Formal Requirements from Structured Natural Language

Jonathan Deneke

Giannakopoulou D., Pressburger T., Mavridou A., Schumann J. (2020) Generation of Formal Requirements from Structured Natural Language. In: Madhavji N., Pasquale L., Ferrari A., Gnesi S. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2020. Lecture Notes in Computer Science, vol 12045. Springer, Cham. https://doi.org/10.1007/978-3-030-44429-7_2

Problem

- Associating requirements with formulas that can be processed by analysis tools
- Ensuring that the formulas conform to the language semantics

Previous Approach

- Requirement writing using specific patterns (SPIDER[5], SpeAR[2], Prospec[3])
- SALT[1] (Structured Assertion Language for Temporal logic) general purpose specification and assertion language
- Using EARS when combined with LTL[6]

Background

- SALT operators
  - Qualifiers: inclusive/exclusive, required/optional
  - Scope: before, after, between
  - Propositional: not, and, or, implies
  - Future Temporal: until, always, eventually, next
  - Past Temporal: since, historically, once, previous

Presented Approach in Paper

Requirements Language

- Fields in a requirement: [scope, condition], component, shall, [timing], response
  - Scope: state dependent behavior
  - Condition: specifies scope
  - Component: part the requirement applies to
  - Shall: similar to EARS
  - Timing: when does response need to occur
  - Response: what the component does or needs to do
- Example:
  - In roll_hold mode ROLLAP shall immediately satisfy abs(roll_angle) < 6 -> roll_hold_reference = 0.
  - When in roll_hold mode when steady_state & calm_air AP shall always satisfy abs(roll_err) ≤ 1.0

Composition Formalization

- Scope interval: (Left, Right) where a requirement must hold
- Scope endpoints: FiM/LiM, FNiM/LNiM, FFiM/FLiM, FTP, LAST
- Baseform: the expectation of the requirement within each scope interval

Verify Formalization

- Trace generator: example executions
- Formula Retriever: set of all possible verification tuples $<t,\Phi_{ft},\Phi_{pt}>$
- Oracle: computes the truth value of t on a trace
- Semantics Evaluator: takes a trace, a verification tuple and the expected value from the oracle and checks if Φ_{ft} and Φ_{pt} evaluate to the expected value of the trace
- Equivalence Checker: takes a tuple and checks if Φ_{ft} and Φ_{pt} are equivalent formulas

Evaluation

- The restricted natural language FRETISH provides a very concrete and comprehensive language structure.
- FRETISH has many of the same advantages seen in EARS plus a few additional ones

References

1. Bauer, A., Leucker, M.: The theory and practice of SALT. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 13–40. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20398-5_3
2. Fifarek, A.W., Wagner, L.G., Hoffman, J.A., Rodes, B.D., Aiello, M.A., Davis, J.A.: SpeAR v2.0: formalized past LTL specification and analysis of requirements. In: NfM 2017, pp. 420–426 (2017)
3. Gallegos, I., Ochoa, O., Gates, A., Roach, S., Salamah, S., Vela, C.: A property specification tool for generating formal specifications: Prospec 2.0. In: SEKE 2008, pp. 273–278 (2008)
4. Giannakopoulou D., Pressburger T., Mavridou A., Schumann J. (2020) Generation of Formal Requirements from Structured Natural Language. In: Madhavji N., Pasquale L., Ferrari A., Gnesi S. (eds) Requirements Engineering: Foundation for Software Quality. REFSQ 2020. Lecture Notes in Computer Science, vol 12045. Springer, Cham. https://doi.org/10.1007/978-3-030-44429-7_2
5. Konrad, S., Cheng, B.H.C.: Facilitating the construction of specification pattern-based properties. In: Proceedings of RE 2005, pp. 329–338. IEEE (2005)
6. Lúcio, L., Iqbal, T.: Formalizing EARS - first impressions. In: 1st International Workshop on Easy Approach to Requirements Syntax (EARS), pp. 11–13 (2018)