ARSENAL: Automatic Requirements Specification Extraction from Natural Language

PI: Shalini Ghosh, co-PI: Natarajan Shankar, Supervisor: Patrick Lincoln, Key Personnel: Daniel Elenius
Industrial Collaborator: Wilfried Steiner (TTTech), Student Researcher: Wenchao Li (UC Berkeley), Other Sr. Personnel: Sam Owre

Bridging the Informal/Formal Gap

- *Initial software system designs are often developed in informal natural language*

➡ Facilitates discussion among stakeholders in early design

➡ Leads to confusion, lack of automation, and errors

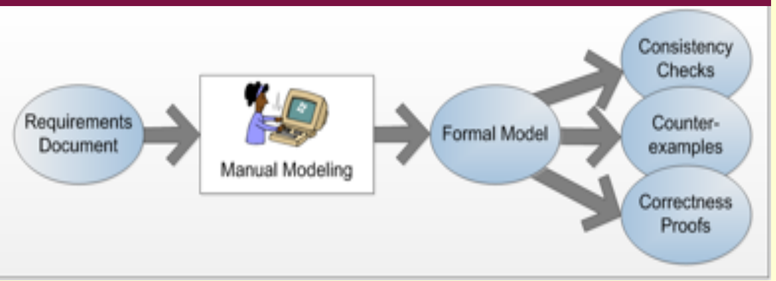
- *Formal design specifications are desirable*

➡ Eliminate ambiguity, allow consistency checking, and facilitate test generation

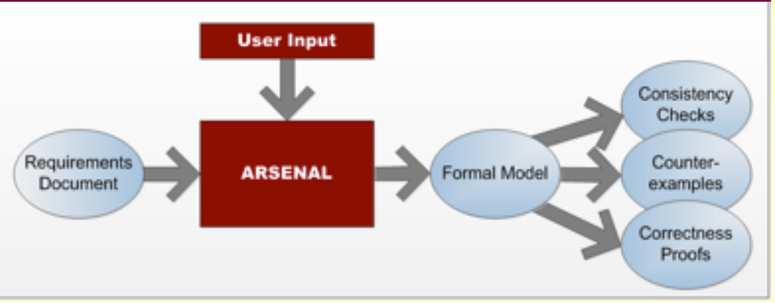
➡ Are more rigorous, and hence more difficult for designers

Goal: ***Bridge the gap between semi-formal natural language requirements and precise formal specifications***.

Requirements Modeling Today



Requirements Modeling with ARSENAL

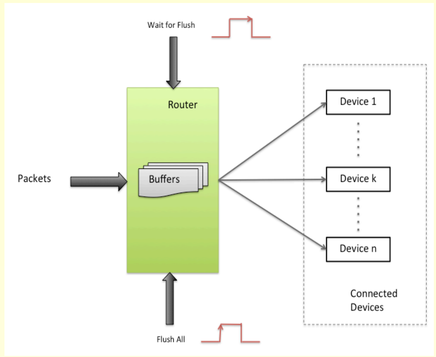


What is ARSENAL?

Robust, scalable, trainable system to

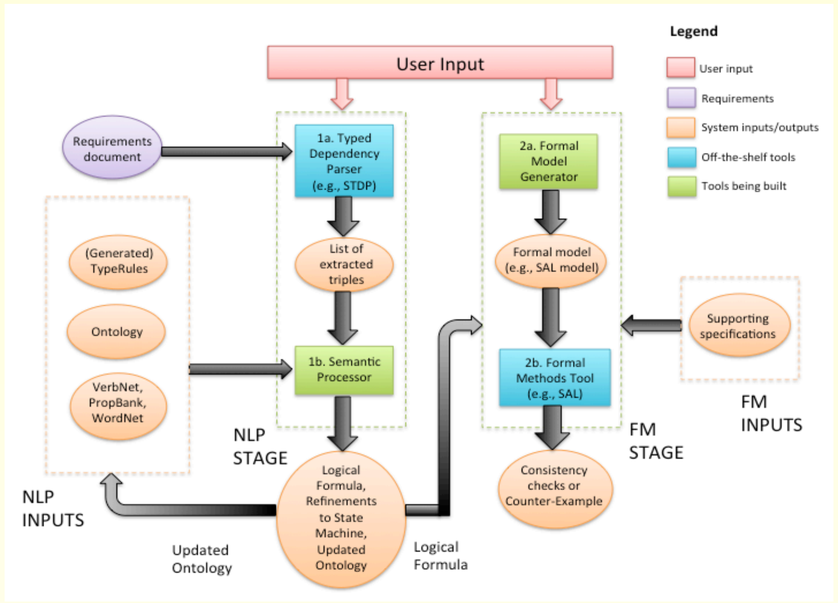
- Extract relevant information from requirements written in semiformal natural language
- Create formal models of requirements
- Facilitate formal analysis of system properties encoded in natural language requirements

Example: Router*



* This is a test example to demonstrate the power of ARSENAL. We are doing actual case studies using the TTTech TT-GbE End System IP requirements document -- it provides functional requirements for an Ethernet-compliant network interface controller IP, one of the solutions used to improve the safety and reliability of networked computer systems in the transportation and industrial segments.

From Requirements to Formulas



Benefits of ARSENAL

- Computational approach to improve requirements engineering
  - Automatic consistency checking within a document
  - Mechanical consistency checks across documents
  - Checks for vacuous assertions
- Enables detection of system issues early in the design cycle
- Facilitates mechanical validation of critical complex systems: Enables automatic checking of designs against requirements
- Enables maintenance of formal specifications throughout the design/build/test/maintain lifecycle, and the checking of those requirements at each stage
- Useful in also capturing business logic, security policies, documentation

Specifications of Router Model

Requirements:

1. If the WaitForFlush signal is asserted, the router shall stop routing packets to connected devices until the FlushAll signal is triggered.
2. If the FlushAll signal is not triggered, buffers shall not be flushed.
3. If the router stops routing packets to connected devices, all buffers shall be flushed.

ARSENAL generated LTL formulas:

1. ([ ] ((assert(WaitForFlush)) -> stop(routing_packets, router, connected_devices)) U (trigger(FlushAll))))
2. ([ ] ((!trigger(FlushAll)) -> (!flush(buffers))))
3. (((stop(routing_packets,router,connected_devices)) -> (flush(buffers))))

Inconsistency detected by ARSENAL:

- State: WaitForFlush is asserted, FlushAll is not triggered.
- Spec 2 => Buffer is not flushed, Spec 3 => Buffer is flushed