AI2D-RST Annotation Manual

Version 1.0

Contents

1 Introduction **2**
1.1 Background 2
1.2 What to do with missing annotations? 3

2 The annotation tool **5**
2.1 How to use the annotation tool? 5
2.2 Commands for macro-grouping and grouping 7
2.3 Commands for connectivity 7
2.4 Commands for RST 7
2.5 Generic commands 8

3 Macro-grouping **10**

4 Grouping **15**

5 Connectivity **19**

6 Discourse structure **23**

Chapter 1

Introduction

1.1 Background

The AI2D-RST dataset is based on a dataset developed the Allen Institute for Artificial Intelligence Diagrams dataset (AI2D), which contains nearly 5000 diagrams with crowd-sourced annotations (Kembhavi et al. 2016). The dataset was originally intended for developing algorithms that can understand diagrams and answer questions about their content, but is now being used to study the multimodal structure diagrams as well (Alikhani & Stone 2018).

The AI2D-RST dataset provides an alternative description of the diagrams in the original AI2D dataset, which features multiple layers of description to capture aspects of their organisation (Hiippala & Orekhova 2018). Each diagram in the AI2D-RST dataset is represented by a Diagram object with three layers of annotation, which account for structural regularities, connectivity and discourse structure. In this context, the notion of 'structural regularities' refers to the organization and properties of diagrams that are provided visually to the viewer, covering essentially all material offered up for interpretation. That being said, a description of structural regularities aims to capture what is shown, not what is intended or 'meant'. The communicative intentions of a diagram are captured by the description of discourse structure. This division of labour allows interrogating discrepancies between structural regularities and communicative intentions.

1.2 What to do with missing annotations?

In some cases, the original crowd-sourced description of diagrams in AI2D is incomplete or erroneous. We do not 'compensate' for missing annotations to avoid providing mixed signals to any machine learning models trained using AI2D-RST. To exemplify, we do not annotate a connection between labels T2 and T0, T1 and T3 in Figure 1.1, as defined in Chapter 5, because the connecting line is missing. We do, however, draw a rhetorical relation between the label T2 and labels T0, T1 and T3, as the annotation for discourse structure does not rely on connectivity annotation (for more details, see the use case in Figure 6.7).

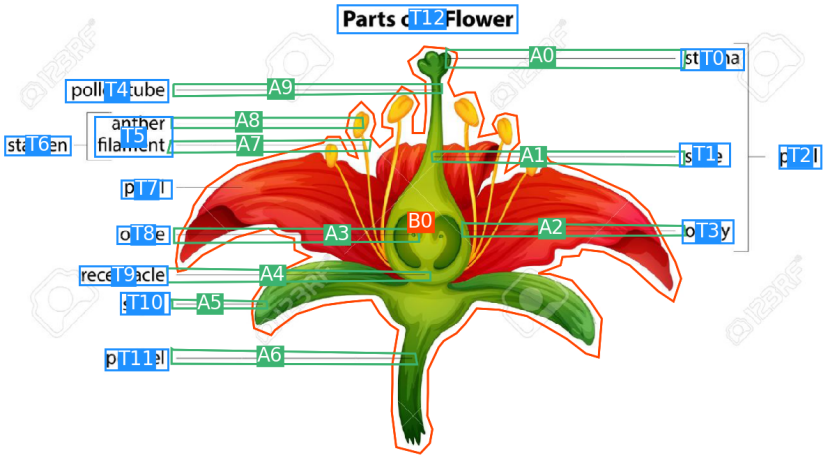

Figure 1.1: Diagram 3208 shows two missing elements, namely the 'bracket' lines associated with labels T6 and T2. T5, in turn, uses a single bounding box for two separate labels.

Figure 1.2 shows an incomplete annotation, in which the illustration has not been segmented to a sufficient detail and some of the identifiers are missing. As AI2D-RST does not revise the layout segmentation, we consider diagrams such as this as candidates for deletion.

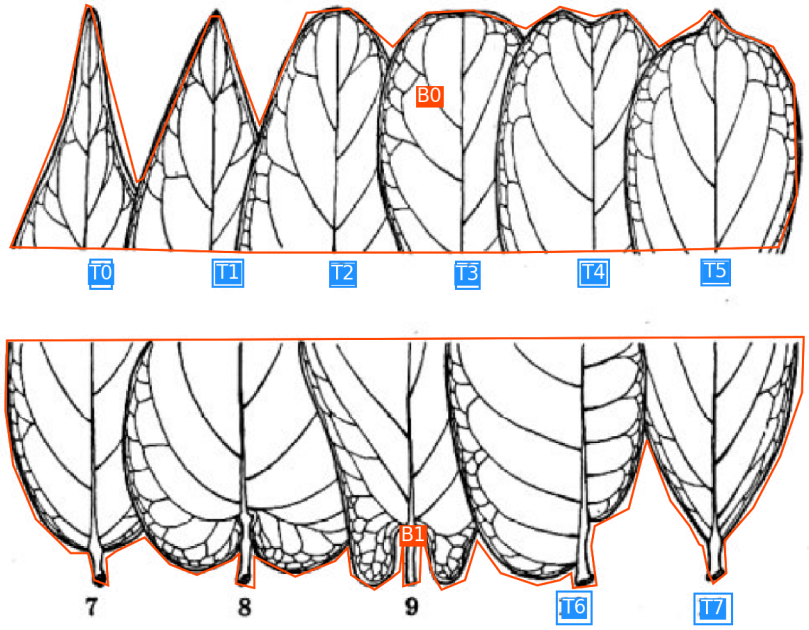

Figure 1.2: Diagram 4493 features insufficient detail for a reliable description using AI2D-RST.

Chapter 2

The annotation tool

The AI2D-RST dataset is accompanied by an in-house annotation tool available at `https://github.com/DigitalGeographyLab/AI2D-RST`. This chapter describes the commands available in the annotation tool and how they are used to describe the diagrams and their structure. The annotation tool is written in Python 3.6 and makes extensive use of libraries such as *matplotlib* (Hunter 2007), *NetworkX* (Hagberg et al. 2008) and *OpenCV* (Bradski & Kaehler 2013) and stores the annotation in pandas DataFrames (McKinney 2010).

2.1 How to use the annotation tool?

The annotator has two modes of operation: *normal* and *revision*. The normal mode is intended for normal annotation tasks, whereas the revision mode is reserved for checking and revising the annotation once the annotation process is complete for a batch of diagrams. The annotator may be opened in normal mode using the following command:

```
python annotate.py -a dataframe.pkl -i images/ -o output.pkl
```

In the above command, *dataframe.pkl* is the pandas DataFrame containing a batch of diagrams, *images/* is the directory containing the original AI2D diagram images and *output* is the pandas DataFrame in which the annotation is stored. To open the annotator in review mode, add the flag `-r` to the command as shown below:

```
python annotate.py -a dataframe.pkl -i images/ -o output.pkl -r
```

The annotation tool is used to perform three annotation tasks for each diagram: describing grouping, connectivity and discourse structure (RST). The annotator allows moving between the annotation tasks using the commands `group`, `conn` and `rst`. The active annotation task is indicated by the prefix of the annotator command line, e.g. [GROUPING], [CONNECTIVITY] or [RST]. The active task defines the available commands, which are described in Sections 2.2, 2.3 and 2.4. Generic commands available for all annotation tasks are described in Section 2.5.

The grouping annotation provides the foundation for annotating the diagram for connectivity and discourse structure using RST. For this reason, the grouping information is made available in both connectivity and RST layers, and updated to reflect any changes made in the grouping annotation. Figure 2.1 shows how the grouping information is represented in other annotation layers.

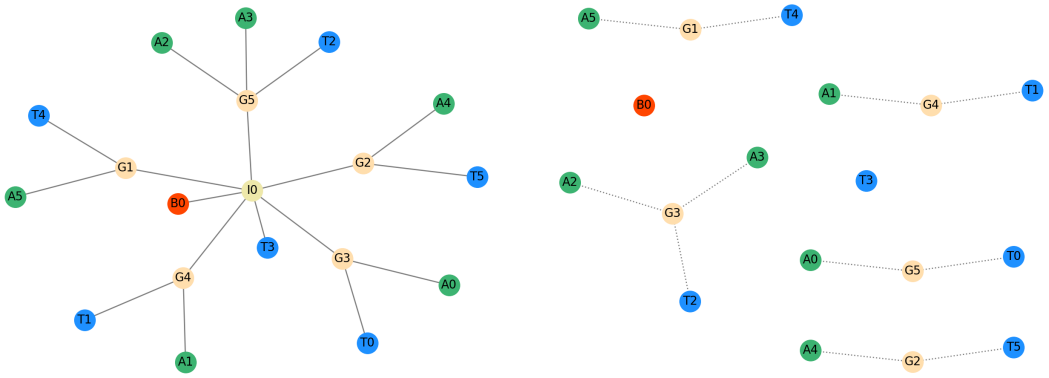

Figure 2.1: Completed grouping annotation (left) and incomplete connectivity annotation (right) for diagram 0.png. The grouping information is visualised using dashed edges on the right-hand side as a part of the connectivity annotation. The groups G1–5 and their terminal nodes may be picked up in the connectivity annotation. Note that grouping edges are automatically removed from connectivity and RST graphs when they are marked as complete.

When the annotation for the current layer is complete, the layer may be marked as complete using the command `done`. This prevents further annotation of the layer. The entire diagram is marked as complete when all annotation layers have been marked complete. The annotator will then move to the next diagram in the batch. Completed diagrams and layers are not opened for annotation again if the annotator is restarted in normal mode. To move to the next diagram in the batch without marking the annotation complete, use the command `next`.

2.2 Commands for macro-grouping and grouping

The syntax for grouping together diagram elements is given in section *Annotation* of Chapter 4.

- `macro`: add macro-group information to diagram element. For instructions, see section *Annotation* in Chapter 3.
- `macrogroups`: print the available macro-groups and their aliases, and the macro-groups defined for the current graph.
- `rm`: remove grouping node from the graph, e.g. `rm g1`.

2.3 Commands for connectivity

The syntax for defining connections between diagram elements is given in section *Annotation* of Chapter 5.

- `ungroup`: remove edges originating from the grouping annotation (visualised using dashed lines).

2.4 Commands for RST

The syntax for defining connections between diagram elements is given in section *Annotation* of Chapter 6.

- `ungroup`: remove edges originating from the grouping annotation (visualised using dashed lines).
- `rm`: remove RST relations or grouping nodes from the graph, e.g. `rm r1`.
- `split`: create multiple copies of diagram elements to allow them to be picked out by multiple RST relations. To split the text T1 into three parts, enter the command `split 3 t1`. For an example, see Figure 6.19 in Chapter 6.

2.5 Generic commands

The following commands are available for all annotation tasks. When referring to diagram elements with sequential identifiers, you can use a shorthand for identifier range: instead of typing out `g1, g2, g3, g4` you can simply write `g1:4`.

- `acap`: create a screenshot with *all* annotation layers in the current diagram. The screenshot is saved into the annotator directory with the name *all_graphs_id.png* where *id* is the identifier of the current diagram.
- `cap`: create a screenshot of the graph for the current annotation layer. The screenshot is saved into the annotator directory with the name *task_id.png* where *task* is the current annotation task and *id* is the identifier of the current diagram.
- `comment`: store a comment about the current diagram.
- `done`: mark the current annotation layer completed, preventing further edits to the graph unless the annotator is running in review mode.
- `exit`: save current work and exit the annotator immediately.
- `export`: save the current graph in Graphviz DOT format. The graph is saved into the annotator directory with the name *id_task.dot* where *id* is the identifier of the current diagram and *task* is the current annotation task.
- `free`: remove the edges leading to a specific node, e.g. `free b0`. To remove edges from multiple nodes, provide a list of valid identifiers, e.g. `free b0, t1`.
- `hide`: Hide the bounding boxes for layout segmentation and show the original diagram. To return the layout segmentation, enter the command `show`.
- `info`: print available commands and their definitions for the current layer.
- `isolate`: remove all nodes without edges from the graph.
- `next`: save current work and move to the next diagram in the batch.
- `reset`: reset the current graph. For the grouping annotation, the `reset` command performs a hard reset that loads the original AI2D annotation. For connectivity and RST annotation, the `reset` command returns the graph to the state it was in when starting the current task.

- `show`: shows the layout segmentation if the segmentation has been hidden using the command hide `hide`. The `show` command can also be used to highlight the bounding boxes for one or more diagram elements, while hiding the bounding boxes for others. To do so, enter the command `show b0` to highlight a single bounding box or `show b0, t1` to highlight multiple bounding boxes at once.

Chapter 3

Macro-grouping

Purpose. Macro-grouping seeks to capture higher-level cues about the organisation of a diagram. Such organisations are responsible for providing cues about generic structures in diagrams, allowing the viewer to generate expectations towards these structures. Figure 3.1 shows the macro-groups identified in the AI2D data using a system network.

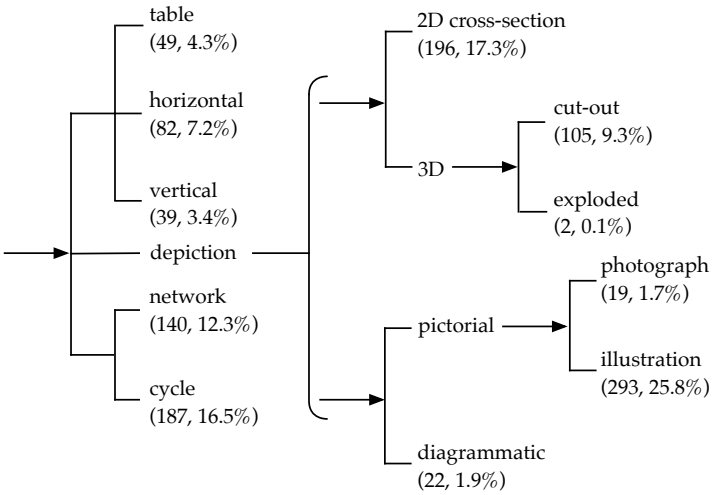

Figure 3.1: A system network representing choices among macro-groups. The numbers below the macro-group indicate their number and frequency in the AI2D-RST corpus.

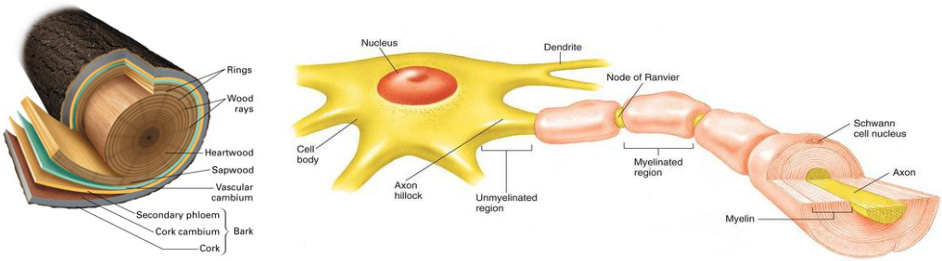

Figure 3.2: Two diagrams (2955 and 3003.png), which both use *cut-outs* to describe the objects they portray. As a macro-group, the presence of a cut-out depiction already signals that the labels and lines do not necessarily identify specific parts, but may refer to entire regions of the cut-out (see e.g. 'Heartwood' on the left and 'Myelin' on the right).

Data structure. Information about macro-grouping is stored as node attributes in the graph representing content hierarchy.

Annotation. Information about macro-grouping can be added to any node in the content hierarchy graph. To do so, enter the command `macro` followed by a valid node identifier `macro g1` or a list of identifiers separated by commas, e.g. `macro g1, g2, g3`. The annotation tool will then ask you to enter the name of the macro-group to be assigned to the nodes. You can either use the full name of the macro-group or its abbreviation, which can be printed using the command `macrogroups`.

Use cases.
Assigning macro-grouping information to an entire diagram. To add macro-grouping information to an entire diagram, enter macro-grouping information under the image constant I0 using the command `macro i0`.

Assigning macro-grouping information to parts of a diagram. If the diagram features multiple macro-groups, add macro-grouping information to the top-level grouping node, which joins together all the elements that macro group, as shown in Figures 3.3 and 3.4.

Assigning macro-grouping information to a specific element. If the diagram features elements that belong to different macro-groups, for instance, a photograph

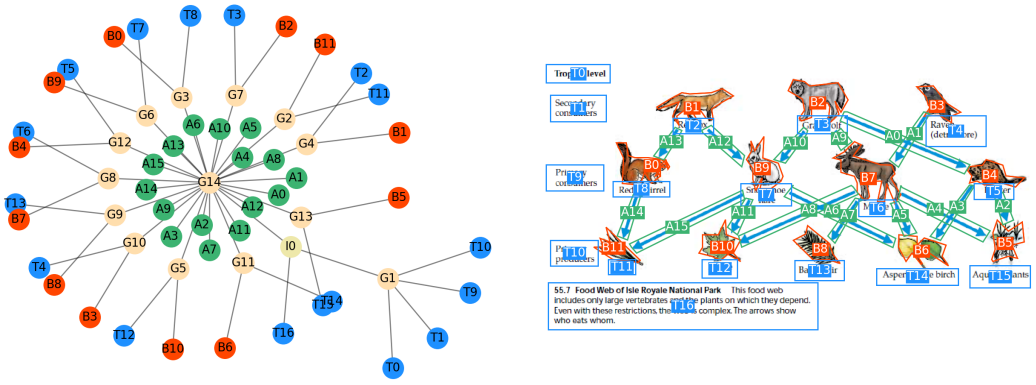

Figure 3.3: The hierarchy of content elements for diagram 274.png. The diagram features two macro-groups, a network (G14) and a vertical organisation (G1), which are both connected to the image constant IO. Macro-grouping information is assigned to grouping nodes G14 (macro g14 ; net) and G1 (macro g1 ; ver).

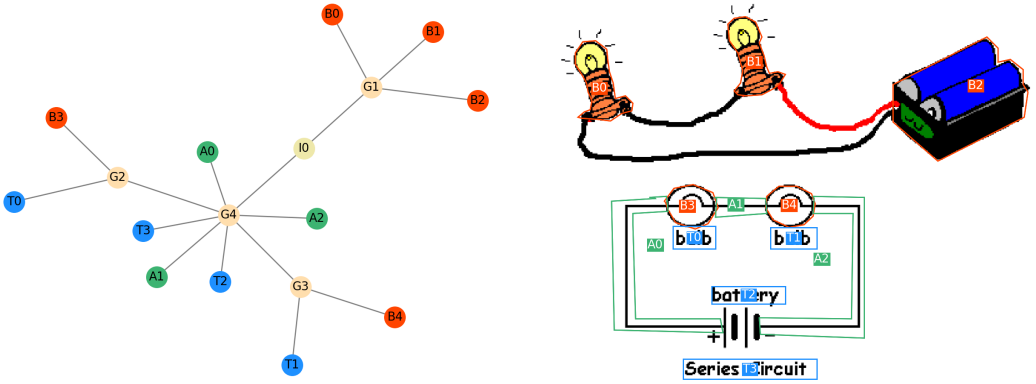

Figure 3.4: The hierarchy of content elements for diagram 1374.png. The diagram features two macro-groups, an illustration (G1) and a diagrammatic representation (G4), which are both connected to the image constant IO. Macro-grouping information is assigned to grouping nodes G1 (macro g1 ; ill) and G4 (macro g4 ; diag).

and an illustration, add macro-grouping information directly to the node.

Removing macro-grouping information. To remove macro-grouping information from a node, assign the macro-group 'none' to the node (e.g. macro b2 ; none).

Annotating table-like structures. Some diagrams feature a table-like organisation, which is annotated using the macro-group table. The annotator will then prompt

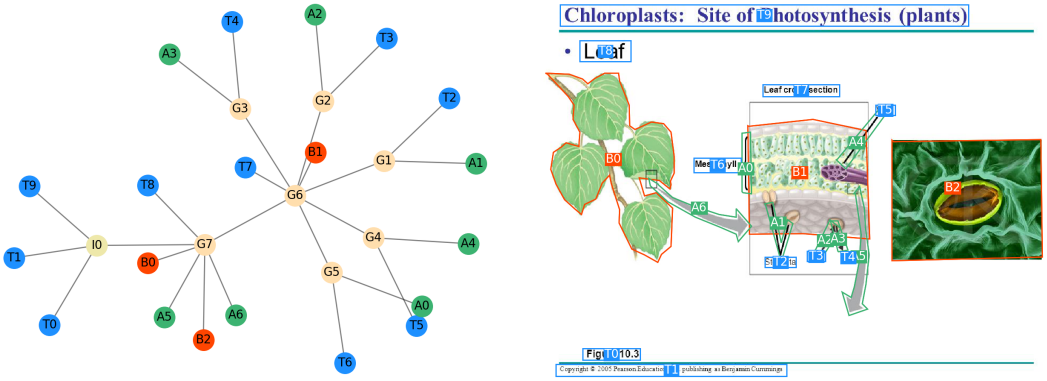

Figure 3.5: The hierarchy of content elements for diagram 4118.png. The diagram features three macro-groups, an illustration (B0), a cut-out (G6) and a photograph (B2), which are organised into a horizontal structure (I0). Macro-grouping information is assigned to nodes B0 (command `macro b0` followed by command `ill`) and B2 (`macro b2`; `photo`), and the grouping node G6 (`macro g4`; `cut`).

for additional details, namely the number of rows, columns and labels in the table. The grouping annotation should be completed first to enable the detailed description of table-like structures. Figures 3.6 and 3.7 exemplify the annotation of tables.

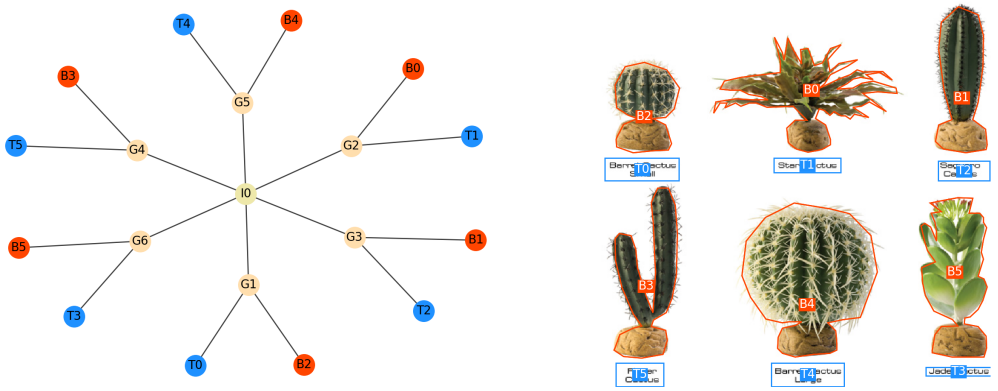

Figure 3.6: Diagram 5 features a table-like organisation with two rows, three columns and no labels. The macro-group information is assigned to the image constant I0 (`macro i0`; `table`). The two rows are populated by G1, G2, and G3 on the first row and G4, G5 and G6 on the second. The axes do not have labels.

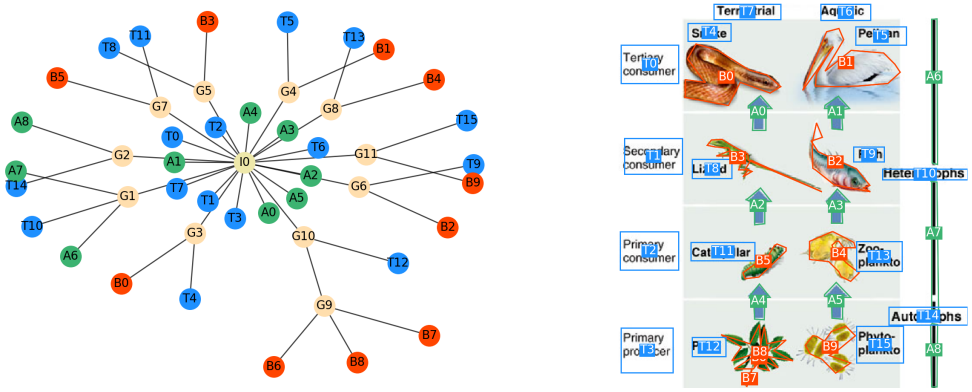

Figure 3.7: Diagram 16 features a table-like organisation with four rows, two columns and labels on three axes. The macro-group information is assigned to the image constant I0 (macro i0 ; table). Each row contains two groups (G3, G4; G5, G6, etc.). The labels for axis 1 consist of T0-3, while for axis 2 the labels are T7 and T6. Finally, for axis 3, the labels are groups G1 and G2. The need for groups shows why the grouping annotation must be completed before annotating tables for their structure.

Chapter 4

Grouping

Purpose. The purpose of grouping is to describe what kinds of groups are presented visually to the viewer. In AI2D-RST, the description of grouping is based on well-known principles of visual perception originally developed within Gestalt psychology. The principles of pattern perception applicable to static objects described in Ware (2012, 181–187) include:

- *proximity*: things that are placed close to each other are grouped together
- *concentration*: regions with similar element density are grouped together
- *similarity*: elements with similar shapes are grouped together
- *connectedness*: connected elements are grouped together
- *continuity*: elements that are smooth and continuous are easier to perceive than those with abrupt changes
- *symmetry*: symmetrical elements are grouped together
- *closure*: closed contours are likely to be perceived as independent elements

Some principles of grouping are clearly evident in the original AI2D annotation for layout segmentation (see e.g. element B6 in Figure 4.1, in which the two clouds are grouped together based on *similarity*). In AI2D-RST annotation, these principles are used to motivate the choices made when grouping together diagram elements, when constructing the hierarchy of content elements that serves as a basis for the

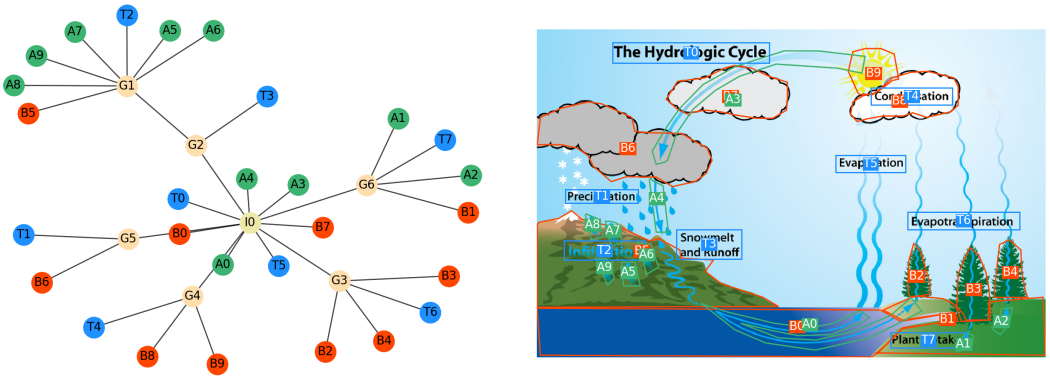

Figure 4.1: Diagram 1499 illustrates several principles of grouping active in diagrams. To begin with, the elements in group G1 may be grouped together based on the principle of *proximity*. The elements in group G3, in turn, may be grouped together due to principles of *similarity* and *connectedness*. This also illustrates a shortcoming in the AI2D layout segmentation: the crowd-sourced annotators have ignored the wavy lines, which act as connectors that complete the cycle in terms of connectivity.

annotation of connectivity and discourse structure.

Data structure. Grouping is represented using a tree graph. This means that all edges in the graph are undirected and the maximum number of edges between any two nodes in the graph is one.

Annotation. The graph is initially populated by nodes from the original AI2D annotation. Nodes are grouped together by entering their identifiers in the annotation tool. To exemplify, entering the command `a4, t5` in the annotation tool joins nodes A4 and T5 together under a group, which is then assigned an unique identifier, as illustrated by group G5 in Figure 4.2. Groups can be grouped together to increase the depth of the content hierarchy as necessary.

Use cases.
Labels and connecting lines. Labels and the lines that indicate what is being labelled are grouped together in the hierarchy, as shown in Figure 4.2.

Excessive detail in the layout segmentation. The crowd-sourced annotators responsible for the original layout segmentation have occasionally picked out objects in

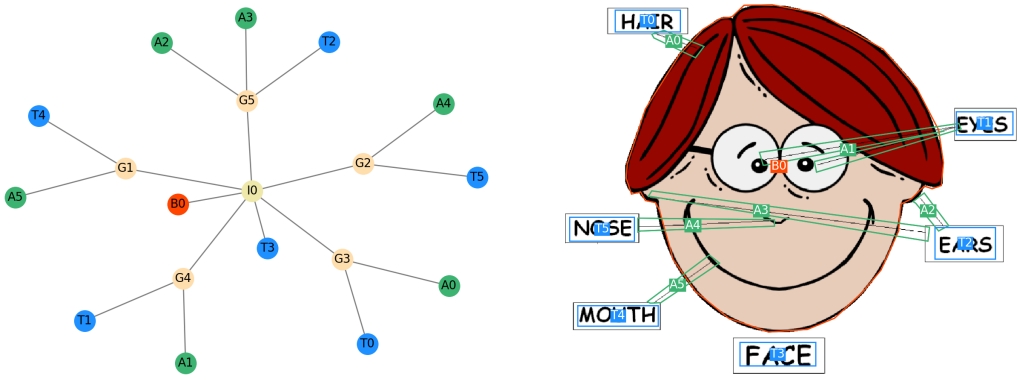

Figure 4.2: The hierarchy of content elements for diagram 0.png. Note that the hierarchy makes strong assumptions about the independence of the elements. The illustration B0 is connected directly to the root node I0, which stands for the entire diagram. The labels and connecting lines, in turn, are grouped together, as the lines are meaningless without their accompanying labels.

great detail. The parts of a whole are grouped together in the grouping annotation, as exemplified by Figure 4.3.

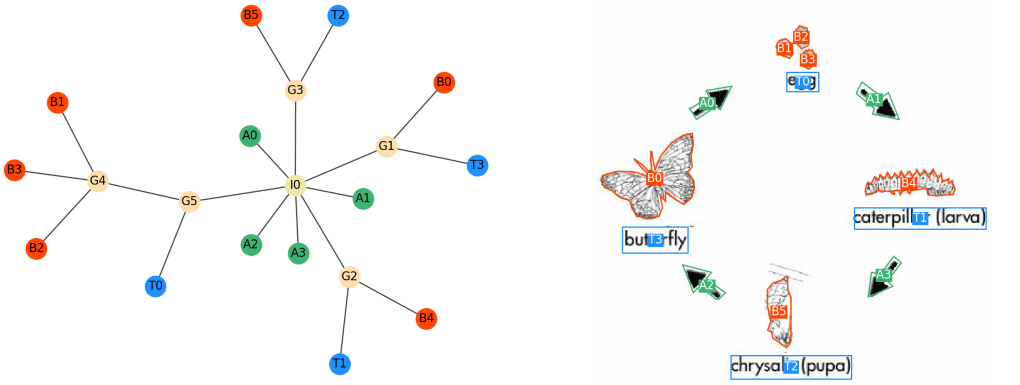

Figure 4.3: The hierarchy of content elements for diagram 55.png. Note that the crowd-sourced annotators have picked out each of the eggs B1–3. These nodes have been grouped together under the group G4 and the grouped together with their label T0. Together, the eggs and the label form the group G5.

Labels and lines that refer to multiple elements. A single label can be used to designate parts of one or more diagram elements, as exemplified by Figure 4.4. In this case, group each label and their connecting lines together, and associate this group

with the image constant I0. Finally, add a connection from the label (source) to all elements designated using the lines (targets) in the connectivity annotation.

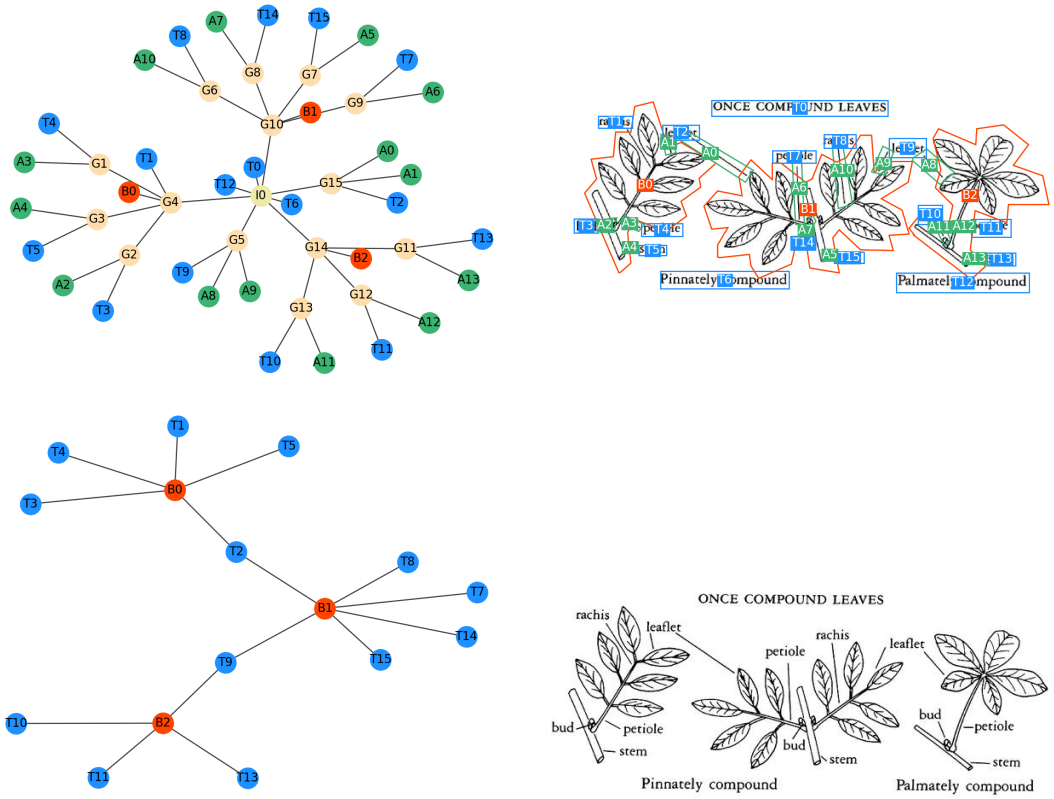

Figure 4.4: In diagram 4724.png, the labels for 'leaflet' (T2 and T9) designate multiple objects in the image. Both labels and their lines are grouped together in the content hierarchy (G5 and G15) and connected directly to the image constant I0. The connections between labels and their targets are described in the connectivity layer (note the two outgoing undirected connections from T2 and T9).

Chapter 5

Connectivity

Purpose. The purpose of connectivity is to describe connections between diagram elements, which are signalled visually using arrows and lines. Because connectivity (or *connectedness*) is a strong pre-attentive feature (Ware 2012, Marriott et al. 2012), the grouping annotation serves as the foundation for describing connectivity. This means that groups of elements can be picked out as sources and targets of connections when annotating a diagram for connectivity.

Data structure. Connectivity is represented using a directed graph, which may also feature cycles. A single node may have multiple connections. The connection type is stored as an attribute of the edge. A bidirectional connection is represented using two directional edges, which connect the source node to the target and vice versa.

Annotation. The directed graph is initially populated using the content hierarchy from the grouping annotation. The edges representing grouping information are visualised as dotted lines to separate them from edges that represent connectivity. The dotted edges are removed when the connectivity annotation is marked as complete.

There are two main types of connections, directed and undirected, which are represented using the edges of the connectivity graph:

- undirected (—)
- directed (⟶), which are further split into

– unidirectional ($\longrightarrow$)

– bidirectional ($\longleftrightarrow$)

All connection types have aliases in the annotation tool. To exemplify, entering the command `t4 - b0` in the annotation tool would create an undirected connection between the label T4 and the illustration B0 in Figure 4.2. The corresponding aliases are `t4 > b0` for unidirectional connections and `t4 <> b0` for bidirectional connections.

Use cases.

Annotating labels for connectivity. The labels and lines and arrows that support them are grouped together in the content hierarchy. In the connectivity annotation, however, the connections are drawn directly between the label and the element that is being labelled, as illustrated in Figure 5.1.

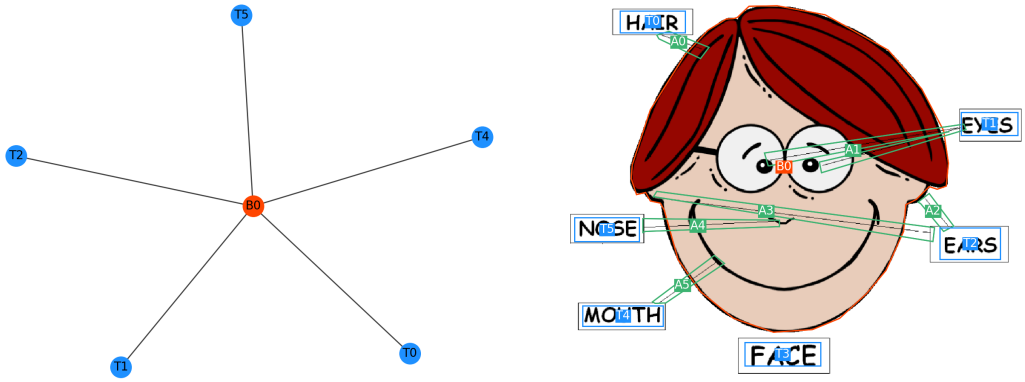

Figure 5.1: Diagram 0 shows undirected connections between the labels T0, T1, T2, T4 and T5 and the illustration B0.

Connections can be drawn between both nodes and groups. Figure 5.2 uses both individual nodes and groups of nodes as sources and targets. If the annotation would only account for the connections that are presented visually, the network diagram would have disconnections, as the incoming and outgoing connections terminate and originate at different elements (B7 and B6; B8, respectively).

Variation among diagrammatic elements. The visual appearance of diagrammatic elements that are used to signal connectivity show considerable variation. What

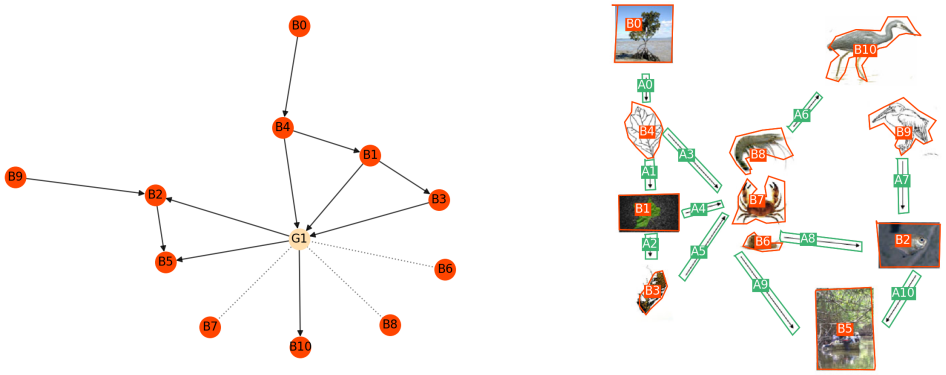

Figure 5.2: Diagram 1857 shows a network diagram, whose connectivity graph is presented on the left. The dashed edges indicate that nodes B6, B7 and B8 make up the group G1 in the grouping annotation. This group serves as a source (B10, B5, B2) and a target (B4, B1, B3) for incoming and outgoing connections, although the connections terminate at and originate in different elements of the group.

needs to be kept in mind that the connectivity annotation does **not** seek to describe the properties of arrows and lines, but the connections that hold between diagram elements and their directionality. Figures 5.3, 5.4 and 5.5 give examples of variation among arrows and connecting lines.

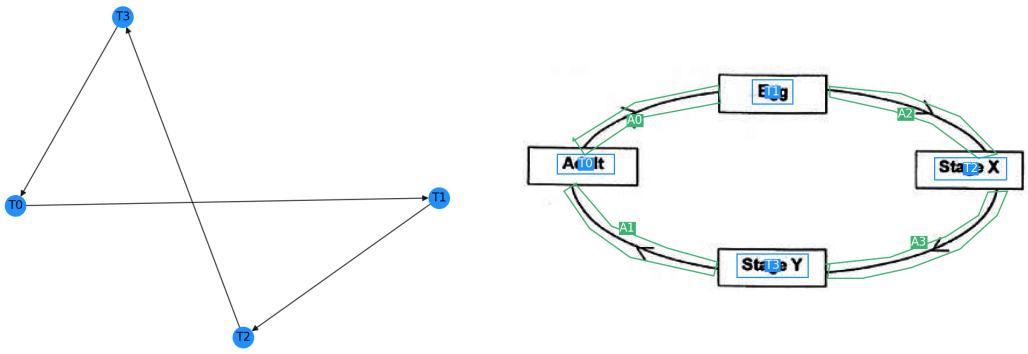

Figure 5.3: Diagram 415 features arrows with arrowheads in the middle. Their position does not affect the directed connections that these arrows set up between the diagram elements.

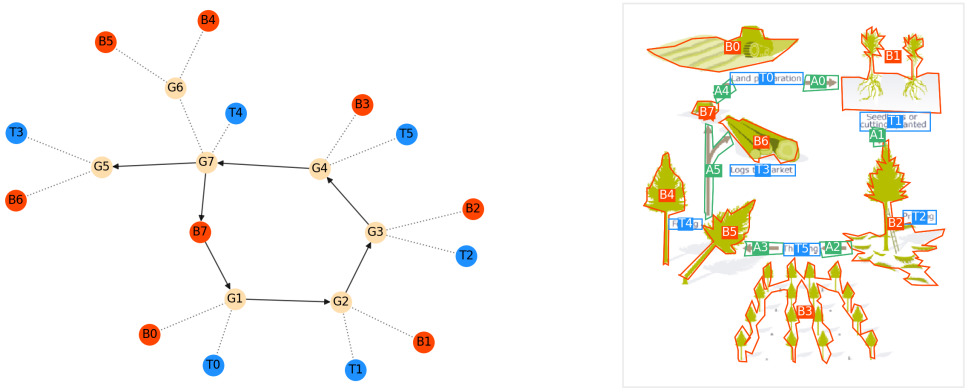

Figure 5.4: Diagram 616 features a double-headed arrow (A5), which is used to designate a connection that originates in group G6 and terminates in group G5 and element B7. Describing this connection in the annotator requires defining a target with multiple identifiers using the command `g1 > g7, b7`.

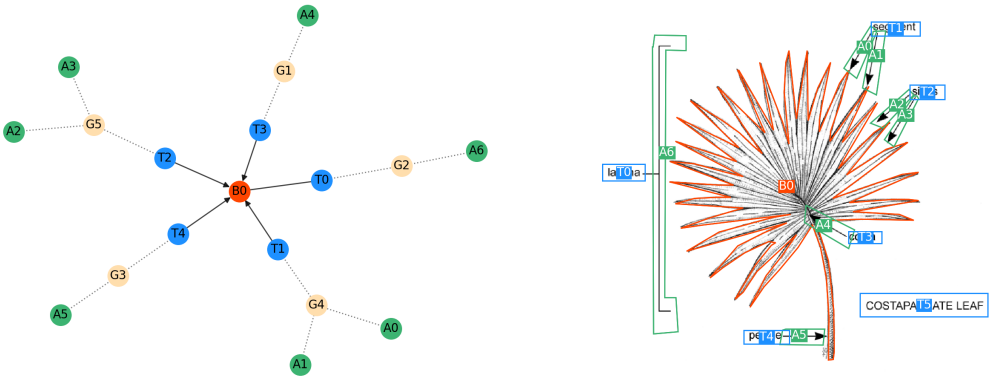

Figure 5.5: Diagram 569 features a double-headed line used to designate a part of an object. In the connectivity annotation, this is simply annotated as an undirected connection between the label and the object that is being labelled.

Chapter 6

Discourse structure

Background. Rhetorical Structure Theory (RST; Mann & Thompson 1988) was originally developed as a theory of text organization. The underlying idea behind RST is that a text can be considered coherent (and thus well-formed), if the readers can understand how the different parts of a text are related to each other, and how these parts serve the communicative goals of the entire text. To better understand how texts achieve their intended communicative goals, Mann & Thompson (1988) proposed breaking texts down into their component parts, which typically consist of clauses or their rough equivalents. Describing the relationships between these parts could offer insights on how coherence emerges in the text as a whole.

These relationships are referred to as *rhetorical relations* in RST. Each relation consists of at least two parts. Some parts are assumed to be more important than others in a relation. These parts are described using the term *nucleus*, whereas the less important parts are named *satellites*. The satellites are considered subservient to the nucleus. The actual rhetorical relations between these parts are described using a fixed set of relations. These relations are accompanied by criteria for their application and a schema for drawing diagrams that represent the rhetorical structure of a text. This schema and the key concepts of RST are illustrated with annotations in Figure 6.1.

To summarise, the parts of a text are described using the term *span*, which take on the role of nuclei or satellites when participating in a rhetorical relation. A span may consist of a text part or another rhetorical relation. Thus a typical description of a text using RST results in a hierarchical tree structure, whose depth increases as the text progresses. This causes a problem when RST is extended to describing discourse relations in multimodal artefacts, which do not necessarily follow a linear

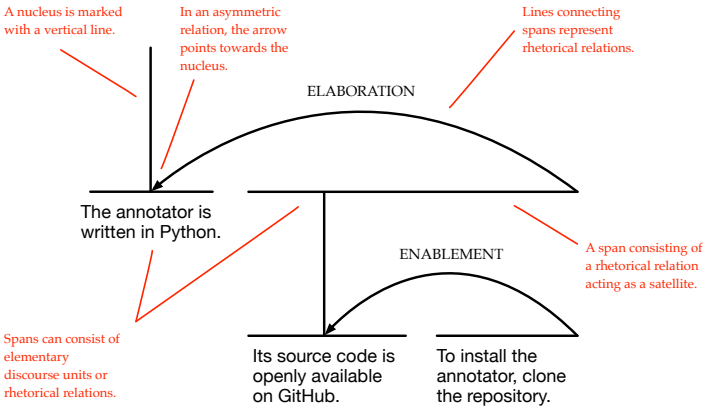

Figure 6.1: An example description of a short text using RST.

organization, but exploit the two-dimensional layout space when drawing discourse relations. Additional challenges emerge in defining the parts that should be picked out for analysis: multimodal artefacts often feature elements that are smaller than clauses, such as text fragments. To exemplify, consider labels in a diagram, which traditional RST would not recognise as analytical units. Furthermore, the lack of a generic framework for decomposing graphic elements remains a problem.

The description of discourse relations in AI2D-RST builds on the previous work of applying RST to multimodal artefacts or their parts (see e.g. Bateman 2008, Thomas 2009, Taboada & Habel 2013, Hiippala 2015). The following list describes key principles when applying RST in AI2D-RST:

- **Build on existing annotation**: graphic elements are not decomposed further than what is available for the original layout segmentation.
- **Preserve hierarchical organisation**: if certain diagram elements are picked out as spans in multiple RST relations, the `split` command can be used to create multiple copies of the diagram element (see use case *Diagram elements that participate in multiple rhetorical relations* below).
- **Rhetorical relations may act as nuclei**: rhetorical relations that hold between large spans are rare in written texts, but frequent in diagrams (see e.g. Figure 6.8).

Purpose. The AI2D-RST dataset uses Rhetorical Structure Theory (Mann & Thompson 1988, Taboada & Mann 2006) to describe the communicative *intentions* of a diagram. In contrast to the description of macro-groups, grouping and

connectivity, the annotation for the discourse structure consists of what Mann & Thompson (1988, 246) describe as plausibility judgements: the RST analysis is based on the analyst's assumptions about the designer's plausible intentions. In other words, the rhetorical relations are intended to capture what the designer likely meant.

Data structure. Discourse structure is represented using a directed graph. Each RST relation is represented as a node. A relation node must have edges towards nucleus and satellites or nuclei, depending on whether the relation is asymmetric (i.e. it has a nucleus and a satellite) or symmetric (i.e. it has multiple nuclei). Relation nodes can be picked up as nuclei or satellites in other RST relations, which means that the discourse structure graph is hierarchical. Each relation nodes includes attributes for node type (*kind*), relation name (*rel_name*), unique identifier (*id*), and nucleus and satellites or nuclei.

Annotation. The discourse structure graph is initially populated by elements and groups defined in the grouping annotation. To define a new rhetorical relation, enter the command `new`. The annotator will then prompt you to enter the alias for the relation. A list of available relations and the relations currently defined in the graph can be printed out using the command `rels`. Depending on whether the relation is asymmetric or symmetric, the annotator will then request the identifier of either nucleus and satellites, or nuclei. If the satellites or nuclei include more than one identifier, separate them using a comma, e.g. `b0, g1`. If the same diagram element participates in multiple rhetorical relations, the element may be split using the command `split`, as described in the use case *Diagram elements that participate in multiple rhetorical relations*.

Use cases.
Common text-image relations. Diagrams often draw discourse relations between written language and/or various types of images. Bateman (2008, 163) observes that the visuo-spatial combination of "label-connector-labelled" can be used to realise various kinds of rhetorical relations. For this reason, we define several use cases to unify their annotation.

- ELABORATION is used for both part-whole relations, as exemplified by Figure 6.9, and more extensive descriptions that add information to what is being described, as exemplified by Figure 6.2.
- IDENTIFICATION is used for cases in which one diagram element identifies another element, not just its parts. The element acting as the identifier may be either a word or an abstract alphanumeric identifier, as exemplified by Figures 6.3 and 6.4.
- CLASS-ASCRIPTION asserts that an element belongs to a particular class, as exemplified by Figure 6.5. In other words, this relation designates that an element is an instance of some class.
- PROPERTY-ASCRIPTION is used for cases in which the label is used to describe a property of the object that is being described, as exemplified by Figure 6.6.

Note that a text-image relation does not have to fall within these four relations: they simply cover the most common relations (for rarer alternatives, see e.g. CIRCUMSTANCE in Figure 6.8). Common text-image relations can also be combined in a single diagram, as shown in Figure 6.7, which features both ELABORATION and IDENTIFICATION.

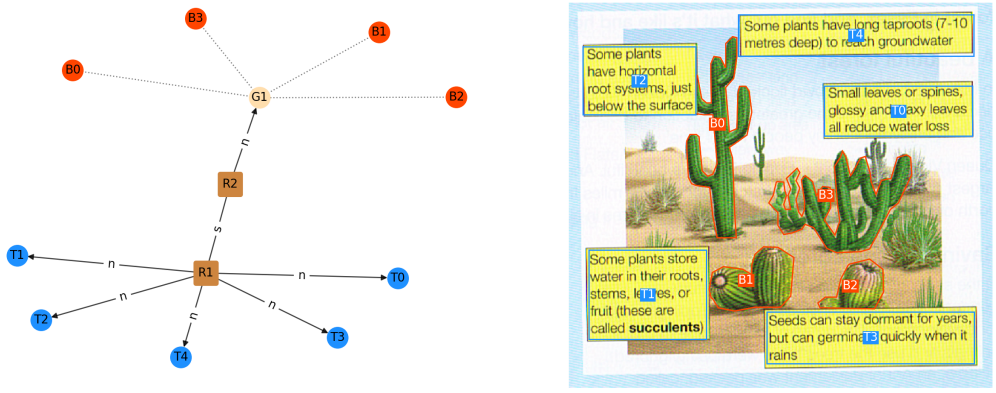

Figure 6.2: The labels T0–3 elaborate the group of illustrations G1. Note that the illustrations are grouped together, as the layout segmentation picks out parts of a larger illustration. Due to their shared rhetorical function, all labels are annotated as nuclei of the JOINT relation R1, as described in the use case *Elements with similar rhetorical functions*. This span acts as the satellite of an ELABORATION relation R2, whose nucleus is the illustration B0.

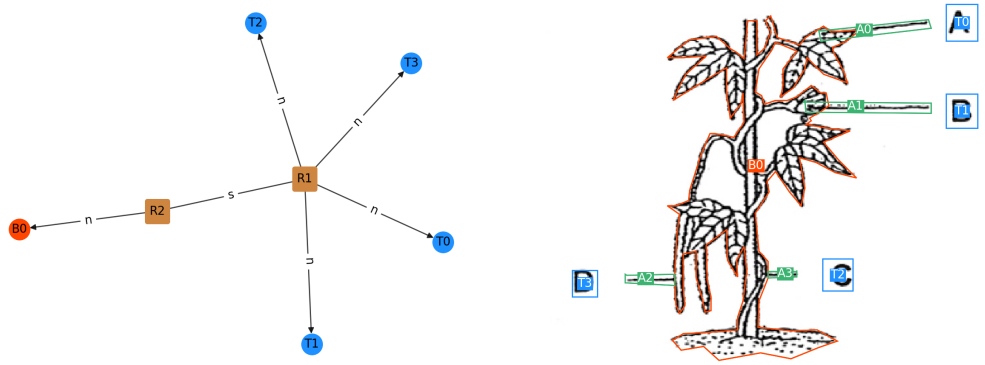

Figure 6.3: The labels T0–3 provide identifiers to parts of illustration B0. Due to their shared rhetorical function, all labels are annotated as nuclei of the JOINT relation R1, as described in the use case *Elements with a shared rhetorical purpose*. This span acts as the satellite of an IDENTIFICATION relation R2, whose nucleus is the illustration B0.

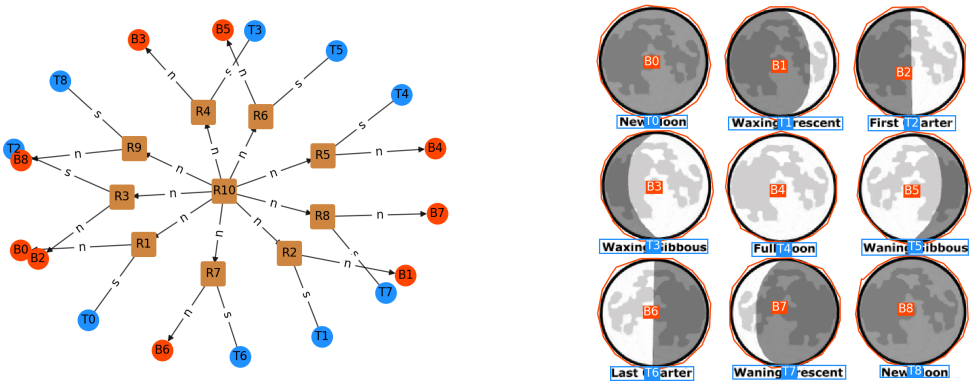

Figure 6.4: Illustrations B0–8 shows a sequence that identifies the different phases of the moon. Relations R1–9 capture this relation using IDENTIFICATION, in which the illustration is the nucleus (or 'identified') and the label is the satellite (or 'identifier'). Finally, all phases are annotated as nuclei of the SEQUENCE relation R10.

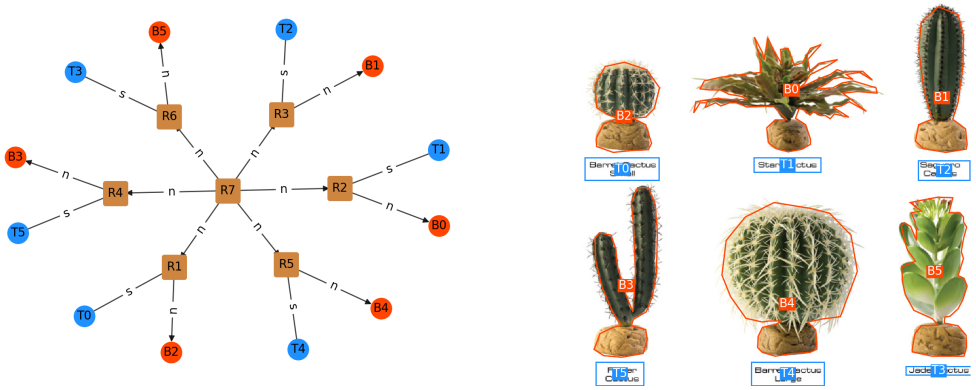

Figure 6.5: Photographs B0–5 show different types of cacti, whereas the labels T0–5 describe the class of each cactus. These relations are annotated as CLASS-ASCRIPTION (R1–6), in which the photograph is the nucleus and the label is the satellite. Finally, these relations are annotated as nuclei of the JOINT relation R7.

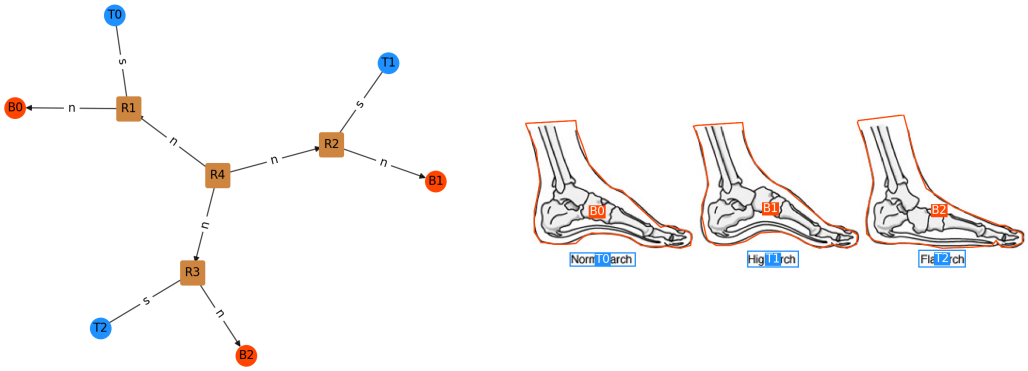

Figure 6.6: Diagram 1007 shows arches with various properties: 'normal', 'high' and 'flat'. The relation PROPERTY-ASCRIPTION captures something predicated of the object, e.g. "The arch is normal". In this example, the nuclei consist of the objects (B0–2) and their predicates (T0–2). Finally, the relation R1–3 are annotated as nuclei of the JOINT relation R4.

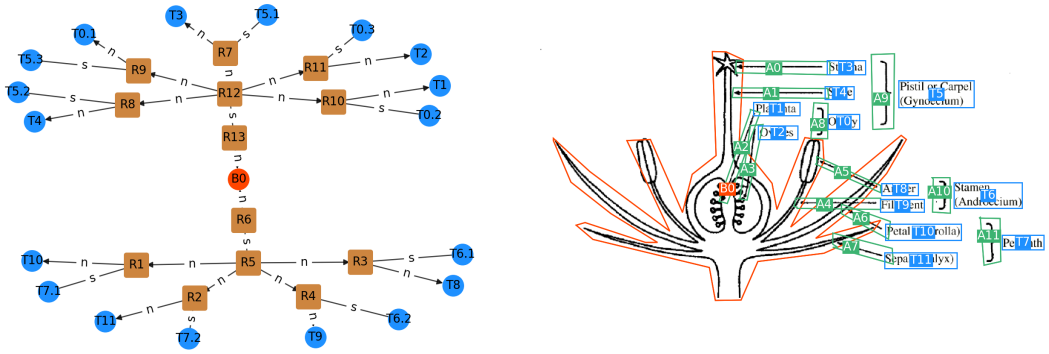

Figure 6.7: Diagram 3194 features labels that describe labels. These labels act as the satellites of IDENTIFICATION relations R1–4 and R7–11. Because a single label (e.g. T7) is used to describe multiple labels (e.g. T10 and T11), the label T7 needs to be split into two to preserve hierarchical structure of the graph. Due to their shared rhetorical functions, all labels are annotated as nuclei of the JOINT relations R5 and R12, as described in the use case *Elements with similar rhetorical functions*. These spans act as the satellites of an ELABORATION relations R6 and R13, whose nucleus is the illustration B0.

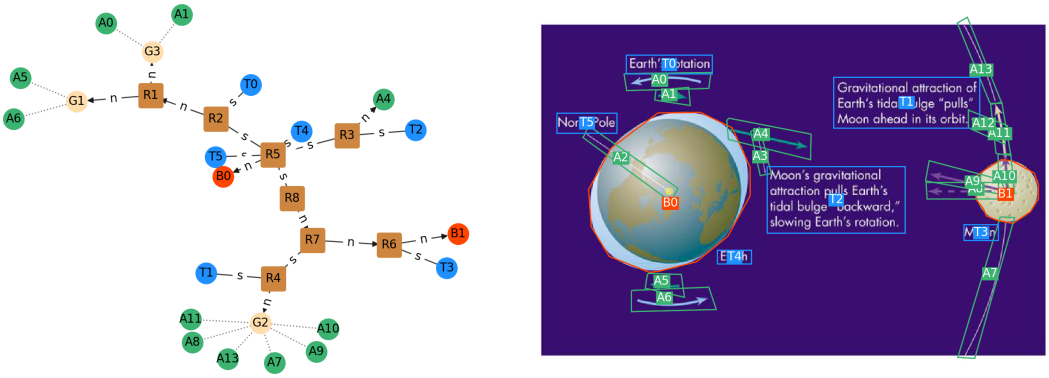

Figure 6.8: Diagram 2605 illustrates the reciprocal effect of gravity on the moon and the Earth. This is captured using the symmetric CONTRAST relation R8, in order to avoid arbitrary assignments of nuclearity, that is, deciding which celestial object is the nucleus and which is the satellite. The CONTRAST relation R8 is drawn between the top-level relations among both celestial objects. Two CIRCUMSTANCE relations, R3 and R4, help the viewer to understand that the celestial objects are to be contrasted with each other. In both cases, the nucleus of the CIRCUMSTANCE relation consists of an arrow (A4 and A11), whose frame of interpretation is provided by the satellites (T2 and T1).

Elements with a shared rhetorical purpose. Multiple elements of a single diagram can perform similar communicative tasks as a part of the discourse structure. These elements can be annotated as nuclei of the relation JOINT, if the participating elements exhibit strong similarities in the grouping and connectivity annotations. In the case of Figure 6.9, similarity means that each element is grouped together with a connecting line (see Figure 4.2) and they are all connected to the same element (see Figure 5.1). Figure 6.9 exemplifies how this JOINT relation is then picked out as the satellite of an ELABORATION relation.

Annotating sequences without cycles. Sequences without cycles are annotated using the symmetric relation SEQUENCE. The rhetorical spans participating in a sequence may consist of individual elements, groups or rhetorical relations, as exemplified by Figure 6.10.

Annotating sequences with cycles. Sequences with cycles are annotated using the symmetric relation CYCLIC SEQUENCE. The rhetorical spans participating in a sequence may consist of individual elements, groups or rhetorical relations, as exemplified by Figure 6.11. In order to distinguish between between sequences

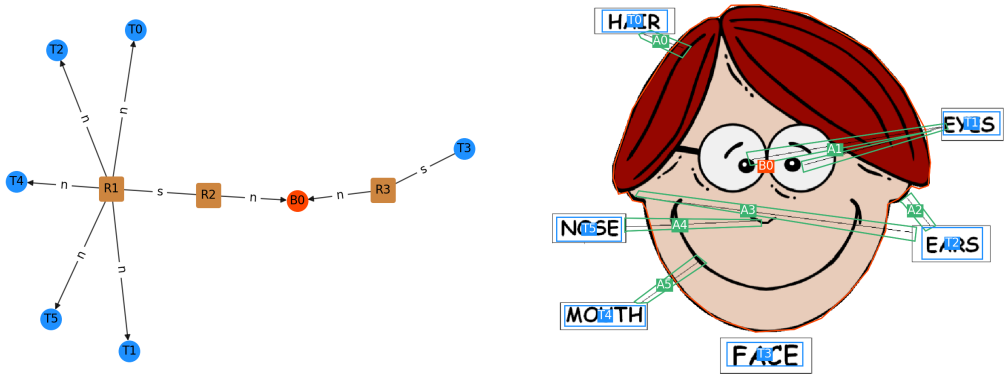

Figure 6.9: The labels T0, T1, T2, T4 and T5 describe parts of the illustration B0. Due to their shared rhetorical function, all labels are annotated as nuclei of the JOINT relation R1. This span acts as the satellite of an ELABORATION relation R2, whose nucleus is the illustration B0. The label T3 is annotated as the satellite of a PREPARATION relation R3, whose nucleus is the illustration B0.

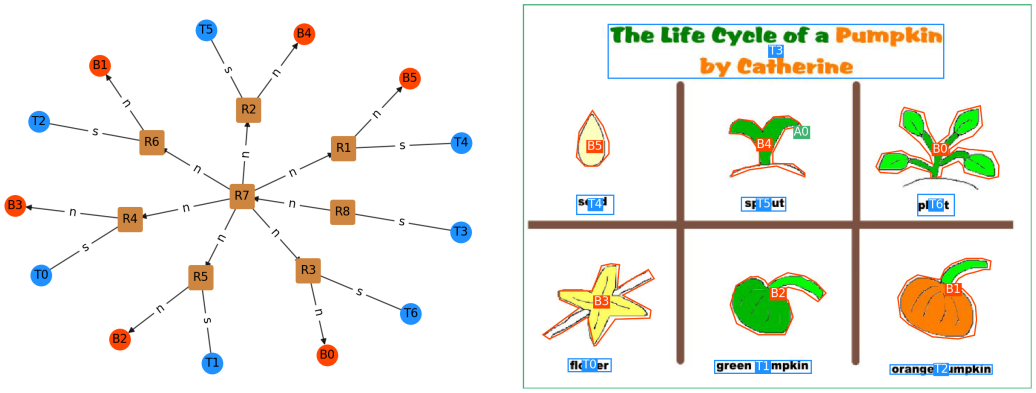

Figure 6.10: Diagram 2301 illustrates the life cycle of a pumpkin. Each stage of this cycle is represented using a combination of an illustration and a label. Each combination is annotated as IDENTIFICATION, in which the nucleus is an illustration (e.g. B5) and the satellite is a label (e.g. T4). These relations (R1–R6) act as the nuclei of the SEQUENCE relation R7. Note also the error in the layout segmentation, which defines an arrow (A0) that surrounds the entire diagram.

without cycles, which are described in the use case *Annotating sequences without cycles*, one can draw on the description of connectivity. If the connectivity annotation features cyclic connections (see e.g. Figure 5.3), it is likely that the CYCLIC SEQUENCE relation should be used instead of SEQUENCE. A single diagram can also feature multiple independent cyclic sequences, as shown in Figure 6.12. Al-

ternatively, the cycles may pick out the same diagram elements, as illustrated by Figure 6.19: this scenario is described in greater detail in the use case *Diagram elements that participate in multiple rhetorical relations*. Cycles may also feature stages and phases, as shown in Figure 6.13 and elaborated in use case *Annotating stages and phases*.

Annotating cycles with alternative stages or phases. Diagrams with cycles can feature alternative stages or phases or exits from the cycle. These alternatives are described using the DISJUNCTION relation, as described in Figure 6.14.

Annotating rhetorical relations between groups. Groups of diagram elements defined in the grouping annotation can be picked out as a part of the rhetorical structure, as illustrated by the group G1 in Figure 6.11.

Annotating rhetorical relations in networks. Networks as such do not possess a rhetorical structure: they simply set up connections between diagram elements, which are covered by the connectivity annotation. To accommodate possible rhetorical relations *within* the nodes of a network, such as illustrations and their labels, and titles of diagrams, all nodes of the network are annotated as nuclei of the symmetric relation CONNECTED, as illustrated in Figure 6.15.

Annotating rhetorical relations in networks with labelled connections. Diagrams often use arrows to represent various types of processes, whose meaning may be derived from their context (Alikhani & Stone 2018). In some cases, the processes represented by arrows may be described explicitly using labels. This needs to be accounted by the rhetorical structure, as exemplified by Figure 6.16.

Annotating rhetorical relations in networks with labels for multiple nodes. Sometimes networks are laid out in a way that allows using a single label for a group of nodes in the network. These diagrams typically feature multiple macro-groups, as exemplified in Figure 3.3. Figure 6.17 exemplifies this situation in a food web with an accompanying set of labels.

Annotating diagram elements that participate in multiple rhetorical relations. Diagram elements may be readily picked out as nuclei or satellites by multiple rhetor-

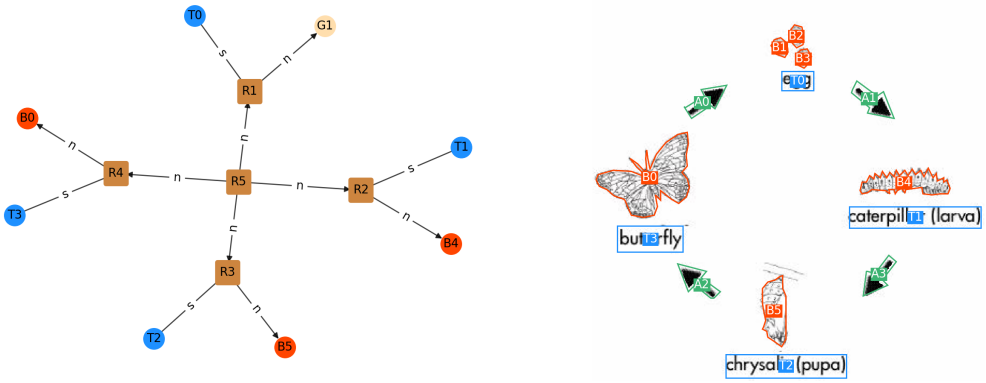

Figure 6.11: Diagram 55 illustrates the life cycle of a butterfly. Each stage of this cycle is represented using a combination of an illustration and a label. Each combination is annotated as IDENTIFICATION, in which the nucleus is an illustration (e.g. G1 consisting of B1–3) and the satellite is a label (e.g. T4). These relations act as nuclei of the CYCLIC SEQUENCE relation R5.

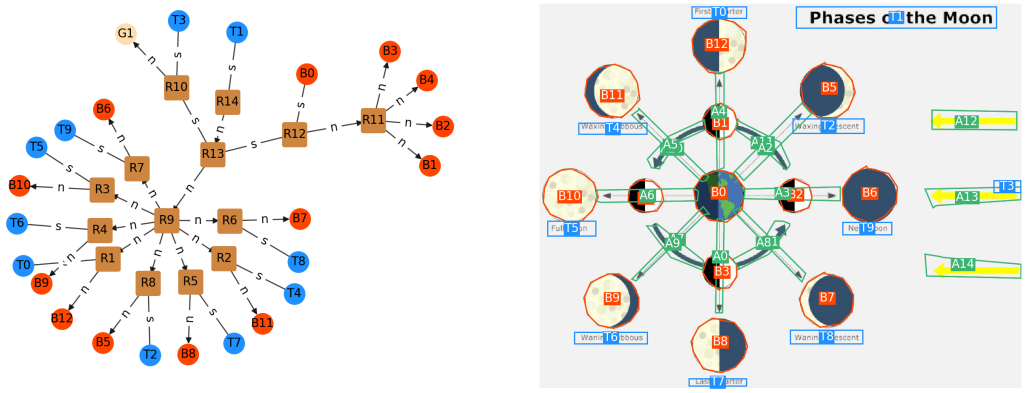

Figure 6.12: Diagram 1253 shows the phases of the moon as viewed from the Earth, which are represented by the arrows A0–7. These arrows are excluded from the description of rhetorical structure, as this information is captured by the connectivity annotation. In terms of RST, the diagram features two CYCLIC SEQUENCE relations R9 and R11. R9 consists of the moon phases and their labels, which act as the nuclei and satellites of IDENTIFICATION relations R1–8. The inner cycle, R11, is the nucleus of the CIRCUMSTANCE relation R12, whose satellite is the illustration B0. In other words, the illustration B0 provides provides a frame of interpretation for the cycle R11. Their combination acts as the satellite of another CIRCUMSTANCE relation, R13, which sets the frame of interpretation for the outer cycle R9. Note that the CIRCUMSTANCE relation R13 has two satellites: the other is the IDENTIFICATION relation R10, which provides another frame of interpretation: the appearance of the moon depends on the direction of sunlight. Finally, the PREPARATION relation R14 integrates the title text T1 into the structure.

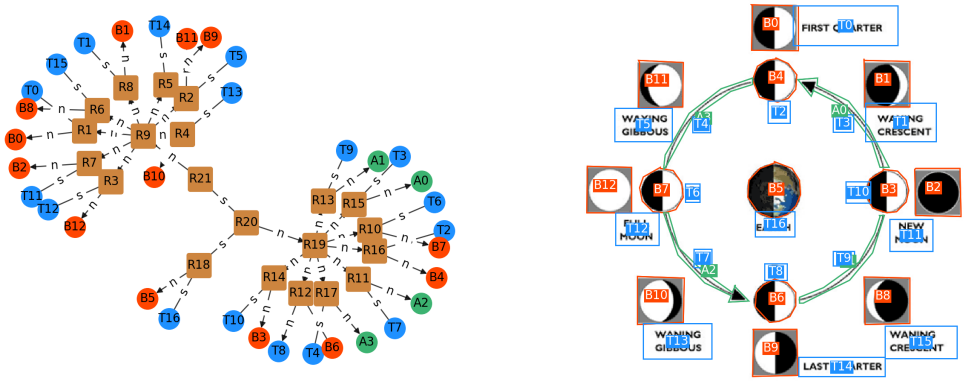

Figure 6.13: Diagram 139 shows two concurrent CYCLIC SEQUENCE relations R9 and R19. The outer cycle, R9, is a normal cyclic sequence, as described in the use case *Annotating sequences with cycles*, whose nuclei are the IDENTIFICATION relations R1–8. The inner cycle features stages and phases, as described in the use case *Annotating stages and phases*, whose nuclei consist of the IDENTIFICATION relations R9–15 and R18. The IDENTIFICATION relation R18 provides a framework for interpreting the inner cycle R19, forming a satellite of the CIRCUMSTANCE relation R21. This combination, in turn, acts as a satellite in another CIRCUMSTANCE relation, R21, which sets up the frame of interpretation of the outer cycle R9.

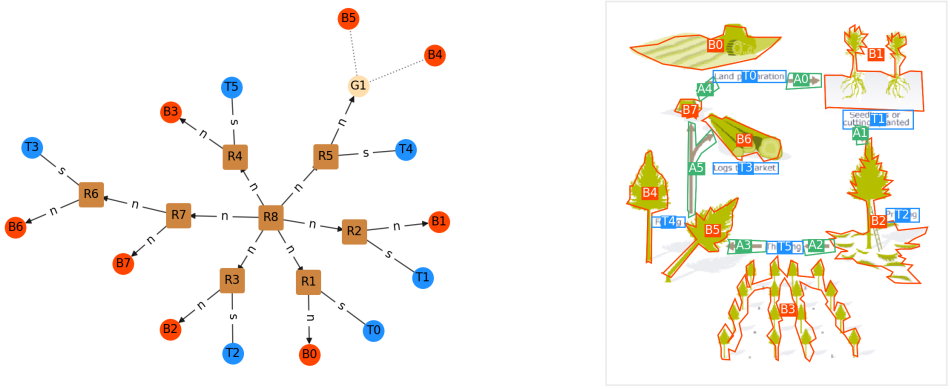

Figure 6.14: Diagram 616 features a cycle with an exit indicated by arrow A5, which shows two alternative outcomes in the cycle. This is expressed using the DISJUNCTION relation R7, which is one of the nuclei in the CYCLIC SEQUENCE relation R8. Other relations include IDENTIFICATION relations R1–R6 that hold between the illustrations and their labels.

ical relations. This kind of re-use and its consequences to the multimodal applications of RST is well known from previous literature (André & Rist 1995, Bateman 2008). Allowing this kind of direct re-use would require abandoning the principle of hierarchy, as exemplified by Figure 6.18. Figure 6.19 illustrates how the hierar-

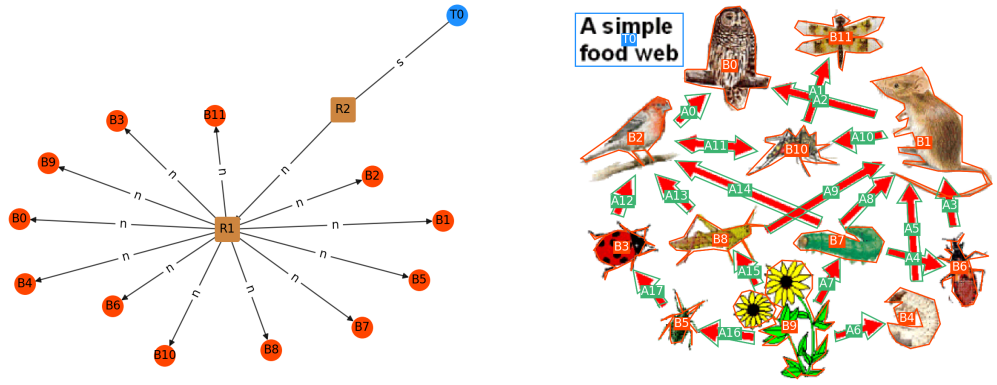

Figure 6.15: Diagram 22 shows a network, whose nodes consist of illustrations B0-11. These are annotated as nuclei of the CONNECTED relation R1. This relation also acts as the nuclei of the PREPARATION relation R2, in which the title T0 acts as the satellite.

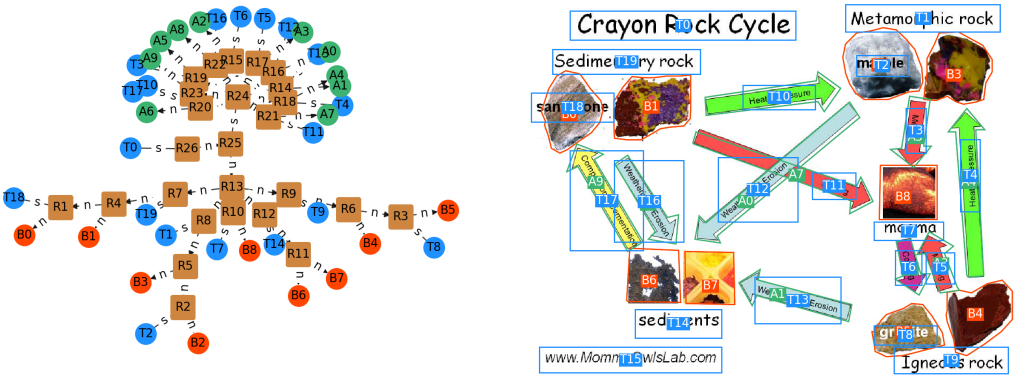

Figure 6.16: Diagram 1362 shows a diagram entitled Crayon Rock Cycle, which is essentially a network of interactions. The arrows and their labels participate in IDENTIFICATION relations, which are grouped together by the JOINT relation R24. This relation acts as a satellite of a CIRCUMSTANCE relation R25, providing a frame of interpretation for the CONNECTED relation R13. The CONNECTED relation sets up relations between the 'nodes' of the network. To zoom in on one such node, the IDENTIFICATION relation R1 involves the image of standstone as nucleus (B0) and the label T18 as satellite. Relation R1 and image B1 form the nuclei of a JOINT relationship R4, which is the nucleus of a PROPERTY-ASCRIPTION relation R17, whose satellite is the label 'Sedimentary rock' T19.

chy can be preserved by splitting the participating nodes using the split command.

Annotating stages and phases. Diagrams that represent sequences or cycles can use labels to describe both the stages of the sequence and its phases (Hiippala &

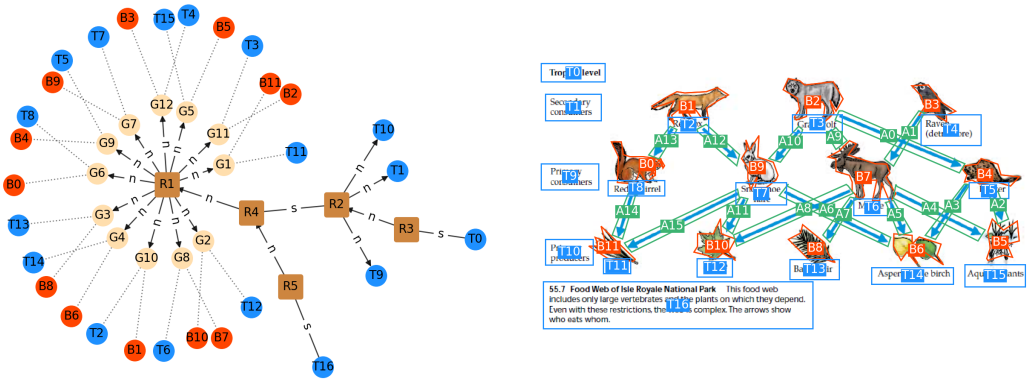

Figure 6.17: Diagram 274 features a food web with two macro-groups: a *network* and a *vertical* organisation that provides labels for the nodes in the network (see Figure 3.3). The node labels T1, T9 and T10 are nuclei of JOINT relation R2. This relation is the nuclei for the PREPARATION relation R3, whose satellite is the title T0. The relationship between the vertical organisation and the network is captured by the CLASS-ASCRIPTION relation R4, whose satellite is the relation R2, while the nucleus is the CONNECTED relation R1. Finally, the caption in T16 acts as the satellite of the PREPARATION relation R5, whose nucleus is the CLASS-ASCRIPTION relation R4.

Orekhova 2018, 1927). Such phases can be represented using the combination of an arrow and a label, as exemplified by Figure 6.19, which obviously need to be covered by the analysis of rhetorical structure. These relations are picked up as nuclei of the relation conveying sequentiality, such as CYCLIC SEQUENCE relations in Figures 6.19 and 6.13.

Annotating scales and legends. Diagrams often use scales and legends to add information to what is being represented. Figure 6.20 shows an example of a diagram with a scale, whereas Figure 6.21 shows an example with a legend.

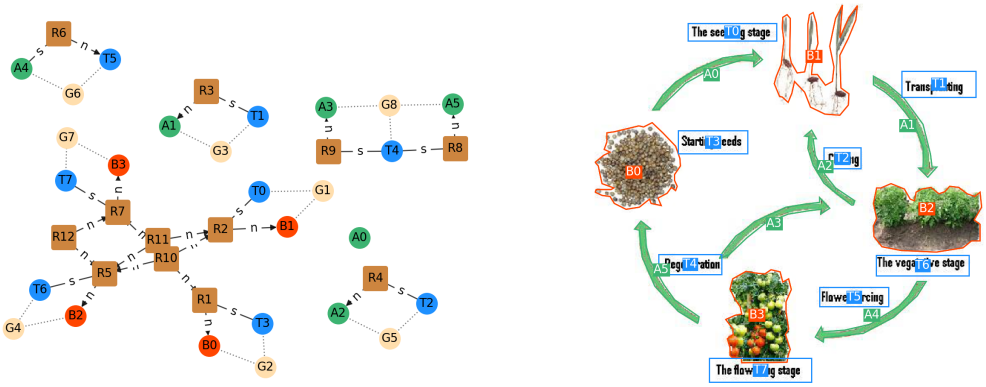

Figure 6.18: Diagram 573 features three cyclic sequences, which all involve illustrations B1, B2, B3 and their labels. In addition, text T4 describes both arrows A5 and A3. Defining three CYCLIC SEQUENCE relations between the participating relations R1, R2, R5 and R7 (all IDENTIFICATION) creates several cyclic connections (R11, R12), which are not allowed in a well-formed tree graph.

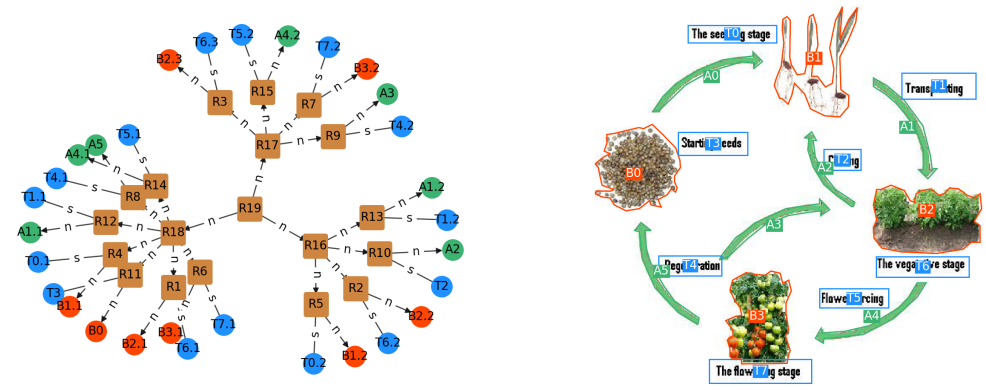

Figure 6.19: Representing the three cyclic sequences using a tree graph requires splitting the nodes for several illustrations, arrows and their labels. Each CYCLIC SEQUENCE relation (R16–18) consists of several IDENTIFICATION relations that hold between either illustrations or arrows and their labels. In order to avoid the situation presented in Figure 6.18, the nodes that participate in multiple relations have been split using the command `split`. To exemplify, the combination of the illustration B2 and its label T6 have been split into three parts using command `split 3 b2, t6`, which creates three nodes for each node. These different 'views' into the combinations of an illustration and its label participate in IDENTIFICATION relations (R1-3), which act as the nuclei of different CYCLIC SEQUENCE relations. The three cycles are joined together using the JOINT relation R19.

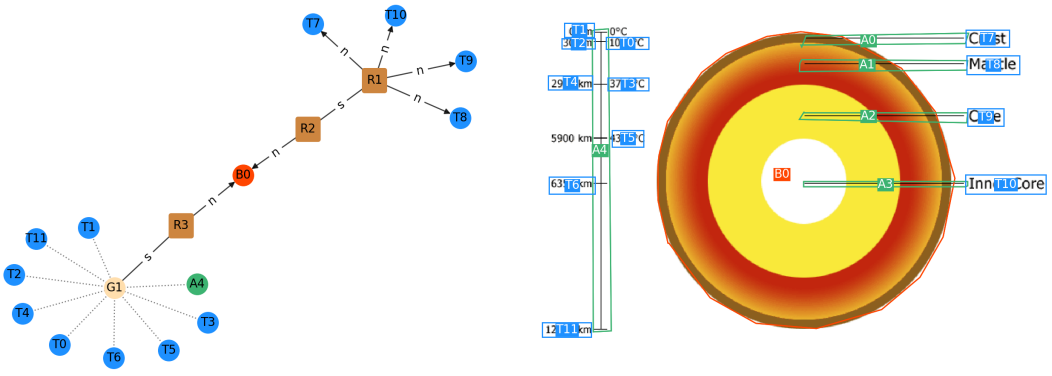

Figure 6.20: Diagram 4060 features a slice depiction of the Earth, which is accompanied by labels and a scale. The scale provides information on several properties of Earth, such as radius and temperature at different depths. Because scales are based on a spatial arrangement, a description of their rhetorical structure provides little value. For this reason, the whole scale is simply related to the object whose properties (e.g. size, temperature) describes, in this case, the illustration B0, using the PROPERTY-ASCRIPTION relation R3. Note that the labels, which all serve a similar rhetorical purpose, namely ELABORATION (R2), have been joined together under the JOINT relation R1, as described in the use case *Elements with a shared rhetorical purpose*.

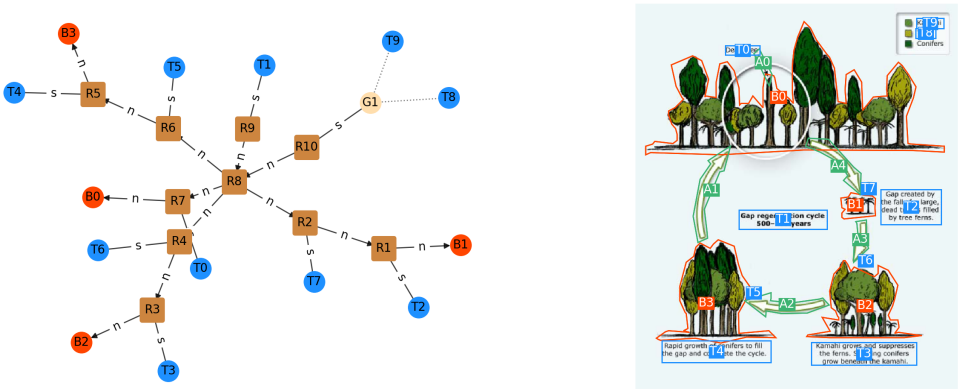

Figure 6.21: Diagram 2203 features a cycle accompanied by a legend, which is only partially annotated for two items (T8 and T9). Because legends are based on a spatial arrangement, a description of their rhetorical structure would add little value. For this reason, the group consisting of the legend items (G1) is taken to ELABORATE the entire cycle, which is represented by the CYCLIC SEQUENCE relation R8. Note that each step of the cyclic sequence consists of an illustration (e.g. B3), a descriptive text (e.g. T4) and an identifier (e.g. T5). The relation between B3 and T4 is described using an ELABORATION relation (R5). This relation acts as the nucleus of an IDENTIFICATION relation R6. Because this relation is the top-level relation for these groups, they are marked as nuclei of the CYCLIC SEQUENCE relation R8.

Bibliography

Alikhani, M. & Stone, M. (2018), Arrows are the verbs of diagrams, *in* 'Proceedings of the 27th International Conference on Computational Linguistics', Santa Fe, New Mexico, USA, pp. 3552–3563.

André, E. & Rist, T. (1995), 'Generating coherent presentations employing textual and visual material', *Artificial Intelligence Review* **9**, 147–165.

Bateman, J. A. (2008), *Multimodality and Genre: A Foundation for the Systematic Analysis of Multimodal Documents*, Palgrave Macmillan, London.

Bradski, G. & Kaehler, A. (2013), *Learning OpenCV: Computer Vision in C++ with the OpenCV Library*, 2 edn, O'Reilly, Sebastopol, CA.

Hagberg, A., Swart, P. & Schult, D. (2008), Exploring network structure, dynamics, and function using NetworkX, *in* 'Proceedings of the 7$^{\text{th}}$ Python in Science Conference', pp. 11–15.

Hiippala, T. (2015), *The Structure of Multimodal Documents: An Empirical Approach*, Routledge, New York and London.

Hiippala, T. & Orekhova, S. (2018), Enhancing the AI2 Diagrams dataset using Rhetorical Structure Theory, *in* 'Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)', European Language Resources Association (ELRA), Paris, pp. 1925–1931.

Hunter, J. (2007), 'Matplotlib: A 2D graphics environment', *Computing in Science & Engineering* **9**(3), 90–95.

Kembhavi, A., Salvato, M., Kolve, E., Seo, M., Hajishirzi, H. & Farhadi, A. (2016), A diagram is worth a dozen images, *in* 'Proceedings of the 14th European Conference on Computer Vision (ECCV)', Springer, Cham, pp. 235–251.

Mann, W. C. & Thompson, S. A. (1988), 'Rhetorical Structure Theory: Toward a functional theory of text organization', *Text* **8**(3), 243–281.

Marriott, K., Purchase, H., Wybrow, M. & Goncu, C. (2012), 'Memorability of visual features in network diagrams', *IEEE Transactions on Visualization and Computer Graphics* **18**(12), 2477–2485.

McKinney, W. (2010), Data structures for statistical computing in Python, *in* S. van der Walt & J. Millman, eds, 'Proceedings of the 9th Python in Science Conference', pp. 51–56.

Taboada, M. & Habel, C. (2013), 'Rhetorical relations in multimodal documents', *Discourse Studies* **15**(1), 65–89.

Taboada, M. & Mann, W. C. (2006), 'Rhetorical Structure Theory: looking back and moving ahead', *Discourse Studies* **8**(3), 423–459.

Thomas, M. (2009), Localizing pack messages: A framework for corpus-based cross-cultural multimodal analysis, PhD thesis, University of Leeds.

Ware, C. (2012), *Information Visualization: Perception for Design*, third edn, Elsevier, Amsterdam.