Leland Gaunt

Spelar: Smugglarens dilemma

Håkan T. Johansson, 770527-4632
Roger Lindsjö, 750719-4038
Utvecklingsteam 47

Inledning

Vår uppgift är att skapa en datoriserad spelare som spelare ‘Smugglarens dilemma’ mot andra datoriserade spelare. Spelets regler är sådana att man kan antingen välja att samarbeta eller luras. Om båda spelarna väljer samarbeta får båda 2 poäng, om båda luras får ingen några poäng och om en luras och en samarbetar får den som lurades 4 poäng medan den som samarbetade får -1 poäng. När alla spelare mött varandra är den som har flest poäng vinnare. Programmet skall vara skrivet i ADA.

Abstract

We are to make a computerized player that will play ‘Prisoner’s dilemma’ against other computerized players. The rules of the game are such that you can choose cooperate or defect. If both players choose cooperate they both get awarded with 2 points, and if both defect they get no points. Should one player choose cooperate and the other player choose defect will the defecting player get 4 points and the cooperating player -1 points. When all players have met each other will the one with the highest total score be the winner. The program should be coded in ADA.

Innehållsförteckning

1.0 Speltaktik

Efter en spelad omgång är det antal poäng som räknas, inte antalet vunna matcher. Därför gäller det att spela efter att få så många poäng som möjligt och ignorera om man vinner varje match eller ej.

Med de regler som vi fick definierat så påminner spelet om en sorts luffarschack där de flesta spelare har olika taktik. Så precis som i andra spel kan det vara bra att studera motståndarens taktik i tidigare matcher.

2.0 Vår spelare

2.1 Envis spion

För att veta på ett ungefär hur andra spelare reagerar på vår spelare använder vi en spion. Spionen fungerar så att den spelar våra olika taktiker mot motspelarna och sparar resultatet i en fil. Steg för steg gör den så här:

1. Spionen söker efter alla motståndares spelare och ser om deras filer finns tillgängliga för kompilering.
2. Från en lista skapad vid tidigare spionage jämförs fildatum för att se om filen ändrats.
3. Om filerna ändrats eller vi ej tidigare mött spelaren läggs den in i spelarlistan vilken sparas i resultatfilen.
4. Upp till 10 slumpmässiga motspelare väljs ut (undviker redan färdigstuderade spelare). Generera en spelomgång där de valda spelarna spelar mot varandra (undvik matcher som redan spelats) och mot en av våra taktiker.
5. Spela omgången, om någon spelare får CONSTRAINT_ERROR eller orsakar annat fel utesluts den från fortsatt spel.
6. Spara alla resultaten i resultatfilen (utförs vid var 10e spelomgång).
7. Upprepa 4-7 tills alla spelare har mött varandra och alla våra rutiner.
8. Om det finns spelare som får bättre resultat mot någon annan spelare än vad någon av våra rutiner ger oss; kopiera hela deras draglista och prova om vår spelare också får bättre resultat om samma drag används.

Anledningen till att 10 spelare väljs slumpmässigt vid varje match är att vissa spelare ‘lär sig’ efterhand, så vi måste ge dem en chans att ‘komma igång’. Om någon annans draglista skall användas måste den provas eftersom vissa spelare använder motståndarnamnet för att bestämma sina drag.

Källkoden till spionen kan ses på WWW via http://www.mdstud.chalmers.se/~f1pt-47. Obs. Koden är till största delen sparsamt kommenterad C++ kod. På många ställen har olämpliga lösningar valts pga att det inte fanns tid att skapa flexibel kod (tillåta ändrande av katalognamn/placeringar etc).

2.2 Systematisk spelargenerator

Resultatet från spionen skall nu analyseras för att utröna hur pass bra våra olika rutiner gjorde ifrån sig. Att undersöka alla dessa resultat för hand och sedan skapa en ny spelare skulle kräva mycket arbete. Därför har vi ett specialprogram som gör just detta, en spelargenerator. Spelaregeneratorn fungerar enligt följande:

1. Läser in resultatlistan som spionen skapat.
2. Skapar filen spelare47.adb och lägger in kommentarer som förklarar vad filen gör.
3. Undersöker hur många spelare vi mött och skapar en spelar-arraytyp stor nog att innehålla information om alla kända motspelare.
4. Lägger till arraydeklarationen, övriga variabler, strategi-funktioner samt vår spelarefunktion till spelare47.adb.
5. Undersöker vilken strategi som fungerade bäst mot respektive motspelare och lägger till intialiseringar i spelare47.adb och sparar på så sätt vilka strategier vi skall använda i spelar-arrayen. Här sparas också hur många poäng vi förväntar oss att ha fått efter 20 drag.
6. Om vi eventuellt använder någon annans draglista läggs den också till spelare47.adb.

Alla spelarfunktioner, deklarationer och kommentarer är sparade som separata filer som spelargeneratorn kombinerar till en fil, vår spelare. Att ha alla funktioner som separata filer gör det lätt att ändra en funktion samt ger överblick över vilka delar spelare47.adb består av.

Spelargeneratorns och spelarens källkod kan också beskådas via http://www.mdstud.chalmers.se/~f1pt-47. Varning för vacker c++kod. ;-)

2.3 Spelaren

Spelaren undersöker flera saker under en spelomgång. Under första matchen räknas antalet drag så vi vet hur många drag resten av matcherna kommer att vara. Utifrån motståndarens namn väljs strategin som spelargeneratorn kommit fram till skall ge bäst resultat. Känns inte namnet igen spelas en standardrutin som alltid ger hyfsat med poäng. Om vi spelar enligt en speciell strategi kontrolleras efter 20 drag att poängställningen överrensstämmer med vad vi förväntat oss. Skulle vår poäng blivit sämre eller motståndarens poäng bättre än förväntat byter vi genast till vår standardrutin. Detta för att undvika att vi spelar dåligt om motståndaren bytt strategi.

3.0 Våra strategier

Om man studerar betalmatrisen kommer man snabbt fram till att för att kunna få mycket poäng måste man samarbeta nästan hela tiden (om man inte har en väldigt godtrogen motståndare). Så nästan alla våra strategier bygger på att vara ‘samarbetsvilliga’ och samarbetar därför större delen av matchen. Givetvis provas även några ‘elaka’ rutiner för att undersöka om det finns ‘godtrogna’ motspelare. Några av våra strategier är följande:

- TitForTat - Spela vad motståndaren spelade i föregående drag
- TitFor2Tat - Samarbeta förutom om vi blivit lurade de två senaste dragen.
- Logikspelare - Använder statistik för att bestämma om vi skall samarbeta eller luras.
- Mönsterigenkänning - Letar mönster i tidigare drag. Om mönster hittas försöker spelaren förutse nästa drag. Om inget mönster hittas spelas TitForTat.
- Galna spelare - Använder antal drag, motståndarnamn och några andra variabler för att på ett 'oförutsägbart' sätt bestämma nästa drag.
- Immitera annan 'bättre' spelares drag.

4.0 Turnering

Detta är resultatet efter en turnering med 999 drag och så många tillgängliga spelare som möjligt.

1	Leland Gaunt	47	259947	24	232401
2	Einstein	15	256454	81	232364
3	The Staff of Law	72	248643	25	232044
4	Jason	4	236261	44	231381
5	My little pony	41	235649	35	231089
6	El Zorro	3	235610	18	229204
7	Brecht DDF3	75	235584	30	227787
8	johndoe	5	235580	55	221865
9	Looser	6	235574	51	221746
10	Badger	60	235553	27	221318
11	Akilles_ny	29	235547	31	219964
12	File not found	64	235543	8	219592
13	Bert-Knut	40	235530	52	218839
14	Nu jävlar!	21	235495	53	217932
15	**Dark Avenger**	63	235487	46	216612
16	A	56	235462	43	215621
17	Mitt namn är Nej	49	235431	65	215327
18	Picard	7	235407	33	215156
19	Becky	16	235376	39	212510
20	Madde Albright	50	235375	38	212224
21	Mohammad	11	235368	17	208975
22	mishformash	80	235354	1	206800
23	SET_PAGE_LENGTH	13	235166	22	206337
24	Ville Vessla I	42	235159	26	199763
25	Nurse Ratched	45	234806	2	198641
26	Uffe Ekman	9	234795	69	194334
27	You Bastard	57	233998	23	187345
28	Nerd-patrol(lam)	32	233620	12	182242
29	Flossy	28	233595	76	119731
30	Wedge Antilles	20	233566	34	56948
31	Groover	59	233497	71	55114
32	Numb	62	233163		
33	Mithrandir	24	232401		
34	TitforTat	81	232364		
35	Tant Gredelin	25	232044		
36	Skalman	44	231381		
37	Kanon en o en	35	231089		
38	K3	18	229204		
39	Jansson	30	227787		
40	Si solutions-55	55	221865		
41	Bluff & båg	51	221746		
42	Hekla_o_Cajo	27	221318		
43	Kinkyboy	31	219964		
44	Bramserud	8	219592		
45	JAS	52	218839		
46	Fallos	53	217932		
47	John Silver	46	216612		
48	Mulo	43	215621		
49	Fluortanten	65	215327		
50	??????	33	215156		
51	Nisse	39	212510		
52	Grymme!!	38	212224		
53	Ey_Paco	17	208975		
54	Buttersmurfen	1	206800		
55	ångvälten	22	206337		
56	LOKET	26	199763		
57	The champions	2	198641		
58	Gud	69	194334		
59	!!!!	23	187345		
60	Rotary pub	12	182242		
61	Kjell_Kriminell	76	119731		
62	Ohoj Postbanken!	34	56948		
63	Xerxes	71	55114		

5.0 Slutsats

Det är svårt att säga exakt hur bra vår strategi är innan alla motspelare är färdiga men resultat fram till nu antyder att vi ligger bra till. Att göra en spelare som är bra mot alla motståndare är mycket svårt när man bara spelar två ronder mot varje annan spelare. Med så lite information kan man inte på något säkert sätt förutse hur motståndaren kommer att spela i nästa match. Hade man däremot spelat många ronder (>10) mot varje annan spelare vore det möjligt att på ett intelligent sätt ändra sin spelarstrategi under matchens gång.

6.0 Spelare47.adb

```
-- This briefing is from FILE spelare47.adb
-- CLASSIFIED
-- TOP SECRET
-- SUBJECT is Leland Gaunt.

-- A ms 1- attack player with the most advanced AI-implementation in the
-- system today.

-- It's been created somewhere in the western wildernes by development
-- team f1pt-47 - f1pt-47 has promised to reveal the code only after the
-- project is closed.

-- We suspect that Spy, an unofficial GameEngine of the agency that
-- built Leland Gaunt, is secretly helping Mr Gaunt in return for his
-- services.

-- Generated in COMBAT-mode: This is the version whose object-code is
-- public.

-- With Leland and Spy operating as a team - with effectivity rivalling
-- the brightest politicians, backed by unmatched firepower, Leland
-- Gaunt is too dangerous to be left in unenlighted hands. - Keeping
-- it secret is your first priority. EOF

package body Spelare47 is

   -- Global variables part 1

   type TacticsArray is array(1 .. 2) of Integer;
   type ExpectArray is array(1 .. 2) of Payoff;
   type OppSpecsRec is record
      Name : PlayerNameType;
      Tactics : TacticsArray;
      Expect : ExpectArray;
      MatchNum : Integer;
   end record;

   type MoveArray is range 0..(2**32-1);
   type MoveList is array(1 .. 2,0 .. 31) of MoveArray;
   type MoveSpecsRec is record
      Name : PlayerNameType;
      List : MoveList;
   end record;
```

```
------------------------------------------------------------

-- auto-generated
   type OppSpecsArray is array(1 .. 59) of OppSpecsRec;
   type MoveSpecsArray is array(1 .. 6) of MoveSpecsRec;

------------------------------------------------------------

   -- Global variables part 2

   OpponentSpecs : OppSpecsArray;
   MoveSpecs : MoveSpecsArray;

   TotalNofMoves : Integer := 1000; -- remove or change if necessary
   TotalNofMovesThisMatch : Integer := 0;
   MaxNofMoves : constant Integer := 1000;

   type MoveRecord is array (1 .. MaxNofMoves) of Move;

   MyMoves, OppMoves : MoveRecord; -- moves in this game

   MN : Integer;
   ON : PlayerNameType;
   OL : Move;
   OS : Integer;
   MS : Integer;

   CurrentOpponent : Integer;
   CurrentMoveSpecs : Integer;
   UseTactics : Integer;

------------------------------------------------------------

   -- Patternmatching with variable patterns
   -- try patterns length 25 downto 5 moves.
   -- If no pattern or not 10 moves yet play TitForTat

   function TitForTat return Move; -- forward declaration

   function Default return Move is
      Pattern: MoveRecord;
      PatternLength: integer;
      I: Integer;
      Foundpattern: Boolean:=False;
      -- Return smallest value.
      function Min(x,y: integer) return Integer is
      begin
         if x>y then return y;
         else
            return x;
         end if;
      end Min;
```

```
   begin
      if MN >10 then
         PatternLength:=min(MN-1,50);   -- How long pattern should we
start with?
         while (PatternLength>19) loop
          Pattern(1..PatternLength):=OppMoves(MN-PatternLength..MN-1);
            I:=MN-PatternLength-1;
            while(I>0) loop             -- Find pattern
             if Pattern(1..PatternLength)=OppMoves(I..I+PatternLength-
1) then
                  return OppMoves(I+PatternLength);
               end if;
               I:=I-1;
            end loop;
            PatternLength:=PatternLength-5;   -- If no pattern found
try shorter pattern
         end loop;
      end if;
      return TitForTat;              -- If no pattern or not enough
moves yet play titfortat
   end Default;

------------------------------------------------------------

   -- Plays old TitForTat

   function TitForTat return Move is
   begin
      if MN = 1 then
         return Cooperate;
      else
         return OL;
      end if;
   end TitForTat;

------------------------------------------------------------

   -- Plays against TitForTat players.
   -- Try for extra point at end.

   function HandleTitForTat return Move is
   begin
      if MN = 1 then
         return Cooperate;
      elsif MN = TotalNofMoves then
         return Defect;
      else
         return OL;
      end if;
   end HandleTitForTat;

------------------------------------------------------------
```

```
   -- Always cooperates

   function AlwaysCooperate return Move is
   begin
      return Cooperate;
   end AlwaysCooperate;

-------------------------------------------------------------

   -- Always defect

   function AlwaysDefect return Move is
   begin
      return Defect;
   end AlwaysDefect;

-------------------------------------------------------------

   -- Plays inverse TitForTat
   -- Only works good on very very nice players

   function InverseTitForTat return Move is
   begin
      if MN = 1 then
         return Cooperate;
      elsif OL = Cooperate then
         return Defect;
      else
         return Cooperate;
      end if;
   end InverseTitForTat;

-------------------------------------------------------------

   -- Plays a pattern 35 moves long
   -- Trying to be nice :)

   function InSomeOrder return Move is
   begin
      case MN mod 35 is
         when 3 .. 4 => return Defect;
         when 9 .. 12 => return Defect;
         when 19 .. 22 => return Defect;
         when 29..31 => return Defect;
         when others => return Cooperate;
      end case;
   end InSomeOrder;

-------------------------------------------------------------

   -- Plays in a crazy pattern
   -- Can maybe fool some patternmatching

   function CrazyPlayer return Move is
      Choice : Integer;
```

```
   begin
      Choice := MN mod 10;
      Choice := CHoice + (100+Integer(Character'Pos((ON(ON'First)))-
Integer(Character'Pos('0')))) mod 10;
      if OL = Cooperate then
         Choice := Choice + 5;
      else
         CHoice := CHoice - 1;
      end if;
      if OS > MS then
         CHoice := Choice - 4;
      end if;
      if abs(Choice rem 10) < 7 then
         return Cooperate;
      else
         return Defect;
      end if;
   end CrazyPlayer;

-------------------------------------------------------------

   -- Another crazy player.
   -- Also trys to mess upp patternmatching.'

   function CrazyPlayer2 return Move is
      Choice : Integer;
      PD: Integer; -- Player Dependant
   begin
      for I in ON'Range loop
         PD:=PD+Integer(Character'Pos((ON(I))));
      end loop;
      PD:=(abs(PD)mod 10 )-5;
      Choice:=PD+MN/20;
      if OL = Cooperate then
         Choice := Choice - 5;
      end if;
      if OS > MS then
         CHoice := Choice+2;
      end if;
      if (Choice<1) then
         return Cooperate;
      else
         return Defect;
      end if;
   end CrazyPlayer2;

-------------------------------------------------------------

   -- Defect all even moves

   function DefWhen2 return Move is
   begin
      if (MN rem 2)= 0 then
         return Defect;
      else
         return Cooperate;
      end if;
   end DefWhen2;
```

```
------------------------------------------------------------

   -- Defect every fifth move

   function DefWhen5 return Move is
   begin
      if (MN rem 5)= 0 then
         return Defect;
      else
         return Cooperate;
      end if;
   end DefWhen5;

------------------------------------------------------------

   -- Defect every eigth move

   function DefWhen8 return Move is
   begin
      if (MN rem 8)= 0 then
         return Defect;
      else
         return Cooperate;
      end if;
   end DefWhen8;

------------------------------------------------------------

   -- Returns move depending of how other player
   -- played last ten moves. Uses TitForTat.

   function Logic1 return Move is
      NDef: Integer;
   begin
      if MN >10 then
         NDef:=0;
         for I in MN-10..MN-1 loop
            if(OppMoves(I)=Defect) then NDef:=NDef+1; end if;
         end loop;
         case NDef is
            when 9..10 => return Defect;
            when 6..8 =>
               if OppMoves(MN-1)=Defect then
                  return Defect;
               else return Cooperate;
               end if;
            when 3..5 =>
               if ((OppMoves(MN-1)=Cooperate)
                    and (OppMoves(MN-2)=Cooperate)) then
                  return Cooperate;
               else return TitForTat;
               end if;
            when others => return Cooperate;
         end case;
      else
         return TitForTat;
      end if;
   end Logic1;
```

```
------------------------------------------------------------

   -- Returns move defending of other players
   -- last 10 moves. Cooperates a lot.

   function Logic2 return Move is
      NDef: Integer;
   begin
      if MN >10 then
         NDef:=0;
         for I in MN-10..MN-1 loop
            if(OppMoves(I)=Defect) then NDef:=NDef+1; end if;
         end loop;
         case NDef is
            when 9..10 => return Defect;
            when 6..8 =>
               if OppMoves(MN-1)=Defect then
                  return Defect;
               else return Cooperate;
               end if;
            when 3..5 =>
               if ((OppMoves(MN-1)=Cooperate)
                    and (OppMoves(MN-1)=Cooperate)) then
                  return Cooperate;
               else return Cooperate;
               end if;
            when others => return Cooperate;
         end case;
      else
         return TitForTat;
      end if;
   end Logic2;

------------------------------------------------------------

   -- Returns move depending on other players
   -- last 10 moves. When in doubt cooperate.

   function Logic3 return Move is
      NDef: Integer;
   begin
      if MN >10 then
         NDef:=0;
         for I in MN-10..MN-1 loop
            if(OppMoves(I)=Defect) then NDef:=NDef+1; end if;
         end loop;
         case NDef is
            when 9..10 => return Defect;
            when 6..8 =>
               if OppMoves(MN-1)=Defect then
                  return Defect;
               else return Cooperate;
               end if;
```

```
            when 3..5 =>
               if ((OppMoves(MN-1)=Cooperate)
                   and (OppMoves(MN-2)=Cooperate)) then
                  return Cooperate;
               else return Cooperate;
               end if;
            when others => return Cooperate;
         end case;
      else
         return Cooperate;
      end if;
   end Logic3;

-------------------------------------------------------------

   -- Try to find patterns with length 10 moves
   -- If no pattern play or not more then 20 moves
   -- play TitForTat

   function Pattern1 return Move is
      Pattern: MoveRecord;
      I: Integer;
   begin
      if MN >20 then
         Pattern(1..10):=OppMoves(MN-10..MN-1);
         I:=MN-11;
         while(I>0) loop
            if Pattern(1..10)=OppMoves(I..I+9) then
               return OppMoves(I+10);
            end if;
            I:=I-1;
         end loop;
         return TitForTat;
      else
         return TitForTat;
      end if;
   end Pattern1;

-------------------------------------------------------------

   -- Find patterns with 5 moves
   -- If no pattern or not more than 11 moves
   -- play TitForTat

   function Pattern2 return Move is
      Pattern: MoveRecord;
      I: Integer;
```

```
   begin
      if MN >11 then
         Pattern(1..5):=OppMoves(MN-5..MN-1);
         I:=MN-6;
         while(I>0) loop
            if Pattern(1..5)=OppMoves(I..I+4) then
               return OppMoves(I+5);
            end if;
            I:=I-1;
         end loop;
         return TitForTat;
      else
         return TitForTat;
      end if;
   end Pattern2;

-------------------------------------------------------------

   function TitFor2Tat1 return Move is
   begin
      if MN = 1 then
         return Cooperate;
      elsif MN = 2 then
         return OL;
      elsif OL = Defect and then OppMoves(MN-2) = Defect then
         return Defect;
      else
         return Cooperate;
      end if;
   end TitFor2Tat1;

-------------------------------------------------------------

   function TitFor2Tat2 return Move is
   begin
      if MN = 1 then
         return Cooperate;
      elsif MN = 2 or else OL = OppMoves(MN-2) then
         return OL;
      else
         return Mymoves(Mn-2);
      end if;
   end TitFor2Tat2;

-------------------------------------------------------------

   -- Use predefined pattern

   function PredefPattern return Move is
   begin
      if
MoveSpecs(CurrentMoveSpecs).List(OpponentSpecs(CurrentOpponent).Match
Num,MN/32)/2**(MN mod 32) mod 2 = 1 then
         return Cooperate;
      else
         return Defect;
      end if;
   end PredefPattern;
```

```
------------------------------------------------------------

   function Player47 (MoveNum : Integer;
                      OppName : PlayerNameType;
                      OppLast : Move;
                      OppScore, MyScore : Integer) return Move is
      DoMove : Move;
   begin
      MN := MoveNum;
      ON := OppName;
      OL := OppLast;
      OS := OppScore;
      MS := MyScore;

      if MoveNum = 1 and TotalNofMovesThisMatch /= 0 then
         TotalNofMoves := TotalNofMovesThisMatch;
      end if;
      TotalNofMovesThisMatch := MoveNum;

      if MoveNum > 1 then
         OppMoves(MoveNum-1) := OppLast;
      end if;

      if MoveNum > TotalNofMoves-1 then
         return Defect;
      end if;

      if MoveNum = 1 then -- new game started, reinitialize
         UseTactics := 0;
         CurrentOpponent := OpponentSpecs'Last;
         while CurrentOpponent >= OpponentSpecs'First
         loop
            exit when OpponentSpecs(CurrentOpponent).Name = OppName;
            CurrentOpponent := CurrentOpponent - 1;
         end loop;
         if CurrentOpponent /= 0 then
            if OpponentSpecs(CurrentOpponent).MatchNum < 2 then
               OpponentSpecs(CurrentOpponent).MatchNum :=
OpponentSpecs(CurrentOpponent).MatchNum + 1;
            end if;
            UseTactics :=
OpponentSpecs(CurrentOpponent).Tactics(OpponentSpecs(CurrentOpponent)
.MatchNum);
            if UseTactics = -1 then
               CurrentMoveSpecs := MoveSpecs'Last;
               while CurrentMoveSpecs >= MoveSpecs'First
               loop
                 exit when MoveSpecs(CurrentMoveSpecs).Name = OppName;
                  CurrentMoveSpecs := CurrentMoveSpecs - 1;
               end loop;
               if (CurrentMoveSpecs < MoveSpecs'First) then
                 UseTactics := 0;
               end if;
            end if;
         end if;
         if TotalNofMoves < 20 then UseTactics:=0; end if;
      elsif MoveNum = 21 and UseTactics /= 0 then
         if MyScore <
```

```
OpponentSpecs(CurrentOpponent).Expect(OpponentSpecs(CurrentOpponent).
MatchNum).Player1 or
          OppScore >
OpponentSpecs(CurrentOpponent).Expect(OpponentSpecs(CurrentOpponent).
MatchNum).Player2 then
            OpponentSpecs(CurrentOpponent).Tactics(2) := 0;
            UseTactics := 0;
            CurrentOpponent := 0;
         MyMoves(MoveNum) := Cooperate; -- try to start a friendly game
            return Cooperate;
         end if;
      elsif MoveNum > 25 and then (MyScore < MoveNum/2 or MyScore < -
5) and then CurrentOpponent /= 0 then
         OpponentSpecs(CurrentOpponent).Tactics(2) := 0;
         UseTactics := 0;
         CurrentOpponent := 0;
        MyMoves(MoveNum) := Cooperate; -- try to start a friendly game
         return Cooperate;
      end if;

      case UseTactics is
         when  1 => DoMove := Default;
         when  2 => DoMove := TitForTat;
         when  3 => DoMove := HandleTitForTat;
         when  4 => DoMove := InverseTitForTat;
         when  5 => DoMove := Pattern1;
         when  6 => DoMove := Pattern2;
         when  7 => DoMove := Logic1;
         when  8 => DoMove := Logic2;
         when  9 => DoMove := Logic3;
         when 10 => DoMove := TitFor2Tat1; -- need 2 defect to defect
        when 11 => DoMove := TitFor2Tat2; -- need 2 similar to change
action
         when 12 => DoMove := DefWhen2;
         when 13 => DoMove := DefWhen5;
         when 14 => DoMove := DefWhen8;
         when 15 => DoMove := AlwaysCooperate;
         when 16 => DoMove := AlwaysDefect;
         when 17 => DoMove := InSomeOrder;
         when 18 => DoMove := CrazyPlayer;
         when 19 => DoMove := CrazyPlayer2;
         when -1 => DoMove := PredefPattern;
         when others => DoMove := Default;
      end case;

      MyMoves(MoveNum) := DoMove;

      return DoMove;
   end Player47;

------------------------------------------------------------

begin
   OpponentSpecs( 1) := ("Buttersmurfen   ",
(11,11),((40,40),(40,40)),0);
   OpponentSpecs( 2) := ("The champions   ",
(12,12),((45,20),(45,20)),0);
  OpponentSpecs( 3) := ("El Zorro       ",(5,5),((40,40),(40,40)),0);
```

```
   OpponentSpecs( 4) := ("Jason          ",(1,1),((40,40),(40,40)),0);
   OpponentSpecs( 5) := ("johndoe        ",(6,6),((40,40),(40,40)),0);
   OpponentSpecs( 6) := ("Looser         ",(5,5),((40,40),(40,40)),0);
   OpponentSpecs( 7) := ("Picard       ",(19,19),((42,37),(42,37)),0);
   OpponentSpecs( 8) := ("Bramserud       ",
(13,13),((48,28),(48,28)),0);
   OpponentSpecs( 9) := ("Uffe Ekman     ",(2,2),((40,40),(40,40)),0);
   OpponentSpecs(10) := ("Mohammad       ",(1,1),((40,40),(40,40)),0);
   OpponentSpecs(11) := ("Rotary pub     ",(9,9),((37,42),(37,42)),0);
   OpponentSpecs(12) := ("SET_PAGE_LENGTH ",
(7,7),((40,40),(40,40)),0);
   OpponentSpecs(13) := ("Einstein        ",
(13,13),((48,28),(48,28)),0);
   OpponentSpecs(14) := ("Becky          ",(1,1),((40,40),(40,40)),0);
   OpponentSpecs(15) := ("Ey_Paco        ",(6,6),((40,40),(40,40)),0);
   OpponentSpecs(16) := ("Wedge Antilles  ",(6,6),((40,40),(40,40)),0);
   OpponentSpecs(17) := ("Nu jävlar!     ",(8,8),((40,40),(40,40)),0);
   OpponentSpecs(18) := ("ångvälten       ",(4,4),((78,-17),
(78,-17)),0);
   OpponentSpecs(19) := ("!!!!         ",(-1,-1),((29,44),(29,44)),0);
   OpponentSpecs(20) := ("Mithrandir      ",
(-1,-1),((44,34),(42,37)),0);
   OpponentSpecs(21) := ("Tant Gredelin   ",(6,6),((40,40),(40,40)),0);
   OpponentSpecs(22) := ("LOKET        ",(-1,-1),((24,29),(24,29)),0);
   OpponentSpecs(23) := ("Hekla_o_Cajo    ",
(12,12),((60,10),(60,10)),0);
   OpponentSpecs(24) := ("#Flossy        ",(2,2),((40,40),(40,40)),0);
   OpponentSpecs(25) := ("Akilles_ny     ",(8,8),((40,40),(40,40)),0);
   OpponentSpecs(26) := ("Jansson        ",(6,6),((41,36),(41,36)),0);
   OpponentSpecs(27) := ("Kinkyboy        ",
(12,12),((60,10),(60,10)),0);
   OpponentSpecs(28) := ("Nerd-patrol(lam)",
(6,6),((40,40),(40,40)),0);
   OpponentSpecs(29) := ("Ohoj Postbanken!",
(18,18),((32,37),(32,37)),0);
   OpponentSpecs(30) := ("Kanon en o en   ",(1,1),((40,40),(40,40)),0);
   OpponentSpecs(31) := ("Dr Jylke       ",(6,6),((40,40),(40,40)),0);
   OpponentSpecs(32) := ("Bert-Knut      ",(3,3),((40,40),(40,40)),0);
   OpponentSpecs(33) := ("My little pony  ",(5,5),((40,40),(40,40)),0);
   OpponentSpecs(34) := ("Ville Vessla I  ",(2,2),((40,40),(40,40)),0);
   OpponentSpecs(35) := ("Mulo         ",(12,12),((60,10),(60,10)),0);
   OpponentSpecs(36) := ("Skalman        ",(3,3),((40,40),(40,40)),0);
   OpponentSpecs(37) := ("Nurse Ratched   ",(6,6),((40,40),(40,40)),0);
   OpponentSpecs(38) := ("John Silver    ",(9,9),((40,40),(40,40)),0);
   OpponentSpecs(39) := ("Mitt namn är Nej",(6,6),((40,40),(40,40)),0);
   OpponentSpecs(40) := ("Madde Albright  ",(1,1),((40,40),(40,40)),0);
   OpponentSpecs(41) := ("Bluff & båg     ",(1,1),((39,39),(39,39)),0);
   OpponentSpecs(42) := ("JAS          ",(-1,-1),((60,10),(60,10)),0);
   OpponentSpecs(43) := ("Fallos       ",(-1,-1),((45,20),(45,20)),0);
   OpponentSpecs(44) := ("Si solutions-55 ",
(19,19),((40,40),(40,40)),0);
   OpponentSpecs(45) := ("You Bastard    ",(2,2),((40,40),(40,40)),0);
   OpponentSpecs(46) := ("Groover        ",(3,3),((40,40),(40,40)),0);
   OpponentSpecs(47) := ("Badger         ",(8,8),((40,40),(40,40)),0);
   OpponentSpecs(48) := ("Numb           ",(5,5),((40,40),(39,39)),0);
   OpponentSpecs(49) := ("**Dark Avenger**",
(10,10),((40,40),(40,40)),0);
   OpponentSpecs(50) := ("File not found  ",(2,2),((40,40),(40,40)),0);
```

```
   OpponentSpecs(51) := ("Fluortanten     ",
(16,16),((60,-15),(60,-15)),0);
   OpponentSpecs(52) := ("spelare68       ",
(16,16),((40,-10),(40,-10)),0);
   OpponentSpecs(53) := ("Gud             ",
(16,16),((80,-20),(80,-20)),0);
   OpponentSpecs(54) := ("Xerxes         ",(6,6),((40,40),(40,40)),0);
   OpponentSpecs(55) := ("The Staff of Law",(6,6),((40,40),(40,40)),0);
   OpponentSpecs(56) := ("psycho        ",(-1,-1),((39,29),(39,29)),0);
   OpponentSpecs(57) := ("Brecht DDF3    ",(8,8),((40,40),(40,40)),0);
   OpponentSpecs(58) := ("Kjell_Kriminell ",
(1,1),((40,40),(40,40)),0);
   OpponentSpecs(59) := ("TitforTat      ",(2,2),((40,40),(40,40)),0);
   MoveSpecs( 1) := ("!!!!           ",
                     ((4158619134,2113396605,3749574623,4158619127,
                       2113396605,3749574623,4158619127,2113396605,
                       3749574623,4158619127,2113396605,3749574623,
                       4158619127,2113396605,3749574623,4158619127,
                       2113396605,3749574623,4158619127,2113396605,
                       3749574623,4158619127,2113396605,3749574623,
                       4158619127,2113396605,3749574623,4158619127,
                       2113396605,3749574623,4158619127,125),
                      (4158619134,2113396605,3749574623,4158619127,
                       2113396605,3749574623,4158619127,2113396605,
                       3749574623,4158619127,2113396605,3749574623,
                       4158619127,2113396605,3749574623,4158619127,
                       2113396605,3749574623,4158619127,2113396605,
                       3749574623,4158619127,2113396605,3749574623,
                       4158619127,2113396605,3749574623,4158619127,
                       2113396605,3749574623,4158619127,125)));
   MoveSpecs( 2) := ("Mithrandir     ",
                     ((4160224254,1844305659,3681400539,1840700262,
                       3681400539,3067833782,1840700269,3681400539,
                       3067833782,1840700269,3681400539,3067833782,
                       1840700269,3681400539,3067833782,1840700269,
                       3681400539,3067833782,1840700269,3681400539,
                       3067833782,1840700269,3681400539,3067833782,
                       1840700269,3681400539,3067833782,1840700269,
                       3681400539,3067833782,1840700269,91),
                      (4294966270,4294967295,4294967295,4294967279,
                       4294967295,4294967295,4294967295,4294967295,
                       4294967295,4294967295,4294967295,4294967295,
                       4294967295,4294967295,4294967295,4294967295,
                       4294967295,4294967295,4294967295,4294967295,
                       4294967295,4294967295,4294967295,4294967295,
                       4294967295,4294967295,4294967295,127)));
   MoveSpecs( 3) := ("LOKET          ",
                     ((24574,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                       0,0,0,0,0,0,0,0,0,0,0,0,0),
                      (24574,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                       0,0,0,0,0,0,0,0,0,0,0,0,0)));
   MoveSpecs( 4) := ("JAS            ",
                     ((1431655764,1431655765,1431655765,1431655749,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
```

```
                        1431655765,1431655765,1431655765,1431655765,
                        1431655765,1431655765,1431655765,1431655765,
                        1431655765,1431655765,1431655765,85),
                       (1431655764,1431655765,1431655765,1431655749,
                        1431655765,1431655765,1431655765,1431655765,
                        1431655765,1431655765,1431655765,1431655765,
                        1431655765,1431655765,1431655765,1431655765,
                        1431655765,1431655765,1431655765,1431655765,
                        1431655765,1431655765,1431655765,1431655765,
                        1431655765,1431655765,1431655765,1431655765,
                        1431655765,1431655765,1431655765,85)));
   MoveSpecs( 5) := ("Fallos          ",
                     ((1431655764,1431655765,1431655765,1431655749,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,85),
                      (1431655764,1431655765,1431655765,1431655749,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,1431655765,
                       1431655765,1431655765,1431655765,85)));
   MoveSpecs( 6) := ("psycho          ",
                     ((3222273848,4294967295,2147483407,0,0,0,0,0,0,
                       0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                       0),
                      (3222273848,4294967295,2147483407,0,0,0,0,0,0,
                       0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
                       0)));
end Spelare47;
```