INTRODUCTION/MOTIVATION

**Goal**
- Formalize High-Quality Natural Language Requirements

**Case: Generic Patient Controlled Analgesic (GPCA) Infusion System**
- Case study from the University of Minnesota (1)
- Sought to generate an ideal Requirements Document
- Intended for as an example document for those who want to study system level requirements
- Features
  - Hierarchy composition of components
  - Macros to capture transition behavior
  - Rising edge behavior on events

**Target Formalization: SpeAR**
- A tool that allows for the generation of formal requirements primarily based on PLTL with natural language-like expressivity.
- Leverages model checking to analyze the composition of formal requirement definition for entailment, consistency, realizability, and traceability

CHALLENGE: FRAME CONDITION

The ability to maintain the prior values of unconstrained free variables in a formal model trace
- Proper frame condition definition is essential in synchronous languages

**Pros**
- Succinct to define behavior
  - “If inputs don’t change the system should maintain current state”
- Easy to understand / assume
  - “Only stimulus can change system state”
- Relative to programming languages
  - Sequential order execution enforces the frame

**Cons**
- Not naturally provided by SpeAR, the user must formally define this

**Solution**
- Flatten model & completely constrain

CONCLUSION

High Quality Natural Language Requirements benefit from formalization and analysis

**Examples**
- Determined the explicit Frame Condition for the system
- Identified the initial mode and transition logic
- Determined a Nondeterministic Transition – This was an Error in the documentation

**Future Work**
- Improve tool support to meet the challenges demonstrated in complex systems

REFERENCES


CHALLENGE: HIERARCHY

Nested state machines representing a cluster of nodes.
- The authors were defining similar transitions that occur on a collection of nodes

**Pros**
- Compact for the user
- Reduces the overall connection definition set
- Easy to read
- Removes the clutter of a number to transitions that define the same behavior

**Cons**
- Forces a number of complex transition behavior to become implicit to the system definition
- No direct representation in SpeAR

**Solution**
- Define explicit triggers in the flattened representation

CHALLENGE: FRAME CONDITION

r_no_change : if (not trigger_a and
not trigger_b and
not trigger_c and
not trigger_d and
not trigger_e and
not trigger_f and
not trigger_g and
not trigger_h and
not trigger_i and
not trigger_m and
not trigger_n )
then state equal to prev_state

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions

Formalize Transitions