Politechnika Śląska w Gliwicach

Wydział Automatyki, Elektroniki i Informatyki

Instytut Informatyki

Rozprawa doktorska

Rozszerzenie biomolekularnego automatu Shapiro

Sebastian Sakowski

Promotor
Dr hab. Tadeusz Krasiński
Uniwersytet Łódzki
Wydział Matematyki i Informatyki

Gliwice 2010

Spis treści

Wstęp

W ostatnich latach informatykę zaczęto określać, jako dyscyplinę nauki o przetwarzaniu, przechowywaniu i przekazywaniu informacji w dowolnym środowisku, a więc zarówno opartym na elektronice, jak i na mechanice kwantowej, czy genetyce molekularnej [45], [46], [47], [48]. W chwili obecnej rozwiązania oparte na kwantach i łańcuchach DNA są jednak we wczesnym etapie rozwoju. W niedalekiej przyszłości spodziewać się jednak należy znacznego rozwoju tych technologii.

Problemy techniczne i ograniczenia fizyczne miniaturyzacji układów scalonych sprawiły, że trudno jest zwiększyć wydajność obliczeń wykonywanych przez komputery oparte na przepływie elektronów. Obecnie prowadzone badania naukowe nad nowymi technologiami mogącymi zastąpić konwencjonalne komputery, koncentrują się głównie na dwóch nowych możliwościach: obliczeniach kwantowych oraz DNA obliczeniach. Bardzo obiecujące wydaje się zastosowanie DNA, czyli związku chemicznego kodującego informacje w organizmach żywych, gdyż DNA ze względu na swoje właściwości ma duży potencjał w kodowaniu, przetwarzaniu i magazynowaniu informacji. W pracach naukowych znaleźć można wiele przykładów implementacji znanych problemów informatycznych za pomocą odpowiedniego kodowania łańcuchów DNA oraz stosowania metod genetyki molekularnej. Są to jednak pomysły za mało zaawansowane technologicznie, aby możliwe było zbudowanie na ich podstawie ogólnodostępnych i uniwersalnych rozwiązań. Dzięki bardzo małym rozmiarom, możliwościom upakowania informacji oraz zgodności z organizmami żywymi DNA ma bardzo duży potencjał, jako materiał do budowy urządzeń mogących przetwarzać informację w bardzo szczególnych warunkach np.: biochipów analizujących choroby, a także innych nowoczesnych nanomaszyn.

Dynamiczny rozwój badań nad możliwościami wykonywania DNA obliczeń rozpoczął się w 1994 roku od eksperymentu Leonarda Adlemana [2], jednego z współautorów szyfrowania RSA. Zastosował on łańcuchy DNA do znanego problemu informatycznego (problemu drogi Hamiltona w grafie). Od tego czasu dziedzina ta znacznie rozwinęła się. Pojawiły się różne teoretyczne rozważania oraz praktyczne implementacje obliczeń za pomocą DNA. W 2001 roku grupa naukowców z Instytutu Weizmanna (Shapiro i inni) [4] opracowała programowalny automat skończony. Wszystkie elementy tej prostej 2-stanowej 2-symbolowej niedeterministycznej maszyny zbudowane są z łańcuchów DNA oraz jednego enzymu restrykcyjnego. Naprzemienne cięcie i łączenie łańcuchów DNA

reprezentujących poszczególne elementy automatu doprowadza do sekwencji terminalnej, której obecność w roztworze oznacza akceptację słowa wejściowego.

Automat Shapiro z teoretycznego punktu widzenia reprezentuje prosty model. Dążąc do uniwersalnych DNA obliczeń, konieczne jest w pierwszej kolejności zbadanie możliwości rozszerzania tego automatu. Autorzy zwracają jednak uwagę, że odkrycie enzymów restrykcyjnych pozostawiających dłuższe lepkie końce umożliwi rozszerzenie ich automatu. Zespół Shapiro i inni [4] podaje, że możliwe jest zakodowanie za pomocą lepkich końców, powstających po cięciu enzymem *Fok*I, automatu co najwyżej 3-stanowego. Ograniczenie wynika z długości lepkich końców. Rozszerzenie do trzech stanów zostało wykonane przez dwa zespoły badawcze [35], [40].

W pracy przedstawiona została nowa idea rozszerzania automatu Shapiro polegająca na zwiększeniu liczby enzymów restrykcyjnych działających autonomicznie w jednej mieszaninie. Podane zostało kodowanie łańcuchów DNA dla 6-stanowego 2-symbolowego automatu z wykorzystaniem dwóch enzymów restrykcyjnych. Określone zostały również warunki arytmetyczne konieczne do dalszego rozszerzania automatu Shapiro. W pracy zaproponowano również tworzenie „bibliotek łańcuchów DNA", które umożliwiają wielokrotne użycie raz przygotowanych łańcuchów DNA. Eksperyment laboratoryjny potwierdził doświadczalnie możliwość rozszerzania automatu Shapiro do większej liczby stanów z zastosowaniem dwóch enzymów restrykcyjnych.

Rozdział pierwszy omawia teoretyczne modele komputerów i związane z nimi klasy języków formalnych. W pracy przybliżone są również podstawowe informacje o DNA (rozdział 2) oraz krótko omówione wybrane praktyczne implementacje problemów informatycznych za pomocą DNA. Ze względu na główną tematykę pracy w rozdziale trzecim zostanie omówiony dokładnie 2-stanowy 2-symbolowy automat Shapiro. W rozdziale czwartym przedstawiono nową ideę konstruowania automatów opartych na wielu enzymach restrykcyjnych (dokładnie dwóch). Przedstawiono również rozważania teoretyczne dotyczące możliwości rozszerzeń do dowolnej liczby stanów i dowolnej liczby symboli. Rozdział 5 prezentuje laboratoryjny eksperyment implementujący automat opisany w rozdziale 4 z zastosowaniem dwóch enzymów restrykcyjnych. Podano w nim również metody laboratoryjne (inne niż użyte przez zespół Shapiro i inni) umożliwiające konstruowanie łańcuchów DNA reprezentujących poszczególne elementy automatu. Warto podkreślić, że nikt wcześniej nie zademonstrował praktycznego wykorzystania dwóch

enzymów restrykcyjnych jednocześnie do przeprowadzania obliczeń na łańcuchach DNA (teoretyczna próba zastosowania wielu enzymów jest w pracy [28]).

Praktyczna implementacja laboratoryjna wykonana została w ramach projektu dofinansowanego przez Rektora Uniwersytetu Łódzkiego, Dziekana Wydziału Matematyki i Informatyki Uniwersytetu Łódzkiego, Zakład Analizy Rzeczywistej i Algebry UŁ oraz Katedrę Genetyki Molekularnej UŁ.

Pragnę podziękować Panu Prof. dr hab. Januszowi Błasiakowi za udostępnienie laboratorium Genetyki Molekularnej, a także Panu Dr Tomaszowi Popławskiemu za pomoc merytoryczną w wykonaniu eksperymentów laboratoryjnych.

Ponadto chciałbym podziękować Panu Prof. dr hab. inż. Tadeuszowi Czachórskiemu za życzliwość i wsparcie w trakcie odbywania studiów doktoranckich i przygotowywania rozprawy doktorskiej.

Cel i teza pracy

Możliwe jest teoretyczne i praktyczne (eksperymentalne) rozszerzenie 2-stanowego automatu Shapiro do większej ilości stanów przez zastosowanie wielu enzymów restrykcyjnych działających autonomicznie w jednej mieszaninie.

Cele pracy:

1. Konstrukcja 6-stanowego 2-symbolowego automatu skończonego za pomocą łańcuchów DNA z zastosowaniem dwóch enzymów restrykcyjnych działających w jednej mieszaninie.
2. Podanie warunków arytmetycznych możliwości rozszerzenia automatu Shapiro do p stanów i r symboli, z uwzględnieniem długości kodów symboli n i długości lepkich końców $k_1,\ldots,k_j$ pozostałych po działaniu j enzymów restrykcyjnych.
3. Wykonanie eksperymentu laboratoryjnego polegającego na praktycznym sprawdzeniu idei zwiększania liczby stanów przez zastosowanie dwóch enzymów restrykcyjnych działających autonomicznie w jednej mieszaninie na słowie wejściowym.

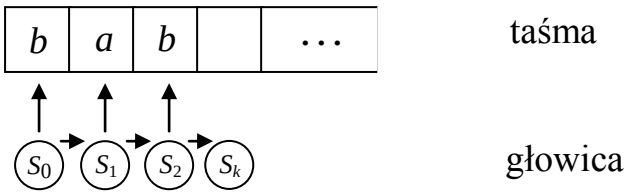
Rozdział 1

Teoretyczne modele komputerów

Pierwsze modele obliczeń powstały na początku dwudziestego wieku w głowach matematyków, a dokładniej logików matematycznych zajmującymi się podstawami matematyki. Ich celem nie były komputery, których jeszcze nie wymyślono, ale problemy takie jak: precyzyjne pojęcie algorytmu, pojęcie funkcji obliczalnej, automatyczne dowodzenie twierdzeń. Te teoretyczne prace stały się podstawą budowy komputerów, opartych na przepływie elektronów. W rozdziale tym omówimy standardowe teoretyczne modele urządzeń obliczających [14], [17]. Ponieważ główne wyniki pracy dotyczą automatów skończonych, więc dokładniej omówimy ten model. Aby dokładniej umiejscowić problemy rozwiązywane przez zaproponowany w rozprawie model, krótko opiszemy również podstawową, uznawaną powszechnie klasyfikację języków formalnych, czyli problemów (hierarchia Chomsky'ego).

1.1. Automaty skończone

Najprostszym modelem obliczeń jest automat skończony, mogący przyjmować skończoną liczbę stanów $s_0,...,s_m$. Ma on niewielką moc obliczeniową i stosowany może być do rozwiązywania prostych problemów. Automat skończony *M* możemy wyobrazić sobie, jako następujące urządzenie.



Rys. 1. Schemat ideowy automatu skończonego.

Na taśmie (potencjalnie nieskończonej z prawej strony) podzielonej na komórki zapisywane są słowa utworzone z alfabetu automatu *M*. Głowica czyta kolejne symbole na taśmie i w zależności od swego stanu i wczytanego symbolu zmienia stan. Jeżeli głowica po wczytaniu wszystkich symboli danego słowa znajdzie się w jednym ze stanów końcowych, to słowo jest akceptowane przez automat *M*.

Deterministyczne automaty skończone

Automaty skończone jednoznacznie (deterministycznie) określające stan po wczytaniu kolejnego symbolu słowa wejściowego nazywamy deterministycznymi automatami skończonymi. Formalnie definiujemy je w następujący sposób.

Definicja 1

Deterministycznym automatem skończonym (w skrócie **DAS**) nazywamy uporządkowaną piątkę

$$M = (Q, \Sigma, s_0, F, \delta)$$

gdzie:

$Q = \{s_0, ..., s_m\}$ - zbiór skończony, którego elementy nazywany stanami *M*,

$\Sigma = \{a_1, ..., a_n\}$ - zbiór skończony, zwany alfabetem *M*,

$s_0 \in Q$ - wyróżniony stan, zwany stanem początkowym,

$F \subset Q$ - wyróżniony podzbiór stanów, zwany stanami końcowymi,

$\delta : Q \times \Sigma \to Q$ - funkcja, zwana funkcją przejść *M*, która każdemu stanowi Q i symbolowi alfabetu Σ przypisuje nowy stan automatu *M*.

Słowo A utworzone z symboli Σ jest akceptowane przez automat $M = (Q, \Sigma, s_0, F, \delta)$, gdy

$$\hat{\delta}(s_0, A) \in F .$$

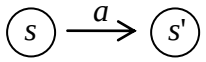
gdzie przez $\hat{\delta}(s_0, A)$ oznaczyliśmy stan automatu po wczytaniu słowa A zaczynając od stanu s_0.

Oznacza to, że słowo jest akceptowane, gdy automat *M* znajdzie się w jednym ze stanów końcowych po wczytaniu całego słowa *A*, rozpoczynając działanie od stanu początkowego s_0.

Zbiór wszystkich słów utworzonych z alfabetu Σ i akceptowanych przez automat *M* oznaczamy przez *L*(*M*), tzn.

$L(M) = \{A \in \Sigma^* :$ słowo *A* jest akceptowane przez automat *M*$\} = \{A \in \Sigma^* : \hat{\delta}(s_0, A) \in F\}$.

Każdy automat skończony $M = (Q, \Sigma, s_0, F, \delta)$ możemy przedstawić za pomocą skierowanego grafu *G*(*M*). Wierzchołkami grafu są stany automatu *M*, natomiast każde pojedyncze przejście $\delta(s, a) = s'$ jest skierowaną krawędzią grafu *G*(*M*).

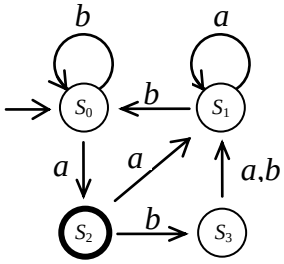


Stan początkowy oznacza się dodatkową strzałką $\rightarrow (s_0)$, a stany końcowe pogrubionym okręgiem (s).

Przedstawienie automatu skończonego za pomocą grafu jest podejściem równoważnym, gdyż wszystkie informacje z definicji 1 są zawarte w grafie tzn.: ilość stanów, stan początkowy, stan końcowy, funkcja przejść δ.

Przykład 1

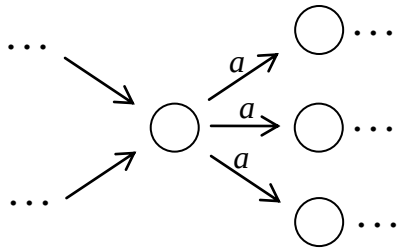
Przykład 4-stanowego 2-symbolowego deterministycznego automatu skończonego *M* został przedstawiony na poniższym rysunku.



Rys. 2. Przykład deterministycznego automatu.

Niedeterministyczne automaty skończone

Pewnym uogólnieniem deterministycznych automatów skończonych są niedeterministyczne automaty, które różnią się od poprzednich tym, że po wczytaniu kolejnego symbolu słowa mogą przejść w jeden z kilku możliwych stanów. W grafie automatu niedeterministycznego mogą występować różne drogi dla tego samego słowa.



Rys. 3. Niedeterminizm automatu skończonego.

Formalnie niedeterministyczne automaty skończone definiujemy w następujący sposób.

Definicja 2

Niedeterministycznym automatem skończonym (w skrócie **NAS**) nazywamy uporządkowaną piątkę

$$M = (Q, \Sigma, s_0, F, \delta)$$

gdzie:

$Q = \{s_0,...,s_m\}$ - zbiór skończony, którego elementy nazywany stanami *M*,

$\Sigma = \{a_1,...,a_n\}$ - zbiór skończony, zwany alfabetem *M*,

$s_0 \in Q$ - wyróżniony stan, zwany stanem początkowym,

$F \subset Q$ - wyróżniony podzbiór stanów, zwany stanami końcowymi,

$\delta : Q \times \Sigma \to \beta(Q)$ - funkcja, zwana funkcją przejść *M*, która danemu stanowi Q i symbolowi z alfabetu Σ przypisuje pewien zbiór stanów automatu *M*.

Słowo $A \in \Sigma^*$ jest akceptowane przez niedeterministyczny automat $M = (Q, \Sigma, s_0, F, \delta)$, gdy

$$\hat{\delta}(s_0, A) \cap F \neq \emptyset$$

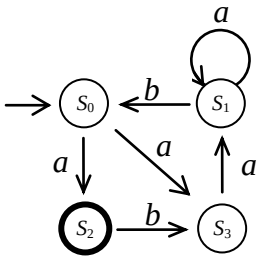
gdzie przez $\hat{\delta}(s_0, A)$ oznaczyliśmy zbiór wszystkich stanów automatu otrzymanych po wczytaniu słowa *A* zaczynając od stanu s_0. Zbiór wszystkich słów utworzonych z alfabetu Σ i akceptowanych przez automat *M* oznaczamy przez *L*(*M*), tzn.

$L(M) = \{A \in \Sigma^* :$ słowo *A* jest akceptowane przez automat *M*$\} = \{A \in \Sigma^* : \hat{\delta}(s_0, A) \cap F \neq \emptyset\}$.

Podobnie jak w przypadku deterministycznych automatów skończonych każdy niedeterministyczny automat skończony możemy przedstawić w postaci grafu.

Przykład 2.

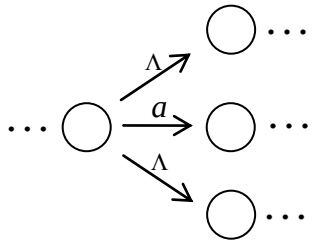
Poniższy graf reprezentuje 4-stanowy 2-symbolowy niedeterministyczny automat skończony.



Rys. 4. Przykład niedeterministycznego automatu.

Niedeterministyczne automaty skończone z Λ - przejściami

Automaty te są modyfikacją automatów niedeterministycznych, a ich odmienność polega na możliwości wykonywania pewnych przejść samorzutnie tzn. bez wczytania następnego symbolu słowa wejściowego. W grafie tego automatu wyróżniamy krawędzie oznaczone symbolem słowa pustego Λ reprezentujące tzw. Λ - przejścia .



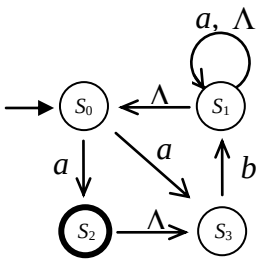
Rys. 5. Λ- przejścia w automacie niedeterministycznym.

Definicja niedeterministycznych automatów z Λ przejściami (w skrócie **NAS-Λ**) jest taka sama jak niedeterministycznego automatu skończonego z wyjątkiem funkcji przejść $\delta : Q \times (\Sigma \cup \{\Lambda\}) \to \beta(Q)$, która danemu stanowi Q i symbolowi Σ lub słowu pustemu Λ przypisuje zbiór stanów automatu *M*.

Podobnie jak przedstawione automaty skończone, każdy niedeterministyczny automat z przejściami Λ można przedstawić w postaci grafu skierowanego.

Przykład 3.

Niech $\Sigma = \{a, b\}$, wówczas poniższy graf reprezentuje niedeterministyczny automat z Λ przejściami.



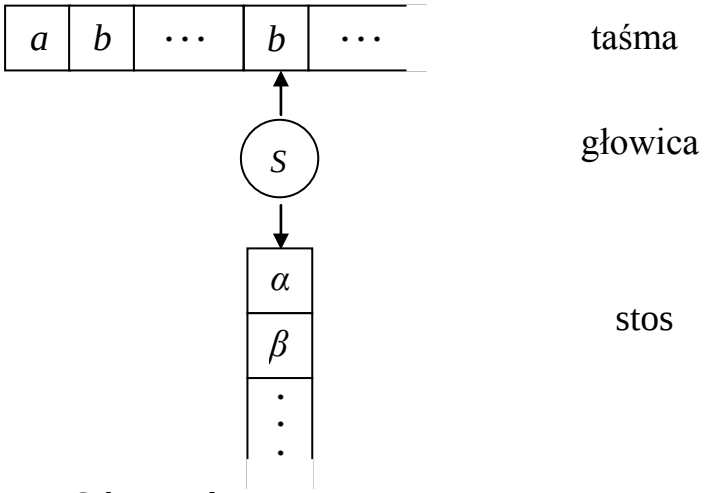
Rys. 6. Przykład grafu niedeterministycznego automatu z Λ przejściami.

Wszystkie powyżej omówione automaty akceptują tę samą klasę języków tzw. klasę języków regularnych (na mocy twierdzenia Kleene`go). Inne podejście do języków regularnych omówimy przy klasyfikacji Chomsky'ego).

Krótko omówimy teraz inne standardowe modele obliczeń (komputerów), choć nie będą one używane w pracy.

1.2. Automaty ze stosem

Automaty te posiadają dodatkowy element, nie występujący w automatach skończonych, a mianowicie prostą pamięć (stos) do której automat może dodawać i pobierać informacje. Intuicyjnie automat ze stosem możemy przedstawić w następujący sposób (Rys.7).



Rys. 7. Schemat ideowy automatu ze stosem.

W automacie tym z samej definicji dopuszczalny jest niedeterminizm oraz Λ przejścia. Formalna definicja automatu ze stosem jest następująca.

Definicja 3

Automatem ze stosem (w skrócie **AZS**) nazywamy układ

$$M = (Q, \Sigma, \Gamma, \delta, s_0, \perp, F)$$

gdzie:

Q - zbiór skończony, którego elementy nazywamy stanami *M*,

Σ - zbiór skończony, zwany alfabetem *M*,

Γ - zbiór skończony, zwany alfabetem stosu *M*,

$\delta : Q \times (\Sigma \cup \{\Lambda\}) \times \Gamma \to \beta(Q \times \Gamma^*)$ - funkcja przejść, która każdej trójce (s, a, γ), gdzie $s \in Q$, $a \in \Sigma \cup \{\Lambda\}$ i $\gamma \in \Gamma$, przyporządkowuje pewien skończony podzbiór par z $Q \times \Gamma^*$,

$s_0 \in Q$ - wyróżniony stan, zwany stanem początkowym *M*,

$\perp \in \Gamma$ - wyróżniony symbol, zwany symbolem początkowym na stosie,

$F \subset Q$ - wyróżniony podzbiór stanów, zwany stanami końcowymi.

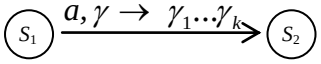
Jeżeli $\delta(s,a,\gamma) \to \{(s_1,A_1),...,(s_k,A_k)\}$, to oznaczamy również

$$(s,a,\gamma) \to (s_1,A_1)$$
$$\vdots$$
$$(s,a,\gamma) \to (s_k,A_k)$$

Każdy pojedynczy ruch $(s,a,\gamma) \to (s_i,A_i)$ oznacza, że jeżeli głowica znajdzie się w stanie *s*, wczytuje symbol *a* na taśmę i na wierzchu stosu jest symbol γ, to może zmienić stan na s_i, a symbol γ jest zamieniony przez słowo A_i.

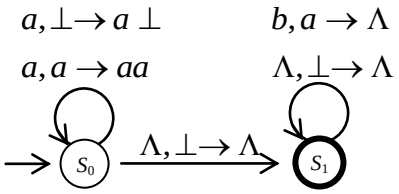
Słowo $A \in \Sigma^*$ jest akceptowane przez automat ze stosem *M*, gdy istnieje taki układ ruchów, że po wczytaniu całego słowa automat znajdzie się w jednym ze stanów końcowych (lub w myśl równoważnej definicji stos jest pusty).

Podobnie jak poprzednio omawiane automaty także automat ze stosem można przedstawić w postaci skierowanego grafu. Stany w grafie AZS są wierzchołkami, natomiast krawędzie reprezentują przejścia $(s_1,a,\gamma) \to (s_2,\gamma_1...\gamma_k)$.



Przykład 4

Graf automatu ze stosem został przedstawiony na poniższym rysunku.

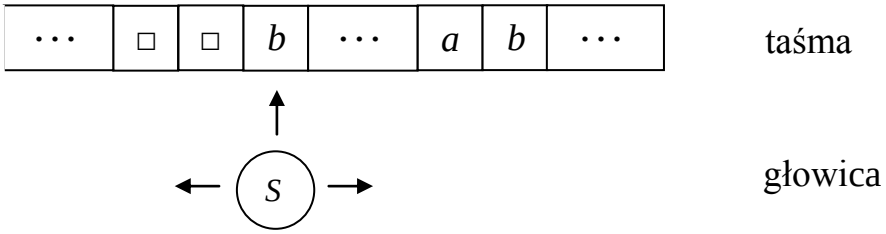


Rys. 8. Graf automatu ze stosem.

Automaty ze stosem akceptują szerszą klasę języków niż automaty skończone, tzw. klasę języków bezkontekstowych. Omówimy je przy klasyfikacji Chomsky'ego.

1.3. Maszyny Turinga

Maszyna Turinga jest automatem mogącym rozwiązywać problemy o największej złożoności i uważana jest za właściwy model komputera. Mogą one obliczać wartości każdej funkcji posiadającą „procedurę" obliczania. Zbudowana jest z teoretycznie nieskończenie (w obie strony) długiej taśmy, która podzielona jest na komórki (Rys. 9). Komórka może zawierać symbol ze skończonego zbioru symboli. W skład maszyny Turinga wchodzi również głowica, przyjmująca skończoną ilość stanów i która może czytać, zapisywać i usuwać symbole. Głowica może przesuwać się w prawo lub w lewo taśmy.



Rys. 9. Schemat ideowy maszyny Turinga.

Definicja 4

Maszyną Turinga (w skrócie **MT**) nazywamy układ:

$$M=(Q, \Sigma, \Gamma, \delta, s_0, \square, F)$$

gdzie:

Q - zbiór skończony, którego elementy nazywamy stanami M,

Σ - zbiór skończony zwany alfabetem M,

Γ - zbiór skończony zwany alfabetem taśmy M, taki że $\Sigma \subset \Gamma$,

δ: $D \rightarrow Q \times \Gamma \times \{L,R\}$, gdzie D jest pewnym podzbiorem $Q \times \Gamma$ (δ nazywamy funkcją przejścia lub funkcją ruchów M),

$s_0 \in Q$ - wyróżniony stan, zwany stanem początkowym M,

$\square \in \Gamma$ - wyróżniony symbol, zwany symbolem pustej komórki,

$F \in Q$ - wyróżniony podzbiór stanów zwanych stanami końcowymi M.

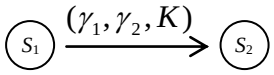
Słowo $A \in \Sigma^*$ jest akceptowane przez M, gdy istnieje skończona ilość ruchów M prowadząca od stanu początkowego głowicy s_0, czytającej pierwszy symbol słowa A, do stanu głowicy, który jest pewnym stanem końcowym $s' \in F$.

Możemy zatem zdefiniować język akceptowany przez *M*:

$$L(M):=\{A \in \Sigma^*: M \text{ akceptuje słowo } A\}.$$

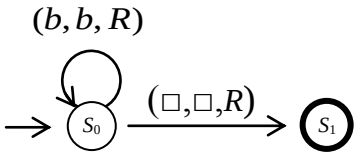
Język formalny *L* nazywamy rekursywnie przeliczalnym, gdy istnieje maszyna Turinga *M* taka, że *L=L(M).*

Każdą maszynę Turinga można reprezentować skierowanym grafem, co ułatwia konstruowanie konkretnych maszyn Turinga. Wierzchołkami tego grafu są stany, natomiast krawędzie łączące stany s_1 i s_2 oznaczamy trójką (γ_1, γ_2, K), o ile $\delta(s_1, \gamma_1) = (s_2, \gamma_2, K)$ (gdzie $s_1, s_2 \in Q, \gamma_1, \gamma_2 \in \Gamma$ i $K \in \{L, R\}$).



Przykład 5

Przykładowy graf maszyny Turinga został przedstawiony na poniższym rysunku.



Rys. 10. Graf Maszyny Turinga.

1.4. Klasyfikacja Chomsky'ego języków i gramatyk

W 1959 Noam Chomsky podał klasyfikację języków ze względu na gramatyki generujące te języki. Formalnie gramatykę *G* możemy zdefiniować jako czwórkę *G*=(Σ, *V*, *S*, *P*), gdzie:

V - skończony zbiór symboli niekońcowych,

Σ - alfabet gramatyki, czyli symbole końcowe,

S - symbol początkowy $S \in V$,

P - skończony zbiór produkcji $P \subset (V \cup \Sigma)^+ \times (V \cup \Sigma)^*$.

Produkcję $(u, v) \in P, u \in (V \cup \Sigma)^+, v \in (V \cup \Sigma)^*$, zapisujemy zwykle jako $u \to_G v$. Gramatyka *G* generuje język $L(G) \subset \Sigma^*$ w następujący sposób: wychodząc od zmiennej początkowej *S* stosujemy produkcje *G* w skończonej ilości w dowolnej kolejności, aż otrzymamy słowo $A \in \Sigma^*$. Zapisujemy to

$$S \Rightarrow_G A.$$

Wówczas

$$L(G) = \{A \in \Sigma^* : S \Rightarrow_G A\}.$$

Narzucając różne warunki na postać produkcji otrzymujemy różne klasy języków. Zostało to wykorzystane przez N. Chomsky'ego do pewnej klasyfikacji języków, zwanej hierarchią Chomsky'ego.

Hierarchia ta dzieli gramatyki na 4 kategorie: gramatyki typu 0, 1, 2, 3. Najogólniejsze to gramatyki typu 0, a najbardziej ograniczone typu 3. Poniższa tabela przedstawia klasyfikację Chomsky'ego języków, gramatyk i odpowiadających im automatów.

Gramatyka	Inna nazwa	Języki	Automaty
typu 0	GF	JRP	MT
typu 1	GK	JK	ALO
typu 2	GBK	JBK	AZS
typu 3	GR	JR	DAS, NAS, NAS-Λ

Rys. 11. Klasyfikacja Chomsky'ego.

Omówimy te gramatyki zaczynając od najprostszych:

a) **Gramatyki typu 3** - są to gramatyki regularne.

Gramatyki te generują języki regularne. Zbiór wszystkich języków regularnych oznaczamy przez JR. Jest to klasa języków akceptowanych przez Deterministyczne Automaty Skończone (DAS), Niedeterministyczne Automaty Skończone (NAS) oraz Niedeterministyczne Automaty Skończone z Λ przejściami (NAS-Λ).

Produkcje gramatyk regularnych są typu: $X \to aY,\ Z \to b,\ X,Y,Z \in V,\ a \in \Sigma,\ b \in \Sigma \cup \{\Lambda\}$.

Przykłady produkcji: $X \to aY,\ Y \to bY,\quad X,Y \in V,\ a,b \in \Sigma$.

b) **Gramatyki typu 2** - są to gramatyki bezkontekstowe.

Gramatyki te generują języki bezkontekstowe. Zbiór wszystkich języków bezkontekstowych oznaczamy przez JBK. Klasa języków bezkontekstowych jest akceptowana przez Automaty ze Stosem (AZS).

Produkcje gramatyk bezkontekstowych są typu: $X \to \alpha,\ X \in V, \alpha \in (V \cup \Sigma)^*$.

Przykłady produkcji: $X \to aXa \mid bXb \mid a \mid b \mid \Lambda,\quad X \in V,\ a,b \in \Sigma$.

c) **Gramatyki typu 1** - są to gramatyki kontekstowe.

Gramatyki te generują języki kontekstowe (JK), które są akceptowane przez Automaty Liniowo Ograniczone.

Produkcje gramatyk kontekstowych są typu: $\alpha \to \beta,\ \alpha, \beta \in (V \cup \Sigma)^* \setminus \{\Lambda\},\ i\ |\alpha| \le |\beta|$.

Przykłady produkcji: $XaY \rightarrow aYZbSa, \quad X \rightarrow YZ, \quad X,Y,Z,S \in V, \; a,b \in \Sigma.$

d) **Gramatyki typu 0** - są to gramatyki frazowe.

Gramatyki te generują języki rekursywnie przeliczalne. Zbiór wszystkich języki rekursywnie przeliczalne oznaczamy przez JRP. Jest to klasa języków akceptowanych przez Maszyny Turinga (MT).

Produkcje są typu: $\alpha \rightarrow \beta, \quad \alpha \in (V \cup \Sigma)^* \setminus \{\Lambda\}, \quad \beta \in (V \cup \Sigma)^*.$

Przykłady produkcji*:* $\mathrm{X}a\mathrm{Y} \rightarrow aYZbSa, \; XY \rightarrow Z \; XbYa \rightarrow \Lambda$, $\mathrm{X,Y,Z,S} \in V, \; a,b \in \Sigma$.

Rozdział 2

Wprowadzenie do DNA obliczeń

Poznanie budowy DNA i jego własności spowodowało, że organizmy żywe zaczęto postrzegać jako bardzo złożone systemy informatyczne (biologiczne systemy informatyki). Szczególne własności DNA sprawiają, że możliwe jest zakodowanie w nich informacji o procesach zachodzących w organizmach żywych. Rozwój metod genetyki molekularnej umożliwił wykonanie praktycznych doświadczeń polegających na implementacji laboratoryjnej problemów informatycznych kodowanych odpowiednio zaprojektowanymi fragmentami DNA. W rozdziale tym przybliżona zostanie struktura DNA (dokładne informacje w [38]) oraz przedstawione zostaną wybrane implementacje DNA obliczeń.

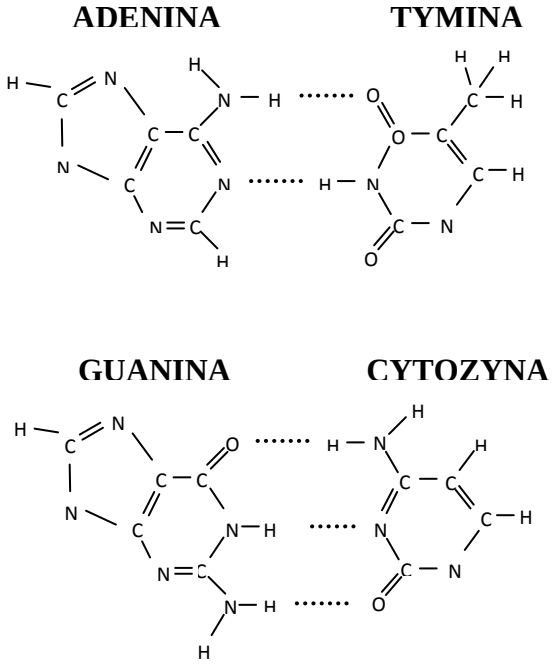
2.1. Budowa i podstawowe operacje na DNA

Struktura DNA (zwana modelem helisy B-DNA) została odkryta przez Jamesa Watsona i Francisa Cricka w 1953 roku na podstawie zdjęć rentgenowskich DNA wykonanych przez Rosalind Franklin i Maurice'a Wilkinsa. Zaproponowany model budowy DNA został potwierdzony wieloma późniejszymi badaniami strukturalnymi [38].

DNA jest polimerem, czyli związkiem chemicznym składającym się z wielokrotnie powtórzonych jednostek zwanych deoksyrybonukleotydami, nazywanych krótko nukleotydami. Każdy nukleotyd składa się z zasady azotowej, cukru deoksyrybozy oraz grupy fosforanowej. Nośnikiem informacji w DNA są jednak tylko zasady azotowe (pochodne puryny i pirymidyny). Zasadami purynowymi są adenina (A) i guanina (G), a pirymidynowymi tymina (T) i cytozyna (C). Pozostałe elementy budowy DNA tworzą szkielet cukrowo-fosforanowy niekodujący informacji. Zasady azotowe łączą się ze sobą komplementarnie tzn.: adenina łączy się zawsze z tyminą (dwa wiązania wodorowe), a guanina z cytozyną (trzy wiązania wodorowe) - rysunek 12.

W modelu Watsona i Cricka dwa łańcuchy nukleotydów połączone komplementarnie oplatają się helikalnie. Charakterystyczna tzw. polarność DNA związana jest z ukierunkowaniem łańcuchów DNA. Jeden z końców DNA ma grupę 5'-OH, natomiast drugi koniec grupę 3'-OH. Grupy te nie są połączone z innymi nukleotydami. Przyjmuje się, że niezwiązana grupa 5'-OH ulokowana jest po lewej stronie w uproszczonym zapisie DNA i oznacza się jako 5'. Natomiast grupa 3'-OH znajduje się po prawej stronie i oznaczamy ją jako 3'. Kolejność zasad kodujących informację genetyczną zapisuje się w kierunku 5'→3'.

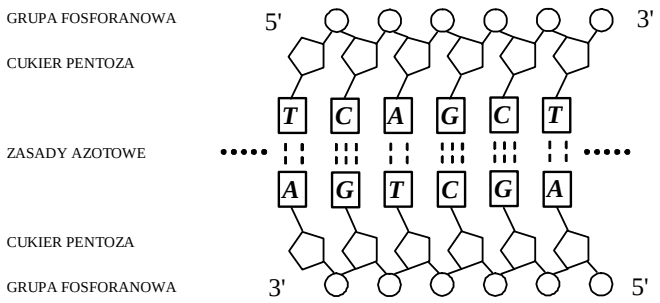
Formalizowanie zapisu kolejności zasad azotowych jest bardzo ważne przy badaniu DNA, gdyż przykładowo sekwencje ACG oraz GCA kodują inne aminokwasy.



Rys. 12. Komplementarność zasad azotowych.

Ze względu na kodowanie informacji o budowie białek i RNA (te związki określają wszystkie cechy organizmów żywych), DNA zajmuje wyjątkową pozycję wśród związków chemicznych występujących w organizmach żywych. Model budowy DNA (helisy B-DNA), opracowany przez Watsona i Cricka, charakteryzuje się następująco (Rys. 13):

1. zbudowany jest z dwóch równoległych i przeciwbieżnych łańcuchów nukleotydów jeden ma kierunek 5'→3', a drugi 3'→5',
2. łańcuchy te są helikalnie zwinięte wokół wspólnej osi,
3. podwójny łańcuch DNA zawiera około 10 par nukleotydów na jeden skręt,
4. brak jest ograniczeń co do kolejności sekwencji w łańcuchu DNA,
5. pary zasad adenina-tymina (A-T) łączą się ze sobą dwoma wiązaniami, natomiast guanina-cytozyna trzema wiązaniami wodorowymi.



Rys. 13. Ogólna budowa DNA.

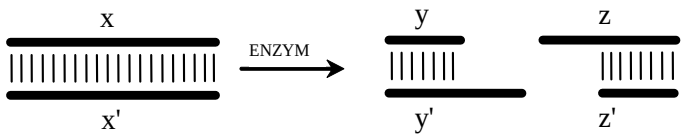
Warto zwrócić uwagę, że po opracowaniu modelu budowy DNA przez Watsona-Cricka organizmy żywe zaczęto postrzegać jako bardzo złożone procesy przetwarzające informacje [38]. Systemy informatyczne, dzięki którym organizmy żywe mogą istnieć zaczęto nazywać biologicznymi systemami informatyki [45], [46], [47], [48]. W systemach tych DNA nie jest bezpośrednią matrycą do syntezy białek, funkcję tę pełni RNA. Wyróżniamy informacyjny RNA (mRNA), transportujący RNA (tRNA) oraz rybosomowy RNA (rRNA). Informacja zapisana w DNA jest poddawana procesowi transkrypcji (przepisywania) z DNA na informacyjny RNA. Pozostałe dwa rodzaje RNA (tRNA oraz rRNA) stanowią część mechanizmu syntezy białek. Kodowanie informacji w RNA odbywa się, podobnie jak w DNA, za pomocą zasad azotowych z tą różnicą, że zamiast tyminy (T) w RNA występuje uracyl (U).

Podstawową jednostką określającą informację genetyczną w organizmach żywych jest gen, czyli określony odcinek DNA. Geny nie kodują informacji w sposób ciągły, wyróżniamy odcinki kodujące informacje („eksony”) oraz odcinki niekodujące („introny”). Zespół genów nazywamy genomem, który wraz z procesami przetwarzającymi informację w nim zawartą stanowi bardzo złożony system informatyczny. W biologicznych systemach informatyki DNA pełni rolę zewnętrznej pamięci zawierającej informacje o sposobie porządkowania swobodnych atomów i molekuł [46], [48]. Biorąc pod uwagę, że biologiczne systemy informatyki rozwijały się miliardy lat, są one prawdopodobnie najbardziej rozwiniętymi systemami informatyki znanymi człowiekowi. W biologicznych systemach informatyki podstawowa informacja reprezentowana jest w postaci bitów molekularnych, które są reprezentowane czterema zasadami azotowymi (adeniną, guaniną, tyminą, cytozyną w DNA oraz dodatkowo uracylem w RNA).

W latach siedemdziesiątych ubiegłego stulecia nastąpił rozwój metod sztucznej rekombinacji DNA, czyli wymiany fragmentów DNA na inne (w szczególności syntezę dowolnego łańcucha DNA). Umożliwiło to sterowanie informacją zapisaną w DNA i powstanie nowej dziedziny zwanej inżynierią genetyczną. Możliwe to było dzięki opracowaniu technik działania na DNA.

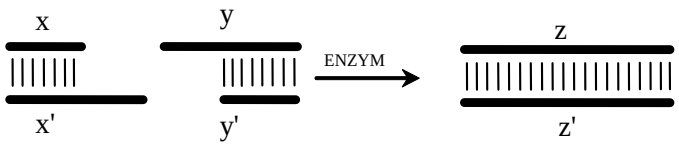
Wyróżniamy następujące podstawowe operacje (działania) na DNA:

1. **Cięcie**. Związki chemiczne zwane enzymami restrykcyjnymi umożliwiają precyzyjne cięcie łańcuchów DNA (Rys. 14) w określonym miejscu np. enzym *Bse*XI rozpoznaje następującą sekwencję nukleotydów GCAGC, a następnie tnie łańcuch DNA w odległości 8 nukleotydów od znalezionej sekwencji.



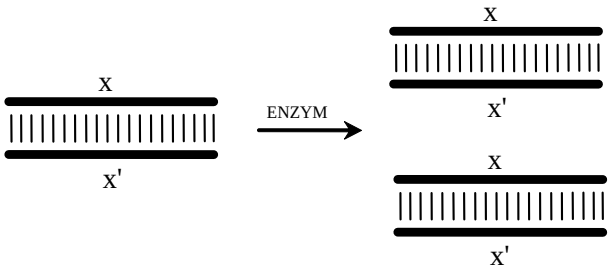
Rys. 14. Cięcie DNA.

2. **Łączenie**. Enzymy ligazy umożliwiają łączenie za sobą łańcuchów DNA (Rys.15) np. połączenie dwóch łańcuchów DNA z komplementarnymi odcinkami w jeden długi.



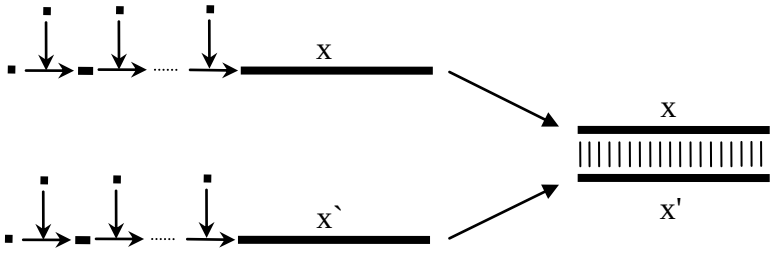
Rys. 15. Łączenie łańcuchów DNA.

3. **Kopiowanie**. Enzymy polimerazy kopiują informacje z jednego łańcucha DNA na drugi (Rys. 16) np. polimeraza DNA tworzy kopię komplementarną do nici matrycy.



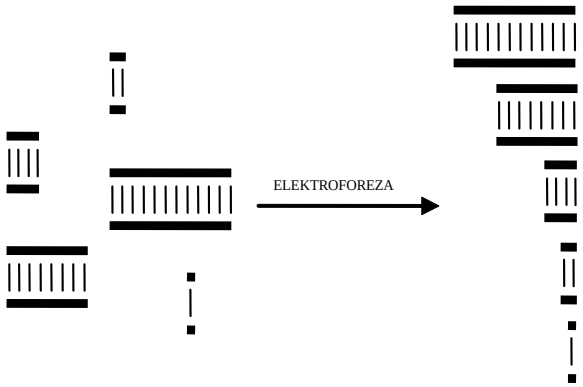
Rys. 16. Kopiowanie DNA.

4. **Synteza**. Współczesne laboratoria biologii molekularnej umożliwiają syntezę wcześniej zaprojektowanego łańcucha DNA o dowolnej sekwencji zasad (Rys. 17). Wytwarzane są początkowo dwa jednoniciowe fragmenty DNA, a następnie zlepiane są one komplementarnie ze sobą.



Rys. 17. Synteza DNA.

5. **Sortowanie**. Do rozdzielania łańcuchów DNA o różnej długości wykorzystuje się elektroforezę na żelu. Analizowany roztwór DNA nanosi się na płytkę uformowaną z żelu elektroforetycznego. Wzdłuż krawędzi płytki biegną elektrody. Pod wpływem stałego napięcia elektrycznego łańcuchy DNA ulegają stopniowemu rozdzieleniu i sortowaniu ze względu na długość łańcuchów DNA, czyli ilości nukleotydów (Rys.18).



Rys. 18. Sortowanie DNA.

2.2. Praktyczne implementacje problemów informatycznych

Podstawowe operacje na DNA umożliwiły implementacje laboratoryjne wielu problemów informatycznych. Badania doświadczalne nad DNA obliczeniami koncentrują się głównie na praktycznym zaimplementowaniu, metodami inżynierii genetycznej, znanych algorytmów informatycznych za pomocą odpowiedniego konstruowania łańcuchów DNA. Badania te mają na celu udowodnienie, że możliwe jest rozwiązanie problemów informatycznych przez kodowanie danych wejściowych łańcuchami DNA, z użyciem znanych operacji na DNA. Często są to problemy NP-zupełne, które można rozwiązać korzystając z masowej równoległości wykonywanych obliczeń. Świadczy to o potencjalnej przewadze DNA obliczeń nad klasyczną implementacją za pomocą przepływu elektronów.

2.2.1. Wybrane doświadczenia laboratoryjne

W 1994 roku Leonard Adleman [2] jako pierwszy rozwiązał doświadczalnie (laboratoryjnie) znany problem informatyczny (drogi Hamiltona w grafie) za pomocą odpowiednio zaprojektowanych cząsteczek DNA. Ogólna idea metody Adlemana polega na zakodowaniu danych wejściowych w postaci relatywnie krótkich jednoniciowych łańcuchów (oligonukleotydów). W roztworze łańcuchy te łączą się komplementarnie na wiele sposobów, a pewne z nich reprezentują rozwiązanie problemu w postaci dwuniciowych łańcuchów DNA. Zadanie praktyczne polega na wykryciu, za pomocą operacji na DNA, takich właściwych łańcuchów. Adleman wykorzystał komplementarność w cząsteczkach DNA, czyli własność do łączenia się zasad: adeniny z tyminą oraz guaniny z cytozyną. W probówce umieścił nici DNA kodujące wszystkie wierzchołki oraz krawędzie grafu i po kilku minutach wygenerował szukaną drogę w grafie (wśród wielu innych niewłaściwych) reprezentowaną przez łańcuch DNA. Ze względu na pracochłonność metod biotechnologicznych odczyt informacji wynikowej zajął mu jednak bardzo dużo czasu. Do wykrycia właściwego wyniku Adleman wykorzystał następujące metody biotechnologiczne: PCR (reakcja łańcuchowa polimeryzacji), elektroforezę (rozdział na żelu) oraz amplifikację (rozdział oparty na powinowactwie).

W 1997 roku M. Ogihara i inni [24] udowodnili, że DNA może zostać użyte do implementacji standardowego modelu przetwarzania używanego w informatyce, a mianowicie bramek logicznych (sieci logicznych). Główna idea tej implementacji jest następująca. Dla danej sieci logicznej każdy wierzchołek (wejściowy i bramki logiczne) ma przypisany unikalny jednoniciowy łańcuch DNA o tej samej długości (parzystej) 2N. Podobnie każdej krawędzi przypisany jest też jednoniciowy łańcuch DNA długości 2N,

którego pierwsza część jest komplementarna do drugiej części łańcucha odpowiadającego początkowi krawędzi, a druga część jest komplementarna do pierwszej części łańcucha odpowiadającego końcowi wierzchołka krawędzi. Zatem jeżeli w roztworze występują łańcuchy odpowiadające wierzchołkom oraz krawędzi łączącej je, to w roztworze powstaną łańcuchy DNA długości 4N. Za pomocą elektroforezy w żelu usuwa się z roztworu łańcuchy o mniejszej długości (<4N). Proces przetwarzania informacji przebiega etapami wykonywanymi nieautonomicznie. Konieczna jest ingerencja człowieka (laboranta) w poszczególnych etapach eksperymentu.

Ciekawe badania dotyczące bramek logicznych wykonane zostały w Warszawie przez zespół z Politechniki Warszawskiej oraz z Instytutu Biotechnologii i Antybiotyków. Grupa ta przedstawiła szereg prac [23], [41], [42], [43], implementujących systemy wnioskujące oraz funkcje logiczne realizowane za pomocą łańcuchów DNA.

Kolejne ciekawe badania doświadczalne przeprowadził N. Seemann. Z badań autora [33], [34] wynika, że DNA ma wiele właściwości umożliwiających potencjalne zastosowanie w nanotechnologii szczególnie w systemach samowytwarzania struktur. Seeman podał jako pierwszy techniczne, inne niż liniowa, możliwości budowy struktur opartych na DNA. Do przetwarzania informacji nadają się znakomicie struktury zbudowane z DNA o czterech ramionach kończących się lepkimi końcami. Elementy te można wykorzystać do samowytwarzania struktur przestrzennych, a nawet implementować problemy informatyczne oparte na samowytwarzaniu. W 1998 roku E. Winfree i inni [44] przedstawili, oparty na matematycznej „teorii części", programowalną strukturę samowytwarzania. Programowanie zostało osiągnięte przez komplementarność łańcuchów DNA oraz odpowiednie kodowanie lepkich końców struktur zbudowanych z DNA.

W 1999 roku D. Faulhammer i inni [11] zaimplementowali za pomocą DNA i RNA znany NP-problem zupełny SAT na przykładzie „problemu skoczka szachowego", który polega na rozmieszczeniu skoczków na szachownicy $n \times n$ tak, by wzajemnie nie atakowały się. Wykonanie praktyczne polegało na wytworzeniu, najpierw w postaci łańcuchów DNA, biblioteki wszystkich możliwych rozmieszczeń skoczków na szachownicy, a następnie dokonanie transkrypcji ich w postaci RNA. Każda ze zmiennych reprezentująca obecność skoczka w danym polu może przyjmować wartość 0 (brak skoczka na tym polu) lub 1 (skoczek jest obecny na tym polu). Ogólnie metoda polega na zaznaczeniu „złych" łańcuchów RNA (odpowiadających złym ustawieniom skoczków) poprzez dodanie do

roztworu odpowiednich komplementarnych łańcuchów DNA oraz na niszczeniu za pomocą *RNasy H* (rybonukleaza H niszczy wiązania fosfodiestrowe RNA połączonych z DNA).

W 2001 roku zespół [4] pod kierunkiem E. Shapiro przedstawił model pierwszego autonomicznego automatu skończonego zbudowanego z DNA oraz enzymu restrykcyjnego *Fok*I. Dwa lata później Shapiro i inni [5] przedstawili ulepszony automat zbudowany z DNA. Zostanie on dokładnie omówiony w następnym rozdziale. W 2009 r. zespół Shapiro i inni [26] zademonstrował również prosty system wnioskujący zbudowany z DNA oraz enzymu restrykcyjnego *Fok*I.

W 2003 roku M. Stojanovic i inni [36] przedstawili implementację za pomocą DNA algorytmu gry w kółko i krzyżyk (problem tic-tac-toe). Automat ten został nazwany MAYA i do jego działania użyta została cząsteczka zbudowana z DNA o katalitycznych właściwościach (*Deoxyrybozym*). Istota konstrukcji polega na budowie odpowiednich łańcuchów DNA umieszczonych w poszczególnych probówkach (odpowiadającym polom gry) w postaci charakterystycznych struktur w kształcie „spinki". Ruchy oponenta polegają na dodaniu do wszystkich probówek tego samego łańcucha DNA, odpowiadającego numerowi pola, w którym oponent chce postawić symbol kółka. W poszczególnych probówkach zachodzą reakcje na DNA, w konsekwencji których jedna z nich zmienia barwę dzięki fluorescencji (jest to odpowiedź DNA komputera na ruch oponenta). Autorzy zwracają uwagę, że metoda może zostać w przyszłości wykorzystana do szybkiego diagnozowania we krwi wirusów, nowotworów itd.

2.2.2. Porównanie implementacji praktycznych

Prace naukowe w zakresie DNA obliczeń koncentrują się na dwóch rodzajach badań. Pierwszym są teoretyczne rozważania nad możliwościami budowy modeli (systemów) zbudowanych z DNA. Drugi kierunek badań to doświadczenia laboratoryjne polegające na opracowaniu różnych implementacji znanych algorytmów lub modeli przetwarzania informacji używanych w informatyce z zastosowaniem metod inżynierii genetycznej. Dostrzegalna jest różnorodność koncepcji budowy danych wejściowych oraz wyjściowych, a także procesu przetwarzania informacji. Poniższa tabela (Rys. 19) zestawia poszczególne implementacje praktyczne obliczeń za pomocą DNA.

Eksperyment	**Rozwiązany problem**	**Kodowanie informacji**		**Przetwarzanie**
		Wejście	**Wyjście**	
Adleman (1994 r.)	Droga Hamiltona w grafie	ssDNA	dsDNA	Komplementarne zlepianie oligonukleotydów
Ogihara i inni (1997 r.)	Bramki logiczne	ssDNA	dsDNA	Komplementarne zlepianie oligonukleotydów
Winfree i inni (1998 r.)	Samowytwarzanie struktur przestrzennych	dsDNA	dsDNA	Komplementarne łączenie geometrycznych struktur DNA w kształcie krzyża
Faulhammer i inni (1999)	Rozmieszczenie skoczków	ssRNA ssDNA	ssRNA	Komplementarne zlepianie RNA z DNA. Trawienie: enzym *RNasa H*
Shapiro i inni (2001)	Problemy rozwiązywane przez dwustanowe automaty skończone	dsDNA	dsDNA	Komplementarne łączenie lepkich końców Trawienie: enzym *Fok*I
Stojanovic i inni (2003)	Gra w „kółko i krzyżyk”	ssRNA dsDNA	ssRNA	Komplementarne łączenie Trawienie: *Deoksyrybozym*

Rys. 19. Porównanie wybranych implementacji praktycznych DNA obliczeń.

Sposoby kodowania słowa wejściowego w implementacjach praktycznych różnią się znacząco od siebie. W pracy L. Adlemana [2] wykorzystane są jednoniciowe oligonukleotydy DNA, które zawierają informacje o rozwiązywanym problemie. Podobną koncepcję przedstawiają M. Ogihara i inni [24], którzy konstruują jednoniciowe łańcuchy DNA realizując laboratoryjnie bramki logiczne. W swoich badaniach E. Shapiro i jego zespół [4], [5] używa natomiast jako wejścia dwuniciowe fragmenty DNA, którym koduje dowolne słowo z alfabetu podlegające przetwarzaniu. Kolejny autor (D. Faulhammer i inni [11]) konstruuje natomiast dwuniciowe biblioteki DNA, które ulegają niszczeniu w procesie przetwarzania informacji (realizacji algorytmu). Zupełnie inne podejście do konstruowania słowa wejściowego przedstawia M. Stojanovic i inni [36]. W implementacji tej operuje się na specyficznych cząsteczkach zbudowanych z DNA. Charakteryzują się one w odróżnieniu od poprzednich rozwiązań zupełnie inną konstrukcją - mają kształt „spinki”. W każdym z

podejść programowanie odbywa się przez umieszczenie w roztworze, w którym znajdują się dane wejściowe, odpowiedniego jedno lub dwuniciowego DNA.

W procesie przetwarzania informacji w poszczególnych implementacjach praktycznych występują również znaczne różnice. Jednak we wszystkich podejściach cały proces przetwarzania informacji zachodzi dzięki komplementarności nici DNA. W podejściu L. Adlemana [2] oraz N. Seemana [44] wykorzystuje się wyłącznie komplementarność. Natomiast w pozostałych rozwiązaniach zastosowano dodatkowo operację cięcia DNA realizowaną enzymami restrykcyjnymi [4], *Deoksyrybozymem* [36] lub *RNazą H* [11]. Proces przetwarzania informacji w pracach E. Shapiro i inni [4], [5] oraz D. Faulhammera i inni [11] polega na „niszczeniu" odpowiednio zaprojektowanego wcześniej słowa wejściowego. Natomiast w przypadku L. Adlemana [2] oraz N. Seemana [44] proces przetwarzania informacji związany jest z „samowytwarzaniem", czyli automatycznym konstruowaniem struktur zbudowanych z DNA.

Zakończenie procesu przetwarzania informacji zostało rozwiązane w różny sposób w poszczególnych implementacjach. U L. Adlemana [2] wyjściem jest dwuniciowy łańcuch DNA o liniowej strukturze, określonej długości oraz sekwencji zasad, który należy wykryć w roztworze. W przypadku N. Seemana [44] proces przetwarzania kończy się odpowiednią strukturą DNA, będącą rozwiązaniem problemu; jest ona jednak bardziej skomplikowana (wielowymiarowa) niż w podejściu L. Adlemana. Automat E. Shapiro i inni [4] kończy natomiast działanie, gdy w roztworze pojawi się odpowiednia cząsteczka terminalna, świadcząca o akceptacji słowa wejściowego.

Na obecnym etapie rozwoju metod używanych w biotechnologii występują jednak duże problemy z odczytem rezultatów działań na DNA. Do analizy danych wyjściowych stosuje się głównie metodę PCR (reakcja polimeryzacji), za pomocą której powiela się określone sekwencje DNA oraz metodę elektroforezy na żelu. W przypadku przetwarzania informacji uzyskanego przez M. Stojanovica i inni [36] zastosowano bardzo ciekawą i wymagającą mniejszego nakładu pracy metodę fluorescencji.

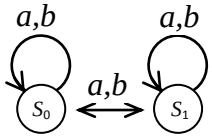
Rozdział 3

Dwustanowy automat Shapiro

W 2001 roku Y. Benenson, T. Paz-Elizur, R. Adar, E. Keinan, Z. Livneh, E. Shapiro [4] przedstawili implementację laboratoryjną automatu skończonego (czyli najprostszego modelu komputera) zbudowanego z DNA (będziemy go nazywali automatem Shapiro). W 2003 roku zespół pod kierunkiem E. Shapiro [5] przedstawił ulepszony automat skończony zbudowany z DNA (bez konieczności stosowania enzymu *ligaza*). Idea budowy automatu Shapiro zbliżona jest do hipotetycznego biomolekularnego urządzenia opracowanego w 1973 roku przez CH. Bennetta [7] oraz do teoretycznego modelu Maszyny Turinga zbudowanego z DNA i przedstawionego w 1995 roku przez P. Rothemunda [28]. Zespół pod kierunkiem E. Shapiro [4], [5] udowodnił jednak, że możliwe jest praktyczne skonstruowanie automatów skończonych zbudowanych z DNA, działających autonomicznie tzn. bez dodatkowego działania pośredniego człowieka za wyjątkiem wprowadzenia danych wejściowych oraz „odczytania" danych wyjściowych.

3.1. Opis działania

Każdy komputer, także zbudowany z DNA, powinien być implementacją jednego ze znanych modeli teoretycznych komputerów. Najprostszym modelem przetwarzania informacji jest automat skończony opisany dokładnie w rozdziale 1. Implementację laboratoryjną prostego niedeterministycznego automatu skończonego zbudowanego z DNA przedstawił zespół pod kierunkiem E. Shapiro [4], [5]. Odpowiednie kodowanie stanów oraz symboli za pomocą DNA umożliwiło autorom budowę dowolnego 2-stanowego 2-symbolowego automatu skończonego. Ma on $8=2^3$ możliwych ruchów, które przedstawione są na poniższym rysunku.

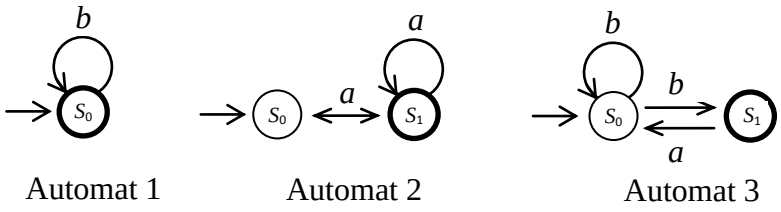


Rys. 20. Możliwe przejścia automatu 2-stanowego.

Zatem dla tego modelu poprzez wybór ruchów możemy skonstruować dowolny niedeterministyczny automat skończony 2-stanowy 2-symbolowy.

Przykład 6.

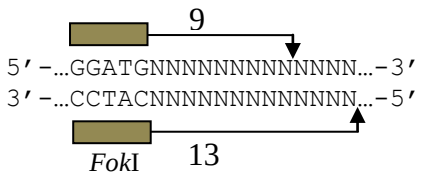
Wybrane automaty oparte na modelu 2-stanowym 2-symbolowym:



Rys. 21. Przykłady automatów 2-stanowych 2-symbolowych.

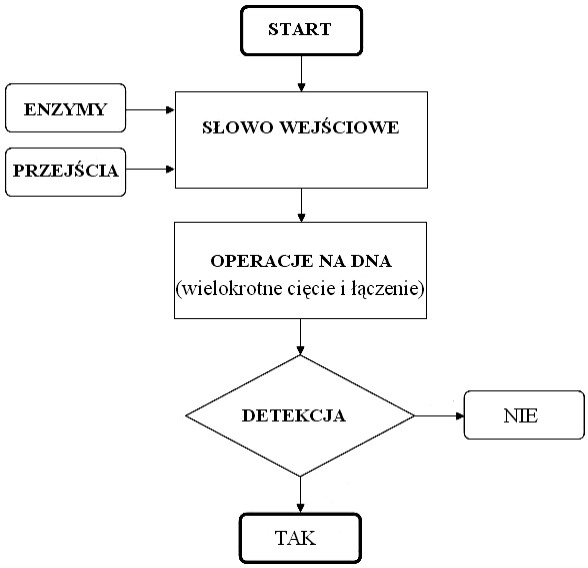
Automat Shapiro umożliwia przeprowadzanie autonomicznego i programowalnego przetwarzania informacji za pomocą wyboru odpowiedniego zestawu łańcuchów DNA reprezentujących reguły przejść. Model automatu 2-symbolowego 2-stanowego przedstawiony został za pomocą, adekwatnego do teoretycznego modelu automatu skończonego, zbioru odpowiednio dobranych łańcuchów DNA. Konieczne było zatem zakodowanie za pomocą DNA: symboli, krawędzi (czyli przejść) oraz stanów.

Dodatkowym elementem budowy automatu Shapiro jest enzym (odpowiednik teoretyczny głowicy), czyli związek chemiczny umożliwiający wykonywanie operacji cięcia łańcuchów DNA. Idea budowy modelu komputera opartego na teoretycznie działających enzymach została po raz pierwszy przedstawiona w 1973 roku przez Ch. Bennetta [7]. Teoretyczna analiza możliwości budowy maszyny Turinga (ale bez implementacji laboratoryjnej) z zastosowaniem komercyjnych enzymów restrykcyjnych dostarczanych przez firmę *New England Biolabs* wykonana została po raz pierwszy przez P. Rothemunda [28]. Dopiero w 2001 roku zespół pod kierunkiem E. Shapiro [4] zastosował laboratoryjnie komercyjny enzym restrykcyjny o nazwie handlowej *Fok*I do budowy automatu skończonego. Enzym ten rozpoznaje określoną sekwencję zasad azotowych w łańcuchu DNA, a dokładniej sekwencję GGATG na łańcuchu DNA w kierunku 5'→3'. Po znalezieniu tej sekwencji enzym wykonuje cięcie DNA po 9 symbolach (dowolnych) w kierunku 5'→3' oraz po 13 symbolach (dowolnych) w kierunku 3'→5' (Rys.22).



Rys. 22. Działanie enzymu *Fok*I.

W modelu automatu Shapiro proces przetwarzania informacji, czyli akceptacji lub nieakceptacji słowa, polega na przemiennym cięciu oraz łączeniu odpowiednio zbudowanych łańcuchów DNA. Schemat ideowy przetwarzania informacji w automacie Shapiro przedstawia się następująco (Rys. 23)



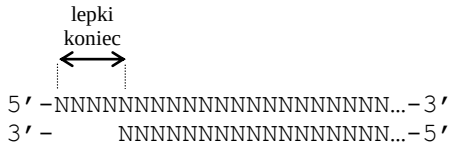
Rys. 23. Schemat ideowy automatu Shapiro.

Zgodnie ze schematem ideowym przedstawionym na rysunku 36 automat rozpoczyna działanie po zmieszaniu ze sobą w wielu kopiach (wielozbiór): słowa wejściowego, wybranych przejść oraz enzymu *Fok*I. Dodatkowo w pierwszym modelu automatu Shapiro zastosowano enzym *ligaza*, który przyspiesza łączenie się łańcuchów DNA. Podczas przeprowadzenia przez autorów testów eliminacji zbędnych elementów automatu udało się pominąć *ligazę*, która wymagała dostarczania energii w postaci związku chemicznego ATP. Przetwarzanie informacji odbywa się w wyniku naprzemiennego cięcia oraz łączenia łańcuchów DNA. Doprowadza to do akceptacji (lub nieakceptacji) słowa wejściowego. Zakończenie działania (akceptacja) związane jest z pojawieniem się w roztworze łańcucha DNA o określonej długości. Zostaną teraz omówione dokładnie wszystkie elementy budowy automatu Shapiro, a mianowicie: kodowanie słowa wejściowego oraz przejść. Dodatkowo omówiony zostanie sposób kodowania par <stan, symbol> oraz mechanizm działania enzymu *Fok*I.

Budowa słowa wejściowego

Podstawowym elementem automatu Shapiro jest słowo wejściowe $x \in \{a,b\}^*$ nad alfabetem $\Sigma = \{a,b\}$, kodowane określonymi sekwencjami zasad azotowych: adeniną (A),

tyminą (T), guaniną (G), cytozyną (C). Łańcuch kodujący słowo zaczyna się tzw. lepkim końcem, który koduje pierwszy symbol słowa *x* oraz jednocześnie stan początkowy.



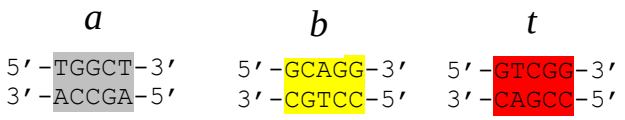
Rys. 24. Lepki koniec słowa wejściowego.

W automacie Shapiro zakodowano sześć par <stan, symbol>, które reprezentowane są fizycznie odpowiednimi sekwencjami w lepkim końcu (Rys. 25).

<stan,symbol>	*a*	*b*	*t*
S_0	5′-GGCT-3′	5′-CAGG-3′	5′-TCGG-3′
S_1	5′-TGGC-3′	5′-GCAG-3′	5′-GTCG-3′

Rys. 25. Sekwencje zasad azotowych kodujących pary <stan, symbol>.

Słowo wejściowe rozpoczyna się lepkim końcem po którym kodowane są kolejne symbole a, b z alfabetu Σ słowa *x*. Dodatkowo wprowadzono trzeci symbol t oznaczający sekwencję terminalną, której znaczenie wyjaśnimy dalej. Poszczególne symbole kodowane są w następujący sposób (Rys. 26).

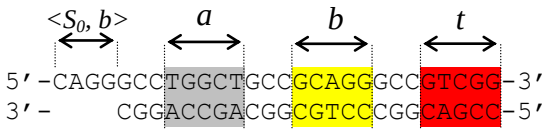


Rys. 26. Kodowanie symboli w automacie Shapiro.

Dodatkowo między symbolami umieszczone zostały sekwencje GCC niekodujące symbole słowa, ale umożliwiające właściwe działanie enzymu *Fok*I.

Przykład 7

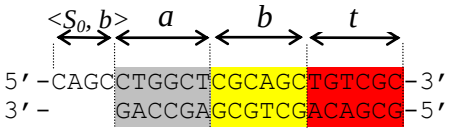
Słowo wejściowe *bab* przy stanie początkowym S_0 kodowane jest następująco (Rys. 27).



Rys. 27. Kodowanie słowa $x = bab$.

W pierwszej pracy zespołu Shapiro i inni z 2001 roku [4] symbole kodowane były sześcioma nukleotydami oraz nie zastosowano sekwencji GCC (Rys. 28). Kodowanie słowa

wejściowego *bab* przy stanie początkowym S_0 w pierwszej pracy zespołu Shapiro było następujące.



Rys. 28. Kodowanie słowa $x = bab$ w pierwszej pracy zespołu Shapiro.

Zastosowanie sekwencji GCC wynikało z testów laboratoryjnych mających na celu eliminację któregoś z elementów automatu, a tym samym uproszczenie jego budowy. W eksperymencie z 2003 roku [5] pominięta została *ligaza*, która przyspiesza łączenie łańcuchów DNA. Eliminacja *ligazy* doprowadziła jednak do zmniejszenia efektywności całego procesu. Dopiero zmiana w kodowaniu słowa wejściowego (zastosowanie sekwencji GCC) zwiększyła wydajność zachodzącego procesu bez użycia *ligazy*.

Kodowanie przejść

Kolejnym elementem budowy są wszystkie możliwe przejścia (ruchy) dla modelu automatu 2-symbolowego 2-stanowego. Każde przejście zostało zakodowane innym łańcuchem DNA, w taki sposób, aby umożliwiało to odpowiednie działanie enzymu *Fok*I.

T1: $S_0 \xrightarrow{a} S_0$	5′-GGATGC -3′ 3′-CCTACGCCGA-5′	T5: $S_1 \xrightarrow{a} S_0$	5′-GGATG -3′ 3′-CCTACACCG-5′
T2: $S_0 \xrightarrow{a} S_1$	5′-GGATGCC -3′ 3′-CCTACGGCCGA-5′	T6: $S_1 \xrightarrow{a} S_1$	5′-GGATGC -3′ 3′-CCTACGACCG-5′
T3: $S_0 \xrightarrow{b} S_0$	5′-GGATGC -3′ 3′-CCTACGGTCC-5′	T7: $S_1 \xrightarrow{b} S_0$	5′-GGATG -3′ 3′-CCTACCGTC-5′
T4: $S_0 \xrightarrow{b} S_1$	5′-GGATGCC -3′ 3′-CCTACGGGTCC-5′	T8: $S_1 \xrightarrow{b} S_1$	5′-GGATGC -3′ 3′-CCTACGCGTC-5′

Rys. 29. Kodowanie przejść w automacie Shapiro.

Łańcuchy DNA kodujące przejścia zawierają sekwencję rozpoznawaną przez enzym *Fok*I, czyli GGATG w kierunku 5'→3'. Umożliwia to sterowanie procesem cięcia łańcucha DNA kodującego słowo wejściowe.

Przetwarzanie informacji

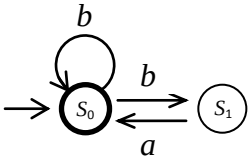
Proces przetwarzania rozpoczyna się od umieszczenia w probówce wszystkich elementów budowy automatu: enzymu *Fok*I, łańcuchów DNA reprezentujących słowo wejściowe oraz przejść wybranego automatu. Pierwszym etapem jest połączenie łańcuchów DNA reprezentujących słowo z łańcuchami reprezentującymi przejścia, realizowane dzięki

komplementarności. Zastosowanie enzymu o katalitycznych właściwościach (enzym *ligaza*) umożliwia przyspieszenie łączenia łańcuchów o komplementarnych lepkich końcach. Łączenie łańcuchów DNA za pomocą *ligazy* było jedynym etapem przetwarzania informacji wymagającym dostarczania energii (w postaci ATP). Eliminacja enzymu *ligaza* w procesie przetwarzania informacji [5] wymagała zmiany sposobu kodowania słowa oraz przejść, jednak umożliwiła zespołowi Shapiro i inni zmniejszenie wydatkowania energii na wykonywanie operacji na DNA.

Proces przetwarzania słowa polega na stopniowej analizie symboli za pomocą enzymu *Fok*I (operacja cięcia). Wyjaśni to poniższy przykład, który przedstawia działanie modelu Shapiro dla określonego automatu i słowa akceptowanego.

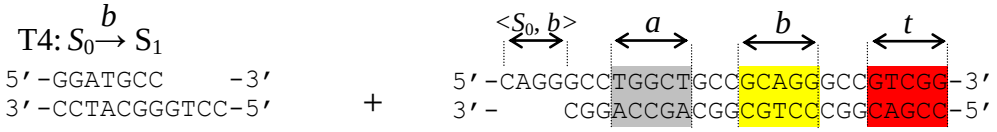
Przykład 8

Proces przetwarzania informacji dla automatu M (Rys. 30) i słowa $x = bab$.



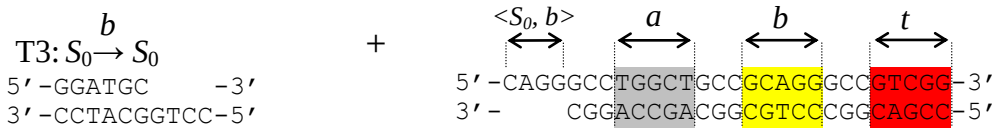
Rys. 30. Przykład 2-stanowego automatu skończonego.

Automat M akceptuje następujący język regularny $L_M = L((b+ba)^*)$. Do budowy automatu przedstawionego na rysunku 30 konieczne jest zakodowanie w postaci łańcuchów DNA słowa $x = bab$ (Rys. 27) oraz zastosowanie łańcuchów DNA kodujących przejścia T3, T4, T5 (Rys. 29). Lepki koniec CAGG słowa x jest komplementarny do lepkiego końca GTCC przejść T3 oraz T4. Dla słowa $x = bab$ prawidłowe jest połączenie z przejściem T4, gdyż umożliwi to wykonanie kolejnych przejść i akceptację całego słowa.



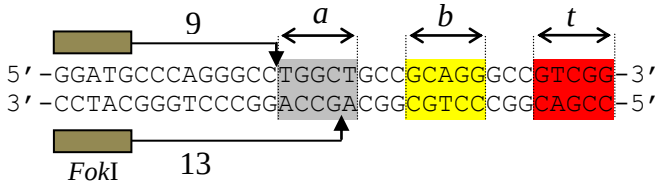
Rys. 31. Łączenie słowa wejściowego z przejściem T4.

Możliwe jest połączenie z przejściem T3, ale to nie doprowadzi jednak do sekwencji terminalnej.

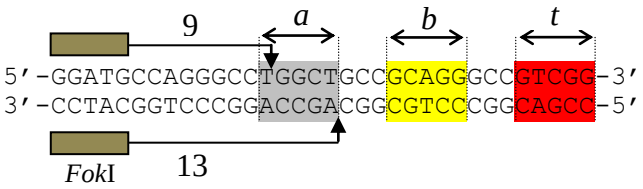


Rys. 32. Łączenie słowa wejściowego z przejściem T3.

Kolejnym etapem procesu przetwarzania jest cięcie łańcuchów enzymem *Fok*I.

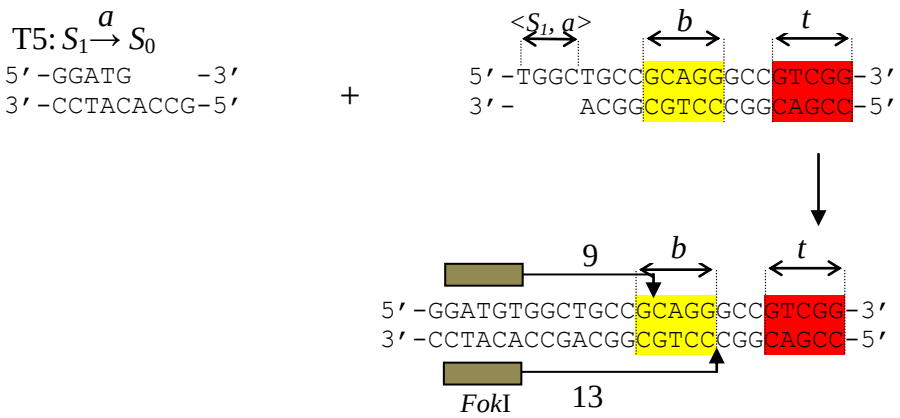


Rys. 33. Pierwsze cięcie enzymem *Fok*I dla przejścia T4.



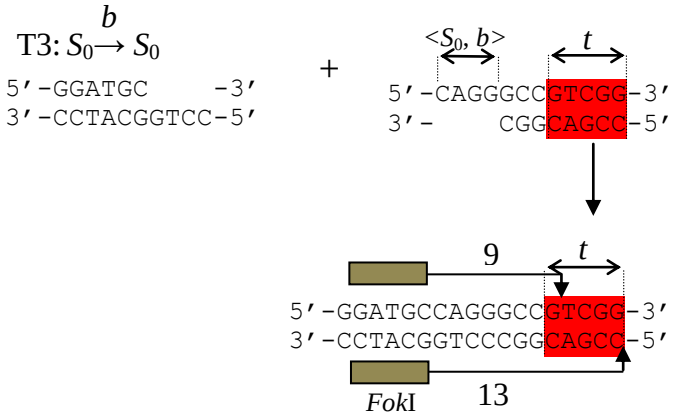
Rys. 34. Pierwsze cięcie enzymem *Fok*I dla przejścia T3.

W wyniku działania enzymu *Fok*I (pierwsze cięcie) usunięty zostaje symbol b ze słowa wejściowego, po którym zostaje lepki koniec TGGC kodujący parę $< S_1, a >$. Lepki koniec ACCG łańcucha DNA kodującego przejście T5 jest komplementarny do tego lepkiego końca słowa. Wykonane zostanie połączenie tych łańcuchów i drugie cięcie enzymem *Fok*I.



Rys. 35. Drugie cięcie enzymem *Fok*I.

Kolejne przejście T3 jest komplementarne do lepkiego końca uzyskanego w wyniku cięcia enzymem (Rys. 36).



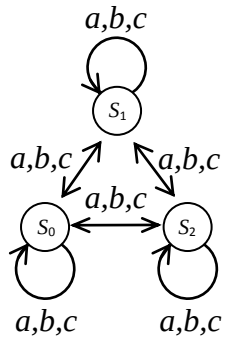
Rys. 36. Trzecie cięcie wykonane enzymem *Fok*I.

Powyższe cykliczne operacje cięcia (dokładnie trzy cięcia enzymem *Fok*I) prowadzą do sekwencji terminalnej i pojawienia się lepkiego końca TCGG, który koduje parę $< S_0, t >$. Ponieważ w automacie stan S_0 jest końcowym, więc oznacza to akceptację słowa i zakończenie działania automatu w stanie S_0.

3.2. Znane rozszerzenia automatu Shapiro

Badania E. Shapiro i innych [4], [5], udowodniły, że możliwe jest autonomiczne i programowalne przetwarzanie informacji za pomocą naprzemiennych operacji cięcia i łączenia łańcuchów DNA. Model ten ograniczony jest jednak do dwóch stanów i dwóch symboli (automat 2-stanowy 2-symbolowy). Badania przedstawione przez zespół Shapiro skupiły uwagę wielu naukowców. Prowadzono dalsze prace zmierzające do rozszerzenia liczby stanów oraz akceptowanych symboli.

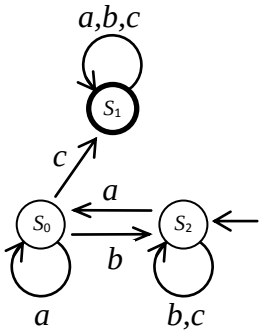
W 2005 roku Soreni i inni [35] przedstawili możliwość wykonania praktycznego (laboratoryjnego) automatu, opartego na cięciu i łączeniu łańcuchów DNA, o większej liczbie stanów i symboli (automat 3-stanowy 3-symbolowy). Autorzy przedstawili również koncepcję unieruchomienia wejściowego łańcucha DNA na płytce reprezentującej mikromacierz. Przedstawiono możliwość rozszerzenia automatu Shapiro do modelu 3-stanowego 3-symbolowego (Rys. 37).



Rys. 37. Wszystkie możliwe przejścia dla automatu 3-stanowego 3-symbolowego.

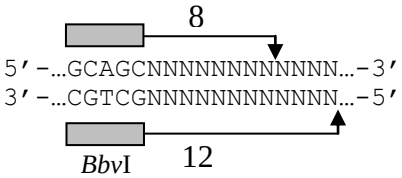
Przykład 9

W doświadczeniu przebadano eksperymentalnie następujący automat 3-stanowy 3-symbolowy.



Rys. 38. Automat 3-stanowy 3-symbolowy badany w eksperymencie.

W doświadczeniu tym użyty został enzym restrykcyjny *Bbv*I oraz *ligaza* przyspieszająca łączenie łańcuchów DNA. Enzym *Bbv*I rozpoznaje sekwencję GCAGC, a następnie tnie łańcuch DNA po 8 oraz 12 nukleotydach.



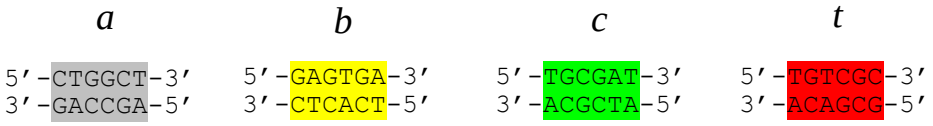
Rys. 39. Działanie enzymu *Bbv*I.

Cały proces przetwarzania zgodny jest ze schematem ideowym automatu Shapiro (Rys. 23). Sposób konstruowania słowa wejściowego zbliżony jest do metody użytej w automacie Shapiro. Słowo wejściowe w postaci łańcuchów DNA zawiera sekwencję rozpoznawaną przez enzym *Bbv*I. Dodatkowym nowym elementem jest umieszczenie biotyny na końcu łańcucha DNA reprezentującego słowo wejściowe. Biotyna umożliwia przytwierdzenie słowa wejściowego do mikromacierzy i wykonanie równoległego działania na wielu słowach wejściowych. Symbole z alfabetu $\Sigma = \{a, b, c\}$ kodowane są podobnie jak w modelu Shapiro, a pary <stan, symbol> reprezentowane są następującymi sekwencjami nukleotydów (Rys. 40).

<stan,symbol>	*a*	*b*	*c*	*t*
S_0	5′-CTGG-3′	5′-GAGT-3′	5′-TGCG-3′	5′-TGTC-3′
S_1	5′-TGGC-3′	5′-AGTG-3′	5′-GCGA-3′	5′-GTCG-3′
S_2	5′-GGCT-3′	5′-GTGA-3′	5′-CGAT-3′	5′-TCGC-3′

Rys. 40. Sekwencje zasad azotowych kodujących pary <stan, symbol>.

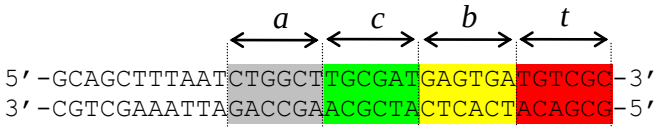
Poszczególne symbole kodowane są sześcioma nukleotydami w następujący sposób (Rys. 41).



Rys. 41. Kodowanie symboli w automacie 3-stanowym 3-symbolowym.

Przykład 10

Przykładowo słowo *acb* kodowane jest w następujący sposób (Rys. 42).



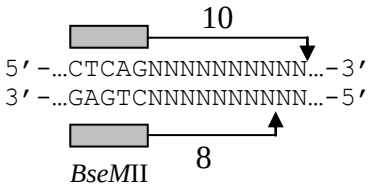
Rys. 42. Kodowanie słowo acb.

Kolejnym elementem automatu są ruchy reprezentowane odpowiednio skonstruowanym zbiorem łańcuchów DNA kodujących relacje przejść. Autorzy opracowali bibliotekę 27 łańcuchów DNA reprezentującą wszystkie możliwe przejścia dla automatu 3-stanowego 3-symbolowego. Model ten został eksperymentalnie przetestowany dla czterech różnych słów *a*, *ac*, *acb* oraz *bc*. Każde słowo zostało unieruchomione na jednym z czterech sektorów mikromacierzy (biochip).

Autorzy zwracają uwagę, że podana laboratoryjna implementacja umożliwia teoretycznie rozszerzenie automatu Shapiro do modelu automatu 3-stanowego 37-symbolowego. Zwiększanie liczby stanów powoduje jednak znaczny wzrost liczby potrzebnych przejść (ruchów) koniecznych do zakodowania za pomocą DNA. Przykładowo dla automatu 3-stanowego 3-symbolowego potrzebna jest biblioteka 27 molekuł przejść, kodujących wszystkie możliwe krawędzie dla grafu takiego automatu. Autorzy zwracają uwagę, że opracowanie automatu 3-stanowego o większej liczbie symboli wymaga zwiększania kombinacji zasad azotowych w lepkich końcach.

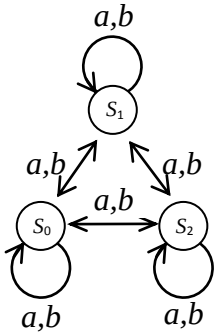
W swoim doświadczeniu M. Soreni i inni [35] skonstruowali 3-stanowy 3-symbolowy automat kodując symbole automatu, różnymi kombinacjami sześciu zasad azotowych (6 bp). W rzeczywistości możliwe jest zakodowanie większej liczby symboli. Zatem na mocy wzoru na ilość wariacji 4 elementowych ze zbioru 4 elementowego otrzymujemy, że jest ich $4^4 = 256$. Konieczna jest eliminacja palindromowych lepkich końców np. TCGA, ze względu na możliwość komplementarnego zlepienia się ich ze sobą. Ponieważ każdy 4 symbolowy palindrom jest jednocześnie określony przez dwa pierwsze symbole więc jest ich $4^2 = 16$. W związku z tym liczba możliwych lepkich końców została zredukowana do 240. Dla każdego symbolu kodowanego 6 parami zasad w automacie 3 stanowym muszą istnieć 3 różne lepkie końce kodujące wszystkie możliwe pary - ten symbol i dowolny stan. To daje liczbę możliwych symboli 240/3=80. Ponieważ każdy lepki koniec (niepalindromowy) ma komplementarny lepki koniec, więc aby uniknąć ich zlepienia (podobnie jak w przypadku palindromów), więc ilość symboli możliwych do zakodowania jest w końcu równa 80/2=40. Sugerowane jest, zatem maksymalne zakodowanie 39 symboli oraz 1 symbol terminalny oparte na tej technologii.

W 2004 rok O. Unold, M. Troć, T. Dobosz, A. Trusiewicz [40], grupa naukowców z Wrocławia rozszerzyła i zweryfikowała w laboratorium automat Shapiro o trzech stanach stosując inny enzym restrykcyjny *Bse*MII, który umożliwia wykonanie cięcia na łańcuchu DNA w następujący sposób (Rys. 43).



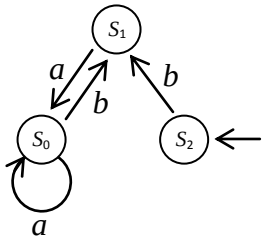
Rys. 43. Działanie enzymu *Bse*MII.

Kodowanie za pomocą łańcuchów DNA umożliwiło autorom opracowanie modelu 3-stanowego 2-symbolowego.



Rys. 44. Możliwe przejścia dla automatu 3-stanowego 2-symbolowego.

Model został zweryfikowany w laboratorium dla następującego automatu *M*.



Rys. 45. Automat *M* testowany laboratoryjnie.

Pary <stan, symbol> zostały zakodowane za pomocą łańcuchów DNA w następujący sposób (Rys. 46).

<stan,symbol>	*a*	*b*	*t*
S_0	5′-TC-3′	5′-AC-3′	5′-GC-3′
S_1	5′-AT-3′	5′-TA-3′	5′-GG-3′
S_2	5′-CA-3′	5′-CT-3′	5′-CG-3′

Rys. 46. Sekwencje zasad azotowych kodujących pary <stan, symbol>.

Model ten podobnie jak automat Shapiro jest automatem 2-symbolowym. Autorzy zaproponowali następujący sposób kodowania symboli (Rys. 47).

a	*b*	*t*
5′-CATC-3′ 3′-GTAG-5′	5′-CTAC-3′ 3′-GATG-5′	5′-CGGC-3′ 3′-GCCG-5′

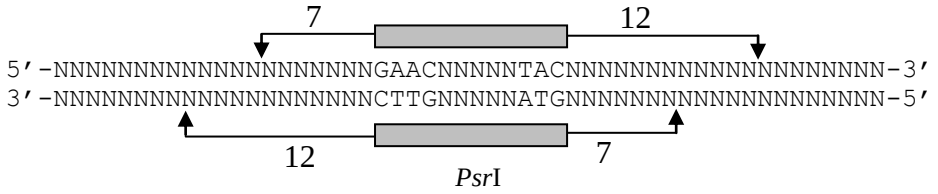
Rys. 47. Kodowanie symboli.

Zmniejszono liczbę nukleotydów użytych do kodowania symboli. Możliwe to było przez zastosowanie enzymu *Bse*MII, który pozostawia po wykonanej operacji cięcia dwa nukleotydy w lepkim końcu. Dodatkowo autorzy zakodowali za pomocą łańcuchów DNA 18 możliwych przejść dla modelu automatu 3-stanowego 2-symbolowego. Proces przetwarzania podobnie jak w automacie Shapiro polega na stopniowej analizie słowa wejściowego przez wykonywanie naprzemiennych operacji cięcia i łączenia.

W przedstawionych dwóch pracach [35], [40] nie podano możliwości dalszego rozszerzania modelu automatu Shapiro do większej liczby stanów. Autorzy nie omówili także ograniczeń, jakie ma koncepcja autonomicznego automatu skończonego zbudowanego z DNA i enzymów restrykcyjnych (automat Shapiro).

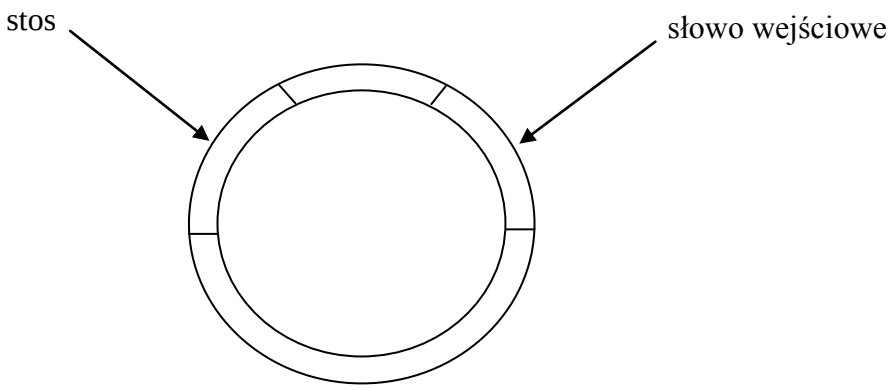
Inny model, oparty na idei Shapiro został zaprezentowany w [9]. Jest to model automatu ze stosem oraz automatu z dwoma stosami (ten ostatni jak wiadomo jest

równoważny maszynie Turinga). Nową myślą, która umożliwia konstrukcję takich automatów, było użycie kolistych łańcuchów DNA oraz zastosowanie enzymów restrykcyjnych trawiących specyficznie łańcuchy DNA np.: *Psr*I - w dwie strony jednocześnie (Rys. 48).



Rys. 48. Działanie enzymu *Psr*I.

Kolisty łańcuch DNA reprezentuje jednocześnie słowo wejściowe i stos (Rys. 49).



Rys. 49. Idea stosu i słowa wejściowego na kolistym DNA.

Po odpowiednim zakodowaniu symboli taśmy, stosu, stanów oraz ruchów (podobnie jak w przypadku automatu Shapiro) można oddać ruchy automatu ze stosem. Wadą tego podejścia było to, że w trakcie działania dany ruch musi przyłączyć się jednocześnie do dwóch lepkich końców rozciętego kolistego łańcucha DNA. Nie ma mechanizmu, który zapobiegałby przyłączeniu się jednego ruchu do jednego końca i drugiego ruchu do drugiego lepkiego końca, co powodowałoby złe działanie tego automatu.

Rozdział 4

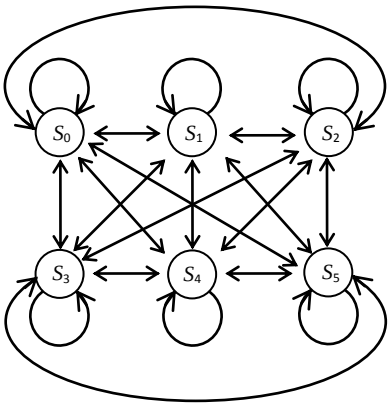
Rozszerzenie automatu Shapiro

W poprzednim rozdziale omówiony został model automatu 2-stanowego 2-symbolowego (automat Shapiro) [4], [5] oraz znane jego rozszerzenia do trzech stanów [35], [40]. Zespół Shapiro podaje, że automat oparty na enzymie *Fok*I może być rozszerzony maksymalnie do trzech stanów oraz kilkudziesięciu symboli akceptowanych. Autorzy jednocześnie zwracają uwagę, że odkrycie enzymów restrykcyjnych pozostawiających po cięciu dłuższe lepkie końce mogłoby zwiększyć złożoność wykonanych obliczeń.

W rozdziale tym przedstawiona zostanie nowa idea rozszerzania automatu Shapiro, która polega na zwiększeniu liczby działających enzymów w jednej mieszaninie (w jednej probówce) [19].

4.1. Sześciostanowy automat

Rozszerzenie modelu automatu Shapiro do 6-stanów możliwe jest przez zastosowanie dwóch enzymów restrykcyjnych *Bse*XI oraz *Eco*57I działających w jednej mieszaninie. Enzymy zostały tak dobrane, że działając na słowie wejściowym (tnąc słowo wejściowe) pozostawiają dwa rodzaje lepkich końców. Umożliwia to zakodowanie większej ilości przejść automatu (dokładnie 72 przejścia) i jest wystarczające dla modelu 6-stanowego 2-symbolowego (Rys. 50).



Rys. 50. Wszystkie możliwe przejścia dla 6-stanowego automatu.

Wszystkie krawędzie tego grafu posiadają etykietę: a, b. Ze względu na złożoność grafu etykiety te nie zostały umieszczone na rysunku 50.

4.1.1. Kodowanie informacji

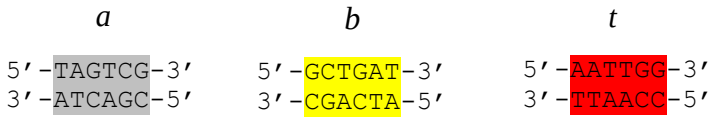
Podobnie jak w automacie Shapiro słowo wejściowe zaczyna się lepkim końcem. Wszystkie możliwe pary <stan, symbol> oraz ich kody w postaci łańcuchów DNA zostały przedstawione w poniższej tabeli (Rys. 51).

<stan,symbol>	a	b	t
S_0	5′-GTCG-3′	5′-TGAT-3′	5′-TTGG-3′
S_1	5′-AGTC-3′	5′-CTGA-3′	5′-ATTG-3′
S_2	5′-TAGT-3′	5′-GCTG-3′	5′-AATT-3′
S_3	3′-GC-5′	3′-TA-5′	3′-CC-5′
S_4	3′-AG-5′	3′-CT-5′	3′-AC-5′
S_5	3′-CA-5′	3′-AC-5′	3′-AA-5′

Rys. 51. Kodowanie par <stan, symbol>.

Wyróżniamy dwa rodzaje lepkich końców. Pierwszy rodzaj (na łańcuchu 5'→3') składa się z czterech nukleotydów i koduje stany S_0, S_1, S_2 oraz odpowiadające im symbole. Drugi rodzaj lepkich końców (na łańcuchu 3'→5') składa się z dwóch nukleotydów i koduje stany S_3, S_4, S_5 wraz z odpowiadającymi im symbolami.

Symbole a, b z alfabetu Σ w słowie wejściowym kodowane są sześcioma nukleotydami. Podobnie jak w automacie Shapiro wprowadzono trzeci symbol t oznaczający sekwencję terminalną. Poszczególne symbole kodowane są w następujący sposób (Rys. 52).

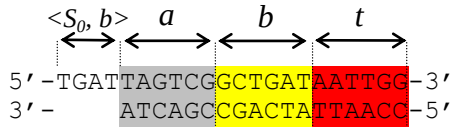


Rys. 52. Kodowanie symboli w modelu 6-stanowym.

Przykład 11

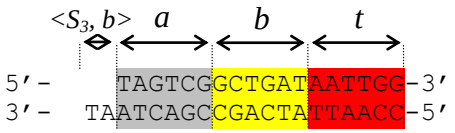
Słowo $x = bab$ w zależności od stanu początkowego kodujemy w następujący sposób. W przypadku, gdy stanem początkowym jest S_0 lub S_1 lub S_2, to lepki koniec

znajduje się na łańcuchu 5'→3' i składa się z czterech nukleotydów. Przykładowo dla pary $<S_0, b>$ (Rys. 53).



Rys. 53. Kodowanie słowa $x = bab$ gdy stanem początkowym jest S_0.

Dla stanów S_3, S_4, S_5 lepki koniec znajduje się na łańcuchu 3'→5' i składa się z dwóch nukleotydów. Na przykład dla stanu S_3 (Rys. 54).

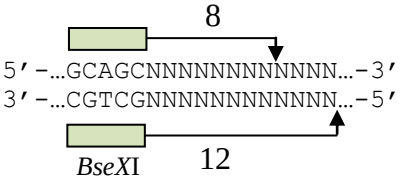


Rys. 54. Kodowanie słowa $x = bab$ gdy stanem początkowym jest S_3.

Kolejnym elementem budowy przedstawionego 6-stanowego automatu są ruchy (dokładnie 72) zakodowane łańcuchami DNA (Rys. 57). Zostały one tak zakodowane łańcuchami DNA, aby nie wchodziły ze sobą w konflikty.

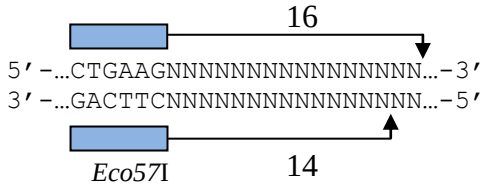
Symbol „N" oznacza jeden z 4 symboli: A, T, G, C. Przy wykonywaniu eksperymentu praktycznego ważne jest, aby symbole „N" nie tworzyły sekwencji rozpoznawanej przez enzym, gdyż doprowadzać może to dodatkowego cięcia łańcuchów DNA. Ze względu na większą trwałość wiązań wodorowych (trzy wiązania wodorowe) między guaniną a cytozyną powinno się stosować kombinację tych dwóch symboli w sekwencji kodowanej symbolami „N".

W modelu tym użyto dwa enzymy restrykcyjne *Bse*XI oraz *Eco*57I. Zastosowane enzymy zostały tak dobrane, aby po wykonaniu cięcia łańcucha DNA pozostawiały dwa rodzaje lepkich końców. Pierwszy enzym (o nazwie *Bse*XI) rozpoznaje sekwencję GCAGC (Rys. 55) i tnie łańcuch DNA po 8 nukleotydach w kierunku 5'→3' oraz po 12 nukleotydach w kierunku 3'→5'.



Rys. 55. Cięcie enzymem *Bse*XI .

Drugi enzym (o nazwie *Eco*57I) użyty w modelu rozpoznaje sekwencję CTGAAG i tnie łańcuch DNA po 16 nukleotydach w kierunku 5'→3' oraz po 14 nukleotydach w kierunku 3'→5'.



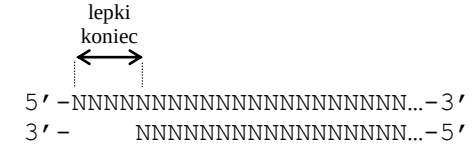
Rys. 56. Cięcie enzymem *Eco*57I .

PRZEJŚCIE	KODOWANIE DNA	PRZEJŚCIE	KODOWANIE DNA
T1: $S_0 \xrightarrow{a} S_0$	5′-GCAGCNN -3′ 3′-CGTCGNNCAGC-5′	**T19:** $S_3 \xrightarrow{a} S_0$	5′-GCAGCNNNNCG-3′ 3′-CGTCGNNNN -5′
T2: $S_0 \xrightarrow{a} S_1$	5′-GCAGCNNN -3′ 3′-CGTCGNNNCAGC-5′	**T20:** $S_1 \xrightarrow{a} S_3$	5′-CTGAAGNNNNN -3′ 3′-GACTTCNNNNNTCAG-5′
T3: $S_0 \xrightarrow{a} S_2$	5′-GCAGCNNNN -3′ 3′-CGTCGNNNNCAGC-5′	**T21:** $S_1 \xrightarrow{a} S_4$	5′-CTGAAGNNNNNN -3′ 3′-GACTTCNNNNNNTCAG-5′
T4: $S_0 \xrightarrow{b} S_0$	5′-GCAGCNN -3′ 3′-CGTCGNNACTA-5′	**T22:** $S_1 \xrightarrow{a} S_5$	5′-CTGAAGNNNNNNN -3′ 3′-GACTTCNNNNNNNTCAG-5′
T5: $S_0 \xrightarrow{b} S_1$	5′-GCAGCNNN -3′ 3′-CGTCGNNNACTA-5′	**T23:** $S_1 \xrightarrow{b} S_3$	5′-CTGAAGNNNNN -3′ 3′-GACTTCNNNNNGACT-5′
T6: $S_0 \xrightarrow{b} S_2$	5′-GCAGCNNNN -3′ 3′-CGTCGNNNNACTA-5′	**T24:** $S_1 \xrightarrow{b} S_4$	5′-CTGAAGNNNNNN -3′ 3′-GACTTCNNNNNNGACT-5′
T7: $S_3 \xrightarrow{a} S_3$	5′-CTGAAGNNNNNNNNCG-3′ 3′-GACTTCNNNNNNNN -5′	**T25:** $S_1 \xrightarrow{b} S_5$	5′-CTGAAGNNNNNNN -3′ 3′-GACTTCNNNNNNNGACT-5′
T8: $S_3 \xrightarrow{a} S_4$	5′-CTGAAGNNNNNNNNNCG-3′ 3′-GACTTCNNNNNNNNN -5′	**T26:** $S_3 \xrightarrow{a} S_1$	5′-GCAGCNNNNNCG-3′ 3′-CGTCGNNNNN -5′
T9: $S_3 \xrightarrow{a} S_5$	5′-CTGAAGNNNNNNNNNNCG-3′ 3′-GACTTCNNNNNNNNNN -5′	**T27:** $S_4 \xrightarrow{a} S_1$	5′-GCAGCNNNNTC-3′ 3′-CGTCGNNNN -5′
T10: $S_3 \xrightarrow{b} S_3$	5′-CTGAAGNNNNNNNNAT-3′ 3′-GACTTCNNNNNNNN -5′	**T28:** $S_5 \xrightarrow{a} S_1$	5′-GCAGCNNNGT-3′ 3′-CGTCGNNN -5′
T11: $S_3 \xrightarrow{b} S_4$	5′-CTGAAGNNNNNNNNNAT-3′ 3′-GACTTCNNNNNNNNN -5′	**T29:** $S_3 \xrightarrow{b} S_1$	5′-GCAGCNNNNNAT-3′ 3′-CGTCGNNNNN -5′
T12: $S_3 \xrightarrow{b} S_5$	5′-CTGAAGNNNNNNNNNNAT-3′ 3′-GACTTCNNNNNNNNNN -5′	**T30:** $S_4 \xrightarrow{b} S_1$	5′-GCAGCNNNNGA-3′ 3′-CGTCGNNNN -5′
T13: $S_0 \xrightarrow{a} S_3$	5′-CTGAAGNNNNNN -3′ 3′-GACTTCNNNNNNCAGC-5′	**T31:** $S_5 \xrightarrow{b} S_1$	5′-GCAGCNNNTG-3′ 3′-CGTCGNNN -5′
T14: $S_0 \xrightarrow{a} S_4$	5′-CTGAAGNNNNNNN -3′ 3′-GACTTCNNNNNNNCAGC-5′	**T32:** $S_2 \xrightarrow{a} S_2$	5′-GCAGCNN -3′ 3′-CGTCGNNATCA-5′
T15: $S_0 \xrightarrow{a} S_5$	5′-CTGAAGNNNNNNNN -3′ 3′-GACTTCNNNNNNNNCAGC-5′	**T33:** $S_2 \xrightarrow{a} S_1$	5′-GCAGCN -3′ 3′-CGTCGNATCA-5′
T16: $S_0 \xrightarrow{b} S_3$	5′-CTGAAGNNNNNN -3′ 3′-GACTTCNNNNNNACTA-5′	**T34:** $S_2 \xrightarrow{a} S_0$	5′-GCAGC -3′ 3′-CGTCGATCA-5′
T17: $S_0 \xrightarrow{b} S_4$	5′-CTGAAGNNNNNNN -3′ 3′-GACTTCNNNNNNNACTA-5′	**T35:** $S_2 \xrightarrow{b} S_2$	5′-GCAGCNN -3′ 3′-CGTCGNNCGAC-5′
T18: $S_0 \xrightarrow{b} S_5$	5′-CTGAAGNNNNNNNN -3′ 3′-GACTTCNNNNNNNNACTA-5′	**T36:** $S_2 \xrightarrow{b} S_1$	5′-GCAGCN -3′ 3′-CGTCGNCGAC-5′

T37: $S_2 \xrightarrow{b} S_0$	5′-GCAGC -3′ 3′-CGTCGCGAC-5′	**T55:** $S_4 \xrightarrow{b} S_5$	5′-CTGAAGNNNNNNNNNGA-3′ 3′-GACTTCNNNNNNNNN -5′
T38: $S_5 \xrightarrow{a} S_5$	5′-CTGAAGNNNNNNNNGT-3′ 3′-GACTTCNNNNNNNN -5′	**T56:** $S_5 \xrightarrow{a} S_4$	5′-CTGAAGNNNNNNNGT-3′ 3′-GACTTCNNNNNNN -5′
T39: $S_4 \xrightarrow{a} S_0$	5′-GCAGCNNNTC-3′ 3′-CGTCGNNN -5′	**T57:** $S_5 \xrightarrow{a} S_3$	5′-CTGAAGNNNNNNGT-3′ 3′-GACTTCNNNNNN -5′
T40: $S_5 \xrightarrow{a} S_0$	5′-GCAGCNNGT-3′ 3′-CGTCGNN -5′	**T58:** $S_5 \xrightarrow{b} S_5$	5′-CTGAAGNNNNNNNNTG-3′ 3′-GACTTCNNNNNNNN -5′
T41: $S_3 \xrightarrow{a} S_0$	5′-GCAGCNNNNAT-3′ 3′-CGTCGNNNN -5′	**T59:** $S_5 \xrightarrow{b} S_4$	5′-CTGAAGNNNNNNNTG-3′ 3′-GACTTCNNNNNNN -5′
T42: $S_4 \xrightarrow{b} S_0$	5′-GCAGCNNNGA-3′ 3′-CGTCGNNN -5′	**T60:** $S_5 \xrightarrow{b} S_3$	5′-CTGAAGNNNNNNTG-3′ 3′-GACTTCNNNNNN -5′
T43: $S_5 \xrightarrow{b} S_0$	5′-GCAGCNNTG-3′ 3′-CGTCGNN -5′	**T61:** $S_2 \xrightarrow{a} S_3$	5′-CTGAAGNNNN -3′ 3′-GACTTCNNNNATCA-5′
T44: $S_1 \xrightarrow{a} S_1$	5′-GCAGCNN -3′ 3′-CGTCGNNTCAG-5′	**T62:** $S_2 \xrightarrow{a} S_4$	5′-CTGAAGNNNNN -3′ 3′-GACTTCNNNNNATCA-5′
T45: $S_1 \xrightarrow{a} S_0$	5′-GCAGCN -3′ 3′-CGTCGNTCAG-5′	**T63:** $S_2 \xrightarrow{a} S_5$	5′-CTGAAGNNNNNN -3′ 3′-GACTTCNNNNNNATCA-5′
T46: $S_1 \xrightarrow{a} S_2$	5′-GCAGCNNN -3′ 3′-CGTCGNNNTCAG-5′	**T64:** $S_2 \xrightarrow{b} S_3$	5′-CTGAAGNNNN -3′ 3′-GACTTCNNNNCGAC-5′
T47: $S_1 \xrightarrow{b} S_1$	5′-GCAGCNN -3′ 3′-CGTCGNNGACT-5′	**T65:** $S_2 \xrightarrow{b} S_4$	5′-CTGAAGNNNNN -3′ 3′-GACTTCNNNNNCGAC-5′
T48: $S_1 \xrightarrow{b} S_0$	5′-GCAGCN -3′ 3′-CGTCGNGACT-5′	**T66:** $S_2 \xrightarrow{b} S_5$	5′-CTGAAGNNNNNN -3′ 3′-GACTTCNNNNNNCGAC-5′
T49: $S_1 \xrightarrow{b} S_2$	5′-GCAGCNNN -3′ 3′-CGTCGNNNGACT-5′	**T67:** $S_3 \xrightarrow{a} S_2$	5′-GCAGCNNNNNNCG-3′ 3′-CGTCGNNNNNN -5′
T50: $S_4 \xrightarrow{a} S_4$	5′-CTGAAGNNNNNNNNTC-3′ 3′-GACTTCNNNNNNNN -5′	**T68:** $S_4 \xrightarrow{a} S_2$	5′-GCAGCNNNNNTC-3′ 3′-CGTCGNNNNN -5′
T51: $S_4 \xrightarrow{a} S_3$	5′-CTGAAGNNNNNNNTC-3′ 3′-GACTTCNNNNNNN -5′	**T69:** $S_5 \xrightarrow{a} S_2$	5′-GCAGCNNNNGT-3′ 3′-CGTCGNNNN -5′
T52: $S_4 \xrightarrow{a} S_5$	5′-CTGAAGNNNNNNNNNTC-3′ 3′-GACTTCNNNNNNNNN -5′	**T70:** $S_3 \xrightarrow{b} S_2$	5′-GCAGCNNNNNNAT-3′ 3′-CGTCGNNNNNN -5′
T53: $S_4 \xrightarrow{b} S_4$	5′-CTGAAGNNNNNNNNGA-3′ 3′-GACTTCNNNNNNNN -5′	**T71:** $S_4 \xrightarrow{b} S_2$	5′-GCAGCNNNNNGA-3′ 3′-CGTCGNNNNN -5′
T54: $S_4 \xrightarrow{b} S_3$	5′-CTGAAGNNNNNNNGA-3′ 3′-GACTTCNNNNNNN -5′	**T72:** $S_5 \xrightarrow{b} S_2$	5′-GCAGCNNNNTG-3′ 3′-CGTCGNNNN -5′

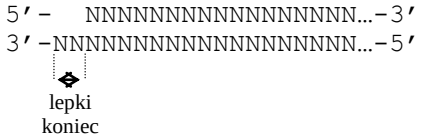
Rys. 57. Kodowanie przejść automatu 6-stanowego.

Wyróżniamy dwa rodzaje lepkich końców pozostałych po cięciu łańcuchów DNA enzymami *Bse*XI oraz *Eco*57I. Pierwszy lepki koniec (działanie enzymem *Bse*XI) zawiera cztery symbole i znajduje się na łańcuchu DNA w kierunku 5'→3'.



Rys. 58. Lepki koniec dla enzymu *Bse*XI .

Drugi lepki koniec, powstaje w wyniku działania enzymu *Eco*57I, zawiera 2 symbole i znajduje się na nici 3'→5'.



Rys. 59. Lepki koniec dla enzymu *Eco*57I.

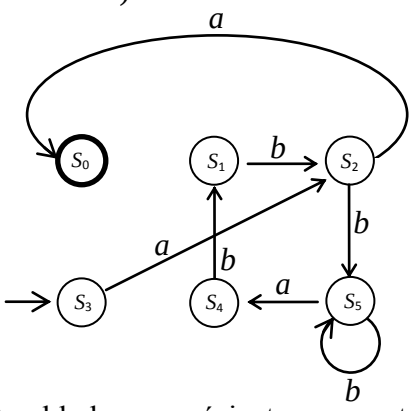
Użycie dwóch enzymów pozostawiających dwa rodzaje lepkich końców zwiększyło liczbę możliwych kombinacji do zakodowania symboli w lepkim końcu. Umożliwiło to zakodowanie 6 stanów i 2 symboli.

4.1.2. Przetwarzanie informacji

Proces przetwarzania informacji w 6-stanowym automacie jest zbliżony do działania automatu Shapiro. Pierwszym etapem jest wybór automatu, który uzyskuje się przez wybór odpowiednich przejść wybranych z 72 możliwych dla 6-stanowego 2-symbolowego automatu oraz umieszczenie odpowiadających im kodów w postaci łańcuchów DNA w probówce. Kolejnym etapem jest umieszczenie w probówce słowa wejściowego $x \in \{a,b\}^*$ oraz enzymów (*Bse*XI, *Eco*57I). Stopniowa analiza słowa wejściowego x przebiega przez naprzemienne cięcie i łączenie łańcuchów DNA realizowane enzymami restrykcyjnymi (cięcie) oraz enzymem o nazwie *ligaza* (łączenie). Proces cięcia i łączenia przebiega automatycznie i autonomicznie, aż do całkowitego strawienia łańcuchów DNA i pojawienia się w roztworze określonej sekwencji nukleotydów (sekwencji terminalnej *t*) - oznacza to zakończenie działania automatu.

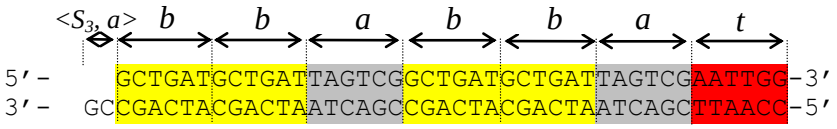
Przykład 12

Rozważmy 6-stanowy automat *M* przedstawiony na poniższym rysunku oraz słowo $x = abbabba$ (akceptowane przez ten automat).



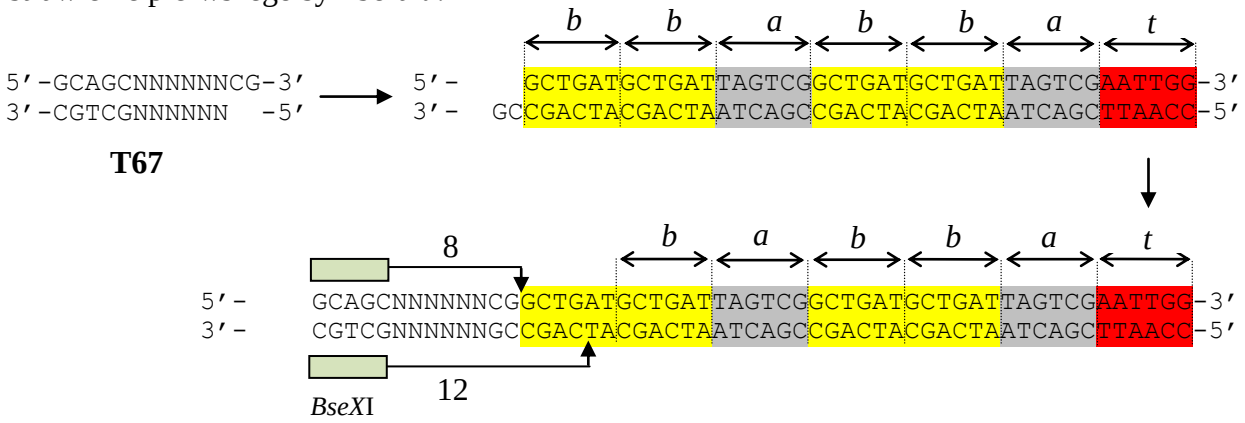
Rys. 60. Przykładowy sześciostanowy automat.

Słowo $x = abbabba$ zostało zakodowane w następujący sposób



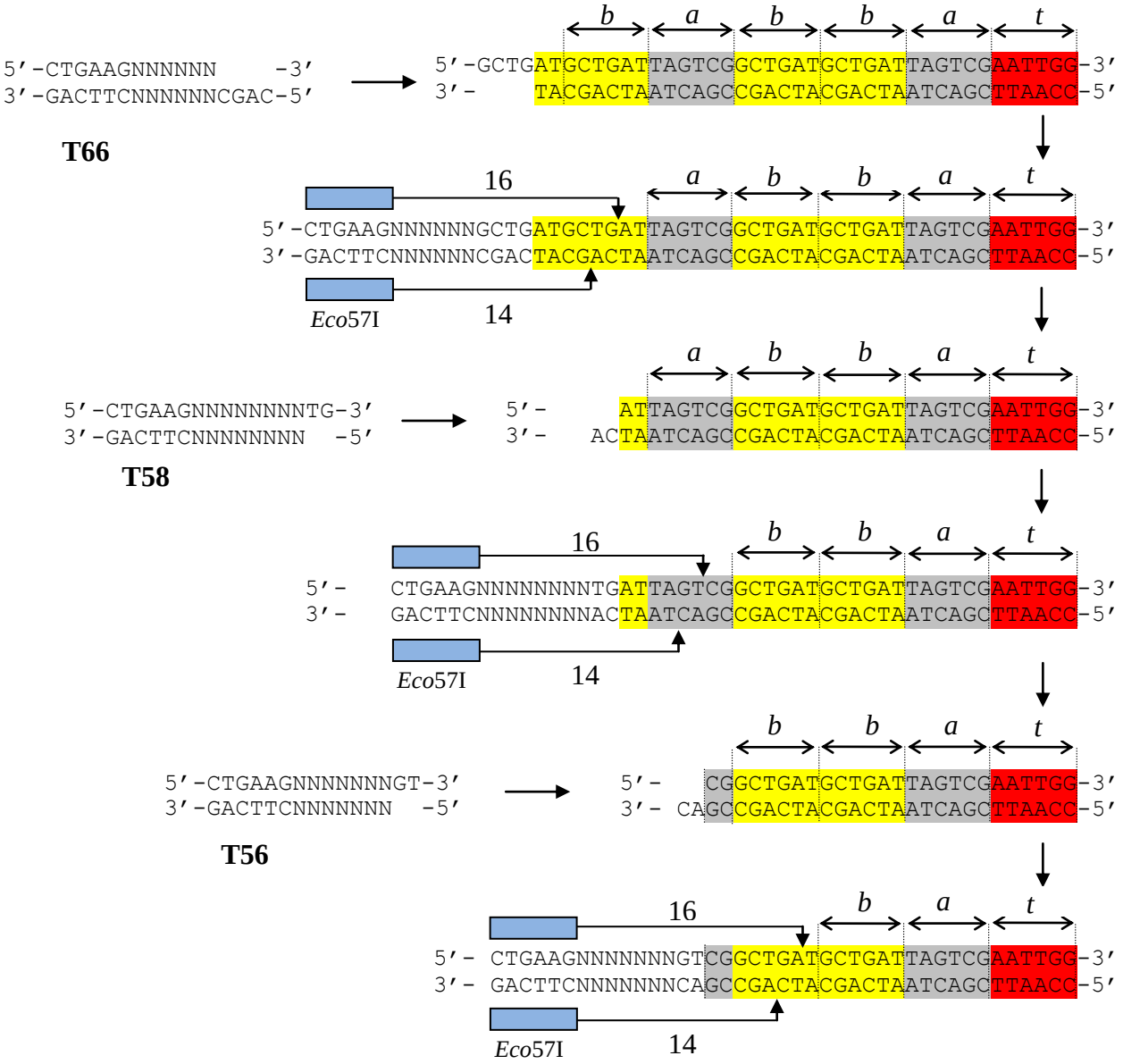
Rys. 61. Kodowanie słowa wejściowego $x = abbabba$.

Dodatkowo zakładamy, że stan początkowy znajduje się w S_3, a więc wykorzystujemy lepki koniec zbudowany z dwóch nukleotydów w kierunku 3'→5'. Następnie w probówce umieszczamy łańcuchy DNA kodujące ruchy (krawędzie) automatu *M*. Są to odpowiednio T30, T34, T49, T56, T58, T66, T67. Po wprowadzeniu do probówki wszystkich elementów automat rozpocznie działanie. Lepkie końce słowa oraz przejścia T67 są komplementarne, więc połączą się ze sobą. Sekwencja GCAGC jest rozpoznawana przez enzym *Bse*XI, który wykona automatyczne cięcie słowa wejściowego i tym samym strawienie pierwszego symbolu *a*.



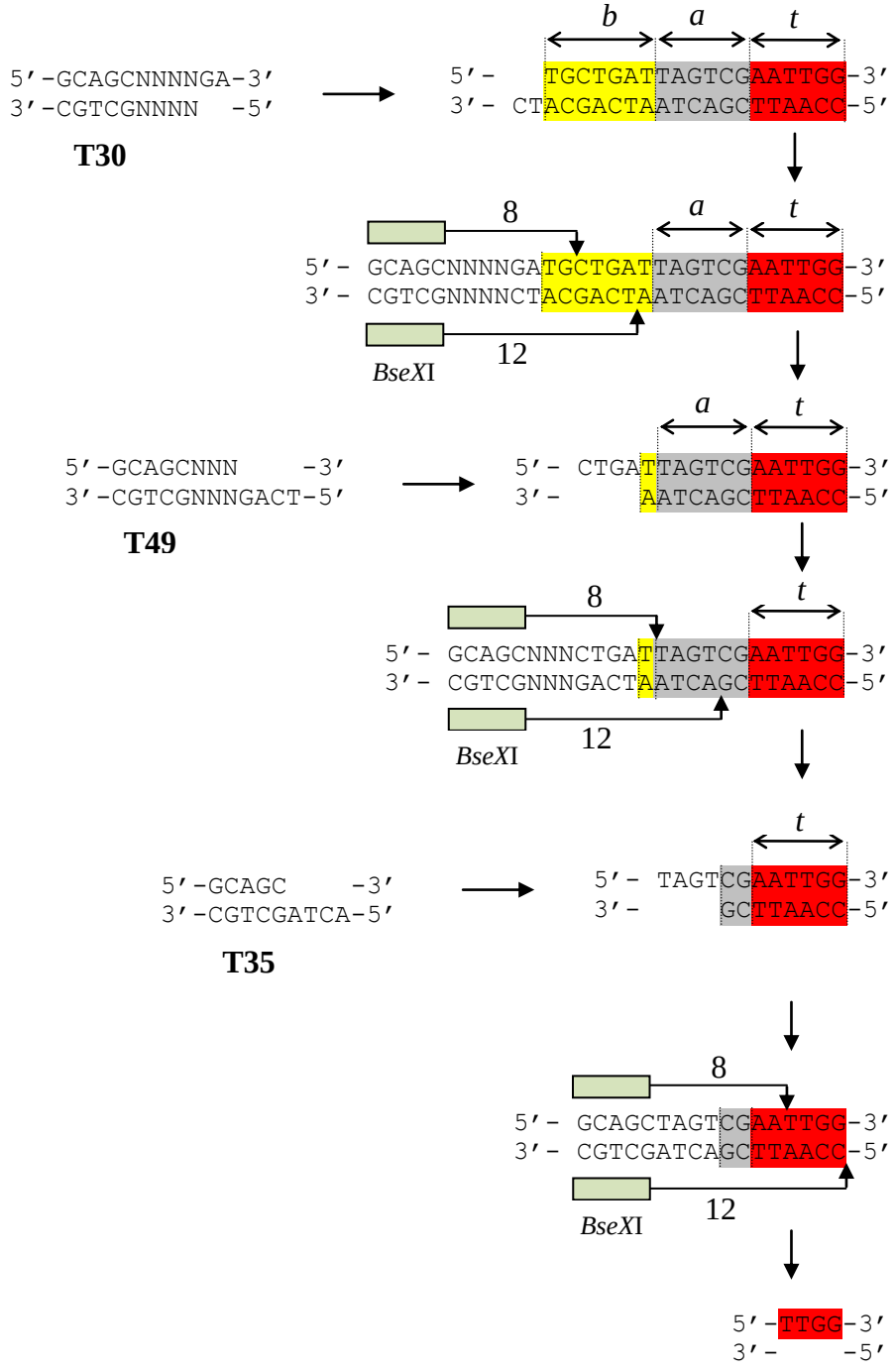
Rys. 62. Pierwsze cięcie enzymem *BseXI*.

W wyniku działania enzymu powstaje (na słowie *x*) lepki koniec GCTG w kierunku 3'→5' komplementarny do lepkiego końca ruchu T66. Kolejne trzy cięcia wykonywane są enzymem *Eco*57I i tym samym trawione zostają kolejne trzy symbole słowa *x*.



Rys. 63. Trzykrotne cięcie enzymem *Eco*57I.

Kolejne trzy cięcia wykonywane są enzymem *Bse*XI.



Rys. 64. Trzykrotne cięcie enzymem *Bse*XI.

Ostatecznym efektem działania enzymów (dokładnie siedem cięć enzymami *Bse*XI oraz *Eco*57I) jest otrzymanie łańcucha terminalnego TTGG w kierunku 3'→5', który koduje parę $<S_0, t>$. Ponieważ S_0 jest stanem końcowym więc słowo jest akceptowane.

4.2. Analiza możliwości rozszerzania automatu Shapiro

W poprzednim paragrafie omówiliśmy rozszerzenie automatu Shapiro do 6 stanów. W pracach [35], [40] podane zostały inne rozszerzenia automatu Shapiro. W tym paragrafie przedstawimy teoretyczne rozważania dotyczące możliwości rozszerzeń automatu Shapiro do dowolnej ilości stanów i dowolnej ilości symboli. Podamy arytmetyczne warunki, kiedy jest to możliwe. Ponieważ rozważania te są teoretyczne, więc przyjmujemy również dowolną ilość symboli, za pomocą których kodujemy symbole alfabetu Σ (zamiast przyjętych w komputerach biomolekularnych 4 symboli A, T, G, C).

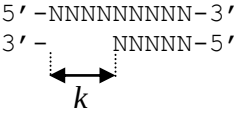
Załóżmy, że chcemy skonstruować metodą Shapiro niedeterministyczny automat skończony M o p stanach i r symbolach. Zatem $Q=\{s_1,...,s_p\}$, $\Sigma=\{a_1,...,a_r\}$. W myśl metody Shapiro symbole z Σ i pary z $Q\times\Sigma$ kodujemy za pomocą odpowiedniego układu zasad (czyli symboli A, T, G, C). W naszych teoretycznych rozważaniach przyjmujemy, że dany jest dodatkowy alfabet $A=\{A_1,...,A_q\}$ złożony z q symboli, za pomocą których kodujemy symbole z Σ i pary $Q\times\Sigma$ (czyli A zamienia alfabet {A, T, G, C}).

Załóżmy, że symbole z Σ kodujemy za pomocą łańcuchów (słów) o długości n utworzonych z symboli A.

gdzie litera „N” oznacza symbol z A.

Oczywiście uwzględniamy tutaj pewną komplementarność (czyli pewną relację symetryczną w zbiorze A). Dla uproszczenia zakładamy, że jedna nić jednoznacznie wyznacza drugą komplementarną.

Ustalmy, że lepki koniec (powstały w wyniku cięcia enzymem restrykcyjnym) jest k wyrazowy. Zatem kod dowolnej pary <stan, symbol> ma postać.

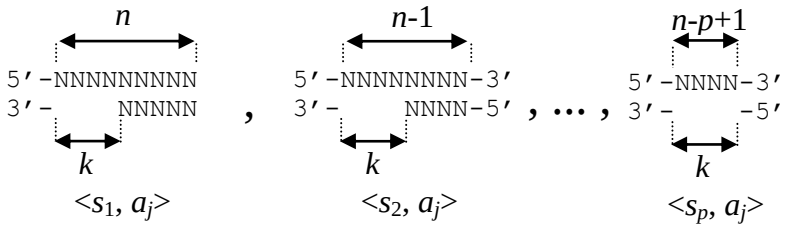


Oczywiście

$$1\le k\le n.$$

Problemem jest ustalenie zależności między liczbami p, r, q, n, k, by była możliwa konstrukcja dowolnego NAS o p stanach, r symbolach i q symbolach kodujących.

Ponieważ dla ustalonego symbolu $a_j \in \Sigma$ mamy p par $<s_1, a_j>,\ldots,<s_p, a_j>$, więc kod symbolu a_j musi mieć p różnych lepkich końców.



Zatem otrzymujemy pierwszą zależność

$$k = n - p + 1,$$

lub równoważnie

$$p = n - k + 1.$$

Oczywiście wystarczy zależność $p \le n - k + 1$, ale ze względu na minimalność konstrukcji przyjmujemy $p = n - k + 1$. Ponieważ każda para $<s_i, a_j>$, $i=1,\ldots,p$, $j=1,\ldots,r$ musi mieć swój unikalny kod (czyli lepki koniec), więc kody symboli $a_1,\ldots,a_r$ muszą być tak wybrane, by wszystkie lepkie końce (k wyrazowe) były różne.

Zatem aby możliwa była konstrukcja dowolnego NAS o p stanach, r symbolach w alfabecie i q symbolach kodujących, muszą istnieć liczby $n,k \in N$ takie, że:

1. $1 \le k \le n$,
2. $p = n - k + 1$,
3. istnieje układ r ciągów n wyrazowych utworzonych z q symboli kodujących o tej właściwości, że wszystkie lepkie końce k wyrazowe tych ciągów są różne.

Dla danych $r,n,q,k \in N$, oznaczmy przez $l(r,n,q,k)$ ilość układów r ciągów o własnościach 1, 2, 3. Zatem NAS automat M o p stanach i r symbolach można skonstruować metodą Shapiro, kodując symbole ciągami n wyrazowymi (utworzonymi z q symboli) i używając enzymu restrykcyjnego pozostawiającego k wyrazowy lepki koniec, gdy:

1. $p = n - k + 1$ (wystarczy $p \le n - k + 1$),
2. $l(r,n,q,k) > 0$.

Zatem należy rozważyć kombinatoryczny problem:

Problem. Dla jakich liczb $r,n,q,k \in N$, $1 \le k \le n$

$$l(r,n,q,k) > 0.$$

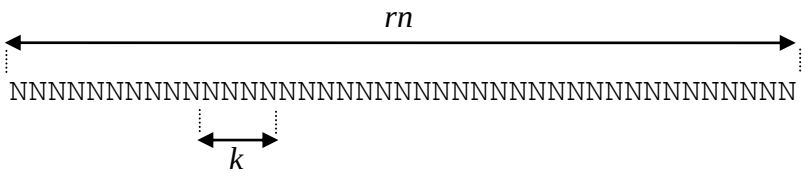
Jeżeli dla danych r,n,q,k mamy $l(r,n,q,k) > 0$ to wtedy ilość stanów

$$p = n - k + 1.$$

Dokładne wyznaczenie $l(r,n,q,k)$ jest problemem trudnym. Podamy tylko oszacowanie $l(r,n,q,k)$ od dołu przez wyrażenie arytmetyczne $l^*(r,n,q,k)$ zależne w prosty sposób od r,n,q,k. Dla r,n,q,k dla których $l^*(r,n,q,k) > 0$ oczywiście będziemy mieli

$$l(r,n,q,k) \geq l^*(r,n,q,k) > 0.$$

Zatem zamiast problemu wyznaczenia $l(r,n,q,k)$ rozważymy trochę inny problem. Obliczymy (dokładniej tylko oszacujemy) ilość ciągów rn wyrazowych utworzonych z q symboli dla których wszystkie k wyrazowe kolejne podciągi (możliwe lepkie końce) są różne.



Dla danych r,n,q,k oznaczmy ilość takich ciągów przez $l^*(r,n,q,k)$. Oczywiście

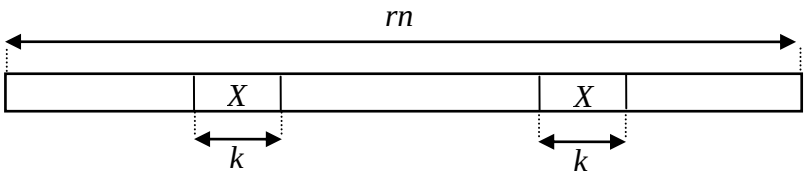
$$l(r,n,q,k) \geq l^*(r,n,q,k),$$

gdyż każdy ciąg rn wyrazowy utworzony z q symboli dla których wszystkie kolejne k podciągi są różne po podzieleniu na r różnych części daje nam układ r ciągów n wyrazowych spełniających warunek 3. Zatem wystarczy wyznaczyć wzór na $l^*(r,n,q,k)$. Zanalizujmy ten problem.

Wszystkich ciągów rn wyrazowych o wyrazach ze zbioru q elementowego jest

$$q^{rn}.$$

Z tego zbioru musimy usunąć ciągi, w których przynajmniej 2 podciągi kolejnych k wyrazów są identyczne



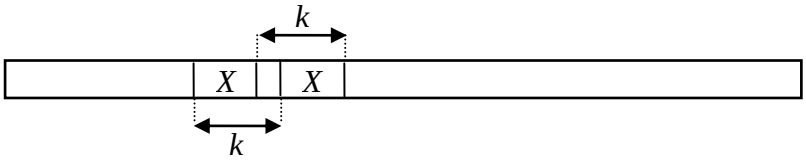
Obliczymy liczbę tych ostatnich:

1. jeżeli 2 podciągi w ustalonych miejscach kolejnych k wyrazów są identyczne i rozłączne (tak jak na rysunku), to ich liczbę daje znany wzór z kombinatoryki:

$$q^k \, q^{rn-2k} = q^{rn-k}$$

(tworzymy dowolny k wyrazowy ciąg ze zbioru q elementowego – to daje liczbę q^k , a następnie pozostałe wyrazy ciągu wybieramy dowolnie - to daje liczbę q^{rn-2k}),

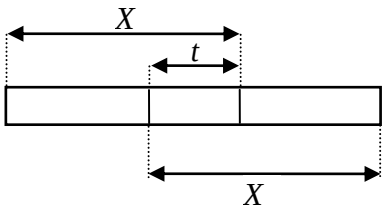
2. jeżeli 2 podciągi w ustalonych miejscach kolejnych k wyrazów są identyczne i „zachodzą na siebie”



to ich ilość jest też równa

$$q^k\, q^{rn-2k} = q^{rn-k}.$$

Istotnie, jeżeli te podciągi mają „wspólną część” t elementową



to dowolnych elementów w X jest tylko k-t (pozostałe t wyrazów jest jednocześnie wyznaczone przez identyczność tych podciągów). Zatem liczba takich słów X jest równa

$$q^{k-t}.$$

Dowolnych elementów poza tymi dwoma podciągami jest rn-k-(k-t) =rn-2k+t, a więc ich liczba jest równa

$$q^{rn-2k+t}.$$

Łącznie ciągów rn wyrazowych o powyższej własności jest

$$q^{k-t}\, q^{rn-2k+t} = q^{rn-k}.$$

Ponieważ możliwych rozmieszczeń 2 podciągów k wyrazowych w ciągu rn wyrazowym jest równa

$$(rn-k)+(rn-k-1)+\ldots+1=\frac{(rn-k+1)(rn-k)}{2},$$

więc liczba możliwych rn ciągów, w których dwa k wyrazowe podciągi powtarzają się jest

$$\leq \frac{(rn-k+1)(rn-k)}{2} q^{rn-k}.$$

Otrzymujemy tylko nierówność, gdyż niektóre z tych ciągów są liczone wielokrotnie np.

	X		X		X	

Zatem liczba ciągów rn elementowych, w których żadne 2 podciągi k wyrazowe nie powtarzają się (czyli $l^*(r,n,q,k)$) spełnia nierówność

$$l^*(r,n,q,k) \geq q^{rn} - q^{rn-k}\frac{(rn-k+1)(rn-k)}{2},$$

$$l^*(r,n,q,k) \geq q^{rn-k}\left(q^k - \frac{(rn-k)(rn-k+1)}{2}\right),$$

stąd

$$l(r,n,q,k) \geq q^{rn-k}\left(q^k - \frac{(rn-k)(rn-k+1)}{2}\right).$$

Stąd otrzymujemy częściową odpowiedź na problem. Jeżeli

$$q^k > \frac{(rn-k)(rn-k+1)}{2},$$

to

$$l(r,n,q,k) > 0.$$

Odnosząc to do problemu istnienia automatu Shapiro otrzymujemy.

Twierdzenie 1. Możliwa jest konstrukcja automatu Shapiro o p stanach i r symbolach, w których kodujemy symbole ciągami n elementowymi o wyrazach ze zbioru q elementowego oraz używamy jednego enzymu restrykcyjnego pozostawiającego po cięciu k wyrazowy lepki koniec ($1 \leq k \leq n$), gdy

1. $p = n-k+1$ (wystarczy $p \leq n-k+1$),
2. $q^k > \frac{(rn-k)(rn-k+1)}{2}$.

W szczególnym przypadku, gdy q=4 (jest tak, gdy używamy symboli A, C, T, G do kodowania) to otrzymujemy

Wniosek. Możliwa jest konstrukcja automatu Shapiro o p stanach i r symbolach w którym kodujemy symbole łańcuchami DNA o długości n i używamy jednego enzymu restrykcyjnego pozostawiającego po cięciu k wyrazowy lepki koniec ($1 \leq k \leq n$), gdy

1. $p = n - k + 1$ (wystarczy $p \le n - k + 1$) ,

2. $4^k > \frac{(rn-k)(rn-k+1)}{2}$.

Przykład 13.

Weźmy:

1. liczbę stanów p=4,
2. liczbę symboli r=2,
3. długość kodów symboli n=6,
4. długość lepkich końców k=3.

Wtedy

$$p = n - k + 1,$$

i

$$4^3 > \frac{9 \cdot 10}{2} .$$

Zatem przy tych ustaleniach jest możliwy automat Shapiro.

Przykład 14.

Weźmy p=5, r=2, n=6, k=2. Wtedy

$$p = n - k + 1,$$

ale nie zachodzi nierówność

$$4^2 > \frac{10 \cdot 11}{2} .$$

W tym przypadku nasze twierdzenie nie daje odpowiedzi.

Przypadek użycia większej liczby enzymów.

Rozważmy teraz przypadek, gdy zastosujemy większą ilość enzymów. Dla ułatwienia rozważań rozpatrzmy najpierw przypadek 2 enzymów pozostawiających lepkie końce różnej długości. Oznaczmy te długości przez k_1 i k_2. Zatem po cięciu otrzymujemy

Ponieważ $k_1 \neq k_2$, więc ich lepkie końce nie spowodują konfliktu. Z rozważań dotyczących przypadku jednego enzymu wynika, że ilość stanów automatu przy cięciu pierwszego enzymu może być równa

$$p_1 = n - k_1 + 1.$$

Podobnie dla drugiego enzymu

$$p_2 = n - k_2 + 1.$$

Zatem łączna (maksymalna) ilość stanów, przy użyciu 2 enzymów, jest równa

$$p = 2n - (k_1 + k_2) + 2.$$

Twierdzenie 2. Możliwa jest konstrukcja automatu Shapiro o p stanach i r symbolach, w którym kodujemy symbole ciągami n elementowymi o wyrazach ze zbioru q elementowego oraz używamy 2 enzymów restrykcyjnych pozostawiających po cięciu lepkie końce k_1 i odpowiednie k_2, gdy:

1. $p = 2n - (k_1 + k_2) + 2$ (wystarczy $p \leq n\text{-}k+1$),
2. $l(r, n, q, k_1) > 0$,
3. $l(r, n, q, k_2) > 0$.

Gdy zastosujemy twierdzenie 1 to otrzymamy bardziej efektywne kryteria istnienia takich automatów.

Twierdzenie 3. Możliwa jest konstrukcja automatu Shapiro o p stanach i r symbolach, w którym kodujemy symbole ciągami n elementowymi o wyrazach ze zbioru q elementowego oraz używamy 2 enzymów restrykcyjnych pozostawiających po cięciu lepkie końce k_1 i odpowiednie k_2, gdy:

1. $p = 2n - (k_1 + k_2) + 2$ (wystarczy $p \leq 2n - (k_1 + k_2) + 2$),
2. $q^{k_1} > \dfrac{(rn - k_1)(rn - k_1 + 1)}{2}$,
3. $q^{k_2} > \dfrac{(rn - k_2)(rn - k_2 + 1)}{2}$.

Przykład 15.

Gdy n=6 i k=4 to ilość stanów $p = n - k + 1 = 3$ (lub mniej).

Gdy n=6 i k_1=3, k_2=4, to ilość stanów $p = 2n - (k_1 + k_2) + 2 = 12 - 7 + 2 = 7$.

Gdy n=4 i k_1=2, k_2=3, to ilość stanów $p = 8 - 5 + 2 = 5$.

W przypadku, gdy użyjemy większą liczbę enzymów to rozumując analogicznie jak w przypadku użycia 2 enzymów otrzymamy następujące twierdzenia.

Twierdzenie 4. Możliwa jest konstrukcja automatu Shapiro o p stanach i r symbolach, w którym kodujemy symbole ciągami n elementowymi o wyrazach ze zbioru q elementowego oraz używamy j enzymów restrykcyjnych pozostawiających po cięciu lepkie końce $k_1,\ldots,k_j$, gdy:

1. $p = jn-(k_1+\ldots+k_j)+j$,
2. $l(r,n,q,k_1)>0$,

……,

j+1. $l(r,n,q,k_j)>0$.

oraz bardziej efektywne kryterium.

Twierdzenie 5. Możliwa jest konstrukcja automatu Shapiro o p stanach i r symbolach, w którym kodujemy symbole ciągami n elementowymi o wyrazach ze zbioru q elementowego oraz używamy j enzymów restrykcyjnych pozostawiających po cięciu lepkie końce $k_1,\ldots,k_j$, gdy:

1. $p = jn-(k_1+\ldots+k_j)+j$,
2. $q^{k_1} > \frac{(rn-k_1)(rn-k_1+1)}{2}$

……,

j+1. $q^{k_j} > \frac{(rn-k_j)(rn-k_j+1)}{2}$.

Rozdział 5

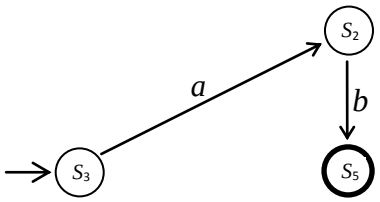
Praktyczna implementacja laboratoryjna

W konstrukcji teoretycznej rozszerzenia automatu Shapiro do większej liczby stanów, podanej w poprzednim rozdziale, zastosowano dwa enzymy restrykcyjne: *Bse*XI, *Eco*57I. Zostały one użyte ze względu na odpowiedni sposób działania. Enzymy te pozostawiają lepkie końce różnej długości oraz miejsca cięcia są w dużej odległości od sekwencji rozpoznawanej. Z praktycznego (laboratoryjnego) punktu widzenia okazało się jednak, że lepszym rozwiązaniem jest użycie innych enzymów restrykcyjnych, które działają w dokładnie taki sam sposób (te same miejsca rozpoznawane oraz w wyniku ich działania powstają takie same lepkie końce). Przyczyną tej zmiany jest fakt, że nowe enzymy: *Acu*I oraz *Bbv*I mają takie same warunki środowiska reakcyjnego (temperatura, bufor reakcyjny) w przeciwieństwie do teoretycznie zaplanowanych: *Bse*XI, *Eco*57I z których każdy działa w innych warunkach środowiska reakcyjnego.

W rozdziale tym przedstawione zostaną wyniki eksperymentu mającego na celu laboratoryjne sprawdzenie działania modelu 6-stanowego opracowanego w rozdziale 4. Doświadczenie zostało wykonane w Katedrze Genetyki Molekularnej Uniwersytetu Łódzkiego.

1. Automat badany w doświadczeniu

W doświadczeniu testowany był następujący automat działający, z zastosowaniem dwóch enzymów restrykcyjnych (*Acu*I, *Bbv*I), autonomicznie w jednej mieszaninie (Rys.65).

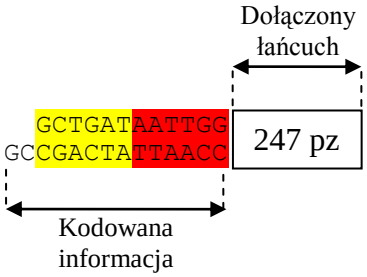


Rys. 65. Automat *M* testowany w doświadczeniu.

Wybrane zostały przejścia: T66 oraz T67 (w postaci łańcuchów DNA) ze zbioru 72 przejść (Rys. 57) kodowanych łańcuchami DNA i reprezentujących wszystkie możliwe przejścia dla automatu 6-stanowego 2-symbolowego. Automat *M* został tak dobrany, aby do pozytywnego zakończenia działania automatu na słowie $x=ab$ konieczne było trawienie

łańcucha DNA (reprezentującego słowo wejściowe) dwoma enzymami restrykcyjnymi: *Acu*I oraz *Bbv*I. Słowo *x=ab* jest akceptowane przez wybrany automat *M*.

Ze względu na specyfikę metod genetyki molekularnej do każdego fragmentu DNA kodującego informację (słowo wejściowe, przejście T66, przejście T67, sekwencja terminalna) dołączone zostały dodatkowe łańcuchy DNA o określonych długościach. Do słowa *x* dołączono łańcuch długość 247 pz, do przejścia T66 długości 77 pz, do przejścia T67 długości 75 pz i do łańcucha terminalnego długości 332 pz. Na poniższym rysunku przedstawiono łańcuch DNA kodujący słowo wejściowe *x=ab* (Rys. 66).



Rys. 66. Przykład DNA z dodatkowym łańcuchem długości 247 pz.

2. Schemat ideowy doświadczenia

Doświadczenie składało się z dwóch etapów. Pierwszy etap eksperymentu polegał na konstrukcji dwuniciowych fragmentów DNA odpowiadających słowu wejściowemu, wybranym przejściom, oraz sekwencji terminalnej. Z przygotowanych wcześniej wektorów plazmidowych wycięto za pomocą enzymów restrykcyjnych fragmenty DNA zawierające odpowiednie sekwencje wykorzystywane do konstrukcji automatu. Dopiero po przygotowaniu wszystkich elementów automatu w postaci łańcuchów DNA możliwe było wykonanie właściwego eksperymentu polegającego na weryfikacji działania automatu.

W drugim etapie sprawdzono eksperymentalnie działanie automatu opracowanego teoretycznie w rozdziale 4. Głównym zagadnieniem było sprawdzenie laboratoryjne możliwości naprzemiennego cięcia dwoma enzymami restrykcyjnymi słowa wejściowego *x=ab* wykonane autonomicznie w jednej mieszaninie. Działanie automatu badano umieszczając w probówce:

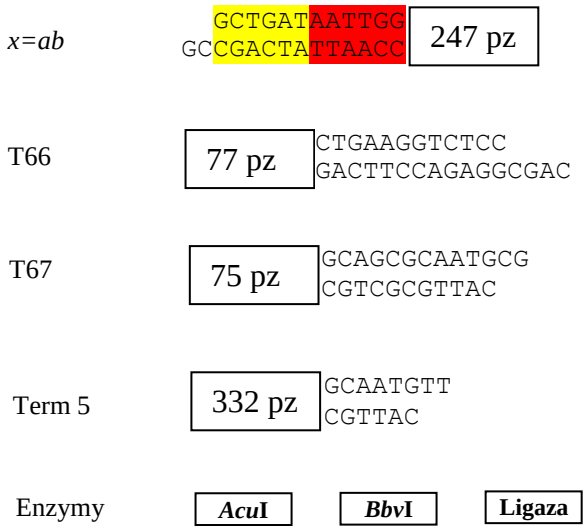
1. słowo *ab* w postaci łańcuchów DNA,
2. przejście T66 w postaci łańcuchów DNA,
3. przejście T67 w postaci łańcuchów DNA,

4. sekwencję terminalną Term 5 w postaci łańcuchów DNA,

5. enzymy restrykcyjne *Acu*I i *Bbv*I,

6. enzym *ligazę*.

Do łańcuchów DNA z punktów 1, 2, 3, 4 dołączono dodatkowo fragmenty DNA (z odpowiedniej strony) o określonych długościach 247 pz, 77 pz, 75 pz, 332 pz.



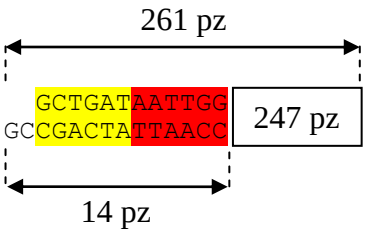
Rys. 67. Elementy automatu testowanego w eksperymencie.

Właściwa długość tzn. uwzględniająca fragmenty kodujące informację wynosi dla:

- słowa wejściowego - 261 pz,
- przejścia T66 - 93 pz,
- przejścia T67 - 88 pz,
- sekwencji terminalnej Term 5 - 340 pz.

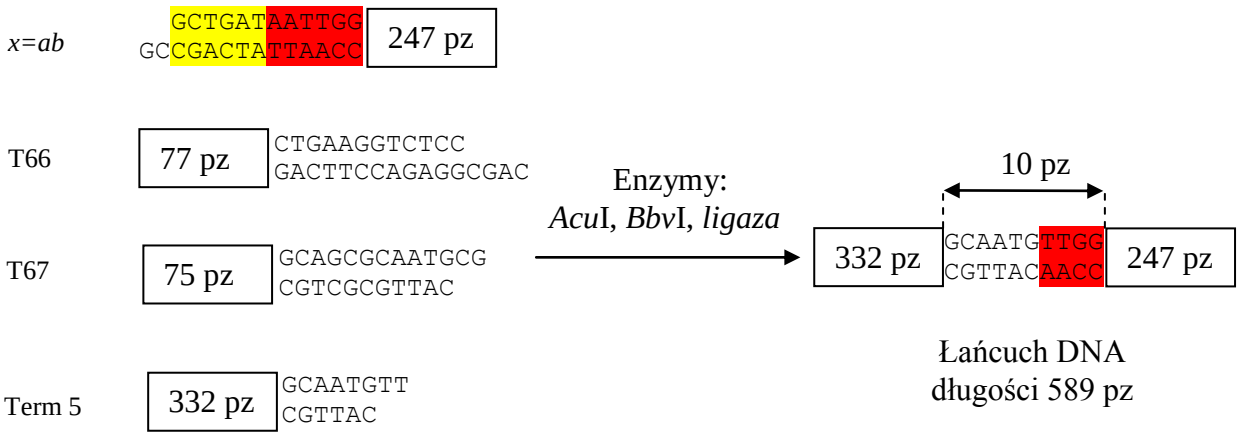
Przykład 16.

Przykładowo dla słowa wejściowego *x=ab* sekwencja kodująca informację jest długości 14 pz, co daje w połączeniu z fragmentem nie kodującym informację (długości 247 pz) łączną długość 261 pz.



Rys. 68. Długość łańcucha DNA kodującego słowo *x=ab*.

Autonomiczne cięcie (dwie endonukleazy *Acu*I i *Bbv*I) i łączenie (*ligaza*) fragmentów DNA kodujących badany automat powinno doprowadzić do powstania lepkiego końca komplementarnego jedynie do fragmentu DNA reprezentującego sekwencję terminalną odpowiadającą stanowi końcowemu S_5. W roztworze pojawi się łańcuch DNA o długości 589 pz (Rys. 69), który może powstać jedynie w wyniku prawidłowego działania badanego automatu i jest on znacznie dłuższy od pozostałych fragmentów DNA (340 pz, 261 pz, 93pz, 88 pz) w roztworze.



Rys. 69. Schemat ideowy doświadczenia.

3. Przebieg doświadczenia

W doświadczeniu sprawdzono działanie ogólnego modelu automatu 6-stanowego 2-symbolowego, a w szczególności autonomicznego działania dwóch enzymów w jednej mieszaninie na słowie wejściowym. Konstrukcja teoretyczna automatu 6-stanowego nie brała pod uwagę specyfiki metod laboratoryjnych koniecznych do implementacji praktycznej automatu z zastosowaniem dwóch endonukleaz. Konieczne było zatem praktyczne sprawdzenie ogólnej metody rozszerzania automatu Shapiro z zastosowaniem dwóch enzymów restrykcyjnych (*Acu*I i *Bbv*I). Dodatkowo utworzono metodą klonowania do wektorów plazmidowych „bibliotek łańcuchów DNA" reprezentujących przejścia automatu. Wprowadzenie takiego rozwiązania umożliwia wielokrotne użycie raz przygotowanych fragmentów DNA.

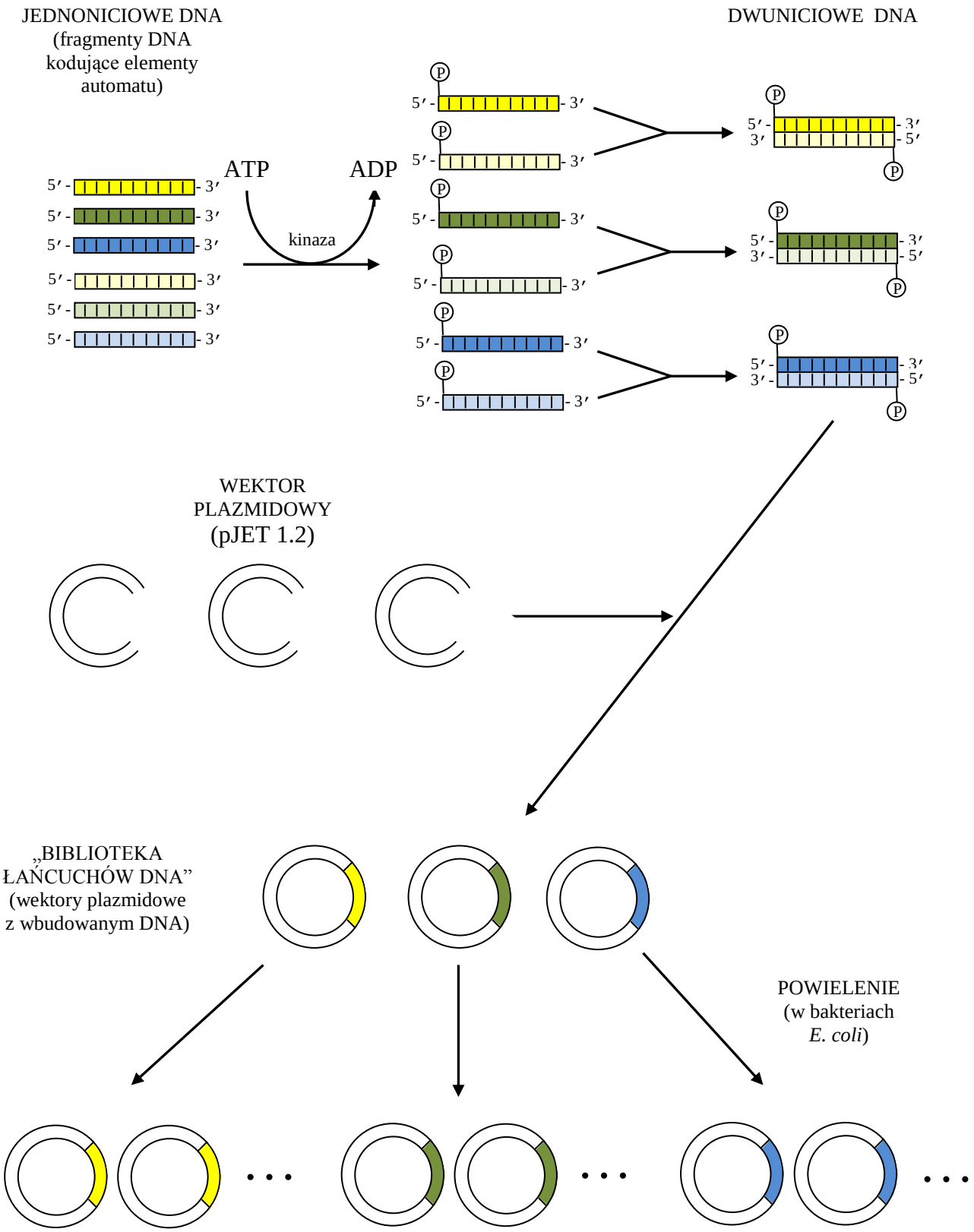
Eksperyment składał się z dwóch głównych etapów. Pierwszy etap polegał na przygotowaniu odpowiednich dwuniciowych fragmentów DNA reprezentujących słowo

wejściowe, przejścia i sekwencję terminalną. W etapie drugim przygotowane uprzednio fragmenty DNA poddano autonomicznemu działaniu dwóch endonukleaz.

Etap 1. Przygotowanie zbioru dwuniciowych fragmentów DNA

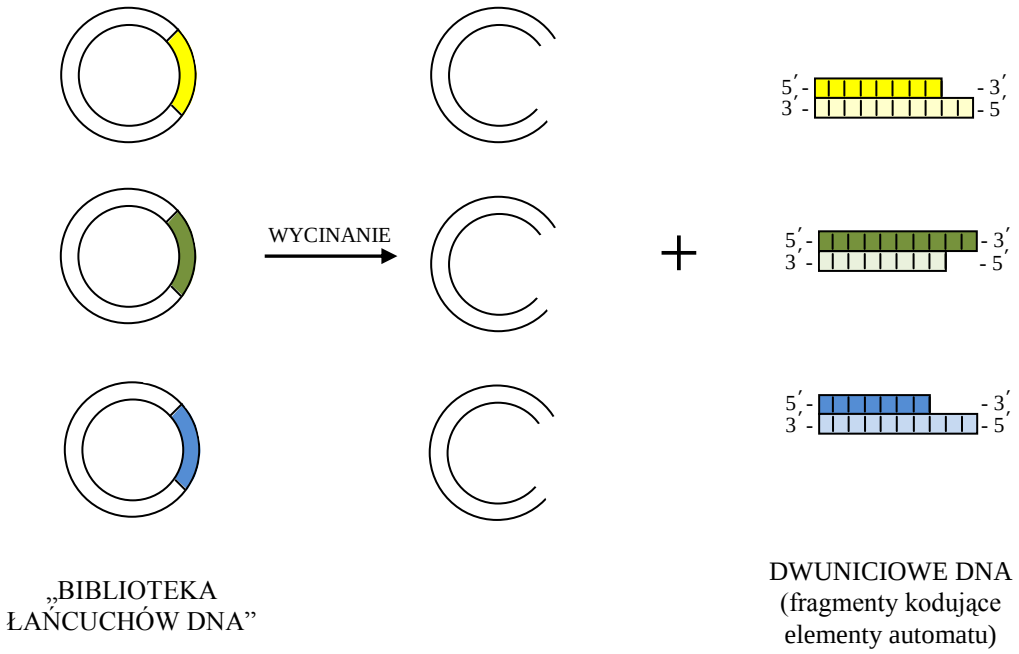
W etapie tym przygotowany został zbiór fragmentów DNA reprezentujący poszczególne elementy testowanego w eksperymencie automatu tzn.: przejścia T66 i T67, słowo *x=ab,* sekwencja terminalna Term 5. Fragmenty te powstały w wyniku renaturacji (zlepienia) jednoniciowych oligonukleotydów o określonej sekwencji DNA (zamówionych w firmie komercyjnej) w dwuniciowe łańcuchy DNA. Dodatkowo zastosowano *kinazę polinukleotydową*. Enzym ten fosforyluje końce 5' jednoniciowych oligonukleotydów w obecności ATP co zwiększa wydajność klonowania dwuniciowych fragmentów DNA (zob. 5.1).

Następnie otrzymane fragmenty DNA zostały wklonowne (zob. 5.2) do wektora plazmidowego (pJET 1.2) dzięki czemu możliwe jest dowolne powielenie (kopiowanie) wektora plazmidowego w komórkach bakteryjnych (*Escherichia coli*), a więc i łańcuchów kodujących informację. Zastosowanie wektora plazmidowego umożliwiło również przygotowanie zbioru łańcuchów DNA łatwych do przechowywania i ponownego użycia („biblioteki łańcuchów DNA"). Dzięki temu możliwa jest konstrukcja innych automatów i wielokrotne wykorzystanie raz przygotowanej „biblioteki łańcuchów DNA". Dodatkową zaletą użycia wektorów plazmidowych na tym etapie eksperymentu jest możliwość wycięcia fragmentów DNA o różnej długości, które są niezbędne do prawidłowego działania automatu. Na rysunku 70 przedstawiono schemat ideowy wykonanych czynności laboratoryjnych zaczynając od fosforylacji pojedynczych fragmentów DNA, a kończąc na powieleniu plazmidów z wbudowanymi fragmentami DNA.



Rys. 70. Przygotowanie „biblioteki łańcuchów DNA”.

Zastosowanie „biblioteki łańcuchów DNA” umożliwia łatwe powielenie przygotowanych łańcuchów DNA, a następnie ich wycięcie (Rys. 71) z wektorów plazmidowych za pomocą enzymów restrykcyjnych (zob. 5.7).



Rys. 71. Wycinanie fragmentów DNA z „biblioteki łańcuchów DNA”.

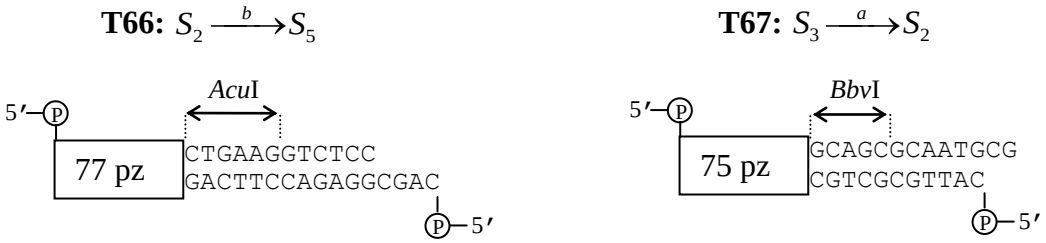
Ogólny schemat wytworzenia dwuniciowych fragmentów DNA różni się od zaproponowanego przez Shapiro i inni [4], [5]. W automacie Shapiro dwuniciowe fragmenty DNA wytworzono w następujący sposób. Pierwszym etapem było trawienie wektora plazmidowego pBluescript II SK+ (2961 pz) enzymem restrykcyjnym *Fok*I i uzyskanie czterech łańcuchy DNA o różnych długościach: 1457 pz, 1036 pz, 181 pz, 287 pz. Z powstałych czterech fragmentów wybrano najdłuższy łańcuch (długość 1457 pz), a następnie sprawdzono sekwencję zasad azotowych lepkiego końca powstałego po trawieniu enzymem *Fok*I. W kolejnym etapie zamówiono oligonukleotydy kodujące informacje o poszczególnych elementach automatu w postaci łańcuchów DNA, które komplementarnie połączono ze sobą. Uzyskane dwuniciowe fragmenty DNA zawierały lepkie końce komplementarne do fragmentu długości 1457 pz powstałego po trawieniu enzymem *Fok*I wektora plazmidowego pBluescript II SK. Dwuniciowe fragmenty DNA kodujące informacje połączono lepkimi końcami z fragmentami długości 1457 pz. Ostatnim etapem było powielenie fragmentów DNA pożądanej sekwencji i wielkości w otrzymanych łańcuchach metodą PCR mające na celu uzyskanie fragmentów DNA odpowiednich długości.

Etap 2. Weryfikacja działania automatu

W doświadczeniu badane było słowo *x=ab* (Rys. 72) akceptowane przez automat *M* przedstawiony na rysunku 65.

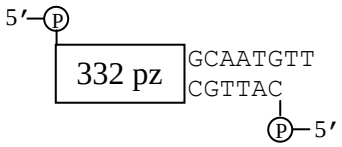
Rys. 72. Fragment DNA reprezentujący słowo *x=ab*.

Wczytując słowo *x=ab* automat będzie znajdował się kolejno w stanach S_3, S_2, S_5. Przejścia zostały zakodowane, zgodnie z ogólnym modelem automatu 6-stanowego przedstawionego w rozdziale 4 (Rys. 73).



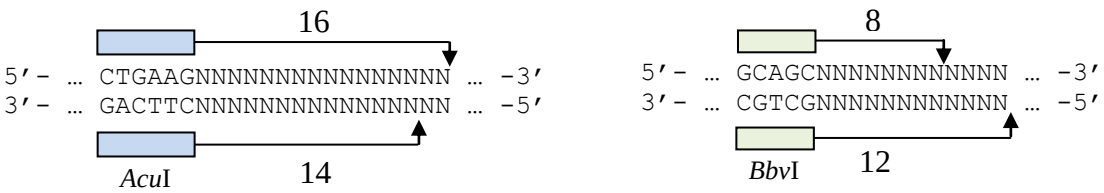
Rys. 73. Fragmenty DNA reprezentujące przejściaT66 oraz T67.

Dodatkowo przyjęto, że stanem końcowym w automacie jest stan S_5 zakodowany fragmentem DNA (Term 5) w następujący sposób.



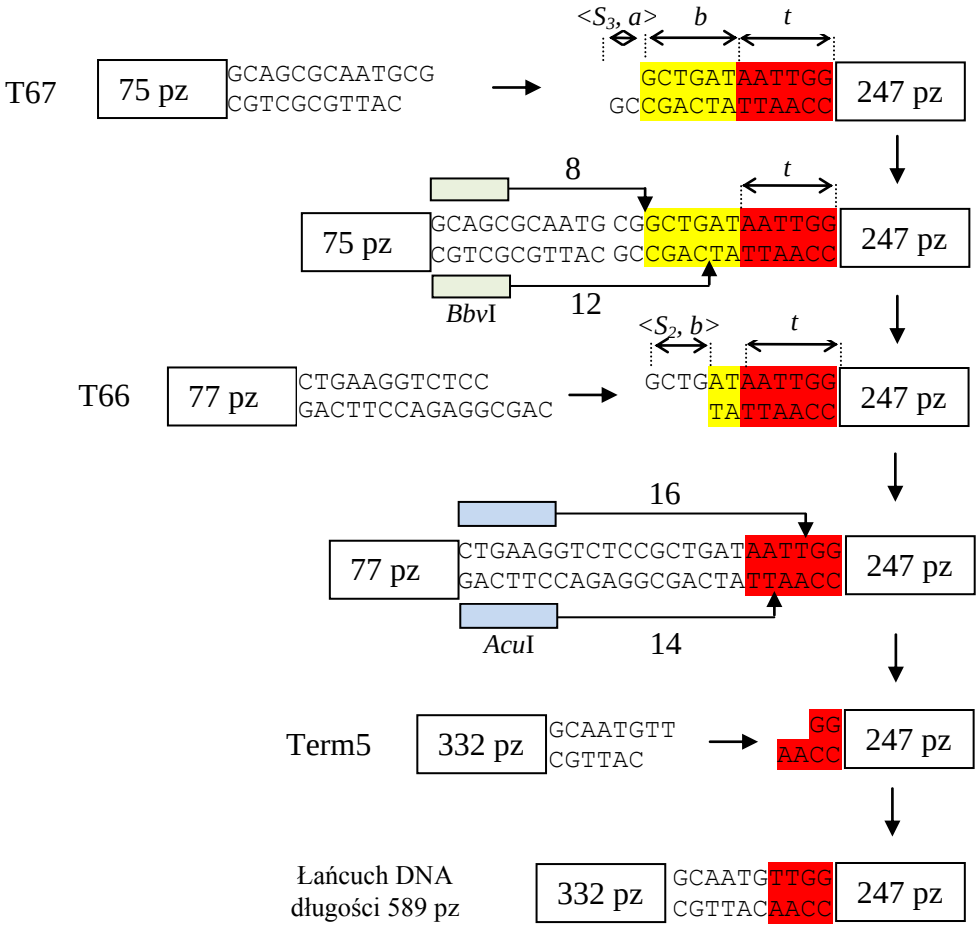
Rys. 74. Fragment DNA reprezentujący Term 5.

W doświadczeniu użyte zostały dwa enzymy restrykcyjne *Acu*I oraz *Bbv*I. Enzym *Acu*I rozpoznaje sekwencję CTGAAG i tnie łańcuch DNA po 16 nukleotydach w kierunku 5'→3' oraz po 14 nukleotydach w kierunku 3'→5'. Natomiast enzym *Bbv*I rozpoznaje sekwencję GCAGC i tnie łańcuch DNA po 8 nukleotydach w kierunku 5'→3' oraz po 12 nukleotydach w kierunku 3'→5'.



Rys. 75. Cięcie enzymami *Acu*I oraz *Bbv*I.

Przy prawidłowym przebiegu doświadczenia automat kończy działanie w stanie S_5. Wtedy możliwe jest przyłączenie się lepkiego końca fragmentu terminalnego (Term 5) do lepkiego końca powstałego po działaniu enzymów (*Acu*I, *Bbv*I) na słowie *x=ab*. Zatem pojawi się łańcuch o długości 589 pz. Łańcuch tej długości może pojawić się w roztworze jedynie w przypadku, gdy cały proces przetwarzania informacji przebiegł prawidłowo. Obecność tego łańcucha oznacza również, że słowo *x=ab* zostało zaakceptowane przez automat. Na rysunku 76 przedstawiony został planowany przebieg eksperymentu. Stopniowe trawienie (analiza) słowa dwoma enzymami restrykcyjnymi doprowadza do powstania lepkiego końca AA, a następnie komplementarnego przyłączenia łańcucha DNA długości 338 pz (lepki koniec TT) i prawidłowego zakończenia działania automatu - akceptacji słowa *x=ab*.

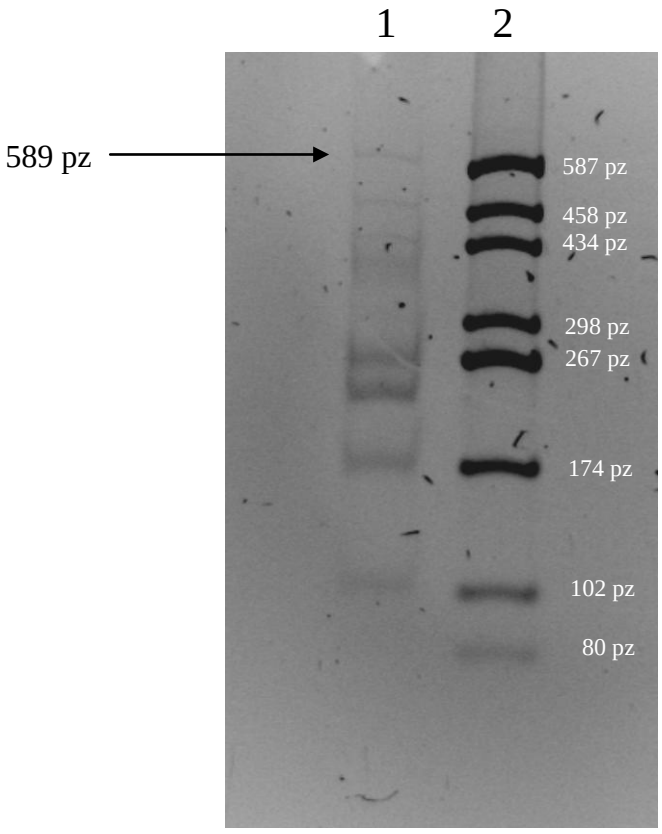


Rys. 76. Działanie badanego automatu.

Reakcja została przeprowadzona przez zmieszanie w jednej probówce typu Eppendorf:

- DNA reprezentującego słowo *x=ab*,
- DNA reprezentującego Term 5,
- DNA reprezentującego przejście T66,
- DNA reprezentującego przejście T67,
- enzymu *Acu*I,
- enzymu *Bbv*I,
- enzymu *ligaza*,
- buforu firmowego NEBuffer 2,
- ATP.

Całość inkubowano przez 1,5 h w temperaturze 37 °C. Produkt reakcji oczyszczono trzykrotnie mieszaniną fenol:chloroform:alkohol izoamylowy (25:24:1), a następnie rozdzielono metodą elektroforezy w 8% żelu poliakrylamidowym (Rys. 77).



Rys. 77. Rozdział elektroforetyczny produktów powstałych po zakończeniu działania automatu. Kanały: 1 - produkt powstały po zakończeniu działania automatu (autonomiczne działanie dwóch endonukleaz *Acu*I oraz *Bbv*I); 2 - standard wielkości DNA uzyskany z pUC19/*Hae*III.

Analiza preparatu poreakcyjnego zawierającego fragmenty DNA powstałe w wyniku autonomicznego działania dwóch enzymów restrykcyjnych w jednej mieszaninie pozwoliła na identyfikację fragmentów DNA o oczekiwanej wielkości tzn. 589 pz. Analiza zdjęcia wykonanego po elektroforezie na żelu poliarylamidowym wskazuje, że przetwarzanie informacji (działanie automatu) zaszło według oczekiwań, co potwierdza słuszność idei zwiększania liczby enzymów działających w jednej mieszaninie. Dodatkowo widoczne są inne fragmenty DNA reprezentujące poszczególne elementy testowanego automatu oraz produkty reakcji powstałe w wyniku działania enzymów restrykcyjnych.

4. Materiały

4.1. Szczepy bakteryjne

- szczep bakterii *Escherichia coli* JM107.

4.2. Wektory plazmidowe

- pJET 1.2 (Fermentas).

4.3. Podłoża bakteryjne

- L-Bulion, L-agar.
- Podłoże SOC:

  Podłoże przygotowywano przez dodanie do podłoża SOB jałowego 20% roztworu glukozy (do stężenia końcowego 20 mM) oraz 2M $MgCl_2$ (do stężenia 100 mM). Podłoże SOC stosowano do przygotowywania komórek kompetentnych *E. coli*.

4.4. Bufory i roztwory

- Bufory do elektroforezy w żelach agarozowych:

  boranowy: 0,04 M Tris-HCl pH=7,6; 0,04 M kwas borowy; 0,002 M EDTA.
- Bufor TE, stosowany do rozpuszczania preparatów DNA:

  0,01 M Tris-HCl,

  0,001 M EDTA,

  pH=8,0.
- Bufor do elucji DNA z żeli poliakrylamidowych:

  0,5 M octan amonowy ,

  1mM EDTA pH=8,0,

  0,1% SDS,

  0,1M octan magnezu.

4.5. Enzymy restrykcyjne

- *Bbv*I , *Acu*I - endonukleazy użyte w doświadczeniu,
- *Bsm*AI, *Cla*I, *Bsr*DI, *Not*I, *Acu*I - endonukleazy użyte do wycinania odpowiednich fragmentów z wektora plazmidowego,
- enzymy stosowano zgodnie z zaleceniami producenta (NEB oraz Fermentas).

4.6. Syntetyczne oligonukleotydy

- słowo *ab*:
  ab_1 (5'-TAACTGAAGTCAATCTAAAGTATCGGCTGATAATTGGGAGCAA-3')
  ab_2 (5'-TTGCTCCCAATTATCAGCCGATACTTTAGATTGACTTCAGTTA-3').

- przejście T66:
  T66_1 (5'-ACTCAAAGGCGGTAATACGGTTATCCACAGCTGAAGGTCTCCGCTG-3')
  T66_2 (5'-CAGCGGAGACCTTCAGCTGTGGATAACCGTATTACCGCCTTTGAGT-3').

- przejście T67:
  T67_1 (5'-ATCAGGGGATAACGCAGGAAAGAACATGTGCAGCGCAATGCG-3')
  T67_2 (5'-CGCATTGCGCTGCACATGTTCTTTCCTGCGTTATCCCCTGAT-3').

- sekwencja terminalna:
  Term5_1 (5'- GCGTTTTTCCATAGGCTCCGCCCCCCTGACGAGCATCACAAAAA TCGACGCTCAAGTCAGAGGTGGCGAAGCAATGTT-3')
  Term5_2 (5'-AACATTGCTTCGCCACCTCTGACTTGAGCGTCGATTTTTGTGATG CTCGTCAGGGGGGCGGAGCCTATGGAAAAACGC-3').

4.7. Pozostałe odczynniki

Bromek etydyny (Sigma),

Tris base (Sigma),

ligaza T4 (NEB),

Ampicylina (Sigma),

TEMED (Serva).

4.8. Standardy do określania wielkości DNA

- Fragmenty DNA plazmidu pUC 19 otrzymane po trawieniu enzymem *Hae*III:
  587 pz, 434 pz, 298 pz, 267 pz, 257 pz, 174 pz, 102 pz, 80 pz, 18 pz, 11 pz.
- Fragmenty DNA faga *phi*X174 otrzymane po trawieniu enzymem *Bsu*RI (*Hae*III):
  1353 pz, 1078 pz, 872 pz, 603 pz, 310 pz, 281 pz, 271 pz, 234 pz, 194 pz,
  118 pz, 72 pz.
- 100 pz - standard do oznaczania wielkości DNA firmy *New England BioLabs*.

5. Metody

5.1. Fosforylacja oligonukleotydów i wytworzenie dwuniciowych fragmentów DNA.

Syntetyczne oligonukleotydy są pozbawione grupy fosforanowej na końcu 5', a grupa ta jest niezbędna do wklonowania fragmentu DNA w wektor plazmidowy. Końce 5' na pojedynczych łańcuchach DNA poddano fosforylacji *kinazą polinukleotydową*. Enzym ten przenosi grupę fosforanową z ATP na koniec hydroksylowy 5' oligonukleotydów. Komplementarne do siebie oligonukleotydy zmieszano ze sobą, a następnie inkubowano w celu samoistnego wytworzenie dwuniciowych fragmentów DNA z jednoniciowych oligonukleotydów. W wyniku komplementarnego zlepienia oligonukleotydów powstały dwuniciowe łańcuchy DNA (Dodatek A). Fragmenty te nie zawierały lepkich końców niezbędnych do testowania automatu - końce te powstaną w kolejnych etapach eksperymentu przy udziale enzymów restrykcyjnych.

5.2. Klonowanie fragmentów DNA do wektora plazmidowego.

Dwuniciowe fragmenty DNA zostały wstawione do plazmidu pJET 1.2, a następnie wprowadzone do bakterii *E. coli* metodą transformacji. Klonowanie wykonane zostało zestawem do klonowania łańcuchów DNA o nazwie *CloneJET*^{TM} (Dodatek C). Zestaw ten zawiera:

- wektor plazmidowy pJET 1.2,
- *ligaza* DNA,
- bufor reakcyjny,
- szczep *E.coli* JM107.

5.3. Izolacja i pozyskiwanie DNA plazmidowego

Izolację DNA plazmidowego wykonano zestawem *Gene MATRIX Plasmid Miniprep DNA Purification Kit*, który umożliwia szybkie i wydajne pozyskiwanie wektorów plazmidowych (Dodatek D). Pierwszy etap polegał na hodowli bakterii zawierających wektor plazmidowy w obecności antybiotyku. Następnie wykonano lizę bakterii oraz oczyszczanie preparatu z białek. Ostatnim etapem przygotowania plazmidów było wytrącenie kwasów nukleinowych z roztworu w celu ich zatężenia. Uzyskane w ten sposób wektory plazmidowe zostały użyte do kolejnych etapów eksperymentu.

5.4. Sprawdzenie poprawności sekwencji łańcuchów DNA

DNA otrzymanych klonów wysłane zostały do Pracowni Sekwencjonowania i Syntezy Oligonukleotydów Instytutu Biochemii i Biofizyki Polskiej Akademii Nauk w celu sprawdzenia poprawności uzyskanych sekwencji DNA. Wektory plazmidowe użyte w doświadczeniu nazwano w następujący sposób:

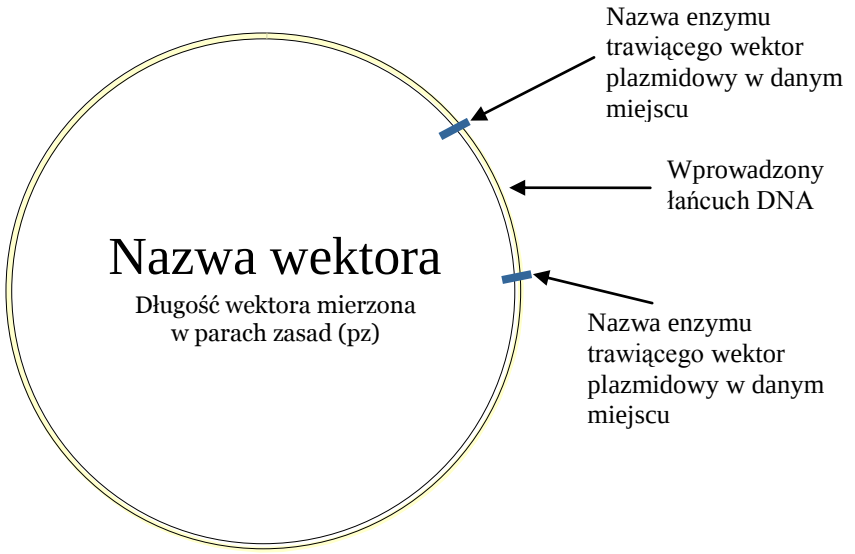
- pPSAB - zawiera fragment kodujący słowo *x=ab*,
- pPST66 - zawiera fragment kodujący przejście T66,
- pPST67 - zawiera fragment kodujący przejście T67,
- pPSTER - zawiera fragment kodujący sekwencję terminalną Term5.

Wyniki sekwencjonowania potwierdziły obecność właściwych fragmentów DNA w wektorach plazmidowych (Dodatek B).

5.7. Trawienie DNA enzymami restrykcyjnymi.

Wektory plazmidowe z wklonowanymi fragmentami DNA kodującymi informację wymagały przetworzenia w celu uzyskania łańcuchów DNA o określonych długościach oraz specyficznych lepkich końcach. Konieczne było zatem opracowanie metody wycinania odpowiednich fragmentów DNA z wektora plazmidowego, tak aby uzyskać odpowiedni lepki koniec kodujący <stan, symbol> oraz określoną długość łańcucha DNA.

Poniższy rysunek przedstawia schemat ideowy wycinania fragmentów DNA z wektora plazmidowego. Na rysunku przedstawiony został również sposób opisu plazmidu używany w dalszej części pracy.

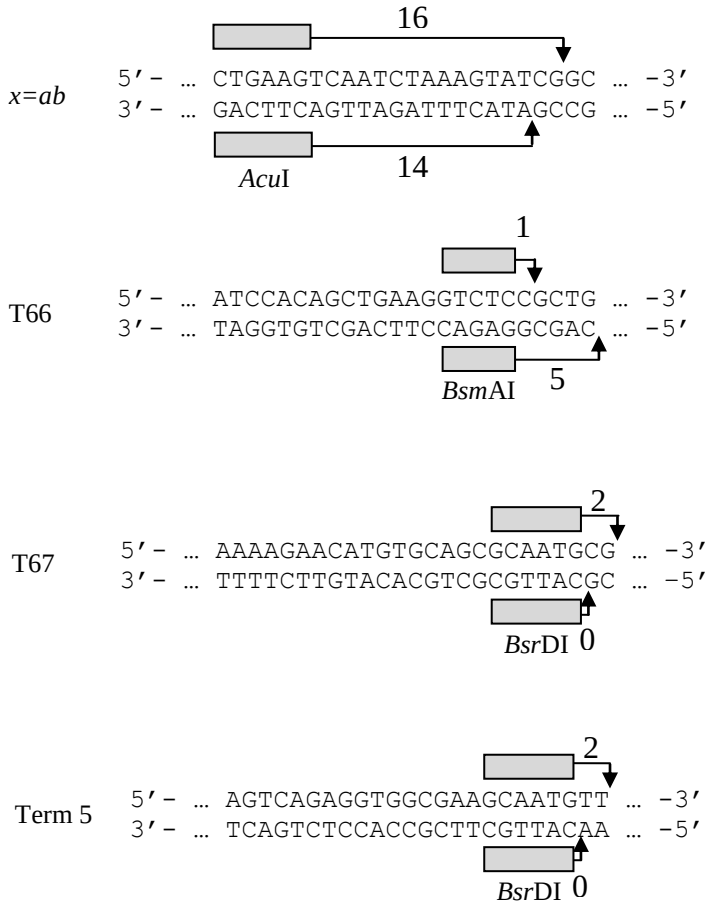


Rys. 78. Schemat opisu wycinania fragmentów DNA z plazmidu.

5.7.1. Uzyskanie lepkiego końca

W badanym automacie lepki koniec koduje informacje o aktualnym stanie oraz wczytanym symbolu. Konieczne zatem było opracowanie metody uzyskiwania lepkich końców dla poszczególnych elementów automatu tzn.: przejścia T66 i T67, słowa wejściowego, sekwencji terminalnej. Główna idea uzyskania lepkich końców polegała na trawieniu wektora plazmidowego enzymami restrykcyjnymi. Konieczne było zatem odpowiednie umieszczenie miejsca restrykcyjnego w zamówionych oligonukleotydach, tak aby w wyniku cięcia wektora plazmidowego powstał odpowiedni lepki koniec. W tym celu zastosowano następujące enzymy restrykcyjne:

1. *Acu*I - dla słowa wejściowego (uzyskano lepki koniec GC w kierunku 3'→5',
2. *Bsm*AI - dla przejścia T66 (uzyskano lepki koniec CGAC w kierunku 3'→5'),
3. *Bsr*DI - dla przejścia T67 (uzyskano lepki koniec CG w kierunku 5'→3'),
4. *Bsr*DI - dla łańcucha Term5 (uzyskano lepki koniec TT w kierunku 5'→3').

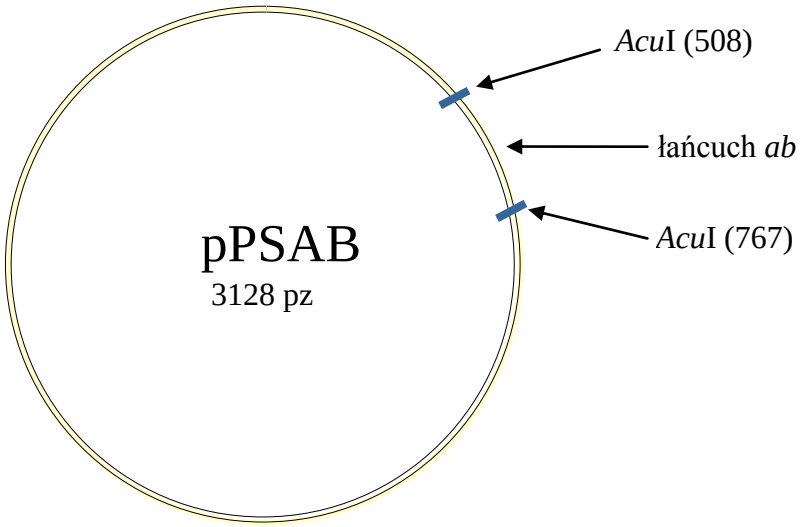


Rys. 79. Uzyskanie lepkiego końca dla poszczególnych elementów automatu.

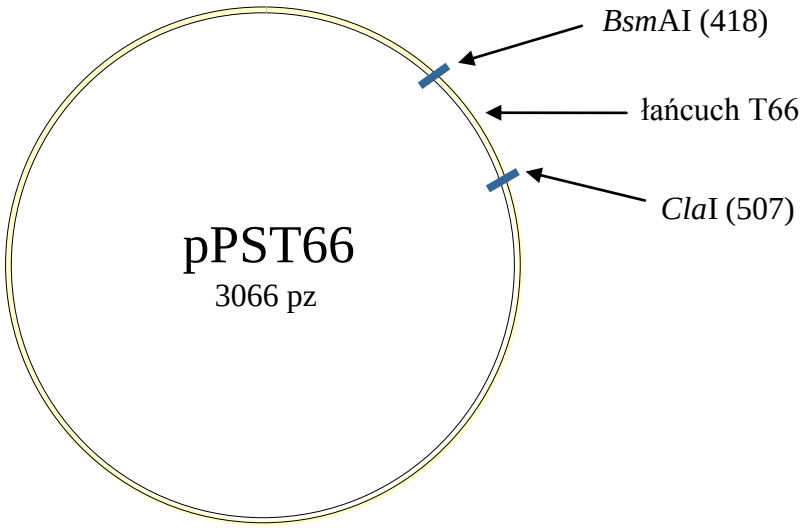
5.7.2. Wycinanie fragmentów z wektora plazmidowego

Po uzyskaniu lepkiego końca z jednej strony łańcuchów DNA należało teraz trawić wektory plazmidowe tak aby otrzymać łańcuchy o określonej długości i lepki koniec z drugiej strony. W tym celu przeanalizowano występowanie miejsc restrykcyjnych dla różnych endonukleaz w plazmidzie pJET 1.2. Do uzyskania fragmentów DNA o określonej długości wybrano następujące enzymy restrykcyjne: *Acu*I (słowo *x=ab*, długość 261 pz), *Cla*I (przejście T66, długość 93 pz), *Not*I (przejście T67, długość 88 pz), *Bsm*AI (Term 5, długość 340 pz).

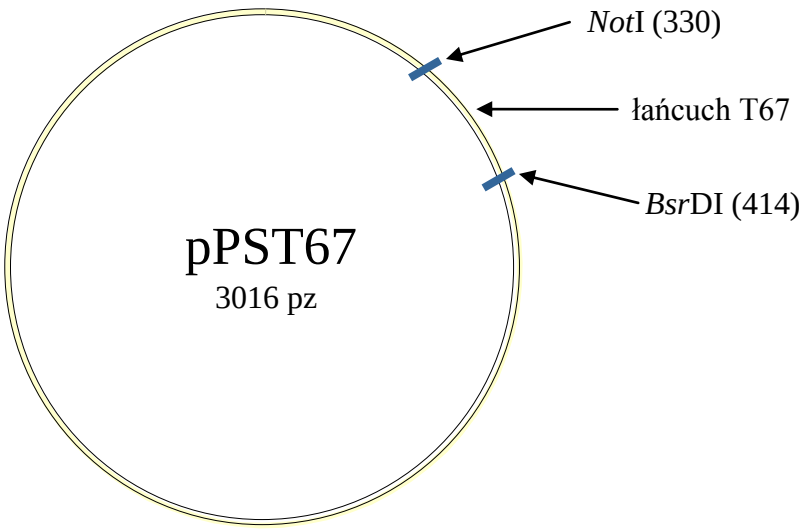
Poniżej przedstawione zostały graficznie wektory plazmidy użyte w doświadczeniu (pPSAB, pPST66, pPST67, pPSTER) wraz z zastosowanymi enzymami restrykcyjnymi.



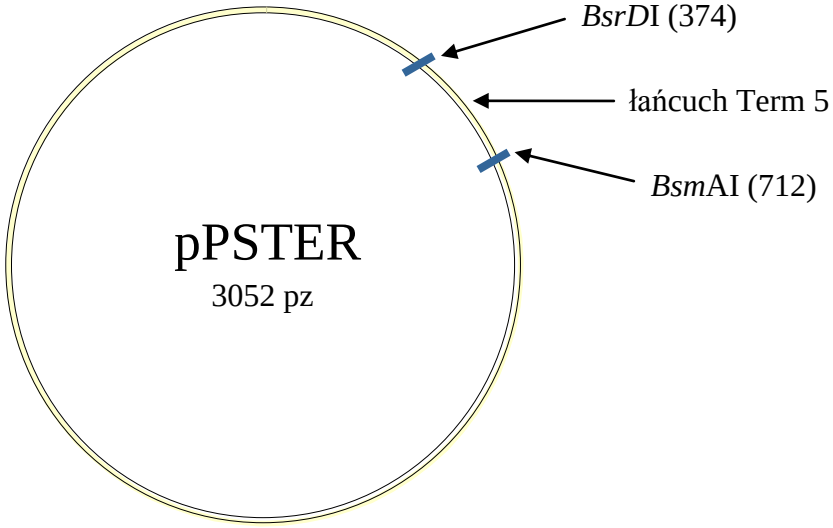
Rys. 80. Miejsca cięcia enzymami dla słowa *x=ab*.



Rys. 81. Miejsca cięcia enzymami dla przejścia T66.



Rys. 82. Miejsca cięcia enzymami dla T67.



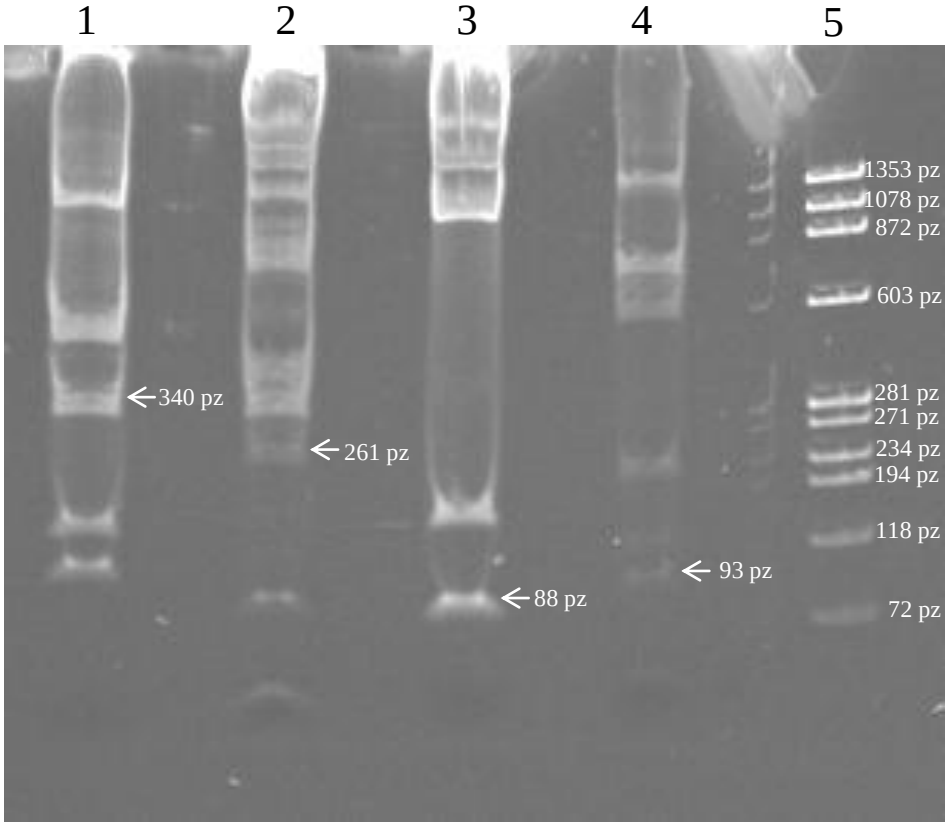
Rys. 83. Miejsca cięcia enzymami dla Term 5.

5.6. Elektroforeza i izolacja DNA z żelu.

W celu oddzielenia żądanych łańcuchów od pozostałych fragmentów wektora plazmidowego wykorzystano elektroforezę w żelu poliakrylamidowym. Fragmenty DNA pożądanych wielkości wycięto następnie z żelu poliakrylamidowego oraz oczyszczono z żelu i użyto w kolejnym etapie weryfikacji działania automatu.

Analiza elektroforetyczna dla poszczególnych wektorów plazmidowych została przedstawiona na poniższym rysunku (Rys. 87):

1. sekwencja terminalna Term 5 trawiona enzymem *Bsr*DI w temperaturze 65°C (NEBuffer 2) oraz *Bsm*AI w temperaturze 55°C (NEBuffer 2) - 340 pz.
2. słowo *x=ab* trawione w dwóch miejscach enzymem *Acu*I w temperaturze 37°C (NEBuffer 2) - 261 pz,
3. przejście T67 trawione enzymem *Bsr*DI w temperaturze 65°C (NEBuffer 2) oraz *Not*I w temperaturze 37°C (Bufor pomarańczowy Fermentas) - 88 pz,
4. przejście T66 trawione enzymem *Bsm*AI w temperaturze 55°C (NEBuffer 4) oraz *Cla*I w temperaturze 37 °C (NEBuffer 4) - 93 pz.



Rys. 84. Rozdział elektroforetyczny preparatów plazmidowych zawierających elementy automatu. Kanały: 1 - produkt powstały po trawieniu plazmidu pPSTER endonukleazą *Bsr*DI oraz *Bsm*AI (Term5, długość 340 pz); 2 - produkt powstały po trawieniu plazmidu pPSAB endonukleazą *Acu*I (słowo *x=ab*, długość 261 pz); 3 - produkt powstały po trawieniu plazmidu pPST67 endonukleazą *Bsr*DI oraz *Not*I (T67, długość 88 pz); 4 - produkt powstały po trawieniu plazmidu pPST66 endonukleazą *Bsm*AI oraz *Cla*I (T66, długość 93 pz); 5 - standard wielkości DNA uzyskany z *phi*X174 DNA/*Bsu*RI.

Rozdział 6

Dyskusja wyników

Głównym celem pracy było określenie możliwości rozszerzenia automatu Shapiro opartego na łańcuchach DNA. W pracy podano nową ideę rozszerzenia automatu Shapiro, której główną myślą było zastosowanie dwóch (lub więcej) enzymów restrykcyjnych działających autonomicznie w jednej mieszaninie. Automat Shapiro był prostym 2-stanowym 2-symbolowym automatem skończonym, w którym zastosowano tylko jeden enzym restrykcyjny. Jego autorzy podali jedynie, że możliwe jest rozszerzenie tego automatu do 3 stanów, co zostało potwierdzone eksperymentalnie przez dwa zespoły badawcze w [35] i [40]. Zespół Shapiro i inni [4] przewidywał w 2001 roku możliwość odkrycia w niedalekiej przyszłości enzymów restrykcyjnych pozostawiających znacznie dłuższe lepkie końce (>4 pz), które umożliwiłyby znaczne zwiększenie liczby stanów automatu (>3 stany). Aktualnie dostępnych jest kilkaset różnych enzymów restrykcyjnych trawiących łańcuchy DNA w różny sposób. Ciągle odkrywane są nowe endonukleazy, jednak enzymy te tną łańcuchy DNA pozostawiając stosunkowo krótkie lepkie końce tzn. w przedziale od 1 pz do 5 pz. Można jednak wykorzystać inne właściwości endonukleaz do rozszerzania modelu automatu Shapiro takie jak:

1. rozpoznawanie różnych sekwencji zasad azotowych w łańcuchach DNA,
2. trawienie łańcuchów DNA w różnej odległości od sekwencji rozpoznawanej,
3. pozostawianie lepkich końców na łańcuchu DNA w kierunku 5'→3' lub w kierunku 3'→5'.

Powyższe własności enzymów restrykcyjnych umożliwiają zwiększenie liczby stanów w automacie Shapiro przez zastosowanie wielu endonukleaz działających autonomicznie w jednej mieszaninie. Użycie dwóch enzymów pozostawiających dwa rodzaje lepkich końców (długości 4 pz w kierunku 5'→3' oraz długości 2 pz w kierunku 3'→5') zwiększyło liczbę możliwych kombinacji do zakodowania w lepkim końcu. Umożliwiło to zakodowanie wszystkich 72 przejść wystarczających dla modelu automatu 6-stanowego 2-symbolowego. Wymagało to odpowiedniego dobrania lepkich końców, gdyż kodują one wszystkie pary <stan,symbol>. Również enzymy restrykcyjne zostały tak dobrane, aby odległości od sekwencji rozpoznawanej były stosunkowo duże. Zwiększyło to możliwości sterowania działaniem enzymów restrykcyjnych na słowie wejściowym. Ze względu na konieczność jednoznacznego kodowania informacji (aktualny stan i wczytany

symbol) każdy lepki koniec musiał być unikalny tzn. różnić się od pozostałych sekwencją zasad azotowych.

W pracy przedstawiono również rozważania dotyczące teoretycznych możliwości rozszerzania automatu Shapiro do dowolnej ilości stanów i dowolnej ilości symboli podając warunki arytmetyczne kiedy jest to możliwe. Przebadano teoretyczne możliwości kodowania automatów p stanowych i r symbolowych w których kodujemy symbole ciągami n elementowymi o wyrazach ze zbioru q elementowego oraz używając j enzymów restrykcyjnych pozostawiających po cięciu $k_1,\ldots,k_j$ wyrazowe lepkie końce. Ze względu na teoretyczny charakter rozważań przyjęto dowolną ilość symboli q, za pomocą których kodujemy symbole danego alfabetu Σ (zamiast przyjętych w komputerach biomolekularnych 4 symboli A, T, G, C). W pracy podano następujące twierdzenia.

Twierdzenie 1 Możliwa jest konstrukcja automatu Shapiro o p stanach i r symbolach, w których kodujemy symbole ciągami n elementowymi o wyrazach ze zbioru q elementowego oraz używamy jednego enzymu restrykcyjnego pozostawiającego po cięciu k wyrazowy lepki koniec $(1 \le k \le n)$, gdy:

1. $p = n - k + 1$ (wystarczy $p \le n - k + 1$),

2. $q^k > \dfrac{(rn-k)(rn-k+1)}{2}$.

W pracy doktorskiej badano również przypadek zastosowania większej ilości enzymów. Dla ułatwienia rozważań rozpatrzono najpierw przypadek 2 enzymów pozostawiających lepkie końce różnej długości. Oznaczono te długości przez k_1 i k_2. Zatem po cięciu otrzymujemy

k_1 k_2

Ponieważ $k_1 \neq k_2$, więc ich lepkie końce nie spowodują konfliktu. Z rozważań dotyczących przypadku jednego enzymu wynika, że ilość stanów automatu przy cięciu pierwszym enzymem jest równa

$$p_1 = n - k_1 + 1.$$

Podobnie dla drugiego enzymu

$$p_2 = n - k_2 + 1.$$

Zatem łączna (maksymalna) ilość stanów automatu, przy użyciu 2 enzymów, jest równa

$p = 2n - (k_1 + k_2) + 2$.

Korzystając z twierdzenia 1 otrzymujemy następujące twierdzenie.

Twierdzenie 2. Możliwa jest konstrukcja automatu Shapiro o p stanach i r symbolach, w którym kodujemy symbole ciągami n elementowymi o wyrazach ze zbioru q elementowego oraz używamy dwóch enzymów restrykcyjnych pozostawiających po cięciu lepkie końce długości k_1 i odpowiednio k_2, gdy:

1. $p = 2n - (k_1 + k_2) + 2$ (wystarczy $p \le 2n - (k_1 + k_2) + 2$),
2. $q^{k_1} > \frac{(rn - k_1)(rn - k_1 + 1)}{2}$,
3. $q^{k_2} > \frac{(rn - k_2)(rn - k_2 + 1)}{2}$.

Uogólniono to twierdzenie również na przypadek, w którym użyto j enzymów restrykcyjnych (tw. 5 w Roz. 4).

W pracy doktorskiej przedstawiono również implementację laboratoryjną automatu działającego autonomicznie w jednej mieszaninie z zastosowaniem dwóch enzymów restrykcyjnych. Przeprowadzone doświadczenia laboratoryjne potwierdziły tezę, że możliwe jest rozszerzenie automatu Shapiro przez zastosowanie dwóch enzymów w celu zwiększenia liczby stanów automatu. Zgodnie ze schematem ideowym doświadczenia eksperyment podzielono na dwa główne etapy: przygotowanie zbioru dwuniciowych fragmentów DNA (Etap 1) oraz weryfikacja działania automatu z użyciem dwóch enzymów restrykcyjnych (Etap 2). W pierwszym etapie zlepiono pojedyncze łańcuchy DNA, klonowano je w wektor plazmidowy, a następnie wycinano odpowiednie fragmenty DNA. Drugi etap badał działanie automatu przez naprzemienne cięcie odpowiednio przygotowanych łańcuchów DNA dwoma enzymami restrykcyjnymi oraz łączenie enzymem *ligazą*.

Wymieńmy dwie główne różnice w stosunku do eksperymentu przedstawionego przez zespół Shapiro i inni:

1. **Użycie dwóch enzymów restrykcyjnych.** W automacie Shapiro zastosowano tylko jeden enzym restrykcyjny *Fok*I, za pomocą którego zaimplementowano jedynie 2-stanowy 2-symbolowy automat skończony. Zastosowanie dwóch enzymów restrykcyjnych na odpowiednio zaprojektowanych łańcuchach DNA umożliwiło rozszerzenie automatu Shapiro do modelu 6-stanowego 2-symbolowego. Wymagało to jednak wyboru odpowiednich dwóch enzymów

restrykcyjnych (*Bbv*I, *Acu*I), które działają w takim samym środowisku reakcyjnym (temperatura, pH, jony).

2. **Tworzenie „bibliotek łańcuchów DNA”.** Ze względu na pracochłonność metod genetyki molekularnej przygotowanie łańcuchów DNA kodujących informacje jest kluczowym etapem implementacji praktycznej automatu. Wprowadzenie klonowania w wektor plazmidowy (pJET 1.2) i przechowywanie łańcuchów DNA w bakteriach *Escherichia coli* umożliwia ponowne wykorzystanie raz przygotowanych łańcuchów. Szczepy bakterii łatwo się namnaża (kopiuje) i przechowuje (magazynuje) w warunkach laboratoryjnych. Wykonanie obliczenia dla innego automatu sprowadza się do użycia łańcuchów DNA z „biblioteki łańcuchów DNA”.

Planowane są dalsze badania laboratoryjne w których zostanie sprawdzony wpływ na wyniki końcowe: długości słów, niedeterminizmu automatu, złożoności automatu, liczby użytych enzymów oraz ilości użytych związków chemicznych.

Automat Shapiro jest niewątpliwie ciekawym pomysłem na przetwarzanie informacji za pomocą DNA. Do zalet automatów opartych na enzymach restrykcyjnych można zaliczyć masową równoległość zachodzącego procesu przetwarzania informacji, czyli działanie enzymów jednocześnie na wielu kopiach słowa wejściowego. Zauważalne są jednak ograniczenia techniczne i teoretyczne automatu Shapiro. Pierwszym problemem są lepkie końce, gdyż w praktyce laboratoryjnej mają one stosunkowo małe długości (od 1 do 5 nukleotydów) i w konsekwencji mogą kodować niewiele stanów i symboli. Zastosowanie wielu enzymów restrykcyjnych powoduje z kolei konieczność określenia warunków laboratoryjnych dla działania enzymów w jednej mieszaninie, gdyż na ogół optymalne działanie enzymów zachodzi w różnych warunkach środowiska reakcyjnego. W ostatnich latach odkryto nową grupę enzymów działających w takich samych warunkach: jednakowe pH i temperatura działania (FastDigest® firmy Fermentas), co umożliwia łatwiejsze konstruowanie automatów wielostanowych opartych na enzymach restrykcyjnych. Dodatkową zaletą tych enzymów jest dużo większa szybkość trawienia łańcuchów DNA wynosząca 5-15 minut, podczas gdy dotychczas używane enzymy restrykcyjne wymagają ok. 60 minut do prawidłowego przebiegu reakcji. Umożliwia to zastosowanie większej ilości działających enzymów w jednej mieszaninie do konstrukcji bardziej złożonych automatów.

Dalsze badania nad automatem Shapiro wydają się obiecujące, szczególnie nad zastosowaniami np. w medycynie czy biotechnologii. Automat ten zbudowany jest

wyłącznie ze związków organicznych, co w przyszłości predysponuje go do zastosowania wewnątrz organizmów żywych. Problemem jest jednak umieszczenie łańcuchów DNA w komórkach żywych organizmów, gdyż łańcuchy DNA, które znajdą się poza jądrem komórkowym ulegają zniszczeniu przez enzymy występujące w cytoplazmie. Dlatego konieczne jest umieszczenie takich automatów w oddzielnym środowisku np. wewnątrz błon lipidowych, które oddzielałyby automaty od środowiska organizmów żywych. Obliczenie (np. diagnostyka nowotworu) mogłoby odbywać się wewnątrz błony lipidowej, do środowiska której wnikałoby np. mRNA kodujące nowotwór, a następnie uwalniany mógłby zostać lek np. w postaci siRNA do cytoplazmy komórkowej. Zespół Shapiro podaje, że automat ten może być wykorzystany do diagnozowania i leczenia nowotworów [6]. Wydaje się, że idea automatu Shapiro i jego rozszerzania może zostać wykorzystana w biochipach (mikromacierzach) stosowanych do diagnostyki różnych chorób. Zastosowanie w technice, medycynie i nauce wymaga jednak długotrwałych i kosztownych badań laboratoryjnych, a następnie klinicznych.

Bibliografia

1. Adar R., Benenson Y., Linshiz G., Rosner A., Tishby N., Shapiro E.: *Stochastic computing with biomolecular automata*. PNAS 101, 9960-9965 (2004).

2. Adleman L.: *Molecular computation of solutions to combinatorial problems.* Science 226, 1021-1024 (1994).

3. Amos M.: *Theoretical and Experimental DNA Computation*. Springer. Berlin, Heidelberg, New York 2005.

4. Benenson Y., Paz-Elizur T., Adar R., Keinan E., Livneh Z., Shapiro E.: *Programmable and autonomous computing machine made of biomolecules*. Nature 414, 430-434 (2001).

5. Benenson Y., Adar R., Paz-Elizur T., Livneh Z., Shapiro E.: *DNA molecule provides a computing machine with both data and fuel.* PNAS 100, 2191-2196 (2003).

6. Benenson Y., Gil B., Ben-Dor U., Adar R., Shapiro E.: *An autonomous molecular computer for logical control of gene expression*. Nature 429, 423-429 (2004).

7. Bennett C.: *Logical reversibility of computation*. IBM Journal of Research and Development 17, 525-532 (1973).

8. Bennett C.: *The thermodynamics of computation – a review*. International Journal of Theoretical Physics 21, 905-940 (1982).

9. Cavaliere M., Jonoska, N., Yogev, S., Piran, R., Keinan, E., Seeman, N.: *Biomolecular implementation of computing devices with unbounded memory.* Proceedings 10th International Workshop on DNA Computing, Springer-Verlag, New York. Lecture Notes in Computer Science 3384, 35-49 (2005).

10. Chen P., Jing L., Jian Z., Lin H., Zhizhou Z.: *Differential dependence on DNA ligase of type II restriction enzymes: a practical way toward ligase-free DNA automaton*. Biochemical and biophysical research communications 353, 733-737 (2007).

11. Faulhammer D., Cukras A., Lipton R., Landweber L.: *Molecular computation: RNA solutions to chess problems*. PNAS 97, 1385-1389 (1999).

12. Hagiya M.: *From molecular computing to molecular programming.* Proceedings 6th international Workshop on DNA-based computers. Springer Verlag, Berlin Heidelberg. Lecture Notes in Computer Science 2054, 89-102 (2001).

13. Head T.: *Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behavior.* Bulletin of Mathematical Biology 49, 737-759 (1987).

14. Hopcroft J., Ullman J.: *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley 1979.

15. Ignatova Z., Martinez-Perez I., Zimmermann K.: *DNA computing models*. Springer. Berlin, Heidelberg, New York 2008.

16. Janczak T., Malinowski A., Mulawka J., Nowak R.: *DNA computing - promise for information processing.* Universitatis Jagiellonicae Acta Informaticae, 113-130 (2000).

17. Krasiński T.: *Automaty i języki formalne*. Wydawnictwo Uniwersytetu Łódzkiego. Łódź 2007.

18. Krasiński T., Sakowski S.: *A theoretical model of the Shapiro finite state automaton built on DNA*. Theoretical and Applied Informatics 18, 161-174 (2006).

19. Krasiński T., Sakowski S. : *Extended Shapiro Finite State Automaton*. Foundations of Computing and Decision Science 33, 241-255 (2008).

20. Krasiński T., Sakowski S.: *Przegląd modeli i praktycznych implementacji DNA obliczeń*. Studia Informatica 29, 5-31 (2008).

21. Lipton R.: *DNA solution of hard computational problems*. Science 268, 542-545 (1995).

22. Mao C., LaBean T., Reif J., Seeman N. *Logical computation using algorithmic self-assembly of DNA triple-crossover molecules*. Nature 407, 493-496 (2000).

23. Nowak R., Mulawka J,, Płucienniczak A.: *Molecular associative memory built on DNA.* Proceedings SPIE Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments IV, 52-959 (2006).

24. Ogihara M., Ray A.: *Simulating Boolean circuits on a DNA compute*r. Proceeding First Annual International Conference on Computational Molecular Biology, New York, 326-331 (1997).

25. Păun G., Rozenberg G., Salomaa, A.: *DNA Computing. New Computing Paradigms*. Springer. Berlin, Heidelberg, New York 1998.

26. Ran T., Kaplan S., Shapiro E.: *Molecular implementation of simple logic program*. Nature Nanotechnology 10, 642-648 (2009).

27. Reif J.: *Paradigms for biomolecular computation*. Unconventional models of computation. Springer-Verlag, 72-93 (1998).

28. Rothemund P.: *A DNA and restriction enzyme implementation of Turing machines*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 27. American Mathematical Society, 75-120 (1995).

29. Rothemund P., Winfree E.: *The program-size complexity of self-assembled squares.* Proceedings Third-Second Annual ACM Symposiom on Theory of Computing, ACM Press, 459-468 (1999).

30. Sakakibara Y., Suyama A.: *Intelligent DNA chips: logical operation of gene expression profiles on DNA computers.* Proceedings 11th Workshop on Genome Informatics, 33-42 (2000).

31. Sakamoto K., Kiga D., Komiya K., Gouzu H., Yokoyama S., Ikeda S., Sugiyama H.,Hagiya M.: *State transitions by molecules.* Biosystems 52, 81-91 (1999).

32. Sakamoto K., Gouzu H., Komiya K., Kiga D., Yokoyama S., Yokomori T., Hagiya M.: *Molecular computation by DNA hairpin formation*. Science 288, 1223-1226 (2000).

33. Seeman N.: *DNA engineering and its application to nanotechnology*. Trends in Biotechnology 17, 37-443 (1999).

34. Seeman N.: *DNA Nicks and Nodes and Nanotechnology*. Nano Letters 1, 22-26 (2001).

35. Soreni M., Yogev S., Kossoy E., Shoham Y., Keinan E.: *Parallel biomolecular computation on surfaces with advanced finite automata*. Journal of the American Chemical Society 127, 3935-3943 (2005).

36. Stojanovic M., Stefanovic D.: *A Deoxyribozime-Based Molecular Automaton.* Nature Biotechnology 21, 1069-1074 (2003).

37. Soloveichik D., Winfree E.: *The Computational Power of Benenson Automata*. Theoretical Computer Science 344, 279-297 (2005).

38. Stryer L., Tymoczko J., Berg J.: *Biochemia.* PWN, Warszawa 2005.

39. Unold O., Troć M.: *Restriction Enzyme Computation*. Proceedings 7th International Work-Conference on Artificial and Natural Neural Networks, IWANN 2003, Lecture Notes in Computer Science 2686, 686–693 (2003).

40. Unold O., Troć M., Dobosz T., Trusiewicz A.: *Extended molecular computing model*. WSEAS Transactions on Biology and Biomedicine 1, 15-19 (2004).

41. Wąsiewicz, P., Janczak, T., Mulawaka, J., Płucienniczak, A.: *The inference based on molecular computing*. Cybernetics and Systems 31, 283-315 (2000).

42. Wąsiewicz P., Mulawka J.: *Molecular Genetic Programming*. Soft Computing, Springer-Verlag 5, 106-113 (2001).

43. Wąsiewicz P., Malinowski A., Nowak R., Mulawka J., Borsuk P., Weglenski P., Płucienniczak A.: *DNA computing: Implementation of data flow logical operations*. Future Generation Computer Systems 17, 361-378 (2002).

44. Winfree E., Liu F., Wenzler L., Seeman N.: *Design and self-assembly of twodimensional DNA crystals*. Nature 394, 539-544 (1998).

45. Węgrzyn S.: *Informatic systems (technical, biological)*. Bulletin of the Polish Academy of Sciences. Technical Sciences 49, 643-646 (2001).

46. Węgrzyn S., Klamka J., Bugajski S., Gibas M., Winiarczyk R., Znamirowski L., Miszczak J., Nowak S.: *Nano i kwantowe systemy informatyki*. Wydawnictwo Politechniki Śląskiej, Gliwice 2004.

47. Węgrzyn S., Klamka J., Znamirowski L., Winiarczyk R, Nowak S.: *Nano and quantum systems of informatics*. Bulletin of the Polish Academy of Sciences. Technical Sciences 52, 1-10 (2004).

48. Węgrzyn S., Znamirowski L.: *Zarys nanonauki i informatycznych molekularnych nanotechnologii*. Wydawnictwo Politechniki Śląskiej, Gliwice 2007.

49. Yokomori T.: *Molecular computing paradigm – toward freedom from Turing's charm.* Natural Computing 1, 333-390 (2002).

50. Yurke B., Turberfield A., Mills A., Simmel F., Neumann J.: *A DNAfuelled molecular machine made of DNA.* Nature 406, 605-608 (2000).

Dodatek A

Łańcuchy DNA reprezentujące elementy automatu (przed klonowaniem w wektor plazmidowy):

1. Łańcuch DNA reprezentujący słowo *x=ab*:

```
5′-TAACTGAAGTCAATCTAAAGTATCGGCTGATAATTGGGAGCAA-3′
3′-ATTGACTTCAGTTAGATTTCATAGCCGACTATTAACCCTCGTT-5′
```

2. Łańcuch DNA reprezentujący przejście T66:

```
5′-ACTCAAAGGCGGTAATACGGTTATCCACAGCTGAAGGTCTCCGCTG-3′
3′-TGAGTTTCCGCCATTATGCCAATAGGTGTCGACTTCCAGAGGCGAC-5′
```

3. Łańcuch DNA reprezentujący przejście T67:

```
5′-ATCAGGGGATAACGCAGGAAAGAACATGTGCAGCGCAATGCG-3′
3′-TAGTCCCCTATTGCGTCCTTTCTTGTACACGTCGCGTTACGC-5′
```

4. Łańcuch DNA reprezentujący Term 5:

```
5′-GCGTTTTTCCATAGGCTCCGCCCCCCTGACGAGCATCACAAAAATCGACGCTCAAGTCAGAGGTGGCGAAGCAATGTT-3′
3′-CGCAAAAAGGTATCCGAGGCGGGGGGACTGCTCGTAGTGTTTTTAGCTGCGAGTTCAGTCTCCACCGCTTCGTTACAA-5′
```

Fosforylacja końców oligonukleotydowych *kinazą* polinukleotydową oraz wytworzenie dwuniciowych fragmentów do klonowania.

Protokół:

1. w oddzielnych probówkach typu Eppendorf umieścić po 100 pM oligonukleotydów reprezentujących poszczególne elementy automatu,
2. dodać 100 pM ATP oraz 20 u kinazy T4,
3. reakcję fosforylacji prowadzić w temperaturze 37°C przez 30 minut,
4. oczyścić mieszaniną fenol:chloroform:alkohol izoamylowy (25:24:1),
5. komplementarne oligonukleotydy umieścić w probówce typu Eppendorf w buforze TE (Tris-EDTA, pH 8,0),
6. inkubować w temp. 95°C przez 15 minut, a następnie stopniowo (1°C/1 min) schłodzić do temperatury pokojowej.

Dodatek B

Wyniki sekwencjonowania dla poszczególnych plazmidów podajemy poniżej (tylko łańcuchy w kierunku 5'→3'):

1) Plazmid pPSAB - zawiera fragment kodujący słowo *x=ab*.

GGGACTTATCTACGAGATGGCTCGAGTTTTTCAGCTAGATTTGCTCCCAATTATCAGCCGATACTTTAACTGAACTC
AATCTAAAGTATCGGCTGATAATTGAGAGCAAAACTGAAGTCAATCTAAAGTATCGGCTGATAATTGGGAGCAATAA
CTGAAGTCAATCTAAAGTATCGGCTGATAATTGGGAGCAAATCTTTCTAGAAGATCTCCTACAATATTCTCAGCTGC
CATGGAAAATCGATGTTCTTCTTTTATTCTCTCAAGATTTTCAGGCTGTATATTAAAACTTATATTAAGAACTATGC
TAACCACCTCATCAGGAACCGTTGTAGGTGGCGTGGGTTTTCTTGGCAATCGACTCTCATGAAAACTACGAGCTAAA
TATTCAATATGTTCCTCTTGACCAACTTTATTCTGCATTTTTTTTGAACGAGGTTTAGAGCAAGCTTCAGGAAACTG
AGACAGGAATTTTATTAAAAATTTAAATTTTGAAGAAAGTTCAGGGTTAATAGCATCCATTTTTTGCTTTGCAAGTT
CCTCAGCATTCTTAACAAAAGACGTCTCTTTTGACATGTTTAAAGTTTAAACCTCCTGTGTGAAATTATTATCCGCT
CATAATTCCACACATTATACGAGCCGGAAGCATAAAGTGTAAAGCCTGGGGTGCCTAATGAGTGAGCTAACTCACAT
TAATTGCGTTGCGCTCACTGCCAAATTGCTTTCCAGTCGGGAAACCTGTCGTGCCAGCTGCATTAATGAAATCGGCC
AACGCGCGGGGAGAGGCGGTTTGCGTATTGGGCGCTCTTCCGCTTCCTCGCTCACTGACTCGCTGCGCTCGGTCGTT
CGGCTGCGGCGAGCGGTATCAGCTCACTCAAAGGCGGTAATACGGTTATCCACAGAATCAGGGGATAACGCAGAAAA
GAACATGTGAGCAAAGGCCAGCAAAAGGCCAGGAACCGTAAAAAGGCCGCGTTGCTGGCGTTTTTCATAGCTCCGCC
CCCCTGACGAGCATCACAAAAATCGACGCTCAAGTCAGAGTTGCGAAACCCGACAGGACTATAAAGAATACAGGCGT
TTCCCCTGGAGCTCCCTCGTGCGCTCTCCTGTTCCGACCCTGCCGCTTACCGATACCTGTCCGCCTTTCTCCCTTCG
GAAAGCGTTGCGCCTTTCCTCATAGCTCACCGCTTGTAAGGTATCTCCAT

2) Plazmid pPST66 - zawiera fragment kodujący przejście T66.

GGGACGTATTTTCGGATGGCTCGAGTTTTTCAGCAAGATCAGCGGAGACCTTCAGCTGTGGATAACCGTATTACCGC
CTTTGAGTCAGCGGAGACCTTCAGCTGTGGATAACCGTATTACCGCCTTTGAGTATCTTTCTAGAAGATCTCCTACA
ATATTCTCAGCTGCCATGGAAAATCGATGTTCTTCTTTTATTCTCTCAAGATTTTCAGGCTGTATATTAAAACTTAT
ATTAAGAACTATGCTAACCACCTCATCAGGAACCGTTGTAGGTGGCGTGGGTTTTCTTGGCAATCGACTCTCATGAA
AACTACGAGCTAAATATTCAATATGTTCCTCTTGACCAACTTTATTCTGCATTTTTTTTGAACGAGGTTTAGAGCAA
GCTTCAGGAAACTGAGACAGGAATTTTATTAAAAATTTAAATTTTGAAGAAAGTTCAGGGTTAATAGCATCCATTTT
TTGCTTTGCAAGTTCCTCAGCATTCTTAACGAAAGAGCTCACTTTTGACTGGTATGAAGTTTAGACCTCCTGCGGGA
GATTATTATCAGGTCATATCTCCAGACATTATACTTACCTTCAACGATAGAGAGTTCAGCCTGAGGCGCCTACAGAG
TGAACTAGATATTTTCTTGTGCCGCGTGCACACTACTCTGTGCTGGCCAGTCATGATACAAGTCATGCCAGCTGTAT
AACTGAAGGCTGGTAATGGCCGGGGACGGTATCTGCTGTAATGAGTGCTACTTCCGCTTTGTCTGCACTGAGGTACT
ACGTGAGCGATGTCGCGATGGAAGATCGCTCAGTACCACATCGCCACGATGCATCTGTTTGCTCCGCTATCCCAGGA
TAGACCCTTGCCGGAACGCAGCGACATCATGAACTTCCGACAGCTGCTAGAGCACATCGCTGGCCCGCCGACTGTTT
AACGACAATCAATCGCTCGCTGACCAGAAGCACGCGAATGATGCTCAGTGCACGTGGTCGAATGCCAACATGATGTA
GCAACCAGCTCGACTACAACTAAGAGCTCCACGCGATCCGGCAATCTCAAACTACGAATGGCATAATCGGGTCCTTC
GTTATGACGGTCAGATAGGTGTCGATGCCAAAATGGCCTGACCTAGACCTGACTCACGGCAGCCATTGCATGACCTC
AGTCCGCGGGTAATCTGCTGCAGGAAGCGAACAT

3) pPST67 - zawiera fragment kodujący przejście T67.

TGGCTCGAGTTTTTCAGCAAGATATCAGGGGATAACGCAGGAAAGAACATGTGCAGCGCAATGCGATCTTTCTAGAA
GATCTCCTACAATATTCTCAGCTGCCATGGAAAATCGATGTTCTTCTTTTATTCTCTCAAGATTTTCAGGCTGTATA
TTAAAACTTATATTAAGAACTATGCTAACCACCTCATCAGGAACCGTTGTAGGTGGCGTGGGTTTTCTTGGCAATCG
ACTCTCATGAAAACTACGAGCTAAATATTCAATATGTTCCTCTTGACCAACTTTATTCTGCATTTTTTTTGAACGAG
GTTTAGAGCAAGCTTCAGGAAACTGAGACAGGAATTTTATTAAAAATTTAAATTTTGAAGAAAGTTCAGGGTTAATA
GCATCCATTTTTTGCTTTGCAAGTTCCTCAGCATTCTTAACAAAAGACGTCTCTTTTGACATGTTTAAAGTTTAAAC
CTCCTGTGTGAAATTATTATCCGCTCATAATTCCACACATTATACGAGCCGGAAGCATAAAGTGTAAAGCCTGGGGT
GCCTAATGAGTGAGCTAACTCACATTAATTGCGTTGCGCTCACTGCCAATTGCTTTCCAGTCGGGAAACCTGTCGTG
CCAGCTGCATTAATGAATCGGCCAACGCGCGGGGAGAGGCGGTTTGCGTATTGGGCGCTCTTCCGCTTCCTCGCTCA

CTGACTCGCTGCGCTCGGTCGTTCGGCTGCGGCGAGCGGTATCAGCTCACTCAAAGGCGGTAATACGGTTATCCACA
GAATCAGGGGATAACGCAGAAAGAACATGTGAGCAAAAGGCCAGCAAAAGGCCAGGAACCGTAAAAAGGCCGCGTTG
CTGGCGTTTTTCCATAGGCTCCGCCCCCCTGACGAGCATCACAAAAATCGACGCTCAAGTCAGAGGTGGCGAAACCC
GACAGGACTATAAAGATACCAGGCGTTTCCCCCTGGAA

4) Plazmid pPSTER - zawiera fragment kodujący sekwencję terminalną Term5.

GGCCCGATCTCGGATGGCTCGAGTTTTTCAGCAAGATAACATTGCTTCGCCACCTCTGACTTGAGCGTCGATTTTTG
TGATGCTCGTCAGGGGGGCGGAGCCTATGGAAAAACGCATCTTTCTAGAAGATCTCCTACAATATTCTCAGCTGCCA
TGGAAAATCGATGTTCTTCTTTTATTCTCTCAAGATTTTCAGGCTGTATATTAAAACTTATATTAAGAACTATGCTA
ACCACCTCATCAGGAACCGTTGTAGGTGGCGTGGGTTTTCTTGGCAATCGACTCTCATGAAAACTACGAGCTAAATA
TTCAATATGTTCCTCTTGACCAACTTTATTCTGCATTTTTTTTGAACGAGGTTTAGAGCAAGCTTCAGGAAACTGAG
ACAGGAATTTTATTAAAAATTTAAATTTTGAAGAAAGTTCAGGGTTAATAGCATCCATTTTTTGCTTTGCAAGTTCC
TCAGCATTCTTAACAAAAGACGTCTCTTTTGACATGTTTAAAGTTTAAACCTCCTGTGTGAAATTATTATCCGCTCA
TAATTCCACACATTATACGAGCCGGAAGCATAAAGTGTAAAGCCTGGGGTGCCTAATGAGTGAGCTAACTCACATTA
ATTGCGTTGCGCTCACTGCCAATTGCTTTCCAGTCGGGAAACCTGTCGTGCCAGCTGCATTAATGAATCGGCCAACG
CGCGGGGAGAGGCGGTTTGCGTATTGGGCGCTCTTCCGCTTCCTCGCTCACTGACTCGCTGCGCTCGGTCGTTCGGC
TGCGGCGAGCGGTATCAGCTCACTCAAAGGCGGTAATACGGTTATCCACAGAATCAGGGGATAACGCAGAAAGAACA
TGTGAGCAAAAGGCCAGCAAAAGGCCAGGAACCGTAAAAAGGCCGCGTTGCTGGCGTTTTTCCATAGGCTCCGCCCC
CCTGACGAGCATCACAAAAATCGACGCTCAAGTCAGAGGTGGCGAAACCCGACAGGACTATAAAGATACCAGCGTTT
CCCCCTGGAAAGCTCCCTCGTGCGCTCTCCTGTTCCGACCCTGCCGCTTACCGGATACCTGTCCGCTTTCTCCCTTC
GGGAAGCGTGGCGCTTTCTCATAGCTCCACGCTGTAGTATCTCAGTTCGTGTAGTCGTCGCTCAGCTGGGCTGGTGT
GCACGAACCCCCGTTCAGCCGACGCTGCGCCTA

Dodatek C

Klonowanie fragmentów DNA w wektor plazmidowy zestawem *CloneJET™ Cloning Kit*.

Protokół:

1. dodać 10 μl buforu reakcyjnego do probówki typu Eppendorf,
2. dodać 1 μl wektora plazmidowego pJET 1.2/blunt (50 ng/μl),
3. dopełnić roztwór do 19 μl wodą wolną od nukleaz,
4. dodać 5 u T4 DNA *ligazy*,
5. inkubować mieszaninę w temperaturze pokojowej (22°C),
6. mieszaninę użyć do transformacji *E. coli*.

Transformacja *E. coli* przy zastosowaniu zestawu *CloneJET™*.

Protokół:

1. przygotować płytki agarowe LB zawierające ampicylinę, a następnie podgrzać je do temperatury 37°C przez 20 minut,
2. przygotować kompetentne komórki bakteryjne *E.coli* zgodnie z zaleceniami protokołu *TransformAid™ Bacterial Transformation Kit*,
3. przenieść 2,5 μl mieszaniny zawierającej wektory plazmidowe do probówki typu Eppendorf,
4. dodać 50 μl kompetentnych komórek bakteryjnych *E. coli*,
5. inkubować przez noc w temperaturze 22°C.

Dodatek D

Izolacja i pozyskiwanie plazmidów o wysokiej czystości z hodowli bakterii (*Gene MATRIX Plasmid Miniprep DNA Purification Kit*).

Protokół:

1. zwirować 1,5 ml nocnej hodowli bakteryjnej przy ok. 14000 rpm,
2. zawiesić osad bakteryjny w 250µl buforu do zawieszania *CeIIR*,
3. dodać 200µl buforu lizującego *Lysis Blue*.
4. dodać 350µl buforu *Neutral-B* i wirować w przez 7 minut z prędkością ok. 14000 rpm,
5. wlać supernatant do mikrokolumny i wirować 1 minutę z prędkością 12000 rpm,
6. wyjąć mikrokolumnę, wylać przesącz i powtórnie umieścić mikrokolumnę w probówce,
7. dodać 500µl buforu *PX* i wirować przez 1 minutę z prędkością 12000rpm,
8. wylać przesącz i powtórnie umieścić mikrokolumnę w probówce,
9. dodać 650µl buforu *Wash-PX* i wirować przez 1 minutę z prędkością 12000 rpm,
10. wylać przesącz i wirować 2 minuty z prędkością12000 rpm,
11. mikrokolumnę umieścić w nowej probówce. Dodać 50-100 µl buforu *Elution PX*,
12. mikrokolumnę umieścić na 2 minuty w temperaturze pokojowej i wirować przez 1 min. z prędkością 12000 rpm.

SKRÓTY i SYMBOLE

A	adenina (*adenine*)
ATP	trifosforan adenozyny (*adenosine triphosphorate*)
AZS	automat ze stosem
C	cytozyna (*cytosine*)
DAS	deterministyczny automat skończony
DNA	kwas dezoksyrybonukleinowy (*deoxyribonucleic acid*)
dsDNA	dwuniciowy DNA (*double stranded deoxyrybonukleoid acid*)
dsRNA	dwuniciowy RNA (*double stranded rybonukleoid acid*)
E. coli	bakteria *Escherichia coli*
EDTA	wersenian sodowy
G	guanina (*guanine*)
GF	gramatyka frazowa
GK	gramatyka kontekstowa
GBK	gramatyka bezkontekstowa
GR	gramatyka regularna
JRP	języki rekursywnie przeliczalne
JK	języki kontekstowe
JR	języki regularne
JBK	języki bezkontekstowe
mRNA	informacyjny RNA (*messenger* RNA)
MCS	miejsce klonowania (*multiple cloning site*)
NAS	niedeterministyczny automat skończony
PCR	łańcuchowa reakcja polimerazy (*Polymerase Chain Reaction*)
pz	para zasad

rRNA	rybosomowe RNA (*ribosomal* RNA)
SDS	sól sodowa siarczanu dedycylu
ssRNA	jednoniciowy RNA (*single stranded rybonukleoid acid*)
siRNA	krótkie dwuniciowe RNA (*small interfering rybonukleoid acid*)
SAT	problem spełnialności (*satisfability problem*)
ssDNA	jednonicowy DNA (*single stranded deoxyrybonukleoid acid*)
SOC	podłoże do hodowli bakterii
tRNA	transportujący RNA (*transfer* RNA)
RNA	kwas rybonukleinowy (*rybonucleic acid*)
T	tymina (*thymine*)
TEMED	N, N, N', N'- tetrametylenodiamina
TRIS	*2-amino-2-(hydroksynetylo)-1-3-propandiol*
TM	maszyna Turinga
U	uracyl (*uracil*)