Towards General Evaluation of Intelligent Systems

Using Semantic Analysis to Improve Environments in the AIQ Test

Ondřej Vadinský

Department of Information and Knowledge Engineering
University of Economics Prague

August 2018

Introduction

- To achieve *Artificial General Intelligence*...
- **What is intelligence and how can it be evaluated in an artificial system?**
- *Universal Evaluation of Intelligence*:
  - Hernández-Orallo (2000) *C-Test* based on **Algorithmic Information Theory**
  - Legg and Hutter (2007) *Universal Intelligence definition*
  - Hernández-Orallo and Dowe (2010) *Anytime Intelligence Test*.
  - Legg and Veness (2011, 2013) **Algorithmic Intelligence Quotient Test** (detailed assessment in Vadinský (2018a))

Environment Programs in AIQ Test

- Legg and Veness (2011, 2013):
  - **compute the current reward and observation** from the history of interactions.
  - **randomly generated Turing-complete programs** *(modified BF language)*
  - runtime limits to ensure halting, basic syntax limits to encourage interactivity.
- Hernández-Orallo and Dowe (2010): **non-discriminative environments** do not meaningfully contribute to the agent's evaluation.

Research Questions

- How does chance influence an agent's rewards and observations?
- How do the actions of an agent influence its rewards and observations?
- What are the forms of code that can be considered pointless?

Semantic Analysis Overview

1. Informally specify a **Semantic Class**
   A semantic specification of a set of environment programs.
2. Derive one or more **Syntactic Classes**
   An expression in generalized BF language containing:
   - mandatory instructions
   - variable code fragments that meet given conditions
3. Convert into **Regular Expressions**
   (PCRE in GNU Grep)

Semantic Analysis Example

1. *The agent's reward is always random.*
2. `a%p.z#` where:
   - `a` cannot lead to premature termination, nor can it contain loops that are not closed, nor can it contain the write instruction.
   - `p` can only contain instructions `+-` and can be of a zero length.
   - `z` can contain any instruction.
3. `^[^\[\.]*%[\+\-]*\..*#`

Limits of the Method

- *Necessarily incomplete, and inaccurate* due to:
  - Many possible syntax for any given semantics
  - Syntactic limits of regular expressions
- Results are estimates:
  - **Sufficiently complete** given a finite sample of environment programs.
  - **Sufficiently accurate** with regard to the research questions.

Results Summary

- *Two types of results:*
  - **Syntactic classes** that formalize semantic classes in detail
  - **Proportion estimates** of semantic (and syntactic) classes
- *Random sampling of environment programs results in...*
  - **pointless code** (>74%),
  - **simple programs** (34%),
  - **non-discriminative environments** (17%),
  - more details in the Appendix.

Discussion

- Hernández-Orallo and Dowe (2010) suggest **changing to another reference machine**.
- Semantic analysis gives the necessary details to attempt to **optimize the sampling process**.
- The current method cannot always select the problematic code, only identify the program as problematic.

Improving Environment Programs of the AIQ Test

- *Changes to the BF programs sampler* aimed at:
  - **Removing pointless code**
    (obfuscates environment programs)
    program optimization
  - **Improving discriminative power**
    (wastes resources, distorts the score)
    sample optimization

Removing Pointless Code

- Legg and Veness (2011): some very basic types of pointless code are removed in one pass.
- Implemented changes:
  - *SEP-orig*: **Repetitive replacing procedure** (new default)
  - *SEP-ext*: **New replace patterns based on Semantic Analysis** (optional)
- Examples:

```
NoOpt:     ++----%+.+,>,#
LV:        +  ---%+.+,>,#
SEP-orig:     --%+.+,>,#
SEP-ext:        % . ,>,#
FullOpt:        % . ,> #
```

Improving Discriminative Power

- Legg and Veness (2011) drop programs that:
  - have no read or write instruction;
  - return the same reward.
- Implemented changes:
  - *SDP*: drop programs of **The agent's reward is always random** class identified by Semantic Analysis (optional)
- Examples:

```
LV:        +.%#
LV:        -,%#
SDP:       %.,>,#
SDP:       %[+.>],-->#
FullOpt:   ,[+>]%.#
```

Evaluation

Table: Evaluation of the AIQ test extensions (details in the Appendix)

Method	**SEP-orig**	**SEP-ext**	**SDP**
Descriptive statistics			
(program length)	no difference	decrease	increase
Semantic analysis			
(relevant classes proportion)	no difference	decrease	decrease
Experiments			
(agent scores difference)	negligible	*small increase*	large increase
(t-test significance)	weak	strong	strong

Discussion

- Limits of the implementation:
  - **Not all problematic programs are removed**
  - Some cases with complex conditions require different approach.
  - PCRE can identify the program as problematic but not the code.
- Usage recommendations:
  - *SEP-orig* results are directly comparable to the original test, however *SEP-ext* and *SDP* are not.
  - *SEP-ext* and *SDP* **increase AIQ score representativeness**, the **usage is highly recommended**.

Conclusion

- **Evaluation is important for AGI.**
  - *Universal Intelligence definition* and *AIQ test.*
- **Semantic Analysis identified problems with AIQ test environment programs:**
  - pointless code is abundant
  - simple programs are frequent
  - non-discriminative environments are common
- **Extended version of BF programs sampler for the AIQ test:**
  - reduces proportion of pointless code
  - improves discriminative power

Future Work

- A technical task:
  - **Removing the remaining** types of **pointless code and non-discriminative programs** that need different approach.
- More general questions:
  - *What is the influence of program classes on agents' results?*
  - *How to integrate the results on classes that limit the total achievable rewards into the overall score?*