計算機シミュレーション

3年後期・選択2単位 担当：小栗 清

成績評価はテスト

授業出席は1点

表現モデル

- グラフ
- ペトリネット
- データフローモデル
- 状態遷移モデル
- エンティティアクションモデル
- 静的ロジックモデル
- オブジェクト指向モデル

時間軸

- 実時間
- クロック（同期信号）
- 擬似乱数
- M系列
- 確率分布
- 確率過程（ランダムウォーク）
- 拡散方程式

離散系シミュレーション

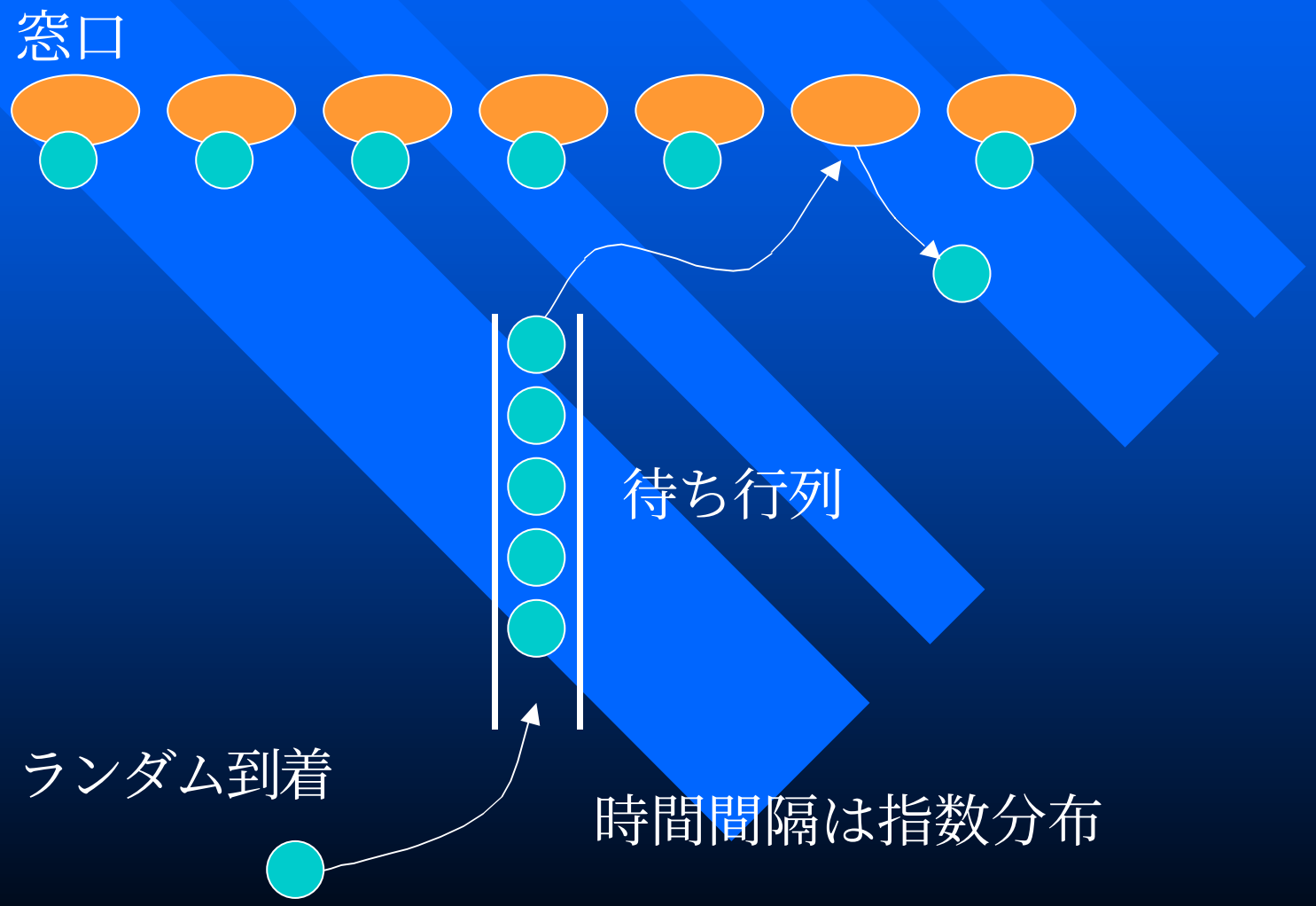
- 待ち行列のシミュレーション
- 論理シミュレーション（イベントドリブン方式）
- 故障シミュレーション（並列方式）
- 並列動作のシミュレーション（同期動作）
- 生態系のシミュレーション（非同期動作）
- 政策決定用シミュレーション
- ・・・

連続系シミュレーション

- 電子回路シミュレーション
- 電磁波シミュレーション
- 電子デバイスシミュレーション
- フライトシミュレーション
- 気象シミュレーション
- ・・・

突然

待ち行列のシミュレーション



指数分布

- 一直線上にランダムに打たれた多数の点を考え，その隣り合う2点間の距離を問題とする.
- 点はこの直線上の任意の位置に長さdxの線素片を考えたとき，その上に~~1つの~~点がある確率がdx/λに等しいように打たれているものとする．ここでλは一定.
- 点の平均密度が1/λである.

市村浩「統計力学」より引用

距離 x の間に点のない確率$P_0(x)$

- いま，この直線上で任意のところに長さxの区間を考えたとき，その上に点が全くない確率$P_0(x)$を求める．x+dxの区間上に，点が全くない確率$P_0(x+dx)$は，x 上に点が無く，次のdx 上にも点が無い確率となるので，
- $P_0(x+dx)=P_0(x)(1-dx/\lambda)$

である.

距離 x の間に点のない確率$P_0(x)$

- $P_0(x+dx)=P_0(x)(1-dx/\lambda)$の左辺をTayler展開して$(dx)^2$以上を無視し，$P_0(x)+(dP_0/dx)dx$とすると,
- $(dP_0(x)/dx)dx= -P_0(x)\cdot dx/\lambda$
- $1/P_0(x)\cdot (dP_0(x)/dx)= -1/\lambda$
- $1/P_0(x)\cdot dP_0(x)= -1/\lambda \cdot dx$　　これを積分して
- $\log P_0(x)=-x/\lambda +const.$
- $P_0(x)=e^{-x/\lambda}$となる．←結論
- ここでx=0，つまり区間の長さが0のときにはその上に点の打たれる確率は0，すなわち，点のない確率が1と考えられるので，積分定数をそれに合わせた.

初めて点がある確率

- それではxとx+dxの間に初めて点がある確率$f(x)dx$は，xの間には点が無く，次のdxの間に点がある確率と考えられるから
- $f(x)dx=P_0(x)\cdot dx/\lambda=e^{-x/\lambda}/\lambda\cdot dx$
- $f(x)=e^{-x/\lambda}/\lambda$ ←結論
- これは間隔の分布の確率である.
- 確率の総和 $\int_0^\infty e^{-x/l}/l\,dx=\left[-e^{-x/l}\right]_0^\infty=0-(-1)=1$
- 平均間隔 $\int_0^\infty x\cdot e^{-x/l}\Big/l\,dx=l$

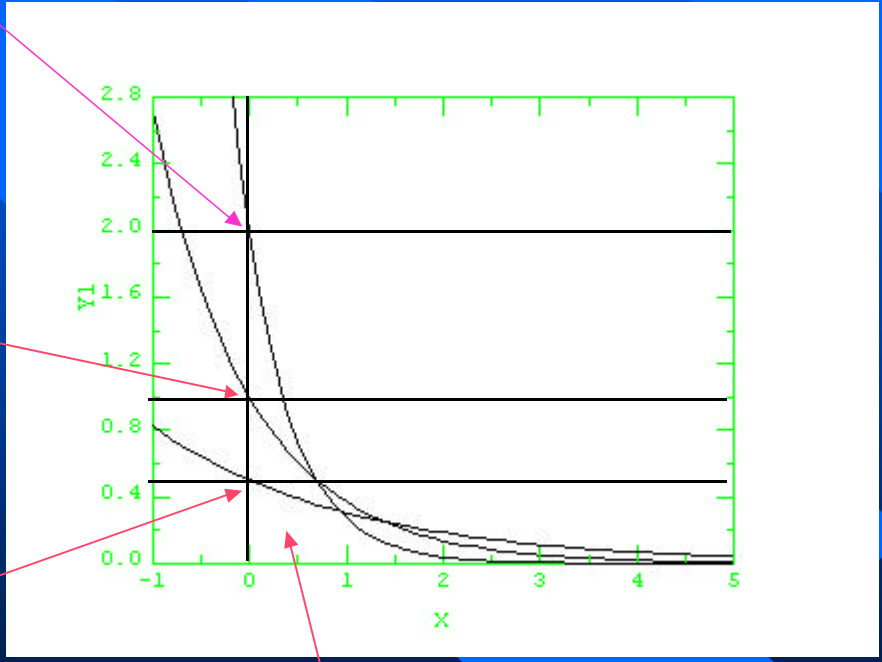
指数分布（間隔の分布）と密度

平均密度=2

$y=2e^{-2x}$

平均間隔=1/2

$y=e^{-x/\lambda}/\lambda$　ここで$1/\lambda$：平均密度



平均密度=1

$y=e^{-x}$

平均間隔=1

平均密度=1/2

$y=e^{-x/2}/2$

平均間隔=2

この部分の面積は1

指数分布の累積関数

- 指数分布の累積を考える.

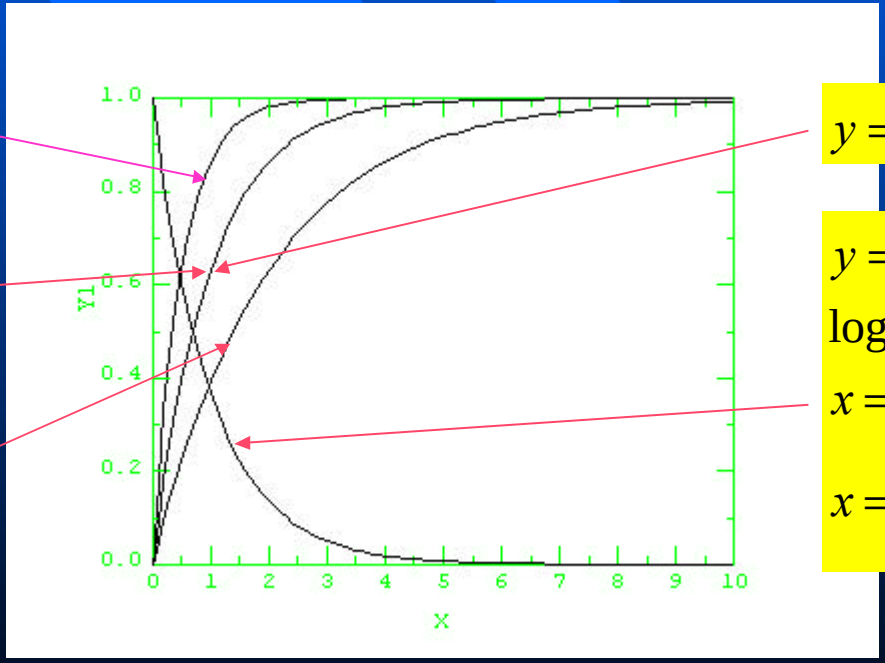
$$\int_0^x e^{-x/l}/l\,dx=\left[-e^{-x/l}\right]_0^x=-e^{-x/l}-(-1)=1-e^{-x/l}$$

ここで1/λ：平均密度

平均密度=2

平均密度=1

平均密度=1/2



$$y=1-e^{-x/l}$$

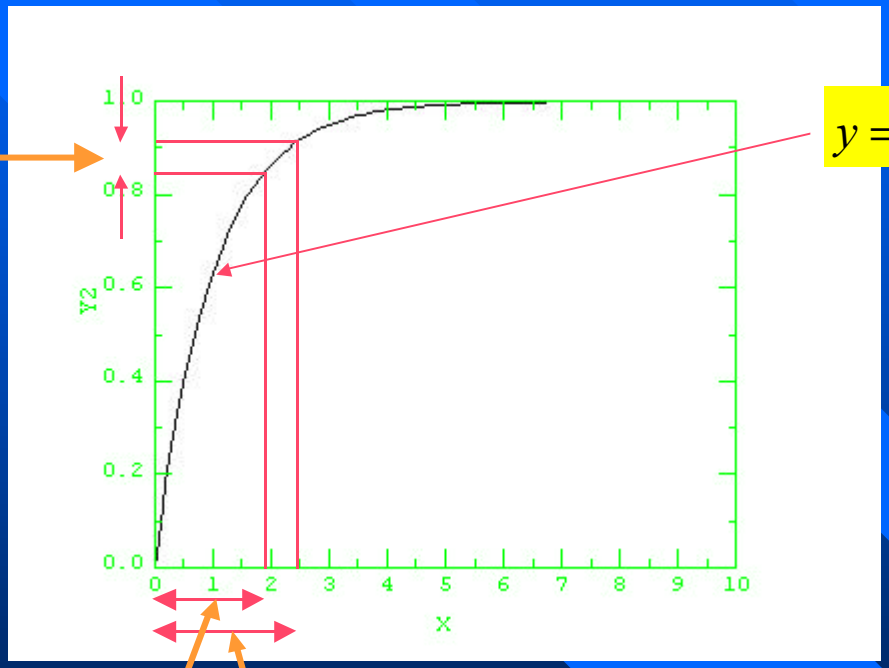
$$y=e^{-x/l}$$

$$\log y=-x/l$$

$$x=-l\log y$$

$$x=l\log\frac{1}{y}$$

指数分布の累積関数の意味



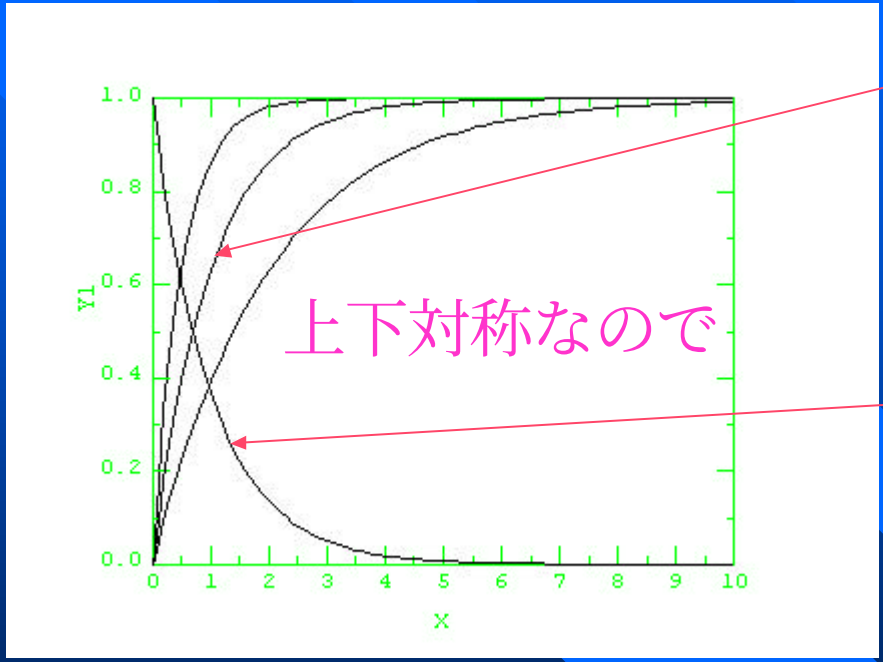
$y = 1 - e^{-x/l}$

間隔が　から　までである確率は

従ってＹ方向に等確率で点を打って，それをＸ方向にマッピングすればその間隔は等確率で出現する.

一様分布と指数分布

ここでλ：平均間隔



$$y = 1 - e^{-x/l}$$

$$y = e^{-x/l}$$

$$\log y = -x/l$$

$$x = -l \log y$$

$$x = l \log \frac{1}{y}$$

YからXへの変換にを使う

待ち行列のシミュレーション

- rnd()を0から1までの一様乱数の発生関数として，平均 t の指数分布の乱数(平均すると 時間間隔 t ごとに事象が起こる場合の事象間の時間間隔)を与える関数は

```
double exp_rnd(int t){
  return t*(log(1/rnd()));}.
```

- 到着間隔の平均を ta，対応時間の平均をtbとして，到着間隔はexp_rnd(ta)，対応間隔はexp_rnd(tb).

待ち行列のシミュレーション

基本データ構造

- 窓口対応中フラグ
- 待ち行列人数
- 現在時刻
- イベントリスト

待ち行列のシミュレーション

1. 最初，現在時刻を0とし，待ち行列に1人入れ，
   exp_rnd(ta)時刻の到着イベントをイベントリストに登録
2. 窓口のいずれかが空いていて待ち行列に人がいるとき，
   空いていた窓口の任意の一個を対応中とし，
   待ち行列の人数を一人減らし，
   (現在時刻＋exp_rnd(tb))時刻のその窓口の
   対応終了を表すイベントをイベントリストに登録
   この操作を窓口が全て対応中となるか待ち行列が空になるまで行い，3.へ
3. 最も時刻が早いイベントの時刻を現在時刻とし，そのイベントが
3a 到着イベントであった時，待ち行列に一人追加するとともに，
   (現在時刻＋exp_rnd(ta))時刻の到着イベントをイベントリストに登録．2.へ。
3b 対応終了イベントであった時，その窓口を空き状態とする．2.へ。

queue.cの宣言部

```
 1  #include <stdio.h>
 2  #include <math.h>
 3  #include <stdlib.h>
 4  #define NofClerk 3
 5  enum event_type {arival,finish};
 6  struct event_node {
 7    long              time;
 8    enum event_type    type;
 9    int               clerk_id;
10     struct event_node *next;
11  };
12  long              mean_arival_interval;
13  long              mean_operation_interval;
14  long              quit_time;
15  int               clerk_busy[NofClerk];
16  int               nof_person_in_queue;
17  long              current_time;
18  struct event_node *event_root;
19  void event_add(long interval,enum event_type type,int clerk_id);
20  long exp_rnd(long mean_interval);
```

BSD系の場合，この辺は環境による.

queue.cのmain関数

```
21  int main(int argc,char **argv){
22    int i;
23    struct event_node *rm_event;
24    if(argc!=4){
25      printf("usage: queue arival_interval operation_interval quit_time¥n");
26      return 1;
27    }
28    sscanf(argv[1],"%d",&mean_arival_interval);
29    sscanf(argv[2],"%d",&mean_operation_interval);
30    sscanf(argv[3],"%d",&quit_time);
31    for(i=0;i<NofClerk;i++) clerk_busy[i]=0;
32    nof_person_in_queue=1;
33    current_time=0;
34    event_root=NULL;
35    event_add(exp_rnd(mean_arival_interval),arival,0);
      …. メインループ
66  }
```

queue.cのmain関数のメインループ

```
36  for(;;){
37   if(current_time>quit_time)return 0;
      … 表示
44   for(;;){
45    if(!nof_person_in_queue) break;
46    for(i=0;i<NofClerk;i++){
47     if(!clerk_busy[i]) break;
48    }
49    if(i==NofClerk) break;
50    clerk_busy[i]++;
51    nof_person_in_queue--;
52    event_add(exp_rnd(mean_operation_interval),finish,i);
53   }
54   rm_event=event_root;
55   event_root=rm_event->next;
56   current_time=rm_event->time;
57   if(rm_event->type==arival){
58    nof_person_in_queue++;
59    event_add(exp_rnd(mean_arival_interval),arival,0);
60   }
61   else if(rm_event->type==finish)
62    clerk_busy[rm_event->clerk_id]=0;
63   else ;
64   free(rm_event);
65  }
```

```
38   /*++++++++++++++++++++++++++++++*/
39   printf("time=%3d nof_p=%3d",
                current_time,nof_person_in_queue);
40   for(i=0;i<NofClerk;i++)
41    printf(" busy[%d]=%d",i,clerk_busy[i]);
42   printf("¥n");
43   /*++++++++++++++++++++++++++++++*/
```

```
67  void event_add(long interval,enum event_type type,int clerk_id){
68    struct event_node *new_event;
69    struct event_node *event;
70    new_event=(struct event_node *)malloc(sizeof(struct event_node));
71    new_event->time=current_time+interval;
72    new_event->type=type;
73    new_event->clerk_id=clerk_id;
74    if(event_root==NULL){
75      new_event->next=NULL;
76      event_root=new_event;
77      return;
78    }
79    if(event_root->time>new_event->time){
80      new_event->next=event_root;
81      event_root=new_event;
82      return;
83    }
84    for(event=event_root;;event=event->next){
85      if(event->next==NULL){
86        new_event->next=NULL;
87        event->next=new_event;
88        return;
89      }
90      if(event->next->time>new_event->time){
91        new_event->next=event->next;
92        event->next=new_event;
93        return;
94      }
95    }
96  }
```

queue.cのevent_add関数

queue.cのexp_rnd関数

長い整数

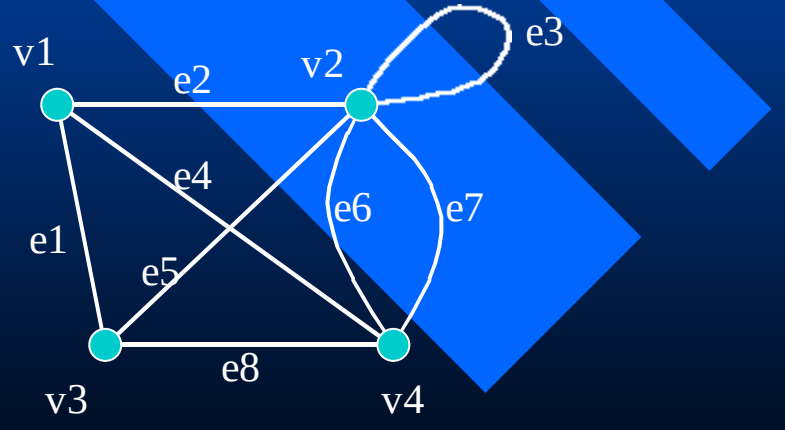
倍精度の浮動小数点数

```
 97  long exp_rnd(long mean_interval){
 98    double r;
 99    r=(double)mean_interval*(log((double)RAND_MAX/(double)random()));
100    return (long)r;
101  }
```

random()は0からRAND_MAXまでの一様分布の乱数

グラフ理論

- グラフとはいくつかの点と，その点を結ぶ線からなる図形
- 点のことを節点，線のことを枝と言う
- 点の位置，線が直線か曲線かは不問

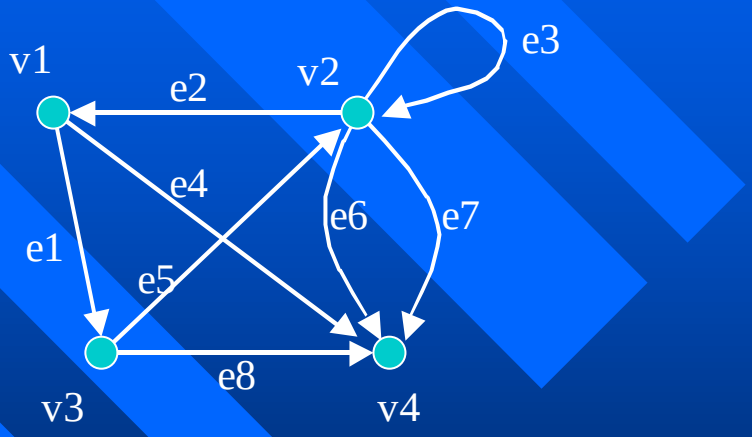


グラフの定義

- グラフGとは空でない節点の集合V
- と枝の集合E
- およびそれらの間の接続関係$\phi : E \to V \times V$
- のことを言う.
- $G=(V, E, \phi)$, $G=(V, E)$
- ϕはEの元を1つ決めたら，Vの元の対が決まる関数，×は直積

いろいろなグラフ

- 無向グラフ
  - 枝に向きが無い
- 有向グラフ
  - 枝に向きが有る
- 重み付きグラフ
  - 枝や節点に値がある



表記法，用語 (1/6)

- $e=(v_i, v_j)$
  - 枝 e が節点 v_i と v_j を結んでいる
  - v_i, v_j を e の端点という
- $e=(\overrightarrow{v_i, v_j})$
  - 有向グラフの場合
  - v_i を e の始点，v_j を終点という
- 節点の次数
  - 節点 v に接続している枝の数

表記法，用語 (2/6)

- セルフループ
  - 両端点が同一の節点である枝
  - その節点の次数は 2
- 正の次数
  - 有向グラフで節点から出る枝の数
- 負の次数
  - 有向グラフで節点に入る枝の数
- 節点が隣接
  - その節点を結ぶ枝が有る

表記法，用語 (3/6)

- 枝が隣接
  - ある節点を共通の端点としてもつとき
- 長さ (n-1) の路
  - $(v_1,v_2),(v_2,v_3),\ldots,(v_{n-1},v_n)$
- 閉路
  - $(v_1,v_2),(v_2,v_3),\ldots,(v_{n-1},v_1)$
- 有向路
  - $(\overrightarrow{v_1,v_2}),(\overrightarrow{v_2,v_3}),\ldots,(\overrightarrow{v_{n-1},v_n})$

表記法，用語 (4/6)

- 単純路
  - 同じ枝が2度以上現れない路
- 初等的な路
  - 同じ節点が2度以上現れない路
- 枝の開放除去
  - 枝をグラフから取り除くこと
- 枝の短絡除去
  - 枝をグラフから取り除き，両端点をくっつける

表記法，用語 (5/6)

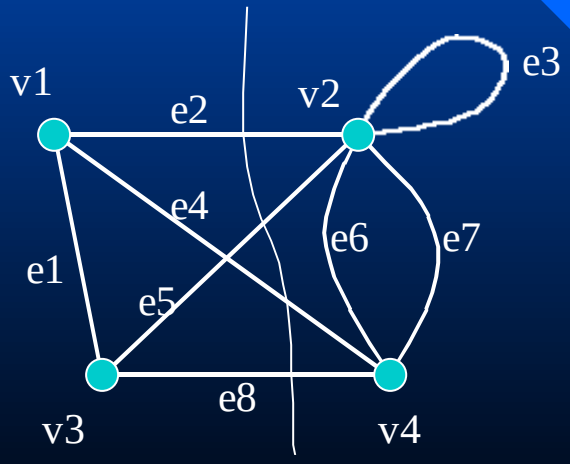
- 節点の除去
  - 節点とその節点を端点とする枝を取り除くこと
- 部分グラフ
  - 枝の開放除去と節点の除去を0回以上行って得られるグラフ
- 木
  - 閉路を含まないグラフ

表記法，用語 (6/6)

- Gを張る木
  - Gの部分グラフでもう1本枝を加えると閉路を含んでしまう木
- 同形
  - 2つのグラフのグラフとしての形が同じ
- A に対するセクショングラフ
  - $G=(V, E)$のVの部分集合A以外の節点を除去したGの部分グラフ

カットセット

- グラフ$G=(V, E)$の節点Vを2つの部分V_1, V_2に分けたとき，V_1の節点とV_2の節点を結ぶ枝の集合
  - カットセットの全ての枝を開放除去すると2つの部分に分離する



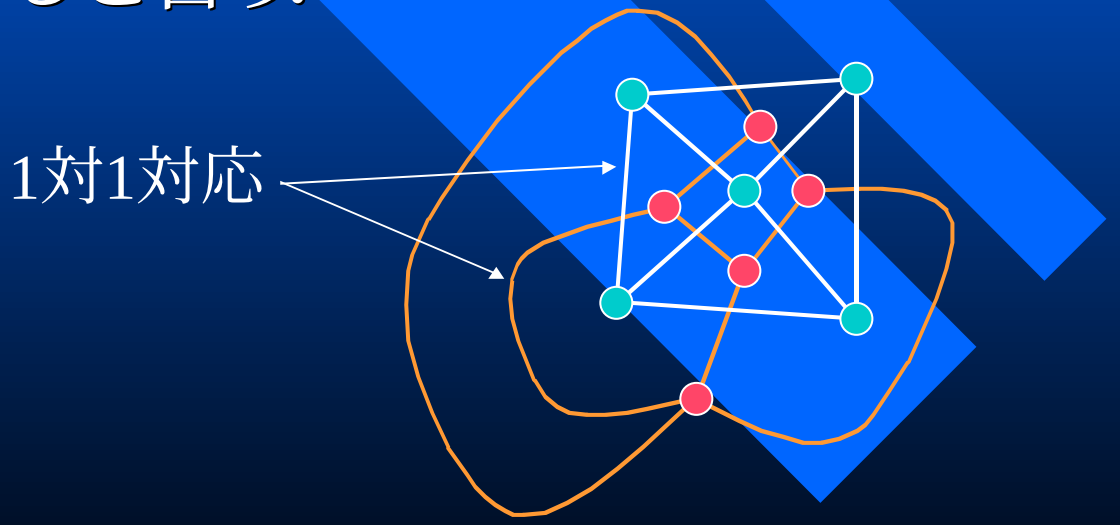
$V_1=\{v_1,v_3\}$　　$V_2=\{v_2,v_4\}$

カットセット$E_1=\{e_2,e_4,e_5,e_8\}$

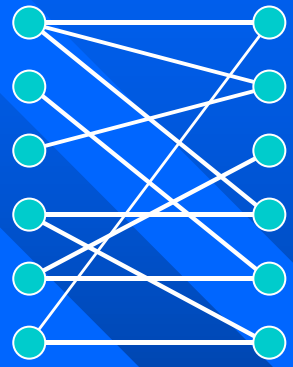
$C(E_1)=(V_1,V_2)$ と書く

双対なグラフ

- 2つのグラフ$G_1=(V_1,E_1)$と$G_2=(V_2,E_2)$が，E_1とE_2の間に1対1対応があり，G_1のカットセットがG_2で単純閉路に，G_1の単純閉路がG_2のカットセットになるとき，G_1とG_2は双対であると言う.



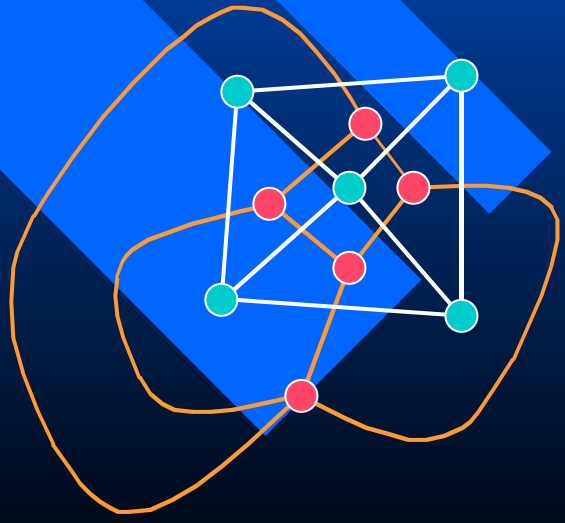
2部グラフ



- 2部グラフである必要十分条件は，長さが奇数の閉路を含まないことである.

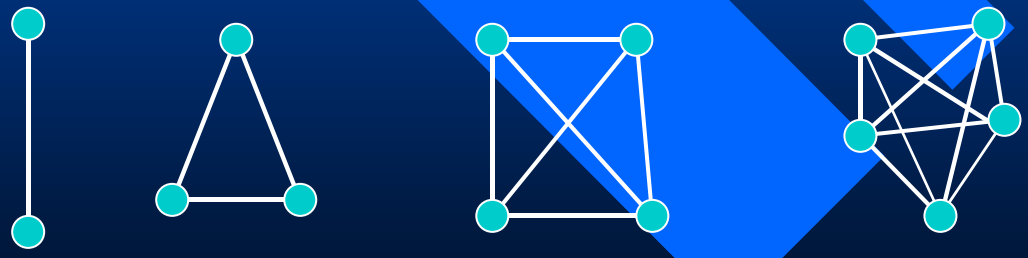
平面グラフ

- グラフ$G=(V,E)$が，平面上に枝の交差なしに描けるとき，Gを平面グラフと言う.
  - Gが平面グラフである必要十分条件はGが双対グラフをもつことである.



完全グラフ

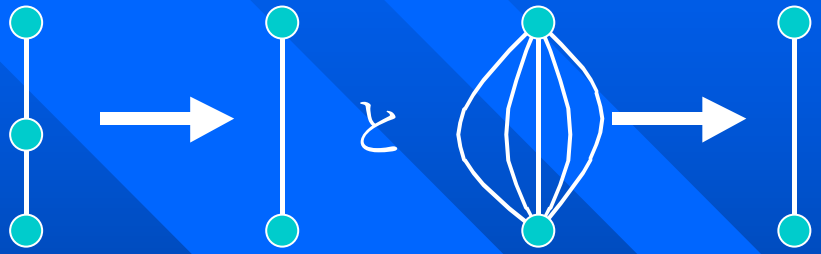
- セルフループを含まず，どの節点対に対しても，それらを端点とする枝が存在するグラフ
- 完全グラフの次数
  - 完全グラフの節点数



クリーク

- Gのクリーク
  - Gの部分グラフで完全グラフであるもの
- 極大クリーク
  - 他のクリークに含まれないもの
- 最大クリーク
  - Gのクリークのなかで次数が最大のもの
- Gのクリーク数
  - Gの最大クリークの次数

直並列グラフ



と

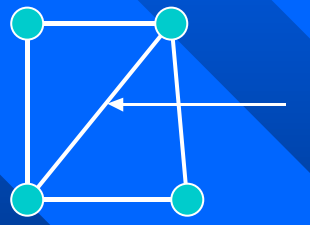
を繰り返し適用して

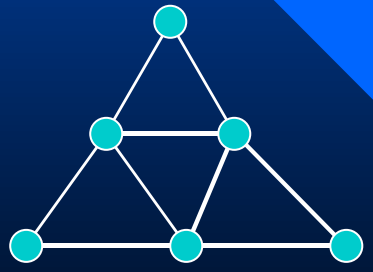
最終的に

とできるグラフ

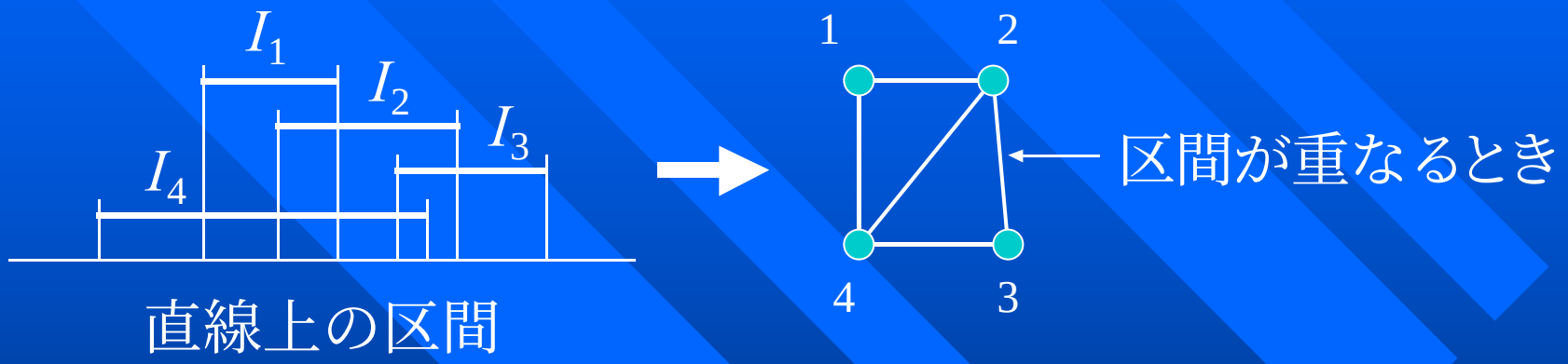
三角化グラフ

- 長さ4以上の初等的閉路が必ず弦を持つグラフ





区間グラフ



- 区間グラフは三角化グラフである.
- ただし逆は成り立たない.

グラフ上の問題(1/2)

- 平面判定問題(m)
- 平面化問題(NP完全)
  - 最少個数の枝を除去して平面グラフにする
- 同形判定問題(?)
  - 部分グラフ同形判定(NP完全)
- 彩色数問題(NP完全)
  - 4色問題
- 最大マッチング問題(NP完全)
  - 隣接しない枝集合の最大のものを求める
  - お見合い問題

グラフ上の問題(2/2)

- 最短路(m^2)
  - ダイクストラのアルゴリズム
- 最長路(NP完全)
- 最小木(m^2)
  - Gを張る木のなかで枝の重みが最少のもの
- スタイナー木(NP完全)
- 最大カットセット(NP完全)
- 頂点被覆問題(NP完全)

P問題とNP問題

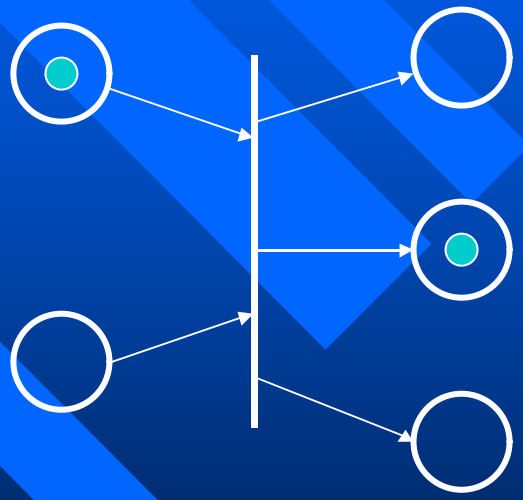
- 決定性算法
  - 一度に一つの場合しか処理を行わない
- 非決定性算法
  - 場合分けを同時に処理する
- P問題
  - 決定性算法で多項式時間かかる問題
- NP問題
  - 非決定性算法で多項式時間かかる問題

NP完全とNP困難

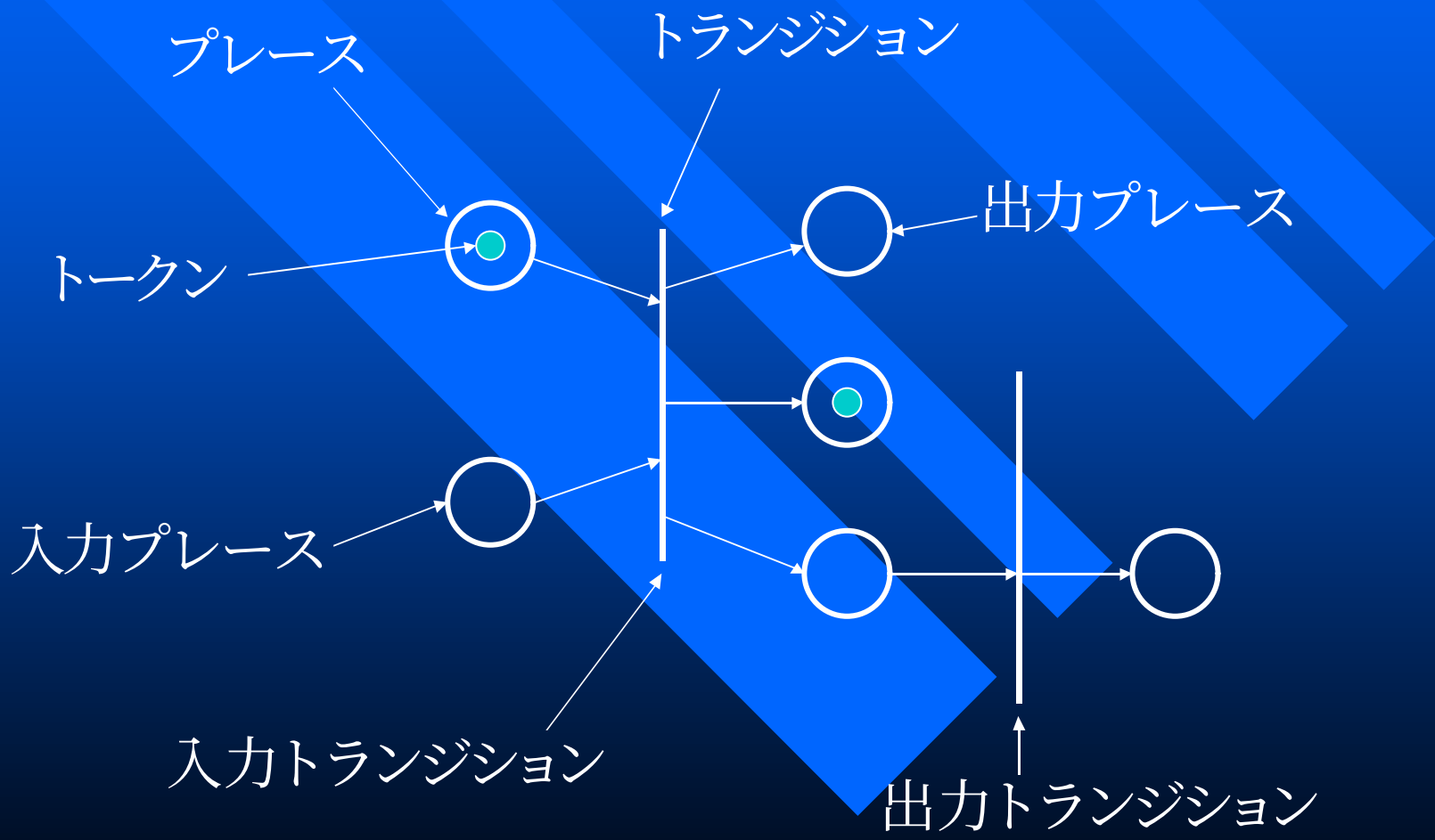
- NP完全問題
  - 決定性算法での多項式時間で，任意のNP問題から変換できるあるNP問題
- NP困難問題
  - 決定性算法での多項式時間で，任意のNP問題から変換できるある問題

ペトリネットとは

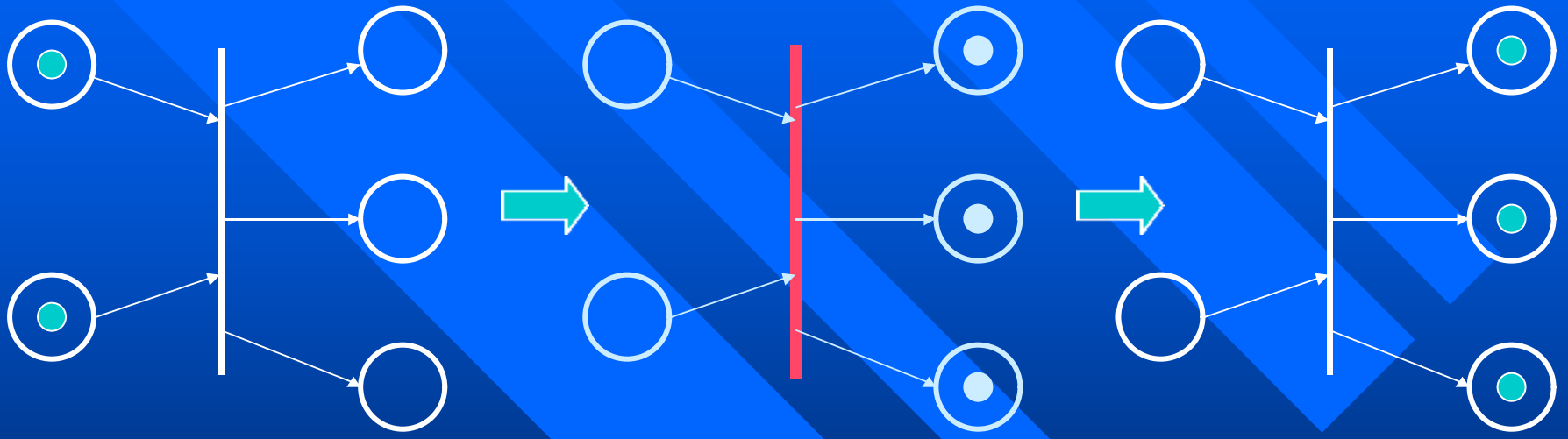
- 円，棒，有向枝，点からなる図形
- 円と棒を節点と見れば二部有向グラフ＋点
- 円（＝プレース）：要求(情報)の居場所
- 棒（＝トランジション）：変化
- 有向枝：因果関係
- 点（＝トークン）：要求(情報)



名称



トークンの動作



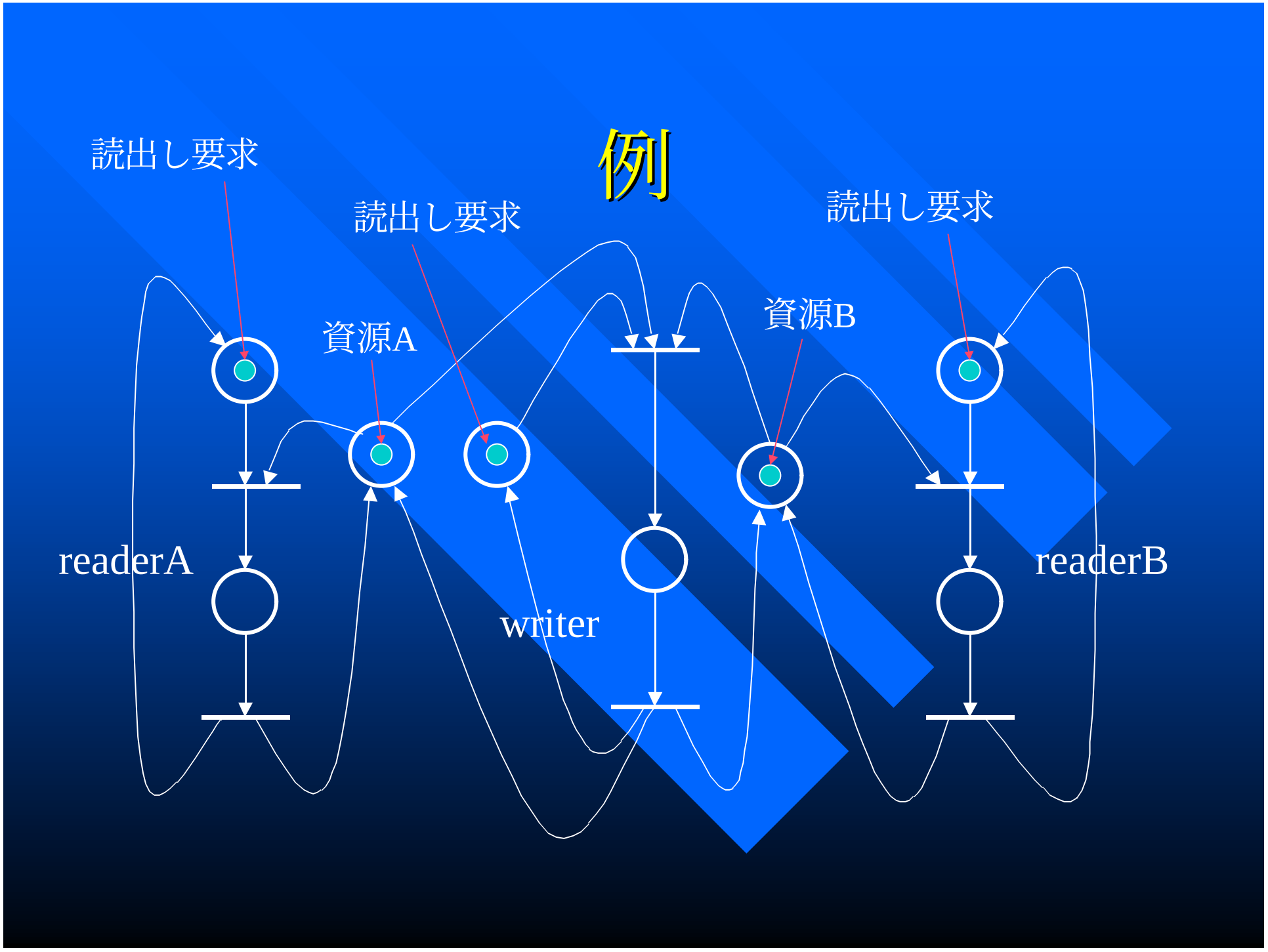
- 入力プレースにトークンがそろうと
- トランジションが発火（ファイアー）して
- 出力プレースのトークンは1個増える
- 入力プレースのトークンは1個減る

全体を考えると

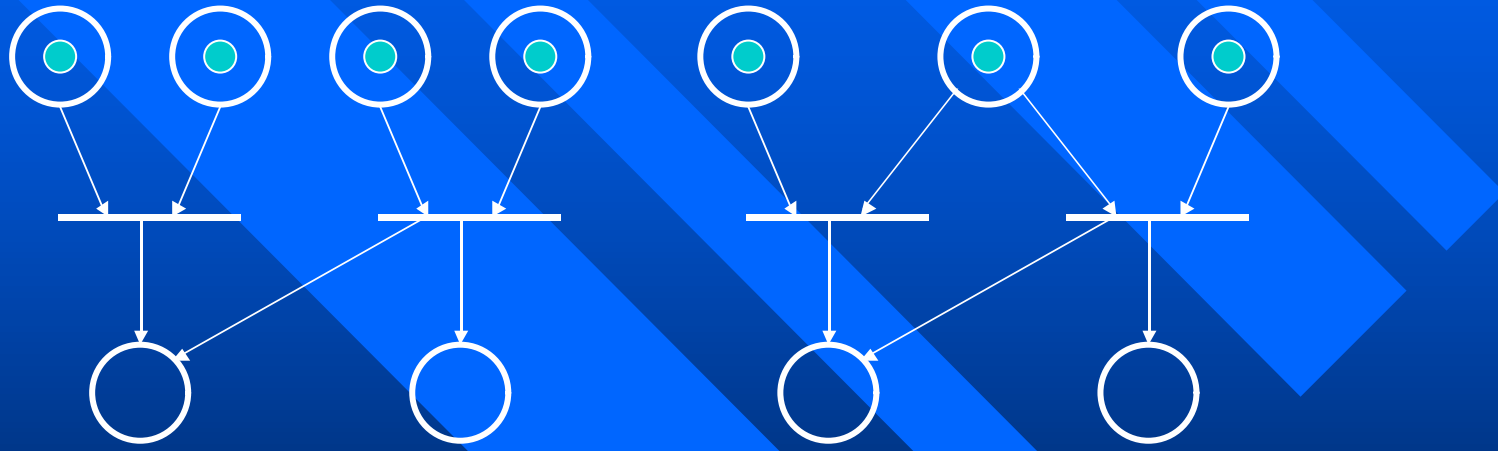
- トークンの配置をマーキングという
  - システム全体の状態が表される
  - プレースにトークンは複数個あっても良い
- 入力プレースにトークンがそろったトランジションを発火可能なトランジションという
- 一時に発火するトランジションは一つ
  - 発火可能なトランジションのどちらが発火するかはペトリネットでは規定されない
- 次々とマーキングが変化していく
  - これをシステムの動作ととらえる

なぜ使うのか

- 非同期システムにおける情報の流れを示す
- 部分クラスとして順序機械を含む
- 並列動作，衝突を含む複雑な非同期システムの振舞いを表現
- 非同期システム＝非同期回路というわけではない，OSの動作，物流，工程管理なども非同期システム


例
読出し要求
読出し要求
読出し要求
資源A
資源B
readerA
writer
readerB

並列動作と衝突



並列動作　　衝突動作

到達可能

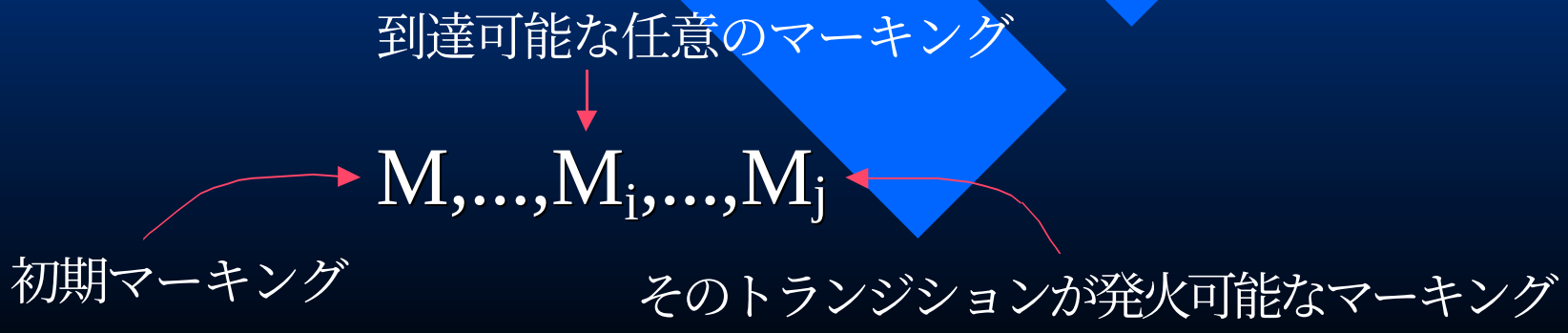
- 1つの発火によりマーキングM_1からM_2へ遷移できるとき
- M_2はM_1から直接到達可能という
- 直接到達可能なマーキングの列 $M_i,M_{i+1},M_{i+2},\ldots,M_j$が存在するとき$M_j$は$M_i$から到達可能という
- 初期マーキングMから到達可能なマーキングの集合（到達可能集合）をR(M)と書く

有界，安全

- プレースがk-有界
  - 初期マーキングから到達可能な任意のマーキングに対して，そのプレースのトークンの数がk以下であること
- プレースが安全
  - プレースが1-有界であること
- ペトリネットが安全
  - 全てのプレースが安全
- 資源が有限なハードウェアで重要

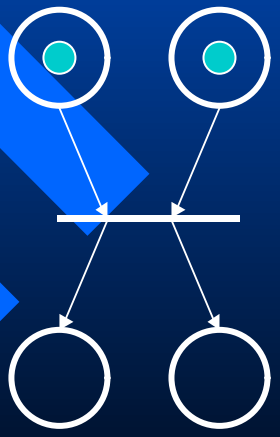
生きている

- トランジションが生きている
  - 初期マーキングから到達可能な任意のマーキングから到達可能なあるマーキングでそのトランジションが発火可能であること
- ペトリネットが生きている
  - すべてのトランジションが生きていること
  - デッドロックに陥らない



保存的

- ペトリネットが保存的
  - 初期マーキングから到達可能な任意のマーキングに対してトークンの総和が一定
- 必要十分条件は
  - 発火可能なトランジションの入力プレース数と出力プレース数が等しいこと



到達可能木

- 有界，安全，生きている，保存的などの性質は到達可能集合を求めることにより調べることができる
- プレースのトークンの数は必ずしも有限ではないので到達可能集合も有限集合とは限らない
- 到達可能木
  - 到達可能集合を有限な形式で表現する

特別な記号ω

- ωは任意の自然数nに対して

  $n<\omega$

  $\omega+n=\omega$

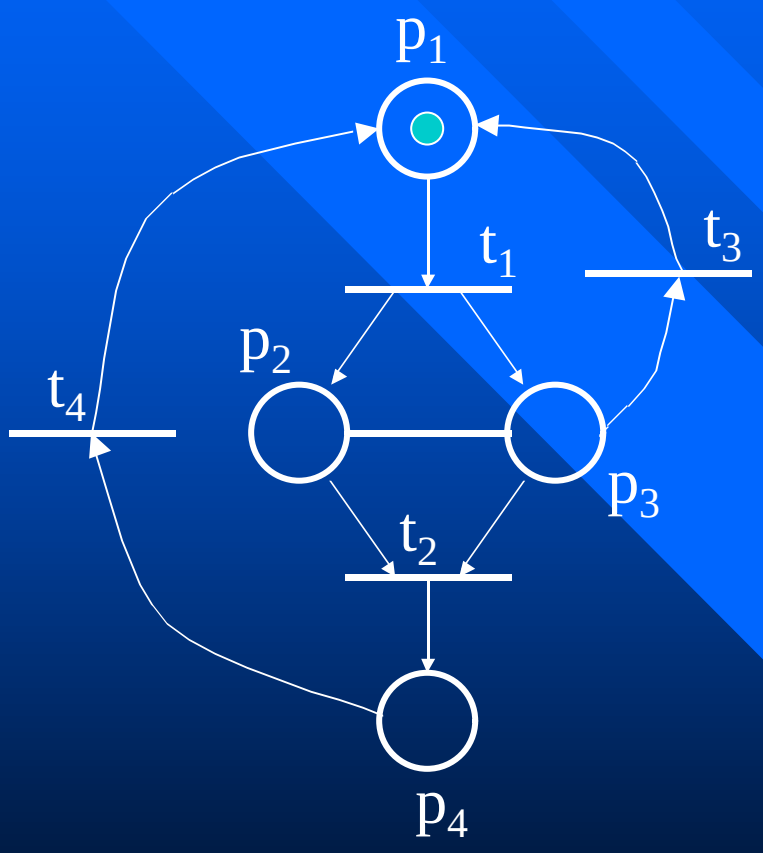
  $\omega-n=\omega$

- ωは非常に大きな数を表す

到達可能木の作り方

- 基本
  - マーキングを○，遷移を→で表す
  - 初期マーキングを最初の○として追加する
  - 今追加した○から直接到達可能なマーキングを表す○を追加し，矢印で結ぶ
- 修正
  - 追加した○とまったく同じマーキングがそこまでの経路上に存在する時，この○で止める
  - 追加した○がそこまでの経路上のマーキングにトークンの数を加えたものになっている時，この○のトークンの数をωとする

例



(1,0,0,0)

t_1

(0,1,1,0)

t_2 t_3

(0,0,0,1) (1,1,0,0)

t_4

(1,0,0,0) (1, ω ,0,0)

t_1

(0, ω ,1,0)

t_2 t_3

(0, ω ,0,1) (1, ω ,0,0)

t_4

(1, ω ,0,0)

含む

同じ

ωがあるので
有界でない
安全でもない
保存的でもない
生きてはいる

ここで終わり

到達可能木から判定

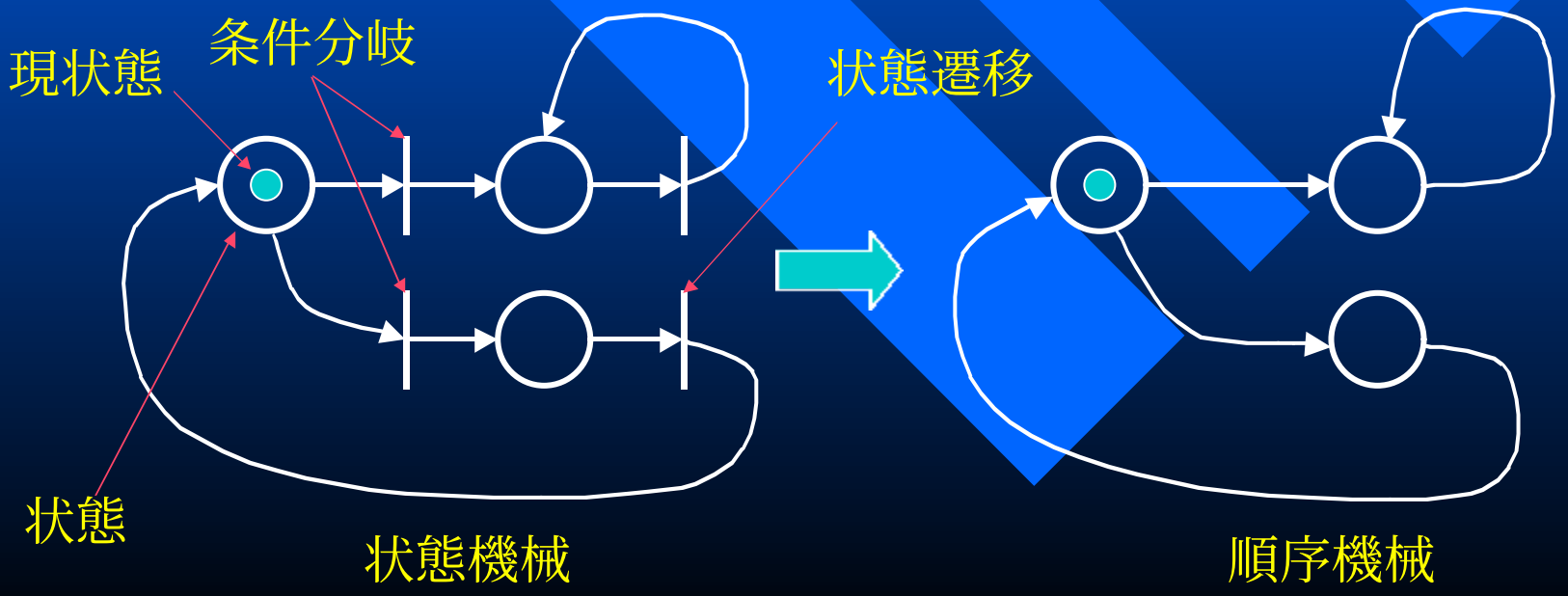
- ωがある時
  - 有界でない
  - 安全でもない
  - 保存的でもない
- ωがなければ
  - 有界
- それ以外も判定が容易
  - 先の例では
    - 「どのトランジションも必ず発火可能となるマーキングに到達可能」であることを容易に確認できる
    - 従って「ペトリネットが生きている」ことがわかる

ペトリネットの部分クラス

- 部分クラスの考え方
  - ペトリネット
    - » 非同期システムのモデル化能力が高い
    - » 解析は容易ではない
  - 一定の制限を設けて解析を容易に
- 部分クラスの例
  - 状態機械
  - マーク付きグラフ
  - 自由選択ネット
  - 単純ペトリネット

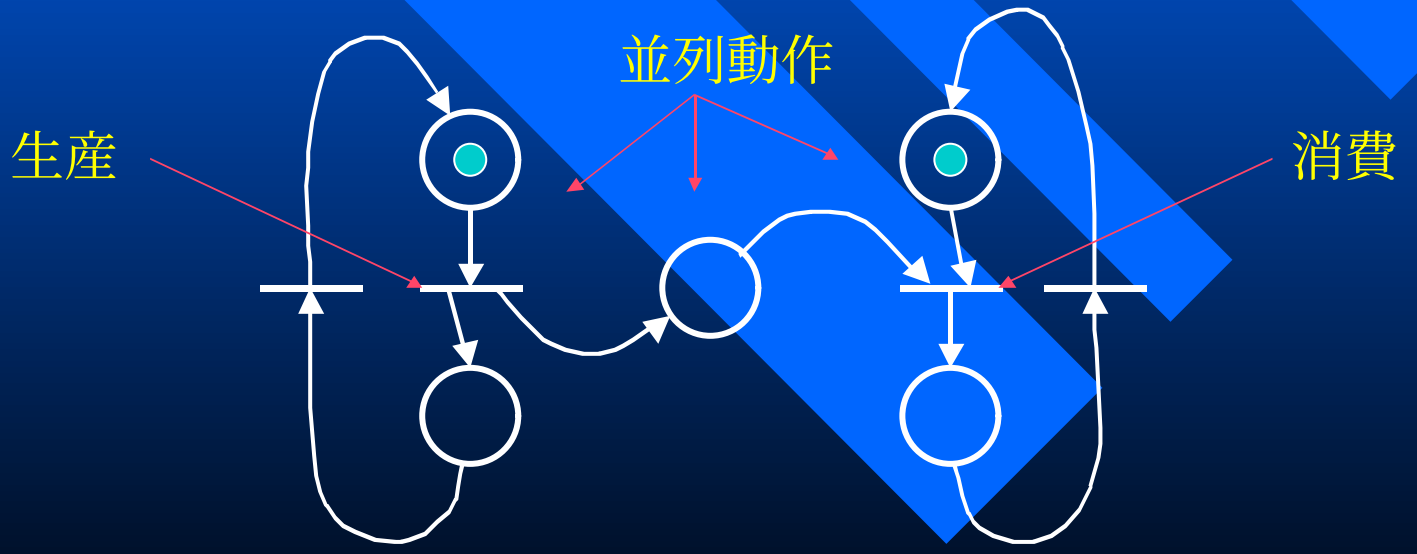
状態機械

- どのトランジションもただ一つの入力プレースとただ一つの出力プレースを持つペトリネットを状態機械（state machine）という
- トークンの数が一個のとき順序機械



マーク付きグラフ

- どのプレースもただ一つの入力トランジションとただ一つの出力トランジションを持つペトリネットをマーク付きグラフ（marked graph）という

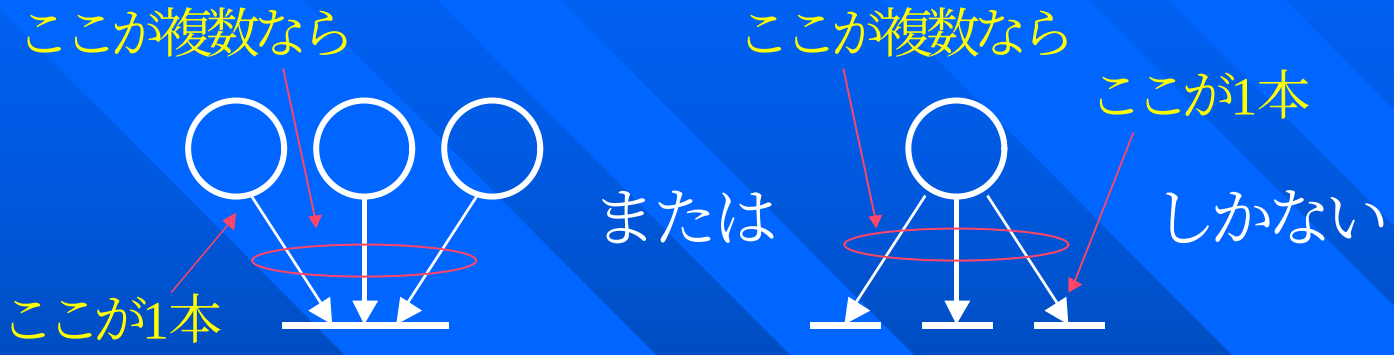


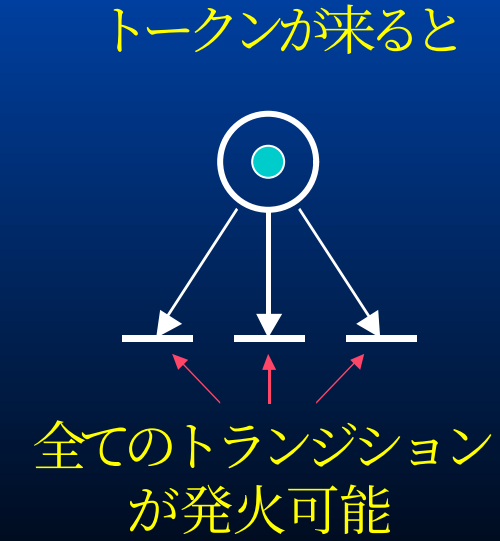
生産者ー消費者問題

双対な関係

- 状態機械とマーク付きグラフは
  プレースとトランジションに関して双対
- 状態機械
  - 並列動作→NG
  - 条件分岐→OK
- マーク付きグラフ
  - 並列動作→OK
  - 条件分岐→NG

自由選択ネット(1)

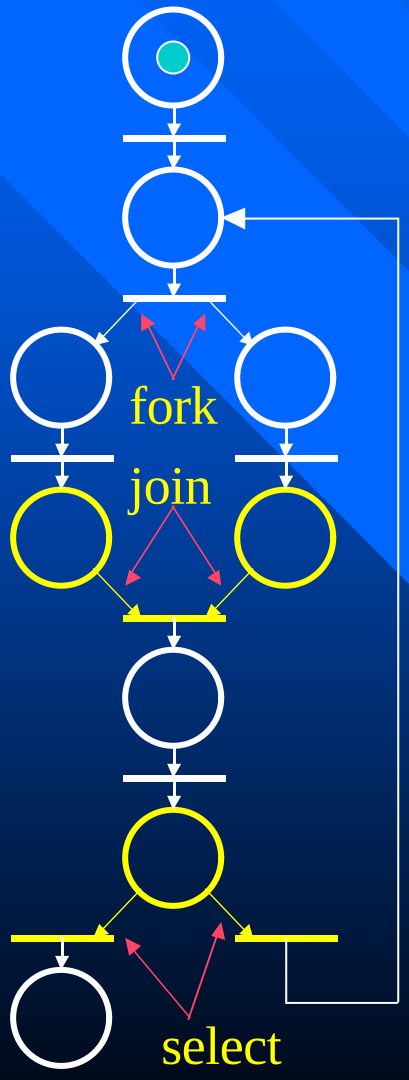


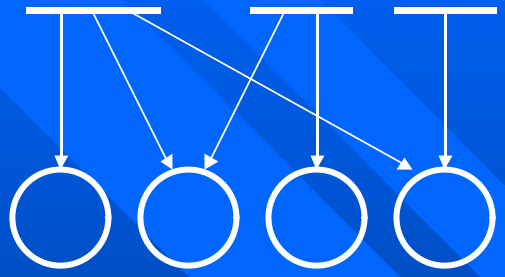


- 任意の一つを自由に選択して発火させることができるので自由選択ネット (free choice net)
- 状態機械とマーク付きグラフを部分クラスとして含む

自由選択ネット(2)

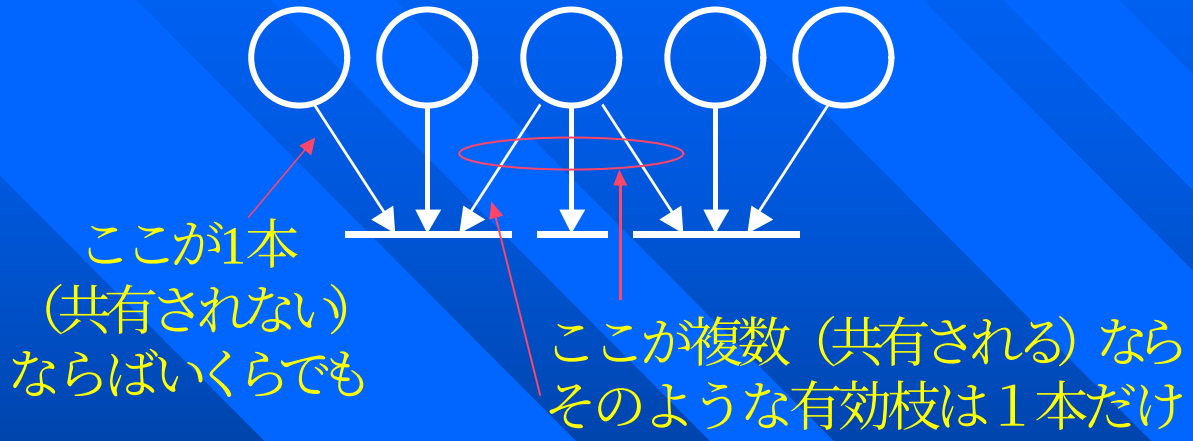
- 例





- トランジションからプレースへの有向枝に関しては何の制約もない
- 並列動作と条件分岐の両方を表現可能

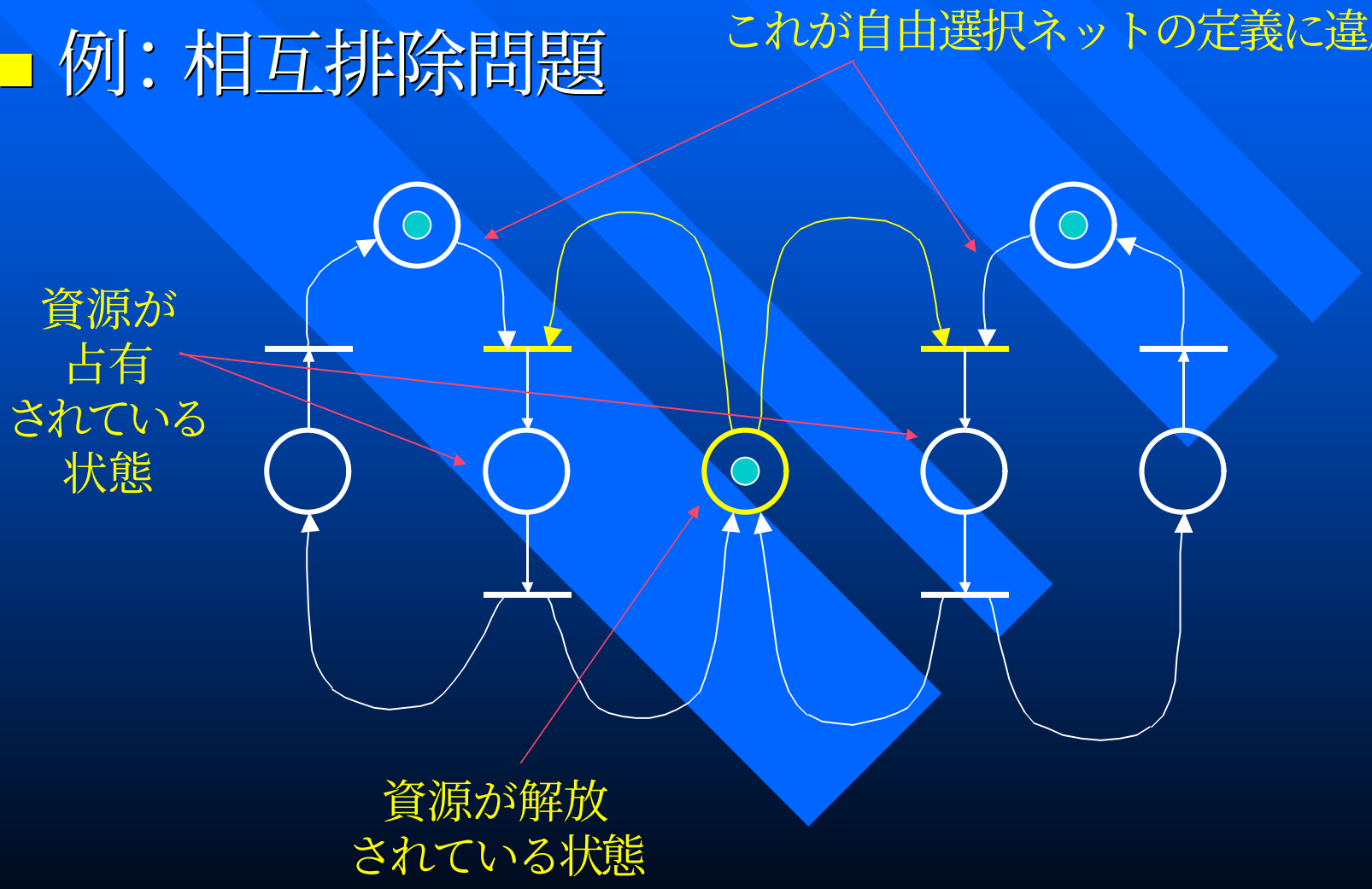
単純ペトリネット(1)



- どのトランジションに関しても，他のトランジションと共有する入力プレースはたかだか一つであるようなペトリネットを単純ペトリネット（simple Petri net）という.
- 自由選択ネットを含む

単純ペトリネット(2)

- 例：相互排除問題



部分クラスの包含関係

マーク付きグラフの経路，閉路

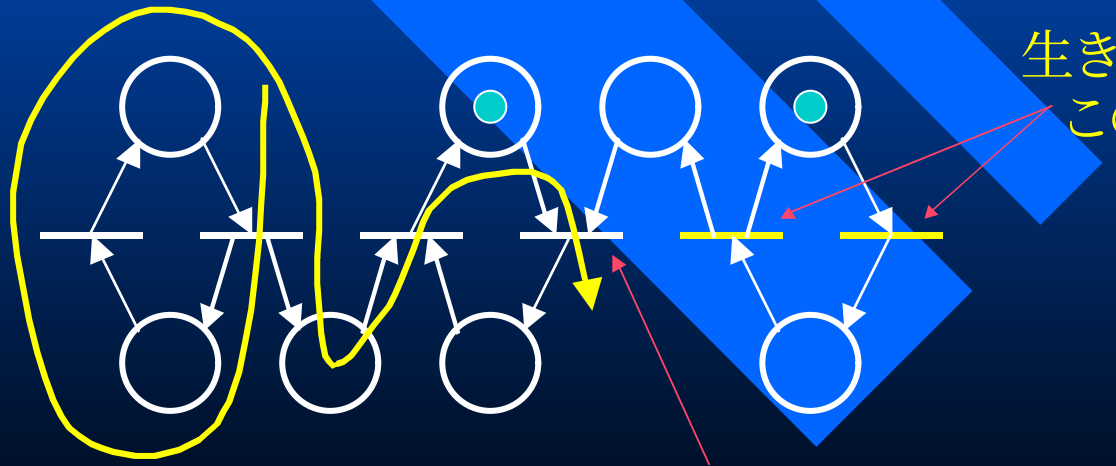
マーク付きグラフでは
出入りが１本づつなので

を考えることができる

- 閉路上のトークンの数は不変

生きている条件

- マーク付きグラフにおいて，あるトランジションが生きているための必要十分条件は，そのトランジションが，トークンの存在しない閉路にも，トークンの存在しない閉路から出る経路にも，含まれていないことである.



発火の回数

- マーク付きグラフの生きていないトランジションが発火できる回数は，トークンの存在しない閉路からそのトランジションへ至る経路上のトークンの数，複数あるときは最小値，に等しい．

到達可能問題

- 初期マーキングMから出発して，任意に与えられたマーキングM'に到達できるかを判定する問題，換言すればM'が$R(M)$の要素かを判定する問題を到達可能問題という.

被覆可能問題(1)

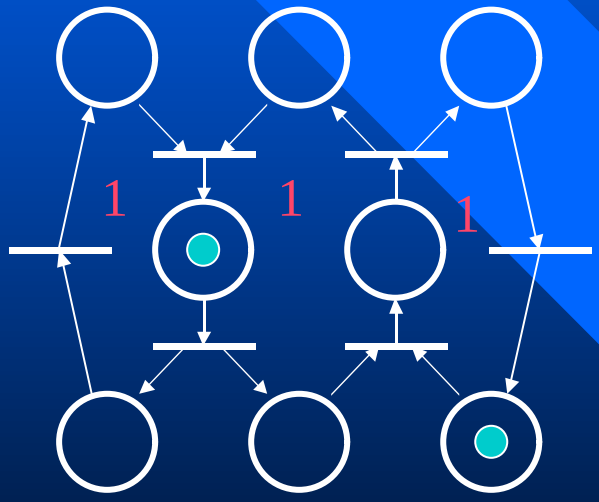
- また，初期マーキングMから出発して，任意に与えられたマーキングM'を被覆するマーキングM''に到達できるかを判定する問題を被覆可能問題という.

被覆可能問題(2)

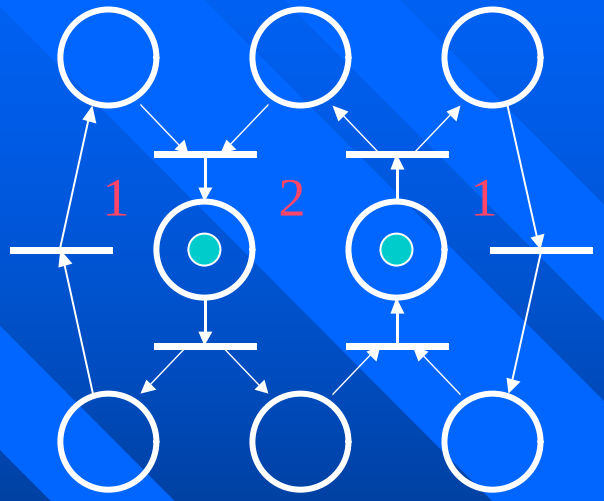
- 生きているマーク付きグラフに於いて，与えられたマーキングM'を被覆するマーキングM''へ初期マーキングから到達可能であるための必要十分条件は，Mにおいて各閉路上に存在するトークンの数が，それぞれ，M'において存在するトークンの数と同じかそれ以上になっていることである.

被覆可能問題(3)

- 例

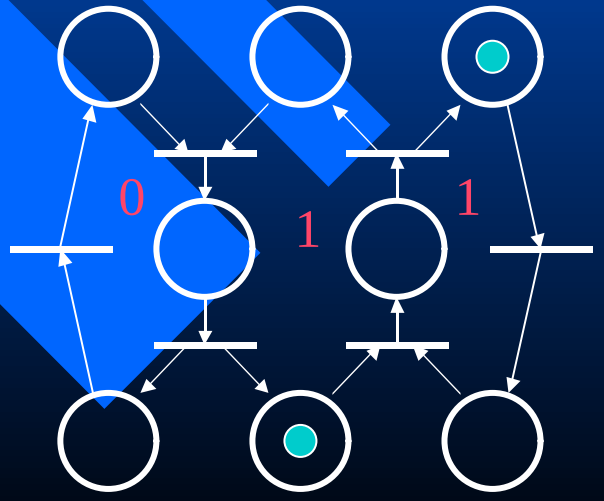


M



被覆可能ではない

M_1



被覆可能

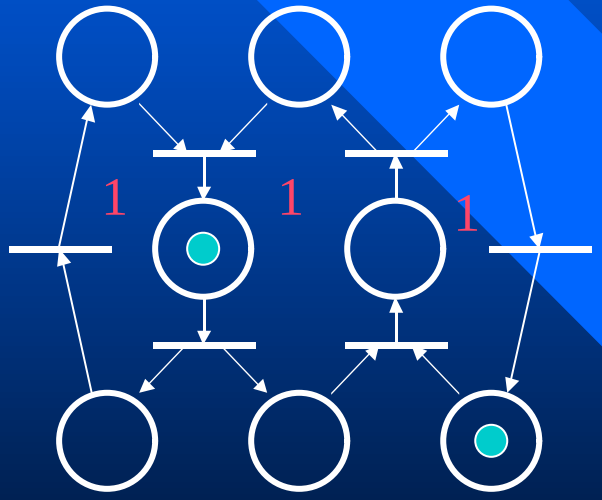
M_2

到達可能性の条件(1)

- 生きているマーク付きグラフと与えられたマーキングM'に対して，初期マーキングMからM'へ到達可能であるための必要十分条件は，各閉路上のトークンの数が，それぞれ，MとM'において同数となっていることである.

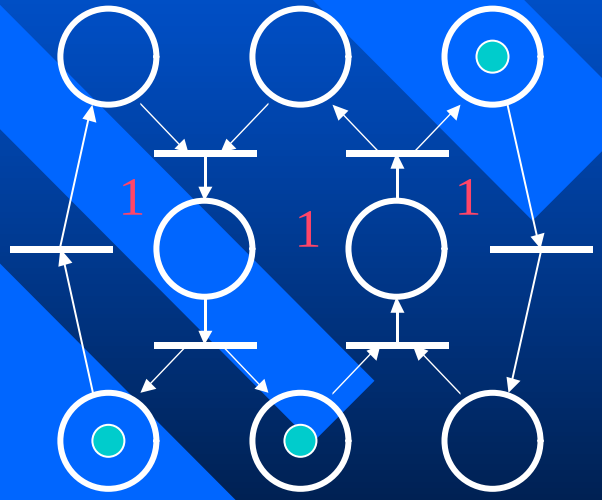
到達可能性の条件(1)

- 例



M

到達可能



M_3

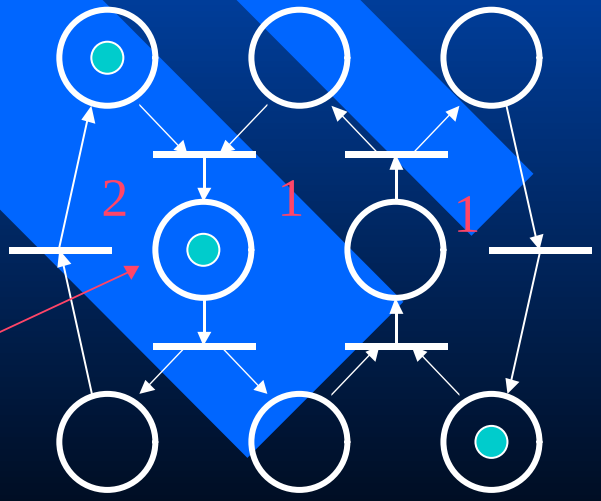
到達可能性

- 生きているマークつきグラフがMからM'へ到達可能ならばM'からMへ到達可能である.

相互に到達可能

安全性の条件

- 生きているマークつきグラフにおいて，あるプレースが安全であるための必要十分条件は，トークンが１つだけ存在する少なくとも一つの閉路にそのプレースが含まれていることである.

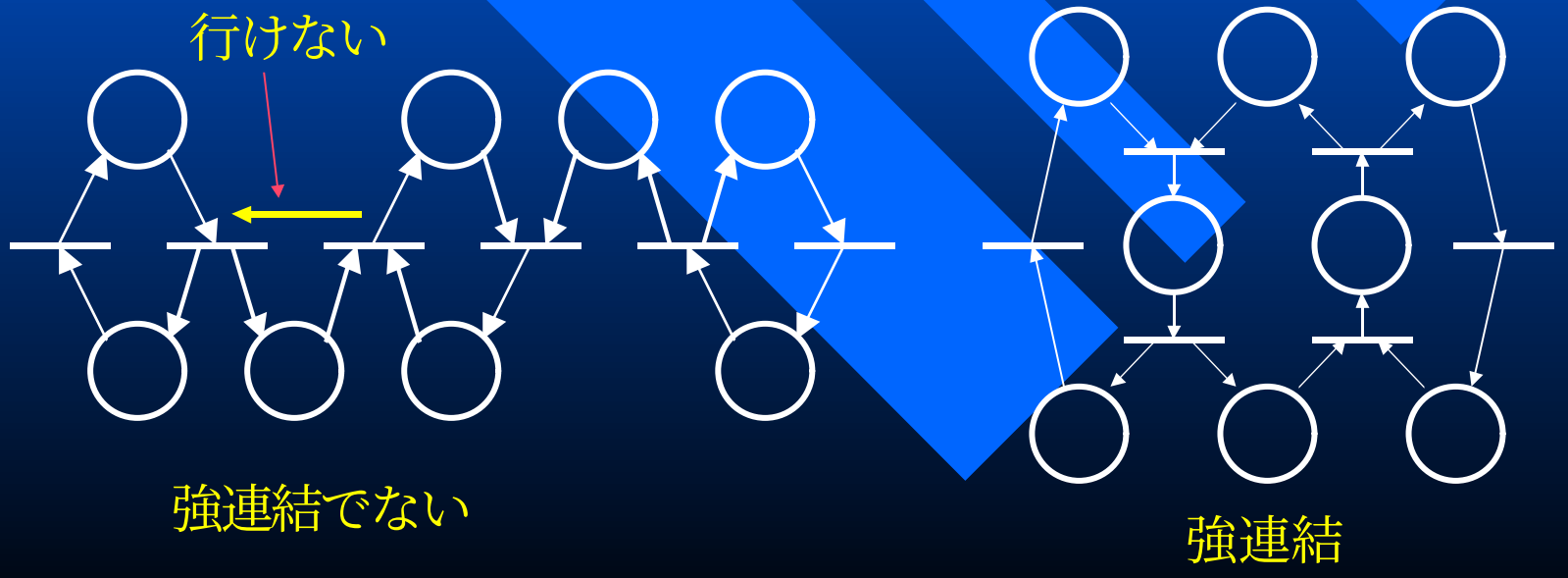


生きていて安全なマーキング

- マーク付きグラフが強連結ならば，生きていて安全であるような初期マーキングが可能である.
- どの閉路にも1個のトークンを置く
- 生きている
- 安全

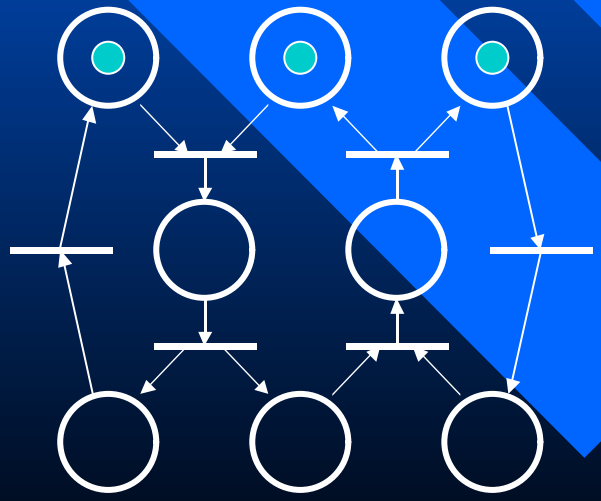
強連結

- プレースとトランジションを節点，有向枝を有向枝とする有向グラフに於いて任意の2つの節点間に経路が存在すること.



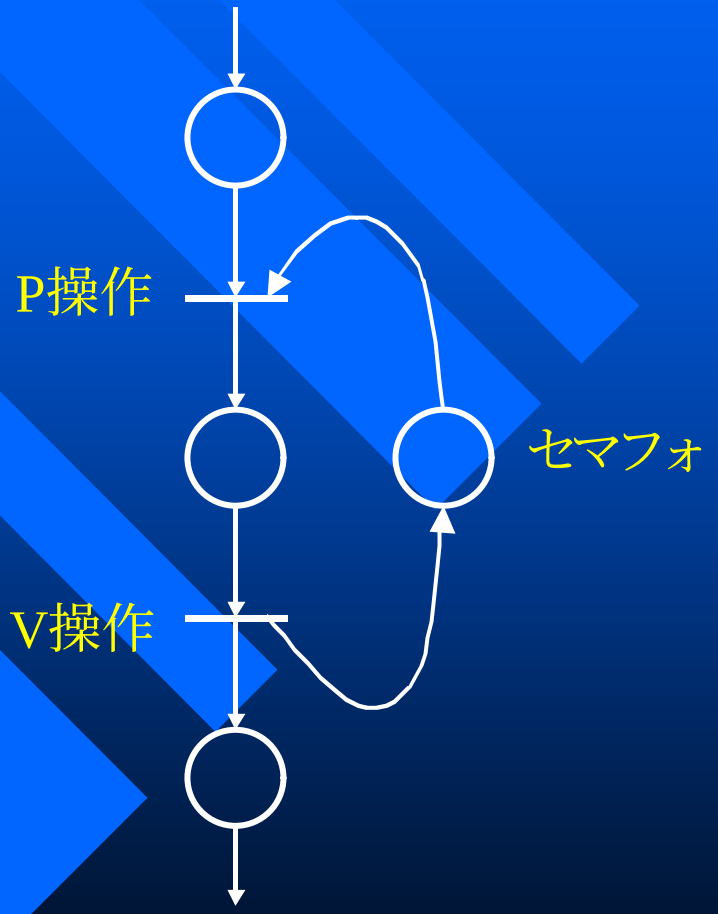
強連結ならば閉路のみに

- 強連結ならば閉路のみに分解できる.
- そのすべての閉路に1個のトークンを置くと,
- 生きていて安全となる



Dijkstraのセマフォ

- セマフォ
  - 0,1,2,...の値をとるデータ
- V操作
  - セマフォの値を+1する
- P操作
  - セマフォの値が正であれば，-1する
  - 0であれば，V操作を待つ
- セマフォでは表現できない同期問題が存在



喫煙者の問題

ペトリネットの限界

- 有界でないときトークンが存在しないことを判定できない.
- not演算が無い
- 抑止枝をもったペトリネット

確率ペトリネット

- 発火可能となってから発火するまでの時間を指数分布とするペトリネット

モデリング

- 情報システムのモデリングを誰が行うか
  - 昔は開発者
    - » どうしても開発の手間を考えてしまう
  - 利用者と開発者の間にモデリングを行う専門家（アナリスト）を配置
- モデリングはWhatを明確にすること
  - Howではない
- データフローモデル
  - T. DeMarcoが事務処理システムのために提案

データフローモデル

- DFD: Data Flow Diagramにより表現
- CASE: Computer Assisted Software Engineeringに使用される
- どのような情報がどのように流れているか
- 情報をどのように加工するかを表現する
- 階層設計をサポート
  - トップダウン: 全体から部分へ
  - ボトムアップ: 部分から全体へ

DFDによる記述

- 一枚のDFDはデータフローのある階層の表現
- DFDは以下の4つのものだけからなる

  →：データフロー，定常的な情報の流れ

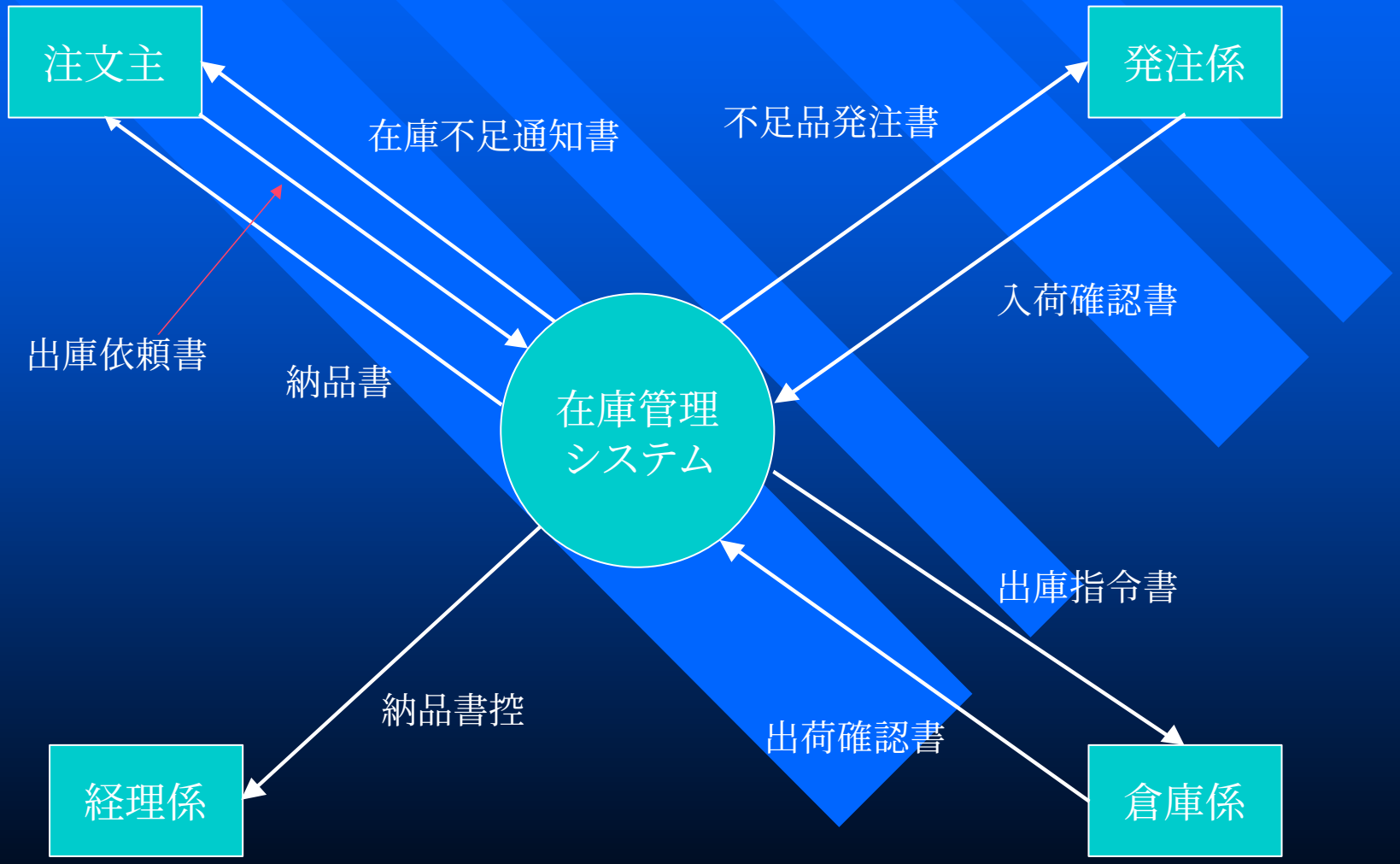
  ○：プロセス，情報の加工

  □：エンティティ，開発しているシステムの外で情報を生成し，吸収する組織，人

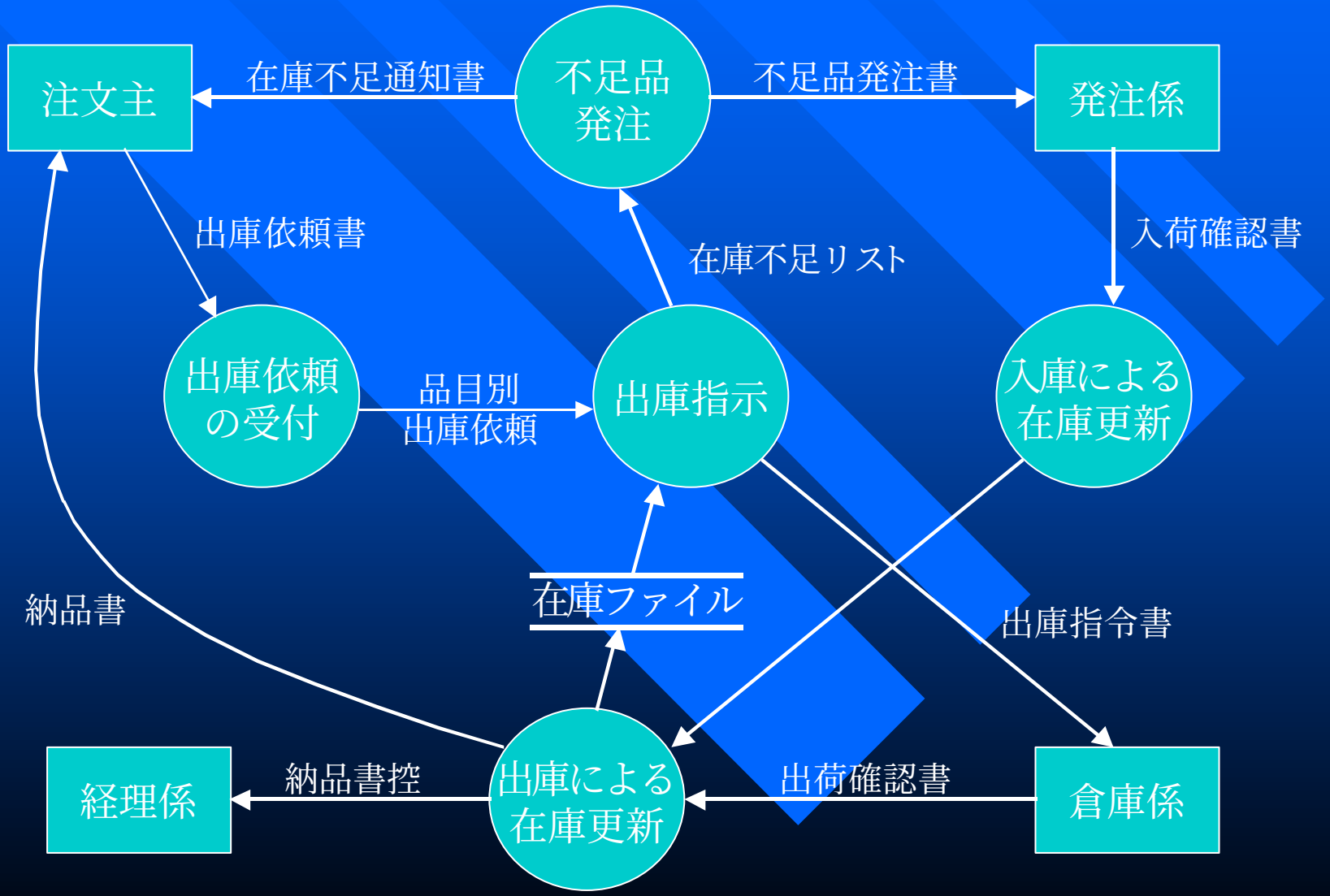
  ＝：ファイル，情報の原本

TOPのDFD

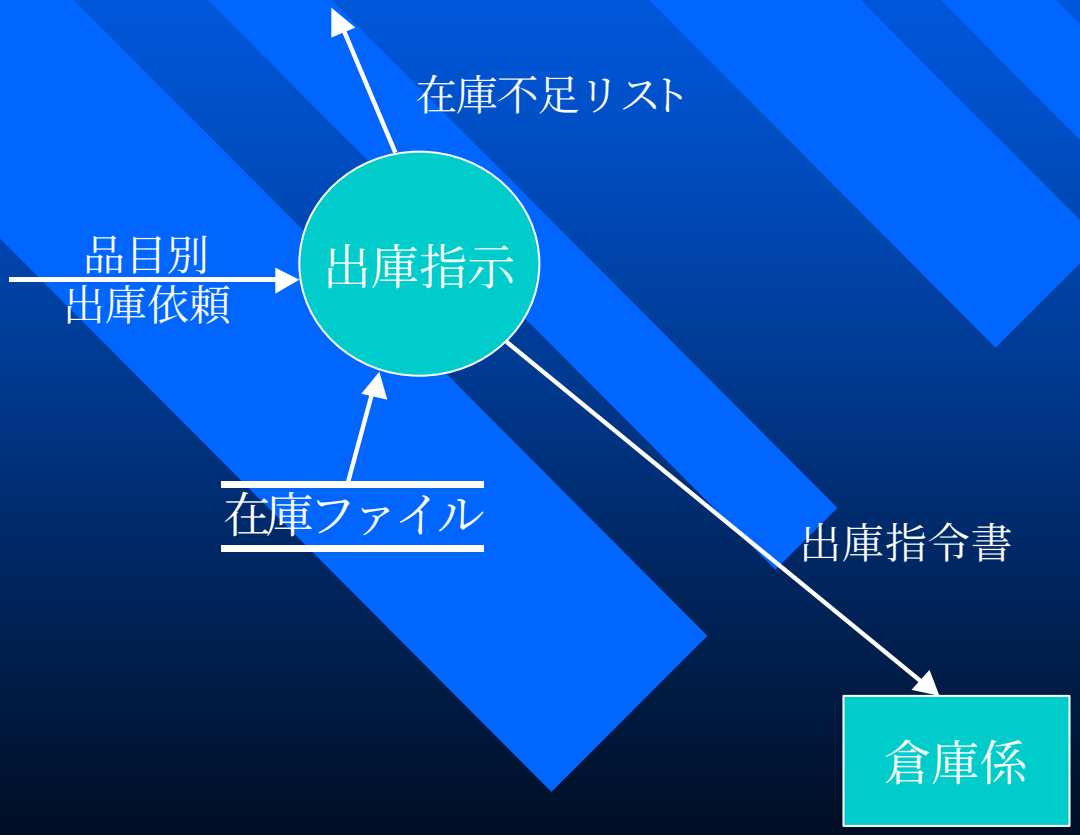
(在庫管理システムに関する全体文脈図)



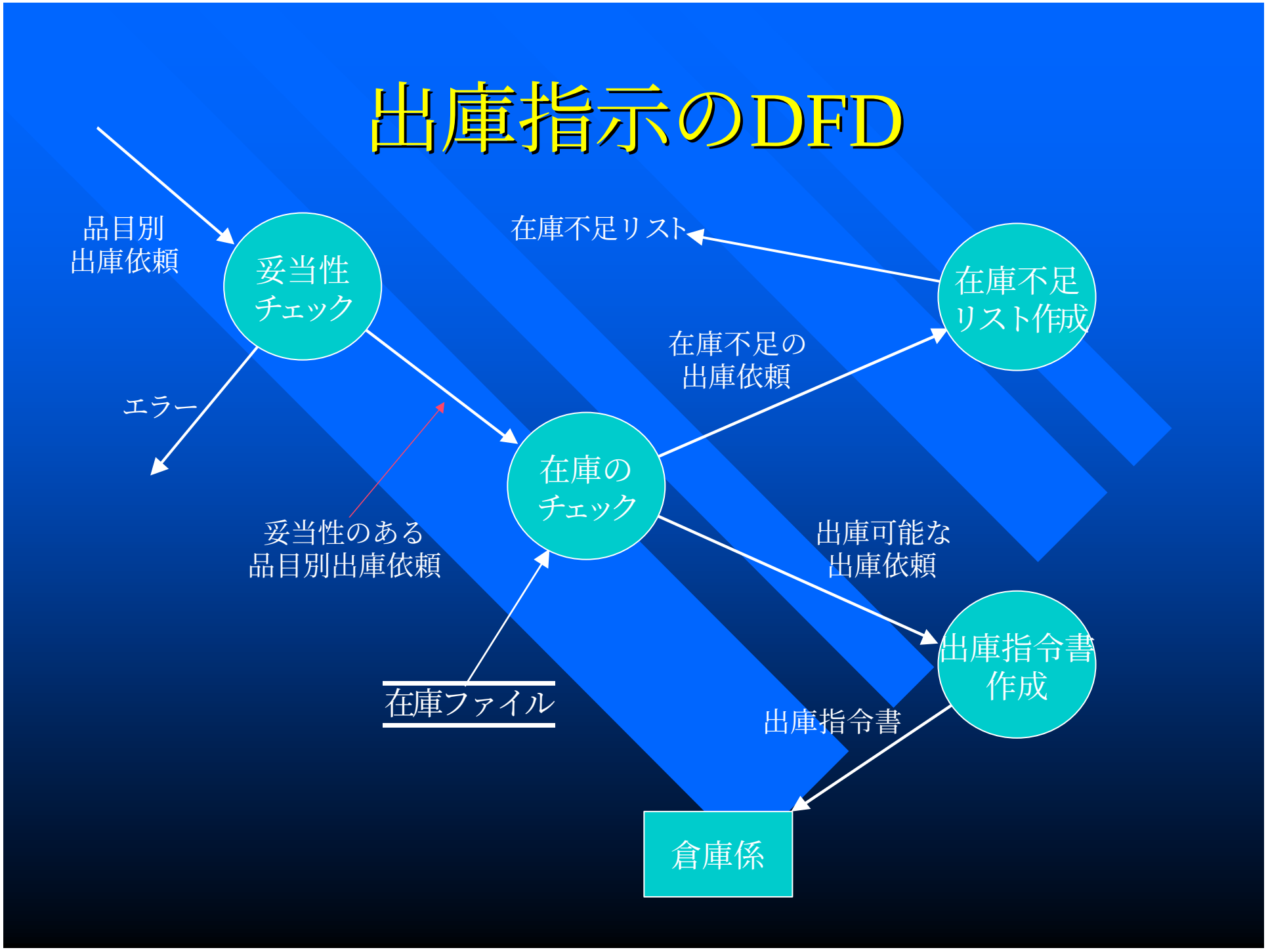
在庫管理システムのDFD



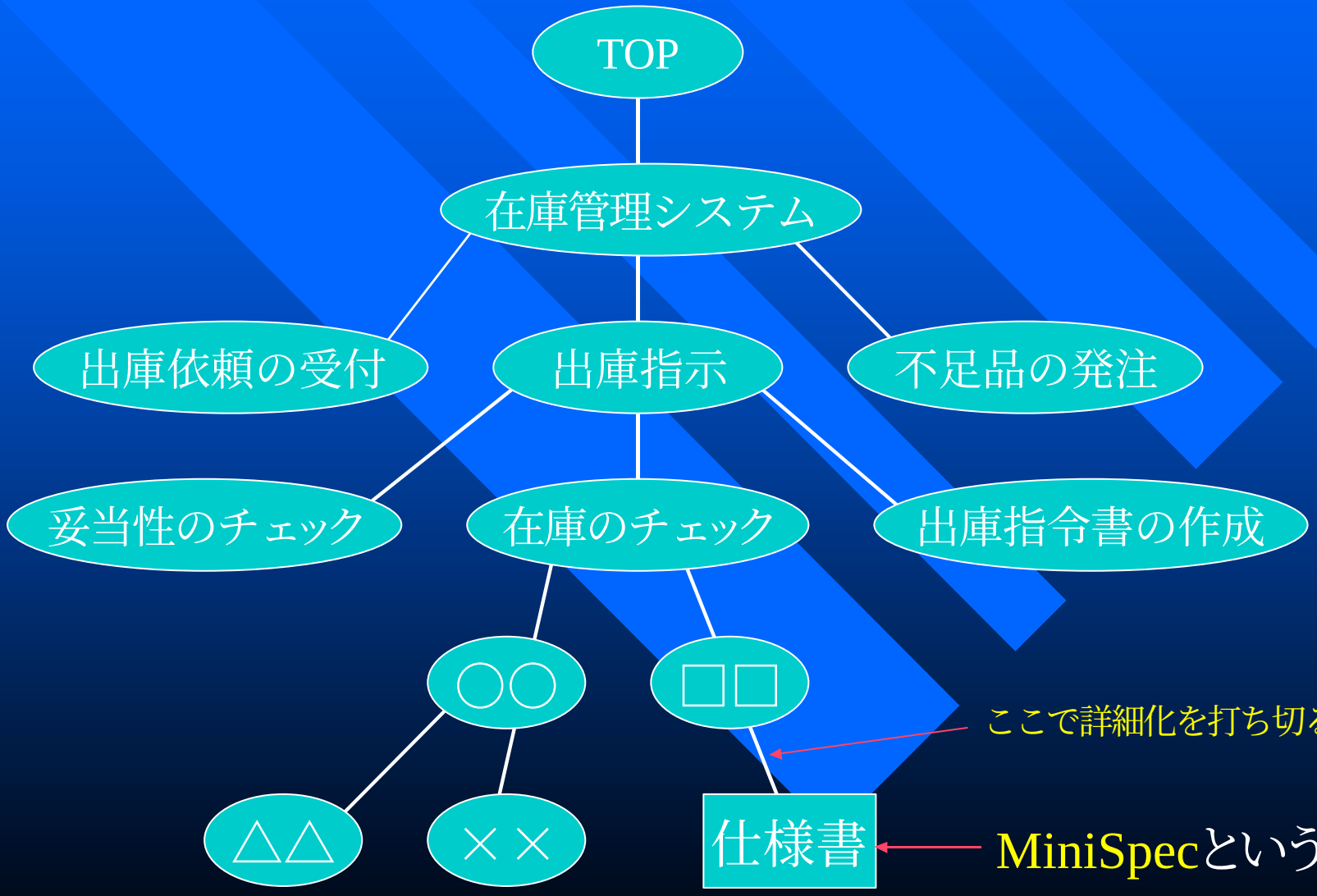
出庫指示だけに着目



出庫指示のDFD


品目別
出庫依頼
妥当性
チェック
エラー
在庫不足リスト
在庫不足
リスト作成
在庫不足の
出庫依頼
在庫の
チェック
妥当性のある
品目別出庫依頼
出庫可能な
出庫依頼
出庫指令書
作成
在庫ファイル
出庫指令書
倉庫係

階層構造


TOP
在庫管理システム
出庫依頼の受付
出庫指示
不足品の発注
妥当性のチェック
在庫のチェック
出庫指令書の作成
○○
□□
△△
××
仕様書
ここで詳細化を打ち切る
MiniSpecという

モデリングのアウトプット

- DFD
- MiniSpec
- DD(Data Dictionary)
  - データフロー毎にデータの形式と内容を定義
  - 記法（正規表現と同じ）

    (a|b): aまたはb

    a+b: aに続けてb

    {a}: aを0回以上繰り返し

    [a]: aを省略化

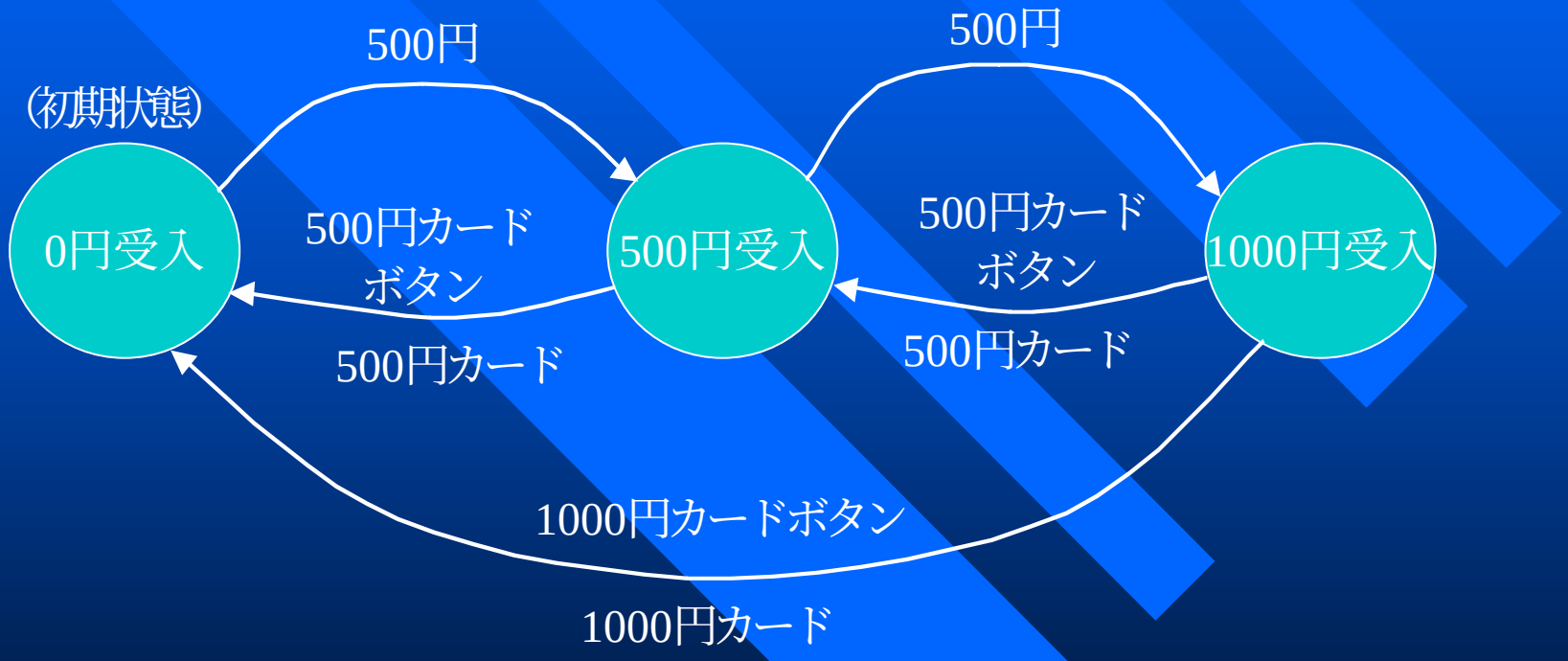
コントロールフローモデル

- D. J. HatleyがDFDにCFD(Control flow Diagramを追加
  - DFD⇔CFD
  - MiniSpec⇔ControlSpec
  - DD⇔TD (Timing Dictionary)
- 詳細化しすぎると，Petri netなどと変わらなくなってしまう
- 利用者と開発者にとって分かりやすいことが第一

状態遷移モデル

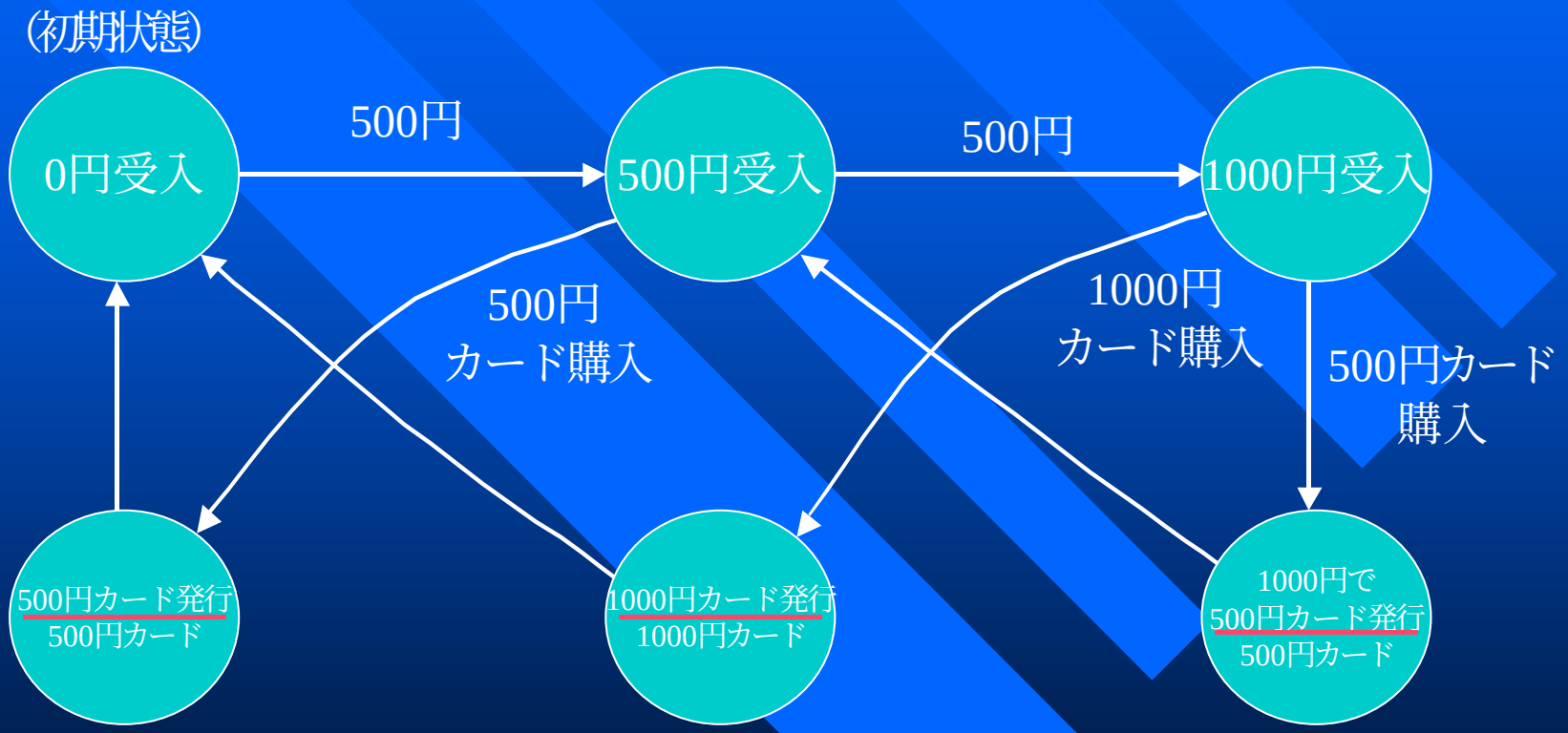
- 状態遷移図(State Transition Diagram)
  - DFDが定常的な情報の流れを記述したのに対し，状態遷移図は情報を操作する動作を記述する.
- 状態を表す○または□と遷移を表す→からなる.
- DFDと区別する必要があるときは

Mealey型状態遷移図



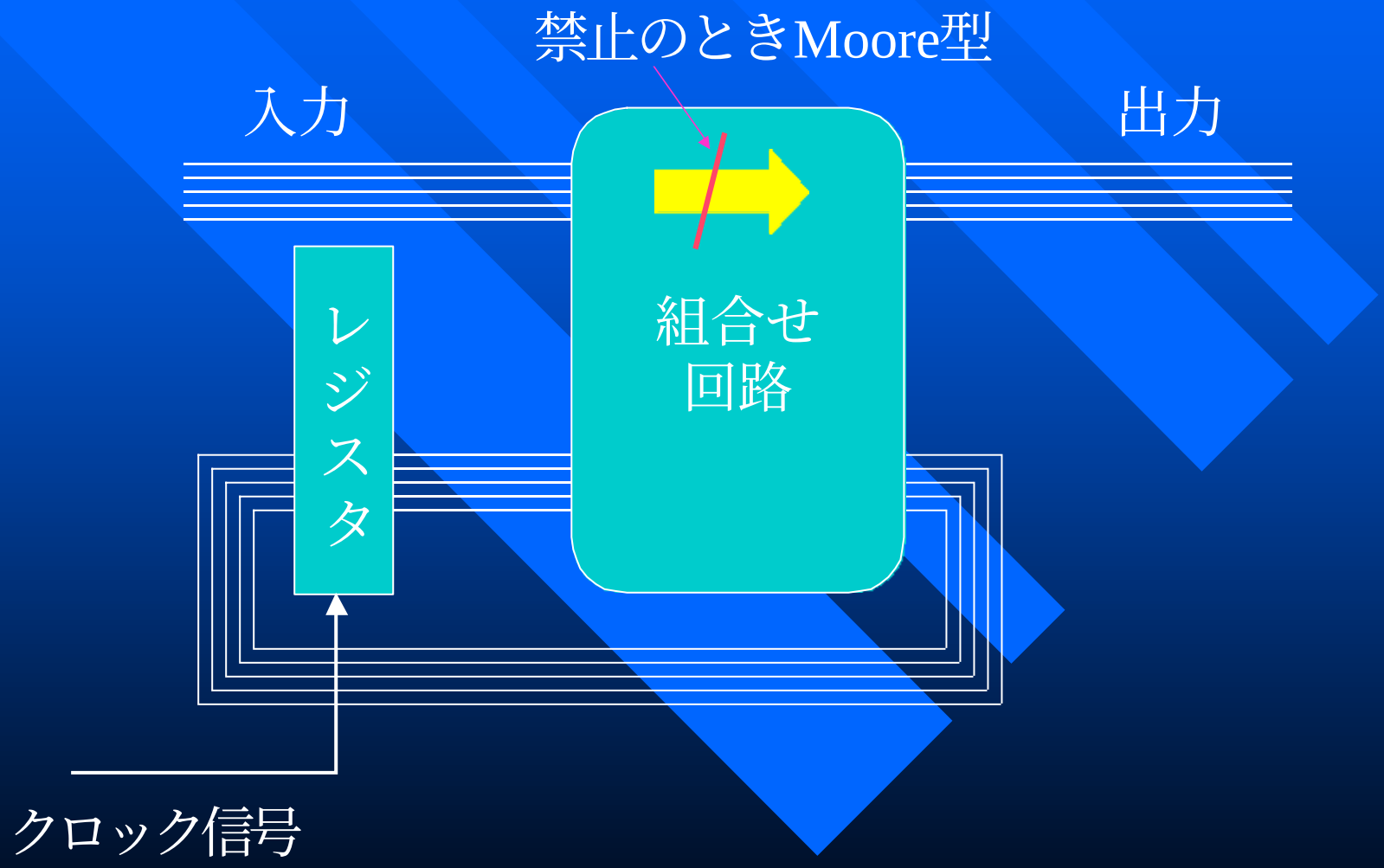
自動販売機の状態遷移図

Moore型状態遷移図



自動販売機の状態遷移図

順序回路

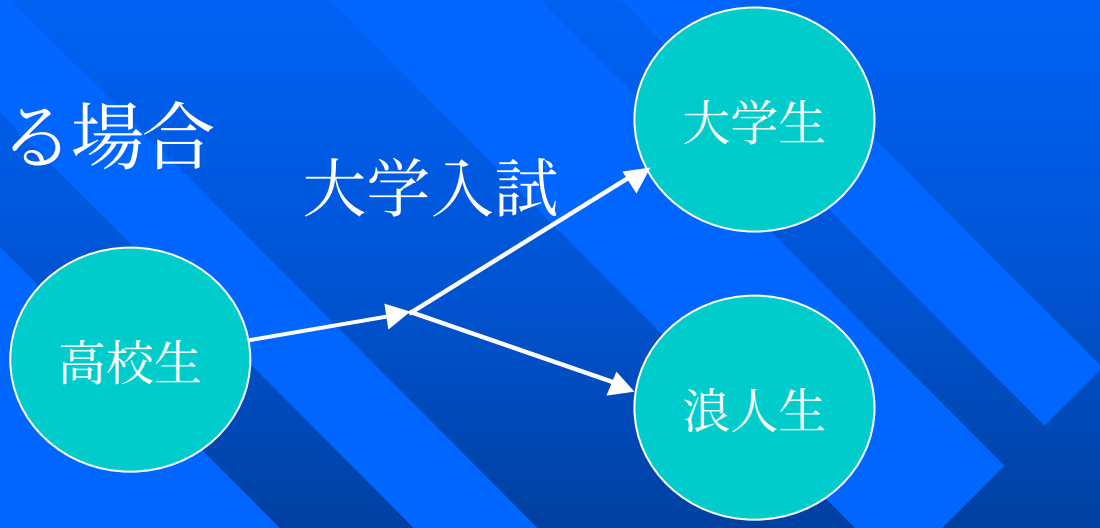


対称世界の認識

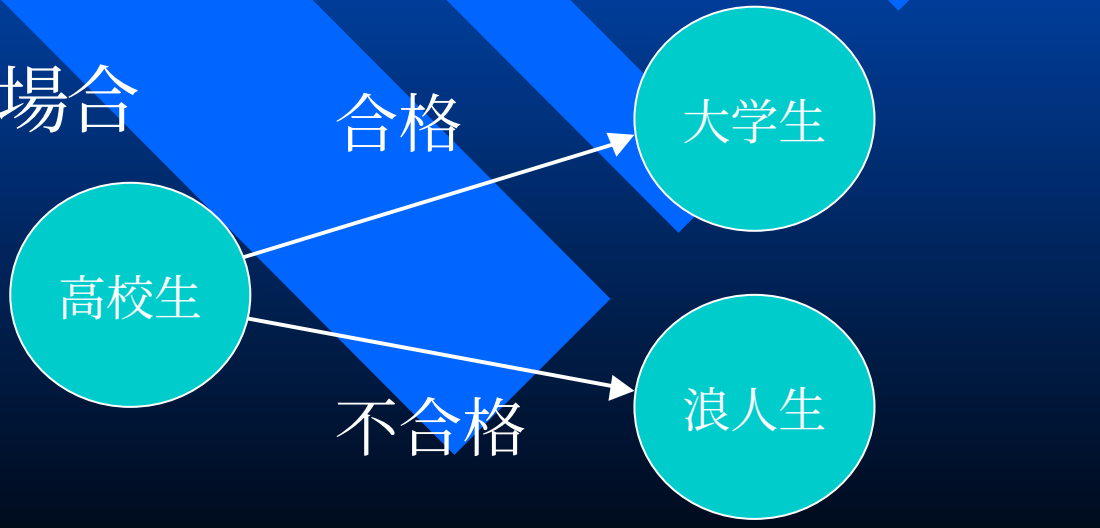
- 状態と状態遷移のきっかけを与える事象を認識し，抽出する作業が必要
- この状態と事象は互いに関連しており（双対性をもつという），何をどう選ぶかは経験によるしかない
- 状態：一定時間持続するもの
- 事象：ごく短い時間に生起して消滅するもの
- 非決定性：同時に複数の状態遷移を許す場合，最初は気にしなくても良い

何を選ぶかということ

- 非決定性となる場合



- 決定性となる場合



形式言語

- 記号で表される対象をどう扱うか
- 語と形式言語の定義
- 言語の演算，連接，閉包
- 言語の生成と受理
- プロダクションによる生成
- 形式文法
- 受理系の構造

記号，アルファベット

- 記号(symbol)，文字
  - a,b,c,...
  - これ以上説明の必要がないものとする
- アルファベット(alphabet)
  - 記号の集合
  - 有限集合
  - 記号0,1からなるアルファベットは{0,1}である
  - アルファベット→ Σ で表現

語

- 語，記号系列，文
  - アルファベットに属する記号を重複を許して有限個並べたもの
  - 01101は{0,1}上の語である
  - 単一の記号からなる語もある
  - 記号→ a で表現
  - 語→$\boldsymbol{a}$で表現
  - 同じ記号aがi個続いている語→ a^i で表現

空語，連接

- 空語，空系列
  - 記号の個数が0個である語
  - 空語→ ε で表現
  - ε はアルファベットの要素ではない
- 連接(concatenation)
  - 語x，yが語であるときxyをxとyの連接という
  - $x=abc$，$y=ca$のとき，$xy=abcca$である
  - すべての語x に対して，$x\varepsilon = \varepsilon x = x$とする

逆，接頭語，接尾語，部分語

- 逆(reverse)
  - 語を構成する記号を逆順に並べたもの
  - 語xの逆→x^Rで表現
  - $\varepsilon^R = \varepsilon$とする
- 接頭語，接尾語，部分語
  - x, y, zが同じアルファベット上の三つの語であるとき，x, y, zはそれぞれ語xyzの接頭語，部分語，接尾語である
  - εは任意の語の接頭語，接尾語，部分語である

語の長さ

- 語の長さ
  - 語に含まれる記号の数
  - 語の長さ→ $|x|$ で表現
  - $|\varepsilon|=0$
  - $|abc|=3$

言語, Σ^*, Σ^+

- 言語，形式言語
  - 与えられたアルファベット上の語の任意の集合
  - 空集合φは言語である
  - {ε}は空集合ではない
- Σ^*
  - アルファベットΣ上のすべての語を含む集合
  - $\Sigma=\{a,b\}$として，$\Sigma^*=\{\varepsilon,a,b,aa,bb,ab,ba,aaa,\ldots\}$
  - Σ上の任意の言語はΣ^*の部分集合
- Σ^+
  - Σ^*からεを除いたもの

言語の演算

- 集合演算
  - 言語は集合なので当然，和，積，補集合の演算が定義できる
- 連接演算
  - L_1をアルファベットΣ_1上の言語，L_2をアルファベットΣ_2上の言語とするとき，集合

    $\{xy \mid x \in L_1 \wedge y \in L_2\}$

    を，言語L_1，L_2の連接といって，L_1L_2で表す
  - ここでxyは語の連接

閉包(1)

- 閉包 L^*(closure) の再帰的定義
  - L を与えられた言語とする
  - $L^0=\{ \varepsilon \}$
  - $n \geqq 1$である任意のnに対して$L^n=LL^{n-1}$
  - Lの閉包 L^*とは$\cup$(nが0から∞まで) L^n

閉包(2)

- L^*は，Lに属する任意の語を任意個選んで，それらを任意の順に並べて得られる語のすべての集合である
- 仮にLにεが含まれなくともL^*にはεが含まれる
- L^*からεを除いたものをLの正の閉包L^+という

言語の生成と受理

- 言語は無限集合
  - アルファベットは有限でも言語は無限集合
  - 有限集合であれば要素をすべて列挙して表現
  - 無限集合では生成系や受理系により表現
- 生成系
  - ある言語に属する語のみを生成し，しかもその言語を覆いつくす能力のある有限のもの
- 受理系
  - ある言語に属する語のみ可と認識する有限のもの

生成とは書換え

- 著作
  - 想起された概念を下位概念に書換え続けて一つの叙述を得る
- 文の訂正
  - 語の書換え
- 数学
  - 公理を書換えて定理を得る

書換え規則(rewriting rule)

- $U\alpha W \rightarrow U\beta W$
  - U, Wはεをも含めて任意の語
  - α，βは特定の語
  - αがεであってはならない
    - » 勝手な位置に部分語を挿入するこは禁止
    - » 逆にβがεであること(特定の部分語を削除)はOK

書換え規則の意味

- $U\alpha W \rightarrow U\beta W$
  - 任意の語の中にもし特定の語αがあればその部分だけ特定の語βに書き換える
  - 曖昧さがなければ，$\alpha \rightarrow \beta$と書く
  - プロダクションとも言う
- 書換え規則(の集合)
  - 記号系列(語)の集合，すなわち言語上の関係

Chomskyの句構造文法

- 語の書換えを行う体系，文法
- 文法に用いるアルファベットの定義
- 書き換え規則の定義
- 文法の定義
- 文法の分類

Chomsky文法のアルファベット

- V_N: 非終端記号，V_T: 終端記号
  - アルファベットは互いに素な二つの部分集合 V_N，V_Tからなる
  - 終端記号は定数，非終端記号は変数
  - 終端記号はa,b,...，非終端記号はA,B,...で表す
  - 非終端記号は自然言語の文，句，節などのような高位の抽象概念に対応

Chomsky文法のプロダクション

- $\alpha \rightarrow \beta$，書換え規則，プロダクション
  - αは$(V_N \cup V_T)^* V_N (V_N \cup V_T)^*$の要素である
    - » $(V_N \cup V_T)^*$は全ての語の集合，言語
    - » $(V_N \cup V_T)^* V_N (V_N \cup V_T)^*$は言語の連接
  - βは$(V_N \cup V_T)^*$の要素である
  - プロダクション(の集合)はこれらの集合間の関係
  - すなわちこれらの集合の直積の部分集合
  - βにはεが含まれるので$\alpha \rightarrow \varepsilon$が有り得る

Chomsky文法

- 文法Gは，4項組によって定義される

$$G = (V_N, V_T, P, S)$$

  - V_N は非終端記号の有限集合
  - V_T は終端記号の有限集合
  - P は書換え規則の有限集合
    - » $V = (V_N \cup V_T)$ としてPは直積$V^* V_N V^* \times V^*$の有限部分集合
  - 開始記号S はV_Nの特別な要素

例

- $V_N=\{S\}$

  $V_T=\{0,1\}$

  $P=\{S\rightarrow 01S,\ S\rightarrow 1\}$

  Sは開始記号
- この文法による文の生成

  $S\rightarrow 01S$を3回適用

  $S\rightarrow 1$を1回適用

  0101011が生成される

文法から生成される言語

- 文形式
  - 文法$G = (V_N, V_T, P, S)$に対して
  - Sは文形式である
  - もし$u\,\alpha\,w$が文形式で，かつ，書換え規則 $\alpha \rightarrow \beta$がPにあるとき，$u\,\beta\,w$は文形式である
  - u，wとは，$V = (V_N \cup V_T)$として，V^*の要素
- 文法Gによって生成される言語$L(G)$
  - 文法Gの文形式のうち語であるものの集合
  - 語とは，非終端記号を含まない文形式

導出

- $\alpha \Rightarrow_G \beta$：導出
  - ある文形式$u\,\alpha\,w$が文法Gのプロダクション$\alpha \rightarrow \beta$によって$u\,\beta\,w$に書き換えられること
- 導出の反射的，推移的閉包
  - $a \Rightarrow_{G^*} a_m$：$a \Rightarrow_G a_1, a_1 \Rightarrow_G a_2, \ldots, a_{m-1} \Rightarrow_G a_m$
  - $a \Rightarrow_{G^*} a$：これは定義
- $\Rightarrow_{Gk}$：導出の回数kを指定
- $\Rightarrow$：どの文法を使うかを指定しない

簡潔な記法

- $L(G)=\{x|x\in V_T^{*}$かつ$S\Rightarrow_{G^*}x\}$
  - 文法Gによって生成される言語$L(G)$
- $\alpha\rightarrow\beta_1|\beta_2|\ldots|\beta_n$
  - 記号系列αに対するプロダクションが$\alpha\rightarrow\beta_1$, $\alpha\rightarrow\beta_2$, …, $\alpha\rightarrow\beta_n$のように複数個存在すること
- $L(G)=\{(01)^n1|n\geqq 0\}$
  - 先の例が生成する言語

文法の階層(1)

- 3型文法
  - 正規文法，右線形文法，左線形文法
  - 正規言語，3型言語
- 2型文法
  - 文脈自由文法
  - 文脈自由言語，2型言語
- 1型文法
  - 文脈規定文法
  - 文脈規定言語，1型言語

文法の階層(2)

- 0型文法
  - 無制限文法，句構造文法
  - 無制限言語，句構造言語，0型言語
- Chomskyの階層
  - 上記の4つの文法と言語のクラス分け
- 機能的集合
  - 要素の帰属問題が機能的に可解
- 機能的に可算な集合
  - 要素の帰属問題が機能的に可解でない

用語

- 手続き
  - 基本操作の適用方法を決めたもの
- アルゴリズム
  - 必ず終了する手続き
- 機能的集合
  - 要素の帰属を決定するアルゴリズムが存在する集合
- 機能的に加算な集合
  - 要素を数え上げる手続きが存在する集合
  - 機能的集合でないものがある

3型文法

- A,B は非終端記号，xは終端記号の系列
- $A \rightarrow xB$，$A \rightarrow x$
  - 右線形文法
- $A \rightarrow Bx$，$A \rightarrow x$
  - 左線形文法
- 正規文法，正規言語，3型言語
- 書換え規則の制限が最も強い

右線形文法の例

- $V_N=\{S\}$
  $V_T=\{0,1\}$
  $P=\{S\rightarrow 01S,\ S\rightarrow 1\}$
  Sは開始記号
- この文法による文の生成
  $S\rightarrow 01S$を3回適用
  $S\rightarrow 1$を1回適用
  0101011が生成される
- $L(G)=\{(01)^n1 | n\geqq 0\}$

2型文法

- A は非終端記号
- α は記号の系列 (任意の語)
- $A \rightarrow \alpha$
  - Aの前後に関係なくαに書換える
- 文脈自由文法，文脈自由言語，2型言語
- 3型文法は2型文法の特別の場合
  - 正規言語は文脈自由文法で生成できる
  - 文脈自由言語の中には正規文法で生成できないものがある

文脈自由文法の例(1)

- $V_N=\{S\}$

  $V_T=\{+,*,a\}$

  $P=\{S\rightarrow +SS|*SS|a\}$
- S⇒*SS⇒*S+SS⇒*a+aa
- aは変数，+は加算，*は乗算の演算子

  演算子は後続の2つの変数に作用

  逆ポーランド記法

  日本語はこの逆

文脈自由文法の例(2)

- $V_N=\{A,S\}$

  $V_T=\{0,1\}$

  $P=\{S\rightarrow 0A1,\ A\rightarrow 0A1,\ A\rightarrow \varepsilon\ \}$
- $S\Rightarrow 0A1\Rightarrow 00A11\Rightarrow 000A111\Rightarrow 000111$
- $L(G)=\{0^n1^n|n\geqq 1\}$

1型文法

- $\alpha \rightarrow \beta$ の形は任意であるが $|\alpha| \leqq |\beta|$
  - 記号が置かれる環境に基づいて書換え
  - $\alpha \rightarrow \varepsilon$ は禁止
- 文脈規定文法，文脈規定言語，1型言語
- $\alpha \rightarrow \varepsilon$ がない2型文法は1型文法の特別の場合
- 生成される言語は機能的集合
  - 3,2,1型言語は機能的集合

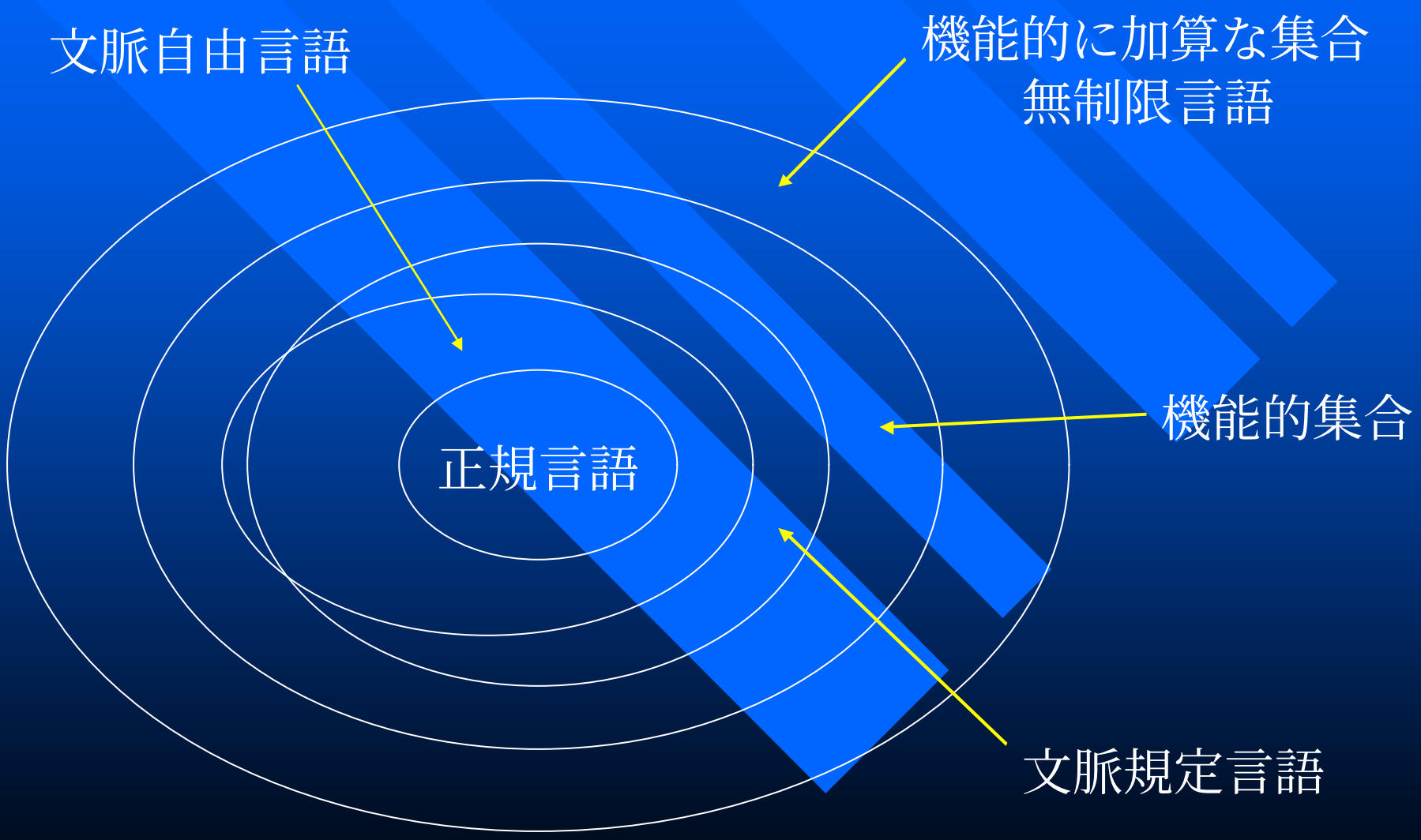
文脈規定文法の例

- $V_N=\{A, B, S\}$
  $V_T=\{a, b, c\}$
  $P=\{S\rightarrow aSAB|abB,\ BA\rightarrow AB,\ bA\rightarrow bb,$
  $bB\rightarrow bc,\ cB\rightarrow cc\}$
- $S\Rightarrow$aSAB$\Rightarrow$aaSABAB$\Rightarrow$aaabBABAB
  $\Rightarrow$aaabBAABB$\Rightarrow$aaabABABB
  $\Rightarrow$aaabAABBB$\Rightarrow$aaabbABBB
  $\Rightarrow$aaabbbBBB$\Rightarrow$aaabbbcBB
  $\Rightarrow$aaabbbccB$\Rightarrow$aaabbbccc
- $L(G)=\{a^n b^n c^n | n \geqq 1\}$

0型文法

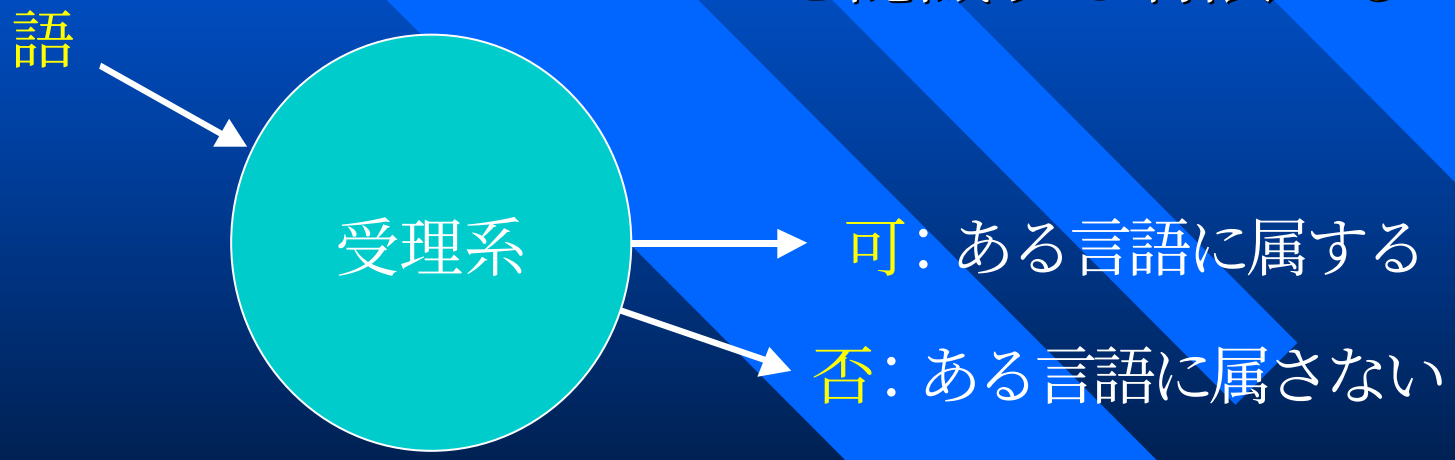
- $\alpha \rightarrow \beta$ の形が任意
  - 記号が置かれる環境に基づいて書換え
  - $\alpha \rightarrow \varepsilon$ もOK
- 無制限文法，句構造文法
- 無制限言語，句構造言語，0型言語
- 3,2,1型文法は0型文法の特別の場合
- 生成される言語は機能的に可算な集合
  - 語の帰属問題が機能的に可解でない無制限言語が存在

文法の階層(3)



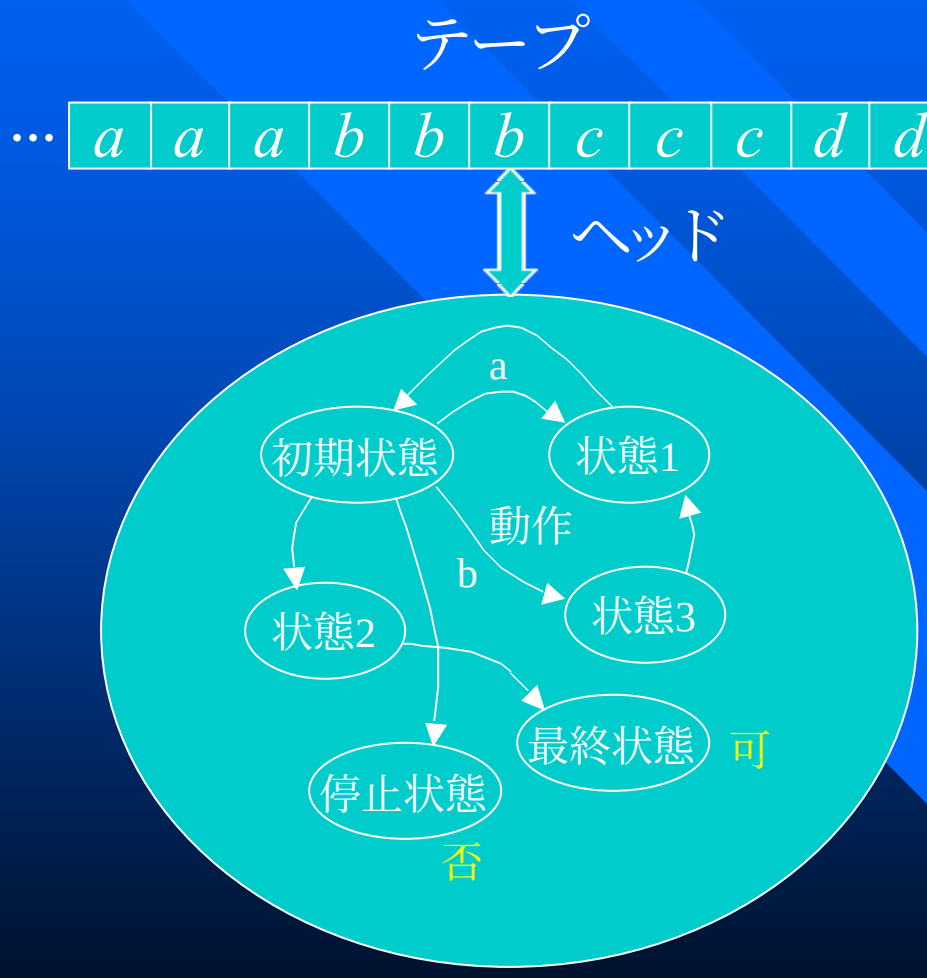
言語の受理

- 受理系
  - ある言語に属する語のみ可と認識する有限のもの



抽象的な機械

受理系の構成



- 語を表記する記号
- 処理に必要な記号
  - これらの記号の数は有限
- 媒体はテープ
  - 長さは必要なだけ
- テープを読んで
  - テープを右に1コマ
  - 左に1コマ
  - 記号の書込み
  - 次の状態へ

  の繰返し

  （動作）
- 動作の無い
  - 停止状態と
  - 最終状態

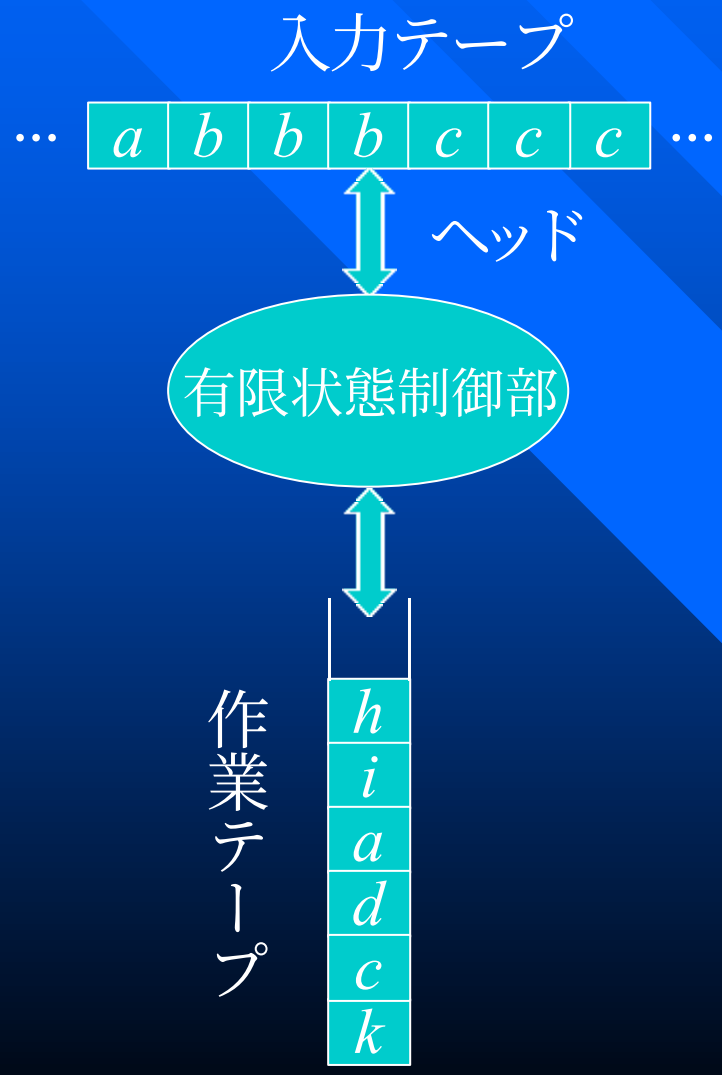
受理系の階層化

- テープの使い方を制約すると
- 有限オートマトン
  - 1方向性有限オートマトン
- プッシュ・ダウン・オートマトン
  - プッシュ・ダウン・スタック
  - Last-In-First-Out(LIFO)
- Turing 機械
  - 線形拘束オートマトン

有限オートマトン

- テープには入力語のみ存在
  - 語の最初の記号から入力を始める
  - 語を読み終えたことが分かる
- テープからの入力に対して
  - 必ずテープを右に1コマ移動
  - 必ず状態遷移をする
    - » 遷移先が同一状態でも良い
  - 書込み，消去は禁止
  - テープの左移動，移動なしは禁止
- 語を読み終えたとき
  - 最終状態にあれば語を受理
  - そうでなければ拒否

プッシュ・ダウン・オートマトン



- 入力テープ，最終状態
  - 有限オートマトンと同様
- 以下の繰返し
  - ① まず両方のテープから読み込む
  - ② 現状態と読み込み結果に基づいて必要であれば作業テープへ書込み
  - ③ 状態遷移
- 作業テープ上の記号は
  - 読んだら無くなる
  - 書いたら追加

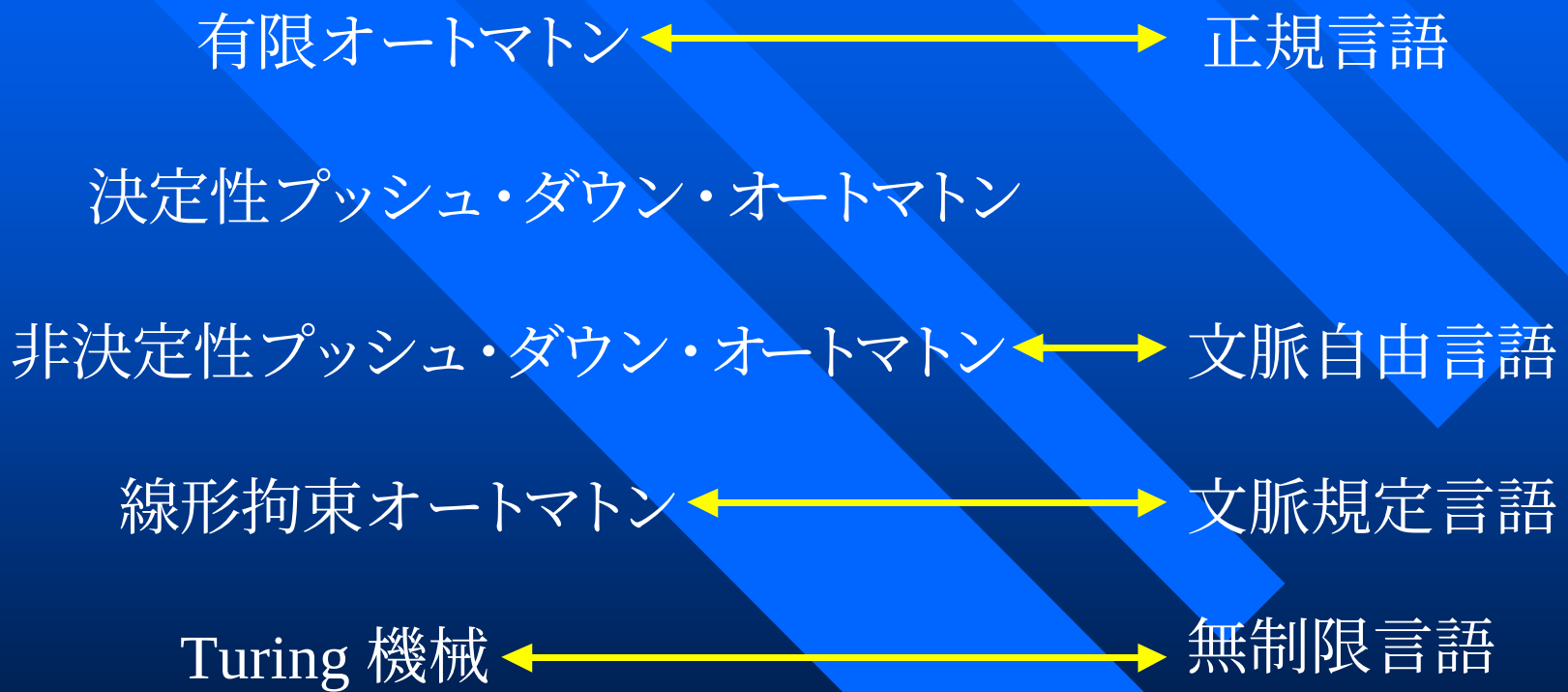
Turing 機械

- テープの使い方に全く制約がないもの
  - テープの長さは無限(必要なだけどれだけでも)
  - テープ上の記号の個数は有限
- Turing は数学者の名前，Turing 賞
- 線形拘束オートマトン
  - 使えるテープの範囲を入力語が存在する範囲に限定したもの
  - 使える範囲を定数倍に拡大しても本質的変化なし

非決定性の受理系

- 一回の動作として両立しない複数のものを平行して行うことができるという概念上の受理系
- 認識過程が並列に存在
- Forkのイメージ
- 現状を表現するものを複製して別の道へ
- 非決定性でないものは決定性

受理系と言語の関係


有限オートマトン
正規言語
決定性プッシュ・ダウン・オートマトン
非決定性プッシュ・ダウン・オートマトン
文脈自由言語
線形拘束オートマトン
文脈規定言語
Turing 機械
無制限言語

正規表現

- アルファベットΣ上の正規表現と正規表現が表す言語(語の集合)の機能的定義
  - ① 正規表現φ⇔空集合φ
  - ② 正規表現 ε ⇔空語 ε からなる集合{ ε }
  - ③ a∈Σの時，正規表現 a ⇔集合{a}
  - ④ 正規表現 s,t ⇔集合 S,T ならば

    正規表現 s+t ⇔集合 S∪T

    正規表現 st ⇔集合 ST (言語と言語の連接)

    正規表現 s* ⇔集合 S* (言語の閉包)

正規表現の例

- 01⇔{01} 語01だけからなる集合
- (0+1)⇔{0,1}
- $(0+1)(0+1)=(0+1)^2$⇔{00,01,10,11}
- (0+1)*⇔{0,1}*={ ε ,0,1,00,01,10,11,...}
- 1(0+1)*1⇔{11,101,111,1001,1011,1101,..}
- 記号の集合であるアルファベットΣを単一の記号からなる語の集合と考えるとこれは言語とみなせる
- Σ={0,1}の時, $(0+1)^2$⇔ Σ^2

正規表現が同値とは

- 二つの正規表現α,βが表す集合がひとしい時，α,βは同値であるといい，α=βと書く
- 例 (0+1)*= (0*1*)*

  (0+1)*⇔{0,1}*

  (0*1*)*⇔{{0}*{1}*}*={ ε ,0,1,00,01,...}*
  - 言語Lの閉包 L*は，L に属する任意の語を重複を許して任意個選んで，それらを任意の順に並べて得られる語のすべての集合である

正規表現の逆

- 正規表現 s ⇔集合 S の時
- 集合 $S^R=\{x^R|x\in S\}$に対応する正規表現 s^R が存在する
- そのような正規表現は
  - $\varepsilon^R=\varepsilon$
  - $a\in\Sigma$ の時, $a^R=a$
  - 正規表現 u,v に対して
    - » $s=u+v$ならば$s^R=u^R+v^R$
    - » $s=uv$ならば$s^R=v^Ru^R$
    - » $s=u^*$ならば$s^R=(u^R)^*$

sed の正規表現

- hello　　一番左のhello
- [.?!] 一番左の.?!のいずれか
- [A-Za-z]　一番左の英字
- [^0-9]　　一番左の数字以外の文字
- [hH]ello　一番左のhelloないしHello
- .　　左端の文字
- ^hello　　左端のhello
- ¥(bye¥)*　一番左のbyeの0回以上の繰返し
- bye$　　右端のbye
- ^$　　空語
- bye *bye　一番左の任意個の空白を挟んだbye
- .*　　全体
- .*bye　　左端から一番右のbyeまで
- ¥.　　一番左の.

エンティティリレイションモデル

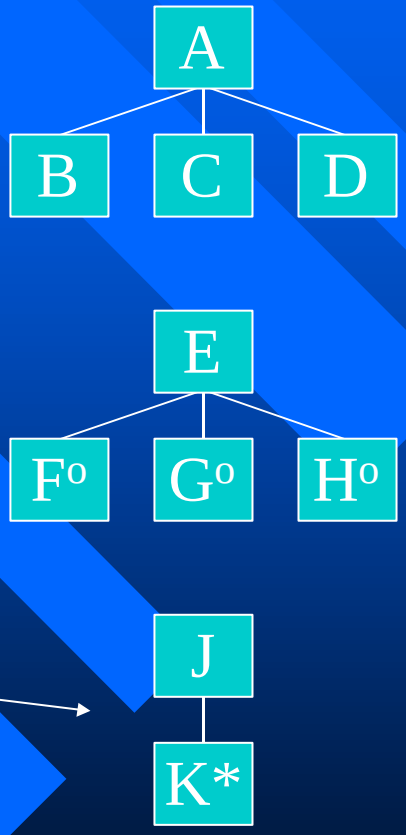
- データフローモデル
  - 定常的な情報の流れ
- 状態遷移モデル
  - 個々の情報操作のふるまい
- エンティティリレイションモデル
  - 個々の情報要素の関連
  - Entiti-Relation Model: ERM
  - データベース

エンティティアクションモデル

- より動的な側面を強調
- Entiti-Action Model
- 情報要素の振舞いもモデリング
- Michael A. Jackson
- エンティティ
  - 情報を操作すること
  - データフローモデルのエンティティはシステム外部の組織
- アクション
  - エンティティの分解

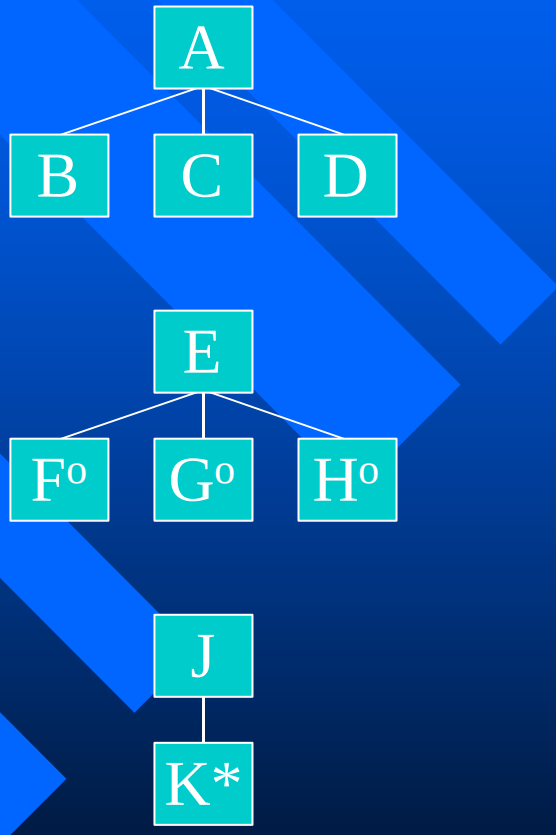
ジャクソン図

- 静的な情報構造と動的な動作構造を共に表現
- エンティティ → 
- 構造
  - 連接(A=BCD)
  - 選択(E=F+G+H)
  - 反復(J=K*)
- 何段にも構造を重ねる
  - 上ほど中小度が高い
  - 下ほど具体性が高い

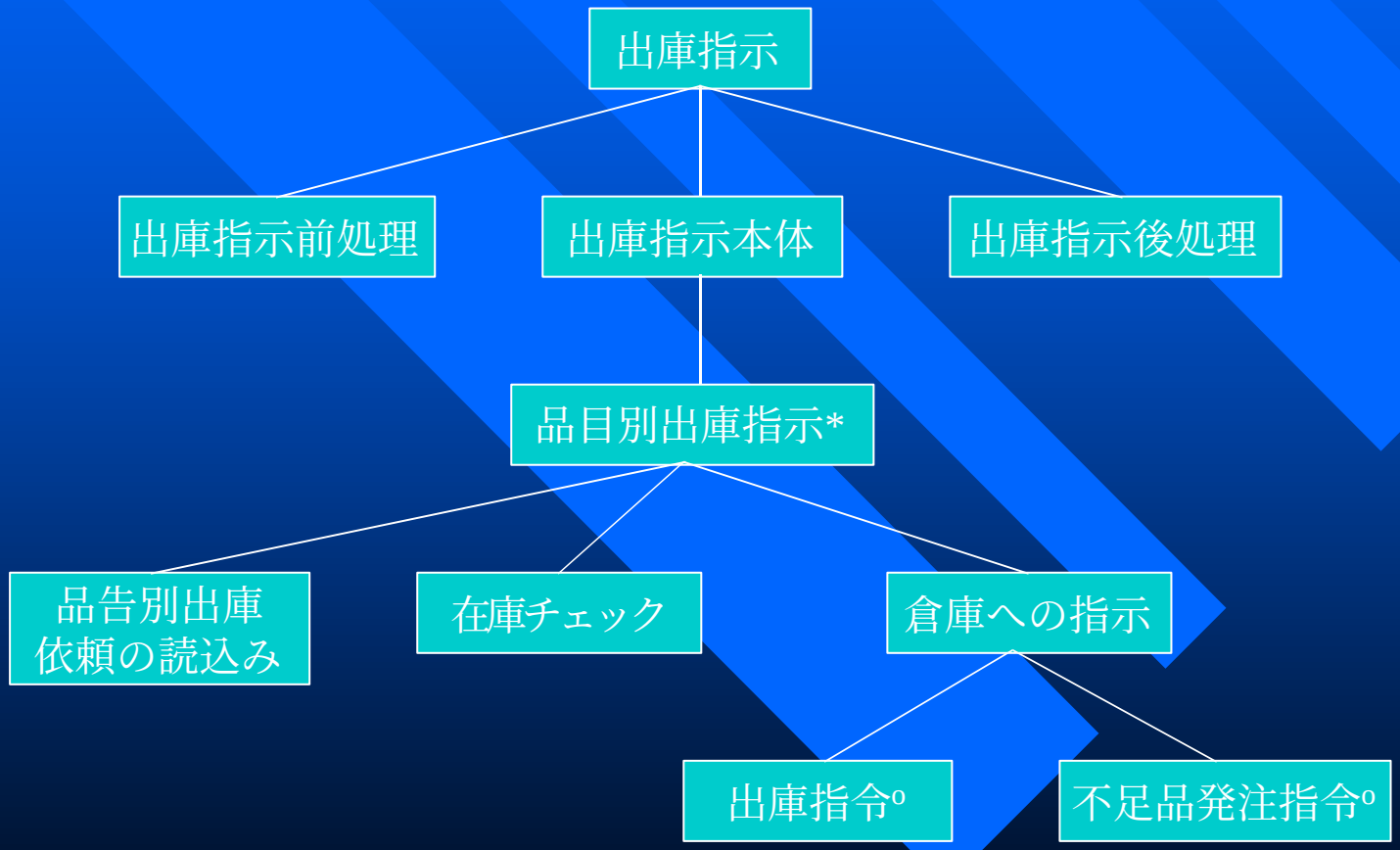


注意すべき点

- 一段の構造に連接・選択・反復の混在禁止
- 上の制限により同じ抽象度の概念の高さが揃いにくくなる
- 連接の要素が動作の場合は左から右へ動作が進む
- 正規表現と対応
- 非決定性状態遷移表現と対応
- 再帰構造は表現できない



例

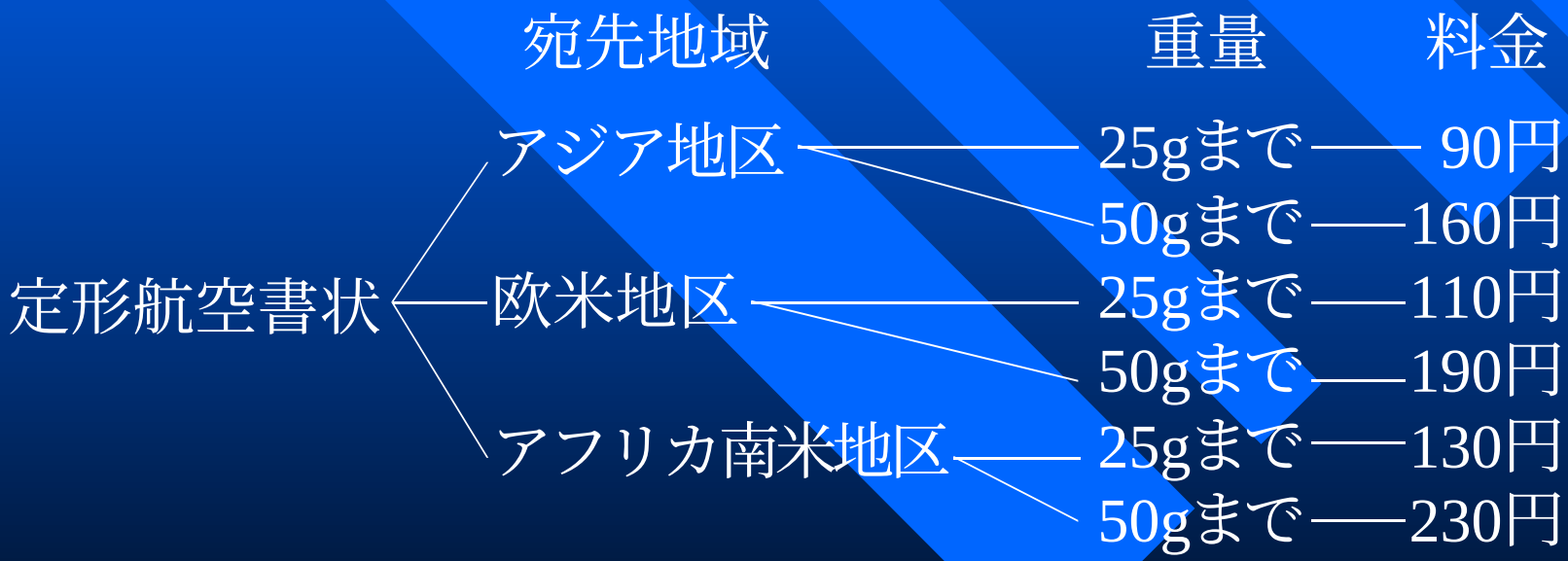

出庫指示
出庫指示前処理
出庫指示本体
出庫指示後処理
品目別出庫指示*
品告別出庫
依頼の読込み
在庫チェック
倉庫への指示
出庫指令o
不足品発注指令o

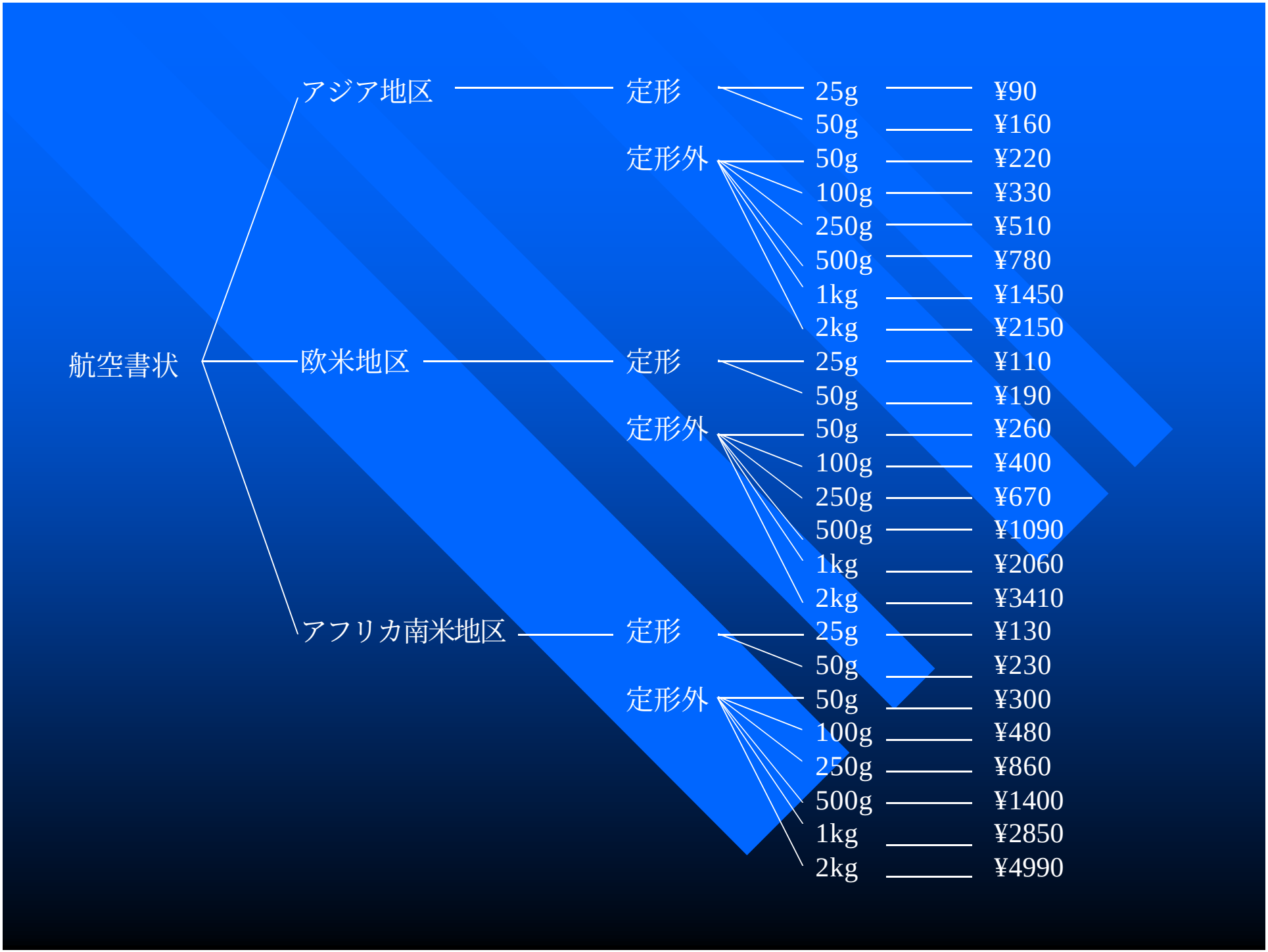
静的なロジックのモデル

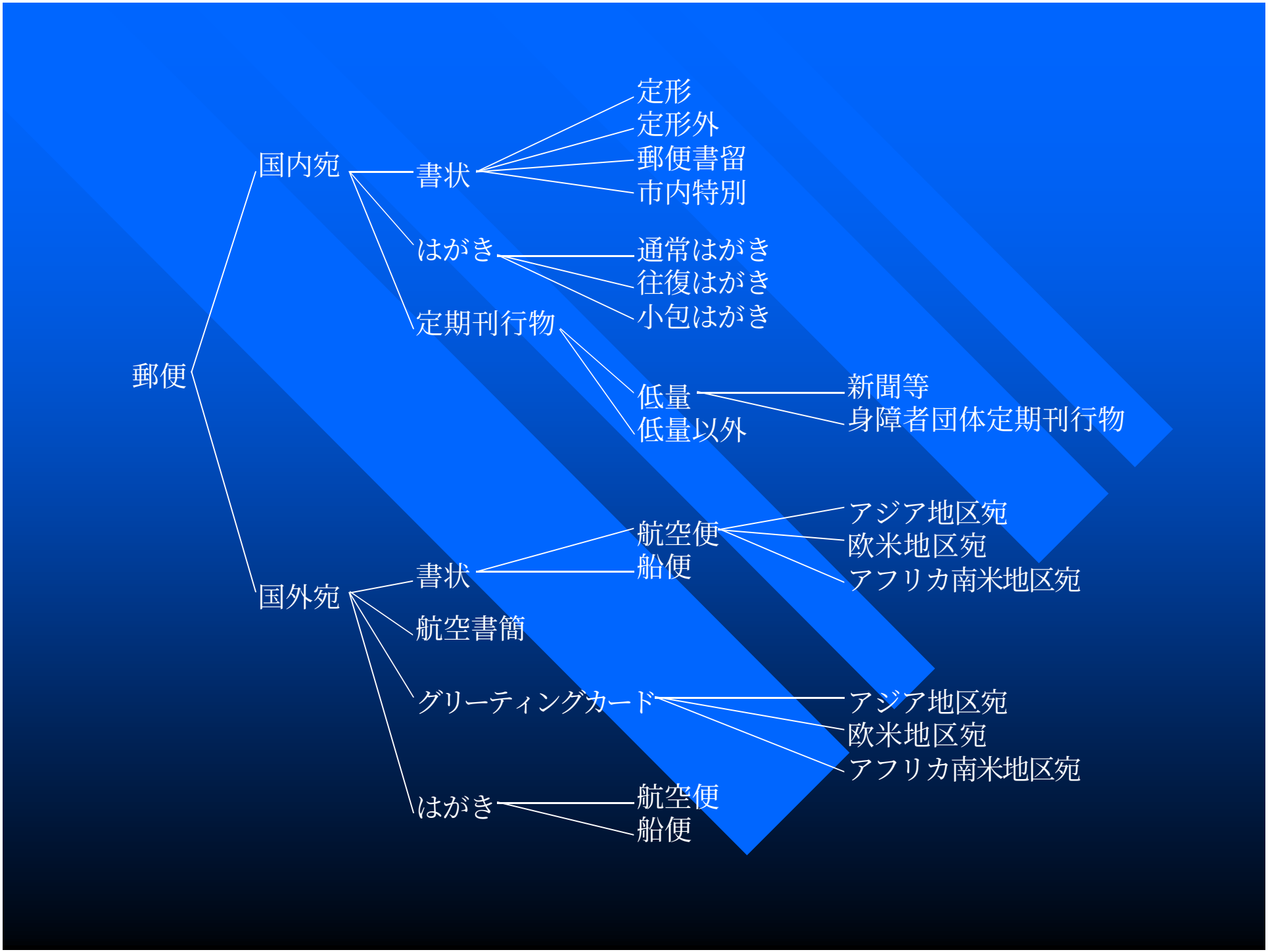
- 決定木
- 決定表
- BDD
  - BDD: Binary Decision Diagram
  - SROBDD: Shared Reduced Ordered BDD
  - ZBDD: Zero-suppressed BDD
- 原因結果グラフ (論理回路図)

決定木 (decision tree)

- 静的な構造は段階的な場合分けの積み重ね




航空書状
アジア地区
定形
25g ¥90
50g ¥160
定形外
50g ¥220
100g ¥330
250g ¥510
500g ¥780
1kg ¥1450
2kg ¥2150
欧米地区
定形
25g ¥110
50g ¥190
定形外
50g ¥260
100g ¥400
250g ¥670
500g ¥1090
1kg ¥2060
2kg ¥3410
アフリカ南米地区
定形
25g ¥130
50g ¥230
定形外
50g ¥300
100g ¥480
250g ¥860
500g ¥1400
1kg ¥2850
2kg ¥4990

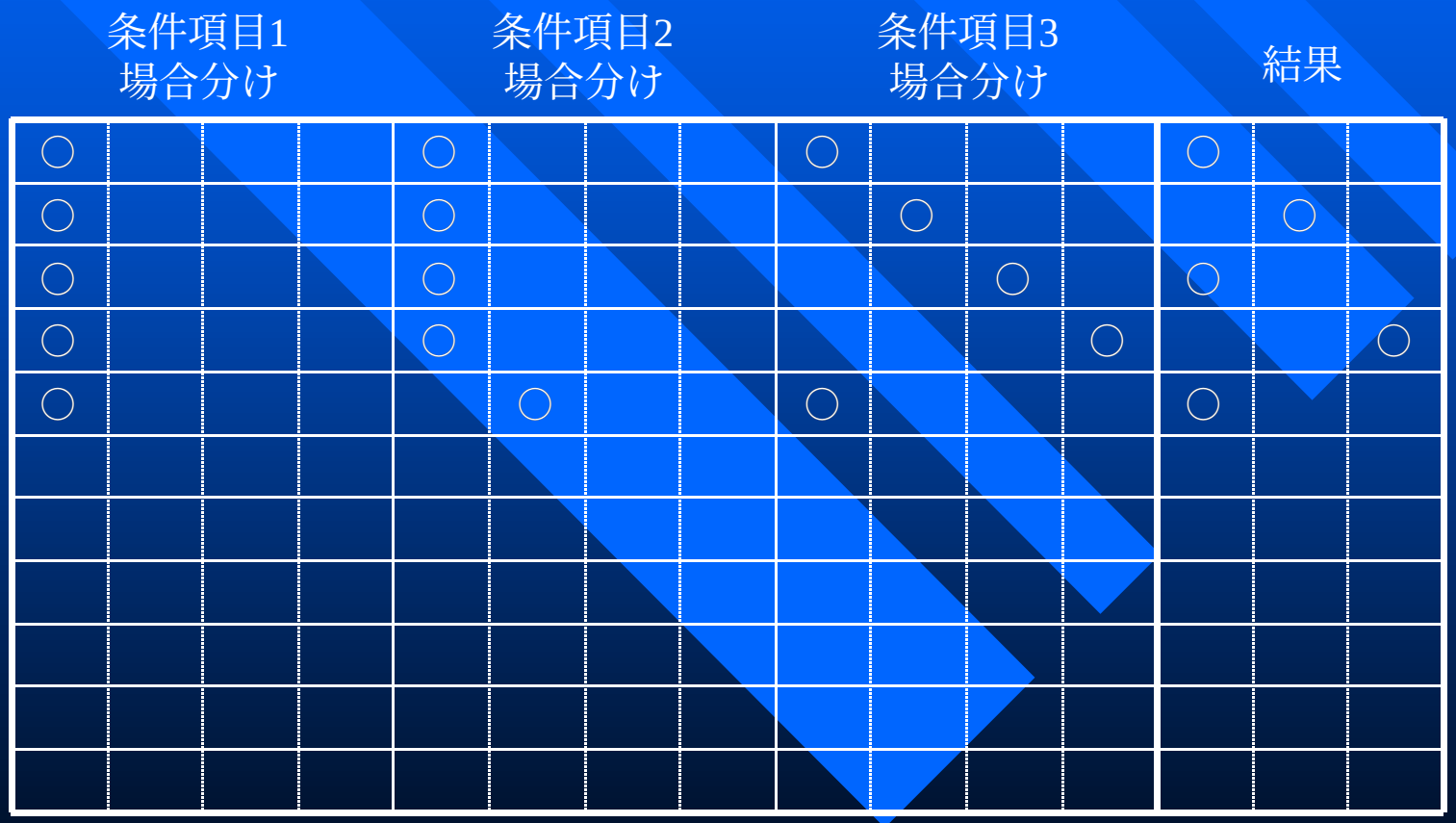

郵便
国内宛
書状
定形
定形外
郵便書留
市内特別
はがき
通常はがき
往復はがき
小包はがき
定期刊行物
低量
新聞等
身障者団体定期刊行物
低量以外
国外宛
書状
航空便
船便
アジア地区宛
欧米地区宛
アフリカ南米地区宛
航空書簡
グリーティングカード
アジア地区宛
欧米地区宛
アフリカ南米地区宛
はがき
航空便
船便

決定木の利点

- 複雑で大規模な条件を系統だてて整理できる点
- 組合せによる抜け落ちを比較的容易に検出できる
  - 全体がコンパクトになることから
  - 気が付かなければそれまで

決定表 (decision table)

タイトル



対応する決定木


条件項目1
条件項目1
条件項目1
結果
タイトル
場合1
場合2
場合3
場合4
場合1
場合2
場合3
場合4
場合1
場合2
場合3
場合4
内容1
内容2
内容3

決定表の簡単化

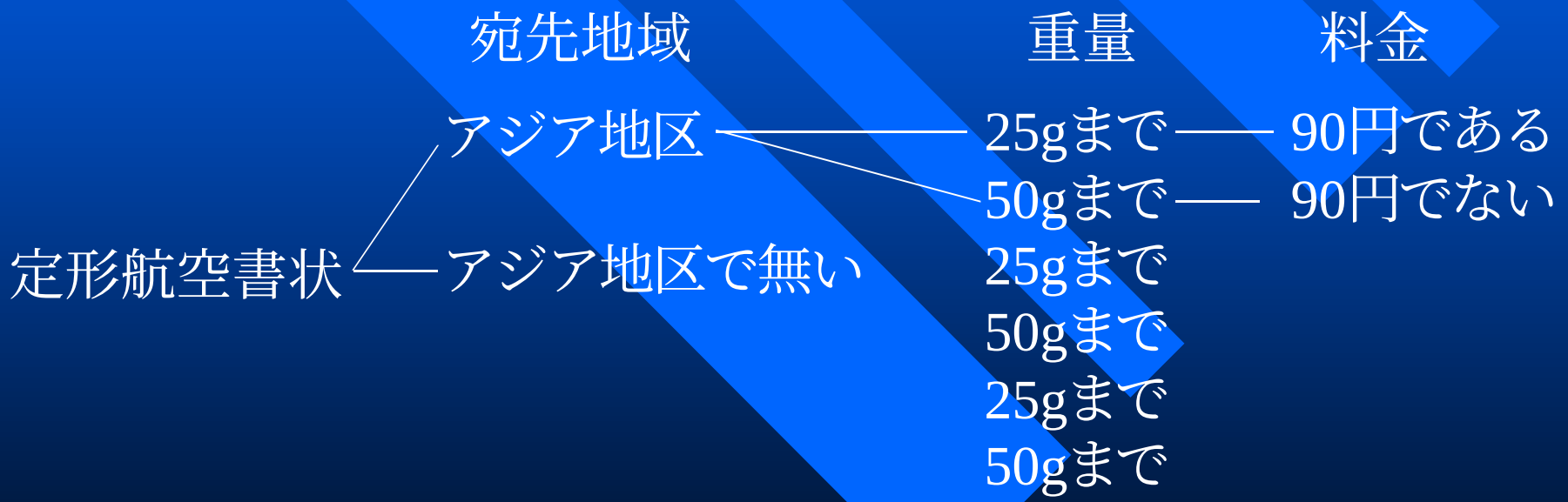
決定表の利点

- 全ての組合せを必然的に考慮することになるので条件の抜け落ちが発生しにくい
- 機械的な操作を適用しやすい

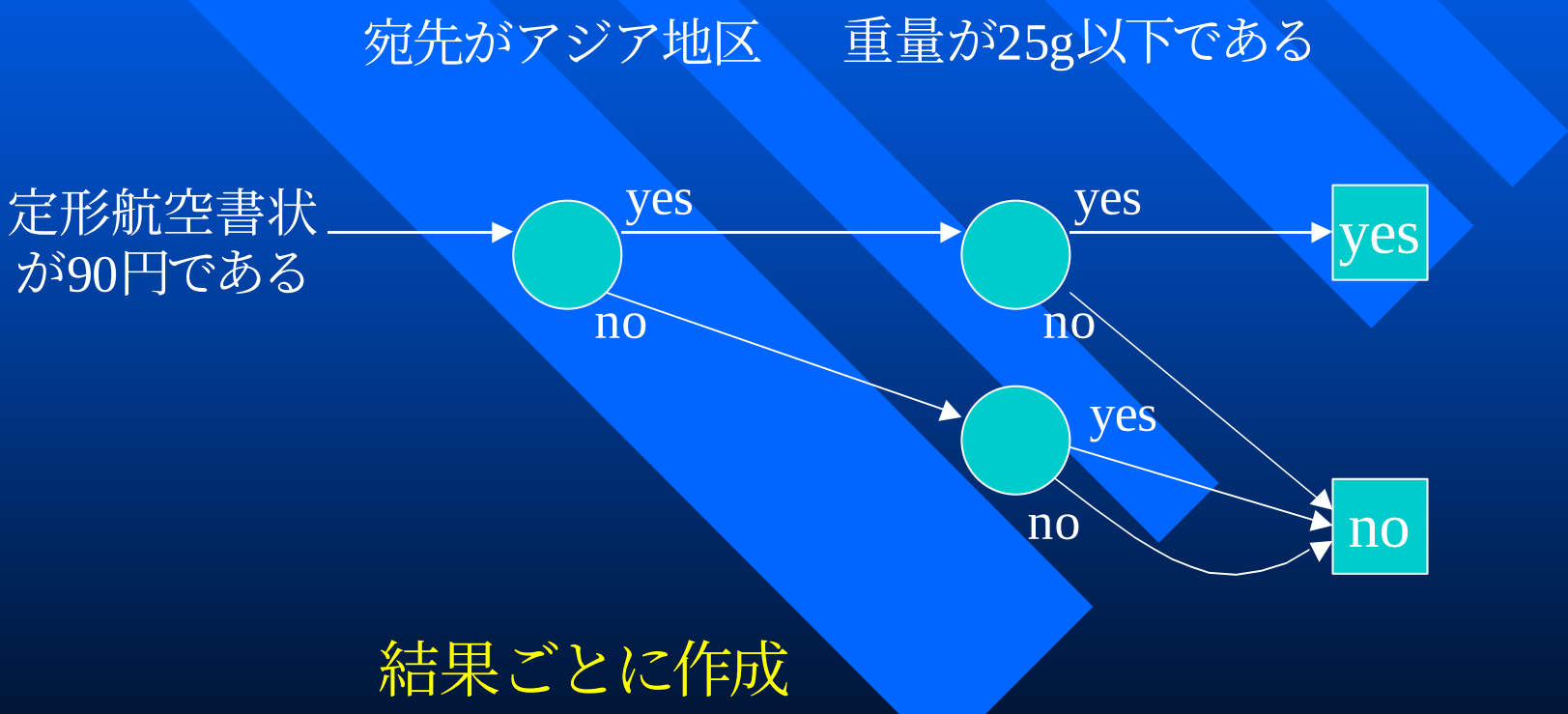
Binary Decision Diagram

- 場合分けが2つになるように全ての条件項目(変数)を選ぶ
- 結果も「～である／～でない」のように2つのいずれかであるように整理する
- 決定木を作る
- これをBinary Decision Treeではなくて Binary Decision Diagramといっている

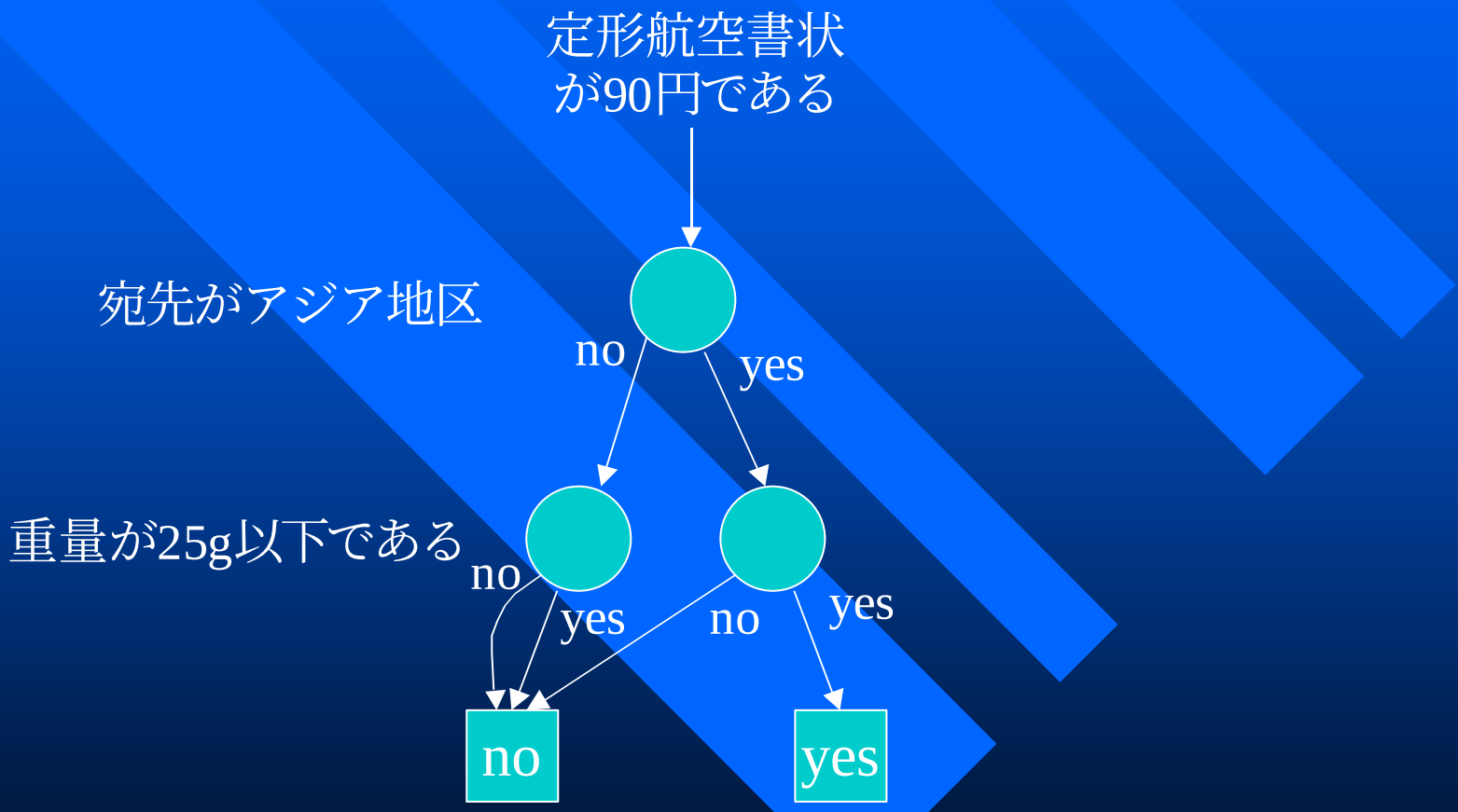
BDDの例 (1)


宛先地域
重量
料金
アジア地区
25gまで
90円である
50gまで
90円でない
定形航空書状
アジア地区で無い
25gまで
50gまで
25gまで
50gまで

BDDの例 (2)


宛先がアジア地区
重量が25g以下である
定形航空書状
が90円である
yes
yes
yes
no
no
yes
no
結果ごとに作成

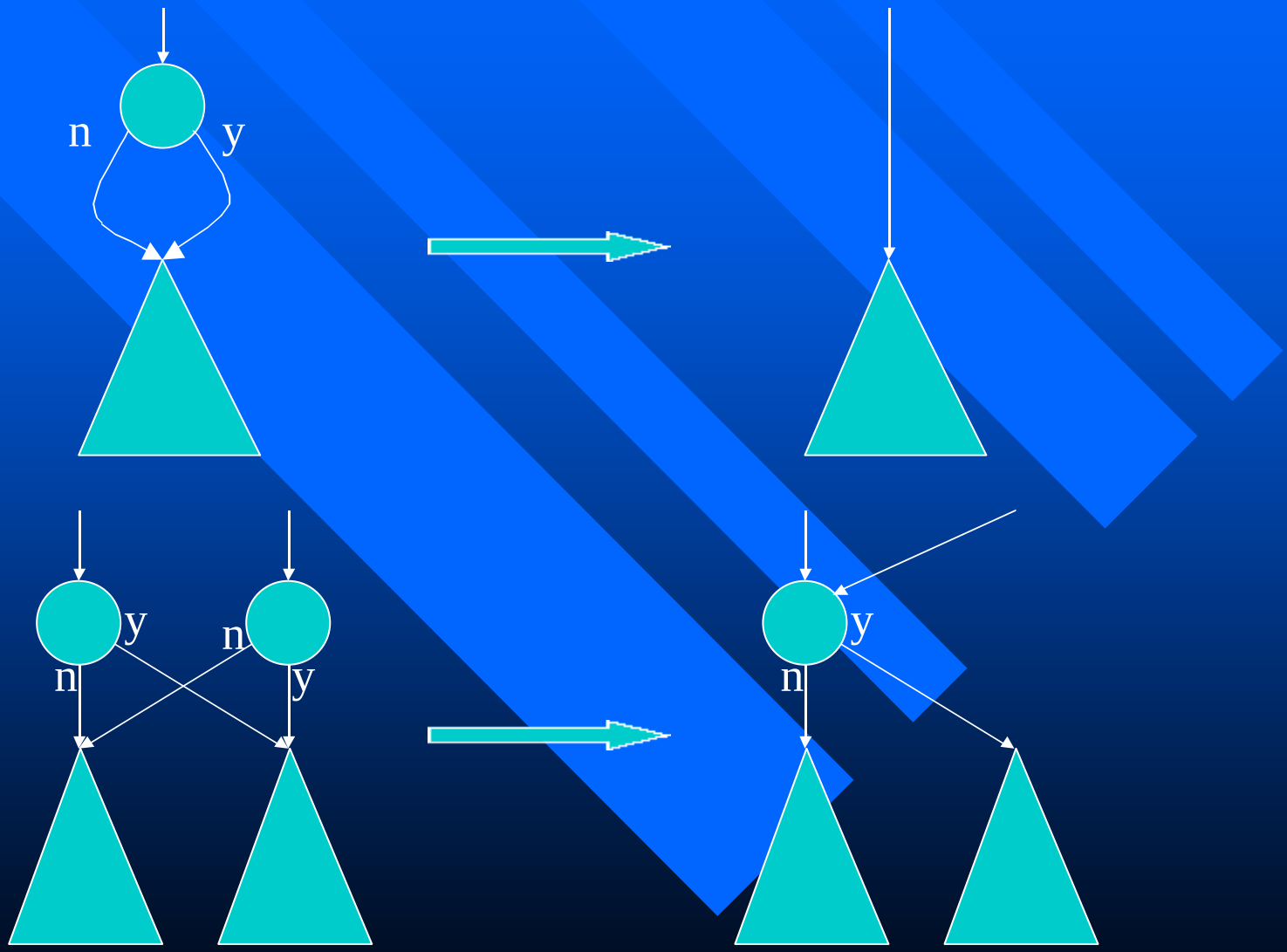
BDDの例 (3)


定形航空書状
が90円である
宛先がアジア地区
no
yes
重量が25g以下である
no
yes
no
yes
no
yes

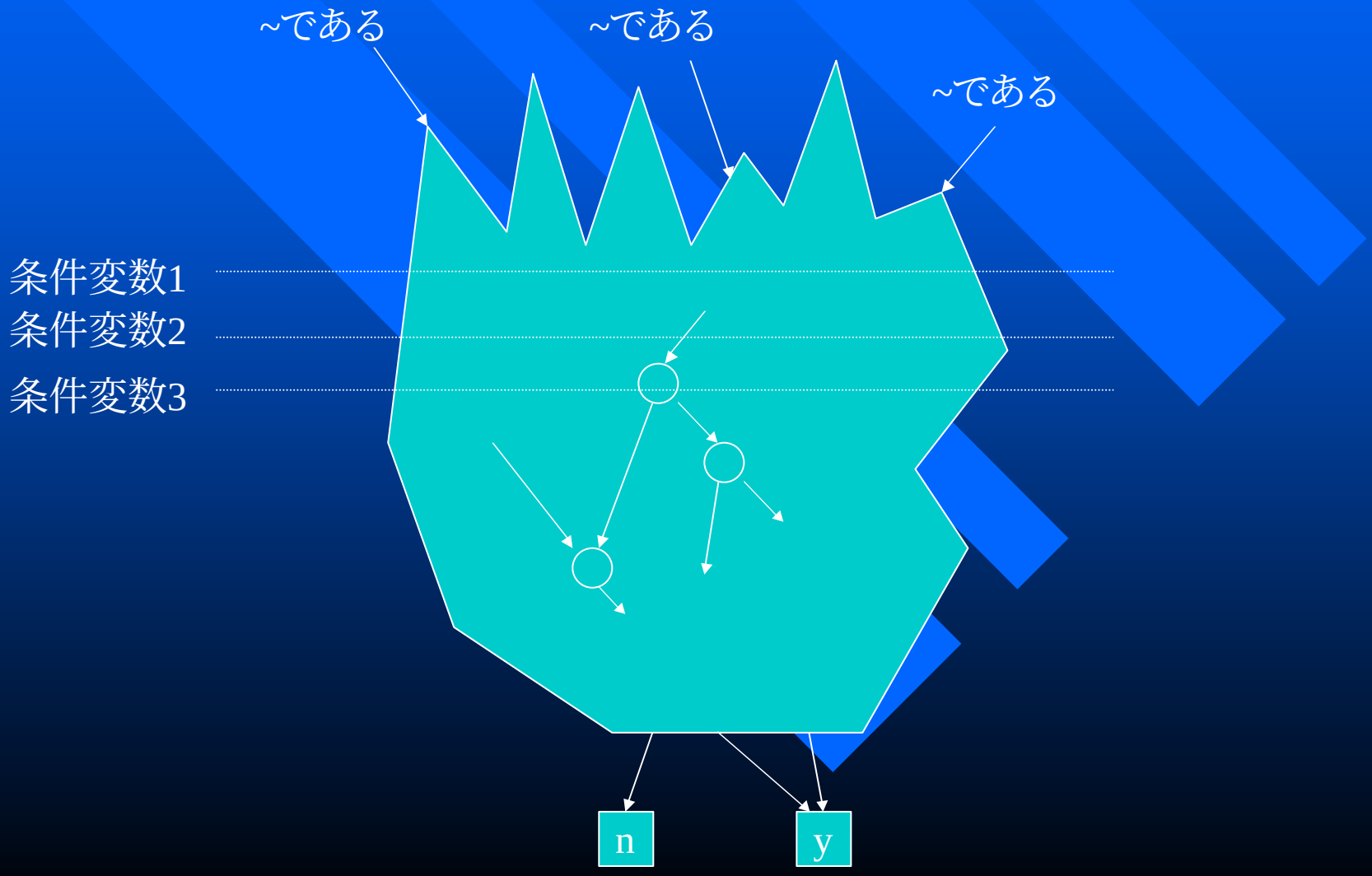
Shared Reduced Ordered BDD

- 結果ごとの複数のBDDをまとめて扱う
- 同じ変数を表す○は同じ高さ
- 次の2つの規則により○を削除，共有

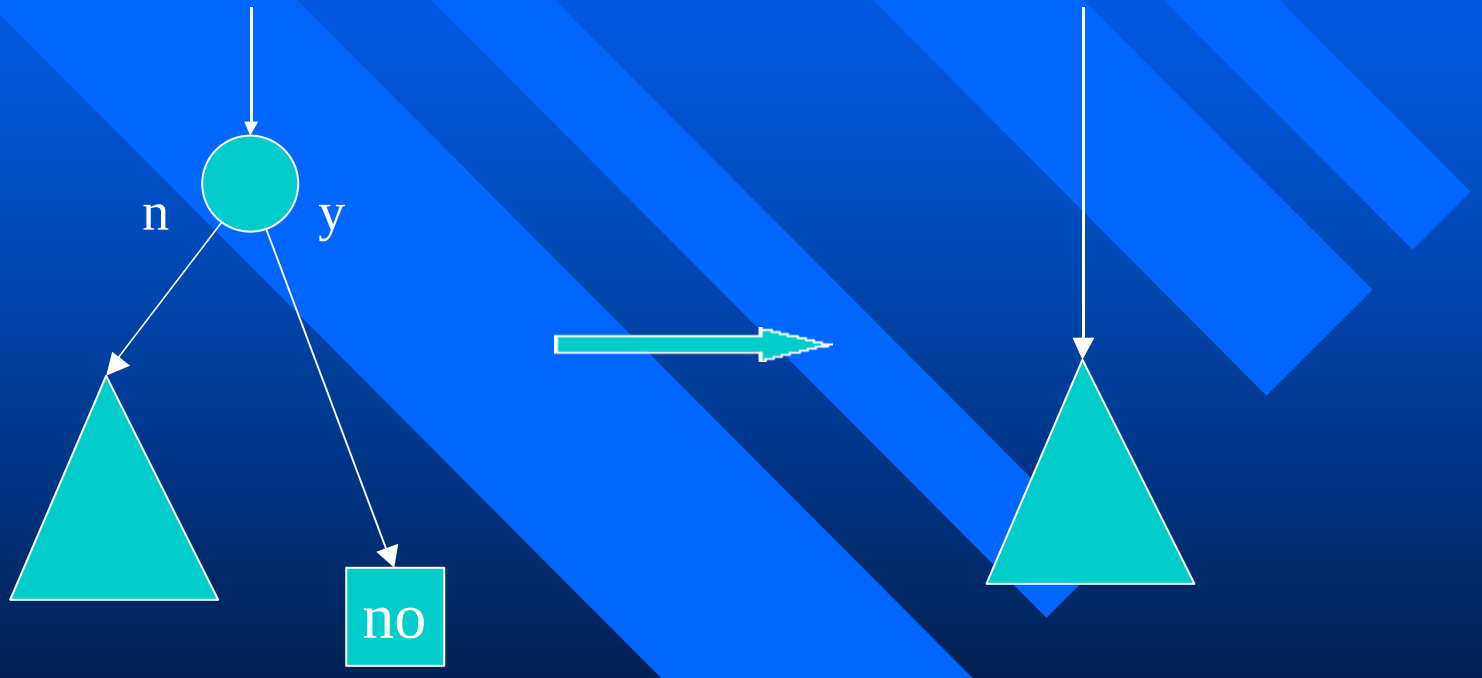
簡約規則



SROBDD
(これを普通BDDという)



ZBDDの簡約規則



乱数の生成

- ガロア体
- 拡大体
- べき表現
- ベクトル表現
- 原始元をかける
- 行列による積表現
- 回路による積表現
- M系列

ガロア体

加減乗除の四則演算が行えるような集合を体という。

たとえば、有理数、実数、複素数全体の集合はそれぞれ体をなす。

元の数が有限なものを有限体またはガロア体と呼び、$GF(q)$ で表す。

q は元の数であり、位数と呼ばれる。

ガロア体は任意の位数に対して存在するわけではなく、

位数が素数のべき p^m のとき、またそのときに限って存在する(天下り)。

位数が素数 p であるガロア体は簡単に作ることができる。

これには p 個の整数の集合 $\{0,1,2,\cdots,p-1\}$ を考え、

これに対して $\bmod p$ で加算、乗算を行えばよい。

拡大体

実数体上(係数が実数)では因数分解できない

(既約である)多項式 x^2+1 の 1 つの根 i (虚数単位)を

実数体に付加することによって複素数体が得られた。

このように体 F 上では既約な多項式

(F の元を係数とする多項式で、

F の元を係数とする形では因数分解できない多項式)

を考え、その根を付加してより大きな体を作ることを、

体の拡大という。

$GF(p)$ に $GF(p)$ 上の m 次既約多項式の根を

1 つ付加することにより、$GF(p^m)$ が得られる(天下り)。

べき表現

ある種の m 次既約多項式 $F(x)$ を選ぶと、

$GF(p^m)$ の非零の全ての元を

$F(x)$ のある根 α のべきで表すことが出来る(天下り)。

このような既約多項式を原始多項式といい、

その根を $GF(p^m)$ の原始元という。

いいかえれば、 α は、 $\alpha^0(=1), \alpha^1, \alpha^2, \cdots, \alpha^{p^m-2}$ がすべて異なり、

(p^m-1) 乗してはじめて $\alpha^{p^m-1}=1$ に戻る元なのである。

これを $GF(p^m)$ の元のべき表現という

ベクトル表現

a は m 次既約多項式 $F(x)$ の根であるから $F(a)=0$。

a^m を残してそれ以外を右辺へ移項すれば、

$G(a)$ を $m-1$ 次以下の多項式として、$a^m = G(a)$ と表せる。

m 乗を超える部分に次々とこれを適用すれば、$GF(p^m)$ の

全ての元 a^i は、a の $m-1$ 次以下の多項式として表せる。

すなわち $a^i = a_0 + a_1 a + \cdots + a_{m-1} a^{m-1}$ と表すことができる。

係数の組 $(a_0, a_1, \cdots, a_{m-1})$ と元が対応するので

$(a_0, a_1, \cdots, a_{m-1})$ を $GF(p^m)$ の元のベクトル表現という。

例

2 元体

$GF(2)$ を 2 元体という。2 元体の加法表、乗法表と逆元を以下に示す。

+	0	1
0	0	1
1	1	0

x	--x
0	0
1	1

•	0	1
0	0	0
1	0	1

x	1/x
0	---
1	1

2 元体の拡大体

$GF(2^m)$ においては $a + a = (1+1)a = 0$ なので

$a = -a$ であり、＋と－は区別されない。

原始多項式の例

$1 + x + x^4$ ：2 元体上の 4 次原始多項式

原始元をかける(1)

2 元体上のb 次原始多項式

$$F(x) = f_0 + f_1 x + \cdots + f_{b-1} x^{b-1} + x^b$$

の原始元となる根をa とする。

従って$GF(2^b)$ の非零の全ての元を

a から初めて次々にa をかけていくことにより

作り出すことができる。

原始元をかける(2)

このことをベクトル表現で考えてみる。

ベクトル表現が$(a_0, a_1, \cdots, a_{b-1})$である$GF(2^b)$のある元

$$a = a_0 + a_1 a + \cdots + a_{b-1} a^{b-1}$$

もちろんこれはaの何乗かであるはずであるが、

これにaを乗じること、

すなわち次の元を作ることを考える

原始元をかける(3)

まず単純には $aa = a_0 a + a_1 a^2 + \cdots + a_{b-1} a^b$ であるが、

右端項の a^b に

$F(a) = f_0 + f_1 a + \cdots + f_{b-1} a^{b-1} + a^b = 0$ から得られる

$a^b = f_0 + f_1 a + \cdots + f_{b-1} a^{b-1}$ を代入すると、

$aa = a_{b-1} f_0 + (a_0 + a_{b-1} f_1) a + \cdots + (a_{b-2} + a_{b-1} f_{b-1}) a^{b-1}$ となる。

従って aa のベクトル表現は

$(a_{b-1} f_0, a_0 + a_{b-1} f_1, \cdots, a_{b-2} + a_{b-1} f_{b-1})$ となる。

行列による積表現

aa のベクトル表現

$(a_{b-1}f_0, a_0 + a_{b-1}f_1, \cdots, a_{b-2} + a_{b-1}f_{b-1})$ は

$$(a_0, a_1, \cdots, a_{b-1}) \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \\ f_0 & f_1 & f_2 & \cdots & f_{b-1} \end{bmatrix}$$

と表せる。

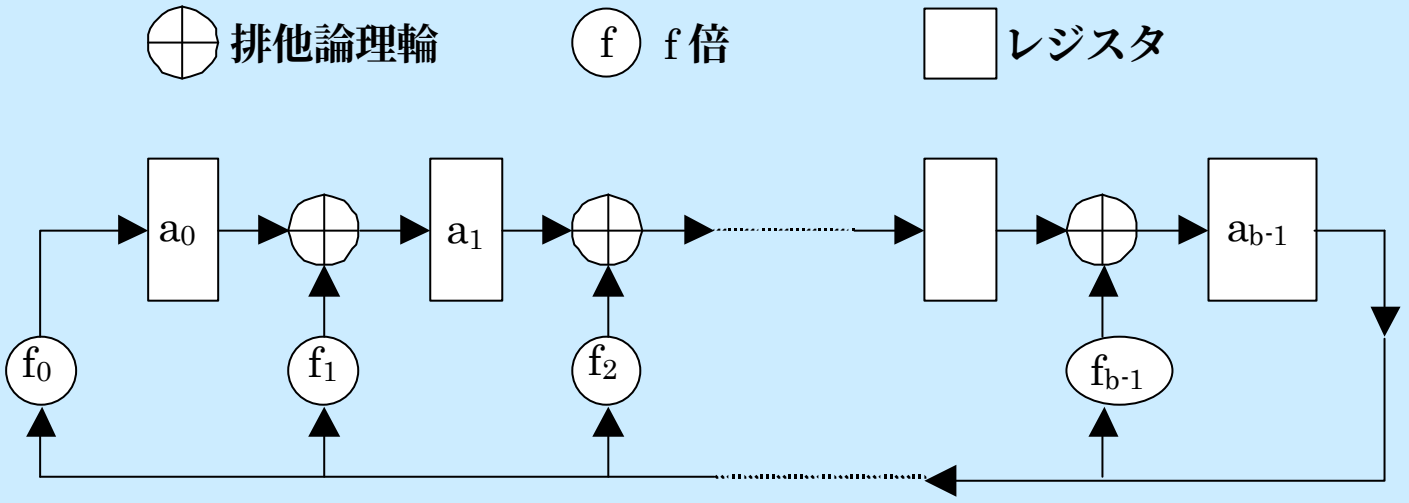
これは a のベクトル表現である $(a_0, a_1, \cdots, a_{b-1})$ に

行列を右からかけた形になっている。

もちろん行列をかける演算は 2 元体上で行うものとする。

回路による表現

行列による積表現は

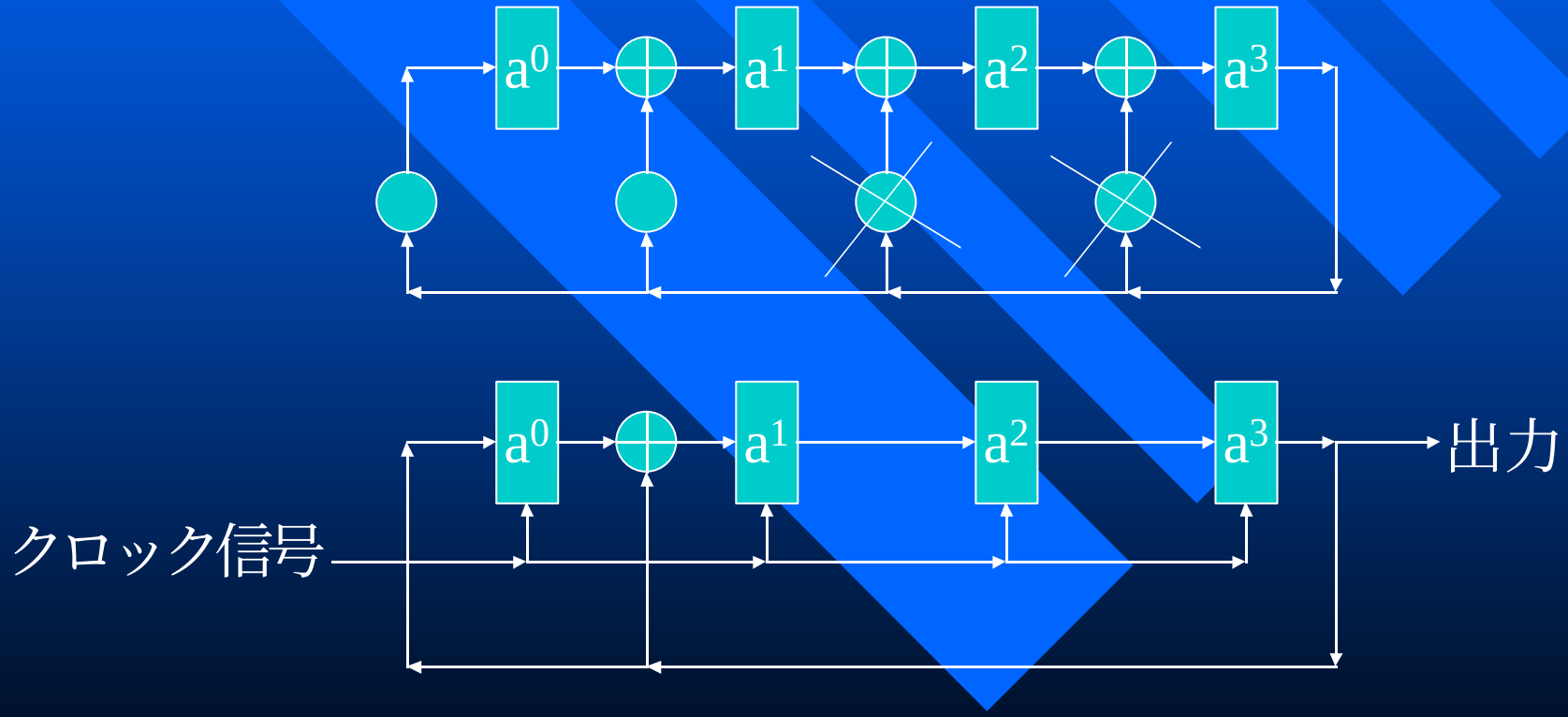


のような回路により実現することができる。

- この回路によって得られる系列を最長周期系列，M系列という.

例：$1+x+x^4$によるM系列

- $1+x+x^4=1+1x+0x^2+0x^3+1x^4$



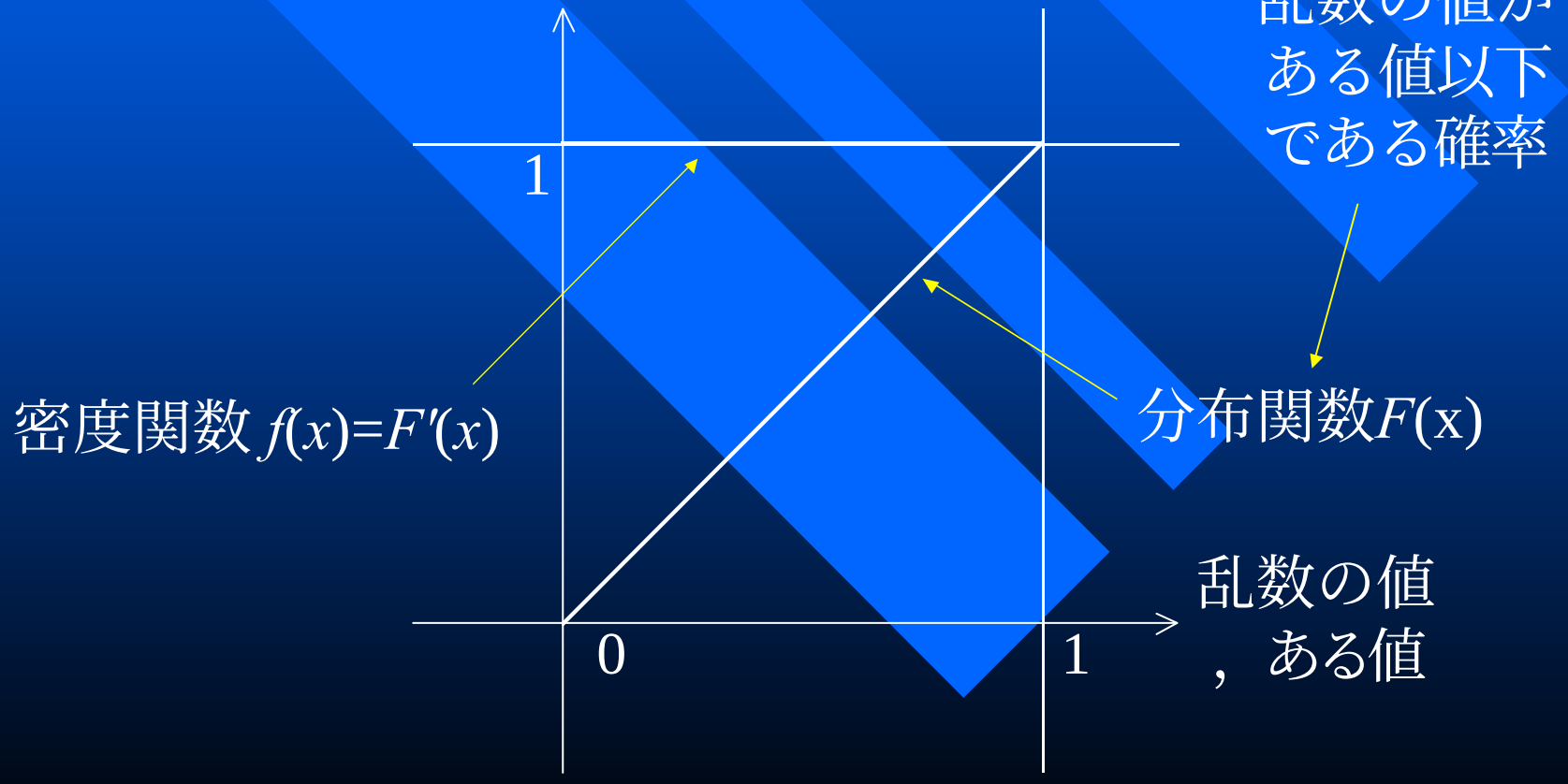
乱数

- 乱数列 $x^1, x^2, x^3, \ldots$
  - x^k が $x^k < c$ を満たす確率は c だけに依存
- 乱数
  - 乱数列の一つ一つ

実数の乱数

- 分布関数
  - 乱数 Xが $X<x$ を満たす確率 $F(x)$をX の分布関数という
- 密度関数
  - $F(x)$ の導関数 $f(x)=F'(x)$ を乱数の密度関数という
- 一様乱数
  - 密度関数が一定の乱数

分布関数，密度関数



整数の一様乱数

- ある範囲内の整数が等確率で現れる乱数
- 整数の一様乱数 rand() から実数の一様乱数 U（$0 \leqq U < 1$）を作る

  $U=(1.0 / (M + 1.0) \times (\mathrm{rand}() + 0.5)$

  ただし，$0 \leqq \mathrm{rand}() \leqq M$

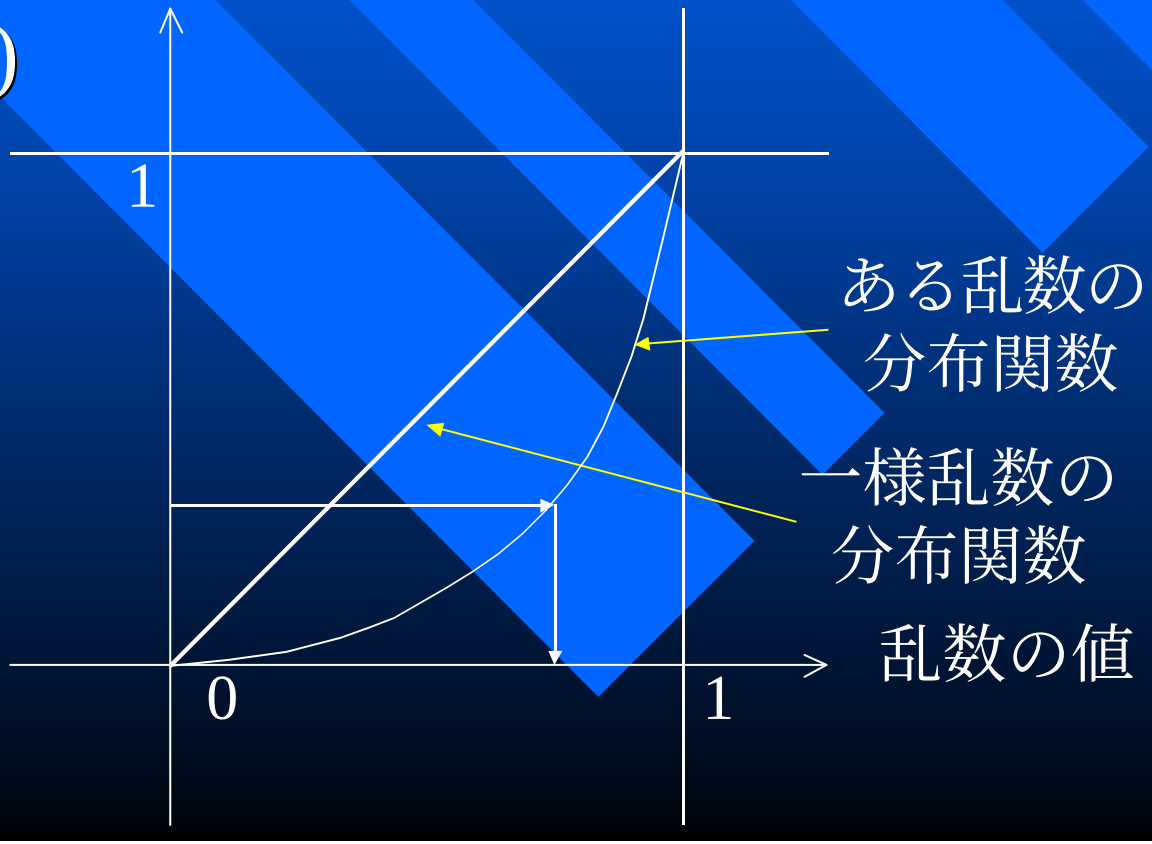
  $M=99$だと

  $U=(1/100) \times (\mathrm{rand}()+0.5)$

  Uが0や1になるのを避けるため

任意の分布の乱数

- 分布関数 $F(x)$ の乱数 Xは
  実数の一様乱数U（$0 \leqq U < 1$）を使って
  $X=F^{-1}(U)$



乱数の発生法

- ノイズ（熱運動）の検出によるもの

東芝製

- 擬似乱数(pseudo random number)発生プログラム

擬似乱数
(pseudo random number)

- 線形合同法
- Knuthの乱数発生法
- Wichmann-Hillの乱数発生法
- M系列乱数
- 松本眞・西村拓士のMersenne Twister

線形合同法

- $x_i = (a\, x_{i-1} + c) \bmod M$ ， i=1,2,3...
- x_0は適当な初期値
- $0 \leqq x_i < M$
- M は２の累乗

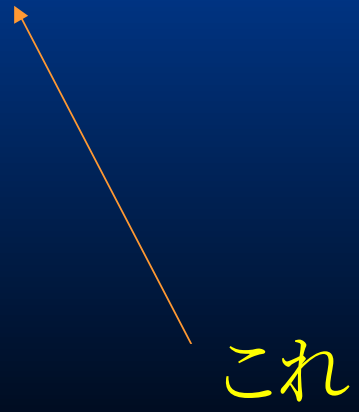
  a は $a \bmod 8 = 5$

  c は奇数

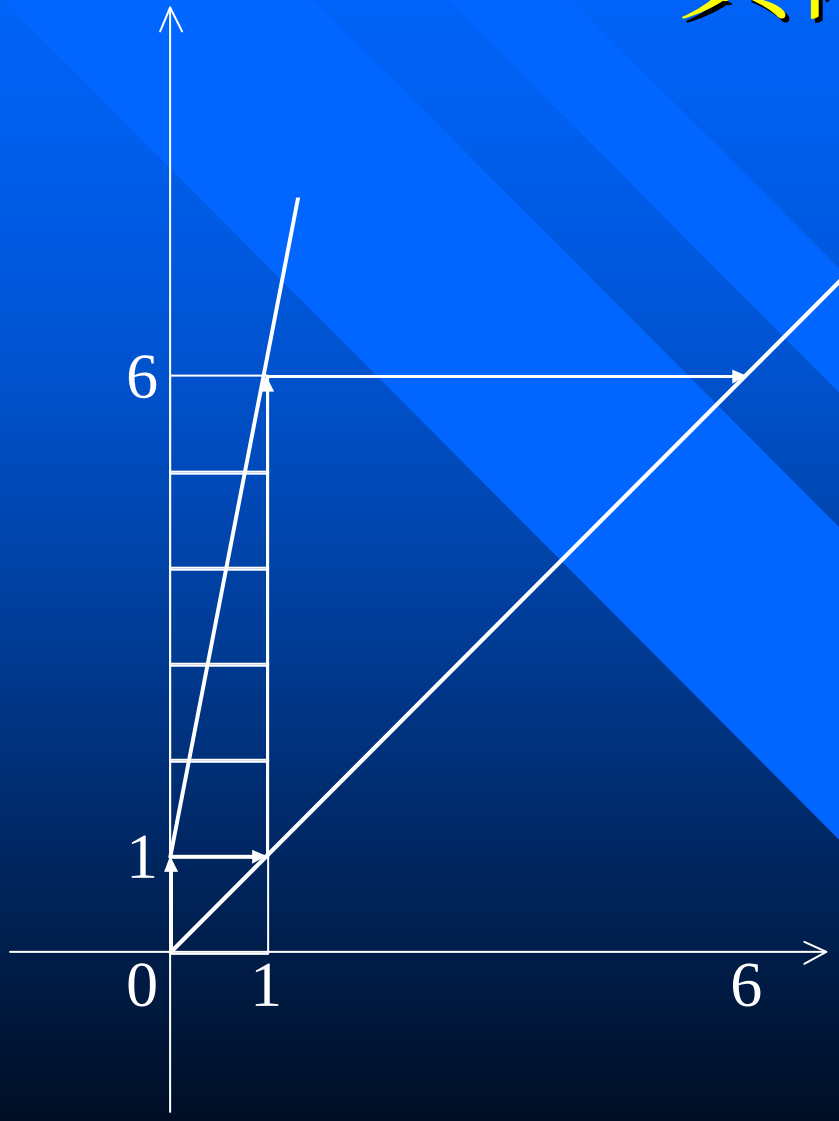
  だと周期はちょうど M

Knuth

- TEX を作った人



具体例で



- M=8, a=5, c=1, x_0=0
- $x_i = (5x_{i-1}+1) \bmod 8$
- 0,1,6,7,4,5,2,3,0,...

Mersenne Twister

- 1996~1997 松本眞・西村拓士
- 周期が2^19937-1
- 623次元超立方体の中に 均等に分布
  - 証明されている
- Cの標準ライブラリのrand()より高速
- メモリ効率が良い
  - 624ワードでOK

MTの仕組み

- 2元体 GF(2) 上の19937次の原始多項式を見つけるとその根は0を除いて

  $\alpha^0(=1), \alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^{2\text{^}19937-2}$
- 根は次々とαをかける事で手に入る
- これは2^19937-1個ある（周期も同じ）
- 19937はメルセンヌ数なので2^19937-1は素数
- これはどんな部分周期も持たない
- 0を除いた根に1,2,3,....と整数を割当る

メルセンヌ数

- 2^M-1が素数となるような数M
- 2,3,5,7,13,17,19,31,61,89,
- 107,127,521,607,1279,2203,2281,3217,4253,4423,
- 9689,9941,11213,19937,21701,...
- 1998年37番目3021377

MT用原始多項式の発見

- 規約性の判定
  - 計算量が次元によらない判定アルゴリズム
  - フロベニウス写像のオーダーを計算
    - » 2^19937-1の素数性をうまく使った

Wichmann-Hillの乱数発生法

- 16ビット処理系用
- x_0=918999161
- x_i=16555425264690 x_{i-1} mod 27817185604309
- 答えは x_i / 27817185604309
- 周期 6.95×10^{12}
- たいした事は無い

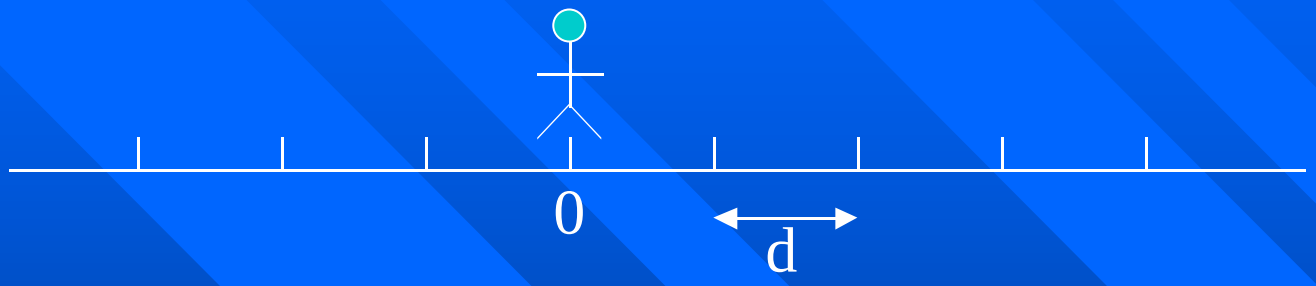
rand()

- ANSI Cで定義されている乱数生成関数
- 線形合同法
- $x_i=(1103515245\ x_{i-1}+12345) \bmod 2^{31}$
- 周期 2^{31}
- たいした事は無い

Knuthの乱数発生法

- 原始多項式 $x^{55}+x^{24}+1$ によるM系列
- たいした事は無い

ランダムウォーク



- $p(x,t)$: 時刻 t に位置 x に存在する確率
  - $x = md$ (m は整数)
  - $t = nT$ (n は負でない整数)
  - $p(0,0)=1$
  - $p(x,0)=0$ $(x \neq 0)$
  - $p(x,t+T)=1/2p(x-d,t)+1/2p(x+d,t)$

漸化式の変形

$$p(x,t+T)-p(x,t)=\frac{(p(x+d,t)-p(x,t))-(p(x,t)-p(x-d,t))}{2}$$

$$\frac{p(x,t+T)-p(x,t)}{T}=\frac{d^2}{2T}\frac{\frac{p(x+d,t)-p(x,t)}{d}-\frac{p(x,t)-p(x-d,t)}{d}}{d}$$

- ここで2つの考え方
  - ① 極限をとる
  - ② 連続関数で近似

①極限をとる

$$\frac{p(x,t+T)-p(x,t)}{T}=\frac{d^2}{2T}\frac{\dfrac{p(x+d,t)-p(x,t)}{d}-\dfrac{p(x,t)-p(x-d,t)}{d}}{d}$$

$\dfrac{d^2}{2T}$ を一定として $T,d\to 0$ の極限をとると

$p(x,t)$ は連続関数となり上式は

$$\frac{\partial p(x,t)}{\partial t}=\frac{d^2}{2T}\frac{\partial^2 p(x,t)}{\partial x^2}$$

という微分方程式となる

このとき初期条件は $p(x,0)=\delta(x)$ ディラックのデルタ関数

②連続関数で近似

$$\frac{p(x,t+T)-p(x,t)}{T}=\frac{d^2}{2T}\frac{\dfrac{p(x+d,t)-p(x,t)}{d}-\dfrac{p(x,t)-p(x-d,t)}{d}}{d}$$

$p(x,t)$ を $x=md$ $t=nT$ のところで定義値をとる
連続関数とみなす

さらに T d が十分小さいとして上式をテーラー展開を使って

$$\frac{\partial p(x,t)}{\partial t}=\frac{d^2}{2T}\frac{\partial^2 p(x,t)}{\partial x^2}$$ で近似する

このとき初期条件は $p(x,0)=\delta(x)$ で①と同じ

拡散方程式

$$\frac{\partial p(x,t)}{\partial t} = \frac{d^2}{2T}\frac{\partial^2 p(x,t)}{\partial x^2}$$ を $$\frac{\partial p(x,t)}{\partial t} = D\frac{\partial^2 p(x,t)}{\partial x^2}$$

と書いて，これを拡散方程式という　D は拡散係数

$t > 0$　$p(x,0) = \delta(x)$　のときこの解は

$$p(x,t) = \frac{1}{\sqrt{4\pi Dt}} e^{-\frac{x^2}{4Dt}}$$

正規分布

平均 m 分散 σ^2 の正規分布を $N(m, \sigma^2)$ と書き

その分布関数は $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-m)^2}{2\sigma^2}}$ である

$p(x,t) = \frac{1}{\sqrt{4\pi Dt}} e^{-\frac{x^2}{4Dt}}$ と比べると $2Dt = \sigma^2$

離散値のまま計算(1)

- 単位時間 T ごとに1コマ d ずつ，それぞれ1/2 の確率で右か左に移動する酔っ払いが右に n^+ 回，左に n^- 回移動した結果，ある時刻 nT で位置 md にいる確率 $p(md, nT)$ を求める

まず $n^+ + n^- = n$　$n^+ - n^- = m$　である

$p(md, nT)$ は n 回の移動のうち右移動が n^+ 回である確率なので

$$p(md,nT) = {}_nC_{n^+}\left(\frac{1}{2}\right)^n = \frac{n!}{n^+!(n-n^+)!}\left(\frac{1}{2}\right)^n$$

離散値のまま計算(2)

$$p(md,nT)={}_nC_{n^+}\left(\frac{1}{2}\right)^n=\frac{n!}{n^+!(n-n^+)!}\left(\frac{1}{2}\right)^n$$ は

$n^+ + n^- = n$ $n^+ - n^- = m$ を使って

$$p(md,nT)=\frac{n!}{n^+!n^-!}\left(\frac{1}{2}\right)^n=\frac{n!}{\left(\frac{n+m}{2}\right)!\left(\frac{n-m}{2}\right)!}\left(\frac{1}{2}\right)^n$$

離散値のまま計算(3)

スターリングの公式：n が非常に大きい時

$$n! \cong n^n \sqrt{2\pi n} e^{-n}$$ または $$\log(n!) \cong (n+\tfrac{1}{2})\log n - n + \tfrac{1}{2}\log(2\pi)$$

を使うと結局

$$p(md, nT) = \frac{2}{\sqrt{2\pi n}} e^{-\frac{m^2}{2n}}$$

連続値と離散値での比較(1)

連続値:
$$p(x,t)=\frac{1}{\sqrt{4\pi Dt}}e^{-\frac{x^2}{4Dt}}$$

離散値:
$$p(md,nT)=\frac{2}{\sqrt{2\pi n}}e^{-\frac{m^2}{2n}}$$

連続値に $x=md$ $t=nT$ $D=\frac{d^2}{2T}$ を代入すると

$$p(md,nT)=\frac{1}{\sqrt{4\pi\frac{d^2}{2T}nT}}e^{-\frac{m^2d^2}{4\frac{d^2}{2T}nt}}=\frac{1}{\sqrt{2\pi nd}}e^{-\frac{m^2}{2n}}$$

となる

連続値と離散値での比較(2)

離散値:

$$p(md,nT)=\frac{2}{\sqrt{2\pi n}}e^{-\frac{m^2}{2n}}$$

と今求めた

が $2d$ 倍大きい

ここ

$$p(md,nT)=\frac{1}{\sqrt{2\pi nd}}e^{-\frac{m^2}{2n}}$$

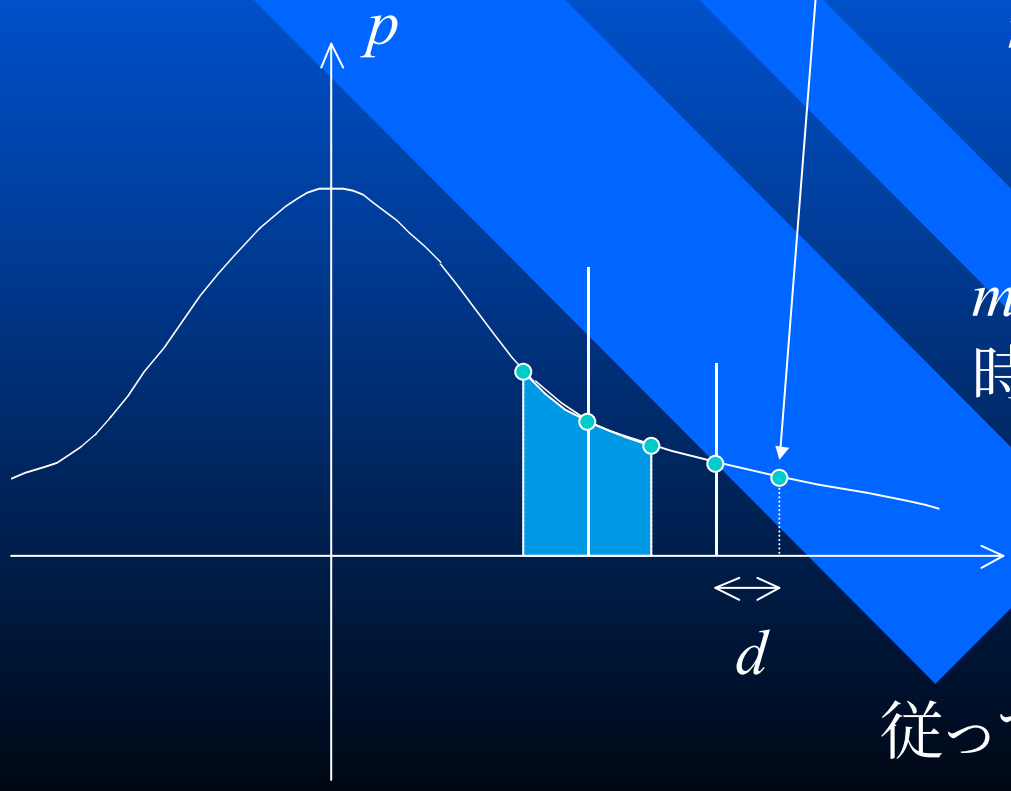
が一致しない

ここが異なる！

連続値と離散値での比較(3)

$$p(md, nT) = \frac{1}{\sqrt{2\pi nd}} e^{-\frac{m^2}{2n}}$$

は　のような連続関数上の値



n が一定の時
m は1つおきの値
しかとらない
ことに注意して

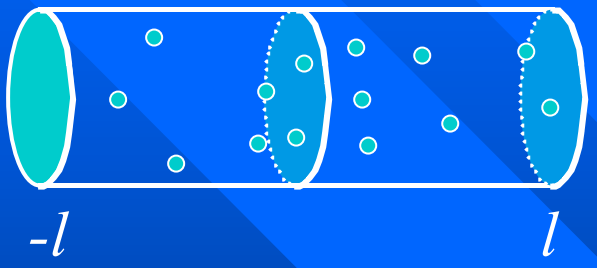
md が離散値しかとらない
時の確率としての値は

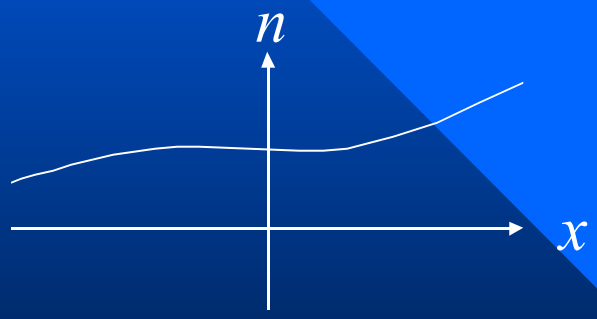
■ の面積

の値となる

従って $2d$ の差が説明された

拡散係数の物理的意味(1)





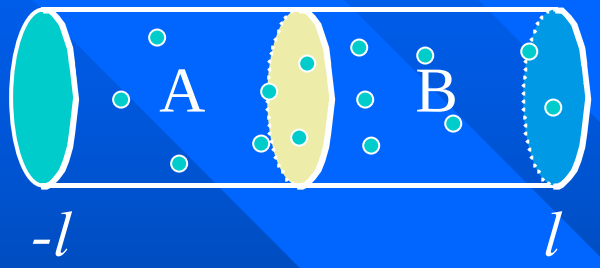
断面積 1 の円筒を考え,
その中の
粒子密度を $n(x,t)$ とする
どの粒子も v_{th} の速度で
右または左に移動しており,
l 進むと一斉に
他の粒子とぶつかり
移動の向きが変わる.
衝突後の向きは
1/2 の確率で
右または左である.

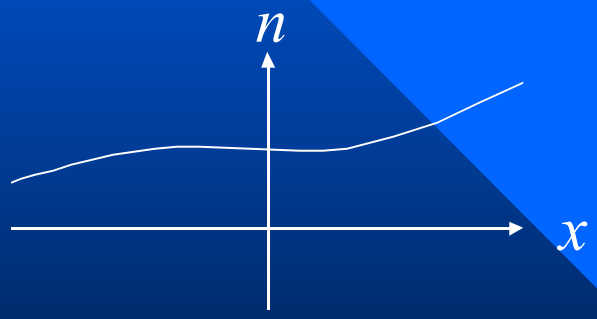
l　平均自由行程

$$t_c = \frac{l}{v_{th}}$$

平均緩和時間

拡散係数の物理的意味(2)





一斉衝突直後に図のAの部分に存在した

$\int_{-l}^{0} n(x)dx$ 個の粒子の半分は

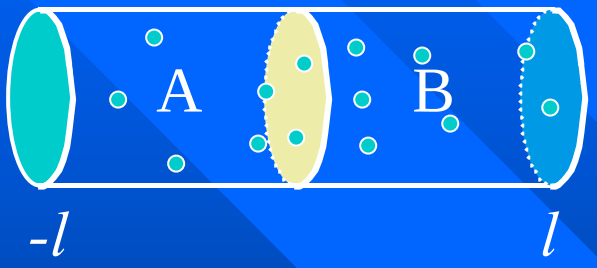
t_c 時間後の一斉衝突までに

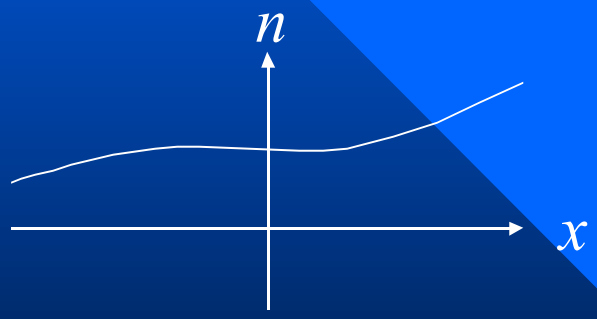
を左から右に通過する.

同様に図の B の部分に存在した $\int_{0}^{l} n(x)dx$ 個の粒子の半分は

を右から左へ通過する.

拡散係数の物理的意味(3)





従って結局，一斉衝突から

一斉衝突までの t_c 時間で

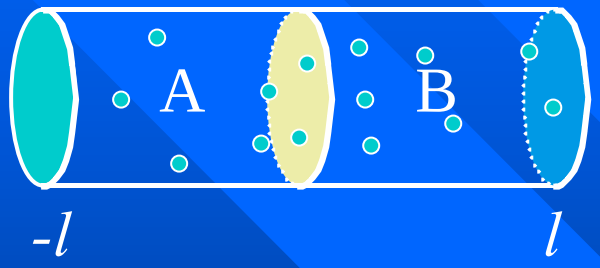
$$\frac{\int_{-l}^{0} n(x)dx - \int_{0}^{l} n(x)dx}{2}$$

個の

粒子が を左から右へ

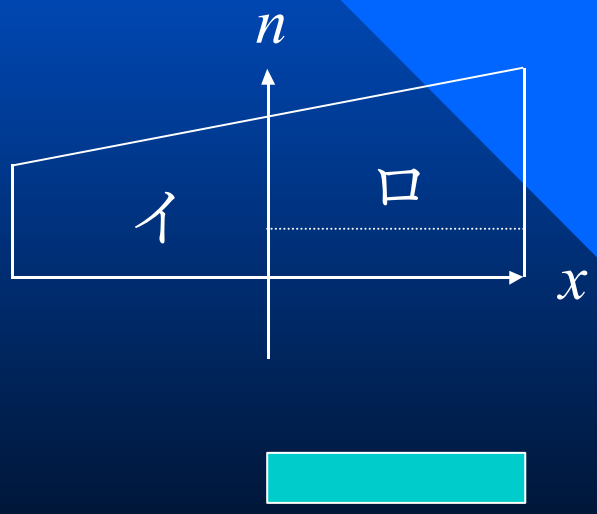
通過する.

拡散係数の物理的意味(4)



$n(x)$ を直線で近似すると

$$\frac{\int_{-l}^{0} n(x)dx - \int_{0}^{l} n(x)dx}{2}$$ は

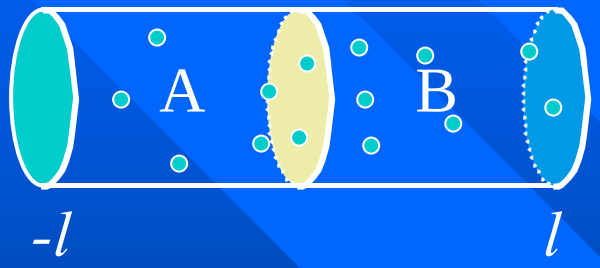


イの面積からロの面積を

引いたものの1/2なので

$$-\frac{1}{2}\frac{dn}{dx}l^2$$ となる.

拡散係数の物理的意味(5)



$-\frac{1}{2}\frac{dn}{dx}l^2$ は t_c 時間に

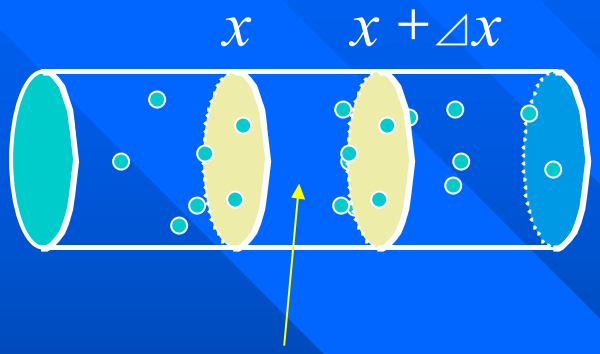
を左から右へ通過する個数

なので，結局 単位時間の通過個数は $-\frac{l^2}{2t_c}\frac{dn}{dx}$

$\frac{l^2}{2t_c}$ を D と書いて拡散係数という

この意味は「密度の勾配に比例して粒子の移動が起こる」である

拡散係数の物理的意味(6)



粒子の密度を $n(x,t)$

拡散係数を D とすると

この部分に Δt 時間に流入・流出する粒子は

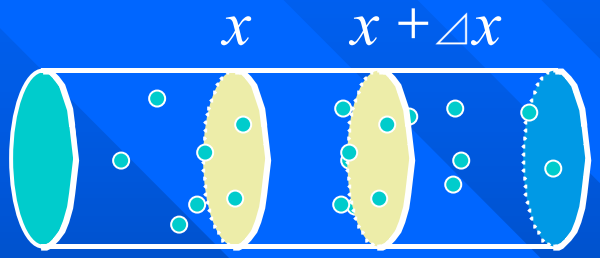
右から $\Delta tD(\frac{\partial n}{\partial x})_{x+\Delta x}$ が流入し

左から $\Delta tD(\frac{\partial n}{\partial x})_{x}$ が流出する．この合計が

密度の上昇となるので

$$\Delta x \frac{\partial n}{\partial t}\Delta t = \Delta tD\left((\frac{\partial n}{\partial x})_{x+\Delta x} - (\frac{\partial n}{\partial x})_{x}\right)$$

拡散係数の物理的意味(7)



$$\Delta x \frac{\partial n}{\partial t} \Delta t = \Delta t D \left((\frac{\partial n}{\partial x})_{x+\Delta x} - (\frac{\partial n}{\partial x})_x \right)$$

は

$$\frac{\partial n}{\partial t} = D \frac{\left((\frac{\partial n}{\partial x})_{x+\Delta x} - (\frac{\partial n}{\partial x})_x \right)}{\Delta x} = D \frac{\partial^2 n}{\partial x^2}$$

となる.

これは拡散方程式である.

Langevin方程式(1)

- ブラウン運動
  - ランダムウォークに対応する物理現象
- Langevin方程式
  - ブラウン運動や拡散の運動方程式
  - 質量mの粒子の1次元(x軸)運動方程式

$$m\frac{d^2x}{dt^2} = X - f\frac{dx}{dt}$$

速度に比例する抵抗を受ける
その比例定数を粘性f

速いとたくさんぶつかる

他の粒子とぶつかる事により発生

平均すると0となる乱数的力

Langevin方程式(2)

$$m\frac{d^2x}{dt^2} = X - f\frac{dx}{dt}$$ ----------①

$$\frac{d^2x^2}{dt^2} = \frac{d}{dt}(\frac{dx^2}{dt}) = \frac{d}{dt}(2x\frac{dx}{dt}) = 2\frac{dx}{dt}\frac{dx}{dt} + 2x\frac{d^2x}{dt^2}$$ なので

$$\frac{m}{2}\frac{d^2x^2}{dt^2} = \frac{m}{2}(2(\frac{dx}{dt})^2 + 2x\frac{d^2x}{dt^2}) = m(\frac{dx}{dt})^2 + mx\frac{d^2x}{dt^2}$$

ここに①を代入して

$$\frac{m}{2}\frac{d^2x^2}{dt^2} = m(\frac{dx}{dt})^2 + (X - f\frac{dx}{dt})x$$

Langevin方程式(3)

$$\frac{m}{2}\frac{d^2x^2}{dt^2} = m\left(\frac{dx}{dt}\right)^2 + \left(X - f\frac{dx}{dt}\right)x$$

は

$$\frac{m}{2}\frac{d^2x^2}{dt^2} - m\left(\frac{dx}{dt}\right)^2 = Xx - fx\frac{dx}{dt} = Xx - f\frac{1}{2}\frac{dx^2}{dt}$$

と変形でき

結局

$$\frac{m}{2}\frac{d^2x^2}{dt^2} - m\left(\frac{dx}{dt}\right)^2 = Xx - \frac{f}{2}\frac{dx^2}{dt}$$

となる

Langevin方程式(4)

場合
ここで多くの粒子についての

$$\frac{m}{2}\frac{d^2x^2}{dt^2} - m\left(\frac{dx}{dt}\right)^2 = Xx - \frac{f}{2}\frac{dx^2}{dt}$$

を平均すると

$$\frac{m}{2}\frac{d^2<x^2>}{dt^2} - m<\left(\frac{dx}{dt}\right)^2> = <Xx> - \frac{f}{2}\frac{d<x^2>}{dt}$$

ここで平均する操作(積分)と微分は順序を交換できるものとした

さらにxとXは独立なので

$$<Xx> = <X><x> = 0\cdot 0 = 0$$

したがって

$$\frac{m}{2}\frac{d^2<x^2>}{dt^2} - m<\left(\frac{dx}{dt}\right)^2> = -\frac{f}{2}\frac{d<x^2>}{dt}$$

Langevin方程式(5)

$$\frac{m}{2}\frac{d^2<x^2>}{dt^2} - m<(\frac{dx}{dt})^2> = -\frac{f}{2}\frac{d<x^2>}{dt}$$

は

$$\frac{d<x^2>}{dt} = u$$

とおくと

$$\frac{m}{2}\frac{du}{dt} - m<(\frac{dx}{dt})^2> = -\frac{f}{2}u$$

これは分散の時間変化率

$$\frac{du}{dt} - 2<(\frac{dx}{dt})^2> = -\frac{f}{m}u$$

となって

結局

$$\frac{du}{dt} + \frac{f}{m}u = 2<(\frac{dx}{dt})^2>$$

エネルギー等分配則

- 熱平衡状態では1自由度あたり $\frac{1}{2}kT$ の
  エネルギーが等しく分配される
- 種々の粒子が混在する時にも成立
- $pV = nRT$ が気体の種類によらない根拠
  - p: 圧力，V: 体積，n: モル数，T: 温度
  - R: 気体定数，これをアボガドロ数で割ると k
  - k: ボルツマン定数

Langevin方程式(6)

$$\frac{du}{dt}+\frac{f}{m}u=2<(\frac{dx}{dt})^2>$$

は運動エネルギーに

エネルギーが等分配されるとする

$$\frac{1}{2}m<(\frac{dx}{dt})^2>=\frac{1}{2}kT$$

によって

$$\frac{du}{dt}+\frac{f}{m}u=\frac{2}{m}kT$$

となる

Langevin方程式(7)

$$\frac{du}{dt}+\frac{f}{m}u=\frac{2}{m}kT$$ は $$\frac{du}{dt}=-\frac{f}{m}u+\frac{2}{m}kT$$ として

変数を分離すると $$\int\frac{1}{-\frac{f}{m}u+\frac{2}{m}kT}du=\int dt$$

$$\int\frac{1}{ax+b}dx=\frac{1}{a}\log|ax+b|$$ を使うと $$-\frac{m}{f}\log\left|-\frac{f}{m}u+\frac{2}{m}kT\right|=t+C$$

さらに $$\log\left|-\frac{f}{m}u+\frac{2}{m}kT\right|=-\frac{f}{m}t+C'$$

Langevin方程式(8)

$$\log\left|-\frac{f}{m}u+\frac{2}{m}kT\right|=-\frac{f}{m}t+C'$$ から

$$C''e^{-\frac{f}{m}t}=-\frac{f}{m}u+\frac{2}{m}kT$$ これに $\frac{m}{f}$ をかけて

$$u=C'''e^{-\frac{f}{m}t}+\frac{2}{f}kT$$

Langevin方程式の答え

Einsteinの関係式

$u = C'''e^{-\frac{f}{m}t} + \frac{2}{f}kT$

Langevin方程式の答え

ここで u は $\frac{d<x^2>}{dt}$ ，すなわち分散の

時間変化率であったことを思い出し，

$e^{-\frac{f}{m}t}$ が十分小さくなった時を考えると $< x^2 >= \frac{2}{f}kTt$

一方，拡散方程式の解と正規分布を比べることによって得られた

$2Dt = s^2$ から $2Dt = s^2 =< x^2 >= \frac{2}{f}kTt$ すなわち

$D = \frac{kT}{f}$

これをアインシュタインの関係式という

粘性fとは(1)

$D=\frac{l^2}{2t_c}$ と $\frac{1}{2}m<(\frac{dx}{dt})^2>=\frac{1}{2}kT$ を使って $D=\frac{kT}{f}$

からfを求める．まず $kT=m<(\frac{dx}{dt})^2>$ なので

$$D=\frac{l^2}{2t_c}=\frac{mv^2}{f}=\frac{mvt_cvt_c}{ft_ct_c}=\frac{ml^2}{ft_ct_c}$$

$$\frac{l^2}{2t_c}=\frac{ml^2}{ft_ct_c}$$

$\frac{1}{2}=\frac{m}{ft_c}$ と変形でき，最終的に $f=\frac{2m}{t_c}$ となる

粘性fとは(2)

元のLangevin方程式 $m\frac{d^2x}{dt^2}=X-f\frac{dx}{dt}$ に $f=\frac{2m}{\tau_c}$

を代入すると $m\frac{d^2x}{dt^2}=X-\frac{2m}{\tau_c}\frac{dx}{dt}$ これを

$\frac{d^2x}{dt^2}=\frac{1}{m}X-\frac{2}{\tau_c}\frac{dx}{dt}$ さらに $\frac{dv}{dt}=\frac{1}{m}X-\frac{2}{\tau_c}v$ と変形して

力を加えてもある一定の流れの速さにしかできない
という粘性をイメージすると

粘性fとは(3) は力Xによって

$$\frac{dv}{dt}=\frac{1}{m}X-\frac{2}{t_c}v$$ で加速度 $\frac{dv}{dt}$ を0とした

$$0=\frac{1}{m}X-\frac{2}{t_c}v$$ は

力を加えてもある一定の
流れの速さにしかできない

ということを表していると考えると，これを変形した

$$\frac{1}{m}X=\frac{2}{t_c}v$$

$$Xt_c=2vm$$

衝突から衝突までの間に受けた
力積により増加した運動量が
衝突により一気に失われる

ということを表している.