A Novel Ontology for Computer Go Knowledge Management

Chang-Shing Lee, Wang Mei-Hui, Tzung-Pei Hong, Guillaume Chaslot, Jean-Baptiste Hoock, Arpad Rimmel, Olivier Teytaud, Yau-Hwang Kuo

A Novel Ontology for Computer Go Knowledge Management

Chang-Shing Lee, Mei-Hui Wang, Tzung-Pei Hong, Guillaume Chaslot
Jean-Baptiste Hoock, Arpad Rimmel, Olivier Teytaud, and Yau-Hwang Kuo

***Abstract*—In order to stimulate the development and research in computer Go, several Taiwanese Go players, including three professional Go players and four amateur Go players, were invited to play against the famous computer Go program, MoGo, in the *Taiwan Open 2009*. The MoGo program combines the online game values, offline values extracted from databases, and expert rules defined by Go expert that shows an excellent performance in the games. The results reveal that MoGo can reach the level of 3 Dan in Taiwan amateur Go environment. But there are still some drawbacks for MoGo that should be solved, for example, the weaknesses in semeai and how to flexibly practice the human knowledge through the embedded opening books. In this paper, a new game record ontology for computer Go knowledge management is proposed to solve the problems that MoGo is facing. It is hoped that the advances in intelligent agent and ontology model can provide much more knowledge to make a progress in computer Go and achieve as much as computer *chess* or *Chinese chess* in the future**.

I. Introduction

Computer Go has proved to be a troublesome game in that the best computer players still play at the level of a good novice [1]. Additionally, Go remains an excellent challenge for computer science research; however Monte Carlo methods have very recently shown significant promise, especially for small versions of the game such as 9×9 games. Werf *et al*. [2] presented a search-based approach for playing Go on small boards. Bouzy and Cazenave [3] presented an AI-oriented survey of computer Go. Computer Go has been developing for the past several years. In 1998, Martin Muller won despite 29 handicap stones against the computer Go "*Many Faces of Go*" [4]. In August 2008, the program "*MoGo*" (http://www.lri.fr/~teytaud/mogo.html) won with an advantage of "only" 9 handicap stones against top-level human players in 19×19 Go—Myung-Wan Kim, who won the 2008 US Open and was a Korean 8th Dan Pro (8P). Additionally, another program, "*CrazyStone*," won with handicaps of 8 and 7 stones against Kaori Aoba, a Japanese 4th Dan Pro (4P) in December 2008; MoGo got some new impressive results in the *Taiwan Open 2009*.

Knowledge is defined as any information that is relevant, actionable, and is based on a person's experience [5]. All knowledge workers share certain characteristic activities. However, when data is annotated, it's done against a framework or ontology [6]. Ontology has become a very powerful way of sharing the knowledge as well as representing the information and its semantics [7]. It is also a conceptualization of a real world domain into a human understandable, machine-readable format consisting of entities, attributes, relationships, and axioms [8]. Moreover, ontology mediation allows us to combine knowledge from the differing ontologies [6]. It has been successfully applied to certain domains. For example, Lee *et al.* proposed a fuzzy ontology to apply for news summarization [9] and they also proposed an ontology-based intelligent decision support agent for capability maturity model integration (CMMI) project monitoring and control [10]. Reformat and Ly [7] proposed an ontology-based approach to provide a rich environment for expressing different types of information including perceptions.

The game of Go is one of the last board games where the strongest humans are still able to easily win against computers. Researchers, however, have discovered new performing algorithms and computers are catching up really fast. The results of the games in the *Taiwan Open 2009* (http://go.nutn.edu.tw/2009/) have indicated that if computers continue to improve at this rate, one more human stronghold may fall in front of machines in less than 10 years. Additionally, the game results also have shown that if the computer can learn more knowledge and strategy from professional Go players through the constructed ontology, then the computer Go will approach the level of professional go player very fast. Hence, this study proposes a novel game record ontology for computer Go knowledge management. With the proposed computer Go knowledge management, the computer Go is expected to become much more intelligent than nowadays. The remainder of this study is organized as follows. Section II briefly describes the Monte Carlo Tree Search (MCTS)-based computer Go. The ontology model for computer Go knowledge representation is presented in Section III. Some case studies in the *Taiwan Open 2009* are presented in Section IV. Conclusions are finally given in Section V.

This work is supported by the National Science Council of Taiwan under the grant NSC97-2221-E-024-011-MY2 and the Pascal Network of Excellence.

Chang-Shing Lee and Mei-Hui Wang are with the Department of Computer Science and Information Engineering, National University of Tainan, Taiwan. (corresponding author to provide phone: 886-6-2606123 ext. 7709; fax: 886-6-2606125; e-mail: leecs@mail.nutn.edu.tw).

Tzung-Pei Hong is the Department of Computer Science and Information Engineering, National University of Kaohsiung, Taiwan.

Guillaume Chaslot is with Department of Computer Science, University of Maastricht, The Netherlands.

Jean-Baptiste Hoock, Arpad Rimmel, and Olivier Teytaud are with the TAO, Lri, Univ. Paris-Sud, Inria Saclay-IDF, UMR Cnrs 8623, Bât 490, Université Paris-Sud F91405 Orsay, France.

Yau-Hwang Kuo is the CREDIT Research Center, National Cheng Kung University, Taiwan.

II. MCTS-BASED COMPUTER GO

In this section, the Monte Carlo Tree Search (MCTS)-based computer Go is briefly described. Minimax and alpha-beta searches are the most common techniques used in computer games. However, in Go, even after pruning by patterns or rules, these approaches are clearly outperformed by MCTS – all recent tournaments have been won by MCTS approaches with usually all the MCTS programs ranked better than non-MCTS programs. Interestingly, some significant improvements in *chess* have been achieved with "forced moves" [14]. Forced moves are moves that are almost mandatory for a player. A combination is a sequence of such moves when an opponent can only play forced moves. Via forced moves, one can increase analysis depth. In *chess*, according to some scholars, this technique is efficient and programs with forced moves can announce checkmate and victory far in advance. In computer Go, forced moves are a difficult concept. Cazenave [15] produced notable results using forced moves; however, forced moves are only forced in the sense that they are necessary for some particular goals, not for a complete victory. In Go, strong players never absolutely try to keep some groups alive. If the opponent can use many stones to kill one group, letting the opponent reach his target is fine, as during that time, influence is extended to another part of the goban, increasing the amount of territory won. Via the MCTS approach, very deep sequences can be produced, allowing computer Go to reach the same complexity as the computer *chess* despite a lack of forced moves.

A. Monte Carlo Evaluation

Brugmann proposed an original evaluation function based on Monte Carlo exploration [11]: the estimate is the probability of winning estimated by a random player. The All-Moves-As-First (AMAF) value of a move improves Monte Carlo evaluation by using statistics on permutations of games: when one simulation is performed, several permutations of this permutation are considered as well. These simulations provide statistics for several moves, other than the first move of the initial simulation. In the MCTS setting, AMAF values are usually called Rapid Action Value Estimation (RAVE) values [17]. The Monte Carlo evaluation can be used as an evaluation function in an alpha-beta engine. However, a recent and considerable improvement is the incremental construction of a tree on top of the Monte Carlo evaluation function. Random simulations are still performed from the initial node, but they are adaptively modified in the part which is in the tree. Outside the tree, the simulation uses the default random policy, whereas in the tree, simulations choose the move that maximizes a score combining two criteria: (1) Exploration criterion: moves that have not been simulated often should be more intensively simulated; and (2) Exploitation criterion: moves that lead to high probabilities of winning should be simulated more often. In this incremental construction, the tree is expanded in the direction of interesting moves as estimated by the Monte-Carlo evaluation. A well-known implementation of this idea is Upper-Confidence Tree (UCT) [13]. The overall MCTS algorithm [12] is presented in Algorithm 1. Moreover, this MCTS is parallel: in Fig. 1, each computation node builds his own tree in memory, and 3 times per second all nodes "share" their trees with other nodes by averaging statistics, as if the algorithm was a simple Monte-Carlo algorithm.

Algorithm 1: Pseudo-code of a MCTS algorithm applied to a two-player game (typically Go or *chess*). *T* is a tree of positions, with each node equipped with statistics (number of wins and losses in simulations starting at this node).

Initialize *T* to a single node, representing the current state.
while Time left > 0 do
- **Simulate one game until a position** *L* is out of *T*.
- **Simulate one game from position L until the game is over** (thanks to the random player, see Algorithm 2).
- **Growth of the tree:** add *L* in *T*.
- **Update statistics in the entire tree:** In UCT, we have to store in each node how many simulations and how many winning simulations have been performed from this node.

End while
Return the move which has been simulated most often from the root

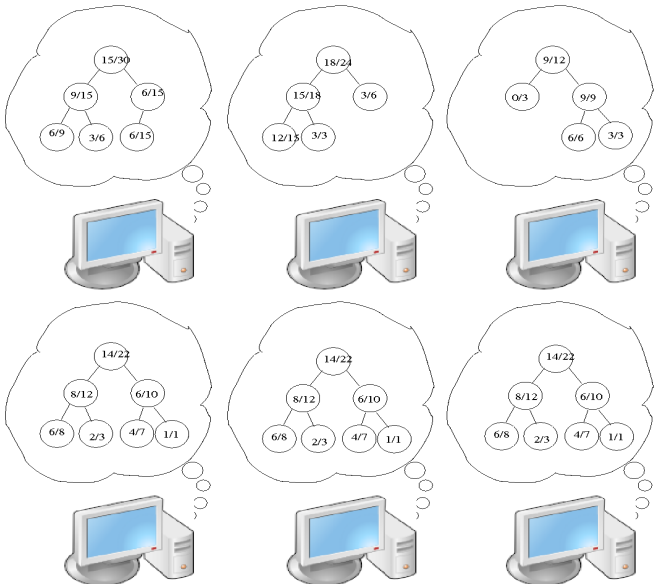


Fig. 1. One step of sharing in the MPI-parallelization of MoGo.

B. Upper Confidence Bound and Heuristic values

When Upper Confidence Bound (UCB) is used, MCTS is called an UCT [13]. Adding RAVE values has been done in [16]. Adding a term based on the patterns [12], [18] and rules to the RAVE values was proposed by [16]. Typically, the value of a pattern is proportional to the frequency of a move *m* in a pattern *p* according to a database of strong games, plus a coefficient tuned empirically for moves matching some expert rules. The rules implemented in MoGo can be found in [16]. The website (http://senseis.xmp.net) provides the Go definitions required for implementing these rules.

C. Mechanism of MoGo

The Monte Carlo player used in MoGo is defined in Algorithm 2. An *atari* occurs when a string representing a group of stones can be captured in one move. For the bandit part of MoGo, MoGo combines RAVE values [17], offline values extracted from databases [12], [16], [18] and expert rules [16]; the algorithm is more completely described in [16].

Algorithm 2: Algorithm for choosing a move in Monte Carlo simulations for the game of Go. Some details have been omitted for clarity; see [16] for

details (in particular the "Nakade" modification).

```
if the last move is an atari then
    Save the stones which are in atari.
else
    Randomly pick up 6 empty locations on the goban.
    if one of them is empty and the 8 surrounding locations are empty then
        Play in this location.
    else
        if there is an empty location among the 8 locations around the last
          move which matches a pattern then
            Play randomly and uniformly in one of these locations.
        else
            if there is a move which captures stones then
                Capture stones.
            else
                if there is a legal move which does not fill a friendly eye then
                    Play randomly such a legal move.
                else
                    Return pass.
                end if
            end if
        end if
    end if
end if
```

III. Ontology-based Knowledge Management

This section describes the Knowledge Management (KM) process for computer Go and the ontology model for computer Go knowledge representation.

A. Knowledge Management Process for computer Go

By applying pattern knowledge and accumulated positional knowledge, a Go player is able to exclude bad moves from consideration and quickly recognize and stop exploration in hopeless positions during planning. If a computer Go also uses such domain knowledge, then it should be much efficient. Fig. 2 shows the process of constructing Go knowledge management. As shown in Fig. 2, there are five processes defined, including a knowledge creation, a knowledge integration, a knowledge storage, a knowledge dissemination, and a knowledge reuse. First, the key knowledge creation transaction is between a Go player and an ontology engineer. It also can be constructed by the computer Go program. The ontology engineering discovers and captures the Go domain knowledge from the Go player through the communication with each other to create the knowledge. Second, the ontology engineer integrates the found knowledge to construct a Go ontology. Third, the Go domain knowledge is stored into the Go knowledge management system. Fourth, the knowledge is then available to computer Go programmer and other Go players through the knowledge reuse and knowledge dissemination processes. Finally, the computer Go programmer develops a novel game record ontology for computer Go knowledge management to play with Go players.

B. Ontology Model

Fig. 3 shows the structure of the domain ontology, including a *domain layer*, a *category layer*, and a *concept layer* [9], [10]. The *domain layer* represents the domain name of an ontology, and consists of various categories defined by domain experts. The *category layer* defines several categories, labeled as "category *1*, category *2*, …, and category *k*." Each concept in the *concept layer* contains a concept name C_i, an attribute set $\{A_{C_i1},\cdots,A_{C_iq_i}\}$, and an operation set $\{O_{C_i1},\cdots,O_{C_iq_i}\}$ for an application domain. Three inter-conceptual relations have been utilized in the domain ontology, namely the *generalization*, the *aggregation*, and the *association*. A *generation* relation between a domain and its corresponding category is called an "*is-kind-of*" relationship. The relationship between each category and its corresponding concepts is the *aggregation*, which denotes the "*is-part-of*" relationship. The *association* indicates a semantic relationship between concepts in the *concept layer*.

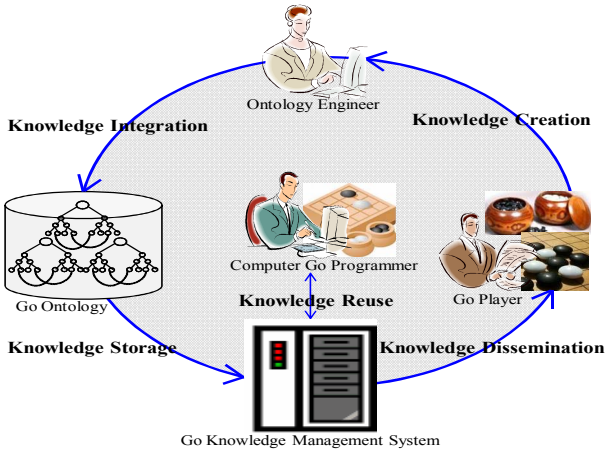


Fig. 2. Knowledge management process for computer Go.

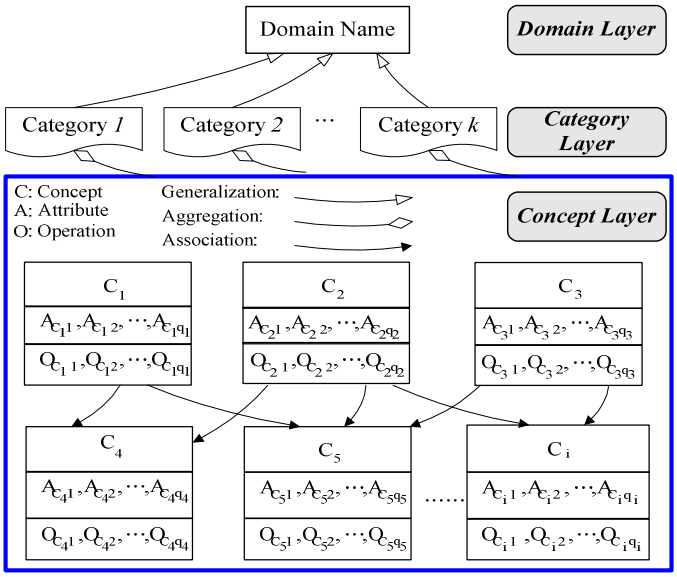
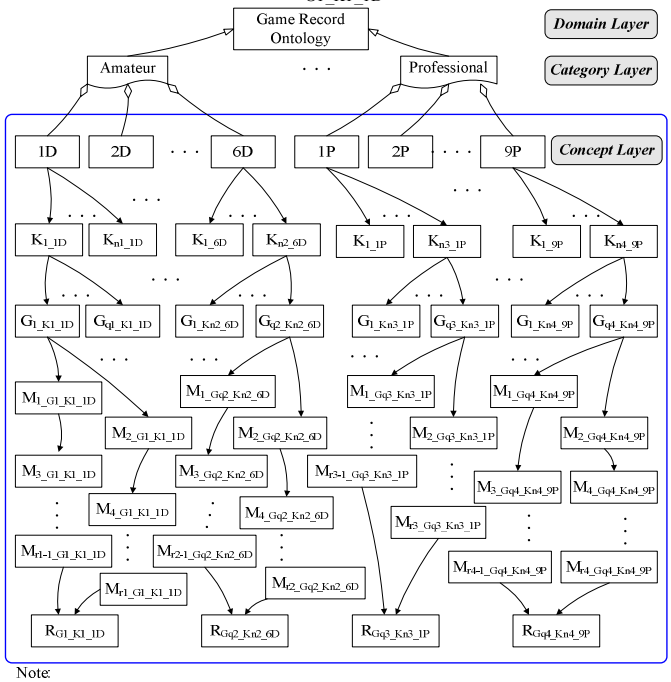


Fig. 3. Structure of the domain ontology.

Based on the structure of the domain ontology, we apply it to define the game record ontology model for computer Go knowledge management, shown in Fig. 4. The domain name of this ontology is "*Game Record Ontology*." The categories in the *category layer* include "*Amateur*" and "*Professional*." Several concepts are located in the *concept layer*. For example, concepts "*1D*," "*2D*," and "*6D*" represent the level of dan on the amateur scale for a Go player: 1 dan, 2 dan, and 6 dan, respectively. On the other hand, concepts "*1P*," "*2P*," and "*9P*" represent the level of dan on the professional scale

for a Go player: 1 dan, 2 dan, and 9 dan, respectively. The built Go game record ontology stores each move of each game played by the Go players against MoGo. For instance, assume there are *n1* 1D Go players (K_{1_1D},…, and K_{n1_1D}). Each 1D Go player may play total *n1* games stored in the game record ontology. So, Fig. 4 indicates that each game of each Go player can be modeled by the game record ontology. For example, the games that the first 1D Go player (K_{1_1D}) once played are represented as $G_{1\_K1\_1D}$, …, and $G_{q1\_K1\_1D}$. For the game $G_{1\_K1\_1D}$, the records of game are represented as moves *1*, *2*, …, and *r1*, that is $M_{1\_G1\_K1\_1D}$, …, and $M_{r1\_G1\_K1\_1D}$, and its result is denoted as $R_{G1\_K1\_1D}$.



Note:
(1) D: Dan (2) P: Professional (3) K: Kishi (Go player) (4) G: Game (5) R: Result
(6) n1, n2, n3, and n4: Total number of Go players with 1D, 2D, 1P, and 9P, respectively.
(7) q1, q2, q3, and q4: Total number of recorded games that Go players (1_1D), (n2_6D), (n3_1P), and (n4_9P) had played, respectively.
(8) r1: Total number of moves for the Game_1 that Go player (1_1D) had played.
(9) r2: Total number of moves for the Game_q2 that Go player (n2_6D) had played.
(10) r3: Total number of moves for the Game_q3 that Go player (n3_1P) had played.
(11) r4: Total number of moves for the Game_q4 that Go player (n4_9P) had played.

Fig. 4. Ontology for the computer Go knowledge management.

IV. Case Study in Taiwan Open 2009

A. Introduction to Taiwan Open 2009

Due to the development of the Computational Intelligence, computer Go has made considerable progress for the past 10 years. Programs are currently competitive at the professional level in 9×9 Go. After the 2008 *World 9×9 Computer Go Championship* (http://go.nutn.edu.tw/) in Taiwan, the MoGo teams enhanced the opening book and the weakness in corners to further improve the performance of MoGo in 9×9 and 19×19 games. Therefore, National University of Tainan (NUTN), National Cheng Kung University (NCKU), Taiwanese Association for Artificial Intelligence (TAAI), and Institute for Information Industry (III) hosted the "*2009 Invited Games for MoGo vs. Taiwan Professional Go Players* (*Taiwan Open 2009*)" on February 10-13, 2009. The computer center of NUTN, the center for research of e-life digital technology (CREDIT) of NCKU, and the Innovative Digitech-Enabled Applications & Services Institute (IDEAS) of III are responsible of organizing the *Taiwan Open 2009*. Three professional Go players, including Jun-Xun Zhou (9P), Shih Chin (2P), and Li-Chen Chien (1P), and several amateur Go players, including Cheng-Wen Dong (5D), Shi-Jim Yen (6D), Shang-Rong Tsai (6D), and Biing-Shiun Luoh (6D), were invited by the organizers to compete with MoGo in the *Taiwan Open 2009*. Additionally, several international experts on computational intelligence and games were also invited to serve as the International Advisory Board Members.

On the first two days of the *Taiwan Open 2009*, MoGo made two new world records by winning a 19×19 game with 7 handicap stones against a 9P professional Go player—Jun-Xun Zhou and a 19×19 game with 6 handicap stones against a 1P professional Go player—Li-Chen Chien. The results of these two games will stay in history. On the third day of this event, MoGo beat 2P professional Go player Shin Chin in the first 9×9 game, that is game 3 which is shown in Fig. 5.

B. Go Knowledge Management for Taiwan Open 2009

In this subsection, eight 9×9 games of *Taiwan Open 2009* are discussed. For more game results, Table I lists the profiles of partial Go players who competed against MoGo. The game adopted the Chinese rule and komi was 7.5. During the tournament, MoGo ran on a supercomputer Huygens, provided by Dutch research organizations, Stichting Academisch Rekencentrum Amsterdam (SARA, The Netherlands) and National Computer Facilities (NCF, The Netherlands). The MoGo program was allowed to use at most 20 nodes of the supercomputer, *i.e.*, 640 cores at 4.7GHz. The games were played over the Kiseido Go Server (KGS) platform (http://www.gokgs.com/). The results of the discussed 9×9 games are listed in Table II. Figs. 6 and 7 show the recorded moves of games 5 and 6, respectively.

TABLE I. Profiles of Partial Go Players Competing with MoGo.

No	Name	Age	Sex	Dan grade
1	Jun-Xun Zhou	28	Male	9P professional
2	Shih Chin	22	Male	2P Professional
3	Shi-Jim Yen	41	Male	6D Amateur

TABLE II. Results of Discussed 9×9 Games.

No	Date	Time per side (min)	White	Black	Result
1	02/10/2009	30	**MoGo**	Yen	W+Resign
2	02/10/2009	50	Yen	**MoGo**	B+1.5
3	02/12/2009	50	**MoGo**	Chin	W+0.5
4	02/12/2009	50	**Chin**	MoGo	W+Resign
5	02/10/2009	50	**Zhou**	MoGo	W+Resign
6	02/13/2009	50	**Zhou**	MoGo	W+Resign
7	02/10/2009	50	MoGo	**Zhou**	B+Resign
8	02/13/2009	50	MoGo	**Zhou**	B+Resign

Games 5 (Fig. 6) and 6(Fig. 7) reveal that MoGo played the same sequences and positions against Zhou for moves 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, and 35. Also, games 7 (Fig. 8) and 8 (Fig. 9) were almost identical, especially from move 1 to move 19. The human Go player, 9P Jun-Xun Zhou, just reproduced the same games, which were very constrainted by the opening, on the fourth game day

(February 13, 2009). On game 6 against Zhou, MoGo played the same moves as the ones on game 5 against Zhou. But from a human's viewpoint, MoGo should have played different moves on game 6 to have a chance to defeat Zhou. It's a very simple inference to get to know that it is definite to change the strategy in order to have a chance to win. However, the current version of MoGo is not smart enough to adopt different strategies on game 6. Hence, if MoGo can learn more about knowledge from the experiences of playing with humans, it may be able to solve this problem—MoGo played the same sequences and positions on games 5 and 6, but MoGo lost the game 5. Therefore, a novel game record ontology for computer Go knowledge management is proposed in this study. After the *Taiwan Open 2009*, we have discovered and learned more about the weaknesses of MoGo. Possibly, after giving MoGo a specific and designed Go ontology, we can then take steps to solve the above-mentioned problem that MoGo is facing.

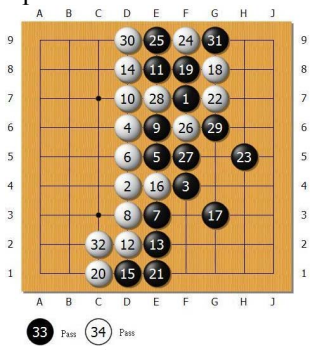


Fig. 5. Game 3.

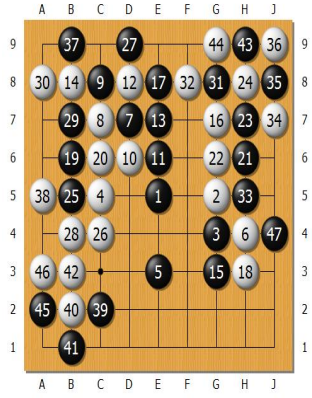


Fig. 6. Game 5.

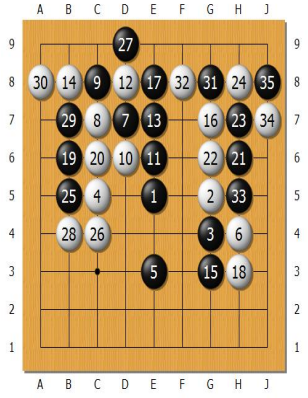


Fig. 7. Game 6.

Consequently, based on part of the selected games results, we apply the ontology model shown in Fig. 4 to the real cases of the *Taiwan Open 2009*. Fig. 10 shows the *Taiwan Open 2009* Go knowledge management ontology. First, the domain name is "Go Game Record Ontology (Taiwan Open 2009)." There are one 5D amateur Go player concept "Cheng-Wen Dong" and three 6D amateur Go players' concepts, including "Shi-Jim Yen," "Shang-Rong Tsai," and "Biing-Shiun Luoh." Concepts "Li-Chen Chien," "Shih Chin," and "Jun-Xun Zhou," are 1P, 2P, and 9P professional Go players, respectively. Take concept "Game 1" for an example. Its attributes can be "Time of game," "Name of Go player," "Event of game," and "Moves of positions and sequences." The ontology is then constructed according to the "When, Who, What, and How" practices.

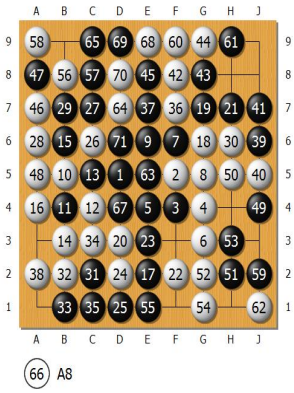


Fig. 8. Game 7.

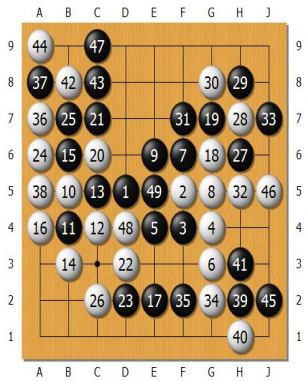


Fig. 9. Game 8.

For using the ontology, we proceed as follows. MoGo currently chooses, in the opening, the move which provided the best score. The score is the number of wins in MoGo's huge set of (offline, built on Grid5000 clusters) high-quality games. With the ontology, after each game: (1) The stability of a move in the opening is defined as the ratio between its score and the score of the move with second best score. By definition, it is at least 1. (2) The move with lower stability is then forbidden for future games in the same sequence of games. Its strategy is applied, then (1) On game 6, shown in Fig. 7, the move E3 would have been replaced by move C6. It is beyond our expertise to ensure that C6 is a good idea. On the other hand, the fact that E3 is a bad idea was pointed out by Pierre Audouard, well-known expert of 9×9 Go in [19]. (2) On game 8, shown in Fig. 9, the second move would be E5 instead of F5, after D5. It is difficult to say if this move is a good move or not. It is at least very aggressive and highly complicated.

V. Conclusion

In this study, a novel game record ontology for computer Go knowledge management is proposed based on the game results in the *Taiwan Open 2009*. During the competition, MoGo made history by winning two 19×19 games with 7 and 6 handicap stones against a 9P and a 1P professional Go players, respectively. In addition, MoGo also secondly defeated a professional Go player in a 9×9 game. With these exciting records, however, MoGo still have something to be solved such as discovering new performing algorithms to improve the semeai, the fights between two groups, and introducing the domain experts' knowledge to opening books. It is hoped that one more human stronghold will fall in front of machines in less than 10 years.

Acknowledgment

The authors would like to thank the support of Prof. H.-S. Huang (President of National University of Tainan, Taiwan), Institute for Information Industry (III), and the Computer Center of National University of Tainan, especially Y.-L. Wang and M.-C. Wang.

REFERENCES

[1] S. M. Lucas and G. Kendall, Evolutionary computation and games, *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 10-18, 2006.

[2] E. van der Werf, J. van den Herik, and J. Uiterwijk, "Solving Go on small boards," *International Computer Games Association (ICGA) Journal*, vol. 26, no. 2, pp. 92-107, 2003.

[3] B. Bouzy and T. Cazenave, "Computer Go: an AI-oriented survey," *Artificial Intelligence Journal*, vol. 132, no. 1, pp. 39-103, 2001.

[4] M. Muller, "Computer Go," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 145-179, 2002.

[5] B. Ghosh and J. E. Scott, "Comparing knowledge management in health-care and technical support organizations," *IEEE Transactions on Information Technology in Biomedicine*, vol. 9, no. 2, pp. 162-168, 2005.

[6] P. Warren, "Knowledge management and the semantic web: from scenario to technology," *IEEE Intelligent Systems*, vol. 21, no. 1, pp. 53-59, 2006.

[7] M. Reformat and C. Ly, "Ontological approach to development of computing with words based systems," *International Journal of Approximate Reasoning*, vol. 50, no. 1, pp. 72-91, 2009.

[8] Q. T. Tho, S. C. Hui, A. C. M. Fong, and T. H. Cao, "Automatic fuzzy ontology generation for semantic web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 6, pp. 842-856, 2006.

[9] C. S. Lee, Z. W. Jian, and L. K. Huang, "A fuzzy ontology and its application to news summarization," *IEEE Transactions on Systems, Man and Cybernetics Part B*, vol. 35, no. 5, pp. 859-880, 2005.

[10] C. S. Lee, M. H. Wang, and J. J. Chen, "Ontology-based intelligent decision support agent for CMMI project monitoring and control," *International Journal of Approximate Reasoning*, vol. 48, no. 1, pp. 62-76, 2008.

[11] B. Brugmann, Monte Carlo Go, 1993. Online at http://www.ideanest.com/vegos/MonteCarloGo.pdf.

[12] G. Chaslot, M. Winands, J. Uiterwijk, H. J. van den Herik, and B. Bouzy, "Progressive strategies for Monte-Carlo tree search," in Proceedings of the 10th Joint Conference on Information Sciences (JCIS 2007), Salt Lake City, USA, 2007, pp. 655–661.

[13] L. Kocsis and C. Szepesvari, "Bandit-based Monte-Carlo planning," Lecture Notes in Computer Science, vol. 4212, pp. 282–293, 2006.

[14] T. Anantharaman, M. S. Campbell, and F. Hsu, "Singular extensions: adding selectivity to brute-force searching," *Artificial Intelligence*, vol. 43, no. 1, pp. 99-109, 1990.

[15] T. Cazenave, "Metaprogramming forced moves," in Proceedings of the 13th European Conference on Artificial Intelligence (ECAI '98), Brighton, UK, 1998, pp. 645-649.

[16] L. Chatriot, S. Gelly, J. B. Hoock, J. Perez, A. Rimmel, and O. Teytaud, "Including expert knowledge in bandit-based Monte-Carlo planning with application to Computer Go," The 10th European Workshop on Reinforcement Learning (EWRL08), Lille, France, 2008.

[17] S. Gelly and S. Silver, "Combining online and offline knowledge in UCT," in Proceedings of the 24th International Conference on Machine Learning (ICML 2007), New York, USA, 2007, pp. 273-280.

[18] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search," in Proceedings of the 5th International Conference on Computers and Games (CG 2006), Turin, Italy, 2006, pp. 72-83.

[19] P. Audouard, G. Chaslot, J. B. Hoock, J. Perez, A. Rimmel, O. Teytaud, "Grid coevolution for adaptive simulations; application to the building of opening books in the game of Go," 1st European Workshop on bio-inspired Algorithms in Games (EvoGAMES), Tubingen, Germany, 2009.

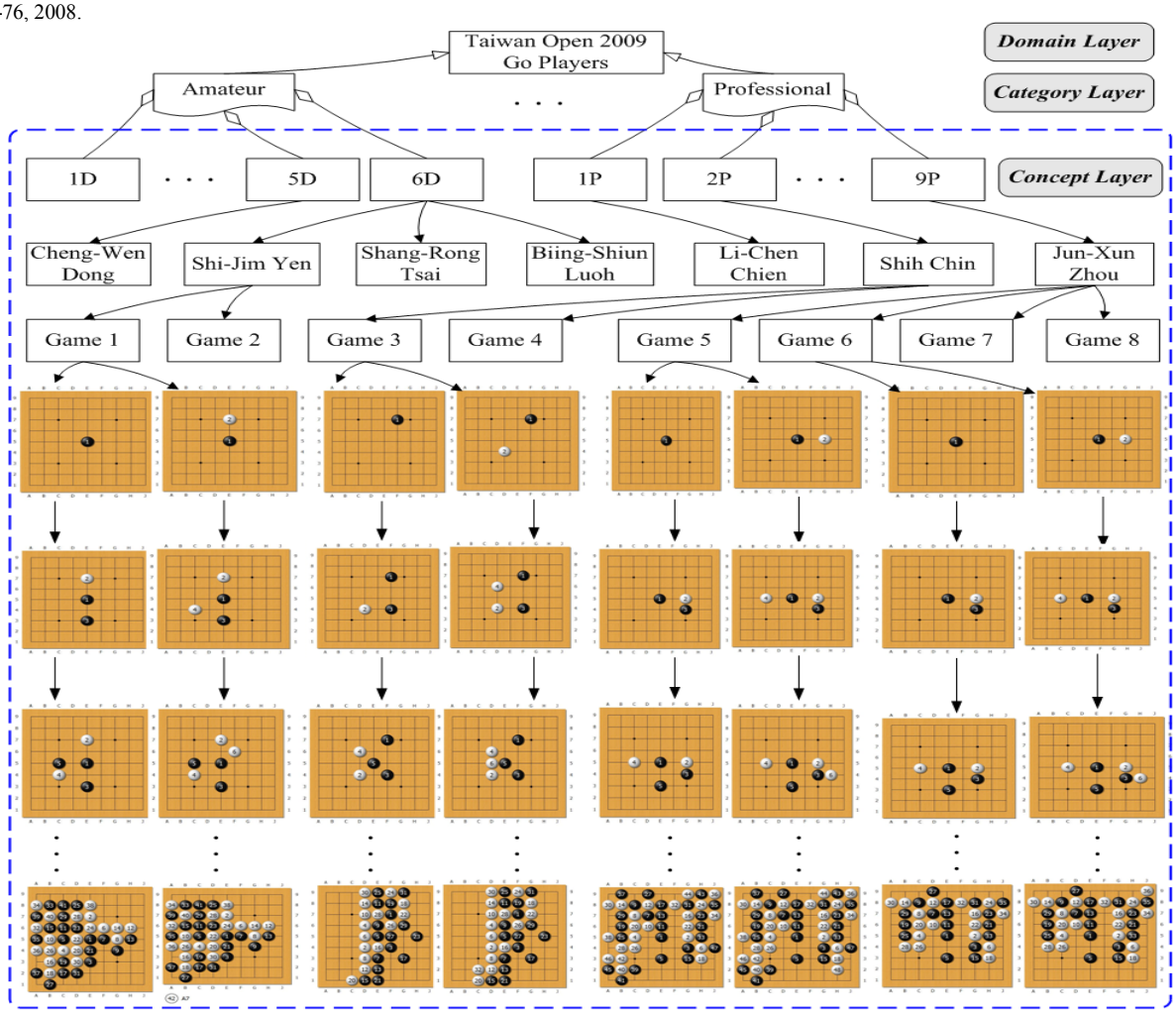


Fig. 10. Taiwan Open 2009 Go knowledge management ontology.