Transcript Segment Graph (TSG) Format Specification

Overview

The Transcript Segment Graph (TSG) format is designed for representing transcript assemblies and splicing relationships, adapting concepts from the Graphical Fragment Assembly (GFA) 2.0 specification. TSG allows for the representation of exons, splice junctions, isoforms, and structural variants in a graph-based format. The format supports multiple graphs within a single file.

Conceptual Model

In the TSG model:

1. **Graphs (G)** represent independent transcript graphs, each with its own set of nodes and edges.
2. **Chains (C)** are used to build the graph structure. They define the nodes and edges that make up the graph.
3. **Paths (P)** are traversals through the constructed graph.
4. The complete TSG is built by combining all nodes and edges from all chains within each graph.
5. After constructing the graph from chains, paths can be defined to represent ways of traversing the graph.

File Format

TSG is a tab-delimited text format with each line beginning with a single letter that defines the record type. Fields within each line are separated by tabs, and each line represents a distinct element of the transcript graph.

Multi-Graph Support

TSG supports multiple graphs within a single file using:

1. **Graph separators (G)** to define the start of each graph section
2. **Graph ID namespace** to qualify element IDs with their respective graph

All elements in a graph section belong to that graph until a new graph section is encountered.

Record Types

Global Header (H)

Contains metadata about the entire file. These appear at the beginning of the file before any graph sections.

```
H  <tag>  <value>
```

Fields:

- `tag`: Identifier for the header entry
- `value`: Header value

Graph Separator (G)

Indicates the start of a new graph section and provides graph metadata.

`G <graph_id> [<tag>:<type>:<value> ...]`

Fields:

- `graph_id`: Unique identifier for the graph
- Optional list of attribute tags in `tag:type:value` format

Example:

```
G gene_a name:Z:BRCA1 source:Z:RefSeq version:Z:GRCh38
```

Nodes (N)

Represent exons or transcript segments.

`N <id> <genomic_location> <reads> [<seq>]`

Fields:

- `id`: Unique identifier for the node (within the current graph section)
- `genomic_location`: Format `chromosome:strand:coordinates` where:
  - `chromosome`: Chromosome name (e.g., "chr1")
  - `strand`: "+" for forward strand, "-" for reverse strand
  - `coordinates`: Comma-separated list of exon coordinates in "start-end" format
- `reads`: Comma-separated list of reads supporting this node, in format `read_id:type`
  - Types might include SO (spanning), IN (internal), SI (significant), etc.
- `seq` (optional): Sequence of the node

Edges (E)

Represent connections between nodes, including splice junctions or structural variants.

`E <id> <source_id> <sink_id> <SV>`

Fields:

- `id`: Unique identifier for the edge (within the current graph section)
- `source_id`: ID of the source node
- `sink_id`: ID of the target node

- `SV`: Structural variant information in format "reference_name1,reference_name2,breakpoint1,breakpoint2,sv

Unordered Groups/Sets (U)

Represent unordered collections of graph elements.

```
U  <group_id>  <element_id_1> <element_id_2> ... <element_id_n>
```

Fields:

- `group_id`: Unique identifier for the unordered group (within the current graph section)
- `element_id_*`: Space-separated list of element identifiers (nodes, edges, or other groups)

Ordered Groups/Paths (P)

Represent ordered collections of graph elements where orientation matters.

```
P  <path_id>  <oriented_element_id_1> <oriented_element_id_2> ... <oriented_element_id_n>
```

Fields:

- `path_id`: Unique identifier for the ordered group (within the current graph section)
- `oriented_element_id_*`: Space-separated list of element identifiers with orientation (+ or -)

Chains (C)

Represent explicit paths through the graph with alternating nodes and edges.

```
C  <chain_id>  <node_id_1> <edge_id_1> <node_id_2> <edge_id_2> ... <node_id_n>
```

Fields:

- `chain_id`: Unique identifier for the chain (within the current graph section)
- Elements: Space-separated list of alternating node and edge identifiers
  - Must start and end with node identifiers
  - Must have an odd number of elements
  - Adjacent elements must be connected in the graph

Attributes (A)

Optional metadata attached to other elements.

```
A  <element_type>  <element_id>  <tag>:<type>:<value>
```

Fields:

- `element_type`: Type of element (N, E, U, P, or C)
- `element_id`: Identifier of the element to attach the attribute to
- `tag`: Name of the attribute

- `type`: Single letter code for the attribute data type
- `value`: Value of the attribute

Inter-Graph Links (L)

Represents connections between elements in different graphs.

```
L  <id>  <graph_id1>:<element_id1>  <graph_id2>:<element_id2>  <link_type>  [<tag>:<type>:<v
```

Fields:

- `id`: Unique identifier for the link
- `graph_id1:element_id1`: Graph-qualified identifier for the first element
- `graph_id2:element_id2`: Graph-qualified identifier for the second element
- `link_type`: Type of link (e.g., "fusion", "reference", "containment")
- Optional list of attribute tags in `tag:type:value` format

Semantics

Graph Sections

Each graph section in a TSG file defines an independent transcript segment graph:

1. **Section Boundaries**: A graph section begins with a G record and continues until the next G record or the end of the file.
2. **Element Scope**: All elements (N, E, C, P, U, A) defined within a graph section belong to that graph.
3. **Element Reference**: Elements can only reference other elements within the same graph section, except through inter-graph links (L).
4. **Element IDs**: Element IDs must be unique within their graph section but can be reused in different graph sections.

Node Semantics

Nodes in TSG represent exons or transcript segments. Each node has a genomic location that includes chromosome, strand, and coordinates. The genomic location specifies where the node is located in the reference genome. Nodes can be supported by different types of read evidence through the `reads` field, and can optionally include the sequence.

Edge Semantics

Edges in TSG represent connections between nodes, such as splice junctions or structural variants. The `SV` field provides details about the genomic context of the connection, including reference names, breakpoints, and the type of structural variant or splice.

Read Continuity

Read continuity is a critical concept in TSG that ensures valid traversals through the graph:

1. **Definition**: Read continuity requires that specific patterns of read support exist between connected nodes in a path, depending on node types.
2. **Node Types and Continuity Requirements**:
   - **SO (Source Node)**: Represents the start of a read. No read continuity required with previous nodes.
   - **SI (Sink Node)**: Represents the end of a read. No read continuity required with subsequent nodes.
   - **IN (Intermediary Node)**: Represents an internal segment of a read. Must share at least one read ID with both its previous and next nodes in the path.
3. **Validation**:
   - For IN nodes, implementations must verify that at least one common read ID exists between the current node and both its adjacent nodes.
   - SO and SI nodes provide more flexible continuity, allowing for extended paths without requiring end-to-end read support.
   - A path can be considered valid even if it doesn't have a single read spanning its entire length.
4. **Constraints**:
   - Each IN node in a valid path must maintain read continuity with its adjacent nodes.
   - The specific read type (SO, SI, IN) determines the continuity requirements at each position in the path.
5. **Representation**: The read IDs in each node's `reads` field implicitly define the continuity relationships in the graph, while the read types determine the continuity constraints.

Chains vs. Paths

Chains and paths serve fundamentally different purposes in the TSG format:

1. **Chains as Graph Construction Elements**:
   - Chains (C) are used to build the TSG graph itself.
   - Each chain contributes nodes and edges to the graph structure.
   - The complete graph is constructed by collecting all nodes and edges from all chains.
   - Chains represent the source evidence (e.g., transcript sequences) from which the graph was built.
2. **Paths as Graph Traversals**:

- Paths (P) are traversals through the already-constructed graph.
- Paths don't add any new structural elements to the graph.
- They represent ways of traveling through the existing nodes and edges.
- Paths can represent transcript isoforms, alternative splicing patterns, or other biological features.

This distinction is critical: chains define what the graph is, while paths define ways to traverse the graph.

Inter-Graph Links

Inter-graph links provide a way to represent relationships between elements in different graphs:

1. **Types of Links**:
   - **Fusion**: Represents a fusion between transcripts in different graphs
   - **Reference**: Indicates that one element references another
   - **Containment**: Shows that one element contains or is a superset of another
   - **Identity**: Indicates that elements across graphs are identical
2. **Usage Scenarios**:
   - Connecting fusion transcripts across genes
   - Linking alternative assemblies of the same region
   - Creating hierarchical relationships between graphs
   - Cross-referencing between different transcript annotations

Processing Model

The typical processing flow for a TSG file depends on whether the graph structure (nodes and edges) already exists:

Case 1: Nodes and Edges Are Explicitly Defined

If the TSG file contains explicit node (N) and edge (E) records:

1. Process the global headers at the beginning of the file
2. For each graph section (started by a G record):
   - Create a new graph with the given ID and attributes
   - Read and create the graph structure directly from the records in the section
   - Process chains (C) as additional evidence or source information
   - Process paths (P) as traversals through the explicitly defined graph
3. Process inter-graph links (L) to establish connections between graphs
4. Validate read continuity by checking for shared read IDs across adjacent nodes in paths

Case 2: Nodes and Edges Are Not Explicitly Defined

If the TSG file does not contain explicit node and edge records, or contains only partial definitions:

1. Process the global headers at the beginning of the file
2. For each graph section (started by a G record):
   - Create a new graph with the given ID and attributes
   - Extract and construct all nodes and edges from chains (C) in the section
   - Build the complete graph structure from these extracted elements
   - Process paths (P) as traversals through the graph constructed from chains
3. Process inter-graph links (L) to establish connections between graphs
4. Verify read continuity for all paths by ensuring adjacent nodes share common read support

Type Definitions for Attributes

- `i`: Integer
- `f`: Float
- `Z`: String
- `J`: JSON
- `H`: Hex string
- `B`: Byte array

Example

```
# Global headers
H  TSG  1.0
H  reference  GRCh38

# First graph
G  gene_a  name:Z:BRCA1  locus:Z:chr17q21.31

# Nodes for gene_a
N  n1  chr17:+:41196312-41196402  read1:SO,read2:SO  ACGTACGT
N  n2  chr17:+:41199660-41199720  read2:IN,read3:IN  TGCATGCA
N  n3  chr17:+:41203080-41203134  read1:SI,read2:SI  CTGACTGA

# Edges for gene_a
E  e1  n1  n2  chr17,chr17,41196402,41199660,splice
E  e2  n2  n3  chr17,chr17,41199720,41203080,splice

# Chains for gene_a
C  chain1  n1  e1  n2  e2  n3
```

```
# Paths for gene_a
P  transcript1  n1+  e1+  n2+  e2+  n3+

# Sets for gene_a
U  exon_set  n1  n2  n3

# Attributes for gene_a elements
A  N  n1  expression:f:10.5
A  N  n1  ptc:i:10
A  P  transcript1  tpm:f:8.2

# Second graph
G  gene_b  name:Z:BRCA2  locus:Z:chr13q13.1

# Nodes for gene_b
N  n1  chr13:+:32315480-32315652  read4:SO,read5:SO  GATTACA
N  n2  chr13:+:32316528-32316800  read4:IN,read5:IN  TACGATCG
N  n3  chr13:+:32319077-32319325  read4:SI,read5:SI  CGTACGTA

# Edges for gene_b
E  e1  n1  n2  chr13,chr13,32315652,32316528,splice
E  e2  n2  n3  chr13,chr13,32316800,32319077,splice

# Chains for gene_b
C  chain1  n1  e1  n2  e2  n3

# Paths for gene_b
P  transcript1  n1+  e1+  n2+  e2+  n3+

# Sets for gene_b
U  exon_set  n1  n2  n3

# Attributes for gene_b elements
A  P  transcript1  tpm:f:3.7

# Inter-graph links (appears after all graph sections)
L  fusion1  gene_a:n3  gene_b:n1  fusion  type:Z:chromosomal
```

In this example:

1. The file begins with global headers that apply to the entire file
2. Two graph sections are defined (gene_a and gene_b), each started by a G record
3. Each graph has its own nodes, edges, chains, paths, and attributes
4. Element IDs (n1, e1, etc.) are local to each graph section
5. An inter-graph link represents a fusion between elements from different graphs

6. To reference elements across graphs (in the L record), graph-qualified IDs are used (graph_id:element_id)

Implementation Considerations

Graph Section Handling

Implementations should:

- Initialize a new graph context whenever a G record is encountered
- Associate all subsequent elements with the current graph until the next G record
- Maintain separate data structures for each graph
- Enforce that connections (edges, chains, paths) only exist between elements in the same graph
- Process inter-graph links separately from within-graph connections

Element ID Resolution

When processing elements:

- Within a graph section, element IDs are resolved in the context of that graph
- In inter-graph links, element IDs must be qualified with their graph ID (graph_id:element_id)
- Implementations should maintain a mapping of (graph_id, element_id) pairs to resolve references

Genomic Location Parsing

Implementations should carefully parse the genomic location field, which contains:

- Chromosome (e.g., "chr1")
- Strand ("+" or "-")
- Coordinates (comma-separated list of "start-end" pairs)

These components are separated by colons.

Read Evidence

Read evidence is recorded with both read identifiers and types. Implementations should:

- Parse the read identifier and read type, separated by a colon
- Support different read types (SO, IN, SI, etc.) as used in the implementation

Validation Requirements

Implementations should validate that:

- Each graph section has a unique graph ID
- Element IDs are unique within their graph section
- Chains have an odd number of elements (starting and ending with nodes)
- Adjacent elements in chains are correctly connected in the graph
- Group identifiers are unique across U, P, and C types within each graph section
- Paths (P-lines) only reference elements that exist in the same graph section
- Inter-graph links only reference elements that exist in their respective graphs

Chain Processing

- Process chains within each graph section independently
- When encountering a chain, extract all nodes and edges and add them to the current graph
- The same node or edge can appear in multiple chains within the same graph
- The structural integrity of each graph is defined by its chains

Path Processing

- Paths do not add new elements to the graph
- Paths must reference existing nodes and edges within the same graph section
- Paths can include orientation information (+ or -) for elements

Read Continuity Verification

When processing TSG files, implementations should:

- Extract read IDs from each node's `reads` field along with their types (SO, SI, IN)
- For each path or chain traversal:
  - For IN nodes: Ensure at least one read ID is shared with both the previous and next nodes
  - For SO nodes: No continuity check required with previous nodes
  - For SI nodes: No continuity check required with subsequent nodes
- Flag paths where IN nodes lack proper read continuity as potentially unsupported by the data
- Recognize that valid paths may be constructed even without end-to-end read support, as long as the IN node continuity requirements are satisfied
- Provide options to filter or annotate paths based on different levels of read continuity stringency

Biological Interpretation in Transcript Analysis

In the context of transcript analysis, the TSG format elements typically represent:

Graphs (G)

- **Genes**: Independent genetic loci
- **Transcription Units**: Coordinated transcriptional regions
- **Alternative Assemblies**: Different representations of the same region

Nodes (N)

- **Exons**: Genomic regions that are transcribed and remain in the mature RNA
- Can include multiple segments (e.g., for complex exon structures)
- Read support indicates which sequencing reads support this exon, with different types of support

Edges (E)

- **Splice Junctions**: Connections between exons
- **Structural Variants**: Genomic rearrangements like fusions, deletions, or insertions
- The SV field provides details on the exact genomic coordinates

Chains (C)

- **Original Transcripts**: Complete transcript sequences observed in the data
- **Source Evidence**: The actual RNA molecules detected
- **Assembled Transcripts**: Transcripts assembled from read data
- These build the structure of the transcript graph

Paths (P)

- **Transcript Isoforms**: Alternative splicing variants
- **Expression Patterns**: Different ways genes are expressed
- **Predicted Transcripts**: Computationally predicted transcript models
- These are ways to traverse the established transcript graph

Inter-Graph Links (L)

- **Fusion Transcripts**: Transcripts spanning multiple genes
- **Reference Relationships**: Cross-references between different transcript annotations
- **Containment Relationships**: Hierarchical organization of transcripts

This separation aligns with how many transcript assembly algorithms work:

1. First, chains of exons and splice junctions are identified from the data
2. Then, potential transcripts are derived by traversing the graph in different ways
3. Finally, relationships between different transcript graphs are established