ICOT Technical Report: TR-471

TR-471

EBG実験システム(対話型EBGシステム)

打橋　知孝(NTT)、
滝　寛和、堀内　英一

April, 1989

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

ＥＢＧ実験システム（対話型ＥＢＧシステム）

--説明木を利用した領域知識の獲得支援システム--

はじめに

実際の知識ベース構築において，専門家からの知識獲得による構築後，知識ベース運用時には事例を用いて知識を洗練していく必要がある。この知識洗練の一手法として説明に基づく学習（ＥＢＧ）が行われている。しかしながら，ＥＢＧの処理を行うための前提条件がきつく実用上問題点が存在する。

今回報告するＥＢＧ実験システムは，ＥＢＧにおいての説明失敗を領域知識の不足または不適当な例の導入と考え，説明失敗時にユーザに刺激となる情報を与え，対話的に知識を獲得・洗練する１つの試みである。

１．背景

1.1.ＥＢＧの基本処理

エキスパートシステム開発において，知識ベースの構築はそのシステムごとに行う必要がある。ところが，この知識ベース構築のための知識獲得は非常に困難な作業であり，知識獲得ボトルネックと呼ばれている。

この知識獲得問題に対する１つのアプローチとして，説明に基づく学習（ＥＢＧ [Mitchell, keller, & Keder-cabelli 86]）が提案されている。

ＥＢＧでは，正しい学習が行えるように，既に存在する豊富な背景知識（領域知識）を活用して事象（訓練例）を説明し，それに基づいて的確な一般化を行った結果を獲得する。つまり説明とは，訓練例が領域知識のもとで成立することを示す一種の証明である。

ＥＢＧにおいては，以下に示す４つの情報を学習システムに与え，２つのプロセスから成る方法で処理が行われる。

与えられる情報

領域知識(domain theory):
　学習する領域（分野）についてのルールや事実

目標概念(goal concept):
　学習される概念の記述

訓練例(training example):
　目標概念の具体例

操作性規範(operationality criterion):
　学習結果の満たすべき規準。学習結果が利用性，有効性の高い記述であるように定めておく。

処理手順

説明プロセス：学習システムに目標概念及びその具体例である訓練例を与えて領域知識に基づく目標概念の説明構造を生成する（説明する）。

一般化プロセス：説明構造中の定数を変数化し，変数の拘束の情報を明らかにする。また与えられる操作性規範を満たすように項目を選択し，説明構造を利用性，有効性の高い記述に変換する。

1.2.従来の問題点

ＥＢＧは，例からの学習に比して，多数の例を利用せずに学習でき，学習に要する計算量が少ないなどの利点がある。そのため実用的な問題にも応用しやすいと考えられている。

しかし，学習システムに与えられる情報などに関して次のような問題点が存在している。

(a)領域知識に関する問題点
説明構造を得るためには領域知識が完全であることが前提になっている。不十分な場合には説明が行えず，学習が行えない事態に陥る。

(b)操作性規範に関する問題点
一般化プロセスは説明構造を利用性，有効性の高い記述に変換するプロセスとも考えられる。この利用性，有効性の評価基準が操作性規範である。操作性規範は対象とする問題領域に非常に強く依存するものであり，学習システムの状況と目的に応じて時間的に変化しうるものであると考えられている。そのため事前に定義するのは困難であり，これを精密に定義することが研究課題となっている。

(c)訓練例に関する問題点
説明に基づく学習では１つの例から１つの概念が学習できる。同目標概念の具体例が多数ある場合に目標概念を説明するのに適した訓練例の選択法が問題である。

(d)説明構造に関する問題点
説明構造が複数存在する場合の解の選び方。

説明に基づく学習の適用に際して，上述の問題点があり，使用条件がきびしいことから，実用上，適用範囲が狭いことが難点となっている。

2．ＥＢＧ実験システム

今回構築したＥＢＧ実験システムは，領域知識，目標概念，訓練例を入力とし，ＥＢＧの処理（説明）に失敗した時にユーザからの不足知識追加を支援して説明成功に導き，最終的に説明木及び獲得できる概念を出力するシステムである。

2.1.背景

1.2.に示したようにＥＢＧの実用に関して種々の問題点が存在する。
ＥＢＧの処理に関して、説明の基本となる領域知識が完全であることがその前提となっているが，実際問題として考えると，領域知識が完全である可能性は低い。
そこで，前記の(a)領域知識に関する問題点 を対象としたシステムを構築し，領域知識が不完全な場合にも対処できるシステムを構築することにした。

2.2.アプローチ

2.2.1.システムの処理方針

従来のＥＢＧシステムでは説明できた場合のみ説明構造や獲得できた概念を出力することができた。しかし，説明できない場合には，説明できなかったという事実しか分からなかった。
そこで，説明失敗時の対応として，ユーザに失敗時の状況を提示し、ユーザから説明を成功させるための情報を得て、不完全な知識を極力完全にしていき、説明成功へと導くというシステムを構築することにした。

2.2.2.システムの持つ機能

説明失敗の原因は次に示すものであると考えられる。

原因１．領域知識（ルール，因果関係）が不足しているために訓練例として与えられた事実（説明木の leaf となる fact）に至る途中の道が切れている。

原因２．説明に適切な事実を用意しなかった（必要な事実を説明に有効でない

ものと見なして選ばなかった）。

この２点より，説明を成功に導くためには，
1)説明に失敗した時にユーザにその旨を示す機能、
2)不足情報（原因１における途中をつなぐためのルールまたは原因２における事実）の入力を支援する機能

が必要であると考えた。しかし，ユーザから情報を獲得するには知識連想の刺激が必要である。そのために失敗時の状況をシステムからの情報として提示することにした。

説明失敗と言っても，全く説明構造を構築しなかったのではなく，情報が不足するまでは構築していたのである。そこで，情報が不足するまで構築を行っていた説明構造（説明木の途中結果）と構造の途絶えた箇所（ここで示される項目が存在すれば説明木の構成は完成する。木完成のために必要な情報と言える）を失敗時の状況として提示し，ユーザへの刺激とすることにした。

さらに，訓練例として入力しておきながら説明試行時に利用しなかった事実にも着目した。この事実が leaf となるようなサブ説明木を考えるとそのサブ説明木の root と前記の機能で示される項目とを結び付けるルールが存在すれば，その説明は成功することになる。
そこで，原因１．の不足するルール連想の刺激として，前記の不足情報の提示機能に加えて説明試行時に使用しなかった事実を leaf とするサブ説明木の root を提示する機能を設定した。

以上示した機能によりユーザとのインタラクティブな処理を進め，領域知識の獲得を行う。ここで操作性規範について考えると，この規準はいわば効率であり，ユーザごとに異なると考え，情報の適宜入力機能に加え，ユーザが適宜操作性規範を決定でき，その結果を表示できるようにした。

また，説明成功の場合には，従来システムと同様に，処理の結果得ることのできる概念の提示の他に，説明木を見て説明を確認できること，一般化するにあたり変数の拘束情報が確認できることができるようにした。

以上示した機能を再度列挙する。
説明成功時に，
1)説明木及び獲得できた概念の表示
説明失敗時に
2)説明木の途中結果及びその木を完成するのに必要な項目の表示
3)説明に使用しなかった事実からの bottom_up parsing 結果表示
4)操作性規範の変更に伴う獲得結果の変化表示

2.3. ＥＢＧ実験システムの処理内容

実験システムには 2.2.2 に示した機能を持たせ，情報追加後再度説明を試行するようにした。具体的な処理内容をつぎに示す。

A)領域知識，訓練例，目標概念を学習システムに入力した後（図１），説明を試みる。
B)説明試行結果を表示する。
　説明成功：説明木，変数の拘束関係（一般化された説明木）及び獲得された概念を表示する。
　説明失敗：構成途中の説明木及びその木を完成させるのに不足している情報を表示する。

この説明失敗の際に表示される項目は，不足している項目名，その引数の数，引数の具体的な値など不足している情報の内容を詳しく示している（図３）。
また，成功時に説明木と同時に示される木（図６-(c)）は項目の因果関係及び変数の拘束関係を示している。

c)その表示に基づき説明木完成に向けて説明の再施行を行う。再試行の内容は次の通り，
1:不足事実の入力による再処理
2:不足ルールの入力による再処理
3:操作性規範の変更による再処理
4:説明木の別候補の表示
5:説明試行時未使用事実についての表示

システムの表示は，領域知識を完全にするための具体的な示唆をユーザに与えており，ユーザは表示された項目に関する入力，または操作性規範の変更（これにより不足する項目の手前までを有効とし，不足する項目を有効範囲外とする）により有効な概念を得ることができる。

以上，A,B,C の繰り返しにより，領域知識として不足しているルールの補充を行うことができる。
また，操作性規範の動的な設定によりユーザの希望の利用性，有効性を持つ記述への変換（一般化）を行うことができる。

3．処理例

ここでは john の自殺（suicide(john)）の例を使って，ＥＢＧ実験システムの処理を示す。入力となる領域知識，目標概念，訓練例を図２に示す。
この入力を基に説明試行を行ったが失敗した場合を説明する。　表示から hate(john,john)及び　gun(colt)　に関する情報が不足していることが分かる（図３）。そこで，まず gun(colt) という事実が確かに観測されるので追加して再試行してみると，gun(colt) が不足しているという表示は消える。しかし，hate(john,john) の不足は表示されたままである（図４）。
hate(john,john) という事実が確認できないので，未使用の事実についての表示をさせてみると　depressed(X):-serious_ill(X). というルールの存在が明らかになった（図５）。
この情報から hate(X,X) と　depressed(Y) が関係づけられれば説明木が完成することが分かる。そこで hate(X,X):-depressed(X)　というルールを追加して再試行すると，説明成功に至り，システムは説明木，変数の拘束関係，獲得できた概念を出力する（図６）。
なお，操作性規範の変更による獲得する概念の変化の表示を図７に示す。

4．まとめ

今回，ＥＢＧによる知識獲得・洗練において，ＥＢＧの適用範囲拡大を目的として，領域知識・目標概念・訓練例を入力として，説明失敗時には各種情報を出力してユーザの不足情報追加を支援して不完全な知識を極力完全にするように働き，説明成功時には説明の詳細及び獲得できる概念を出力するシステム（ＥＢＧ実験システム）の構築を行った。また，このシステムでは各種情報の変更に伴う説明構造・獲得概念の変化を出力できる。

従来のＥＢＧの処理例は今回の john の自殺のように小規模の例が多かった。そこで，ＥＢＧ実験システムを使って比較的大規模な例を作成してみた（付録）。このように様々な例を扱うことによって，ＥＢＧの処理向きの例の検討といった課題にも着手できるようになった。

今後の課題としては，再処理にあたり入力される情報と既に入力されている

情報との矛盾性の管理などの問題が存在するのでその解決が研究課題（仮説推論との融合による解決が考えられる）となる。

```
---------------------- fig. of result ----------------------

領域知識 ファイル名を入力して下さい。
|: suidt1.

目標概念 を入力して下さい。
|: suicide(X):-kill(X,X).

目標概念の具体例 を入力して下さい。
|: suicide(john).

訓練例 ファイル名を入力して下さい。
|: suite1.
投入された訓練例は,
  suicide(john) :- [met(john,tom),carry(tom,colt),serious_ill(john)]

operationality は？（指定しないのなら "nil." を入力）
|: nil.
```

図１．システムへの入力

```
kill(A,B):-hate(A,B),weapon(C),possess(A,C).

depressed(W):-heavy_fall(W).
depressed(W):-serious_ill(W).
possess(U,V):-buy(U,V).
possess(U,V):-get(U,V,P).
get(U,V,P):-met(U,P),carry(P,V).
weapon(Z):-gun(Z).
weapon(Z):-knife(Z).
weapon(Z):-car(Z).
weapon(Z):-ship(Z).
```

(a)領域知識

```
suicide(X):-kill(X,X).
```

(b)目標概念

```
suicide(john):-met(john,tom),carry(tom,colt),serious_ill(john).
```

(c)訓練例

図２．入力情報

説明に失敗しました。

```
suicide(john)
          kill(john,john)
             hate(john,john) <-- 不足
                weapon(colt)
                   gun(colt) <-- 不足
                possess(john,colt)
                      get(john,colt,tom)
                         met(john,tom)
                         carry(tom,colt)


            gun(_1198),
            hate(john,john).に関する情報が必要です。


以下の操作後，説明の再試行を行います。（input number）
      1: fact の入力，              2: ルールの入力，
      3: operationality の変更，  4: 他 説明木の表示
      5: 未使用 fact について，   6: 最初の情報に戻す
      7: 中止
|:
```

図 3．説明失敗時の表示

```
fact を入力して下さい。  （入力終了 --> input "nil."）
|: gun(colt).
fact を入力して下さい。  （入力終了 --> input "nil."）
|: nil.


説明に失敗しました。

suicide(john)
          kill(john,john)
             hate(john,john) <-- 不足
                weapon(colt)
                   gun(colt)
                possess(john,colt)
                      get(john,colt,tom)
                         met(john,tom)
                         carry(tom,colt)


            hate(john,john).に関する情報が必要です。


以下の操作後，説明の再試行を行います。（input number）
      1: fact の入力，              2: ルールの入力，
      3: operationality の変更，  4: 他 説明木の表示
      5: 未使用 fact について，   6: 最初の情報に戻す
      7: 中止
```

図 4．不足事実の追加入力及び再試行結果

```
訓練例で与えられた fact の内、次のものは未使用です。
[serious_ill(john)]

未使用 fact が何から導かれるのか表示します。
上記の 未使用 fact の中から１つを選んで下さい。
|: serious_ill(P).

       serious_ill(_2555) ->
          depressed(_2555)
```

図５．未使用 fact についての表示

```
ルール を入力して下さい。 (入力終了 --> input "nil.")
|: hate(X,X):-depressed(X).
ルール を入力して下さい。 (入力終了 --> input "nil.")
|: nil.
```

(a)不足ルールの追加入力

```
**  説明木  **
suicide(john)
          kill(john,john)
                hate(john,john)
                      depressed(john)
                         serious_ill(john)
                weapon(colt)
                   gun(colt)
                possess(john,colt)
                      get(john,colt,tom)
                         met(john,tom)
                         carry(tom,colt)
```

(b)説明木

```
**  変数の拘束関係  **
suicide(_2601)
          kill(_2601,_2601)
                hate(_2601,_2601)
                      depressed(_2601)
                         serious_ill(_2601)
                weapon(_2671)
                   gun(_2671)
                possess(_2601,_2671)
                      get(_2601,_2671,_2972)
                         met(_2601,_2972)
                         carry(_2972,_2671)
```

(c)一般化した説明木

```
**  獲得した概念  **
```

```
suicide(_2601) :-
            serious_ill(_2601),
            gun(_3548),
            met(_2601,_3769),
            carry(_3769,_3548).
```

(d)獲得した概念

今までに次のルールが追加されています。
hate(_3919,_3919):-depressed(_3919).

(e)追加ルールの表示

図6．説明成功時の表示（ルールの追加による成功）

```
Operationality を入力して下さい。（入力終了 --> input "nil."）
|: depressed(X).
Operationality を入力して下さい。（入力終了 --> input "nil."）
|: nil.

**  説明木  **
suicide(john)
          kill(john,john)
               hate(john,john)
                  depressed(john)
               weapon(colt)
                  gun(colt)
               possess(john,colt)
                     get(john,colt,tom)
                        met(john,tom)
                        carry(tom,colt)

**  獲得した概念  **

suicide(_3937) :-
            depressed(_3937),
            gun(_4778),
            met(_3937,_4951),
            carry(_4951,_4778).
```

図7．操作性規範の変更による変化

```
-------------------- 付録 ----------------------------------------
/*******************************************************
        Domain_Theory(example) for EBG system

Please input

        build_bridge(name,river,style,material)

for Goal_Concept,and
```

```
        budget(name,Pay)                [Pay=    rich / middle /poor]
        river_width(river,width)

        setting_point(name,Place)       [Place=  on_path / on_street /
                                          town_to_town / town_to_plant]
        wind(river,Wind)                [Wind=   windy / calm]
        water_volume(river,Water)       [Water=  lot / little]
        ground_kind(Ground)             [Ground= rock / clay / sand]

for Training_Example.
*****************************************************/

bridge(Name,River,Style,Material):-
                        bridge_style(Name,River,Cost,Style),
                        cost(Name,Cost),
                        materials(Name,River,Cost,Style,Material).

/***cost(Name,Cost):-budget(Name,Pay)***/
cost(Name,high):-  budget(Name,rich).
cost(Name,middle):-budget(Name,middle).
cost(Name,low):-   budget(Name,poor).


/***environment(River,Judge1):-river_width(River,Width),
                              foundation(River,Quality),
                              wind(River,Wind)***/

environment(River,non_girder):-river_width(River,_),
                               foundation(River,bad),
                               wind(River,_).

environment(River,non_suspend):-river_width(River,_),
                                foundation(River,_),
                                wind(River,windy).

environment(River,not_care):-river_width(River,_),
                             foundation(River,_),
                             wind(River,_).

foundation(River,Quality):-river_bed(River,Quality).

/**river_bed(River,Qualify):-water_volume(River,Water),
        ground_kind(River,Ground).**/

river_bed(River,good):-water_volume(River,lot),ground_kind(River,rock).
river_bed(River,passable):-water_volume(River,lot),ground_kind(River,clay).
river_bed(River,bad):-water_volume(River,lot),ground_kind(River,sand).
river_bed(River,excellent):-water_volume(River,little),ground_kind(River,rock).
river_bed(River,good):-water_volume(River,little),ground_kind(River,clay).
river_bed(River,passable):-water_volume(River,little),ground_kind(River,sand).

/***bridge_length(Name,River,Length):-river_width(River,Width).***/
bridge_length(_,River,long):-river_width(River,wide).
bridge_length(_,River,middle):-river_width(River,middle).
```

```
bridge_length(_,River,short):-river_width(River,narrow).

/***endurance(Name,River,Judge):-bridge_weight(Name,River,Weight1),
                                 over_weight(Name,Weight2),
                                 river_bed(River,Quality).***/

endurance(Name,River,necessary):-bridge_weight(Name,River,heavy),
                                 over_weight(Name,_),
                                 river_bed(River,_).

endurance(Name,River,necessary):-bridge_weight(Name,River,_),
                                 over_weight(Name,heavy),
                                 river_bed(River,_).

endurance(Name,River,necessary):-bridge_weight(Name,River,_),
                                 over_weight(Name,_),
                                 river_bed(River,passable).

endurance(Name,River,necessary):-bridge_weight(Name,River,_),
                                 over_weight(Name,_),
                                 river_bed(River,bad).

endurance(Name,River,necessary):-bridge_weight(Name,River,middle),
                                 over_weight(Name,middle),
                                 river_bed(River,good).


endurance(Name,River,moderate):- bridge_weight(Name,River,middle),
                                 over_weight(Name,middle),
                                 river_bed(River,excellent).

endurance(Name,River,moderate):- bridge_weight(Name,River,middle),
                                 over_weight(Name,light),
                                 river_bed(River,_).

endurance(Name,River,moderate):- bridge_weight(Name,River,light),
                                 over_weight(Name,middle),
                                 river_bed(River,good).


endurance(Name,River,unnecessary):-bridge_weight(Name,River,light),
                                   over_weight(Name,light),
                                   river_bed(River,_).

endurance(Name,River,unnecessary):-bridge_weight(Name,River,light),
                                   over_weight(Name,middle),
                                   river_bed(River,excellent).


/***bridge_weight(Name,River,Weight1):-bridge_length(Name,River,Length),
                                       bridge_width(Name,Width).***/
bridge_weight(Name,River,heavy):- bridge_length(Name,River,long),
                                  bridge_width(Name,wide).
bridge_weight(Name,River,heavy):- bridge_length(Name,River,middle),
                                  bridge_width(Name,wide).
```

```
bridge_weight(Name,River,middle):-bridge_length(Name,River,short),
                                  bridge_width(Name,wide).
bridge_weight(Name,River,heavy):- bridge_length(Name,River,long),
                                  bridge_width(Name,middle).
bridge_weight(Name,River,middle):-bridge_length(Name,River,middle),
                                  bridge_width(Name,middle).
bridge_weight(Name,River,light):- bridge_length(Name,River,short),
                                  bridge_width(Name,middle).

/*over_weight(Weight1,Weight2):-bridge_width(Width),bridge_crosser(Vehicle).*/
over_weight(Name,heavy):-bridge_width(Name,_),bridge_crosser(Name,lorry).
over_weight(Name,heavy):-bridge_width(Name,wide),
        bridge_crosser(Name,automobile).
over_weight(Name,middle):-bridge_width(Name,middle),
        bridge_crosser(Name,automobile).
over_weight(Name,middle):-bridge_width(Name,wide),bridge_crosser(Name,human).
over_weight(Name,light):-bridge_width(Name,middle),bridge_crosser(Name,human).

bridge_width(Name,Width):-lane(Name,Width).

lane(Name,wide):-trafic(Name,frequent).
lane(Name,middle):-trafic(Name,rare).

bridge_crosser(Name,Vehicle):-setting_point(Name,Place),
                              cross_point(Name,Place,Vehicle,_).
trafic(Name,Trafic):-setting_point(Name,Place),
                     cross_point(Name,Place,_,Trafic).

/*setting_point(Name,Place):-cross_point(Name,Place,Vehicle,Trafic).*/

cross_point(Name,on_path,human,rare).
cross_point(Name,on_street,automobile,rare).
cross_point(Name,town_to_town,automobile,frequent).
cross_point(Name,town_to_plant,lorry,frequent).

/***bridge_style(Name,River,Cost,Style):-environment(River,Judge1),
                                        bridge_length(Name,River,Length),
                                        endurance(Name,River,Judge2).***/

bridge_style(Name,River,low,plain):-  environment(River,girder_less),
                                      bridge_length(Name,River,short),
                                      endurance(Name,River,unnecessary).
bridge_style(Name,River,low,plain):-  environment(River,girder_less),
                                      bridge_length(Name,River,middle),
                                      endurance(Name,River,moderate).
bridge_style(Name,River,low,plain):-  environment(River,non_suspend),
                                      bridge_length(Name,River,short),
                                      endurance(Name,River,unnecessary).
bridge_style(Name,River,low,plain):-  environment(River,non_suspend),
                                      bridge_length(Name,River,middle),
                                      endurance(Name,River,moderate).
bridge_style(Name,River,low,plain):-  environment(River,not_care),
                                      bridge_length(Name,River,short),
                                      endurance(Name,River,unnecessary).
bridge_style(Name,River,low,plain):-  environment(River,not_care),
```

```
                                        bridge_length(Name,River,middle),
                                        endurance(Name,River,moderate).

bridge_style(Name,River,middle,girder):-environment(River,non_suspend),
                                        bridge_length(Name,River,short),
                                        endurance(Name,River,unnecessary).
bridge_style(Name,River,middle,girder):-environment(River,non_suspend),
                                        bridge_length(Name,River,middle),
                                        endurance(Name,River,moderate).
bridge_style(Name,River,middle,girder):-environment(River,non_suspend),
                                        bridge_length(Name,River,middle),
                                        endurance(Name,River,necessary).
bridge_style(Name,River,high,girder):-  environment(River,non_suspend),
                                        bridge_length(Name,River,long),
                                        endurance(Name,River,necessary).
bridge_style(Name,River,middle,girder):-environment(River,not_care),
                                        bridge_length(Name,River,short),
                                        endurance(Name,River,unnecessary).
bridge_style(Name,River,middle,girder):-environment(River,not_care),
                                        bridge_length(Name,River,middle),
                                        endurance(Name,River,moderate).
bridge_style(Name,River,middle,girder):-environment(River,not_care),
                                        bridge_length(Name,River,middle),
                                        endurance(Name,River,necessary).
bridge_style(Name,River,high,girder):-  environment(River,not_care),
                                        bridge_length(Name,River,long),
                                        endurance(Name,River,necessary).

bridge_style(Name,River,high,suspension):-
                                        environment(River,girder_less),
                                        bridge_length(Name,River,middle),
                                        endurance(Name,River,necessary).
bridge_style(Name,River,high,suspension):-
                                        environment(River,girder_less),
                                        bridge_length(Name,River,long),
                                        endurance(Name,River,necessary).
bridge_style(Name,River,high,suspension):-
                                        environment(River,not_care),
                                        bridge_length(Name,River,middle),
                                        endurance(Name,River,necessary).

/*materials(Name,River,Cost,Style,Material):- water_volume(River,Water),
                                            bridge_length(Name,River,Length).*/
materials(Name,River,high,girder,iron_frame):-water_volume(River,lot),
                                              bridge_length(Name,River,long).
materials(Name,River,high,girder,iron_frame):-water_volume(River,lot),
                                              bridge_length(Name,River,middle).
materials(Name,River,middle,girder,iron_frame):-water_volume(River,lot),
                                               bridge_length(Name,River,short).
materials(Name,River,high,girder,iron_frame):-water_volume(River,little),
                                              bridge_length(Name,River,long).
materials(Name,River,middle,girder,stone):-   water_volume(River,little),
                                              bridge_length(Name,River,long).
materials(Name,River,middle,girder,stone):-   water_volume(River,little),
                                              bridge_length(Name,River,middle).
```

```
materials(Name,River,low,girder,wood):-      water_volume(River,little),
                                               bridge_length(Name,River,middle).
materials(Name,River,middle,girder,stone):-   water_volume(River,little),
                                               bridge_length(Name,River,short).
materials(Name,River,low,girder,wood):-      water_volume(River,little),
                                               bridge_length(Name,River,short).

materials(Name,River,high,suspension,iron_frame):-
                        bridge_length(Name,River,long).
materials(Name,River,_,suspension,iron_frame):-
                        bridge_length(Name,River,middle).

materials(Name,River,high,plain,iron_frame):-bridge_length(Name,River,middle).
materials(Name,River,middle,plain,stone):-bridge_length(Name,River,_).
materials(Name,River,middle,plain,wood):-bridge_length(Name,River,middle).
materials(Name,River,low,plain,wood):-bridge_length(Name,River,short).


/*************************************************************
        Information of rivers
*************************************************************/
/*
river_width(Name,Width)
water_volume(Name,Water)
ground_kind(Name,Ground)
wind(Name,Wind)*/

/*river(r1,middle,lot,rock,windy).*/
river_width(r1,middle).
water_volume(r1,lot).
ground_kind(r1,rock).
wind(r1,windy).

/*river(r2,wide,lot,clay,calm).*/
river_width(r2,wide).
water_volume(r2,lot).
ground_kind(r2,clay).
wind(r2,calm).

end.
```