有理時間を含むプログラムの仕様表現および動作解析*

1M－2

白銀 哲也（筑波大学 工学研究科）†　五十嵐 滋（筑波大学 電子・情報工学系）‡
塩 雅之（筑波大学 工学研究科）†　水谷 哲也（筑波大学 電子・情報工学系）‡

1　はじめに

有理時間を含むプログラムの仕様表現や検証・動作解析についての2種類の方法について提案する．1つは拍車を用いる方法[3]である．拍車は数学的には我々の立場ではそれ自身プロセスの特別な場合として扱われる概念であるが，直観的にはプロセスのスケジューラを一般化した概念であり，時相論理の next operator を一般化したものと考えることもできる．プログラムを拍車の履歴を入力とする一種のオートマトンとして捉える事により検証が容易になった．もう1つは形式的体系「束時相論理」(envelope system)[4, 5]である．時刻で真理値の変化する命題の真理集合を考え，閉包概念を特殊化したenvelopeと呼ぶ集合演算子を用いることにより証明に集合演算の直観を持ち込む事が可能となった。これらを実時間問題に適用して動作解析を行った例を示す．

2　有理拍車に基づいたプログラム検証

2.1　準備

適当な理論 T にプログラム変数等の表現のために可算個の定数を加えて，理論 T_c とする．時刻によってモデルを変える事で値の移り変わりを表す．拍車はプロセスと1対1で対応づけられ $\alpha, \beta, \ldots$ で表すが，そのもとの素拍車と呼ぶ命題 $\dot{\alpha}, \dot{\beta}, \ldots$ も定数として付け加える．また非負有理数全体の集合 $Q^{\geq 0}$ に正の無限遠点 ∞ を付け加えたTを時間の集合と考え，時刻から T_c のモデルへの写像 χ を軌跡と呼ぶ．プログラムの駆動は離散的に行なわれると仮定し，T_c の任意の論理式 p について真理集合 $\{t \mid \chi(t) \models p\}$ が左連続な階段関数状になる軌跡を考える．

2.2　プログラムの定式化

対象プログラムを処理単位ごとに拍車に注目して定式化する．得られた式をプログラム公理と呼ぶ．以下，軌跡は対象プログラム系 P のプログラム公理全体の集合 Γ_P を満たすようなものだけを扱う．

定義 1 軌跡 χ を固定すると，時間項 τ は，観察時刻 t（ただし任意の真理集合の不連続点）によってTの要素として解釈される．直観的には観察時刻以後最初にその事象が成立するまでかかる時間として解釈される．このことから，観察時刻 t_0 における時間項 τ の解釈を「t_0 からみた τ の立ち上がり」と呼ぶ．時間項の定義を以下に示す．

1. u が T の要素ならば u は時間項である．
2. p が T_c の論理式であるとき，$\lceil p \rceil$ は時間項である．これは，「観察時刻以後 p が最初に成立するまでにかかる時間」として解釈される．
3. τ_1，τ_2 が時間項であるとき，$\tau_1;\tau_2$ は時間項である．これは，「観察時刻から τ_1 が示す時間後に観察時刻を移した時 τ_2 が示す時間」として解釈される．

3について，特に τ_2 が有理数 x のとき，$\tau_1;x$ を $\tau_1 + x$ と表す．また素拍車 $\dot{\alpha}$ に対し $\lceil\dot{\alpha}\rceil;\lceil\neg\dot{\alpha}\rceil$ を拍車と呼び，α で略記する．対応するプロセスを α-driven であるという．

本体系の論理式は，時間項の順序関係及びそれらの否定や論理和から再帰的に構成する．曖昧さのないとき，$\lceil\cdot\rceil$ や；，$\tau = 0$ の $= 0$ は省略することが多い．

定義 2 軌跡 χ を固定すると観察時刻 t が与えられれば，t 以後の拍車の立ち上がる順番は定まる．これを表す列を（χ における）t からの初期拍車列と呼ぶ．

プログラムの初期状態，初期拍車列及び各立ち上がり間の時間差から計算実行列 (computation sequence) を得られ，動作解析が容易になる．以下プログラム検証の例を示すが，これらは[3]に詳しい．

2.3　例題1：Dekker の解

プログラム公理 並行プログラム系の例として Dekker の解 [1, 2] を挙げる．2つのプロセスに拍車 α, β を対応づけ，各処理単位は実行時間の上限下限について適当な仮定をおき，以下のように定式化した (P_{left}).

$$
\begin{aligned}
L_L \supset\ & (F_1 \supset (0.9 \leq \neg F_1 \supset \alpha < 0.9) \\
& \quad \wedge\ (\neg F_1 < 0.9 \supset \neg F_1 < \alpha)) \\
& \wedge (F_2 \supset (0.9 \leq \neg F_2 \supset \alpha < 0.9) \\
& \quad \wedge\ (\neg F_2 < 0.9 \supset \neg F_2 < \alpha)) \\
& \wedge (\neg(F_1 \vee F_2) \supset ((F_1 \leq \alpha) \vee (F_2 \leq \alpha))) \\
M_L \quad & \supset (90 \leq \alpha < 120) \wedge (\alpha = \alpha; \neg T_L) \\
P_L \quad & \supset 90 \leq \alpha
\end{aligned}
$$

*Representation and Analysis of Rationally Timed Programs

†Tetsuya Shirogane and Masayuki Shio, Doctoral Program in Engeneering, University of Tsukuba

‡Shigeru Igarashi and Tetsuya Mizutani, Institute of Information Sciences , University of Tsukuba

spurs from 0	time diff. min	time diff. max	state α drv.	state β drv.	state T_L	permission α-OK	permission β-OK	spur expect. α	spur expect. β	next spurs
			L_L	S_R	false	S_R	$\neg T_L$	$0.9 \leq \neg S_R$ $\supset \alpha \leq 0.9$	$0.7 \leq T_L$ $\supset \beta \leq 0.7$	$\beta < \alpha$ $\supset \beta \leq 0.7$
β		0.7	L_L	L_R	false	$L_R \wedge \neg T_L$	×	$0.9 \leq \neg(L_R \vee \neg T_L)$ $\supset \alpha \leq 0.9$	$\alpha < \beta \vee$ $\alpha = \beta = \infty$	$\alpha \leq 0.9$
$\beta\alpha$		0.9	S_L	L_R	false	×	S_L	$\beta < \alpha \vee$ $\alpha = \beta = \infty$	$0.7 \leq \neg S_L$ $\supset \beta \leq 0.7$	$\beta \leq 0.7$
$\beta\alpha\beta$		0.7	S_L	M_R	false	×	–	$\beta < \alpha \vee$ $\alpha = \beta = \infty$	$70 < \beta \leq 130$	$70 < \beta \leq 130$
$\beta\alpha\beta\beta$	70	130	S_L	P_R	true	T_L	–	$0.9 \leq \neg T_L$ $\supset \alpha \leq 0.9$	$70 < \beta$	$\alpha \leq 0.9$
$\beta\alpha\beta\beta\alpha$		0.9	L_L	P_R	true	P_R	*	$0.9 \leq \neg P_R$ $\supset \alpha \leq 0.9$	$70 - \delta_1 < \beta$	$\alpha \leq 0.9$
$\beta\alpha\beta\beta\alpha\alpha$		0.9	M_L	P_R	true	–	*	$90 < \alpha \leq 120$	$70 - (\delta_1 + \delta_2)$ $< \beta$	

表 1： Dekker の解の計算実行列 (初期状態 $L_L, S_R, \neg T_L$)

$$S_L \supset (T_L \supset (0.9 \leq \neg T_L \supset \alpha < 0.9) \wedge (\neg T_L < 0.9 \supset \neg T_L < \alpha)) \wedge (\neg T_L \supset T_L \leq \alpha)$$

飢餓回避 時刻 0 から最初に立ち上がる拍車が β の場合について可能な計算実行列を表 1 にまとめた. α-OK の欄は, α-driven のプロセスについて次の処理単位へコントロールが移るための必要条件を表す (β-OK も同様).

この実行列については各処理単位の実行時間の上限・下限の制約 (特に P_R の実行時間の下限) により可能な場合を減らす事ができ, プログラム公理を満たす軌跡では, 初期拍車列が β から始まるならば拍車は $\beta\alpha\beta\beta\alpha\alpha$ の順に立ち上がって危険領域 M_L に最初に到達し, それまでにかかる時間は各立ち上がりの上限と下限の累計から 70 から 134.1 単位時間の間であることがわかる. なお, 最初に α が β 以前に立ち上がる場合には同様な方法で 0 から 0.7 単位時間以内に M_L が実行されることが導ける.

2.4 例題 2: T 字路での合流

見通しの悪い一方通行路に T 字路で合流する事を考える. 合流する側の車は T 字路の直前で一旦停止するが, 衝突しないように アクセルを踏む条件を設定したい。この問題を並行プログラム系としてモデル化し, さらに拍車の立ち上がりを入力とするオートマトンとして捉え動作解析を行なった結果, 2.3節同様各ブロックの実行時間の上限下限から可能な初期拍車列の場合が減少し有効だった.

3 束時相論理

束時相論理は 2節の方法を真理集合に関する集合演算をもとに体系を再構成したものである. すなわち命題を真理集合として解釈し, それがなす集合束に閉包概念を特殊化した envelope と呼ぶ集合演算子をつけ加えて位相づける. より複雑な概念を集合として扱うことが可能となるが, これは観察時刻により異なるため, 複素平面で観察時刻を実軸, その観察時刻で観察される未来を虚軸として 2 次元的に解釈される. [5] では有理数時間を陽に扱い Dekker の解の飢餓回避問題や CSP の Distributed Termination に関する例題について適用し有効性を確認した.

4 結び

拍車を媒介に有理数時間を明示的に扱う形式的体系と束概念にもとづく形式的体系とを作成し, 並行プログラム系の動作解析を行なった. これらの体系は, 時間間隔のきめ細かい変動やその伝播を自然にかつ厳密に表現できる点および検証の大部分を数学の立場で算術として扱うことができるため, 純粋に論理的に行うより簡単である点で優れている。

参考文献

[1] E. W. Dijkstra : Co-operating Sequential Processes, *Programming Languages*, 1968, 43-112.

[2] 水谷 哲也, 五十嵐 滋, 小宮山 弘樹, 辻 尚史 : 一二の並行プロセスの検証問題について, 第 34 回プログラミングシンポジウム報告集, 1993, pp. 105–116.

[3] 白銀 哲也, 五十嵐 滋, 水谷 哲也, 塩 雅之 : 有理拍車に基づいたプログラムの検証ないし実時間動作解析, 応用数学合同研究集会報告集, 1995, pp. 21-1 – 21-6.

[4] 塩 雅之, 五十嵐 滋, 水谷 哲也, 辻 尚史, 白銀 哲也 : 時間の論理の束モデルを用いた並行プログラム計の検証, 情報処理学会第 50 回 (平成 7 年前期) 全国大会講演論文集, 1995, pp. 4-303 – 4-304.

[5] 塩 雅之, 白銀 哲也, 五十嵐 滋, 水谷 哲也 : 束時相論理による時間を含むプログラム仕様表現と動作解析, 応用数学合同研究集会報告集, 1995, pp. 22-1 – 22-6.