分子コンピューティング概論

萩谷 昌己
東京大学
大学院情報理工学系研究科
コンピュータ科学専攻

生体分子コンピューティングとは

- 生体分子＝情報処理装置
  - 化学反応の自律的制御 ⇒ 自身にコード化
    - 微小・省エネルギー
    - 超並列
    - 分子の物理化学的性質
- 分子コンピューティングの目標
  - 分子反応の持つ潜在的計算能力の理学的解明
  - 分子反応に基づく新しい計算機能の工学的実現
- 参考文献
  - 萩谷, 横森編: DNAコンピュータ, 培風館, 2001.
  - 萩谷編著: 分子コンピュータの現状と展望 – 分子プログラミングへの展開, サイエンス社, 2004

分子コンピューティングの目標

- 分子反応の持つ計算能力の解析と
- その応用
  - 分子による計算を活用した分子計測技術
    ⇒ バイオテクノロジーへの応用
  - プログラムされた自己組織化や分子マシン
    ⇒ ナノテクノロジーへの応用
  - 分子による進化的計算
    ⇒ 分子進化工学への応用
- 分子反応に基づく新しい計算パラダイム

関連分野(生物と情報)

	なまもの	ソフトウェア
分析	分子生物学	バイオインフォマティクス 数理生物
合成	生体分子コンピューティング 分子進化工学	進化的計算 人工生命

進化⊂計算

関連分野(分子)

- 分子エレクトロニクス
  - 分子素子を用いた電子回路
    (既存の計算パラダイム)
  - 分子回路の構成技術(ナノテクノロジー)としての
    分子コンピューティング
- ナノテクノロジー
- 超分子化学
- 量子コンピューティング
- 光コンピューティング
- 分子生物学・バイオテクノロジー
- 分子進化工学

萩谷研ウェット実験室

- 工学部9号館501号室
- 工学部9号館の耐震工事が
  3月末にようやく終わる。
- 4月より、実験再開。
  - 4月19日大掃除


National
ハイビーム電球
150
National
レフ電球
ホワイト


TOMY

分子コンピューティング概論：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

分子コンピューティング概論：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

分子反応の持つ計算能力の解析

- 様々な計算モデル
  - 分子間の反応 vs. 分子内の反応
  - 液相 vs. 固相
  - 試験管、膜、細胞
  - 他律的 vs. 自律的
- （計算モデルの）計算能力の解析
  - 計算可能性
  - 計算量 --- 時間、領域
  - エラーや収量 --- 確率的な解析
  - 現実の分子反応により忠実な解析へ

様々な計算モデルとその解析

- Adleman-Lipton
  - DNAの選択的ハイブリダイゼーションを利用した解の候補のランダムな生成
  - データ並列計算による解の抽出
  - Suyama --- dynamic programming
  - Sakamoto-Hagiya --- SAT Engine
  - Head-Yamamura --- aqueous computing
- Seeman-Winfree
  - 各種形態のDNA分子の自己組織化（self-assembly）
  - 自己組織化による計算

様々な計算モデルとその解析

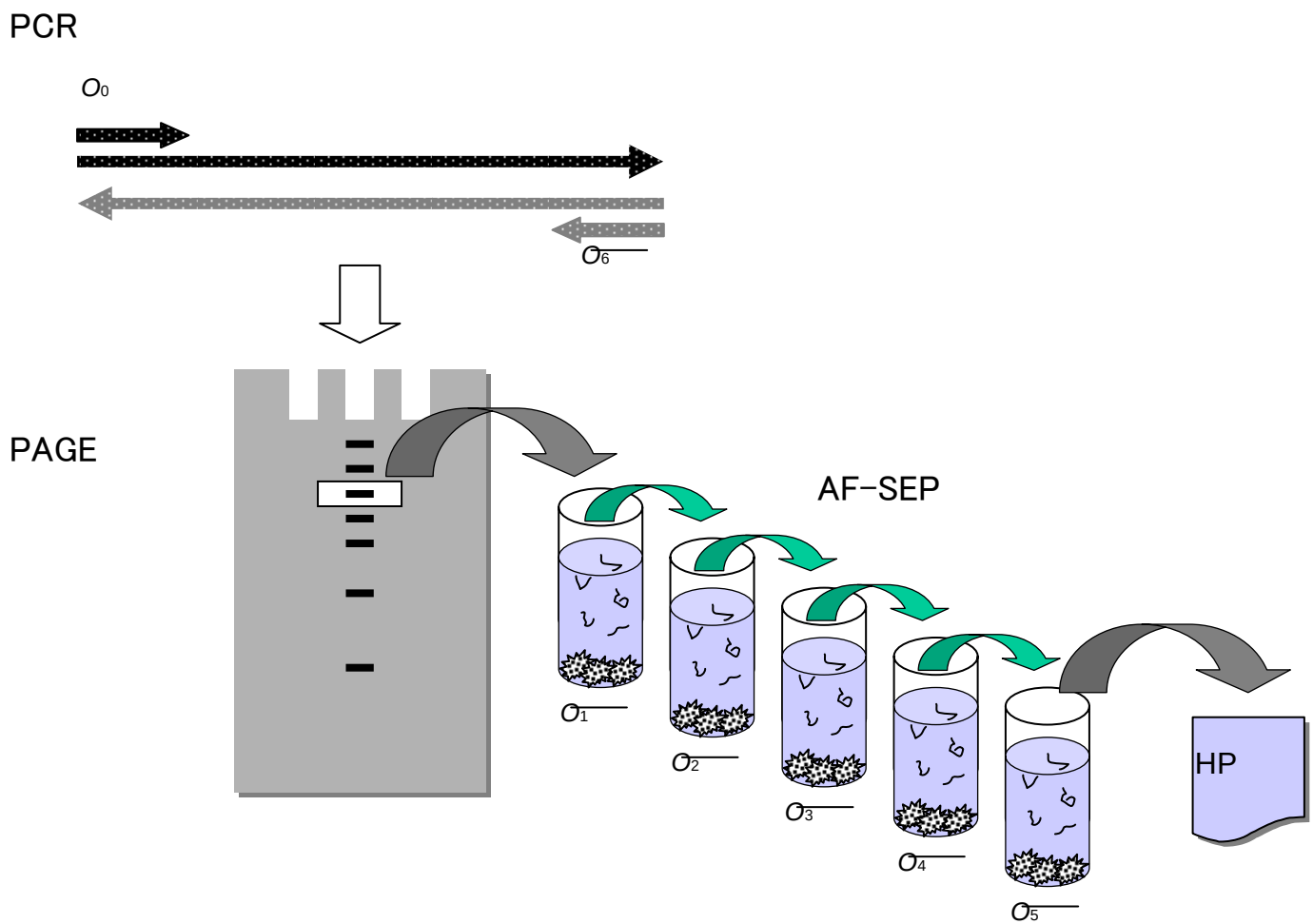
- Head
  - 遺伝子の組み換えによる言語の生成
- Ogihara-Ray
  - ブール回路の並列計算
- Hagiya-Sakamoto
  - 状態機械（Whiplash）
- Shapiro
  - 有限オートマトン

分子マシンのところで

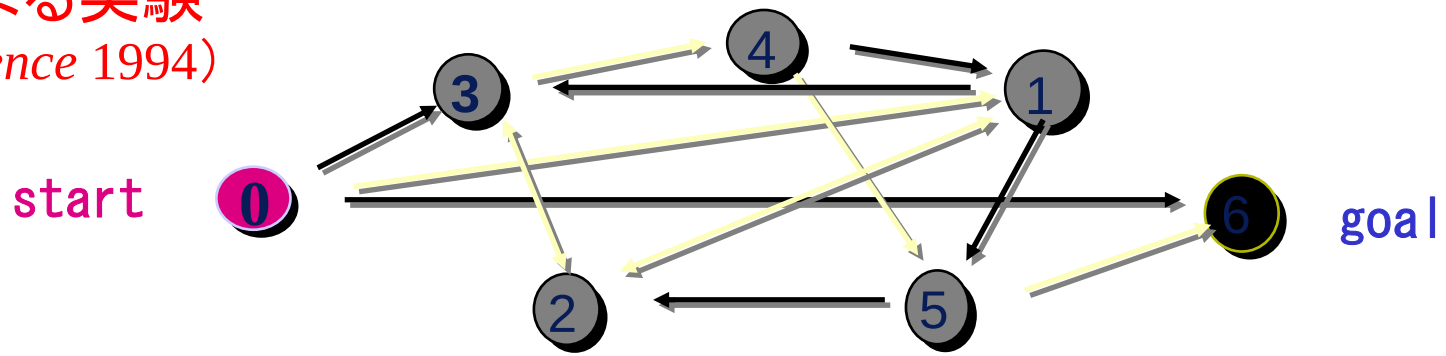
Adleman-Liptonパラダイム

- Adleman (*Science* 1994)
  - ハミルトン経路問題をDNAを用いて解く。
- Lipton, *et al.*
  - SAT問題をDNAを用いて解く。
- 分子による超並列計算
  - 主として組み合わせ的最適化
  - DNAの自己会合によるランダムな生成
    - 解の候補 ＝ DNA分子
  - 生物学実験技術を駆使した解の抽出
- 現状ではバイオテクノロジーのベンチマーク
  - 遺伝子解析への応用を視野に入れた研究

Adlemanの最初のDNAコンピュータ



Adleman
による実験
（*Science* 1994）



頂点３
ATCGATCG

頂点４
TAAGATA

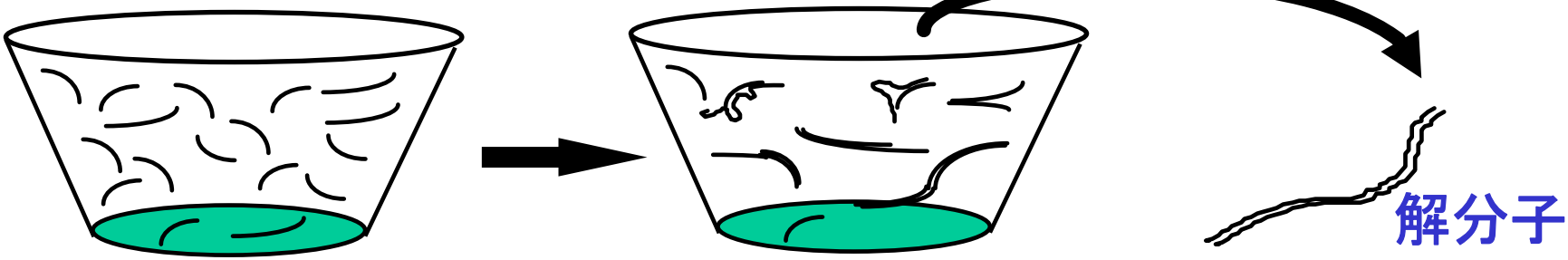
TAGCATTC

有向辺３⟶４

ハイブリダイゼーション

ATCGATCG•TAAGATA
TAGCATTC

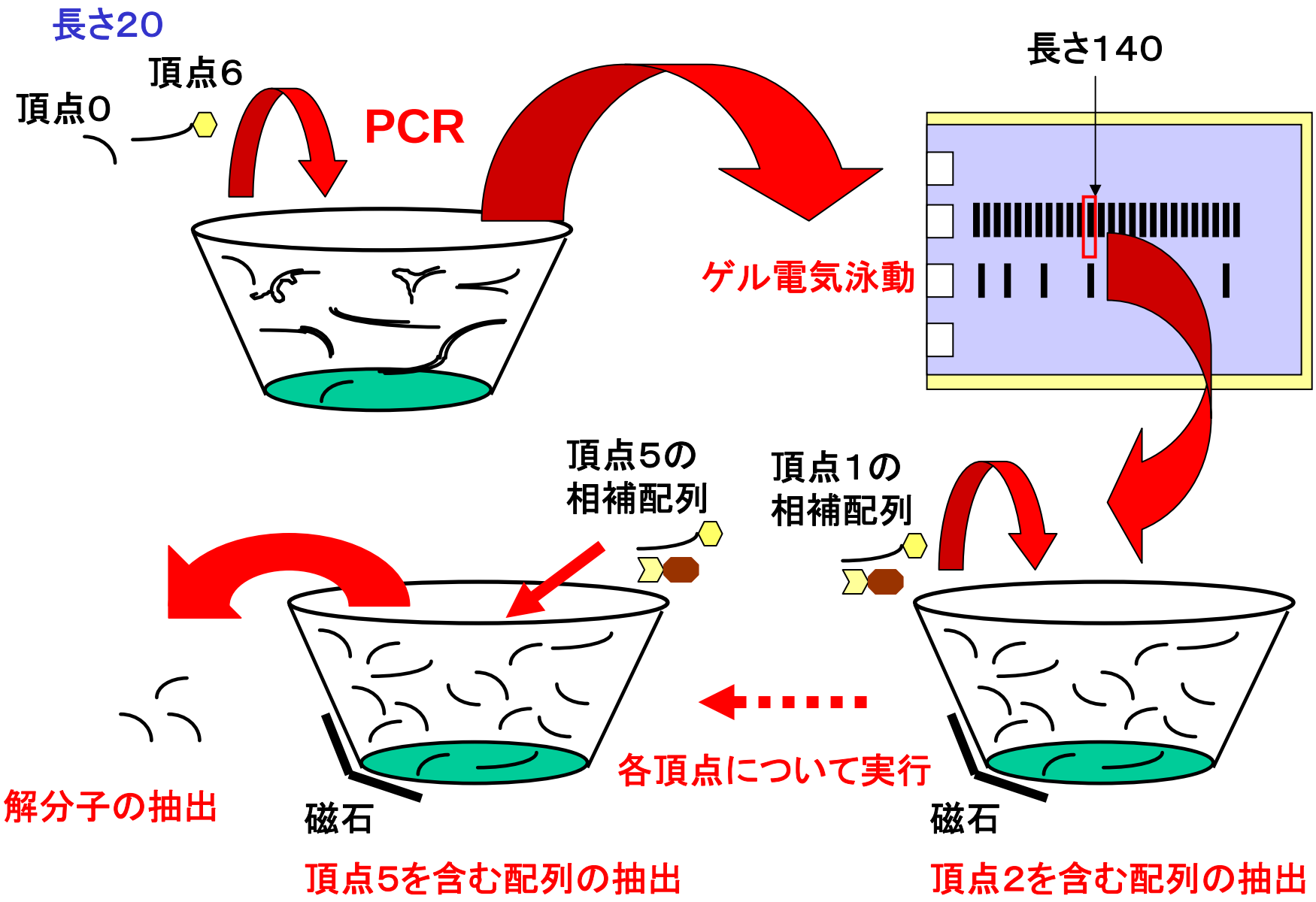
抽出・検知



解分子

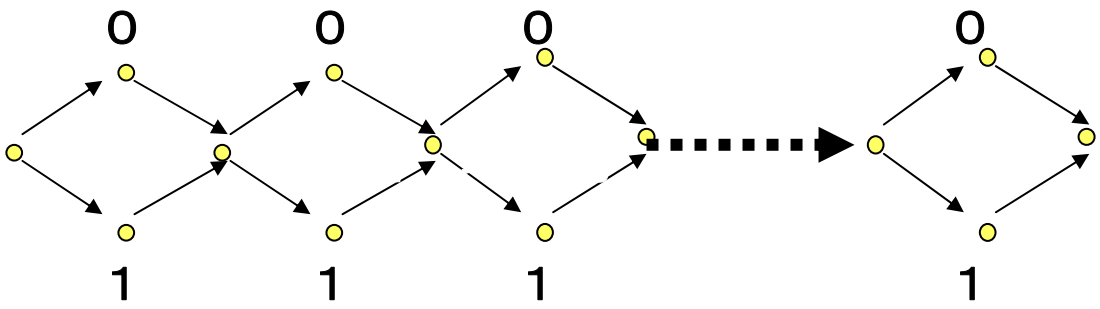
実験操作へ

解の抽出・検知


長さ20
頂点6
頂点0
PCR
長さ140
ゲル電気泳動
頂点5の
相補配列
頂点1の
相補配列
各頂点について実行
解分子の抽出
磁石
磁石
頂点5を含む配列の抽出
頂点2を含む配列の抽出

Adleman-Liptonパラダイム

全ての解候補の生成



全ての
代入の生成
(Lipton 1995,
充足可能性問題)

解の検査と抽出

T_i：文字列の多重集合（試験管）

[Separate命令] T_2＝＋（T_1，ｓ）：ｓを含む配列の抽出

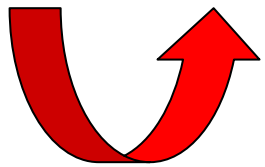
T_2＝－（T_1，ｓ）：ｓを含まない配列の抽出

[Merge命令] T_3＝T_1 ∪ T_2：T_1 と T_2 の混合

[Amplify命令] (T_2, T_3)＝T_1：T_1 の増幅（コピー）

解の検出

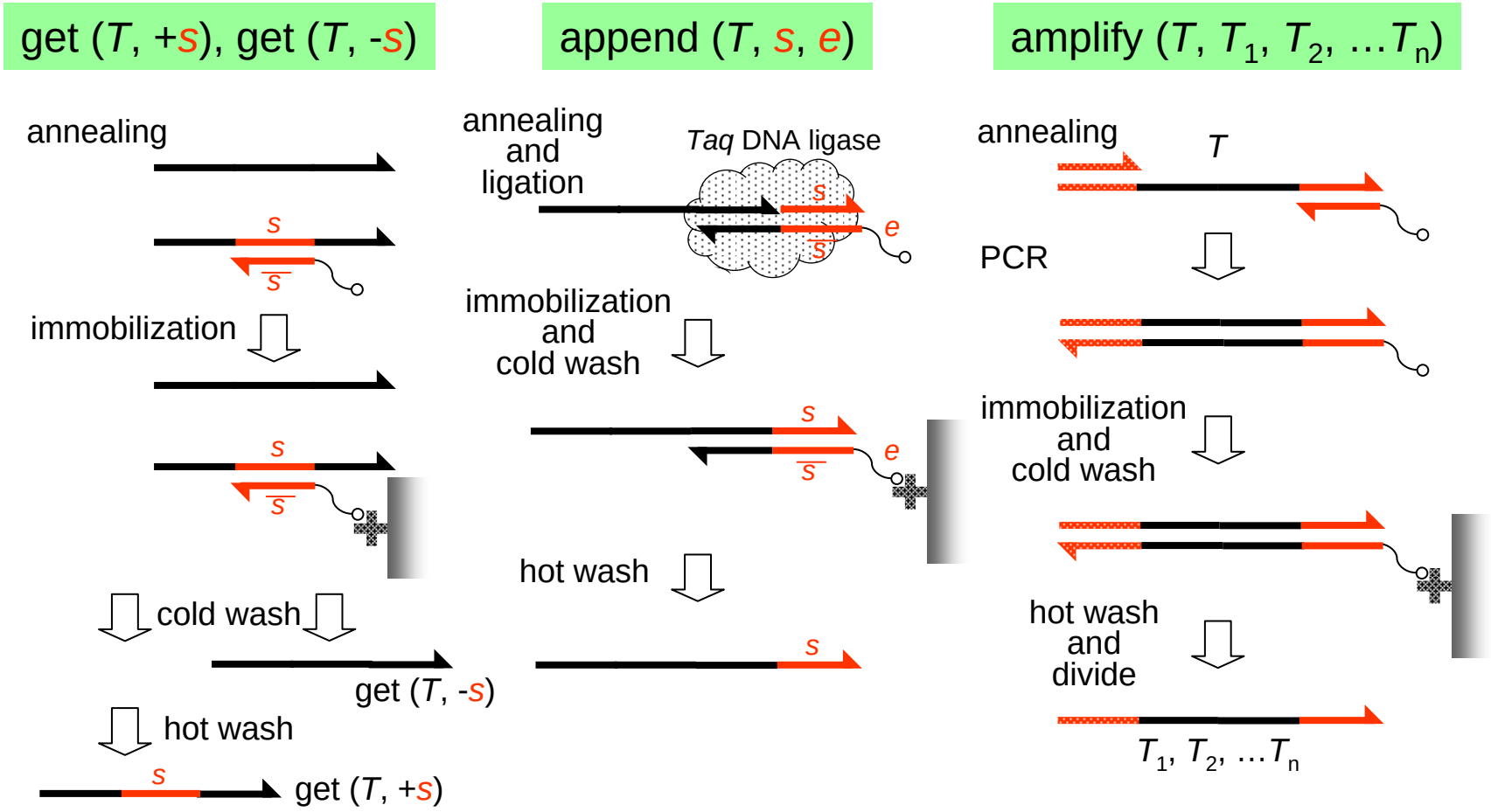
[Detect 命令]



SuyamaのDNAコンピュータ

- “counting”（Ogihara and Ray）
  - $O(2^{0.4n})$ molecules for n-variable 3-SAT
- “dynamic programming” (Suyama)
- 生成と選択の繰り返し
  - 解の候補の部分的な生成
  - 解の候補の選択
- 指数オーダーには変わりはないが、
  $O(2^{0.4n})$ は $O(2^n)$ よりずっと少ない。
- 固相法
  - 磁気ビーズによるアフィニティ・セパレーション
  - 自動化に適している ⇒ Robot！

基本演算とその実装


get (T, +s), get (T, -s)
append (T, s, e)
amplify (T, T1, T2, …Tn)
annealing
s
s
immobilization
s
s
cold wash
get (T, -s)
hot wash
s
get (T, +s)
annealing and ligation
Taq DNA ligase
s
e
s
immobilization and cold wash
s
e
s
hot wash
s
annealing
T
PCR
immobilization and cold wash
hot wash and divide
T1, T2, …Tn

3CNF-SAT を解く
DP的なDNA アルゴリズム

function dna3sat$(u_1, v_1, w_1, \ldots, u_m, v_m, w_m)$
begin
  $T_2 = \{X_1^T X_2^T, X_1^F X_2^T, X_1^T X_2^F, X_1^F X_2^F\}$;
  for $k = 3$ **to** n **do**
    amplify(T_{k-1}, T_w^T, T_w^F);
    for $j = 1$ **to** m **do**
      if $w_j = x_k$ **then**
        T_w^F = getuvsat(T_w^F, u_j, v_j);
      end
      if $w_j = \neg x_k$ **then**
        T_w^T = getuvsat(T_w^T, u_j, v_j);
      end
    end
    $T^T = \text{append}(T_w^T, X_k^T, \overline{X_{k-1}^{T/F} X_k^T})$; $T^F = \text{append}(T_w^F, X_k^F, \overline{X_{k-1}^{T/F} X_k^F})$;
    $T_k = \text{merge}(T^T, T^F)$;
  end
  return detect(T_n);
end

function getuvsat(T, u, v)
begin
  $T_u^T = \text{get}(T, +X_u^T)$; $T'^F_u = \text{get}(T, -X_u^T)$;
  $T_u^F = \text{get}(T'^F_u, +X_u^F)$; /* *can be omitted* */
  $T_v^T = \text{get}(T_u^F, +X_v^T)$;
  $T^T = \text{merge}(T_u^T, T_v^T)$;
  return T^T;
end

Number of operations

$$(n-2) \times (\text{amplify} + 2 \times \text{append} + \text{merge}) + m \times (3 \times \text{get} + \text{merge})$$

3CNF-SAT問題とその解

Problem : 4 variables, 10 clauses

$$(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge$$
$$(\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge$$
$$(x_1 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee \neg x_4) \wedge$$
$$(\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee x_3 \vee x_4) \wedge$$
$$(x_2 \vee \neg x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee x_4)$$

Solution :

YES

$$\{X_1^T X_2^T X_3^F X_4^F\}$$

3CNF-SATを解くDP的アルゴリズム

k のループ : k は変数の番号を動く

j のループ : j は節の番号を動く

if x_k が j 番目の節の3番目のリテラル **then**

1番目のリテラルも二番目のリテラルも

充足しない割り当ては削除する

$X_k^{\ F}$ を残りの割り当てに追加する

($\neg x_k$ が3番目のリテラルの場合も同様)

$k = 3$

x_3

$X_1^{\ T} X_2^{\ T}$ ⟶ $X_1^{\ T} X_2^{\ T} X_3^{\ F}$

~~$X_1^{\ F} X_2^{\ T}$~~ $(x_1 \vee \neg x_2 \vee x_3)$

$X_1^{\ T} X_2^{\ F}$ ⟶ $X_1^{\ T} X_2^{\ F} X_3^{\ F}$

~~$X_1^{\ F} X_2^{\ F}$~~ $(x_1 \vee x_2 \vee x_3)$

3CNF-SATを解くDP的アルゴリズム

k のループ : k は変数の番号を動く

　j のループ : j は節の番号を動く

　　if x_k が j 番目の節の3番目のリテラル **then**

　　1番目のリテラルも二番目のリテラルも

　　充足しない割り当ては削除する

$X_k^{\ F}$ を残りの割り当てに追加する

($\neg x_k$ が3番目のリテラルの場合も同様)

$k = 3$

$\neg x_3$

~~$X_1^{\ T} X_2^{\ T}$~~

$X_1^{\ F} X_2^{\ T}$ $\longrightarrow$ $X_1^{\ F} X_2^{\ T} X_3^{\ T}$ $(\neg x_1 \vee \neg x_2 \vee \neg x_3)$

~~$X_1^{\ T} X_2^{\ F}$~~

$X_1^{\ F} X_2^{\ F}$ $\longrightarrow$ $X_1^{\ F} X_2^{\ F} X_3^{\ T}$ $(\neg x_1 \vee x_2 \vee \neg x_3)$

3CNF-SATを解くDP的アルゴリズム

k のループ : k は変数の番号を動く

j のループ : j は節の番号を動く

if x_k が j 番目の節の3番目のリテラル **then**

1番目のリテラルも二番目のリテラルも

充足しない割り当ては削除する

$X_k^{\ F}$ を残りの割り当てに追加する

($\neg x_k$ が3番目のリテラルの場合も同様)

$k = 4$

x_4

~~$X_1^{\ F} X_2^{\ T} X_3^{\ T}$~~ $(\neg x_2 \vee \neg x_3 \vee x_4)$

~~$X_1^{\ F} X_2^{\ F} X_3^{\ T}$~~ $(x_2 \vee \neg x_3 \vee x_4)$

$X_1^{\ T} X_2^{\ T} X_3^{\ F}$ ⟶ $X_1^{\ T} X_2^{\ T} X_3^{\ F} X_4^{\ F}$

~~$X_1^{\ T} X_2^{\ F} X_3^{\ F}$~~ $(x_2 \vee x_3 \vee x_4)$

10-variable and 43-clause instance of 3SAT

$$
\begin{aligned}
&(\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_3 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4) \\
&\wedge (x_1 \vee x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee \neg x_5) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_5) \\
&\wedge (\neg x_1 \vee \neg x_3 \vee \neg x_5) \wedge (\neg x_2 \vee \neg x_4 \vee x_6) \wedge (\neg x_2 \vee x_3 \vee x_6) \\
&\wedge (x_2 \vee \neg x_3 \vee x_6) \wedge (\neg x_1 \vee \neg x_5 \vee \neg x_6) \wedge (x_2 \vee \neg x_6 \vee x_7) \\
&\wedge (x_1 \vee x_5 \vee x_7) \wedge (\neg x_1 \vee \neg x_5 \vee \neg x_7) \wedge (x_5 \vee \neg x_6 \vee \neg x_7) \\
&\wedge (x_1 \vee \neg x_2 \vee \neg x_7) \wedge (x_1 \vee x_6 \vee \neg x_7) \wedge (\neg x_4 \vee x_6 \vee \neg x_7) \\
&\wedge (x_1 \vee x_4 \vee x_8) \wedge (\neg x_1 \vee x_5 \vee x_8) \wedge (x_2 \vee \neg x_3 \vee x_8) \\
&\wedge (x_1 \vee x_6 \vee x_8) \wedge (x_2 \vee x_5 \vee \neg x_8) \wedge (x_1 \vee x_4 \vee \neg x_8) \\
&\wedge (\neg x_3 \vee \neg x_5 \vee \neg x_8) \wedge (\neg x_2 \vee x_4 \vee x_9) \wedge (x_4 \vee x_7 \vee x_9) \\
&\wedge (x_1 \vee x_7 \vee x_9) \wedge (\neg x_4 \vee x_6 \vee \neg x_9) \wedge (\neg x_1 \vee x_3 \vee \neg x_9) \\
&\wedge (\neg x_2 \vee x_3 \vee \neg x_9) \wedge (x_1 \vee \neg x_7 \vee \neg x_9) \wedge (\neg x_2 \vee x_4 \vee \neg x_9) \\
&\wedge (\neg x_1 \vee x_5 \vee \neg x_9) \wedge (\neg x_4 \vee x_8 \vee x_{10}) \wedge (x_3 \vee \neg x_6 \vee x_{10}) \\
&\wedge (\neg x_2 \vee \neg x_7 \vee x_{10}) \wedge (x_3 \vee \neg x_4 \vee \neg x_{10}) \wedge (\neg x_5 \vee \neg x_6 \vee \neg x_{10}) \\
&\wedge (x_4 \vee x_5 \vee \neg x_{10}) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_{10}) \wedge (x_2 \vee x_8 \vee \neg x_{10}) \\
&\wedge (\neg x_1 \vee x_8 \vee \neg x_{10})
\end{aligned}
$$

DNA Computer Robot based on MAGTRATION™ (Prototype No.1)

DNAコンピュータのプログラミング

function $\mathrm{dna3sat}(u_1, v_1, w_1, \ldots, u_m, v_m, w_m)$
begin
  $T_2 = \{X_1^T X_2^T, X_1^F X_2^T, X_1^T X_2^F, X_1^F X_2^F\};$
  for $k = 3$ **to** n **do**
    $\mathrm{amplify}(T_{k-1}, T_w^T, T_w^F);$
    for $j = 1$ **to** m **do**
      if $w_j = x_k$ **then**
        $T_w^F = \mathrm{getuvsat}(T_w^F, u_j, v_j);$
      end
      if $w_j = \neg x_k$ **then**
        $T_w^T = \mathrm{getuvsat}(T_w^T, u_j, v_j);$
      end
    end
    $T^T = \mathrm{append}(T_w^T, X_k^T, \overline{X_{k-1}^{T/F} X_k^T});$
    $T^F = \mathrm{append}(T_w^F, X_k^F, \overline{X_{k-1}^{T/F} X_k^F});$
    $T_k = \mathrm{merge}(T^T, T^F);$
  end
  return $\mathrm{detect}(T_n);$
end

Pascal/C-level

```
[Instrument]
[Reset Counter] 0
[Home Position] 0
[MJ-Open Lid]
...
[Get1(0)]
[Get2(1)]
[Append(2)]
...
[Exit]
```

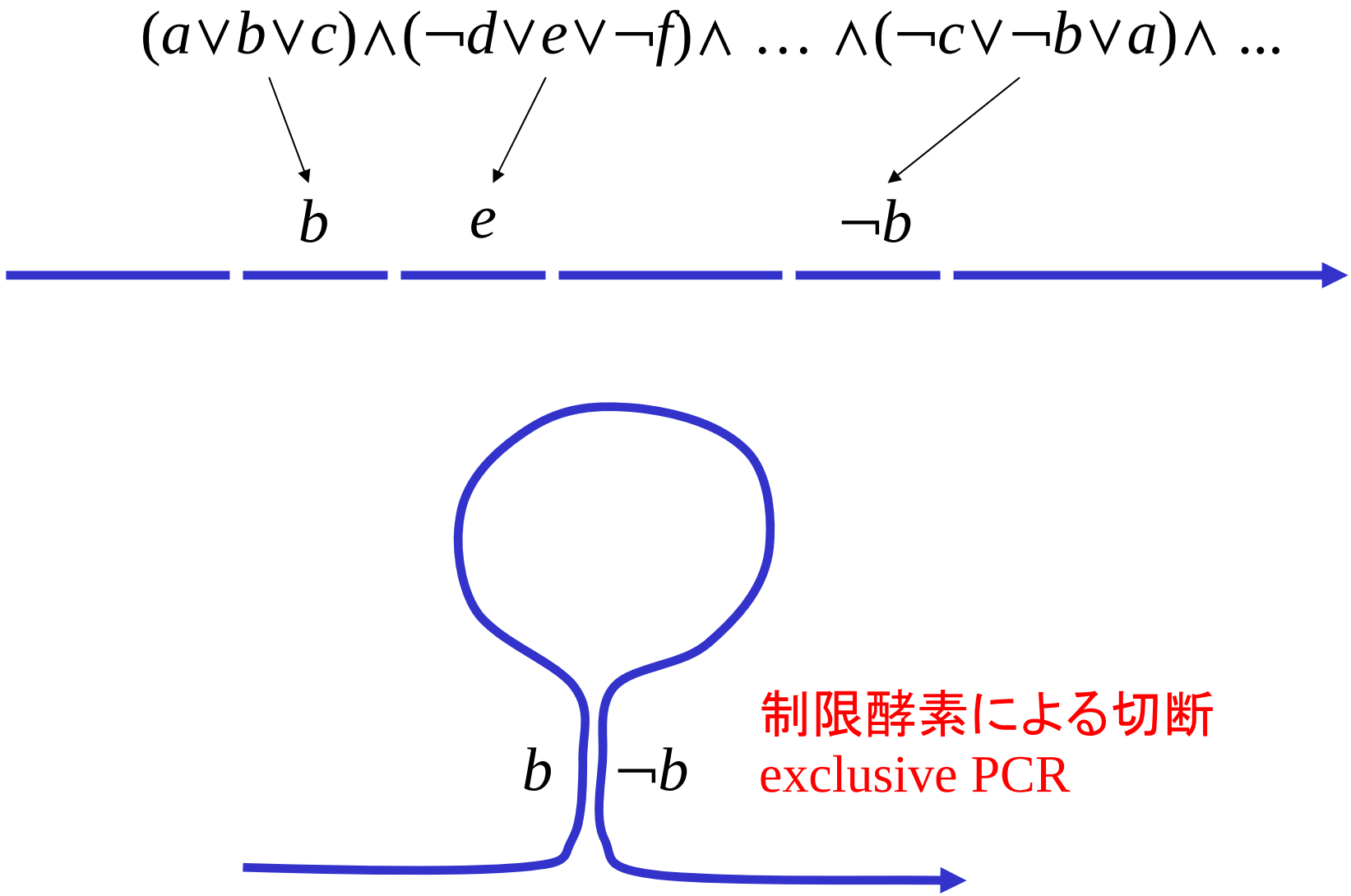
protocol-level

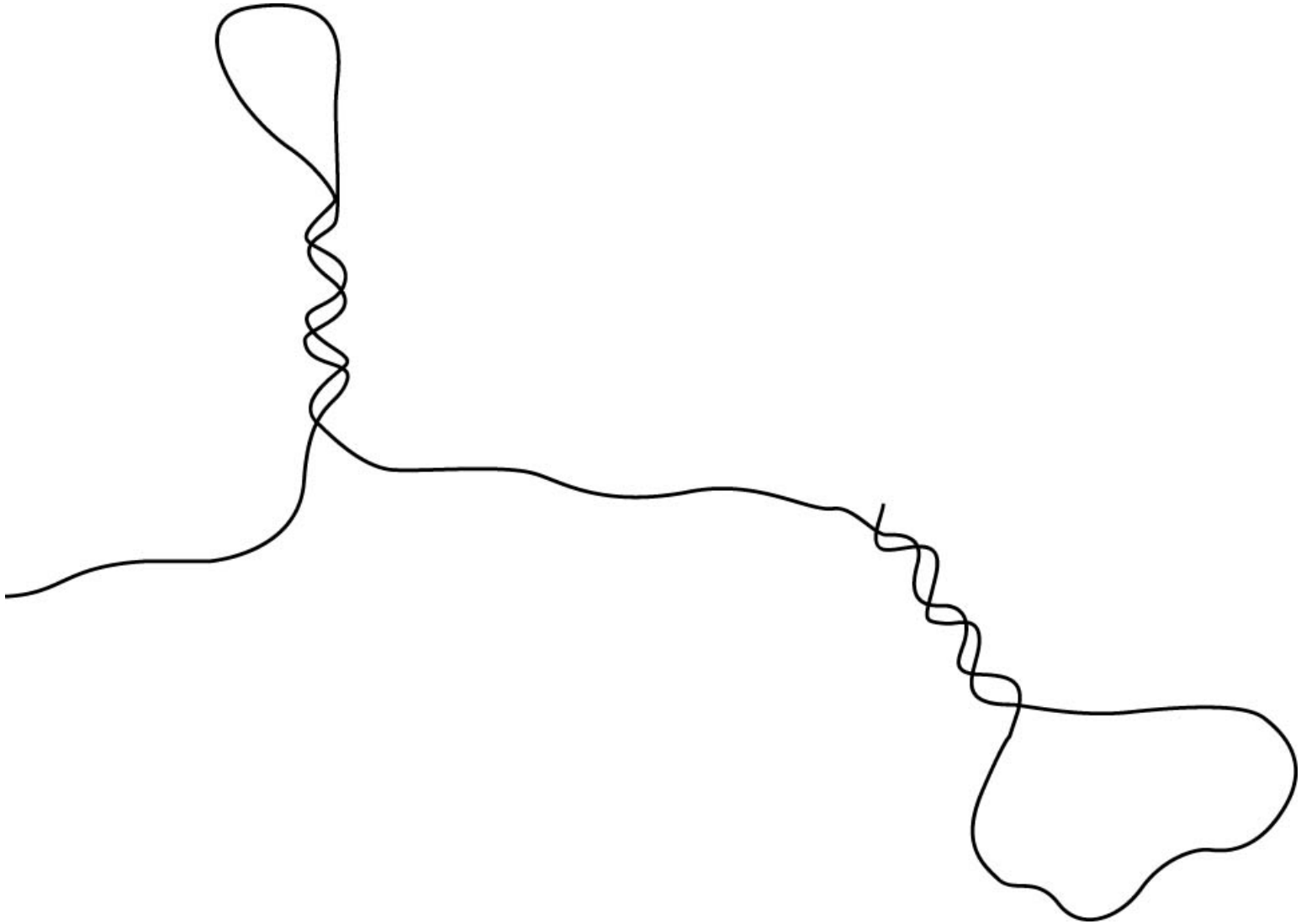
```
(1-1-4)    [MJ-Open Lid]
Do 2
  _SEND    "LID OPEN"
  Do       10
    _SEND  "LID?"
    Wait_msec        500
    _CMP_GSTR        "OPEN"
    IF_Goto  EQ  0   ;open
    Wait_msec  1000
  Loop
Loop
; Time out
End
;open
```

script-level

ヘアピン・エンジン（SATエンジン）

- Sakamoto *et al.*, *Science*, May 19, 2000.
- 特平11-165114
- DNAのヘアピン構造を利用した選択
  - ヘアピンの制限酵素切断
  - exclusive PCR
- 3-SAT
  - 各節から選んだリテラルから成る一本鎖 DNA
  - 相補的なリテラル ＝ 相補的配列
  - 矛盾したリテラルの選択 ⇒ ヘアピン
  - 6-variable 10-clause 3-SAT Problem
- SAT計算の本質的部分 ＝ ヘアピン形成
  - 節や変数の数によらないステップ数
  - Autonomous molecular computation


(a∨b∨c)∧(¬d∨e∨¬f)∧ … ∧(¬c∨¬b∨a)∧ …
b
e
¬b
b
¬b
制限酵素による切断
exclusive PCR



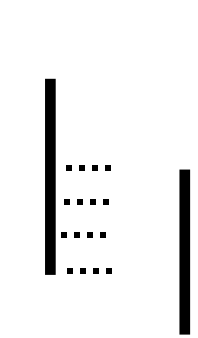
ヘアピン構造による選択

- 制限酵素による切断
  - リテラルを表す配列中に
    制限酵素サイトを挿入
- Exclusive PCR
  - 一般にPCRはヘアピンに対して
    増幅率が低い。
  - exclusive PCRでは、各サイクルで溶液を薄めることにより、ヘアピンと非ヘアピンの増幅率の差を大きくしている。
- 実験操作の数は変数や節の数に依存しない。

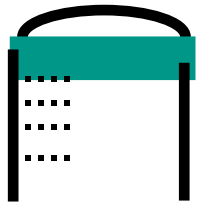
Adleman-Liptonパラダイムに関する現在のコンセンサス

- 電子コンピュータを凌駕するには程遠い。
  - スケールアップ問題
- 「分子が計算する」ことの
  proof of conceptとしては重要。
- 少なくとも、バイオテクノロジーの
  ベンチマークとして使うことができる。
- さらに、遺伝子計測への応用（Suyama）。

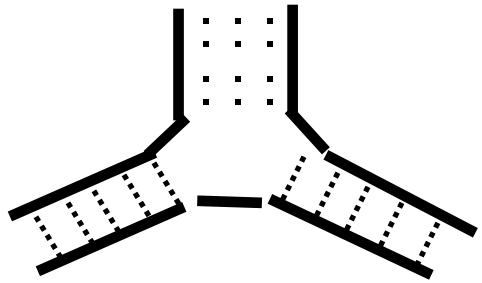
Seeman-Winfreeの DNAの自己組織化による計算



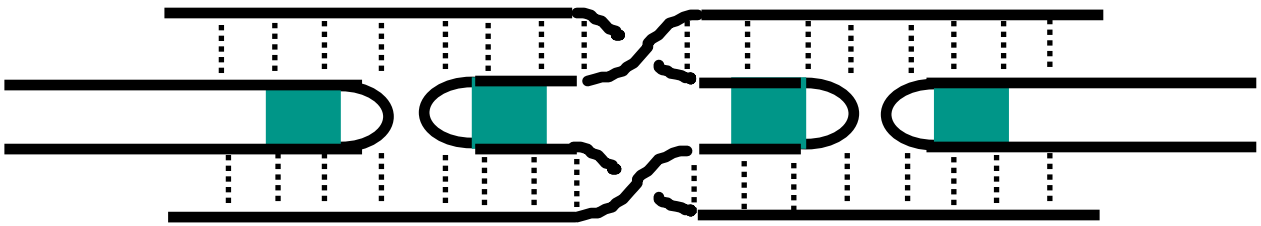
線形



ヘアピン



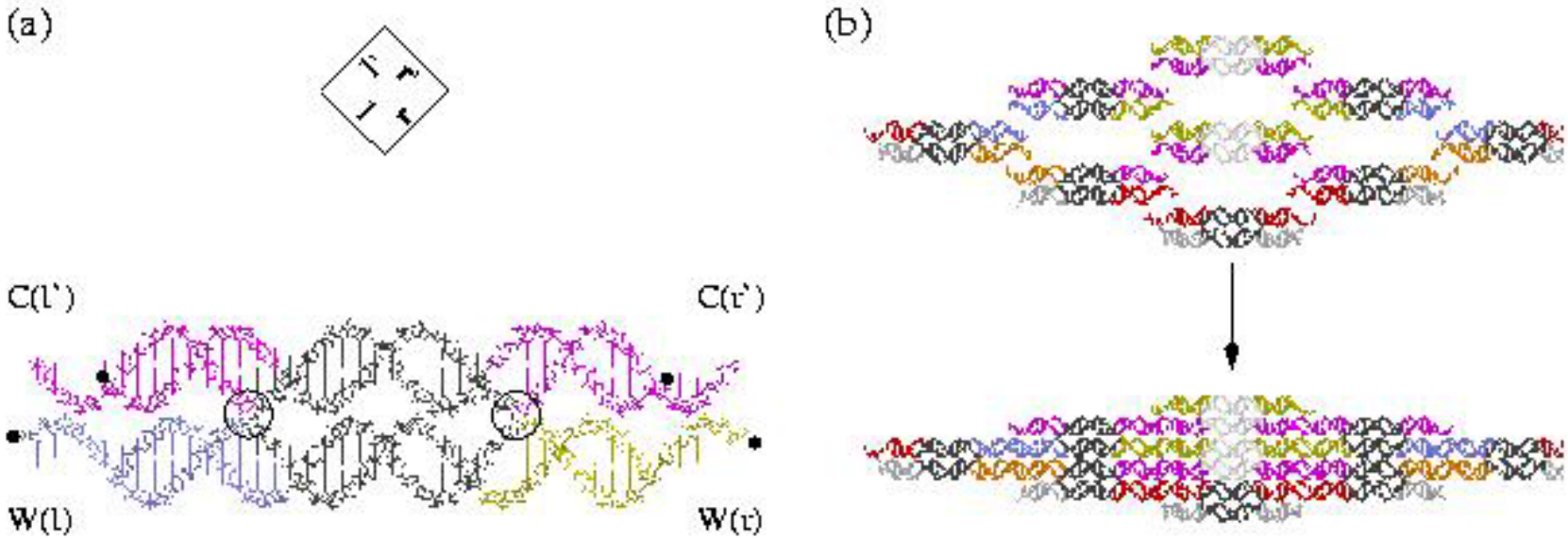
３分岐



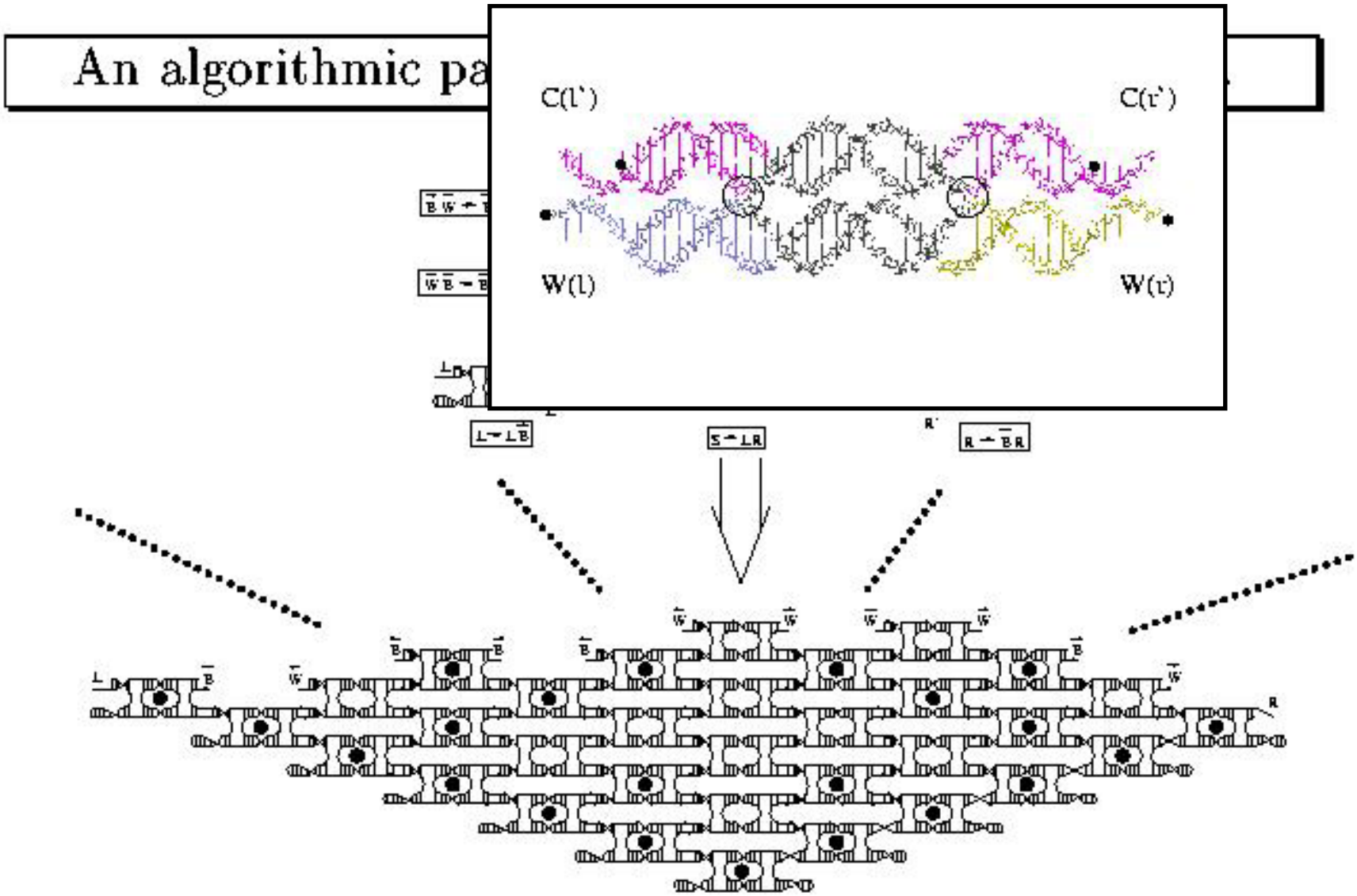
DX（ダブルクロスオーバー）

様々なDNA構造分子

The DNA representation of Wang tiles.



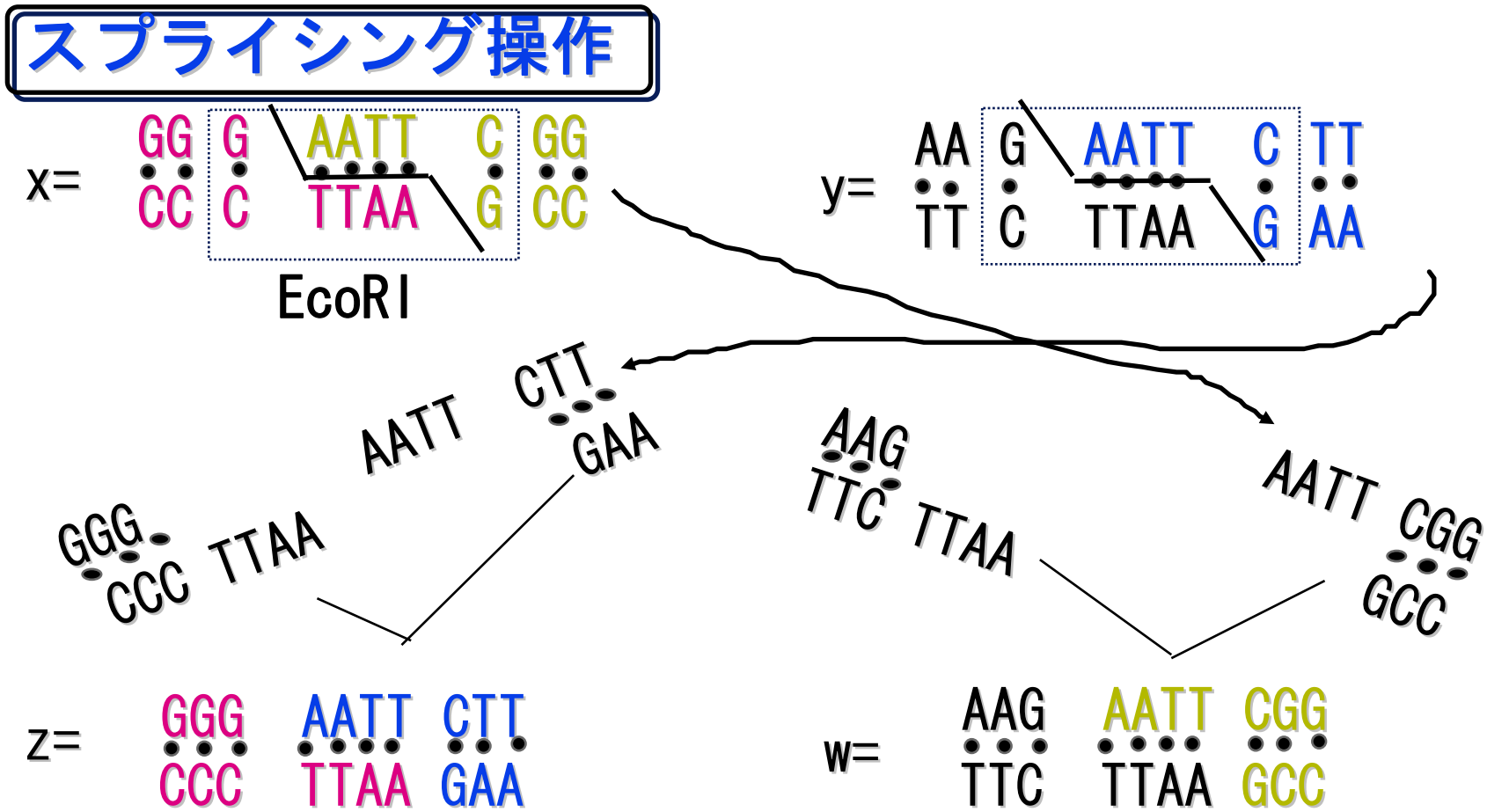
Winfreeのタイリング



Sierpinskiの三角形

Headの遺伝子組み換えによる計算

- 制限酵素とライゲースによる遺伝子組み換えの数学的モデル（スプライシング・モデル）



組み換えによる言語の生成

- スプライシング規則: $r = u_1\$u_2\#u_3\$u_4$
- $(x_1u_1u_2x_2, y_1u_3u_4y_2) \vdash_r (x_1u_1u_4y_2, y_1u_3u_2x_2)$
- R: スプライシング規則の集合
- A: 文字列の集合（公理）
- L: RとAによって生成される言語
  - $x \in A$ ならば、$x \in L$
  - $x, y \in L$ かつ $r \in R$ かつ $(x, y) \vdash_r (z,w)$ ならば、$z,w \in L$
- RとAが有限ならば、Lは正則になる。

分子コンピューティング概論：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

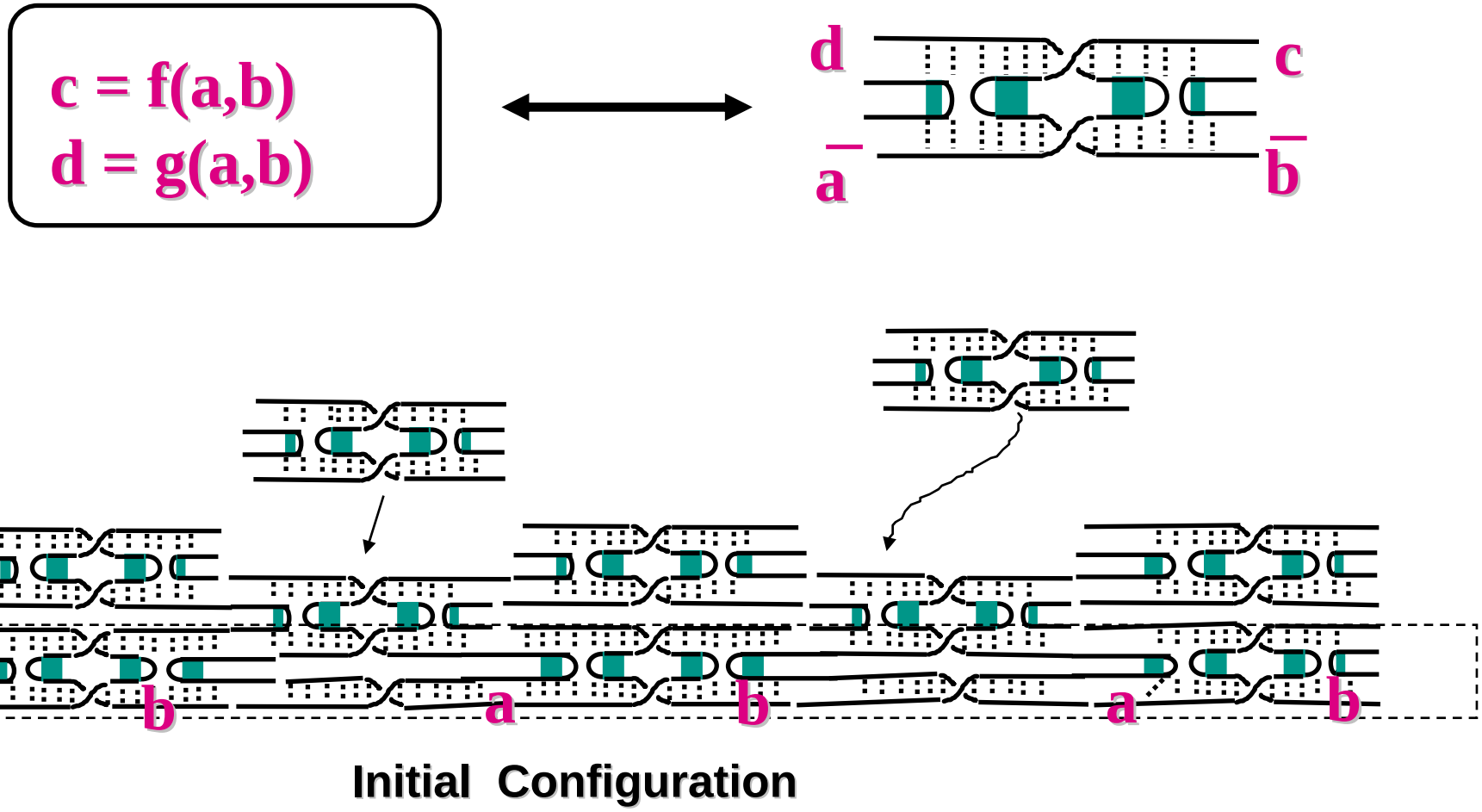
分子計算の計算可能性

- DNAの自己組織化の計算可能性
  - Winfreeの結果
- 遺伝子組み換えの計算可能性
  - スプライシング・モデルの様々な拡張

Winfreeの結果

- （線形）構造分子によって生成される言語族
  ＝正則言語族
- （線形＋ヘアピン＋３分岐）構造分子によって
  生成される言語族
  ＝文脈自由言語族
- （線形＋DX）構造分子によって
  生成される言語族
  ＝帰納的可算言語族
  ＝チューリング計算可能

Winfreeのモデルによる計算過程の例



１次元セルオートマトンの模倣

スプライシング･モデルの拡張

スプライシング・モデル
の生成能力　＜　正則言語族

（スプライシング）＋α？　＝　万能計算能力

いくつかの答え：　環状分子
マルチ試験管
時間差

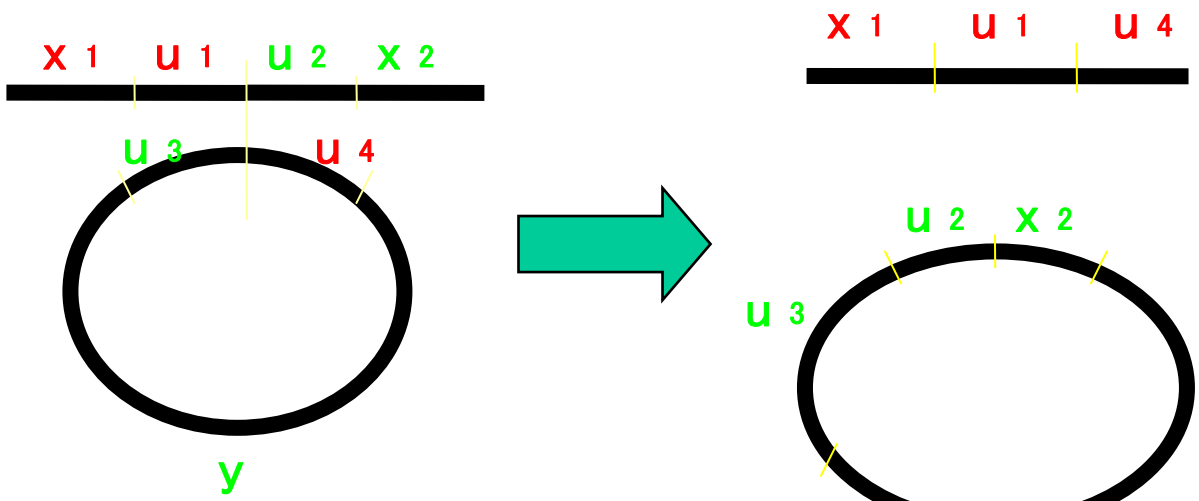
環状組換えシステム

プラス α として

環状文字列（環状DNA）を用いることを許す。

終端記号と非終端記号の区別を許す。

（ e.g. 大腸菌染色体とFプラスミドの組換え ）



スプライシング規則: u1$u2#u3$u4

分子コンピューティング概論：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

分子計算の計算量

- 時間
  - 実験操作の回数
  - 各操作に必要な時間
    - 分子の計算能力の解析にはより本質的
- 領域（＝並列度）
  - 分子の数
    - 最大
    - トータル
  - 分子の大きさ（長さ）
- トレードオフの解析が重要

計算量の解析（Adleman-Lipton）

- Reif (SPAA'95)
  - 非決定性チューリング機械において、入力長 n、空間 s、時間 $2^{O(s)}$ の計算は、我々のPAMモデルのもとで、$O(s)$ ステップのPA-Match操作と、$O(s \log s)$ ステップのそれ以外の操作により、長さ $O(s)$ の集合体を用いて実行することができる。
- Beaver (DNA1, 1995)
  - 多項式ステップの分子コンピュータは、PSPACEを計算する。
- Rooß and Wagner (I&C, 1996)
  - Liptonのモデルを用いて、ちょうど $P^{NP}=\Delta^P_2$ に属する問題を多項式時間で解くことができる。

Rooß and Wagner (I&C, 1996)

- Liptonのモデルを用いて、ちょうど $P^{NP}=\Delta^P_2$ に属する問題を多項式時間で解くことができる。
- BIO({UN,BX,IN},{EM})-P = $P^{NP}= \Delta^P_2$
  - UN: 合併（マージ）　　$T_3 = T_1 \cup T_2$
  - BX: ビット抽出（分離）

    $T_2 = +(T_1, s)$　　$T_2 = -(T_1, s)$
  - IN: 初期化（ランダム生成）
  - EM: 空テスト（検出）
  - -P: 多項式時間（ステップ）
  - P^{NP}: NPオラクルを用いた多項式時間

計算量の解析（自己組織化）

- Rothemund and Winfree (STOC 2000)
  - 任意の非増加非有界の計算可能関数 $f(N)$ が与えられたとき、無限個の N に対して、$f(N)$ より小さい個数のタイルを用いて、$N \times N$ の正方形を自己組織化により生成することができる。
- Winfree, Eng and Rozenberg (DNA6, 2000)
  - ストリング・タイルの線形自己組織化により、有限訪問チューリング機械（テープの各位置を訪れる回数の上限が存在するチューリング機械）の出力言語を生成することができる。

反応のエラーと収量

- 収量
  - 平衡　--- 平衡定数 (K)
  - 平衡に到達する時間 --- 反応速度 (k)
  - 例: $\mathrm{A} \leftrightarrow \mathrm{B}$

$$[\mathrm{B}] = (K/(1+K))(1-e^{-(k+k_{-1})\,t})$$

$$K = k/k_{-1}$$

- エラー
  - 例: ミス・ハイブリダイゼーション
  - エラーの確率は0にはならない。
- 確率的な解析

確率的な解析

- Karp, Keynon and Waarts (SODA'96)
  - 耐エラーのビット評価を達成するために必要な抽出操作の回数は $\Theta(\lceil \log_{\varepsilon} \delta \rceil \times \lceil \log_{\gamma} \delta \rceil)$ である。
- Kurtz (DNA2, 1996)
  - Adlemanの実験における経路生成の熱力学的解析
  - ハミルトン経路の生成に必要な時間 --- $\Omega(n^2)$
- Winfree (1998, Ph.D. Thesis)
  - DNAタイリングの熱力学的解析
- Rose, *et al.* (GECCO'99, *etc.*)
  - 計算論的非一貫性
    （ミス・ハイブリダイゼーションの熱力学的解析）

分子コンピューティング：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

分子システム設計の計算論的側面

- 「分子プログラミング」
- 分子の設計
  - DNAの場合 ＝ 配列設計
  - 構造 ⇒ 配列（inverse folding）
  - 自己組織化パターンの設計・分子マシンの設計
- 分子反応の設計
  - 反応条件や操作順序の設計
  - シミュレーション・ツール
- 分子マシン
  - 分子プログラミングの現在の目標の一つ

分子コンピューティング：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

配列設計

- 配列セットの評価
  - ミスハイブリダイゼーションの回避
    - ハミング距離
    - エネルギー計算 ⇒ mfold（Zuker）、Viennaパッケージ
  - 一様なT_m（融解温度、melting temperature）
- 配列セットの探索
  - 遺伝的アルゴリズム
  - 符号理論 --- 有田のテンプレート法
- 逆問題
  - 構造 ⇒ 配列（inverse folding）
  - Viennaグループ

テンプレート法

- Arita and Kobayashi, 2002

[AT]か [GC] の位置を全配列共通にする

（これをテンプレートと呼ぶ）

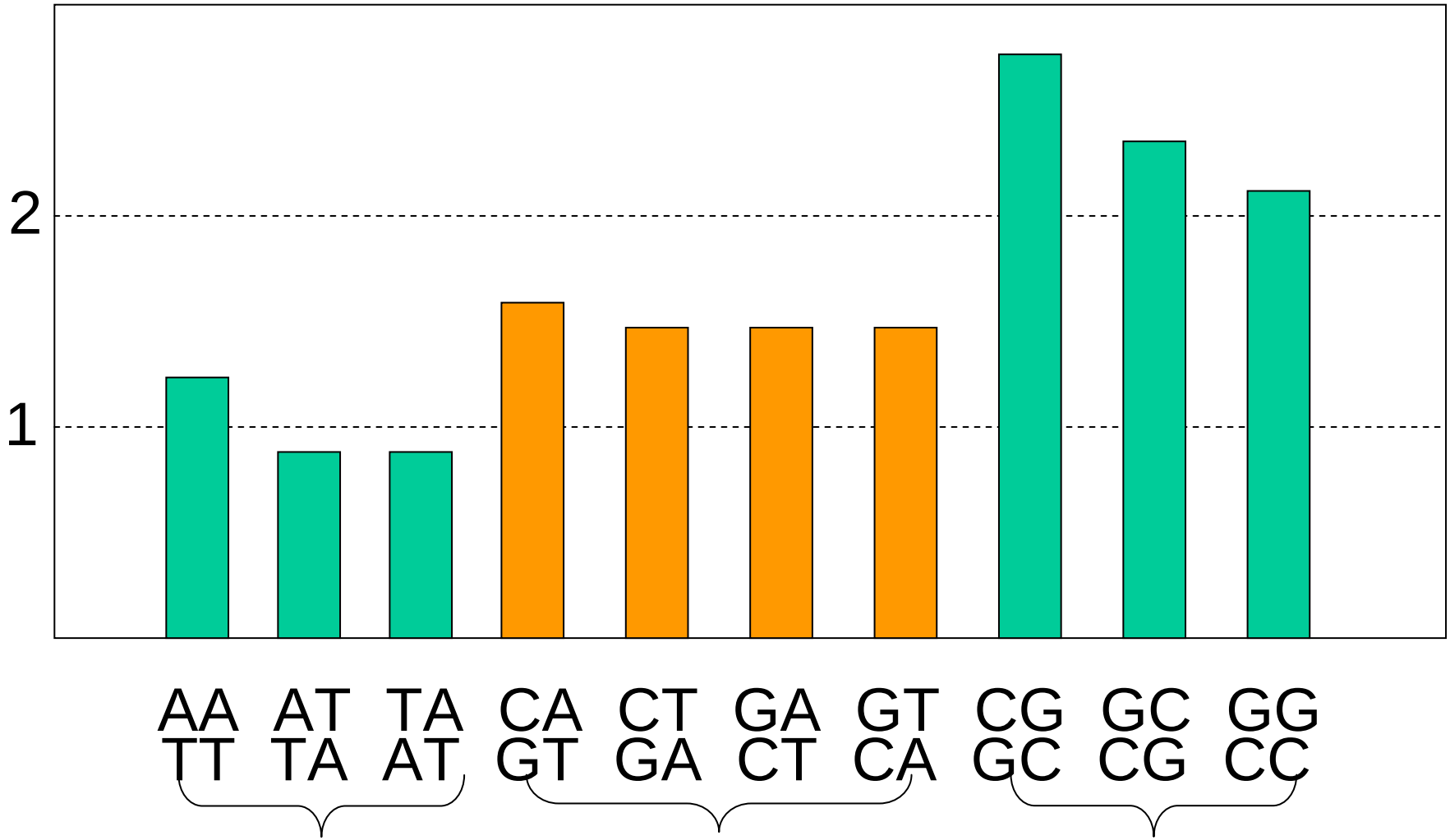
例．011010　より

ACCTGA，TGCTCA，TCGACA，etc.

→　こうすると、全配列の融解温度は揃う。

スタッキング・エネルギー

–ΔG kcal/mol (DNA/DNA) by Sugimoto et al.



ミスマッチを含むテンプレート

- 上手にテンプレートを選べば、
  シフト、リバースの際でも必ずミスマッチをもつ。

例：　１１０１００のとき

１１０１００	１１０１００	１１０１００　１１０１００
１１０１００	１１０１００	１１０１００

どうずらしても、連結した部分を考えても、
ミスマッチは最低２個存在

テンプレートの選び方

- テンプレート T を、以下のパターンに対して、最低 d 個のミスマッチを持つように選ぶ。

  - T^R
  - TT^R 、T^RT
  - TT、T^RT^R

注： T^RはTの逆配列。

T=110100ならば、T^R=001011

テンプレートの例

- 長さ 6 (ミスマッチ 2)

  110100　　（2^6個中）

- 長さ 11 (ミスマッチ 4)

  01110100100, 01011100010, 11000100101
  （2^{11}個中）

- 長さ 23 (ミスマッチ 9）

  01111010110011001010000,
  10110011001010000011110,
  11100000101001100110101　（2^{23}個中）

DNA配列の設計法

“テンプレート + 誤り訂正符号”

１１０１００ (テンプレート)
+０１００１１ (任意の符号語)
ＡＴＣＡＧＧ　(DNA配列)

誤り訂正符号は何でも利用可。

1. BCH 符号
2. Golay 符号
3. Hamming 符号　など。

Inverse Folding

- Viennaグループ
- McCaskillのアルゴリズムの利用
- コスト関数の最小化による配列の探索

$$\Xi(x) = E(x, \Omega) - G(x) = -RT \ln p$$

  - Ω：目的の構造
  - x: 配列
  - $E(x, \Omega)$: x における Ω の自由エネルギー
  - $G(x)$: 配列 x の集団自由エネルギー（McCaskill）
  - p: x における Ω の確率

分子コンピューティング：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

分子反応の設計

- 反応条件
  - 温度
  - 塩濃度
  - 時間
- 操作の順序
- シミュレーション
  - e-PCR
    - http://www.ncbi.nlm.nih.gov/genome/sts/epcr.cgi
  - VNA

VNA: 仮想 DNAのシミュレータ

- 抽象的だが、十分に物理的

  抽象モデルと実際の反応との間の橋渡し

  分子 — 仮想ストランドのハイブリッド

```
abcd
  ||
CDEF
```

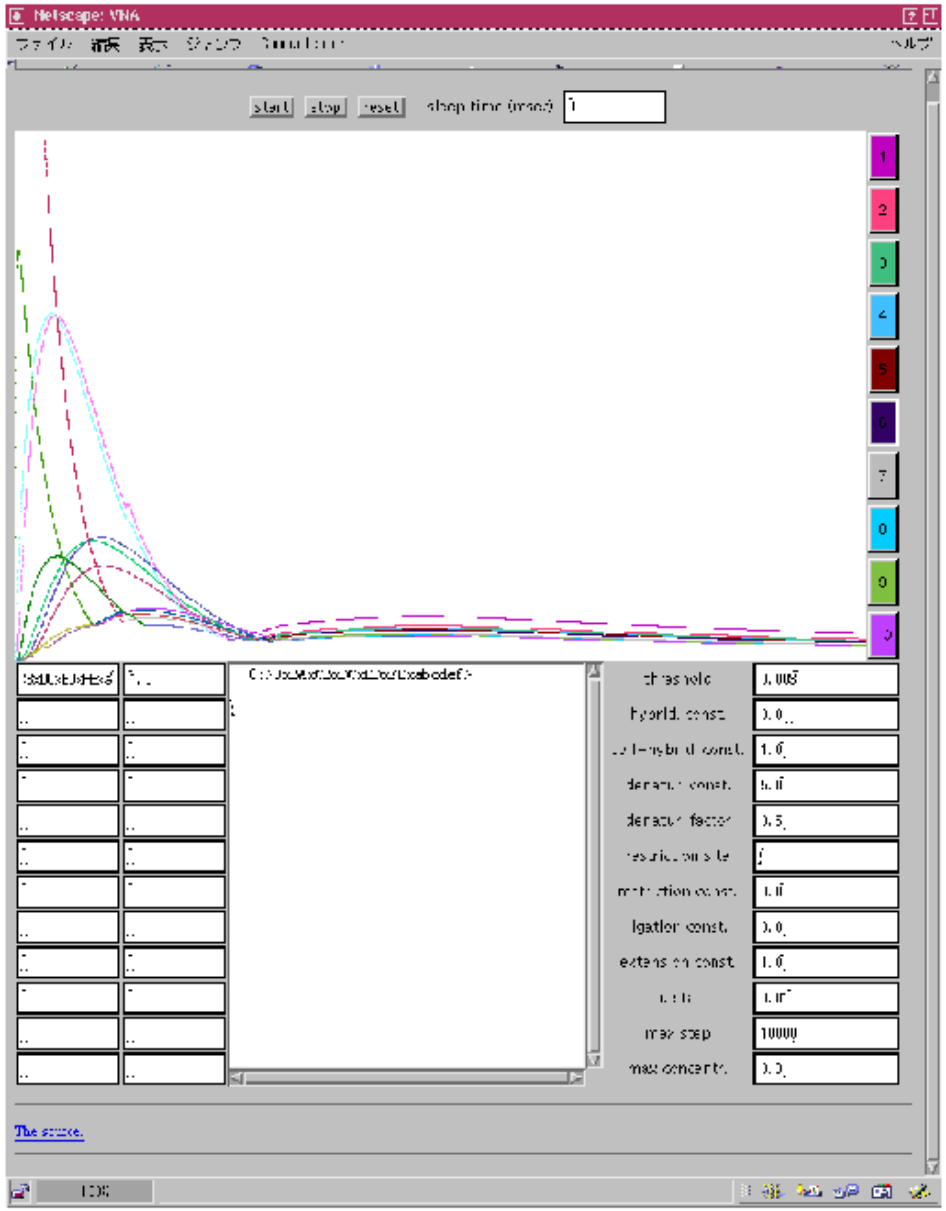
- 反応
  - hybridization
  - denaturation
  - restriction
  - ligation
  - self-hybridization
  - extension

VNA (続き)

- 目的
  - DNA計算の方式の実現可能性の検証
  - 生物学実験の妥当性の検証 (e.g., PCR実験)
  - 生物学実験の適切なパラメータの探索
- 実行例
  - OgiharaとRayによるブール式の計算
  - Winfreeによるdouble-crossover unitの生成
  - PCR実験
- 実装
  - Java ⇒ アプレットとして実行可能

VNA (続き)

- 方法
  - 組み合せ的数え上げ
  - 連続的シミュレーション (微分方程式)
) 融合
- 組み合せ的爆発の回避
  シミュレーション技術における貢献
  - 閾値
  - 確率的
- GAによるパラメータ探索
  - PCR実験の増幅率の最適化


Netscape: VNA
start
stop
reset
step time (msec)
1
2
3
4
5
6
7
8
9
0
threshold
0.003
hybrid. const.
0.0
1.0
0.5
0.0
0.0
0.0
0.0
1.0
0.0
10000
0.0
The source.

分子コンピューティング概論：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

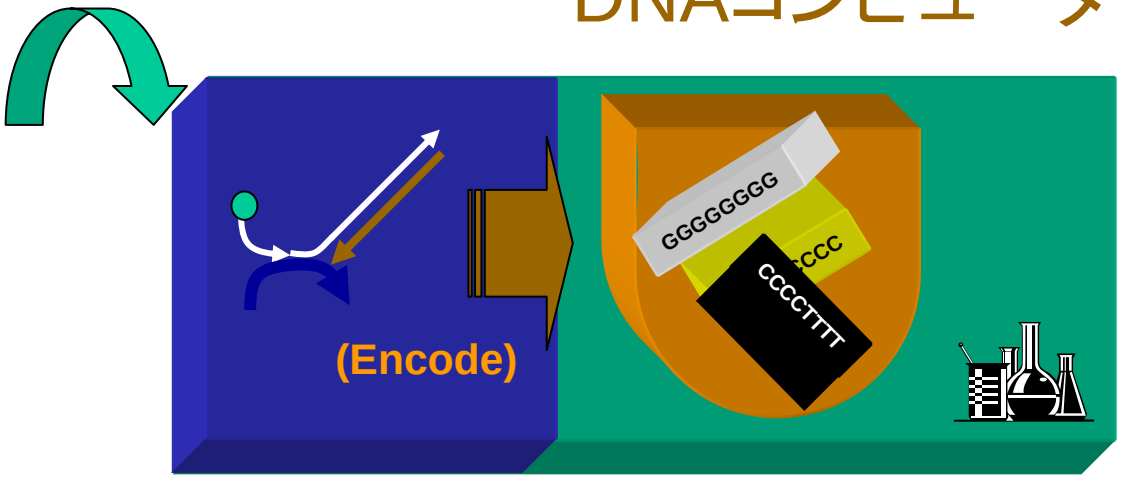
ブール式の学習の遺伝子発現解析への応用

- 知的DNAチップの提案
  - 3つ手法の組み合わせ
    - DNA符号化数（陶山）
    - DNA計算を用いたブール式の学習アルゴリズム
      （例からのブール式の帰納的学習）
    - DNAチップ技術
  - 知的DNAチップ実現方法
- 知的DNAチップの遺伝子発現解析への応用の提案

DNA計算を用いた遺伝子発現解析の情報処理

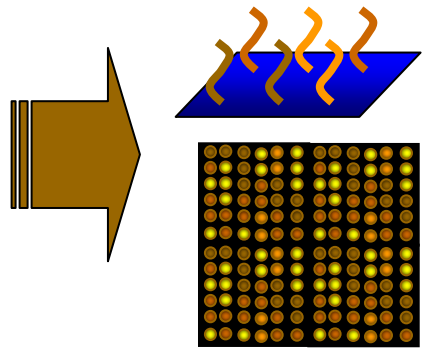
遺伝子の発現

直接入力:



DNAコンピュータ

出力:



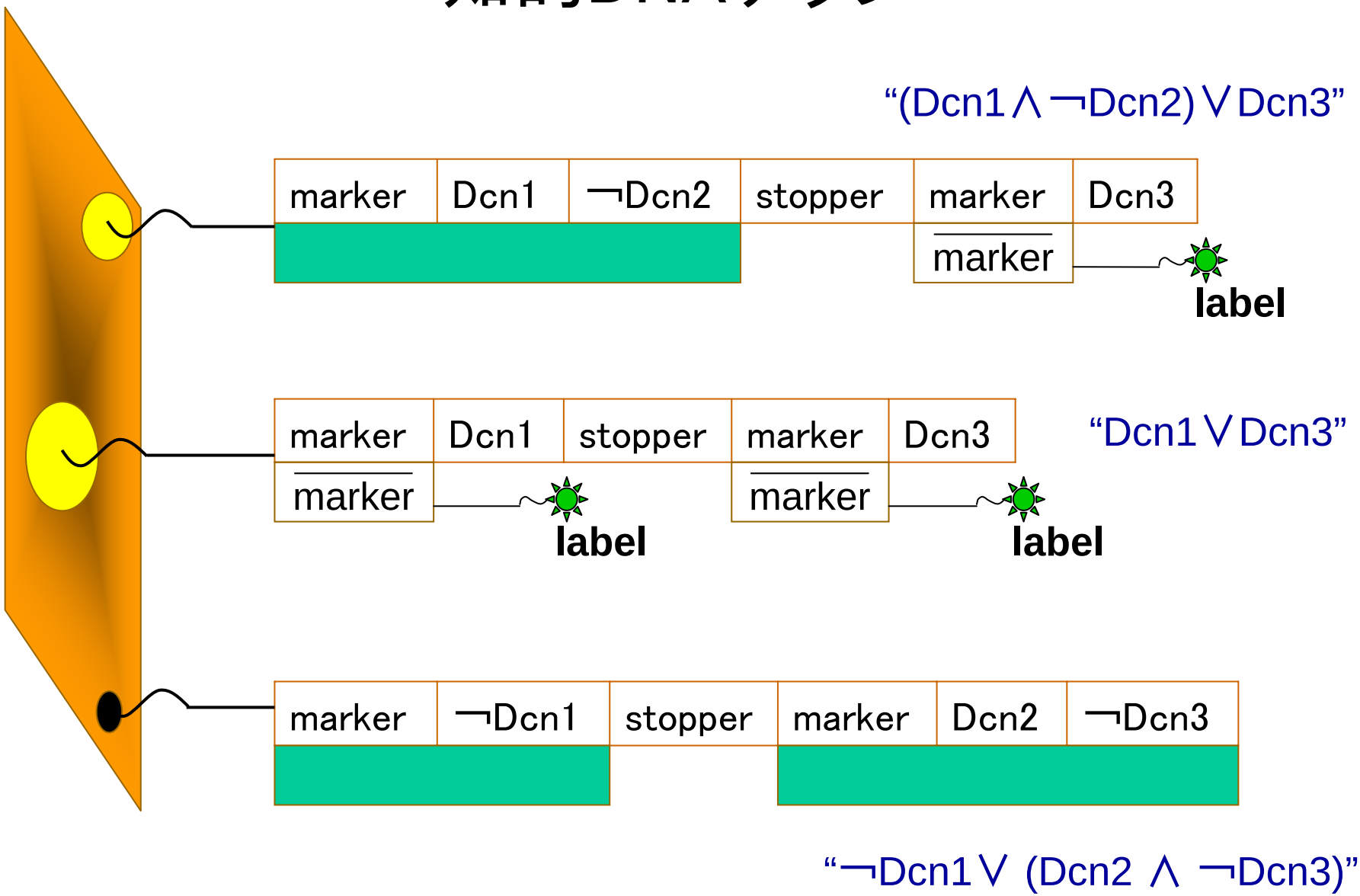
DNA符号化数

試験管内での情報処理
in vitro

DNAチップ上での出力

- DNA分子の直接入力
- 超並列性

知的DNAチップ


“(Dcn1∧¬Dcn2)∨Dcn3”
marker
Dcn1
¬Dcn2
stopper
marker
Dcn3
marker
label
marker
Dcn1
stopper
marker
Dcn3
“Dcn1∨Dcn3”
marker
label
marker
label
marker
¬Dcn1
stopper
marker
Dcn2
¬Dcn3
“¬Dcn1∨ (Dcn2 ∧ ¬Dcn3)”

分子コンピューティング概論：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

DNAナノテクノロジー
DNAによる自己組織化

- DNAの網
- 分子糊としてのDNA
  - DNAによるナノ粒子の自己組織化
  - DNAによるナノワイアの自己組織化
- DNAタイル
  - DNA自身による構造形成
- プログラムされた自己組織化

DNAによるナノ粒子の自己組織化 初期の研究

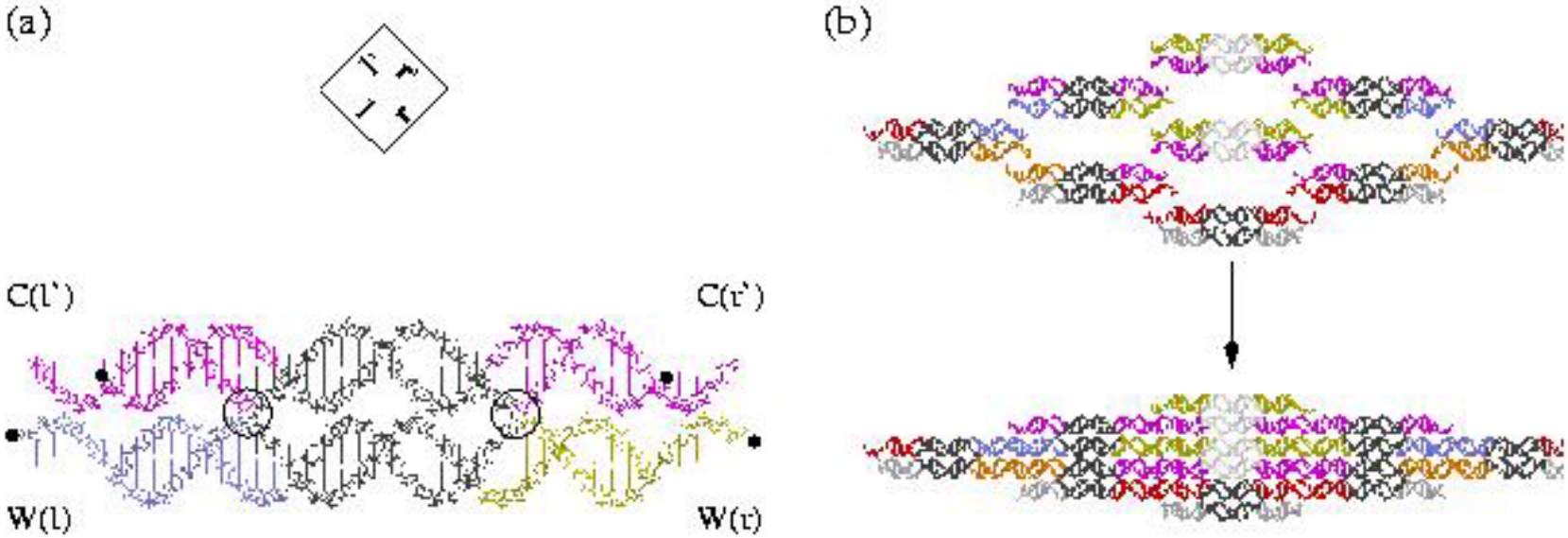
- C. A. Mirkin *et al.*

  DNA-based method for rationally assembling nanoparticles into macroscopic materials. *Nature* **382**, 607-609 (1996)
- A. P. Alivisatos *et al.*

  Organization of 'nanocrystal molecules' using DNA. *Nature* **382**, 609-611 (1996)

Winfree-SeemanのDNAタイル（double crossover分子）

The DNA representation of Wang tiles.



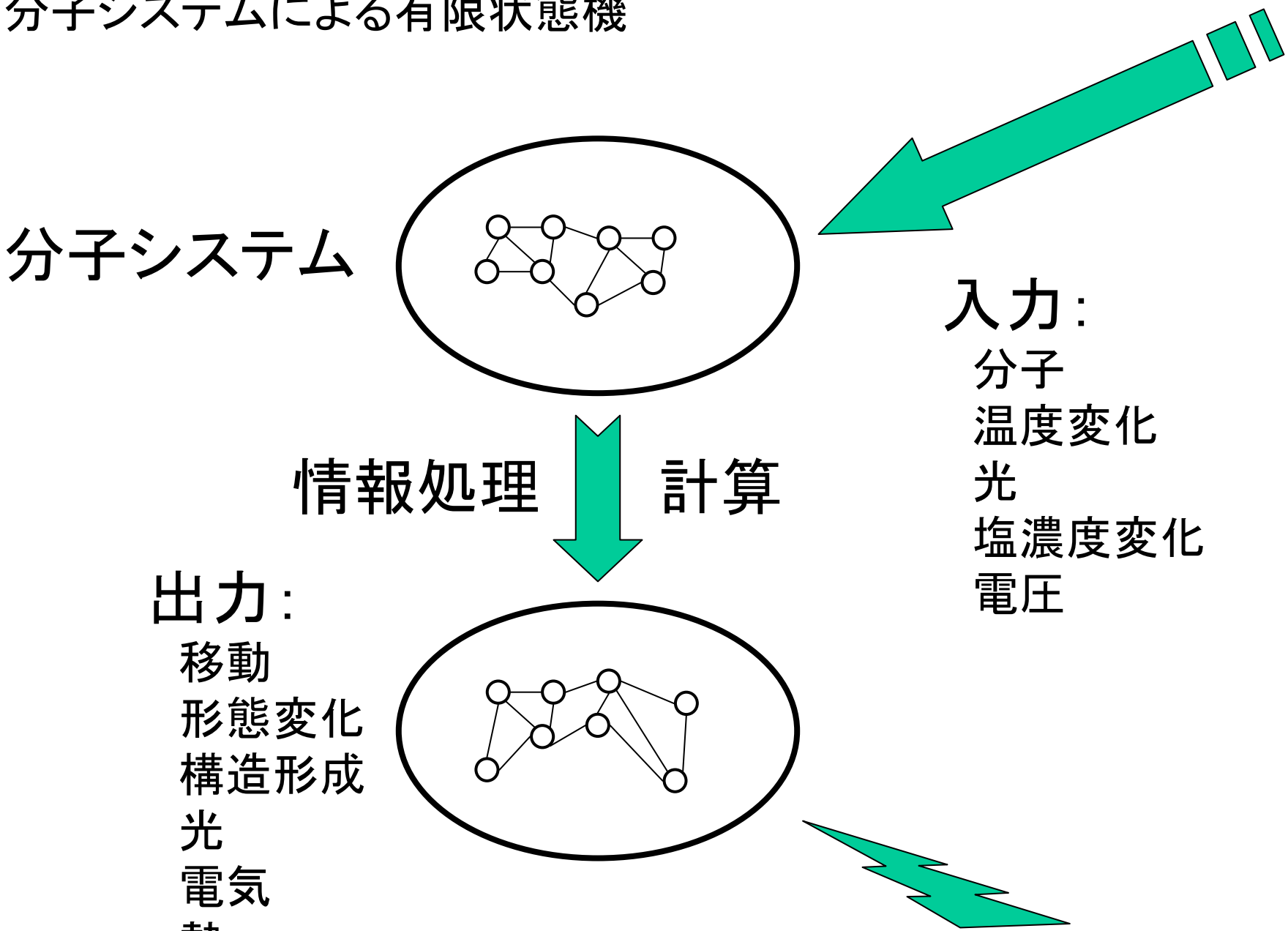
分子コンピューティング概論：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

分子マシン

- アクチュエータとしてのマシン
  - モーター
  - トランスポータ
- 抽象的なマシン --- 有限状態機械（オートマトン）
  - 有限の状態を持つ。
  - その状態を自律的もしくは入力に従って遷移させる。
  - 出力を行うかもしれない。
  - 汎用コンピュータへの第一歩
  - 多くの応用
    - スイッチ・センサー
    - メモリ（記憶保持・アドレシング）
- 両者はまだ渾然一体としている。

分子システムによる有限状態機

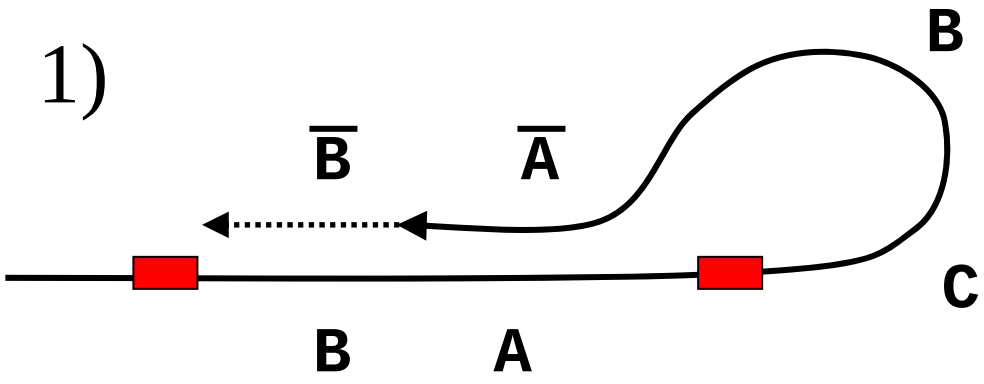


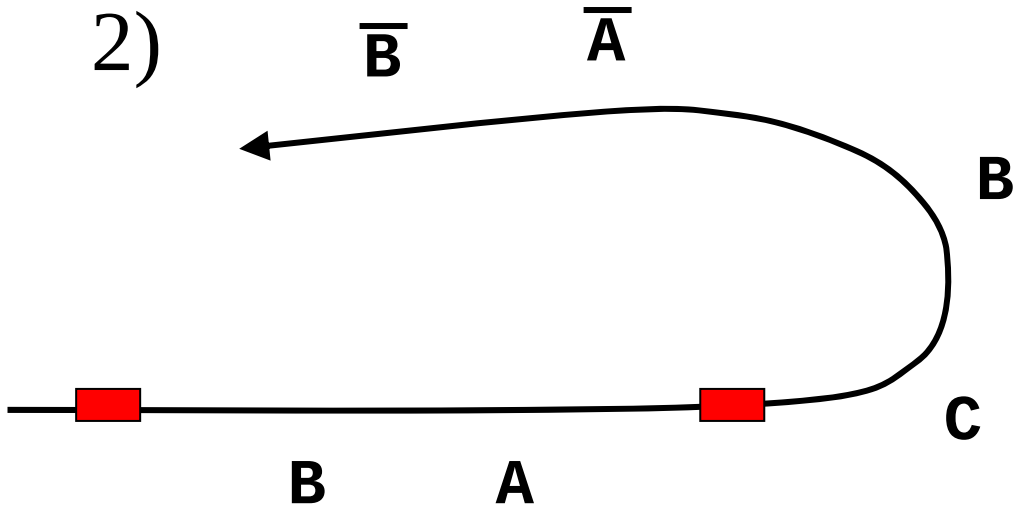
分子（DNA）状態機械

- 末端配列機械
  - 末端配列が状態を表現
  - Whiplash PCR（鞭打ちPCR）
    - 状態が遷移するにつれ長くなる。
  - Shapiroのオートマトン
    - 状態が遷移するにつれ短くなる。
- 形態機械（Conformational Machine）
  - 分子の形態が状態を表現
  - YurkのMolecular Tweezers（分子ピンセット）
  - SeemanのPX-JX_2 Switch
  - 我々のHairpin-Based Machine

Whiplash PCR (WPCR)

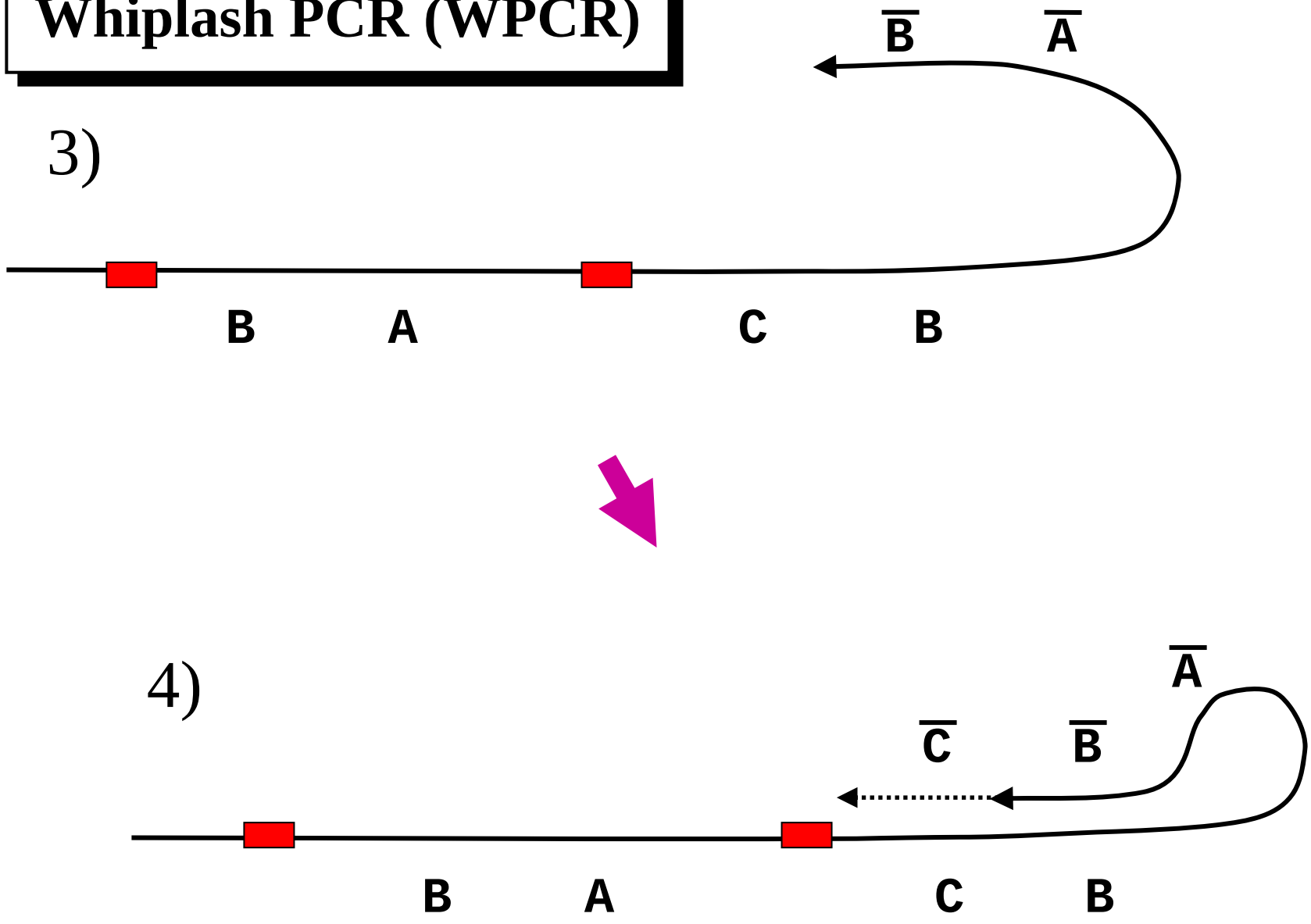
: stopper sequence



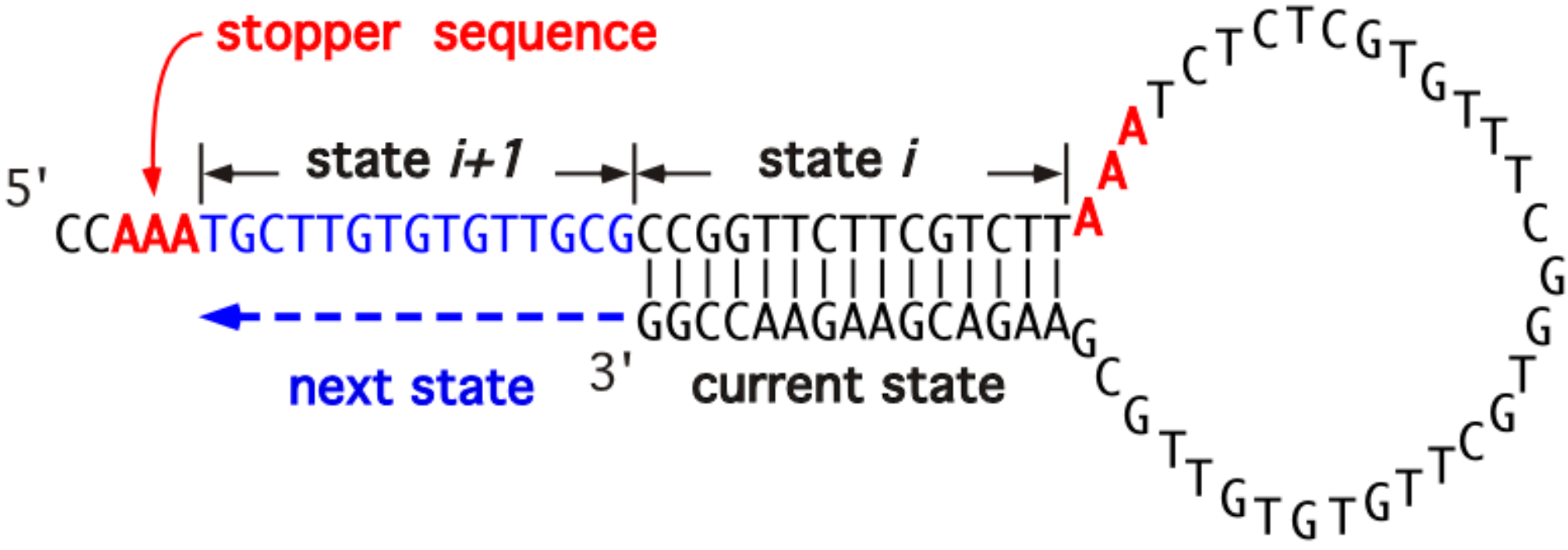


Komiya *et al.*

Whiplash PCR (WPCR)

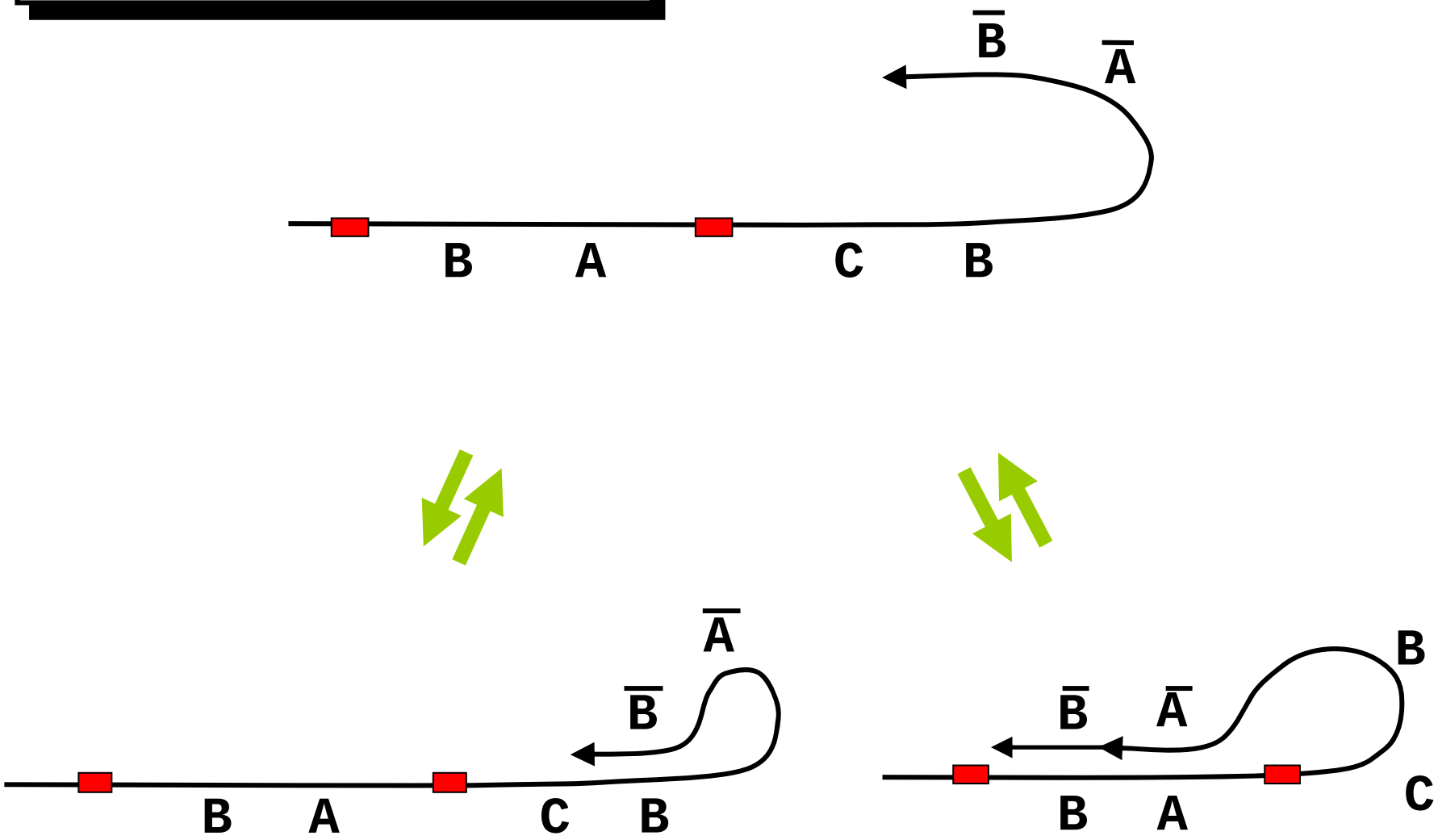


Polymerization Stop



Polymerization by DNA polymerase
with dATP, dCTP, dGTP

Back-hybridization

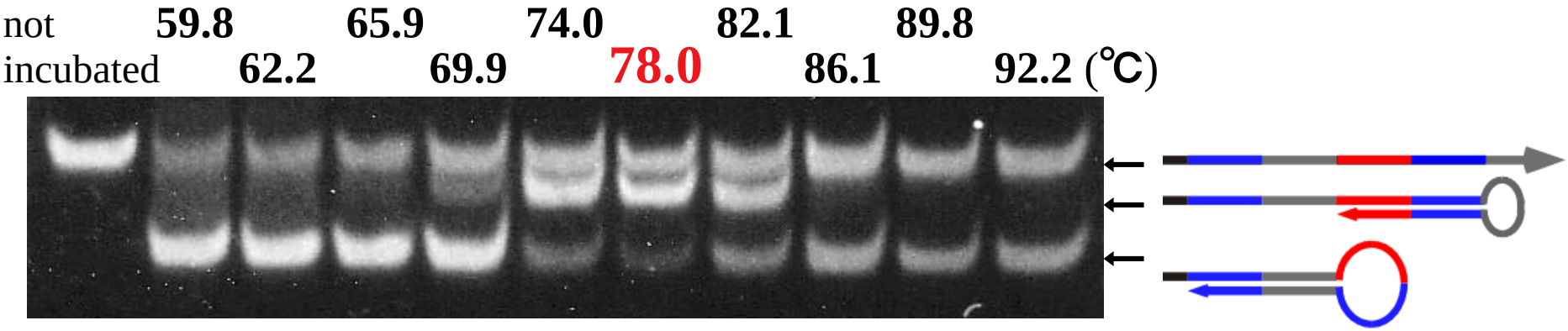


Competing Alternative Hairpin Forms

Temperature optimization for WPCR

- 8 M urea 8% PAGE

Komiya, et al.



in 1X *Pfx* buffer
(the composition unknown)
1 mM $MgSO_4$
0.2 mM dATP, dCTP, dGTP
1.5 units Platinum *Pfx* DNA polymerase

Thermal schedule

94℃ for 1 min.

↓

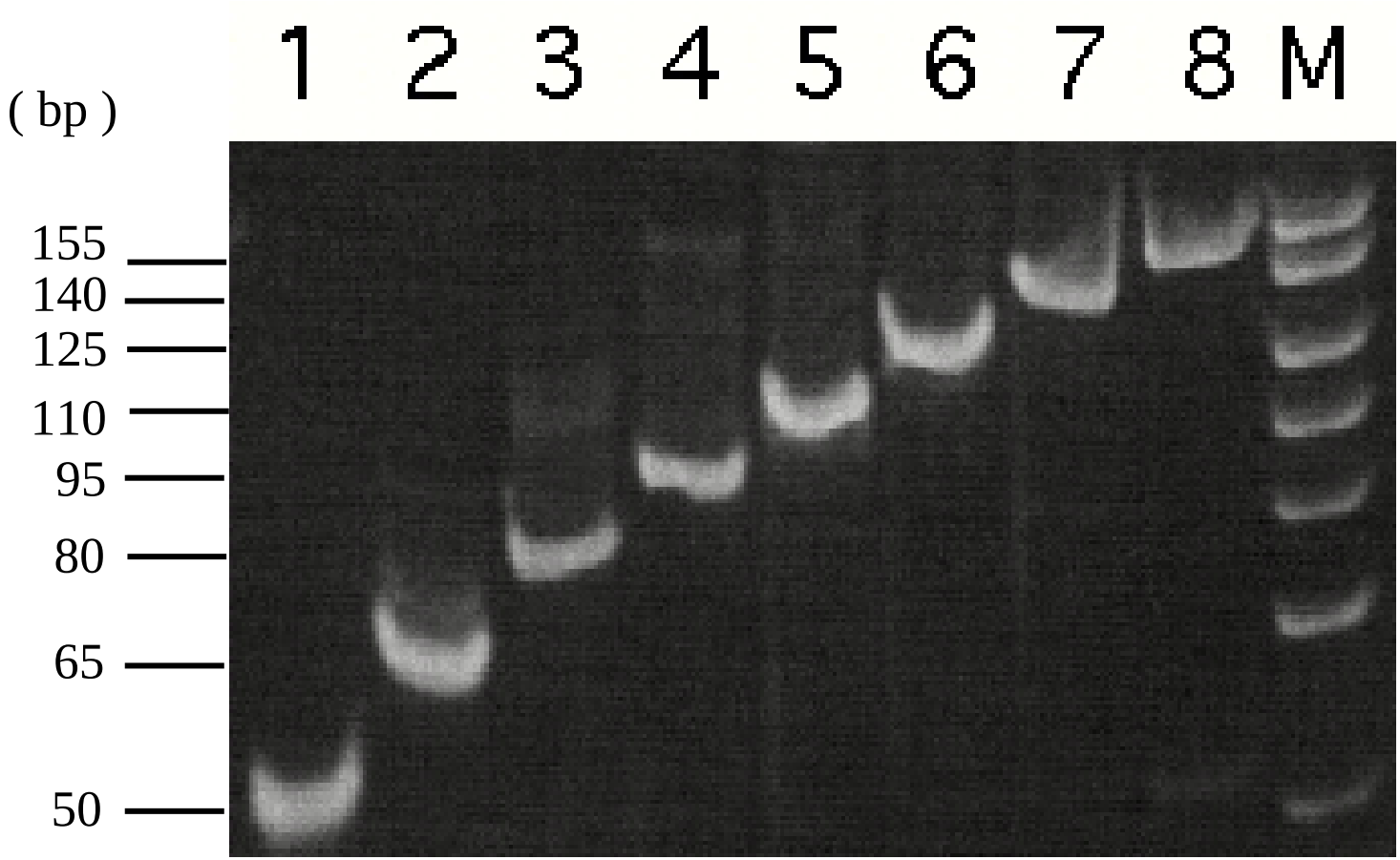
x ℃ for 5 min.

x =59.8 ~ 92.2

Successful implementation of transitions

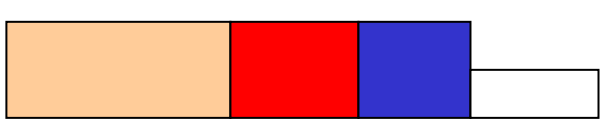
- 12% PAGE

Komiya, et al.



ShapiroのDNAオートマトン

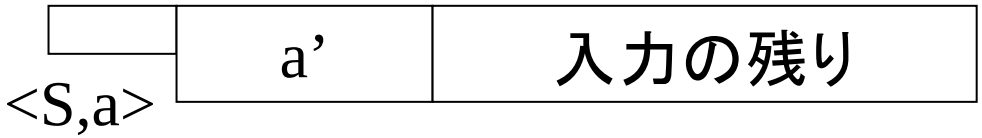
IIS型制限酵素
認識部位　スペーサ

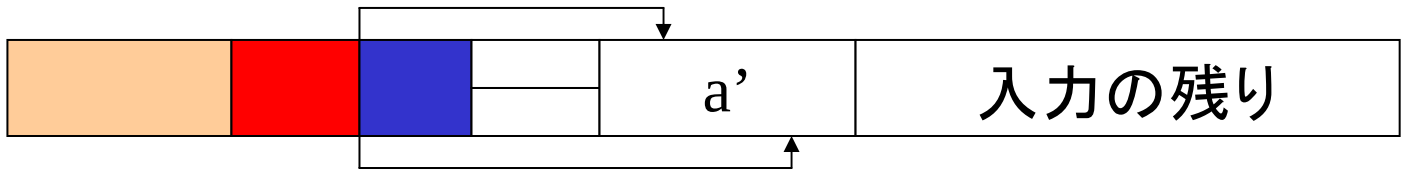


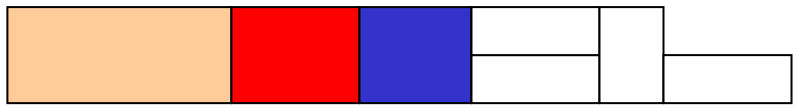
S,a → S’

<S,a>

遷移分子







入力文字a’の配列は、各S’に対して<S’,a’>を含んでいる。
遷移分子は、スペーサを調整して、適切な個所で切断する。

ShapiroのDNAオートマトン

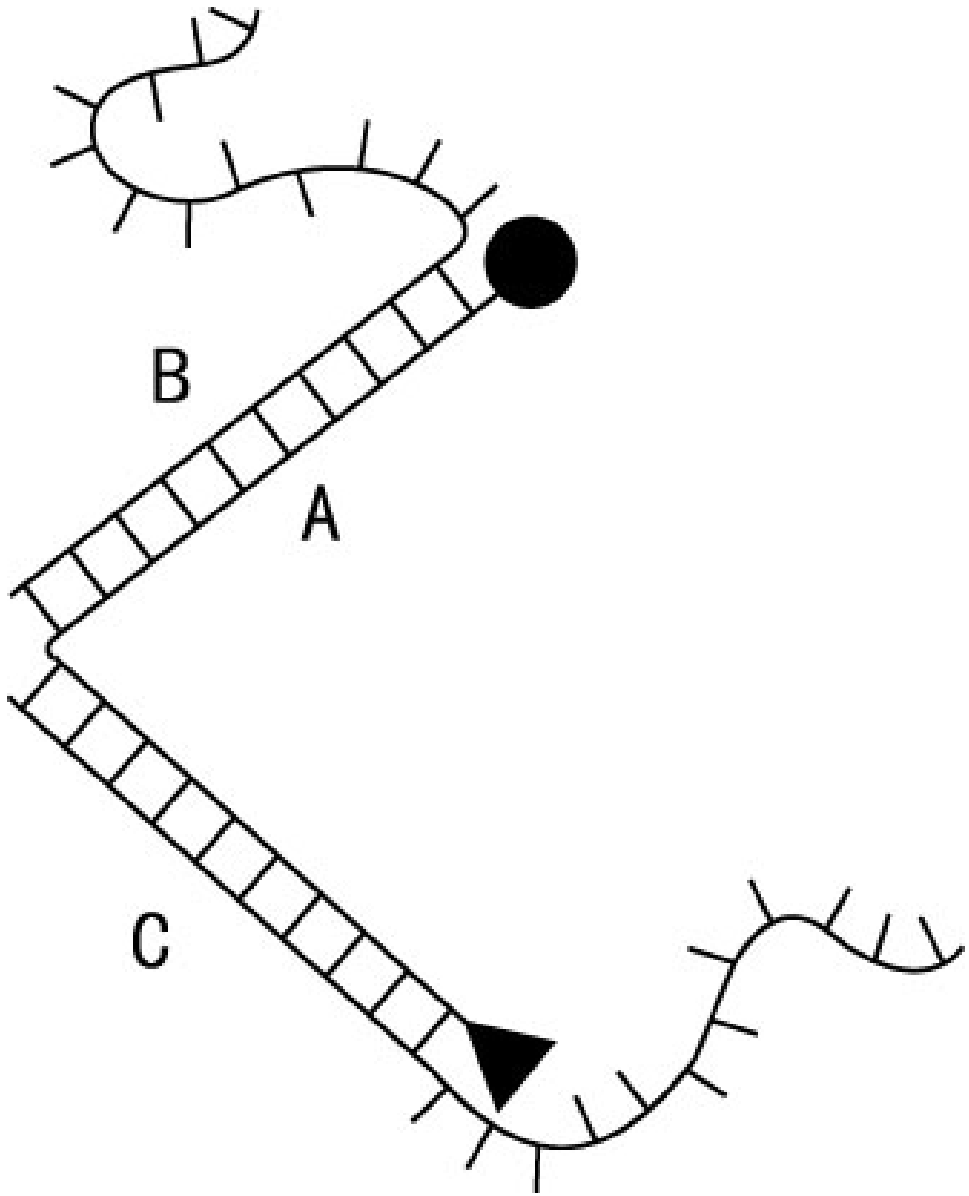
- *Nature* 2001
- 2入力文字、2状態
- *Fok*I

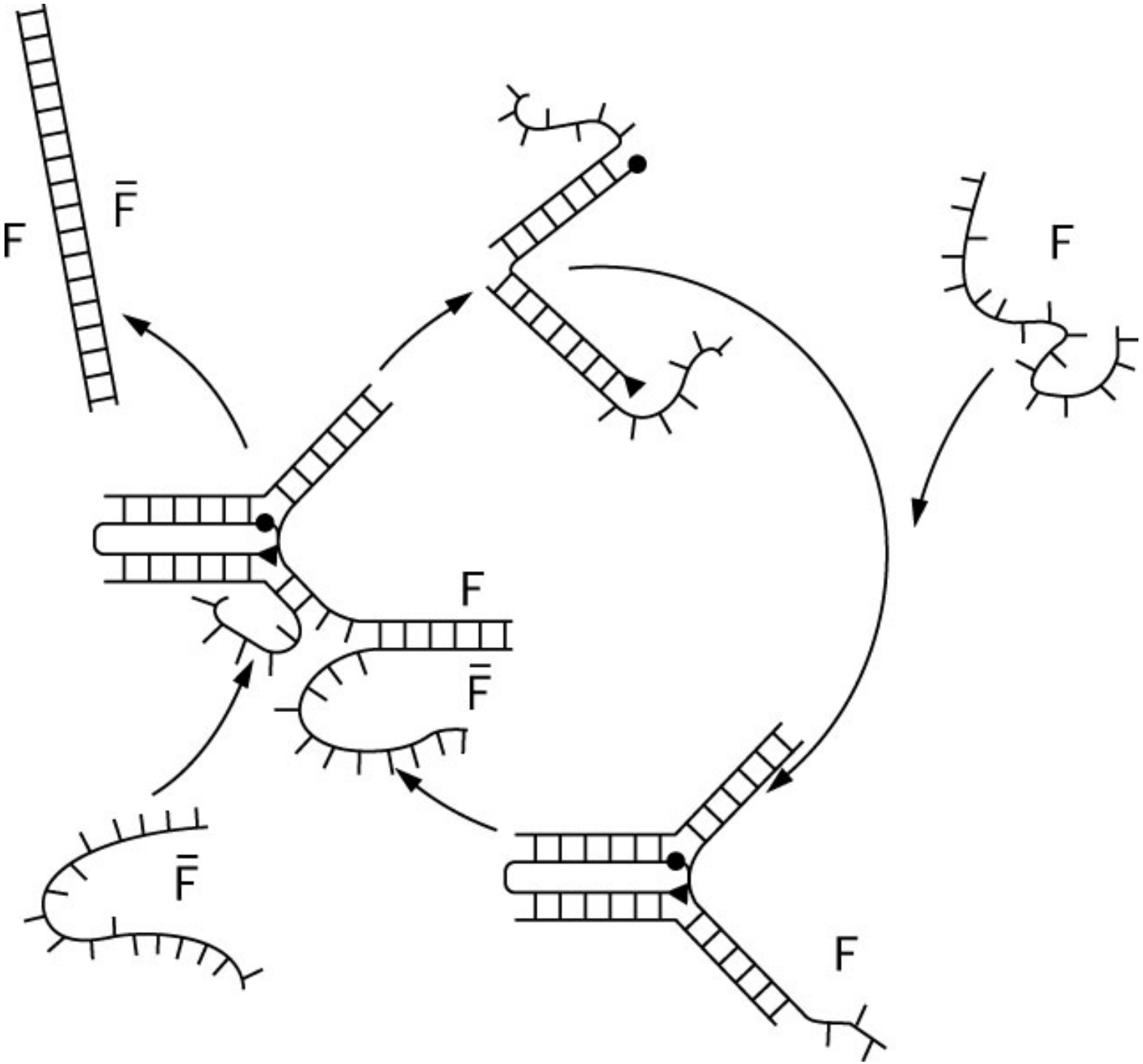
a=**CTGGCT**　　b=**CGCAGC**

5′-p…22…GGATGTAC
3′-GGT…22…CCTACATGCCGAp　　S0,a→S0

5′-p…22…GGATGACGAC
3′-GGT…22…CCTACTGCTGCCGAp　　S0,a→S1

YurkeのDNAピンセット




F
F̄
F
F̄
F̄
F
F

分子コンピューティング概論：目次

- 分子反応の持つ計算能力の解析
  - 計算モデル・計算可能性・計算量
- 分子システム設計の計算論的側面
  - 分子の設計・分子反応の設計
- 分子反応の持つ計算能力の応用
  - 知的分子計測
  - 自己組織化
  - 分子マシン
- 分子反応に基づく新しい計算パラダイム
  - 膜コンピューティング、アモルファス・コンピューティング
  - 光や量子との融合
  - 分子エレクトロニクスとの融合

新しい計算パラダイム

- 膜コンピューティング
  - Paun
- アモルファス・コンピューティング
  - MIT のグループ
    - Abelson と Sussman
    - Knight
- その他にも…
  - Smart Dust
  - Programmable Matter
  - Quantum-Dot Cell Automaton
  - …

細胞膜モデル

- G. Paun (1998)
- 細胞膜(membrane)による計算過程の制御
- スーパーセルシステム＝万能計算モデル

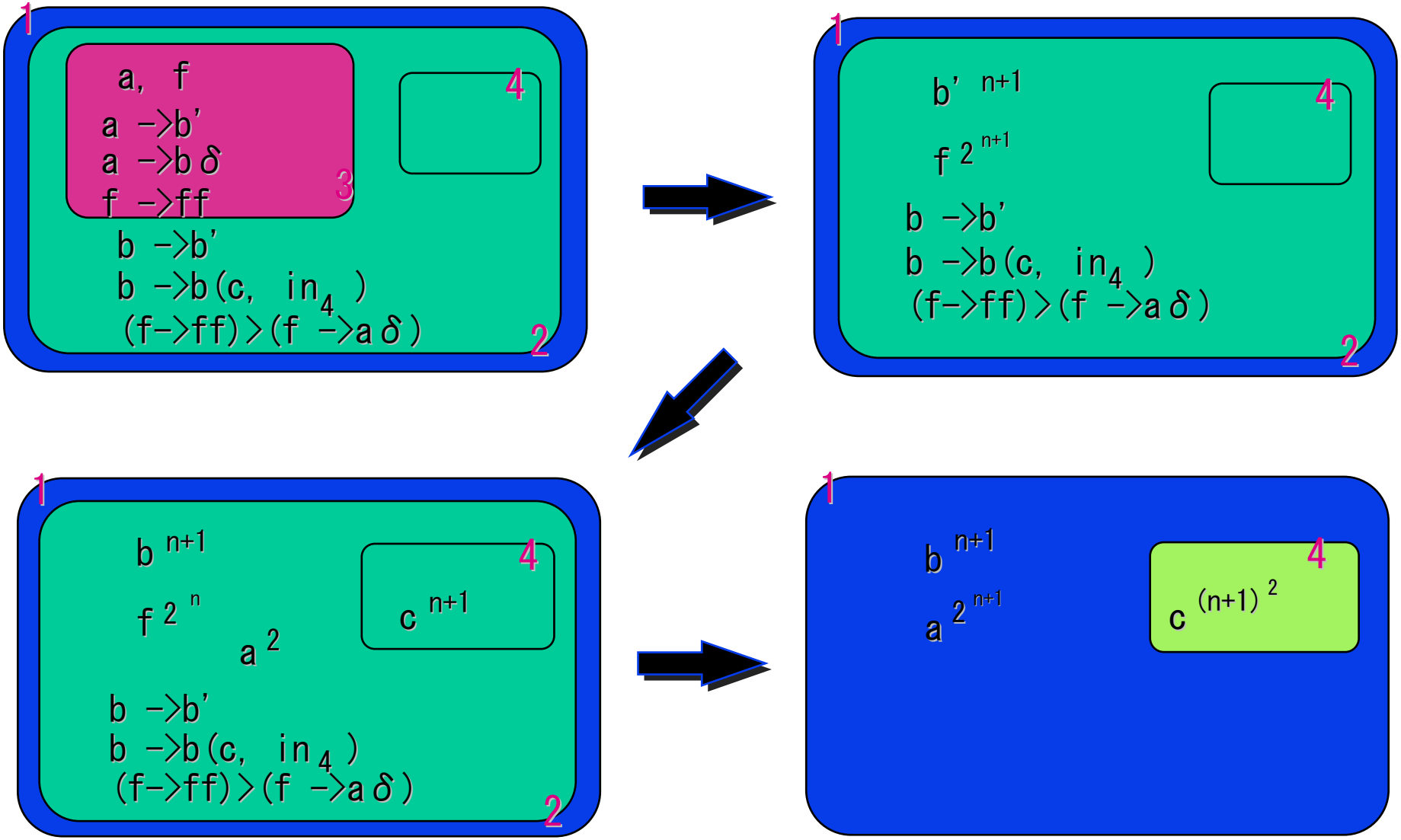
（例） $G=(V, \mu, M_1, \ldots, M_4, (R_1, \rho_1), \ldots, (R_4, \rho_4), 4)$

$V=\{a, b, b', c, f\}$　アルファベット

$\mu=[_1[_2[_3]_3[_4]_4]_2]_1$　膜構造

M_i　膜 i 内要素の多重集合

(R_i, ρ_i)　膜 i 内の規則の順序集合



細胞膜モデルによる n^2 の計算

アモルファス計算

- 自己組織化のための新しい計算パラダイム
  - 微細加工技術と細胞工学
  - 低コストで様々なプロセッサ
- Computational particle
  - 小さい計算力と少量メモリー
  - 不規則配置、可動性
  - 非同期、局地的相互作用
  - 誤った挙動、環境の影響
  - 同一プログラム
  - 自分たちの位置や方向に関する情報をもたない
  - 近接のparticleと短距離（半径r）の通信をする。
- 全体としては超並列計算システムになっている。
- 回路の自己組織化のシミュレーション

Amorphous Computingとは

- 背景
  - 微細加工技術と細胞工学
  - 低コストで様々なプロセッサを作る
    （厳密に正確な動作は不要）
  - 新しい計算パラダイムとしての研究
- 不規則に配置されて、
  非同期に局地的に相互作用するような，
  計算機能の要素“computational particle”の
  集合としてモデル化。
- 効果的にプログラムするにはどうすればよいか。
  - 生物学的な組織の形成に関連？
  - 生物学は単なるメタファーでなく，実装に使えないか？

Computational particleの性質

- 誤った挙動を示す可能性もある。
- 環境に影響される。
- 何らかの動作をするかもしれない。
- 動き回るかもしれない。
- 小さい計算能力と少量のメモリーを持つ。
- 全particleは同様にプログラムされている。
  （局所状態の保持や乱数発生は可能）
- 自分たちの位置や方向に関する情報をもたない。
- 近接のparticleと短距離(半径r)の通信をする。
- 全体としては超並列計算システムになっている。

Wave propagationによるパターン形成

- 最初の“anchor” particleから始めてメッセージ（ホップの情報を持つ）を伝達していく。
- 生物学的なパターン形成と関連。
- ２つのanchor particleにより「成長の阻害」や「屈動性(tropism)」などもプログラムできる。
- Cooreによるgrowing-point language(GPL)でプログラミングし、コンパイルしてparticleにセット。

量子ドット・コンピュータ

- キワモノか？
- 量子コンピュータとは違う。
- 量子ドット・セル・オートマトン（QCA）
  - ドミノのように四個の量子ドットを並べる。
  - ドミノ内ではトンネル効果によって電子が移動。
  - ドミノの相互作用により状態が伝搬する。
- 配線が必要無い？
- しかし、量子ドットを正確に配置しなければならない。

計算モデル論（萩谷分）

① DNA/RNAの二次構造を予測する方法、動的プログラミングにより最小エネルギーおよび分配関数を求める方法、二次構造から配列を設計する方法について説明せよ。（文献を調べるとよい。）

② DNAを用いた自己組織化や分子マシンの実現可能性や応用について述べよ。

BASICS

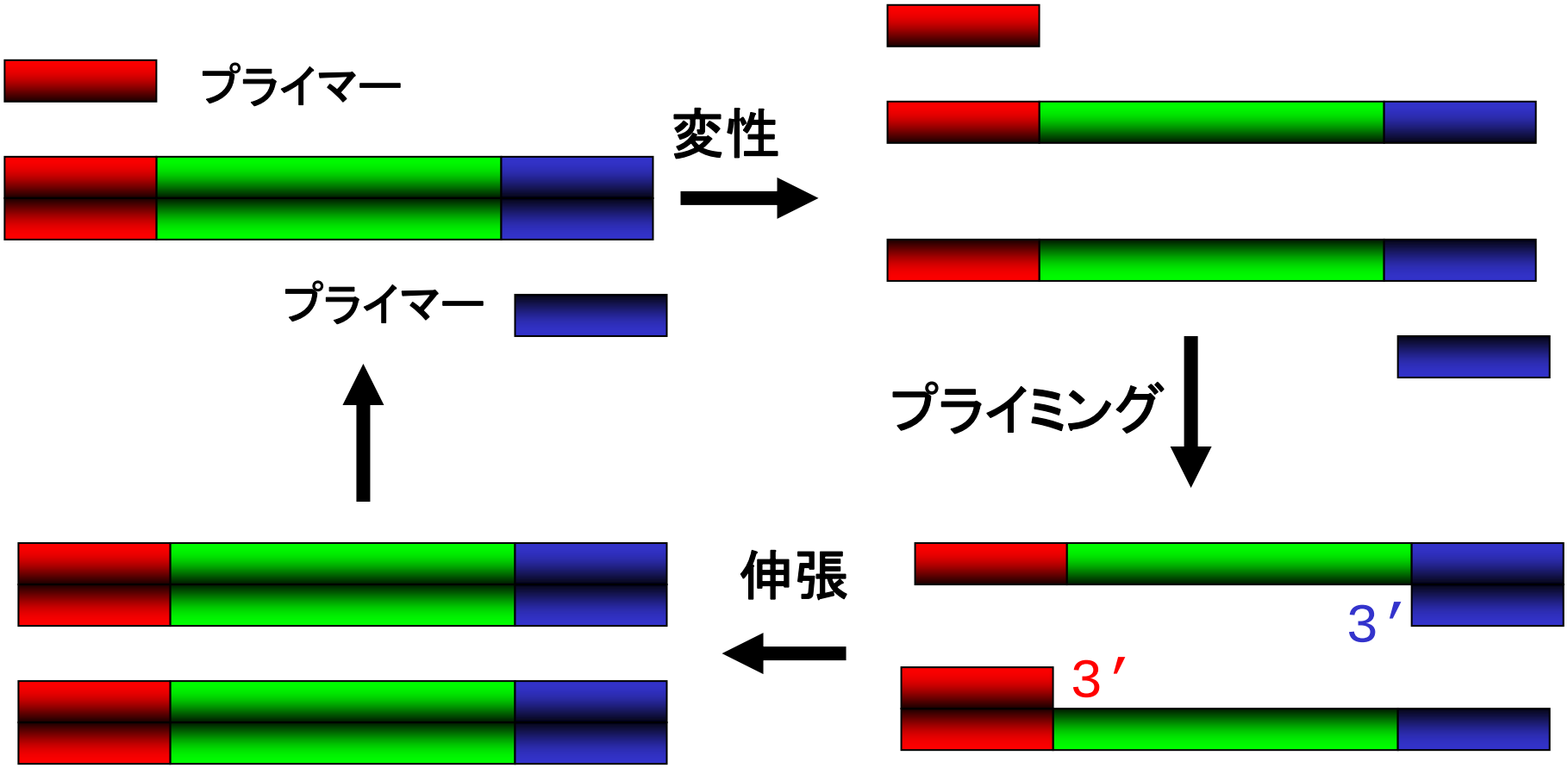
DNA

- 糖
  - デオキシリボース
- リン酸
- 塩基
  - プリン塩基 --- ６角形と5角形の二つのリング
    - アデニン（A：Adenine）
    - グアニン（G：Guanine）
  - ピリミジン塩基 --- ６角形一つ
    - チミン（T：Thymine）
    - シトシン（C：Cytosine）

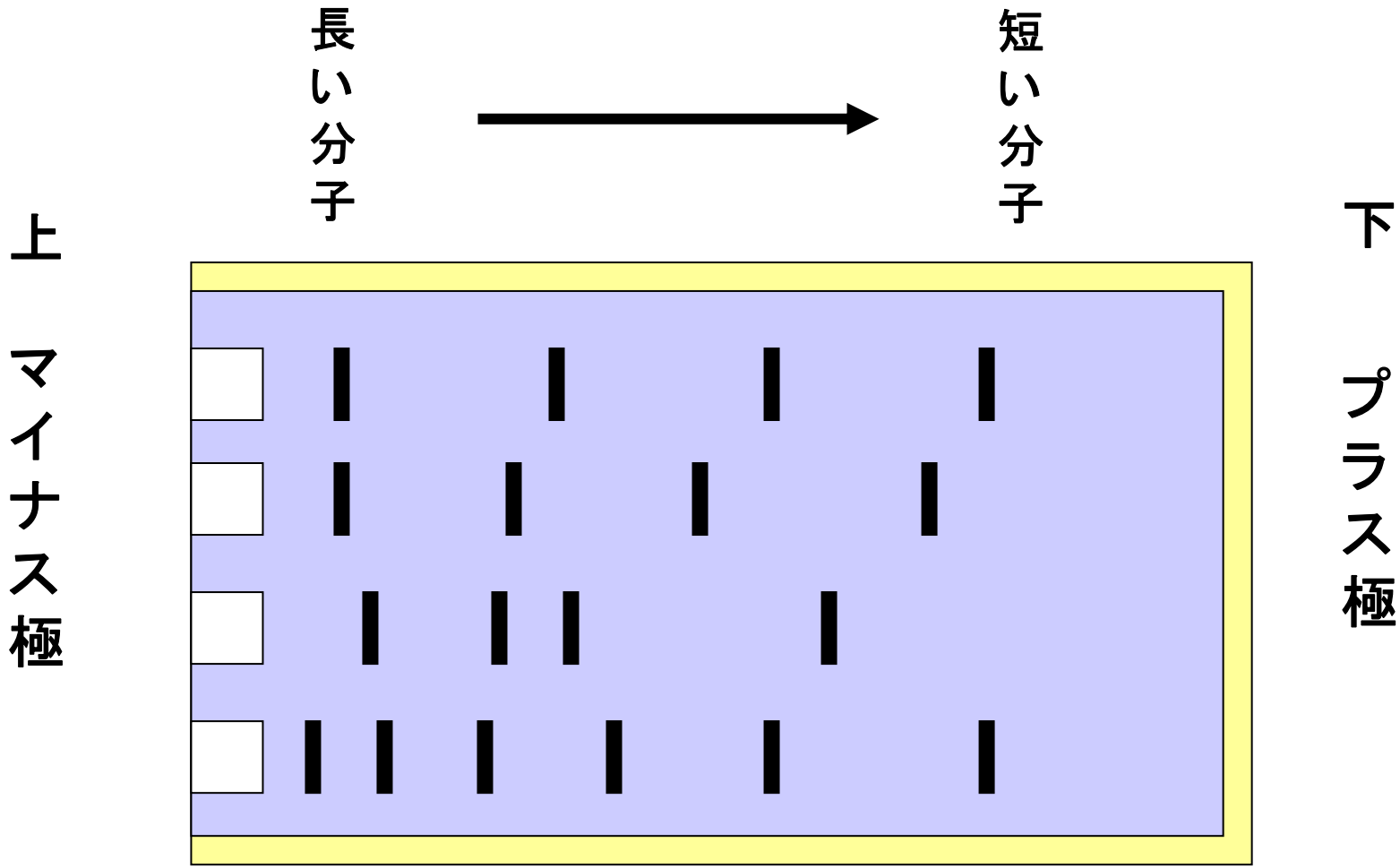
実験操作

- PCR（ポリメラーゼ連鎖反応）
- ゲル電気泳動
- アフィニティ・セパレーション
- 制限酵素による切断
- ライゲースによる連結
- クローニングとシーケンシング

PCR（ポリメラーゼ連鎖反応）

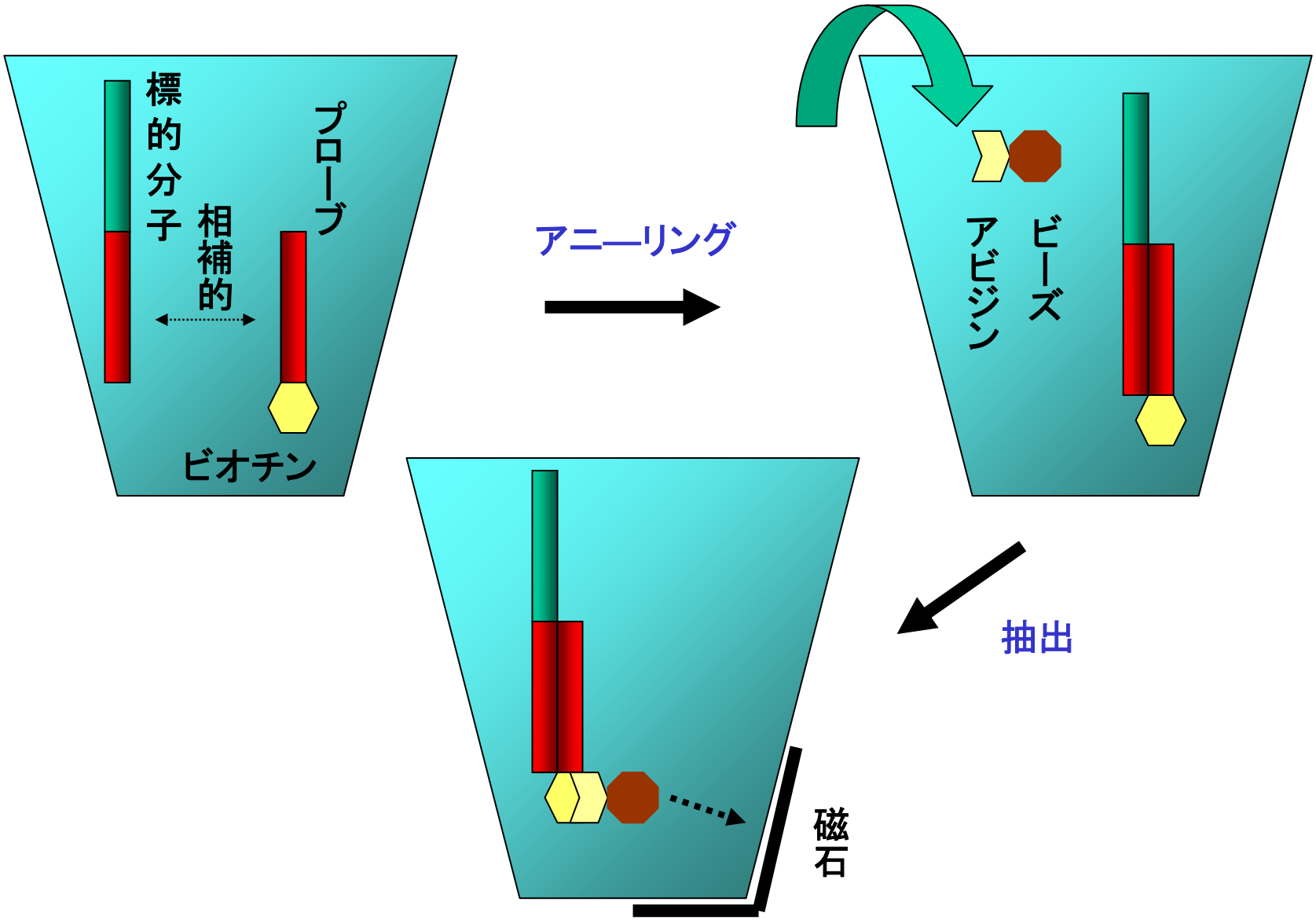


ゲル電気泳動



ポリアクリルアミドゲル電気泳動

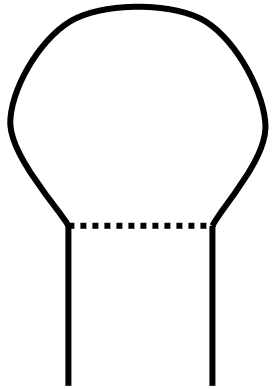
DNA一本鎖の分離



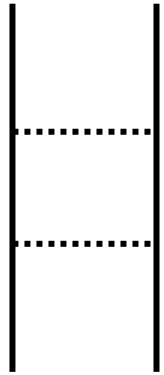
DNA（RNA）の二次構造とその予測

DNA（RNA）の二次構造

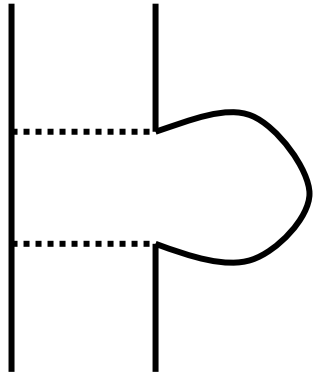
- ベースペア *i.j* の集合
- *k*-ループ --- *k* 個のベースペアで囲まれたループ
  - 1-ループ
    - ヘアピン（hairpin）
  - 2-ループ
    - スタック（stack）
    - バルジ（bulge）
    - 内部（interior）
  - マルチ・ループ（multi-loop）
- ループに対してエネルギーが割り当てられる。



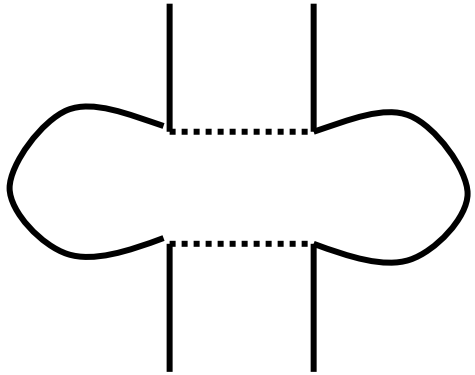
ヘアピン



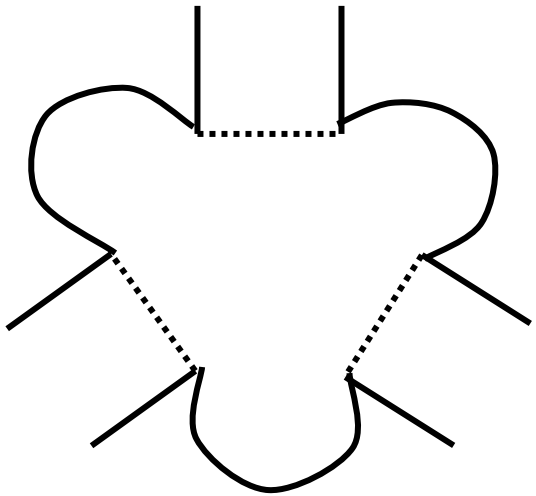
スタック



バルジ



内部



3-ループ

これらの構造にエネルギーを割り当てる。
（nearest neighborモデル）

動的プログラミング

- $W(i, j)$: i 番目のベースと j 番目の間の最小エネルギー
- $V(i, j)$: i と j がペアである場合の最小エネルギー

- $W(i, j) = \min(W(i+1, j),\ W(i, j-1),\ V(i, j),\ \min_{i \le k < j}(W(i, k)+W(k+1, j)))$

- $V(i, j) = \min(\mathrm{eh}(i, j),\ \mathrm{es}(i, j)+V(i+1, j-1),\ VBI(i, j),\ VM(i, j))$
  - $\mathrm{eh}(i, j)$: ヘアピンのエネルギー
  - $\mathrm{es}(i, j)$: スタックのエネルギー

動的プログラミング

- $VBI(i, j) = \min_{\substack{i<i'<j'<j \\ i'-i+j-j'>2}}(\mathrm{ebi}(i, j, i', j')+V(i', j'))$
  - $\mathrm{ebi}(i, j, i', j')$: 内部ループのエネルギー
  - $O(n^4)$ になってしまう。

- $VM(i, j) = \min_{i<k<j-1}(W(i+1, k)+W(k+1, j-1))$
  - マルチ・ループのエネルギーが 0 の場合。

内部ループ

- 内部ループのエネルギー $\mathrm{ebi}(i, j, i', j')$ が、ループの長さ $(i'-i+j-j') \times c$ ならば、

- $VBI(i, j) = \min_{l}(VBI(i, j, l))$
- $VBI(i, j, l) = \min(VBI(i+1, j, l-1) + c,$
  $VBI(i, j-1, l-1) + c,$
  $c \times l + V(i+1, j-l+1),$
  $c \times l + V(i+l-1, j-1))$
  - $O(n^3)$ になる。

マルチ・ループ

- マルチ・ループのエネルギーの近似:

$$a + b \times k' + c \times k$$

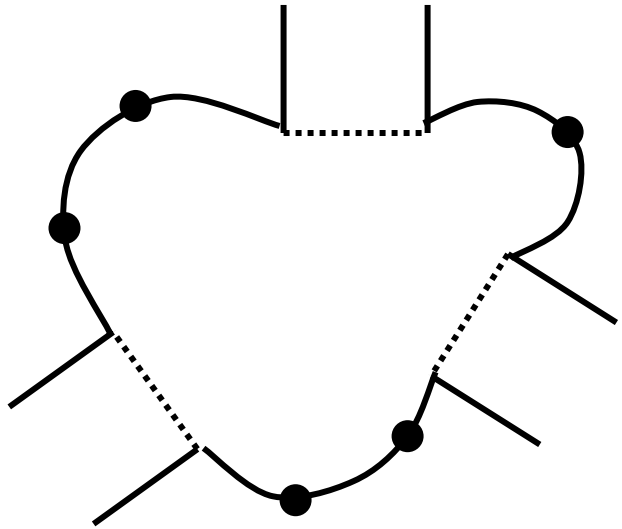
k': ペアを組まない塩基の数

k: ペアの数

  - $O(n^3)$ になる。

$k' = 5$

$k = 3$



McCaskillのアルゴリズム

- 個々の構造のエネルギーを求めるのではなく、可能なすべての構造のエネルギーの分布を求める。
  - 分配関数（partition function）
  - 特定のベースペアが形成される確率
- 動的プログラミングによる。

基本配列

- w[i,j]
  iとjの間の最小エネルギー
- ww[k,j]
  kがペアを作っている条件のもとでのw[k,j]
  実際にはjとj-1の場合だけ記憶すればよいので、
  再利用することができる。
- v[i,j]
  iとjがペアを作っている場合の
  iとjの間の最小エネルギー

- 配列はすべてINF（無限大）で初期化する。

```
for (j=2; j<=n; j++)
    for (i=j-1; i>=1; i--) {
        ww[i,j] = ww[i,j-1];
        if (i.jがペア)
            ww[i,j] = min(ww[i,j], v[i,j]);
        for (temp=INF, k=i+1; k<=j; k++)
            temp = min(temp, w[i,k-1]+ww[k,j]);
        w[i,j] = min(temp, ww[i,j]);
    }
```

マルチループのための配列

- v[i,j]
  iとjがペアを作っている場合の
  iとjの間の最小エネルギー
- vm[i,j]
  iとjの間のペアをマルチループに属すると
  仮定したときの最小エネルギー
  最低一個はペアを含む。
- vvm[k,j]
  kがペアを作っている条件のもとでのvm[k,j]
  実際にはjとj-1の場合だけ記憶すればよいので、
  再利用することができる。

```
for (j=2; j<=n; j++)
    for (i=j-1; i>=1; i--) {
        if (i.jがペア) {
            v[i,j] = min(v[i,j], ヘアピンのエネルギー);
            for (l=i+2; l<j-1; l++)
                for (k=l-1; k>i; k--)
                    if (k.lがペア)
                        v[i,j] = min(v[i,j],
                                        v[k,l]+2ループのエネルギー);
            for (temp=INF, k=i+2; k<=j-1; k++)
                temp = min(temp, vm[i+1,k-1]+vvm[k,j-1]);
            v[i,j] = min(v[i,j], temp+MLclosing+MLintern);
        }
        vmとvvmの設定;
    }
```

マルチ・ループ

MLclosing

- マルチ・ループのエネルギーの近似:

$$a + b \times k' + c \times k$$

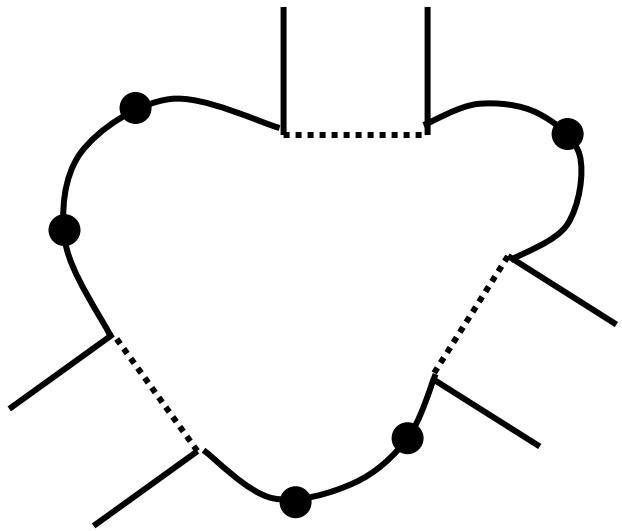
k': ペアを組まない塩基の数

k: ペアの数

MLintern

MLbase

– $O(n^3)$ になる。



$k' = 5$

$k = 3$

vmとvvmの設定:

```
vvm[i,j] = vvm[i,j-1]+MLbase;
if (i.jがペア)
    vvm[i,j] = min(vvm[i,j], v[i,j]+MLintern);
for (temp=INF, k=i+1; k<=j; k++) {
    temp = min(temp, vm[i,k-1]+vvm[k,j]);
    temp = min(temp, MLbase*(k-i)+vvm[k,j]);
}
vm[i,j] = min(temp, vvm[i,j]);
```

分配関数

- 構造が現れる確率は、
  構造のエネルギーを G としたとき、
  ボルツマン因子 exp(-G/kT) に比例する。
- 分配関数 Z とは、
  すべての構造のボルツマン因子を
  足し合わせたもの。
- エネルギー G の構造の出現確率は、
  exp(-G/kT)/Z で与えられる。

分配関数の計算

- 二次構造を網羅しながら
  最小エネルギーを求めていたところを、
  二次構造を網羅しながら、
  ボルツマン因子を足し合わせればよい。

最小エネルギー	分配関数
G	exp(-G/kT)
初期値INF	初期値0
min	＋
＋	＊

基本配列

- w[i,j]
  iとjの間の分配関数
- ww[k,j]
  kがペアを作っている条件のもとでのw[k,j]
  実際にはjとj-1の場合だけ記憶すればよいので、
  再利用することができる。
- v[i,j]
  iとjがペアを作っている場合の
  iとjの間の分配関数

- 配列はすべて0で初期化する。

```
for (j=2; j<=n; j++)
    for (i=j-1; i>=1; i--) {
        ww[i,j] = ww[i,j-1];
        if (i.jがペア)
            ww[i,j] = ww[i,j]+v[i,j];
        for (temp=0, k=i+1; k<=j; k++)
            temp = temp+w[i,k-1]*ww[k,j];
        w[i,j] = temp+ww[i,j];
    }
```

マルチループのための配列

- v[i,j]
  iとjがペアを作っている場合の
  iとjの間の分配関数
- vm[i,j]
  iとjの間のペアをマルチループに属すると
  仮定したときの分配関数
  最低一個はペアを含む。
- vvm[k,j]
  kがペアを作っている条件のもとでのvm[k,j]
  実際にはjとj-1の場合だけ記憶すればよいので、
  再利用することができる。

```
for (j=2; j<=n; j++)
    for (i=j-1; i>=1; i--) {
        if (i.jがペア) {
            v[i,j] = v[i,j]+ヘアピンの分配関数;
            for (l=i+2; l<j-1; l++)
                for (k=l-1; k>i; k--)
                    if (k.lがペア)
                        v[i,j] = v[i,j]+v[k,l]*2ループの分配関数;
            for (temp=0, k=i+2; k<=j-1; k++)
                temp = temp+vm[i+1,k-1]*vvm[k,j-1];
            v[i,j] = v[i,j]+temp*expMLclosing*expMLintern;
        }
        vmとvvmの設定;
    }
```

vmとvvmの設定:

```
vvm[i,j] = vvm[i,j-1]*expMLbase;
if (i.jがペア)
    vvm[i,j] = vvm[i,j]+v[i,j]*expMLintern;
for (temp=0, k=i+1; k<=j; k++) {
    temp = temp+vm[i,k-1]*vvm[k,j];
    temp = temp+expMLbase^(k-i)*vvm[k,j];
}
vm[i,j] = temp+vvm[i,j];
```