Chapter 1

Literature Survey

In 2011, Simon Davies created a AI bot for the Zerg race in StarCraft: Brood War following the Behaviour Oriented Design development methodology. This project aims to use the work Simon performed and implement improvements and generalise his code for bots of other races.

StarCraft has recently seen much development in the AI area, due to the release of a convenient API for the purpose of AI development. What follows is a rough review of work in this area, including a brief explanation of StarCraft, the field of AI and agents in general and Behaviour Oriented Design.

1.1 StarCraft

StarCraft is a real-time strategy game where the player commands an army from one of 3 distinct races. It was released in 1998 by Blizzard Entertainment and its expansion StarCraft: Brood War was released later the same year [2]. During a game, the player must harvest two limited resources, minerals and vespene gas, and use them to construct buildings, recruit units and research upgrades with the ultimate goal of destroying the opponent. The game can be said to be split into 3 stages: The opening, where choosing the most suitable order of construction is vital due to limited resources; the mid-game, where players build up for larger attacks and must expand to gain more resources; and the end-game, where one player has gained the advantage and should be looking to make a very large push to secure it [32].

The 3 races differ from each other substantially in both play style and unit ability. The insectile Zerg focus on small, fast and cheap units and swarm tactics; the psychic and advanced Protoss focus on expensive but extremely powerful units; and the human Terran are the balance between the two, not as weak or as numerous as the Zerg, but not as powerful or as expensive as the Protoss. They also focus on attacking from range. Due to the games age and popularity, there are many online communities that have collated a large amount of information about the game and effective strategies, such as the StarCraft wiki [9] and team

liquid [10].

StarCraft has regularly found itself used as a subject in academia, probably due to its prominence and well-balanced game play. Uses can vary from essays on what StarCrafts design communicates [17], analysis of the games network traffic [15] , to algorithmically generating playable maps [29]. The creation of the Brood War Application Programming Interface (BWAPI) in 2010 [3] has led to increased usage as a platform to explore and construct AI and a target for AI competitions [4].

1.2 Artificial Intelligence

AI is a wide ranging field concerned with the research and development of machines/programs capable of intelligence. It includes a wide variety of sub-fields like "computer vision, natural language, decision theory, genetic algorithms and robotics" [22]. There is the concept of strong AIs, which are the kind of AI visualised by the general public and science fiction writers, true "machines that think", conscious and capable of human emotion [21]. AIs that dont attempt to simulate the entirety of human intelligence can be called weak AI, and its probably safe to say all AIs created to date are weak AI, including those found in games.

The concept of an agent is popular one in AI. An agent is a system capable flexible, autonomous action in some environment [31]. Thats a rather vague definition, so to expand it; an agent is a computer system which has a goal to accomplish, resides in an environment which it can effect some change upon and can react to changes in that environment. An environment does not have to be physical, but there are other classifications to consider [25]:

- Accessible vs. inaccessible How much information is available to the agent about the environment? Accessible environments provide complete, accurate and up-to-date information about the environments state.
- Deterministic vs. non-deterministic What happens when the agent acts? In deterministic environments an action has a single guaranteed effect, with no uncertainty.
- Static vs. dynamic Is anything else changing the environment? Static environments only change due to actions by the agent.
- Episodic vs. non-episodic How is the agent rated? An agent rated in an episodic environment takes part in several discrete episodes which remain independent, so past and future performances arent important. For example, an AI taking part in a tournament could consider each game to be a single episode.
- Discrete vs. continuous How many actions are available to the agent? Discrete environments have a very limited and fixed number actions and a small amount of required knowledge in it.

StarCraft can be argued to be an inaccessible, deterministic, dynamic, continuous environment with the possibility to be episodic and accessible depending on game settings [16]. The majority of StarCraft games have the fog of war setting enabled, so only objects within unit vision are visible and have up-to-date information. It is possible to disable the setting, turning it into an accessible environment. Almost all agent actions are deterministic, though there is some small randomness in ranged attacks. StarCraft is played against another agent, so must be dynamic. The number of units and the number of positions they can be in make it more like a continuous environment than a discrete one, though I think the number of actions available is just extremely high rather than infinite.

1.3 Behaviour Oriented Design (BOD)

BOD is a AI development methodology conceived by Bryson [13]. It details both an agent architecture and a design methodology. The BOD architecture has 2 parts, a modular library of behaviours (actions, senses and state) and parallel rooted, ordered, slip-stack, and hierarchical (POSH) reactive plans for action selection. The methodology specifies how to decompose an agent's behaviours and emphasises an iterative development approach for implementation [12]. BOD focuses on the design of the AI system, aiming to produce agents that are easy to extend and maintain. It's modularity allows many different AI techniques to be used in tandem, encouraging the developer to use whatever approach fits an individual problem best [18].

1.3.1 Architecture

As stated, the BOD architecture consists of 2 main parts: The Behaviour Library Behaviours are like object oriented design (OOD) objects and encapsulate how to do something, including perception and actions. Behaviours should be modular, can have state, can implement learning techniques, or even be wrappers for external libraries. This separation from the action selection also facilitates code re-use and allows modification of behaviours without affecting plans [18]. POSH plans and action selection POSH plans are a hierarchy of behaviours and their triggers [18]. The primitives of the hierarchy are actions and senses. These primitives are formed into 3 groupings, action patterns, competences and drive collections, in order of increasing complexity. Action patterns are simple sequences of primitives; competences are basic reactive plans, a prioritised list of actions and triggers; drive collections are the root of the hierarchy and determine where the agents attention should be focused [12].

1.3.2 Decomposition

BOD provides a set of initial steps to begin the decomposition of behaviours and the construction of plans. The steps are as follows [12]:

1. Specify at a high level what the agent is intended to do.

2. Describe likely activities in terms of sequences of actions. These sequences are the basis of the initial reactive plans.

3. Identify an initial list of sensory and action primitives from the previous list of actions.

4. Identify the state necessary to enable the described primitives and drives. Cluster related state elements and their primitives into specifications for behaviours. This is the basis of the behaviour library.

5. Identify and prioritize goals or drives that the agent may need to attend to. This describes the initial roots for the POSH action selection hierarchy.

6. Select a first behaviour to implement.

As BOD is meant to be performed iteratively, getting these steps right initially isn't paramount, but will save some effort later on.

1.3.3 Methodology

BOD champions an iterative development cycle and rapid prototyping. It specifies to only implement a section at a time, to fully test and debug each section and to re-evaluate the specification after each section is complete. During this evaluation, the focus is to simplify, if feasible, anything that has gotten too complex.

1.4 Simon Davies' AI

In 2012, Simon Davies used BOD to create a simple AI bot for StarCraft [16]. The strategy it follows is quite simple; it builds up a simple army until it has a certain number of units and then it attacks the enemy until it has lost a certain number of units. It also builds defences and expands its resource harvesting capacity.

This strategy leaves the bot weak in the early game, and stronger in the late game. It doesn't use information about its opponents to modify this strategy, so often lost when opponents tried an early game push. The bot also has problems with the implementation being a bit rough; there's only two hard coded base expansions; melee (close combat) units will try to attack air units; and forces wait on a single slow scout to find the enemy, when they could do it themselves.

The bot has plenty of opportunity for expansion and improvement mainly in the strategy choice and unit control areas. The behaviours are implemented through a collection of managers for each subsection of the game, and the demarcations make sense. Most of the behaviours and the action patterns are specific to the Zerg race, but many look to be trivial to generalise and the structure of some competences should apply to the other races as well.

1.5 Possible ideas for expansions

This section covers recent AI strategies implemented for StarCraft and offers some opinion on their applicability for improving Simon's bot.

1.5.1 Brood War Terrain Analyser (BWTA)

This is an extension for BWAPI that provides useful information about the current map [11]. This include things like likely base locations, choke points and unwalkable terrain. It does this by identifying obstructed areas of the map, creating and pruning a Voronoi diagram of this areas, using this to identify map regions, and identifying where they connect as choke points [23]. It computes base locations by identifying regions with resources. This would be in some use for expanding Simon's bot, it would help a lot with scouting and expansion and identifying choke points could reduce the cost of base defence (so it could be effective earlier). It appears to be available through JNIBWAPI (the Java interface Simon used) though there may be issues to work around [5].

1.5.2 Improving unit micro-management

The micro-management (micro) of military units in Simon's bot is very simple. Enemies are prioritised based on distance and type and friendly units are told to move to and attack the highest priority unit. There are other works that show possible paths for improvement.

Potential fields

Adapted from an idea in physics, this is where individual entities in the game are given an attractive or repulsive force. So your units would be attracted to weak units while being repulsed from stronger or more dangerous units. Units can also be affected by changes in their own circumstances, perhaps retreating while under fire. The advantage of potential fields is that a single function can control lots of units, however the strength of the forces involved must be figured out manually (though someone has used genetic algorithms to improve theirs [24]). The Berkeley Overmind [1] implemented potential fields to such success that it was able to beat an ex-professional player [20]. There are also an open source implementation available [7].

Monte-Carlo planning

Planning is usually associated with large scale strategy, due to its time cost and the large search space of micro-management. Wang et al [33] have used a Monte-Carlo planning method. Monte-Carlo methods rely on entering random samples into a simulation and analysing the results [6]. Wang et al created a simulation of the game state and then evaluated several predefined stochastic plans. They tested their AI in game and an imbalanced scenario (their AI was at a disadvantage) and it was able to beat beginner players and the original AI

with some consistency, though was still beaten by an expert. This does show some potential for planning at a micro level, but there are problems related to the speed of the simulation and the amount of expert knowledge needed to create the plans. They haven't made their implementation public, so it's unlikely this could be integrate into Simon's bot.

Bayesian Modelling

There is a project that focuses on using Bayesian models for all aspects of a StarCraft AI [28] which includes unit control [27]. This uses the distribution of game elements to decide what to do with a unit next. Units receive tactical goals as sensory inputs and maintain a finite state machine (FSM) for different modes (attacking, defending, etc.). It uses re-enforcement learning to build its probability tables and succeed quite well in trails against both the original AI and against winners of the 2010 AIIDE competition. Their implementation is available open source.

1.5.3 Macro strategies

Macro is a term for the overall strategy and generally refers to economic and military plans. Simon's bot doesn't have very good macro, it has a single strategy and it sticks to it. Improving and generalising his work will require creating several strategies, and should look into making them adaptable.

Planning

There are many replays of StarCraft games publicly available [8], some of which are of high level play. Using BWAPI, it's possible to retrieve lots of data from these replays and data mine the results. There are several papers that do this and use the data to create Bayesian models of what the opponents are likely to be doing [27, 19, 30]. This has the advantage that the bot gains expert-level knowledge without its creator needing to know expert strategies and it also has quite a low operational footprint. Planning and prediction seem to be popular in the StarCraft AI field, probably due to the availability of vast amounts of completed game data.

Case-based reasoning

Certicky [14] implemented cased-based reasoning (CBR) for army composition. CBR is a cyclical process where there is a databases of problem cases with solutions. For each problem, a case is chosen that best fits. The solution is adapted to fit the current problem and if it works it is added back to the database as a new case. Certicky constructed his cases based on the ratios of units that the opponent has and the solutions are the recommend ratios for the bots army. This process has the potential to be useful, but it needs quite a bit of expert knowledge about common army ratios and their counters. Additionally,

this relies on good scouting and accurate information, if the bot picks the wrong case it could be disastrous.

Use more gun

Safadi and Ernst [26] take the interesting position that a bots macro strategy is fairly unimportant. Their view is that planning and pattern recognition is something humans excel so well at that a bot couldn't hope to match it, the bot would require a database of possible strategies which would require continuous updates as the meta-game evolves. Instead the bot should play to it's advantages in the multi-tasking area; 300 unique action per minute is considered the average for StarCraft professionals, but it would be trivial for a computer to match and beat that. There does seem to be some truth in this, as it's basically how the Berkeley Overmind succeeds.

Bibliography

[1] Berkeley overmind. http://overmind.cs.berkeley.edu/#desc. Retrieved November 22, 2013.

[2] Blizzard Entertainment starcraft. http://us.blizzard.com/en-us/games/sc/. Retrieved November 21, 2013.

[3] Bwapi. https://code.google.com/p/bwapi/. Retrieved November 21, 2013.

[4] BWAPI competitions held for bwapi. https://code.google.com/p/bwapi/wiki/Competitions. Retrieved November 21, 2013.

[5] Jnibwapi. https://code.google.com/p/jnibwapi/issues/detail?id=2. Retrieved November 22, 2013.

[6] Monte-carlo method. http://www.encyclopediaofmath.org/index.php/Monte-Carlo_method. Retrieved November 22, 2013.

[7] A starcraft ai bot. https://code.google.com/p/bthai/. Retrieved November 22, 2013.

[8] Starcraft replays. http://www.teamliquid.net/replay/. Retrieved November 22, 2013.

[9] Starcraft wiki. http://starcraft.wikia.com/wiki/StarCraft_Wiki. Retrieved November 21, 2013.

[10] Team liquid. http://www.teamliquid.net/. Retrieved November 21, 2013.

[11] A terrain analyser for bwapi. https://code.google.com/p/bwta/. Retrieved November 21, 2013.

[12] Joanna J Bryson. The behavior-oriented design of modular agent intelligence. *Agent technologies, infrastructures, tools, and applications for e-services*, pages 61–76, 2003.

[13] Joanna Joy Bryson. Intelligence by design: principles of modularity and coordination for engineering complex adaptive agents. 2001.

[14] Martin Certicky and Michal Certicky. Case-based reasoning for army compositions in real-time strategy games. In *Scientific Conference of Young Researchers 2013*, pages 70–73. FEI, Technical University of Košice, 2013.

[15] Alberto Dainotti, Antonio Pescape, and Giorgio Ventre. A packet-level traffic model of starcraft. In *Hot Topics in Peer-to-Peer Systems, 2005. HOT-P2P 2005. Second International Workshop on*, pages 33–42. IEEE, 2005.

[16] Simon Davies. Development of an ai for the real-time-strategy game starcraft using behaviour oriented design. Master's thesis, University of Bath, 2012.

[17] Alexander R Galloway. Starcraft, or, balance. *Grey Room*, (28):86–107, 2007.

[18] Swen Gaudl, Simon Davies, and Joanna J Bryson. Behaviour oriented design for real-time-strategy games.

[19] Jesse Hostetler, Ethan W Dereszynski, Thomas G Dietterich, and Alan Fern. Inferring strategies from limited reconnaissance in real-time strategy games. *arXiv preprint arXiv:1210.4880*, 2012.

[20] Haomiao Huang. Skynet meets the swarm. `http://arstechnica.com/gaming/2011/01/skynet-meets-the-swarm-how-the-berkeley-overmind-won-the-2010-starcraft-ai-competition/`, 2011. Retrieved November 22, 2013.

[21] Ray Kurzweil. The singularity is near: When humans transcend biology. 2005.

[22] P McCorduck. Machines who think: A personal inquiry into the history and prospects of artificial intelligence, ak peters. *Natick, Mass*, 2004.

[23] Luke Perkins. Terrain analysis in real-time strategy games: An integrated approach to choke point detection and region decomposition. *AIIDE*, pages 168–173, 2010.

[24] Espen Auran Rathe and Jørgen Bøe Svendsen. Micromanagement in starcraft using potential fields tuned with a multi-objective genetic algorithm. 2012.

[25] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. Artificial intelligence: a modern approach. 1995.

[26] Firas Safadi and Damien Ernst. Organization in ai design for real-time strategy games. 2010.

[27] Gabriel Synnaeve and Pierre Bessiere. A bayesian model for rts units control applied to starcraft. In *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*, pages 190–196. IEEE, 2011.

[28] Gabriel Synnaeve and Pierre Bessiere. A bayesian tactician. In *Proceedings of the Computer Games Workshop at the European Conference of Artificial Intelligence 2012*, 2012.

[29] Julian Togelius, Mike Preuss, Nicola Beume, Simon Wessing, J Hagelback, and Georgios N Yannakakis. Multiobjective exploration of the starcraft map space. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 265–272. IEEE, 2010.

[30] Ben George Weber and Michael Mateas. A data mining approach to strategy prediction. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 140–147. IEEE, 2009.

[31] Michael Wooldridge and Nicholas R Jennings. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(02):115–152, 1995.

[32] Sangho Yi. Adaptive strategy decision mechanism for starcraft ai. *EKC 2010*, pages 47–57, 2011.

[33] Wang Zhe, Kien Quang Nguyen, Ruck Thawonmas, and Frank Rinaldo. Using monte-carlo planning for micro-management in starcraft.