Temporal Logicに基づくシステム記述と推論

房岡璋　世木博久　高橋和子
（三菱電機中央研究所）

I. はじめに

ダイナミカル・システムの挙動を記述し，システムの諸性質を解析する一方法として，Temporal Logicに基づいた方法論について述べる。例として，簡単なプラント・コントローラを取り上げ，その制御規則の自動作成ができることを示す。

従来プラント・システムを記述する方法として，例えばladder diagramや特殊なプログラミング言語等が用いられているが，これらは推論方法に欠けるため，複雑なシステムを取り扱う場合には不十分であることが多い。システムを設計する際，設計者はシステムのダイナミカルな挙動を把握して，設計が要求仕様を満足していることを検証しなければならない。従って，システムの挙動を適切にモデル化して記述でき，システムの諸性質を論理的に推論できるような記述・推論体系が必要とされる。

ここでは記述・推論体系の基礎としてTemporal Logicを用いる。プラント・コントローラのようなシステムを対象とする場合は，そのダイナミクス，つまりシステムの時間的に推移する様々な性質を扱うことになる。例えばシステム・オペレータはシステムを初期状態から目標とする最終状態に導くにはどのように制御すれば良いかを考える。すなわち，制御の自動作成が問題となる。またシステムが現在の状態にあるのは，いかなる経路を辿ったかを推論する必要もおこる。これはcausal argumentと言われる。また逆に，システムが現在の状態からどのような異なった状態に到達しうるかを導くqualitative simulationの問題もある。この種の問題は，システム・ダイナミクスに内在する因果関係について，backwardにあるいはforwardに推論を繰り返し行なうことによって解かれる。そのために，ここでは“ω-graph”というシステム・ダイナミクスのグラフ表現を導入する。これにより上に述べた問題は，ω-graph上の簡単な決定問題に帰着されることを示す。

II. Temporal Logicによるシステム表現

A. システム記述

最初にここで用いるTemporal Logic（TLと略す）[1]について簡単に説明する。TLは古典論理に時間の概念を陽に導入したものである。様相オペレータとしてここでは，□（always），◇（eventually），○（next），$\mathcal{U}$（until）の四つを用いるが，その意味は次の通りである。

□P：いつもPが成り立つ。
◇P：いつかはPが成り立つ。
○P：次の時刻にPが成り立つ。
P$\mathcal{U}$Q：Qが成り立つ（最初の）時刻までPが成り立つ。

一般に，対象とするシステムは様々な抽象的レベルで記述できる。ここでは，システムはいくつかのobjectから構成されており，それらは階層構造をもつと考える。トップレベルに於いては，システムは二つのobject：コントローラと被制御対象から成っているとする。各objectは“flag”と呼ばれる内部状態を持つ。objectのflagは，

他の objectからは見えないとする。objectは他のobjectに対してメッセージを送ることができ，これをそのobjectの“action”と呼ぶ。actionは次の形で表現する。

if S then X←m

ここでSはobjectのflagを表わす述語であり，Xはメッセージmを送る相手のobjectであって，上式は，“Sが満足されている時はいつもaction X←mを行なう”と解釈される。actionは一単位時間内に完了するもの（これを“eventと”呼ぶ）と考える。コントローラの制御規則は，eventを用いた次のような式の集合で与えられるとする。

if p_i then A_i $(i=1,\cdots,n)$

但し，piは論理式で，Aiはeventを表わす。上式は次のTLにおける式と同じであると解釈する。

$$\Box[(p_1 \supset A_1) \wedge \cdots \wedge (p_n \supset A_n)]$$

ダイナミカル・システムの簡単な例として温度調節コントローラを持つボイラーを考える（Figure 1）。

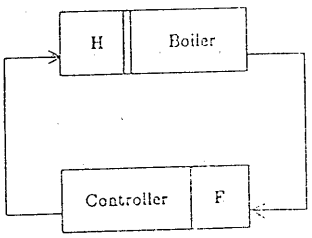

Fig.1 A model of boiler

ボイラーは，水温の状態を示す次式で規定される五つの内部状態Ei（i=1,…,5）を持つとする。

$$(E_1 \equiv \theta < \alpha);\quad (E_2 \equiv \theta = \alpha);$$
$$(E_3 \equiv \alpha < \theta < \beta);\quad (E_4 \equiv \theta = \beta);$$
$$(E_5 \equiv \beta < \theta)$$

但し，α，β（α<β）は，ある一定の温度を表わす。ボイラーは更に，点火の状態を示すもう一つの別の内部状態Hをもつ。ある時点でHがONならばボイラーは，いつかは水温がより高い状態に移るとする。ボイラーが点火するのは，コントローラからメッセージ“ON”を受け取った時だけで，点火を止めるのは，メッセージ“OFF”を受け取った時に限る。ボイラーがその内部状態をEからE'に変える時は，コントローラのflagにメッセージE'を送る。コントローラはその内部状態Fをボイラーからメッセージを受け取った時にのみ変えるとし，その時ボイラーにメッセージを送る。

以上のようなシステム・ダイナミクスをTLで定式化すると以下のように表現できる。

FRAME AXIOMS

$\Box[E_1 \vee E_2 \vee E_3 \vee E_4 \vee E_5]$
$\Box[\neg(E_i \wedge E_j)]$ *for each* $i,j\ (i \neq j)$
$\Box[F_0 \vee F_1 \vee F_2 \vee F_3 \vee F_4 \vee F_5]$
where $F_i = F \leftarrow E_i$ $(i=1,\cdots 5)$
and $F_0 = \neg[(F \leftarrow E_1) \vee (F \leftarrow E_2) \vee \cdots \vee (F \leftarrow E_5)]$
$\Box[\neg(F_i \wedge F_j)]$ *for each* $i,j\ (i \neq j)$
$\Box[E_i \wedge \circ E_j \equiv F_j]$ *for each* $i,j\ (j = i \pm 1)$
$\Box[H \vee \neg H]$

MESSAGE PASSING

$\Box[X \leftarrow m \supset X = m]$
$\Box[X = m \supset X = m\ \mathcal{U}\ X \leftarrow m']$
where X is a flag and m and m' are messages $(m \neq m')$

DYNAMICS

$\Box[E_1 \wedge \neg H \supset \circ E_1 \wedge F_0]$
$\Box[E_1 \wedge H\ \mathcal{U}\ E_2 \supset \Diamond(E_1 \wedge \circ E_2 \wedge F_2) \wedge E_1\ \mathcal{U}\ E_2]$
$\Box[E_2 \wedge \neg H \supset \circ E_1 \wedge F_1]$
$\Box[E_2 \wedge H \supset \circ E_3 \wedge F_3]$
$\Box[E_3 \wedge \neg H\ \mathcal{U}\ E_2 \supset \Diamond(E_3 \wedge \circ E_2 \wedge F_2) \wedge E_3\ \mathcal{U}\ E_2]$
$\Box[E_3 \wedge H\ \mathcal{U}\ E_4 \supset \Diamond(E_3 \wedge \circ E_4 \wedge F_4) \wedge E_3\ \mathcal{U}\ E_4]$
$\Box[E_4 \wedge \neg H \supset \circ E_3 \wedge F_3]$
$\Box[E_4 \wedge H \supset \circ E_5 \wedge F_5]$
$\Box[E_5 \wedge \neg H\ \mathcal{U}\ E_4 \supset \Diamond(E_5 \wedge \circ E_4 \wedge F_4) \wedge E_5\ \mathcal{U}\ E_4]$
$\Box[E_5 \wedge H \supset \circ E_5 \wedge F_0]$

B. 要求仕様のTLによる表現

ダイナミカル・システムの要求仕様としてよく扱われる性質としては，実現可能性，安定性，観測可能性がある。

1. 実現可能性

実現可能性（eventuality）とは，システムが初期状態から出発して，制御規則によって将来望ましい状態を実現しうるという要請である。これはTLでは次式のように表現される。

$$\textit{Initial condition} \supset \Diamond(\textit{the final state})$$

2. 安定性

安定性（stability）とは，システムがある状態に到達した後，引き続いてその状態をとり続けるという要請であり，

$$\varphi \supset \Box\varphi$$

と表現される。通常，実現可能性と安定性は一緒に使われて，次式のように書かれることが多い。

$$\textit{Initial condition} \supset \Diamond\Box(\textit{the final state}).$$

3. 観測可能性

観測可能性（observability）とは，被制御対象の状態をしかるべき遅延ののちにコントローラが観測しうるという要請である。この条件は，被制御対象の任意の状態Eに対して，あるコントローラの状態Fが存在して，

$$\Box[(E=a) \supset (\neg R\,U\,(F=a) \wedge \Diamond(F=a)]$$

と表わされる。ここでaは状態のとる値であり，Rは適当な遅延条件に対応する式とする。

被制御対象の状態は，その状態に到達する前にコントローラに伝えられてもよい。この時には，条件は次式のように表わされる。

$$\Box[(F=a) \supset (\neg R\,U\,(E=a) \wedge \Diamond(E=a))]$$

Ⅲ. 推論方法

A. ω-graph

TLによって表現されたシステムの諸性質について推論を行なうために，そのモデルを構成することからはじめる。そのためには，ここでは“ω-graph”と呼ぶグラフを導入する。ω-graphは様相論理における tableau method[2] に基づいている。TLの式Fが与えられたとき，まず最初にFを現在の時点に関する部分と，次の時点以降の部分とに分解する。すなわち，Fを論理的に等しい

$$\text{(1)} \quad \bigvee_i (A_i \wedge \bigcirc B_i)$$

の形に変形する。ここで A_i はリテラルの積あるいは true であるとする。この分解の規則（decomposition rules）としては次のようなものを用いる。

$$\Box F \longrightarrow F \wedge \bigcirc\Box F$$
$$\Diamond F \longrightarrow F \vee \{\neg F \wedge \bigcirc(\Diamond F, F)\}$$
$$F_1\,U\,F_2 \longrightarrow F_2 \vee \{\neg F_2 \wedge F_1 \wedge \bigcirc(F_1\,U\,F_2)\}$$

ここで，◇Fに対する分解規則に於ける（◇F, F）のような式を，“マーク付き式”と呼ぶことにする。このマークは，Fが将来のある時点で必ず実現されなくてはならないという要請を表わすためにつける。

Fのω-graphとは，Fのモデルのグラフ表現であり，グラフの各頂点がTLの式に対応し，各弧にはその時点で成立するリテラルが付随している。ω-graphの構成の仕方は次の通りである。

1) Fに対応する頂点 N_0 をつくる。これを initial node と呼ぶ。
2) Fを(1)の形に分解し，式 B_i に対応する頂点 N_i がすでにあれば，N_0 から N_i へラベル A_i をつけた弧をつくる。もし N_i がなければ，新たにつくる。
3) 各 N_i に対し，今度はそれを initial node として，もはや新たな頂点・弧がつくられなくなるまで2)を繰り返す。

ここで，各弧につけられたリテラルをその弧の“handle”と呼ぶことにする。また，マークの付いた式に対応する頂

点を“transitive node”，他の頂点を“ω-node”と言う。ω-graphの有向道をFの“挙動（behavior）”と言い，特に各頂点に対して，そのloopを高々一つしか含まない挙動を“skelton behavior”と呼ぶ。Fに対応するinitial nodeがω-nodeであり，かつω-nodeが閉路をもつならば，その閉路を構成する各弧のhandleの無限列は，Fの一つのモデルを与える。

ω-graphはω-language[3]に対する有限状態オートマトンと見なすことができる。従って，FをTLの式，Gfをそのω-graph，L(F)をGfに対応するオートマトンにより受理される言語とするならば，次の関係が成立する。

(i) $F = F_1 \wedge F_2 \iff L(F) = L(F_1) \cap L(F_2)$
(ii) $F = F_1 \vee F_2 \iff L(F) = L(F_1) \cup L(F_2)$
(iii) $F = true \iff L(F) = \Sigma^\omega$
(iv) $F = false \iff L(F) = \phi$
(v) $F_1 \supset F_2 \iff L(F_1) \subseteq L(F_2)$,

但し，Σはあるアルファベットを表わす。ω-languageの性質より，(v)の包含関係は決定可能である。

この対応関係に注意して，(v)の左辺の関係を決定するためにここでは次のような“containment algorithm”を用いる。今，F_1, F_2をTLの式としてG_i, $L(F_i)$ $(i = 1, 2)$をそれぞれF_iのω-graph，F_iに対応するω-languageとする。このとき $L(F_1) \supseteq L(F_2)$が成立する（簡単のため，これを“G_1がG_2を含む”と言う）か否かを決定するには，次の二つのルールを適用すればよい。

(1) [Checking Rule 1]
G_iのinitial nodeをN_i $(i = 1, 2)$とする。頂点の組(N_1, N_2)に対し，各N_iを始点とする弧E_{ij} $(j = 1, \cdots, m_i)$のhandle H_{ij}について，次の条件が成り立っているかを調べる。

(*) $H_{2j} \supset \bigvee H_{1j}$ for each j.

もし上式が成立していなければ，$L(F_1) \supseteq L(F_2)$は成立しない。さもなければ，各handleの組$(H_{1k}, H_{2\ell})$に対し，もし$H_{1k} \wedge H_{2\ell} \neq false$ならば次の条件が成立しているかを調べる。

$$\bigcup_k L(F_{1k}) \supseteq L(F_{2\ell}).$$

但し，F_{1k}, $F_{2\ell}$はそれぞれ弧E_{1k}, $E_{2\ell}$の終点の頂点に対応するTLの式を表わす。このルールは新たな頂点の組が現われなくなるまで適用する。

(2) [Checking Rule 2]
N_2がω-nodeであり，N_2を含む閉路（その閉路の頂点の列を$(N_2, T_1, \cdots, T_i, \cdots, T_k, N_2)$と書く）が存在する時は，更に次の二条件を満たすようなω-node W_iがG_1に存在するか否かを確める。

(a) N_1を始点とするbehaviorでその頂点の系列が$(N_1, \cdots, S_m, W_i, \cdots, S_n, W_i)$と書ける。
(b) 頂点の無限列の組：$(N_1, \cdots, S_m)\cdot(W_i \cdots, S_n, W_i)^\omega$と$(N_2, T_1, \cdots, T_i, \cdots, T_k, N_2)^\omega$について，対応する順番の頂点の組は[Checking Rule 1]の条件(*)を満たしている。

もしこのようなW_iがG_1に存在しなければ$L(F_1) \supseteq L(F_2)$は成立しない。

この手続きは，G_1, G_2の頂点の数は有限なので，明らかに終了する。

B. ω-graph上の推論

プラント・コントローラに関する様々な推論は，そのダイナミクスに対応するω-graph上の簡単な決定問題に帰着される。典型的な推論として次のものが考えられる。

1. 制御規則の自動作成

制御規則の自動作成とは，システムを望ましい最終状態に導くような制御規則を自動的に導出する問題である。これは，Dynamics と Goal が与えられたとき，式：

$$\textit{Dynamics} \wedge \textit{Unknown Control Rules} \supset \textit{Goal}.$$

を満足するような Unknown Control Rules の式を決定することである。ω-graph を用いると，Goal の式に対応する ω-graph が，Dynamics ∧ Unknown Control Rules に対応する ω-graph を含む条件を求めることに他ならない。この手続きの例を Ⅳ で示す。Ⅱ.B で述べたシステムの実現可能性・安定性・観測可能性という性質の検証も，対応する式の ω-graph を構成し，上述の containment algorithm を適用することにより行なわれる。

2. Qualitative Simulation と Causal Argument

qualitative simulation [4] とは，システムの状態が時間とともにどのように推移してゆくか，すなわち，システムが現在の状態から到達しうる“定性的に異なる状態を次々に求めてゆく推論である。これは次式を満足するような未知の状態の系列 $X_1, X_2, \cdots$ を決定することに他ならない。

$$\textit{Dynamics} \wedge P \supset \Diamond(\neg P \wedge X_1)$$
$$\textit{Dynamics} \wedge X_1 \supset \Diamond(\neg X_1 \wedge X_2)$$
$$\cdots$$

但し，P は現在の状態を表わす述語である。この推論も ω-graph を用いて同様に行なえるが，この場合は skelton behavior のみに着目すれば十分である。

causal argument [4] とは，システムを現在の状態に導いた過去の状態を見出す推論である。すなわち，

$$\textit{Dynamics} \supset \Diamond(X_1 \supset \Diamond P),$$
$$\textit{Dynamics} \supset \Diamond(X_2 \supset \Diamond X_1),$$
$$\cdots$$

を満足するような過去の状態の系列 $X_1, X_2, \cdots$ を決定する問題であり，これは qualitative simulation を backward に行なうことになる。

Ⅳ. 具体例

ω-graph 上で制御規則の自動作成の手続きがどのように行なわれるかを見るために，Ⅱで述べたボイラーを例に取り上げる。

Figure 2 にボイラーの Frame Axioms と Dynamics に対応する ω-graph G_D を示す。G_D を用いて，システムを最終的にある安定な状態，例えば $E_2 \vee E_3 \vee E_4$ に導く制御規則を見出すことを考える。この場合，制御規則 C は次式を満たす。

$$\textit{Dynamics } D \wedge \textit{Controls } C \supset \Diamond\Box(E_2 \vee E_3 \vee E_4)$$

ボイラーの制御規則は，

$$\textit{if } E_i \textit{ then } H \leftarrow \textit{ON/OFF} \quad (i = 1,\ldots,5)$$

の形で表わされる。但し，$H \leftarrow$ ON/OFF は，H が ON に set されるか，または OFF に set されるかいずれか一方のみおこることを示す。この規則を TL の式で書き換えると，

$$C \equiv \Box[(E_1 \supset H_1) \wedge (E_2 \supset H_2) \wedge \cdots \wedge (E_5 \supset H_5)]$$
$$\equiv \Box[E_1H_1 \wedge E_2H_2 \wedge E_3H_3 \wedge E_4H_4 \wedge E_5H_5]$$

となる。ここで，H_i は

$$H_i = H/\neg H.$$

を満たす。

$D \wedge C$ に対応する ω-graph G_C は G_D に現われる E_i を E_iH_i $(i=1,\cdots,5)$ に置き換えたグラフになることは容易に分かる。

最初に，制御規則 C が実現可能性

$$\Diamond(E_2 \vee E_3 \vee E_4) \equiv \Diamond E_2 \vee \Diamond E_3 \vee \Diamond E_4$$

を満足する条件を考える。他の場合も同様に行なえるので，$\Diamond E_3$ を例にとる。この実現可能性 $\Diamond E_3$ に対する ω-graph G_E を Figure 3 に示す。条件：

(*) $\quad \textit{Dynamics } D \wedge \textit{Control } C \supset \Diamond E_3$

を満足するためには，G_E が G_C を含まな

ければならない。Ⅱで述べた containment algorithm を (G_E, G_C) に対して適用する。ここで，G_Eの頂点 Ω，G_Cの頂点 S_1, S_5は ω-node であることに注意する。

1. initial node の組 (S_0, T_0) に対しまずG_CにおいてS_0からS_1への弧に着目する。その handle は

$$E_1 H_1 \bar{H} F_0 \vee E_2 H_2 \bar{H} F_1$$

である。

2. 今，この弧が存在する，すなわちこの handle が false ではないと仮定する。次に，(S_1, T_0) に対して，Checking Rule を適用する。

3. S_1がループ $E_1 H_1 \bar{H} F_0$をもつと仮定すると，S_1は ω-node であるから，behavior $(E_1 H_1 \bar{H} F_0)^\omega$を含むことになる。しかるに，$T_0$は $(E_1)^\omega$ の形の behavior を持ち得ない。

4. 従って，もし

$$(E_1 H_1 H F_0 + E_2 H_2 H F_1) \cdot E_1 H_1 H F_0 = false$$

ならば，(**) は満足されない。これより，

$$H_1 = H.$$

を得る。

5. 同様な議論を繰り返すことにより結局，次のような H_iに対する条件を導くことができる。

$$H_1 = H, \quad H_2 = H \quad H_4 = \bar{H} \quad H_5 = \bar{H}$$

安定性に対する要請も同様な手続きで行なわれ，最終的に次のような制御規則を導出することができる。

if F_1 then $H \leftarrow ON$; if F_2 then $H \leftarrow ON$;
if F_4 then $H \leftarrow OFF$; if F_5 then $H \leftarrow OFF$

Ⅴ. むすび

プラント・コントローラのようなダイナミカル・システムを形式的に取り扱う方法論として，Temporal Logic に基づいた記述・推論方法について述べた。推論方法として ω-graph によるアプローチを導入した。ω-graph による表現は，システム・ダイナミクスについてのモデルを与え，その上で様々な推論を統一的に行なえるので，通常の natural deduction や tableau method よりもすぐれていると考えられる。現在，エキスパート・システムのような実際的応用のために，ω-graph 上の推論を自動的に行なうシステムを開発中である。

[参考文献]

[1] Manna,Z. and Pnueli,A. "Verification of Concurrent Programs, Part 1: The Temporal Framework," Report No. STAN-CS-81-836, Department of Computer Science, Stanford University, CA, June 1981.

[2] Wolper,P.L., "Synthesis of Communicating Processes from Temporal Logic Specifications," Report No. STAN-CS-82-925, Department of Computer Science, Stanford University, CA, August 1982.

[3] McNaughton,R., "Testing and Generating Infinite Sequences by a Finite Automaton," Information and Control 9 (1966) 521-530.

[4] de Kleer,J. and Brown,J.S., "Foundations of Envisioning," In Proc. AAAI-82. Pittsburgh, Pennsylvania, August, 1982, pp. 434-437.

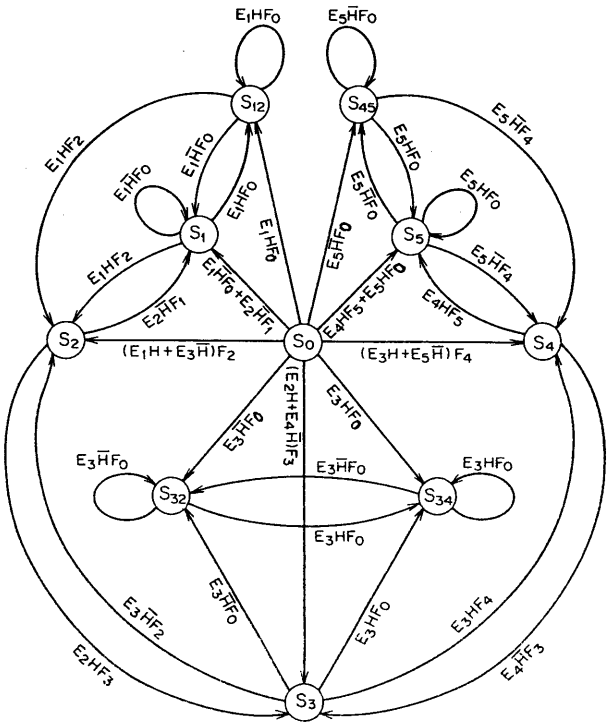

Fig.2 ω-graph G_D of the system dynamics

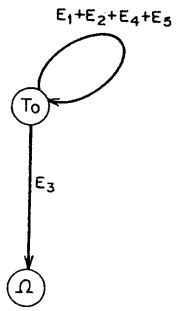

Fig.3 ω-graph G_E of the eventuality