Published in final edited form as:

Commun ACM. 2013 January ; 56(1): 83–93. doi:10.1145/2398356.2398376.

Abstractions for Genomics

Vineet Bafna,

Computer Science and Engineering, UC San Diego

Christos Kozanitis,

Computer Science and Engineering, UC San Diego

Alin Deutsch,

Computer Science and Engineering, UC San Diego

Lucila Ohno-Machado,

Division of Medical Informatics, UC San Diego

Andrew Heiberg, and

Computer Science and Engineering, UC San Diego

George Varghese

Computer Science and Engineering, UC San Diego

We are a product of “nature and nurture”. By this, we mean that our *phenotype* (the composite of all outward, measurable, characteristics including our health parameters) is a function of two things: our *genotype* (the DNA program inside all cells), and the environment (all inputs to a human such as food and medicine). This is analogous to how the output of a program (e.g., a search engine) is a function of both the program and the input (keywords typed by user). Using the same input with a different program (e.g. Google search vs. Bing) can result in different output.

In this analogy, the role of the medical professional is to provide information that is either “diagnostic” (i.e. is there a bug in the program based on observed output?), “prognostic” (e.g., can we predict the output/outcome, given specific inputs such as diet?), or “therapeutic” (e.g., can any specific input such as a drug lead to the desired output). Also, the Electronic Medical Record (EMR) of a patient can be described as an archive of previously acquired inputs and outputs.

Unlike computers, the human “program” was largely hidden away. As a consequence, medicine has traditionally been “depersonalized” with doctors providing treatments by comparing the patient’s phenotype (symptoms) against empirical observations of outputs from a large number of individuals. There was only limited customization based on coarse classes such as ‘race’. All of this changed with the sequencing of the human genome in early 2000, and the subsequent drop in costs from hundreds of millions of dollars down to \$1000 on small desktop sequencing machines. The ability to cheaply read the program of each human underlies the great promise of *personalized medicine*— treatments based on symptoms and the patient’s distinctive DNA program.

We frame this point with a classic example. The blood-thinner drug, Warfarin, is widely prescribed to prevent blood clots. The dosage is critical: too high, and the patient can bleed to death; too low and the drug may not prevent life-threatening blood-clots. Often, the right dosage is established through multiple visits to the clinic and regular testing. However, recent reports [16] suggest that knowledge of the patient's genetic program can help establish the right dosage. This approach (Genetic association or a *discovery workflow*) is outlined below.

- Collect a sample of affected, and 'genetically matched' control individuals; sample DNA, and catalog variations.
- Identify and report variations that co-segregate (i.e., correlate) with the affected/control status of the individual.
- Follow up on the genetic basis of the correlation through expensive follow-up studies and experiments in animal models and clinical trials. Transfer knowledge to the clinic.

Even with its successes, the discovery approach has problems. First, studies are resource-intensive and require identifying and sequencing large cohorts of individuals with and without a disease. Second, it is unclear how to apply the study results to a specific individual, especially one that is genetically different from the investigated cohort. Finally, data reuse is difficult: significant computation must be expended to dig out data from a previous study and much care is required to reuse the data. We contrast the discovery workflow with *personalized medicine*. Here, a physician treating individual *A* may query a database either for: treatments suitable for patients who have genetic variations similar to those of *A*; or query for patients genetically similar to *A* for treatments and dosages that worked well for these patients.

The notion of computational tools enabling personalized medicine is gaining currency. Some groups have proposed specialized databases for cancer genomes (e.g., [14]) although details are still emerging. Here, we take a more general viewpoint, arguably allowing for broader access to genomic information, and enabling both discovery and personalized medicine.

Specifically, we start with a shift in perspective implicit in the personalized genomics community. Instead of sequencing small cohorts on a need-to basis, we assume that individuals are sequenced at birth, and their personal genome is a part of their EMR (medical record), and available to be queried by researchers and medical personnel. This scenario is realistic considering rapidly dropping sequencing costs. We claim (along with others) that this shift in perspective (see Figure 1) enables personalized medicine and largescale discovery.

In the example of choosing Warfarin dosage for a patient, the medical team might do the following: (a) identify a collection of individuals genetically similar to the patient and on a warfarin regimen; (b) query their genomes and EMRs for genetic variation in candidate genes and warfarin dosage, respectively; and, (c) choose the appropriate dosage based on the patient's specific genetic program. The ability to logically select from a very large database

of individuals using the phenotype as a key removes the first problem described with the discovery workflow. Using genetic variations specific to an individual A as a key to return treatments (that work well for such variations) addresses the second problem. Finally, if the accompanying software system has good abstractions, then the third problem (computational burden to reuse data) is also greatly eased. Our paper focuses on proposing key software abstractions for genomics— we suggest that like other CS areas (VLSI/Systems), software abstractions will enable genomic medicine.

We start by describing basic genetics using programming metaphors in Section 1. Section 2 describes trends in sequencing. We describe how genetic variations are called today in Section 3. Our vision for a vast genomic database built in layers is described in Section 4—the key idea is the separation of ‘evidence’ and ‘inference’. In Section 5, we propose a language for specifying genome queries. We end in Section 7) by outlining research directions for other areas of computer science to further this agenda.

Scope

We limit our scope to genomics, ignoring dynamic aspects like transcript, protein expression, and pathway databases. Genomic information is traditionally analyzed using two complementary paradigms. First, *comparative genomics*, where different species are compared— most regions are dissimilar, and the conserved regions are functionally interesting [7, 6]. The second approach is *population genomics*, where genomes from a single population are compared under the baseline hypothesis that the genomes are identical and it is the variations that define phenotypes and are functionally interesting. Our paper focuses on population genomics and its application to personalized medicine. Finally, we do not discuss specific sequencing technologies such as strobe sequencing versus color space encoding.

1. GENETICS FOR COMPUTER SCIENTISTS

We provide a quick introduction to genetics for computer scientists; a standard reference such as [1] can provide more details.

All living creatures consist of cells; each cell is like a computer running a program which is its DNA. The program uses 3 billion characters (*nucleotides/basepairs*) from a fixed alphabet $\{A, C, G, T\}$. Humans are *diploid* — there are two programs controlling each cell, one inherited from the father and one from the mother. Further, each program is broken up into 23 “modules” called chromosomes and within each chromosome are sparsely scattered small functional blocks called genes. The module pairs from the father and mother are called homologous chromosomes and each human has a pair of (homologous) genes from each parent.

The ‘hardware’ of the cell consists of cellular organelles and proteins— the cellular machinery. Proteins perform specific cellular functions such as catalyzing metabolic reactions and transducing signals. The gene contains the ‘code’ for manufacturing proteins. Each gene executes in one of many ‘ribosomes’ (analogous to a CPU). Information travels from the nucleus (where the packaged DNA resides) to the ribosome via a ‘messenger’

molecule (mRNA) that is essentially a copy of the coding DNA. The ribosome ‘reads’ the code 3 bases (one codon) at a time. Each codon is analogous to an OpCode instructing the ribosome to attach a specific amino acid to the protein sequence being constructed. Thus, the DNA program provides instructions for making the hardware which in turn performs all cellular functions.

A change (*mutation*) in the DNA can change the amino-acid, and correspondingly, the cellular machinery resulting in a different phenotype (output). In the Mendelian paradigm, each of the two homologous copies of the gene controls one *phenotypic trait* (e.g., ability to clot blood). A mutation in one gene might impact the phenotype strongly (*dominant*), not at all (*recessive mutation*) or somewhere in between. In fact, most phenotypes are *complex*—controlled by the paired copies of multiple genes. Nevertheless, DNA controls traits so even the simplest queries on DNA are useful — e.g., *Compared to a ‘normal’ individual, are there parts of the patient’s DNA program that have mutated.*

Advancements in sequencing has made it possible to cheaply scan an individuals genetic program for mutations (or, *variations*). First, a physical process is used to randomly shear genomic DNA into small *inserts* of size 500–1000bp. The sequencing machine deciphers the DNA from small *fragments, or reads* (length $L \simeq 100\text{bp}$) at one or both ends of the inserts. Thus, the genomic information is presented as a collection of small strings over A,C,G,T, sampled from a random location on a (maternal or paternal) chromosome. It is natural to assume that these fragments will be assembled like a giant jigsaw puzzle. However, this is complex and expensive because of the large amounts of repetitive portions in human genomes.

Mapping, and Variation

An alternative to assembly used today is to *align*, or *map* the non-repetitive fragments of a sampled genome (the *donor/patient genome*) to a reference human genome. The current reference is a single (haploid) copy of each chromosome sampled from multiple individuals. Mapping involves finding a location on the reference where the genomic substring matches the query fragment up to a small number of errors. The errors might be sequencing errors or true variations in the query, relative to the reference. Mapping works because string search is more tractable than assembly and any pair of human genomes is identical to 1 in 1000 locations.

A true deviation in the donor sequence relative to the reference is called a *variation*. The simplest variation is a *SNV* (single nucleotide or single character variation). Recall that donor genome consists of two copies. A variation that occurs in both copies is called *homozygous*; a variation that occurs in only one copy is called *heterozygous*. The specific value of the variant is called an *allele*. For example, suppose that the DNA at homologous chromosomes of individual *A* compared to the reference is as follows

... ATG ... GAGTA ...	Reference Assembly
... ACG ... GAGTA ...	Maternal chromosome 1
... ATG ... GAGCA ...	Paternal chromosome

Then, individual A is bi-allelic, or heterozygous at the 2 SNV sites, has the *genotype* ... C/T ... C/T ..., and the genotypes are resolved into two *haplotypes* ... $C \dots T \dots$, ... $T \dots C \dots$

Sites containing SNVs that are prevalent in a population demarcate chromosomal positions as varying, or polymorphic. Consequently, these locations are called *SNPs* (Single Nucleotide Polymorphisms). In discovery workflows, we test populations to see if the occurrence of variation correlates or *associates* with the phenotype status of the individual.

So far we have talked of simple variations involving one, or a small number of changes at a location. By contrast, we also have *structural variations* in which large (one kbp to several million bases) genomic fragments are deleted, inserted, translocated, duplicated, or inverted, relative to the reference [19].

2. SEQUENCING TRENDS

The following trends are relevant to designing a genomic software architecture:

- *Reducing costs:* While the Human genome project cost $\sim \$100M$, human re-sequencing costs for a redundant (15 $\times$) coverage are now under \$5K, and projected to go below \$1K. This implies that universal sequencing may be realizable, and archival and analysis, not sequencing, will dominate cost.
- *Small Read lengths:* New sequencing technologies have largely sacrificed length and sequence quality to massive parallelism and high throughput. As ‘single-molecule’ sequencing technologies emerge, reads may become long enough ($\sim 100Kbp$) to allow *de novo* assembly. Nevertheless, raw reads will continue to remain first class entities.
- *Prohibitive Assembly Costs, Paired end sequencing:* Repetitive regions cover $\sim 40\%$ of the human genome. If the read-length is shorter than the length of repetitive sequence, the genome cannot be assembled uniquely. Longer reads or reads sequenced from ends of long clones (paired end reads) are necessary to resolve repeats and assemble sequences *de novo*. Thus, today sequenced reads are mapped to a standard human reference to identify variations, and these variations are correlated to phenotype variation.
- *Computer System Costs will soon dominate:* Some studies [18, 15] have shown that the costs of disk storage for genomes is now greater (and decreasing slower) than the costs of sequencing.

We start with some exemplar queries on genomic data that illustrate the difficulty of genomic analysis and show that there is no consensus on the best method. Abstractions must be flexible to handle a variety of methods.

3. VARIATION CALLING

The key to efficiency in genomics is the premise that an individual’s genetic record can be summarized succinctly by a much smaller list of individual genetic variations. While we develop this premise further in our layering proposal (Figure 3a)), we provide the reader

with intuition as to how variants are called *today*. The expert should skip this section and proceed to our layering proposal. We start with querying for Single Nucleotide Variations (SNVs) in the donor genome, the simplest form of variation.

3.1 Calling SNVs

Figure 2 shows how a mutation may be called. Consider the reference allele *C*. We see two copies of the donor genome with a *G* allele, and some copies with a *C*, indicating a heterozygous SNV. If the variation is homozygous, we expect all overlapping READs to have a *G* in that position. Even this simple call can be confounded. First, some of the reads may have been mapped to the wrong place on the reference; such as the top donor read in the figure. The G/T mutation may not be correct and the alignment of the incorrectly mapped read might present many variations. Even if the READ is mapped correctly, sequencing errors could incorrectly appear as heterozygous mutations.

Thus, mutation callers use various statistical methods informed by mapping quality (e.g., the number of potential places in the genome a read can map to) of the read, the quality score of a base call, and the distribution of bases or alleles in the READs for that location. Some mutation callers even use evidence based on the surrounding locations (e.g., an excess of insertion/deletion events nearby suggests alignment problems). The decision itself could be based on frequentist, Bayesian inference, or other machine learning techniques. While SNP callers use various inference techniques, they all use the same evidence: the set of reads overlapping the location of the SNP in question.

3.2 Calling Structural Variations

In addition to small nucleotide changes, larger structural variations involving insertion, deletion, and translocation of large genomic regions also form an important source of genomic variation. Consider the example of deletions in the donor genome (Figure 2b) in which a large segment of DNA is deleted, relative to the reference). If both copies of the donor genome are deleted, the deletion is *homozygous*; otherwise, it is *heterozygous* deletion. Deletions can be detected using a number of techniques:

Paired-end mapping—In Paired-end sequencing, both ends of large genomic fragments (sampled randomly from the donor genome) are sequenced. These genomic fragments have been size-selected to be tightly distributed around a specified length L ($\simeq 500$). If paired reads end up mapping much further apart than L (*length-discordance*), we can infer a deletion in the donor relative to the reference (e.g., read ‘a’ in Figure 2b). If the deletion is heterozygous, we should see a mix of concordant and discordant reads at the breakpoints of the deletion.

Depth of coverage—“Depth” at a position refers to the count of reads mapped to the position. Deleted regions of the donor chromosome will have reduced coverage — roughly half for heterozygous deletions and zero for homozygous ones. Thus, read ‘b’ in Figure 2b maps within the deleted region, but reads a,c do not.

Single end mapped, and split-reads—When a read maps to the breakpoint of the deletion on the donor, it cannot be mapped back to the reference (Figure 2b, read ‘c’). In the case of a ‘clean’ deletion, the prefix and suffix of the fragment could be mapped separately, and such split-reads are indicative of deletion events.

Loss of heterozygosity—Consider the SNV locations on the donor genome. While sampling multiple polymorphic sites, we expect a mix of heterozygous and homozygous sites. At a deletion, the single chromosome being sampled displays a loss of heterozygosity.

Even within these four categories, a number of design decisions are made by individual tools to account for repetitive sequences, and reconcile conflicting evidence. Variant inference remains a challenging research problem.

4. LAYERING FOR GENOMICS

Our vision is inspired by analogy with systems and networks. For example, the Internet has successfully dealt with a wide variety of new link technologies (from fiber to wireless) and applications (from email to social networks) via the “hourglass” model using the key abstractions of TCP and IP (Figure 3a).

Similarly, we propose that Genomic Processing software be layered into an instrument layer, a compression layer, an evidence layer, an inference layer, and a variation layer that can insulate genomic applications from sequencing technology. Such modularity requires us to forgo efficiencies that can be gained by leaking information across layers. For example, biological inferences can be sharpened by considering which sequencing technology is being used (e.g., Illumina versus Life Technologies) but we suggest that modularity is paramount.

Some initial interfaces are in vogue. Many instruments now produce sequence data as ‘fastq’ format. The output of mapping reads is often represented as SAM/BAM format, although other compressed formats are being proposed [10]. At a higher level, standards such as VCF are used to describe variants (See Figure 3a).

We propose additional layering between the mapped tools and applications. Specifically, we separate the collection of *evidence* required to support a query (deterministic, large data movement, standardized) from the *inference* (probabilistic, comparatively smaller data movement, little agreement on techniques). We assert that while Inference methods vary considerably, the Evidence for inferences is fairly standard. To gather this evidence in a flexible and efficient manner, we propose a Genome Query Language (GQL).

While we do not address it, a careful specification of a variation layer (Figure 3a) is also important. While the data format of a variation is standardized using, for example, VCF, the interface functions are not.

4.1 The case for an evidence layer

Genomes, each several hundred gigabytes long, will be produced at different locations around the world. To realize the vision of Figure 1, individual laboratories at other parts of the world need to process this genomic information to reveal variations and correlate them

with medical outcomes/phenotypes at each place a discovery study or personalized medicine assay is undertaken. The obvious alternatives are not workable, as described below.

Downloading raw data—Transporting $\sim 100\text{Gb}$ for each of ~ 1000 genomes across the network is infeasible today. Compression can mitigate ($5\times$), but not completely avoid this problem. Finally, massive computational infrastructure, must be replicated at every study location for analysis.

Downloading variation information—Alternatively, the genomic repositories could run standard variant calling pipelines [4] and produce much smaller lists of variations in a standard format such as VCF. Unfortunately, variant calling is still an inexact science; researchers often want to use their own callers and almost always want to see the “evidence” for specific variants. Thus discovery applications are very likely to need the raw genomic evidence. By contrast, personalized genomics applications might only query called variants and a knowledge-base that correlates genotypes and phenotypes. However, even medical personnel may occasionally need to review the raw evidence for important diagnoses.

Our approach provides a desirable compromise. We *allow the retrieval of evidence for variations on demand using a query language*. The query server itself will utilize a large compute (cloud) resource, and implement a query interface that can return the subset of reads (the *evidence*) that support specific variations. Indeed, some recent approaches have hinted at this evidence layer, including SRA, and Samtools, but in a limited scenario useful mainly for SNV/SNP calling. TABIX extends the vertical slicing in Samtools to specifying ranges for multiple file formats (not just BAM files). However, our approach (GQL) goes further and allows querying for intervals with putative structural variation (e.g., with discrepant READs supporting a deletion) or copy number changes. GQL also supports multiple types of inference, changing definitions of variation, and pooling of evidence across instrument types.

Consider a complex biological query: “identify all deletions that disrupt genes in a certain biological network, and the frequency of those deletions in a natural population”. For *any statistical inference algorithm*, the evidence would consist of mapped reads that satisfy certain properties, including (a) length-discordant reads; (b) reads with reduced depth of coverage; and, (c) reads with one end unmapped. The EL will support queries to get these reads.

- The separation allows Inference Layer designers to start thinking of alternate forms of evidence to improve the confidence of their queries, such as split-end reads that map to the deletion breakpoints.
- The EL often poses a ‘data bottleneck’ as it involves sifting through large sets of genomic reads. By contrast, the inference layer may be compute intensive, but typically works on smaller amounts of data (filtered by the evidence layer). We can implement EL on the cloud while the Inference Layer can be implemented either on the cloud or on client workstations.

- A standardized EL will allow vendors time to creating a fast and scalable implementation. By contrast, the Inference Layer today is a moving target.

In the following, we develop this intuitive idea further by describing a Genome Query Language to support the Evidence Layer.

5. QUERYING THE GENOME VIA GQL

We want a query language that is: ‘complete’ (i.e., capable of handling all evidence level queries), efficient (The query language must have an efficient implementation), easy to express and using standard I/O. Ideally, it should select from a reads and output a subset of reads in a standard format, like BAM. We propose using a standard SQL like syntax (SELECT from READS WHERE condition) which will be simple and easy to use and familiar for most programmers. However, we argue that using a standard relational database does not work well.

We have two fundamental types of relations. The first are *Reads* mapped to the human genome often expressed as BAM files. Second, we have tables of intervals representing ‘interesting’ (functional) regions of the genome. While simple selection queries are exactly as in relational languages (Select From Reads Where mapped pairs are far apart), many useful queries require a form of joining of relations using interval intersection and not equality as the join operator. For example (also see Figure 4), one might want to join a relation consisting of READS that map to exons in a specific gene, but where the paired-ends are far apart (indicating a deleted exon). We define a special *MapJoin* operator to allow this. We note that some of the newer technologies allow for multiple reads to be generated from the same physical clone. While not explicitly discussed here, the relations can easily be extended to this case. We also find it extremely helpful to have a third operator we call Project Interval, which computes the largest continuous interval containing a merged representation of what may be a number of smaller intervals. Examples of queries using these operations are described below.

While geometric queries were considered for GIS databases, genomics provides a sufficiently important domain to warrant database support from the ground up. We thus define a language called GQL and show it can be used to write a number of biologically interesting queries. GQL is based on a *Genome Query Algebra-GQA* described in the appendix.

5.1 Sample Queries

We have gathered many queries from practicing biologists to understand the power of GQA. Here, we show some examples of expressive power of the algebra. In a companion technical paper, we show that GQA’s expressive power captures the language of First Order Logic over the relations above, as well as a signature of aggregation functions.

1. What is the genotype at a specific position (e.g., SNV)?

Query: Let A denote a relation containing the single point interval $\langle \text{chr}, i, i \rangle$. The evidence for the genotype at the position is provided by alignments of READS that map to the location, and we can either query for the mapped reads, or for the

alignments themselves, which are often stored as a mapped read attribute (e.g., $R.A_{\text{ALIGNSTR}}$). Thus:

GQL: $\text{SELECT } R.ID, R.A_{\text{ALIGNSTR}} \text{ FROM MAPJOIN } R, A$

2. What are the diploid haplotypes (phased genotypes; see Section 1) across a set of linked loci in a dataset?

Query: This is a harder query. To assemble haplotypes we need a collection of reads each of which (perhaps along with their paired-end reads) connect at least two polymorphic sites. Let attribute $R.CloneId$ denote the clone identifier so that the paired-end reads r_1, r_2 derived from the same clone satisfy $r_1.CloneId = r_2.CloneId$. Also, let relation S denote the collection of point intervals, one for each variant locus.

- a. Find a subset of READS mapping to the loci, and the count of sites the reads or their paired-ends map to (call this count c):

GQL: $RC = \text{SELECT } R.CloneId, c = \text{count}(\ast) \text{ FROM MAPJOIN } R, S$
 $\text{GROUPBY } R.CloneId$

- b. Return IDs of reads with count ≥ 2 :

GQL: $\text{SELECT } R.ID \text{ FROM } R, RC$
 $\text{WHERE } R.CloneId = RC.CloneId \text{ AND } (RC.c \geq 2)$

3. What genomic loci are affected by Copy Number Variations (CNVs)?

Query: If the number of donor reads mapping to a region exceeds some threshold T then the inference might be that the region has been duplicated in the donor genome. Such CNVs have been implicated as an important variation for many disease phenotypes. To gather evidence, we would like all of the intervals where the number of mapped reads exceeds threshold t (for example). Let $G.loc$ denote a specific chromosome and location.

- a. Compute for each location, the number of reads that map to the location:

GQL: $V = \text{SELECT } G.loc, c = \text{count}(\ast) \text{ FROM MAPJOIN } R, G \text{ GROUPBY } G.loc$

- b. Return all ‘merged regions’ where the read count exceeds threshold t .

GQL: $\text{SELECT MERGED } RS.loc \text{ FROM } V \text{ WHERE } V.c > t$

4. Identify all regions in the donor genome with large deletions.

Query: As discussed earlier, the evidence for deletion comes from a variety of sources. We use *discrepant* paired-end mapping. Paired-end reads from clones of length 500 (for example) should map $\simeq 500\text{bp}$ apart on the reference genome. If instead, the ends happen to map *discrepantly far* (e.g., l apart for some $l \gg 500$, like $l \simeq 10000$), they support the case for a deletion in the donor genome. Thus, our goal is to identify all regions with at least t discrepant paired-end reads.

- a. Use a join in which each record contains the mapping locations of the read as well as its paired-end.

GQL: READS already contains this join.

- b. Select records containing discrepant reads.

GQL: $H_2 = \text{SELECT } * \text{ FROM READS WHERE } \text{abs}(\text{loc} - \text{rnateloc}) > 10000$

- c. Select intervals containing at least t discrepant reads.

GQL: $\text{SELECT MERGED G.loc FROM } H_2 \text{ GROUPBY G.loc, c=count(*) WHERE } C > t$

5.2 Population based queries

The true power of querying genomes comes from the ability to query populations. Indeed, existing tools (Samtools) allow for the ability to extract reads from multiple individuals at specific locations corresponding to polymorphic sites. We want to extend the full power of genomic queries to interrogate populations. Recalling the warfarin example, we would like to query for warfarin dosage and genetic variation in candidate genes (genes identified through a discovery work flow) among individuals on the warfarin regimen. Specifically, we could be interested the following: **Colloquial:** *Report warfarin dosage and genomic intervals and reads in individuals s.t. the copy number of mapped reads is at least twice the expected coverage in the interval.*

The query for this would be similar to that of a single individual, but repeated via a ‘join’ with a population P , using

GQL: $\text{SELECT } * \text{ FROM } P, \text{MAPJOIN } R, E \text{ WHERE } P.WD = \text{TRUE}$

Using previous arguments, we can count the read depth, and report high CNV regions. A similar idea applies to a *personalized workflow*, where we would be interested in copy numbers similar to the specific query individual.

Group Inference without Accurate Individual Inference—The ability to query populations has an important benefit. Often, individual genomes may have very little evidence for SNV calls at low-coverage sequencing. However, if a large number of affected individuals in a population (e.g., 800 out of 1000) all show the same SNV, while controls do not, we can reliably predict an association even with unreliable calls for individuals. While more work is required to demonstrate the benefits of group inference, the important point is GQL provides query support for group inference.

6. A PROTOTYPE IMPLEMENTATION

We have developed a prototype implementation, *genomequery* of a subset of GQL (Figure 5). The uploaded genome (in BAM format) can be queried using a simple text interface that allows the user to write a GQL query. The query is compiled and executed, and the output returned as a (smaller) BAM file that can be visualized using jbrowse, or downloaded to the client for further analysis.

The implementation of *genomequery* has a customized parser that converts GQL to an intermediate representation resembling GQA. Thus, there are customized procedures for each of the algebraic operations, with some concessions for efficiency (mainly w.r.t memory). Specifically, we use interval trees to implement $\bowtie$, and customized indices (including Strength Vectors; see below) for efficient querying. Details on the implementation will be provided elsewhere.

7. THE WAY FORWARD: CHALLENGES FOR COMPUTER SCIENTISTS

So far we have made the case for a set of genomic layers, including an Evidence Layer where evidence is retrieved using a Genome Query Language. Successful implementation of this vision will depend upon some new ideas from computer science: specific areas are marked in the subsection titles

7.1 Query Power – Database Theory

Is GQL sufficiently powerful to address all Evidence Layer queries needed in practice? The goal is to have the Evidence Layer handle as much of the data-intensive computation as possible while preserving performance; without the performance goal, any query can be trivially satisfied by passing the entire genome to the inference layer. It helps to note that GQL's expressive power coincides with that of first order logic over the schema of the three relations R, G, P , a signature of aggregation functions, and a group-by operator. However, user feedback may require extensions. In implementing extensions, care must be taken to balance expressive power with efficient evaluation.

7.2 Query Speed — Database Systems

We have designed a corresponding algebra GQA as an internal representation for optimization and evaluating query plans. As an example, assume that queries on populations will automatically be decomposed into queries on individuals. Consider queries of the general form `SELECT  $a$  FROM MAPJOIN  $R, G$  WHERE  $b$` . The two steps are

1. Select for relations that satisfy constraints b .
2. Project (while removing duplicates) on to attributes a

We use a location based index L_{toR} , where $L_{toR}(l)$ is a pointer to first read that maps to the point-interval l . For each individual, we keep a compressed index of the mapped reads in memory. The index can be used for Select operations based on specific locations (e.g, reads that map to specific genes).

However many queries involve scanning the entire genome for maximal intervals. For example, find all maximal regions where there is a disproportionate increase in the number of mapped reads (High copy number). For efficient implementation of these queries, we construct special indices that allow filtering for READS according to a user-defined constraint. Define a *strength vector* S_θ for a constraint θ as a vector of length G (the entire genome). For any location $l \in G$, $S_\theta[l]$ gives the strength of the evidence at that location, and can be precomputed for common constraints θ . To reduce memory, we also choose a minimum strength cut-off, and maintain $C_{\theta,t}$ as a sorted sequence of intervals $i_1, i_2, \dots$ such

that each i_j is a maximal interval satisfying $S\theta[l] \leq t$ for all $l \in i_j$. The compressed vectors reduce memory and computation requirements and can be recomputed on demand.

7.3 EMRs – Information Retrieval

The phenotypes associated with each sequenced individual are already in patient medical records. Initial results from the eMERGE network indicate that, for a limited set of diseases, it is possible to utilize the EMR for phenotype characterization in genome wide association studies (GWAS) within a reasonable margin of error[9, 13]. It is anticipated that most healthcare institutions will be utilizing EMRs by 2014, given incentives provided by the HITECH act[17]. Increasing adherence to interoperability standards[5] and advances in biomedical natural language processing[12], make efficient querying possible. However, there is currently no integration of genotype and phenotype data. GQL should be useful both for interrogating a single genome or interrogating several genomes across groups of individuals, but will need to integrate with existing EMR systems so that phenotype data could be queried together with genomes.

7.4 Privacy – Computer Security

The genome is the ultimate unique identifier. All privacy is lost once we have access to the genome of an individual, but the current regulation, the Health Information Portability and Accountability Act (HIPAA) is silent about this identifier[2, 3, 11]. Although the Genetic Information Nondiscrimination Act (GINA) addresses accountability for use of genetic information[8], privacy laws will need to change to ensure that sensitive information is available only to the appropriate person. Checking that a given study satisfies a specific privacy definition requires formal reasoning about the data manipulations that generated the disclosed data, which is impossible without a declarative specification (such as GQL) of such manipulations.

7.5 Provenance – Software Engineering

GQL is an ideal vehicle for recording provenance of study conclusions. Current scripts often consist of code that is often too ad-hoc for human readability, and span various programming languages, too low-level for automatic analysis. In contrast, publishing the set of declarative GQL queries along with their results would significantly enhance the clarity and reproducibility of a study's claims.

Provenance queries also enable scientists to reuse the data of previously published computing-intensive studies. Instead of running her expensive queries directly on the original input databases, the investigator would launch an automatic search for previously published studies whose provenance queries correspond to (parts of) the computation needed by her own queries. The results of the provenance queries can be directly imported and used as partial results of the new study's queries, skipping re-computation. This scenario corresponds in relational database practice to that of 'rewriting queries using views'.

7.6 Scaling — Probabilistic Inference

Learning the correlation between diseases and variations can be tackled differently if there are a large number of genomes. It may be less critical to accurately evaluate individual

variations for such a discovery problem, because erroneous variations are unlikely to occur over a large group of randomly chosen individuals. More generally, what other inference techniques are there that leverage the presence of data at scale? As an example, Google leverages the large data it has to find common misspellings. Note that accurately screening individual variations is still needed for personalized medicine.

7.7 Crowd-sourcing — Data Mining

Patterson suggests crowd-sourcing to solve hard challenges like cancer[14]. To enable this, the query system must have mechanisms to allow a group to work coherently on a problem. For example, imagine that talented high school science students look for genetic associations from cases and controls for a disease. A potentially useful GQL mechanism would be selecting a *random* subset of cases and controls, which are nevertheless genetically matched (arising from a single, mixing population). One could then query for a random subset of 100 individuals using a fraction of network bandwidth, while still providing similar statistical power for detecting associations.

7.8 Reducing Costs – Computer Systems

Personalized medicine must be commoditized to be successful; this requires computer systems research. For example, since most genomes are read only, could solid-state disks (SSDs) be leveraged? Second, efficient decomposition between the cloud and the workstation is key to reducing data traffic in and out of the cloud. Third, while genomics has been dominated historically by expensive parallel computers, it makes economic sense to adapt genomic software to harness the parallelism present in cheap multicore CPUs.

8. CONCLUSION

Genomics is moving from an era of scarcity (a few genomes with imperfect coverage) to abundance (universal sequencing with high coverage and cheap resequencing when needed). This shift requires rethinking genomic processing from ad hoc tools that support a few scientists to commodity software that can support a world of medicine. The history of computer systems teaches us that as systems move from scarcity to abundance, modularity is paramount; ad hoc software must be replaced by a set of layers with well defined interfaces. That these trends have been recognized by industry can be seen by the shift from machine specific formats such as Illumina to standards such as BAM, and from vendor specific variant formats to VCF. The 1000 genomes project has gained momentum and a large number of sequences are already accessible. However, much of the progress today has been to define data formats without powerful interface functionality; to use an Internet analogy this is as if TCP packet formats were defined without the socket interface.

Our paper proposes going beyond the layering implicit in current industry standards to enable personalized medicine and discovery. We advocate the separation of evidence from inference, and the separation of individual variation from variation across groups (Figure 3a)). More concretely, we propose a specific interface between the EL and the IL via a Genome Query Language (GQL). While GQL is based on a relational model using a virtual

interval relation, we have found that effort is required beyond standard relational optimization to allow GQL to scale to large genomes and large populations.

We described several benefits to separating evidence from inference. A genome repository accessible by GQL offers the ability to reuse genomic data across studies, the ability to logically assemble case-control cohorts, and the ability to rapidly change queries without ad-hoc programming. GQL also offers the ability to reduce the quality of inference on an individual basis when doing group inference on large populations. We described some simple ideas to scale GQL to populations using compressed strength indices and by doing EL processing in the cloud.

We emphasize that GQL and the EL are only our initial attempt at capturing abstractions for genomics. We hope our paper starts a larger conversation between computer scientists and biologists to tease out the right interfaces and layers for medical applications of genomics. Beyond abstractions, much work remains to be done to complete the vision including better large scale inference, system optimizations to increase efficiency, information retrieval to make medical records computer readable, and security mechanisms. While the work is vast, so is the opportunity. It is not often that computer scientists get a chance to work on a large system such as the Internet or UNIX that can change the world.

Acknowledgments

The authors acknowledge support from the NIH sponsored iDASH project (Grant U54 HL108460), NIH 5R01-HG004962, and a CSRO scholarship to CK. We thank Rajesh Gupta, Vish Krishnan, Ramesh Rao, and Larry Smarr for useful discussions and support.

REFERENCES

1. Alberts B, et al. *Molecular Biology of the Cell*. Garland Science. 2007
2. Annas GJ. HIPAA regulations - a new era of medical-record privacy? *N. Engl. J. Med.* Apr.2003 348:1486–1490. [PubMed: 12686707]
3. Benitez K, Malin B. Evaluating re-identification risks with respect to the HIPAA privacy rule. *J Am Med Inform Assoc.* 2010; 17:169–177. [PubMed: 20190059]
4. DePristo MA, et al. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat. Genet.* May.2011 43:491–498. [PubMed: 21478889]
5. Dolin RH, Alschuler L. Approaching semantic interoperability in Health Level Seven. *J Am Med Inform Assoc.* 2011; 18:99–103. [PubMed: 21106995]
6. David Haussler, DHaussler. *Nat. Biotechnol.* Mar.2011 29:243. [PubMed: 21390032]
7. Haussler D, et al. Genome 10K: a proposal to obtain whole-genome sequence for 10,000 vertebrate species. *J. Hered.* 2009; 100:659–674. [PubMed: 19892720]
8. Hudson KL, Holohan MK, Collins FS. Keeping pace with the times—the Genetic Information Nondiscrimination Act of 2008. *N. Engl. J. Med.* Jun.2008 358:2661–2663. [PubMed: 18565857]
9. Kho AN, et al. Electronic medical records for genetic research: results of the eMERGE consortium. *Sci Transl Med.* Apr.2011 3:79re1.
10. Kozanitis C, Saunders C, Kruglyak S, Bafna V, Varghese G. Compressing genomic sequence fragments using SlimGene. *J. Comput. Biol.* Mar.2011 18:401–413. [PubMed: 21385043]
11. Malin B, Benitez K, Masys D. Never too old for anonymity: a statistical standard for demographic data sharing via the HIPAA Privacy Rule. *J Am Med Inform Assoc.* 2011; 18:3–10. [PubMed: 21169618]
12. Nadkarni PM, Ohno-Machado L, Chapman WW. Natural language processing: an introduction. *J Am Med Inform Assoc.* 2011; 18:544–551. [PubMed: 21846786]

13. Pathak J, et al. Mapping clinical phenotype data elements to standardized metadata repositories and controlled terminologies: the eMERGE Network experience. *J Am Med Inform Assoc.* 2011; 18:376–386. [PubMed: 21597104]
14. Patterson D. Computer Scientists May Have What It Takes to Help Cure Cancer. *The New York Times*, December 5. 2011
15. Pennisi E. Will Computers Crash Genomics? *Science.* Feb; 2011 331(6018):999–1008. [PubMed: 21350137]
16. Schwarz UI, et al. Genetic determinants of response to warfarin during initial anticoagulation. *N. Engl. J. Med.* Mar.2008 358:999–1008. [PubMed: 18322281]
17. Stark P. Congressional intent for the HITECH Act. *Am J Manag Care.* Dec.2010 16:P24–28.
18. Stein LD. The Case for Cloud Computing in Genome Informatics. *Genome Biology.* May.2010 11(5):207. [PubMed: 20441614]
19. Tuzun E, et al. Fine-scale structural variation of the human genome. *Nat Genet.* Jul; 2005 37(7): 727–732. [PubMed: 15895083]

APPENDIX

Our database has some key relations. The first is R (*Reads*) that describe the set of read fragments output by the sequencer, and all of their properties. For example, for paired-end sequencing, R could contain identifiers for each read r , and its paired-end. A second relation encodes a set of intervals on the genome, where each is specified by the triple “(chromosome, start-position, end-position)”. A specific, and important, case of the interval relation is the relation G (the genome) where each entry is a point interval corresponding to a specific location (*locus*) on the genome. As G is large ($3 \cdot 10^9$ loci), we do not materialize G explicitly, using it instead as a virtual (conceptual) relation to facilitate the relational expression of queries. A final relation P describes the set of all individuals in a population, and will be discussed in a subsequent section.

Standard relational algebra operators and language constructs

Standard algebraic operators we use in the examples below include:

- The projection operator Π where $\Pi_X(E)$ returns, from the relation E evaluates to, the restriction of the tuples to the attributes named in X . In GQL, we express this operation using

SELECT X FROM E

- The selection operator σ , where $\sigma_\theta(E)$ selects the subset of precisely those tuples in the result of expression E that satisfy filter predicate θ . Correspondingly, in GQL:

SELECT * FROM E WHERE θ

- The Cartesian product operator $\times$, where $E \times F$ refers to all tuples of the form (e, f) where $e \in E, f \in F$. In GQL:

SELECT * FROM E, F

- The renaming operator $\rho_N(E)$, which sets to N the name of the relation returned by E .

- The group-by operator Γ , where in $G_1, G_2, \dots, G_k \Gamma_{N_1:F_1(A_1), \dots, N_l:F_l(A_l)}(E)$ E is an algebraic expression that evaluates to a relation, $G_1, \dots, G_k$ are group-by attributes, each F_i is an aggregate function and each N_i, A_i are attribute names. The meaning of the group-by operator is the standard one: all tuples in the result of E are partitioned into groups, such that two tuples are in the same group if and only if they agree on the values of all group-by attributes. Thus, the groups can be identified by the value of the group-by attributes. For each group $(g_1, \dots, g_k)$, the result contains a tuple of attributes $G_1, \dots, G_k, N_1, \dots, N_l$ and values $(g_1, \dots, g_k, n_1, \dots, n_l)$, respectively. Here, each n_i is computed by applying function F_i to the collection of attribute A_i 's values found in the group. In GQL:

$$\text{SELECT } G_1, \dots, G_k, N_1 = F_1(A_1), \dots, N_l = F_l(A_l) \text{ FROM } E \text{ GROUPBY } G_1, \dots, G_k$$

By default, this collection is viewed as a multiset, i.e. duplicate values are not removed. To apply F_i after eliminating duplicates, we use the syntax $N_i : F_i(\text{distinct } A_i)$. The syntax of the count aggregation is an exception, in that no A_i attribute need be specified.>

The Map join and the Project Interval operator

As reads are sequenced, they are also mapped to the reference genome using various mapping algorithms. Some mapping algorithms map each read to a unique location, or not at all; others may choose to identify all intervals where the underlying genomic string is similar (in terms of edit distance) to the read. For now, assume that each read is mapped to its best location by a mapping algorithm M , and the relation R includes the interval of the best map.

For an arbitrary relation of genomic intervals, A , define a *mapjoin* operation $R \bowtie A$ by a pair of tuples $(r, g) \in R \times A$ such that $(r.\text{chr}, r.\text{start}, r.\text{end})$ and $(a.\text{chr}, a.\text{start}, a.\text{end})$ ‘intersect’. Recall that we defined the relation G as the set of all genomic loci. Therefore, $R \bowtie G$, denotes the set of all tuples r, g where g denotes a genomic coordinate that r maps to. In many practical instances, the relation is quite large. If 33% of the genome was mapped on

the average by 10 reads, $R \bowtie G$ would contain $\simeq \frac{1}{3} \cdot 3 \cdot 10^9 \cdot 10 = 10^{10}$ tuples. In GQL:

$$\text{SELECT } * \text{ FROM MAPJOIN } R, G$$

To reduce output size, we define a special *Project-interval* operator $\prod^i \cdot \prod_{A.ID}^i (R \bowtie A)$ outputs all genomic intervals after ‘fusing’ adjacent and overlapping intervals. Thus,

$$\prod_{G.ID}^i (R \bowtie G)$$

would result in the set of all disjoint genomic intervals with some read mapping to them. Correspondingly, in GQL:

$$\text{SELECT MERGED } G.ID \text{ FROM MAPJOIN } R, G$$

Figure 1.

Universal Sequencing, discovery, and personalized medicine. Assume every individual is sequenced at birth. In discovery, we logically select a subset of individuals with a specific phenotype (e.g., disease) and another without the phenotype, and then identify genetic determinants for the phenotype. By contrast, in personalized medicine we retrieve the medical records of all patients genetically similar to a sick patient S.

Figure 2.

Evidence for variation in the donor. (a) The evidence for SNVs is provided by aligning donor reads against the reference sequence. The G/T variation might be a sequencing error as the variant reads maps with too many errors. However, the G/C variation appears to be a true SNV. (b) Paired-end sequencing and mapping provides evidence for deletion in the genome. The dotted rectangle demarcates the region in the reference deleted in exactly one of the two donor chromosomes. Read ‘a’ samples the region around the deletion (marked with the lightning bolt), and maps ‘discordantly’ in the reference; read ‘b’ maps concordantly, but with coverage about half of neighboring regions; read ‘c’ is sampled from the breakpoint and can map only at one end.

Figure 3.
Abstraction for genomics.

Figure 4.

Pictorial illustration of the MapJoin and ProjectInterval operations. (a) The MapJoin operation is a join based on intersection of intervals. In this figure, we project the reads that are in the Mapjoin. (b) The Project Interval is expressed by the ‘Select Merged’ command.

Figure 5.

A prototype implementation of GQL, with visualization using the open source tool jbrowse. Discordant paired-end reads supporting deletions can be seen.