北海道大学工学部研究報告
第106号（昭和56年）

Bulletin of the Faculty of Engineering,
Hokkaido University, No.106 (1981)

点追従制御方式をとる個別輸送システム CVSの運行シミュレータ

栗原正仁*　　中田勝啓**　　加地郁夫*

（昭和56年6月30日受理）

CVS Network Simulator for Personal Rapid Transit systems under Point-follower Control

Masahito KURIHARA, Katsuhiro NAKADA and Ikuo KAJI

(Received June 30, 1981)

Abstract

We developed a simulator for CVS (Computer-controlled Vehicle System) which is classified as PRT (Personal Rapid Transit systems).

Our simulator deals with those models in which vehicles move on a guideway under point-follower control. Vehicles might enter and depart from stations, pass through intersections and join another vehicle flow.

Control programs written in a particular language developed for simple programming can be supplied to the simulator. The simulator is composed of three parts … translator, runtime routines and scanner. The translator tranforms control programs into internal codes. The runtime routines are used when vehicles execute the instructions. The scanner is the most important part. It selects each vehicle, fetches the instruction to be executed, and then decodes it. Thus the vehicle executes the instruction and changes its status. When the scanner has finished with this vehicle, it selects another vehicle. When all vehicles have been scanned, the simulation clock evolves. The scanner also exhibits vehicle flow on a graphic display.

The above principles successfully make the simulator simple, relible, flexible extensible.

1. まえがき

ＣＶＳシミュレータを設計・試作したので報告する。

ＣＶＳ(Computer-controlled Vehicle System)は面的な交通サービスを提供する典型的な都市形個別輸送システムである[1,2,3]。筆者らはＣＶＳの駅部および交差点の個々における輻輳現象を確率論的手法を用いて解析してきた[6~9]。この研究をネットワークの輻輳現象にまで進めるには，理論的な解析のほか計算機シミュレーションによる実験的な考察が必要となろう。

シミュレーションについては，例えば，離散系についてはＧＰＳＳ，微分方程式等で記述される

* 電気工学科　系統工学講座
** 玉川大学工学部

連続系モデルについてはＤＤＳIIIなどの標準モデルが利用可能である。しかし，本ＣＶＳネットワークモデルは，汎用言語によってはもちろん，これらのシミュレーション言語によっても記述がむずかしい。又，都市形軌道交通システムのシミュレータは各メーカーが所内用に作成しているものがある[4,5]が，これらは鉄道や地下鉄のようにダイヤに基づいて運行される，比較的簡単な形状のネットワークを詳細に模擬することを指向しているようである。

本報告で述べるものは，デマンド運転方式の個別輸送システムを指向し，面的な形状のネットワークや車両相互の同期，種々の経路選択アルゴリズムを考慮する連続系シミュレータである。

2．シミュレータの概要

2．1　ＣＶＳ及びモデルの概要

ＣＶＳ[1,2,3]は典型的な都市形個別軌道システムである。同一車両には個人か又は家族などの同一パーティしか乗車せず，需要の発生に応じて空車を配するデマンド運転方式を採用している。車両は網状にはりめぐらされた専用軌道上を完全なコンピュータコントロールによって運転され，都市の面輸送容量を倍加する。

このように小形車両のデマンド運転方式による面的な交通サービスを提供する場合，制御方式として２方式が提案されている。一つは車両追従制御方式である。もう一つは点追従制御方式(Point-follower control)と呼ばれ，計算機内で仮想のセルを軌道上に走行させ，デマンドが発生したときに空いているセルを割り当てるものである。合流点で一緒になるセルには同一番号を付し，番号が同じセルは一つと見なして１車両だけを割り当てるようにして合流点での競合を回避する。

本報告で取り扱うモデルは後者の方式で，各車両は割り当てられたセルに正確に追従すると仮

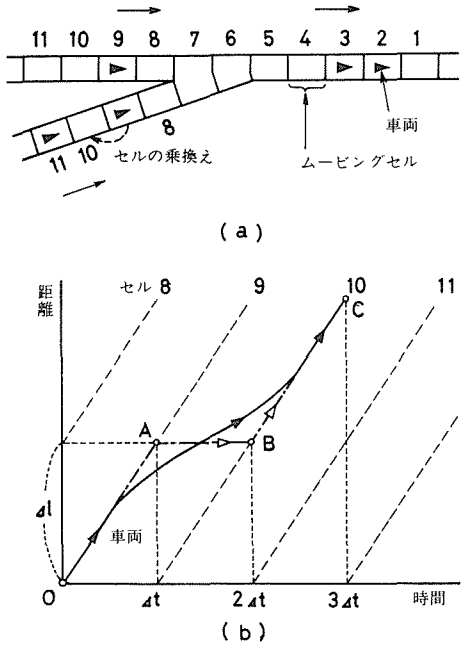

図１　セルの乗換え

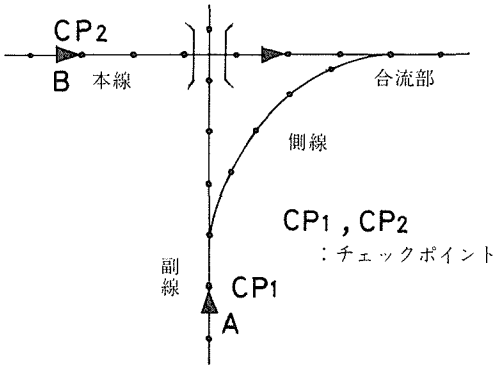

図２　交差点

定する。しかし，この方法ではネットワークが大規模になると，ルート上のすべての合流点で競合のない空いたセルを見つけることが困難になるので，ここでは合流点の手前などで必要に応じて１つ後のセルへの乗換えを許す方式のシミュレーションを考える。

図１はセルの乗換えの状況を示したものである。合流部での衝突を避けるためセル９を割り当てられていた車両がセル10へ乗換える場合，実際には同図(ｂ)の実線のように制御がなされる。しかし，本稿で述べるシミュレータは車両のダイナミクスを模擬するものではなく，車両群のマクロな運行状況をシミュレートするものなので，ここでは車両が折線ＯＡＢＣに沿って制御されると考える。従って，セルの乗換えは Δt 時間の待ち（停止）と等価であるといえる。

交差点の概略を図２に示す。交差点では直進と右折のみを許すものとする。本線と副線の交差部分は立体交差とし，副線上の右折希望車は側線を使って本線に合流する。しかし，図のように右折希望車両Ａが副線上の CP_1 に到来したときに本線上の CP_2 にも車両Ｂが到来した場合には，そのままでは合流部で衝突するので車両Ａが CP_1 で待つ(セルを乗り換える)ことになる。この場合，車両Ａを無制限に待たせる方法のほか，待ちに制限を加える方法として，(１)待行列長を制限する，(２)セルの乗り換え回数を制限するなどが考えられ，これらの制限を越える場合には右折希望車両であっても右折させず強制的に直進させる方式が提案されている。[9]

駅部の概略を図３に示す。本線から分岐した側線上には１バースの乗降駅が設けられている。駅にいる出発準備の完了した車両は，合流部で本線上の走行車両と衝突しないとき，即ち，図のCPに入駅非希望車(本線走行車)がいないときに出駅する。一方，CPに到来した入駅希望車は駅に車両がいなければ入駅できるが，入駅できない場合は，直ちに本線側を走行し，迂回する。

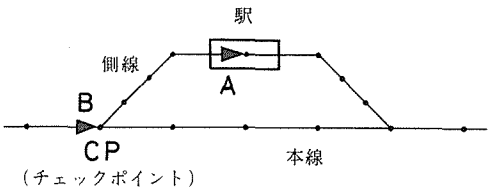

図３　駅　　部

２．２　目的および設計方針

本シミュレータの目的は，（１）ＣＶＳネットワークを実証し，（２）各種制御方式・運行アルゴリズムを評価することである。ＣＶＳは，実際にはまだ研究段階のシステムである。これが実現すればほぼ理想的な交通形態を生ずる反面，システムの効率的な運用の見地から，右折希望車を強制的に直進させたり，入駅希望車を入駅させず迂回させたりするため，すべての乗客が個々の目的地に到達できるのかという疑問も生ずる。第１の目的はこのような定性的な疑問に答える等，システムの概念計画段階で計画立案グループ内でのコンセンサスを得るためのものである。交差点および駅部では待ちに関連して幾つかの方式がある。又，出発地―目的地（OD）間の経路選択アルゴリズムやデマンドのパターンに応じた空車割当てアルゴリズムも種々考えられる。第２の目的は，このような各方式・アルゴリズムを本シミュレータに組み込み，定量的に評価することである。

当面は目的（１）を主眼にするが目的（２）のための拡張も考慮し，次のような方針をたてた。

（ａ）車両挙動の表示

（ｂ）拡張性，柔軟性があり容易にプログラムできるシミュレーション言語

（ｃ）汎用言語との結合

（ａ）は，車両の動きをグラフィックディスプレイに表示しようというものである。これは，

目的（1）のために必要な事項である。これはまた，シミュレータ自身のテストに有効であるばかりでなく，ユーザーにとってもシミュレーション言語で書いた自分のプログラムのテストやデバックに有効であろう。(b)は，現実には建設されておらず模索段階のシミュレーションということから当然である。現時点でもさまざまな方式，アルゴリズムがあるばかりでなく将来にも新しいアイデアが提案される可能性があるので容易にこれらに対拠できなければならない。そこで本シミュレータでは本目的のための簡易シミュレーション言語を提案し，この言語によってユーザーが記述した1台の車両に対する制御プログラムを解釈実行することにより複数台の車両が運行上の制約を満たしながらネットワークを走行するという構造をとることにした。しかも，この簡易シミュレーション言語は固定されたものではなく，ユーザーが容易にこれらの全部または一部を変更したり，新たな機能を付け加えたりできる。(c)は各種のアルゴリズムはシミュレーション言語ではなく汎用言語の方が記述しやすいと考えたためである。この目的のために汎用言語とのリンク命令を用意した。以上の方針をもとにパーソナルコンピュータで試作した。

2.3 車両の位置

本シミュレータでは車両の位置を論理位置，表示位置の2つに区別して表現している。

（1）論理位置

論理位置を（r，p）と表わし，r（＝1，2，…）をルート番号，p（＝1，2，…）をルート内位置と呼ぶ。今考えているモデルにおける論理位置の集合をLとすると，ルート番号の等しい点列(r，1)，(r，2)，…，(r，l_r)がLに属し，(r，l_r＋1)がLに属さないならば，(r，l_r＋2)，(r，l_r＋3)，…∉Lでなければならない。このとき，点列(r，1)，…，(r，l_r)をルート（路線）rと呼ぶ。r，l_r，(r，1)，(r，l_r)を各々，ルート番号，ルートrの長さ，ルートrの始点，ルートrの終点という。

次に車両挙動の制約として，「どの2つの車両も異なる論理位置にいる」ことを要求する。即ち，Lの上を走行する車両iの論理位置を(R_i，P_i)とすると，$i \neq j$ならば(R_i，P_i)≠(R_j，P_j)でなければならない。

（2）表示位置

表示位置は個々の車両の表示にのみ必要な情報である。表示位置は（x，y，d，c）の形で表わし，車両iの表示位置を(X_i，Y_i，D_i，C_i)とする。x，yはグラフィックディスプレイ上の座標であり，非負整数値をとる。dは車両の向きで，図4のように0から7までの8方向を許し，右折を正方向とする。Cは車両の色を整数値で示す。車両の色はその車両の属性を視覚的に示すのに有効である。例えば，実車か空車かの別，駅で乗降中か出発準備完了かの別を示したり，注目したい特定の車両へのマークとしてなど目的に応じた使用法が種々考えられる。

表示位置については，「どの2つの車両も異なる表示位置にいる」という制約は設けない。

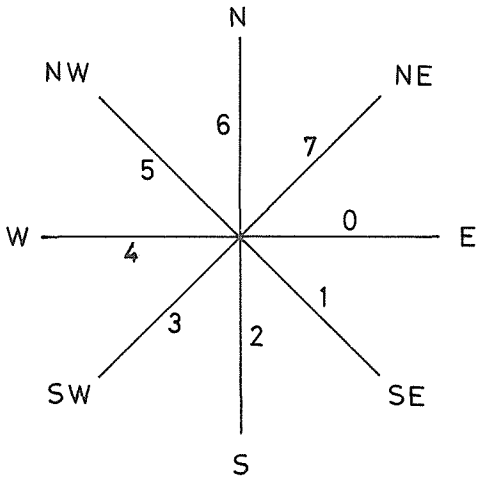

図4 車両の向き

２．４　車両の移動

前述の二種の位置に対応し，車両の移動も論理移動と表示移動に区分される。

（１）論理移動

車両の論理位置（R_i，P_i）を更新することを車両の論理移動という。これにより，時刻ｔ（ｔは整数）に位置（ｒ，ｐ）にいる車両の，時刻ｔ＋１での論理位置は次の４つに区分される。

（ａ）（ｒ，ｐ）：停止（駅での停車，又はセル乗換えに対応する待ち）

（ｂ）（ｒ，ｐ＋１）：同一ルート上を進行

（ｃ）（ｒ’，１）：他ルート始点への分岐（（ｒ，ｐ）を分岐点と呼ぶ）

（ｄ）（ｒ’，ｐ’）：他ルートへの合流（（ｒ’，ｐ’）を合流点と呼ぶ）

但し，（ｃ）と（ｄ）ではｒ’＝ｒであってもよい。又，２つ以上の項目に該当する場合は，より上に示した項目に分類する。

この４区分はそれぞれに対するシミュレータの処理方式の観点から定めたものである。後述のように，同一ルート上の車両は線形リストでつながれるため，（ａ），（ｂ），（ｃ），（ｄ）の順に処理が容易となっている。従って，論理的には（ｄ）のみですべての場合を網羅するけれども，より能率的な処理を可能とするため上記区分を用いる。一方，例えば環状ルートｒの終点（ｒ，l_r）から始点（ｒ，１）への移動は同一ルート上の運行にもかかわらず，上記区分では（ｃ）となる。これは，ルートｒの先頭車両だったものが同ルートの最後尾車両となりリストの変更が必要なため，処理上の立場から（ｃ）とされるものである。

さて，論理移動の際には前述の条件「$i \neq j$ならば $(R_i, P_i) \neq (R_j, P_j)$」を満たすように，必ず進行位置に車両がいないことをチェックしてから行なわなければならないが，進行位置に車両がいる場合について述べる。この場合は次の２つのいずれかの処理になる。

進行位置にいる車が現在時刻の処理（スキャンと呼ぶ）を終了しているときは当車両は進行できず，現在位置で停止し，待ちとなる。当車両の当時刻のスキャンは終了したこととする。

進行位置にいる車が現在時刻のスキャンを終了していないときは，その先行車が当時刻のスキャン内に移動する可能性がある。従って，当車両のスキャンを保留する。その後，進行位置にいる車がスキャンを完了したときに再度当車両をスキャンすることになる。

図５は上記２つの場合の例を示したものである。（ａ）の場合は，車両ａが交差点で右折できないため車両ａ，ｂの待ち行列が生じておりｃは停止する。（ｂ）は車両ａ，ｂが未スキャンのため車両ｃはスキャンを保留する。その後，ａ，ｂがスキャンされて移動した後に再びｃがスキャンされる。その結果ｃは停止することなく現在時刻のスキャンで合流する。

最後に，論理移動に関する重要な規定を述べる。それは，「論理移動は各車について各時刻に１回だけ行なう」ということである。２回以上は行なってならず，又０回も許されない。

（２）表示移動

車両の表示位置（X_i，Y_i，D_i，C_i）を更新することを表示移動という。表示移動には次のようなものが考えられる。

（ａ）x，y，d，c，の全部または一部を直接与えるもの。

（ｂ）現在位置（x，y，d，c）からの変位Δx，Δy，Δd，Δcの全部または一部を与えるもの。

又，あらかじめ移動ユニットΔx，Δy，Δdを定めておき，

（ｃ）ｘ方向へ１ユニット移動（他の７方向についても同様）。

（ｄ）現在の方向（D_i方向）へ１ユニット移動。

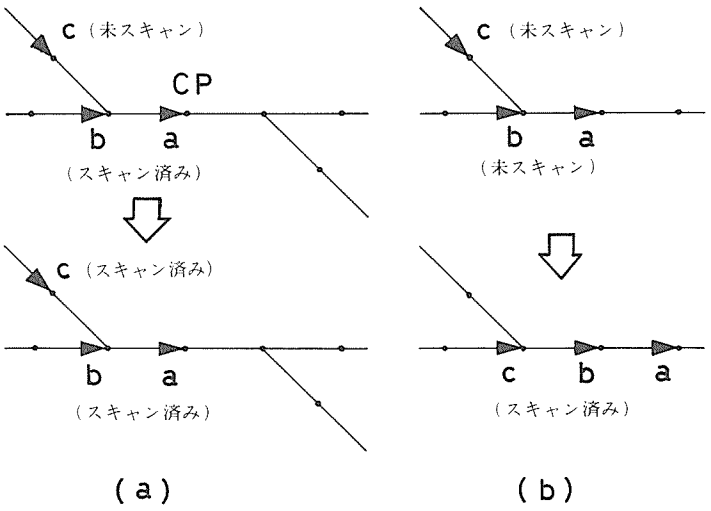

図5 運行位置に車両がいる場合

（e）右（左）へ1ユニット（45°）向きを変える。

などがある。

表示移動は論理移動とは異なり，同一時刻に何回でも行なうことができ，その時刻のスキャンが終了した時点での（X_i，Y_i，D_i，C_i）により車両が表示される。

2．5 車両の運行

車両の運行方法は所定の言語による運行プログラムで記述する。この運行プログラムの各ステートメントを命令と呼ぶ。各車両は各々アドレスカウンタを持っており，この命令列を独自に並列的に実行することによりシミュレーションが進む。本節では運行プログラムを構成する命令系を概観する。命令は以下の5種に分類される。

（1）論理移動命令

これには，2．4（1）で述べた停車，前進，分岐，合流のほか，車両の発生・消滅に関するものがある。

（2）表示移動命令

これには，2．4（2）で述べた5種類に対応するものがある。

（3）判定・判断命令

これは，交差点での分岐・合流に関するチェックやルート選択，駅での出発可否判定やルート選択に関するものである。このうちルートの選択は，現在位置と目的駅および網内の輻輳状況等によって決定されるものであり，高度なアルゴリズムを要する。従って，これに関する命令は，汎用言語(FORTRAN)で書かれたユーザープログラムへのリンク命令とする。

（4）補助命令

これは上記3種以外の基本命令である。このうち最も重要なものとして区切命令がある。この命令は同一時刻で実行される命令列の区切を示すものである。その他には，単純ジャンプ命令，リンク命令，パラメータの設定や統計に関する命令等がある。

（5）複合命令

以上の（1）～（4）の基本命令系から，よく使用される命令列を複合し，1命令としたもので

ある。但し，複合命令はこれを構成する基本命令列にマクロ展開するのではなく，この命令列と等価な効果を１命令として実現するものである。

2．6　シミュレーターの構成

図６にシミュレータの構成を示す。実線で囲まれたブロックはプログラム，破線で囲まれたブロックはデータである。プログラム相互を結ぶ太線は制御が移行可能な経路であり，プログラムとデータを結ぶ矢印はデータの流れを表わす。シミュレータは一点鎖線で囲まれた部分である。

トランスレータはユーザーが記述したソース言語による運行プログラムを内部形式に変換する。命令実行ルーチンは各命令と１対１に対応して命令を実行する。グラフィックハンドラーは１台の車両がスキャンを終了する毎に，グラフィックディスプレイにその車両を表示する。

スキャナーはシミュレータの中枢といえる部分である。前節では各車両が個々に，従って全体としては全車両が並列的に命令を実行すると述べた。しかし，これらの車両群は完全に独立に命令列を実行するわけにはいかない。車両挙動は相互に依存性があり，又，時間の進展に同期をとりながら走行させなければならない。（ここでいう時間は実時間ではなく，論理移動命令の実行回数を尺度にとったシミュレーション上の時間を指す。）従って，このような並列的な処理内容を，これと矛盾しないように直列処理で置き換えなければならない。これを行なうのがスキャナーである。スキャナーは，個別車両情報，フラグ及びそれ自身の内部に持つ情報とからスキャンすべき車両を１台決定する。次に，その車両の持つ個別情報の１つであるアドレスカウンタを参照し，運行プログラムから命令を１つ読み取る。そして，この命令を解読して適切なルーチンを当車両に結びつける。このとき，並列処理と矛盾のないようにするためにスキャンを保留する車両が生じた場合は，これらをリストで結び，後にスキャンを行なうようにする。こうして最終的にすべての車両のスキャンを終了したらクロックを１だけ進めて再度同様の処理を続ける。

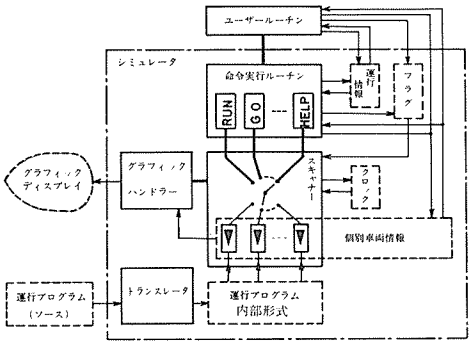

図６　シミュレータの構成

３．シュミレータの詳細

３．１　個別車両情報と運行情報

各車両は車両番号ｉ（＝１，２，…）で識別され，次のような属性（パラメータ）を持っている。

（ａ）アドレスカウンタ　A_i：次に実行する命令のアドレス

（ｂ）論理位置　R_i，P_i

（ｃ）表示位置　X_i，Y_i，D_i，C_i

（ｄ）目的駅番号　S_i

（ｅ）分岐・合流要求フラグ　B_i：次の交差点で他ルートへ分岐したい車両はB_iを１にセットする。又，駅で出発準備完了している車が出発（本線への合流）を要求するときにもこのフラグをセットする。以下では交差点の例を述べる。交差点での典型的な命令列は判断命令・判定命令・分岐命令である。まず車両は判断命令を実行し，経路選択アルゴリズムによってB_iの値が決め

られる。判定命令では，$B_i=1$の車両に対して分岐可能か否か（合流部で衝突しないか）の判定を下す。次の分岐命令では$B_i=1$の車両は指定されたアドレスへジャンプする。$B_i=0$の車両は直後の命令へ進む。従って，判定命令における$B_i=1$の車両の挙動は判定結果により次の3通りが考えられる。

（i）許可：要求が認められ，$B_i=1$のまま次の命令へ進み，分岐する。その後B_iはリセットされる。

（ii）待ち：当時刻では許可できないので，その場で待つ。次の時刻のスキャンではこの判定命令を再度実行する。B_iはセットされたままである。

（iii）拒否：分岐は認められず，B_iをリセットされて強制的に直進させられる。次の分岐命令では分岐しないことになる。

(f) カウンタ　N_i：命令によってはその実行ルーチンがN_iを作業用に用いることがある。

(g) 後続車番号　L_i：車両iと同一ルート上の後続車の番号。もし車両iが最後尾なら$L_i=0$とする。L_iは後に述べる車両リストのリンク部である。

（h）ユーザーパラメータ　$U_i(1)$，$U_i(2)$，…：ユーザーが，上記以外の車両属性や統計のため自由に使えるパラメータ。

次にL_iを用いて構成される車両リストについて述べる。図7はその概念図である。図で黒い三角形で示された車両は実際に表示されている車両である。同じルートr上を走行する車両は先頭車から順に線形リストでつながれており，その先頭車両はTOP〔r〕，末尾車両はBOT〔r〕で示される。RMAXはモデルで与えられたルートの数である。CV，PV，SVはそれぞれ現在スキャン中の車両，その先行車および後続車を示す。

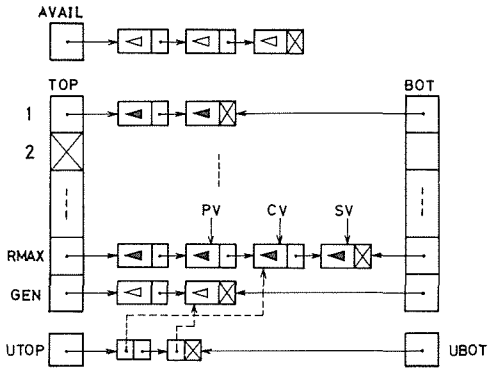

図7　車両リスト

白い三角形は表示されない車両である。AVAILは，ネットワーク上を走行していない車庫内にいる空車のスタック（空車スタックと呼ぶ）のスタックポインタである。TOP〔GEN〕によって指されている車両群はそれぞれ個々の車両発生命令に対応している仮想車両であり，この車両が車両発生命令を実行することによって空車スタックの先頭車1台がネットワーク上に発生される。

以上の各車両パラメータと車両リストとを個別車両情報と呼ぶ。UTOPで指されているリストは現在時刻でのスキャン保留車の車両番号をつないだもので，スキャナーが内部に持つ情報である。

運行情報は，論理位置（r，p）とその位置にいる車両番号i（車両がいなければi＝0とする）を関係付けるものであり，簡単な表（運行表）で実現されるものである。

3.2　スキャナーのアルゴリズム

本来は並列に運行されている個々の車両を直列にスキャンし，並列処理と同等の効果を生じさせるのがスキャナーである。図8にそのアルゴリズムの概要を示す。スキャナーはスキャン中の車両のアドレスカウンタの指す命令を取り出し，解釈して対応する実行ルーチンへ制御をわたす。

実行ルーチンではアドレスカウンタや位置などの車両属性や運行情報を変更し，ＳＴＦをスキャナーに返す。ここでＳＴＦはステイタスフラグと呼ばれ，1，2，3のいずれかの値にセットされる。スキャナーはこの値により当車両のスキャンを続行，終了，保留する。実行ルーチンがＳＴＦにどの値を返すかは命令およびそのときの状態（運行状況）によって異なる。

スキャンの順序は，ルート1からＲＭＡＸそしてＧＥＮまでルート毎に先頭車からとする。ＧＥＮの最後尾までスキャンされたら，スキャン保留車群ＵＴＯＰを先頭からスキャンする。スキャンが終了した車はＵＴＯＰからはずされる。こうしてこのリストが空になるまで続ける。しかし，一般にはこのリストが必ず空になるという保証はないので，ユーザーはそのようなことがないように注意して運行プログラムを記述しなければならない。スキャナーはこのようなデッドロックを検出した場合は残っているスキャン保留車の1台ずつに停止命令を実行させ，これを回避するようにする。このようにして最悪の場合には保留車すべてに停止命令を実行させて次の時刻のスキャンへ進む。

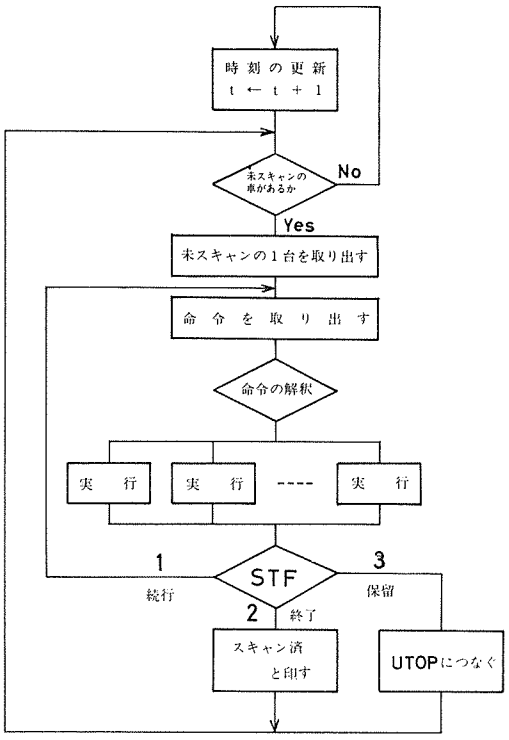

図8　スキャナーのアルゴリズム

3．3　命令系の詳細

命令系は内容により既述の5つに区分できるが，またその命令に対するスキャンの通過形態によりＰ形，Ｔ形，Ｇ形に区分できる。Ｐ（Pass）形はＳＴＦ＝1を返す命令で，車両は当時刻内に必ずこの命令を通過して次の命令へ進む。Ｔ(Terminate)形はＳＴＦ＝2を返す命令で，車両の

```
If (Ri, Pi+1) に車がいない then begin
   運行表を更新; Pi:=Pi+1 ; Ai:=Ai+1; STF:=1 end
else if 先行車両がスキャン済 then STF:=2 else STF:=3.
        (a) Run (1バイト命令)
If Bi=1 then begin
  if Ni=0 then Ni:=n;
  if 同時刻に (r,p) に車がいる，即ち
    . (r, p) に未スキャン車がいる，又は
    . (r, p+1) にスキャン済の車がいる
    then if Ni=0 then begin Bi:=0; Ai:=Ai+4; STF:=1 end
                  else begin Ni:=Ni−1; STF:=2 end
    else begin Ni:=0; Ai:=Ai+4; STF:=1 end
  end else begin Ai:=Ai+4; STF:=1 end.
       (b)Check (4バイト命令)
```

図9　命令実行アルゴリズム

当時刻のスキャンはこの命令で必ず終了し，次の時刻ではその次の命令から実行する。G(Gate)形はある条件が満たされるまで先へ進めない機能を含む命令である。即ち，ＳＴＦ＝２を返し，かつ，アドレスカウンタを更新しない可能性を持つ。この場合，車は次の時刻で再びこの命令を実行する。

以下に主要な命令の機能を示し，幾つかについては図９にアルゴリズムを示す。機能説明で，ｒ，ｐは論理位置，ａは命令のアドレスで，各々ラベルで表現できる。

（１）論理移動命令

○ Run　同一ルート上を１だけ前進する（G形，図９）。

○ Route　ｒ　ルートｒの始点に移動する（G形）。

○ Join ｒ，ｐ　位置（ｒ，ｐ）に合流する（G形）。

○ Stop　１クロックだけ停止する（P形）。

○ Generate ｒ，ｎ　　確率ｎ％で車を１台発生し，ルートｒの始点に進める（ポアソン到来）（G形）。但し，次の場合は発生しない。

（ａ）空車スタックが空のとき。

（ｂ）ルートｒの始点にスキャン済の車がおり，進行できないとき。

○ Terminate　　車両を空車スタックに戻す（T形）。

（２）表示移動命令

○ Origin ｘ，ｙ，ｄ　　X，Y座標および方向をｘ，ｙ，ｄとする（P形）。

○ Color　ｃ　　車両の色をｃにする（P形）。

○ Move ｄ　　ｄ方向（省略の場合は現在の方向）へ１ユニット進める（P形）。

○ Turn ｄ　　向きをｄにする（P形）。

○ Right　　向きを右へ１ユニット（45°）変更（P形）。

（３）判定・判断命令

○ Decide　　分岐要求フラグ B_i の候を決定するユーザールーチンへ飛ぶ（P形）。

○ Check　ｒ，ｐ，ｎ　　$B_i=0$ の車両に対してはそのまま通過させる。$B_i=1$ のときは同時刻に（ｒ，ｐ）に車がいなければ通過させる（おそらく次の命令で分岐する）が，車がいれば待ちとし，次の時刻で本命令を再度実行する。ｎは待ち回数の上限で，これを越えると $B_i=0$ とされて次の命令へ進む。ｎを省略するとｎ＝∞とする。（G形，図９）

○ Branch　ａ　　分岐フラグ $B_i=1$ ならａへジャンプし，そうでないときは直後の命令へ進む（P形）。

○ Enter　ｒ，ａ　$B_i=0$ の車両はそのまま通過させる。$B_i=1$ のときは，ルートｒ上にある駅ｒに入駅可能か否か判定し，可能なら駅ｒに入駅したことを印して，Route ｒの命令のあるアドレスａに分岐する。不可ならば $B_i=0$ にリセットされて次の命令へ進む（P形）。

○ Depart ｒ，ｐ　　出駅可否判定をする。同時刻に（ｒ，ｐ）に車がいなければ本線に安全に合流できる。従って駅ｒから出駅した（他車が入駅可能となる）ことを印して次の命令へ進む。出発不可ならば本命令にとどまる（G形）。

（４）補助命令

○ Go　　現在時刻のスキャンを終了する区切命令（T形）。

○ Jump ａ　　アドレスａにジャンプする（P形）。

○ Help ｎ　　ユーザーが書いたサブルーチン HELP 内の文番号ｎへ飛ぶ。

（５）複合命令

○ Advance n, d　　d方向に同一ルート上をn回進む。これは，Run ; Move d ; Go ;をn回繰り返した命令列と等価である。

○ Curveright　　論理的には同一ルート上を1つ進む。表示は，X，Y座標が進行方向の右斜め45°の位置で，車両の向きは右に90°回転する。これは次の命令列と等価である。

Run ; Right ; Move ; Right ; Go ; 。Curveleftも同様に定義される。

○ Stop n　　nクロック停止する。Stop ; Go ;のn回繰り返しと等価である。

3．4　プログラム例

図10に交差点と駅を1つずつ含むネットワーク例を，図11にその運行プログラム例を示す。T形およびこれを含む複合命令の前にはピリオドを打ち，命令列の区切りを見やすくしている。

ここでラベルの意味について述べる。ラベルのついた特定の命令を実行しようとする車両の論理位置は一定にしておかねばならない。(これはトランスレータによりチェックされる。)このときラベルはアドレス値と同時にこの論理位置の値を持つと考えられる。例えば図11の例でJP3は論理位置(1，2)を表わす。ラベルがオペランドに用いられたときの意味は個々の命令により異なる。例えばこの例におけるJP1はJoin命令では論理位置，Jump命令ではアドレスを表わしている。

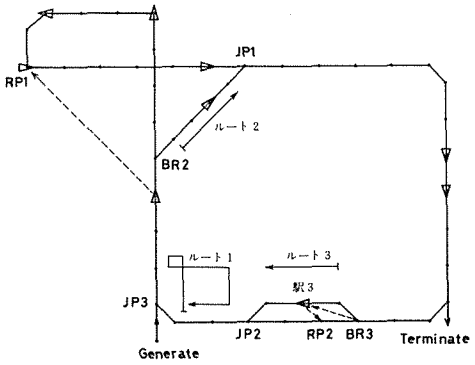

図10　簡単なネットワークの例

このように本シミュレータの運行プログラムはネットワークの図を見ながら容易に作成できる。

```
LABEL OP-CODE       OPERAND
      Generate      1,10
      Origin        100,100,N
      Color         1
      ·Go
      ·Advance      1
JP3   ·Advance      3,N
      Decide
      Check         RP1
      ·Advance      1
      Branch        BR2
      ·Advance      4
      Turn          W
      ·Advance      3
      ·Curveleft
      Run
      Move
      Turn          E
      ·Go
RP1   ·Advance      6
JP1   ·Advance      5
      ·Curveright
      ·Advance      6
      Decide
      Branch        TERM
      ·Curveright
      ·Advance      2
      Decide
      Enter         3,BR3
      ·Advance      1
```

```
LABEL OP-CODE       OPERAND
RP2   ·Advance      2
JP2   ·Advance      2
      Join          JP3
      Move          NW
      Turn          N
      ·Go
      Jump          JP3
BR2   Route         2
      Right
      Move
      ·Go
      ·Advance      3
      Join          JP1
      Move
      Right
      ·Go
      Jump          JP1
BR3   Route         3
      Move          NW
      ·Go
      ·Advance      1
      ·Stop         5
      Depart        RP2
      ·Advance      1
      Join          JP2
      Move          SW
      ·Go
      Jump          JP2
TERM  ·Terminate
```

図11　プログラム例

4. むすび

ＣＶＳ運行シミュレータについて報告した。本シミュレータの特徴は，車両挙動の表示，運行プログラム作成の容易性，システムの柔軟性にある。今後の拡張方向として次のようなものを考えている。

（a）統計データの収集と処理に関する命令を用意し，大型計算機で処理する。

（b）客の到来やデマンドを考慮したモデルを扱えるようにする。

（c）サブルーチン命令などを作り，大規模なネットワークを小さな運行プログラムで記述できるようにする。

参考文献

1） 荒屋：〝新交通システムと自動運転制御〟，信学誌，64，1，43－49（昭56－01）
2） 井口：〝新交通システム〟，電学誌，96，11，10－14（昭51－11）
3） 井口，石井ほか：〝わが国における個別輸送機関（ＣＶＳ）の開発計画について〟，機学誌，75,640.78－86（昭47－05）
4） 宮本ほか：〝軌道輸送システム用計画設計サポートシステム「ＴＲＡＮＳＰＬＡＮ」〟，日立評論，60,10（昭53－10）
5） 荒屋ほか：〝軌道交通システムシミュレータＡＣＴＳ〟，三菱電機技報，53.11（昭54－11）
6） 中田，加地，土肥：〝マルコフ再生理論によるＣＶＳ駅部モデルの車両挙動の解析〟，信学論（Ａ），Ｊ62－Ａ，10（昭54－10）
7） 中田，加地，土肥：〝マルコフ再生理論によるＣＶＳ駅部モデルの客オーバーフローの解析〟，信学論（Ａ），J63－Ａ，10（昭55－10）
8） 栗原，中田，加地：〝ＣＶＳ多バース駅モデルのマルコフ再生理論による解析〟，信学論（Ａ），J64－Ａ，9（昭56－09）
9） 浜松，中田，加地，土肥：〝個別軌道輸送システム十字路付近のモデル化とその解析〟，信学論（Ａ），Ｊ64－Ａ，10（昭56－10）