Massingham and Goldman *BMC Bioinformatics* 2012, **13**:145
http://www.biomedcentral.com/1471-2105/13/145

BMC Bioinformatics

RESEARCH ARTICLE Open Access

Error-correcting properties of the SOLiD Exact Call Chemistry

Tim Massingham* and Nick Goldman*

Abstract

Background: The Exact Call Chemistry for the SOLiD Next-Generation Sequencing platform augments the two-base-encoding chemistry with an additional round of ligation, using an alternative set of probes, that allows some mistakes made when reading the first set of probes to be corrected. Additionally, the Exact Call Chemistry allows reads produced by the platform to be decoded directly into nucleotide sequence rather than its two-base 'color' encoding.

Results: We apply the theory of linear codes to analyse the new chemistry, showing the types of sequencing mistakes it can correct and identifying those where the presence of an error can only be detected. For isolated mistakes that cannot be unambiguously corrected, we show that the type of substitution can be determined, and its location can be narrowed down to two or three positions, leading to a significant reduction in the the number of plausible alternative reads.

Conclusions: The Exact Call Chemistry increases the accuracy of the SOLiD platform, enabling many potential miscalls to be prevented. However, single miscalls in the color sequence can produce complex but localised patterns of error in the decoded nucleotide sequence. Analysis of similar codes shows that some exist that, if implemented in alternative chemistries, should have superior performance.

Background

The collection of technologies described as Second- or Next- Generation Sequencing (NGS) platforms are characterised by the synthesis of complementary strands of DNA from clusters of homologous templates [1]. The chemistry used differs between the platforms but that for the Life Technologies Corporation SOLiD platform is particularly interesting since there is not a one-to-one correspondence between measurements made during sequencing and nucleotides of the sequence being read. Instead the primary output of the SOLiD platform is a 'color sequence', an encoded form of the nucleotide sequence, that has advantages for calling SNPs when comparing the reads to a reference [2]. With the advent of the 5500 Series of machines in November 2010, an improved 'Exact Call Chemistry' (ECC) was introduced that changes the way that the sequence is encoded by the platform and allows mistakes in the measurements to be corrected, hence producing more accurate reads [3].

*Correspondence: tim.massingham@ebi.ac.uk; goldman@ebi.ac.uk
European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridgeshire, UK

The SOLiD sequencing chemistry consists of multiple rounds where probes, consisting of eight bases and a fluorophore, are sequentially ligated to the template sequence to build up a complementary strand. Each round consists of a priming step followed by a repeated cycle of ligating probes to the template, exciting the fluorophores and imaging the resulting emission, then cleaving the flurophore and part of the probe ready for the next cycle. The probe/fluorophore combinations are designed so that the probes interrogate the first two of the eight ligated positions in the template, with each of four fluorophore colors used to indicate four of the 16 possible nucleotide pairs at these positions. The color of the fluorophore for each template is recorded and used later to determine the sequence of the read. After imaging, those templates to which a probe failed to ligate have their previous probe decapped (i.e. dephosphorylated) so they will not be extended on future cycles. This reduces problems analogous to 'phasing' on the Illumina platform [4]. The fluorophore and last three bases are cleaved from the probe, leaving the strand ready to be further extended in the next cycle. The repeated ligation of eight additional bases and then cleaving the end three mean that

the pair of bases of the template sequence that is interrogated moves on by five positions every cycle. After a specified number of cycles, the round is stopped and the complementary strand melted from the template to leave the template ready to be primed for the next round. Each round uses a different primer so that the positions interrogated by the probes change each time; for example, on the first round the first position of the probes on cycles 1, 2, … corresponds to positions 1, 6, … of the template sequence; on round two these become positions 2, 7, …, etc.

After five rounds, probes have been ligated so that every position of the template sequence has been the first position of a probe. The colors recorded and prior knowledge of one of the bases, typically the last adapter base adjacent to the template, are sufficient to determine the nucleotide sequence assuming that no errors have been made. Conventionally, one of the rounds is primed so that it starts one base before the beginning of the template sequence and so the first base interrogated is the last base of a known adapter sequence.

The SOLiD Exact Call Chemistry (ECC) augments the 'two-base-encoding' chemistry with an extra round where a different set of probes is ligated. Each cycle of this round interrogates positions 1, 2 and 4 of each five-nucleotide block of the template. The same four fluorophores are used, each now indicating the presence of one of 16 of the possible 64 combinations of nucleotides at the positions interrogated. These color calls can be used to detect and recover from miscalls made in previous rounds.

Directly decoding the colors from the first five rounds into nucleotide sequence allows catastrophic failures to occur where a single calling error creates errors in every position from then on [2]. As well as allowing some errors to be corrected, the additional round also enables the calls to be translated into nucleotide sequence in a manner that behaves more gracefully in the presence of mistakes [3].

The mathematical basis for the error correcting properties of the SOLiD ECC is the theory of convolutional codes [3], a class of codes which are incorporated into some of the most powerful error-correcting codes in general use, for example those used in the Voyager deep-space exploration program [5]. Convolutional codes are a category of linear codes [6,7] and we use the theory of linear codes to analyse the error correcting properties of the SOLiD Exact Call Chemistry, describing the types of sequencing mistakes it can correct and those cases where the presence of an error can only be detected. For isolated mistakes that cannot be unambiguously corrected, we show that the type of substitution can be determined and its location can be narrowed down to two or three positions in the read, leading to a significant reduction in the the number of plausible alternative reads. If the read is 'corrected' at the wrong location, we show that the nucleotide read ultimately produced contains errors that have a distinctive pattern. Finally, we apply the same techniques used to analyse the Exact Call Chemistry to look at hypothetical alternative chemistries and show that some of them have superior characteristics, being able to correct more errors and inducing simpler patterns of error in the decoded nucleotide sequence when reads are miscorrected.

Methods

Convolutional code theory

We start by defining the terminology we need to analyze the SOLiD code. The chemistry of the sequencing reactions and imaging mean that each fragment of DNA, or 'source sequence' (nucleotides), is encoded into a 'code sequence' (colors) of equal or greater length. The original five rounds produce as many observed colors as there are nucleotides in the fragment; the ECC round adds one-fifth as many additional color calls. The code sequence is then observed with potential errors ('observed sequence'). Since the code sequence is longer than the source sequence, not all possible code sequences correspond to an encoded source sequence; those that do correspond are termed 'valid'. Given an observed sequence, some procedure is used to find the closest valid code sequence ('corrected sequence') and the nucleotide sequence that produces this code sequence is the decoded sequence. Note that the source sequence and decoded sequence are sequences of nucleotides ('base-space'), whereas the code, observed and corrected sequences are colors ('color-space').

In terms of the theory to be presented, colors and nucleotides are just different representations of the same things, members of a set of four elements over which a finite field (i.e. Galois field, GF_4) is defined (see Table 1). To prevent confusion between the various matrices and vectors studied and their concrete representation as nucleotides or colors, all calculations in the text are described in terms of elements GF_4 and the more usual notations are reserved for when an actual sequence of nucleotides or colors is meant.

The ECC code, and any linear code in general, is defined in terms of a $k \times n$ generator matrix G with elements in GF_4. A source sequence s is a row vector of length k consisting of elements of GF_4, and its code sequence c is sG where the vector-matrix multiplication is understood in the usual sense but addition and multiplication of elements is carried out within GF_4 (Table 1). Codes are termed linear because any linear combination of valid code sequences is also a valid code sequence: if s_1 and s_2 are source sequences with corresponding code sequences c_1 and c_2, and w_1 and w_2 are elements of GF_4 then

Table 1 Different representations of GF_4

Representation	Elements				⊕	0	1	α	β	⊗	0	1	α	β
GF_4	0	1	α	β	0	0	1	α	β	0	0	0	0	0
Nucleotides	A	C	G	T	1	1	0	β	α	1	0	1	α	β
Colors	***0***	***1***	***2***	***3***	α	α	β	0	1	α	0	α	β	1
					β	β	α	1	0	β	0	β	1	α

For the purposes of the coding theory presented here, both nucleotides and colors represent elements of the Galois Field over four elements (GF_4) and the correspondence between them is shown below. For example, the color '***2***', the nucleotide 'G' and the element 'α' are considered to be equivalent for the purposes of calculation. A field consists of a set of elements and rules on how to add ($\oplus$) and multiply ($\otimes$) them together; the results of combining two elements are expressed by the Cayley tables above; for example, $\alpha \oplus \beta = 1$ and $\alpha \otimes \alpha = \beta$. The standard rules for associativity and commutativity for multiplication and addition still apply in finite fields, and multiplication is still distributive over addition [8]. One notable difference from ordinary arithmetic is that all elements are self-invertible under addition in GF_4, so addition and subtraction are equivalent operations.

$w_1c_1 + w_2c_2 = (w_1s_1 + w_2s_2)G$ so $c = w_1c_1 + w_2c_2$ is a valid code sequence corresponding to the source sequence $w_1s_1 + w_2s_2$.

Rather than expressing errors in terms of a specific color miscall, we consider types of errors defined by what happens when an element of GF_4 is added to a call. For example, the error type '$+\beta$' transforms the color ***1*** ($\equiv$ 1, an element of GF_4) into ***2*** ($\equiv \alpha$) since $1 \oplus \beta = \alpha$, and transforms ***0*** ($\equiv 0$) into ***3*** ($\equiv \beta$) since $0 \oplus \beta = \beta$ (see Table 1). Although we will be considering color miscalls in the coding sequence, it is instructive to examine the action of these error types on nucleotides to show that they can have a concrete interpretation: $+\beta$ complements the nucleotide, $+\alpha$ results in a transition, +1 preserves whether the nucleotide is an amino- or keto- acid ('transcomplement') and +0 leaves the nucleotide unchanged. The group structure ensures that all possible substitutions can be encoded in this form and, when errors are rare, the sequence of error types will consist mostly of 0s. Adding the sequence of error types to the observed sequence results in the recovery of the (true) code sequence; likewise, adding the error types to the code sequence results in the observed sequence.

Each generator matrix has an associated parity check matrix H that can be used to determine whether a code sequence is valid. The parity check matrix uses a subset of the elements of observed sequence to call a putative decoded sequence and then calculates the expected value of the remaining 'parity' elements by treating the decoded sequence as true. The expected and observed parity sequences are then summed element-wise to create the parity check sequence. H is constructed so that the parity check sequence is the sequence-matrix product xH, where x is the observed sequence. By construction, the parity check of valid code sequence is a sequence of 0s; invalid code sequences have a non-zero parity check. When errors have occurred, the observed sequence can be expressed in terms of the code sequence for the source sequence and a sequence of error types e: $x = c + e$. Since the parity check of a valid code sequence is a sequence of zeros, the parity check of the observed sequence only depends on the sequence of errors and not the source sequence as $xH = (c + e)H = eH$. The value of $eH = xH$ is termed the 'syndrome' of the error, a sequence in GF_4 of length $n - k$ that partitions the set of possible errors into equivalence classes.

Syndromes provide a simple and efficient method of decoding an observed sequence under the assumption that errors are rare and so the most plausible corrected sequence is the valid coding sequence which requires the fewest changes from the observed sequence. An error consisting of multiple changes may belong to the same equivalence class as one with fewer changes but the simpler error is the more likely and so is the best predictor of what error actually occurred. Before decoding begins, a table is constructed mapping every syndrome (σ) to the simplest possible sequence of error types that has it (e_σ); this can be done by enumerating all single errors, followed by doublets, triplets, etc. until every syndrome has been observed at least once and all error types up to a given complexity have been considered. The syndrome table is a function of the code structure only and can be calculated once and distributed with the probe sets. To decode each observed sequence x, its syndrome xH is calculated and then located in the table to find the simplest sequence of error types e_{xH} that could cause it. The sequence $x + e_{xH}$ is then a valid code sequence which can then be decoded. If there are several equally simple sequences of error types, then an error has been detected but the correction is ambiguous.

The ECC code is defined by the architecture shown in figure 1. The encoding 'machinery' looks at windows of length five of the source sequence, moving along one element at a time, and performing the specified additions and multiplications. Two streams of symbols are emitted. For the first stream, corresponding to the standard color encoding, the first two elements of the window are added together. For the second stream, the ECC encoding, only every fifth symbol is recorded, a practice known as puncturing or perforating. This stream multiplies the second and fourth elements of the window by

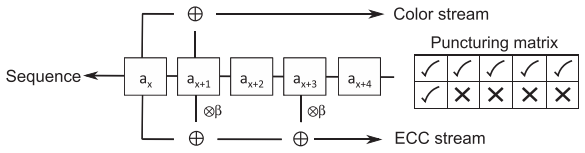

Figure 1 Architecture of the SOLiD ECC encoder. The architecture of the convolutional code for the SOLiD ECC consists of two streams. The nucleotide sequence $a_1, a_2, \ldots$ is passed through the encoder, progressing one position at a time, and the color obtained from the additions and multiplications indicated is emitted from each stream. The top 'color stream' is that produced by the two-base-encoding chemistry and the bottom 'SOLiD ECC stream' is punctured so that only every fifth color member of the sequence is used.

β and sums them with the first element. In terms of the SOLiD chemistry rounds, the color calls from each of the first five rounds correspond to every fifth element of the first stream with a different offset for each round and the calls from the final round correspond to the punctured second stream.

The calculation for both the first and second streams ($i = 1, 2$) can be expressed as the dot product $a \cdot \rho_i$ where a is a row vector of the bases in the window and the ρ_i describe the calculations to be done, with $\rho_1 = 11000$ and $\rho_2 = 1\beta0\beta0$. The ρ_i are the 'probe generators' for the chemistry since they define the color of fluorophore that each probe has attached to it: the probe that interrogates the sequence b has color $b \cdot \rho_i$.

The architecture of the ECC code imposes the block structure shown in figure 2 on the source sequence. The read is partitioned into blocks of length five, each of which can be uniquely recovered from the indicated five elements of the encoded sequence. The remaining elements of the encoded sequence each span two blocks and are in effect parity colors that can be used to detect the presence of errors. Ignoring the parity colors, the generator for each block is the invertible matrix

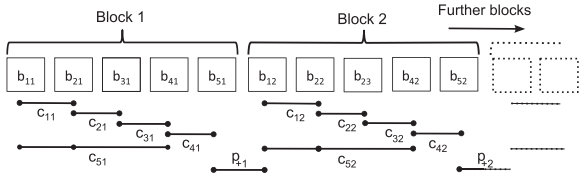

Figure 2 Block structure of encoded read. A read partitioned into blocks of five bases, with block i containing bases $b_{1i}b_{2i}b_{3i}b_{4i}b_{5i}$, showing how the two-base-encoding and ECC color calls are split into five 'data' colors $c_{1i}c_{2i}c_{3i}c_{4i}c_{5i}$, from which the block can be called, and a 'parity' color (p_{+i}) which straddles the block and its downstream neighbour. The data colors are used to determine the nucleotide sequence of the blocks and the parity color is used to detect whether an error has occurred. Note that the data colors are a mixture of both color streams, with the parity color coming from the color stream of the code.

$$G_{\text{Block}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & \beta \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & \beta \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad G_{\text{Block}}^{-1} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ \beta & \beta & \alpha & \alpha & \alpha \\ \beta & \beta & \beta & \alpha & \alpha \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

which can be used to freely convert between a sequence of five nucleotides and its encoding. For example, the nucleotide sequence ACGAT ($\equiv 01\alpha0\beta$) has the color encoding ***13233*** ($\equiv 1\beta\alpha\beta\beta$) since $(01\alpha0\beta)G_{\text{Block}} = (1\beta\alpha\beta\beta)$. Conversely, $(1\beta\alpha\beta\beta)G_{\text{Block}}^{-1} = (01\alpha0\beta)$.

After reordering the encoded sequence appropriately, the generator for the full code, G_{ECC}, can be expressed in terms of the generators for each of the blocks and an additional column for each of the parity colors:

$$G_{\text{ECC}} = \left(\begin{array}{ccccc|ccc} G_{\text{Block}} & 0 & \cdots & & & u_5 & 0 & \cdots \\ 0 & G_{\text{Block}} & 0 & \cdots & & u_1 & u_5 & \ddots \\ \vdots & 0 & G_{\text{Block}} & \ddots & & 0 & u_1 & \ddots \\ & \vdots & \ddots & \ddots & & \vdots & \ddots & \ddots \end{array}\right) = (G^* | P)$$

where u_i is a column vector with five elements consisting of 0s except for a 1 in the i^{th} position, with G^* and P being defined appropriately. The dimensions of G_{ECC} are $k \times n$, where k is the length of the read (source sequence) and n the total number of color calls produced in both the two-base-encoding and ECC chemistry rounds.

The parity-check matrix, H_{ECC}, corresponding to generator G_{ECC}, is

$$H_{\text{ECC}} = \begin{pmatrix} G^{*-1} & 0 \\ 0 & I_{n-k} \end{pmatrix} \begin{pmatrix} P \\ I_{n-k} \end{pmatrix}$$

where I_m is the $m \times m$ identity matrix. By considering the action of this parity-check matrix on an encoded sequence, the calculation of the syndrome can be given concrete form: the first matrix of the factorised form of H_{ECC} takes the observed sequence and inverts it block by block to get a putative nucleotide decoding, with the values of the parity colors preserved. The second matrix calculates the parity colors for the putative decoding and adds them element-wise to the parity colors actually observed. The resulting sequence of length $n-k$ is the syndrome. If the syndrome is composed entirely of 0s, i.e. the parity-check has been passed, then the code sequence and its putative nucleotide decoding are valid. If the syndrome has non-zero entries then an error has been detected.

When an error is known to have occurred but is ambiguous, it may be mistakenly 'corrected' by applying the wrong error type. The equivalence class contains the possible simple error types that could have caused the observed syndrome but only one of them is the one that actually occurred; applying any to the observed sequence

will result in a valid code sequence. We can determine the pattern of nucleotide errors that mistaken correction will result in: if the observed sequence with (unknown) error e_1 is $c + e_1$ and the correction $e_2 \neq e_1$ is applied, then converting back to a nucleotide sequence using the inverse of the block generator results in $(c + e_1 + e_2)\, G^{-1}$. The difference between the nucleotide translation of the corrected sequence and the correct nucleotide sequence is $(e_1 + e_2)\, G^{-1}$ — the pattern of error induced. Note that the pattern does not depend on the correct nucleotide sequence, but only on the two changes being considered. By examining all possible combination of error types belonging to an equivalence class, the full set of patterns that may occur can be determined. The pattern of error may be a single base change, or a more complicated multi-base change.

Practical implementation

The theory described examines the worst case where an error can occur anywhere and there is no additional information about which site it is likely to have affected. Real-life performance of the code depends on additional factors. Firstly, the distribution of where errors occur is not uniform, depending for example on the chemical and physical characteristics of the sequencing process, and may not even be independent between positions; the performance of a code may change depending on the error profile. Secondly, but related, the sequencing platform provides quality information in the form of Phred scores [9] that can be used to help locate the position where an error occurred. Values quantifying the probability that a given call is wrong are known as 'soft information', compared to the 'hard information' of the observed sequence not being a valid encoding.

A convenient way to deal with these extra complications is to simulate encoded sequence under a realistic model of how errors occur and then decode using dynamic programming, on a 'trellis' graph that defines all possible decodings and their relative probabilities, to find the most probable call for each position of the decoded sequence [3]. In practice, we need to consider three classes of differences: those due to variants between sequence under study and the reference it is mapped against, those caused by mutations in the original sequence due to polymerase errors during sample preparation ('generalised error') and errors made calling the encoded sequence. The first type of difference is of interest and will affect many reads mapping to a single location. Of the remaining two types of difference, the former occurs before the sequence is encoded, so encoding provides no protection and limits the maximum accuracy of the platform; errors of the latter class may be correctable.

In the absence of empirical data for how the distribution of calls and miscalls changes over probes and rounds, and how errors are correlated between positions, we have assumed that errors occur independently for every element of the code sequence and errors are picked uniformly from the three possibilities. The probability of an error for a particular round and cycle of ligation is taken to be equivalent to the quality score from a read sampled at random from a real set of data, implicitly assuming that the error characteristics for alternative probe sets (which depend on the ligation efficiencies of the different pentamers) will be the same as for the ECC probe set. The simulation scheme is as follows:

1. Sample a fragment from genome (base-space).
2. Mutate bases of fragment independently with equal probability (generalised error).
3. Encode mutated fragment (convert to code-space).
4. Sample qualities from a set of real data.
5. Mutate the encoded sequence with probabilities defined by quality values.

This scheme outlined has an error model similar to that implicitly assumed when analysing real data since information about both alternative calls and correlation between calls has already been lost during processing into a color-space sequence with a single quality value for each position. The scheme does not simulate the occurrence of insertions and deletions but these are relatively rare compared to substitutions and are most likely to be due to errors introduced during sample preparation rather than calling errors.

Results and discussion

If a read is error-free then it can be decoded unambiguously and the calls from the two-base-encoding chemistry and ECC codes will agree. When an error occurs in any of the two-base-encoding chemistry rounds, it translates into multiple base miscalls. The structure of the ECC code allows such errors to be detected, recovered from, and, in certain circumstances, corrected. The syndrome equivalence classes are defined by the types of error that can occur and so, by examining them, we can classify the cases where errors can be corrected unambiguously and those where additional information is needed to resolve the ambiguity.

Since the elements of the syndrome consist solely of the summation of the observed and expected parity colors, there is a one-to-one correspondence between them and parity colors. An error in a block can only affect two elements of the syndrome, those corresponding to the two parity colors that overlap the block's first and last elements. We refer to these as the upstream and downstream syndromes, respectively, so it is sufficient to concentrate on how errors occur in only one block. All statements we make about the error correcting properties of the encoding are on a per-block basis, so a code that

can correct one error per block can correct multiple errors if spread between blocks. The final block only has a upstream syndrome and so has more limited error correction.

The value of the syndromes for all possible single-color errors that could occur in a block or its parity colors are shown in Table 2. For example, an error of +1 in the second color produces the syndrome $\beta\alpha$, identical to that produced by an error of +1 in the third position. If a syndrome is unique (for example, syndrome $\beta\beta$ arises only from error $+\beta$ in position c_5) then, assuming only a single error has occurred in that block or parity color, that error can be determined and so corrected. Note that while errors in parity color p_-, overlapping a block and the previous one, appear to have unique syndromes, this parity color is also p_+ for the previous block and so the syndrome can actually be caused by multiple different errors.

The syndrome equivalence classes show the types of single-color error that cannot be disambiguated without additional information; a error has been detected but is not correctable. For example, the syndromes 0β, 0α and 01 can be generated by single errors at c_1, c_4 or p_+, and so errors at these positions cannot be distinguished. The equivalence classes define the possible simple changes to the observed sequence that will produce a valid sequence that can be decoded into a string of bases, only one of which produces the correct sequence. When an error is wrongly corrected, the code sequence is changed in two places, the original error and the correction; the structure of the code ensures that the changes should be complementary (both '+1' or both '$+\alpha$', for example).

Using the inverse of the block generator matrix, the pattern of changes that are induced in base-space by miscorrections to the observed sequence can be derived. If errors occur and are miscorrected, such that the difference between the correct and observed values of the data colors is d, then the pattern of differences p induced in base-space is given by $p = dG^{-1}_{\text{Block}}$. For example, an error of type $+\alpha$ at the second position might be erroneously corrected by an error of type $+\alpha$ at the third position, so the differences are $0\alpha\alpha00$ and the pattern induced in base-space is $00\alpha00$ (the sum of the second and third rows of G^{-1}_{Block} multiplied by α). If this occurred to the nucleotide sequence ATGCG then the sequence ATACG would result.

Table 3 Patterns of error for the ECC generator

Positions	Change	Pattern
c_2, c_3	Single	00100
c_4, p_+	Single	00001
c_1, c_4	Triple	01110
c_1, p_+	Quadruple	01111

All possible patterns of error caused in corrected base-space sequence by a single wrongly corrected error in the observed sequence of type $+d$ (where $d = 1, \alpha$ or β). The 'Positions' column indicates all possible pairs of positions at which an error can occur and be wrongfully corrected; it is not necessary to identify which member of the pair corresponds to which form of error as the resulting pattern is the same. In all cases the error pattern should be multiplied element-wise by the error type (d) to get the actual pattern (e.g. a wrongly corrected $+\beta$ error at c_2 results in an error pattern of $00\beta00$).

All patterns induced by single-color errors are listed in Table 3; for example, an error of type $+\alpha$ at the first position may be wrongly corrected at either the fourth position or the parity color, leading to the triple error $0\alpha\alpha\alpha0$ or the quadruple error $0\alpha\alpha\alpha\alpha$, respectively. Note that no pattern ever affects the first position of the block; the structure of the matrix G^{-1}_{block} shows that this position can only be changed by errors in the second, third or fifth positions: single-color errors at the fifth position can be always be unambiguously corrected, and errors at the second or third positions result in either corrections or changes that cancel at the first position.

One notable feature of these more complex errors is that the error type is the same at all affected positions, a property that might help distinguish sequencing errors from genuine variants if the reads are later mapped to a reference. By adding the mapped read to the reference in GF_4, the pattern should be evident if it was due to simple miscall. If the pattern is not evident, the differences are either due to sequence variants or multiple miscalls.

While the consideration of syndromes provides useful information about a code's properties and syndrome decoding is computationally efficient, it is not a replacement for probabilistic methods of decoding since the

Table 2 Syndromes for ECC generator

Error	Interpretation	Position							Equivalence
Type		p_-	c_1	c_2	c_3	c_4	c_5	p_+	classes
$+\beta$	Complement	$\beta0$	0β	$\alpha1$	$\alpha1$	0β	$\beta\beta$	0β	$\{c_5\}$
$+\alpha$	Transition	$\alpha0$	0α	1β	1β	0α	$\alpha\alpha$	0α	$\{c_2, c_3\}$
+1	Transcomplement	10	01	$\beta\alpha$	$\beta\alpha$	01	11	01	$\{c_1, c_4, p_+\}$

All syndromes caused by a single error in a block of five code letters ($c_1c_2c_3c_4c_5$) and two parity letters (p_-p_+) of the SOLiD ECC code. Each row in the table corresponds to a specific type of error at the given position of the code word, with the 'interpretation' of an error type being the effect it would have when considered as a nucleotide substitution. The table entries are the values of the relevant elements of the syndrome, corresponding to the upstream and downstream parity checks for the block for each error type. The error type +0 is not shown since it represents no error; its syndromes would be 00 at all positions. The equivalence classes are listed separately and do not correspond to specific error types. Note that p_- is also p_+ for the preceding block.

latter incorporates the quality information about each call and can use it to make better decisions about correction. However, syndrome decoding techniques may still be of use in conjunction with the more computationally expensive probabilistic methods since the syndrome provides a quick test of whether the observed sequence is valid (no correction needed). Syndrome equivalence classes could also be used to restrict the paths through a trellis to a plausible set, providing a heuristic to speed up the dynamic programming algorithm to find the most probable decoding.

Theoretical bounds

We have shown the type of errors that the ECC code can correct but have not yet addressed whether it is optimal. Before examining specific alternative codes, it is interesting to look at what can possibly be achieved and there are several mathematical results that restrict the performance of any code. Firstly the Hamming [10], Johnson [11] and Singleton [12] bounds place an upper limit on the number of errors that a code can hope to detect or correct but codes meeting these bounds may not exist; the Gilbert–Varshamov bound [13,14] is a lower limit on the performance of the best code that does exist.

For reads of 50 bases encoded into 60 letters (as with the ECC code), the lower bound guarantees that a code exists that can detect any two errors and correct any single error. The upper bounds show that no code can guarantee to correct more than three errors. For reads of length 75 bases encoded into 90 letters, then a code exists that can detect and correct any two errors but no code can guarantee to detect and correct more than four errors. There is no guarantee that a convolutional code can meet this bound and the two most common classes of codes that come close to attaining these bounds, Turbo codes [15] and Low-Density Parity-Check codes [16], require long-range dependencies between positions in the sequence and so cannot be implemented in any plausible sequencing chemistry.

Alternative chemistries

Examining the syndromes in Table 2, we notice that, despite many possible single errors having ambiguous syndromes, not all syndromes are present: the three syndromes $\alpha\beta$, $\beta 1$ and 1α do not occur. The missing syndromes are not truly unused, being generated by multiple errors, but the failure to use them to distinguish single errors suggests that there may exist alternative codes with a greater ability to correct single calling errors. The advantage would derive from using the extra syndromes to partition single-color errors more evenly into equivalence classes. Such alternative codes can be analysed using the same techniques as the ECC code.

Rather than consider all possible convolutional codes, we will focus our attention on a subset that satisfy some reasonable restrictions inspired by the reality of the SOLiD platform. It is desirable that any new chemistry would be backwards compatible with the two-base-encoding chemistry, meaning that the first five rounds of sequencing must use an unaltered two-base-encoding probe set. This backwards compatibility restriction is equivalent to requiring that a new code must have an unpunctured color stream.

While the number of rounds and probe sets could be varied, and probes of differing lengths could be used, to remain comparable to the current ECC chemistry we will focus only on chemistries where a single additional round (and so only one additional probe set) will be used and the probes remain based on pentamers. Analogous to the code structure shown in figure 1, alternative codes that can be implemented with a single additional round are punctured so only every fifth element of the second stream is produced.

By redefining the boundaries of the block structure, the number of different alternative codes that need to be considered can be further reduced: a code whose probe generator starts with one or more 0s is identical to one starting at the first non-zero element with the tail padded with zeros. The probe generator of any code that starts with α or β can be written as αW or βW, where W is the probe generator of a code starting with 1, and the linearity of convolutional codes ensures that the two codes have identical sets of code words: although the mapping between nucleotide sequence and encoded sequence will differ, the error correcting characteristics will be the same.

The block generator for alternative codes that satisfies all the restrictions, and its inverse, can be written as

$$G_{\text{alt}} = \begin{pmatrix} 1 & 0 & 0 & 0 & p_1 \\ 1 & 1 & 0 & 0 & p_2 \\ 0 & 1 & 1 & 0 & p_3 \\ 0 & 0 & 1 & 1 & p_4 \\ 0 & 0 & 0 & 1 & p_5 \end{pmatrix} \quad G_{\text{alt}}^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & x_1 y^{-1} \\ 0 & 1 & 0 & 0 & x_2 y^{-1} \\ 0 & 0 & 1 & 0 & x_3 y^{-1} \\ 0 & 0 & 0 & 1 & x_4 y^{-1} \\ 0 & 0 & 0 & 0 & y^{-1} \end{pmatrix} L \tag{1}$$

where $p_1p_2p_3p_4p_5$ is the generator for the second set of probes, L is the matrix whose upper triangle consists of 0 and whose diagonal and lower triangular elements are all 1, $x = Lp$ and $y = x_5$. The ECC code has the probe generator $p_1p_2p_3p_4p_5 = 1\beta 0\beta 0$. The inverse generator only exists when y is invertible (i.e. $y \neq 0$); when y is not invertible, blocks of colors cannot be individually inverted and the syndrome analysis is not applicable. Due to the structure of L, requiring y to be invertible is the same

as the sum of the elements of the probe generator not being zero.

One further restriction will be placed on the probe generator of alternative codes: the ECC probe generator contains 0 at its fifth position and this is probably a consequence how accurately pentamers with differing final bases can be ligated to the sequence. Codes whose generators use the final position might theoretically have better error correction properties but the increased rate of calling errors, due to incorrect probes being ligated, may cancel any improvement their use may offer; consequently, we will require $p_5 = 0$.

There are 48 probe generators satisfying all the restrictions and the invertibility condition, of which the syndromes and equivalence class for two interesting alternatives are shown in Table 4. The first code has generator $p_1p_2p_3p_4p_5 = 10\beta00$ and has similar equivalence classes to the ECC code but only uses the first three positions of the generator. While the set of syndromes for single-color errors is also incomplete (syndromes 1β, $\alpha1$ and $\beta\alpha$ are unused) and thus the error correcting properties will be similar to the ECC code, the shorter length of the generator means that calculations on the trellis, to determine the most probable decoding, can be carried out four times quicker.

The second alternative code shown in Table 4, with probe generator $1\beta010$, uses all possible syndromes and potentially has better error correcting properties than the ECC code. Comparing the equivalence classes of the new code to those for the ECC code, the new code uses the extra syndromes to split the largest class into two. Whereas the ECC code is unable to distinguish errors at the first, fourth or parity positions, the new code can unambiguously correct errors at the first position and the ambiguity of fourth and parity position errors is reduced.

Table 5 shows the error patterns induced by single-color errors for the two alternative codes considered. Note the reduced number of base-space errors relative to the ECC code (Table 3).

Table 5 Patterns of error for alternative codes

Generator	Positions	Change	Pattern
$10\beta00$	c_1, c_2	Single	01000
	c_3, c_4	Single	00010
	c_3, p_+	Double	00011
	c_4, p_+	Single	00001
$1\beta010$	c_2, c_3	Single	00100
	c_4, p_+	Single	00001

All possible patterns of error caused in corrected base-space sequence by a single, wrongly corrected error in the observed sequence for code with given generator. An error occurs at one of the positions in the 'Positions' column; the other member of 'Positions' is where the observed sequence is wrongfully corrected. As in Table 3, all error patterns should be multiplied element-wise by the error type (1, α or β) to get the actual pattern.

Simulations

The theory described suggests that the code with probe generator $1\beta010$ is more powerful than the ECC code but this does not necessarily mean that it performs better in practice since actual performance will depend on the distribution of different types of error and at which positions they occur. To help quantify the difference in performance between codes, they were compared on artificial data, simulated so that it had an error profile similar to real data but for which the original sequence of bases is known, the error profile being estimated by mapping to a SNP-corrected genome.

One million fragments were sampled uniformly from the positive strand of the genome of *E. coli* DH10b with qualities being sampled from a real sequencing run of the same genome. The probability of generalised error was set to be equivalent to Q_{34}, close to observed values for prepared samples. Three sets of reads were produced, using different codes on the same set of fragments to generate the ECC information, and sequence was called into both base-space and color-space using the Maximum A Posteriori (MAP) criterion. Reads were then mapped to an appropriately encoded reference using the BWA aligner

Table 4 Syndromes for alternative codes

Generator	Error type	Position							Equivalence classes
		p_-	c_1	c_2	c_3	c_4	c_5	p_+	
$10\beta00$	$+\beta$	$\beta0$	1α	1α	0β	0β	$\alpha\alpha$	0β	$\{c_5\}$
	$+\alpha$	$\alpha0$	$\beta1$	$\beta1$	0α	0α	11	0α	$\{c_1, c_2\}$
	$+1$	10	$\alpha\beta$	$\alpha\beta$	01	01	$\beta\beta$	01	$\{c_3, c_4, p_+\}$
$1\beta010$	$+\beta$	$\beta0$	$\alpha1$	1α	1α	0β	11	0β	$\{c_1\}$, $\{c_5\}$
	$+\alpha$	$\alpha0$	1β	$\beta1$	$\beta1$	0α	$\beta\beta$	0α	$\{c_2, c_3\}$
	$+1$	10	$\beta\alpha$	$\alpha\beta$	$\alpha\beta$	01	$\alpha\alpha$	01	$\{c_4, p_+\}$

All syndromes caused by a single error in a block of five code letters ($c_1c_2c_3c_4c_5$) and two parity letters (p_-p_+) for codes with the specified generator. Each row corresponds to a specific type of error at the given position of the code word and the table entries are the values of the relevant elements of the syndrome corresponding to the upstream and downstream parity checks for the block for each error type. The error type +0 is not shown since it represents no error. The equivalence classes are listed separately and do not correspond to specific error types. Note that p_- is also p_+ for the preceding block.

[17] with an edit distance of five. Results, in terms of the total proportion of reads mapping and the proportion mapping with a given number of errors, are shown in Table 6.

Using the ECC information (probe generator $1\beta0\beta0$) to help call reads makes a considerable improvement in both color-space and base-space. Without any correction, 77.1% of simulated reads map to the reference in color-space and only 38.4% of these are perfect, with a further 16.7% having one error. After correction, 48.6% of reads are perfect, with 10.2% containing one error. In base-space, the number of mapped reads increases from 47.2% to 64.3%. This large increase is due to correcting single-color errors that would otherwise induce multiple base errors and prevent the read from being mapped using the 'five differences or fewer' criterion.

All three codes perform better than uncorrected sequence in both color-space and base-space. There was little difference between their performance in absolute terms, although the code generated by $1\beta010$ produced more error-free reads than the other two. This small absolute difference disguises a larger increase in the proportion of corrected reads: under the $1\beta0\beta0$ code, 10.2% of reads were corrected to being perfect compared to 11.4% for the $1\beta010$ code, an 11.8% relative improvement even though the absolute improvement is only 1.2%. The $10\beta00$ code produces more mappable base-space sequence than either of the other codes and the reason for this may be in the pattern of errors in base-space that single-color errors make: comparing the patterns in Tables 3 and 5 shows that, for the two alternative codes, single-color errors predominantly cause a single base error when wrongfully corrected, rather than more complex errors, and so there are fewer base-space errors in total.

While the code generated by $1\beta010$ does perform better than the ECC code, the improvement is not especially dramatic despite the syndrome equivalence classes suggesting superior error-correcting properties. The decoding algorithm uses soft information as well as the hard information from color calls when determining the most probable base at each position of decoded sequence and this is a possible explanation for the small difference between the performance of the codes. The equivalence classes narrow down the possible errors but, rather than randomly picking the correction, MAP decoding uses the quality information to guide the choice and the right correction might be picked the majority of the time even without this extra assistance.

Conclusions

The addition of the Exact Call Chemistry to the SOLiD platform enables many sequencing errors to be detected that would otherwise would pass unnoticed; this in itself provides useful information about the accuracy of reads. Without using quality information, the ability of the Exact Call Chemistry to correct sequencing errors is limited but the number of possible options can often be drastically reduced. The quality of the calls for the encoded sequence can then be used to choose between the options for correction, leading to genuine ability to correct sequencing mistakes. The five-base length of the block places a limit on how frequently errors can occur before the encoding can no longer offer protection. In our simulations, the number of perfect color-space reads increases by 27% when error correction is performed, the majority of these corrections being single errors that might otherwise degrade the base-space translation of the read. The error correction capability offered by the ECC results in measurable gains for the SOLiD platform but there is a trade-off between the extra time and expense and the improvements possible through increasing coverage using additional sequencing runs.

The complex triple and quadruple errors that can be induced in the final nucleotide reads by miscorrection of

Table 6 Number of errors for simulated data

Space	Probe generator	Percentage mapped	Percentage mapped with 0 – 5 errors					
			0	1	2	3	4	5
Color	None	77.1	38.4	16.7	10.5	6.2	3.6	1.8
"	$1\beta0\beta0$	78.4	48.6	10.2	9.7	5.0	3.2	1.7
"	$10\beta00$	78.3	48.4	10.3	9.7	5.0	3.2	1.7
"	$1\beta010$	78.3	49.8	9.1	9.7	4.9	3.2	1.7
Base	None	47.2	38.4	2.3	1.6	1.8	1.8	1.2
"	$1\beta0\beta0$	64.3	49.6	4.7	2.1	3.1	2.6	2.1
"	$10\beta00$	65.5	49.4	6.2	3.0	2.3	2.4	2.2
"	$1\beta010$	64.9	50.8	5.0	2.2	2.4	2.4	2.2

Percentage of reads mapped (five edits or fewer), and mapped with a given number of errors, for one million simulated reads using the codes with probe generators as specified. For comparison, figures are also given using only the two-base-encoding probe set (probe generator 'None'). Since color-space reads have their first position trimmed before mapping to produce reads 49 colors long, the percentages of mapped reads and reads with a given number of errors are slightly inflated compared to those given for base-space reads.

the observed sequence are not well represented by the Phred error model [9], where each site is assigned an individual quality independent of other sites. This may have consequences for downstream analyses that assume the Phred model is a good representation of the probability that a particular site is in error. The patterns of error induced by the largest equivalence class of the ECC code may allow some sequencing errors to be distinguished from genuine variants after mapping to a nucleotide reference. The power of this approach is unlikely to be good and so it will not be an adequate replacement for mapping and variant calling using a more appropriately encoded reference. One possibility would be to encode the reference using the ECC code and match directly against that, using all the available information but making variant calling difficult due to the complex structure of the data. A simpler alternative would be to decode into two-base-encoding colors, using the extra ECC calls to correct the color calls. The corrected color reads can then be mapped against a color-encoded reference using the many tools already developed.

The variation in the ability of the ECC to correct errors at different positions in each block suggests a simple improvement to the platform that would enable it to recover from problematic rounds. Consider, for example, that one of the initial five rounds has experienced some form of gross failure, perhaps due to a bad wash or incomplete melting of the previous primer and sequence from the template, so that every cycle in that round is of poor quality. The priming of the final round could be adjusted to start at a different position of each block, chosen to maximise the chance of correcting previous errors. For the ECC code, the priming would be chosen to ensure that calls from the bad round coincide with positions two or three of the final round (the equivalence class with two elements); the alternative code generated by $1\beta010$ would be primed so that the errors coincide with the first position and correction could be guaranteed. This technique could also help recover from transient errors, like bubbles in the buffer preventing calls being made for a large number of clusters on a particular cycle.

The alternative codes that we describe have equal or superior error correction ability to that of the ECC but the creation of a new probe set is a considerable investment and a more thorough search of alternatives codes should be undertaken before the expense is incurred. All the alternative codes considered satisfy a number of restrictions, several of which could be relaxed to improve the error correction properties. Firstly, more rounds could be used, slowing the sequencing process down but providing more redundancy with which to correct errors. Secondly, the architecture of the encoder was constrained to be backwards compatible with the two-base-encoding sequencing chemistry, completely defining one unpunctured stream. A completely new chemistry could vary both probe sets and also change the puncturing matrix, allowing much more flexibility over the design of the code architecture and potentially creating codes that are capable of unambiguously correcting multiple errors.

While many sequencing errors can be corrected, their occurrence is non-uniform, being more frequent in the later cycles of each round. This leads to a tendency for the poorest reads to contain multiple errors in close proximity. Such bursts of errors cannot be successfully corrected, so the advantage of the ECC is limited to generally high quality reads containing a few sparsely distributed errors. Perhaps the major advantage of the new chemistry is that the sequencing platform can produce reliable nucleotide sequence, without the possibility of a single error causing a catastrophic decoding failure, which allows the reads produced to be analysed with the wealth of tools available that assume nucleotide sequence.

Software

Software reimplementing the decoding algorithm for the two-base- and four-base- encodings (SOLiD ECC), as well as the alternative encodings described, on a trellis to find both the maximum likelihood (Viterbi decoding) and maximum a posteriori [3] (forwards/backwards decoding) nucleotide sequence is available at http://www.ebi.ac.uk/goldman-srv/solid/, distributed under version 3 of the GNU General Public Licence, as is software to simulate two- and four-base encoded reads and some utility functions to manipulate convolutional codes in GF_4. This software is provided solely for the purposes of reproducibility.

Competing interests

This work was supported in part by Wellcome Trust Technology Development grant WT088151MA and a grant to the EBI from Life Technologies Corporation. Life Technologies Corporation had no input into the development of the study, article preparation or decision to publish. No other competing interests are declared.

Acknowledgements

The authors give their thanks to Ewan Birney for suggesting a study of the ECC, and to Marcin Sikora and Alan Blanchard of Life Technologies Corporation (Foster City, CA) who did the original work on the application of convolutional codes to sequencing. TM would like to thank the staff at Life Technologies Corporation for their hospitality during a short visit. The real sequencing data used for the analyses in this article was provided by Life Technologies Corporation. SOLiD® is a registered trademark of Life Technologies Corporation.

Authors contributions

The study was conceived and planned by both authors. The analysis of the Exact Call Chemistry coding was done by TM, who also drafted the manuscript. Both authors read, edited and approved the final manuscript.

Received: 31 January 2012 Accepted: 22 June 2012
Published: 22 June 2012

References

1. Metzker ML: **Sequencing technologies – the next generation.** *Nat Rev Genet* 2010, **11:**31–46.

2. Breu H: **A theoretical understanding of 2 base color codes and its application to annotation, error detection, and error correction.** White paper, Life Technologies 2010, [http://www3.appliedbiosystems.com/cms/groups/mcb_marketing/documents/generaldocuments/cms_058265.pdf. [Accessed: 7 Dec. 2011].
3. Applied Biosystems: **SOLiD System accuracy with Exact Call Chemistry module.** White paper, Life Technologies 2011, [http://www3.appliedbiosystems.com/cms/groups/global_marketing_group/documents/generaldocuments/cms_091372.pdf. [Accessed: 7 Dec. 2011].
4. Erlich Y, Mitra PP, de la Bastide, M, McCombie WR, Hannon GJ: **Alta-Cyclic: a self-optimizing base caller for next-generation sequencing.** *Nat Methods* 2008, **5:**679–682.
5. Yuen JH, Vo QD: **In search of a 2-dB coding gain.** In *TDA Progress Report 42-83, Jet Propulsion Laboratory, Volume July-September 1985*, Jet Propulsion Laboratory 1985:26–33. [http://ipnpr.jpl.nasa.gov/progress_report/42-83/83C.PDF. [Accessed: 7 Dec. 2011].
6. Viterbi AJ: **Convolutional codes and their performance in communication systems.** *IEEE Trans Commun Technol* 1971, **19:**751–772.
7. MacKay D: *Information Theory, Inference, and Learning Algorithms*. Cambridge: Cambridge University Press; 2003.
8. Cohn PM: *Algebra, Volume 1*. second edition. Chichester: Wiley-Blackwell; 1982. [http://goo.gl/maps/44D0].
9. Ewing B, Green P: **Base-calling of automated sequencer traces using *phred*. II. Error probabilities.** *Genome Res* 1998, **8:**186–194.
10. Hamming RW: **Error detecting and error correcting codes.** *Bell Syst Technl J* 1950, **29:**147–160.
11. Johnson SM: **A new upper bound for error-correcting codes.** *IRE Trans Inf Theory* 1962, **8:**203–207.
12. Singleton R: **Maximum distance q-nary codes.** *IEEE Trans Inf Theory* 1964, **10:**116–118.
13. Gilbert EN: **A comparison of signalling alphabets.** *Bell Syst Techn J* 1952, **31:**504–522.
14. Varshamov RR: **Estimate of the number of signals in error correcting codes.** *Dokl Acad Nauk SSSR* 1957, **117:**739–741.
15. Berrou C, Glavieux A, Thitimajshima P: **Near Shannon limit error-correcting coding and decoding: turbo-codes.** In *IEEE International Conference on Communications, Volume 2*, IEEE 1993:1064–1070.
16. Gallager RG: **Low-density parity-check codes.** *IRE Trans Inf Theory* 1962, **8:**21–27.
17. Li H, Durbin R: **Fast and accurate short read alignment with Burrows-Wheeler transform.** *Bioinformatics* 2009, **25:**1754–1760.

doi:10.1186/1471-2105-13-145
Cite this article as: Massingham and Goldman: Error-correcting properties of the SOLiD Exact Call Chemistry. *BMC Bioinformatics* 2012 **13**:145.