プログラムにおける実時間問題の ν-転換による解析と動作条件

富田康治[†] 辻 尚史[††] 五十嵐 滋[††]

本論文では，実時間を扱う並行プログラム系における性質の解析について述べる．ここでは，二つのプロセスからなるものを考え，一方のプロセスにバッファがあり，他方からはそれに空きがあるか否かを調べられないという機械構成の場合に，バッファが溢れないように通信を行うためにはどうするかということを考える．そのためには，実際の処理時間を知ることが必要となる．ここでは ν-転換を用いてこのプログラムの性質の解析を行うことにより，バッファが溢れずに動作するための十分条件を示した．ν-転換は解釈で時刻を陽に扱うため，このような実時間を扱う問題に対しても，解析を自然に行うことができる．

An Analysis of a Real Time Problem Using ν-Conversion and Its Safety Conditions

Kohji Tomita,[†] Takashi Tsuji[††] and Shigeru Igarashi[††]

This paper gives a solution to a kind of producer-consumer problems. We consider a program consisting of the following two parallel processes : one process is a producer, and the other is a consumer with a bounded buffer which receives data from the producer. This study gives sufficient conditions that make the program run without overflowing the buffer, when the producer does not know the status of the buffer. Using the ν-conversion which interpretes programs with rational time, properties of this program are analyzed and sufficient conditions of no overflow are obtained.

1. 序 論

計算機プログラムの正当性の検証等，理論的側面に関する重要性については言をまたない．逐次プログラムに対する検証の体系としては，ホーアの体系[4),2)]やト論理[6)]などがあり，正当性は原理的には証明可能である．しかし並行プログラムに対しては，ホーア論理を並行プログラムに対して拡張した体系[14),15)]や，時制論理 (temporal logic) を用いる方法[13),12),18),1)]など，さまざまな手法が提案されているにもかかわらず，インタリーブな実行のみを扱うなど欠点が多い．特にプログラムにおける明示的な時刻の扱いについては，扱うことを考えていないか，あるいは扱っても元々時間を陽に扱わない体系の上につけたものであるので整合性が悪い．

ν-転換 (ν-conversion)[7)] は数学的理論の上の論理式を ν-定義可能行為 (ν-definable act) に転換するものである．ν-定義可能行為を略して ν-行為 (ν-act) または単に行為 (act) ともいう．ν-行為はプログラムを一般化したものと考えることができ，その意味は形式論理の上で定義されている．これを用いることにより，プログラムの性質の解析や検証を厳密かつ一様に行うことができる[16),9),10)]．

ν-転換を用いた体系で意味を記述したり検証などを行うことにはさらに次のような利点がある．まず，ν-転換を用いた体系は，他の体系における公理や推論規則を定理として示すことができ，より一般的であるといえる[7)]．

また，ν-行為という一般的なものを対象としているので，異なる言語から ν-行為への翻訳規則を定めることにより，様々な言語のプログラムに対する検証等が一様に行える．また，異なる言語のプログラム間の関係を議論できる．すなわち，ν-転換を用いる体系はいわゆるプログラム言語に依存しない．CSP (Communicating Sequential Processes)[5)] から ν-行為への翻訳が文献 3) に示されている．

複数のプログラム間の同値性に関する議論も行われ

† 工業技術院機械技術研究所
Mechanical Engineering Laboratory, AIST
†† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

ており[17]，ν-行為の表現そのものをプログラムと考えて実行するための処理系も作られている[11]．

本論文では，実時間を扱う並行プログラム系における性質の ν-転換を用いた体系における解析について述べる．時制論理は，表現からむしろ時刻を排除する方向であるので，実時間を扱うプログラムに対する検証には向かない．一方，ν-転換を用いた体系では解釈で有理数の時刻を直接取り扱うことができるので自然かつ一様に行うことが可能である．このことを，次のような具体的な例に対して ν-転換の立場での解析を行うことにより示す．

ここでは，二つのプロセスからなる並行処理系において，一方のプロセスにバッファがあり，他方からはそれに空きがあるかどうかがわからないという機械構成の場合に，バッファが溢れない範囲で効率良くデータを送ることを考える．そのためには，実際の処理時間を知ることが必要となる．このような問題設定は，単方向の通信を行う場合や，通信に要する時間が非常に大きくバッファの空きを調べることが現実的でない場合などにおいて起こり得るものであるが，これまで解析の対象とされていなかった．ここでは，このプログラムに対応する ν-定義可能行為を解釈した軌跡 (locus) に対して，バッファが溢れずに動作する十分条件を示す．この例は実時間を扱うプログラムとして典型的なものであり，他の問題に対しても同様に適用することができる．

本論文の構成は次のとおりである．まず，2章で本論文に必要な ν-転換に関する諸定義を示す．次に3章で例題となる実時間問題を示して，4章で動作条件を導く．最後に5章で結論を述べる．

2. ν-転 換

この章では，ν-転換に関連する基本的な定義等のうち，本論文で必要なものについて述べる．

T を適当な数学的理論の体系（例えば FA[20]），$L(T)$ を T の言語とし，$x, y, z, \cdots$ を $L(T)$ の自由変数を表す超変数とする．このとき，νx を質変数 (qualitative) と呼ぶ．$L(T)$ の論理式 A に対して A の中の自由変数のうちの少なくとも一つを質変数に置き換えたもの，例えば $A[x, \nu x, z, \cdots]$，を ν-定義可能行為（ν-行為）といい，この操作を ν-転換という．ここで [ ] は変数などの出現または置換を表す．

ν-行為の解釈の最も単純なものとして基本解釈がある．この解釈では，$A[\boldsymbol{x}, \nu\boldsymbol{x}]$ は，$\boldsymbol{x}$ の新しい値が $\nu\boldsymbol{x}$ で表されるように，現在の状態を新しい状態に変更することを表現する．ここで，状態とは自由変数への値の割当てであり，$\boldsymbol{x}$ は $x_1, \cdots, x_n$ のことである．厳密には次のようになる．$L(T)$ のモデルを M，M のユニバースの元の名前がすべて定数として含まれるように $L(T)$ を拡張した言語を $L(M)$，$L(M)$ の論理式 F が M において割当て σ で真であることを $\sigma \vDash F$ で表す．M における $A[\boldsymbol{x}, \nu\boldsymbol{x}]$ の基本解釈は次のとおり．

現在の状態を σ とする．

1. $\sigma \vDash \exists \boldsymbol{y} A[\boldsymbol{x}, \boldsymbol{y}]$ であれば，A は行動可能 (actable) であるという．このとき M のユニバースの元 $\boldsymbol{\xi}$ で $\sigma \vDash A[\boldsymbol{x}, \boldsymbol{\xi}]$ なるものを一つ選び固定する．割当て σ のうち $\boldsymbol{x}$ の値のみを $\boldsymbol{\xi}$ に変更して得られる割当てを σ' とし，これを新しい状態として採用する．
2. $\sigma \vDash \neg\exists \boldsymbol{y} A[\boldsymbol{x}, \boldsymbol{y}]$ であれば，A は行動不可能 (unactable) であるという．このときは新しい状態は存在しない．

例えば，$\nu x = y \,\&\, \nu y = x$ という行為は変数 x と y の値を置換する行為を表す．

また，ν-行為に対する別の解釈として，行動述語 (predicate of action) と呼ばれる次の命題によって与えられる解釈もある．これを $Pa(A, \hat{\boldsymbol{x}})$ と表す．ただし，$\hat{\boldsymbol{x}}(t)$ は時刻 t における変数列 $\boldsymbol{x}$ の値であり，文脈より明らかなときは単に $\boldsymbol{x}(t)$ とも書く．ここで，t は非負有理数上を動く．

$$\begin{aligned}\forall t \forall \varepsilon > 0 ((\exists \boldsymbol{y} A[\hat{\boldsymbol{x}}(t), \boldsymbol{y}, t] \\ \supset \exists \delta > 0 (\delta < \varepsilon \,\&\, A[\hat{\boldsymbol{x}}(t), \hat{\boldsymbol{x}}(t+\delta), t])) \\ \&\, (\forall \delta \geq 0 (\delta < \varepsilon \supset \neg\exists \boldsymbol{y} A[\hat{\boldsymbol{x}}(t+\delta), \boldsymbol{y}, t+\delta]) \\ \supset \hat{\boldsymbol{x}}(t) = \hat{\boldsymbol{x}}(t+\varepsilon)))\end{aligned}$$

この命題は，行動可能な時刻には行為は無限小時間で行動し，行動可能でなければ変数の値が不変であることを述べている．本論文では解釈として，行動述語による解釈を考えることにする．

$Pa(A, \hat{\boldsymbol{x}})$ をみたす $\hat{\boldsymbol{x}}$ を行為 A の変数列 $\boldsymbol{x}$ の軌跡 (locus) という．$\exists \boldsymbol{y} A[\hat{\boldsymbol{x}}(t), \boldsymbol{y}, t]$ なる t が離散的であれば，$\hat{\boldsymbol{x}}$ は左連続の階段関数となる[8]．

さらに，行為は，変数の値を可能な限り保持するものとする（最小行動の原理，principle of the least action）．A^* は A の変数に順序をつけて，優先順位の高い変数の値をなるべく変えないように行動が決定される A に対応する行為であり，最小行動の原理を満たす．厳密な定義は文献 7) を参照．

最小行動の原理の下では，行為 $R[\boldsymbol{x}, \nu\boldsymbol{x}, t]$ に対し，$a(t) \supset R[\boldsymbol{x}, \nu\boldsymbol{x}, t]$ は A の行動する時刻を $a(t)$ が成り立つ時刻に限定した行為となっている．このような $a(t)$ を R の拍車（spur）という．ただし，$a(t)$ を満たす t は離散的であり，a は $\boldsymbol{x}, \nu\boldsymbol{x}$ を含まないものとする．ここで，R としては，スケジューリングなどに関連する部分は含まないようなアルゴリズムの本質的なもの（通常のプログラムに対応したもの）を考えることが多い．そして，$(a(t) \supset R[\boldsymbol{x}, \nu\boldsymbol{x}, t])^{\#}$ の行動述語による解釈が，$a(t)$ というスケジューラの下での行為 R に対応する逐次プログラムの実行と考えられる．通常行動述語で解釈の対象とするのはこのような実行系までも含めて抽象化した行為である．

ν-行為 R, S に対して，

$$((a(t) \,\&\, \exists \boldsymbol{y} R[\boldsymbol{x}, \boldsymbol{y}, t] \supset R[\boldsymbol{x}, \nu\boldsymbol{x}, t]) \,\&\, (b(t) \,\&\, \exists \boldsymbol{y} S[\boldsymbol{x}, \boldsymbol{y}, t] \supset S[\boldsymbol{x}, \nu\boldsymbol{x}, t]))^{\#}$$

は，$a(t)$, $b(t)$ という拍車の下で R と S を並行に実行するという行為を表すが，これを

$$a(t)\uparrow R \parallel b(t)\uparrow S$$

と書く．この解釈は，$a(t)$ が成り立つ時刻に R に対応するプロセスを実行し，$b(t)$ が成り立つ時刻に S に対応するプロセスを実行するという並行プログラムの実行に対応する．

ν-転換を用いた体系では，実時間を扱う並行プログラムの解析も自然かつ一様に行うことができる．プログラムについてのある性質 P の証明は，そのプログラム（に対応する ν-行為）を行動述語で解釈した結果の任意の軌跡を一つ考え，その軌跡が性質 P を満たすことを証明することにより行うことができる．実時間を扱うプログラムの解析の際には，P は時刻を表す変数 t を含み，異なる時刻における変数の値を表すことなどが必要となるが，このような場合に対しても全く同様に解析することができる．このことを次章以降で，具体例に基づいて示す．

3. プログラムの実時間問題

計算機がデータを生成して端末に送り，端末がそのデータを処理する時，計算機と端末をそれぞれ一つのプロセスと考えると，全体を一つの並行プログラム系と考えることができるので，そこで並行プログラムの問題が生じる可能性のあることがわかる[22]．

実際に起こった問題は，端末にはバッファがあり，データをある程度貯えることができるものの，計算機の側から空きがあるか否かは調べられないという機械構成の場合に，バッファが溢れない程度にできるだけ早くデータを送るにはどうするかというものである．

具体的には，計算機は HITAC M 170 で端末は Tektronix 4025 である．計算機は端末にベクトルコマンド列を送り，端末はそれを解釈してベクトルを生成しグラフィック画面に表示するが，このベクトル生成に時間がかかる．また，画面を消去するコマンドは他に比べて膨大な時間がかかる．

この時，計算機側が待たないでデータを早く送りつけすぎるとバッファが溢れてしまう．一方待ちすぎた場合には画面の表示は正しく行われるが，実行時間がかかりすぎてしまい，不当に待たされることになる．そこで，バッファが溢れない範囲でできるだけ早くデータを送ることが必要となる．

次のプログラム 1 はこの問題に関する計算機の動作の本質的な部分を表す．

```
νx=0;
compute(χx+1);
νu=0 & νt0=t;
νx=x+1 & νw[νx]=χx+1 & νu=u+f(χx+1);
loop
  compute(χx+1);
  if u≥c then
    begin
      wait(u−(t−t0));
      νu=0 & νt0=t
    end;
  νx=x+1 & νw[νx]=χx+1 & νu=u+f(χx+1)
end;
```

また，次のプログラム 2 は想定される端末の動作を表す．

```
νy=0;
loop
  if x>y then
    begin
      νy=y+1;
      display(w[y])
    end
end;
```

ここで，w は端末のバッファを表す一次元配列であり，x, y はそれぞれ計算機側，端末側における配列のインデックスを表す．実際にはバッファの大きさは有限（N とする）であるが，簡単のため w の大きさは無限であるとする．従って $x-y$ がバッファの中

のデータの個数を表す．また，バッファが溢れている状態は $x-y>N$ と表すことができる．

compute(χ_{x+1}) はデータ χ_{x+1} を計算している部分を表す．*wait*(v) は時間 v だけ実行を待つ．*display*$(w[y])$ は端末がデータを表示する部分である．$w[y]$ があるデータ χ_i であるとき，端末はこの処理に時間 $g(\chi_i)$ を要するものとする．プログラム1の f は g を推定した関数である．実際に起きた問題では画面を消去するコマンドのような例外を除いては，$f(\chi_i)$ は大体0にしても正常に動作したが，当初はその原因がわからなかった．

プログラム1の $\nu w[\nu x]=\chi_{x+1}$ の部分で通信を行っている．通信にかかる時間が一定であるとして，これを e_0 とする．

i 回目の $\nu w[\nu x]=\chi_{x+1}$ の実行開始から，次の通信または *wait* の実行開始までにかかる時間を $h(i)$ とする．ただし，$h(1)$ は $x=0$ の時の *compute*(χ_{x+1}) の実行開始から最初の通信の実行開始までにかかる時間とする．これは χ_i の計算にかかる時間，チャネルや TSS などの待ち時間の合計と考えられる．

この二つのプログラム1，2に対応する ν-行為をそれぞれ W, R とし（付録A参照），

$$a(t)\uparrow W \parallel b(t)\uparrow R$$

という ν-行為を行動述語で解釈した軌跡を考える．ただし，$a(t)$, $b(t)$ を成り立たせる時刻の間隔は等しいものとし，それを d とする．また，a と b とのずれを s とする．すなわち，$0\leq s<d$ かつ $a(t)\equiv b(t+s)$ とする．また，$f(\chi_i)$, $g(\chi_i)$ は d の自然数倍であるとする．明らかに $h(i)$ も d の自然数倍である．通信に e_0 だけ時間がかかることから，時刻 $t(\geq e_0)$ におけるバッファの中のデータの個数は $\hat{x}(t-e_0)-\hat{y}(t)$ と表され，x と y の軌跡の間には，任意の $t(\geq e_0)$ に対して，

$$0\leq \hat{x}(t-e_0)-\hat{y}(t)$$

という関係がある．以下で，このような軌跡についてバッファが溢れない条件を考える．

4. 動作条件

本章ではバッファが溢れない条件，すなわち任意の $t(\geq e_0)$ に対し，

$$\hat{x}(t-e_0)-\hat{y}(t)\leq N$$

が成り立つための条件を考える．W の動作は R には影響されないので，まず W だけについて考え，k 番目のデータを送る時刻 t_k を求める．そのため次のように T_m, X_m を定義する．T_m は m 回目の *wait* が終わった後，次にデータを送る時刻を表し，X_m はそのときの x の値を表す．

1. $T_0=a_0+h(1)$, $X_0=0$ とする．ただし，a_0 は $a(a_0)\,\&\,\forall t(a(t)\supset t\geq a_0)$ を満たす．
2. T_{m-1}, X_{m-1} が定義されているものとする．

$$\sum_{i=X_{m-1}+1}^{n-1} f(\chi_i)<c\leq \sum_{i=X_{m-1}+1}^{n} f(\chi_i)$$

なる n が存在するとき，そのような n に対して，T_m, X_m を

$$T_m=T_{m-1}+\max\left\{\sum_{i=X_{m-1}+1}^{n} f(\chi_i),\ \sum_{i=X_{m-1}+2}^{n+1} h(i)\right\},$$

$$X_m=n$$

と定義する．存在しないとき，

$$T_m=\infty,$$

$$X_m=\infty$$

とし，それより大きい m に対しては定義しない．

k 番目のデータを送る時刻 t_k は，$X_m<k\leq X_{m+1}$ なる m に対して，

$$t_k=T_m+\sum_{i=X_m+2}^{k} h(i)$$

である（**図1**）．また，明らかに

$$T_m\leq t_k<T_{m+1}$$

が成り立つ．

次に，R がバッファから k 番目のデータを読む時刻を考える．まず，次のように T'_j, X'_j を定義する．T'_j は，j 回目にバッファが空になった後，次にデータを送る時刻であり，X'_j はその時の x および y の値である．

1. $T'_0=t_1$, $X'_0=X_0(=0)$ である．

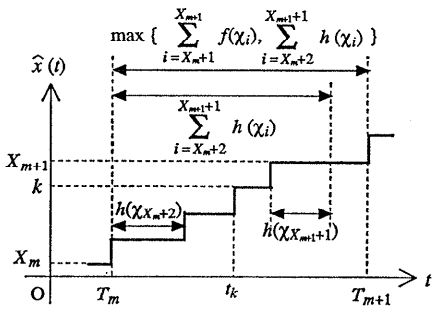

図1 k と m の関係
Fig. 1 The relation between k and m.

2. T'_{j-1}, X'_{j-1} が定義されているものとし，T'_j, X'_j を次のように定義する．

$$T'_{j-1}+\sum_{i=X'_{j-1}+1}^{n} g(\chi_i)\leq t_{n+1}$$

なる $n(\geq X'_{j-1})$ が存在すれば，これを満たす最小の n に対し，

$T'_j=t_{n+1}$,

$X'_j=n$

とする．存在しなければ，

$T'_j=\infty$,

$X'_j=\infty$

とし，それより大きい j に対しては定義しない．

以上の定義から k 番目のデータをバッファから読む時刻 t'_k は次のようになる（**図 2**）．

$X'_j<k\leq X'_{j+1}$ のような j に対して，

$$t'_k=T'_j+e_0+s+\sum_{i=X'_j+1}^{k-1} g(\chi_i).$$

このとき，明らかに

$$T'_j=t'_k<T'_{j+1}$$

が成り立つ．

以上の準備のもとで，バッファが溢れないための条件，任意の $t(\geq e_0)$ に対し $\hat{x}(t-e_0)-\hat{y}(t)\leq N$ は，$\hat{x}$, $\hat{y}$ が Pa を満たす軌跡であるということを用いて同値変形することにより，$k>N$ に対して，$t_k-(t'_{k-N}-e_0)\geq 0$ が成り立つことと同値であることを示すことができる．これは，次のことによる．まず，$\hat{x}, \hat{y}$ が増加関数であること（付録B参照）と $\hat{y}$ が左連続の階段関数となることおよび t'_k の定義を用いて，$k>0$ に対して $\hat{x}(t'_k-e_0)-\hat{y}(t'_k)\leq N$ と同値となる．次に同様に $\hat{x}$, $\hat{y}$ の性質と t_k の定義を用いることにより，求める条件と同値であることを導くことができる．以下では，これが成り立つ条件を考える．

$T_m<k\leq T_{m+1}$, $T'_j<k-N\leq T'_{j+1}$ であるような j, m を考える．このとき

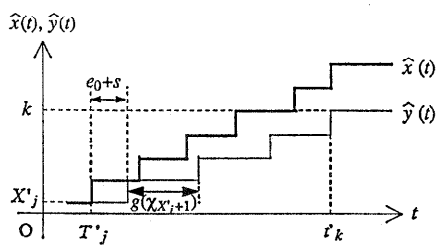

図 2 k と j の関係
Fig. 2 The relation beaween k and j.

$$t_k-(t'_{k-N}-e_0)=T_m-T'_j+\sum_{i=X_m+2}^{k} h(i)-\sum_{i=X'_j+1}^{k-N-1} g(\chi_i)-s$$

が0以上となることがバッファが溢れないための必要十分条件である．

これをこのまま扱うのは複雑なので，計算機が速い場合と遅い場合について分けて考える．まず計算機が速い場合として，$\max h(i)<\min g(\chi_i)$ の場合を考えることにより，計算機が *wait* している時にしかバッファが空にならない場合に話を限定する．この時，すべての j に対して，ある i が存在して，$T'_j=T_i$ が成り立つ．そこで，$T'_j=T_i$ とする．この時，$X'_j=X_i$ である．N 個より多くのデータを送る場合には計算機が *wait* を行うものとし，

$$N\cdot\min f(\chi_i)\geq c \qquad (1)$$

という条件をおく．すると，$k-N<X_m$ が得られ，

$$t_k-(t'_{k-N}-e_0)=\sum_{j=l}^{m-1}\max\left\{\sum_{i=X_j+1}^{X_{j+1}} f(\chi_i),\ \sum_{i=X_j+2}^{X_{j+1}+1} h(i)\right\}+\sum_{i=X_m+2}^{k} h(i)-\sum_{i=X_l+1}^{k-N-1} g(\chi_i)-s$$

となる．これを簡単化することにより，条件(1)に加えて

$$\max\left\{\sum_{i=X_j+1}^{X_{j+1}} f(\chi_i),\ \sum_{i=X_j+2}^{X_{j+1}+1} h(i)\right\}\geq\sum_{i=X_j+1}^{X_{j+1}} g(\chi_i) \qquad (2)$$

という条件が成り立てばよいことがわかる．これはグラフィックに要すると思って計算機が待つ時間と実際に計算にかかる時間との大きい方は，グラフィックに実際に要する時間以上であるということを表す．特に

$$f(\chi_i)\geq g(\chi_i) \qquad (3)$$

であれば十分である．

以上をまとめると，計算機が速い場合には(1)，(3)の二つがバッファの溢れないための十分条件である．これは通常の設計に当たり，一度に N 個までしかデータを送らず，ディスプレイにかかる時間以上に待つというものである．

一方，計算機が遅い場合として，ここでは，

- $g(\chi_i)\geq c$ ならば $i=1$ かつ $f(\chi_1)\geq g(\chi_1)$
- $g(\chi_i)<c$ ならば $g(\chi_i)\leq h(i)$

の場合を考える．これは，画面の消去は最初にしか行わず，その他に対しては，グラフィックに要する時間より計算時間のほうが大きいということを表している．このとき，

$$N\cdot\min h(i)\geq\max\{g(\chi_i)\,|\,g(\chi_i)<c\} \qquad (4)$$

であれば，バッファが溢れない．これは，バッファ中のデータに対する $g(\chi_i)$ の合計が，二つ目のデータを送った時刻以後においては，c を超えない $g(\chi_i)$ のうちの最大値以上にはならないことから示される．実際には有限個のデータしか送らなかったことも影響していると思われるが，f をほとんど0にしても動作したのはこのような状況であったと考えられる．計算機の方が遅いというのは一見奇妙であるが，これは TSS やチャネルの待ち時間をも含んだものを計算時間と呼んでいることによる．OS のスケジューリングやバッファリングの方法によっては，計算機が無駄に待たされたりすることにより計算時間が大きくなることがある．

5. 結　　論

実時間を扱う並行プログラムの ν-転換を用いた解析について述べ，典型的な例に対して性質の解析を行って動作条件を示した．ただし，実際にはデータごとにその大きさが異なるのであるが，その部分を簡単化して，すべて同じ大きさであるものとして取り扱った．また，拍車 $a(t)$ と $b(t)$ の成り立つ時刻の間隔を一定で同じとしたが，これも簡単化のためであり，h, g の定義から見て明らかなように，この制限は本質的な問題ではない．例えばこれらを異なるものとするとき，結論として，(3)式で $b(t)$ の間隔分を左辺に加えるのみでよい．時刻を直接取り扱うことができたため，性質の解析は自然であった．また，ここで示した条件を満たすならバッファは溢れないということは，拍車に関していくつかの条件をおけば，行動述語から直接導ける．すなわち FA 上で形式的に証明することも可能であるが，ここでは読みやすさのために証明の筋道を示すに留めた．他の問題に対しても同様に性質の解析を行うことができる．しかし，このままでは証明が非常に長く繁雑なものとなるので，機械的な支援および補助的な定理などが今後必要となろう．

謝辞　本研究を進めるに当たり，討論に参加し，有益な助言を下さった筑波大学電子・情報工学系細野千春講師，同水谷哲也助手，埼玉短期大学情報処理学科池田靖雄講師に深く感謝いたします．

参 考 文 献

1) Chaochen, Z., Hoare, C. A. R. and Ravn, P.: A Calculus of Durations, *Inf. Process. Lett.*, Vol. 40, pp. 269-276 (1991).
2) Cousot, P.: Methods and Logics for Proving Programs, van Leeuwen, J. (ed.), *Handbook of Theoretical Computer Science*, Elsevier Science Publishers B. V., pp. 842-993 (1990).
3) Gao, T., Hosono, C. and Yamanaka, K.: An Analytic Semantics of CSP, *Fundamenta Informaticae*, Vol. 15, No. 2, pp. 107-122 (1991).
4) Hoare, C. A. R.: An Axiomatic Basis for Computer Programming, *Comm. ACM*, Vol. 12, No. 10, pp. 576-580, 583 (1969).
5) Hoare, C. A. R.: Communicating Sequential Processes, *Comm. ACM*, Vol. 21, No. 8, pp. 666-677 (1978).
6) Igarashi, S.: A Natural Deduction System for Assertions, Nivat, M. (ed.), *Théorie des algorithmes des langages et de la programmation*, Sèminaires IRIA, pp. 39-45 (1974).
7) Igarashi, S.: The ν-conversion and an Analytic Semantics, Mason, R. E. A. (ed.), *Inf. Proc.*, IFIP, Elsevier Science Publishers B.V. (North-Holland), pp. 769-774 (1983).
8) 五十嵐滋，辻　尚史，細野千春，小暮博道，水谷哲也，岸田克己，笹川瑠美：ν-conversion の基礎的概念とその解釈，第 27 回情報処理学会全国大会論文集，pp. 23-24 (1983).
9) Igarashi, S., Mizutani, T. and Tsuji, T.: An Analytical Semantics of Parallel Program Processes Represented by ν-Conversion, *TENSOR N.S.*, Vol. 45, pp. 222-228 (1987).
10) Igarashi, S., Mizutani, T. and Tsuji, T.: Specifications of Parallel Program Processes in Analytical Semantics, *TENSOR, N.S.*, Vol. 45, pp. 240-255 (1987).
11) 池田晴雄：高階論理型プログラム言語 NU の解釈系，筑波大学工学研究科博士論文 (1993).
12) Kröger, F.: *Temporal Logic of Programs*, Springer-Verlag (1987).
13) Lamport, L.: What Good Is Temporal Logic?, Mason, R. E. A. (ed.), *Inf. Proc.*, IFIP, Elsevier Science Publishers B.V. (North-Holland), pp. 657-668 (1983).
14) Owicki, S. and Gries, D.: Verifying Properties of Parallel Programs: An Axiomatic Approach, *Comm. ACM*, Vol. 19, No. 5, pp. 279-285 (1976).
15) Owicki, S. and Gries, D.: An Axiomatic Proof Technique for Parallel Programs I, *Acta Inf.*, Vol. 6, pp. 319-340 (1976).
16) 水谷哲也，細野千春，五十嵐滋：ν-定義可能行為によるプログラムの検証，コンピュータソフトウェア，Vol. 2, No. 3, pp. 47-56 (1985).
17) Mizutani, T., Igarashi, S. and Tsuji, T.: An Analytical Equivalence Theory of Computer Programs, Díez, A., Echeverría, J. and Ibarra, A. (eds.), *Structures in Mathematical Theories*, Reports of the San Sebastian International

Symposium, pp. 199-204 (1990).
18) Pnueli, A. and Harel, E.: Applications of Temporal Logic to the Specification of Real Time Systems, Joseph, M. (ed.), *Proc. Symp. Fromal Techn. in Real-Time and Fault-Tolerant Systems*, Lecture Notes in Computer Science 331, pp. 84-98 (1988).
19) Shoenfield, J. R.: *Mathematical Logic*, Addison-Wesley (1967).
20) Takeuti, G.: *Two Applications of Logic to Mathematics*, Iwanami Shoten, Publishers and Princeton University Press (1978).
21) 富田康治, 辻 尚史, 五十嵐滋: プログラムにおける実時間問題, 日本ソフトウェア科学会第7回大会論文集, pp. 153-156 (1990).
22) 辻 尚史: ν 定義可能行為とプログラムにおける実時間問題, 応用数学合同研究集会報告集, pp. 103-114 (1983).

付 録 A

プログラム2に対応する行為 R として，様々なものが考えられる[7),16)]．例えば l というステージ変数を用いて，次のように機械的に行為へ翻訳することも可能である．ステージ変数は，逐次制御を行うために導入された新しい変数である．ただし，$\dot{l}(0)=0$，すなわち変数 l の初期値は0であるものとする．

$$l=0 \ \& \ \nu y=0 \ \& \ \nu l=1$$
$$\vee l=1 \ \& \ x>y \ \& \ \nu l=2$$
$$\vee l=2 \ \& \ \nu y=y+1 \ \& \ \nu l=3$$
$$\vee l=3 \ \& \ \nu v=t+G(w[y]) \ \& \ \nu l=4$$
$$\vee l=4 \ \& \ v<t \ \& \ \nu l=1$$

ここで，G という関数が含まれているが，これは，実体としては本文で用いた g と同じものであるが，区別のためここでは G と表すこととした．つまり，*display* を，その実行に要する時間だけ待つものとして表した．

このような形の行為は扱いが繁雑になるので，ここでは簡単のため，

$$x>y \ \& \ v<t \ \& \ \nu y=y+1 \ \& \ \nu v=t+G(w[y])$$

という行為を考えることにする．ただし，$\hat{y}(0)=0$，$\hat{v}(0)=0$ とする．

付 録 B

行為 $a(t)\uparrow W \parallel b(t)\uparrow R$ を解釈した軌跡において $\hat{y}$ が単調増加の関数となることを示すのだが，ここでは簡単のため R のみを単独に考える．これは直観的には W が y の値を変化させないことにより正当化されるが，厳密には同様の証明が必要である．

R' を

$$(b(t) \ \& \ \exists \boldsymbol{y} R[\boldsymbol{x}, \boldsymbol{y}, t] \supset R[\boldsymbol{x}, \nu \boldsymbol{x}, t])^{\#}$$

とする．このとき，$\#$ の定義より，

$$R' \equiv b(t) \ \& \ x>y \ \& \ v<t \ \&$$
$$\nu x=x \ \& \ \nu y=y+1 \ \& \ \nu v=t+G(w[y])$$

となる．$Pa(R', \hat{\boldsymbol{x}})$ なる任意の $\hat{\boldsymbol{x}}$ をとると，Pa の定義から，R' が時刻 t で行動可能であれば，すなわち，

$$\exists \boldsymbol{y} R'[\hat{\boldsymbol{x}}(t), \boldsymbol{y}, t]$$

であれば

$$R'[\hat{\boldsymbol{x}}(t), \hat{\boldsymbol{x}}(t+0), t]$$

となる．よって，$\hat{y}(t+0)=\hat{y}(t)+1$ である．ただし，

$$\forall \varepsilon>0 \exists \delta(0<\delta<\varepsilon \& A[t+\delta])$$

の略記として $A[t+0]$ を用いる．

また，時刻 t から時刻 $t+\varepsilon$ まで行動不可能であれば，すなわち $\forall \delta \geq 0(\delta<\varepsilon \supset \neg \exists \boldsymbol{y} R'[\hat{\boldsymbol{x}}(t+\delta), y, t+\delta])$ であれば $\hat{\boldsymbol{x}}(t)=\hat{\boldsymbol{x}}(t+\varepsilon)$ なので明らかに $\hat{y}(t)=\hat{y}(t+\varepsilon)$ である．このことと，行動可能となる時刻が離散的となることとから，$\hat{y}$ は単調増加の関数であることがいえる．

また，このことは，直接 $a(t)\uparrow W \parallel b(t)\uparrow R$ から，$\mathbf{N}(t)$ & $\nu y=y+1$ という行為への軌跡準同型[17)]が存在することからより容易に示すこともできる．ここで $\mathbf{N}$ は自然数であることを表す述語とする．

$\hat{x}$ の単調性についても同様に考えることができる．

(平成3年4月 1 日受付)
(平成5年2月12日採録)

富田 康治（正会員）
1965 年生．1988 年筑波大学第三学群情報学類卒業．1990 年同大学院修士課程理工学研究科修了．同年機械技術研究所入所．現在，物理情報部知識機械工学課にてマシンインテリジェンスの研究に従事．日本ソフトウェア科学会，計測自動制御学会各会員．

辻 尚史（正会員）
1945 年生．東京大学大学院．工学博士．1977 年筑波大学助教授（電子・情報工学系）．プログラム基礎論．「FORTRAN の実際」（共著，サイエンス社）．日本ソフトウェア科学会会員．

五十嵐　滋（正会員）

1937 年生．東京大学大学院．工学博士．1976 年筑波大学教授（電子・情報工学系）．プログラム理論，人工知能論，計算機音楽．Z．マンナ著「プログラムの理論」（訳書，日本コンピュータ協会）．IFIP WG 2.2 正員．日本数学会，日本ソフトウェア科学会会員．1992 年度マン・オブ・ザ・イヤー（米国伝記協会）．