JSPS Project on Molecular Computing

(presentation by **Masami Hagiya)**

- funded by Japan Society for Promotion of Science
- *Research for the Future* Program
  - ***biocomputing*** **field** chaired by Prof. Anzai
    - molecular computing
    - artificial cell (chemical IC)
    - evolutionary computation
    - input/output for molecular computing
    - complex systems
- October 1996 - March 2001
  - 60M-80M yen per year

JSPS Project on Molecular Computing

- project leader - Masami Hagiya (Computer Science)
- members
  - Takashi Yokomori (Computer Science)
  - Akira Suyama (Biophysics)
  - Yuzuru Husimi (Biophysics)
  - Kensaku Sakamoto (Biochemistry)
  - Shigeyuki Yokoyama (Biochemistry)
  - Masayuki Yamamura (Computer Science)
  - Masanori Arita (Genome Informatics)

Some Achievements

- Kensaku Sakamoto, Shigeyuki Yokoyama, and Masami Hagiya
  - Whiplash PCR
  - Hairpin Engine
- Masami Hagiya
  - VNA --- a simulator for DNA reactions
- Takashi Yokomori
  - Formal Models of DNA Computing
- Akira Suyama
  - Solid-based DNA Computing
  - Dynamic Programming DNA Computers
- Yuzuru Husimi
  - Evolutionary Reactor Based on 3SR
- Masayuki Yamamura
  - Aqueous Computing (with Tom Head)

DNA Computing

- Adleman-Lipton Paradigm **Å** massive parallelism
  - random generation by assembly
  - selection by molecular biology experiments

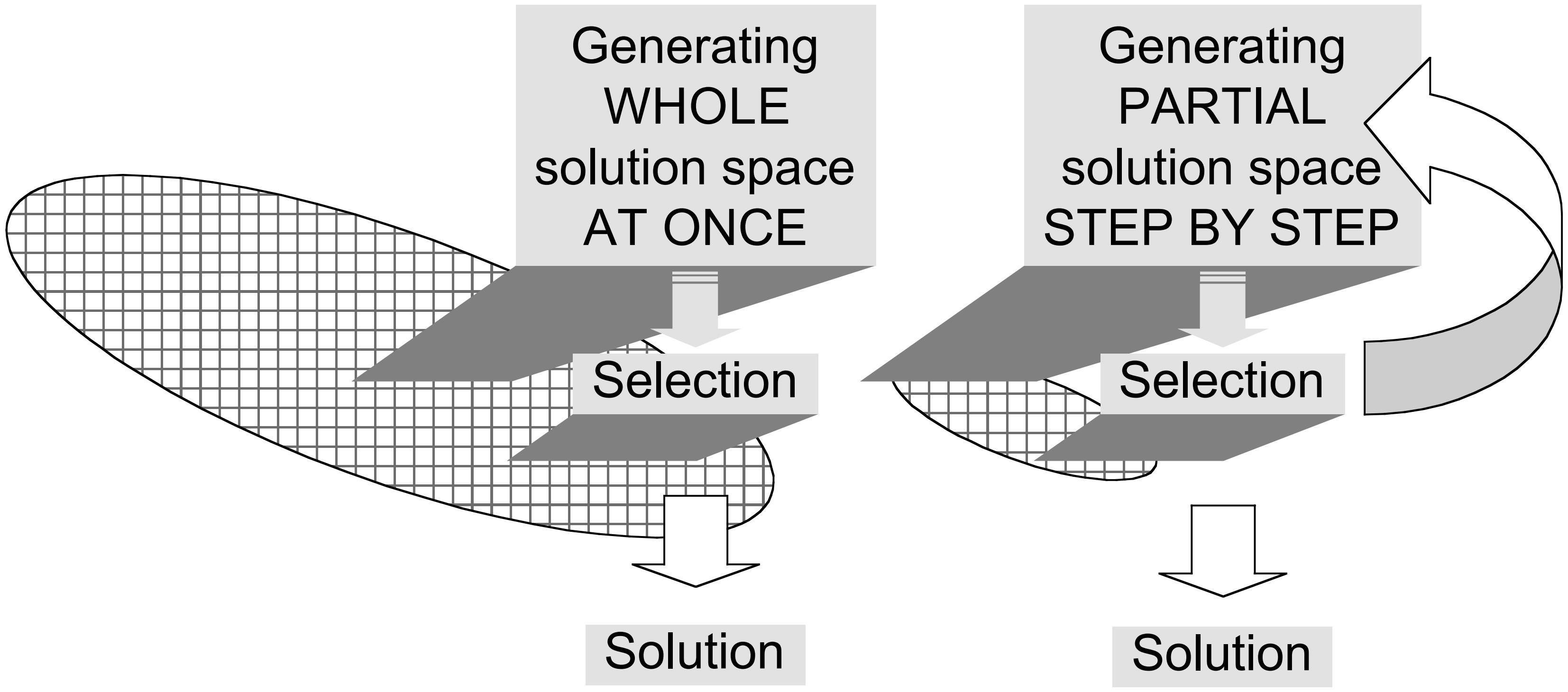
  Å combinatorial optimization

  large size **Å** increase yield and decrease error
- one-pot reaction **Å**
  autonomous computation Å self-organization
  - DNA tiles by Winfree
  - DNA automata by Hagiya et al.
  - DNA boolean circuits by Ogihara and Ray
- theoretical analyses
  - complexity
  - formal languages

Brute Force v.s. Dynamic Programming DNA Computers

Brute Force
Large pool size
Low reaction rate

Dynamic Programming
small pool size
High reaction rate



3-CNF SAT Solution on DP DNA Computer

Problem : 4 variables, 10 clauses

$$(x_1 \int\ x_2 \int\ x_3) \mid (x_1 \int \downarrow x_2 \int\ x_3) \mid$$
$$(\downarrow x_1 \int\ x_2 \int \downarrow x_3) \mid (\downarrow x_1 \int \downarrow x_2 \int \downarrow x_3) \mid$$
$$(x_1 \int \downarrow x_3 \int \downarrow x_4) \mid (\downarrow x_1 \int\ x_2 \int \downarrow x_4) \mid$$
$$(\downarrow x_1 \int\ x_3 \int \downarrow x_4) \mid (x_2 \int\ x_3 \int\ x_4) \mid$$
$$(x_2 \int \downarrow x_3 \int\ x_4) \mid (\downarrow x_2 \int \downarrow x_3 \int\ x_4)$$

Solution :

YES

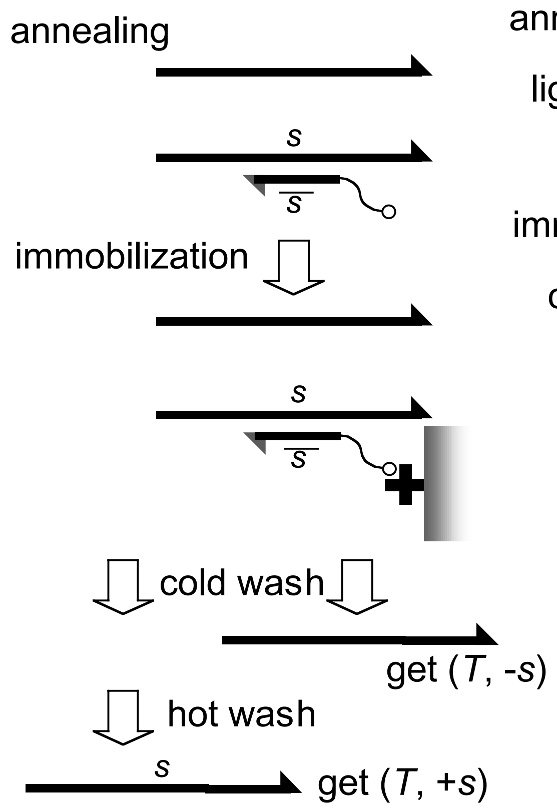
$\{X_1^T X_2^T X_3^F X_4^F\}$

Basic Operations for Dynamic Programming DNA Computers

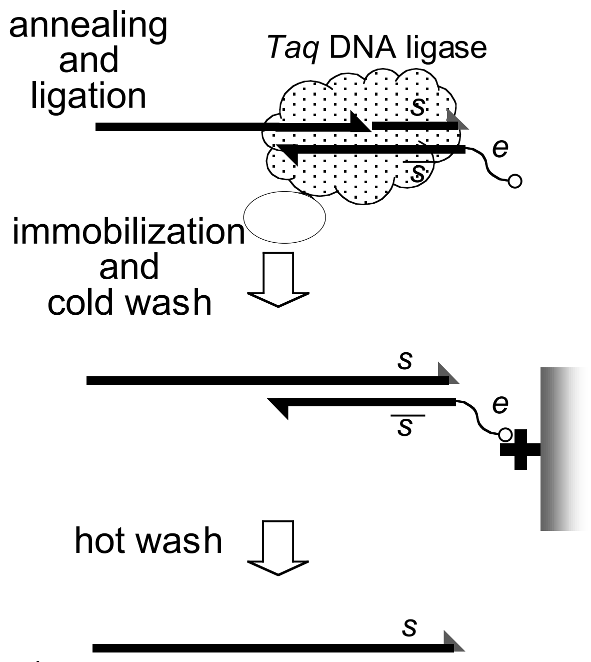
- get (T, $+s$), get (T, $-s$)
  get DNA molecules with a subsequence s (without s) in a tube T
- append (T, s, e)
  append a subsequence s at the end of DNA molecules with a splint e in a tube T
- merge (T, T_1, T_2, …, T_n)
  merge DNA molecules in tubes T_1, T_2, …, T_n into a tube T
- amplify(T, T_1, T_2, …, T_n)
  amplify DNA molecules in a tube T and divide them into tubes T_1, T_2, …, and T_n
- detect(T)
  detect DNA molecule in a tube T

Implementation of Basic Operations

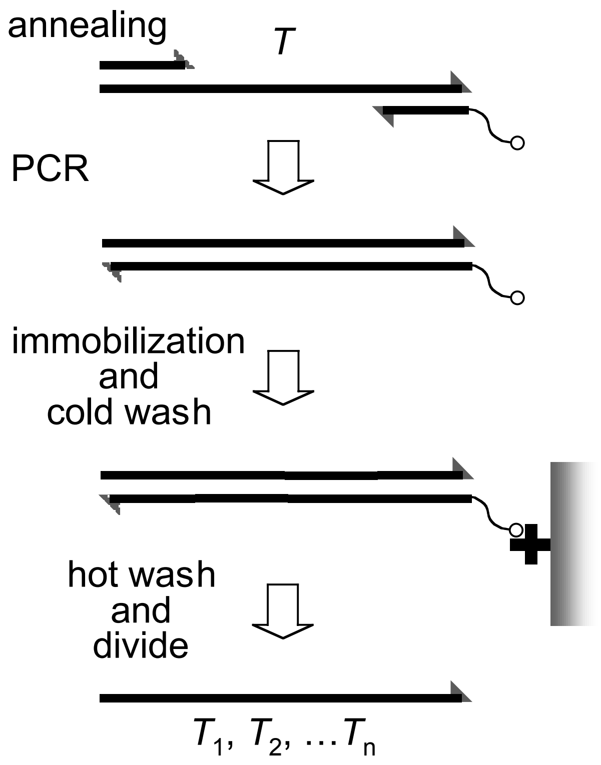
get (T, +s), get (T, -s)



append (T, s, e)



amplify (T, T_1, T_2, …T_n)



DP algorithm for 3CNF-SAT on DNA Computers

```
**procedure** dna 3sat $(u_1, v_1, w_1, \ldots, u_m, v_m, w_m)$
**begin**
    $T_2^T = \{X_1^T X_2^T, X_1^F X_2^T\}$;   input $(T_2^T)$;   $T'^T_2 = \{X_1^T X_2^T, X_1^F X_2^T\}$;   input $(T'^T_2)$;
    $T_2^F = \{X_1^T X_2^F, X_1^F X_2^F\}$;   input $(T_2^F)$;   $T'^F_2 = \{X_1^T X_2^F, X_1^F X_2^F\}$;   input $(T'^F_2)$;
    **for** $k = 3$ **to** $n$ **do**
        merge $(T_w^T, T_{k-1}^T, T_{k-1}^F)$;   merge $(T_w^F, T'^T_{k-1}, T'^F_{k-1})$;
        **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** $w_j = '\downarrow x_k'$ **then**
                $T_w^T$ = getuvsat $(T_w^T, u_j, v_j)$;
            **end**
        **end**
        $T^T$ = append $(T_w^T, X_k^T, \overline{X_{k-1}^{T/F} X_k^T})$;   $T^F$ = append $(T_w^F, X_k^F, \overline{X_{k-1}^{T/F} X_k^F})$;
        amplify $(T^T, T_k^T, T'^T_k)$;   amplify $(T^F, T_k^F, T'^F_k)$;
    **end**
    output (detect $(T_n)$);
**end**
```

```
**procedure** getuvsat $(T, u, v)$
**begin**
    input $(T)$;
    $T_u^T$ = get $(T, +X_u^T)$;   $T'^F_u$ = get $(T, -X_u^T)$;
    $T_u^F$ = get $(T'^F_u, +X_u^F)$;
    $T_v^T$ = get $(T_u^F, +X_v^T)$;
    merge $(T^T, T_u^T, T_v^T)$;
    output $(T^T)$;
**end**
```

Number of operations

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

An Amount of DNA for Computation
Brute Force v.s. Dynamic Programming

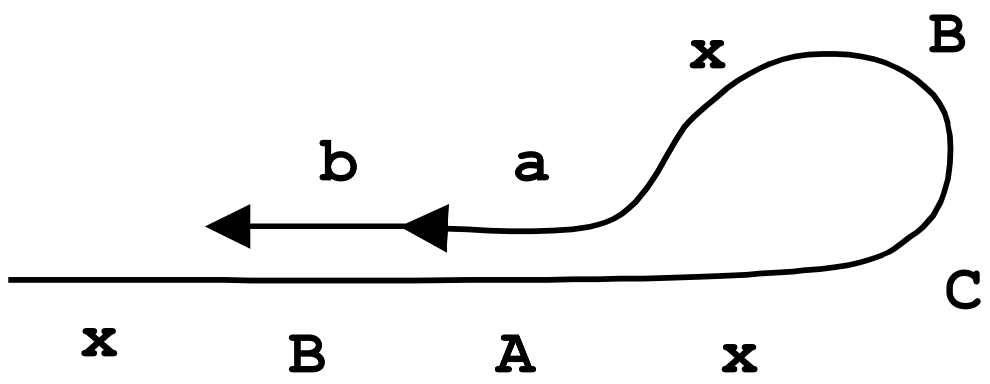
100 variable 3-CNF SAT

Adleman-Lipton's Brute Force DNA Computers	**Dynamic Programming DNA Computers**
2×10^{12} g of dsDNA 1×10^{12} g of ssDNA	**2×10^{-3} g of dsDNA** **(1×10^{-3} g of ssDNA)**

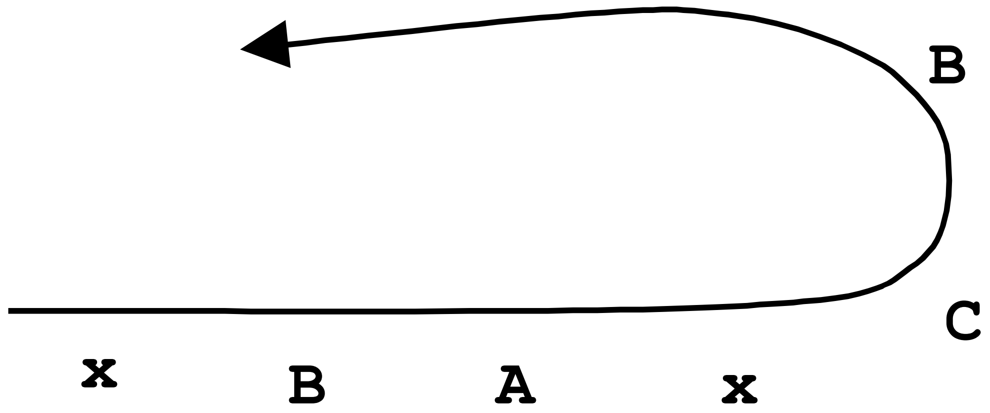
Autonomous Computation Self-Organization

- computational power of assembly (Winfree)
  - string _ regular
  - tree _ context-free
  - tile _ universal
- autonomous state transition (Hagiya et al.)
  - Whiplash PCR
- applications _ not only combinatorial optimization, but also
  - nanotechnology
  - genetic analysis

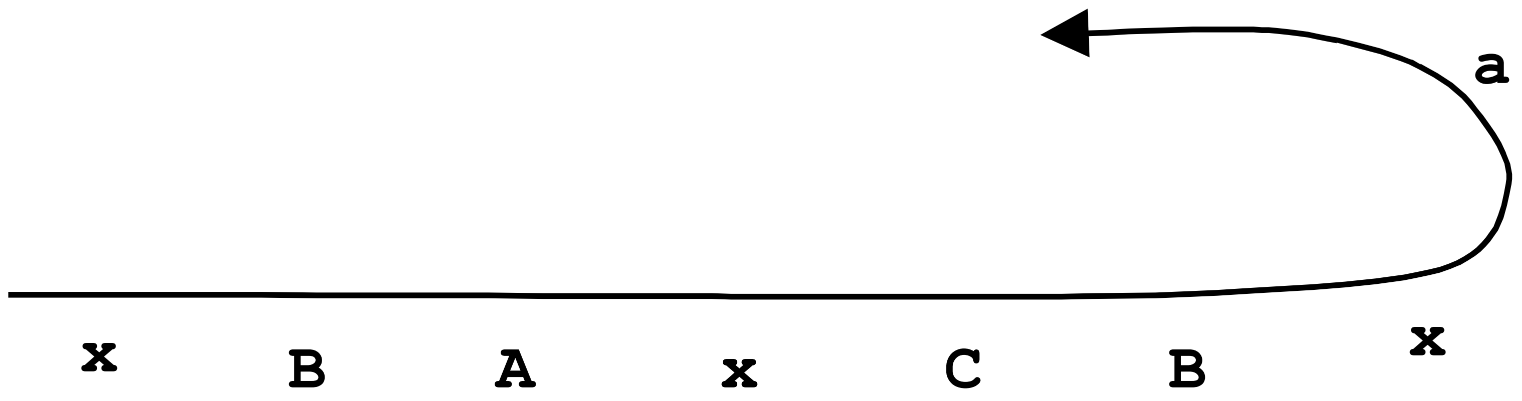
Whiplash PCR



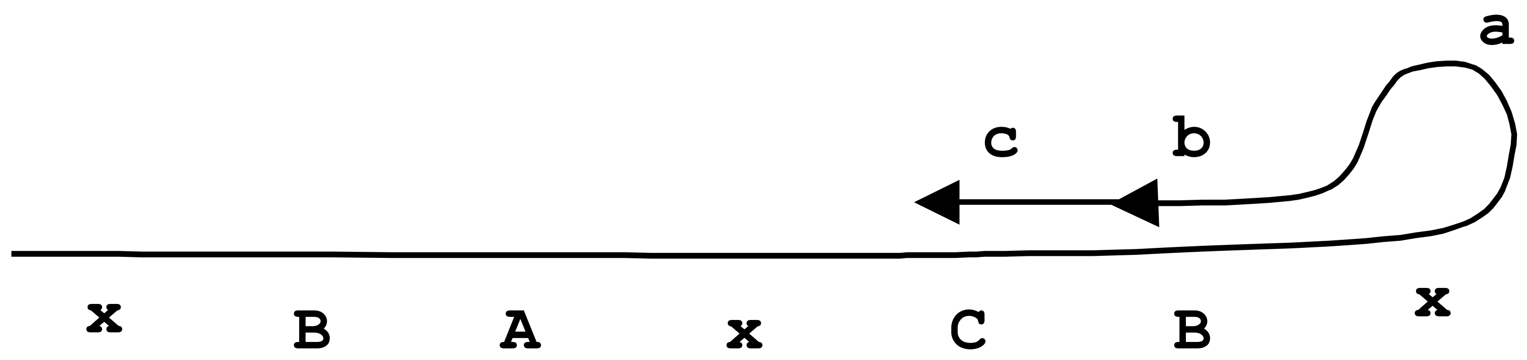
Whiplash PCR



Whiplash PCR



Whiplash PCR



DNA State Machine

- Transition table:

$$\boxed{5'-\mathbf{stopper}-state'_1-state_1-\cdots\cdots-\mathbf{stopper}-state'_n-state_n-3'}$$

  stopper: stopper sequence

  $state'_i-state_i$: state pair

  $$\begin{cases} state_i\text{: state before transition} \\ state'_i\text{: state after transition} \end{cases}$$

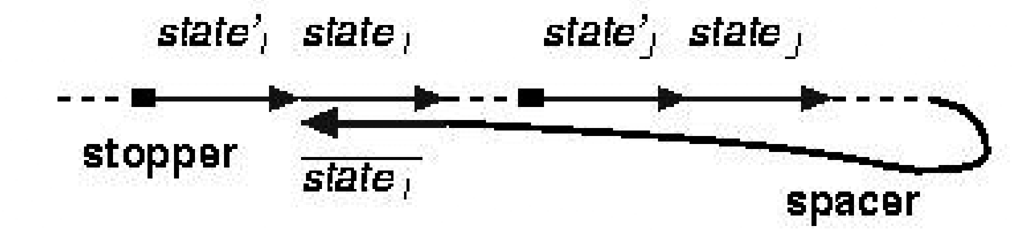
- Current state:

$$\boxed{5'-transition\text{-}table-\mathbf{spacer}-\overline{state_i}-3'}$$

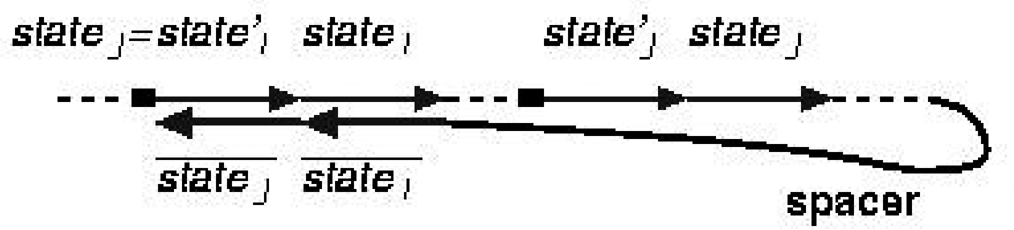
State Transitions by Polymerization

Whiplash PCR

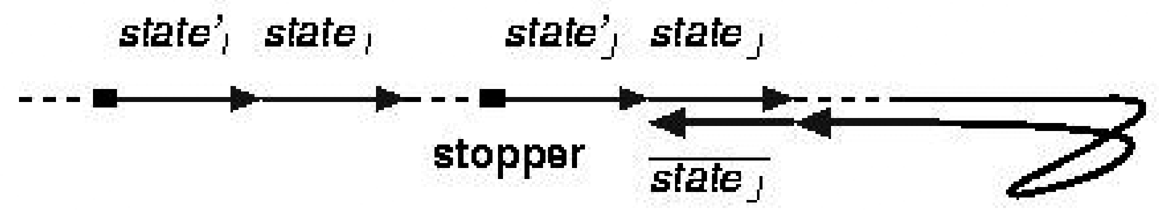
a:



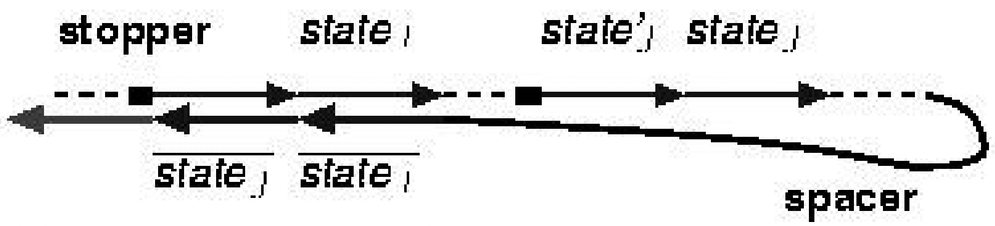
b:



c:



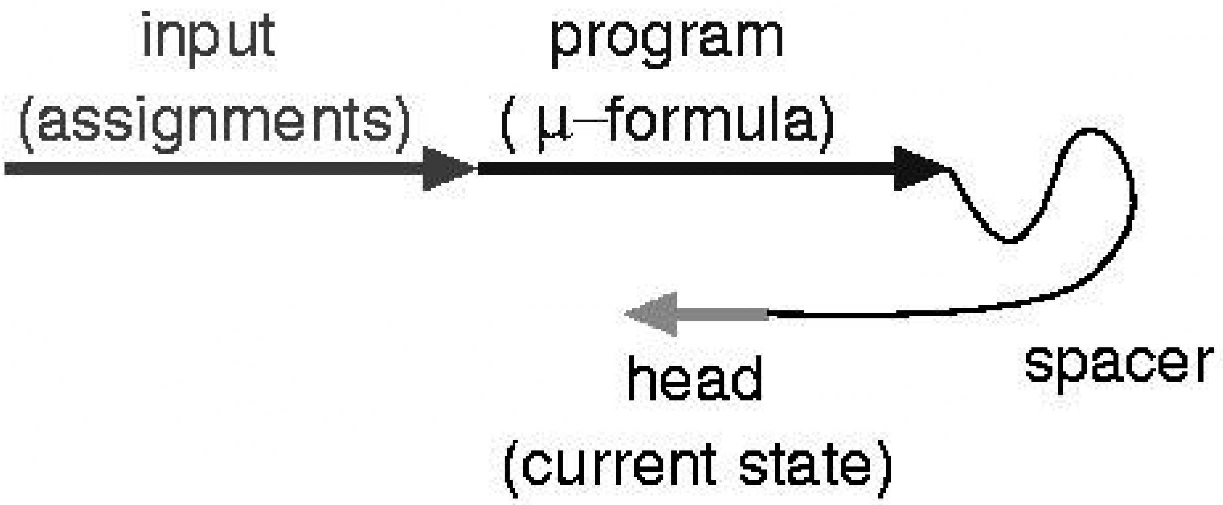
Polymerization Stop



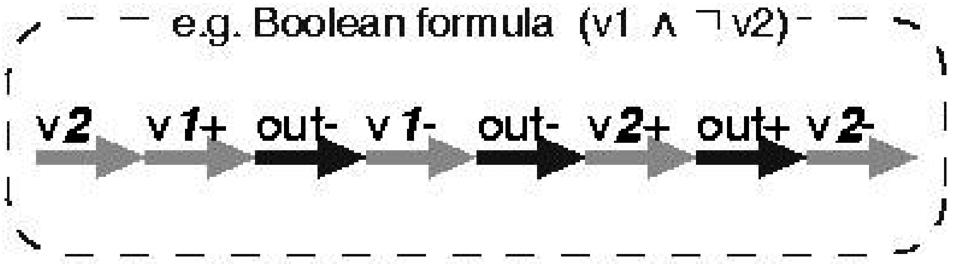
- States are encoded with **three** out of the four common deoxyribonucleotides (dATP, dGTP, dCTP and dTTP).
- A repetition of the missing deoxyribonucleotide works as a **stopper sequence** in polymerization.
- The polymerization buffer contains only complements of these **three** deoxyribonucleotides.

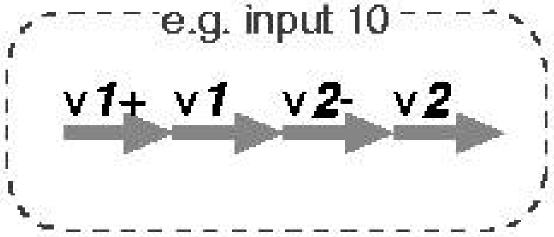
Evaluating Boolean μ-formulas

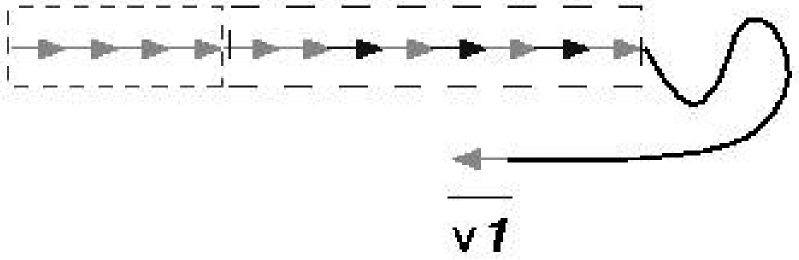
A μ–formula is a Boolean formula in which each input variable occurs at most once.



Encoding


e.g. Boolean formula (v1 ∧ ¬ v2)
v2
v1+
out-
v1-
out-
v2+
out+
v2-


e.g. input 10
v1+
v1
v2-
v2


$\overline{v1}$

VNA: Simulator for Virtual DNA

- a **concrete computational** model based on assembly and state transition
- abstract, but sufficiently physical

  bridging gaps between abstract models and real reactions

  molecule _ hybrid of virtual strands

```
abcd
  ||
  CDEF
```

- reactions

– hybridization	assembly
– denaturation	decomposition
– restriction	decomposition
– ligation	state transition
– self-hybridization	state transition
– extension	state transition

VNA (cont.)

- **purposes**
  - verify feasibility of algorithms for DNA computing
  - verify validity of molecular biology experiments (e.g., PCR experiments)
  - parameter fitting in molecular biology experiments
- examples
  - Ogihara and Ray’s computation of boolean circuits
  - Winfree’s construction of double-crossover units
  - **PCR experiments**
- implementation
  - Java **Å** executable as an applet

VNA (cont.)

- methods
  - combinatorial enumeration
  - continuous simulation (diff. eq.)

  $\Big\}$ **unify**
- avoiding combinatorial explosion
- contributions in simulation technology
  - threshold
  - stochastic
- parameter fitting by GA
  - optimizing amplification in PCR experiments

Goals of Molecular Computing _ concluding remark

- analysis of computational power of biomolecules
  - understanding life
    - origin of life
    - **wet artificial life**
  - engineering applications
    - combinatorial optimization
    - nanotechnology
    - genetic analysis
- **new computational models Å simulation technology**

digital Å analog Å physical