Multi Subsuming Multi Outcome Games

Walter B. Gress V

Drexel University

Friday, June 7 2013

This paper discusses the possibility of using Subsumption Architecture along with the Theory of Games to create a reliable strategy in Real Time Strategy Games

# introduction

The state space in RTSes is immense and trying to derive any sort of game tree in real time for the dynamic elements is processor intensive and difficult, if not impossible by today’s standard algorithm. In this paper I will provide a method that results in an approximation of a solution for the theory of real time strategy games using a combination of game theory and subsumption architecture. Subsumption architecture allows seemingly simple operations, set within an architecture that results as a sort of emergent behavior complex operations and results. Subsumption architecture however, is based on the idea of not necessarily computing with inward data structures, it is a reactive architecture for the most part, but this has allowed robots built with subsumption architecture to, per se, navigate a hall way, pick up a soda can and throw it away without an internal map or direct mechanism.

# SUBSUMPTION ARCHITECTURE

Rodney Brooks developed the subsumption architecture in the 1990s. It is a reactive architecture and bases its decisions on input rather than “mental models.” The AI is structured in layers of finite state machines and one layer can subsume or take over, other layers when necessary. For example, say we have a robot that does the following: wanders around; sees somebody and runs from them; hides when the lights go out. Hide when the lights go out may subsume run away from an intruder which may subsume wander about.

# SUBSUMPTION GAMES

A subsumption game is the following:

1. Each move only requires minimal knowledge of its environment, key being, its past states .
2. Is reactant and performs action only upon input.
3. Reacts predictably to input.
4. Has multiple states and subsumes states according to entries in the game.

# LOOKAHEAD GAMES AND SUMS

In this paper we will discuss something I have dubbed “look-ahead games.” In these games, the previous game influences the present game, and the present game influences the future games. They do this by adding to each game the scores of the prior games. Often times a different outcome than what would expect occurs. These sum games are the summation of the scores of the look-ahead games and are very important in this paper.

# OPTIMAL STRATEGIES

Unlike checkers or go, or even chess, real time strategy games have a huge state space and unlike checkers or go, it likely there is no common optimal strategy. There could be however multiple optimal strategies. I conjecture that these multiple optimal strategies are all related to one another, according to their matrix scores. They may be multiples of another, inverses, etc. It is only logical that one way of winning can be related to another way of winning. Take baseball as example. Individually the batters can hit home runs, grand slams, or just doubles or triples. The defending player can catch a fly ball for an out, etc. Like RTSes, baseball is complex, dynamic game. But in the end, it is simply the team with the most points that wins. There are dozens of optimal strategies but again, in the end its points that matter. That is what I am trying to stress here. There may be multiple optimal strategies in RTSes, but in the end the only thing that matters is the scoreboard or the matrix that represents a winning game.

# NOMENCLATURE

In this article I use the standard matrix for individual games except for one major change. Instead of using scores, we rate each behaviour for the game using an S function. The S function takes the optimal solution for that game. Here plays the subsumption part. The game is iterated. Next, that same cell can be subsumed depending on conditions. When we get to the examples things will become clearer.

# SOME EXAMPLE GAMES

In this first game, there are two identical players. They can choose HEAL for a, ATTACK for b, or DEFEND for c, PATROL for d.PATROL is the base, but DEFEND can subsume it, and is be subsumed by ATTACK, which can be subsumed by HEAL.

Initial Matrix:

|  |  |  |
| --- | --- | --- |
|  | Strategy A | Strategy B |
| Player One | S(0) | S(a) |
| Player Two | S(b) | S(0) |

Round Two:

Healing subsumes attack

|  |  |  |
| --- | --- | --- |
|  | Strategy A | Strategy B |
| Player One | S(0,0) | S(a,a) |
| Player Two | S(0,a) | S(0,0) |

Round Three:

Attack is over

Attacking no longer subsumes. Revert to patrol.

|  |  |  |
| --- | --- | --- |
|  | Strategy A | Strategy B |
| Player One | S(0,0,d) | S(a,a,d) |
| Player Two | S(0,a,d) | S(0,0,d) |

Attacked, by player two this time! Player 2 defends.

|  |  |  |
| --- | --- | --- |
|  | Strategy A | Strategy B |
| Player One | S(0,0d,a) | S(a,a,d,a) |
| Player Two | S(0,a,d,c) | S(0,0,d,c) |

Etc.

Eventually we look at the last matrix, which is a sum of all the previous and assuming, for sake of space and sanity, the above matrix is the last matrix. If so, then Player One with Strategy B is the winner, it has the most attacks.

# SUBSUMPTION SCORE

We are trying to compare sequences to represent wins or losses in the overall game. If, for example, we have two players. One player scores a S(a,a,a,b,b,c) and the second player scores a S(b,b,b,a,c,d) because a subsumes b which subsumes c, the first player has a higher subsumption score.

# PROBLEM IN DETAIL

Given a winning strategy of , where } for all G where G is all the games playable, and n is the number of games played, and every game g is a matrix 2x2 Matrix M of S functions, and S contains a sequence of subsumption behaviors: A. A is complete set of all subsumption behaviors.

g is a winning game if for all

S G is strictly higher (has a higher subsumption score), than his/her opponent.

Strictly higher meaning given a score of subsumption behaviors for that game g, |()| > |

Where S1 represents the first players final score for that cell and S2 the second player final score for that cell.

is the subsumption score where for , the greater the individual a’s, the greater the score. That is, the higher in the subsumption architecture of the player, the better their score: |.

and are uncommon S scores and if , then win and vice versa. If they are equal then there is tie.

# DELTA SCORE

I propose a complete strategy can be met by the use of successive delta scores.

Where is a strategy score, and is the set of all strategy scores, with then,

= { } where each has n members.

The score consists of the summation of multiple subsumption games.

# CONCLUSION

By using game theory we can skirt around some of the issues related to RTS games where the branching factor of the game tree is exceedingly complex. This paper presented a method of using a single solution via multiple outcomes to help mitigate these issues.

# BIBLIOGRAPHY

Santiago Ontanon, G. Synnaeve, Alberto U, Florian Richoux, David Churchill, Mike Preuss. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. IEEE Transactions on Computational Intelligence and AI in Games. IEEE Computational Intelligence Society, 2013, 5 (4) pp. 1-19.

Rodney A. Brooks. Elephants Don’t Play Chess. MIT Artificial Intelligence Laboratory, Cambridge MA 02139, USA.

Russell, Norvig. Artificial Intelligence: A Modern Approach

Steven Tadelis. Game Theory: An Introduction