Cross Pairing PCR (XPCR)

Dr Giuditta Franco

Department of Computer Science, University of Verona, Italy
giuditta.franco@univr.it

In the context of new bio-techniques, proposed for a more precise and reliable DNA computation, novel XPCR-based recombination (extraction, mutagenesis, concatenation) methods have been proposed as combinatorial algorithms, and validated by experiments.

What is the output?

Given a string γ, we compute $XPCR_\gamma(\alpha, \overline{\beta})$.

Input pool: $P = \{\alpha x \beta \mid |\alpha x \beta| = n\}$.

1. $(P_1, P_2) := split(P)$;
2. $P_1 := PCR(\alpha, \overline{\gamma})(P_1) \wedge P_2 := PCR(\gamma, \overline{\beta})(P_2)$;
3. P:= $Mix(P_1, P_2)$;
4. P: = $\cup_{k<n} El_k(P)$;
5. P: = $PCR(\alpha, \overline{\beta})(P)$

XPCR$_\gamma$ procedure for γ-recombinations

Initial pool: heterogeneous pool of DNA double strands, all the same length *n*, prefix α, suffix β.

Given a string γ, *XPCR*$_\gamma$ recombines all the strings that contain (even more than one occurrence of) γ as a substring:

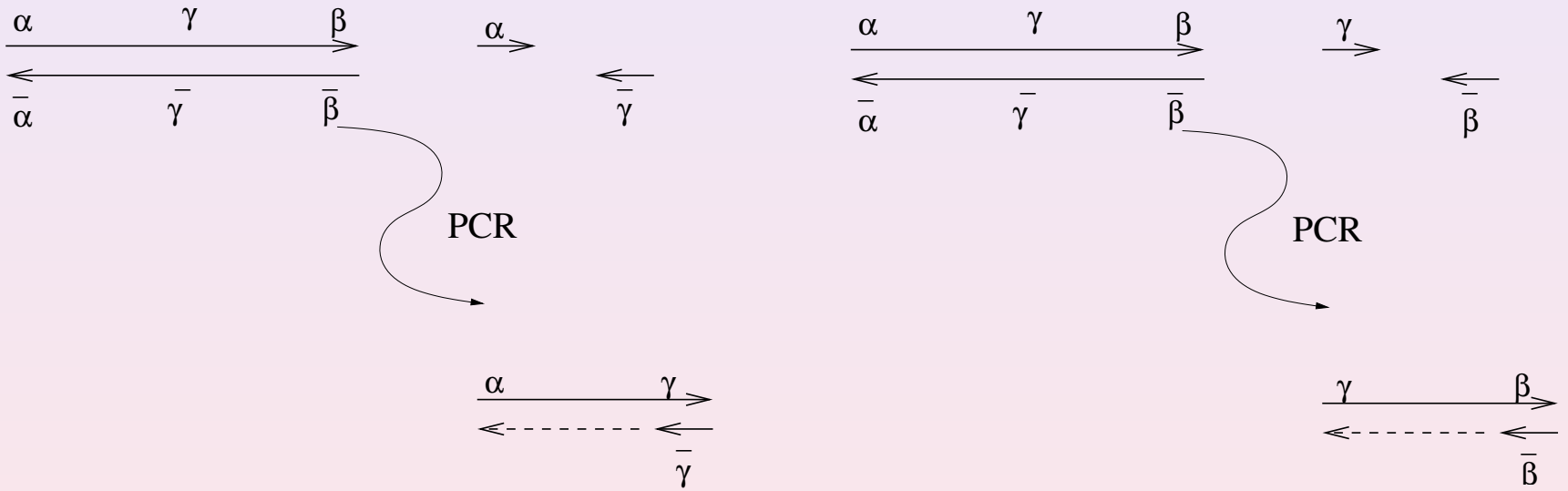
$$\underline{\alpha\phi\gamma}\psi\beta,\ \alpha\delta\underline{\gamma\eta\beta} \xrightarrow{r_\gamma} \underline{\alpha\phi\gamma\eta\beta},\ \alpha\delta\gamma\psi\beta,\ \alpha\phi\gamma\psi\beta,\ \alpha\delta\gamma\eta\beta$$

with $\alpha, \phi, \gamma, \psi, \delta, \eta, \beta$ strings over the alphabet.

$XPCR_\gamma$: Sequence copy

Input Pool P of α-prefixed and β-suffixed strings having length *n*

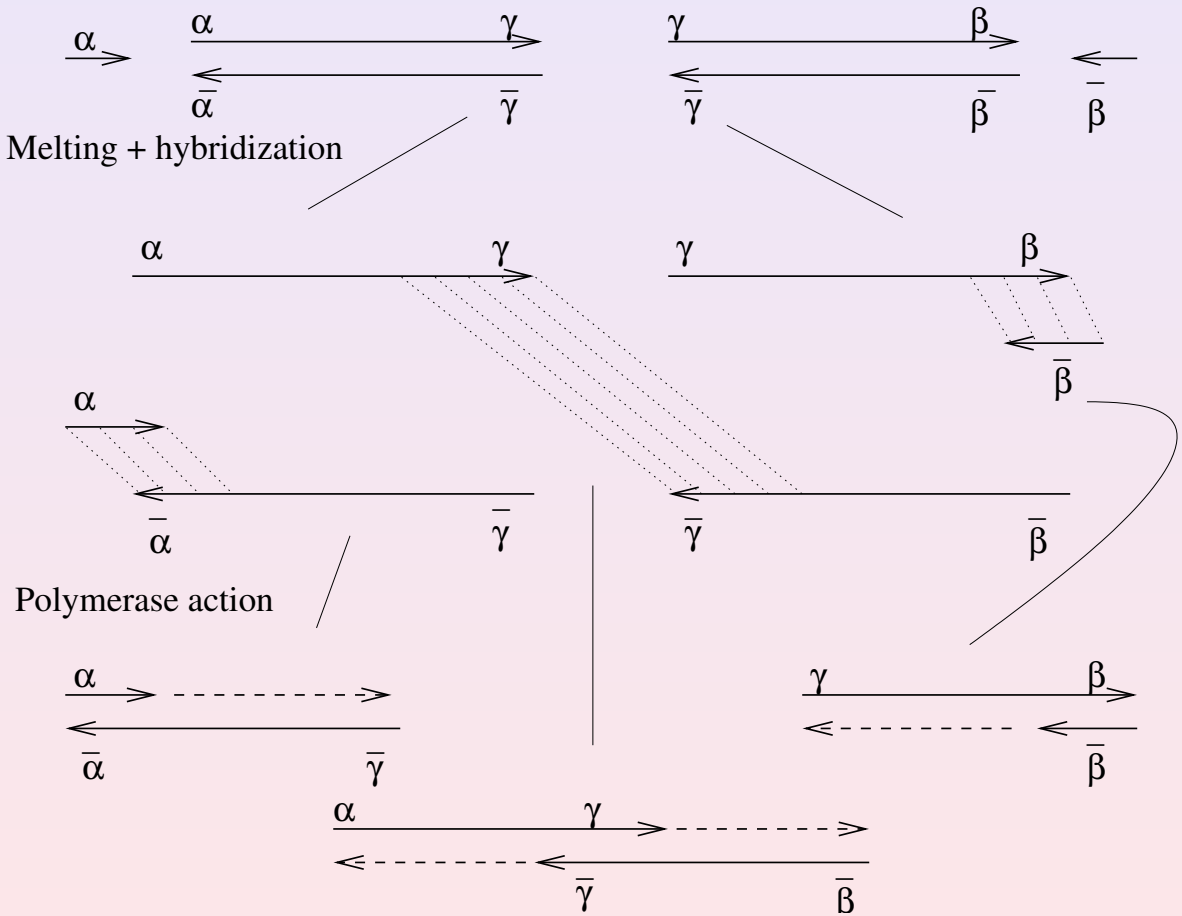
- **split** *P* into P_1 and P_2 (same approximate size)
- **apply** $PCR(\alpha, \overline{\gamma})$ to P_1 **and** $PCR(\gamma, \overline{\beta})$ to P_2



- perform **electrophoresis** on P_1 **and** on P_2 to eliminate the sequences of the initial length
- **mix** the two pools so obtained in a pool *P*

$XPCR_\gamma$: Sequence recombination

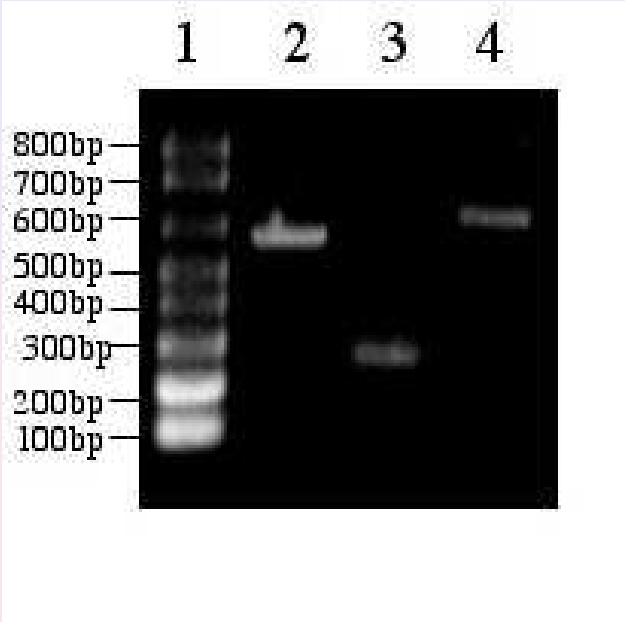
- **apply** $PCR(\alpha, \overline{\beta})$ to P



Output The pool P resulting from the previous step

Experiment for Testing the XPCR

RhoA is a gene involved in regulation and timing of cell division.



Electrophoresis results. 1: molecular size marker ladder (100b). 2: $\alpha\phi\gamma\psi$-strands of human RhoA (582bp), 3: $\gamma\psi\beta$-strands (253bp), 4: cross pairing amplification of $\alpha\phi\gamma\psi\beta$-strands (606bp): 606 = 582 +253 - 229.

About the overlapping string

We noticed that γ may be much longer than the primers (229 bps).

Long lengths guarantee it is different than other substrings, and allow many possible sequences for γ design, however under the following constrains: Tm comparable with primers α and $\overline{\beta}$ (rich of CG).

Operation general enough to assume freedom on the choice of possible γ. Of course, it works for multiple occurrences of γ.

XPCR as an innovative biotechnology

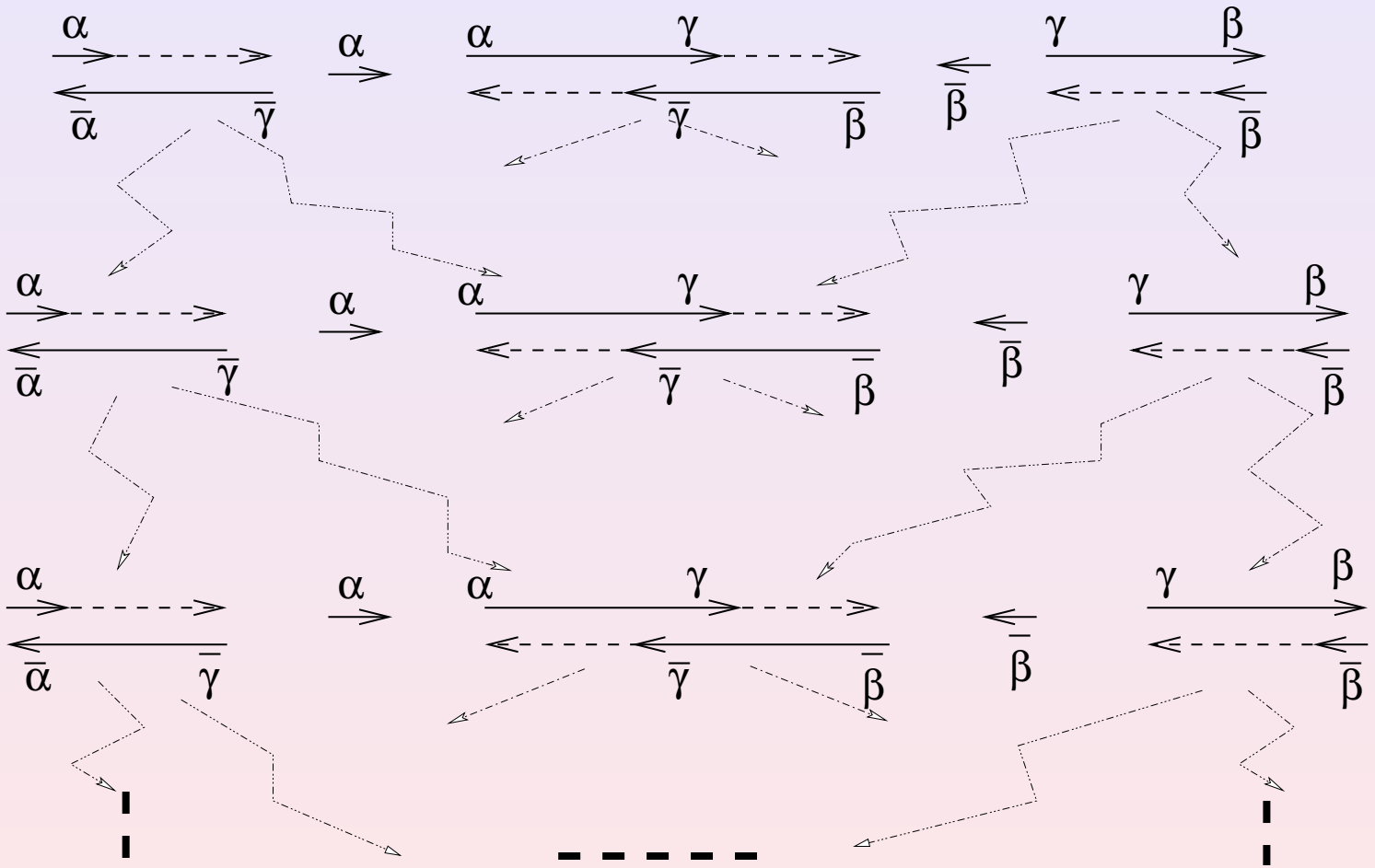
XPCR technique proved to be convenient in literature, in terms of *generation speed*, *material waste* and *efficiency (error)*.

It is cheap and scalable, and can be easily automatized.

It performs a super-exponential amplification, with an optimal efficiency in terms of space and time.

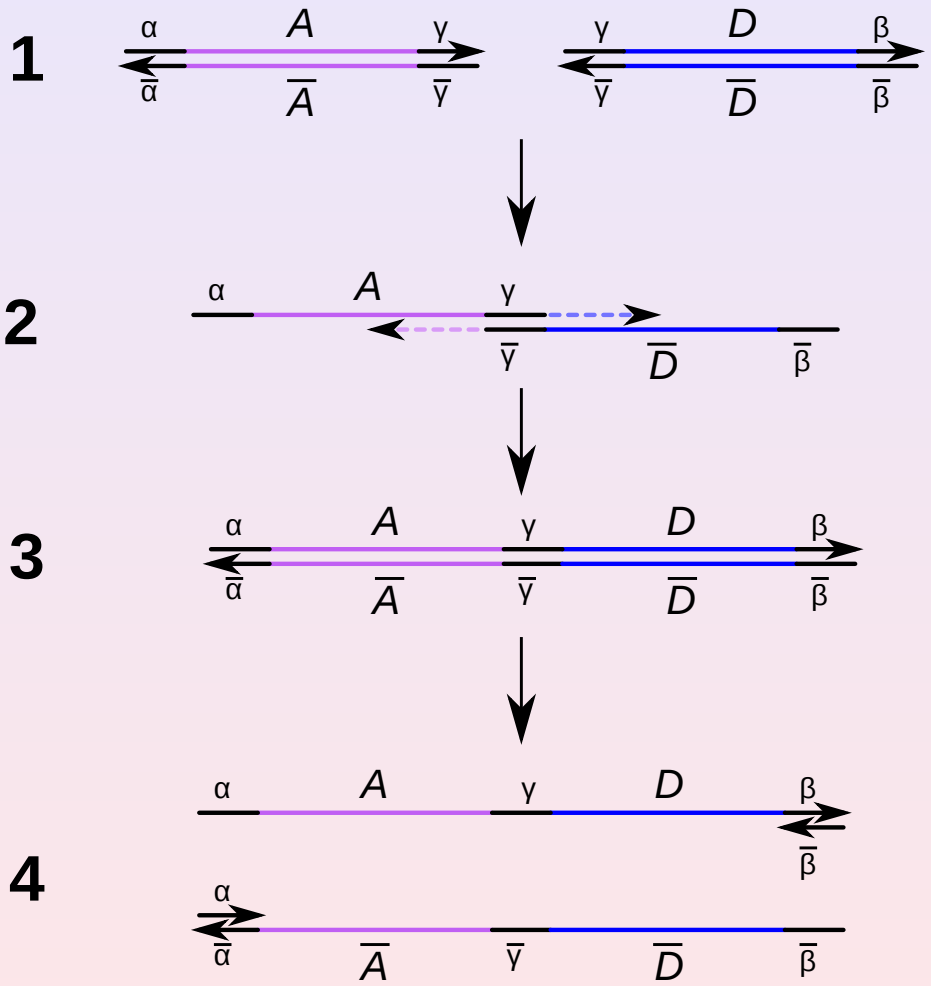
Bad news: PCR amplifies (a lot) even if the two primers do not pair parts of the same string.

XPCR Process



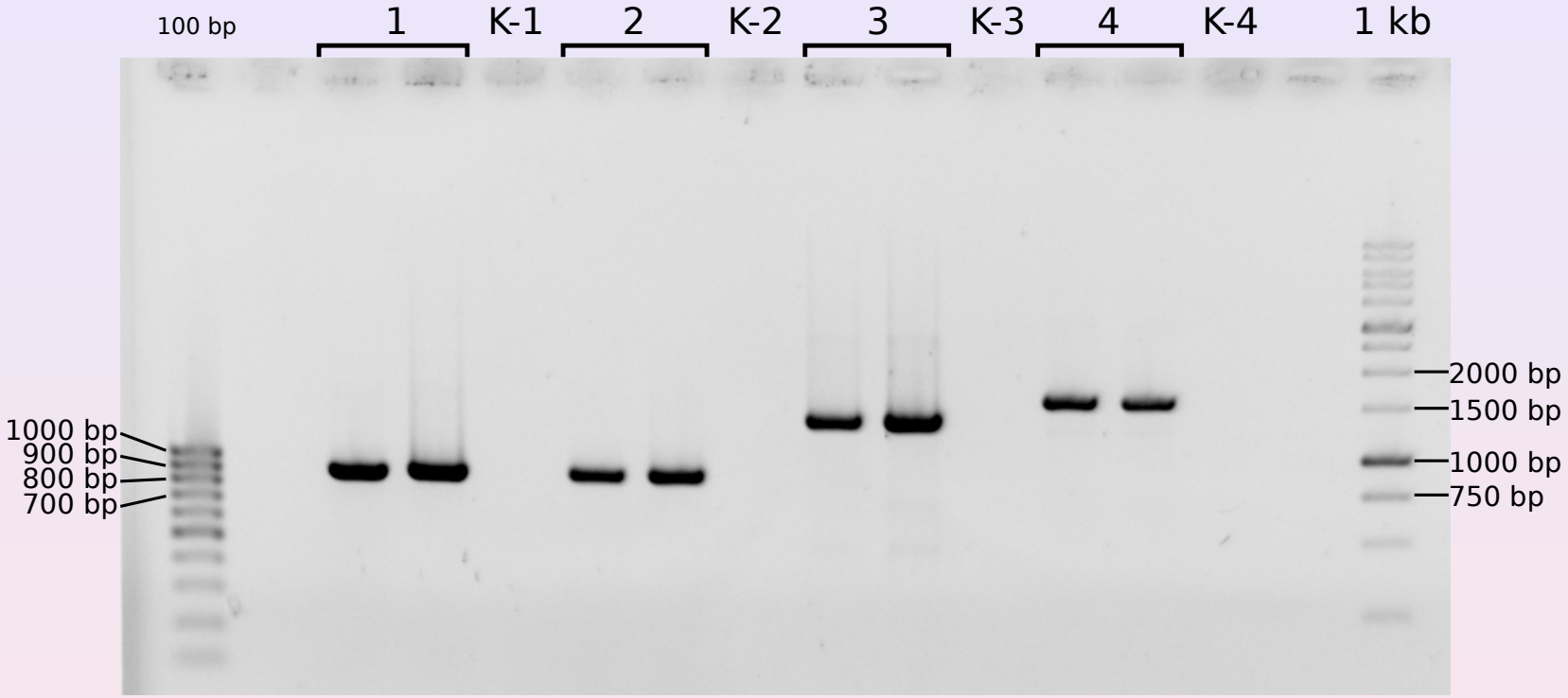
Long (central) strands are seeds of (super)exp amplification

Successful 2-genes concatenation:



Strands in (3) are seeds of (super)exp amplification

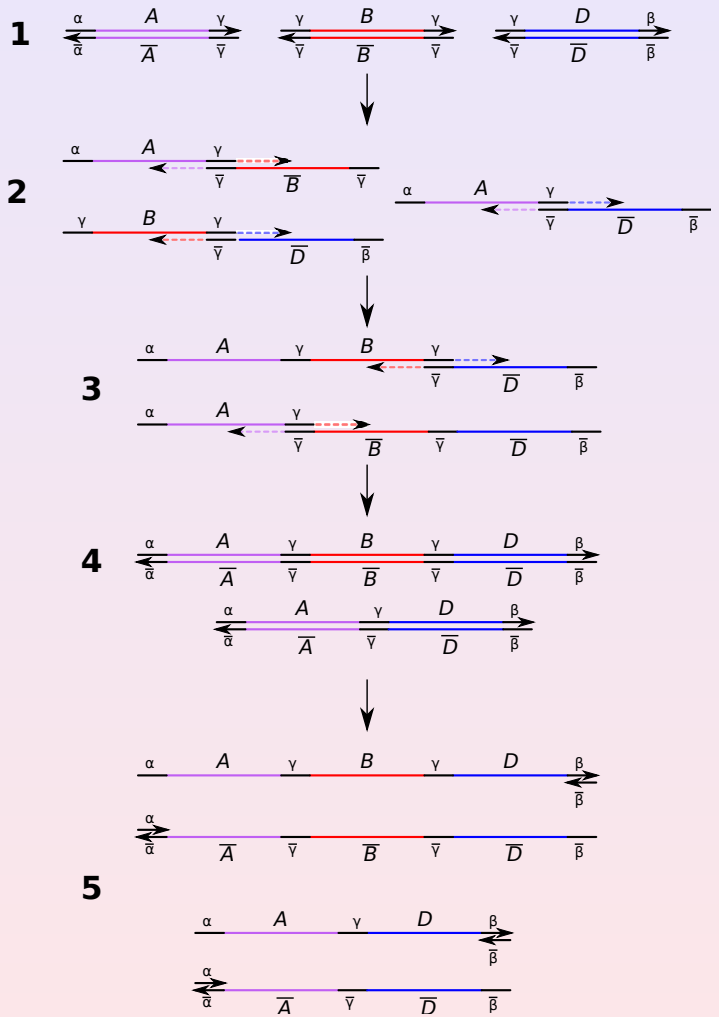
Experimental results for (long) lenghts



Experiments on specific genes from the Burkholderia operon [1]. Reactions with primers $(\alpha, \overline{\beta})$ and templates $\alpha D\gamma$ and $\gamma B\beta$, $\alpha B\gamma$ and $\gamma D\beta$, $\alpha A\gamma$ and $\gamma B\beta$, $\alpha D\gamma$ and $\gamma A\beta$. Expected products $\alpha D\gamma B\beta$, $\alpha B\gamma D\beta$, $\alpha A\gamma B\beta$, $\alpha D\gamma A\beta$, resp. 892, 1393, 1600 bps.

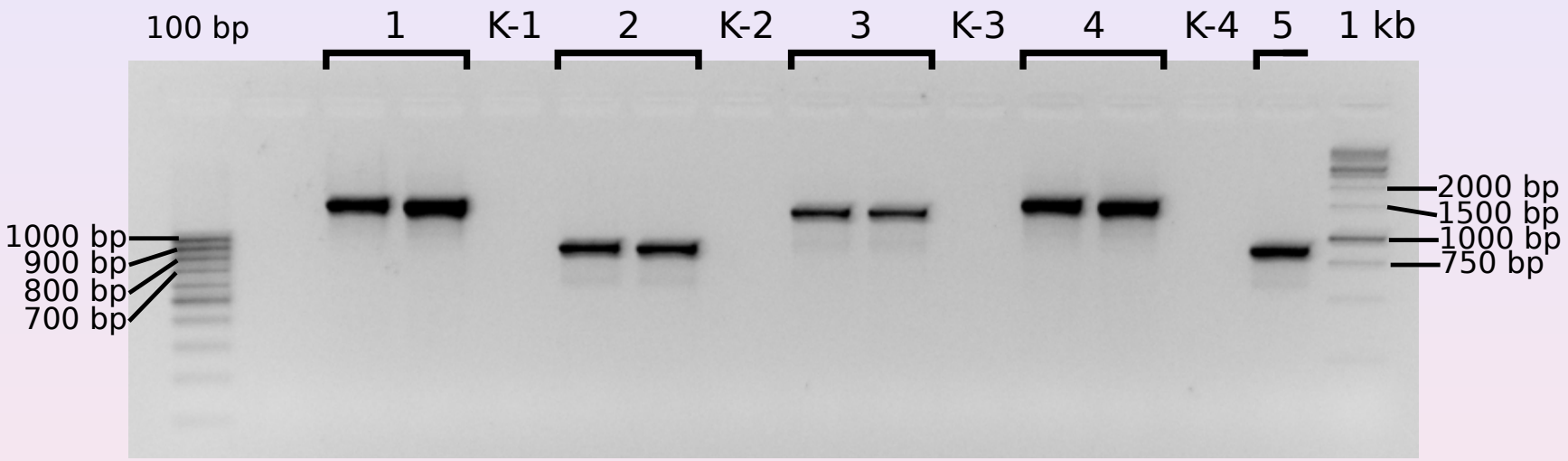
[1] courtesy of F. Bellamoli, Master Thesis, Univ. of Verona, Feb 2014.

Successful 3-genes concatenation



Iteration of two steps is necessary to get only long strings in (4)

Experimental results



Experiments on specific genes from the Burkholderia operon [2]. Reactions (at different ratios) with primers $(\alpha, \overline{\beta})$, and templates $\alpha A\gamma + \gamma B\gamma + \gamma D\beta$ and their permutations. Expected products 1600 long.

[2] courtesy of F. Bellamoli, Master Thesis, Univ. Verona, Feb 2014

Multiple XPCR

Multiple forms of XPCR also work, referred to as *n*-XPCR, when *n* different types of DNA double strings are concatenated in the same test tube.

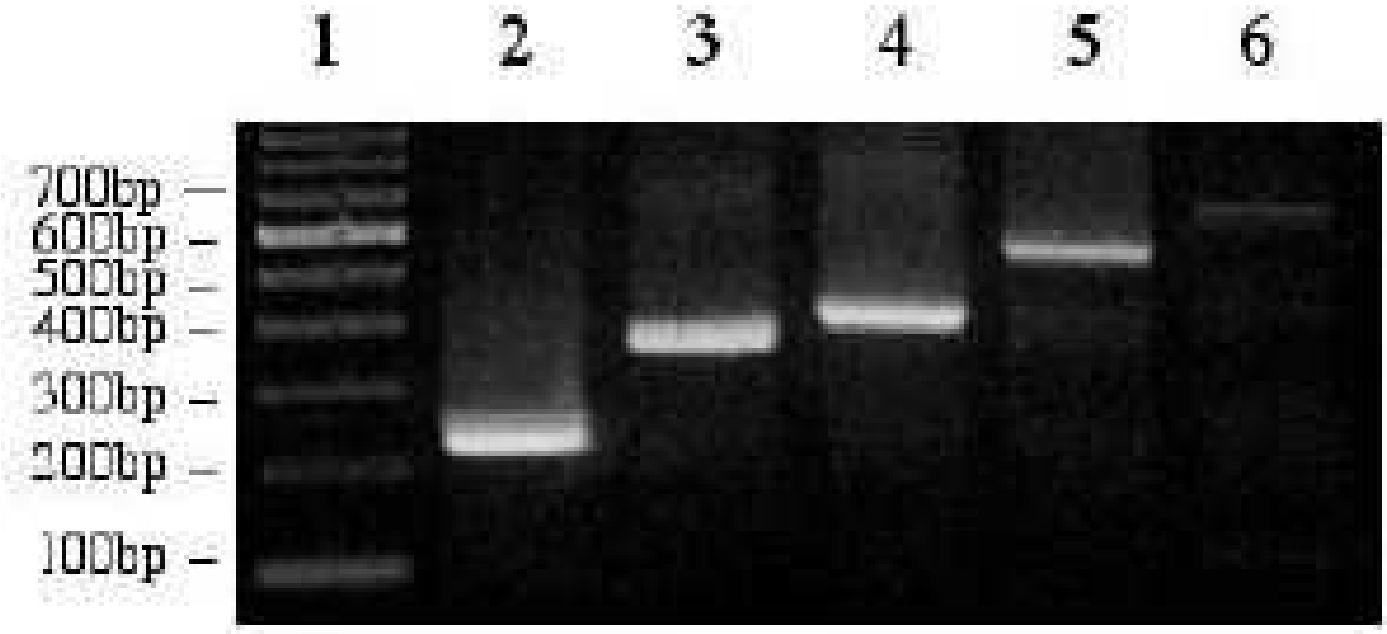
For example $n = 4$, given a pool

$$P = \{\alpha\delta_1\gamma_1, \gamma_1\delta_2\gamma_2, \gamma_2\delta_3\gamma_3, \gamma_3\delta_4\beta\}$$

the final $PCR_{(\alpha,\beta)}(P)$ provides an exponential amplification of the string $\alpha\delta_1\gamma_1\delta_2\gamma_2\delta_3\gamma_3\delta_4\beta$.

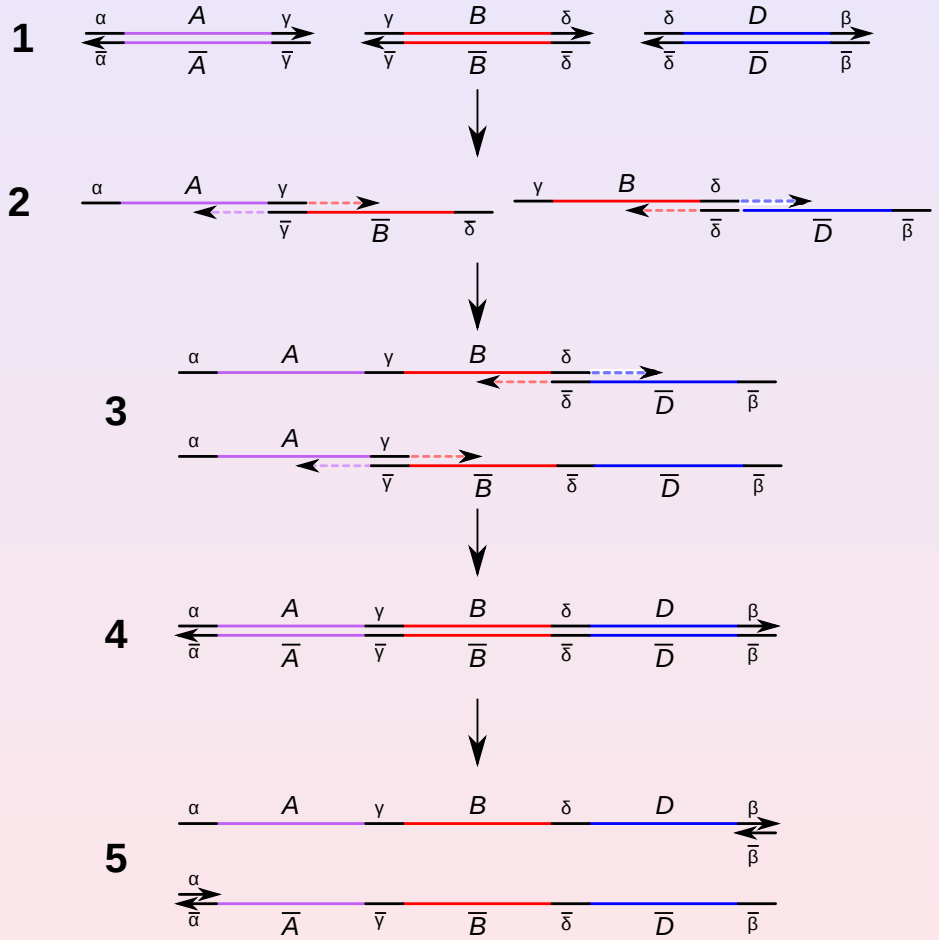
Experimental (limited) success

Multiple XPCR were performed for n=4, 5, 6, 7, 8.



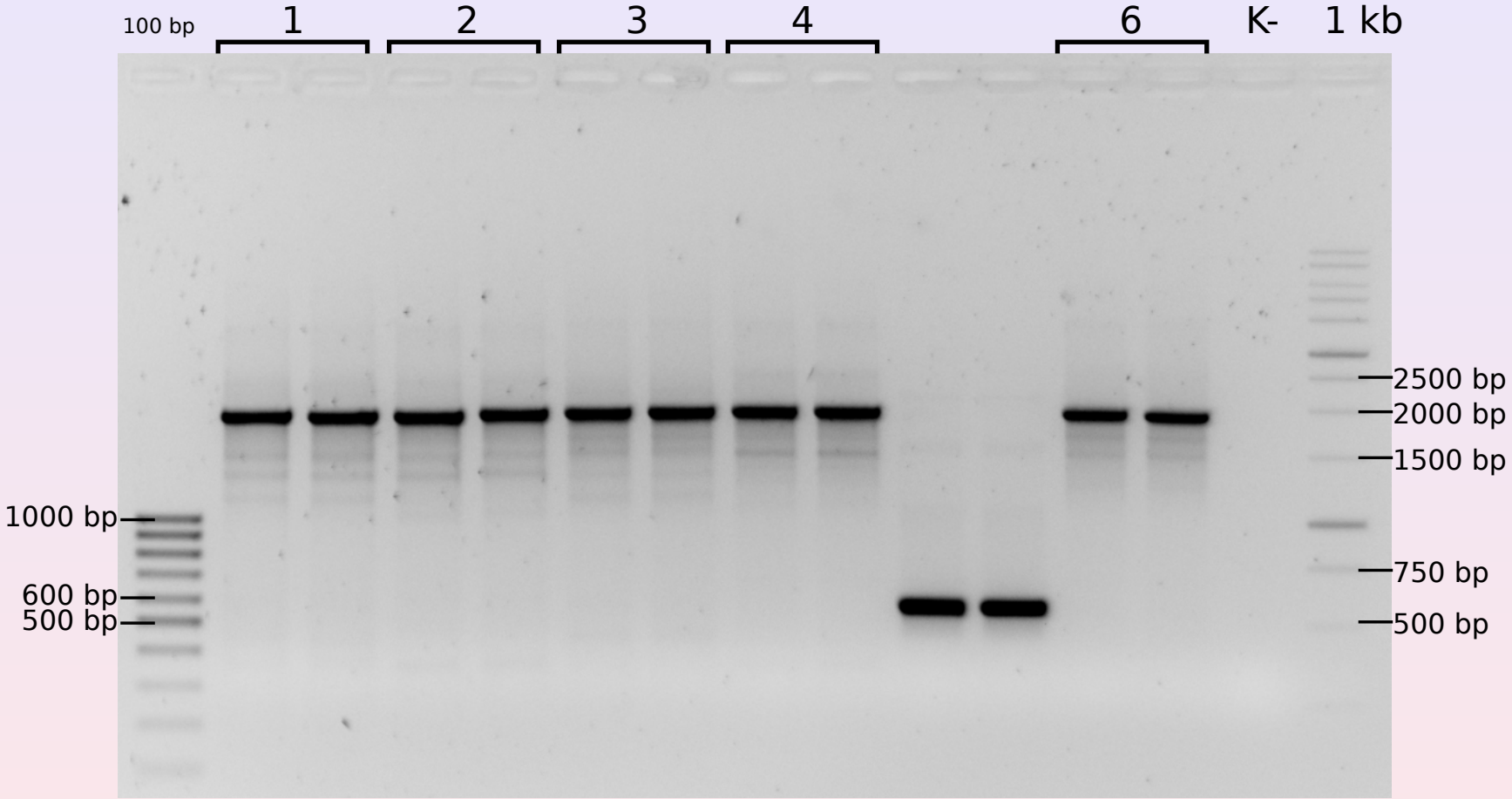
Expected lengths: 217, 356, 407, 538, 626.

Parallel 3-genes concatenation



Parallel concatenation by different overlapping strings γ, δ

Successful experimental results



Experiments on specific genes from the Burkholderia operon [3].
Reactions with primers $(\alpha, \overline{\beta})$ and templates $\alpha A\gamma + \gamma B\delta + \delta D\beta$, $\alpha A\gamma + \gamma D\delta + \delta B\beta$, $\alpha B\gamma + \gamma A\delta + \delta D\beta$, $\alpha B\gamma + \gamma D\delta + \delta A\beta$,

H systems

Tom Head '87 - Splicing Systems $\mathcal{S} = (\Sigma, A, R)$.

In R: $\underline{wpq}x,\ yu\underline{vz} \xrightarrow{r_{(p,q,u,v)}} \underline{wpv}z,\ yuqx,$ with $w, p, q, x \in \Sigma^\star$

Null context splicing rule r_p has $p = u$ and $qv = \lambda$:

$$\underline{\phi p}\psi,\ \delta\underline{p\eta} \xrightarrow{r_p} \underline{\phi p\eta},\ \delta p\psi, \quad \text{where } \phi, p, \psi, \delta, \eta \in \Sigma^\star$$

$XPCR_\gamma$ more general than the 30 enzymes able to perform the operation above.

Splicing Examples

Initial sequences: $l_1 = X_1X_2X_3X_4X_5X_6$, $l_2 = Y_1Y_2Y_3Y_4Y_5Y_6$, $l_3 = X_1Y_2X_3Y_4X_5Y_6$, $l_4 = Y_1X_2Y_3X_4Y_5X_6$.

1. $l_1,\ l_4 \xrightarrow{r_{X_2}} X_1X_2Y_3X_4Y_5X_6,\ Y_1X_2X_3X_4X_5X_6,$
   $l_2,\ X_1X_2Y_3X_4Y_5X_6 \xrightarrow{r_{Y_5}} Y_1Y_2Y_3Y_4Y_5X_6,\ \mathbf{X_1X_2Y_3X_4Y_5Y_6}.$
2. $l_2,\ l_4 \xrightarrow{r_{Y_3}} Y_1Y_2Y_3X_4Y_5X_6,\ Y_1X_2Y_3Y_4Y_5Y_6,$
   $l_1,\ Y_1Y_2Y_3X_4Y_5X_6 \xrightarrow{r_{X_4}} X_1X_2X_3X_4Y_5X_6,\ \mathbf{Y_1Y_2Y_3X_4X_5X_6}.$
3. $l_1,\ l_3 \xrightarrow{r_{X_5}} X_1Y_2X_3Y_4X_5X_6, X_1X_2X_3X_4X_5Y_6,$
   $l_2,\ X_1Y_2X_3Y_4X_5X_6 \xrightarrow{r_{Y_2}} X_1Y_2Y_3Y_4Y_5Y_6,\ \mathbf{Y_1Y_2X_3Y_4X_5X_6}$

A library generated in an NCH system

Let us start with the set of four sequences $J = \{l_1, l_2, l_3, l_4\}$, where (e.g., $n = 6$), $l_1 = X_1X_2X_3X_4X_5X_6$, $l_2 = Y_1Y_2Y_3Y_4Y_5Y_6$, $l_3 = X_1Y_2X_3Y_4X_5Y_6$, $l_4 = Y_1X_2Y_3X_4Y_5X_6$.

Let us consider the *null context splicing rules* r_γ [T. Head, '87]:

$$\underline{\phi\gamma}\psi,\ \delta\underline{\gamma\eta} \xrightarrow{r_\gamma} \underline{\phi\gamma\eta},\ \delta\gamma\psi,\ \phi\gamma\psi,\ \delta\gamma\eta$$

with $\phi, \gamma, \psi, \delta, \eta$ strings over the alphabet $\Sigma = \{A, T, C, G\}$.

Theorem

The n-dimensional library $\{\alpha_1 \cdots \alpha_n \mid \alpha_i \in \{X_i, Y_i\}, i = 1, \ldots, n\}$ *is the null context splicing language generated by the system* $\mathcal{N} = (\Sigma, J, R)$, *where* $J = \{l_1, l_2, l_3, l_4\}$, $R = \{r_{X_2}, r_{Y_2}, \ldots, r_{X_{n-1}}, r_{Y_{n-1}}\}$.

A library generated in an NCH system

Let us start with the set of four sequences $J = \{I_1, I_2, I_3, I_4\}$, where (e.g., $n = 6$), $I_1 = X_1X_2X_3X_4X_5X_6$, $I_2 = Y_1Y_2Y_3Y_4Y_5Y_6$, $I_3 = X_1Y_2X_3Y_4X_5Y_6$, $I_4 = Y_1X_2Y_3X_4Y_5X_6$.

Let us consider the *null context splicing rules* r_γ [T. Head, '87]:

$$\underline{\phi\gamma}\psi,\ \delta\underline{\gamma\eta} \xrightarrow{r_\gamma} \underline{\phi\gamma\eta},\ \delta\gamma\psi,\ \phi\gamma\psi,\ \delta\gamma\eta$$

with $\phi, \gamma, \psi, \delta, \eta$ strings over the alphabet $\Sigma = \{A, T, C, G\}$.

Theorem

The n-dimensional library $\{\alpha_1 \cdots \alpha_n \mid \alpha_i \in \{X_i, Y_i\}, i = 1, \ldots, n\}$ *is the null context splicing language generated by the system* $\mathcal{N} = (\Sigma, J, R)$, *where* $J = \{I_1, I_2, I_3, I_4\}$, $R = \{r_{X_2}, r_{Y_2}, \ldots, r_{X_{n-1}}, r_{Y_{n-1}}\}$.

A library generated in an NCH system

Let us start with the set of four sequences $J = \{I_1, I_2, I_3, I_4\}$, where (e.g., $n = 6$), $I_1 = X_1X_2X_3X_4X_5X_6$, $I_2 = Y_1Y_2Y_3Y_4Y_5Y_6$, $I_3 = X_1Y_2X_3Y_4X_5Y_6$, $I_4 = Y_1X_2Y_3X_4Y_5X_6$.

Let us consider the *null context splicing rules* r_γ [T. Head, '87]:

$$\underline{\phi\gamma}\psi,\ \delta\underline{\gamma\eta} \xrightarrow{r_\gamma} \underline{\phi\gamma\eta},\ \delta\gamma\psi,\ \phi\gamma\psi,\ \delta\gamma\eta$$

with $\phi, \gamma, \psi, \delta, \eta$ strings over the alphabet $\Sigma = \{A, T, C, G\}$.

Theorem

The n-dimensional library $\{\alpha_1 \cdots \alpha_n \mid \alpha_i \in \{X_i, Y_i\}, i = 1, \ldots, n\}$ *is the null context splicing language generated by the system* $\mathcal{N} = (\Sigma, J, R)$, *where* $J = \{I_1, I_2, I_3, I_4\}$, $R = \{r_{X_2}, r_{Y_2}, \ldots, r_{X_{n-1}}, r_{Y_{n-1}}\}$.

DNA Libraries

Goal: generating "efficiently" the combinatorial library of n binary numbers:

$$\{\alpha_1 \cdots \alpha_n \mid \alpha_i \in \{X_i, Y_i\},\ X_i, Y_i \in \{A, T, C, G\}^*,\ i = 1, \ldots, n\}$$

Methods generating libraries have biological, technological, and algorithmic relevance. Namely, to build up large memories, to study rearrangements of antibody genes, or to optimize operon structures.

Quaternary Recombination Algorithm

This method starts from α-prefixed and β-suffixed I_1, I_2, I_3 I_4, works in linear time, by using polymerase extension.

Let P_1 and P_2 be two copies of the pool

$$\{\alpha I_1 \beta,\ \alpha I_2 \beta,\ \alpha I_3 \beta,\ \alpha I_4 \beta\}$$

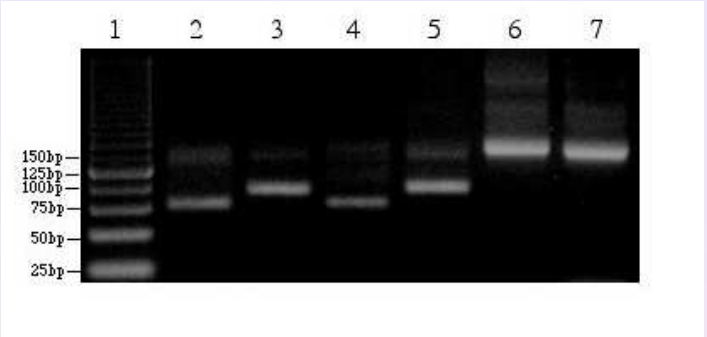
for $i = 2, 3, 4, 5$ **do**

- **perform** $XPCR_{X_i}$ on P_1 and $XPCR_{Y_i}$ on P_2 [4]
- **mix** the two pools into one $P := P_1 \cup P_2$, then **split** P randomly in two new pools P_1 and P_2

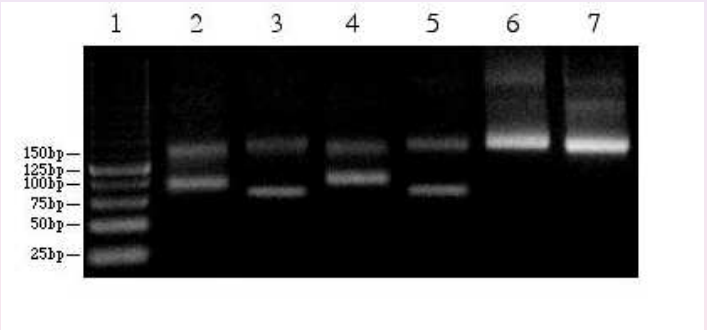
[4] Run together (no intermediate electrophoresis) have a worse efficiency and complexity

Experimental Results

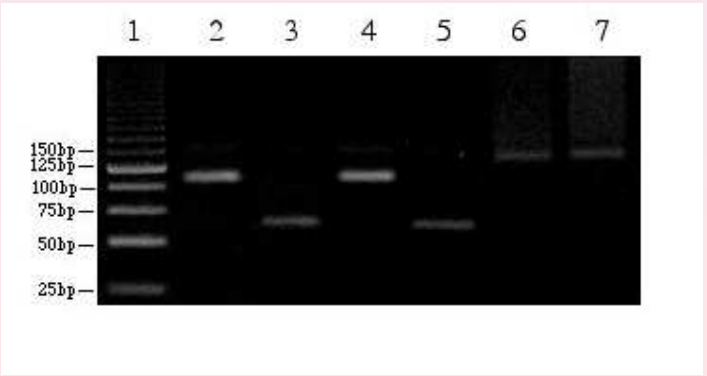
Results $XPCR_{X_i}$ and $XPCR_{Y_i}$ for i=3:



Results $XPCR_{X_i}$ and $XPCR_{Y_i}$ for i=4:



Results $XPCR_{X_i}$ and $XPCR_{Y_i}$ for i=5:



The correctness of XPCR recombination algorithm can be given by the following formulation:

Fact

The n-dimensional library $\{\alpha_1 \cdots \alpha_n \mid \alpha_i \in \{X_i, Y_i\}, i = 1, \ldots, n\}$ *is the null context splicing language generated by the system* $\mathcal{N} = (\Sigma, A, R)$, *where* $\Sigma = \{A, T, C, G\}$, $A = \{I_1, I_2, I_3, I_4\}$, *and* $R = \{r_{X_2}, r_{Y_2}, \ldots, r_{X_{n-1}}, r_{Y_{n-1}}\}$.

Algorithm Correctness Proof (1/2)

Proof. For any recombination $\alpha_1\alpha_2\ldots\alpha_n$ there exists the subset of rules $\{r_{\alpha_2}, r_{\alpha_3}, \ldots, r_{\alpha_{n-1}}\}$ that generates it by means of the following computation starting from the initial sequences.

Let us call l_i the initial sequence containing $\alpha_{i-1}\alpha_i$ as subsequence, for $i = 2, \ldots, n$, and let c, s_1, s_2 be string variables.

By construction, for each value of i there exists only one of such an initial sequence.

Algorithm Correctness Proof (2/2)

$c := l_2$
for $j = 2, \ldots, n-1$
apply $r_{\alpha_j} : \; c,\; l_{j+1} \longrightarrow s_1,\; s_2;$
$c := s_1;$
output: c

The string c contains the substrand $\alpha_1\alpha_2$ before the for cycle, and $\alpha_1\alpha_2 \ldots \alpha_i$ after the cycle corresponding to $j = i - 1$. Since its length remains constant during the computation, after the for cycle the string c is equal to $\alpha_1\alpha_2 \ldots \alpha_n$. □

Large DNA Libraries

- The largest library dates back to 2002 [Braich, Chelyapov, Johnson, Rothemund, Adleman, Science 296], a product of the *mix&split method*: 2^{20} elements.
- A thermodynamically constant library was designed in [Penchovsky, Ackermann, J. of Comput. Biology 10, 2003]: 2^{12} elements.
- *POA*-based [Stemmer, PNAS 91, '94] library in [Gal, Monteith, Macula, Nat Computing, 2008]: 2^5 elements.
- XPCR-based library in [F., Manca, Giagulli, Laudanna, LNCS 3892, 2006]: 2^6 elements.
- Work in progress in Binghamton (NY) to built up a library of 4^7 elements by generalized XPCR.

Generalization

Analogous results hold for the general case of spaces where each of the n variables can assume a number of values $k > 2$, that is $\{1, \ldots, k\}^n$.

If $\alpha_i \in \{Z_i^{(1)}, \ldots, Z_i^{(k)}\}$, the k^2 initial sequences are defined by (n even):

for $h, j = 1, \ldots, k$, $\quad I_{(h-1)k+j} = Z_1^{(h)} Z_2^{(j)} Z_3^{(h)} Z_4^{(j)} \ldots Z_{n-1}^{(h)} Z_n^{(j)}$

and the k-recombination witnesses:

for $j = 1, \ldots, k-1$ (and $n \equiv 0 \pmod 4$)

$$W_j = Z_1^{(j)} Z_2^{(j)} Z_3^{(j+1)} Z_4^{(j+1)} Z_5^{(j)} Z_6^{(j)} \ldots Z_n^{(j+1)}$$

$$W_k = Z_1^{(k)} Z_2^{(k)} Z_3^{(1)} Z_4^{(1)} Z_5^{(k)} Z_6^{(k)} \ldots Z_n^{(1)}$$

Theory and Experimentation

Good performance derives from XPCR, and it requires only 4 initial strings.

As all the biotechniques, it should be equipped with a method to verify that the whole library is really present in the final pool, that is, whether all the splicing rules have worked as expected.

Task. Assuming we have methods to check if a certain sequence is present in the pool, we want to minimize the number of sequences to check in order to verify the success of the experiment (i.e. that the whole library was generated).

Evidence strings: their presence in the final pool guarantees the whole library is there. It can be proved that, for any dimension n of the library, k given sequences are enough, where k is the base of the library.

Evidence strings

It is proved that, for any dimension *n* of the library, two sequences exist such that their simultaneous presence in the pool guarantees that all the recombinations required by the XPCR recombination algorithm occurred.

Let us call i-trio-factor any substring $\alpha_{i-1}\alpha_i\alpha_{i+1}$ where exactly two consecutive variables are X or Y, and consider the corresponding null context splicing rule r_{α_i}.

Lemma

For any $i = 2, \ldots, n-1$ the rule r_{α_i} has been applied in the pool iff in the pool there is a string including a corresponding i-trio-factor.

Actual evidence strings

A couple of strings are recombination witnesses if they include all the possible *i*-trio-factors.

They can be: $\{W_1, W_2\}$ or $\{T_1, T_2\}$, where $(n \equiv 0 \pmod 4)$

$$W_1 = X_1X_2\ Y_3Y_4\ X_5X_6\ Y_7 \ldots Y_n$$
$$W_2 = Y_1Y_2\ X_3X_4\ Y_5Y_6\ X_7 \ldots X_n$$

$$T_1 = X_1\ Y_2Y_3\ X_4X_5\ Y_6Y_7 \ldots X_n$$
$$T_2 = Y_1\ X_2X_3\ Y_4Y_5\ X_6X_7 \ldots Y_n$$

Experiment to find the two sequences

For each $Z_1Z_2Z_3Z_4Z_5Z_6$ of the recombination witnesses W_1 and W_2, its presence was checked in the final pool by means of the following steps:

1. **perform** $PCR(Z_1, \overline{Z_6})$, then **electrophoresis** selecting shortest strands

$$\alpha \underline{Z_1 \ldots Z_6} \beta$$

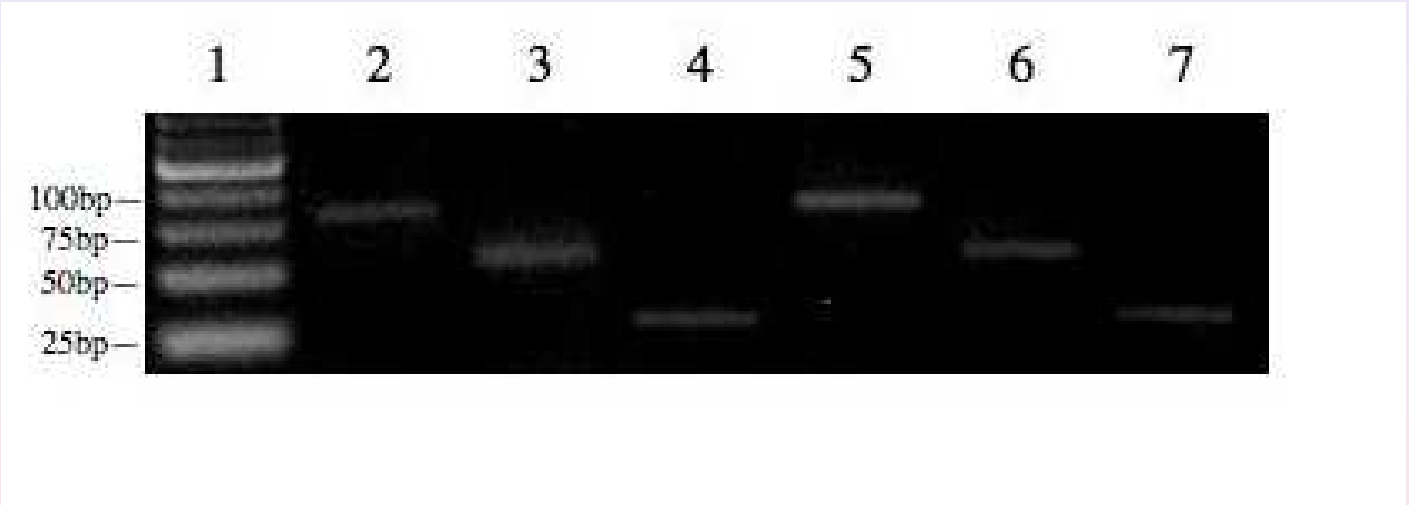
2. **perform** $PCR(Z_2, \overline{Z_5})$, then **electrophoresis** selecting shortest strands

$$Z_1 \underline{Z_2 \ldots Z_5} Z_6$$

3. **perform** $PCR(Z_3, \overline{Z_4})$

$$Z_2 \underline{Z_3 Z_4} Z_5$$

Success



Marker (25bp). 2: PCR $(X_1, \overline{X_6})$ (90bp), 3: $PCR(X_2, \overline{X_5})$ (60bp), 4: $PCR(Y_3, \overline{Y_4})$ (30bp), 5: $PCR(Y_1, \overline{Y_6})$ (90bp), 6: $PCR(Y_2, \overline{Y_5})$ (60pb), 7: $PCR(X_3, \overline{X_4})$ (30pb).

Improved XPCR Recombination: Logarithmic Time

Let variables be encoded by the DNA sequences $X_1, \ldots, X_n$, and their negations by the sequences $Y_1, \ldots, Y_n$ respectively.

Let $Z_1 = \alpha X_1 \gamma X_2 \gamma X_3 \cdots \gamma X_n \beta$ and $Z_2 = \alpha Y_1 \gamma Y_2 \gamma Y_3 \cdots \gamma Y_n \beta$, while r being their length.

input: $P = \{Z_1, Z_2\}$

for $p = 1, \ldots, \lceil \log_2 n \rceil$

$$\begin{cases} P := XPCR_\gamma(P); \\ P := El_r(P) \end{cases}$$

After the k-th cycle *for*, all the possible recombinations of 2^k consecutive literals are present in the pool as subsequences.

Specification of DNA Extraction Problem

Given an input pool P of heterogeneous DNA strands, with the same length and with the same prefix and suffix, and given a string γ:

provide an output pool containing *all and only* the γ-superstrands from P: $P_\gamma = \{\alpha\gamma\beta \mid \alpha\gamma\beta \in P, \alpha, \beta \in \Sigma^*\}$

One additional separation step

Given a string γ, we compute $XPCR_\gamma(\alpha, \overline{\beta})$.

Input pool: $P = \{\alpha x \beta \mid |\alpha x \beta| = n\}$.

1. $(P_1, P_2) := split(P)$;
2. $P_1 := PCR(\alpha, \overline{\gamma})(P_1)$;
3. $P_2 := PCR(\gamma, \overline{\beta})(P_2)$;
4. P:= $Mix(P_1, P_2)$;
5. P: = $\cup_{k<n} El_k(P)$;
6. P: = $PCR(\alpha, \overline{\beta})(P)$;
7. $P := El_n(P)$.

XPCR based $Extract(P, \gamma)$

1. $S := \emptyset$; $L := length(P)$;
2. **for each** $n \in L$ **do**
   1. $R_1 := \emptyset, R_2 := \emptyset, Q := \emptyset, P_1 := \emptyset, P_2 := \emptyset$;
   2. $P := separate(P, n)$;
   3. $P := infix(P, \alpha, \beta)$;
   4. $(P_1, P_2) := split(P)$;
   5. $P_1 := PCR(P_1, \alpha, \bar{\gamma})$;
   6. **for each** $m < n$ **do** $R_1 := mix(R_1, separate(P_1, m))$;
   7. $P_2 := PCR(P_2, \gamma, \bar{\beta})$
   8. **for each** $m < n$ **do** $R_2 := mix(R_2, separate(P_2, m))$;
   9. $Q := mix(R_1, R_2)$;
   10. $Q := PCR(Q, \alpha, \bar{\beta})$;
   11. $Q := separate(Q, n + |\alpha| + |\beta|)$;
   12. $S := mix(S, Q)$;
   13. **output** S [5].

[5] Problem of the γ-chimeras/mermaids.

$XPCR - Mutagenesis(P, \gamma, \delta)$

let $P = \{< \alpha\gamma\beta >\}$, and $Q = \{< \alpha[-18, -1]\ \delta\ \beta[1, 20] >\}$;

1. $(P_1, P_2) := split(P)$;
2. $P_1 := PCR(P_1, \alpha[1, 18], mir(\alpha[-18, -1]))$;
3. $P_2 := PCR(P_2, \beta[1, 20], mir(\beta[-20, -1]))$;
4. $P_1 := separate(P_1, |\alpha|)$; $P_2 := separate(P_2, |\beta|)$;
5. $P_1 := mix(P_1, Q)$;
6. $P_1 := PCR(P_1, \alpha[1, 18], mir(\beta[1, 20]))$;
7. $P_1 := separate(P_1, |\alpha| + |\delta| + 20)$;
8. $P := mix(P_1, P_2)$;
9. $P := PCR(P, \alpha[1, 18], mir(\beta[-20, -1]))$;
10. $P := separate(P, |\alpha| + |\beta| + |\delta|)$;

Output: P.