KIGAS Vol.5, No.1, March, 2001
(Journal of the Korean Institute of Gas)

시간논리와 표현논리를 이용한 운전절차 자동합성 시스템 개발

허 보 경 · †황 규 석
부산대학교 화학공학과
(2001년 1월 4일 접수, 2001년 3월 21일 채택)

Development of Automatic Synthesis System for Operating Procedures Using Temporal Logic and Description Logic

Bo Kyeng Hou, Kyu Suk Hwang
Dept. of Chem. Eng., Pusan National University, Pusan 609-735, Korea
(Received 4 January 2001; Accepted 21 March 2001)

요 약

운전절차 합성 시스템은 운전절차 합성에 필요한 시간과 노력을 줄여주고 좀더 자세하게 운전절차를 분석해 줄뿐 아니라 인적오류를 줄여주거나 제거해주는 역할을 수행한다. 또한 운전절차를 생성하는데 필요한 전문가들의 지식을 가지고 있어 새로운 상황에 사용할 수 있도록 한다. 그러나 기존의 시스템들은 많은 한계점을 가지고 있다. 따라서 본 연구에서는 이러한 문제를 해결하기 위해 시간논리와 표현논리로 공정의 특성지식과 시간적인 제약조건을 다루는 새로운 접근방법을 제안하여 보일러 공장에 적용하여 그 유효성을 입증하였다.

Abstract - OPS(Operating Procedure Synthesis) systems can reduce the time and effort involved in OPS, make the analysis more thorough and detailed, and minimize or eliminate human errors. And OPS systems capture the expertise needed to create operating procedures and allow this experience to be used in the new situations. But there are the limitations of the OPS techniques that have been used. So in order to resolve this problem, in this work we have proposed a new approach to utilize temporal constraints and specific process knowledge in temporal logic and description logic. We have demonstrated its remarkable effectiveness in a boiler plant.

Key words : Operating procedure synthesis, Temporal logic, Description logic

1. 서 론

화학장치 산업에 있어서의 이와 같은 대형 사고는 사고 회사의 기업존폐 자체에도 영향을 줄 뿐 아니라 '89년 미국 파사디나 석유 화학 공장 폭발 사고 이후, 그 공장에서 생산하는 제품인 폴리에틸렌 가격 폭등에서 경험하였듯이 제품공급 불안정으로 인하여 관련 산업전체에 큰 영향을 끼친다. 화학장치 산업의 제품은 기초원료 생산에서 완제품 생산에 이르기까지 다양하지만 중간 반응과정이나 제품으로 되어서까지 유해 화학물질로서 존재 할 수 있는 경우가 많기 때문에 이들을 적정 압력과 적정 온도, 적정 흐름과 기밀을 유지하지 못할 때는 인명살상, 중독 등의 중대재해를 동반하게 된다. 그리고 개시 조업(start-up)의 비용은 전체 설계와 건설비용의 5~15%정도를 차지하는 것으로 보고된 바 있다. 또한 개시 조업의 지연 사유를 살펴보면, 장치 부족으로 인한 지연이 40~61%, 디자인의 부적합으로 인한 지연이 10%, 조작 실수로 인한 지연이 13~30%을 차지하고 있다[1].

†주저자 : kshwang@hyowon.pusan.ac.kr

지하고 있다[1].

따라서 화학공장의 자동화와 안전성 향상을 위해서는 공장의 개시 · 정지 · 비상사태 · 보수 유지 조업에 관한 운전절차를 합성하고 분석하는 작업이 필요하다. 공정의 초기상태(initial state)로부터 최종 목표상태(final or goal state)로 제약조건(constraint)을 침범(violation)하지 않고 안전하게 운전목표를 달성시키는데 필요한 조작 절차를 생성하는 운전절차 합성시스템은 오류가 없는 운전절차를 합성하는데 도움을 주고 특정 공정에 대한 전문가들의 지식을 구조 · 문서화하여 보관하고 공유할 수 있다[2-4].

또한 조업자 교육용 시스템으로의 활용 뿐 아니라 공정의 이상상황에 유연하게 대처할 수 있는 시스템으로 발전할 수 있을 것으로 예상된다. 즉, 조업자는 공정의 이상상황을 인식한 후에 정상적인 공정상태로 되돌리기 위해 필요한 조작을 발견해야 한다. 화학공장은 운전상황간의 인과관계가 매우 복잡할 뿐 아니라 조업자가 인식해야 될 데이터의 양이 엄청나므로 적절한 대처 조작을 제시하는 것은 매우 어려운 일이다. 그러나 운전절차 합성 시스템은 공정의 이상을 발견하는 이상진단 시스템의 도움을 받아 공정의 이상을 없애는데 필요한 조작을 매우 효과적으로 제시해 줄 수 있다.

2. 제안된 시스템의 구조

본 연구에서 제안된 운전절차 합성 시스템은 Allegro common lisp for windows 5.01로 작성되어 있다. 시스템의 구조는 Fig. 1과 같이 구성되어 있으며, 각 구성요소의 기능은 아래와 같다.

(1) GUI(Graphic User Interface): 사용자가 대상공정의 연결관계, 공정의 초기상태와 목표상태, 제한조건 등과 같은 정보를 입력할 뿐 아니라 최종적으로 합성된 운전절차를 사용자에게 제시해 주는 모듈이다.

(2) Knowledge Base: 각 장치의 객체지향 조작연산자, 화학물질의 물리적 · 화학적 특성, 조업자의 경험지식 등을 라이브러리로 작성하여 저장해 둔 지식베이스이다.

(3) LOOM-OPSTEL: 사용자로부터 입력받은 대상공정의 지식과 데이터를 계층화할 뿐 아니라 논리적 질의에 응답하기 쉬운 형태의 네트워크로 변형시키는 모듈이다. 즉, 장치 사이의 물리적 · 기능적인 인과관계를 유추하고 운전절차의 합성에 필요한 정보들을 추출하는 역할을 수행한다.

(4) Mapping-Module: 운전상황의 인식을 위해 필요한 모듈로 조작에 의해 변화된 공정상태를 인식하여 운전상황의 존재유 · 무를 판단한다.

(5) Temporal-Logic-Verifier: 시간적인 제약조건을 체크하고 운전상황의 진행과정을 인식하기 위해 정의된 시간논리식을 확장하여 평가하는 모듈이다.

(6) Safety-Checker: 전역적 · 지역적 제약조건을 침범하는 위험상황이 발생하는지의 여부를 판단하는 모듈이다.

(7) Goal-Tree-Generator: 정형화된 목표트리를 생성할 수 있는 경우, 지식베이스를 참조하여 대상공정의 목표트리를 생성하는 모듈이다.

(8) OPSTEL-Planner: 중간목표 순서화, 조작연산자의 선택, 그리고 운전목표의 달성 여부를 체크하여 위험상황을 발생시키지 않고 목표상태에 도달하는데 필요한 운전절차를 합성하는 모듈이다.

(9) Simulator: OPSTEL-Planner가 제시한 후보조작의 실행으로 인한 공정상태 변화를 확인하는데 사용되는 모사기 모듈이다.

(10) Flow-Path-Searcher: 필요한 공정흐름 경로를 발견하기 위해 사용되는 탐색 모듈이다.

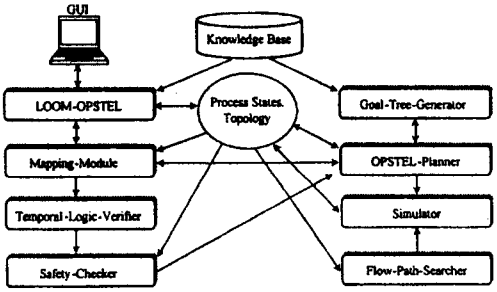

Fig. 1. System architecture

3. 공정구조의 표현 방법

본 연구에서는 이와 같은 문제를 해결하기 위해 장치가 가지고 있는 기능들을 장치중심의 상태함수로 표현한 객체지향(object-oriented) 조작 연산자(operator)모델을 제시한다. 객체지향 조작연산자의 각 슬롯(slot)은 장치 이름(name), 물질 수지(mass-balance), 에너지 수지(energy-balance), 장치의 운전을 위해 필요한 준비조건(operating-precondition), 운전 제약조건(operating-constraint), 입출력 포트(port)로 구성되어 있다(Fig. 2).

장치의 연결 구조(topology)는 새로운 장치의 정의 및 추가, 공정 흐름(process flow)의 방향성 표현이 용이한 유향그래프(digraph)를 사용하여 나타낸다. 유향그래프는 장치를 나타내는 노드(node, 원으로 표시), 노드와 노드 사이의 연결 관계를 나타내는 아크(arc, 선으로 표시)로 구성된다. 여기서, 아크에 표시된 노드쌍(node pair)은 흐름의 방향성을 의미한다. 예를 들어, 노드 A와 노드 B 사이에 존재하는 아크의 노드쌍이 {A, B}이면, 노드 A에서 노드 B로 흐름을 의미하며 만일 {A, B}와 {B, A}가 동시에 존재하면, 노드 A와 노드 B 사이에는 양방향의 흐름이 가능함을 의미한다. 또한 역류는 아크에 존재하는 노드쌍의 순서를 바꿔주어 나타낸다.

```
(defclass process-unit (generic-process-unit)
 ((name :accessor name :initform nil)
  (mass-balance :accessor mass-balance
    :initform
     '((sigma (mass-flow-{pi}) over pi in PORTS)
      (for-each j in *chemical-species*
        (sigma (x-{j}-{pi} * mass-flow-{pi})
          over pi in PORTS))
      (for-each pi in PORTS
        (1 - (sigma (x-{j}-{pi})
                    over j in *chemical-species*)))))
  (energy-balance :accessor energy-balance
    :initform
     '(((sigma (mass-flow-cp-{pi} * T-{pi})
               over pi in PORTS) + q-in)
       (sigma (mass-flow-cp-{pi}) over pi in PORTS)
               (for-each pi in PORTS
                 (mass-flow-cp-{pi} -
           (mass-flow-{pi} * cp-{pi})))))
  (operating-precondition :accessor operating-precondition
    :initform nil)
  (operating-constraint :accessor operating-constraint
    :initform nil)))
```

Fig. 2. Operator of process unit

그러나 유향그래프는 단순히 장치의 물리적인 연결구조와 물질흐름 방향만을 나타낼 뿐 그 연결의 기능적 의미는 알려주지 못한다. 따라서 본 연구에서는 장치 연결의 기능적 관계를 나타내기 위해 표현논리(description logics, DL)를 이용하는 방법을 제시하고자 한다.

DL은 객체의 정의에 필요한 정의들을 concept로 객체 사이의 관계를 role로 표현할 수 있는 정의언어(definition language)를 지원하며 기본적인 concept와 relation으로부터 복잡한 형태의 concept와 relation을 생성한다. DL이 사용하는 표현 언어는 Table 1과 2와 같다[4]. 예를 들어, 냉각 자켓(cooling jacket)을 가지고 파이프에만 연결되어 있는 장치와 냉각 자켓이 없고 3 부분으로 구성되어 있는 장치의 concept는 각각 다음과 같다. 본 논문에서는 표기의 일관성을 유지하기 위해 concept는 대문자로 시작하고, role은 소문자로 시작한다.

$$\text{Device} \sqcap (\forall \text{connected-to.Pipe}) \sqcap (\exists \text{has-part.Cooling-jacket}) \quad (1)$$

$$\text{Device} \sqcap (\forall \text{has-part.} \neg \text{Cooling-jacket}) \sqcap (\geq 3 \text{ has-part}) \quad (2)$$

이와 같이 정의된 장치의 타입을 정의하는 concept와 relation은 DL의 Tbox에 저장되며, 다른 장치의 concept를 정의하는데 이용된다. 예를 들어, 냉각 반응기(cooled reactor)와 교반기가 달려 있는 냉각 반응기의 concept는 각각 다음과 같이 정의된다.

$$\text{CoolReactor} := \text{Reactor} \sqcap (\exists \text{has-part.Cooling-jacket}) \quad (3)$$

$$\text{StCoolReactor} := \text{CoolReactor} \sqcap (\exists \text{has-part.Stirring-unit}) \quad (4)$$

Table 1. Role expressions

Description	Syntax	Semantics
top role	$\top \times \top$	$\Delta^x \times \Delta^x$
conjunction	$R \sqcap S$	$R^x \cap S^x$
disjunction	$R \sqcup S$	$R^x \cup S^x$
composition	$R \circ S$	$R^x \circ S^x$
identity	$id(C)$	$\{<d, d> \mid d \in C^x\}$
inverse	R^{-1}	$\{<d, d'> \mid <d', d> \in R^x\}$
transitive closure	R^+	$\cup_{1 \le n}(R^x)^n$
transitive reflexive closure	R^*	$\cup_{0 \le n}(R^x)^n$

Table 2. Concept expressions

Description	Syntax	Semantics
top	$\perp$	Δ^x
bottom	$\top$	$\varnothing$
conjunction	$C \sqcap D$	$C^x \cap D^x$
disjunction	$C \sqcup D$	$C^x \cup D^x$
negation	$\neg C$	$\Delta^x - C^x$
exists restriction	$\exists R.C$	$\{d \in \Delta^x \mid R^x(d) \cap C^x \neq \varnothing\}$
value restriction	$\forall R.C$	$\{d \in \Delta^x \mid R^x(d) \subseteq C^x\}$
number restriction	$\geq nR$	$\{d \in \Delta^x \mid \lvert R^x(d)\rvert \geq C\}$
number restriction	$\leq nR$	$\{d \in \Delta^x \mid \lvert R^x(d)\rvert \leq C\}$
qualified number restriction	$\geq nR.C$	$\{d \in \Delta^x \mid \lvert R^x(d) \cap C^x\rvert \geq C\}$
qualified number restriction	$\leq nR.C$	$\{d \in \Delta^x \mid \lvert R^x(d) \cap C^x\rvert \geq C\}$
exists restriction	$\exists A.C$	$\{d \in dom\ A^x \mid A^x(d) \in C^x\}$
value restriction	$\forall A.C$	$\{d \in \Delta^x \mid dom\ A^x \Rightarrow A^x(d) \in C^x\}$
attribute value map	$A = B$	$\{d \in \Delta^x \mid A^x = B^x\}$
attribute value map	$A \neq B$	$\{d \in \Delta^x \mid A^x \neq B^x\}$

Tbox(terminological component)에 정의되어 있는 concept는 장치의 인스턴스(instance)로 구체화되어 Abox(assertional component)라는 곳에 저장된다.

REACTOR2: has_part COOL17,
REACTOR2: Reactor,
COOL17:Cooling-jacket (5)

Object REACTOR2는 Reactor의 인스턴스이고 role has-part에 의해 Cooling-jacket의 인스턴스인 COOL17과 연관되어 있음을 보여준다. DL은 Tbox안에 있는 concept들을 분류하여 각 객체가 속해 있는 concept를 찾아내므로 REACTOR2가 CoolReactor임을 유추한다.

본 연구에서는 대규모 화학공장의 운전절차 합성을 위해 개발된 지식 기반 구조인 LOOM-OPSTEL은 DL 기반의 프로그래밍 프레임워크(programming framework)인 LOOM [4]을 기본으로 하여 장치중심의 상태함수로 표현된 객체지향 조작연산자 모델의 자료 검색과 추론에 필요한 계층구조를 형성하도록 설계하였다.

본 연구에서는 DL을 이용하여 효율적으로 장치 사이의 물리적 · 기능적인 인과관계를 적절하게 표현할 수 있을 뿐 아니라 운전절차의 합성에 필요한 정보들을 유용하게 추출할 수 있다. 그리고 LOOM-OPSTEL의 지식 계층화 구조는 최상위 레벨의 지식 유형을 표현하는 DL layer, DL로 표현된 지식의 구체적인 정보를 가지고 있는 장치의 객체지향 조작연산자 모델 layer, 그리고 최하위 레벨인 각 장치의 객체지향 조작연산자 모델에 정의되어 있는 물질 수지와 에너지 수지의 제한조건 전파 모델 layer로 구성되어 있다(Fig. 3).

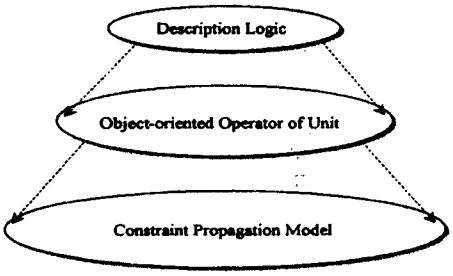

Fig. 3. LOOM-OPSTEL knowledge framework

4. 공정상태의 변화 모사 방법

제약조건 전파(constraint propagation)란 변수를 나타내는 셀(cell)과 셀 사이의 제약조건(constraint)을 이용하여 셀의 값을 결정하는 방법을 말한다(Fig. 4). 예를 들어, 셀 A와 셀 B의 값을 알고 있는 경우는 constraint adder의 룰(rule)인 A + B = C를 이용하여 셀 C의 값을 구하고, 셀 A와 셀 C의 값으로 셀 B의 값을 구하는 경우는 B = C - A 룰을 이용한다. 나머지의 경우도 위와 같은 방법으로 구한다.

즉, 임의의 공정 변수 값이 바뀌는 경우, 제약조건으로 연결되어 있는 다른 공정 변수의 값을 자동으로 변화시켜 준다. 이러한 계산능력을 이용하여 조작에 의한 공정상태의 변화를 모사할 뿐 아니라 사용자가 입력한 초기상태와 목표상태의 논리적 타당성을 검증하기 위해 의존 네트워크(dependency network) 기능을 이용할 수 있다. 즉, 사용자가 논리에 맞지 않는 서로 상반된 상태를 입력하는 경우, 제약조건 전파는 의존 네트워크 기능을 사용하여 이러한

모순을 유발시키는 원인을 찾아내어 논리적으로 타당한 초기상태와 목표상태의 입력이 가능하도록 하는 것이다.

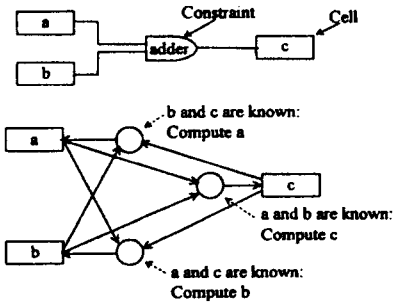

Fig. 4. Constraint propagation

5. 공정 제약조건의 표현과 조작연산자의 선택 방법

일반적으로 복잡한 화학공장의 운전절차 자동합성을 위해서는 일반화된 화학공장의 표준운전절차(Standard Operating Procedure, SOP)를 이용해야 가능하다. 왜냐하면 엄청나게 많은 운전목표들 중에서 과연 어느 목표를 우선적으로 달성해야 되는지의 문제와 한 목표를 달성하기 위해 2개 이상의 조작을 필요로 하는 경우, 과연 많은 조작들 중에 어느 조작을 선택해야 하는지의 문제를 해결할 수 없다.

기존의 운전절차 합성 시스템은 전역적 제약조건과 지역적 제약조건(특정 장치의 가동을 위한 운전 준비조건)은 이용하지만 임의의 시간간격 동안에 존재하는 시간적 제약조건을 표현하지 못하므로, Fig. 5와 같은 공정에서 A의 x리터와 B의 y리터가 될 때까지 C로 채우는(charging) 운전절차나 A, B가 C로 유입되는 동안에 C안의 A의 농도가 u와 w 사이의 일정한 값을 유지하는데 필요한 운전절차를 합성할 수 없다.

본 연구에서는 과거·현재·미래에 대한 반드시 만족되어야 하는 운전상황의 인과적이고 시간적인 관계나 제약조건들을 시간논리로 규정하여 잘못된 탐색경로를 최소화하는데 이용하는 전략을 제안한다. 즉 상태공간상에 존재하는 공정변수와 규정된 운전상황으로 정의된 시간논리식의 값을 체크한 다음, 그 정보를 이용하여 후보 조작연산자를 선택한다.

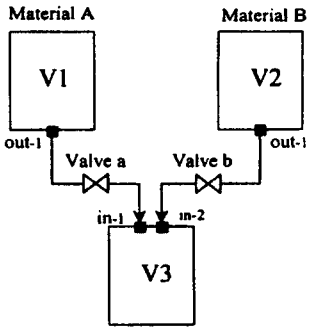

Fig. 5. A process

효과적인 연산자의 선택과 위험상황 발생 여부를 확인하기 위해서 U(Until), □(Always), ◇(Eventually), ○(Next)로 구성된 일차(first order) 선형 시간논리(Linear Temporal Logic, LTL) 연산자와 ∀(All), ∃(Some)과 같은 한정사(quantifier), 그리고 ∧(And), ∨(Or), ⇒(Implies), ¬(Not)와 같은 논리 기호(logical symbol)를 이용하여 전역적·시간적 제약조건을 표현한다. LTL의 기본적인 작성규칙과 그 의미는 Table 3과 같다. 여기서, f, f1, f2는 formula로써 흐름경로의 형태와 상태변수의 값을 맵핑하여 운전상황을 파악하는 연산자이다. 흐름경로의 형태에 관한 제약조건인 경우, flow-mapping-operator(start-unit, start-port, target-unit, destination-port, [material, {material-condition}], <time>)를 사용한다. flow-mapping-operator의 인수 중에서 {material-condition}과 <time>은 optional 인수이며, <time> 인수는 제약조건을 만족시키는데 필요한 지속시간을 나타낸다.

Table 3. Meaning of temporal modality

Temporal Modality	Meaning
$\bigcirc f$	f holds in the next state
$\Box f$	f holds in current state and in all future states
$\Diamond f$	f either holds now or in some future state
$f_1 \cup f_2$	Either now or in some future state f_2 holds and until that state f_1 holds

시작장치(start-unit)와 목표장치(target-unit) 사이의 흐름경로 형태는 막혀 있는 구간(Block), 폐쇄된 구간(Trap), 흐르고 있는 구간(Flow), 그리고 분기된 구간(Branch)으로 분류된다(Fig. 6).

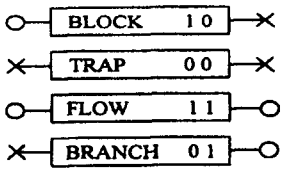

Fig. 6. Type of flow conditions

상태변수의 값에 관한 제약조건인 경우, state-mapping-operator(unit-name)를 사용하여 장치의 조작연산자의 슬롯 값을 참조하여 만족여부를 확인한다. 예를 들어, 온도(Temp), 압력(Press), 농도(Conc), 수위(Level) 등과 같은 공정변수 연산자나 퍼지 유무를 확인하는 Purge 연산자가 있다.

또한 $\forall[x:\Upsilon(x)]f \equiv \forall x.\Upsilon(x) \Rightarrow f$는 $\Upsilon(x)$을 만족하는 모든 x가 참이면, f가 유지됨을 의미하고, $\exists[x:\Upsilon(x)]f \equiv \exists x.\Upsilon(x) \wedge f$는 $\Upsilon(x)$을 만족하는 어떤 x가 참이면, f가 유지됨을 의미한다.

예를 들어 Fig. 5와 같은 공정의 시간적 제약조건을 LTL로 나타내면 다음과 같다.

(1) V1에 있는 물질 A가 x리터이고 V2에 있는 물질 B가 y리터가 될 때까지 V3에 채운다(단, 두 흐름은 서로 독립적인 비동기화 결합이다).

(Flow(V1, port-1, V3, port-1, [A])∨
Flow(V2, port-1, V3, port-2, [B]))∪
(Level(V1, [A, {x}])∧Level(V2, [B, {y}]))

(2) V1의 물질 A, V2의 물질 B가 V3에 유입되는 동안에 V3안의 A의 농도가 u와 w 사이의 일정한 값을 유지한다(단, 두 흐름은 서로 종속적인 동기화 결합이다).

(Flow(V1, port-1, V3, port-1, [A])∧
Flow(V2, port-1, V3, port-2, [B]))∪
(Concentration(V3, [A, {u<x(A)<w}])))

공정의 현재상태에서 선택 가능한 조작연산자의 선택은 조작에 의한 공정상태의 변화를 인식한 후, 시간논리로 표현된 제약조건들의 침범여부를 결정함으로 이루어진다. 여기서, 사용되는 공정제약조건의 진행(progression) 알고리즘은 Fig. 7과 같다.

Inputs: An *LTL* formula f and a state s
Output: A new *LTL* formula f' representing the progression of f through the state s

Algorithm *Progress(f, s)*
Case
1. $f=\emptyset$ ($\emptyset$ contains no temporal modalities): f':= TRUE if $s \models f$, FALSE otherwise
2. $f=f_1 \wedge f_2$: $f' := Progress(f_1, s) \wedge Progress(f_2, s)$
3. $f=f_1 \vee f_2$: $f' := Progress(f_1, s) \vee Progress(f_2, s)$
4. $f=\neg f_1$: $f' := \neg Progress(f_1, s)$
5. $f=\bigcirc f_1$: $f' := f_1$
6. $f=f_1 \cup f_2$: $f' := Progress(f_2,s) \vee (Progress(f_1,s) \wedge f)$
7. $f = \Diamond f_1$: $f' := Progress(f_1,s) \vee f$
8. $f = \Box f_1$: $f' := Progress(f_1,s) \wedge f$
9. $f=\forall[x:\Upsilon(x)]f_1$: $f' := \wedge_{(c:s \models \Upsilon(x/c))} Progress(f_1(x/c),s)$
10. $f=\exists[x:\Upsilon(x)]f_1$: $f' := \vee_{(c:s \models \Upsilon(x/c))} Progress(f_1(x/c),s)$

Fig. 7. Progression algorithm

6. Case study

본 연구에서 다루는 대상공정은 보일러 공장의 버너 영역으로 연료가스, 기름, 공기, 점화가스, 그리고 연료 기름을 분사시키는 atomizing steam은 각각 해당 라인을 통하여 버너주위로 유입되어 ignitor에 의해 점화된 후에 연소된다(Fig. 8).

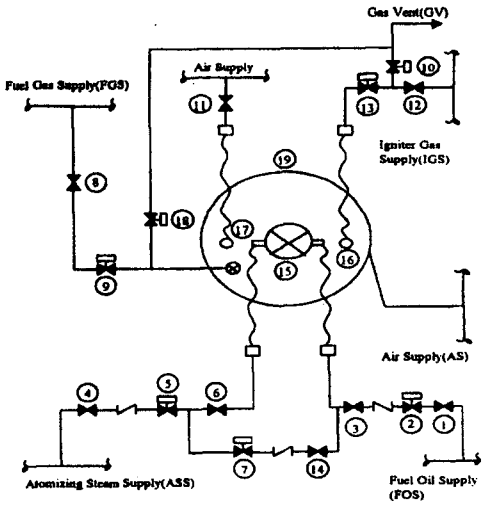

Fig. 8. Burner section of boiler plant

특히 버너점화 작업은 보일러 플랜트의 운전과정 중에서 가장 복잡한 중간목표 상호작용을 보이므로 제시한 방법론을 해당공정에 적용하여 그 유효성을 검증하고자 하고자 한다. 이 공정의 초기상태와 목표상태는 Table 4와 같다.

Table 4. Initial and final states

Units	Valve Number	Initial state	Final state
Fuel Oil Burner Block Valve	①	open	open
Fuel Oil Trip Valve	②	closed	open
Fuel Oil Burner Throttle Valve	③	closed	open
Atomizing Steam Burner Valve	④	open	open
Atomizing Steam Trip Valve	⑤	closed	open
Atomizing Steam Burner Throttle Valve	⑥	closed	open
Scavenge Valve	⑦	closed	open
Fuel Gas Burner Cock Valve	⑧	closed	closed
Fuel Gas Trip Valve	⑨	closed	closed
Igniter Gas Vent Valve	⑩	open	open
Flame Scanner Cooling Air Valve	⑪	open	open
Igniter Gas Block Valve	⑫	open	open
Igniter Gas Valve	⑬	closed	closed
Scavenge Block Valve	⑭	closed	closed
Burner	⑮	open	open
Igniter	⑯	off	off
Flame Scanner	⑰	on	on
Fuel Gas Vent Valve	⑱	closed	closed
Air Register	⑲	closed	open

이러한 공정의 운전절차 합성이 어려운 이유는 Igniter Gas valve, Scavenge Block Valve, 그리고 Igniter와 같이 장치의 조작이 한번 이상 실행되어 초기상태와 목표상태가 동일하고 중간목표간의 복잡한 상호작용이 존재할 뿐 아니라 다음과 같은 제약조건을 내포하고 있기 때문이다.

(1) 버너가 점화되기 전에는 버너 주위로 유입되는 공기흐름은 차단되어야 한다.

□¬Flow(AS, p2, AR, p1,[air])∪
((Flow(ASS, p2, Burner, p1, [steam])∧
Flow(FOS, p2, Burner, p1, [oil])∧
Flow(IGS, p2, Igniter, p1, [gas])∧
On(Igniter))

(2) 버너를 점화하기 이전에 먼저 버너 안에 존재하는 모든 불순물들을 완전히 제거하는 scavenge 작업이 완료되어야 한다.

(¬Flow(ASS, p2, Burner, p1, [steam])∧
¬Flow(FOS, p2, Burner, p1, [oil])∧
¬Purge(Burner)∧
¬Flow(ASS,p2, Burner, p2, [steam]))
⇒○Flow(ASS, p2, Burner, p2 ,[steam], <2minute>)

(3) scavenge 작업은 2분의 시간이 경과한 후, 해당 scavenge의 흐름경로를 즉시 차단해야 완료된다.

Flow(ASS, p2, Burner, p2 ,[steam], <2minute>)
⇒○¬Flow(ASS, p2, Burner, p2, [steam])

(4) Ignitor에 의해 버너가 점화되면, 점화기로 유입되는 가스는 차단되어야 한다.

(Flow(ASS, p2, Burner, p1, [steam])∧
Flow(FOS, p2, Burner, p1, [oil])∧
Flow(IGS, p2, Igniter, p1, [gas])∧
On(Igniter))
⇒○¬Flow(IGS, p2, Igniter, p1, [gas])

(5) Fuel oil을 분사시키는 atomizing steam이 버너로 유입되고, 버너가 scavenge작업으로 퍼지된 후에 fuel oil이 버너로 유입되어야 한다.

(Flow(ASS, p2, Burner, p1, [steam])∧
¬Flow(FOS, p2, Burner, p1, [oil])∧
Purge(Burner))
⇒○Flow(FOS, p2, Burner, p1, [oil])

(6) Fuel oil이 버너로 유입되고 있는 중간에 scavenge 작업이 이루어져서는 안 된다.

Flow(ASS, p2, Burner, p2 ,[steam])⇒
□¬Flow(FOS, p2, Burner, p1, [oil])

(7) 버너가 점화되어 ignitor로 유입되는 가스가 차단되면, 점화기에 공급되는 전원은 차단되어야 한다.

(Flow(ASS, p2, Burner, p1, [steam])∧
Flow(FOS, p2, Burner, p1, [oil])∧
¬Flow(IGS, p2, Igniter, p1, [gas])∧
On(Igniter))⇒○Off(Igniter)

(8) 버너의 점화작업을 위해 버너주위로 유입되는 공기의 흐름이 차단된 직후, 점화기에 전원을 공급한다.

(¬Flow(AS, p2, AR, p1, [air])∧
¬Flow(ASS, p2, Burner, p2 ,[steam])∧
¬Flow(ASS, p2, Burner, p1, [steam])∧

¬Flow(FOS, p2, Burner, p1, [oil])∧
Purge(Burner)∧Off(Igniter))⇒○On(Igniter)

(9) 점화기에 전원이 공급된 직후, 점화기에 가스를 공급하여 점화기를 점화한다.

(¬Flow(IGS, p2, Igniter, p1, [gas])∧
On(Igniter)))⇒○Flow(IGS, p2, Igniter, p1, [gas])

(10) 기름이 버너로 유입되기 위해서는 먼저 atomizing steam이 유입되어야 한다.

(□¬Flow(FOS, p2, Burner, p1, [oil])∪
Flow(ASS, p2, Burner, p1, [steam])

(11) Ignitor의 점화를 위한 점화용 가스는 항상 Gas Vent로 vent되어야 한다.

□Flow(IGS, p1, GV, p1, [gas])

Table 5와 같이 합성된 운전절차에서 볼 수 있는 것과 같이 각 중간목표의 달성에 필요한 조작의 선택 시에 시간논리로 표현된 제약조건들을 이용하므로 각 중간목표의 달성에 관여하지 않는 조작들이 제거됨을 알 수 있다. 조작실행으로 변화된 공정상태의 변화를 시간논리를 사용하여 인식한 후, 중간목표 사이의 상호작용을 피할 수 있는 장치의 조작연산자를 선택한다. 시간논리로 표현된 제약조건에 의해 조작연산자를 선택할 수 없는 경우, 장치들의 현재상태와 목표상태의 차이를 비교하여 그 차이를 줄일 수 있는 조작을 선택한다. 예를 들어, Atomizing Steam Burner Throttle Valve와 Air Register를 여는 조작이 해당된다.

7. 결 론

화학공장의 운전절차 합성을 위해 개발된 지식 기반 구조는 DL 기반의 프로그래밍 프레임워크로써 장치중심의 상태함수로 표현된 객체지향 조작연산자 모델의 자료 검색과 추론에 필요한 계층구조를 형성하여 장치 사이에 존재하는 물리적·기능적 연결관계 및 상태 정보를 명확하게 제시해 줌을 알 수 있었다. 또한 이와 같은 지식 기반 구조를 바탕으로 화학공장의 OPS에 필요한 지식베이스를 구성하였다.

제안된 방법론을 중간목표간의 상호작용이 매우 심한 보일러 공장의 버너점화 작업에 필요한 운전절차 합성에 적용해 본 결과, 그 유효성을 입증할 수 있었다.

Table 5. Synthesized operating procedure

Operating procedure	Progressed constraints	Applied constraints
close Air Register	(1), (10), (11)	(1)
on Igniter	(1), (8), (10), (11)	(8)
open Igniter Gas Valve	(1), (9), (10), (11)	(9)
open Atomizing Steam Trip Valve	(1), (2), (10), (11)	(2)
open Scavenge Valve	(1), (2), (10), (11)	(2)
open Scavenge Block Valve	(1), (2), (10), (11)	(2)
wait 2 minute	(1), (2), (10), (11)	(2)
close Scavenge Valve	(1), (3), (10), (11)	(3)
open Atomizing Steam Burner Throttle Valve	(1), (10), (11)	(10)
open Fuel Oil Trip Valve	(1), (5), (11)	(5)
open Fuel Oil Throttle Valve	(1), (5), (11)	(5)
close Igniter Gas Valve	(1), (4), (6), (11)	(4), (11)
off igniter	(1), (6), (7), (11)	(7)
open Air Register	(1), (6), (11)	(1)

참고문헌

1. Rafael, B. P.: Ph.D. Dissertation, Tokyo Institute of Technology, Yokohama, Japan (1997).
2. Lakshmanan, R. and Stephanopoulos, G., "Synthesis of Operating Procedures for Complete Chemical Plants- I. Hierarchical Structured Modelling for Nonlinear Planning", Comput. chem. Engng., 2, 985-1002 (1988).
3. Tomita, S., Hwang, K. S., O'Shima E., and McGreavy, C., "Automatic Synthesizer of Operating Procedures for Chemical Plants by use of Fragmentary Knowledge", Journal of Chemical Engineering of Japan, 22, 364-372 (1989).
4. Satter, U., "Terminological knowledge representation systems in a process engineering application", Ph.D Thesis, 1-152 (1998).