Korean J. Chem. Eng., **22**(4), 503-511 (2005)

An Automatic Synthesis System for Start-Up Operating Procedures of Boiler Plants

Sung Gun Lee, Sang il Han, Yoon Sup Byun, Kyu Suk Hwang†, Young Han Kim*, Dongil Shin and En Sup Yoon*****

Department of Chemical Engineering, Pusan National University, Busan 609-735, Korea
*Department of Chemical Engineering, Dong-A University, Busan 604-714, Korea
**Department of Chemical Engineering, Myongji University, Yongin, Kyunggido 449-728, Korea
***School of Chemical Engineering, Seoul National University, Seoul 151-742, Korea

(*Received 17 January 2005 • accepted 11 April 2005*)

Abstract−As boiler plants which supply steam and heat into chemical plants become larger in scale and more complicated, accidents caused by the wrong understanding and erroneous operations are increasing. In this paper, we present a computer-aided system that should prevent such mistakes and synthesize operating procedures for start-up of steam boiler plants: 1) This system generates a goal tree by generalizing and classifying the operational situations hierarchically, and uses the relationship between an operation and a function to search for primitive operations in the lowest level of the goal tree. 2) It simulates changes in operational situations caused by the action of primitive operations with heuristic knowledge of transition relationships between operational situations. 3) It determines the priority of low-level operational situations and primitive operations by referring to the database on hazardous situations and properties. We have applied the developed system to a large-scale boiler plant to demonstrate its effectiveness.

Key words: Operating Procedure Synthesis, Boiler Plant, Start-up, Goal Tree, Heuristic Knowledge

INTRODUCTION

Though the control systems of boiler plants have progressed rapidly as steam boiler plants become more large-scale and complicated, operations at unsteady states such as start-up and shutdown are still managed according to the heuristic knowledge of human operators and to standard operating procedures. If human operators make an error because of their wrong recognition, the interlock system of the boiler plant may stop the plant, causing economic losses because steam and heat cannot be supplied to downstream plants. In previous works reported in the literature, computer-aided systems for boiler plants have been limited to the energy-saving systems and to the expert systems for fault diagnosis. In this study, we develop a computer-aided system that automatically synthesizes operating procedures for start-up and should prevent hazardous accidents due to the erroneous operations and the wrong recognition by human operators as well.

Operating procedure synthesis is the methodology that searches for the sequence of primitive operations capable of moving the plant states from an initial state to a final goal state without violating operational and safety constraints in the plant. This work is accomplished effectively by use of constraints of the considered process in contrast to the simple planning methods in the artificial intelligence area. After each operation of the plant device is done, the process states have to be simulated and updated by the simulation methodology, which has to consider the operational situations of the process, because there can be different types of changes in process states by the actions of primitive operations according to the present process state.

Prior researches suggested nonlinear planning [Lakshmanan and Stephanopoulos, 1988a, b, 1990] and goal tree methodology [Rivas and Rudd, 1974; Hwang et al., 1991] to solve combinatorial explosion problems due to the blind search on all state space. These methodologies are applicable to synthesizing operating procedures for start-up of a large-scale boiler plant. However, nonlinear planning methodology has difficulty using the state information of a process efficiently because it replaces state space with the planning space, and it is difficult to control searching algorithms. Especially, Lakshmanan [1988a, b, 1990] automatically generated subgoals to purge the hazardous material when no primitive operation was applicable to reach the final goal state, and then determined the order of primitive operations with the nonlinear planning method. He indicated that operating procedure synthesis for large-scale chemical plants needs an expert system to formulate the necessary subgoals. In other words, as the planning space increases, the complexity of searching subgoals increases rapidly in nonlinear planning. Soutter [1997] represented the information of subgoals with operator forms, and then he developed the system using the nonlinear planning methodology and demonstrated its effectiveness. However, his methodology could not overcome the disadvantage of not using the numerical information.

In this paper, we propose a tree methodology which divides an operational goal into a hierarchy of a detailed list of subgoals according to their order. The lowest-level subgoals in the goal tree are accomplished by the selection of related operations of plant devices. For that methodology, we propose a subgoal ordering method using generalized knowledge of standard operating procedure for start-up of boiler plants and heuristic knowledge of human operators.

START-UP OPERATION OF BOILER PLANTS

Boiler plants consist of air supply sections, Boiler Feed Water (BFW) supply sections, fuel gas and oil supply sections, burner units, and drum units (see Fig. 1). After BFW is preheated by an economizer, it flows into an upper drum and is transformed into steam

†To whom correspondence should be addressed.
E-mail: kshwang@pusan.ac.kr

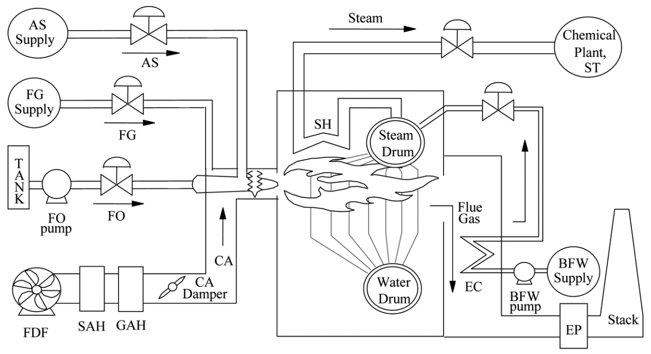

Fig. 1. Structure of a boiler plant.

SH: Super Heater	SAH: Steam Air Heater
FO: Fuel Oil	GAH: Gas Air Heater
CA: Combustion Air	FDF: Forced Draft Fan
EP: Electrostatic Precipitator	BFW: Boiler Feed Water
EC: Economizer	AS: Atomizing Steam
FG: Fuel Gas	ST: Steam Turbine

by a burner. The steam is changed to a state of high pressure and high temperature through a Super Heater (SH), and then it is supplied into other downstream (chemical) plants. The air required to burn fuel oil and gas in the burner is transported into the furnace by a Forced Draft Fan (FDF) after being preheated through a Steam Air Heater (SAH) and a Gas Air Heater (GAH). Fuel gas, oil, and atomizing steam to atomize fuel oil are fed into the burner through each supply line.

The human operators use the following procedure to determine the order of primitive operations during the start-up of a large boiler plant:

1. They construct the final goal tree composed of the detailed subgoals necessary to reach the final goal state, and determine the priority of the subgoals by referring to the possible hazardous situations.
2. They aim for the primitive operations necessary to achieve each subgoal by searching the classified library for relations between the primitive operations and their expected function, and then simulate changes of the process states by executing the candidate operators searched for, with the heuristic equation and the state-transition knowledge.
3. They select the next subgoal to be achieved from the changed process states and the goal tree.
4. They repeat the procedure 1 to 3 for all of the selected subgoals.

In this research, we first constructed a generalized goal tree for a practical boiler plant to utilize the inference method of human operators at the start-up of the boiler plant so that this goal tree is also available for the generation and ordering of subgoals for boiler plants of different structures.

OPERATING PROCEDURE SYNTHESIS SYSTEM FOR BOILER PLANTS

Our proposed Operating Procedure Synthesis system for Boiler Plants (OPSBP) consists of eight modules (see Fig. 2):

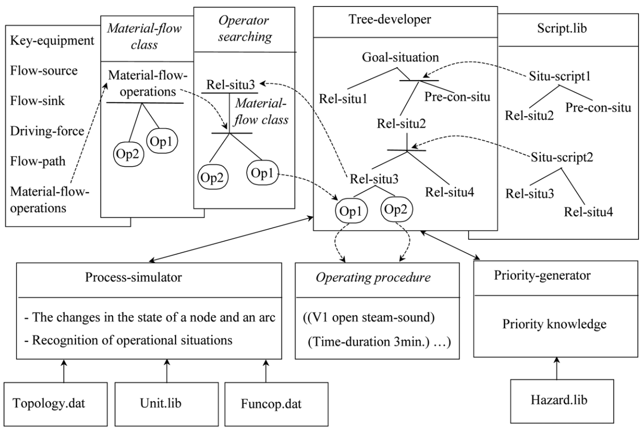

Fig. 2. Overview of the proposed operating procedure synthesis system.

Topology.dat: Data on unit connections, material flow paths, and information about an initial state and a final state of the process.

Funcop.dat: Data on functional rules representing the unique functional characteristics of each piece of equipment.

Unit.lib: Hierarchical units library-classified to the functional types of process units by using object-oriented modeling method.

Hazard.lib: Library of all possible hazardous situations in boiler plants.

Script.lib: Library in which generalized change patterns of necessary operational situations are stored in order to recognize how operational situations progress.

Tree-developer: Construct goal trees, which represent causal relationships among subgoals to be activated to change the process state from an initial state to a goal state.

Process-simulator: Simulate changes of the process state by primitive operations with the functional rules of each unit.

Priority-generator: Module to determine the priority among subgoals by using the hazardous situation data in Hazard.lib.

The proposed OPSBP system works as follows. First, the system reads the connection structure, initial state and final goal state of the model process stored in Topology.dat. Then Tree-developer constructs a goal tree with hierarchical structure using the information about causal relations among operational situations to be satisfied to accomplish some operational goal state, which is stored in Script.lib, and searches an appropriate operation using the relationship between the operational situation and the device operation. The search for the primitive operations necessary to achieve each subgoal on the constructed goal tree is executed by finding the primitive operations with satisfying functional characteristics to achieve the corresponding subgoal. To represent the function of each primitive operation, it is classified into six categories such as a material-flow operation, a flow-blocking operation, a flow-control operation, a switch operation, a control-mode-change operation and a time-duration operation. Because the function of the primitive operations concerned with the start-up of boiler plants is mostly about material flows, the system searches for an available flow path satisfying a corresponding subgoal and finds the primitive operations sufficient to produce a material flow at the corresponding flow path. The priority order between subgoals to be achieved is determined by Priority-generator module using Hazard.lib. Using the forward chaining method with the functional rules of each unit in Funcop.dat, process-simulator simulates changes in process state by the selected primitive operation and checks whether the changed process state violates safety constraints or not. If the process state is safe, the corresponding primitive operation is added to the operating procedure. The system applies all the candidate primitive operations as stated above, and if hazardous situations do not occur and the final operational goal has been achieved, the synthesized operating procedure is finally determined.

KNOWLEDGE REPRESENTATION OF PROCESS STRUCTURE AND PROCESS DECOMPOSITION

The process structure is represented by a directed graph composed of nodes and arcs. Elements (units, valves, mixing points, and splitting points) that change a process state are represented as nodes, and pipes connecting a unit with others are represented as arcs. In this paper, the boiler plant is broken down into a number of lines; the functional relationships between lines are classified in terms of materials in the lines such as air, steam, oil, water, gas, etc. There are functional dependent relationships among lines, and lines consist of units on the flow path of the lines (see Fig. 3). The unit model representation is a hierarchical structure using object-oriented description (see Fig. 4). By using this model, we can both facilitate modification and updating of data and ensure reusability of knowledge, which is encapsulated as objects.

SYNTHESIS OF THE GOAL TREE

We introduce a goal tree to represent subgoals required for expanding high-level operational situations into low-level ones. Here, each subgoal on the goal tree corresponds to each operational situation.

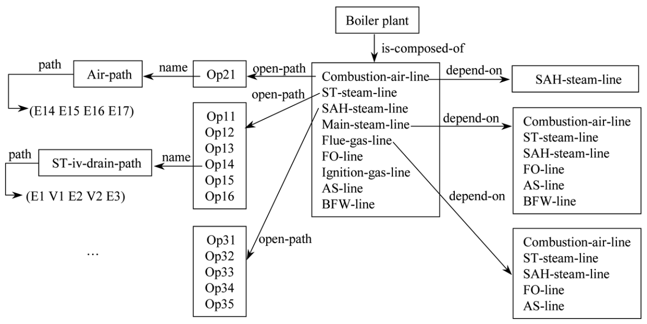

Fig. 3. Process decomposition.

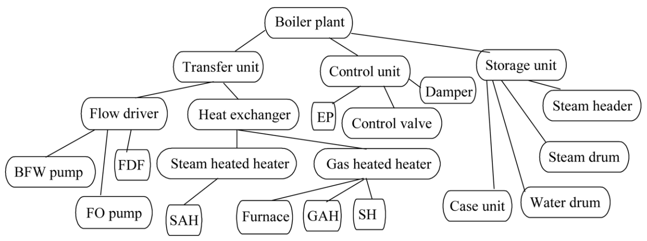

Fig. 4. Unit hierarchy.

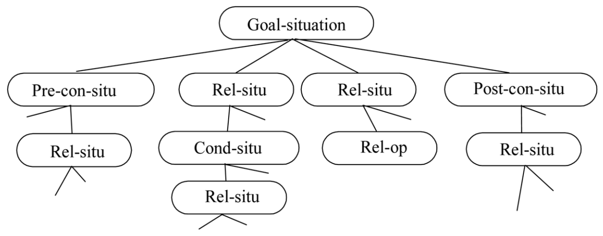

Fig. 5. Relation among operational situations.

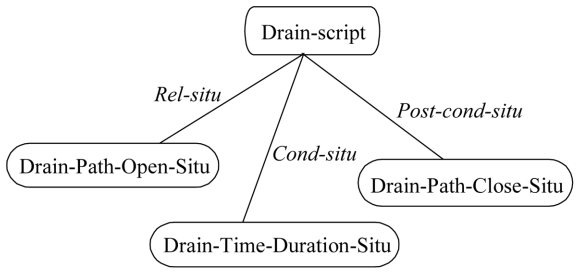

Fig. 6. Script for a generalized operational situation.

The expansion of the goal tree is achieved by using general situation expansion knowledge in Script.lib and specific situation expansion knowledge about specific processes. Each operational situation is expanded into the low-level operational situations by using the corresponding Rel-situ as shown in Fig. 5. Meanwhile, if some conditions must be satisfied before the corresponding high-level operational situation is achieved, it is called pre-constraint-situation (Pre-con-situ) shown in Fig. 5. If there exist some conditions to be finally satisfied before the corresponding high-level operational situation is achieved, this is called post-constraint-situation (Post-con-situ). In the case of time constraints, when time-duration is needed to accomplish its high-level operational situation, it is called Cond-situ (see Figs. 5 and 6).

1. Expansion Method of General Operational Situations

General operational situations in boiler plants are generalized in the form of a formal script and used to expand high-level operational situations into low-level ones. In other words, the operational situations are expanded into the low-level situations by referring to the operational situation script that is composed of general low-level operational situations, a Pre-con-situ, a Post-con-situ and a Cond-situ, and then the total goal tree is constructed.

For example, the drain task for the steam turbine (ST) inlet or outlet valve is generalized as follows. First, we let the steam flow by opening the flow path; and after a certain amount of time, if all the corresponding conditions are satisfied, we close the drain valve (see Fig. 6).

Data structure of script is as follows:

```
(script ‘script-frame
     ‘((general-pre-con-situ (pre-constraint-situation))
       (general-rel-situ (low-level operational situations))
       (general-cond-situ (time-duration))
       (general-post-con-situ (post-constraint-situation))))
```

Script examples for goal tree expansion of boiler plants are shown below.

1. To activate the line-up task of a steam line, the drain task of each unit on the line should be accomplished before, and finally that task is completed by opening a corresponding steam line.

```
('Steam-line-lineup-script 'script-frame
   '((general-pre-con-situ (Steam-Line-Equipment-Drain-Situ))
     (general-rel-situ (Steam-Line-Open-Situ))))
```

2. Generally, to drain water out of some units and pipes, we first open drain path for the units or pipes, and then close the drain path after time necessary for the drain task has passed.

```
('Drain-script 'script-frame
   '((general-rel-situ (Drain-Path-Open-Situ))
     (general-cond-situ (Drain-Time-Duration-Situ)
     (general-post-con-situ (Drain-Path-Close-Situ))))
```

Fig. 7 shows the expansion procedure for an operational situation using script, and it shows how the operational situation related to producing steam in the boiler plant is expanded into low-level operating situations. It first searches the corresponding script for producing steam in the boiler plant (that is, Steam-Gen-Script), and then each of low-level situations of the searched script (that is, Water-Feed-Drum-Situ, Heating-Situ) is specified upon the model process, and it becomes the low-level situation of the high level situation.

2. Expansion Method for Specific Operational Situations

There is specific expansion knowledge to expand high-level operational situations about specific process into low-level ones differently from the way using general expansion knowledge. For example, the type of a pre-heater could be different in each boiler plant, and the attribute value representing the working-situation of the pre-heater is different in each boiler plant. In this work, the low-level operational situations about the working-situation of the pre-heater in the model plant are the Steam Air Heater (SAH) working-situation, the Gas Air Heater (GAH) working-situation and the economizer working-situation, and this specific knowledge is controlled by Tree-developer module to construct the total goal tree. Meanwhile, the ordering of the operational situations on this constructed goal tree is determined by using the priority determination knowledge, to be mentioned in Section 4.

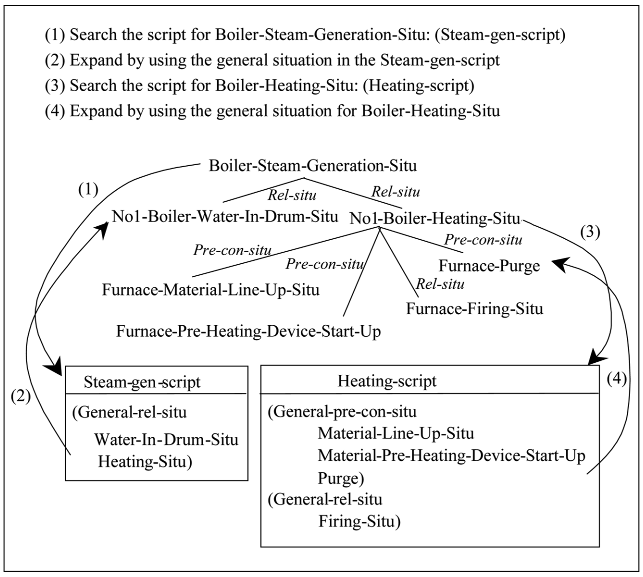
(1) Search the script for Boiler-Steam-Generation-Situ: (Steam-gen-script)
(2) Expand by using the general situation in the Steam-gen-script
(3) Search the script for Boiler-Heating-Situ: (Heating-script)
(4) Expand by using the general situation for Boiler-Heating-Situ
Boiler-Steam-Generation-Situ
Rel-situ
Rel-situ
No1-Boiler-Water-In-Drum-Situ
No1-Boiler-Heating-Situ
Pre-con-situ
Pre-con-situ
Pre-con-situ
Rel-situ
Furnace-Purge
Furnace-Material-Line-Up-Situ
Furnace-Firing-Situ
Furnace-Pre-Heating-Device-Start-Up
(1)
(2)
(3)
(4)
Steam-gen-script
(General-rel-situ
Water-In-Drum-Situ
Heating-Situ)
Heating-script
(General-pre-con-situ
Material-Line-Up-Situ
Material-Pre-Heating-Device-Start-Up
Purge)
(General-rel-situ
Firing-Situ)

Fig. 7. Expansion procedure of an operational situation.

Table 1. Classification of operation types

Class type of operations	Instances of operations
Material-flow operation	(Valve open operating-speed/amount), (Equipment on operating-speed/amount)
Time-duration operation	(Time-duration time)
Flow-blocking operation	(Valve close operating-speed/amount)
Control-mode-change operation	(Equipment change-mode (auto/manual/cascade))
Flow-control operation	(Valve/equipment set-value)
Switch operation	(Switch on)/(Switch off)

3. Searching for Primitive Operations to Achieve Operational Situations

Primitive operations with functional characteristics necessary to achieve the subgoals are searched for at the final step of the goal tree expansion. They are classified as material-flow operation, flow-blocking operation, flow-control operation, switch operation, control-mode-change operation and time-duration operation from the viewpoint of its functional type (see Table 1). General operational situations can be achieved by the combination of the six operation types above.

The knowledge for searching out primitive operations in terms of operation types is stored in Operation-type.lib, and the key-equipment (Key-equipment shown in Table 2) of the model process is used to find available primitive operations by matching the model process to the elements of their Operation-type.lib. For example, the components necessary to search for the material-flow operation are a key-equipment, a flow-source, a flow-sink, a flow-path, etc. Here, the key-equipment is a flow target, the flow-source is a material source to provide the material continuously, the flow-sink is a sink to store the material without limit, and the flow-path is a path to make the material flow. Meanwhile, the components necessary for searching the flow-control operation are a control mode (manual, auto or cascade mode), a control variable, a manipulation variable, a control source (an equipment to execute a control operation), a manipulation amount, etc.

Each line-up task is reached to the desired operational situations by material-flow operations. The way to search for the material-flow operations is shown in Fig. 8. The material-flow operations are a set of primitive operations to open valves on the line related to creating the flow path with the key-equipment and the flow-source, which is necessary to achieve the corresponding operational situation.

4. Priority Ordering of Operational Situations and Primitive Operations

It is useful to reduce backtracking and minimize the state space for ordering of the subgoal and action by determining the priority in the expansion of the goal tree. To do this, first, the system con-

Table 2. Library for searching for primitive operations

Classes	Slot variables	Values
Class for material-flow operations	Key-equipment	Drum
(in case of Water-Flow-In-Drum-Situ)	Flow-source	BFW-tank
	Flow-sink	Drum
	Driving-force	BFW pump
	Flow-path	(E40 E41 E42 E43 E44 V45 V46 V47 E45 E46 E47 E48)
	Material-flow-operations	(pump on), (control-valve on), (motor-valve open), …
Class for flow-control operations	Control-mode	Manual mode
(in case of Drum-Level-Normal-Control-Situ)	Control-variable	Level
	Manipulation-variable	Flow
	Manipulation-source	BFW-flow-remote-controller
	Manipulation-amount	25 Ton/Hour
	Flow-control-operations	(motor-valve-of-BFW-remote-controller open)

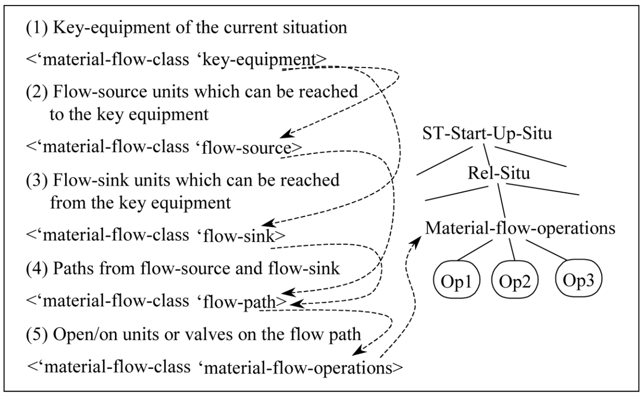
(1) Key-equipment of the current situation
<'material-flow-class 'key-equipment>
(2) Flow-source units which can be reached to the key equipment
<'material-flow-class 'flow-source>
(3) Flow-sink units which can be reached from the key equipment
<'material-flow-class 'flow-sink>
(4) Paths from flow-source and flow-sink
<'material-flow-class 'flow-path>
(5) Open/on units or valves on the flow path
<'material-flow-class 'material-flow-operations>
ST-Start-Up-Situ
Rel-Situ
Material-flow-operations
Op1
Op2
Op3

Fig. 8. Searching for primitive operations for material flow.

structs the general priority knowledge by using the hazardous situations library, and then determines the priority of subgoals and operations by applying the general priority knowledge to the goal tree.

A hazardous situations library is represented as follows.

```
(HAPOSS1 'water-hammer-in-steam-turbine
    '(and (ST-Pre-Working-Situ id nil)
          (ST-Working-Situ id t)))
```

The following is the detailed meaning of the expression above: If the steam turbine works before the precondition for its operation is satisfied, a water hammer at the steam turbine can occur. An alternative description of the above expression is as follows, and this meaning is the same with the one above.

```
(HAPOSS1 'water-hammer-in-steam-turbine
    '(and (ST-Working-Situ first)
          (ST-Pre-Working-Situ second)))
```

Therefore, to achieve a desired operational situation safely, the precondition for the primitive operation is satisfied first, and then the steam turbine working-situation has to be activated. The general description about the priority is stored at the *situ-priority* variable as follows:

Priority1: (setf *situ-priority* '(ST-Pre-Working-Situ ST-Working-Situ))

The concrete meaning of the constructed priority knowledge is as follows: First, the preconditions of the steam turbine are satisfied. Secondly, the operational situations of the steam turbine must be satisfied.

The following example shows that if the pump operates before formulating an open-path, which means the opening state of the flow path is formulated, then a rupture of the pump would occur. Therefore, the priority knowledge for the pump operation is generated as follows: that is, after the primitive operations are executed to formulate the open-path for protecting the pump rupture, and to switch on the pump to work.

SIMULATION OF PROCESS STATE BY PRIMITIVE OPERATION

1. Representation of Process States

The attribute variables of a node and an arc are used to represent the process state.

(1) The state of a node: attribute variables such as apval, func-utility, and working are used to represent the state of a node.

Apval: the apval attribute of a node says whether a fluid can flow through the node or not. If that is possible, the attribute value of a node is true.

Func-utility: the func-utility attribute value shown in Table 3 says that the utility is necessary to activate the node.

Working: the working attribute stores the operating amount of a unit and a valve. For example, an operating amount of a steam turbine is given by the first order equation in terms of the flow-rate of feeding steam.

(2) The state of an arc: it is described by the values of temperature, pressure, flow-rate, composition, and phase.

(3) The description of rules for simulation: the simulation of the process state is carried out with the functional rules (funcop) of a unit using the forward-chaining method, where funcop is a set of rules representing the state change of a node and connected arcs caused by a primitive operation according to the present operational

Table 3. Examples of func-utility

Unit	Func-utility
Steam turbine	Steam-flow
SAH	Steam-flow
GAH	Flue-gas-flow
Furnace	Flame
EP	Switch
FDF	Steam-turbine

situations.

```
(node ‘funcop
    ‘((IF ((<working conditions of func-utility>)
          (<working amount of the node>)
          (<time-duration conditions>)
         ((<input-arc name> (<process state such as pressure, temperature, flow, composition, and phase>))))
    THEN ((<output-arc name
          (<process variables such as pressure, temperature, and flow>
                ADD ((input-arc temp mul α)
                     (input-arc press mul β)
                     (input-arc flow mul γ)))
          (<process variables such as composition and phase>
                ADD or DELETE or SAME or PT <input-arc name> m))
    AR (<the connectivity list between input-port and output-port>))))
```

Here, α, β, and γ are parameters, and m is a new phase or composition.

The following is an example.

```
(‘v101 ‘funcop
    ‘((IF ((u nil) (working steam-sound) (time-duration ((less 26)))
          (s (s101 (press 15) (temp 15) (phase gas) (comp H2O) (flow 12))))
    THEN ((s102 (press add ((s101 press mul 0.2)))
                (temp add ((s101 temp mul 0.6)))
                (comp same s101)
                (phase add s101 liquid)
                (flow add ((s101 flow mul 0.2)))))
    AR ((s101) (s102)))
```

‘Funcop’ rule is composed of an IF-part, a THEN-part, and an AR-part. The conditions listed in an IF-part, such as the working state of a func-utility and a node, the conditions of time-duration and the process states of input-arcs, are checked to know if the rule matches the states of input-arcs and the node after the primitive operation is executed. If a matching rule is found, the state change of the model process is carried out by using THEN-part and AR-part of the rule. A THEN-part changes the process state by using the state-transition function represented in a THEN-part in accordance with the present operational situations and the state of inputs of the unit. The following is the state-transition function in terms of the temperature (T), pressure (P), and flow-rate (F). This form is used easily by the human operators in a practical plant to estimate the process state:

$$Y=\alpha T+\beta P+\gamma F+\delta \text{ (here, Y=T, P, or F)}$$

Here, T, P, and F at the right-hand side of the function are the state values of the input-arc, and Y at the left-hand side is the state variables of the output-arc.

The values of state variables comp and phase are changed by removing or adding the composition and phase to the values of the input-arc. We use the special forms such as keyword PT and SAME to change process states effectively. Here, keyword PT means that the materials of an input-arc pass through an output-arc without changes in the state variables at the nodes such as a mixing-point and a splitting-point. The keyword SAME means that special state variables go through the output-arc without changes. For example, the composition and phase of the output-arc are for the most part equal to those of the input-arc of the opened valves.

2. Simulation of Process State

The basic simulator, time-duration simulator, and control simulator are used to simulate changes in the process state by candidate primitive operations necessary to achieve subgoals in the goal tree.

In the case of material-flow operation and switch operation, the basic simulator is used to simulate the effect to the process state and behavior by execution of the primitive operation. The following are simulation steps of the basic simulator:

(1) Values of attribute variables such as apval and working of the node are changed by the primitive operation.

(2) Lines directly affected by the primitive operation or indirectly related to the primitive operation in terms of the dependency relationship (the keyword depend-on shown in Fig. 3) are searched.

(3) Values of state variables of the output-arcs connected with the corresponding node are calculated by forward chaining of the functional rules of the node. The state variables such as temperature, flow-rate, and pressure are changed by using the functional rules in attribute variable, funcop of the node. The composition or phase variable of the output-arc of the corresponding unit is determined by adding and removing elements shown in the functional rules without using numerical equations.

The control simulator maintains process state variables that pass through the controller with set value. The detailed simulation steps are as follows:

(1) The state variable of the target arc is kept at the set value of the controller. In other words, the value of state variable related to the purpose attribute of the controller is set as the set-point value of the controller. For example, if the purpose attribute value is the temp-change, the temperature of the target arc is changed to the set-point value.

(2) For the line next to the target arc, the basic simulator is executed.

An example of a time-duration operation is such as gradually opening a valve for ten minutes at the rate of one wheel per one minute or waiting five minutes until the drain task is completed. This primitive operation updates the state variable, time-duration.

CASE STUDY

A prototype system is demonstrated to show the effectiveness and basic features of the proposed system, with application to real-life boiler plants. To automate the operating procedure synthesis for boiler plants, our system has used techniques such as synthesis of a goal tree, expansion of operational situations, search for primitive operations necessary to satisfy the corresponding operational situation, simulation of changes in process state by primitive operations, and progression of the operational situations in accordance with the changed process state.

The model process is a part of a boiler plant, the steam turbine process shown in Fig. 9, to operate the FDF necessary to provide air to the furnace. The result of situations expanded for this process is shown in Fig. 10.

Fig. 10 explains the expansion step of operational situations for start-up of boiler plants. Table 4 shows the final operating procedure generated to satisfy the working-situation (Material-Driver-Start-Up-Situ) for the material transport units (steam turbine, FDF) and the firing situation (Burning-Situ) of the burner in the boiler plant. Here, Material-Driver-Start-Up-Situ means the operational situation related to operating equipment necessary to transport material for firing the burner into the furnace.

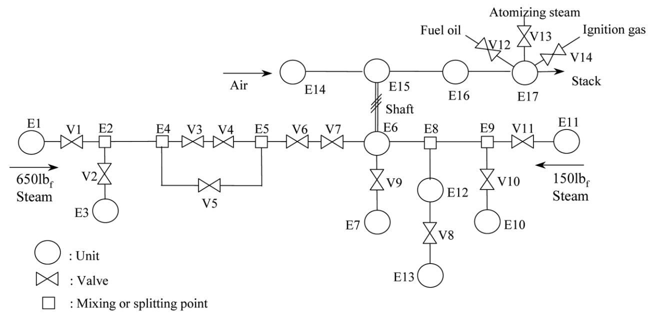

Fig. 9. Model process.

E1: High pressure steam header
E3: Steam inlet valve drain out
E6: Steam turbine
E7: Steam turbine chest drain out
E10: Steam turbine exhaust drain out
E11: Middle pressure steam header
E12: Steam turbine case
E13: Steam turbine case drain out
E14: Atmosphere
E15: FDF
E16: FDF damper
E17: Furnace
V1: Steam turbine inlet valve
V2: Steam turbine inlet drain valve
V3: Steam turbine inlet block valve
V4: Steam turbine flow control valve
V5: Steam turbine warming-up valve
V6: Steam turbine trip valve
V7: Steam turbine governor valve
V8: Steam turbine case drain valve
V9: Steam turbine chest drain valve
V10: Steam turbine exhaust line drain valve
V11: Steam turbine exhaust block valve
V12: Fuel oil throttle valve
V13: Atomizing steam throttle valve
V14: Ignition gas throttle valve

...
- ***Material-Driver-Start-Up-Situ***
  - *FDF-Start-Up-Situ*
    - *FDF-Damper-Auto-Mode-Situ*
    - *ST-Start-Up-Situ*
      - *ST-Drain-Situ*
        - *ST-Exhaust-BV-Drain-Situ*
          - *ST-Exhaust-BV-Drain-Path-Open-Situ*
          - *ST-Exhaust-BV-Drain-Time-Duration-Situ*
          - *ST-Exhaust-BV-Drain-Path-Close-Situ*
        - *ST-Case-Drain-Situ*
      - *ST-Warming-Up-Situ*
- ***Burning-Situ***
  - *Ignition-Situ*
  - *Atom-Steam-Inlet-Situ*
  - *Fuel-Oil-Inlet-Situ*

...

Water-Steam-Line-Service-Situ
Steam-Generation-Situ
Boiler-Plant-Start-Up-Situ

Fig. 10. Ordering of expanded situations.

Table 4. Synthesized operating procedure

Operating procedure synthesized by computer
1. (E16 auto-mode)
2. (V11 open steam-sound)
3. (Time-duration 3 min)
4. (V10 close)
5. (V11 open 100%)
6. (V1 open steam-sound)
7. (Time-duration 3 min)
8. (V2 close 100%)
9. (V1 open 100%)
10. (V5 open 100%)
11. (V3 open 100%)
12. (V5 close 100%)
13. (V8 close 100%)
14. (V9 close 100%)
15. (V14 open 1 kg$_f$)
16. (Ignition-sparker on)
17. (V13 open 55 °C/h)
18. (V12 open 55 °C/h)

The synthesized operating procedure for operating the steam turbine (E6 shown in Fig. 9) and the FDF (E15) to provide air to the furnace is given as follows. First, the control mode of the FDF damper (E16) is set as the auto mode (E16 auto-mode), and then the drain operation on the steam turbine case (E12) is executed. The steam turbine exhaust block valve (V11) on the drain line of the steam turbine case is opened until the steam outflow sounds. After 3 minutes the drain of condensate from the drain path (E9, V10, E10) is finished (V10 is initially open), and the turbine exhaust line drain valve (V10) is closed. Secondly, the valve V11 is opened fully and then the drain task of the steam turbine case (E12) is finished. Here, V2, V4, V6, V7, V8, V9, and V10 are open at initial state. Besides, to satisfy preconditions for operating the steam turbine, the steam turbine inlet valve (V1) is opened until steam outflow sounds. After 3 minutes the drain of condensate from the drain path (E1, V1, E2, V2, E3) is finished, the steam turbine inlet drain valve (V2) is closed, and then the drain task of the steam turbine inlet valve is finished. The valve V1 is opened fully and the steam turbine warming-up valve (V5) is opened, and then the steam turbine warming-up task is carried out. Normally, to operate the steam turbine, the main valve (V3) on the path for working the steam turbine is opened. Then, to close the path for the warming-up task of the steam turbine, the steam turbine warming-up valve (V5) is closed. Finally, after the steam turbine is operated, the turbine case drain valve (V8) and the steam turbine chest drain valve (V9) are closed to isolate the drain path of the steam turbine.

In order to complete the firing of the burner in the boiler plant, the ignition gas throttle valve (V14) is opened until its pressure becomes 1 kg_f and the ignition-sparker makes a spark for burning the burner. Then the atomizing steam throttle valve (V13) and the fuel oil throttle valve (V12) are opened to provide atomizing steam and fuel oil, and regulated to increase the temperature of steam at the rate of 55 °C per hour.

To generate the operating procedure described above, the goal tree is expanded first to search for these primitive operations. The steps of the goal tree expansion are as follows. For the expansion of the operational situation (Material-Driver-Start-Up-Situ) of a unit to supply the driving force needed to transport material, the system finds the FDF which supplies the driving force and then expands its high level operational situation into its low-level one (FDF-Start-Up-Situ). In addition, for the operation of the FDF, FDF-Start-Up-Situ is expanded into the operational situation (FDF-Damper-Auto-Mode-Situ) that controls the amount of air according to the operation amount of the FDF, and then is expanded as the low-level one (ST-Start-Up-Situ) which makes the steam turbine operate.

To operate the steam turbine, the drain situation (ST-Drain-Situ) of the steam turbine is finished and the steam turbine is warmed up (ST-Warming-Up-Situ), and then primitive operations for formulating a steam flow path to work the steam turbine are created. The detailed description for operating the steam turbine is as follows.

The drain situation of the steam turbine is expanded into the drain situation (ST-Exhaust-BV-Drain-Situ) for the steam turbine exhaust block valve and the drain situation (ST-Case-Drain-Situ) for lines connected to the steam turbine case. Then the drain task for the steam turbine exhaust block valve is expanded as primitive operations for creating its drain path (ST-Exhaust-BV-Drain-Path-Open-Situ). After the drain path is created, the time-duration operation to last for the amount of time necessary for completing the drain situation (ST-Exhaust-BV-Drain-Time-Duration-Situ) is executed. After material in the drain path is removed, the flow-blocking operation is executed to block the drain path (ST-Exhaust-BV-Drain-Path-Close-Situ). For the warming-up situation (ST-Warming-Up-Situ) of the steam turbine, the drain situation of units and pipes on lines for the corresponding lines is executed. Meanwhile, if the line-up situation of fuel oil, fuel gas and ignition gas necessary to ignite the burner is finished, the ignition situation (Ignition-Situ) of the burner in the boiler plant starts at the boiler furnace. In order to complete the burning situation (Burning-Situ), atomizing steam (Atom-Steam-Inlet-Situ) and fuel oil (Fuel-Oil-Inlet-Situ) feed in the burner and are regulated to increase the temperature of steam at the rate of 55 °C per hour.

Finally, after the synthesis of an operating procedure for achieving Material-Driver-Start-Up and Burning-Situ is finished, we have found the repeated primitive operations (V1, V5, V11), appearing twice in the synthesized operating procedure (see Table 4). To achieve some operational situations in large-scale boiler plants, sometimes it is unavoidable to repeat the same primitive operation at least twice. To obtain more detailed operating procedures, the full history of changing operational situations and how the execution of a primitive operation at the present operational situation affects process states have to be recognized.

CONCLUSIONS

In this research, we have developed the following methodology for the automatic synthesis of operating procedures for large-scale boiler plants:

1. First, operational situations are represented as the general functional pattern, a script. A selection method of related primitive operation candidates from plant structure data and unit library by using general search knowledge has been developed for the application to boiler plants with different structure.
2. The proposed model includes time-duration operations and uses them to simulate the temporal changes in the process state. In addition, it includes the amount of operation as well as the simple on/off or open/close operations.
3. The constraint situation, condition situation, and priority knowledge are used to determine the primitive operations or the operational situations ordering in the process of goal tree expansion. The priority ordering knowledge is created by referring to the hazardous situation knowledge base.
4. We have used heuristic functions to simulate changes of the process because it reduces effort and time necessary for defining rules to simulate specific situations.

The operating procedure synthesis system proposed by this work can help synthesize a safe operating procedure in case of modifying a plant structure and help generalize heuristic knowledge of experts. And this system can be applied to operator training systems and developing an operation aid system capable of effectively coping with hazardous situations in chemical plants.

ACKNOWLEDGMENT

Financial support from the Korea Science and Engineering Foundation (Grant No. R01-2003-000-10218-0) and the MOCIE through the CANSMC is gratefully acknowledged.

REFERENCES

Aelion, V. and Powers G. J., "A Unified Strategy for the Retrofit Synthesis of Flowsheet Structures for Attaining or Improving Operating Procedures," *Comput. Chem. Eng.*, **5**, 349 (1991).

Ärzen, K. E., "Grafect for Intelligent Supervisory Control Applications," *Automatica*, **30**(10), 1513 (1994).

Crooks, C. A., Evans, S. F. and Macchietto, S., "An Application of Automated Operating Procedure Synthesis in the Nuclear Industry," *Comput. Chem. Eng.*, **18**, 385 (1994).

Foulkes, N. R., Walton, J. J., Andow, P. K. and Galluzzo, M., "Computeraided Synthesis of Complex Pump and Valve Operations," *Comput. Chem. Eng.*, **9**(10), 1035 (1988).

Fusillo, R. H. and Powers, G J., "A Synthesis Method for Chemical Plant Operating Procedures," *Comput. Chem. Eng.*, **11**(4), 369 (1987).

Fusillo, R. H. and Powers, G J., "Operating Procedure Synthesis using Local Models and Distributed Goals," *Comput. Chem. Eng.*, **9**(10), 1023 (1988).

Gerzson, M., Csaki, Z. S. and Hangos, K. M., "Qualitative Model Based Verification of Operating Procedures by High Level Petri Nets," *Comput. Chem. Eng.*, **18**(1), 565 (1994).

Hou, B. K., Hwang, K. S., Choi, S. H., Shin, D. and Yoon, E. S., *Operating Procedure Synthesis for Shutdown of Boiler Plants*, Submitted to 4th IFAC Workshop on On-Line Fault Detection and Supervision in the Chemical Process Industries (CHEMFAS-4), Jejudo (Chejudo) Island, Korea, June (2001).

Hwang, K. S., Tomita, S. and O'shima, E., "Development of a Computer-Based System for Synthesizing the Operating Procedure of a Chemical Plant. Organization of Knowledge for Plant Operation," *International Chem. Engng.*, **31**(1), 134 (1991).

Lakshmanan, R. and Stephanopoulos, G., "Synthesis of Operating Procedures for Complete Chemical Plantsi: Hierarchical, Structured Modelling for Nonlinear Planning," *Comput. Chem. Eng.*, **9**(10), 985 (1988).

Lakshmanan, R. and Stephanopoulos, G., "Synthesis of Operating Procedures for Complete Chemical Plants II: A Nonlinear Planning Methodology," *Comput. Chem. Eng.*, **9**(10), 1003 (1988).

Lakshmanan, R. and Stephanopoulos, G., "Synthesis of Operating Procedures for Complete Chemical Plants III: Planning in the Presence of Qualitative, Mixing Constraints," *Comput. Chem. Eng.*, **3**, 301 (1990).

Li, H. S., Lu, M. L. and Naka, Y., "A Twotier Planning Methodology for Synthesis of Operating Procedures," *Comput. Chem. Eng.*, **21**(1), 899 (1997).

Naka, Y., Lu, M. L. and Takiyama, H., "Operational Design for Startup of Chemical Processes," *Comput. Chem. Eng.*, **9**, 997 (1997).

O'shima, E., "Safety Supervision of Valve Operations," *J. of Chem. Eng. of Japan*, **11**(5), 390 (1978).

Pradubsripetch, D., Lee, S., Adriani, A. and Naka, Y., "Real Time Generation of Operating Procedures to Support Flexible Startup Operation," *Comput. Chem. Eng.*, **20**, 1203 (1996).

Rivas, J. R. and Rudd, D. F., "Synthesis of Failuresafe Operations," *AIChE J.*, **20**, 320 (1974).

Rostein, G. E., Lavie, R. and Lewin, D. R., "Automatic Synthesis of Batch Plant Procedures: Process-Oriented Approach," *AIChE J.*, **40**(10), 1650 (1994).

Soutter, J. K., *An Integrated Architecture for Operating Procedure Synthesis*, Ph. D. Dissertation, Loughborough University, Loughborough, UK (1997).

Tomita, S., Hwang, K. S., O'Shima, E. and McGreavy, C., "Automatic Synthesizer of Operating Procedures for Chemical Plants by use of Fragmentary Knowledge," *J. of Chem. Eng. of Japan*, **22**(4), 364 (1989).

Verwijs, J. W., Kosters, P. H., van den Berg, H. and Westerterp, K. R., "Reactor Operating Procedures for Startup of Continuously Operated Chemical Plants," *AIChE J.*, **1**, 148 (1995).

Verwijs, J. W., van den Berg, H. and Westerterp, K. W., "Startup Strategy Design and Safeguarding of Industrial Adiabatic Tubular Reactor Systems," *AIChE J.*, **2**, 503 (1996).

Viswanathan, S., Johnsson, C., Srinivasan, R., Venkatasubramanian, V. and Ärzen, K. E., "Automating Operating Procedure Synthesis for Batch Processes: Part I. Knowledge Representation and Planning Framework," *Comput. Chem. Eng.*, **22**(11), 1673 (1998).

Viswanathan, S., Johnsson, C., Srinivasan, R., Venkatasubramanian, V. and Ärzen, K. E., "Automating Operating Procedure Synthesis for Batch Processes: Part II. Implementation and Application," *Comput. Chem. Eng.*, **22**(11), 1687 (1998).