Annex D: Description of Imports

Sortal Imports

This section describes the functions of sortal imports. These are mostly seen in the files ‘sortTypes.py’, ‘parse_disjunctive_sort.py’, ‘convertA’, and ‘convertS’.

Import	Description
attributeSort	An Attribute Sort specifies a composition of a simple {Sort} (its base sort) with any other {Sort} (its weight/label sort) under the object-attribute relationship. An individual of the attribute sort is an individual of the base sort that has a form of the weight sort or label sort assigned as an attribute. For more information, refer to Annex C: Relevant Terminology.
disjunctiveSort	The DisjunctiveSort class represents a disjunctive sort additionally as an ordered set of component sorts. In its canonical version, component sorts cannot themselves be disjunctive sorts. For more information, refer to Annex C: Relevant Terminology.
label	A label is an alphanumerical data entity with a discrete behavior, i.e., the value of the sum of two labels is the collection of both labels, unless both labels are identical, in which case is either label. The Label class extends on the Individual class. It defines the characteristic individual for labels. A label is represented as a string. This characteristic individual accepts no parameters. It specifies an {sortal.map. ExactMap} as default. Forms of labels adhere to a discrete behavior.
line2D, line3D	A line is a linear, connected, non-bounded planar curve with a discrete behavior.

Import	Description
	The Line class extends on the Individual class. It defines the characteristic individual for lines. A line is represented as a direction vector and a position vector specifying the root of the line. This characteristic individual accepts no parameters. It specifies a {sortal.map. similarityMap} as default. Forms of lines adhere to a discrete behavior.
lineSegment2D, lineSegment3D	A line-segment is a connected and bounded segment of a line with an interval behavior. The line defines the co-descriptor of the line-segment, the boundary of the segment is defined by the start and end positions of the line-segment. Vectors or points may be used to define the start and end positions. The LineSegment class extends on the Line class and implements the Interval interface. It defines the characteristic individual for line-segments. A line-segment is represented as a line with two rational scalars specifying the tail and head relative to the line's root. This characteristic individual accepts no parameters. It specifies a {sortal.map. similarityMap} as default. Forms of line-segments adhere to an interval behavior.
point2D, point3D	A point is a 0-dimensional geometric data entity with a discrete behavior. The Point class extends on the Individual class. It defines the characteristic individual for points. A point is represented as a position vector. This characteristic individual accepts no parameters. It specifies a {sortal.map. similarityMap} as default. Forms of points adhere to a discrete behavior.
primitiveSort	Specifies a single data type. An individual of a primitive *sort* has a data value of the specified type. For more information, refer to Annex C: Relevant Terminology.
rule	Rule class that takes as input the rule description, LHS and RHS sides of rule.

Import	Description
similarityMap	Mapping function used to distinguish that two individuals of the same sort type (e.g. two line segments ls1 and ls2, or two points p1 and p2) are of the same sort type despite being in different locations or having different coordinates, and that they can be mapped together.
vector2D, vector3D	A vector specifies a position in a two-dimensional Cartesian space. If normalized, it only specifies a direction. The Vector class defines a vector as a pair of {coordinate}'s and a w factor to reflect the vector's infinity characteristic A Vector object is never modified after creation; thus, it can be used multiple times.
Weight, rWeight	A weight specifies a value for the shade of black to white of a plane or a line segment, or perhaps the width of a line. Weight can be defined from a range of 0 to the maximum value. A special version of weight, called rWeight, performs addition and subtraction of weights arithmetically.

Rhino-specific Imports

This section describes the functions of Rhino-specific imports.

Import	Description
addColor	This method adds color to an object or several objects. It prompts for the selection of the objects and the RGB value of the color to change the objects to. Note that color is not visible, however, in Print Preview (commonly used when viewing line segment widths).
addDesc	This method adds a description to the User Text dictionary of a Rhino object under the key 'Desc'. If applied onto a point or a line segment that already has a description, it changes the text of the description.
addEnum	This method adds an enumerative value to the User Text dictionary of a Rhino object under the key 'Enum'. If applied onto a point or a line segment that already has an enumerative value, it changes the contents.
addLabel	This method adds a label (visualized as text dot) to either a point or a line segment. If applied onto a pre-existing text dot or a line segment that already has a label, it changes the text of the label.
addWeight	This method adds weight to a point, text dot or line segment, or several objects at once. With points and text dots this is shown as grayscale color (0 – black to 255 – white), and with line segments this is shown as line width. Line width is only visible when Rhino is set to Print Preview.
attachLabel	This method attaches a pre-existing text dot as a label to a pre-existing line. It changes the User Text dictionary value that corresponds to the key 'Label' to the text of the text dot, stores the GUID of the selected text dot in the User Text dictionary of the line under the key 'LabelDot', and moves the text dot to the midpoint of the line it is attached to.
clearFrame	This method prompts for the selection of a frame and clears all Rhino objects inside that frame.

clearShape	This method takes as input a list of GUIDs or a list of lists of GUIDs, and deletes them from the Rhino workspace.
convertA	The method 'convertA.toSortal' takes as input an agnostic tuple and returns a sortal metaform.
convertR	The method 'convertR.toAgnostic' takes as input a list of Rhino GUIDs, the reference point for the frame the shape is inside of, and returns an agnostic tuple.
convertS	The method 'convertS.toAgnostic' takes as input a sortal metaform and returns an agnostic tuple.
create_rule	This method takes as input the rule name, rule description, and prompts for the selection of the frames where the LHS and RHS shapes are contained in. If the rule name given already matches that of a pre-existing rule, then there is the option to either overwrite the pre-existing rule or rename the rule currently being created. It returns the rule instance and the rule name.
delete	This method clears the sort and rule register of any previous data, deletes all Rhino objects present in the window, and sets up empty frames and layers.
delLabel	This method removes the label of a line segment by setting the value for the key 'Label' in the User Text dictionary of the selected line to empty, meaning that in future handling the line will not have a label attribute. It also deletes the text dot associated with the line, i.e. the text dot that corresponds to the GUID stored in the User Text dictionary of the line under the key 'LabelDot'.
extract	This class of methods uses the function 'extract.getIndividuals' to prompt for a frame where the shape to be extracted is housed in. The method returns the list of Rhino GUIDs corresponding to the objects inside the frame, the layer name the objects are drawn on, and the reference point or insertion point of the frame block instance selected.
find_rule_appns	This method takes as input a subshape and shape (agnostic form), and a rule. It checks if the subshape is part of the larger shape before generating matches and applications for both shapes. The method then checks which applications of the larger shape match those of the subshape, convert the LHS-RHS-rule application combinations from sortal to agnostic, and combine all these into a list that is returned.

visualA	This set of methods can be found in ‘visualAgnostic.py’ and is a class that takes as input the agnostic form to be visualized, the layer the form is to be drawn on, and the reference point for the figure. It has a method called ‘plotAgnostic’ that draws the figure and returns the list of Rhino GUIDs corresponding to the shape just drawn. An instance of ‘visualA’ stores the object GUIDs of the Rhino objects generated by its ‘plotAgnostic’ method.