Ontological Knowledge for Goal-Driven Autonomy Agents in Starcraft

Dustin Dannenhauer
Department of Computer Science and Engineering
Lehigh University, Bethlehem PA 18015, USA
dtd212@lehigh.edu

Introduction

Starcraft, a commercial Real-Time Strategy (RTS) game that has enjoyed world-wide popularity (including televised professional matches), is a challenging domain for automated computer agents. Evidence of this difficulty comes not only from characteristics of the game (massive state space, stochastic actions, partial visibility, etc.) but also from three years of competitive entries in tournaments (i.e. AIIDE Annual Starcraft Competition) in which the best automated entry performs poorly against a human expert. We are interested in taking a new research direction: using semantic knowledge, such as description logic, to represent the game state with abstract concepts in order to perform high level actions.

Goal-driven autonomy (GDA) is a conceptual model for reasoning over goals in autonomous agents. The model makes use of a planner which produces both a plan and corresponding expectations. Expectations are what the agent expects to be true after executing actions in the plan. GDA reasons about goals in four components: discrepancy detection, explanation, goal formulation, and goal management. The process of a GDA agent begins with the goal management component sending a goal to the planner. The planner then produces a plan and expectations for what should be true in the future, as plan actions are executed. The agent begins executing the plan and, concurrently, the discrepancy detection component observes any expectations that are not satisfied. Upon identifying an unmet expectation, the explanation component generates an explanation for why the discrepancy occurred and sends the explanation to the goal formulation component, which may create a new goal. This new goal is sent to the goal management component, which then adjusts the priority of each goal and decides which goal(s) to send to the planner, and the cycle continues. An important aspect of the GDA model is that each component is open to implementation.

Almost all previous work on GDA has been non-hierarchical (for example, discrepancy detection in the ARTUE (Molineaux, Klenk, and Aha 2010) system uses set-difference). Nor has previous work made use of an ontology. We propose using an ontology to represent the state of the environment in order to provide more knowledge-rich reasoning within GDA. At this time we are not committed to a particular formalism for an ontology, and are currently looking into using a Web Ontology Language capable of expressing Description Logics (OWL-DL) in combination with a reasoner that supports Semantic Web Rule Language (SWRL) rules. One such existing reasoner is Pellet (Sirin et al. 2007). Real-time strategy games have seen previous work that represents the game state as an ontology (Sanchez-Ruiz et al. 2007). Starcraft is one such RTS game that is getting increased attention from research communities because of available API's, large amounts of online data (professionally played games, wikis, strategy discussion), stochastic gameplay, and large state space. RTS games are typically programmed with managers (Scott 2002), the work presented here could be used as the civilization manager of a multi-manager system. We now walk through an example, in Starcraft, in which each step of the GDA model could use an ontology to reason about high level strategies with explicit knowledge.

1. **Discrepancy Detection:** A hypothetical plan achieving the goal "surround the enemy base" could produce an expectation such as "the agent controls regions 5 and 6". This kind of expectation is possible with the use of rules, such as a SWRL rule: "A player controls a region if there is at least 1 unit in the region and that player owns every unit in the region." (Reasoning over regions is possible with the API Brood War Terrain Analyzer (Perkins 2010) that automatically decomposes maps into regions). The discrepancy detector could identify if this expectation was met by adding the expectation statement to the ontology (player X controls region Y), running it through a reasoner, and checking consistency. Upon an inconsistency the reasoner provides an explanation trace of what facts are conflicting. In this trace would be the SWRL rule which would have as its consequent the fact that player X controls region Y. The trace also shows which antecedents of the rule are inconsistent, which enables useful explanation.

   An alternative method (faster but more knowledge intensive) would be to query the ontology (i.e. SPARQL queries could be used for OWL-DL ontologies) for each expected fact: (1) agent controls region 5, (2) agent controls region 6. Querying is more knowledge intensive be-

cause there is no immediate connection between a missing fact (missing expectation) and its corresponding explanation (there is no inconsistency trace). SPARQL queries may be justified for primitive expectations (such as counting the number of fighting units we have) because there is no connecting explanation in the ontology (no rule where the consequence is that the agent owns a unit). Here we focus on reasoning over the ontology and detecting expectations via inconsistencies with corresponding traces.

2. **Explanation:** Continuing the example above, the explanation component could identify which part of the rule was not satisfied via the trace. If there were no units in the region, then a valid explanation could be that somehow our units were not able to travel to the region (perhaps there is an enemy force between the agent's units and the target region). But if the other antecedent was inconsistent, i.e., not all the units were ours, that would be evidence for a different explanation, perhaps that the enemy controls the target region.
3. **Goal Formulation:** Either explanation would provide rich knowledge for the goal formulation component. If the explanation was that agent's units were not able to travel to the region, the new goal could be to send different types of units (i.e. flying vs. ground), or take a different path. If the explanation was that the enemy controlled the target region, then perhaps the new goal could be to send a more powerful force, with different units. Unit types could be chosen to be those specifically effective at defeating the enemy units that destroyed the first force. Which units are effective against other units can be represented in the ontology.
4. **Goal Management:** The goal management component would take the goals produced from the goal formulation component and adjust the priority of the current goal. Perhaps the new information in the ontology warrants abandoning the "surround the enemy base" goal and instead pursuing the newly suggested goal from the goal formulation component.

Most previous GDA systems reason at the unit level of a game (i.e. unit U3 is at (5,6)). While data at the unit level is important, it lacks the more abstract knowledge (i.e. data at a region level) that is needed to produce high-level strategies, such as those employed by humans. Expert human Starcraft players are admired for both their low-level unit micromanagement abilities as well as their high level strategic abilities. Ontologies allow for the representation of both low-level unit details and high level details, including knowledge that bridges these different levels. The ability to reason at the region level (such as labeling a region has "unknown", "contested", "owned by the agent" or "owned by the enemy") opens the door for higher level strategies. The kinds of strategies we describe here are more abstract than the strategies used in EISBot (Weber, Mateas, and Jhala 2011). EISBot's strategy selection component determines what types of units to produce and which upgrades to research. Our strategies reason about where and how to attack the enemy (which also includes which unit types and upgrades to pursue).

A motivation for using ontologies is that STRIPS representations are too limited to represent events that happen in the real world (or complex domains), more structured representations are needed in order to capture more complex constraints (Gil and Blythe 2000). While ontologies are knowledge-rich and more expressive than STRIPS, a recurring problem is knowledge engineering. Most work on GDA has assumed that GDA knowledge is given by the user or domain expert, aside from the work done by Jaidee et al. where expectations and goal formulation are learned via reinforcement learning (Jaidee, Muñoz-Avila, and Aha 2011a),(Jaidee, Muñoz-Avila, and Aha 2011b)) and Weber et al. where goals are formulated from cases made from Starcraft game traces (Weber, Mateas, and Jhala 2010). To reduce the knowledge engineering burden of creating ontologies, we propose using automated text-extraction tools to build initial ontologies. For self-contained domains such as Starcraft, a significant amount of textual information is available online which could be used to build an ontology. There has been previous work that demonstrated extracting text from the manual for the RTS game Civilization II to create rules based on the strategy described in the manual (Branavan, Silver, and Barzilay 2011). Additionally, there are automated tools for creating semantic web ontologies which may be able to be used to create initial ontologies from online resources (Laender et al. 2002).

Research Plan

We plan to manually design an ontology with the help of a domain expert for a GDA agent that can play Starcraft. This will most likely require new GDA algorithms to integrate such an ontology into the GDA agent. We will then use others' strategies for extracting text to automatically build ontologies from text data. We expect the GDA agent with the hand crafted ontology will outperform the agents with automatically created ontologies (due to the likelihood that automatically extract ontologies will have more errors). A future research task will be to explore techniques to refine automatically learned ontologies.

The following is the research plan:

1. Perform a literature overview on GDA research (done)
2. Investigate Starcraft as a domain for experimentation (done)
3. Investigate use of ontologies in the context of GDA (done)
4. Implement a system exhibiting the results of (3) and test it in Starcraft (in progress)
5. Perform a literature review of ontology extraction
6. Investigate automated extraction techniques of ontologies from textual/web information sources
7. Apply (6) on Starcraft strategy online resources
8. Investigate learning techniques to refine ontologies from episodic knowledge
9. Perform an empirical evaluation on Starcraft comparing: GDA with hand-crafted ontologies vs. with automatically extracted ontologies

10. Perform empirical evaluations on Starcraft comparing GDA with ontologies against previous tournament winners and/or skilled human players

Progress

We have currently looked into taking a case-based reasoning approach to the knowledge engineering burden required to use scoring algorithms from IBM's Watson (Ferrucci 2012) for the goal management/selection component of GDA (Dannenhauer and Muñoz-Avila 2013). This relies on the presence of oracles (possibly humans) and uses evidence scoring algorithms to rank goals in order to choose the best one. We are currently working towards creating an expert given ontology for Starcraft as well as a GDA implementation of a Starcraft bot for step 4 in our research plan.

References

Branavan, S.; Silver, D.; and Barzilay, R. 2011. Learning to Win by Reading Manuals in a Monte-Carlo Framework.

Dannenhauer, D., and Muñoz-Avila, H. 2013. Case-Based Goal Selection Inspired by IBM's Watson. In *Case-Based Reasoning Research and Development*. Springer. 29–43.

Ferrucci, D. A. 2012. Introduction to This is Watson. *IBM Journal of Research and Development* 56(3.4):1–1.

Gil, Y., and Blythe, J. 2000. How Can a Structured Representation of Capabilities Help in Planning. In *Proceedings of the AAAI–Workshop on Representational Issues for Real-world Planning Systems*.

Jaidee, U.; Muñoz-Avila, H.; and Aha, D. W. 2011a. Case-Based Learning in Goal-Driven Autonomy Agents for Real-Time Strategy Combat Tasks. In *Proceedings of the ICCBR Workshop on Computer Games*, 43–52.

Jaidee, U.; Muñoz-Avila, H.; and Aha, D. W. 2011b. Integrated Learning for Goal-Driven Autonomy. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, 2450–2455. AAAI Press.

Laender, A. H.; Ribeiro-Neto, B. A.; da Silva, A. S.; and Teixeira, J. S. 2002. A Brief Survey of Web Data Extraction Tools. *ACM Sigmod Record* 31(2):84–93.

Molineaux, M.; Klenk, M.; and Aha, D. W. 2010. Goal-Driven Autonomy in a Navy Strategy Simulation. In *AAAI*.

Perkins, L. 2010. Terrain Analysis in Real-Time Strategy Games: An Integrated Approach to Choke Point Detection and Region Decomposition. In *AIIDE*, 168–173.

Sanchez-Ruiz, A.; Lee-Urban, S.; Munoz-Avila, H.; Diaz-Agudo, B.; and Gonzalez-Calero, P. 2007. Game AI for a Turn-based Strategy Game with Plan Adaptation and Ontology-based Retrieval.

Scott, B. 2002. Architecting an RTS AI. *AI Game Programming Wisdom* 2.

Sirin, E.; Parsia, B.; Grau, B. C.; Kalyanpur, A.; and Katz, Y. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics: science, services and agents on the World Wide Web* 5(2):51–53.

Weber, B. G.; Mateas, M.; and Jhala, A. 2010. Case-Based Goal Formulation. In *Proceedings of the AAAI Workshop on Goal-Driven Autonomy*.

Weber, B. G.; Mateas, M.; and Jhala, A. 2011. Building Human-Level AI for Real-Time Strategy Games. In *Proceedings of the AAAI Fall Symposium on Advances in Cognitive Systems*, 329–336.