Systèmes Intelligents : Raisonnement et Reconnaissance

James L. Crowley

Deuxième Année ENSIMAG Deuxième Sémestre 2007/2008

Séance 5 26 mars 2008

Raisonemment avec Rélations et Schémas

La Logique Temporelle d'Allen :.....2
Les Rélations.....3
l'Incertain :.....4
Table de Transitivité.....5
Propagation de Contraintes.....6
Les intervalles de référence :.....8
Exemples.....9

Representation Structurée de la Connaissance..... 10
Définition des Classes en CLIPS.....10
Héritage des définitions des classes.....13
Héritage simple.....13
Héritage multiple.....13

Buts :
Voir une technique de raisonnement pratique dérivée d'une analyse formelle.
Voir une technique de représentation de l'incertain.
Voir une technique de réduction de la complexité par décomposition hiérarchique.

La Logique Temporelle d'Allen :

Les buts du système:

1) Exprimer la connaissance relative et imprécise.
2) Permettre l'incertitude (expression des contraintes).
3) Supporter le raisonnement dans des échelles variables
4) Supporter la persistance (raisonnement par défaut).
   (ex : si j'ai garé ce matin ma voiture dans le parking,
   elle doit toujours être là, même si on ne peut le prouver).

Dans le système d'Allen le temps est représenté par des intervalles :

Intervalle : Un ensemble T = {t} ordonné de points tel que
(t^-) (t T) ($t^- < t$) et
(t^+) (t T) ($t^+ > t$)

la paire t^-, t^+ sont des points limites ("end-points").

Il y a 13 relations qui peuvent être définies entre deux intervalles.

Il y a 7 relations de base, et leurs inverses :

Les Rélations

Pour les intervalles t et s, les 7 relations de base sont :

nom	anglais	notation	Schéma	définition
égal	equal	t = s	|-t-| |-s-|	$(t^- = s^-) \wedge (t^+ = s^+)$
avant	before	t < s	|-t-| |-s-|	$t^+ < s^-$
recouvrement	overlap	t o s	|-t-| |-s-|	$(t^- < s^-) \wedge (t^+ > s^-) \wedge (t^+ < s^+)$
rencontre	meets	t m s	|-t-| |-s-|	$t^+ = s^-$
pendant	during	t d s	|-t-| |---s---|	$(t^- > s^-) \wedge (t^+ < s^+)$
départ	starts	t s s	|-t-| |---s---|	$(t^- = s^-) \wedge (t^+ < s^+)$
fin	finishes	t f s	|-t-| |---s---|	$(t^- > s^-) \wedge (t^+ = s^+)$

Les relations inverses sont notées :

Relation	Inverse	
t < s	t > s	
t = s	t = s	(Nota : = est symétrique)
t o s	t oi s	
t m s	t mi s	
t d s	t di s	
t s s	t si s	
t f s	t fi s	

Les relations entre des intervalles sont représentées par un réseau de symboles.

Les flèches entre les intervalles sont étiquetées par la liste des relations possibles.

Il est possible de simplifier par remplacant pendant, départ et fin
par une relation "dur"
et de remplacer leur inverses par
pendant inverse, départ inverse et fin inverse par "cont"

l'Incertain :

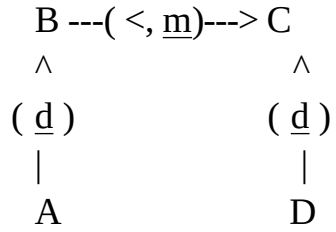
L'utilisation d'une liste permet de gérer l'incertitude par des contraintes.

Exemples :

t_1 d t_2	t_1 --(d)--> t_2
(t_1 d t_2) ou (t_2 d t_1) ou ($t_1 < t_2$)	t_1 --(d, di, <)--> t_2
($t_1 < t_2$) ou ($t_1 < t_2$) ou (t_1 m t_2)	t_1 --(<, >, m)--> t_2

Quand un nouvel intervalle est introduit dans le réseau, toutes ses conséquences sont calculées par propagation des contraintes par transitivité :

par exemple, soit :



on a tout de suite A --(<)--> D

Table de Transitivité

La propagation est définie par un tableau 12 x 12 des relations de "transitivité".
(= est omis).

exemple : pour (A < B) et (B ? C)

	(B ? C)	Contrainte sur (B ? C)
(A < B)	(B < C)	(<)
(A < B)	(B > C)	No Info
(A < B)	(B d C)	(< o m d s)
(A < B)	(B di C)	(<)
(A < B)	(B o C)	(<)
(A < B)	(B oi C)	(< o m d s)
(A < B)	(B m C)	(<)
(A < B)	(B mi C)	(< o m d s)
(A < B)	(B s C)	(<)
(A < B)	(B si C)	(<)
(A < B)	(B f C)	(< o m d s)
(A < B)	(B fi C)	(<)

Propagation de Contraintes

Le tableau de transitivité est utilisé afin de développer les relations possible de chaque paire d' intervalles.

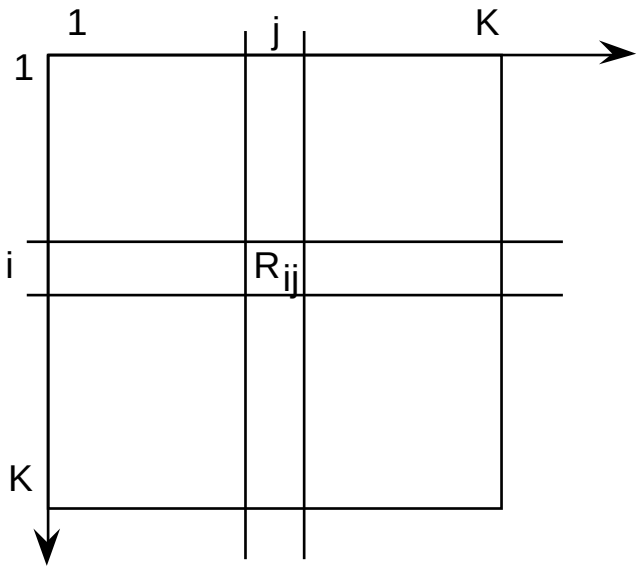
Soient A --(R_{AB})--> B --(R_{BC})--> C

Relations Possible de A --(R_{AC})--> C R_{AC} = Transitivite(R_{AB},R_{BC})

Transitivite (R_{AB},R_{BC})
R_{AC} <- NIL;
r_{ab} R_{AB}
r_b R_{BC}
R_{AC} := R_{AC} $T(r_{ab}, r_{BC})$;
RETURN R_{AC};

Quand une nouvelle relation est affirmée, il faut propager les contraints sur les autres relations.

Pour K intervalles, on peut voir le reseau comme une tableau de K x K relations. Chaque entre du tableau est la liste de relations d'intervalle i vers intervalle j.



L'affirmation d'un nouveau relation permet de reduire les listes de relations possible.

Par exemple, s'il est affirmé que les relations possible entre A et B est la liste

NewR_{AB}, on remplace R_{AB} par NewR_{AB}.
Ceci impose les nouveau contraints sur tout les autres relations

Soient A --(R_{AI})--> I --(R_{IJ})--> J
^----------(R_{AJ})----------^

Relations Possible de A--(R_{AB})--> B

Propogstate (RAB)
interval i
interval j
R_{Aj} := R_{Ai} ∩ Transitivite (R_{Ai},R_{ij});

Il faut propager les contraints sur tout le réseau.
Ily a $(N-1)^2/2$ pairs d'intervalles.

Coût : pour N intervalles, $O(N^2)$ opérations.

Pour le réduire, Allen applique une structure hiérarchique sur les intervalles.

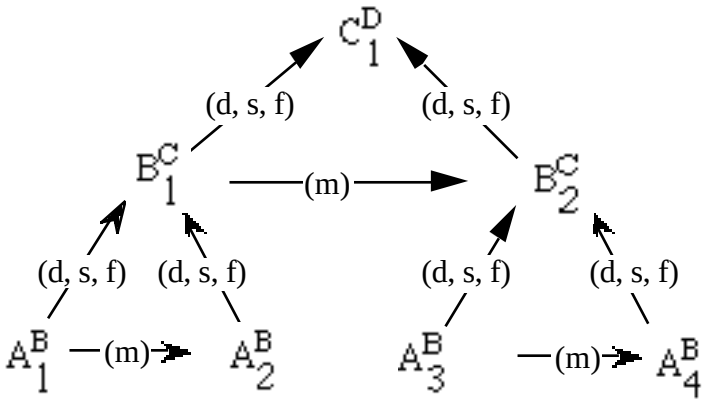
Les intervalles de référence :

Un intervalle de référence est un "ensemble" d' intervalles
tel que les relations avec les autres "ensembles" sont complètement déterminées.

Il s'agit d'une hiérarchie d' intervalles.

Notation : A_k^B : Interval A_k de la reference B.

Exemple :



Si deux intervalles ne sont pas liés directement, il suffit de parcourir le graphe pour chercher un chemin entre eux.

Exemples

1) **:** Soient les relations suivantes dans une logique de relations temporelles

Événement A rencontre événement B : (A m B)
Événement B rencontre événement C : (B m C)
Événement D est après événement A : (D > A)
Événement D est avant événement C : (D < C)

a) Quelles sont les relations possibles entre D et B obtenues par transitivité avec A?

T(D > A, A m B) = (d, f, oi, mi, >)

b) Quelles sont les relations possibles entre D et B obtenues par transitivité avec C?

T(D < C, C mi B) = (<, o, m, d, s)

c) Quelles sont les relations possibles entre D et B après une propagation de contraintes?

(D d B)

Representation Structurée de la Connaissance

Intelligence : (Petit Robert)

"La faculté de connaître et comprendre.

On a vu que Connaissance = Compétence.

Qu'est que c'est à comprendre?

Association entre choses perçues et choses connus.

Pour comprendre une scène ou une histoire, il faut trouver une correspondance
Entre les éléments perçus et les éléments qui représente les connaissances. Les schemas fournissent une représentation déclarative pour la connaissance.

Elle permet de raisonner et comprendre les scènes, les histoires (orales ou écrites), les textes, les idées ou les plans.

L'outil principal de programmation de schémas est la programmation orientée objet. Les concepts de représentation de connaissance structurée et programmation orientée objet ont évolué en parallèle dans l'IA et le génie logiciel, avec une influence mutuelle.

En Génie-logiciel on a vu dévelopé

Simula-67 (1967)
SmallTalk (Goldberg, 1973)
Flavors : Un outil "programmation par Objet" en Common Lisp.
Eiffel

Les outils modernes incluent C++ et Java.

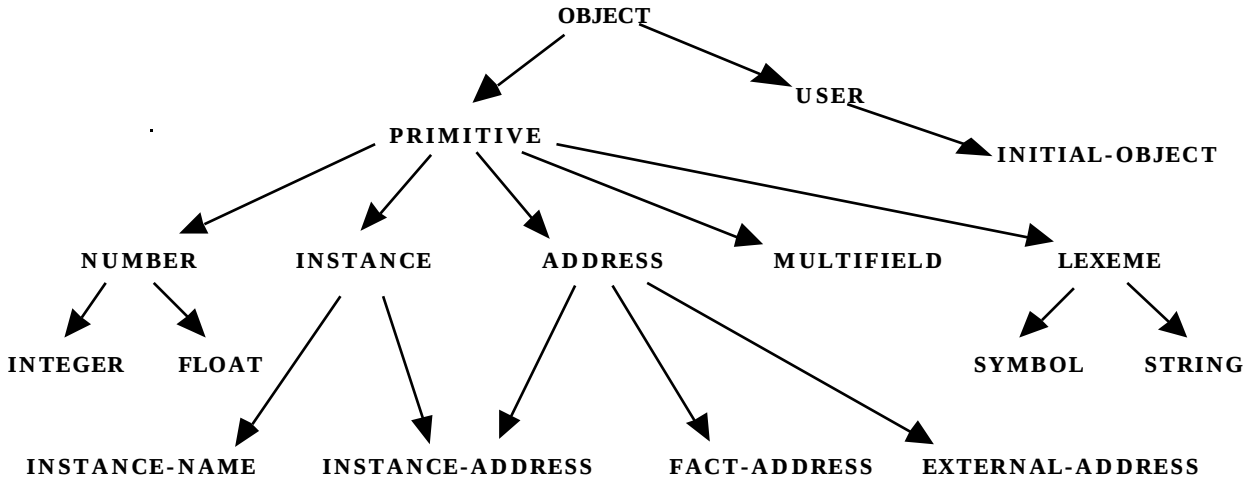
En Intelligence Artificielle on a vu dévelopé

Schémas (Psychologie Cognitive, 1930's)
Réseau Sémantique : (Quinlan, Carbonnel 1968).
Frames : (Minsky, 1970's)
Units : (KRL 1970's)
KRL -> KEE -> COOL (CLIPS 5)

Définition des Classes en CLIPS

Rappel que en MYCIN les "faits" était fortement typé par une structure de donnée. Le même idéa se trouve dans le programmation orienté objet. Les propriétés sont decrits par les "facets".

Les Classes prédefini en clips :



CLIPS> (list-defclasses)

Une classe est définit par une expression :

```
(defclass <name> [<comment>]
  (is-a <superclass-name>+)
  [<role>]
  [<pattern-match-role>]
  <slot>*
  <handler-documentation>*)
```

(1) A (defclass) must have a class name, <class>.
(2) There is at least one superclass name, <superclass>, that follows the is-a.
(3) A (defclass) has zero or more slots.
(4) Each slot has zero or more **facets**; types, <facet>, which describe the characteristics of the slot.

Exemple :

```
(defclass PERSON (is-a USER) (role concrete)
    (pattern-match reactive)
    (slot nom  (default "John Doe")
             (create-accessor read-write))
    (slot age (create-accessor read-write))
    (multislot address (create-accessor read-write))
)
```

```
(defclass ENSI (is-a PERSON) (role concrete)
    (slot year (create-accessor read-write))
    (slot OPTION (default MD) (create-accessor read-write))
)
```

En CLIPS, une schéma est défini par un instance d'une classe

```
(make-instance <instance> of <class> (<slot> <value>)...)
(defclass PERSON (is-a USER)
   (slot name)
   (multislot address)
   (slot age)
)

(make-instance [Jim] of PERSON (name "jim")(age 18))
[Jim]  ; CLIPS returns the name of the instance made.

CLIPS>(send [Jim] get-age)
18
```

la définition d'une slot entraine le définition des "methodes" (Handlers en CLIPS)

put-<slot>	écrire une valeur
get-<slot>	lire une valeur
init-<slot>	initialiser une valeur.

Héritage des définitions des classes

Les classes et instances peuvent être organisés en hiérarchies qui permettent de définir les slots, les valeurs et les méthodes par défaut.

Héritage simple : Une superclass par classe
Héritage multiple : Plusieurs superclass par classe.

Héritage simple

Par exemple,

```
(defclass A  (is-a USER))
(defclass B  (is-a A))
(defclass C (is-a A))
(defclass D (is-a A))
```

Héritage multiple

<subclass> (is-a <superclass1> <superclass2> ... <superclass-n>)

S'il y a plus qu'une superclass, l'héritage est dit "multiple".

La liste "is-a" est le "**class precedence list**"
L'héritage donne priorité selon l'ordre dans cette liste.

Ordre : Spécifique -> Générale

Règle de l'héritage Multiple

1) Une classe à priorité sur ses superclass
2) Gauche à droite dans la liste "is-a"

L'héritage est appliqué à les superclass dans une sorte de recherche en profondeur d'abord.