Nuclear Engineering and Technology 54 (2022) 1635–1643

Contents lists available at ScienceDirect

Nuclear Engineering and Technology

journal homepage: www.elsevier.com/locate/net

Original Article

A formal approach to support the identification of unsafe control actions of STPA for nuclear protection systems

Sejin Jung, Yoona Heo, Junbeom Yoo*

Konkuk University, Republic of Korea

ARTICLE INFO

Article history:
Received 19 July 2021
Received in revised form
13 September 2021
Accepted 21 October 2021
Available online 29 October 2021

Keywords:
Hazard analysis
STPA
Unsafe control action
Formal software requirements specification
Software fault tree analysis

ABSTRACT

STPA (System-Theoretic Process Analysis) is a widely used safety analysis technique to identify UCAs (Unsafe Control Actions) resulting in potential losses. It is totally dependent on the experience and ability of analysts to construct an information model called *Control Structures*, upon which analysts try to identify unsafe controls between system components. This paper proposes a formal approach to support the manual identification of UCAs, effectively and systematically. It allows analysts to mechanically extract *Process Model*, an important element that makes up the *Control Structures*, from a formal requirements specification for a software controller. It then concisely constructs the contents of *Context Tables*, from which analysts can identify all relevant UCAs effectively, using a software fault tree analysis technique. The case study with a preliminary version of a Korean nuclear reactor protections system shows the proposed approach's effectiveness and applicability.

1. Introduction

Safety-critical systems such as nuclear power plants should prove safe from the identified hazards before in-operation [1], and safety demonstration is required by government authorities and international standards [2–4]. Hazard analysis [5] is a widely used technique for identifying potential hazardous state of a system systematically. FTA (Fault Tree Analysis), FMEA (Failure Mode and Effect Analysis) and HAZOP (Hazard And OPerability) are traditional techniques based on cause-effect relationships. On the other hand, STPA (System-Theoretic Process Analysis) [6] focuses on unsafe controlling interactions between system components, based on the STAMP model [7]. STPA is now widely used in hazard analysis of various safety-critical systems in industry.

STPA works in 4 steps: (1) define the purpose of the analysis, (2) model the control structure, (3) identify unsafe control actions (UCA), and (4) identify loss scenarios [7]. It identifies hazardous controls, i.e., UCA on system components, which violate safety constraints, from control structure models. A control structure, therefore, plays the most important role in STPA. It encompasses a hierarchy of control loops consisting of various controlling and controlled elements, and each element includes process models to cope with others precisely. But constructing a control structure, including process models, is all about safety analysts understanding of target systems and specifications and based on their own experiences and abilities. It often takes the most time in the entire STPA process.

Several studies [8–10] have been proposed to help STPA analysts make the control structure easier [9]. uses monitored and controlled functions of the Asmeta abstract state machine to identify process model of the controller, and [10] proposes a method of extracting process model from the Parnas's Four-Variable model. They all use formal models to obtain insights to construct control structures, but they also need to build the formal models additionally. This paper assumes a situation in which a formal requirements specification is prepared.

This paper provides a formal approach to construct process models of control structures and context tables for identifying UCAs effectively and systemically. It uses the *NuSCR* [11] formal requirement specifications and the *NuFTA* [12] software fault tree analysis. *NuSCR* is a formal requirements specification language for safety-level digital controllers in nuclear power plants, in full support with a visual tool-set *NuDE* [13] to extract process models mechanically. The process model generated from the *NuSCR*, however, often results in too many irrelevant combinations in context tables to run the analysis in time [14]. We use the software fault tree analysis technique and tool *NuFTA* to reduce the

* Corresponding author. C386, Engineering Building, Seoul, 05029, Republic of Korea.
E-mail address: jbyoo@konkuk.ac.kr (J. Yoo).

https://doi.org/10.1016/j.net.2021.10.033

combinations into a valid scope. It performs the backward analysis on the specifications and produces minimal cut-sets. From the context tables, analysts can identify relevant UCA effectively. This paper also provides a systematic process that help analysts apply this formal approach to STPA analysis.

We performed a case study with a preliminary version of an RPS (Reactor Protection System) in a Korean nuclear power plant. It developed a formal requirements specification in *NuSCR* [15] for the purpose of design diversity, and the case study showed that the proposed formal approach to construct control structures of STPA analysis is effective and applicable. The paper is organized as follows. Section 2 briefly overviews STPA, *NuSCR* and *NuFTA* as a background. Section 3 explains the proposed formal approach to construct control structures of STPA, and Section 4 describes the case study. Related studies to assist STPA analysis are summarized in Section 5. Section 6 concludes the paper and provides remarks on future research directions.

2. Background

2.1. STPA

STPA (System-Theoretic Process Analysis) [6] is a hazard analysis technique, based on STAMP (System-Theoretic Accident Model and Process) [1]. The system is viewed as a hierarchical structure that higher-level components *control* lower-level components and lower-level components *feedback* to higher level components according to the STAMP "*control structure*" model. STPA focuses on identifying hazardous controls (*i.e., UCA*) between the controlling and controlled components as described in Fig. 1.

Unsafe control action (UCA) is a control action that may lead to a hazard in a particular context and environment. UCAs are classified in 4 types – '*Not providing causes hazard*,' '*Providing causes hazard*,' '*Too early, too late, out of order*,' and '*Stopped too soon, applied too long*.' A context table is used to analyze and identify UCAs. It consists of control actions, types in 4 categories, and "*context*" which is a specific set of process model variables/values, leading the system into a hazardous state. Safety analysts then identify various loss scenarios with the context table.

2.2. NuSCR

NuSCR [11] is a formal software requirements specification language, tailored for control software in nuclear power plants. It consists of 4 elements, such as FOD (Function Overview Diagram), FSM (Finite State Machine), TTS(Timed Transition System) and SDT (Structured Decision Table). <Fig. 2> depicts a part of the formal SRS (Software Requirements Specification) [15] for a preliminary version of the KNICS RPS BP (Bistable Processor) in Korea. The FOD was modeled with the *NuSCR* CASE tool [13], and <Fig. 2 (a)> is a FOD of the *g_LO_SG1_LEVEL* module, which is '*fixed set-point falling trip logic*' for steam generator #1 in the RPS BP. <Fig. 2 (b)> is the TTS model for the *th_LO_SG1_LEVEL_Trip_Logic* node. It models the trip (*i.e.,* shutdown of nuclear reactors) conditions for the input sensor variables with timing constraints. <Fig. 2 (c)> is the SDT model for the *g_LO_SG1_LEVEL_Trip_Out* module. It decides to fire the shutdown signal out.

2.3. NuFTA

Fault tree analysis (FTA) [5] is a top-down, deductive failure analysis technique about how the undesired event of a system occurred, by using a Boolean logic combination of lower-level basic events. *NuFTA* is a software fault tree analysis (SFTA) technique and tool [12] to perform fault tree analysis on *NuSCR* formal requirements specifications and calculate minimal cut-sets (MCS) for top failures/accidents. <Fig. 3> shows the fault tree, whose top event is "*f_LO_SG1_LEVEL_Trip_Out* = *True*." It means the condition "*When the shutdown signal fires*."

It also shows a full set of MCS which lead to the *shutdown* event, 106 in total. They are all analyzed, calculated and displayed in the *NuFTA*. This paper uses the *NuFTA* analysis and MCS calculations to select meaningful combinations of process model variables in the STPA context tables. In addition to, the use of MCSs can generate combinations of process model variables which have conditions related to timing-constraints that occur by TTS in the *NuSCR* semi-automatically. Such conditions are important to identify timing-related conditions of hazardous actions in STPA, but it is difficult to identify those conditions with simple combinations of variables. It also can reduce the combinations of process model variables into a valid scope to analysis compared with previous studies [14].

3. A formal approach to support identifying unsafe control actions

The proposed formal approach to support identifying unsafe control actions of STPA consists of 5 steps as described in <Fig. 4>. It supports two steps of the STPA, which are the STPA step 2 and step

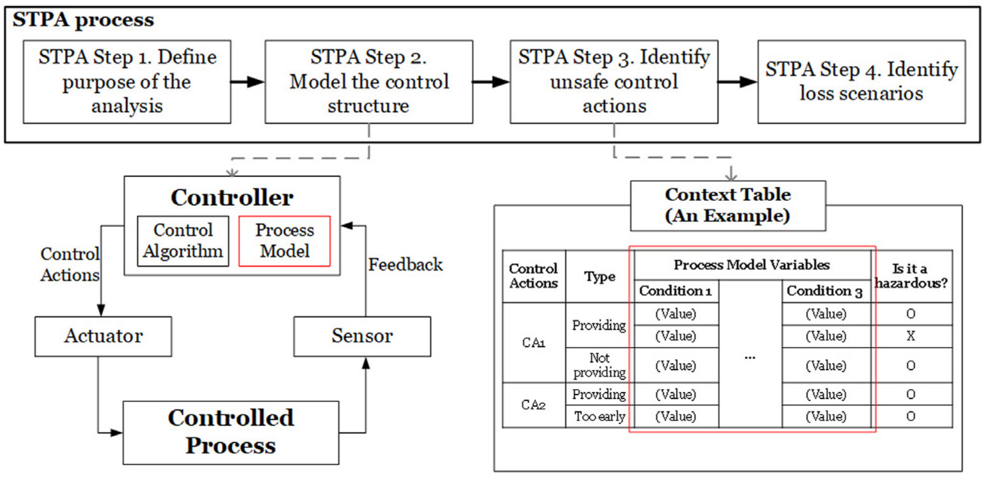
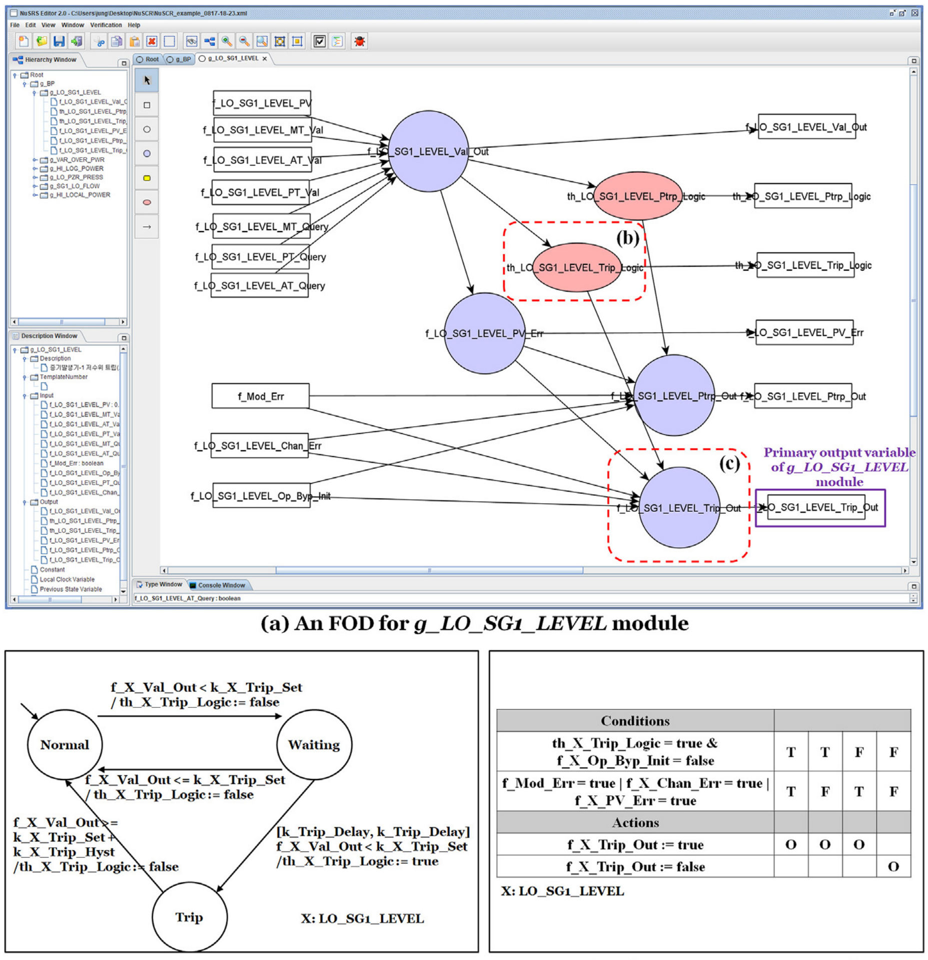


Fig. 1. A typical control structure of STPA.



f_LO_SG1_LEVEL_PV
f_LO_SG1_LEVEL_MT_Val
f_LO_SG1_LEVEL_AT_Val
f_LO_SG1_LEVEL_PT_Val
f_LO_SG1_LEVEL_MT_Query
f_LO_SG1_LEVEL_PT_Query
f_LO_SG1_LEVEL_AT_Query
f_LO_SG1_LEVEL_Val_Out
th_LO_SG1_LEVEL_Ptrp_Logic
th_LO_SG1_LEVEL_Trip_Logic
f_LO_SG1_LEVEL_PV_Err
f_Mod_Err
f_LO_SG1_LEVEL_Chan_Err
f_LO_SG1_LEVEL_Op_Byp_Init
f_LO_SG1_LEVEL_Ptrp_Out
f_LO_SG1_LEVEL_Trip_Out
(b)
(c)
Primary output variable of g_LO_SG1_LEVEL module
f_X_Val_Out < k_X_Trip_Set / th_X_Trip_Logic := false
Normal
Waiting
f_X_Val_Out <= k_X_Trip_Set / th_X_Trip_Logic := false
f_X_Val_Out >= k_X_Trip_Set + k_X_Trip_Hyst /th_X_Trip_Logic := false
[k_Trip_Delay, k_Trip_Delay] f_X_Val_Out < k_X_Trip_Set /th_X_Trip_Logic := true
Trip
X: LO_SG1_LEVEL

Conditions				
th_X_Trip_Logic = true & f_X_Op_Byp_Init = false	T	T	F	F
f_Mod_Err = true | f_X_Chan_Err = true | f_X_PV_Err = true	T	F	T	F
Actions				
f_X_Trip_Out := true	O	O	O	
f_X_Trip_Out := false				O

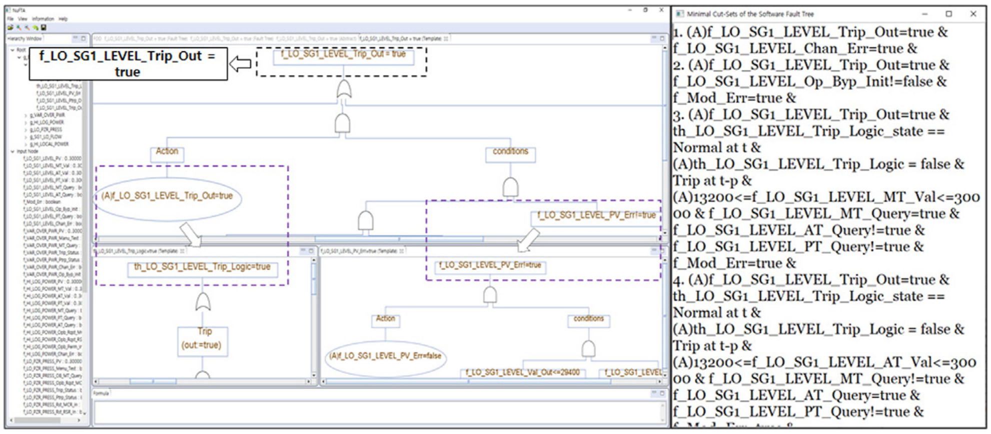
X: LO_SG1_LEVEL

(a) An FOD for *g_LO_SG1_LEVEL* module

(b) A TTS for *th_LO_SG1_LEVEL_Trip_Logic* node

(c) A SDT for *f_LO_SG1_LEVEL_Trip_Out* node

Fig. 2. A part of *NuSCR* specification of *g_LO_SG1_LEVEL* module.



f_LO_SG1_LEVEL_Trip_Out = true
f_LO_SG1_LEVEL_Trip_Out = true
Action
conditions
(A)f_LO_SG1_LEVEL_Trip_Out=true
f_LO_SG1_LEVEL_PV_Err!=true
th_LO_SG1_LEVEL_Trip_Logic=true
f_LO_SG1_LEVEL_PV_Err=true
Trip (out=true)
Action
conditions
(A)f_LO_SG1_LEVEL_PV_Err=false
f_LO_SG1_LEVEL_Val_Out<=29400
Minimal Cut-Sets of the Software Fault Tree

1. (A)f_LO_SG1_LEVEL_Trip_Out=true &
f_LO_SG1_LEVEL_Chan_Err=true &
2. (A)f_LO_SG1_LEVEL_Trip_Out=true &
f_LO_SG1_LEVEL_Op_Byp_Init!=false &
f_Mod_Err=true &
3. (A)f_LO_SG1_LEVEL_Trip_Out=true &
th_LO_SG1_LEVEL_Trip_Logic_state ==
Normal at t &
(A)th_LO_SG1_LEVEL_Trip_Logic = false &
Trip at t-p &
(A)13200<=f_LO_SG1_LEVEL_MT_Val<=300
00 & f_LO_SG1_LEVEL_MT_Query=true &
f_LO_SG1_LEVEL_AT_Query!=true &
f_LO_SG1_LEVEL_PT_Query!=true &
f_Mod_Err=true &
4. (A)f_LO_SG1_LEVEL_Trip_Out=true &
th_LO_SG1_LEVEL_Trip_Logic_state ==
Normal at t &
(A)th_LO_SG1_LEVEL_Trip_Logic = false &
Trip at t-p &
(A)13200<=f_LO_SG1_LEVEL_AT_Val<=300
00 & f_LO_SG1_LEVEL_MT_Query!=true &
f_LO_SG1_LEVEL_AT_Query=true &
f_LO_SG1_LEVEL_PT_Query!=true &

Fig. 3. A fault tree generated by the *NuFTA* for the reactor shutdown condition.

3, with NuSCR formal specifications. The proposed process first extracts relevant information for "process models" with <Algorithm 1>, and then generates valid combinations of process model variables for "context tables" with <Algorithm 2>, with the support of the formal specification method NuSCR in NuDE [11,13] and the software fault tree analysis method *NuFTA* [12]. The proposed approach finally produces a set of context tables for supporting the identifying unsafe control action step.

- (Step 1) Identify output variables concerning control actions from *NuSCR* formal specifications (NuSCR in NuDE)

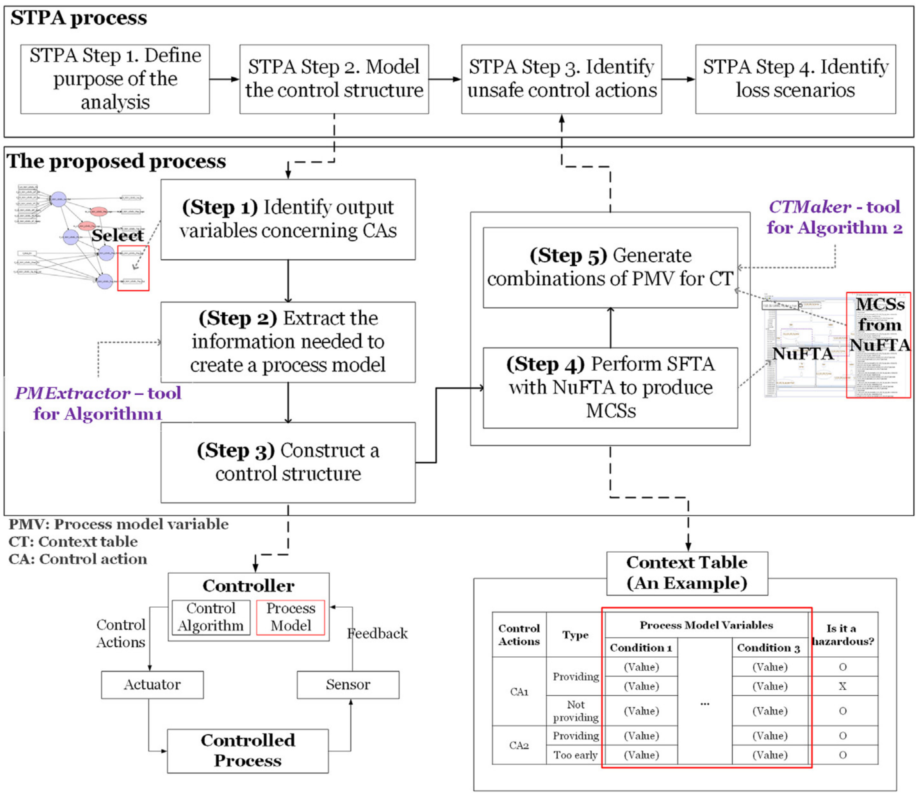

STPA process
STPA Step 1. Define purpose of the analysis
STPA Step 2. Model the control structure
STPA Step 3. Identify unsafe control actions
STPA Step 4. Identify loss scenarios
The proposed process
Select
(Step 1) Identify output variables concerning CAs
(Step 2) Extract the information needed to create a process model
(Step 3) Construct a control structure
(Step 4) Perform SFTA with NuFTA to produce MCSs
(Step 5) Generate combinations of PMV for CT
PMExtractor – tool for Algorithm1
CTMaker - tool for Algorithm 2
NuFTA
MCSs from NuFTA
PMV: Process model variable
CT: Context table
CA: Control action
Controller
Control Algorithm
Process Model
Control Actions
Feedback
Actuator
Sensor
Controlled Process
Context Table (An Example)
Control Actions
Type
Process Model Variables
Condition 1
Condition 3
Is it a hazardous?
CA1
CA2
Providing
Not providing
Too early
(Value)
...
O
X

Fig. 4. A formal approach to constructing control structures and context tables of STPA.

- (Step 2) Extract the information needed to create a process model with "*PMExtractor*" Tool (Algorithm 1)
- (Step 3) Construct a control structure
- (Step 4) Perform SFTA of NuSCR with *NuFTA* to produce MCSs for the output variables identified (*NuFTA*)
- (Step 5) Generate valid combinations of process model variables for context tables with "CTMaker" Tool (Algorithm 2)

(Step 1) The first step is to discern a few relevant output variables, which might be related or connected to control actions, from all output variables in the NuSCR formal requirements specification. The NuDE environment visibly illustrates the formal specification as shown in <Fig. 2>, and analysts choose appropriate output variables based on the content of SRS.

(Step 2) We then extract valuable information needed to create process models with the support of PMExtractor which implements < Algorithm 1> below. It checks all variables and internal states in FODs that correspond to the selected output variables, recursively. Variables and internal states which do not have direct effect on output variables are excluded in this step.

Algorithm 1. Extracting Variable Information

```
Algorithm 1 Extracting Variable Information
Input: FOD: one FOD for extracting variables, connectedVarSet: a set of extracted variables, TargetVariable:
selected variables to check FOD
1: for all Node n∈FOD.getAllVariableNodes() do
2:     if checkTarget(n.target, TargetVariable) is true then
        //check if the target node of n is the same as TargetVriable
3:         connectedVarSet.add(n.name)
4:         if n is FSM || TTS then
5:             connectedVarSet.add(n.name+state)
6:         end if
7:         ExtractingVariableInformation(FOD, connectedVarSet, n)
8:     end if
9: end for
```

(Step 3) An expert constructs a "control structure" with the information about variables and internal states, obtained from the previous step. The information is useful in defining "process models." The analysts play a particularly important role at this step, since it is totally based on the experience and ability of the analysts. Some abstraction often requires to do construct an efficient control structure with process models.

(Step 4) From the control structure constructed, the analysts identify UCAs from context tables. As the table often includes too much irrelevant information for the analysts to do it in time, this paper uses software fault tree analysis method NuFTA to construct valid and compact context tables. From the NuSCR formal requirements specification in NuDE environment, NuFTA constructs software fault trees for each output variable identified in (Step 1) and calculates a set of minimal cut-set (MCS) - logic formula which are only composed of relevant variables and states with their values, mechanically.

(Step 5) We then construct an appropriate size of context tables from the MCS formula with the support of CTMaker which implements <Algorithm 2> below. It inputs MCSs, a process model in control structures, and the selected output variables, and then constructs a context table consisting of only the appropriate process model variables and values for the UCA analysis. The analysts can decide whether the control action is hazardous or not in a certain condition/situation which the context table provides.

Algorithm 2. Generating Context Table

```
Algorithm 2 Generating Context Table
Input: MCS: a set of generated MCSs, PM: process model, OutputVariable: an output variable related to control
action
1: CT = createContextTable(PM)
2: for all m∈MCS do
3:     split whole formula of m into sets of items that express variable and values each to mitem
4:     CT.addNewRow()
5:     if m.topEventVariableValue is true then //Generally, output variable and top-event condition of SFT
7:         set CT.currentRow is UCA case 1, 3, or 4
8:     else if m.topEventVariableValue is false then
9:         set CT.currentRow is UCA case 2
10:    end if
11:    for all mitem∈m do
12:        if mitem.variable.checkExist(PM) is true then
13:            add value of mitem.variable into the CT.current
14:        end if
15:    end for
16: end for
17: for all c∈CT do
18:    if c.UCAcase == 3 && c.checkTimedConstraint() is false then
19:        remove c from CT
20:    end if
21: end for
```

4. Case study

This section introduces the case study we performed upon a preliminary version of RPS (Reactor Protection System) BP (Bistable Processor) in a Korean nuclear power plant. The project had developed a formal requirements specification in NuSCR [15] for the purpose of design diversity. We applied the proposed formal approach to the typical STPA analysis on the RPS BP, and demonstrated the effectiveness and applicability of the proposed method.

4.1. The target system

The target system of the case study is an initial version of the BP software module in the RPS in Korea [15]. The BP software consists of 18 independent modules corresponding to 18 sensor inputs. If a trip (i.e., reactor shutdown) condition for each module is satisfied, then the module fires a trip signal immediately. The g_LO_SG1_LEVEL module in <Fig. 2> is an example of the module at the category of fixed set-point falling trip logic. The case study uses the module to perform the STPA analysis and apply the proposed formal approach to identify unsafe control actions in STPA.

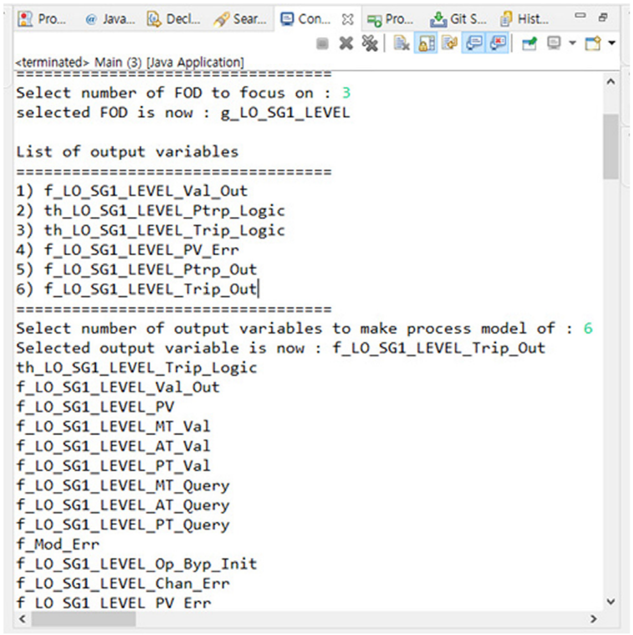
4.2. The STPA results

We performed the STPA analysis on the g_LO_SG1_LEVEL module in <Fig. 2> according to the process illustrated in <Fig. 4>.

[**STPA Step 1**] This step starts the STPA analysis with identifying system-level accidents and hazards of the RPS.

- System-level accidents
  - A-1. People injured or killed
  - A-2. Environment contaminated
  - A-3. Equipment damage
  - A-4. Loss of electrical power generation
- System-level hazards
  - H-1. Release of radioactive materials
  - H-2. Reactor temperature too high
  - H-3. Equipment operated beyond limits
  - H-4. Reactor shutdown

[**STPA Step 2**] The STPA analysts then construct a "control structure" for the RPS, based on their experiences and abilities. The additional 3 steps which this paper proposes help the analysts avoid doing it from scratch.



```
=================================
Select number of FOD to focus on : 3
selected FOD is now : g_LO_SG1_LEVEL

List of output variables
=================================
1) f_LO_SG1_LEVEL_Val_Out
2) th_LO_SG1_LEVEL_Ptrp_Logic
3) th_LO_SG1_LEVEL_Trip_Logic
4) f_LO_SG1_LEVEL_PV_Err
5) f_LO_SG1_LEVEL_Ptrp_Out
6) f_LO_SG1_LEVEL_Trip_Out
=================================
Select number of output variables to make process model of : 6
Selected output variable is now : f_LO_SG1_LEVEL_Trip_Out
th_LO_SG1_LEVEL_Trip_Logic
f_LO_SG1_LEVEL_Val_Out
f_LO_SG1_LEVEL_PV
f_LO_SG1_LEVEL_MT_Val
f_LO_SG1_LEVEL_AT_Val
f_LO_SG1_LEVEL_PT_Val
f_LO_SG1_LEVEL_MT_Query
f_LO_SG1_LEVEL_AT_Query
f_LO_SG1_LEVEL_PT_Query
f_Mod_Err
f_LO_SG1_LEVEL_Op_Byp_Init
f_LO_SG1_LEVEL_Chan_Err
f_LO_SG1_LEVEL_PV_Err
```

Fig. 5. Extracted variables for f_LO_SG1_LEVEL_Trip_Out variable by PMExtractor.

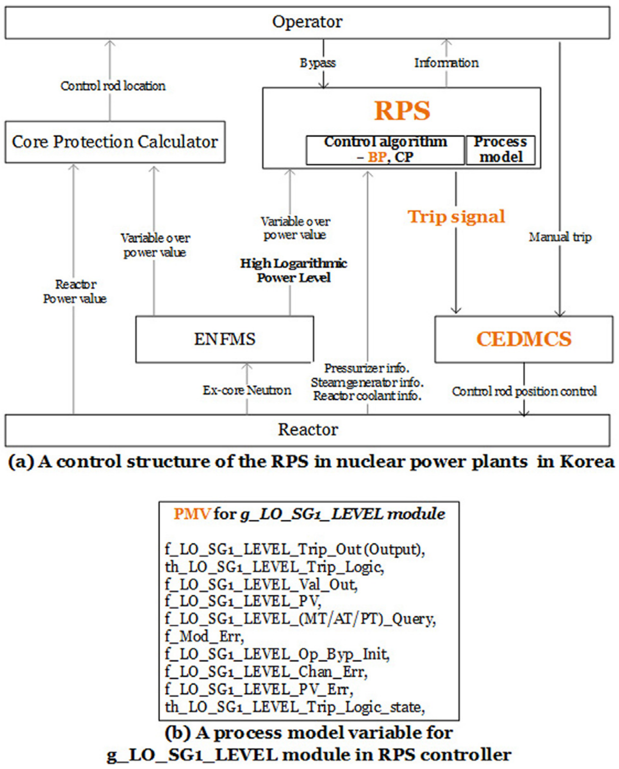

Operator
Control rod location
Bypass
Information
RPS
Core Protection Calculator
Control algorithm – BP, CP
Process model
Trip signal
Manual trip
Variable over power value
Variable over power value
High Logarithmic Power Level
Reactor Power value
ENFMS
CEDMCS
Pressurizer info.
Steam generator info.
Reactor coolant info.
Ex-core Neutron
Control rod position control
Reactor
(a) A control structure of the RPS in nuclear power plants in Korea
PMV for g_LO_SG1_LEVEL module
f_LO_SG1_LEVEL_Trip_Out (Output),
th_LO_SG1_LEVEL_Trip_Logic,
f_LO_SG1_LEVEL_Val_Out,
f_LO_SG1_LEVEL_PV,
f_LO_SG1_LEVEL_(MT/AT/PT)_Query,
f_Mod_Err,
f_LO_SG1_LEVEL_Op_Byp_Init,
f_LO_SG1_LEVEL_Chan_Err,
f_LO_SG1_LEVEL_PV_Err,
th_LO_SG1_LEVEL_Trip_Logic_state,
(b) A process model variable for g_LO_SG1_LEVEL module in RPS controller

Fig. 6. A control structure of the RPS in nuclear power plants in Korea.

- (Step 1) “Identify output variables concerning control actions”

The BP is the primary logic for deciding a trip signal, and we identified f_LO_SG1_LEVEL_Trip_Out as an output variable related to determining the occurrence of control actions, as marked in <Fig. 2 (a)>.

- (Step 2) “Extract the information needed to create a process model”

We extract the necessary information through PMExtractor. It reads the selected output variable (f_LO_SG1_LEVEL_Trip_Out) and extracts 13 (related) inputs and internal variables automatically from the NuSCR formal requirements specifications as shown in <Fig. 5>.

- (Step 3) “Construct a control structure”

The analysts then construct a control structure using the output variable and the process model as shown in <Fig. 6>. They combined the 3_Query variables into one for convenience, deleted 3_Val variables, and add the th_LO_SG1_LEVEL_Trip_state variable for storing internal state information.

The control structure consists of several components such as operator, RPS, CEDMCS (Control Element Drive Mechanism Control System), and ENFMS(Ex-core Neutron Flux Monitoring System). The RPS provides a control action (i.e., a trip signal) to the CEDMCS when a safety-related variable reaches its set-point. CEDMCS controls control rods' position according to the trip signal from RPS or manual operations from the operator either. <Fig. 6 (b)> shows the process model variables for g_LO_SG1_LEVEL module. They are abstracted and modified manually by the analysts from which were extracted by PMExtractor at the previous step.

[**STPA Step 3**] The analysts now identify unsafe control actions from the control structure with context tables. The additional 2 steps which this paper proposes prevent analysts from suffering from too much unnecessary content of context tables.

- (Step 4) “Perform SFTA of NuSCR with NuFTA to produce MCSs”

NuFTA reads the NuSCR formal requirements specification, constructs fault trees as shown in <Fig. 3>, and generates 106 MCSs for the shutdown case and 56 MCSs for normal case. <Table 1> shows a few MCSs for each case as an example. The MCSs also have timing-related constraints occurred from TTS node, for example, waiting at t-p or waiting for [480, 480] represents a condition of TTS internal state in order to satisfy the top-event. The ‘waiting for [480, 480]’ means the internal state of TTS should be in waiting state at minimal 480 delay time to maximum 480. The 162 MCSs in total will be used as contexts (i.e., process model variables) combinatorially to identify UCAs of all types.

- (Step 5) “Generate combinations of process model variables for context tables”

Analysts check all 162 contexts in the context table and determine whether the context is hazardous or not, based on their experience and knowledge. CTMaker generates the whole 162 combinations of relevant contexts (process model variables) with actual values mechanically as shown in <Table 2>, and then analysts can construct the context table with the process model variables as shown in <Table 3>. It shows a part of the context table, including 7 hazardous contexts and 1 non-hazardous, as an example. The UCA for CT #4 would be “The steam generator level was maintained below the set-point for 24 cycles, so the trip should have fired but not.”

[**STPA Step 4**] The analysts finally identify loss scenarios for the UCAs determined. For example, we could think the loss scenario of

Table 1
A few MCSs generated from NuFTA.

f_LO_SG1_LEVEL_Trip_Out = True	
MCS for CT #4, 7	f_LO_SG1_LEVEL_Trip_Out = true & th_LO_SG1_LEVEL_Trip_Logic_state = Trip at t & th_LO_SG1_LEVEL_Trip_Logic = true & Waiting for [480, 480] & 0 <= f_LO_SG1_LEVEL_PV <= 12899 & f_LO_SG1_LEVEL_MT_Query ! = true & f_LO_SG1_LEVEL_AT_Query ! = true & f_LO_SG1_LEVEL_PT_Query ! = true & f_LO_SG1_LEVEL_Op_Byp_Init = false & f_Mod_Err ! = true & f_LO_SG1_LEVEL_Chan_Err ! = true & f_LO_SG1_LEVEL_PV_Err = false & 0 <= f_LO_SG1_LEVEL_PV <= 29400 & f_LO_SG1_LEVEL_MT_Query ! = true & f_LO_SG1_LEVEL_AT_Query ! = true & f_LO_SG1_LEVEL_PT_Query ! = true & 600 <= f_LO_SG1_LEVEL_PV <= 30000 & f_LO_SG1_LEVEL_MT_Query ! = true & f_LO_SG1_LEVEL_AT_Query ! = true & f_LO_SG1_LEVEL_PT_Query ! = true
MCS for CT #8	f_LO_SG1_LEVEL_Trip_Out = true & th_LO_SG1_LEVEL_Trip_Logic_state = Normal at t & th_LO_SG1_LEVEL_Trip_Logic = false & Waiting at t-p & 12900 <= f_LO_SG1_LEVEL_PV <= 30000 & f_LO_SG1_LEVEL_MT_Query ! = true & f_LO_SG1_LEVEL_AT_Query ! = true & f_LO_SG1_LEVEL_PT_Query ! = true & f_Mod_Err = true
f_LO_SG1_LEVEL_Trip_Out = False	
MCS for CT #5	f_LO_SG1_LEVEL_Trip_Out = false & f_LO_SG1_LEVEL_Op_Byp_Init ! = false & f_Mod_Err ! = true & f_LO_SG1_LEVEL_Chan_Err ! = true & f_LO_SG1_LEVEL_PV_Err = false & 0 <= f_LO_SG1_LEVEL_MT_Val <= 29400 & f_LO_SG1_LEVEL_MT_Query = true & f_LO_SG1_LEVEL_AT_Query ! = true & f_LO_SG1_LEVEL_PT_Query ! = true & 600 <= f_LO_SG1_LEVEL_MT_Val <= 30000 & f_LO_SG1_LEVEL_MT_Query = true & f_LO_SG1_LEVEL_AT_Query ! = true & f_LO_SG1_LEVEL_PT_Query ! = true

Table 2
An example of results by CTMaker.

Extracted variable values by CTMaker	
Order of variables	[f_X_Trip_Out, th_X_Trip_Logic, f_X_Val_Out, f_X_PV, (f_X_MT_Query, f_X_AT_Query, f_X_PT_Query), f_Mod_Err, f_X_Op_Byp_Init, f_X_Chan_Err, f_X_PV_Err, th_X_Trip_Logic(_state), (f_X_MT_Val, f_X_AT_Val, f_X_PT_Val)]
Interpreted results of MCSs for CT #4, 7	[true, true, N/A, 0 < = x < = 12899 & 0 <= x <= 29400 & 600 <= x <= 30000, (false, false, false), false, false, false, false, Trip at t & Waiting for [480, 480], (N/A, N/A, N/A)]
Interpreted results of MCSs for CT #5	[false, N/A, N/A, N/A, (true, false, false), false, true, false, false, N/A, (0 <= x <= 29400 & 600 <= x <= 30000, N/A, N/A)]
Interpreted results of MCSs for CT #8	[true, false, N/A, 12900 <= x <= 30000, (false, false, false), true, N/A, N/A, N/A, Normal at t & Waiting at t-p, (N/A, N/A, N/A)]

CT#4 such as.

1. The BP logic operation not implemented correctly,
2. Trip signal occurs but not received by CEDMCS,
3. ENFMS provides spurious feedback, or
4. CEDMCS receives a trip signal but fails to operate.

5. Related work

STPA has been studied in several ways, such as combination with formal methods, use in test case generation, and application to systems [8]. proposed an automatable method to generate hazardous control actions given certain information about the system [16]. proposes a formalization method for defining safety requirements from safety requirements/constraints written in natural language derived from the STPA, and they use a four-variable model and SCR to model the safety requirements. In Ref. [17], the authors proposed an extended STPA process using SysML block definition diagram and internal block diagram to construct control structure. Safety analysis methods about using petri net or colored petri net (CPN) to STPA are studied in several ways [18–20]. They used the petri net to construct a hierarchical control structure model and process model to support performing STPA with the formal approach [10]. proposes an extraction method of process model variables for the STPA based on Four-Variable model that consists of monitoring, input, controlled, and output variables.

The authors of [9] uses abstract state machine (ASM), defined by Asmeta language, to generate process model variables from 'monitoring' and 'controlled' functions of the model. The context table is also composed by combining values of these functions. FSTPA-I [21] is a formal framework that addresses the hazard identification of the STPA, the paper defines a system specification and behavior diagram necessary for STPA and defines a function, called the control action condition function, for identifying hazards from the behavior specifications [22]. introduces an integrated method of STPA and state machine analysis to support STPA by analyzing the dynamic behavior of the systems. It uses an FSM to model the dynamic behavior of the controller and to determine the system states which can affect the control actions for the STPA by extending a control action table.

There are several approaches of helping STPA to construct control structure or context table more easily using formal models. A context table, which is generated with the process model systematically, helps analysts to identify hazardous contexts in controls systematically [8,14]. The proposed approach of this paper extracts a process model and constructs a context table from NuSCR formal requirements specifications with two algorithms systematically and mechanically. The systematic and mechanical generation of process model and context table can help analysts to reduce the analysis efforts to identification of important information for STPA. Especially, the proposed method can generate detailed and actual cases of contexts, which contain complex conditions like timing-related constraints, in generating context tables by using NuFTA. The use of NuFTA also reduces the size of context tables in contrast to simple combination of variables generated exponential size of context table, which are not possible to analyze, in the previous study [14].

6. Conclusions and future work

This paper proposes a formal approach to support the manual identification of UCAs, effectively and systematically. It allows analysts to mechanically extract Process Model, an important element that makes up the Control Structures, from a NuSCR formal requirements specification for a software controller. It then concisely constructs the contents of Context Tables, from which analysts can

Table 3
An example of the context table with the process model variables generated by CTMaker.

Control Action	Type	No.	Contexts (Process Model Variables)										Hazardous?
Trip signal	Not provided causes hazard		f_X_Trip_Out	th_X_Trip_Logic	f_X_Val_Out	f_X_PV	f_X_(MT/AT/PT)_Query	f_Mod_Err	f_X_Op_Byp_Init	f_X_Chan_Err	f_X_PV_Err	th_X_Trip_Logic_State	
		1	TRUE	N/A	N/A	N/A	N/A	N/A	N/A	TRUE	N/A	N/A	**O**
		2	TRUE	N/A	N/A	N/A	TRUE & TRUE & TRUE	TRUE	N/A	N/A	N/A	Waiting at t&t0 = false&Waiting at t-p & Waiting not for [480,480]	**O**
		3	TRUE	N/A	0<=x<=13199&0<=x<=29400&600<=x<=30000	N/A	TRUE & FALSE & FALSE	FALSE	FALSE	FALSE	FALSE	Trip at t & t0 = true & Trip at t-p	**X**
		4	TRUE	TRUE	N/A	0<=x<=12899&0<=x<=29400&600<=x<=30000	FALSE & FALSE & FALSE	FALSE	FALSE	FALSE	FALSE	Trip at t & Waiting for [480, 480]	**O**
	Providing causes hazard	**5**	FALSE	N/A	N/A	N/A	TRUE & FALSE & FALSE	FALSE	TRUE	FLASE	FLASE	N/A	**O**
		6	FALSE	N/A	0<=x<=12899&0<=x<=29400&0<=x<=29400	N/A	FALSE & TRUE&TRUE	FALSE	N/A	FALSE	FALSE	Waiting at t&t0 = false&Waiting at t-p&Waiting no for [480,480]	**O**
	Too early, too late, out of order	**7**	TRUE	TRUE	N/A	0<=x<=12899&0<=x<=29400&600<=x<=30000	FALSE & FALSE & FALSE	FALSE	FALSE	FALSE	FALSE	Trip at t & Waiting for [480, 480]	**O**
	Stopped too soon, applied to long	**8**	TRUE	FALSE	N/A	12900<=x<=30000	FALSE & FALSE & FALSE	TRUE	N/A	N/A	N/A	Normal at t & Waiting at t-p	**O**

identify all relevant UCAs effectively, using the NuFTA software fault tree analysis technique. The case study with a preliminary version of a Korean nuclear reactor protections system shows the proposed approach's effectiveness and applicability. This paper also provides a systematic process of 5 steps that help analysts apply this formal approach to STPA analysis. The case study also shows the proposed approach's effectiveness and applicability. We are now developing an integrated STPA analysis environment to support the entire SPTA analysis as well as the process of 5 steps which this paper proposes.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This paper was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2021R1F1A1047246).

References

[1] N. Leveson, Safeware: System Safety and Computers, Addison Wesley, 1995.
[2] International Electrotechnical Commission (IEC), IEC 61508, Functional Safety of Electrical, Electronic and Programmable Electronic, E/E/PE) safety-related systems, 2000.
[3] Nuclear Regulatory Commission (NRC), Criteria for Use of Computers in Safety Systems of Nuclear Power Plants, RG 1.152), 2004.
[4] Institute of Electrical and Electronics Engineers (IEEE), IEEE-704.3.2, IEEE Standard Criteria for Programmable Digital Devices in Safety Systems of Nuclear Power Generating Stations, 2016.
[5] C.A. Ericson, Hazard Analysis Techniques for System Safety, John Wiley & Sons, 2015.
[6] N.G. Leveson, J.P. Thomas, STPA Handbook, Cambridge, Ma, USA, 2018.
[7] N. Leveson, Engineering a Safer World: Systems Thinking Applied to Safety, MIT press, 2011.
[8] J.P. Thomas, Extending and Automating a Systems-Theoretic Hazard Analysis for Requirements Generation and Analysis, Massachusetts Institute of Technology (MIT), 2013. Ph.D. Dissertation.
[9] F. Al-Shareefi, A. Lisitsa, C. Dixon, Abstract state machines and system theoretic process analysis for safety-critical systems, in: Brazilian Symposium on Formal Methods, Recife, Brazil, Nov 29 –, 2017. Dec 1.
[10] M. Chen, L. Wang, J. Hu, T. Feng, An extraction method of STPA variable based on four-variable model, in: International Conference on Intelligent and Interactive Systems and Applications, Hongkong, 2018. June 29-30.
[11] J. Yoo, T. Kim, S. Cha, J.S. Lee, H.S. Son, A formal software requirements specification method for digital nuclear plant protection systems, J. Syst. Software 74 (1) (2005) 73–83.
[12] S. Jung, J. Yoo, Y.J. Lee, A software fault tree analysis technique for formal requirement specifications of nuclear reactor protection systems, Reliab. Eng. Syst. Saf. 203 (2020) 107064.
[13] E.S. Kim, D.A. Lee, S. Jung, J. Yoo, J.G. Choi, J.S. Lee, NuDE 2.0: a formal method-based software development, verification and safety analysis environment for digital I&Cs in NPPs, Journal of Computing Science and Engineering 11 (1) (2017) 9–23.
[14] Y. Seo, An Extended Process of STPA and Implementation of an Automatic Assistant Tool for Reactor Protection System Software, Master's Thesis, Konkuk University, 2016.
[15] Korea Atomic Energy Research Institute (KAERI), SRS for Reactor Protection System KNICS-RPS-SRS121. Technical reports, 2003.
[16] Y. Zhou, L. Wang, J. Hu, Y. Wang, Safety analysis and requirements verification of electronic checklist system based on STPA, in: 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, Nov 24-26, 2017, 2017.
[17] N.Y. Choi, B.G. Lee, Hazard analysis process based on STPA using SysML, Journal of Internet Computing and Services 20 (3) (2019) 1–11 (In Korean).
[18] R. Wang, W. Zheng, C. Liang, T. Tang, An integrated hazard identification method based on the hierarchical Colored Petri Net, Saf. Sci. 88 (2016) 166–719.
[19] Q. Xu, J. Lin, Safety analysis of communication-based train control system by STPA and colored petri net, in: International 2019 Cyberspace Congress, CyberDI and CyberLife, Beijing, China, 2019. Dec 16–18.
[20] D. Zhu, S. Yao, C. Xu, STAMP-based hazard analysis for computer-controlled systems using petri nets, Int. J. Perform. Eng. 14 (9) (2018) 1997.
[21] P. Asare, J. Lach, J.A. Stankovic, Fstpa-I, A formal approach to hazard identification via system theoretic process analysis. ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS), Philadelphia Pennsylvania, 2013. April 8-11, 2013.
[22] A. Abdulkhaleq, S. Wagner, Integrating State Machine Analysis with System-Theoretic Process Analysis, Software Engineering 2013-Workshopband, Gesellschaft für Informatik e.V., Bonn, 2013.