Outlines

1. Introduction
2. Background
3. Related work
4. Refinement Maps and Refinement Map Templates
5. Conclusion and future work
Critical devices still have safeness issues.
54 Class-1 recalls on infusion pumps due to software issues.
Testing is not enough!
Formal verification can address testing limitations.
Refinement-based verification is a formal verification technique that has been demonstrated to be effective for verification of software correctness at the object code level.
We have proposed a novel approach to synthesize formal specifications from natural language requirements.

Our verification approach is based on the theory of Well-Founded Equivalence Bisimulation (WEB) refinement.

To overcome the difference between specifications and implementations, WEB refinement uses the concept of a refinement map.
A TS $M = \langle S, R, L \rangle$ is a three tuple in which $S$ denotes the set of states, $R \subseteq S \times S$ is the transition relation that provides the transition between states, and $L$ is a labeling function that describes what is visible at each state.
Definition

WEB Refinement: Let $M = \langle S, R, L \rangle$, $M' = \langle S', R', L' \rangle$, and $r: S \rightarrow S'$. $M$ is a WEB refinement of $M'$ with respect to refinement map $r$, written $M \approx r M'$, if there exists a relation, $B$, such that $\langle \forall s \in S :: sB(r.s) \rangle$ and $B$ is a WEB on the TS $\langle S \uplus S', R \uplus R', L \rangle$, where $L.s = L'(s)$ for $s$ and $S'$ state and $L.s = L'(r.s)$ otherwise.
Background-Synthesis of Formal Specifications

1. **Natural Language Requirement/s**
2. **Enju Parser**
3. **Parse Tree**
4. **APERs**
5. **Formal Model Synthesis Procedure**
6. **APs List**

- **Formal specification (TS)**

Diagram:

- Natural Language Requirement/s
- Enju Parser → Parse Tree
- Parse Tree → APERs
- Formal Model Synthesis Procedure → APs List
- APs List → Formal specification (TS)
Related Work

- Rabiah et al. (2016) developed a reliable autonomous robot system by addressing A* path planning algorithm reliability issue. A refinement process was used to capture more concrete specifications by transforming High-Level specification into equivalent executable program.

- Cimatti (2015) et al. proposed a contract-refinement scheme for embedded systems. The contract-refinement provides interactive composition reasoning, step-wise refinement, and principled reuse refinements for components for the already designed or independently designed components.

- Klein et al. (2014) introduced a new technique called State Transition Diagrams (STD). It is a graphical specification technique that provides refinement rules, each rule defines an implementation relation on STD specification.
Miyazawa (2011) et al. proposed a refinement strategy that supports the sequential C implementations of the state flow charts. The proposed design benefited from the architectural features of model to allow a higher level of automation by retrieving the data relation in a calculation style and rendering the data into an automated system.

Spichkova (2008) proposed a refinement-based verification scheme for interactive real time systems. The proposed work solves the mistakes that rise from the specification problems by integrating the formal specifications with the verification system. The proposed scheme translates the specifications to a higher-order logic, and then uses the theorem prover (Isabelle) to prove the specifications.
Our strategy for constructing the refinement maps is as follows.

- A specification state can be constructed from an implementation state by determining the APs that are true in the implementation state. If a specification has $n$ APs, then we construct one predicate function for each AP.

- The predicate functions take the implementation state as input and output a predicate value that indicates if the AP is true in that state or not.

- The collection of such predicate functions is the refinement map.
BO = \([NB \land (NB_c < NB_m)] \lor [EB \land (EB_c < EB_m)]\)

\(P = P \land (P_c < P_m)\)

\(R = R \land (R_c < R_m)\)

\(BA = [BP_1 \land (BP_{1c} < BP_{1m})] \lor [BP_2 \land (BP_{2c} < BP_{2m})] \lor \ldots \lor [BP_n \land (BP_{nc} < BP_{nm})] \lor [TB \land (TB_c < TB_m)]\)

- BO: Bolus delivery
- P: Prime process
- R: Refill process
- BA: Basal delivery
AI = \([BP_1 \land (BP_{1c} < BP_{1m})] \lor [BP_2 \land (BP_{2c} < BP_{2m})] \lor \ldots \lor [BP_n \land (BP_{nc} < BP_{nm})] \lor [TB \land (TB_c < TB_m)] \lor [NB \land (NB_c < NB_m)] \lor [EB \land (EB_c < EB_m)]\)

CDTC = (\(DT \neq HDT\)) \land (CDTC_c < CDTC_m)

UR = FLAG

RTVR = (CRV \neq HRV) \land (RTVR_c < RTVR_m)

AI: Active Infusion

CDTC: Changing Drug Type and Concentration

UR: User Reminder

RTVR: Reservoir Time and Volume Recomputing
Refinement Map Templates

- Process template.

**Example**

\[
BA = [BP_1 \land (BP_{1c} < BP_{1m})] \lor [BP_2 \land (BP_{2c} < BP_{2m})] \lor \ldots \lor [BP_n \land (BP_{nc} < BP_{nm})] \lor [TB \land (TB_c < TB_m)]
\]

- Projection template.

**Examples**

**UR** = FLAG

- Value change template.

**Example**

\[
CDTC = (DT \neq HDT) \land (CDTC_c < CDTC_m)
\]
Conclusion and future work

NLP

Refinement maps

Develop generic refinement map templates

Refinement-based verification
The End