Formal Patterns for Medical Safety

Mu Sun, José Meseguer and Lui Sha

University of Illinois at Urbana-Champaign
Motivation

Many medical systems:

1. involve a collection of devices connected to the patient
2. the entire patient-plus-devices system can be viewed as a real-time and cyber-physical system
3. which are safety-critical, with strong qualitative and quantitative requirements.

To gain high assurance about the safety of such systems, formal methods can provide formal executable models of such systems, and formal specification and verification techniques to ensure they meet their safety requirements.
Their distributed features and their real-time nature make medical systems quite complex and hard to design and verify.

Yet their proper functioning and their safety-critical nature makes their verification essential.

One important source of complexity arises from unforeseen interactions between the patient and the different devices, and between the devices themselves.
Methods to reduce medical system complexity and to increase medical system safety are very much needed.

System complexity has many aspects, including the complexity and associated cost of:

- designing
- verifying
- developing
- maintaining and evolving

such systems.
The main goal of this talk is to propose the use of **formal patterns for medical safety** to reduce medical system complexity and increase safety.

By a “formal pattern” I mean a solution to a commonly occurring software problem that is:

1. **as generic as possible**
2. **formally specified**, with precise semantic requirements
3. **executable**, and
4. **comes with strong formal guarantees**.

A formal pattern can be applied to a **potentially infinite** set of concrete instances, where each such instance is **correct by construction** and enjoys the formal guarantees of the pattern.
To develop formal patterns for distributed systems with features such as those mentioned above an appropriate semantic framework is needed, one supporting:

1. concurrency
2. real time behavior
3. executability, and
4. formal verification methods and tools.

I will use rewriting logic as a semantic framework satisfying (1)–(4), and will show in a number of examples its adequacy to specify and verify formal patterns of this nature.
Rewriting Logic and Maude in a Nutshell

Rewriting logic is a flexible logical framework to specify concurrent systems.

- A concurrent system specified as rewrite theory $\mathcal{R} = (\Sigma, E, R)$ where:
  - $\Sigma$ is signature defining the syntax of the system and of its states
  - $E$ is a set of equations defining system’s states as an algebraic data type
  - $R$ is a set of rewrite rules of the form $t \rightarrow t'$, specifying system’s local concurrent transitions.

- Rewriting logic deduction consists of applying rewriting rules $R$ concurrently, modulo the equations $E$.

Maude is a high-performance rewrite engine capable of executing rewrite theories. Maude additionally provides several model checkers and theorem proving tools.
Parameterized theories provide formal models of **generic patterns with interfaces** specifying semantic requirements.

They provide a **formal contract**: if the semantic requirements are met they ensure specific correctness guarantees.

Formally, a **parameterized theory** $B[P]$ with parameter $P$ is a theory inclusion $P \hookrightarrow B$. As a theory transformation it is the function $\lambda H \in (P/Th). B[H]$, where $Th$ denotes the category of theories and theory interpretations, $(P/Th)$ its coslice category for $P$, and $B[H]$ is defined as the pushout:

$$
\begin{array}{ccc}
B[P] & \xrightarrow{H'} & B[H] \\
\uparrow J & & \uparrow J' \\
P & \xrightarrow{H} & T
\end{array}
$$
An Implanted Cardiac Pacemaker:

- has a rate adaptation interface to adjust the patients heart rate during exercise
- must not pace too fast for too long and should allow sufficient resting time between fast pacing periods
- must not change the pacing rate too drastically over time
Patient Controlled Analgesia. A **Morphine Infusion Pump**:

- has an interface to increase morphine injection rate (bolus dose)
- morphine injections must be administered with **sufficient time between doses** and with a specified **maximum number of bolus doses** per hour
A Mechanical Ventilator:

- used on a sedated patient has a pause interface
- but cannot be paused for too long or too often
All the safety properties for the three devices can be captured as a bound on the stress and relax durations of device operation.
The Command Shaper Pattern

Each medical device is wrapped in a command-shaper module that monitors incoming device commands to ensure stress-relax safety.

Formalized in Real-Time Maude and proved correct by Sun, Meseguer and Sha, Proc. WRLA 2010.
The command shaper is a parameterized rewrite theory:

Parameterized Definition of the Command Shaper

Theory of SR-Safety

Instantiated Command Shaper

Device Specific Model of SR-Safety
Since the Command Shaper Pattern is simultaneously a mathematical model and an executable specification, it can be used not only for specification and verification, but also to prototype medical systems.

In an RTRTS 2011 paper Mu Sun and José Meseguer have demonstrated how the Command Shaper Pattern can be used to prototype safe medical systems interacting in real time with various actual medical devices (e.g., an infusion pump), and detailed software models of patient behavior (e.g., heart behavior).