CLIPS によるエキスパートシステムの構築

準備：　ダウンロード
以下から "CLIPSWin.zip" をダウンロードし、解凍するだけ。
http://www.filewatcher.com/m/CLIPSWin.zip.328828.0.0.html

参考資料
(1)唯一の日本語
http://www.iluminado.jp/businessruleruleengine/rule-base-etc/25-clips1.html
(2)最も分かりやすい入門
http://math.hws.edu/eck/cs453/s03/about_clips.html
(3)要約的
http://www.cse.msstate.edu/~hansen/classes/AIspring04/slides/program3.pdf
(4)例が多い入門（１～５）
http://iweb.tntech.edu/bhuguenard/ds6530/ClipsTutorial/CLIPS%20tutorial%201.htm
(5)レファレンスマニュアル（必要な時必要な所を読む、全 402 ページ）
http://clipsrules.sourceforge.net/documentation/v630/bpg.pdf

（Ｉ）CLIPS の人頭アイコンWクリックで Window が開き、CLIPS>とプロンプトが出た所から話を始める。

STEP1:Rule Base の Expert System を使う場合、ユーザは次の２つを用意する。
(1)Fact リスト：推論のためのデータ（ワーキングメモリに格納される）
Fact リストの読み込みは、
CLIPS>　の後に手入力でもよいし、手入力の時は、(assert (....))とする。
CLISP> (load-facts "～.txt")で読める。
(2)Knowledge Base：Rule 集（～.clp　というファイル名にする）
Knowledge Base の読み込みは、手入力のほか、Window のファイルメニューからも読み込める(Load)し、
CLISP> (load "～.clp")
としてもよい。

STEP2：次は実行、
CLIPS>(run)
run は、agenda が空になるまで実行を繰り返す。
(halt)：でも実行を止めることが出来る。

STEP3：終了は、
CLIPS> (exit)

この他、重要なコマンドは、

(clear)：メモリから全 Knowledge Base と Fact を消す。Shutting Down と RestartCLIPS に相当。
(reset)：Fact だけをメモリから削除、agenda は reset。

(initial-fact)：ワーキングメモリは空だと CLIPS が走れない。reset が行われると、ワーキングメモリが空になるので、初期 Fact(initial-fact)を入れる必要がある。

(facts)：ワーキングメモリにある全 Fact を表示（ワーキングメモリの中身は window でも見られる）

(rules)：全ルールを表示

(agenda)：現在 agenda にあるルールを表示（agenda には実行可能なルールが入り、window でも見られる）

デバッグ用には、
(watch all)：全デバッグメッセージが得られる。

(set-break rule-name)：rule-name で示されるルールがブレークポイントになる。

(remove-break rule-name：set-braek でセットしたブレークポイントを削除する。

（準備１）CLIPS では全てが１つのリストになっていて、式は prefix 形式になっている。prefix とは、例えば、
(father chichiro ichiro)：父郎は一郎の父
(+ 2 3)： "2　足す　3" を計算
といった具合に、最初に演算子がくる。

もう少し厳密に言うと、()内がリストで、リスト内の最初の項目が関数とか命令を示し、２項目以降は引数である。

（準備２）リストの要素（atom）を作る時に注意が必要な文字は次の通りである。
"で始まる行はコメント行になる
<,?,$,~,|,&,,：後述
"...."：文字列

（ＩＩ）ＦＡＣＴ
(1)Fact の 2 つの型がある。

(a) 型１：例 (mother marie irene) や (data 3 6 1) といったもので、（test,and,or,not,declare,logical,object,exsists,forall 以外の）シンボルで始まる。
() 内の順序に意味がある（mother irene marie)では親子が逆になる。

(b)型２："テンプレートとスロット"と呼ばれる。例、
(family (father pierre) (mother marie) (child irene))
family はテンプレート名、father,mother,chaild はスロットと呼ばれ、pierre,marie,irene はスロットの値と呼ばれる。データはスロット値で識別できるので、並びの順序は意味がない。

(2)Fact をワーキングメモリに読み込む方法
次の３つが、
(a)コマンドラインから、
CLIPS> (assert (mother marie irene))
という風に入れる。

(b)次のような Fact のリスト、"～.txt "を作っておき、
CLIPS> (load-fasts "～.txt"))
とする。
＝＝
Fact のテキストファイルは、
(mother marie irene)
(father pierre irene)
の様に、Fact だけを書き"assert"は書かない
＝＝
（注：(save-facts "～.txt")で保存可能）

(c)次のような deffacts ファイル（～.clp）を作って、File メニューの Load で読み込んでおき、
CLIPS>(reset)で読み込む.

===
(deffacts def
(mother marie irene)
(father pierre irene)
)
===

(3)Fact の削除
CLIPS > (retract ルール番号)
ルール番号は、
?rn <- ルール
（例えば、?rn <- (mother marie irene)）
で、変数　?rn　に Fact(mother marie irene)の番号が得られる。

（ＩＩＩ）ルール
プロダクションシステムには普通複数の If...Then___ルールが存在する。
（１）　If...Then___　ルールの形
Ｉｆ～～～Ｔｈｅｎ_____（プロダクションルール）は、CLIPS では次の様に表現される。

(defrule ルール名
　　条件部
　　=>
　　アクション部
)

この条件部を RHS（左辺）、アクション部を LHS(左辺)と呼ぶこともある。
===
プロダクションシステムでは、条件部が真のルールのアクション部を実行する。
例えば、
(defrule　rule-room
　(love i you)

=>
　　(printout　t　"you love me" crlf)
)
は、もしワーキングメモリに(You love me）という Fact が入っていると、I love you.と出力する、という意味である。

===

（２）条件部は、
(.....)
(.....)
(.....)
の様に（....）の並びである。

（....）には次の２つのタイプがある。

(a)ワーキングメモリに入っている Fact とマッチングを取るパターン
上の(love i you)が、これに当たる。

(b)(test (eq　?x　?y))などの TRUE,FALSE を返す命令
（ここでは感じをつかんでおき、詳細は次章終了後見るとよい。
）、
（注：?x、?y の様に?で始まる文字列は変数(次章で説明)で、上は２つの変数?x と?y の値が等しいかを test している）test の他に、not, or, and, exists, forall, logical がある。

・test は、後に続く式の真偽をテストするコマンドである。test の中で使える演算子は次の通りである。
ep：等しい（例：(test(eq ?a ?)）：変数?a と?b の中身が同じかどうか、答えは TRUE または FALSE）
neq：等しくない
以下は数値の比較演算子、
>:（例:　test(> ?a ?b)：変数 ?a　の値が ?b　の値より大きければ真）
<:（例:　test(< ?a 37)：変数 ?a　の値が 37 より小さければ真）
この他、=<,　<>,　>=,　<= が使える。

・not は、次の形にマッチする Fact がワーキングメモリにあるか否か(無ければ真)

例：　(not (father ?a ?b))：ワーキングメモリに(father ?a ?b)にマッチする Fact があるか否か。（注：無ければ真）

・or は、”または”、
例：　(or (mother sazae ?) (mother ? sazae))：サザエは母を持つか、母なら真。

その他、・and, ・exists, ・forall,・logical　は、必要なら自分で調べること。

・"="は、実際に計算する、ことを意味する。

・"<-"は、Fact の番号を求めるのに使用する。例えば、
?idx <- (mother fune sazae)
なら、(mother fune sazae)という Fact の番号（アドレス）が変数?idx に入る。

(c)変数値への制約
・"｜"は、or を意味する：例(color red|green|blue)は、(color red)or(color green)or(color blue)の意味。
・"~"は、anything but を意味する：例(color ~red)は、赤以外、の意味。
・"&"は、制約の結合を意味する：例(color ?c&~black&~white)は、変数?c に入るのは白黒以外。
・":"は、関数の呼び出しを意味する：例（atai ?a&:(> ?a 0)　?b&:(>= ?b ?a))は、
(atai 10 25) にはマッチするが、(atai 0 1）にはマッチしない。

（３）アクション部は、
(.....)
(.....)
(.....)
の様に、行動を（...）の並びで書く。
（....）には次の２つがある。
(a)ワーキングメモリに Fact を書き込む。
例えば、
(assert (I love you))

(b)プリント文

(printout t "Hello")
t は、標準出力を表す。

（ＩＶ）ルールの条件部(LHS とも言う)のマッチングと変数
CLIPS の変数は、PROLOG の変数と同じで、C 言語の変数と扱いが異なるので注意が必要。
変数は、文字列の前に？を付ける。例：?x, ?data など。

準備：ワーキングメモリには、次の Fact が入っているものとする。
＝＝ワーキングメモリ＝＝
(mother fune sazae)
(mother sazae tara)
(father masuo tara)
(mothr ukifune fune)
＝＝

（１）先ずは、変数を使わないルールの場合
If...Then...ルールは、ワーキングメモリに、
(mother fune sazae)
(mother sazae tara)
が有ったら、新たな Fact　(grandma hune tara)を作る、というものとします。
ルールは、以下の通りである。
```
(defrule rule-room
        (mother fune sazae)
        (mother sazae tara)
=>
        (assert　(grandma fune tara))
)
```

（２）変数を使う場合
```
(defrule rule-room
        (mother ?g ?m)
        (mother ?m ?c)
=>
        (assert　(grandma ?g ?c))
)
```
と書ける。

CLIPS の推論エンジンは、ワーキングメモリから変数（?g,?m.?c）に合う Fact を探し出してくれる。同じ変数名にはおなじ項目が対応付けられる。
ゆえにこれを実行すると、ワーキングメモリに新たな Fact
(grandma fune tara)と
(grandma ukifune sazae)
が加わる。

（３）変数への数値の代入
(a)アクション部（RHS とも呼ばれる）では、変数に数値を入れることができる。
命令名は bind です。
例：(bind ?x 5)とか(bind ?y (+ ?a ?b))で、前者は?x に 5 を、後者は?a と?b の和を?y に入れる。

(b)グローバル変数も定義できまる
（グロバル変数名は　?*...*　のように、*ではさむこと）
例：
```
(defglobal
   ?*var* =17
   ?*gv* = "five"
)
```
（私が試したところでは、グローバル変数の値は変更できない。グローバル定数という感じ）

(c)multifield value と呼ばれる複数の値（リスト）を持った変数も利用できる
変数の前に $ が付くと 、マルチフィールドバリューと呼ばれ、その変数は Fact の中にある複数の項目のリストとマッチングが取られる。
例えば、パターン (data $?vals) は (data 10 17 83) という Fact とマッチし、(10 17 83) が ?vals に代入される。 (注： $ は変数名の一部ではない)

マルチフィールドバリューは、数や記号や文字列が並んだものである。CLIPS では（テンプレートの場合以外）入れ子のリストを許さないため、マルチフィールドバリューとして、リストをつないでいく方式を使う。

例えば、input で始まり father を含む Fact とマッチングを取る時、
(input $?before father $?after)
とすると、before には father の前の部分、after には father より後ろの部分が入る。そし

て、もし、ルールが、
(input $?before father $?after)＝＞(assert (before-father ?before))
だったとして、before が(when I think of my)　だったとすると、
"(before-father when I think of my)"
という新たな Fact が作られる。（注："(before-father (when I think of my))"ではない）
マルチフィールドバリューで使われる他の関数は以下の通りである。
(create$ X Y Z...) -- マルチフィールドバリューを作る。例: (create$ a b c) は (a b c)を作る。
(nth$ N L) --　L というマルチフィールドバリューの N 番目の項目を返してくる。（注：番号は 0 から始まる）
(first$ L) -- (nth$ 1 L)と同じ。
(rest$ L) -- マルチフィールドバリュー L の２項目以降を返してくる。
(member$ X L) -- マルチフィールドバリュー L に X が含まれているかを調べ、含まれていればその位置を返してくる。
(subsetp L1 L2) -- マルチフィールドバリュー L１ が マルチフィールドバリュー L2 のサブセットかを調べる。
(explode$ str) -- 文字列 str からマルチフィールドバリューを作る。例：(explode$ "I can't do that") は (I can't do that)を作る。
(implode$ L) -- マルチフィールドバリュー L L を文字列にする（explode の逆)。

explode$は readline(文字列入力)と組み合わせて、入力値を Fact にする時に使える。
例えば：

```
(defrule converse
    ?t <- (talk)     ; Fires when there is a (talk) fact.
=>
    (retract ?t) ; Get rid of (talk) fact.
    (bind ?input-string (readline)) ; Read a line from user.
    (bind ?input (explode$ ?input-string)) ; Convert it.
    (assert (input ?input)) ; Put input into working memory.
)
```

同様に、implode は出力に使える。

（４）入力命令
(a)(read)は、キーボードからの、数値、記号、文字列を読み込む。
例：(bind ?x (read))

(b)(readline)は、キーボードから文字列を入力する。

（５）Fact 内の要素をアクション部で変更したい場合
テンプレートタイプの Fact を使用する。
ここでは、変数 x,y,z を変える

```
(deftemplate hensu
        (slot x)
        (slot y)
        (slot z)
)

(defrule dainyux
        ?i1 <- (hensu (x 0))
=>
        (modify ?i1 (x 1))
)

(defrule dainyuy
        ?i2 <- (hensu (y 0))
=>
        (modify ?i2 (y 2))
)

(defrule dainyuz
        ?i3 <- (hensu (z 0))
        (hensu (x ?x))
        (hensu (y ?y))
=>
        (modify ?i3 (z (+ ?x ?y)))
)
```

===
CLIPS>(assert (hensu (x 0) (y 0) (z 0)))
として(run)
===

（Ｖ）実行ルールの選択

プロダクションシステムでは条件部が真のルールが複数あった場合、何らかのアルゴリズムで１つを選んで実行する。

各ルールには salience という実行優先順位値を割り当てることができる。salience の高いルールが agenda の上位に置かれる。

agenda に実行可能（条件部が真の）ルールを入れる時、一番上に入れる depth-first、一番下に入れるのを breadth-first と呼び、

(set-strategy breadth)

で breadth-first にできる。

一つのルールが、agenda に何回も出てくることもあるが、一度マッチした条件で何度も実行されてはいけない。そのために、Fact には番号を付け、番号順にぐるぐるマッチングをとるようにしている。

（ＶＩ）サンプルプログラム

(1)参考資料(1)の例

```
(defrule rule-room
        (room)
=>
        (assert  (chair))
        (printout  t  "There is a room." crlf)
)


(defrule rule-chair
        (room)
        (chair)
=>
        (printout  t  "There is a chair." crlf)
)
```

====

CLIPS> (assert (room))

をしておき、(run)

====

(2)体温が 37 度以上なら WM に(high)と書き、WM に(high)と有ったら「医者へいけ」と出力

```
(defrule hl
      (tmp ?t)
      (test (> ?t 37))
=>
      (assert    (high))
      (printout    t    "fire next rule" crlf)
)

(defrule warning
      (high)
=>
      (printout    t    "Goto Hospital" crlf)
)
```

＝＝＝Fact には＝＝＝

(tmp 38)

などと書く

＝＝＝＝＝＝＝＝＝＝

(3)サザエさん（変数なし）

```
(defrule rule-room
            (mother hune sazae)
            (mother sazae tara)
=>
            (assert    (grandma fune tara))
)
```

===

次の Fact を作っておく

```
(mother ukifune fune)
(mother fune sazae)
(mother sazae tara)
(father masuo tara)
```

===

(4)サザエさん（変数使用）

```
(defrule rule-room
        (mother ?g ?m)
        (mother ?m ?c)
=>
        (assert  (grandma ?g ?c))
)
```

(5)以前見た映画しかやっていなかったら TDL にゆく

```
(defrule onaji
    (ima ?i)
    (mae ?m)
    (test (eq ?i ?m))
=>
    (printout  t  "Goto TDL" crlf)
)

(defrule tigau
    (ima ?i)
    (mae ?m)
    (test (neq ?i ?m))
=>
    (assert  (mae ?i))
    (printout  t  "Goto theater" crlf)
    (halt)
)
```

＝＝＝Fact には＝＝＝

```
(ima ponyo)
(mae sen)
```

とか入れておく

＝＝＝＝＝＝＝＝＝＝

(6)アクション部でスロットの値を変える方法

```
(deftemplate hensu
        (slot x)
        (slot y)
```

```
        (slot z)
)

(defrule dainyux
        ?i1 <- (hensu (x 0))
=>
        (modify ?i1 (x 1))
)

(defrule dainyuy
        ?i2 <- (hensu (y 0))
=>
        (modify ?i2 (y 2))
)

(defrule dainyuz
        ?i3 <- (hensu (z 0))
        (hensu (x ?x))
        (hensu (y ?y))
=>
        (modify ?i3 (z (+ ?x ?y)))
)
```

===
(assert (hensu (x 0) (y 0) (z 0)))
で Fact を入れておく
===

以下、いくつかのテストに使ったサンプルを付けたしておく。
(7)色々のテストに使った

```
(defrule warn
    (high)
=>
    (printout    t    "Goto H" crlf)
    (halt)
)
```

```
(defrule hl
      (tmp ?t)
      ?f <- (tmp ?t)
      (test (> ?t 37))

=>
      (retract ?f)
      (assert    (high))
      (bind ?i (+ ?t 1))
      (assert    (tmp ?i))
      (printout    t    "r0" crlf)
)
===
(assert (tmp 38))
===

(8)グローバル変数をアクション部で変えられないテスト
(defglobal
    ?*v* = 10
)

(defrule warn
      (high)
=>
      (printout    t    "Goto H" crlf)
      (halt)
)

(defrule hl
      (tmp ?t)
      (test (> ?t 37))

=>
      ?*v* = 20
```

```
    (printout    t    "r0="    ?*v*    crlf)
)
```

(9)テンプレートの中の値を変更する（参考資料(4)より）

```
(deftemplate taion
        (slot name)
        (slot tmp)
)

(defrule hl
        ?f1 <- (hl ?name)
        ?f2 <- (taion (name ?name) (tmp ?tmp))

=>

        (modify ?f2 (tmp (+ ?tmp 1)))
        (retract f1)
        (printout    t    "hennkousita" crlf)
)
```

＝＝＝次のように実行＝＝＝

```
"(assert (taion (name n) (tmp 36)))"
"(assert hl n)"
"(run)"
```

＝＝＝＝＝＝＝＝＝＝＝＝＝＝

(10)(9)をもっと簡単にしたもの

```
(deftemplate taion
        (slot tmp)
)

(defrule hl
        ?f1 <-    (hl n)
        ?f2 <- (taion (tmp ?tmp))
```

```
=>

        (modify ?f2 (tmp (+ ?tmp 1)))
        (retract f1)
        (printout    t    "r0=" crlf)
)
```

＝＝＝この様に実行＝＝＝

```
"(assert (taion (tmp 36)))"
"(assert (hl n))"
"(run)"
```

＝＝＝＝＝＝＝＝＝＝＝＝

（ＶＩＩ）本格的サンプル（ネットから拾った）

(1)ＰＣ購入エキスパートシステム

PC 購入時に、PC の利用形態や予算などを聞いてきて、最適な PC を推薦してくれる。

エキスパートシステムは、samplefromNet1s.clp

使い方は、clp ファイルをロードし、

CLIPS>(reset)

CLIPS>(run)

システムのもう少し詳しい説明は、samplefromNet1d.mht 参照

(2)庭園設計エキスパートシステム

庭の設計には先ず、一番目につく中背の植物と常緑の低木を決め（backbone と呼ぶ)、次に色や模様を考え、１年生、多年生、球根、地衣類、などを考慮して植物を選ぶ。このエキスパートシステムは、現在植えてある植物名を入力すると、バックボーン、１年生、多年生、ツタ類、地衣類、などの数を考え、追加すべき植物を示してくれる。

CLIPS ファイルは、samplefromNet2s.clp

使い方は、上と同様、

CLIPS>(reset)

CLIPS>(run)

（ここで、今、植えてある植物を聞いてくるので、

rose holly magnolia hostas coneflower lamium cosmos

などと答える）

システムの詳しい説明は、samplefromNet2d.mht。

⑶Queue へ数値を入れたり出したりする例
CLIPS ファイルと説明は、samplefromNet3s.clp
使い方は、上と同様、
CLIPS>(reset)
CLIPS>(run)

（ＶＩＩＩ）こんなものできるかな？
⑴１日１回天気（雨または晴）を入力し、２日晴れ続いたら３日めに「植物に水をやって！」と言ってくる。要するに３日以上水をやらない日をつくらない。

⑵株価を毎日入力し、２日連続で上がったら売れ、２日連続で下がったら買え、と言ってくる。

⑶子供のメタボは
ウエスト 80cm 以上
血圧 125mmHg 以上
血中脂質 120mg/dl 以上
血糖値 100mg/dl 以上
の内、２以上当てはまるとメタボと診断されます。
ウエスト、血圧、血中脂質、血糖値
を入力し、メタボか否かを出力する。

⑷薬の選択
⑷a薬品Ａは、成分ｓ（咳止め）と ａ を含む
⑷b薬品Ｂは、成分ｇ（解熱）と ｂ を含む
⑷c薬品Ｃは、成分ｃ（鎮痛）と ｄ を含む
⑷d薬品Ｄは、成分ｓ（鎮痛）と ａ を含む
⑷x成分ｂ と ｄ は同時に摂ってはいけない。
質問：頭痛と発熱がある場合、飲むべき薬はどれか？