Minimal Explanations for Unsatisfiability in Mission-Time Linear Temporal Logic (MLTL)

Your Name

September 5, 2025

Contents

List of Figures

List of Tables

Chapter 1

Introduction

- Motivation: challenges in debugging MLTL specifications.
- Problem statement: unsatisfiable specifications are difficult to interpret.
- Thesis goals: create a tool that extracts unsat cores, adapts them to runtime verification, and presents minimal explanations.
- Contributions:
    1. Tool: MLTL Unsat Core Tool.
    2. Method: adaptation of unsat-core extraction to minimal variable+timestep explanations.
    3. HCI: visualization + user study on interpretability.
- Thesis structure overview.

Chapter 2

Background and Related Work

- Mission-Time Linear Temporal Logic (MLTL).
- Unsatisfiable cores: SAT/SMT methods (QuickXplain, Z3, etc.).
- Runtime verification: goals and challenges.
- Visualization and HCI in formal methods tools.

Chapter 3

System Design and Implementation

- Tool architecture: backend solver + frontend (React).
- Input format: traces and specifications.
- Workflow: trace → solver → unsat core.
- Example run with toy problem.

3.1 Overview of the Tool Architecture

This section provides a high-level description of the tool workflow:

$$\text{Input} \rightarrow \text{Parser} \rightarrow \text{QuickXplain} \rightarrow \text{Z3} \rightarrow \text{Output}$$

3.2 Input: Human-Readable MLTL Requirements

- Present syntax and examples of user input formulas.
- Explain why a parser is required: to bridge human-readable MLTL into solver-compatible formats.

3.3 Parsing and Translation

3.3.1 Parser

- Converts human-readable MLTL into an abstract syntax tree (AST).

3.3.2 Unrolling

- Translates the AST into propositional logic over bounded time, suitable for Z3.
- Provide a small worked example:

  - Input: formula in MLTL syntax.
  - Step 1: AST representation.
  - Step 2: Propositional expansion.

3.4 Integration with QuickXplain

- Explain how QuickXplain organizes satisfiability checks (SAT/UNSAT).
- Show pseudocode or a flowchart of the QuickXplain loop.

3.5 Z3 as a Satisfiability Oracle

- Justify the choice of Z3: established, efficient, widely used.
- Describe input requirements: propositional formulas.
- Explain output: Z3 returns either SAT or UNSAT.

3.6 Output: Results to the User

- Case 1: All requirements are satisfiable → return "SAT."
- Case 2: Requirements are unsatisfiable → return unsat core (minimal conflicting subset).

This output sets the stage for Chapter 4 (Methodology), where unsat cores are refined into minimal variable + timestep explanations.

Chapter 4

Methodology: Minimal Explanations

- Problem framing: minimal variables and timesteps.
- Adaptation of existing algorithms to runtime verification.
- Pseudocode for explanation extraction.
- Example walk-through: large trace with conflict at $t = 51$.

Chapter 5

Visualization and Human-Centered Design

- Design goals: reduce cognitive overload, highlight key variables.
- Interface features: variable highlighting, timestep focus.
- Rationale for design choices.
- Screenshots/mockups.

Chapter 6

Evaluation

- Study design: participants, tasks, measures.
- Pilot study results and refinements.
- Main study: results (quantitative and qualitative).
- Analysis of tool effectiveness.

Chapter 7

Discussion

- Summary of findings.
- Lessons for runtime verification tools.
- Limitations of current approach.

Chapter 8

Conclusion and Future Work

- Summary of contributions.
- Implications for verification and HCI.
- Directions for extending this work (scalability, industrial applications).

Bibliography