INTERLOCUTOR: COMMODUS

InterLocutor:Commodus takes a set of input genomic loci and systematically compares it to selected functional annotation tracks. It can be used to pick out significantly enriched annotation terms. In its annotation term mining capacity, interLocutor will scour all string encoded items associated with intersecting annotation loci (such as GO-terms, KEGG pathways, binding/interaction sites, motifs/domains, etc). and find frequently intersecting terms.

INPUT

1. **QuerySet:** The query locus set that will be mined for intersecting annotation terms.
2. **ControlSets:** A set of locus sets each with the same number of loci as the QuerySet and matched for any other criteria the user might desire (length, G/C content, type of locus, etc). These will be used to generate control statistics for each mined annotation term.
3. **MinSupport:** A fraction of the query locus set or a fixed number of loci. This will be the minimum acceptable support for a mined annotation term. It is used to filter the unary annotation terms and speed up generation of significant combinations.
4. **SpAssDBConnection:** An instance of the HocusLocus DB manager that has been set to the appropriate species and assembly.
5. **TargetAnnotationTracks:** An user-selected list of tracks from the appropriate species and assembly that contain functional annotation loci of interest. These may include regulatory functional elements such as transcription factor, RNA binding-protein, and miRNA binding sites, loci associated with Gene Ontology or pathway terms, disease associations, domain information, or any other locus-conformable annotation database.

SUPPORTING STRUCTURES

1. **AnnoTerm:** A class that holds bitsets (each the same length as the query locus set) for corresponding to the query set and each of the control sets.
2. **locutus:** A HashMap having annotation terms as keys and AnnoTerm data structures as values.

ALGORITHM

The terms already associated with the QuerySet will be the first ones mined. This allows us to aggregate chromosome and strand information for our set along with the available ids, types, sources, and other string annotations. Processing each QuerySet locus in turn, the above named annotations are parsed out and added to the locutus object. The bit in the current term's locutus bitset corresponding to the current locus is flagged true. The process is continued with the remainder of the query locus set. Now, a filtering step is run to remove all terms that don't meet the minSupport criteria. The control sets are then parsed and any QuerySet annotation already in the locutus object are updated with presence flags from the control set.

Next, the SpAssDBConnection is used to retrieve all loci from the first given TargetAnnotationTrack. A LIA instance is initialized and both the QuerySet and the newly retrieved TargetAnnotationTrack LocusSet are added. The intersection is performed and the resulting LocusNexus (a matrix where each row contains two intersecting loci, one from the QuerySet and one from the Target LocusSet) is retrieved. Each row of the LocusNexus is parsed through and annotations from the Target locus (id, source, type, other string annotations besides chromosome and strand) are added to the locutus object. For each of these annotations, the bit corresponding to the QuerySet locus is flagged in the term's bitset. This is repeated for each row in the LocusNexus. The locutus object is again filtered remove all terms not meeting the minSupport criteria. The ControlSets are then each intersected with the current TargetAnnotationTrack and the appropriate control bitsets in all annotation terms indexed from the QuerySet updated with the intersecting control loci. This whole process is repeated with each of the TargetAnnotationTracks.

Finally, a recursive process to find all combinations of found annotation terms that meet the minSupport condition is initiated. A depth-first traversal is conducted over the tree of possible combinations of terms and the recursion is terminated only at nodes that don't meet the minSupport criterion. When any two terms are combined, the corresponding bitsets are intersected and the minSupport for the new compound term is determined. Those compound terms meeting the minSupport condition are allowed to continue with the recursion. As new and exisiting nodes are encountered by the recursion, summaries of the AnnoTerms are created including the support for the term in the QuerySet and each of the ControlSets as well as ratios of QuerySupport/AvgControlSupport and the derived Fisher-Exact Test significance statistic. This continues until all supported compound nodes are created and summaries for all unary and compound nodes generated. The algorithm terminates here.

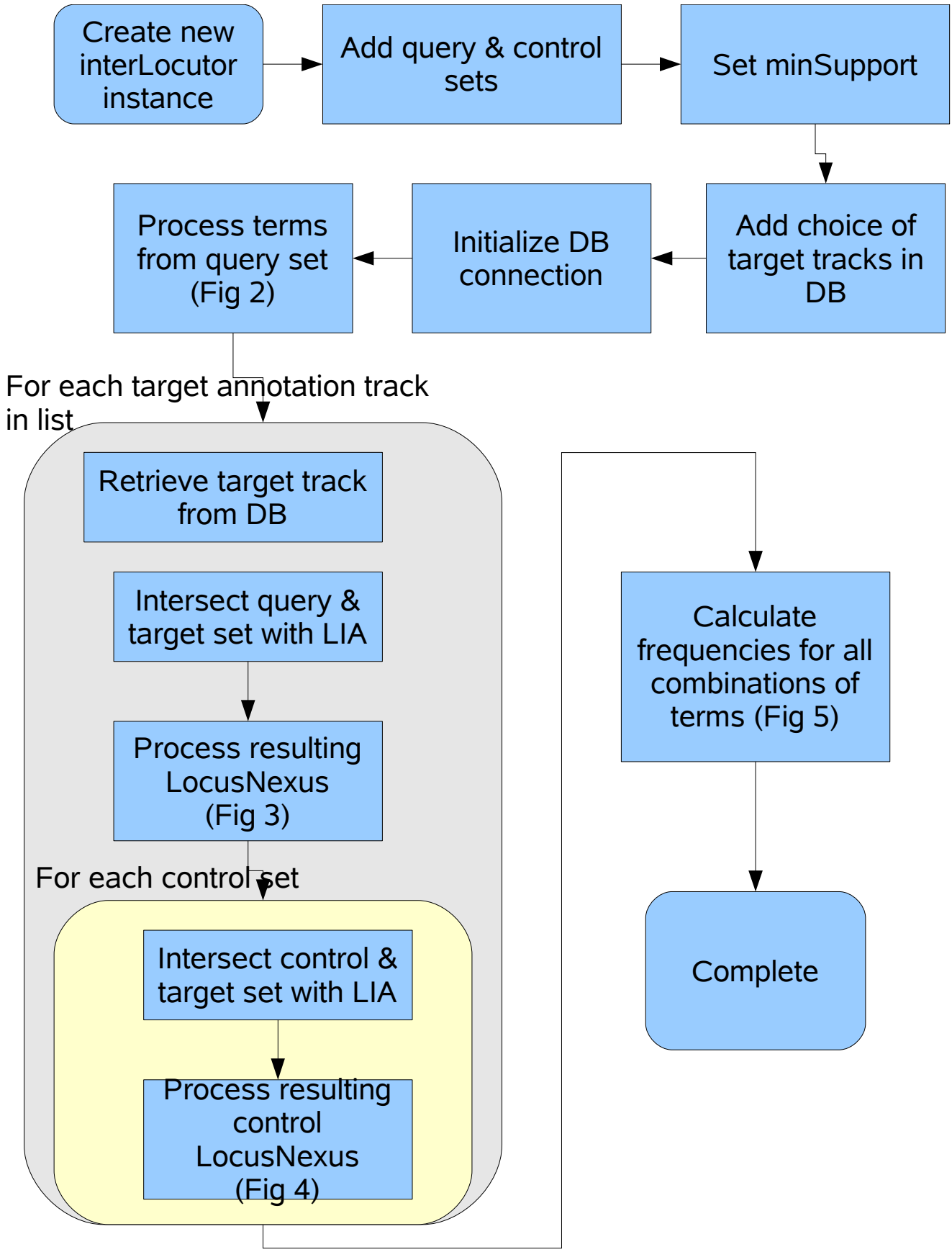


FIGURE 1: INTERLOCUTOR OVERALL FLOW

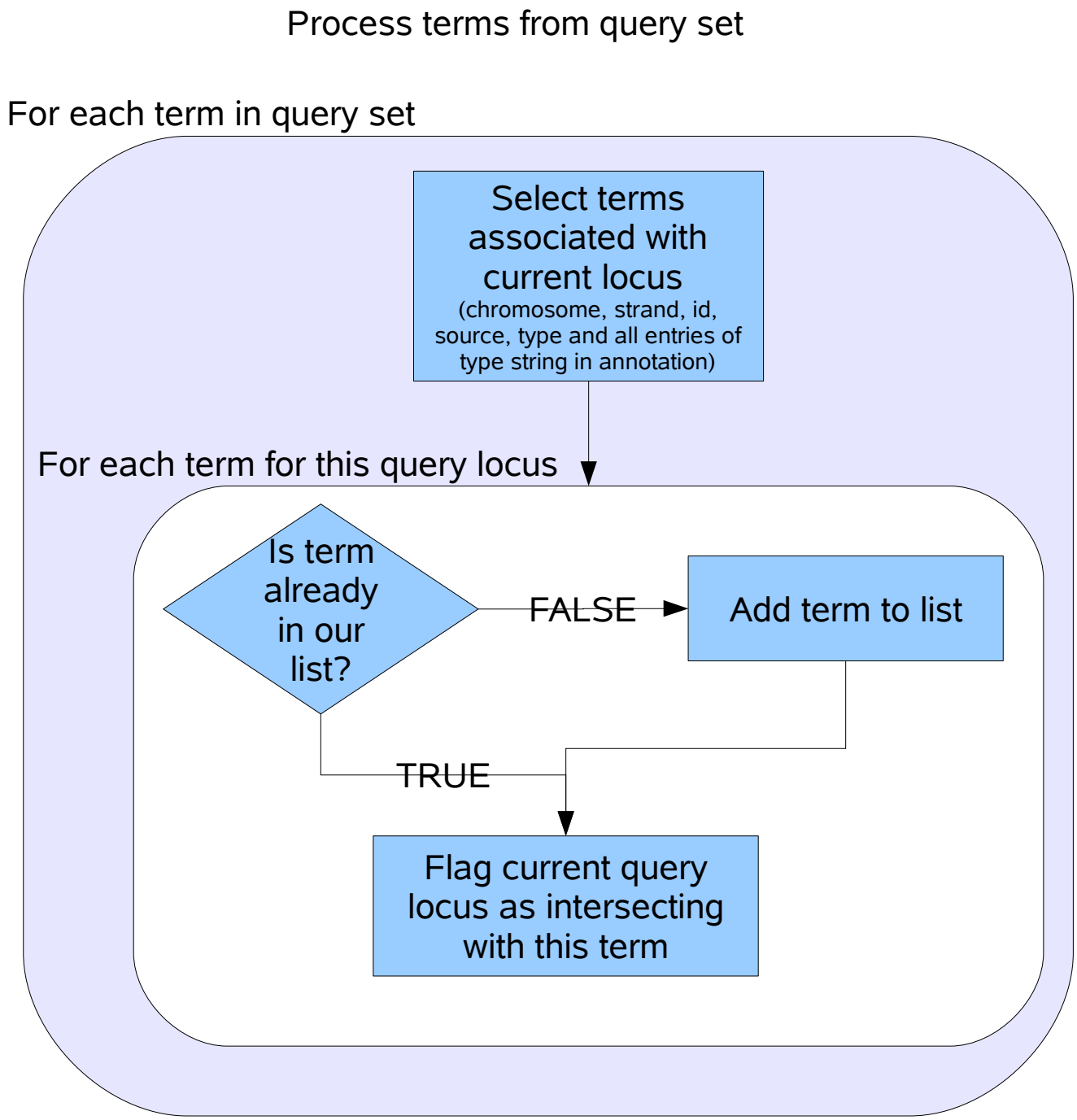


FIGURE 2: PROCESS QUERYSET LOCUS ANNOTATIONS

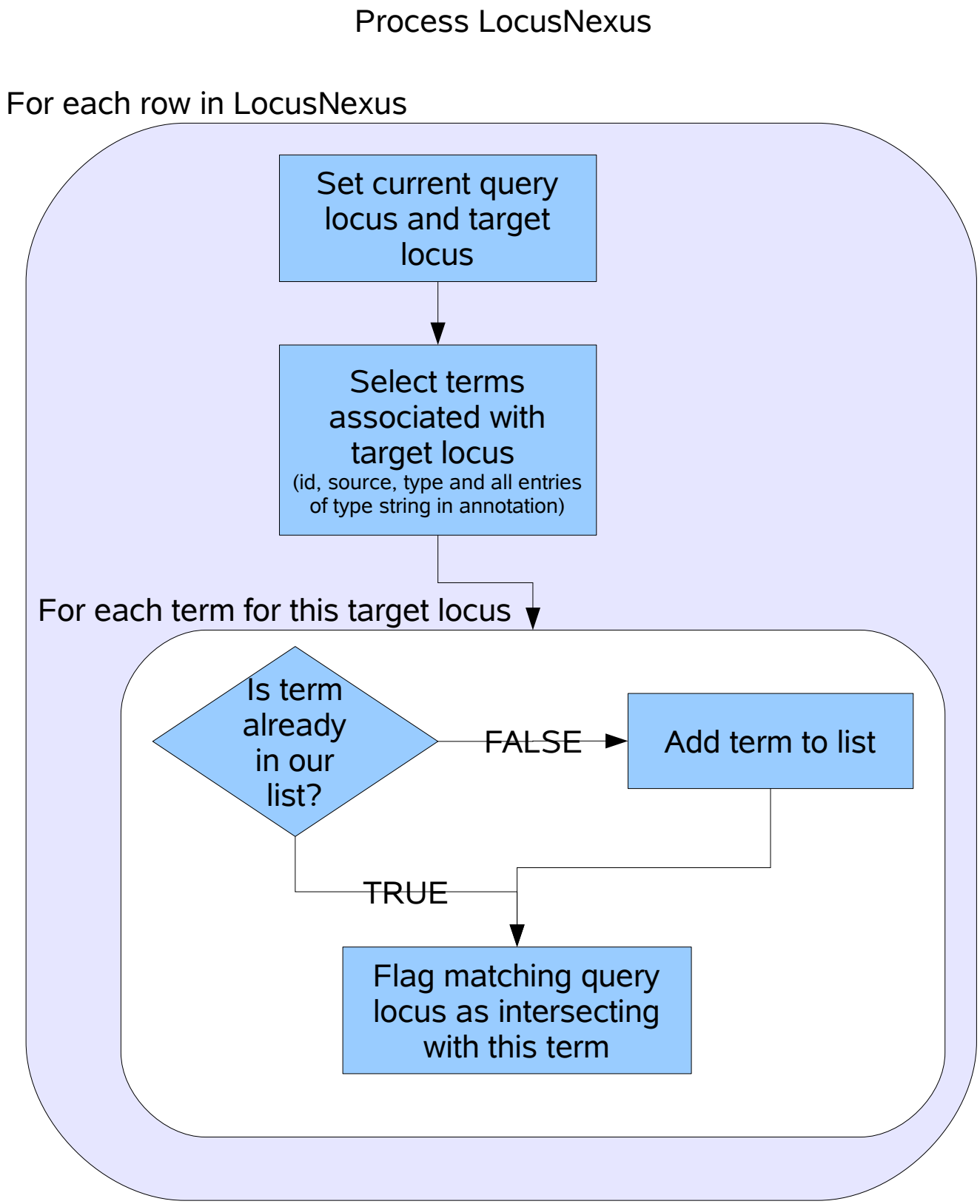


FIGURE 3: PROCESS TARGETTRACK ANNOTATIONS USING LOCUSNEXUS

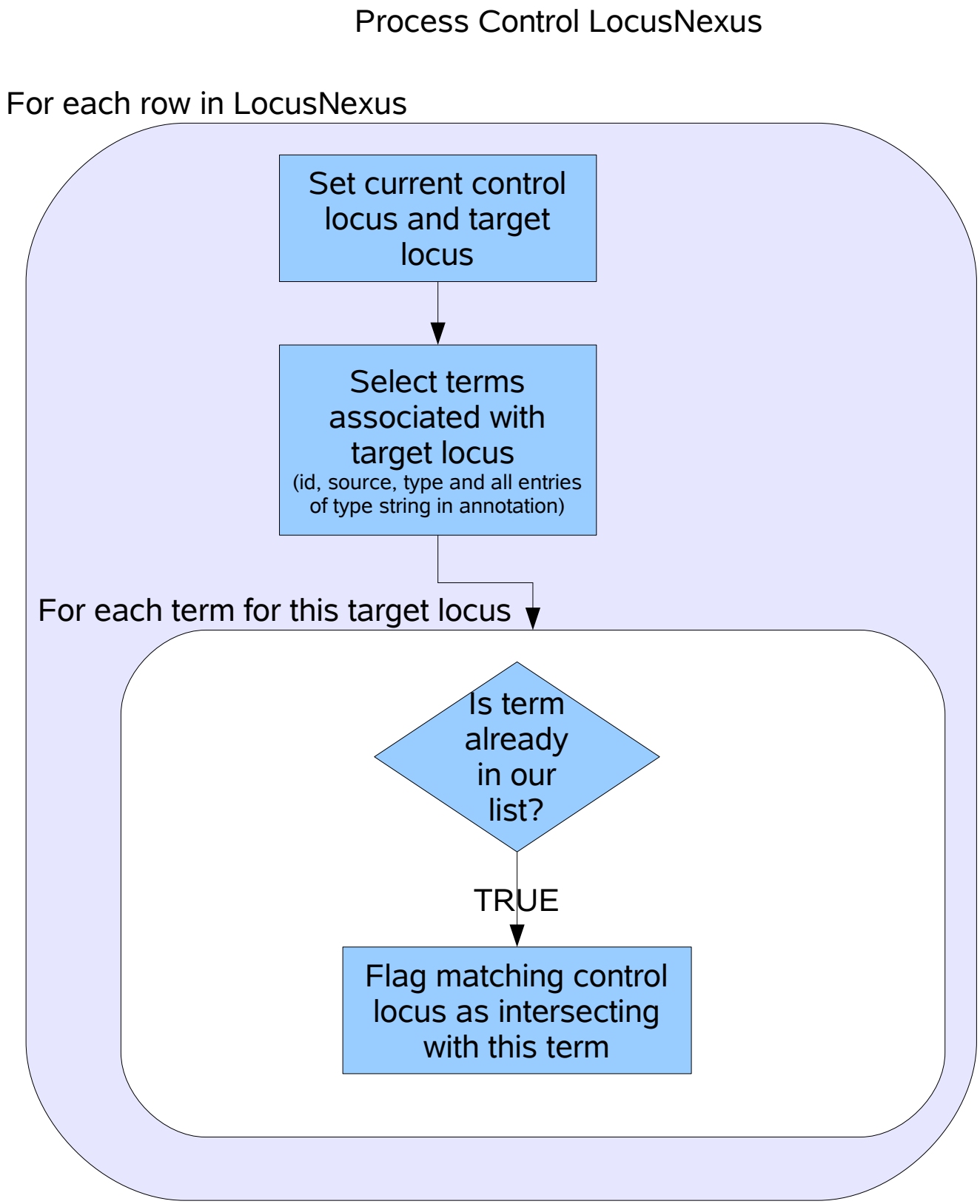


FIGURE 4: PROCESS CONTROLSET INTERSECTIONS WITH TARGETTRACK

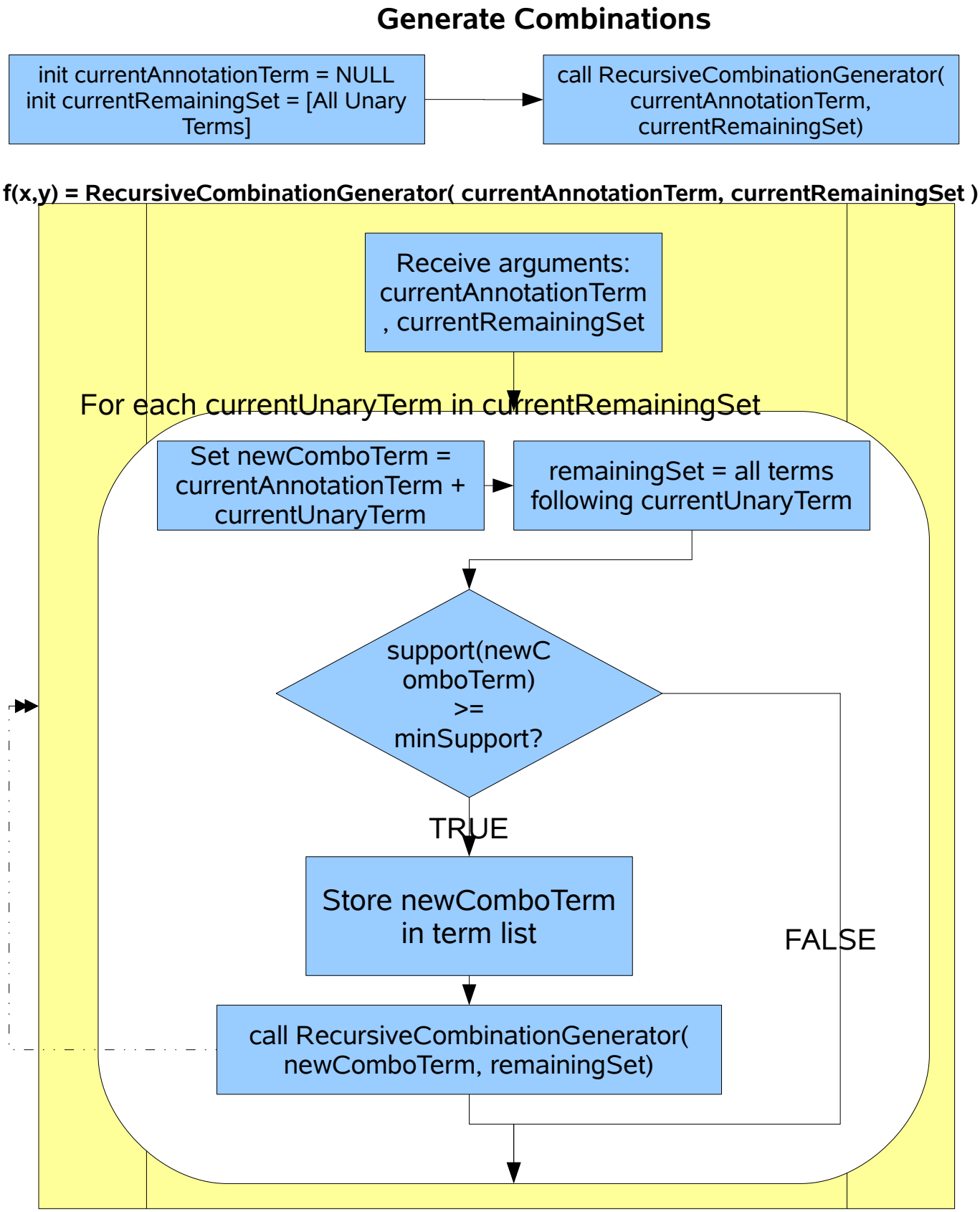


FIGURE 5: GENERATION OF ALL POSSIBLE COMBINATIONS OF UNARY TERMS USING A RECURSIVE ALGORITHM

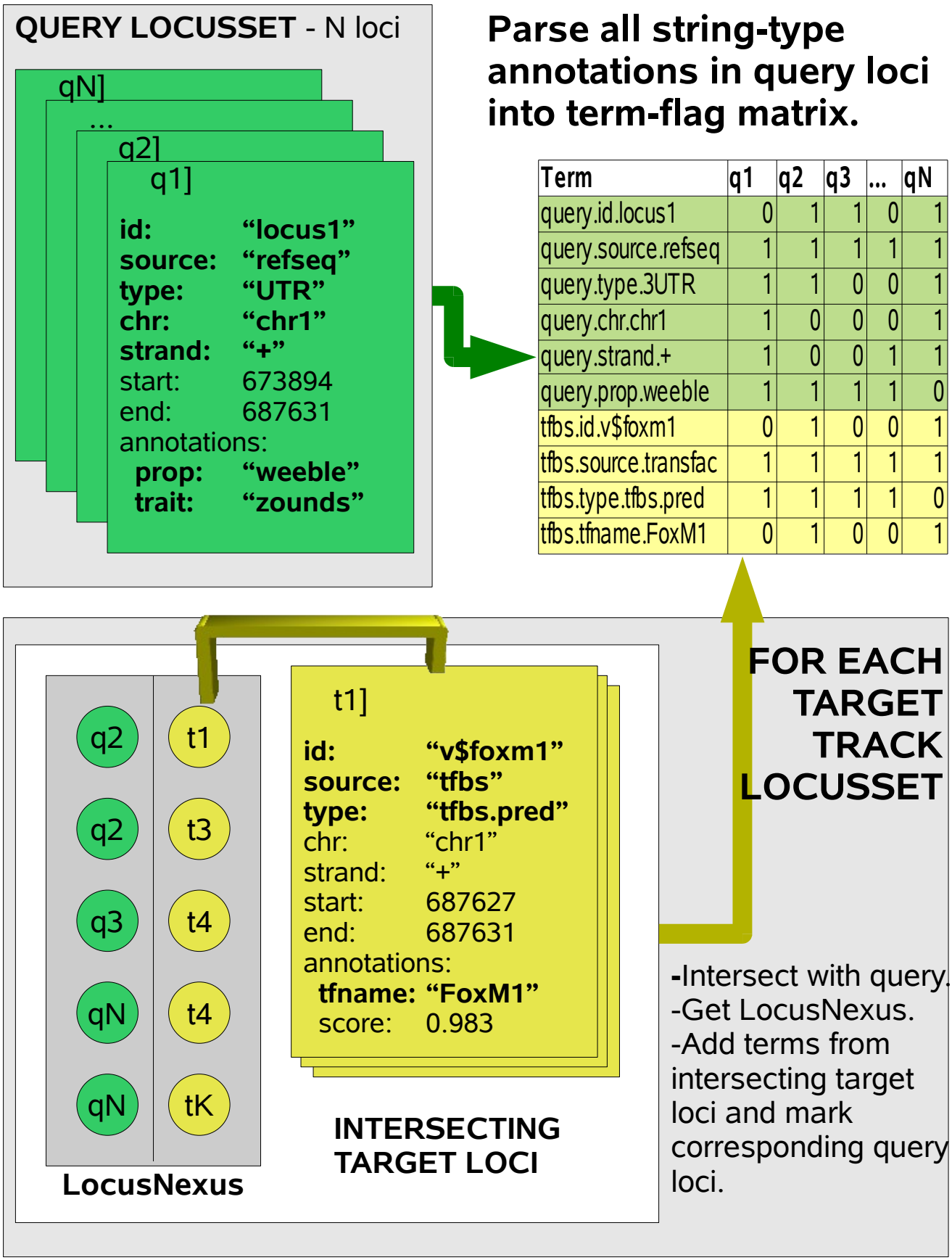
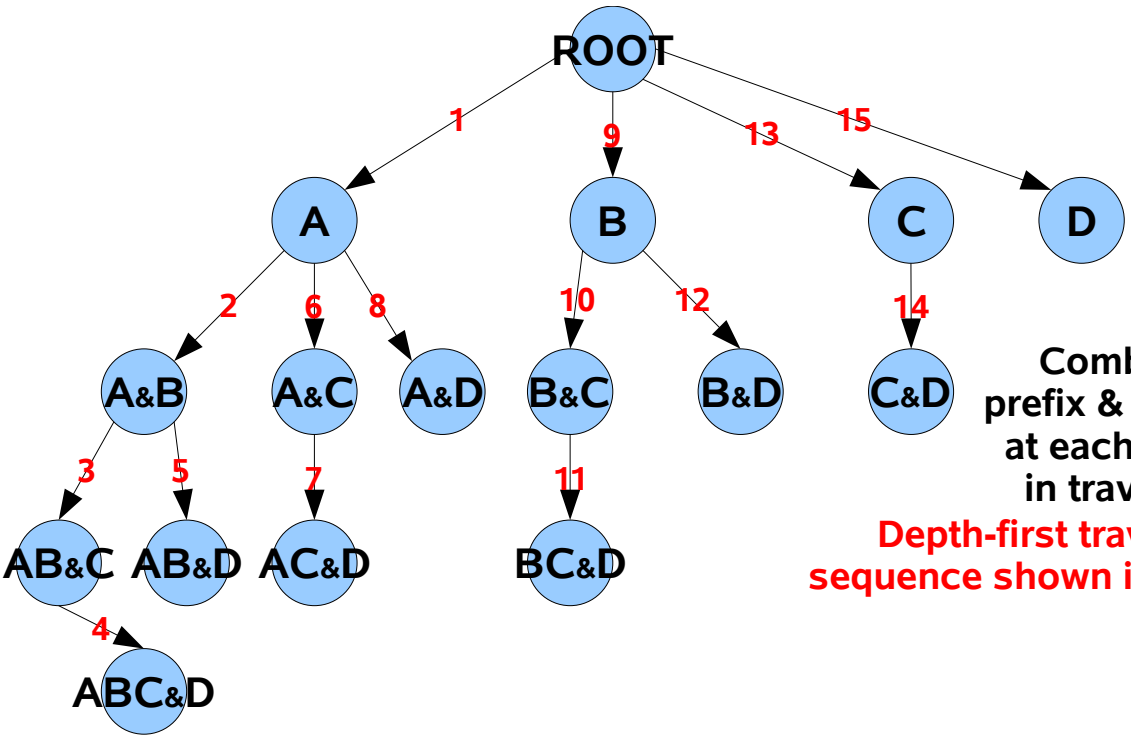


FIGURE 6: GRAPHICAL VIEW OF ANNOTATION MINING IN QUERYSET AND TARGETTRACKS



RECURSIVE DEPTH FIRST TRAVERSAL OF COMBINATIONS

The tree of possible combinations when traversed recursively is an economical way to generate all possible combinations of n terms. We use a depth-first traversal to minimize memory utilization for large lists of terms and to efficiently prune branches that do not meet the minSupport.

At each node, the prefix bitset is intersected with the suffix bitset. If the result meets the minSupport criteria, the new compound term is added to the output and it's child branches are processed.

	bit for each query locus				
Term	**q1**	**q2**	**q3**	**...**	**qN**
query.term1	0	1	1	0	1
...	1	0	1	0	1
query.termN	1	1	0	0	1
track1.term1	1	0	0	0	1
...	0	0	0	1	0
trackN.termN	1	1	0	0	0
query.term1_&_query.term2	0	1	0	0	0
...	0	0	0	0	0
track1.term2_&_trackN.term3_&_trackN.termN	1	1	0	0	0

one-term rows

compound term rows

FIGURE 7: RECURSIVE ALGORTHIM USED TO GENERATE COMPOUND ANNOTATION TERMS(ABOVE) WITH EXAMPLE OUTPUT TABLE FROM THE COMMODUS MODULE (BELOW)

INTERLOCUTOR: CONNEXUS

Interloutor:Connexus is used to cast a set of query loci into the context of functional interactions including protein-protein binding and transcriptional or post-transcriptional regulatory binding. It can however also be used to build a network of any sort of mappable functional association intersecting the query set (for example clusters of loci associated with particular pathways). This is accomplished through the use of annotation tracks containing binding sites or other functional linkages between loci.

INPUT

1. **QuerySet:** The query locus set that will be mined for intersecting annotation terms.
2. **IdentifyingAnnotation:** A string indicating the annotation used to merge loci into a single node – for example all the binding sites for a given protein would be merged. The annotations for each locus will be searched for a key matching IdentifyingAnnotation and the value there will be used as the unique identifier in the Graph. If IdentifyingAnnotation is empty, null or set to "id", or if the IdentifyingAnnotation is not found as a key in a locus's annotation set, the Locus ID will instead be used.
3. **SpAssDBConnection:** An instance of the HocusLocus DB manager that has been set to the appropriate species and assembly.
4. **TargetAnnotationTracks:** An user-selected list of tracks from the appropriate species and assembly that contain functional linkage loci of interest. These may include regulatory functional elements such as transcription factor, RNA binding-protein, and miRNA binding sites, loci associated with Gene Ontology or pathway terms, disease associations, domain information, or any other locus-conformable linkage database.

SUPPORTING STRUCTURES

1. **Graph:** Composed of nodes (each corresponding to a LocusSet) and edges between the nodes denoting functional linkages.
2. **NodeMap:** A HashMap mapping a given unique id to its node in the Graph.

ALGORITHM

The Graph is initialized. The SpAssDBConnection is used to retrieve all loci from the first given TargetAnnotationTrack. A LIA instance is initialized and both the QuerySet and the newly retrieved TargetAnnotationTrack LocusSet are added. The intersection is performed and the resulting LocusNexus (a matrix where each row contains two intersecting loci, one from the QuerySet and one from the Target LocusSet) is retrieved. Each row of the LocusNexus is processed and the query and target ids are parsed using the IdentifyingAnnotation. If either query or target id is not present in the NodeMap, new nodes (each containing a new LocusSet containing the new locus) are created in the graph to represent them. If either id is already present in the NodeMap, the corresponding node is retrieved and if the current locus is not present in the contained LocusSet, it is added. A directed edge is added from the target id node to the query id node. This is repeated for each row in the LocusNexus. This whole process is repeated with each of the TargetAnnotationTracks.

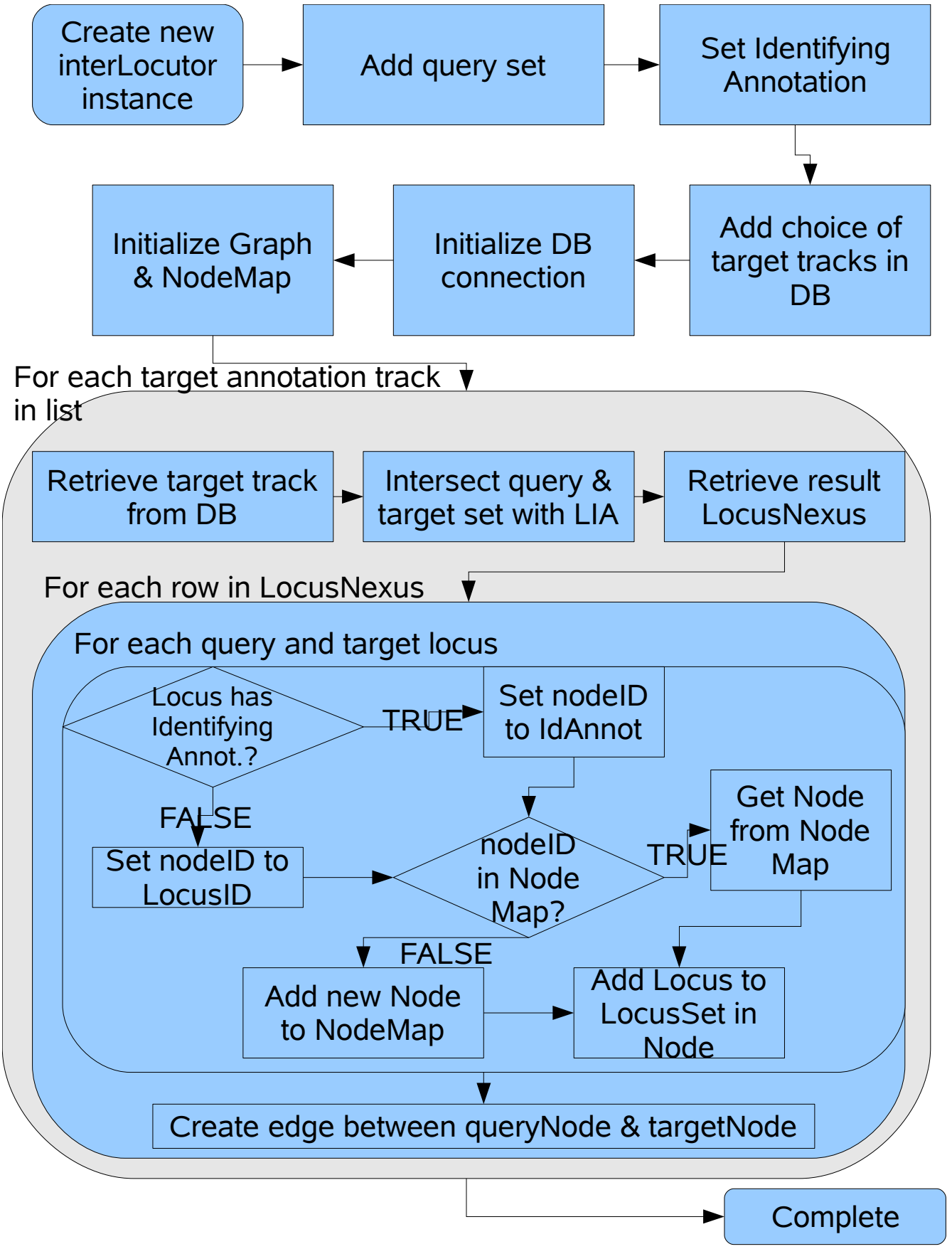


FIGURE 8: FLOW-DIAGRAM FOR GRAPH GENERATION FROM INTERSECTIONS

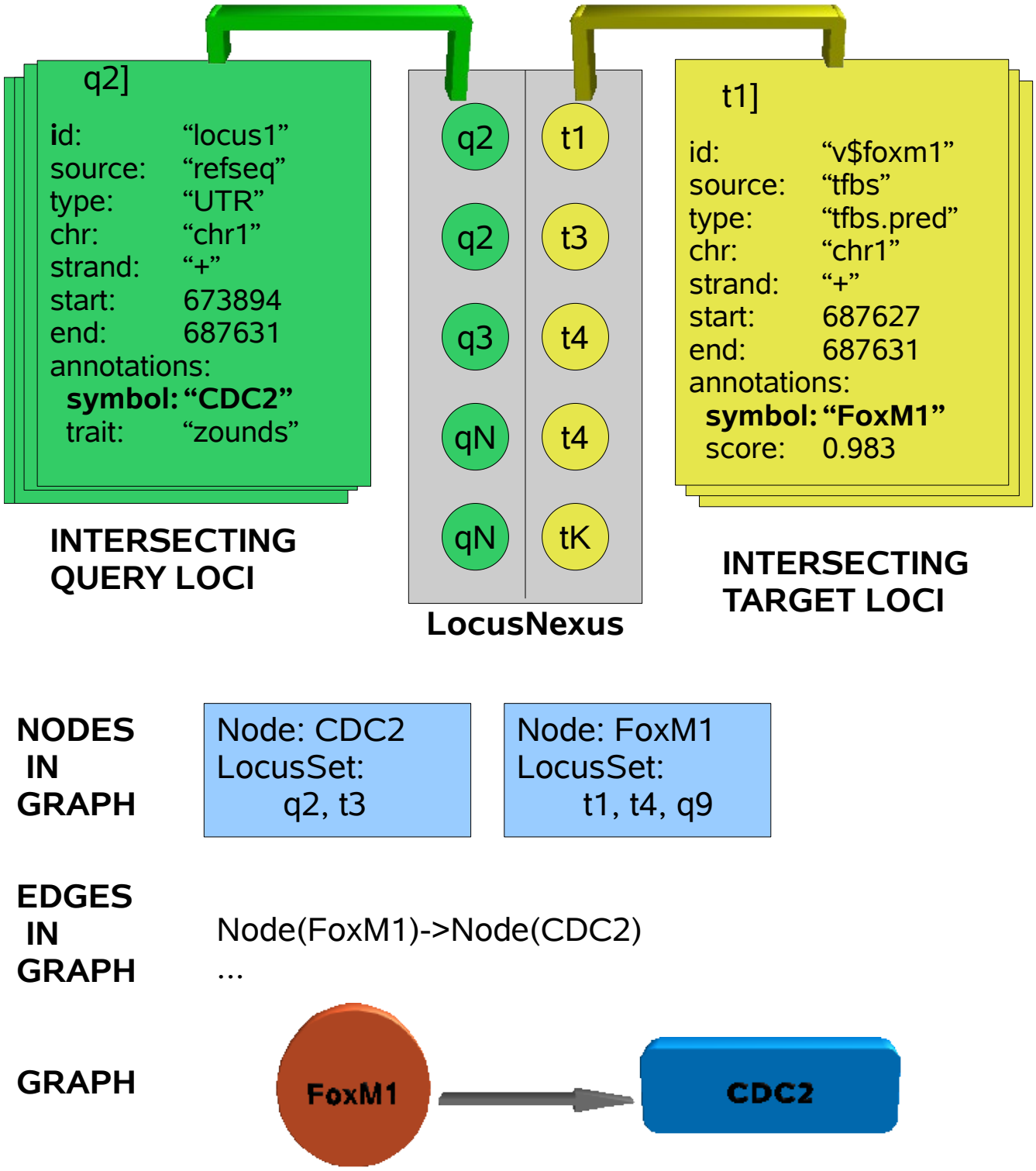


FIGURE 9: CONVERTING A LOCUSNEXUS ROW INTO NODES AND EDGES IN A GRAPH USING IDENTIFYINGANNOTATIONS.

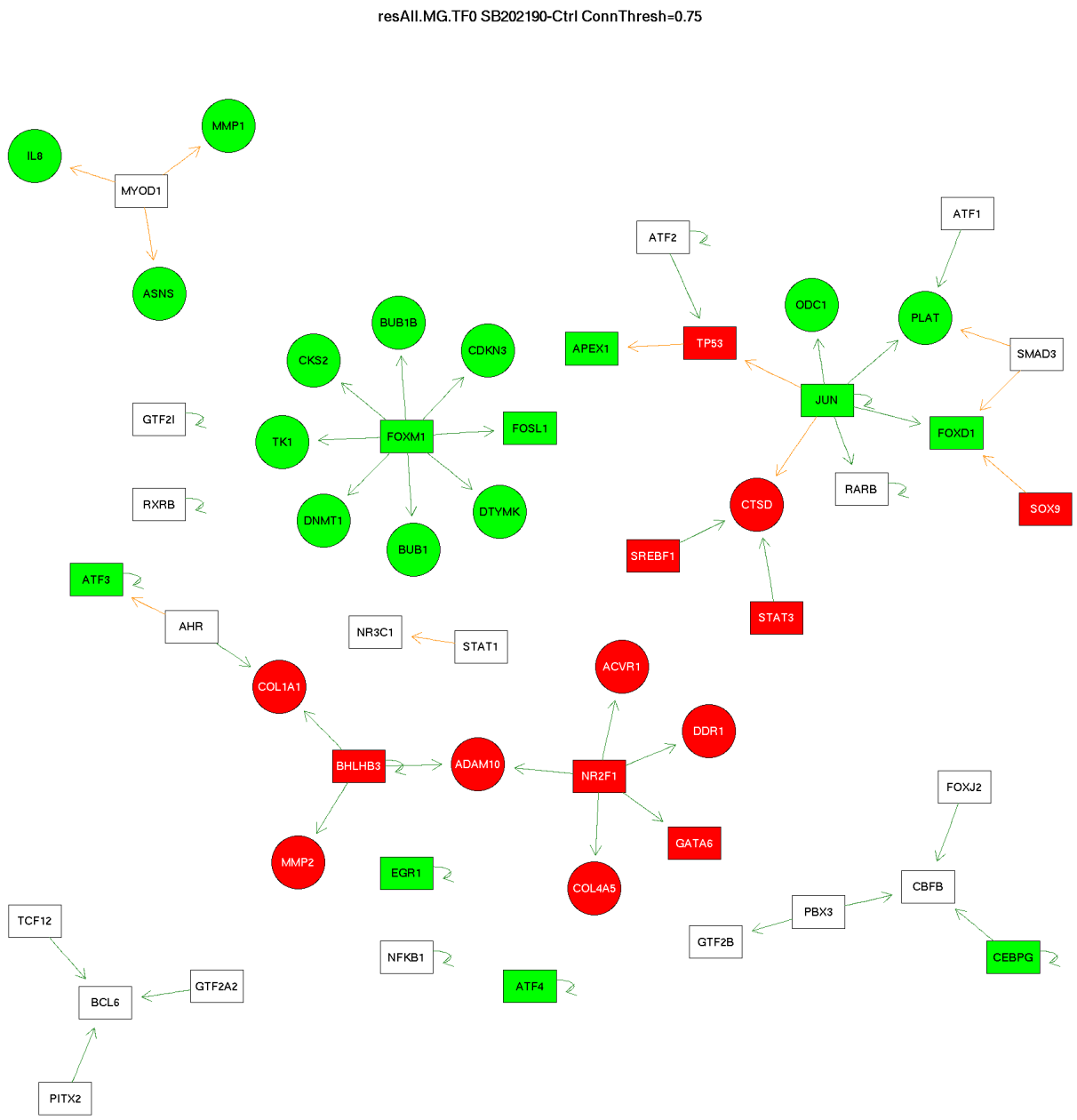


FIGURE 10: SAMPLE OUTPUT FROM CONNEXUS MODULE – LOCI HAVE BEEN TRANSFORMED INTO A NETWORK VIEW OF REGULATORY INTERACTIONS BETWEEN TRANSCRIPTION FACTORS (BOXES) AND OTHER GENES (CIRCLES)