Automated Generation of New Concepts from General Game Playing

Yuichiro Sato[1(✉)] and Tristan Cazenave[2]

[1] Japan Advanced Institute of Science and Technology, Nomi, Japan
sato.yuichiro@jaist.ac.jp
[2] Université Paris-Dauphine, Paris, France
cazenave@lamsade.dauphine.fr

Abstract. In this paper, we propose algorithms to extract explicit concepts from general games and these concepts are useful to understand semantics of games using General Game Playing as a research domain. General Game Playing is a research domain to invent game players which are able to play general games without any human intervention. There are many approaches to General Game Playing, for example, UCT, Neural Network, and Simulation-based approaches. Successful knowledge acquisition is reported in these approaches. However, generated knowledge is not explicit in conventional methods. We extract explicit concepts from heuristic functions obtained using a simulation based approach. Concepts to understand the semantics of Tic-tac-toe are generated by our approach. These concepts are also available to understand the semantics of Connect Four. We conclude that our approach is applicable to general games and is able to extract explicit concepts which are able to be understood by humans.

1 Introduction

An intelligent system is able to adapt itself to its environment. To invent artificial intelligence, it is a good strategy to make a program which learns knowledge from the environment. We can use our living world itself as the environment for a system, e.g., Natural Language Processing. In this study, we focus on artificial environments, i.e., games. A game has concrete rules, and is easy to simulate and evaluate without any human intervention. We tried to discover new concepts from experience in this artificial environment. We use General Game Playing (GGP) as a research domain. GGP is a research domain to invent game players which are able to play general games.

There are many General Game Players. J. Méhat et al. developed a UCT based player [8], and D. Michulke et al. developed a machine learning based player [9]. Also, simulation and knowledge based approaches are studied. T. Kaneko et al. developed a successful method to learn logical features using a Boolean network approach [5]. P. Skowronski et al. extracted features for selective search extensions [12]. H. Finnsson et al. also extracted knowledge for general games to improve search efficiency of their player [1,3]. However, generated

T. Cazenave et al. (Eds.): CGW 2013, CCIS 408, pp. 71–80, 2014.
DOI: 10.1007/978-3-319-05428-5_6,

knowledge is not explicit in these conventional research. In this study, we propose the generation of new concepts which are explicit enough to be understood by humans.

2 Method

We made our General Game Player using ECLiPSe Prolog and Java. It reads games expressed in the Game Description Language (GDL) [7] through a parser [11] into a Prolog engine, then the engine is called from Java through a Java-Prolog interface to simulate a game.

We automatically produced heuristics for the game by statistical analysis of simulation results. GDL is convertible to first-order predicate logic, therefore we can write GDL in Prolog [8]. Also, we can write positions of games as Prolog like facts. Thus, we made heuristics out of a set of Prolog rules. For each rule, the body is a subset of a game position, and the argument of head is the evaluation value of the position.

$$evaluation(value) : -a\_subset\_of\_position\_of\_game. \tag{1}$$

We used Tic-tac-toe and Connect Four as subject games. In Tic-tac-toe, x player and o player try to make a straight line on a 3×3 two-dimensional board using his/her mark. In Connect Four, white player and red player also try to make a straight line with one's color, but the board size is 7×6. Additionally, it has gravity, so the players need to drop the tokens of their color from the top and pile them up. Therefore, players need to play Connect Four using a different strategy to Tic-tac-toe.

Even though the game rules are different, the representation of the board is shared by both games. In both games, a board is represented by a set of cell terms with arity 3, two arguments are for the coordinate and one is for the mark or color on the cell. This situation is convenient for our purpose.

3 Automated Generation of Heuristic Functions from Simulations

We tried automated generation of heuristic functions from random simulations of Tic-tac-toe and Connect Four. J. Clune successfully generated heuristic functions for GGP [2]. M. Kirci et al. successfully extracted winning positions of general games from playouts [6]. We statistically evaluated positions.

We simulated Tic-tac-toe games till we got 10,000 playouts with each player as the winner. Then, we counted up what kind of subset of position is included in playouts and calculated the frequency to appear for each player. Finally, we picked the top 10 % of frequent subsets and made Prolog rules which have the frequency as its evaluation value as written in Algorithm 1. We cut off Prolog rules in the heuristic functions which have greater body size than a specific size.

We made three heuristic functions by using three different cut off sizes, 1, 2 and 3.

The following rules were included in generated heuristic functions.

$$evaluation(0.238...) : -cell(3,1,o), cell(2,2,o). \quad (2)$$

$$evaluation(0.237...) : -cell(3,3,o), cell(2,2,o), cell(1,3,x). \quad (3)$$

These heuristics are appropriate for Tic-tac-toe. An evaluation value of a position is a sum of evaluation values of heuristic functions which match to the position. We generated heuristic functions in the same way for Connect Four by 1000 simulations for each player. For Connect Four, we used a cut off size of 1 and 2. We omitted cut off size 3 to reduce simulation time.

We evaluated the performance of each heuristic function by simulation. In the simulation, a 1-depth alpha-beta search player with each heuristic function played against both a random player and a 1-depth alpha-beta search player without heuristics. The 1-depth alpha-beta search player chooses a winning move when it is found by 1-depth search. Otherwise, the player chooses a random move. We did 10,000 simulations for Tic-tac-toe, 1000 simulations for Connect Four. Against a random player, for both games, winning ratios tend to improve when the cut off size gets greater, as can be seen in Table 1. If the cut off size is 0, it means that the player has no heuristic function. The same tendency was seen against an alpha-beta search player in Table 2. These results suggest that this algorithm successfully extracted features of games properly and encoded them as heuristic functions.

Algorithm 1 makeStatisticalHeuristics(*playouts*)

```
M ⇐ Hash Map
for all playout p in playouts do
  S ⇐ getSubset(p)
  for all subset s in S do
    if M contains s then
      increment counter for s
    else
      create hash for s and set the counter as 1
    end if
  end for
end for
for all m in M do
  v ⇐ counter for m / size of playouts
  add Prolog rule type heuristic, "evaluation(v):-m." into H
end for
H ⇐ pick up rules which have top 10% of its evaluation value from H
return H
```

Table 1. Evaluation of heuristic functions for Tic-tac-toe and Connect Four against a random player.

		cut off size			
		0	1	2	3
game	player	win(%)/lose(%)/draw(%)			
Tic-tac -toe	x	80.73/12.32/6.95	93.38/4.53/2.09	90.17/7.45/2.38	96.49/2.76/0.75
	o	50.57/40.69/8.74	65.64/27.50/6.86	68.10/28.11/3.79	73.65/24.77/1.58
Connect Four	white	81.4/18.6/0	84.4/15.6/0	90.2/9.8/0	-
	red	71.3/28.6/0.1	70.4/29.6/0	77.4/22.6/0	-

Table 2. Evaluation of heuristic functions for Tic-tac-toe and Connect Four against a 1-depth alpha-beta search player.

		cut off size			
		0	1	2	3
game	player	win(%)/lose(%)/draw(%)			
Tic-tac-toe	x	67.98/27.51/4.51	85.81/13.57/0.62	74.53/25.47/0	91.42/8.58/0
	o	27.00/68.71/4.29	41.78/55.40/2.82	41.95/57.87/0.18	50.42/49.58/0
Connect Four	white	57.4/42.6/0	62.2/37.8/0	79.7/20.3/0	-
	red	40.1/59.9/0	40.4/59.6/0	54.4/45.6/0	-

4 Automated Generation of New Concepts for Games from Heuristic Functions

We consider that heuristic functions include essential concepts about games. We tried to extract them as explicit new concepts. This is the Predicate Invention, one of the research areas of Inductive Logic Programming [4].

First, for all Prolog rules in each heuristic function, we made pairs of terms included in its body. Then we replaced arguments with variables in the pairs if an argument in one of the terms in the pair relates to the corresponding argument in the other term. If an argument was a number, the argument was replaced by a new variable and the other was replaced by the sum of the variable and the difference between the two arguments. If the arguments were the same strings, we replaced them with a new variable. We introduced variables into original terms in this way. After removing duplicates, finally we got explicit new concepts from Prolog like heuristic functions, as written in Algorithm 2.

We extracted new concepts from heuristic functions for Tic-tac-toe generated in Sect. 3. Typical concepts are as follows.

$$concept1(X_0, X_1, X_2) : -cell(X_0, X_1, X_2), cell(X_0, X_1 + 1, X_2). \quad (4)$$

$$concept11(X_0, X_1, X_2) : -cell(X_0, X_1, X_2), cell(X_0 + 2, X_1 + 2, X_2). \quad (5)$$

$$concept20(X_0, X_1) : -cell(X_0, X_1, x), cell(X_0 + 2, X_1 + 2, o). \quad (6)$$

We are able to interpret these concepts as human language. Equation 4 means "a cell and its right cell are marked by the same symbol". Equation 5 means "a

cell and its lower right cell are marked by the same symbol". Equation 6 means "a cell is marked by x and its lower right cell is marked by o". These are natural concepts for humans to play Tic-tac-toe. Concepts which have a variable as the role argument are general concepts which are available for both Tic-tac-toe and Connect Four. Concepts which do not have a variable as the role argument are specialized for Tic-tac-toe and not available for Connect Four because role symbols are different between Tic-tac-toe and Connect Four.

Algorithm 2 generateNewConcept(*prolog_rules*)

```
for all rule r in prolog_rules do
  P ⇐ all pairs of terms in the body of r
  for all pair (p1, p2) in P do
    for i = 0 to arity of p1 do
      if i-th argument of p1, ai and i-th argument of p2, bi are instances of the
      same class then
        if ai is an instance of number then
          replace ai to a new variable and bi to sum of the variable and bi − ai
        else if ai equals to bi then
          replace ai and bi to a new variable
        end if
      end if
    end for
    if generated pair (p1, p2) is not in C then
      add generated pair (p1, p2) to C as a new concept
    end if
  end for
end for
return C
```

5 Applying Automated Generated Concepts to Games

We tried to use the generated concepts to reconstruct heuristic functions. We send queries to a Prolog engine whether each generated concept matches to subsets of simulated playouts. If the query matched, we saved the matching result and counted up how many times it matched. Then we assert a Prolog like heuristic rule of which an evaluation value is the ratio of number of matches frequency to number of subsets as written in Algorithm 3. In our experience, reconstructed heuristic functions are made of only binary relations because all of the generated concepts in Sect. 4 are binary relations on terms.

For Tic-tac-toe, we made heuristic functions from 10,000 random game playouts with each player as the winner, then we evaluated performance using 10,000 simulations against both a random game player and a 1-depth alpha-beta search player. In the results, improvements were seen compared to the player who has no heuristic function, for example, winning ratio of x player improved 14 %, but

not to players whose heuristic function is better than the 1-body size heuristic function as seen in Tables 3 and 4. We think this is because they lack 1-body and 3-body Prolog rules as we mentioned above.

We also made heuristic functions for Connect Four in the same way. We made two different heuristic functions, one is made from only 10 simulations and the other is made from 100 simulations. Even though only a few simulations, for white player, the generated heuristic functions have good performance. Only 10 simulations are enough to compete with 1-body heuristic functions. 100 simulations are enough to be competitive with 2-body heuristic functions as seen in Tables 3 and 4. This is proof that concepts learned from experience of small games can play bigger games.

We successfully generated new concepts for games from experience of Tic-tac-toe. However, for red player, the result is not good. The difference is that white is the first player and red is the second player. From random game simulation results, it is suggested that the second player has a disadvantage compared to the first player. The difference of performance is possibly due to this property.

Algorithm 3 makeStructuredHeuristics(*concepts*,*playouts*)

```
for all concept c in concepts do
  M ⇐ Hash Map
  for all playout p in playouts do
    S ⇐ getSubset(p)
    for all subset s in S do
      if prolog query of c matches to s then
        r ⇐ matching result of s
        if M contains r then
          increment counter for r
        else
          create a hash for r and set the counter as 1
        end if
      end if
      increment the number of subsets size
    end for
  end for
  for all m in M do
    v ⇐ the counter for m / size
    add a heuristic, "evaluation(v):-m." into H
  end for
end for
return H
```

Table 3. Evaluation of heuristic functions made of new concepts for Tic-tac-toe and Connect Four against a random player.

game	player	simulation size	win(%)/lose(%)/draw(%)
Tic-tac-toe	x	10000	94.60/3.77/1.63
	o	10000	56.91/38.05/5.04
Connect Four	white	10	90.6/9.4/0
	white	100	92.3/7.7/0
	red	10	61.8/38.2/0
	red	100	69.4/30.6/0

Table 4. Evaluation of heuristic functions made of new concepts for Tic-tac-toe and Connect Four against a 1-depth alpha-beta search player.

game	player	simulation size	win(%)/lose(%)/draw(%)
Tic-tac-toe	x	10000	89.64/10.36/0
	o	10000	39.16/56.87/3.97
Connect Four	white	10	74.5/25.5/0
	white	100	78.3/21.7/0
	red	10	29.5/70.5/0
	red	100	44.3/55.7/0

6 Automated Generation of Ternary Concepts from Binary Concepts

Generated concepts in Algorithm 2 are relationships between a cell and another cell, i.e., binary relationships. We tried to make ternary relationships as a conjunction of binary relationships. If two binary concepts are satisfied at the same time and a cell is shared in both concepts, the situation is a ternary relationship. Therefore, ternary relationships are made by unification of pairs of terms in each binary concept as written in Algorithm 4.

We generated ternary concepts from the binary concepts generated in Sect. 4. Typical generated concepts are as follows.

$$concept67(X_1, X_2, X_3) : -cell(X_1, X_2, X_3), cell(X_1, X_2 + 1, X_3), cell(X_1, X_2 + 2, X_3). \quad (7)$$

$$concept155(X_1, X_2, X_3) : -cell(X_1, X_2, X_3), cell(X_1 + 1, X_2 + 1, X_3), cell(X_1 - 1, X_2 - 1, X_3). \quad (8)$$

Equation 7 and 8 represent an idea of line in Tic-tac-toe. Important ternary concepts are successfully generated by this algorithm. The same algorithm has a possibility to make more complex concepts.

We also made heuristic functions made of ternary concepts for Connect Four by Algorithm 3. To reduce generation time, we input 10 simulations and ternary concepts generated as above to Algorithm 3 and made heuristic functions. Then,

Algorithm 4 makeTernaryConcepts(*concepts*)

```
P ⇐ makePairsOfConcept(concepts)
for all pair (c1, c2) in P do
  for all term t1 in body of c1 do
    for all term t2 in body of c2 do
      if t1 is able to unificate to t2 then
        t3 ⇐ unification result of t1 to t2
        b1 ⇐ make new body as like t3 appearing in c1 by unification
        b2 ⇐ make new body as like t3 appearing in c2 by unification
        if b1 is not equals to b2 then
          R ⇐ make new concept by conjunction of b1 and b2
        end if
      end if
    end for
  end for
end for
R ⇐ remove overlap from R
return R
```

Table 5. Evaluation of heuristic functions made of ternary concepts for Connect Four against a random player.

game	player	simulation size	win(%)/lose(%)/draw(%)
Connect Four	white	10	72.7/27.3/0
	red	10	71.7/28.3/0

Table 6. Evaluation of heuristic functions made of ternary concepts for Connect Four against a 1-depth alpha-beta search player.

game	player	simulation size	win(%)/lose(%)/draw(%)
Connect Four	white	10	39.7/60.3/0
	red	10	37.8/62.2/0

we evaluated it using 1,000 simulations against a random game player or a 1-depth alpha-beta search player. Results are not good as seen in Table 5 and 6. We think this is due to noises from less important concepts. In Algorithm 3, all input concepts are concerned, therefore, not only important concepts like Eq. 7 or 8 but also less important concepts affect evaluation values. This result suggests a limitation of Algorithm 3 and a need for better algorithm.

7 Discussion

In this study, we successfully extracted the essential concepts included in the game automatically. The proposed algorithm is applicable to general games. The algorithm learns new concepts without any supervised signals but from

experience in a certain environment. The generated concepts are fundamental features of the environment, and can be used to play some games with different rules.

A typical concept learned in this study is as follows.

$$concept(X_0, X_1, X_2) : -cell(X_0, X_1, X_2), cell(X_0, X_1 + 1, X_2). \quad (9)$$

This concept is applicable to all games in which players put a mark or piece on a two-dimensional board. Even more, through appropriate filters, it is possible to apply the concept to computer vision to understand semantics of a picture. If one makes a filter which converts a picture to Prolog like facts, this concept is applicable for recognizing semantics of series of squares with contents. In other words, concepts generated in this study are available to understand semantics of our living world, not only of artificial game worlds.

It is necessary for a human being to input supervised signals when an artificial intelligence learns concepts about our living world. For example, when we do Natural Language Processing to generate ontology, we need to input well written texts which are written by human beings [10]. It is impossible to learn ontology from automatically generated strings. However, in this study, we successfully generated concepts about our world without any supervised signals by human beings. This property is due to the special advantage of games, i.e., environments having concrete rules. In other words, human beings do not have to judge a meaning of record of random games because a simulator is able to judge it, i.e., who is the winner according to its game rules. This tremendously desirable advantage is available only in games, not in other fields. In our study, it is proven that we are able to generate essential concepts about our living world from games according to the advantage.

8 Conclusions

In our study, we automatically generated concepts which were applicable to understand the world of the games for General Game Playing. Obtained concepts were general and useful to understand several games. General Game Playing is a desirable research area for automatically learning concepts of our living world. To apply obtained concepts to more games and to generate more complex concepts are the problems which remain to be solved.

Acknowledgments. The authors would like to express their appreciation to Mr. Abdallah Saffidine for his contribution to the stimulating discussions, Prof. Erick Alphonse for his comments on Inductive Logic Programming, Prof. Hiroyuki Iida and Japan Advanced Institute of Science and Technology for funding and Dr. Kristian Spoerer for proof reading.

References

1. Björnsson, Y., Finnsson, H.: CADIAPLAYER: a simulation-based general game player. IEEE Trans. Comput. Intell. AI Games **1**, 4–15 (2009)

2. Clune, J.: Heuristic evaluation functions for general game playing. In: Proceedings of the National Conference on Artificial Intelligence, pp. 1134–1139 (2007)
3. Finnsson, H., Björnsson, Y.: Simulation-based approach to general game playing. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, pp. 259–264 (2008)
4. Inoue, K., Furukawa, K., Kobayashi.: Abducing rules with predicate invention. In: The 19th International Conference on Inductive Logic Programming (2009)
5. Kaneko, T., Yamaguchi, K., Kawai, S.: Automatic feature construction and optimization for general game player. In: Proceedings of Game Programming Workshop 2001, pp. 25–32 (2001)
6. Kirci, M., Sturtevant, N., Schaeffer, J.: A GGP feature learning algorithm. KI - Künstliche Intell. **25**, 35–42 (2011)
7. Love, N., Hinrichs, T., Haley, D., Schkufza, E., Genesereth, M.: General game playing: game description language specification. http://games.stanford.edu/readings/gdl_spec.pdf (2008)
8. Méhat, J., Cazenave, T.: A parallel general game player. KI-Künstliche Intell. **25**, 43–47 (2011)
9. Michulke, D., Thielscher, M.: Neural networks for state evaluation in general game playing. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS(LNAI), vol. 5782, pp. 95–110. Springer, Heidelberg (2009)
10. Mohamed, T.P., Hruschka, E.R.J., Mitchell, T.M.: Discovering relations between noun categories. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 1447–1455 (2011)
11. Schiffel, S.: http://www.general-game-playing.de/index.html (2008)
12. Skowronski, P., Björnsson, Y., Winands, H.M.M.: Automated discovery of search-control features. In: Proceedings of the Twelfth International Advances in Computer Games Conference, pp. 182–194 (2009)