INTERNATIONAL ORGANISATION FOR STANDARDISATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND AUDIO

ISO/IEC JTC1/SC29/WG11 MPEG2020/M53384
April 2020, Alpbach, AT

Source	ISO/IEC JTC1/SC29/WG11
Status	Input Document
Title	**Using Philips' Proposed Format for MPEG-G Part 6 to Accommodate GPress, GTRAC and GTC Compressors**
Authors	Shubham Chandak (Stanford University), Patrick Y.H. Cheung (Royal Philips)*, Qingxi Meng (University of Illinois at Urbana-Champaign), Mikel Hernaez (Center for Applied Medical Research at University of Navarra, UIUC), Idoia Ochoa (Tecnun at University of Navarra, UIUC)

* Corresponding Author

Introduction

Here we briefly describe how aspects of GPress [1] and GTRAC/GTC [2,3] can be integrated with the proposed format. The algorithms and details of integration are already discussed in some detail in other documents including the submission for the previous meeting [4]. We describe how the variables such as transform_algorithm, n_dependencies, dependency_attribute_id and compression_algorithm can be set to achieve this integration.

GPress

For GTF files, Gpress uses conditional compression of the start/end, strand and frame columns based on the feature column. This can be achieved by defining appropriate transform_algorithm. In particular we can use the following:

- Reorder_transform – reorder one attribute based on values of another attribute (can be used for frame columns)
- GTF_start_end_tranform – specialized algorithm from GPress for start_end attribute
- GTF_strand_transform - specialized algorithm from GPress for strand attribute

The value of the parameter n_dependencies can be set to 1 for these since the only dependency is the feature attribute, and the dependency_attribute_id[0] is set to 2 (corresponding to feature attribute).

For sparse RNA sequencing expression data, we can use transform_algorithm “sparse_transform” for converting the sparse matrix data into coordinate and value streams. In this case, n_dependencies variable is set to 0.

For bulk RNAseq expression data, GPress compresses the different columns separately which can be achieved by considering the columns as distinct attributes. Most attributes are compressed directly, while the first column (target ID) can be tokenized (setting compression_algorithm to TOKENIZATION which is similar to that used in MPEG-G part 2 for read names).

GPress uses BSC as the compression algorithm for the attributes (after the transform) which can be supported by the format by appropriately setting compression_algorithm to BSC.

Finally, the random access in GPress can be incorporated directly in the proposed format since both use block (chunk)-based random access strategies. We need to use supplementary indices for random access by genome position and by gene id. The linkage between the GTF and expression data can be achieved as discussed in [4].

More details can be found in the references.

GTRAC and GTC

GTRAC/GTC can be used as a compression_algorithm for the genotype (GT) attribute within each chunk. Note that GTRAC/GTC offers both improved compression and random access to a row or column of the genotype data. Since the file format already uses chunks for random access, we can combine this chunk-level random access with the finer random access from GTRAC/GTC within a chunk.

References

1. Meng, Qingxi, Idoia Ochoa, and Mikel Hernaez. "GPress: a framework for querying General Feature Format (GFF) files and feature expression files in a compressed form." bioRxiv (2019): 833087.
2. Tatwawadi, K., Hernaez, M., Ochoa, I., & Weissman, T. (2016). GTRAC: fast retrieval from compressed collections of genomic variants. Bioinformatics, 32(17), i479-i486.
3. Danek, Agnieszka, and Sebastian Deorowicz. "GTC: how to maintain huge genotype collections in a compressed form." Bioinformatics 34.11 (2018): 1834-1840.
4. M52159 Proposal of a Unified File Format for Genomic Annotations

Annex A Syntax and Semantics of Compressor

The syntax and semantics of Compressor are excerpts from subclause 6.4.1.3 of “Philips’ Response to CE1 (Phase 1) of MPEG-G Part 6” (M53381).

Table A1 – Compressor syntax

Syntax	**Type**
compressor {	
compressor_ID	st(v)
reserved	u(7)
transform	u(1)
if (transform) {	
transform_algorithm_ID	st(v)
n_dependencies	u(8)
}	
n_compression_algorithms	
for (i=0; i<n_compression_algorithms; i++) {	
compression_algorithm_ID[i]	st(v)
compression_algorithm_pars[i]	st(v)
}	
}	

dataset_type specifies the type of data in the dataset for which the encoding parameters are defined. The possible values are: 0 = non-aligned content; 1 = aligned content; 2 = reference; 3 = annotation.

n_compressors specifies the number of compressors, i.e. configurations of transform and compression algorithms, defined for the annotation dataset.

compressor_ID is the unique identifier of the compressor within the dataset, with the values 0 and 1 reserved respectively for no compression and the default compressor. It is used in Table Data Attribute Parameter Set as specified in subclause **Error! Reference source not found.** to associate the corresponding configuration of transform and compression algorithms with an attribute.

transform is a flag, and if set to 1, indicates that the compressor involves data transform before compression. Otherwise, no data transform is involved.

transform_algorithm_ID is the identifier of the transform algorithm being applied, optionally followed by a comma and then a URI that points to the codes of the transform algorithm. The URI shall be compliant with IETF RFC 3986 and IETF RFC 7320. If the ID is known and the codes are already installed, an MPEG-G compliant software can directly perform the transform/inverse-transform operation. If the ID is unknown and a URI is available, then the software should prompt the user to download and install the codes, and register the ID and a pointer to the executables for future use. If the ID is unknown and there is no URI, then the software should inform the user that the algorithm is not available.

n_dependencies specifies the number of dependency attributes for the transform.

n_compression_algorithms specifies the number of compression algorithms applied on an attribute in sequential order.

compression_algorithm_ID[i] is the identifier of the i-th compression algorithm being applied, optionally followed by a comma and then a URI that points to the codes of the compression algorithm. The URI shall be compliant with IETF RFC 3986 and IETF RFC 7320. If the ID is known and the codes are already installed, an MPEG-G compliant software can directly perform the transform/inverse-transform operation. If the ID is unknown and a URI is available, then the software should prompt the user to download and install the codes, and register the ID and a pointer to the executables for future use. If the ID is unknown and there is no URI, then the software should inform the user that the algorithm is not available.

compression_algorithm_pars[i] is a string of parameters in a predefined format required by the i-th compression algorithm.

Annex B Syntax and Semantics of Table Data Attribute Parameter Set

The syntax and semantics of Table Data Attribute Parameter Set for attribute definitions are excerpts from subclause 6.4.4.2 of “Philips’ Response to CE1 (Phase 1) of MPEG-G Part 6” (M53381).

Table B1 – Table Data Attribute Parameter Set Syntax

Syntax	**Key**	**Type**	**Remarks**
table_data_attribute_parameter_set {	*tdap*		
attribute_ID		u(16)	
attribute_name		st(v)	
attribute_metadata		gen_text	
attribute_type		u(8)	
attribute_default_value		st(v)	
attribute_missing_value		st(v)	
compressor_ID		u(8)	
if (transform) {			transform defined in compressor in Table A1
for (i=0; i<n_dependencies; i++) {			
reserved		u(5)	
dependency_table_data_ID[i]		u(3)	
dependency_attribute_ID[i]		u(16)	
}			
}			
compressor_common_data		compressor_common_data	
}			

attribute_ID is the identifier of the attribute unique within Table Data. It is the same as the index of the attribute in attribute_parameter_set of Table Data Attribute Information.

attribute_name is the name of the attribute.

attribute_metadata is the metadata of the attribute, which can include a description on the meaning and format of the attribute value and its belonging attribute group.

attribute_type specifies the data type of the attribute. The possible values and their respective data type definitions are listed in Table B2.

Table B2 – Attribute type definitions

attribute_type	Type	Number of bytes
0	Null terminated string	variable
1	Char	1
2	Boolean	1
3	Uint8	1
4	Int8	1
5	Uint16	2
6	Int16	2
7	Uint32	4
8	Int32	4
9	Uint64	8
10	Int64	8
11	Float	4
12	Double	8
13	Start_end (pair of uint32)	8

attribute_default_value is the default value of the attribute, mainly used for sparse encoding when most values equal to the default are excluded.

attribute_missing_value is the missing value of the attribute to be used in place of a null value in the output after decompression.

compressor_ID is the ID of one of the compressors defined in Dataset Parameter Set for compressing the data of the attribute.

(**dependency_table_data_ID[i]**, **dependency_attribute_ID[i]**) correspond to the table ID and and attribute ID of the i-th dependency attribute required by the transform algorithm (if transform == 1) within the compressor referenced by compressor_ID.

compressor_common_data stores the codebooks/statistical models used by the associated compressor to apply commonly on all chunks.