2015 年 2 月 4 日システムソフトウエア試験問題

学科　　学籍番号＿＿＿＿＿＿＿＿＿＿　　氏名＿＿＿＿＿＿＿＿＿＿

1．　つぎの文章の空白を埋めなさい．（各 1 点：合計 50 点）

(計算機アーキテクチャ：10 点)

- コンピュータは，CPU(中央処理装置)とメモリ，およびその他の入出力装置から構成されている．このうちの CPU は，命令の読み取りとデコードなどを行う(A)　　部と，実際の計算を行う(B)　　部の 2 つから構成されている．(A)部には，演算対象や，読み取った命令と変数などを格納する(C)　　と呼ばれる小規模で高速なメモリが配置されている．
- CPU は命令の(D)　　と(E)　　を繰り返す．一つ前の命令の(E)が終わってから次の命令の(D)をするのではなく，一つ前の命令の(E)と次の命令の(D)を同時に行うことによって高速化を実現する(F)　　処理が一般的に用いられている．
- CPU が次に(D)と(E)をする命令のアドレスは(G)　　と呼ばれる(C)に保持される．(H)　　コールでは(G)をスタックにプッシュし，(H)の終了時に(G)の値をスタックからポップして，呼び出しの直後の命令の(D)と(E)に復帰する．
- 計算結果に応じて(G)の値を書き換える，つまり，実行するプログラムのアドレスを変えるのが(I)　　命令である．この命令を読み取った場合，前述の(F)処理で命令の(D)と(E)をする場所が確定しなくなるため，CPU の計算速度が低下する．
- 短い時間間隔で CPU に割り当てるプログラムを切り替える(J)　　処理により，複数のプログラムが同時実行されているかのように見せることができる．

(オペレーティングシステム:20 点)

- 実行中のプログラムは，(A)　　と呼ばれ，OS からメモリやファイルなどの計算資源の割り当てを受ける．これはさらに(B)　　と呼ばれる単位に分けて実行される．
- 複数のプログラムが同時に実行されている(C)　　環境では，メモリや，補助記憶装置，その他の入出力装置 1 台に対して同時に複数のアクセス要求が来ることがある．このような場合に，アクセス要求を調停することも OS の重要な役目である．
- 補助記憶装置に対するアクセス要求の調停は，(D)　　の位置の調整問題になる．この問題の最も効率的なアクセスの調停方法は(E)　　アルゴリズムである．
- メモリの動的確保と解放は，メモリ領域の(F)　　化を招く．これを防ぐために，MMU というアドレス変換ハードウエアが用いられる．これは，(G)　　アドレスと論理アドレスを対応づける装置であり，(H)　　と呼ばれる固定サイズメモリを単位としてアドレス変換が行われる．
- 補助記憶装置の一部を利用して，物理メモリの大きさよりも大きな論理メモリ空間を実現する手法を(I)　　と呼ぶ. この実現方式としては, (A)が確保しているメモリ単位で, 補助記憶装置上のバッキング・ストア領域に退避・復帰させる(J)　　と，MMU で管理している(H)単位でバッキング・ストアと(G)メモリとの間で退避・復帰させる(K)　　の 2 種類がある．
- (K)のために，(G)メモリの空き容量がある程度減った時点でバッキング・ストアに不要なメモリを退避させておく処理が必要になる．これを行うのが，(L)　　デーモンである．このアルゴリズムには，最も古いものを退避させる(M)　　アルゴリズムや，最

も長期間使われていないものを退避させる LRU アルゴリズムがある．但し，(M)アルゴリズムでは，(G)メモリが大きい場合の方が退避・復帰動作の回数が増える(N)　　の異常と呼ばれる現象が発生するため， LRU アルゴリズムのほうがよく用いられる．

- コンピュータに接続される装置ごとに異なる操作コマンドをopen, close, read, writeなど統一的なインタフェースで扱えるようにするのが(O)　　の役割である．これは，Ｃ言語のプログラムで表現すると，(P)　　を要素とする構造体になる．この構造体が一つのデバイスに対応しており，Unix や Linux の場合は，全デバイスをこの構造体の(Q)　　として扱っている．
- (R)　　とは，補助記憶装置上の論理的なデータの１次元並びであり，実際にはセクタ上のデータブロックを繋げた構造になっている．このデータブロックのつなげ方には２種類があり，データブロックを順次繋げていく(S)　　構造と，データブロックを指すポインタの一次元配列を用いる(T)　　構造がある．

(プログラミング言語処理系：20 点)

- プログラミング言語処理系は，C, C++, Pascal, Basic,などの(A)　　言語で書かれたプログラムテキストを，(1)CPU が解釈実行可能な(B)　　語に変換する(C)　　や，(2)逐次解釈・実行するインタープリタ，などで用いられる．
- プログラムテキストは，(D)　　文法で表現可能な字句の集まりとして表現可能であり，プログラムテキストからこれを抽出する字句解析のためにオートマトンが用いられる．
- 字句解析の後には，構文解析が行われる．通常は(E)　　文法を用いてプログラミング言語を定義することができる．
- 構文解析では，(F)　　に対してどのような順序で(G)　　が適用され，入力として与えられた終端記号の列が生成されたのかを分析する．この分析結果は(H)　　木で表すことができる．
- (H)木は, 分岐のない中間節をまとめ, 演算子を分枝節に引き上げることで(I)　　木（演算子木とも呼ばれる）に変換することができる．
- 構文解析の方法は大きく分けて下降型の構文解析と，上昇型の構文解析法がある．下降型構文解析では(J)　　導出を前提とした文法の下での構文解析が行われ, 上昇型構文解析では(K)　　導出を前提とした構文解析が行われる．
- 下降型構文解析の例としては(L)　　構文解析，上昇型構文解析の例としては，(M)　　構文解析がある．(M)構文解析では，アクション表と GOTO 表をもとにして，スタックを用いて (N)　　と reduce の操作を行いながら構文解析が行われる. このうち reduce は, 生成規則の(O)　　辺に対応する部分を見つけて(P)　　辺に置き換えるという(Q)　　操作に対応している．
- 文法と文法にマッチした文字列に対する計算手続きを与えて，構文解析器を作るソフトウエアのことを(R)　　と呼ぶ．この例としては，Unix や Linux 上の Yacc やこれを拡張した Bison などがある．これらの（R）では入力の先読みをすることで構文解析の曖昧さを回避する(S)　　というアルゴリズムである．
- Yacc や Bison は単体で用いられることはほとんどなく，字句解析用オートマトンを生成する(T)　　や flex と併用して，(C)やインタープリタが作成される．

2．　以下の正規表現（A～D）に対応する記号の集合（すなわち言語）L(A)～L(D)があるとき，L(A)～L(D)の関係はどのようになっているか？Venn 図を書いて表しなさい．但し，全体集合は X で表し，空列 ε がどこに入るかも図中に示すこと．（10 点）

A:	a*b*
B:	b*a*
C:	aa*b*
D:	bb*a*

3．　下記の文法のもとで、「3-2-1」の構文解析を行った結果を、導出木、構文木の形で表わしなさい。（導出木 7 点＋構文木 3 点：合計 10 点）

<算術式> →<算術式><加減演算子><項>|<項>

<項>→ <項><乗除演算子><因子>|<因子>

<因子>→ <変数>|<数値>|(<算術式>)

<数値>→1|2|... |9

<変数>→A|B|C|...|Z

<加減演算子>→+|－

<乗除演算子>→×|÷

4.　Linux 上で，右図①，②，③のコマンドを 10 秒以下の間隔で順に入力した．

(ア) この 5 秒後に，kill 25461 というコマンドを実行すると，端末はどのようになるか？（3 点）

(イ) (ア)の代わりに kill 25460 というコマンドを実行すると，どのようになるか？（3 点）

```
$ ps –f                                                    ・・・①

UID        PID   PPID   C STIME TTY      TIME   CMD
twada     25441 25440   0 16:54   pts/2    00:00:00 -csh
twada     25457 25441   0 16:55   pts/2    00:00:00 ps –f

$ ( sleep 100 ; echo "Wake up" )&                          ・・・②

[1] 25460

$ ps –f                                                    ・・・③

UID        PID   PPID   C STIME TTY      TIME   CMD
twada     25441 25440   0 16:54   pts/2    00:00:00 -csh
twada     25460 25441   0 16:56   pts/2    00:00:00 -csh
twada     25461 25460   0 16:56   pts/2    00:00:00 sleep 100
twada     25462 25441   0 16:56   pts/2    00:00:00 ps –f

$
```

(ウ) （ア）（イ）の代わりに kill –KILL 25441 というコマンドを実行すると，端末はどうなるか？(但し-KILL は強制的なプロセス停止を意味する)（4 点）

5. 下記の文法から下に示すアイテム集合とそれらの間の遷移図を得た．集合 7,8 の空欄を全て埋め，状態遷移の矢印一本を追加し，図を完成させなさい．また，右のアクション表と GoTo 表を完成させ，shift-reduce conflict が起きる場所全てを〇で囲みなさい．(集合 8，7，矢印：各 3 点，表 9 点，〇2 点：合計 20 点)

(1) <E> →<E>+<B>

(2) <E> →<B>

(3) <B> →<B>*<T>

(4)<B>→(<E>)

(4) <B> →<T>

(5) <T> → 0

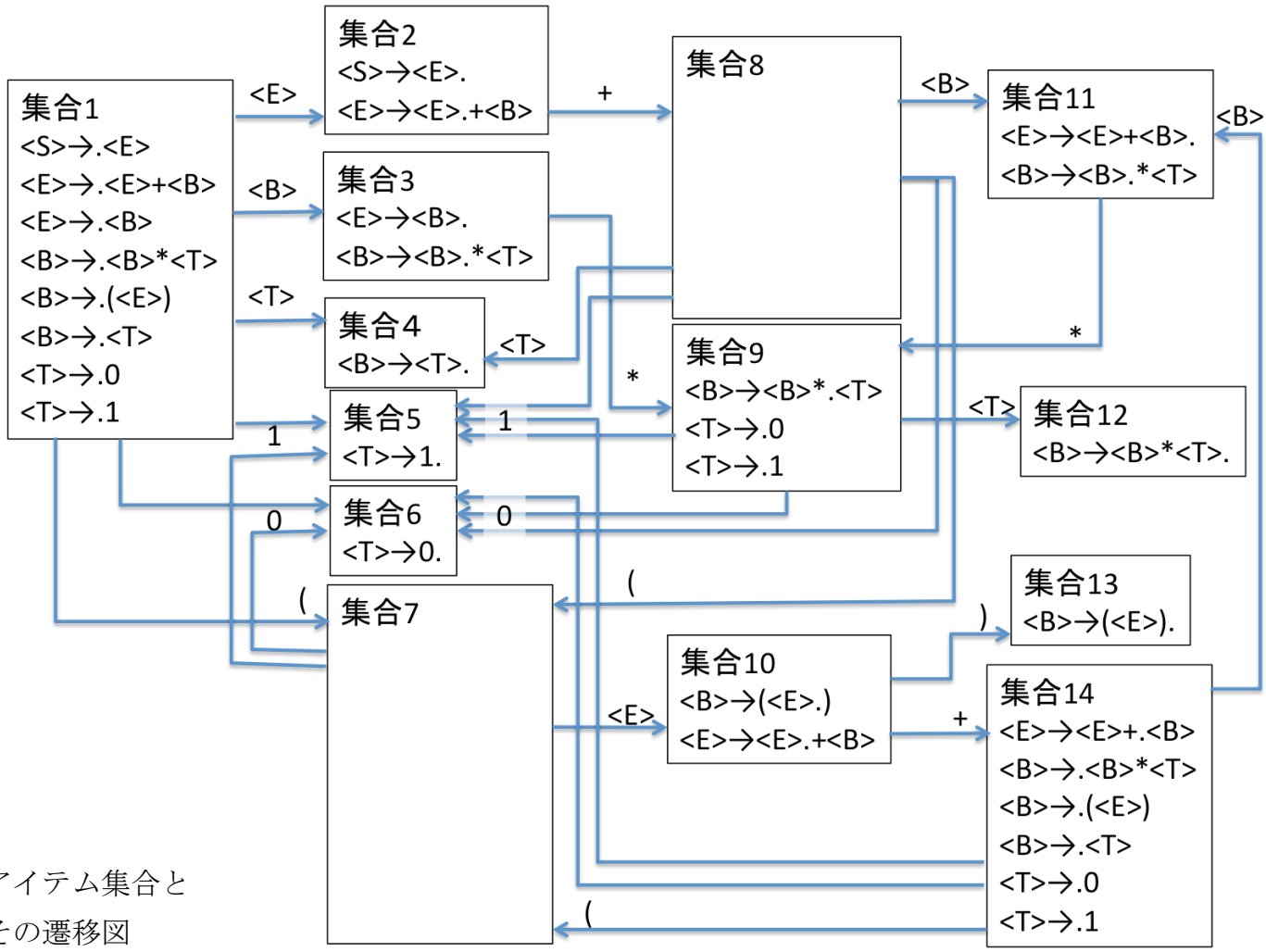
(6) <T> → 1

アクション表

	+	*	(	)	0	1	$
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							

Go To表

E	B	T



アイテム集合と
その遷移図

2015 年 2 月 4 日システムソフトウエア試験問題解答例

学科　　学籍番号＿＿＿＿＿＿＿＿＿＿　　氏名＿＿＿＿＿＿＿＿＿＿

1．　つぎの文章の空白を埋めなさい．（各 1 点：合計 50 点）

(計算機アーキテクチャ：10 点)

- コンピュータは，CPU(中央処理装置)とメモリ，およびその他の入出力装置から構成されている．このうちの CPU は，命令の読み取りとデコードなどを行う(A) 制御部と，実際の計算を行う(B) 演算部の 2 つから構成されている．(A)部には，演算対象や，読み取った命令と変数などを格納する(C) レジスタと呼ばれる小規模で高速なメモリが配置されている．
- CPU は命令の(D) 読み取りと(E) 実行を繰り返す．一つ前の命令の(E)が終わってから次の命令の(D)をするのではなく，一つ前の命令の(E)と次の命令の(D)を同時に行うことによって高速化を実現する(F) パイプライン処理が一般的に用いられている．
- CPU が次に(D)と(E)をする命令のアドレスは (G) プログラムカウンタと呼ばれる(C)に保持される．(H) サブルーチンコールでは(G)をスタックにプッシュし，(H)の終了時に(G)の値をスタックからポップして，呼び出しの直後の命令の(D)と(E)に復帰する．
- 計算結果に応じて(G)の値を書き換える，つまり，実行するプログラムのアドレスを変えるのが(I) 条件分岐命令である．この命令を読み取った場合，前述の(F)処理で命令の(D)と(E)をする場所が確定しなくなるため，CPU の計算速度が低下する．
- 短い時間間隔で CPU に割り当てるプログラムを切り替える(J)時分割処理により，複数のプログラムが同時実行されているかのように見せることができる．

(オペレーティングシステム:20 点)

- 実行中のプログラムは，(A)プロセスと呼ばれ，OS からメモリやファイルなどの計算資源の割り当てを受ける．これはさらに(B)スレッドと呼ばれる単位に分けて実行される．
- 複数のプログラムが同時に実行されている(C)マルチプログラミング環境では，メモリや，補助記憶装置，その他の入出力装置 1 台に対して同時に複数のアクセス要求が来ることがある．このような場合に，アクセス要求を調停することも OS の重要な役目である．
- 補助記憶装置に対するアクセス要求の調停は，(D)ヘッドの位置の調整問題になる．この問題の最も効率的なアクセスの調停方法は(E)C-Look アルゴリズムである．
- メモリの動的確保と解放は，メモリ領域の(F)断片化を招く．これを防ぐために，MMU というアドレス変換ハードウエアが用いられる．これは，(G)物理アドレスと論理アドレスを対応づける装置であり，(H)ページと呼ばれる固定サイズメモリを単位としてアドレス変換が行われる．
- 補助記憶装置の一部を利用して，物理メモリの大きさよりも大きな論理メモリ空間を実現する手法を(I)仮想記憶と呼ぶ．この実現方式としては，(A)が確保しているメモリ単位で，補助記憶装置上のバッキング・ストア領域に退避・復帰させる(J)プロセス・スワップと，MMU で管理している(H)単位でバッキング・ストアと(G)メモリとの間で退避・復帰させる(K)デマンド・ページングの 2 種類がある．
- (K)のために，(G)メモリの空き容量がある程度減った時点でバッキング・ストアに不要なメモリを退避させておく処理が必要になる．これを行うのが，(L)ページアウトデーモンである．このアルゴリズムには，最も古いものを退避させる(M)FIFO アルゴリズムや，最も長

期間使われていないものを退避させる LRU アルゴリズムがある．但し，(M)アルゴリズムでは，(G)メモリが大きい場合の方が退避・復帰動作の回数が増える(N)Brady の異常と呼ばれる現象が発生するため， LRU アルゴリズムのほうがよく用いられる．

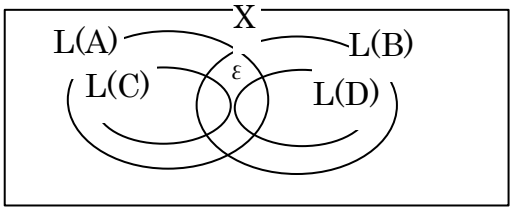
- コンピュータに接続される装置ごとに異なる操作コマンドを open, close, read, write など統一的なインタフェースで扱えるようにするのが(O)デバイスドライバの役割である．これは，C 言語のプログラムで表現すると，(P)関数のポインタを要素とする構造体になる．この構造体が一つのデバイスに対応しており，Unix や Linux の場合は，全デバイスをこの構造体の(Q)配列として扱っている．
- (R)ファイルとは，補助記憶装置上の論理的なデータの 1 次元並びであり，実際にはセクタ上のデータブロックを繋げた構造になっている．このデータブロックのつなげ方には 2 種類があり，データブロックを順次繋げていく(S)リンク構造と，データブロックを指すポインタの一次元配列を用いる(T)インデックス構造がある．

(プログラミング言語処理系：20 点)

- プログラミング言語処理系は，C, C++, Pascal, Basic,などの(A) 高級言語で書かれたプログラムテキストを，(1)CPU が解釈実行可能な(B) 機械語に変換する(C)コンパイラや，(2)逐次解釈・実行するインタープリタ，などで用いられる．
- プログラムテキストは，(D)正規文法で表現可能な字句の集まりとして表現可能であり，プログラムテキストからこれを抽出する字句解析のためにオートマトンが用いられる．
- 字句解析の後には，構文解析が行われる．通常は(E) 文脈自由文法を用いてプログラミング言語を定義することができる．
- 構文解析では，(F) 出発記号に対してどのような順序で(G) 生成規則が適用され，入力として与えられた終端記号の列が生成されたのかを分析する．この分析結果は(H) 導出木で表すことができる．
- (H)木は，分岐のない中間節をまとめ，演算子を分枝節に引き上げることで(I) 構文木（演算子木とも呼ばれる）に変換することができる．
- 構文解析の方法は大きく分けて下降型の構文解析と，上昇型の構文解析法がある．下降型構文解析では(J)最左導出を前提とした文法の下での構文解析が行われ，上昇型構文解析では(K)最右導出を前提とした構文解析が行われる．
- 下降型構文解析の例としては(L) LL 構文解析，上昇型構文解析の例としては，(M) LR 構文解析がある．(M)構文解析では，アクション表と GOTO 表をもとにして，スタックを用いて (N)shift と reduce の操作を行いながら構文解析が行われる．このうち reduce は，生成規則の(O)右 辺に対応する部分を見つけて(P)左 辺に置き換えるという(Q)還元操作に対応している．
- 文法と文法にマッチした文字列に対する計算手続きを与えて，構文解析器を作るソフトウエアのことを(R) コンパイラ・コンパイラと呼ぶ．この例としては，Unix や Linux 上の Yacc やこれを拡張した Bison などがある．これらの（R）では入力の先読みをすることで構文解析の曖昧さを回避する(S) LALR というアルゴリズムである．
- Yacc や Bison は単体で用いられることはほとんどなく，字句解析用オートマトンを生成する(T) lex や flex と併用して，(C)やインタープリタが作成される．

2．　以下の正規表現（A～D）に対応する記号の集合（すなわち言語）L(A)～L(D)があるとき，L(A)～L(D)の関係はどのようになっているか？Venn 図を書いて表しなさい．但し，全体集合は X で表し，空列 ε がどこに入るかも図中に示すこと．（10 点）

A:	a*b*
B:	b*a*
C:	aa*b*
D:	bb*a*



3．　下記の文法のもとで、「3-2-1」の構文解析を行った結果を、導出木、構文木の形で表わしなさい。（導出木 7 点＋構文木 3 点：合計 10 点）

<算術式> →<算術式><加減演算子><項>|<項>

<項>→ <項><乗除演算子><因子>|<因子>

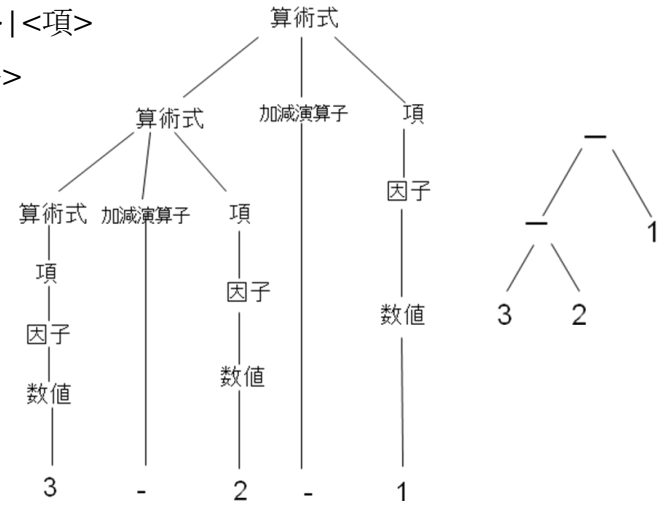
<因子>→ <変数>|<数値>|(<算術式>)

<数値>→1|2|... |9

<変数>→A|B|C|...|Z

<加減演算子>→+|－

<乗除演算子>→×|÷



4.　Linux 上で，右図①，②，③のコマンドを 10 秒以下の間隔で順に入力した．

(ア) この 5 秒後に，kill 25461 というコマンドを実行すると，端末はどのようになるか？（3 点）

端末の画面に”Wake Up”が表示される．

(イ) (ア)の代わりに kill 25460 というコマンドを実行すると，どのようになるか？（3 点）

②のコマンドで起動された全プロセスが停止・消滅する．

```
$ ps –f                                                    ・・・①

UID        PID   PPID   C STIME TTY      TIME   CMD
twada     25441 25440   0 16:54   pts/2   00:00:00 -csh
twada     25457 25441   0 16:55   pts/2   00:00:00 ps –f

$ ( sleep 100 ; echo "Wake up" )&                          ・・・②

[1] 25460

$ ps –f                                                    ・・・③

UID        PID   PPID   C STIME TTY      TIME   CMD
twada     25441 25440   0 16:54   pts/2   00:00:00 -csh
twada     25460 25441   0 16:56   pts/2   00:00:00 -csh
twada     25461 25460   0 16:56   pts/2   00:00:00 sleep 100
twada     25462 25441   0 16:56   pts/2   00:00:00 ps –f

$
```

(ウ) （ア）（イ）の代わりに kill –KILL 25441 というコマンドを実行すると，端末はどうなるか？（但し-KILL は強制的なプロセス停止を意味する）（4 点）

コマンドを解釈実行していたシェルプロセスが停止し，ログアウトする．

5. 下記の文法から下に示すアイテム集合とそれらの間の遷移図を得た．集合 7,8 の空欄を全て埋め，状態遷移の矢印一本を追加し，図を完成させなさい．また，右のアクション表と GoTo 表を完成させ，shift-reduce conflict が起きる場所全てを〇で囲みなさい．(集合 8，7，矢印：各 3 点，表 9 点，〇2 点：合計 20 点)

(1) <E> →<E>+<B>

(2) <E> →<B>

(3) <B> →<B>*<T>

(4)<B>→(<E>)

(4) <B> →<T>

(5) <T> → 0

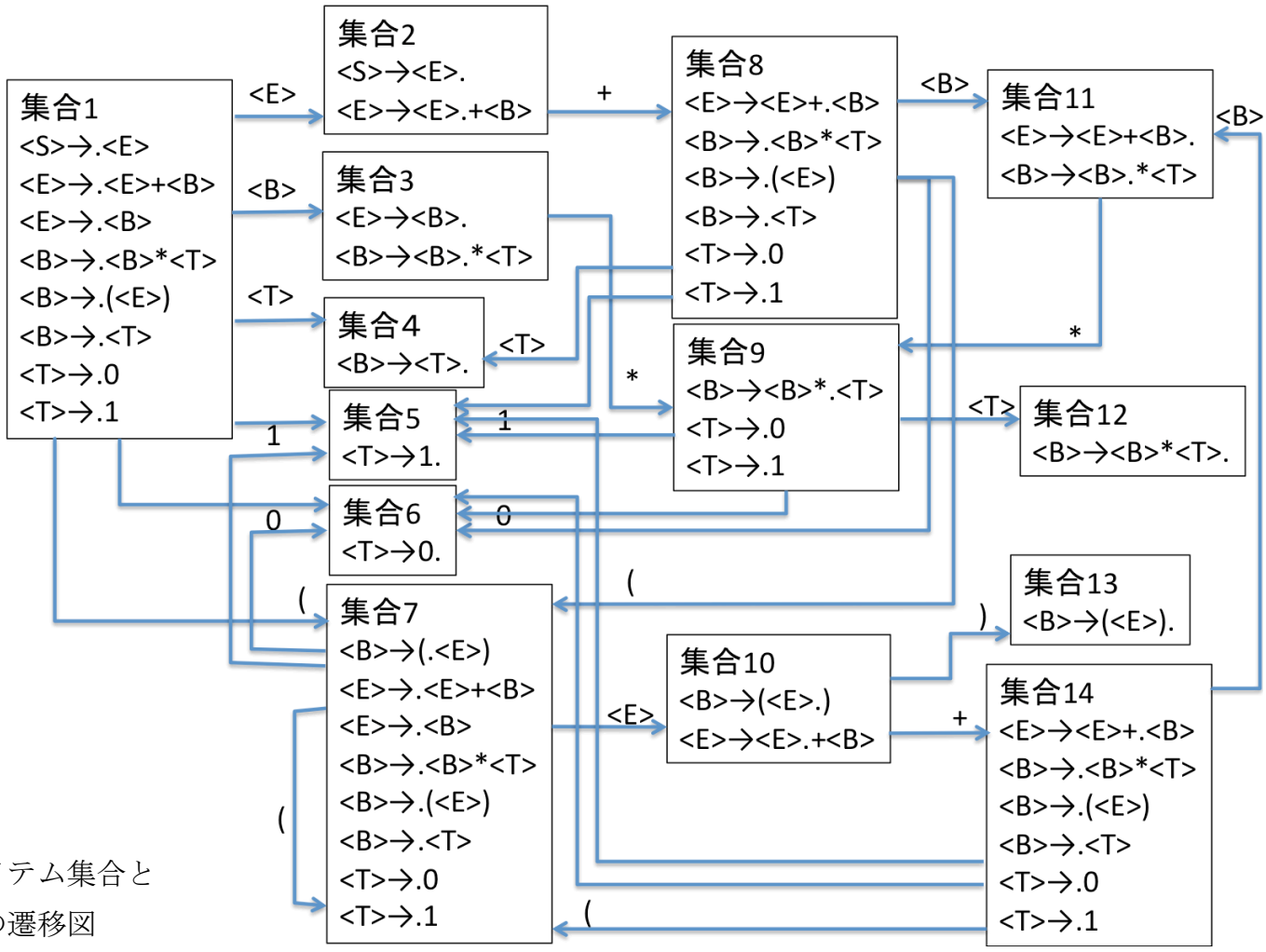
(6) <T> → 1

アクション表

	+	*	(	)	0	1	$
1			s7		s6	s5	
2	s8						acc
3	r2	s9 r2	r2	r2	r2	r2	r2
4	r4	r4	r4	r4	r4	r4	r4
5	r6	r6	r6	r6	r6	r6	r6
6	r5	r5	r5	r5	r5	r5	r5
7			s7		s6	s5	
8			s7		s6	s5	
9					s6	s5	
10	s14			s13			
11	r1	s9 r1	r1	r1	r1	r1	r1
12	r3	r3	r3	r3	r3	r3	r3
13	r4	r4	r4	r4	r4	r4	r4
14			s7		s6	s5	

Go To表

E	B	T
2	3	4
10		
	11	4
		12
	11	



アイテム集合とその遷移図