An Introduction to rule language

- Based on text rather than automata, easy to edit, interpret and run.

- Divided into segmentations, separating the relationship between cards and the process of the game.

- C-style function to describe the process of the game. Two-layer (user layer and system layer) architecture, system interfaces to decribe directive operations to cards are predefined, similar to sys-call in OS.

- 2-letter to describe card. A series of cards are represented as a string.

- Regular expression to match a series of cards.

Advantages: compatible to different games, easy to implement.

Framework

```
<config> players=3;cards=2; </config>        //the number of players and packs of cards
<var>
        int a;
        string b;
</var>                                       //global variables
<func>
        string test(){return testing"}
</func>                                      //global functions

<card> … </card>                             //comparison between single card

<mode> … </mode>                             //relationship between cards

<body> … </body>                             //the procession of the game

<end> … </end>                               //the end of the game and exit
```

About cards

- single card

2-letter, H,D,S,C represent for heart, diamond, spade and club. JB and JS represent for big joker and little joker.

e.g. HA - heart A SJ - spade J C0 - club 10

- series of cards

a sequence of 2-letter substring representing each card. The order of the cards obey the following rules.

- JB and JS are put ahead.
- divide cards into groups, each group has cards with the same number.
- big gourps of cards are put ahead, small groups of cards are put behind.
- groups of the same size are put in the order of the cards' number, A first and K last.

e.g. HAHAHAS3 JBHAHA JSJBH6H6H6SASA

Regular Expression

• some marks mean the same as python regular expressions, including [] . ^

• | means 'or' in the level of 'number or color', || means 'or' in the level of 'single card'

• (S6){2} means two spade 6s. (S6){2+} means at least two spade 6s. (S6){c} means the number of spade 6 is c, c is a variable predefined. (S6){2+:c} means at least two spade 6s, the number of such cards is defined as a variable c.

•A lower-case letter is defined as a variable in the first appearance and should be consistent in the later if it appears without mark $ or #. Mark $ means this variable should be redefined here. Mark # means this variable should increase by 1 here. For example, (.a){2} means two cards with the same number, ((.a$)(.a)){2+} means two pairs, (.a#){5+} means a straight with at least 5 cards.

Comparison between card(s)

- Single card
  - ➢ In the <card></card> section
  - ➢ In the form of '(func_name) (pattern)|(pattern) (priority_list)'
  - ➢ If the two cards to be compared meet the pattern in the second part, find their first matching pattern in the priority list. The pattern behind is smaller than the pattern ahead. If the two card hit the same pattern, continue to the next list.
  - ➢ the global variables quoted are marked '%variable%'
  - ➢ one possible version is shown as follows.

```
list: ..|.. JB JS .2 .A .K .Q .J .0 .9 .8 .7 .6 .5 .4 .3
slist: ..|.. .K .Q .J .0 .9 .8 .7 .6 .5 .4 .3 .2 .A
```

```
list: ..|.. JB JS %colour%%number% .%number% %colour%.
list: a.|a. .1 .K .Q .J .0 .9 .8 .7 .6 .5 .4 .3 .2
```

Comparison between card(s)

- Series of cards
  - ➢ In the <mode></mode> section
  - ➢ In the form of 'name:pattern:compare_list:superclass_play:superclass_card:compare_play:compare_card'.
  - ➢ Compare_list refers to the chosen list in the <card></card> section to compare single cards.
  - ➢ Find the first class whose pattern the cards meet. Cards are smaller than the Cards of its superclass. If one class is not the superclass nor sonclass of another, they are uncomparable.
  - ➢ Cards of the same class are compared according to the last two parts, which are used to compare cards in different parts of the game. $1, $2 refer to the first and second cards to compare. The return code 2 means uncomparable, code 1 means the first is bigger, code -1 means the first is smaller, code 0 means they are equal.

Comparison between card(s)

- one possible version

```
list: ..|.. JB JS .2 .A .K .Q .J .0 .9 .8 .7 .6 .5 .4 .3
slist: ..|.. .K .Q .J .0 .9 .8 .7 .6 .5 .4 .3 .2 .A
```

```
bomb,(.a){4+},list,,,{
    $1.size>$2.size =>1;
    $1.size<$2.size =>-1;
    $1(1)>$2(1)     =>1;
    $1(1)==$2(1)    =>0;
    $1(1)<$2(1)     =>-1
},{}

straight,(.a#){5+},slist,bomb,,{
    $1.size!=$2.size    =>2;
    $1(2)>$2(2)         =>1;
    $1(2)==$2(2)        =>0;
    $1(2)<$2(2)         =>-1
},{}

straight_3,((.a#)(.a){2}){2+},slist,bomb,,{
    $1.size!=$2.size    =>2;
    $1(4)>$2(4)         =>1;
    $1(4)==$2(4)        =>0;
    $1(4)<$2(4)         =>-1
},{}

straight_3_1,((.a#)(.a){2}){2+:c}(.b$){c},slist,bomb,,{
    $1.size!=$2.size    =>2;
    $1(4)>$2(4)         =>1;
    $1(4)==$2(4)        =>0;
    $1(4)<$2(4)         =>-1
},{}
```

Function

- C-style function
- Supported variable type: int, string, bool and the corresponding array type.
- Supported operation: + - * / % && || !
- Key words table
  - bool
  - case
  - default
  - false
  - for
  - int
  - return
  - string
  - true
  - void

```
int next_player(int num){
    switch{
        case (num==3):return 1;
        default:return num+1;
    }
}
```

```
begin();
for (;get_cardsnum(0)>3;){
    deliver(0,present_player,top_card(0),false);
    present_player=next_player(present_player);
}
```

System Interface Function Reference

string get_cards(int num) //get cards of the player whose id number is 'num'
begin() //restart the game
int get_cardsnum(int num) //get the number of cards of the player whose id number is 'num'
string top_card(int num) //get the first card of player whose id number is 'num'
bool find(int num,string base,int length,int times,bool strict) //find whether the player has the straight whose length is 'length' and repeated time is 'times', the variable 'strict' means whether to take the color of the card into consideration
bool match(string card,string regex) // match the card to the regular expression 'regex'
# bool deliver(int sender,int receiver,string cards,bool show) //deliver card from one player to another, variable 'show' means whether to make this process public
# void next_round() // clear the card in the center and go to the next round
int compare(string card1,string card2) // compare between cards
# int play_card(int num,string tocompare) // the player plays a hand, the cards to compare is 'tocompare'
# void enable(int num) // enable one player
# void disable(int num) // disable one player
# int state_point(int num,int[] set,bool candrop) // the player deliver a number, 'set' refers to legal set of numbers he can deliver, boolean 'candrop' decides whether or not he can drop this chance.
# string show_card(int num,int length) // one player show his cards
# show_public(string message) // show a message in the public area
# show_private(int num,string message) // show a private message in the area of a certain player
# add_score(int num,int score) // add scores to certain when the game ends

One Possible Rule File For the Game Doudizhu

```
<config>cards = 2;player_number = 3;</config>
<var>score = 0;host = 0;</var>
<card>
list:..|.. JB JS .2 .1 .K .Q    .J .0 .9 .8 .7 .6 .5 .4 .3
</card>
<mode>
bomb:(.a){4+}::{
            if($1.size>$2.size)
                        return 1;
            else if($1.size<$2.size)
                        return -1;
            else if($1(1)>$2(1))
                        return 1;
            else if($1(1)<$2(1))
                        return -1;
            else
                        return 0;
}
```

One Possible Rule File For the Game Doudizhu

```
straight:(.a#){5+}:bomb:{
        if($1.size!=$2.size)
                return 0;
        if($1(3)>$2(3))
                return 1;
        else if($1(3)<$2(3))
                return -1;
        else
                return 0;
}
straight_3:((.a#)(.a){2}){2+}:bomb:{
        if($1.size!=$2.size)
                return 0;
        if($1(4)>$2(4))
                return 1;
        else if($1(4)<$2(4))
                return -1;
        else
                return 0;
}
```

One Possible Rule File For the Game Doudizhu

```
straight_3_1:((.a#)(.a){2}){2+:a}(.b$){a}:bomb:{
        if($1.size!=$2.size)
                return 0;
        if($1(4)>$2(4))
                return 1;
        else if($1(4)<$2(4))
                return -1;
        else
                return 0;
}
straight_3_2:((.a#)(.a){2}){2+:a}((.b$)(.b)){a}:bomb:{
        if($1.size!=$2.size)
                return 0;
        if($1(4)>$2(4))
                return 1;
        else if($1(4)<$2(4))
                return -1;
        else
                return 0;
}
```

One Possible Rule File For the Game Doudizhu

```
straight_2:((.a#)(.a)){3+}:bomb:{
        if($1.size!=$2.size)
                return 0;
        if($1(5)>$2(5))
                return 1;
        else if($1(5)<$2(5))
                return -1;
        else
                return 0;
}
common_3:(.a){3}:bomb:{
        if($1(1)>S2(1))
                return 1;
        else if($1(1)<$2(1))
                return -1;
        else
                return 0;
}
```

One Possible Rule File For the Game Doudizhu

```
common_3_1:(.a){3}(.b):bomb:{
        if($1(1)>S2(1))
                return 1;
        else if($1(1)<$2(1))
                return -1;
        else
                return 0;
}
common_3_2:(.a){3}(.b){2}:bomb:{
        if($1(1)>S2(1))
                return 1;
        else if($1(1)<$2(1))
                return -1;
        else
                return 0;
}
```

One Possible Rule File For the Game Doudizhu

```
pair:(.a){2}:bomb:{
        if($1(1)>S2(1))
                return 1;
        else if($1(1)<$2(1))
                return -1;
        else
                return 0;
}
single:(..):bomb:{
        if($1(1)>S2(1))
                return 1;
        else if($1(1)<$2(1))
                return -1;
        else
                return 0;
}
</mode>
<func></func>
```

One Possible Rule File For the Game Doudizhu

```
<body>
//发牌
int num = 1;
while (get_cardnum(player[0])>3){
        deliver(player[0], player[num], topcard(player[0]));
        num = (num=1）% (players + 1);
        if (num==0) num = 1;
}
//叫牌
int[] y_num = {0, 1, 2, 3};
int host, tmp, point;
host = 0;
int tmp;
post_yield(player[1], y_num);
tmp = get_yield(player[1]);
if (tmp1!=0){
        host = 1 ;
        point = tmp;
}
```

One Possible Rule File For the Game Doudizhu

```
update(y_num, tmp);
post_yield(player[2], y_num);
tmp = get_yield(player[2]);
if (tmp!=0){
        host = 2;
        point = tmp;
}
update(y_num, tmp);
post_yield(player[3], y_num);
tmp = get_yield(player[3]);
if (tmp!=0){
        host = 3;
        point = tmp;
}
if (host==0) begin;
while (get_cardnum(player[0])!=0){
        deliver(player[0], player[host], topcard(player[0]));
}
```

One Possible Rule File For the Game Doudizhu

```
//出牌
int current_player, pre_owner;
current_player = host;
card[] pre_card = NONE;
pre_owner = 0;
card[] tmpcard;
while (get_cardnum(player[1])*get_cardnum(player[2])* get_cardnum_player[3]!=0){
        enable(current_player);
        tmpcard = get_card(current_player, pre_card);
        if (tmpcard!=NONE){
                deliver(player[current_player], player[0], tmpcard);
                pre_card = tmpcard;
                pre_owner = current_player;
        }
        disable(current_player);
        current_player = (current_player+1) % 4;
        if (current_player == 0) current_player = 1;
        if (pre_owner==current_player) pre_card = NONE;
}
end();</body>
```

One Possible Rule File For the Game Doudizhu

```
<end>if (get_cardnum(player[host]==0)){
        add_score(player[host], score * 2);
        for(int i=1;i<=players;i++){
                if(i!=host)
                        showmessage(player[i],"Sorry you have failed!>.<");
                else
                        showmessage(players[i],"Haha,you are the winner ^0^");
        }
}
else{
        add_score(player[1], score);
        add_score(player[2], score);
        add_score(player[3], score);
        add_score(player[host],-score);
        for(int i=1;i<=players;i++){
                if(i==host)
                        showmessage(player[i],"Sorry you have failed!>.<");
                else
                        showmessage(players[i],"Haha,you are the winner ^0^");
        }
}</end>
```