The example shows in order:

1. A precise deletion with known breakpoint, a one base micro-homology, and a sample that is homozygous for the deletion.
2. An imprecise deletion of approximately 105 bp.
3. An imprecise deletion of an ALU element relative to the reference.
4. An imprecise insertion of an L1 element relative to the reference.
5. An imprecise duplication of approximately ~~21Kb~~21kb. The sample genotype is copy number 3 (one extra copy of the duplicated sequence).
6. An imprecise tandem duplication of 76bp. The sample genotype is copy number 5 (but the two haplotypes are not known).

5.4 Specifying complex rearrangements with breakends

An arbitrary rearrangement event can be summarized as a set of novel **adjacencies**. Each adjacency ties together 2 **breakends**. The two breakends at either end of a novel adjacency are called **mates**.

There is one line of VCF (i.e. one record) for each of the two breakends in a novel adjacency. A breakend record is identified with the tag "SVTYPE=BND" in the INFO field. The REF field of a breakend record indicates a base or sequence s of bases beginning at position POS, as in all VCF records. The ALT field of a breakend record indicates a replacement for s. This "breakend replacement" has three parts:

1. The string t that replaces places s. The string t may be an extended version of s if some novel bases are inserted during the formation of the novel adjacency.
2. The position p of the mate breakend, indicated by a string of the form "chr:pos". This is the location of the first mapped base in the piece being joined at this novel adjacency.
3. The direction that the joined sequence continues in, starting from p. This is indicated by the orientation of square brackets surrounding p.

These 3 elements are combined in 4 possible ways to create the ALT. In each of the 4 cases, the assertion is that s is replaced with t, and then some piece starting at position p is joined to t. The cases are:

REF	ALT	Meaning
s	t[p[	piece extending to the right of p is joined after t
s	t]p]	reverse comp piece extending left of p is joined after t
s	]p]t	piece extending to the left of p is joined before t
s	[p[t	reverse comp piece extending right of p is joined before t

The example in Figure 1 shows a 3-break operation involving 6 breakends. It exemplifies all possible orientations of breakends in adjacencies. Notice how the ALT field expresses the orientation of the breakends.

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO
2	321681	bnd_W	G	G]17:198982]	6	PASS	SVTYPE=BND
2	321682	bnd_V	T	]13:123456]T	6	PASS	SVTYPE=BND
13	123456	bnd_U	C	C[2:321682[	6	PASS	SVTYPE=BND
13	123457	bnd_X	A	[17:198983[A	6	PASS	SVTYPE=BND
17	198982	bnd_Y	A	A]2:321681]	6	PASS	SVTYPE=BND
17	198983	bnd_Z	C	[13:123457[C	6	PASS	SVTYPE=BND

5.4.1 Inserted Sequence

Sometimes, as shown in Figure 2, some bases are inserted between the two breakends, this information is also carried in the ALT column:

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO
2	321682	bnd_V	T	]13 : 123456]AGTNNNNNCAT	6	PASS	SVTYPE=BND;MATEID=bnd_U
13	123456	bnd_U	C	CAGTNNNNNCA[2 : 321682[	6	PASS	SVTYPE=BND;MATEID=bnd_V

6 BCF specification

VCF is very expressive, accommodates multiple samples, and is widely used in the community. Its biggest drawback is that it is big and slow. Files are text and therefore require a lot of space on disk. A normal batch of 100 exomes is a few ~~GB~~gigabytes, but large-scale VCFs with thousands of exome samples quickly become hundreds of ~~GBs~~gigabytes. Because the file is text, it is extremely slow to parse.

Overall, the idea behind is BCF2 is simple. BCF2 is a binary, compressed equivalent of VCF that can be indexed with tabix and can be efficiently decoded from disk or streams. For efficiency reasons BCF2 only supports a subset of VCF, in that all info and genotype fields must have their full types specified. That is, BCF2 requires that if e.g. an info field `AC` is present then it must contain an equivalent VCF header line noting that `AC` is an allele indexed array of type integer.

6.1 Overall file organization

A BCF2 file is composed of a mandatory header, followed by a series of BGZF compressed blocks of binary BCF2 records. The BGZF blocks allow BCF2 files to be indexed with tabix.

BGZF blocks are composed of a VCF header with a few additional records and a block of records. Following the last BGZF BCF2 record block is an empty BGZF block (a block containing zero type of data), indicating that the records are done.

A BCF2 header follows exactly the specification as VCF, with a few extensions/restrictions:

- All BCF2 files must have fully specified contigs definitions. No record may refer to a contig not present in the header itself.
- All INFO and GENOTYPE fields must be fully typed in the BCF2 header to enable type-specific encoding of the fields in records. An error should be thrown when converting a VCF to BCF2 when an unknown or not fully specified field is encountered in the records.

6.2 Header

The BCF2 header contains the following items:

Field	Type	Notes
magic	char[3]	The characters "`BCF`"
major_version	uint8_t	2
minor_version	uint8_t	1
l_text	uint32_t	Length of the "text" field, including the terminating NUL character
text	char[l_text]	VCF format header text, NUL-terminated

The "magic" field and version numbers can be used to quickly examine the file to determine that it's a BCF2.1 file. The "text" field contains the standard VCF header lines in text format, from `##fileformat=VCFv4.1` to `#CHROM ...` inclusive, terminated by a NUL character.

Because the type is encoded directly in the header, the recommended extension for BCF2 formatted files is *.bcf*. BCF2 supports encoding values in a dictionary of strings. The string map is provided by the keyword `##dictionary=S0,S1,...,SN` as a comma-separate ordered list of strings. See the "Dictionary of strings" section for more details.

6.2.1 Dictionary of strings

Throughout the BCF file most string values are be specified by integer reference to their dictionary values. For example, the following VCF record:

```
##INFO=<ID=ASP,Number=0,Type=Flag,Description="X">
##INFO=<ID=RSPOS,Number=1,Type=Integer,Description="Y">
##INFO=<ID=dbSNPBuildID,Number=1,Type=Integer,Description="Z">
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens">
#CHROM POS ID REF ALT QUAL FILTER INFO
20 10144 rs144773400 TA T . PASS ASP;RSPOS=10145,dbSNPBuildID=134
20 10228 rs143255646 TA T . PASS ASP;RSPOS=10229;dbSNPBuildID=134
```