

Automatically generating error messages from FOPL – based specifications

Trisha Quan

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
tkq@andrew.cmu.edu

Ciera Christopher Jaspán

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
cchristo@cs.cmu.edu

Jonathan Aldrich

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
jonathan.aldrich@cs.cmu.edu

Abstract

When a system fails to meet its specification, it can be difficult to find the source of the error and how to fix it. Error reporting logic is an algorithm and tool that displays an error message specific to the parts of a specification that contribute to failure. Additionally, ERL uses a heuristic to determine which object in the system is responsible for the error. Results from a small user study suggest that the combination of a more focused error message and a responsible object for the error helps users to find the failure in the system more effectively.

Introduction

First order predicate logic (FOPL) is used by various specifications. However, it is usually difficult to understand error messages produced by FOPL-based specifications. Either they return a generic human-generated error message for the specification, or they simply return the broken specification itself. Error reporting logic (ERL) creates more useful error messages with the following contributions:

- Singling out an object as the failing object in the system
- Providing a directed error message for the part of the specification that actually failed
- Doing this in an automated manner, without human input.

System Description

Our system recursively breaks down a FOPL-based specification until it gets to an *atomic predicate*, a predicate which defines some domain specific specification. By domain, we are referring to any FOPL-based specification system that uses ERL.

ERL breaks down specifications into various sub predicates, which are disjoint by logical connectives to be

searched for the source of error. There are both basic and complicated splits.

An example of a basic split would be “And”. If we were given (x AND y), we would split this into two atomic predicates, x and y , then create error messages for whichever of the two, possibly both, were wrong.

An example of a more complicated split would be “Forall”. For the more complicated splits, we determine which object is responsible for an error by blaming the closest universal quantifier from the point of failure.

Evaluation

We integrated ERL into AcmeStudio, an Eclipse plug-in that allows a user to create Acme architectural specifications and systems. We gave half of our user study participants a version of AcmeStudio with an ERL extension. We then asked all participants to find errors that we created and fix them.

Our quantitative results were inconclusive due to a small sample size. However, our qualitative results were promising. The participants indicated that a more focused message and a responsible object were useful, and that a system like ERL is useful in creating more effective error messages.

Conclusion

Error reporting logic presents a user with a precise error message by automatically parsing the specification to select only the predicates involved in the failure. It uses a heuristic to assign fault to a particular object so that the user receives a direction for the point of failure. Our evaluation results suggest a need for ERL in FOPL specifications.