Integration of Different Reasoning Modes in a Go Playing and Learning System

Tristan Cazenave

LIP6, Université Pierre et Marie Curie
4, place Jussieu
75252 PARIS CEDEX 05, FRANCE
Tristan.Cazenave@lip6.fr

Abstract

Integrating multiple reasoning mode is useful in complex domains like the game of Go. Go players use various forms of reasoning during a game. Reasoning at the tactical level is completely different from reasoning at the strategic level. Choosing a plan requires a different form of reasoning than knowing how to execute a plan. This paper gives examples of the integration of these reasoning modes into a single system. Rule-based reasoning, Constraint-based reasoning and Case-based reasoning are used in this hierarchical order. Constraint-based reasoning uses the results of Rule-based reasoning, and Case-based reasoning uses the results of Constraint-based reasoning and Rule-based reasoning.

Introduction

Integrating multiple reasoning modes is useful in complex domains like the game of Go. Go players use various types of reasoning during a game. Reasoning at the tactical level is completely different from reasoning at the strategic level. Choosing a plan requires a different type of reasoning than knowing how to execute a plan. This paper gives examples of the integration of these reasoning modes into a single system. This work has some similarities with the work by Epstein and Gelfand [Epstein and Gelfand 1996].

The first section describes computer Go. The second section shows how rules are used and learned in our system at the tactical level. The third section describes some constraints to choose a plan at the strategic level. The fourth section provide a way to use Case-Based Reasoning to choose the more appropriate move to follow a plan. The last section gives the results of our computer Go system.

Computer Go

The game of Go

Go was developed three to four millennia ago in China; it is the oldest and one of the most popular board game in the world. Like chess, it is a deterministic, perfect information, zero-sum game of strategy between two players. In spite of the simplicity of its rules, playing the game of Go is a very complex task. Robson [Robson 1983] proved that Go generalized to NxN boards is *exponential in time*. More concretely, Van den Herik [Van den Herik 1991] and Allis [Allis 1994] use complexity measures of different games to compare them. They define the *whole game tree complexity* A. Considering the average length of actual games L and average branching factor B, we have $A = B^L$. The *state-space complexity* of a game is defined as the number of legal game positions reachable from the initial position of the game. In Go, L≈150 and B≈250 hence the game tree complexity $A \approx 10^{360}$. Go state space complexity, bounded by $3^{361} \approx 10^{172}$, and game tree complexity are far larger than those of any other perfect-information game. Moreover, a position is takes time to evaluate, on the contrary of chess where positions can be evaluated very fast. This makes Go very difficult to program. Computer Go has been recognized as a challenge for Artificial Intelligence [Selman 1996].

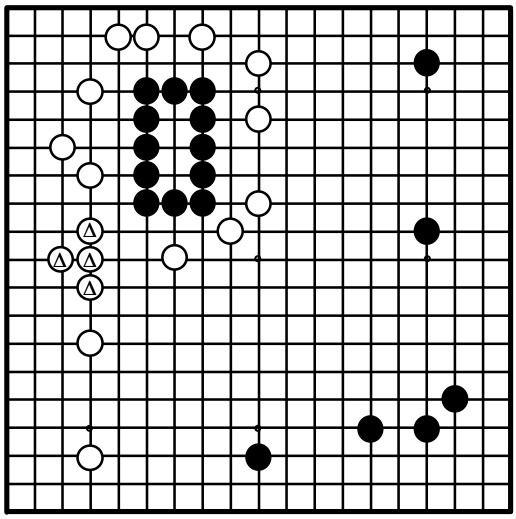

Figure 1

The board is made of 19 vertical lines and 19 horizontal

lines which cut themselves into 361 *intersections*. At the beginning the board is empty. Each player (Black or White) moves alternatively in adding one *stone* on an empty intersection. Two adjacent stones of the same color are *connected* and they are part of the same *string*. For example, the white stones of Figure 1 marked with Δ are connected and are part of the same string. Empty adjacent intersections of a string are the *liberties* of the string. The string of four marked white stones of Figure 1 has eight liberties. When a move fills the last liberty of a string, this string is removed from the board. The repetitions of positions are forbidden. According to the possibility of being captured or not, the strings may be *dead* or *alive*. A player *controls* an intersection either when he has an alive stone on it, either when the intersection is empty but adjacent to alive stones. The aim of the game is to control more intersections than the opponent. The game ends when the two players pass.

In spite of the simplicity of the rules, a Go player uses a lot of concepts to understand a position and to play a move. This paragraph briefly shows some intuitive definitions of these concepts. At the lower level, a player looks at the *safety* of the strings in performing look-ahead. When a string has enough liberties, the string is said to be safe. A player also checks if an intersection is controlled by one player or not. An *eye* is a small enclosed area, Figure 2 gives an example of an eye on intersection A. In this figure, B is one of the four *diagonal* intersections of A. When searching to make an eye, it is important to control diagonal intersections.

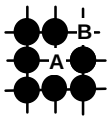

Figure 2

A virtual *connection* is a spatial configuration that enables to connect strings whatever the opponent plays. Figure 3 gives an example of a 'Bamboo join'. If the white player plays at A, black plays at B and connects its stones. If white plays at B, then black at A connects. The four stones are virtually connected.

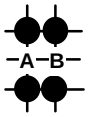

Figure 3

Using these tactical results, a Go player starts its strategic reasoning with the use of *groups*. A group is a complex concept for human players. It may be either a set of intersections that are virtually connected, either a set of intersections that gather the same properties. A group has a *status*. A status is dead or alive and it is derived from other intuitive concepts like *influence*, *fight*, *circling*, *life-base*. The reader does not need explanations of these concepts to understand the following sections.

Different levels in a Go program

As it is impossible to search the entire tree for the game of Go, the best Go playing programs rely on a knowledge intensive approach. They are generally divided in two modules:

- A tactical module that develops narrow and deep search trees. Each tree is related to the achievement of a goal of the game of Go.
- A strategic module that chooses the move to play according to the results of the tactical module.

Strategic reasoning is concerned with groups of stones. A group of stones is a set of stones of the same color, each stone can be connected to each other.

Different types of reasoning are required in these modules. The tactical module uses rules to decide what moves to try in the search trees. The strategic module has to choose a plan and to execute it. A good way to choose a plan is to use constraints on the groups calculated by the tactical module. Choosing a move that executes the plan can be done by comparing the present situation with cases previously encountered in games.

Rule based reasoning

Rule based reasoning is used in the tactical module of the system. The rules are used to decide what moves to try in a search tree. These rules are automatically created by an Explanation Based Learning system named Introspect [Cazenave 1996]. Introspect is an introspective learning system [Cox 1996], such systems have been formalized in [Mitchell 1986] [Laird 1986] [Dejong 1986] and they have received attention more recently in [Ram & Leake 1995]. The rules learned by Introspect enable to consider only between 1 and 5 moves out of the 250 possible moves on a board. They exponentially decrease the size and time of the brute force search tree. This enables our Go program to look 60 moves ahead in some tactical positions. The formalism used to represent these rules is first order predicate logic. The rules are learned by Introspect, only given the rules of the game in predicate logic.

Example of a (simple) learned rule used to find connections between strings of stones :

Connect (S1 S2 I friend) :- Color (S1 friend), Color (S2 friend), Liberty (I S1), Liberty (I S2), Move (I friend).

This rule tells that if an intersection I is a liberty of strings S1 and S2 that are friend strings, playing a black stone at I enables to achieve the goal Connect between the two strings.

The target concepts of the Explanation Based Learning module are the tactical subgoals of the game of Go :

Remove a string, Make a string alive, Connect two strings, Disconnect two strings, Make an eye and Remove an eye. Each of these target concepts is defined using rules in predicate logic. For example the target concept for the tactical goal RemoveString is defined using this rule:

RemoveString (S I friend) :- Color (S enemy), Move (I enemy), NumberOfLibertiesBeforeMove (S 1), Liberty (I S), LegalMove (I enemy).

Thousands of rules are created by using the rules of the game to specialize the tactical goals.

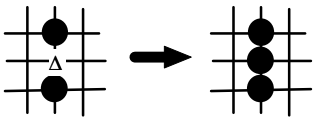

Figure 4

The example learned rule is learned by explaining why the move marked Δ in the Figure 4 connects the two black strings. The initial target concept defining the Connect goal is:

ConnectedAfterMove (S1 S2) :- Color (S1 C), Color (S2 C), ElementOfAfterMove (I S1), ElementOfAfterMove (I S2).

The rules used to specialize the target concept in this example are:

ElementOfAfterMove (I S) :- Liberty (I S), Color (S C), Move (I C).

Connect (S1 S2 I friend) :- Move (I friend), ConnectedAfterMove (S1 S2).

Note that there are different predicates to describe the board after the move and the board before the move. This is to prevent side effects to happen, and to avoid incomplete explanations.

At each node of the proof tree, learned rules are used to select useful moves to try. Knowledge about the moves to try in the search trees are represented using predicate logic rules because these rules represent theorems about the moves useful or necessary to try and the moves not to try.

Constraint based reasoning

Constraints can be used in games to choose a plan [Nigro & Cazenave 1996]. They are used in the Go program to choose plans at the strategic level. For example :

Save (G2) :- Neighbor (G2 G1), Territory (G2) < Territory (G1), Territory (G2) + PotentialTerritory (G2) / 2 < 9, NumberOfEyes (G2) < 2

This constraint tells that it is interesting to save group G2 if it has a neighboring group G1, and G2 has less territory than G1, and if the sum of the territory of G2 and of the potential territory of G2 divided by two is less than 9, and if the number of eyes of G2 is less than 2.

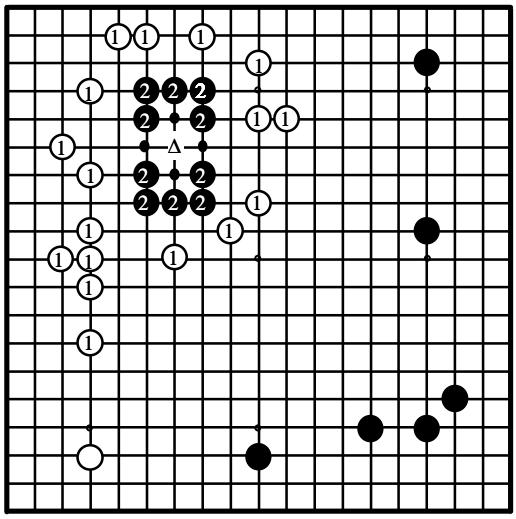

Figure 5

The constraints are about groups. Groups are constructed using the results of the tactical module. For example, each point of territory is the result of a proof tree. The proof tree is developed for proving that a string S that belongs to group G can be connected to the intersection I if Friend plays first. If no opponent string can connect to the same intersection I, then this intersection is a territory of the group G that contains the string S.

In the Figure 5, we give a board where the example rule with the constraints applies. The group G2 is marked with 2, and the groups G1 with 1. The points of territory of the group G2 are marked with little black points. There are more than forty points of territory for the white group, mainly on the upper left side of the board. G2 has five points of territory and only one eye. The constraints of the example rule are verified, so the goal Save (G2) is active.

Using constraints is the obvious way to describe that groups are unsafe under some critical threshold of the numbers representing their properties. Groups have a lot of numerical properties that are related to their safety, so constraints enable to express easily knowledge about the safety of groups.

Case-based Reasoning

Once a plan has been chosen, the program has to choose how to apply the plan. This is the next part of the strategic level. Tactical goals and strategic plans are calculated on a set of typical positions. It provides a set of cases with

associated moves. The moves can be the moves to play or the moves not to play.

There are different degrees of similarity between the predicates describing groups. For example, the conditions 'Territory (G) = 9' and 'Territory (G) = 10' are very similar. Whereas the conditions 'NumberOfEyes (G) = 1' and 'NumberOfEyes (G) = 2' are very dissimilar.

Each move is associated to the goal it achieves. The moves that achieve the plans chosen by the constraints are selected. Each plan has a value, a move can achieve multiple plans. The value of each move is the sum of the value of the plans the move achieves. The move with the highest value is played.

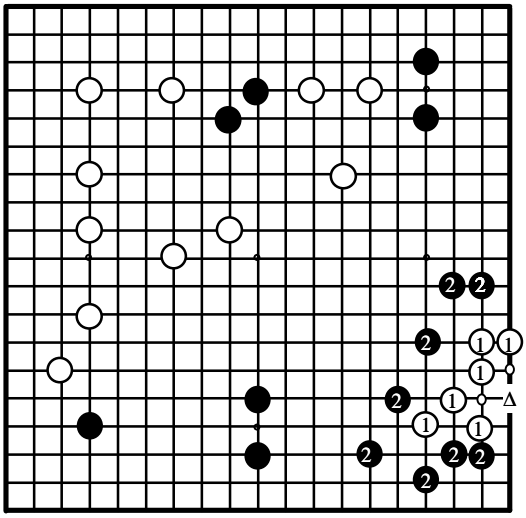

Figure 6

In the Figure 6, the situation of the groups marked with 1 is very similar to the situation of the group marked with 2 in Figure 5. Group 1 in Figure 6 has 3 points of territory and only one eye, whereas the neighboring enemy group has much more territory. The solution remembered to save the group 1 in Figure 6 is to play the move that enables to make two eyes and therefore to live (preventing forever the opponent to remove the group from the board). According to the stored move marked Δ in Figure 6, the system will choose to play the move marked Δ in Figure 5.

Case-Based Reasoning enables imprecision in the use of knowledge. Strategy is naturally imprecise. Allowing to define similarity with some reference groups enables to have a concise and general representation of strategic knowledge.

Results

The Go program plays a move in 10 seconds on a Pentium 133 MHz, for each move it proves about 450 tactical theorems, each theorem requires between 4 and 600 nodes in a search tree to be proved, at each node of each tree, the rules learned by Introspect are called to find the useful moves to try. Introspect has learned thousands of tactical rules. All the learned rules are compiled into a 1 000 000 lines C++ program. The strategic level chooses plans using constraints on some properties of the groups. The groups and their properties are built using the results of the tactical level. When strategic plans are chosen, moves related to the plans are chosen using information about previously seen similar situations.

Gogol competed in the international computer Go tournament held during IJCAI97 together with 40 other participants. It finished 6 out of 40 participants. The five first programs are commercial programs that have required a lot of person*years of work. It has outperformed other commercial systems that have required more than 10 person*years of work.

Conclusion

We have shown how to integrate multiple reasoning modes in a complex domain that requires different forms of reasoning. Rule-based reasoning is used at the tactical level in our Go program to select the useful moves to try when searching. Constraint-based reasoning is used to select interesting plans according to constraints. Once the plans are chosen, Case-based reasoning is used to select the moves that enable the plans to work. Each move has a value that is the sum of the values of the plans the move achieves. The resulting Go program has good results in international competitions (best non-commercial program). This approach combining multiple types of reasoning can also be used in other domains that are complex enough to require different kind of knowledge to use knowledge [Pitrat 1990].

A lesson learned from applying multimodal reasoning to a very complex task like the game of Go, is that in complex domains, as we need a lot of knowledge, using multiple reasoning modes is appropriate because there are different types of knowledge. Each type of knowledge is suited to a particular reasoning mode. The problem is to split a system into modules, and to choose a reasoning mode for each module. In our application, we chose to separate our system in three modules: A theorem prover that uses rules and predicate logic for exact computations. A module based on constraints to choose plans according to predefined thresholds. A Case-Based Reasoning module that enables imprecision in the recognition of how much a move enables to achieve a strategic goal that cannot be exactly foreseen.

Human Go players also use different reasoning mode when studying a position. They search very fast when reading tactical sequences of moves, using complex learned patterns to choose the moves to try. They have a less rigorous reasoning mode when they think strategically. As we have shown with the game of Go, we believe that the ability to switch between reasoning modes is necessary to have good performances in many complex cognitive tasks.

References

Allis, L. V. 1994. Searching for Solutions in Games an Artificial Intelligence. Ph.D. diss., Vrije Universitat Amsterdam, Maastricht.

Cazenave, T. 1996. Système d'Apprentissage par Auto-Observation. Application au Jeu de Go. Ph.D. diss., Université Paris 6.

Cox, M. T. 1996. Introspective Multistrategy Learning : Constructive a Learning Strategy Under Reasoning Failure. Ph.D. diss., Georgia Institute of Technology, College of Computing, Atlanta.

Dejong, G. and Mooney, R. 1986. Explanation Based Learning : an alternative view. *Machine Learning* 1 (2).

Epstein, S. L. and Gelfand J. J. 1996. Pattern-based learning and spatially-oriented concept formation in a multi-agent, decision-making expert. *Computational Intelligence* 12 (1):199-221.

Laird, J.; Rosenbloom, P. and Newell A. 1986. Chunking in SOAR : An Anatomy of a General Learning Mechanism. *Machine Learning* 1 (1).

Mitchell, T. M.; Keller, R. M. and Kedar-Kabelli S. T. 1986. Explanation-based Generalization : A unifying view. *Machine Learning* 1 (1), 1986.

Nigro, J.-M. and Cazenave, T. 1996. Constraint-based explanations in games. In Proceedings of IPMU'96, Granada, Spain.

Pitrat, J. 1990. *Métaconnaissance - Futur de l'Intelligence Artificielle*. Hermès, Paris.

Ram, A. and Leake, D. 1995. *Goal-Driven Learning*. Cambridge, MA, MIT Press/Bradford Books.

Robson, J. M. 1983. The Complexity of Go. In Proceedings IFIP, 413-417.

Selman, B.; Brooks, R. A.; Dean, T.; Horvitz, E.; Mitchell, T. M.; Nilsson, N. J. 1996. Challenge Problems for Artificial Intelligence. In Proceedings AAAI-96, 1340-1345.

Van den Herik, H. J.; Allis, L. V.; Herschberg, I. S. 1991. Which Games Will Survive ? Heuristic Programming in Artificial Intelligence 2, the Second Computer Olympiad (eds. D. N. L. Levy and D. F. Beal), pp. 232-243. Ellis Horwood. ISBN 0-13-382615-5. 1991.