, r5} → { }
h12' (rr12') {r1, r2, r3, r4, r5} → { }
h13 (rr13) {r1, r2, r3, r4, r5} → { }
h13' (rr13') {r1, r2, r3, r4, r5} → { }

On peut enchaîner l'application de plusieurs méta-heuristiques jusqu'à obtenir un ordre total.
h11(h12') :: (rr12') puis (rr11) → { }
h12'(h11) :: (rr11) puis (rr12') → { }
h11(h13') :: (rr13') puis (rr11) → { }
h13'(h11) :: (rr11) puis (rr13') → { }



Les heuristiques traduites en règles de ré-écriture **(rr..)** sont un moyen d'exprimer **explicitement** des connaissances de contrôle. Cependant ...

Certaines règles de ré-écriture sont intrinsèquement prioritaires ::

(rr6) b1, b2 : but
b1, b2 ∈ {but} + contradictoires(b1, b2) → problème.insoluble

(rr7) f1, f2 : fait
f1, f2 ∈ {fait} + contradictoires(f1, f2) → problème.insoluble

(rr8) card ({but}) = 0 → problème.résolu

Certaines règles de ré-écriture entrent mutuellement en conflit, peuvent se combiner, …

⇨ Il existe des connaissances de contrôle **sur** les connaissances de contrôle. Ce sont des **mrr** qui expriment les conditions de l'application, l'ordre dans lequel les appliquer et les conditions d'arrêt de l'application des règles de ré-écriture

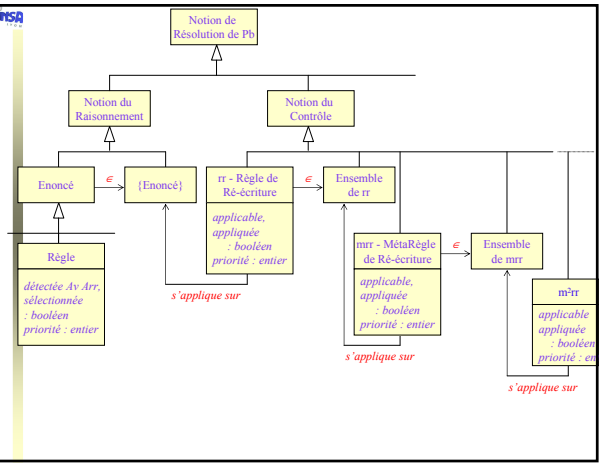
Ordonnancement de rr : →

mrr11) {rr} → {rr} ordonnée selon la relation d'ordre α (par exemple rr12 puis rr11)
mrr12) {rr} → {rr} ordonnée selon la relation d'ordre β (par exemple rr11 puis rr12)
mrr13) ...

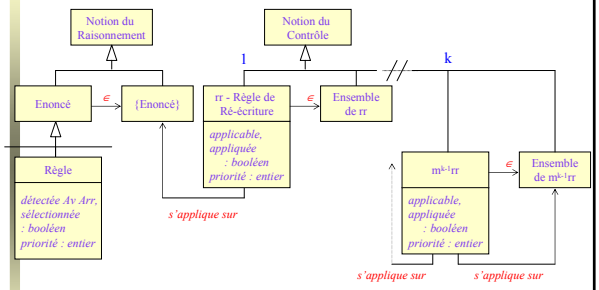
→ *Ordonnancement de* mrr ? →

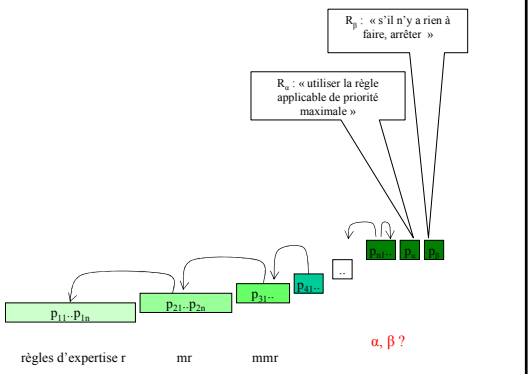
mmrr11) {mrr} → {mrr} ordonnée relation d'ordre λ (par exemple mrr12 puis mrr13, puis...)
mmrr12) {mrr} → {mrr} ordonnée relation d'ordre μ ...

⇨ Les notions de mmrr, mmmrr, …et ensembles de mmrr, mmmrr, … sont, elles aussi, des notions primitives du langage L'.



Question : Pour arrêter la suite, peut-on envisager un niveau k dans lequel seraient exprimées explicitement des connaissances de contrôle de niveau k-1 et k ?





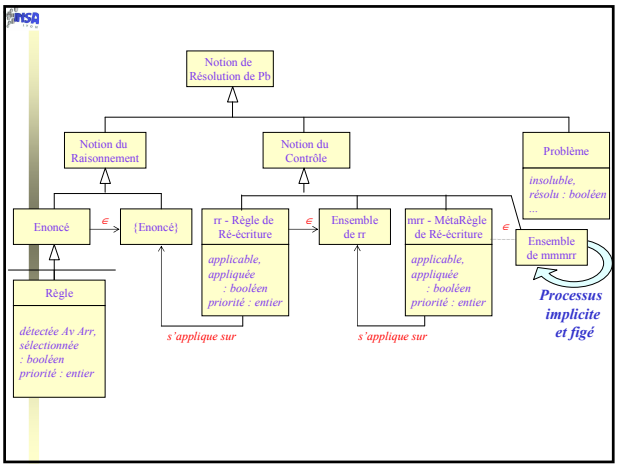
En clair :

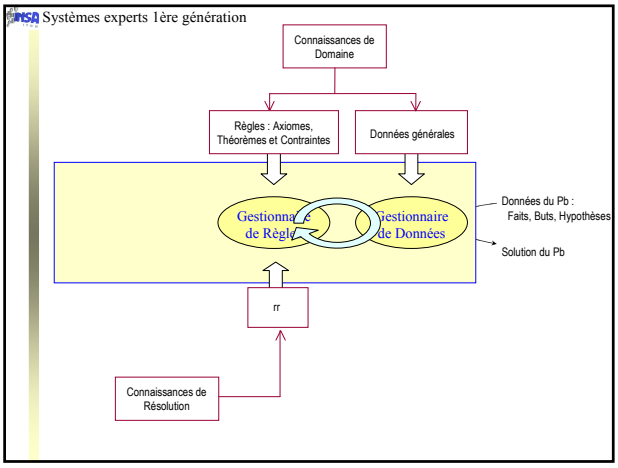
a) Toutes les notions nécessaires à la résolution de problèmes sont des primitives d'un méta-langage L'. L'automatisation de cette résolution nécessite que ce langage L' soit un langage formel exécutable.

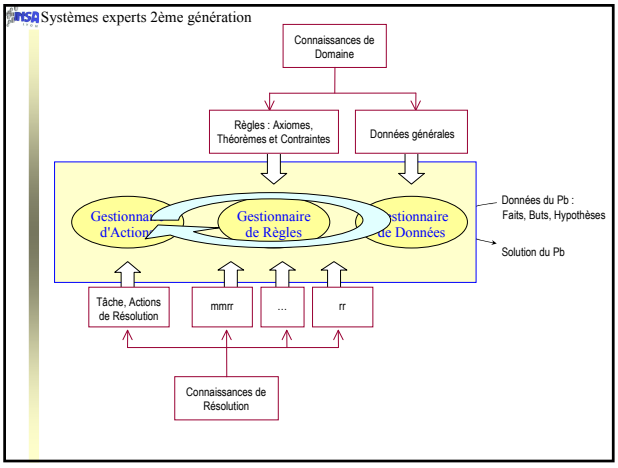
b) rr , mrr, mmrr, mmmrr, ... est une suite *infinie* de notions nécessaires à la modélisation du contrôle de cette résolution et ces notions sont *irréductibles* les unes aux autres.

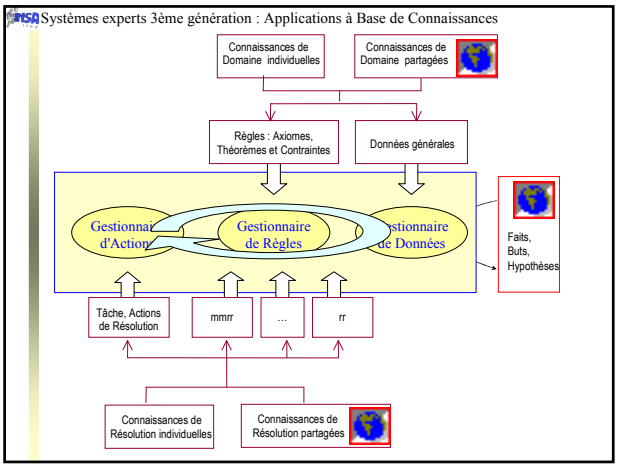
Soit la résolution est modélisée (exprimée) dans un langage réflexif et/ou extensible à l'infini (i.e. en langue naturelle) et elle n'est pas exécutable.

Soit une partie de la résolution n'est pas explicitement représentée dans le modèle. Elle est prise en charge par un processus implicite figé, impératif et invariable.


Notion de Résolution de Pb
Notion du Raisonnement
Notion du Contrôle
Problème
insoluble, résolu : booléen
...
Enoncé
{Enoncé}
rr - Règle de Ré-écriture
applicable, appliquée : booléen
priorité : entier
Ensemble de rr
mrr - MétaRègle de Ré-écriture
applicable, appliquée : booléen
priorité : entier
Ensemble de mmrr
Règle
détectée Av Arr, sélectionnée : booléen
priorité : entier
s'applique sur
s'applique sur
Processus implicite et figé


Systèmes experts 1ère génération
Connaissances de Domaine
Règles : Axiomes, Théorèmes et Contraintes
Données générales
Gestionnaire de Règles
Gestionnaire de Données
Données du Pb : Faits, Buts, Hypothèses
Solution du Pb
rr
Connaissances de Résolution


Systèmes experts 2ème génération
Connaissances de Domaine
Règles : Axiomes, Théorèmes et Contraintes
Données générales
Gestionnaire d'Actions
Gestionnaire de Règles
Gestionnaire de Données
Données du Pb : Faits, Buts, Hypothèses
Solution du Pb
Tâche, Actions de Résolution
mmrr
...
rr
Connaissances de Résolution


Systèmes experts 3ème génération : Applications à Base de Connaissances
Connaissances de Domaine individuelles
Connaissances de Domaine partagées
Règles : Axiomes, Théorèmes et Contraintes
Données générales
Gestionnaire de Règles
Faits, Buts, Hypothèses
Tâche, Actions de Résolution
mmrr
...
rr
Connaissances de Résolution individuelles
Connaissances de Résolution partagées