A Formal Embedded Software Verification using Software Fault Tree Analysis

Junbeom Yoo

Division of Computer Science and Engineering
Konkuk University, 1 Hwayang-dong, Gwangjin-gu, Seoul, Republic of Korea
E-mail: jbyoo@konkuk.ac.kr

Extended Abstract

In developing embedded software such as nuclear reactor protection systems (RPS), safety analysis [1] is the process performed in order to guarantee software safety, as well as development and verification processes. Fault tree analysis (FTA) [2] is one of the most widely used safety analysis techniques, often generated and applied manually. Increasing use of formal specification has made it possible to generate software fault trees mechanically, but they all have an intrinsic limitation to the information they can contain. They do not have any information beyond that captured in their specifications or source code. In this extended abstract, we used the generated software fault tree from a different standpoint, verification purpose.

Figure 1 An overview of embedded software verification using software fault trees

Figure 1 depicts an overview of the proposed software verification technique using software fault tree as a starting point of formal verification. We regard the fault tree as an abstract model of the software, containing information only about its root-node (top failure). We therefore can check it against requirements properties quickly and analyze the verification results easily, in comparison with other verification techniques such as model checking [3]. The proposed technique translates the abstract model and properties both into Verilog programs, and performs a formal verification; VIS’s combinational equivalence checking [4]. We used a prototype version of the KNICS RPS [5] in Korean nuclear power plants to demonstrate its effectiveness, and it showed that the mechanically generated fault tree is a good starting point for verifying a software model quickly against requirements properties regarding safety. We are currently focusing on developing a CASE tool, which mechanically generates software fault tree from NuSCR [6] formal requirements specification.
Acknowledgement
This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency NIPA-2010-(C1090-0903-0004)

References

The 5th International Symposium on Embedded Technology (ISET)  
13-14 May, 2010  
The Grand Hotel Daegu, Daegu, Republic of Korea

A Formal Embedded Software Verification using Software Fault Tree Analysis

JUNBEOM YOO
Dependable Software Laboratory
http://dslab.konkuk.ac.kr
KONKUK University

Contents

- Software Fault Tree Analysis

- Formal Verification using Software Fault Tree
  - SFT-to-Verilog Translation
  - Property-to-Verilog Translation
  - VIS Equivalence Checking
    - Case Study: KNICS RPS BP (Ver.00)

- Conclusion

- References
Software Fault Tree Analysis

**Fault Tree Analysis (FTA)**
- One of widely used safety analysis techniques
- Manually construct a fault tree and analyze with
- Quality of FTA
  - Totally depends on experience and knowledge of FTA experts

Software Fault Tree Analysis

**Software Fault Tree Analysis (SFTA)**
- Target: Software
- Software has no wear-out failure as hardware, since reflects developer’s logic.
- Mechanical construction of SFT is recommended.
  - from specification
    - HiP-HOPS (Hierarchically Performed Hazard Origin and Propagation Studies) [1]
    - RIDL (Reliability Imbedded Design Language) [2]
    - NuSCR [3] (our concerns)
    - Statecharts [4,5]
    - RSML [6]
  - from source code
    - Ada83/95 [7,8]
    - FBD [9]

- Intrinsic limitation to the information they contain within
We use mechanically constructed SFTs for verification purpose.

Software Fault Tree Analysis on NuSCR

- **NuSCR** [10]
  - Formal requirements specification language for RPS (Reactor Protection System)
  - Targeting PLC (Programmable Logic Controller) Software

- **Automatic SFT Construction from NuSCR** [3]
In this presentation, I argue that if we want a device to be “smarter”, it needs to not just add more memory and faster CPUs, it needs to be smarter. The most obvious way of doing this is to start processing information in a fashion similar to how a human does. This does not mean that we should ignore advances in human-machine interfaces, or in other technologies such as new ways of doing this is to start processing information in a fashion similar to how a human does. This does not mean that we should ignore advances in human-machine interfaces, or in other technologies such as new ways of doing this is to start processing information in a fashion similar to how a human does. This does not mean that we should ignore advances in human-machine interfaces, or in other technologies such as new ways of doing this is to start processing information in a fashion similar to how a human does.

This vision can be achieved in several ways. One of the end goals is to enable the phone to operate more like a human. One of the most important changes that is required is to progress from task-driven to context-driven behavior, which enables the smart phone to offer new compelling features of different industries and technologies are combined to build a brand new technology or product that contains new features not present in any of the original products used to build it. This is harder than it appears. For example, consider how applications on not just smart phones, but other types of computers, can intelligently transform content for consumption on different devices that have different features. Context-awareness and semantics can also provide seamless content consumption, which can provide appropriate services based on those semantics. Examples include changing avatars and other features can be augmented with specific technologies and vendor data create data “stovepipes” that isolate data, instead of making those data shareable and reusable by other applications.

The largest improvements will lie in applying Knowledge Engineering and Cognitive Science to build converged devices, applications, and services. For example, if the device can learn behavioral patterns of the user, then it can proactively identify short-term context changes to identify changed goals. This in turn enables the device to offer context-sensitive services to provide a more enjoyable experience to the user. If the device can learn behavioral patterns of the user, then it can proactively identify short-term context changes to identify changed goals. This in turn enables the device to offer context-sensitive services to provide a more enjoyable experience to the user. If the device can learn behavioral patterns of the user, then it can proactively identify short-term context changes to identify changed goals. This in turn enables the device to offer context-sensitive services to provide a more enjoyable experience to the user. If the device can learn behavioral patterns of the user, then it can proactively identify short-term context changes to identify changed goals. This in turn enables the device to offer context-sensitive services to provide a more enjoyable experience to the user. If the device can learn behavioral patterns of the user, then it can proactively identify short-term context changes to identify changed goals. This in turn enables the device to offer context-sensitive services to provide a more enjoyable experience to the user. If the device can learn behavioral patterns of the user, then it can proactively identify short-term context changes to identify changed goals. This in turn enables the device to offer context-sensitive services to provide a more enjoyable experience to the user.

The 5th International Symposium on Embedded Technology, Daegu, Korea
The 5th International Symposium on Embedded Technology, Daegu, Korea

Making Smart Phones Smarter

John Strassner*

* Division of IT Convergence Engineering

Pohang University of Science and Technology, Pohang, Republic of Korea
E-mail: johns@postech.ac.kr

Requirements Properties to Verify

Fairness:
1. If the value of input variables is out of bounds, the system should fire a shutdown signal immediately. (th_X_Trip := 0)
2. Only after all conditions resulting in the "out of bounds error" are canceled, it can stop fire the shutdown signal.
3. If the trip condition regarding the operation variable f X is satisfied, it should fire a shutdown signal immediately.
4. If the trip condition regarding the operation variable f X is canceled, it should stop fire the shutdown signal immediately.

Correctness:
1. When it comes into the state "Normal", no shutdown signal should be fired.
2. When it comes into the state "Waiting", no shutdown signal should be fired.
3. When it comes into or remains at the state "Trip", a shutdown signal should be fired.
4. If the trip condition regarding the operation variable f X is canceled, it should stop fire the shutdown signal immediately.

Property-to-Verilog Translation

Translation from Properties into Verilog program

```verilog
module Property_Formula(X, X_Valid, f_Module_error, f_Channel_error, h_X_Selpoint, Current_State, Previous_State, th_X_Trip);
input X_X, X_Valid, f_Module_error, f_Channel_error, h_X_Selpoint, Current_State, Previous_State;
output th_X_Trip;

assign th_X_Trip =
if(conditions)
   (property_formula);
else
   (default_value);
endmodule
```

- If value of input variables is out of bounds, system should fire a shutdown signal immediately.
- Only after all conditions resulting in the "out of bounds error" are canceled, it can stop firing the shutdown signal.
- If the trip condition regarding the operation variable f X is satisfied, it should fire a shutdown signal immediately.
- If the trip condition regarding the operation variable f X is canceled, it should stop firing the shutdown signal immediately.

- When it comes into the state "Normal", no shutdown signal should be fired.
- When it comes into the state "Waiting", no shutdown signal should be fired.
- When it comes into or remains at the state "Trip", a shutdown signal should be fired.
- If the trip condition regarding the operation variable f X is canceled, it should stop firing the shutdown signal immediately.
Making Smart Phones Smarter

John Strassner*

* Division of IT Convergence Engineering
E-mail: johns@postech.ac.kr

The 5th International Symposium on Embedded Technology, Daegu, Korea

11

VIS Equivalence Checking

- **VIS 2.0** [12]
  - Formal verification tool-set
    - Equivalence checking
    - Simulation
    - Model checking
    - Model synthesis
  - Verilog front-end

- **Features**
  - Powerful, but
  - No GUI

VIS Analyzer 3.0 [13]

- **VIS assisting tool**
  - Automating verification processes
  - Visualizing analysis results

The largest improvements will lie in applying Knowledge Engineering and Cognitive Science to build converged devices, applications, and services. For example, if the device can learn behavioral patterns of the user, its environment, and/or business rules change.

The vision can be achieved in several ways. One of the end goals is to progress from

In this presentation, I argue that if we want a device to be "smarter", it needs to

...technology...
Analysis of the Verification Result

- According to the verification result:
  - We could find the fault quickly, which had been found through full-scale verification i.e. model checking.

![Diagram showing SFT-to-Verilog translation](image_url)

**before (with an error)**

**after modification**

---

Conclusion

- We propose a safety-focused verification technique using software fault trees which mechanically generated from formal specifications:
  - Software fault tree is an abstract model of its software specification, containing information only regarding the root-node of the fault tree.
  - We use a SFT as a starting point of formal verification.

- The verification technique consists in:
  - SFT-to-Verilog translation
  - Property-to-Verilog translation
  - VIS equivalence checking

- Case study:
  - A prototype version of KNICS RPS BP introduced in [14]
In this presentation, I argue that if we want a device to be “smarter”, it needs to be able to understand the context of the user and provide services accordingly. This vision can be achieved in several ways. One of the end goals is to make the device more intelligent by providing context-aware services.

Typically, smart phones are also easier to use. The majority feature intuitive user interfaces, and offer more advanced telecommunication capabilities. This differentiates them from “feature phones”. For example, while both a smart phone and a feature phone may offer email, only a smart phone provides features such as address book changes. This is based on the use of an operating system, a set of standard application interfaces. This can range from simple interface customization to task-driven to achieve, not just by adding more memory and faster CPUs. For example, while both a smart phone and a feature phone may offer email, only a smart phone provides advanced telecommunication capabilities. This differentiates them from “feature phones”. For example, while both a smart phone and a feature phone may offer email, only a smart phone provides features such as address book changes. This is based on the use of an operating system, a set of standard application interfaces. This can range from simple interface customization to task-driven to achieve, not just by adding more memory and faster CPUs.

The largest improvements will lie in applying Knowledge Engineering and Cognitive Science to build a more intelligent device. Cognitive science can also provide a compelling user experience. For example, if new context is detected, then a smart phone can provide context-sensitive services to provide a more enjoyable experience to the user.

Context-awareness can help drive a new genre of compelling experiences. For example, a device can attach new services when the needs of the user, its environment, and/or business rules change. This is based on the use of an operating system, a set of standard application interfaces. This can range from simple interface customization to task-driven to achieve, not just by adding more memory and faster CPUs. For example, while both a smart phone and a feature phone may offer email, only a smart phone provides features such as address book changes. This is based on the use of an operating system, a set of standard application interfaces. This can range from simple interface customization to task-driven to achieve, not just by adding more memory and faster CPUs. For example, while both a smart phone and a feature phone may offer email, only a smart phone provides features such as address book changes. This is based on the use of an operating system, a set of standard application interfaces. This can range from simple interface customization to task-driven to achieve, not just by adding more memory and faster CPUs.

Thus, a smart phone is more than just a phone. It is a device that can understand its environment and provide services accordingly. This can include providing context-sensitive services, such as offering information about nearby restaurants or supermarkets. This can be achieved by using knowledge engineering and cognitive science to build a more intelligent device.

References