Runtime Verification of Hard Realtime Systems with Copilot: A Tutorial

Ivan Perez

KBR @ NASA Ames Research Center

Alwyn E. Goodloe

NASA Langley Research Center

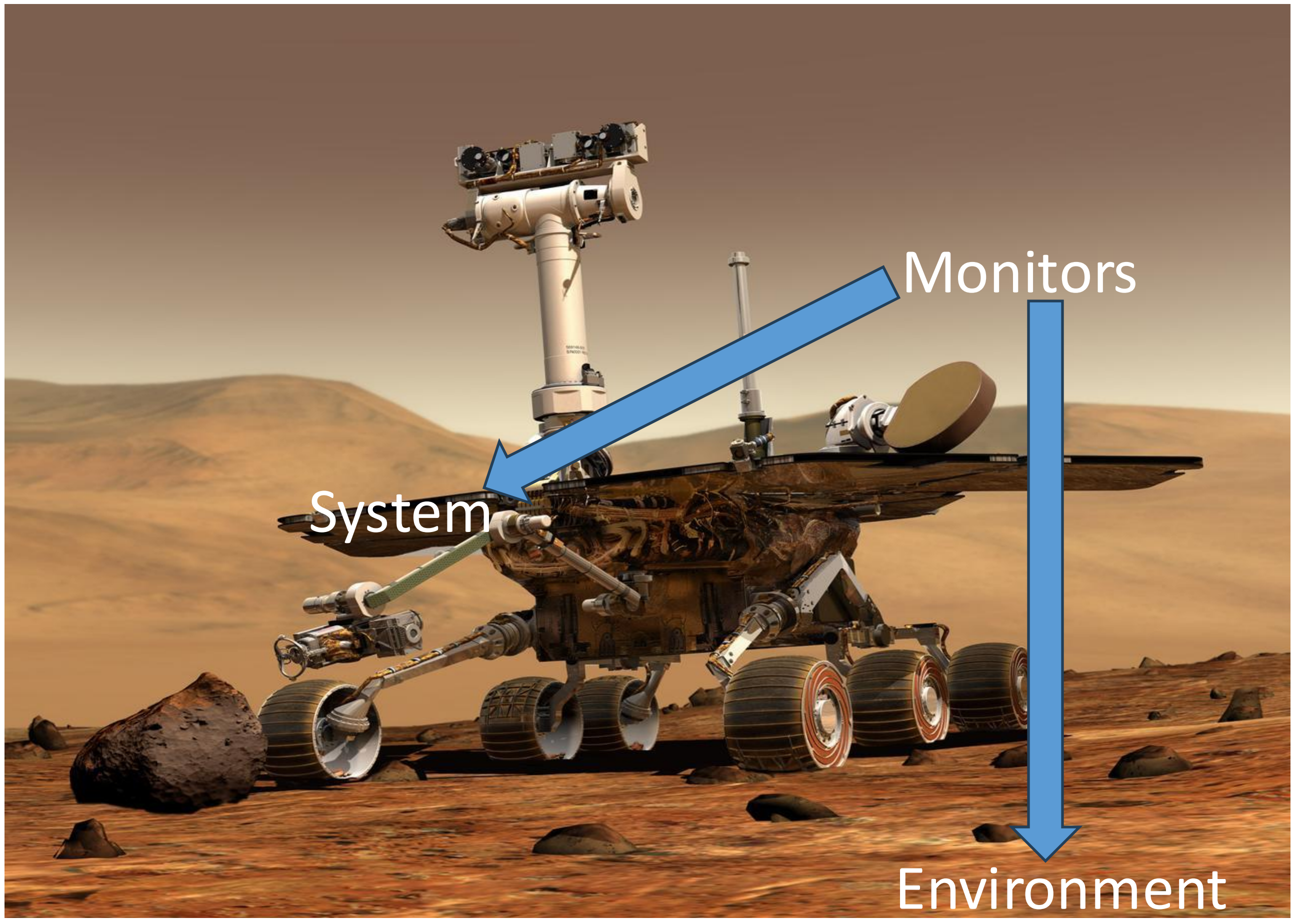
Frank Dedden

System F Computing

RV Motivation

- Formal verification proves a correctness property for every execution of a program correct
    - Most software is too large and requires very specialized workforce
- Testing demonstrates correctness on specific test cases
- Runtime verification (RV) detects if a correctness property is violated during execution and invokes procedures to steer the system into a safe state
    - A form of dynamic system verification

RV in Practice



https://photojournal.jpl.nasa.gov/catalog/PIA04413

Foundations of RV

- Given a specification ϕ of the property we want to check
  - Specification logics: linear temporal logics (LTL), regular expressions, …
- A trace τ of the execution capturing information about the state of a system under observation (SUO)
  - System must be instrumented to capture the trace
- An RV monitor checks for language inclusion $\tau \in \mathfrak{L}(\phi)$
  - Accept all traces admitting ϕ

RV frameworks synthesize monitors from specifications

RV Engineer Checklist

- Specify the property to be checked
- Identify the trace to be captured
- Synthesize a monitor that checks the property using an RV framework
- Create handler that steers the system to a safe state when the property is violated
- Install monitor

Copilot

- Copilot is a language and runtime verification framework targeting hard real-time safety-critical systems
- Stream based specification language similar to Lustre and LOLA
- Employs sampling rather than extensive code instrumentation
  - Appropriate for monitoring safety of CPS systems
- Copilot specifications are translated into MISRA C99 monitors or to BlueSpec and Verilog for implementation in FPGAs
- Effort started in 2008 as a research program
  - Galois and the National Institute of Aerospace (NIA)
- Copilot has evolved into a NASA software engineering tool
  - Adapted NASA Software Engineering development processes
  - Open source
  - Monitors classified as “Mission Support Software” and flown on NASA flights

Copilot Language

- Copilot language implemented as a Haskell Embedded Domain Specific Language (EDSL)
- Users can be productive in Copilot without having to learn Haskell
- Users can write many useful specifications/programs using only a small set of Copilot combinators
- There is an expanding library of predefined combinators to aid in the writing concise specifications
- The Copilot language has been used for general purpose programming of embedded systems
  - Not just for RV

Questions?

Contact Information:

Ivan Perez.
ivan.perezdominguez@nasa.gov

Alwyn Goodloe.
a.goodloe@nasa.gov

Frank Dedden.
frank@systemf.dev