Capturing Requirements

From Natural Language to Formal Models

Entalus
Computer Science Labs

www.entalus.com

November 2023

Need. There is a lack of precise, unambiguous, consistent and relative complete requirements for software systems, software libraries, application programming interfaces, software infrastructures such as compilers and operating systems, programming languages, and data formats as well as other computer-related engineering standards. The consequences are dramatic, since the overwhelming majorities of software deficiencies is directly attributable to insufficient requirements [1]. Moreover, deficiencies introduced at early stages of the development lifecycle tend to be most costly to fix, typically in the order of two to three magnitudes [2].

As a result, we are now in a situation where engineering costs of fixing software bugs alone run to two trillion dollar each year in the US alone [3]. Cybercrime [4,5,6], which is thriving on software deficiencies, now is an eight trillion Dollar problem, that is 913 million Dollars an hour, and rising [8]. Software deficiencies also open the door for a new generation of software supply chain attacks, where attackers aggressively implant vulnerabilities directly into dependencies [9], and adversaries increasingly find their way into builds and deployments to deploy rogue software [10].

This sorry state of affairs is not sustainable, and we need to secure the software supply chain through a paradigmatic shift away from prevailing coding, testing, and fixing cycles towards effective and efficient early lifecycle development. The main barrier which needs to be overcome is to precisely capture requirements, which are usually expressed in stylized natural language and/or diagrams, as formal and machine-analyzable models.

Approach. Semantic parsers convert natural language utterances to semantic representations (SR), which are often referred to as logical forms, meaning representations, or programs. These representations are typically executed against an environment (e.g. database, knowledge base) to yield a desired output (e.g. answer to

a question). Semantic parsing can thus be understood as extracting the precise meaning of an utterance, with numerous applications ranging from information retrieval, natural language understanding, human computer interaction, code generation, and the de-hallucination of LLMs. Semantic parsing has a long history going back to the logician R. Montague who famously argued that there is “no important theoretical difference between natural languages and the artificial languages of logicians.” Recent progress in semantic parsing has been fueled, in particular, by the dramatic advances of machine learning technologies.

ARSENAL [11| is a semantic parser generator. Inputs to ARSENAL include (1) a grammar for an SR language, (2) rules for determining the well-formedness of SRs, and (3) a pretty-printer for SRs in natural language. With these inputs, ARSENAL first generates well-formed SRs by sampling the SR grammar and filtering out non-well-formed SRs. Second, ARSENAL paraphrases the sampled well-formed SRs by pretty-printing natural language utterances, Third, ARSENAL uses supervised learning on the generated input-output pairs for learning a semantic parser. This learning of this translation is warm-started with an open-source LLM (e.g. Bart, Llama-2).

Semantic parsers as generated by ARSENAL output (1) a meaning representation, (2) a reformulated natural language utterance of the input expression for supporting users in evaluating whether translations indeed preserve the semantics of the original input sentence, and (3) scores for generated SRs, which are based on the confidence of the trained machine learning model, for measuring the model’s performance, identifying weaknesses in the grammar and trained model, and for improving translation results.

Typical examples on the use of ARSENAL are semantic parsers for capturing the meaning, in linear temporal logic, of industrial requirements specification languages for embedded systems such as EARS [13] or CLEAR [14[. Natural language utterances with their semantic representations can now be analyzed by means of formal analysis tools. For instance, realizability of the given requirements are checked automatically on their logical meaning. And artefacts produced by the formal analysis tools, such as counterexamples, proofs, and unsatisfiable cores for explaining, say, the root cause of non-realizability, are fed back to the human specifier to support him in capturing the intended functionality with a precise, unambiguous, and consistent requirement specification.

ARSENAL has recently also been used for semantically parsing and analyzing the 5G standard, which consists of more than 500 documents. These specifications require support for a number of basic datatypes (e.g. arithmetic, bit expressions) but also conditions (e.g. freshness, acceptability, order), and actions (e.g. produce, send, and request) which are combined into statements (e.g. conditionals, message flow graphs). Moreover, ARSENAL is the backbone of a semantic browser for 5G documents for investigating huge amounts of interrelated requirements specifications and for understanding implications of change.

We have been noticing many times that large parts of industrial specifications are actually executable programs in disguise. In these situations, an ARSENAL semantic

parser might produce PVS programs as target semantic meanings, which themselves are then autocoded into efficient C or Rust code by the PVS2C autocoder. Now, autogenerated code by PVS is *correct-by-construction* with respect to the meaning, as captured by the ARSENAL semantic parser generator, of natural language utterances.

Engineering Support.

- Generating semantic parser for clients' requirements formats and style guides
- Implementing and maintaining client-specific extensions to ARSENAL
- Designing and prototyping suitable formal verification pipelines for formally analyzing clients' natural language requirements
- Conducting case studies on clients' product development to capture and analyze potential benefits
- Designing and prototyping semantic browsers with ARSENAL semantic parsers as core components
- Working out client-specific roadmaps for the sustainable implementation of formal requirements capture in the context of clients' development environments

Contact.

Harald Ruess
Principal Partner, Entalus LLC

email: harald.ruess@entalus.com
phone: +1 (941) 312-2144

References

[1] Lutz, Analyzing SW Errors in Safety-Critical Embedded Systems, CalTech, 1994.
[2] Dabney, *Return on Investment of Independent Verification and Validation Studies*, Phase 2B, NASA IV&V, 2003.
[3] https://www.synopsys.com/blogs/software-security/poor-software-quality-costs-us/
[4] Dragos, *ICS/OT Cybersecurity Year in Review*, 2022.
[5| Carnegie, *Timeline of Cyber Incidents Involving Financial Institutions*, 2022.
[6] DNI Haines, Opening Statement on the 2023 Annual Threat Assessment of the U.S. Intellligence Community, 2023.
[7] https://cybersecurityventures.com/cybercrime-to-cost-the-world-8-trillion-annually-in-2023/
[8] Ghosh, Shalini, et al. *ARSENAL: automatic requirements specification extraction from natural language.* NASA Formal Methods: 8th International Symposium, 2016.
[9] https://techcrunch.com/2022/07/27/protestware-code-sabotage/
[10]https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know
[11] https://github.com/SRI-CSL/arsenal-base
[12] Kamath, Das, *Survey on Semantic Parsing,* Automated Knowledge Base Construction, 2019.
[13] https://alistairmavin.com/ears/

[14] Bhatt, Devesh, et al. *The CLEAR way to transparent formal methods.* 4th Workshop on Formal Integrated Development Environment. 2018.

[15] Montague, Richard. *The proper treatment of quantification in ordinary English.* Approaches to Natural Language, 1973.