Annexe 4

Langage Mvl

A.4.1. Rappels sur les treillis et bitreillis

Un treillis est une structure mathématique définie soit à partir d'un triplet $(T, \vee, \wedge)$ où T est un ensemble et $\vee$, $\wedge$ deux lois binaires internes idempotentes commutatives associatives et vérifiant l'«absorbtion»

$$[\forall\, x, y \quad x \wedge (x \vee y) = x \quad \text{et} \quad x \vee (x \wedge y) = x]$$

ou alors, un treillis est défini à partir de $(T, \leq)$ où $\leq$ est une relation d'ordre réticulée [$\forall x, y\ \exists$ inf et sup de $\{x, y\}$], les opérations sont alors $x \wedge y = \inf \{x, y\}$ et $x \vee y = \sup \{x, y\}$. Réciproquement pour un treillis $(T, \vee, \wedge)$, on parlera de l'ordre associé défini par $x < y \Leftrightarrow x \wedge y = x$.

ORDRE INDUCTIF : ordre tel que toute chaîne est majorée, tout ensemble inductif possède un max (axiome du choix version Zorn).

ORDRE NOETHÉRIEN : ordre où toute chaîne admet un max (toute suite croissante est stationnaire). Pour la décroissance, on parle d'ordre artinien.

BON ORDRE : ensemble ordonné où toute partie possède un plus petit élément. L'ordre est alors total et tout élément possède un successeur.

TREILLIS COMPLET : ensemble ordonné tel que toute partie possède un «inf» et un «sup».

TREILLIS DISTRIBUTIF : $\vee$ $\wedge$ sont mutuellement distributives

TREILLIS MODULAIRE : treillis tel que l'on ait :

$\forall$ a, b, c $\quad a < c \Rightarrow a \vee (b \wedge c) \geq (a \vee b) \wedge c$, (tout treillis distributif est modulaire).

TREILLIS COMPLÉMENTÉ : treillis possédant deux éléments (notés 0 et 1) tels que $\forall x \ \exists \neg x$ (appelé complément de x) $x \wedge \neg x = 0$ et $x \vee \neg x = 1$

0 est alors minimal pour $\leq$, neutre pour $\wedge$ et absorbant pour $\vee$

1 est maximal pour $\leq$, absorbant pour $\wedge$ et neutre pour $\vee$

TREILLIS DE BOOLE : treillis distributif et compémenté, c'est alors avec la loi définie par $a + b = (a \wedge \neg b) \vee (\neg a \wedge b)$ et $\wedge$, un anneau unitaire commutatif et idempotent.

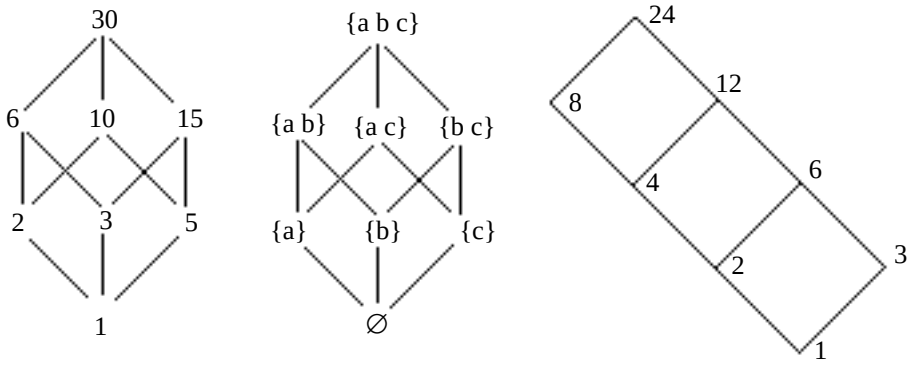

Figure A.4.1 Exemples d'algèbres de Boole Soit l'ensemble des diviseurs de 30 pour la relation de divisibilité, 1 est le minimum, 30 le maximum, $\vee$, $\wedge$ sont les opérations pgcd, ppcm (figure gauche). Soit, maintenant l'ensemble des parties de {a, b, c} pour la relation d'inclusion, $\vee$, $\wedge$ sont les opérateurs $\cap$, $\cup$ (figure centrale). On voit l'isomorphisme de ces deux structures, différentes par ailleurs du treillis des diviseurs de 24 (à droite).

IDÉAL I : c'est une partie stable pour $\vee$ et permise pour $\wedge$ (section commençante de $\leq$ stable par $\vee$). Un idéal premier est un idéal maximal $\forall x \quad x \in I$ ou $\neg x \in I$
FILTRE F : partie stable pour $\wedge$ et permise pour $\vee$. Un ultrafiltre est un filtre maximal $\forall x \quad x \in F$ ou $\neg x \in F$
ALGÈBRE MONADIQUE [Halmos 62] : algèbre de Boole possédant une application notée $\exists$ normalisée, $\exists\, 0 = 0$ extensive $p \leq \exists\, p$ et quasi-multiplicative $\exists\, (p \wedge \exists\, q) = \exists\, p \wedge \exists\, q$
Cette structure est utilisée en logique formelle pour asseoir les notions de modèle et d'interprétations d'une théorie :
On montre que $\exists$ est idempotente, croissante et réalise un morphisme pour $\vee$.
Un idéal monadique étant un idéal stable par $\exists$, une logique monadique est la donnée d'une algèbre monadique et d'un idéal monadique I.
I est appelé l'ensemble des antithèses et $T = \{p / \neg p \in I\}$ qui est un filtre, est l'ensemble des thèses.
(B, I) est complète $\Leftrightarrow$ I maximal $\forall p,\ p \in I$ ou $\neg p \in I$
(B, I) est consistant $\Leftrightarrow$ I propre $\forall p$, non($p \in I$ et $\neg p \in I$)
Un modèle est une logique monadique où I est vide, et une interprétation est un homomorphisme d'une algèbre monadique dans un modèle, qui est telle que l'image de I soit 0.

BITREILLIS

(B, $\wedge$, $\vee$, .,+, $\neg$) est un bitreillis si et seulement (B, $\wedge$, $\vee$) et (B, .,+) sont des treillis et si $\neg$ est une involution de B réalisant un isomorphisme de $\wedge$ sur $\vee$ de $\vee$ sur $\wedge$ et de . et + sur eux mêmes.
La «négation» est la symétrie par rapport à la droite (I TF) dans les figures suivantes. Certains auteurs ont introduit une négation faible comme symétrie par rapport à la bissectrice.

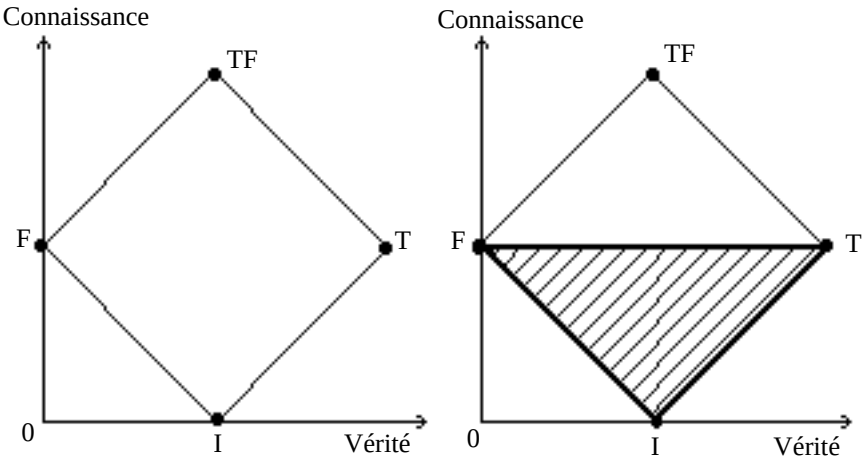

Figure A.4.2 Le plus petit bitreillis non trivial est celui de la logique booléenne [Belnap 77] (il comprend 4 éléments) : T (vrai), F (faux), I (incertain), TF (contradiction).
La théorie des possibilités correspond aux points des segments FI et IT, les logiques multivaluées à ceux du segment FT, et la théorie des probabilités haute et basse à la zone hachurée (I et TF faisant partie à chaque fois du bitreillis).

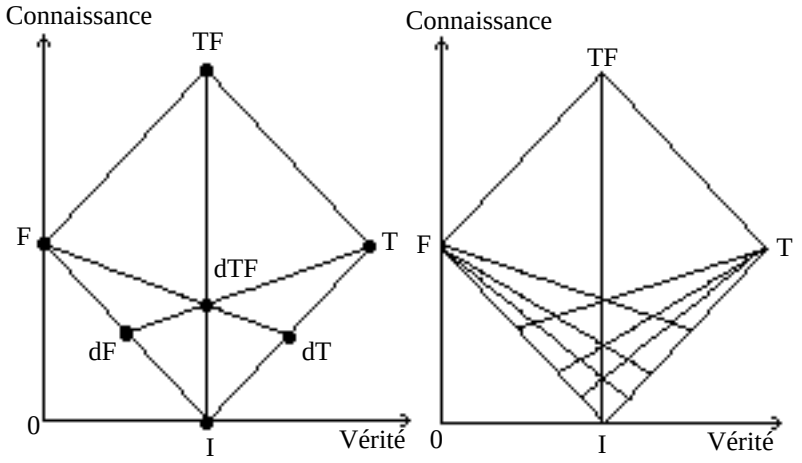

Figure A.4.3 En ce qui concerne la logique des défauts elle peut être représentée par le bitreillis {T, F, I, TF, dT, dF, dTF} à gauche, et la logique dénombrable des défauts ou défauts hiérarchiques à droite par 3 chaînes notées T, dT, d2T, d3T .... F, dF, d2F, d3F,..... et TF, dTF, d2TF, d3TF,.....

Dans le système Mvl [Ginsberg 88, 91], [Desvignes 92] ces quatre opérations sont appelées respectivement «and», «or», «dot» et «plus».

Les ordres partiels associés seront notés $\leq_t$ et $\leq_k$ afin d'ordonner suivant une vérité croissante et une connaissance croissante.

BITREILLIS BASÉ SUR DES MONDES

Soit W un ensemble de «mondes» (intuitivement des univers possibles), on dira que la proposition P du langage L a la valeur de vérité (U, V) avec U, V inclus dans W, si P est vrai de façon booléenne dans chaque élément de U et faux dans chacun de ceux de V.
Pour un tel couple (U, V), un monde de $U \cap V$ est dit inconsistant alors qu'un monde de $W - (U \cup V)$ est dit indéterminé.

En posant : $(U, V) \leq_t (U', V') \Leftrightarrow U \subset U'$ et $V' \subset V$

et : $(U, V) \leq_k (U', V') \Leftrightarrow U \subset U'$ et $V \subset V'$

on obtient un bitreillis $BW = P(U)*P(W)$ dans lequel les éléments distingués sont F = (Ø, W) pour le faux, T = (W, Ø), I = (Ø, Ø) l'indéterminé et enfin TF = (W, W) la contradiction.
Cette façon de construire un bitreillis est particulièrement adaptée aux systèmes de maintien de la cohérence ATMS [Doyle 79], [DeKleer 86].
Dans un tel bitreillis, à chaque proposition P, on associera le couple (Jp, J¬p) formés pour Jp par les «justifications» de P, ainsi si $Q \rightarrow P$ et $R\&S \rightarrow P$ sont des règles, on aura Jp = ({Q}, {R, S}) formé par deux justifications possibles amenant à P, et pour J¬p, par les justifications de ¬P.

Les relation d'ordres sont donc :
$J \preccurlyeq J'$ (J moins général que J') $\Leftrightarrow J' \sqsubset J$
$P_1 \leq_t P_2$ (plus de vérité pour P_2) $\Leftrightarrow (J_1, K_1) \leq_t (J_2, K_2) \Leftrightarrow J_1 \preccurlyeq J_2$ et $K_1 \succcurlyeq K_2$
$P_1 \leq_k P_2$ (plus d'information pour P_2) $\Leftrightarrow (J_1, K_1) \leq_k (J_2, K_2)$
$\Leftrightarrow J_1 \preccurlyeq J_2$ et $K_1 \preccurlyeq K_2$
Les hypothèses qui ont une justification vide sont des éléments minimaux pour $\leq_t$, les axiomes et leurs déductions sont maximaux pour $\leq_t$.
Plus précisément, se plaçant donc dans la logique des propositions ordinaire avec (T = vrai, F = faux, I = incertain, TF = contradictoire), (il ne sera pas question de la négation c'est à dire qu'au lieu d'avoir systématiquement [A et non (A) $\rightarrow$ TF], on aura seulement des règles particulières [A et B $\rightarrow$ TF]. Dans le fonctionnement du système, on met des axiomes (des faits ayant une justification vide), des hypothèses (des faits H ayant (H) comme justification), et une base de règles sous forme de clauses de Horn [A_1 et A_2 et ... $\rightarrow$ A] est une règle, alors cela signifie que (A_1 A_2 ...) sera une justification pour A.
Un environnement est une conjonction de propositions, il est dit consistant s'il n'entraîne pas de contradiction.
Une justification est une liste d'hypothèses.
Un label L(A) pour une proposition A est une disjonction de justifications (les différents «mondes» qui permettent de déduire A)
L(A) est minimal si aucune des justifications présentes n'est sur-ensemble d'une autre (ce sont les plus générales et il n'y a pas de redondance).
L(A) est consistant si chaque justification présente est consistante.

L(A) est complet si tout environnement consistant où A peut être déduit est un sur-ensemble d'une des justifications déjà présente. (aucune justification n'est oubliée)
L(A) est correct (ou sain ou encore bien construit) si chaque justification présente entraîne A.
Le rôle du système est de mettre à jour la liste de justifications de chaque assertion, en en faisant un label sain, consistant minimal et complet, et de le recalculer chaque fois que l'on rajoute une justification c'est à dire une nouvelle règle. Par exemple, si on rajoute (A, B) comme justification de TF, cela signifie que A et B ne peuvent plus coexister.

A.4.2. Le langage de programmation MVL (Multi Valued Logics)

Il s'agit, comme pour FRIL, d'un Prolog fonctionnant essentiellement en chaînage-arrière, mais où les "valeurs de vérité» des propositions peuvent être de type très divers y compris calculées par exemple I ou (T et F) pour la valeur dT, et le mode d'inférence choisi selon différentes logiques non classiques.
Afin d'illustrer le fonctionnement de MVL, nous reproduisons l'exemple du manuel de référence sans trop insister sur les détails de syntaxe et de codage un peu particuliers. Signalons seulement que les clauses, comme en Prolog, se présentent comme clauses de Horn avec la conclusion en tête et parenthèsées, sous la forme :
(<= C P_1 P_2 ... P_n) et pour le chaînage arrière, la conclusion C étant remplacée par les prémisses P_1 ... ou bien (=> C P_1 P_2 ... P_n) pour le chaînage avant non utilisé ici, ou encore (if C P) pour une règle $P \rightarrow C$ et sa contraposée $\neg C \rightarrow \neg P$.

```
: value true              ; permet de définir comme vraies les clauses qui vont suivre
(idn P)                   ; fonction renvoyant la valeur de vérité de la proposition P
(bcs 'P)                  ; permet de poser un but, de procéder à la résolution de P
(contents)                ; donne la liste des clauses chargées
(denotes n)               ; renvoie la clause de numéro n
(stash 'P : value false)  ; permet de rajouter une clause P avec sa valeur («assert»).
(load-logic)  ; permet au début d'une utilisation, de choisir la logique (premier ordre,
ATMS, défauts, défauts hiérarchiques, temporelle, probabiliste)
```

EXEMPLE 1 LE SYSTÈME ATMS (MAINTIEN DE LA COHÉRENCE)

Les valeurs de vérité déclarées sont alors les constantes f, t, i, tf, ou bien des couples de la forme (Jp, J¬p) ou encore nil pour des faits non justifiés (les hypothèses).
On définit alors le programme P_0 comme la suite des clauses définissant un petit arbre généalogique :

```
(femme pam)          (femme pat)          (femme ann)
(femme liz)          (homme tom)          (homme bob)
(homme jim)          (parent pam bob)     (parent tom bob)
(parent bob ann)     (parent tom liz)     (parent bob pat)
(parent pat jim)
(<= (enfant x y) (parent y x))
(<= (mere x y) (parent x y) (femme x))
(<= (ancetre x y) (parent x y))
(<= (ancetre x y) (parent x z) (ancetre z y))
```

```
(<= (diff x y) (not (equal x y)))
                ; ce sont ici les fonctions lisp «not» et «equal» qui sont utilisées
(<= (soeur x y) (parent p x) (parent p y) (femme x) (diff x y))
(<= (frere x y) (parent p x) (parent p y) (homme x) (diff x y))
(if (pauvre x) (not (riche x)))
(if (vivant x) (not (mort x)))
(if (triste x) (not (heureux x)))

(riche bob)                          ; un nouveau fait assez simple
(<= (riche x) (enfant x y) (riche y))

: value nil       ; les règles A, B, C, D sont de contexte vide, ce sont les hypothèses
(not (riche x))  ; règle A, tout le monde est pauvre a priori
(not (mort x))   ; règle B
(<= (heureux x) (riche x) (vivant x))                          ; règle C
(<= (triste x) (mort y) (vivant x)
      (or (frere x y) (soeur x y) (ancetre x y) (ancetre y x)) )      ; règle D
```

On pose la question (riche bob), la réponse fournie par le système est ((nil) (A)) ce qui est conforme.
On rajoute alors le fait (mort bob)
En reposant la question (riche ann), la réponse est encore ((nil) (A)).
pour la question (triste ann), la réponse ((B D) (B C)) signifie qu'avec les hypothèses B et D, Ann doit être triste, mais qu'avec les hypothèses B et C, elle est heureuse.

EXEMPLE 2 LA LOGIQUE DES DÉFAUTS

Les seules valeurs de vérité sont ici les constantes f, t, i, tf, df, dt, dtf.
On prend le même programme P_0 avec la valeur t, et les mêmes règles A, B, C, D mais avec la valeur dt (vrai par défaut).
A la question (riche x), le système répond x = bob avec la valeur vraie.
A la question (not (riche x)), il donne toutes les personnes (y compris bob) avec la valeur dt (il ne va donc pas vérifier le contraire d'un but automatiquement).
A la question (heureux x), la réponse est x = bob (valeur dt)
On introduit alors (mort bob) et on demande (riche x), la réponse est x = pat, ann, bob : vrai (ce dernier reste riche tant qu'une nouvelle clause n'est pas présente pour dire qu'un mort n'est plus riche).
La demande (heureux x) fournit x = pat, ann (valeur dt) à cause de C.
La demande (not (heureux x)) donne x = pat, ann, pam, liz, jim, tom (valeur dt) par la règle D.

EXEMPLE 3 LA LOGIQUE DES DÉFAUTS HIÉRARCHIQUES

Les valeurs sont les mêmes avec dt_1 df_1 et dt_2, df_2 ...
Les règles de P_0 sont les mêmes (valeur vraie) A, B, D sont données avec la valeur dt1, enfin C est donnée avec dt2.
A la question (heureux x), la réponse est toujours x = bob (valeur dt2)
On introduit alors (mort bob) et on demande (heureux x) qui ne donne aucune réponse car «triste» devient prioritaire. Enfin la demande (not (heureux x)) donne encore x = pat, ann, pam, liz, jim, tom (valeur dt1), on a donc une connaissance plus fine qu'avec l'exemple 2.

EXEMPLE 4 LOGIQUE TEMPORELLE

Les valeurs sont les constantes ainsi que des entiers par exemple (true-at 4) signifiant vrai à l'instant 4. Ainsi peut-on déclarer une même proposition P (true-at t_0) puis (false-at t_1), un prédicat prédéfini permet d'indiquer la persistance d'une valeur de vérité dans l'intervalle de t_0 à t_1 - 1 c'est (propagate P).
(delay t P) est vrai en t + t_0 dès que P est vrai en t_0
Le programme est cette fois constitué par les mêmes clauses vraies de P_0 sauf les deux dernières et par :

```
: value true
(heritage bob 6)            ; dans 6 mois, puis la règle A est remplacée par :
(<= (riche x) (enfant x y) (propagate (riche y)) (heritage y) (delay t (mort y)))
(<= (heureux x) (riche x) (propagate (vivant x)))        ; règle B
(<= (triste x) (propagate (vivant x)) (propagate (mort y)) (mere x y))
      ; règle C la mère reste toujours triste de la mort de son enfant

(<= (triste x) (propagate (vivant x)) (or (frere x y) (soeur x y) (ancetre x y)
      (ancetre y x))
      (or (mort y) (delay 1 (mort y)) (delay 2 (mort y))
      (delay 3 (mort y)) (delay 4 (mort y))))
      ; règle D les autres membres de la famille observent 5 mois de deuil
: value (true-at 0)
(riche bob)
(vivant x)
: value (true-at 3)
; la proposition est différée 3 mois après, on aurait pu dire aussi : value (true-at 6)
(heritage bob)
(mort bob)

: value (true-at 20)
(mort jim)
```

La question (triste pat) fournit les valeurs de vérité étalées dans le temps :
(0 i) (3 t) (6 tf) (7 t) (8 i) (20 t)
En effet en 6 on a le deuil plus l'héritage qui amène une contradiction de sentiments pour Pat, à 7 mois le deuil dure toujours, à 8 mois rien ne permet comme en 0 de se prononcer, et à 20 mois la mort de son fils Jim provoque à nouveau la tristesse.

LOGIQUE PROBABILISTE

Pour une règle A et B et C et D et ... $\rightarrow$ Q la probabilité est calculée :
par p(Q) = p(A).p(B).p(C)...
Si P $\rightarrow$ Q et R $\rightarrow$ Q sont 2 règles de même conclusion alors l'indépendance est supposée et p(Q) = p(P) + p(R) - p(P).p(R). Mais la version actuelle ne possède pas $p(\neg A) = 1 - p(A)$.