

Detecting and classifying mutations in genetic code with an application to β -thalassaemia

Rapin Sunthornwat^a, Elvin J. Moore^{a,*}, Yaowadee Temtanapat^b

^a Department of Mathematics, King Mongkut's University of Technology North Bangkok, Bangkok 10800, Thailand

^b Department of Computer Science, Thammasat University, Phatumthani 12121, Thailand

*Corresponding author, e-mail: elvinmoo@gmail.com, ejm@kmutnb.ac.th

Received 19 Mar 2010

Accepted 26 Jan 2011

ABSTRACT: β -thalassaemia is a common disease in peoples of the Northern and Northeastern regions of Thailand. The general causes of the disease are mutations in the β -globin gene. In this paper we discuss methods based on finite-state automata theory, regular expressions, and partially ordered sets that can be used for detecting and classifying mutations in genes. The methods are applied to three main problems. The first problem is the reliable detection in a given β -globin gene of mutations that are known to cause β -thalassaemia. It is shown that for most known mutations in Thailand a 5-base pattern is an optimal size search pattern for reliably detecting if that mutation is or is not present in a given β -globin gene. The second problem is to find all differences between a standard normal β -globin gene and a suspected abnormal gene, to identify any differences as point mutations or frameshift mutations, and to list important biochemical effects of the mutations. The third problem is to list the types of bases in a given gene pattern as purines or pyrimidines and the types of codons as weak (4 hydrogen bonds), mixed (5 hydrogen bonds), or strong (6 hydrogen bonds). Fast and easy-to-use Matlab programs have been developed for each of these three problems. The programs could be useful when large commercial packages are not available.

KEYWORDS: point mutations, frameshifts, binary codes, finite-state automata, regular expressions

INTRODUCTION

β -thalassaemia is a very common disease in Thailand¹⁻³. It is a varied group of disorders of haemoglobin (Hb) synthesis, most of which result from point mutations within the β -globin gene or the immediate flanking sequence. It is often a hereditary disease. More than 200 different β -thalassaemia mutations have now been characterized worldwide, but only approximately 20 mutations are common in Thailand^{2,3}.

Methods for classifying and analysing patterns in genetic code have important applications. Genetic code consists of sequences of four nucleotides (bases): adenine (A), cytosine (C), guanine (G), and thymine (T) in DNA and A, C, G and uracil (U) in RNA which are arranged into codons, where a codon is a triplet of bases which codes for the production of an amino acid⁴⁻⁶. Many authors have suggested different coding schemes for the DNA and RNA bases and codons and different methods for analysing symmetry in genetic codes. Many of these coding schemes are based on binary codes. The coding schemes used by authors depend on the applications in which they are

interested. For example, Wilhelm and Nikolajewa⁷⁻¹⁰ have proposed a binary code based on amino acid properties, such as purine or pyrimidine and number of hydrogen bonds, in order to obtain a simple symmetry classification of the codons and to study the historical evolution of the genetic code. Sanchez, Morgado and Grau¹¹ have used a binary code based on Boolean algebra, partially ordered sets, and Hasse diagrams¹² to analyse the hydrophobic properties of proteins produced from different sequences of codons and to analyse if mutations in a gene are likely to cause minor or major differences in the function of the gene. Jimenez et al¹³ have discussed a hypercube structure to explain conservative and non-conservative amino acid substitutions. Petoukhov¹⁴ and He¹⁵ have developed three different coding schemes in order to study symmetry properties of the genetic code and the relationship between the degeneracy of the code and the positions of the bases in the codons. Authors have also analysed the genetic codes as circular codes. A review of circular codes in genes is given in Ref. 16.

There are many mathematical techniques that have been used for detecting and matching patterns in genetic code. These techniques include regu-

lar expressions and finite-state automata theory¹⁷⁻¹⁹, Boolean algebra and partially ordered sets^{11,12}, and group theory²⁰. Andrzej²¹ and Nowzari et al²² have also used DNA to implement finite-state automata.

The structure of this paper is as follows. In the second section, we summarize information about the main mutations that are responsible for β -thalassaemia in Thailand. In the third section, we discuss the properties of some of the many different coding schemes that have been proposed for the analysis of symmetry properties of the genetic codes and for the detection and classification of mutations. In the fourth section, we briefly describe the mathematical methods based on regular expressions, finite-state automata theory, and partially ordered sets that can be used to detect and classify mutations. In the following three sections, we discuss the three different algorithms and Matlab programs that we have developed to detect and classify mutations and we summarize the results obtained from them. The final section of the paper contains a discussion and conclusions.

MUTATIONS AND β -THALASSAEMIA IN THAILAND

A β -globin gene is a sequence of nucleotides that stores genetic information involved in the synthesis of the haemoglobin protein. There are many errors that can occur in protein synthesis, including errors (mutations) in the original DNA code and copying errors in the DNA to mRNA, mRNA to tRNA, and tRNA to protein stages. In many common types of mutations and copying errors, the symmetry properties of the genetic code are important. A detailed review of the errors that can occur in the synthesis of a protein from a DNA code can be found in Ref. 23.

As stated above, at least 200 mutations associated with β -thalassaemia have been reported world wide. However, within each population at risk for β -thalassaemia only a small number of common mutations are usually found³. A list of 24 of the main mutations found in Thailand can be found in Refs. 1, 2. Table 1 shows the DNA sequence in a normal β -globin gene and some of the mutations that occur in the Thai population¹⁻³. The codon positions listed in Table 1 refer to the standard codon numbering for a β -globin gene^{24,25}. The data in Table 1 will be used as examples for our detection and classification algorithms.

The mutations shown in Table 1 can be classified into four groups as follows^{4,5}.

(1) *Transcription mutation*. This is a point mutation that arises from an error in the transcription process which is written back as a permanent mutation into a

new DNA strand. A mutation of this type is the change of base from A to G at position -28(A→G).

(2) *Nonfunctional mRNA*. This can be caused by a mutation in which a codon is replaced by a stop codon which stops protein synthesis before it is complete. A mutation of this kind is the change of base from A to T in codon 17(A→T).

(3) *Frameshift mutation (frameshift or framing error)*. In this mutation a number of bases not divisible by 3 may be added to or deleted from the DNA strand. A frameshift causes the reading of subsequent codons to be changed, so that most codons after the mutation will code for different amino acids. Any resulting proteins will probably not be functional because they could be abnormally short, abnormally long, and/or contain the wrong amino acids. An example of a frameshift is the deletion of 4 bases at codons 41-42 (-CTTT).

(4) *mRNA processing mutation*. This mutation occurs when an intron is cut. An intron is a base sequence within a gene that is transcribed but is then excised from the mRNA before translation into a protein. A cut in an intron can cause an untranslated sequence in the transcription process to be wrong. For example, a mutation at base one of the intron which is labelled (IVSInt1, G→T) gives an inserted base near a cut point (consensus change).

SYMMETRY IN THE GENETIC CODE

In the standard coding pattern found in most, but not all⁸, living organisms, 20 distinct amino acids are coded for by the 64 possible codons. A table showing this standard coding pattern and biochemical properties of the codons is given in Table 2. The data in this table is used in our programs to associate any mutations detected with their biochemical effects.

As stated in the introduction, many different types of coding have been proposed for the DNA and RNA nucleotides and codons. In this paper, we use three different types of coding. We will use the simple coding in which each of the four bases is represented by the single letter A,T,C,G for DNA or A,U,C,G for RNA to detect mutations by direct comparison of the sequence of bases in a 'normal' gene and an 'abnormal' gene. We use the binary coding scheme of Ref. 11 in classifying the types of mutation that cause β -thalassaemia in Thailand, and the binary coding scheme of Refs. 7, 8, 10 in the classification of biochemical properties of codons.

Table 1 DNA mutations in β -globin gene causing β -thalassaemia.

	5'-	TCCTAAGCCA	GTGCCAGAAG	AGCCAAGGAC	AGGTACGGCT	GTCATCACTT	
	51	AGACCTCACC	CTGTGGAGCC	ACACCCTAGG	GTTGGCCAAT	CTACTCCCAG	
	101	GAGCAGGGAG	GGCAGGAGCC	AGGGCTGGGC	ATAAAAGTCA	GGGCAGAGCC	←(a)
	151	ATCTATTGCT	TACATTTGCT	TCTGACACAA	CTGTGTTTAC	TAGCAACCTC	
	201	AAACAGACAC	CATGGTGCAC	CTGACTCCTG	AGGAGAAGTC	TGCCGTTACT	
(b)→	251	GCCCTGTGGG	GCAAGGTGAA	CGTGGATGAA	GTTGGTGGTG	AGGCCCTGGG	←(c)
(d)(e)→	301	CAGGTTGGTA	TCAAGGTTAC	AAGACAGGTT	TAAGGAGACC	AATAGAAACT	
	351	GGGCATGTGG	AGACAGAGAA	GACTCTTGGG	TTTCTGATAG	GCACTGACTC	
	401	TCTCTGCCTA	TTGGTCTATT	TTCCCACCCT	TAGGCTGCTG	GTGGTCTACC	←(f)
(g)→	451	CTTGGACCCA	GAGGTTCTTT	GAGTCCTTTG	GGGATCTGTC	CACTCCTGAT	
	501	GCTGTTATGG	GCAACCCTAA	GGTGAAGGCT	CATGGCAAGA	AAGTGCTCGG	
	551	TGCCTTTAGT	GATGGCCTGG	CTCACCTGGA	CAACCTCAAG	GGCACCTTTG	
	601	CCACACTGAG	TGAGCTGCAC	TGTGACAAGC	TGCACGTGGA	TCCTGAGAAC	
	651	TTCAGGGTGA	GTCTATGGGA	CCCTTGATGT	TTTCTTTCCC	CTTCTTTTCT	
	701	ATGGTTAAGT	TCATGTCATA	GGAAGGGGAG	AAGTAACAGG	GTACAGTTTA	
	751	GAATGGGAAA	CAGACGAATG	ATTGCATCAG		-3'	

(a) -28(A→G); (b) 17(A→T); (c) 28(G→A)HbE; (d) IVSInt1(G→T); (e) IVSInt1(G→C); (f) 35(C→A); (g) 41-42(CTTT deletion) ATG in row 5 is the start codon Source: Refs. 1, 2

Table 2 Properties of amino acids.

Amino acid	Codon	H-bonds strength	Rings	Chemical properties	Binary code	Amino acid	Codon	H-bonds strength	Rings	Chemical properties	Binary code
Alanine	GCT	6-strong	4	NPL-HDPB	001110	Phenylalanine	TTT	4-weak	3	NPL-HDPB	101010
Alanine	GCC	6-strong	4	NPL-HDPB	001111	Phenylalanine	TTC	4-weak	3	NPL-HDPB	101011
Alanine	GCA	6-strong	5	NPL-HDPB	001101	Proline	CCT	6-strong	3	NPL-HDPB	111110
Alanine	GCG	6-strong	5	NPL-HDPB	001100	Proline	CCC	6-strong	3	NPL-HDPB	111111
Arginine	CGT	6-strong	4	PC	110010	Proline	CCA	6-strong	4	NPL-HDPB	111101
Arginine	CGC	6-strong	4	PC	110011	Proline	CCG	6-strong	4	NPL-HDPB	111100
Arginine	CGA	6-strong	5	PC	110001	Serine	TCT	5-mixed	3	PL-HDPL	101110
Arginine	CGG	6-strong	5	PC	110000	Serine	TCC	5-mixed	3	PL-HDPL	101111
Arginine	AGA	5-mixed	6	PC	010001	Serine	TCA	5-mixed	4	PL-HDPL	101101
Arginine	AGG	5-mixed	6	PC	010000	Serine	TCG	5-mixed	4	PL-HDPL	101100
Asparagine	AAT	4-weak	5	PL-HDPL	010110	Serine	AGT	5-mixed	5	PL-HDPL	010010
Asparagine	AAC	4-weak	5	PL-HDPL	010111	Serine	AGC	5-mixed	5	PL-HDPL	010011
Aspartic acid	GAT	5-mixed	5	NC	000110	Threonine	ACT	5-mixed	4	PL-HDPL	011110
Aspartic acid	GAC	5-mixed	5	NC	000111	Threonine	ACC	5-mixed	4	PL-HDPL	011111
Cysteine	TGT	5-mixed	4	PL-HDPL	100010	Threonine	ACA	5-mixed	5	PL-HDPL	011101
Cysteine	TGC	5-mixed	4	PL-HDPL	100011	Threonine	ACG	5-mixed	5	PL-HDPL	011100
Glutamic acid	GAA	5-mixed	6	NC	000101	Tryptophan	TGG	5-mixed	5	NPL-HDPB	100000
Glutamic acid	GAG	5-mixed	6	NC	000100	Tyrosine	TAT	4-weak	4	PL-HDPL	100110
Glutamine	CAA	5-mixed	5	PL-HDPL	110101	Tyrosine	TAC	4-weak	4	PL-HDPL	100111
Glutamine	CAG	5-mixed	5	PL-HDPL	110100	Valine	GTT	5-mixed	4	NPL-HDPB	001010
Glycine	GGT	6-strong	5	NPL-HDPB	000010	Valine	GTC	5-mixed	4	NPL-HDPB	001011
Glycine	GGC	6-strong	5	NPL-HDPB	000011	Valine	GTA	5-mixed	5	NPL-HDPB	001001
Glycine	GGA	6-strong	6	NPL-HDPB	000001	Valine	GTG	5-mixed	5	NPL-HDPB	001000
Glycine	GGG	6-strong	6	NPL-HDPB	000000	START	ATG	4-weak	5	NPL-HDPB	011000
Histidine	CAT	5-mixed	4	PC	110110	(Methionine)					
Histidine	CAC	5-mixed	4	PC	110111	STOP	TAA	4-weak	5	stop codon	100101
Isoleucine	ATT	4-weak	4	NPL-HDPB	011010	(Ochre)					
Isoleucine	ATC	4-weak	4	NPL-HDPB	011001	STOP	TAG	4-weak	5	stop codon	100100
Isoleucine	ATA	4-weak	5	NPL-HDPB	011001	(Amber)					
Leucine	TTA	4-weak	4	NPL-HDPB	101001	STOP	TGA	5-mixed	5	stop codon	100001
Leucine	TTG	4-weak	4	NPL-HDPB	101000	(Opal)					
Leucine	CTT	5-mixed	3	NPL-HDPB	111010	Notes:					
Leucine	CTC	5-mixed	3	NPL-HDPB	111011	NPL-HDPB=nonpolar-hydrophobic					
Leucine	CTA	5-mixed	4	NPL-HDPB	111001	PL-HDPL=polar-hydrophilic					
Leucine	CTG	5-mixed	4	NPL-HDPB	111000	PC=positively charged					
Lysine	AAA	4-weak	6	PC	010101	NC=negatively charged					
Lysine	AAG	4-weak	6	PC	010100	Binary code of Sanchez et al ¹¹					

Classifying biochemical properties of mutations using binary coding of Sanchez et al

Sanchez et al¹¹ developed a binary code and used the mathematics of Boolean algebra, Boolean lattices, partially ordered sets, and Hasse diagrams¹² as a means of classifying mutations by their effects on gene function. The key biochemical ideas they use are as follows. Firstly, the four bases can be divided into purines or pyrimidines having 2 or 3 hydrogen bonds. Secondly, they use the observation in Ref. 13 that the strength of internal bonding in a codon is determined in the order: $C_2 > H_2 > C_1 > H_1 > C_3 > H_3$, where C_i , $i = 1, 2, 3$ is the chemical type of the base in position i of a codon and H_i is the number of hydrogen bonds. Thirdly, they use the property that adenine (A) in position 2 gives a strongly hydrophilic amino acid, whereas uracil (U) in position 2 gives a strongly hydrophobic amino acid. They then use the idea that most mutations that can survive are likely to cause only minor modifications to the type of amino acid and to the physico-chemical properties of the protein that is produced. Major modifications are more likely to cause severe damage which could mean that the host would not survive.

Sanchez et al use the binary coding G=00, A=01, U,T=10, and C=11 and a partial ordering of these codes based on $0 < 1$ in the same position. Then $00(G) < 01(A) < 11(C)$ and $00(G) < 10(U) < 11(C)$ are ordered but $01(A)$ and $10(U)$ are not ordered (not comparable). In this coding the 'hydrophilic' base A and 'hydrophobic' base U are not comparable. Sanchez et al¹¹ also introduced a dual coding scheme based on the complementarity properties $C \leftrightarrow G$, $A \leftrightarrow U$ or T , in which 0 and 1 are interchanged but the ordering $0 < 1$ is retained. The primal (original) and dual schemes are useful for describing codon-anticodon symmetry and bonding between a base and its complement since two strands are complements if and only if the primal coding of one strand is the same as the dual coding of the other strand.

Sanchez et al combine their 2-bit binary codes for bases to develop partially ordered 6-bit binary codes with the partial ordering based on $0 < 1$ in the same positions of the code. Partially ordered sets can be represented by Hasse diagrams¹² in which a subset of totally ordered elements is called a chain. In the Hasse diagram for 6-bit codes of Sanchez et al¹¹, the hydrophobic-hydrophilic property of the corresponding ordered codons change slowly along a chain. If two codons are on the same chain, then they say that if codon 1 is less than codon 2 then codon 1 is a *deduction* from codon 2. In the Hasse diagram,

codons containing equal numbers of 0's in their codes are placed at the same level with the primary code for CCC (11111) at the lowest level and the primary code for GGG (00000) at the highest level. The codons in the same level are then arranged from left to right so that a more hydrophobic codon is at the left of a less hydrophobic codon. They note that the same Hasse diagram can be used to represent both the primal and dual coding schemes.

Sanchez et al¹¹ state that codon deductions in this partially ordered set of codes have physico-chemical and biological meanings. They reflect the role of polarity and hydrophobicity in protein evolution in such a way that deductions generally correspond to only small changes in the physico-chemical properties of amino acids. Sanchez et al examine many physico-chemical properties of amino acids and show that codons related through deductions typically correspond to smaller changes in properties than codons that are not related. Sanchez et al also note that in the mutations of the β -globin gene, single-point mutations are frequently related through deductions to the corresponding codons in a 'normal' gene and that the effect of a mutation is more severe when the codons are not related.

A simple method of testing whether two codons are related through deductions is to subtract the 6-digit binary codes. If all differences in the bits are 0 or +1 or all differences are 0 or -1 then the codes are related. If +1 and -1 both occur then the codes are not ordered (not comparable) and therefore not related through deductions. For example, CCU (00001) and GCU (11001) are related through deductions since $00001 - 11001 = 11000$, whereas CCU (00001) and GCA (11010) are not related since $00001 - 11010 = (-1)(-1)00(-1)1$.

We use the Sanchez et al coding scheme to detect and classify mutations in β -globin genes according to the type of mutation, changes in amino acids between the normal gene and the mutated gene, and properties such as whether they are hydrophilic or hydrophobic, number of H-bonds, number of rings, chemical properties, binary code, deduction or not deduction.

The binary coding of Wilhelm and Nikolajewa

As stated in the introduction, Wilhelm and Nikolajewa^{7,8,10} developed their binary code to obtain a simple symmetry classification of the codons and to study the historical evolution of the genetic code. Their code is C=00, G=10, A=11, and U or T=01. For the first digit, 0 represents a pyrimidine (C or T or U) and 1 represents a purine (G or A). For the second digit, 0 corresponds to strong bases (C

or G) and 1 corresponds to weak bases (A, T or U). In the six-digit binary code of a codon, 0's in odd positions correspond to purines, 1's in odd positions correspond to pyrimidines, 0's in even positions correspond to strong bases and 1's in even positions correspond to weak bases. Wilhelm and Nikola-jewa show that the recognized symmetries of the genetic code, namely family-nonfamily symmetry, Halitsky symmetry, complementary symmetry, purine-pyrimidine symmetry, codon-anticodon symmetry, codon-reverse codon symmetry, and sense-anticodon symmetry can be represented in a simple table of 8 rows with each row corresponding to the 8 possible purine-pyrimidine possibilities in a codon and 4 columns representing strong, strong-weak, weak-strong, or weak hydrogen bonding in the first 2 bases of a codon. Strong codons have strong bases (3 H-bonds, C or G) in positions 1 and 2, strong-weak codons have strong bases in position 1 and weak bases (2 H-bonds, A, T or U) in position 2, weak-strong codons have weak bases in position 1 and strong bases in position 2, and weak codons have weak bases in positions 1 and 2.

FINITE STATE AUTOMATA AND REGULAR EXPRESSIONS

In this section, we consider the application of regular expressions and finite-state automata^{17-19, 22} to searching for particular subsequences (substrings) of DNA in a DNA sequence and show how these methods can be used to detect mutations for β -thalassaemia in Thailand. We will describe an automata model for detecting mutations responsible for β -thalassaemia in a long sequence of code for the β -globin gene. We follow the method described in Ref. 17. We want to develop a DFA to find mutation patterns in a β -globin genetic code and report the position of these patterns.

Basic theory

A deterministic finite automaton (DFA) $A = \{Q, \Sigma, \delta, q_0, F\}$ consists of¹⁹: (1) a finite set of states Q ; (2) a finite set of input symbols (an alphabet) Σ ; (3) a transition function $\delta : Q \times \Sigma \rightarrow Q$; (4) a start state $q_0 \in Q$; (5) a set of final or accepting states $F \subseteq Q$.

For the genetic code, the alphabet is the set of four bases $\{A, T, C, G\}$ (DNA) or $\{A, U, C, G\}$ (RNA). For binary coding, the alphabet is the set $\{0, 1\}$. Standard methods of specifying a DFA are a transition diagram and a transition table. A transition diagram is a directed graph²⁶ showing the possible transitions and a transition table is a table showing the transitions for each state and each input symbol. An example of a

Table 3 Transition table for DFA for detecting pattern GCTAG.

	G	C	T	A
$\rightarrow q_0$	q_1	q_0	q_0	q_0
q_1	q_1	q_2	q_0	q_0
q_2	q_1	q_0	q_3	q_0
q_3	q_1	q_0	q_0	q_4
q_4	q_5	q_0	q_0	q_0
$* q_5$	q_5	q_5	q_5	q_5

transition table is shown in Table 3. A DFA will detect a search pattern of symbols from its alphabet if and only if that pattern causes a transition from the start state to any final state.

Regular expressions are widely used as a simple method of specifying search patterns. Many computer packages and operating systems (e.g., Microsoft Word, Matlab, Unix/Linux) contain programs for searching using regular expressions. These programs are based on a standard result of the theory of languages that every regular expression is equivalent to a DFA and every DFA is equivalent to a regular expression (Kleene's theorem²⁷).

Detection of mutations in DNA code of β -globin gene by using finite automata

An example of an abnormal genetic code pattern in a β -globin gene is shown in Table 1. The general steps in designing an automaton to find a particular mutation are as follows: (1) select a search pattern that uniquely identifies a particular mutation; (2) write a regular expression for this pattern¹⁹; (3) construct a λ -NFA (non-deterministic finite automaton with empty strings λ as acceptable input) corresponding to the regular expression¹⁹; (4) convert the λ -NFA to a DFA by the subset construction method¹⁹.

As we show in the next section, a suitable search pattern for uniquely detecting a particular mutation in a DNA sequence typically consists of 5 bases. For example, a suitable search pattern for uniquely detecting the mutation in Codon 17: A \rightarrow T, is GCTAG. This is the smallest-length pattern which does not occur in the normal code and which only appears in the mutated code at codon 17. The DFA for the pattern GCTAG can be represented by the transition table shown in Table 3. The alphabet for the DFA is $\Sigma = \{G, C, T, A\}$, the set of states is $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, the start state is q_0 and the final set of accepting states is $F = \{q_5\}$. The states are defined as: q_1 means the symbol G has been read, q_2 means the string GC has been read, q_3 means the string GCT has been read,

q_4 means *GCTA* has been read and the final state q_5 means the required pattern *GCTAG* has been found. The start state q_0 means that none of the previous patterns have been read.

We have written a Matlab program for this DFA based on the algorithm in Ref. 17. Although the DFA in Table 3 can only find one occurrence of a pattern, our program is designed to find any number of occurrences by restarting after a detection of the required pattern. We have tested this program on the mutated code sequence and corresponding normal code sequence given in Table 1. The tests show that the program correctly finds exactly one occurrence of the pattern in the abnormal code at the correct position and no occurrences of the pattern in the normal code.

OPTIMAL-SIZE SEARCH PATTERNS FOR UNIQUE DETECTION OF β -THALASSAEMIA MUTATIONS

For a given mutation, we define an optimal-size search pattern as the smallest-length pattern of genetic code that occurs exactly once in an abnormal DNA sequence if the mutation exists and does not occur in a DNA sequence if the mutation does not exist. In other words, we want a pattern that gives no ‘false positives’ and no ‘false negatives’.

As an example of the method of constructing a pattern, we consider the mutation in Codon 17: A→T. Using the data given in Table 1, we find that sections of the abnormal code that contain the mutation and the corresponding normal codes are as shown in Table 4.

The idea of the algorithm is that it systematically searches genetic codes for patterns of increasing length that contain the mutation until it terminates (1) successfully by finding a pattern that occurs exactly once in an abnormal code and that does not occur in the normal code, or (2) unsuccessfully by searching an abnormal code without finding the search pattern, or (3) unsuccessfully by reaching a maximum size of search pattern.

We usually initialize the search with a pattern of 3 bases and set the maximum allowed size of pattern to 6 bases. For the mutation in codon 17, the initial 3-base search pattern is CTA and the other possible patterns are GCT and TAG. The systematic search for patterns of length 4 searches in the order GCTA,

Table 4 Sections of abnormal and normal code containing A → T mutation at codon 17.

Abnormal code	... TGGGGCTAGGTGAA...
Normal code	... TGGGGCAAGGTGAA...

CTAG, GGCT, TAGG. The systematic search for patterns of length 5 searches in the order GCTAG, GGCTA, CTAGG, GGGCT, TAGGT.

A Matlab program has been written to implement the above algorithm. The searches in the Matlab program use the Matlab regular expression function *regexp*. We can summarize the results of the program as follows.

- (1) For search patterns with 3 symbols (bases) we find many occurrences of each pattern in both abnormal and normal codes.
- (2) For search patterns with 4 symbols, we again find many occurrences of each pattern in both abnormal and normal codes but fewer than for 3 symbols.
- (3) For search patterns with 5 symbols, we usually find one occurrence of the pattern if the mutation is present and 0 occurrences if the mutation is not present.
- (4) For search patterns with 6 symbols, we sometimes find 0 occurrences even if the mutation is actually present in the code. This can occur if there are two overlapping mutations or mutations near each other. For example, the mutations IVS1nt1(G-T) and IVS1nt(G-C) (see Table 1) are close to each other. In this case, using GTTGCT to search for IVS1nt(G-C) fails because the G-T mutation in IVS1nt1(G-T) has changed the abnormal code to TTTGCT.

A search pattern with 5 symbols is usually an optimal-size pattern. The optimal-size 5-base patterns for six of the common β -thalassaemia point mutations in Thailand are shown in Table 5.

DETECTING AND CLASSIFYING MUTATIONS IN β -GLOBIN GENES

In this section we use the simple coding in which the DNA bases are represented by C,G,A,T and the RNA bases by C,G,A,U. We have developed an algorithm and Matlab program that compares a normal and an abnormal genetic code and detects all point mutations and frameshifts in the abnormal code. The program then uses the data in Table 2 to construct a table

Table 5 Optimal size patterns for searching for common Thai mutations.

Mutation	Normal pattern	Abnormal pattern	Optimal pattern
Codon 17: (A-T)	GCAAG	GCTAG	GCTAG
Codon -28: (A-G)	ATAAA	ATGAA	ATGAA
Codon 28: (G-A)HbE	GTGAG	GTAAG	GTAAG
IVS1nt1: (G-T)	AGGTT	AGTTT	AGTTT
IVS1nt1: (G-C)	TGGTA	TGCTA	TGCTA
Codon 35: (C-A)	TACCC	TAACC	TAACC

comparing the biochemical properties of the normal and mutated codons.

The algorithm is given below. As it is quite long, we also give examples of the behaviour of parts of the algorithm.

The detection and classification algorithm

The inputs consist of the following. (1) Normal and abnormal β -globin genetic codes as unbroken strings of DNA base letters A,T,C,G.

(2) 5-base pattern CATGG required to identify the START codon ATG (methionine) in the normal β -globin code.

(3) Amino acid properties in Table 2.

The outputs consist of the following.

(1) *Codon positions in normal genetic code* containing the following information.

(a) Standard Genbank labels²⁴ for codons in normal β -globin gene.

(b) Codon numbering with first complete codon or part codon in normal code given label 1, second complete codon given label 2, etc. Note: Part codons can occur in the normal genetic code if there are 1 or 2 extra bases at the beginning or end of the code.

(c) Positions in normal genetic code of first bases of each codon.

(2) *List of point mutations* in abnormal code with allowance made for frameshifts. The following information is given.

(a) Normal codon. Triplet of bases of the normal codon, Genbank label, position of mutated base in normal genetic code.

(b) Mutated codon. Triplet of bases of the mutated codon, position of mutated base in abnormal genetic code.

(3) *List of frameshift deletions from and frameshift additions to abnormal code* with the following information.

(a) For frameshift deletions, bases present in normal genetic code but not in abnormal code. For frameshift additions, bases present in abnormal genetic code but not in normal code.

(b) Position of base in normal code at which the frameshift begins, and the Genbank label of the corresponding normal codon.

(c) Position of base in abnormal code at which the frameshift begins.

(4) *Detailed list of point mutations and their biochemical effects*. The information includes the following:

(a) Normal codon: position of mutated base in normal genetic code, Genbank label of codon, triplet of bases, amino acid, number H-bonds, number rings, biochemical properties, Sanchez et al¹¹ binary code.

(b) Mutated codon: triplet of bases, amino acid, number H-bonds, number rings, biochemical properties, Sanchez et al¹¹ binary code and deduction or not deduction.

The matching process is a systematic codon by codon comparison of normal and abnormal codes that searches for point mutations and possible frameshifts. The matching algorithm can be divided into initialization, a matching loop, and final processing. Initialization consists of the following steps. (1) Search for position of START codon in normal β -globin genetic code using pattern CATGG.

(a) If pattern is found, assign genbank labels and codon positions to normal code as described above under item 1 of outputs.

(b) If pattern is not found, assign triplets of bases to codons starting from the first base in the normal code.

(2) Set base position counter j_{norm} for the normal code to 1, and base position counter j_{abnorm} for abnormal code to 1.

The matching loop terminates when all bases in either the normal or the abnormal codes have been read. On termination of the loop, the algorithm continues with the final processing step. The steps in the matching loop are as follows.

(1) Read 'codon' (codon or part codon) starting at j_{norm} in the normal code and compare with corresponding number of bases starting at j_{abnorm} in abnormal code.

(2) If the normal and the abnormal 'codons' are the same, increment base counter j_{norm} in normal code and j_{abnorm} in abnormal code to first base positions of next codons and return to step 1 of matching loop.

(3) If the two codons are different, record base counter positions j_{norm} and j_{abnorm} . Continue to step 4.

(4) Test for point mutations and frameshifts by comparing 3 consecutive codons in normal and abnormal codes starting at positions j_{norm} and j_{abnorm} . Then,
(a) If less than 3 consecutive codons in the normal and abnormal codes are different, then differences are regarded as point mutations. Continue to step 5.

(b) If 3 consecutive codons are different, but less than 4 bases in these 3 codons are different, then differences are regarded as point mutations. Continue to step 5.

(c) Otherwise continue to step 6 to test for possible frameshift.

(5) For each point mutation do the following:

(a) Record the information listed in item 2 of outputs.

(b) Increment the base position counters j_{norm} and j_{abnorm} to the first base positions of the next unread codons in the normal and abnormal codes. Return to step 1 of matching loop.

(6) Test for possible frameshift as follows:
 (a) Find positions of the first bases that are different in the 3 consecutive normal and abnormal codons identified as different in step 4. Set frameshift base position counters $fbnorm$ in normal code and $fbabnorm$ in abnormal code to the positions of the first bases that are different.

(b) *Test for Frameshift Addition* in abnormal code. From the normal genetic code select a search pattern of five consecutive bases starting at position $fbnorm$. Then search forward in the abnormal code starting at $fbabnorm$ for this 5-base pattern. Then: (i) If pattern is found, record extra bases in the abnormal code as a possible frameshift addition. Continue to step (c) to test for frameshift deletion. (ii) If pattern is not found, continue to step (c).

(c) *Test for Frameshift Deletion* in abnormal code. From the abnormal code pattern, select five consecutive bases starting at position $fbabnorm$. Then search forward in the normal code starting at position $fbnorm$ for this 5-base pattern. Then: (i) If pattern is found, record extra bases in the normal code as a possible frameshift deletion in the abnormal code. Continue to step (d). (ii) If pattern is not found, continue to step (d).

(d) (i) If both a possible frameshift addition and a possible frameshift deletion have been found, then if frameshift addition is the shorter code, continue to step 7, but if frameshift deletion is the shorter code, continue to step 8. (ii) If only a frameshift addition is found, continue to step 7. If only a frameshift deletion is found continue to step 8. (iii) If neither addition nor deletion is found, then the differences in the 3 codons will be regarded as point mutations in 3 consecutive codons. Return to step 5.

(7) *Frameshift addition*. Do the following:

(a) Record the information listed in item 3 of outputs.
 (b) Reset the base position counter $jnorm$ for the normal pattern to the position of the first base in the next unread normal codon.

(c) Reset the base position counter $jabnorm$ to the base in the abnormal pattern corresponding to the base at $jnorm$ in the normal code pattern. Return to step 1 of the matching loop.

(8) *Frameshift deletion*. Do the following:

(a) Record the information listed in item 3 of outputs.
 (b) Reset the base position counter $jnorm$ for the normal code pattern by incrementing $jnorm$ to the position of the first base in the first normal codon after the frameshift deletion.

(c) Reset the base position counter $jabnorm$ to the base in the abnormal pattern corresponding to the base at $jnorm$ in the normal pattern. Return to step 1 of the

matching loop.

The final processing part consists of the following.

(1) If all normal genetic code has been read, then any remaining bases in the abnormal code are regarded as frameshift additions and the data listed in item 3 of outputs are recorded.

(2) If all abnormal genetic code has been read, then any remaining bases in the normal code are regarded as frameshift deletions and the data listed in item 3 of outputs are recorded.

(3) A table containing the data listed in item 4 of outputs is prepared using the information listed in Table 2.

To illustrate the action of the above algorithm for frameshift additions and deletions, we will use the two sections of code in Table 6. For the frameshift addition example, we assume that the shorter code is normal and for the frameshift deletion example, we assume that the shorter code is abnormal. We also assume that the first codon in the shorter code is *ATC* and the first codon in the longer code is *ACA*, and that the base differences start at **C** in the longer code and **T** in the shorter code.

Table 6 Example of genetic code showing frameshift additions and deletions.

Code	Position	Code
<i>Frameshift addition example</i>		
Normal	47	... <i>ATCGACT</i> ...
Abnormal	52	... <i>ACAAGATGTCGACT</i> ...
<i>Frameshift deletion example</i>		
Normal	52	... <i>ACAAGATGTCGACT</i> ...
Abnormal	47	... <i>ATCGACT</i> ...

For the frameshift addition example, the positions of the first bases in the codons are assumed to be $jnorm=47$ and $jabnorm=52$. For the frameshift deletion example, the positions are assumed to be $jnorm=52$ and $jabnorm=47$.

The algorithm proceeds as follows.

(1) In both examples, step 4 detects differences in 3 consecutive codons and step 4(c) then transfers control to step 6 of the loop. Then step 6(a) finds that the differences in the 3 codons start at **T** in the shorter code and **C** in the longer code. For the frameshift addition example, the frameshift base position counters are set at $fbnorm=48$, $fbabnorm=53$, and for the frameshift deletion example, the counters are set at $fbnorm=53$ and $fbabnorm=48$.

(2) For the frameshift addition example, step 6(b) selects the 5-base pattern TCGAC starting at $fbnorm =$

48 in the normal code and searches forward in the abnormal code for this pattern. The pattern is found at position $53+7=60$ in the abnormal code. The 7-base pattern CAAGATG is accepted as a possible frameshift addition and control is transferred to step 6(c). For the frameshift deletion example, step 6(b) selects the 5-base pattern CAAGA starting at $fbnorm = 53$ and searches forward in the abnormal code for this pattern. We assume that the pattern is not found. Therefore a frameshift addition does not exist and control is transferred to step 6(c).

(3) For the frameshift addition example, step 6(c) selects the 5-base pattern CAAGA starting at $fbabnorm=53$ and searches forward in the normal code. We assume that a match is not found and therefore that a possible frameshift deletion is not found. Control is then transferred to step 6(d) and then to step 7. In step 7(b), the normal base position counter is reset to $jnorm=50$ as the next unread codon after ATC is GAC which starts at position $47+3=50$. In the abnormal code, codon GAC starts at position 62 and therefore the new $jabnorm=62$. For the frameshift deletion example, step 6(d) transfers control to step 8. In step 8(b), the next normal codon after the deletion is CGA starting at position $52+9=61$ and therefore the new $jnorm=61$. In the abnormal code, codon CGA starts at position 49 and therefore the new $jabnorm=49$.

Results

The Matlab program based on this algorithm has been successfully tested on 'normal' and 'abnormal' codes containing many point mutations and frameshift additions and deletions. In this paper, we only show an example of the results of this program when applied to the normal and abnormal β -thalassaemia codes listed in Table 1. The results for the mutations are shown in Table 7. The results in the deductions column show that most of the mutations correspond to deductions. This agrees with the suggestion by Sanchez et al¹¹ that mutations corresponding to deductions are more likely to survive than mutations that do not correspond to deductions. The program also successfully detected the frameshift deletion in the abnormal code of CTTT at normal code position 467, normal codon number 85, and abnormal code position 467.

CLASSIFYING BASES AS PURINE-PYRIMIDINE AND CODONS AS WEAK-MIXED-STRONG

In this section we will use the Wilhelm and Nikolajewa binary coding⁷ of C=00, G=10, A=11, U or T=01 discussed in an earlier section. We have written an algorithm and a Matlab program that uses

this binary coding and regular expressions to identify and count pyrimidines, purines, weak codons, strong-weak codons, weak-strong codons and strong codons in a genetic code.

The regular expression to detect a purine in a 2-digit binary code is $1[01]$ and for pyrimidine a regular expression is $0[01]$. Note: In Matlab, the Unix/Linux expressions for regular expressions are used in which $[01]$ means 0 or 1. For weak codons, the regular expression for a 6-digit binary code is $[01]1[01]1[01][01]$, for weak-strong codons the regular expression is $[01]1[01]0[01][01]$, for strong-weak codons the regular expression is $[01]0[01]1[01][01]$ and for strong codons the regular expression is $[01]0[01]0[01][01]$.

Algorithm for counting purines, pyrimidines, weak, strong-weak, weak-strong and strong codons

The steps in the algorithm are as follows:

- (1) Read in a genetic code as an unbroken string of DNA or RNA base letters.
- (2) Search for a 5-base pattern that detects the START codon in the code (CATGG is used for the β -globin gene).
- (3) Identify and label codons in genetic code.
- (4) Convert DNA or RNA code to binary code with C=00, G=10, A=11, U or T=01.
- (5) Starting at the first codon in the binary code pattern, search and count all occurrences of regular expression $0[01]$ (pyrimidines).
- (6) Repeat step 5 for $1[01]$ (purines), $[01]1$ (weak bases), $[01]0$ (strong bases), $[01]1[01]1[01][01]$ (weak codons), $[01]0[01]1[01][01]$ (strong-weak codons), $[01]1[01]0[01][01]$ (weak-strong codons), $[01]0[01]0[01][01]$ (strong codons).

Results

As an example we show the results of applying the program to the normal β -globin gene pattern given in Table 1. Because of the lengths of some of the outputs we only show selected results in Table 8. A comparison of the counts of the normal code and the abnormal code with the deleted frameshift restored shows that the net effect of the mutations has been to change two purines to pyrimidines, increase the number of weak and weak-strong codons by one, and reduce the number of strong-weak and strong codons by one.

CONCLUSIONS AND RECOMMENDATIONS

The results of this paper can be summarized as follows. The program for detecting known mutations

Table 7 Mutations in β -globin gene: positions in normal and mutated code and biochemical effects.

Position normal	Codon# normal	Codon mutated	Codon normal	Amino mutated	Amino normal	H-bonding mutated	H-bonding normal	# Rings mutated	# Rings normal
131	-28	ATG	ATA	START(MET)	ILE	4-weak	4-weak	5	5
263	17	TAG	AAG	STOP(Amber)	LYS	4-weak	4-weak	5	6
290	26	AAG	GAG	LYS	GLU	4-weak	5-mixed	6	6
302	30	AGT	AGG	SER	ARG	5-mixed	5-mixed	5	6
308	32	CTA	GTA	LEU	VAL	5-mixed	5-mixed	4	5
449	79	ACC	CCC	THR	PRO	5-mixed	6-strong	4	3

Chem. props. mutated	Chem. props. normal	Binary mutated	Binary normal	Deduction	
NPL-HDPB	NPL-HDPB	011000	011001	Yes	Note: NPL-HDPB=nonpolar-hydrophobic PL-HDPL=polar-hydrophilic PC=positively charged NC=negatively charged Binary code of Sanchez et al ¹¹
STOP	PC	100100	010100	No	
PC	NC	010100	000100	Yes	
PL-HDPL	PC	010010	010000	Yes	
NPL-HDPB	NPL-HDPB	111001	001001	Yes	
PL-HDPL	NPL-HDPB	011111	111111	Yes	

Table 8 Counts of weak, strong-weak, weak-strong and strong codons, and pyrimidine and purine bases in normal, abnormal and frameshift adjusted abnormal genes.

Pattern	weak codons (2,2)	s-w codons (3,2)	w-s codons (2,3)	strong codons (3,3)	pyrimidinecount	purinecount
Normal	50	79	63	67	371	406
Abnormal	53	81	64	60	369	405
Abnormal (Frameshift adjusted)	51	78	64	66	373	404

Note: (m, n) means first base in codon has m H bonds and second base has n H bonds. The strength of codon bonding in order of increasing strength is weak, strong-weak, weak-strong, and strong.

could be useful for deciding if a patient has that particular mutation in their β -globin gene. The program for comparing two genetic code patterns could be useful for quickly detecting any differences and classifying the biochemical effects of the differences. This program can be used to compare any 'normal' and 'abnormal' gene patterns and is not restricted to the β -globin gene. The program for listing and counting pyrimidine (purine) bases and strength of H-bonding in codons can also be applied to any gene pattern. The methods and programs we have considered in this paper can be applied to other problems involving detection and classification of differences between genetic codes. The programs are small and fast and are suitable for use on notebook or PC computers. The programs could be useful for doctors and researchers who do not have access to a large commercial package. The programs described in this paper use the Matlab package. However, they could easily be converted to a C or C++ program or developed as stand-alone packages.

Sherva et al²⁸ have examined the severity of the

health effects due to mutations in the β -globin gene. Their data show that a particular mutation in a single gene can have mild or severe effects depending on the existence or otherwise of mutations in other genes and also on environmental factors. In this paper, we have only tried to classify the biochemical effects of mutations in a single gene. The detection and classifying program will detect all mutations in a gene (more precisely the differences between two genetic code patterns). If data is available on the health effects on combinations of genetic mutations, then the final processing stage of this program could be adapted to detect the appropriate combinations. However, this modification is beyond the scope of the present paper.

Acknowledgements: The research in this paper was partially supported by financial assistance from the Graduate College, King Mongkut's University of Technology North Bangkok.

REFERENCES

1. Nopparatana C, Panich V, Saechan V, Sriroongrueng V, Nopparatana C, Rungjeadpha J, Pornpatkul M, Laosombat V, Fukumaki Y (1995) The spectrum of β -thalassemia mutations in southern Thailand. *Southeast Asian J Trop Med Publ Health* **26**, [suppl 1], 229–34.
2. Fucharoen S (1997) Hemoglobinopathies in Southeast Asia: Molecular biology and clinical medicine. *Hemoglobin* **21**, 299–319.
3. Sanguansermsri T, Tanaratankorn P, Steger HF, Chomchuen S, Tongsong T, Chanprapas P, Sirivantanapa P, Sirichotiyakul P (2000) Prenatal diagnosis of Homozygous β -thalassemia by fetal Hb typing analysis using automated HPLC. *Thai J Hematol Transf Med* **10**, 91–101.
4. Solomon EP, Berg LR, Martin DW (2002) *Biology*, 6th edn, Brooks/Cole, Australia.
5. Ricki L (2005) *Human Genetics: Concepts and Applications*, 6th edn, McGraw-Hill, Boston.
6. *The Australasian Genetics Resource Book* (2007) Centre for Genetics Education. www.genetics.edu.au.
7. Willhelm T, Nikolajewa S (2004) A new classification scheme of the genetic code. *J Mol Evol* **59**, 598–605.
8. Wilhelm T, Nikolajewa S (2004) A purine-pyrimidine classification scheme of the genetic code. *Bioforum Eur* **6**, 46–9.
9. Codon Symmetries of the Genetic Code, Leibniz Institute for Age Research. www.imb-jena.de/~sweta/genetic_code_and_evolution/genetic_code_symmetry.html.
10. Nikolajewa S, Friedel M, Beyer A, Wilhelm T (2006) The new classification scheme of the genetic code its early evolution and tRNA usage. *J Bioinformatics Comput Biol* **4**, 609–20.
11. Sánchez R, Morgado E, Grau R (2005) A genetic code Boolean structure. I. The meaning of Boolean deductions. *Bull Math Biol* **67**, 1–14.
12. Rosen KH (2003) *Discrete Mathematics and its Applications*, 5th edn, McGraw Hill, New York.
13. Jiménez-Montaña MA, de la Mora-Basáñez CR, Pöschel T (1996) The hypercube structure of the genetic code explains conservative and non-conservative aminoacid substitutions in vivo and in vitro. *Biosystems* **39**, 117–25.
14. Petoukhov SV (2001) *The Biperiodic Table of Genetic Code and Number of Protons*, Russian Book Chamber, Moscow.
15. He MX (2003) Symmetry in structure of genetic code. In: *Proceedings of Multidisciplinary Conference on Ethics and Science of Future*, Moscow, pp 1–14.
16. Michel CJ (2008) A 2006 review of circular codes in genes. *Comput Math Appl* **55**, 984–8.
17. Dardel F, Képès F (2006) *Bioinformatics: Genomics and Post-genomics*, Wiley, Chichester.
18. Keedwell E, Narayanan A (2005) *Intelligent Bioinformatics: The Application of Artificial Intelligence Techniques to Bioinformatics Problems*, Wiley, London.
19. Hopcroft JE, Motwani R, Ullmann JD (2007) *Introduction to Automata Theory, Languages and Computation*, 3rd edn, Addison-Wesley, Boston.
20. Vincent A (2001) *Molecular Symmetry and Group Theory: A Programmed Introduction to Chemical Applications* 2nd edn, Wiley, London.
21. Robert N, Andrzej P *Non-Deterministic Finite State Automata Built on DNA*. citeseerx.ist.psu.edu/viewdoc/summary?
22. Nowzari-Dalini A, Elahi E, Ahrabian H, Ronaghi M (2006) A new DNA implementation of finite state machines. *Int J Comput Sci Appl* **3**, 51–60.
23. Parker J (1989) Errors and alternatives in reading the universal genetic code. *Microbiol Rev* **53**, 273–98.
24. GenBank. National Institute of Health, USA. www.ncbi.nlm.nih.gov/genbank/GenbankSearch.html
25. Ashby E *Transcription and Translation, Project Synergy*, Maryland Faculty Online www.mdfaconline.org/science/science.tbh_act.doc
26. Gross J, Yellen J (2006) *Graph Theory and Its Applications*, 2nd edn, Chapman & Hall/CRC, Boca Raton.
27. Lawson MV (2004) *Finite Automata*, Chapman and Hall/CRC, Boca Raton, p 103.
28. Sherva R, Sripichai O, Abel K, Ma Q, Whitacre J, Angkachatchai V, Makarasara W, Winichagoon P, Svasti S, Fucharoen S, Braun A, Farrer LA (2010) Genetic modifiers of Hb E/ β^0 thalassemia identified by a two-stage genome-wide association study. *BMC Med Genet* **11**, 1471–2350.