TinyAPL - Agent Oriented Programming Language Architecture

Vasili Braga, Dumitru Ciorbă, Irina Cojuhari

Abstract

An Agent Oriented Programming Language (AOPL) represents a quite complex structure containing at least 3 parts: an interface, meaning a Domain Specific Language (DSL), an object oriented structure with included knowledge representation and reasoning capabilities (KR&R), and the simulation medium for a Multi-Agent System (MAS). The purpose of this paper is to provide a possible implementation for an Agent Oriented Programming Language Framework.

Keywords: artificial intelligence, intelligent agents, domain specific languages, agent oriented programming, knowledge representation and reasoning.

1. Introduction

Designing an AOPL [1, 2, 5], one must consider the principles of agent-oriented programming, also the known Multi Agent Systems (MAS) architectures with more than 20 years of evolution [4] and the unsolved problems.

The implementation of MAS and intelligent agent's (IA) knowledge representation and reasoning [3] have to be in synergy with same language design and grammar. It can be inappropriate to have different grammar structure for different purposes, even for inter-agent communication [6] the approach must be the same.

This paper will present a common frame for this abstract agent oriented and domain specific language - *tinyAPL* grammar, architecture and implementations tools and mechanisms.

2. Related Work

First step in choosing a grammar begins with the supposition, that the language has to be practical to represent at least three things: object oriented [7, 8] structures, knowledge & reasoning and inter-agent communication, also it must definitely be able to allow event driven programming [9] for the simulation phase.

The biggest conflicts of the early approaches were the common practices of representation for AOPLs, KR&R and Agent Communication Languages (ACL). If for AOPL the main base of inspiration was object oriented languages like C++ or Java, for KR&R and ontologies the common practice was to use XML schema [10] and finally the main concern for inter-agent communication was a semantic agnostic protocol.

Also for the survival and the spreading of the language the *tinyAPL* grammar is imperative to have a low entry barrier for programmers, thus it must be based on very common principles for markup languages. The analysis of latest practices permitted to identify the JSON-LD [11] knowledge representation framework. JSON-LD is initially object oriented and allows a pretty decent ontology representation. For the reasoning capacities this language has to be augmented with framing capabilities [12]: IF_ADDED, IF_NEDDED and for the event oriented approach was added IF_CHANGED and IF_TIMER triggers.

The chosen tool for the parser implementation is the ANTLR [13] (ANother Tool for Language Recognition) - a parser generator for reading, processing, executing, or translating structured text, used to build languages, tools, and frameworks. The C++ language was chosen for the implementation of rendered MAS.

3. Architecture Definition

The *tinyAPL* architecture consists of three big modules: PREPROCESSOR, RENDER and SIMULATION units (Figure1). The PREPROCESSOR unit must prepare all the text for a future parsing: including all files with the libraries, replacing the text from the #DEFINE statements, also it has to make the first syntactic check.

The RENDER unit is definitely more complex, but roughly it consists of two modules: the rendering of the abstract object library and the instantiation of the MAS.

The Simulation module has the purpose of running the simulation for the designed and rendered Multi-Agent System, and it is implementing the reasoning, communication and interactions.

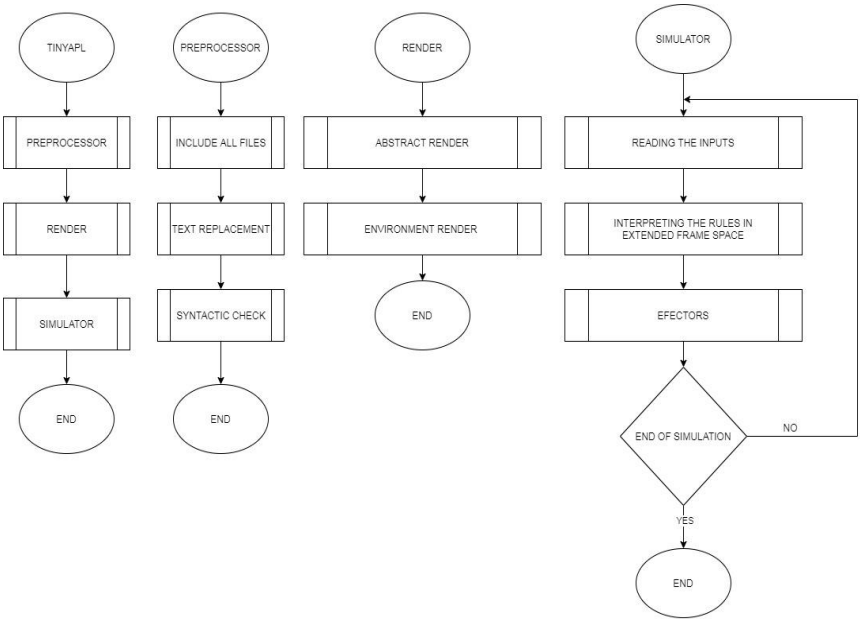

Figure 1. TinyAPL architecture diagram

3.1 The Rendering Unit Architecture

The Rendering unit must transform all parsed tree into linked memory objects, resolve the inheritance issues and build the environment and the communication channels.

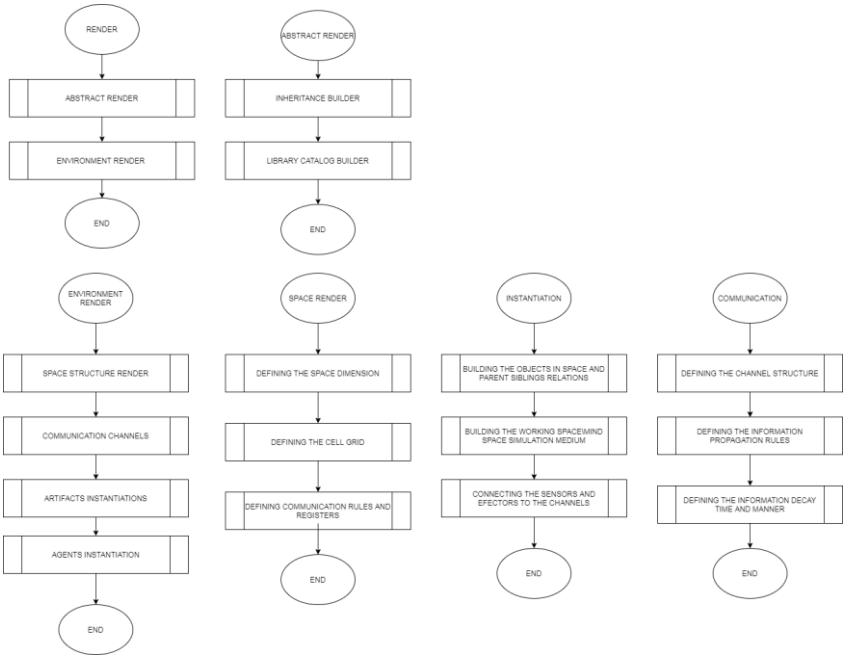

Figure 2. The Rendering unit architecture diagram

4. The Code Example

An example of code can be presented in the following way:

```
{     ABSTRACT :{
      ORGAN: {NAME : olfactive_sensor, PASSIVE & SENSOR,
      MEMORY : { NAME : short_term_memory, DECAY : 10, CLOCK
: 1, VOLUME : 100,
     REGISTER :{NAME : olfactive_history, INTERN, ARRAY [2]:
[10,9], READ & WRITE},
              RULES:{
                    IF_TIMER {
                       this->olfactive_history.push_back(VOLUME)   }
   } }  },
      AGENT: { NAME : being},
      AGENT: { NAME : human, IS_A : being },
      AGENT: { NAME : doctor, IS_A : human }
```

```
},
ENV:{
    AGENT: { NAME : Vasile, IS_A : human, POSITION : {[1,2,3], ABSOLUTE}, ROTATION:{[0,0,0],RELATIVE}},
    ARTIFACT: { NAME : Vasile, IS_A : human, POSITION : {[1,2,3], ABSOLUTE}, ROTATION:{[0,0,0],RELATIVE}},
    AGENT: { NAME : Vasile, IS_A : human, POSITION : {[1,2,3], ABSOLUTE}, ROTATION:{[0,0,0],RELATIVE}}
} }  }
```

5. Future Work and Conclusion

The presented abstract framework for the *tinyAPL* language must be extended for the believable agents [14] representation through the Artificial Endocrine System [15] and an agent society structure [16]. Even so the architecture is quite solid for any kind of intelligent agent system representation and implementation.

Acknowledgments. This work is supported by the 154/23.10.19A project.

References

[1] MB. Van Riemsdijk. *20 Years of Agent-oriented Programming in Distributed AI: History and Outlook*. Proceedings of the 2Nd Edition on Programming Systems, Languages and Applications Based on Actors, Agents, and Decentralized Control Abstractions. New York,USA: ACM; 2012. pp. 7–10.

[2] T. Kosar, PE. Martinez López, PA. Barrientos, and M. Mernik. *A preliminary study on various implementation approaches of domain-specific language.* Information and Software Technology, 2008, pp. 390–405.

[3] M. Gelfond and Y. Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach.* Cambridge University Press, 2014.

[4] V. Mascardi, D. Weyns, A. Ricci, CB. Earle, A. Casals, M. Challenger, et al. *Engineering Multi-Agent Systems: State of Affairs and the Road Ahead.* SIGSOFT Softw Eng Notes. 2019, pp. 18–28.

[5] LS. Sterling and K. Taveter. *Agent Programming Platforms and Languages.* The Art of Agent-Oriented Modeling. The MIT Press, 2009.

[6] T. Finin, R. Fritzson, D. McKay, and R. McEntire. *KQML As an Agent Communication Language*. Proceedings of the Third International Conference on Information and Knowledge Management. New York, NY, USA: ACM; 1994, pp. 456–463.

[7] O. De Troyer and R. Meersman. *A logic framework for a semantics of object oriented data modelling. OOER '95: Object-Oriented and Entity-Relationship Modeling.* Springer Berlin Heidelberg; 1995, pp. 238–249.

[8] A. Ricci and A. Santi. *Concurrent Object-oriented Programming with Agent-oriented Abstractions: The ALOO Approach*. Proceedings of the 2013 Workshop on Programming Based on Actors, Agents, and Decentralized Control. New York, NY, USA: ACM, 2013, pp. 127–138.

[9] C. Elliott. *Declarative event-oriented programming.* Proceedings of the 2nd ACM SIGPLAN international conference on Principles and practice of declarative programming. New York, NY, USA: Association for Computing Machinery; 2000, pp. 56–67.

[10] K. Aberer, T. Catarci, P. Cudré-Mauroux, T. Dillon, S. Grimm,M.-S. Hacid, et al. *Emergent Semantics Systems. Semantics of a Networked World Semantics for Grid Databases*. Springer Berlin Heidelberg, 2004, pp. 14–43.

[11] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, and N. Lindström. *JSON-LD 1.0.* W3C Recommendation, 2014.

[12] R.B. Atman and N.F. Abernethy. *Frame-based knowledge representation system and methods.* US Patent, 6442566, 2002.

[13] T. Parr. *The definitive ANTLR 4 reference.* Pragmatic Bookshelf, 2013.

[14] A. Bryan Loyall. *Believable Agents: Building Interactive Personalities.* Ph.D., School of Computer Science Carnegie Mellon University, 1997.

[15] H. Samani. *Multimodal cognitive processing using artificial endocrine system for development of affective virtual agents*. Труды СПИИРАН,2018, pp. 56–75.

[16] V. Braga. *Un concept de agent credibil în cadrul unei societăți bazat pe sisteme de valori.* In: Ştiinţele socioumanistice şi progresul tehnico-ştiinţific: conf. şt. interuniv., 2019, pp. 166-172.

Vasili Braga, Dumitru Ciorbă, Irina Cojuhari

Technical University of Moldova
E-mails: vasili.braga@ati.utm.md, dumitru.ciorba@fcim.utm.md,
irina.cojuhari@ati.utm.md