

A DOMAIN SPECIFIC LANGUAGE AND TOKENIZATION FOR CONSTRUCTION SEQUENCES

A.1 GEOMETRY

The domain specific language used to represent geometric prompts, construction sequences and the final profiles, share a common vocabulary for geometric entities. Scalar values, points and infinite lines are represented by single tokens as shown in Table 3. More complex entities are then constructed using a combination of special tokens denoting the entity type, followed by primitive tokens representing the geometry as shown in Table 4.

Table 3: Geometry and scalar values represented by single tokens

Token	Explanation
Length	Signed lengths in the range $[-1, 1]$ are quantized into 127 bins. An odd number of bins is deliberate chosen so that the values of -1.0 , 0.0 and 1.0 can be recovered exactly when the quantized values are converted back to floating point numbers.
Angle	Angles in the range $[0, 2\pi)$ are quantized into 121 bins. Care is taken to ensure the that the angles 0.0 and 2π map to the center of the first bin. As 120 has prime factors 2, 3 and 5, this quantization strategy allows common angles like 0.0 , π , $\pi/2$ and $\pi/3$ to be exactly reconstructed.
Point	Points are quantized onto a 127×127 grid. An odd number of grid cells in each direction is chosen to ensure that the center point in the domain can be exactly reconstructed when the quantized representation is de-quantized.
Inflin	Directed infinite lines are represented in hessian normal form. The angle between the line and the x -axis is encoded into 121 bins and the signed distance to the origin is encoded into 127 bins. Infinite lines are then represented as single tokens with 121×127 possible values.
Area	Area values are used in the geometric prompt to control profile thickness. After normalization, the profiles' areas will be in the range $[0, 1]$ and can be quantized into 127 bins.
Complexity	The number of curves in the profile is used as a measure of complexity. Complexity tokens have 127 possible values with the last token value used for profiles containing more than 127 curves.
NumLoops	The number of loops in the profile is represented by a single token with 127 possible values.
SmoothVertices	The fraction of vertices which are tangent continuous is quantized into 127 bins.

Table 4: Geometric entities defined by a combination of multiple tokens. Each multi-token entity starts with a special token denoting the entity type followed by a list of tokens to define the required geometry.

Tokens	Explanation
BoundedLine Point // Start point Point // End point	Bounded line segments start with a BoundedLine token followed by two point tokens representing the start and the end point of the line segment.
Arc Point // Start point Point // Mid point Point // End point	Arcs are represented with an Arc type token following three point tokens representing the start, mid and end points of the arc segment.
Circle Point // Center point Length // Radius IsCCW // Counter-clockwise	Oriented circles are represented with a point token for the center point and a length token for the circle’s radius. An additional flag indicates whether the circle is oriented clockwise or counter-clockwise.
BoundingBox Point // Min point Point // Max point	A bounding box is encoded by two points at the min and max corners.
DimensionDatum Point // Datum point Incline // x-axis Incline // y-axis	The dimension datum defines a fixed reference point from which other geometry can be constructed. It consists of a datum point with two infinite lines passing through it, aligned with the x and y coordinate system axes.
BoltHole Point // Center point Length // Radius Length // Clearance disk	Bolt holes are common in mechanical parts. They are used in the geometric prompt to define both the hole’s position and the required clearance around it. The tokenization contains the center point, a radius representing the size of the hole’s aperture and a clearance disk representing material which should be kept in place surrounding the hole to prevent tear-out of the bolt.

A.2 GEOMETRIC PROMPT

The generation is guided by geometric prompts, which specify the geometry the generated profile should match, along with other important properties such as profile area and center of gravity. Appendix B.2 describes the rationale for each component in the geometric prompts in more detail and Appendix D shows some examples of how the prompt can be used to control the shape. The tokenization of the prompts is shown in Table 5 which includes lower level entities from Tables 3 and 4. An illustration of a geometric prompt and the generated profile is shown in Figure 3.

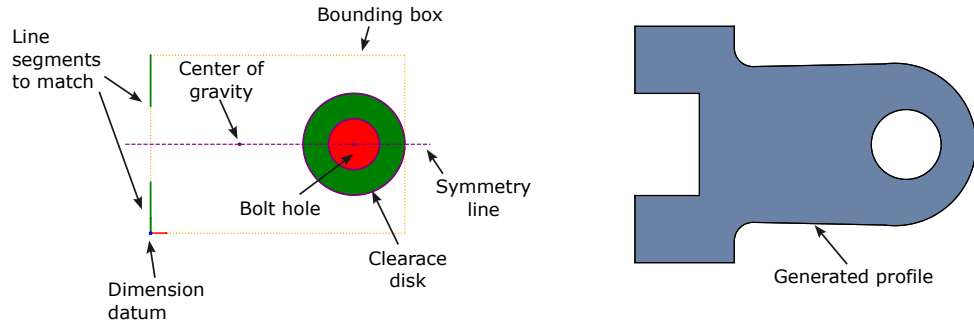


Figure 3: The geometric elements used in a geometric prompt and the resulting generated profile.

Table 5: Geometric prompt entities

Tokens	Explanation
StartOfPrompt	A special token denoting the start of the geometric prompt.
DimensionDatum	The dimension datum. The construction steps will build up the profile geometry relative to this fixed datum point.
BoundingBox	The desired bounding box for the profile being generated.
Area	The desired area of the profile being generated.
Complexity	The approximate number of curves we would like the generated profile to contain.
NumLoops	The desired number of loops for the generated profile.
SmoothVertices	The desired fraction of vertices which we would like to be tangent continuous.
CenterOfGravity Point	The desired center of gravity of the profile.
SymmetryLines ListOf(Infile)	Lines defining the mirror symmetries we would like the generated profile to have.
BoundLines ListOf(BoundLine)	A list of line segments which the the generated profile should include.
BoltHoles ListOf(BoltHole)	A list of bolt holes which should be included in the generated profile.
EndOfPrompt	A special token denoting the end of the geometric prompt.

A.3 CONSTRUCTION STEPS

The construction steps are introduced in Section 3.2 and a small number of examples given. The remaining construction steps are listed in Table 6. Each construction step contains an special token describing the step type, then a series of input and output geometry, encoded as described in Tables 3 and 4.

The list of construction steps starts with a special token `StartOfConstruction` and ends with token `EndOfConstruction`. Prior to construction steps which make use of parameter values, a `UseParameterN` parameter index token is provided in the sequence as described in Appendix A.4. As soon as sufficient geometry has been created to produce a curve in the final profile, the a `CreatedCurve` token is added, followed by the `BoundLine`, `Arc` or `Circle` tokens as described in Table 4. We found the introduction of these tokens to be critical to performance of the network. Without them the profiles predicted at the end of the sequence often contained self-intersection. The complete construction sequences for a number of generations are shown in full in Appendix E.

A.4 PARAMETERS

The construction sequences reduce the degrees of freedom in the generated profile to a small number of parameters which can be used for interactive shape editing. The values of these parameters are predicted by the model, but can be adjusted by the designer and the sequences replayed with floating point precision. Before any parameter value is used in a construction step, a special `UseParameterN` tag token is included indicating the index of the parameter to be used. There are 32 separate tokens, `UseParameter0`, `UseParameter1`, ..., allowing up to 32 separate param-

eters to be defined. After this token the parameter value is given. This will either be a `Length` or `Angle` token indicating the parameter value.

A.5 PROFILE GEOMETRY

The sequences finish with the profile geometry, which is encoded using a domain specific language similar to Wu et al. (2021); Xu et al. (2022). In loops consisting of multiple arc and line segments, the start point of each curve is the end point of the previous curve. Lines can then be defined by their end points and arcs represented by mid points and end points. A special token indicates the start of a new loop. As circles are closed curves, they are always in separate loops and are represented by a center point and radius. The tokenization for the profile geometry is shown in Table 7.

Table 6: Examples of construction steps continued

Description	Explanation	Example
CircleReverseCircle Input: circle ₁ Output: circle ₂	Given an oriented circle, return the same circle with the opposite orientation.	
CirclePointPointArc Input: circle, point _{start} , point _{end} Output: point _{arc}	Given a directed circle and a start and end point on the circle. Find the mid point of the arc which would trim the circle to the span which starts at the start point and follows the circles direction to the end point.	
LineDatumParallelLine Input: line ₁ , point _{datum} , Output: line ₂	Given a line and a datum point, find and return the line running parallel to the given line passing through the datum point.	
LineLineFillet Input: line ₁ , line ₂ , radius Output: arc	Given two directed lines and a radius, find and return the fillet arc which can be created along with its ordered start and end point.	
LineCircleParallelLine Input: line ₁ , circle ₁ Output: line ₂	Given an oriented line and a circle, compute the line parallel to the original that is tangent to the circle. There are exactly two such lines, one on each side of the circle, the orientation of the input line determines which one is selected.	
LineSymLineLine Input: line ₁ , line _{sym} Output: line ₂	Given a line and a symmetry line, find and return the image of the line reflected across the symmetry line.	
PointLineSymPointStep Input: point ₁ , line _{sym} Output: point ₂	Given a point and a symmetry line, find and return the image of the point reflected across the symmetry line.	
LineReverseLine Input: line ₁ Output: line ₂	Given a directed line, return it with reversed direction.	
LineAxisRotatedLine Input: line ₁ , angle Output: line ₂	Given a line a point and an angle, construct and return a line rotated by an angle in either the clockwise or counter-clockwise direction.	
PointRadiusCircle Input: point, radius Output: circle	Given a point and a radius, create and return a circle.	

Table 7: The tokenization of the final profile. As profiles contain only closed loops, the end point of the previous curve can be used as the start point of the current curve

Tokens	Explanation
StartOfProfile	A special token indicating the start of the profile
PolyArcLineLoop	Indicates the start of a loop containing multiple lines and arcs
SingleCircleLoop Point // <i>Center point</i> Length // <i>Radius</i>	Indicates a loop comprising a single circle
ProfileLine Point // <i>End point</i>	A profile line defined by it's end point. The end point of the previous curve is used as this line's start point.
ProfileArc Point // <i>Mid point</i> Point // <i>End point</i>	A profile arc defined by it's mid point and end point. The end point of the previous curve is used as this arcs's start point.
EndOfProfile	A special token indicating the end of the profile data

B EXTRACTING CONSTRUCTION SEQUENCES FROM PROFILE GEOMETRY

B.1 PRE-PROCESSING

Before the extraction of the construction sequences starts, the profiles are normalized so they fit into a unit square centered on the origin. This allows a consistent set of tolerances to be used in the downstream processing. A length tolerance is defined as $1/127$ model units, matching the size of the quantization bins used for tokenization. Points are considered coincident if the distance between them is smaller than this tolerance. Vectors are considered parallel when they deviate by less than 1 degree.

The following pre-processing is then applied.

- Curves shorter than the length tolerance are removed. The end points of adjacent curves are moved to close the gaps.
- Collinear line segments are merged
- The curves in each loop are lexicographically sorted, based on the end point of line and arc segments.
- Loops are ordered with the outer loop first and inner loops lexicographically sorted. For multi-curve loops the sort is conducted using the end point from the first curve. For circles, the bottom point is used.

B.2 EXTRACTING GEOMETRIC PROMPTS

Rather than training our model to auto-complete profiles from a random subset of curves as in Seff et al. (2021), we select the geometry to include in our geometric prompts based on common design requirements. The geometric information extracted and the reasoning for extracting it is described below.

Line segments: When extrusions are combined to build a 3d model, they often align with one another at planar faces. Consequently when defining profile geometry, it's useful to be able to specify line segments which will become a part of the convex hull of the generated profile. To construct the geometric prompt, we begin by identifying all line segments on the convex hull. From these, we select a subset, aiming to preserve groups of symmetrical segments. Groups that can be removed without reducing the convex hull of the line set are then discarded at random.

Bolt holes: Bolts are commonly used to connect mechanical components together. As such, it is often necessary for mechanical components to contain bolt holes at specific locations. We identify bolt holes in profiles as circular inner loops. These are specified by the hole center, a radius representing the size of the hole and a clearance disk which defines the required strength of the part around the bolt. Bolts can fail by tearing out of the bolt hole if the amount of material around them is insufficient to withstand transverse forces, hence the size of the clearance around bolts is an important factor designers want to control.

Profile area: This allows the thickness of the generated profile to be controlled, allowing the physical strength of the part to be balanced against weight and material utilization.

Fraction of tangent continuous vertices: Allowing designers to control the proportion of tangent-continuous vertices lets them decide whether the generated shape will have sharp or rounded corners. Tangent continuous vertices can be easily detected by comparing the tangents at the ends of adjacent curves.

Symmetry lines: Many mechanical parts exhibit symmetries and the ability to enforce symmetry is required by designers. Symmetries are identified using a 2d implementation of Li et al. (2015) and incorporated into the geometric prompt as infinite lines.

Bounding box: Parts will be generated in a unit square and then scaled and positioned when imported into the CAD application. An intuitive way for designers to specify the size and aspect ratio for the generated profile is using an oriented bounding box in the coordinate system of the CAD software. This can be mapped to a 2d bounding box which fits snugly into the unit square. The

generated profile can then be transformed and scaled back into the CAD systems coordinates. The generative model can also exploit the aspect ratio of the bounding box as an additional shape control.

Center of gravity: In cases where the desired profile covers only a small fraction of the bounding box area, and is poorly defined by other geometry in the prompt, the center of gravity can be used to guide the shape. This control is especially useful when the shape needs to avoid regions which would lead to clashes in the assembly context.

Profile complexity: It's often desirable to have control over the complexity of the profile. Generative algorithms can create unwanted additional detail when the complexity is too high, and can give very simple shapes when the complexity is too low. The number of edges in the profile is used as a complexity measure in the geometric prompt.

Number of internal loops: Designers often want to create designs which are both strong and lightweight. By providing control over the number of internal loops in the profile, in conjunction with the profile area, we can encourage the model to produce additional internal loops to lightweight the shape.

Dimensioning datum: Designers often want to parameterize profiles based on some fixed geometry. In parametric CAD this is often the fixed origin of the coordinate system or a reference point created by a previous modeling feature. To allow designers control over this fixed geometry, we include a datum point in the geometric prompt. The datum point used is randomly selected from the following:

- A corner of the profile's bounding box
- A mid point of one of the bounding box edges
- The center of the bounding box
- The center of the largest circle in the profile

Some examples of how the geometric prompt controls the shape can be seen in Appendix D

B.3 CONSTRUCTION SEQUENCE EXTRACTION

An overview of the construction sequence extraction algorithm was given in Section 3.1. Here we give more detail on each phase of the algorithm.

B.3.1 ANALYSIS PHASE

The profile analysis phase is similar to the first step of a heuristic auto-constrainer. The geometry of the profile curves are analyzed and the following groups of geometric entities are detected

- Groups of collinear points
- Groups of parallel lines
- Fillet arcs and their adjacent curves
- Symmetry lines and the points and curves placed symmetrically about them
- Groups of concentric circles or arcs

B.3.2 IDENTIFICATION OF FREQUENTLY USED DISTANCES

The following distances between pairs curves are recorded

- The distance between all pairs of parallel lines
- The difference in radius between all pairs of concentric circles
- The radii of all circles and arcs

The frequency with which these values are found is then counted. Length parameters are created preferentially for length values which appear more frequently.

B.3.3 IDENTIFICATION OF SOURCE LINES

The construction sequences are designed to start with geometry defined in the prompt and, step by step, construct the geometry needed to build the profile. To enable this we identify the following “source lines”, from which subsequent construction lines will be derived by a series of offsets.

- Symmetry lines.
- Infinite lines derived from all line segments provided in the prompt.
- A horizontal and a vertical line passing through the dimension datum.
- Any group of parallel lines which is not parallel or anti-parallel to one of the source lines identified so far must then be accounted for. A `LineAxisRotatedLine` step is created which rotates the closest coordinate system axis line around the dimension datum to produce a new source line parallel to the lines in the group.

B.3.4 IDENTIFICATION OF OFFSETS BETWEEN PARALLEL LINES

Given a set of directed parallel lines, it is possible to construct any line from any other given the appropriate signed offset distance. Groups of parallel lines are processed individually to discover chains of offsetting operations which best construct the group.

Pairs of lines which can be constructed from a symmetric offset of a symmetry line are identified and these symmetric constructions are used preferentially.

The algorithm then tries to describe every line in the remaining group by a sequence of offsets starting from the “source lines”. An undirected fully connected graph is constructed with lines as the nodes and the distances between lines as the edges. Edges are then deleted from this graph, starting with the edges corresponding to the least frequent distances identified above. Edges cannot be removed if doing so would make the graph disjoint. Finally, the order in which lines are created is defined. We start with the source lines and traverse the graph, following the Dijkstra shortest path algorithm. This traversal defines which lines are the input and output of `LineOffsetLine` steps.

B.3.5 IDENTIFICATION OF OFFSETS BETWEEN CIRCLES

Similar to the algorithm for parallel lines described above, offsetting operations between concentric circles/arcs are created based on the most frequent length values in the entire profile. Each group of concentric circles includes at least one operation which creates the initial circle from a center point and radius.

B.3.6 CREATING THE CONSTRUCTION STEPS

Next an extensive list of construction steps, which encompasses all the relationships identified between geometric entities is created. The criteria for creating the steps is as follows

LineXLine: In places where a vertex of the profile is coincident with the intersection of two lines, a `LineXLine` step is created. The directions of the input lines always follows the directions of the corresponding line segments in the profile.

LineLineFilletStep: In places where a fillet arc exists between two line segments, a `LineLineFillet` step is created

PointLineSymPoint: In places where a vertex in the profile can be defined by mirroring another vertex across a symmetry line, a pair `PointLineSymPoint` step are created. The decision as to which point is the original and which point will be mirrored is made later in the graph simplification phase. At this stage two operations are created using the two different points as the input.

LineOffsetLine: Line offsetting steps can be created for each line and offset found by the procedure described in Appendix B.3.4

LineDatumParallelLineStep: In cases where a line is require, which is parallel to a “source line” and passes thorough one of the end points of a line segment given in the prompt, a `LineDatumParallelLineStep` is created. As these steps do not introduce new parameter values they will be used in preference to `LineOffsetLine` steps.

LineCircleParallelLine: A list of circles is built which includes the clearance disks of all prompt holes along with any circles or arcs in the profile. Lines in the profile which touch these circles tangentially are then found. Steps are created which offset a source line to a position which touches the circle. There are two solutions for tangentially touching lines and the direction of the initial line is used to select between them. As above, these steps do not introduce new parameter values, causing them to be selected preferentially.

CirclePointPointArc: For each arc in the profile, which is not defined by a fillet, a step is created which finds the arc's mid-point from the underlying circle and the start and end points of the arc. A direction flag indicates whether the clockwise or counter clockwise arc is selected. The introduction of these steps allow arcs to be derived from circles provided the end points are known.

LineXCicle: For each profile vertex which is at a non-tangential intersection between a line and an arc, a LineXCicle step is created.

B.3.7 GRAPH COMPLETION

The construction steps are then used to build a bipartite dataflow graph with geometry and construction steps as nodes and edges representing the flow of geometry into and out of the operations. The directed graph is then searched for geometry nodes without incoming edges. Geometry introduced by the prompt is expected not to have incoming edges, while other geometry without incoming edges is problematic. New construction steps need to be added to the graph to allow this geometry to be constructed from the prompt. In many cases lines or circles can be created from existing geometry by introducing LineReverseLine or CircleReverseCircle steps. Circles can also be created using PointRadiusCircle steps provided the center point has been constructed. Graphs which contain geometry without incoming edges following the graph pruning phase will be discarded.

B.3.8 PARAMETERS

The LineOffsetLine, CircleOffsetCircle, PointRadiusCircle, LineLineFilletStep steps require input length parameters and the LineAxisRotatedLine steps requires an input angle parameter. The required parameters are identified and special geometry nodes are created to represent their values. A single parameter can be an argument to multiple construction steps, for example, it's common for multiple fillets to have the same radius.

B.3.9 CYCLE BREAKING

The initial dataflow graph will contain cycles, which must be broken before processing continues. Cycles in the graph are detected by a depth first search algorithm and processed one at a time. The list of edges forming each cycle is found and the edge to remove is chosen based on the following criteria:

Essential edges: Starting from the geometry required to define the final profile we examine the incoming edges. If only one edge is incoming then this geometry is derived from the output of a single construction step. The edge must then be added to an "essential edges" set. We then move to the construction step which gave rise to these each of the essential edges and repeat this procedure. The entire graph is traversed and the complete set of essential edges found. The traversal stops when we reach a geometry node with more than one incoming edge. We cannot remove any essential edges from the graph.

Hops from prompt: We want to find the shortest possible construction sequences for each piece of geometry. Starting at the prompt geometry we use Dijkstra's algorithm to find the smallest number of hops to each node in the graph. As the graph contains cycle, care must be taken to avoid visiting nodes more than once.

Construction step priorities: Construction steps have the following priorities.

- Creating arcs as fillets is highest priority
- Next creating lines which are tangent to circles

- Next creating points from line circle intersections
- Then operations mirroring lines or points
- Finally all other operations

The cycles are then broken by removing one edge from each cycle. Essential edges are never removed. The non-essential edges which have nodes with the largest number of hops from prompt geometry are removed in preference. The priorities of the operations at the tail end of each edge is then used as a tie breaker.

B.3.10 GRAPH PRUNING

Geometry nodes with more than one incoming edge are equivalent to over-constrained but consistent geometry in a traditional constraint solver. The graph pruning algorithm selects one incoming edge for each geometry and removes branches of the graph which lead to other solutions. The choice is made based on a cost function which is evaluated for each edge in the graph.

- Geometry provided in the prompt has a cost of zero
- Construction steps add a cost of 0.1 to all output edges
- Parameters incur a cost of 1.0.
- Any geometry which is not defined by the prompt and has no predecessor nodes is given an infinite cost

Once the costs for the edges have been computed, we loop over the geometry nodes. For geometry node with more than one incoming edge, the edge with the smallest cost is kept and all others will be removed. Branches of the graph which do not end at profile geometry are then removed. Profiles are discarded from the dataset if any geometry required to build the final shape could not be constructed from the geometry in the prompt.

B.3.11 SEQUENCE ORDERING

The dataflow graph now contains all the information required to create the construction sequence. The nodes in the graph are first ordered using a lexicographical topological sort. This respects the dependencies between geometry and construction steps defined by the graph while allowing ambiguity in the ordering to be resolved by secondary criteria. The order in which curves appear in the final profile is the primary criteria used and a lexicographical ordering of the intermediate geometry is used to break ties. Infinite lines are ordered based on the lexicographically lowest point where they intersect the bounding box of the profile.

The parameters required to define the shape are then ordered based on the first construction step which utilizes them. Each parameter is allocated an index based on this order, which is used in the `UseParameterN` tokens described in Appendix A.4.

C ADDITIONAL RESULTS

C.1 RL ALGORITHMS

The hyper-parameters used for RL model alignment are detailed in Table 8. Training was performed on 3 NVIDIA A10 GPUs with the AdamW optimizer and a learning rate of 1×10^{-6} for one thousand steps. Figure 4 shows the training performance over steps for the different reinforcement learning algorithms used. Applying a KL penalty proved essential. In its absence, the model exploited the reward function by consistently generating invalid sequences. Since rewards for invalid sequences were masked, only generations for extremely simple prompts yielded nonzero rewards, leading to a trivially high average reward despite no meaningful learning.

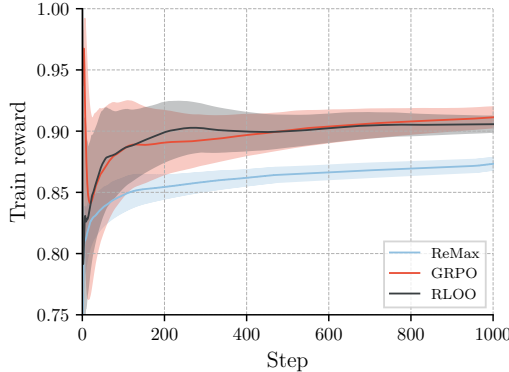


Figure 4: RL training rewards

Table 8: Hyperparameter comparison across different RL algorithms.

Hyperparameters	ReMax	GRPO	RLOO
Effective batch size	768	144	144
Group sample size	–	16	16
Learning rate	1e-6	1e-6	1e-6
Top-p sampling	0.3	0.3	0.3
Token-level KL penalty	0.1	–	0.1
Sequence-level KL penalty	–	0.01	–
Policy clipping ratio	–	0.2	–

C.2 CONSTRUCTION STEP PRECISION

As each construction step represents an atomic operation with well defined output geometry, we can measure the accuracy with which the transformer learns to perform the construction steps. These values are shown for all construction step models in Table 9. Distances are measured in the unit square in which the profile is constructed. Angles are measured in radians and comparisons between boolean of flags are 1.0 when the flags agree and 0.0 when they disagree. All values are averaged over the 6427 prompts in the test set.

We observe that the transformer can learn to exactly duplicate tokens with perfect accuracy. For example, when a circle is reversed, only the `IsCCW` flag changes and perfect agreement is observed for the center point and radius. The precision for construction steps with a single token as input and output is also very good, hence the good agreement in distance from the origin and angle when lines are reversed.

More complex geometric operations, with multiple curves as arguments have bigger errors. Examples of this are line-line and line-circle intersections (`LineXLine` and `LineXCircledist`) and the creation of the mid point of fillet arcs (`LineLineFilletprecision`).

Applying the reinforcement learning algorithms to the sequences has only a small effect. This implies the network primarily learns how to perform the geometric operations in supervised pre-training. The RL rewards then help the network to combine the individual operations into CAD programs which give rise to syntactically and geometrically valid profiles rather than improving its ability to perform the low level geometric computations.

Table 9: Comparison of construction step metrics among different models

Metrics	Construction steps model	Construction steps model (ReMax)	Construction steps model (GRPO)	Construction steps model (RLOO)
<code>CircleOffsetCircle_{center}</code> (↓)	0.0000	0.0000	0.0000	0.0000
<code>CircleOffsetCircle_{radius}</code> (↓)	0.0109	0.0107	0.0124	0.0133
<code>CirclePointPointArc_{dist}</code> (↓)	0.1413	0.1438	0.1413	0.1102
<code>CircleReverseCircle_{ccw}</code> (↑)	1.0000	1.0000	1.0000	1.0000
<code>CircleReverseCircle_{center}</code> (↓)	0.0000	0.0000	0.0000	0.0000
<code>CircleReverseCircle_{radius}</code> (↓)	0.0000	0.0000	0.0000	0.0000
<code>SolutionExists</code> (↑)	0.9981	0.9983	0.9987	0.9988
<code>LineAxisRotatedLine_{angle}</code> (↓)	0.0204	0.0242	0.0243	0.0226
<code>LineAxisRotatedLine_{dist}</code> (↓)	0.0130	0.0148	0.0132	0.0139
<code>LineCircleParallelLine_{angle}</code> (↓)	0.0017	0.0059	0.0020	0.0009
<code>LineCircleParallelLine_{dist}</code> (↓)	0.0239	0.0248	0.0236	0.0231
<code>LineDatumParallelLine_{angle}</code> (↓)	0.0020	0.0019	0.0022	0.0017
<code>LineDatumParallelLine_{dist}</code> (↓)	0.0061	0.0063	0.0066	0.0066
<code>LineLineFillet_{precision}</code> (↓)	0.0347	0.0367	0.0362	0.0365
<code>LineLineFillet_{validcircle}</code> (↑)	0.9320	0.9372	0.9522	0.9504
<code>LineOffsetLine_{angle}</code> (↓)	0.0040	0.0040	0.0039	0.0039
<code>LineOffsetLine_{dist}</code> (↓)	0.0095	0.0097	0.0095	0.0094
<code>LineReverseLine_{angle}</code> (↓)	0.0015	0.0007	0.0006	0.0007
<code>LineReverseLine_{dist}</code> (↓)	0.0001	0.0001	0.0001	0.0001
<code>LineXCircle_{dist}</code> (↓)	0.0866	0.0870	0.0745	0.0787
<code>LineXLine</code> (↓)	0.0245	0.0203	0.0194	0.0184
<code>PointLineSymPoint</code> (↓)	0.0060	0.0066	0.0057	0.0049
<code>PointRadiusCircle_{center}</code> (↓)	0.0000	0.0000	0.0000	0.0000
<code>PointRadiusCircle_{radius}</code> (↓)	0.0000	0.0000	0.0000	0.0000
<code>SymlineOffsetLineLine_{angle}</code> (↓)	0.0003	0.0004	0.0003	0.0005
<code>SymlineOffsetLineLine_{dist}</code> (↓)	0.0055	0.0055	0.0056	0.0056

D SHAPE CONTROL VIA GEOMETRIC PROMPTING

Geometric prompts contain information which the designer can use to control the generated profile shape. Here we show some qualitative examples of shape control. In each of the figure in this section, a single value in the geometric prompt is varies and the other are held fixed at the values shown in Table 10. The positions of the line segments which the profile should match are shown in green, symmetry lines are marked with dashed purple lines and the center of gravity shown as a purple dot.

Table 10: Default properties used in the geometric prompt in this section.

Property	Value
Area	0.4
Symmetry	Vertical symmetry line
Tangent continuous vertex fraction	0.5
Number of edges	16
Center of gravity	0.0, 0.0
Top width	0.1

Designers often want control over the mechanical strength and weight of the parts being modeled. We observe that changing the requested area of the generated profile provides some control over thickness and consequently allows mechanical strength to be increased at the expense of increased mass. The generated shape with three different requested areas is shown in Figure 5.

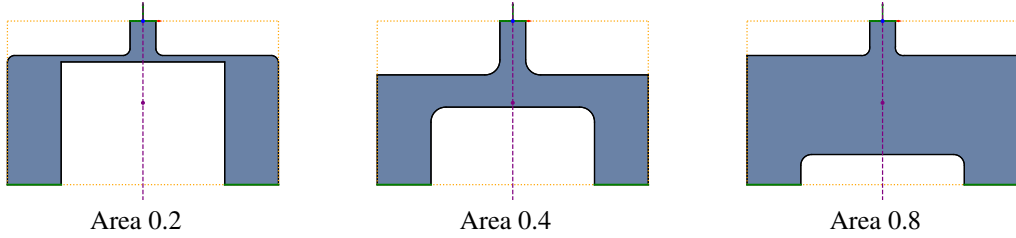


Figure 5: Profiles generated when three different profile areas are requested. When an area of 0.2 model units is requested, the generated profile contains thin regions. Increasing the requested area allows shapes without this thin region.

The ability to make designs symmetrical is also important for designers. In many cases, designers start by drawing symmetry lines and then use constraints to keep various profile curves symmetric around these lines. In Figure 6 we show how including horizontal and vertical symmetry lines in the geometric prompt effects the resulting generation. Notice that the bounded line segments which the profile is requested to match (green lines) exhibit symmetry consistent with a vertical symmetry line only. When a horizontal symmetry line is requested, the network is unable to match both the bounded lines and the requested symmetry. Figure 7 shows the detailed construction sequence when both horizontal and vertical symmetries are included. From this sequence we can see how symmetries in the final shape are derived by mirroring points over the symmetry lines provided.

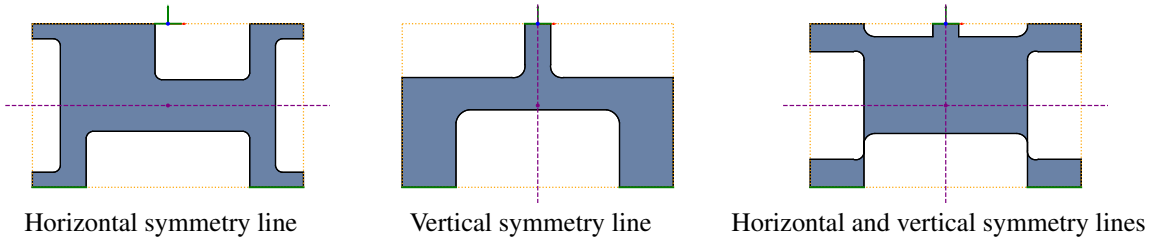


Figure 6: Requesting different symmetry lines in the geometric prompt.

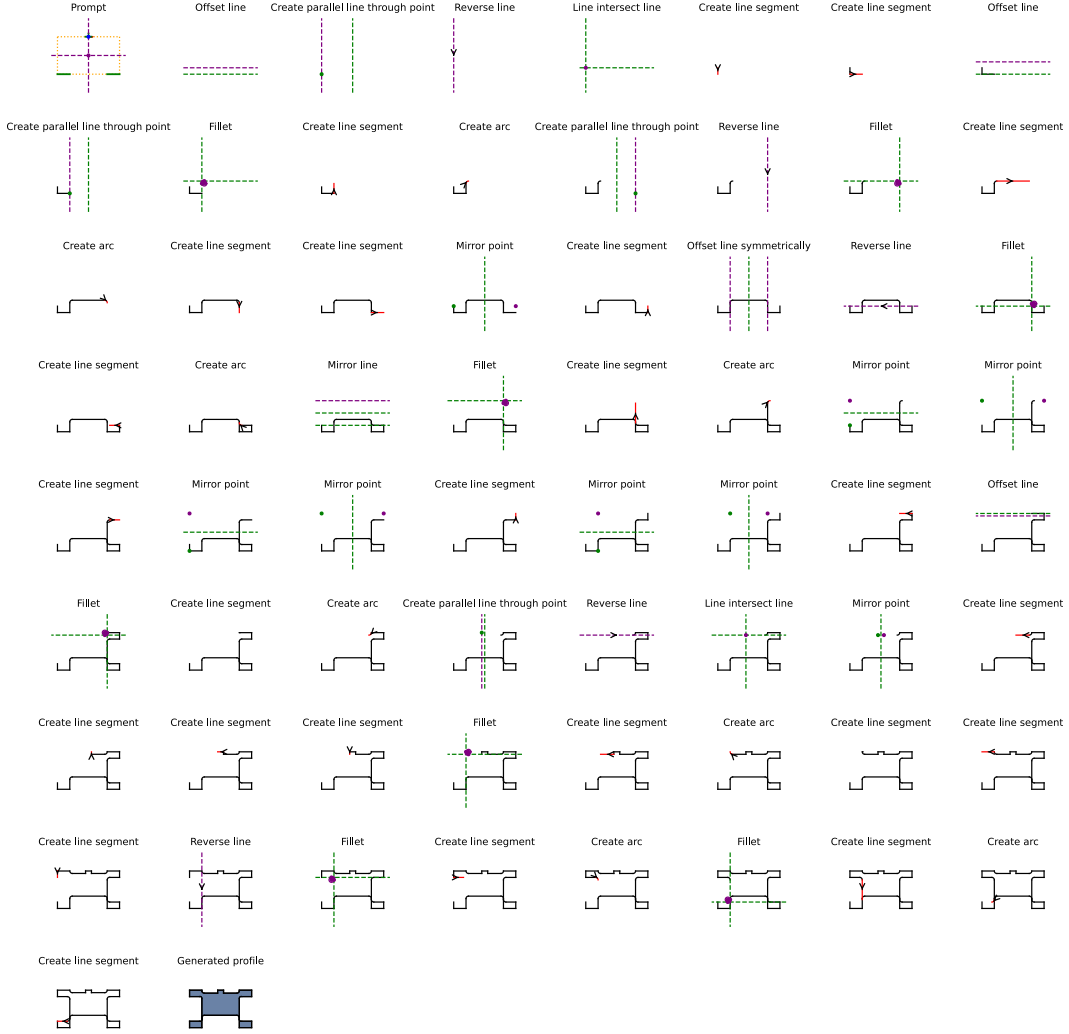


Figure 7: Construction sequence for the example with both horizontal and vertical lines. Points present in the prompt are mirrored over the two symmetry lines and used to build the final shape.

A useful way to provide some high level control over the style of the shape is by requesting sharp corners or tangent continuous vertices. In Figure 8 we see how increasing the requested fraction of tangent continuous vertices allows the shape to be varied from a rectilinear form (left) to the smoothly curved shape shown on the right.

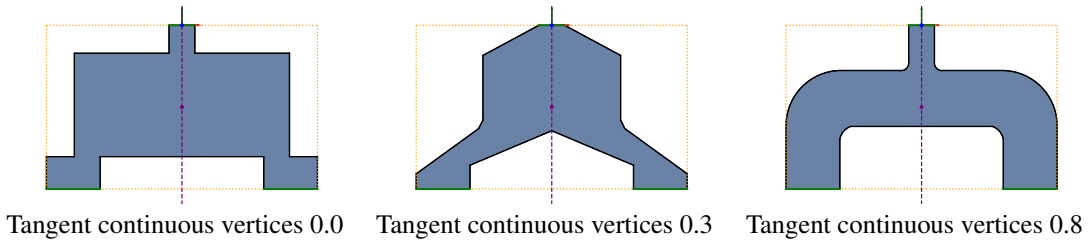


Figure 8: Profiles obtained by varying the requested fraction of tangent continuous vertices.

The complexity of the generated profile can be adjusted by specifying the number of edges in the geometric prompt. Figure 9 illustrates how increasing the number of edges produces more intricate

shapes. However, we find that when higher complexity is requested, the additional details are often poorly defined. In practice, reducing the requested complexity can help eliminate these unnecessary or ambiguous features.

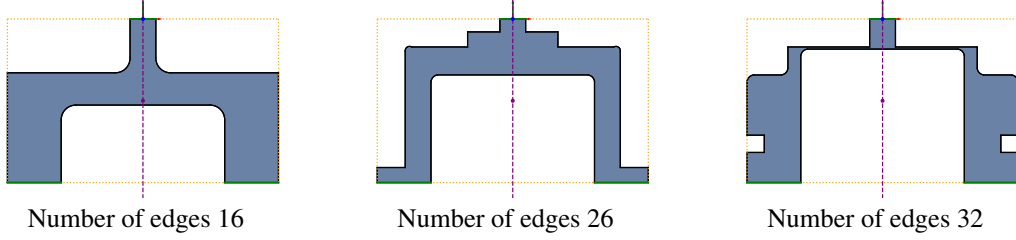


Figure 9: Controlling shape complexity by requesting different numbers of edges in the geometric prompt.

In cases where the desired shape is ambiguous, additional control can be exerted by specifying the desired center of gravity for the profile. In Figure 10 we show how the raising and lowering the center of gravity along the symmetry line changes the final shape.

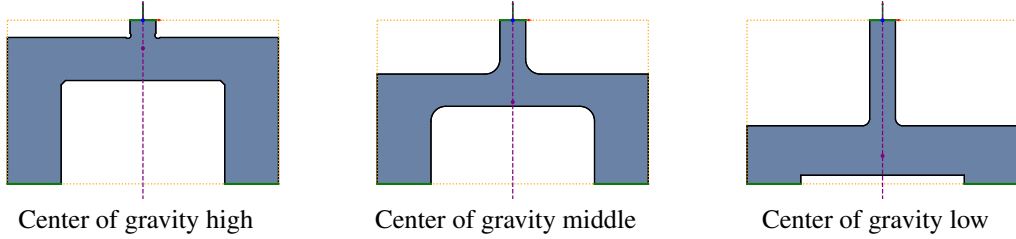


Figure 10: The effect of raising and lowering the requested center of gravity. The center of gravity is marked as a purple point.

In addition, we demonstrate that changing the bounded line segments provided in the prompt, while keeping all other properties at the values defined in Table 10, results in sensible shape changes.

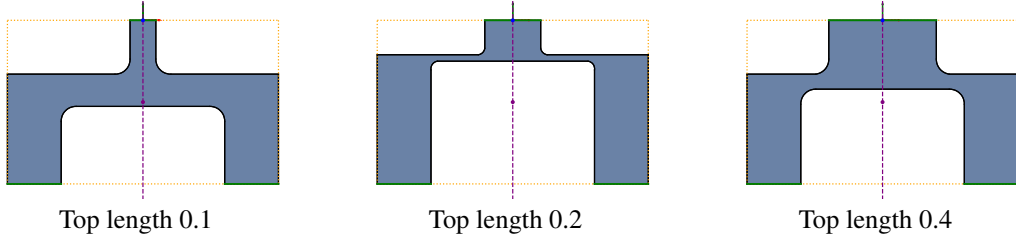


Figure 11: The effect of varying the length of the bounded line segment (green) at the top of the figure. As this length is changed, the top part of the profile widens in a natural way.

E GENERATED CONSTRUCTION SEQUENCE EXAMPLES

In this section we show some additional construction sequences generated by the model and relate these to the parametric behavior of the shape when the driving parameters are varied.

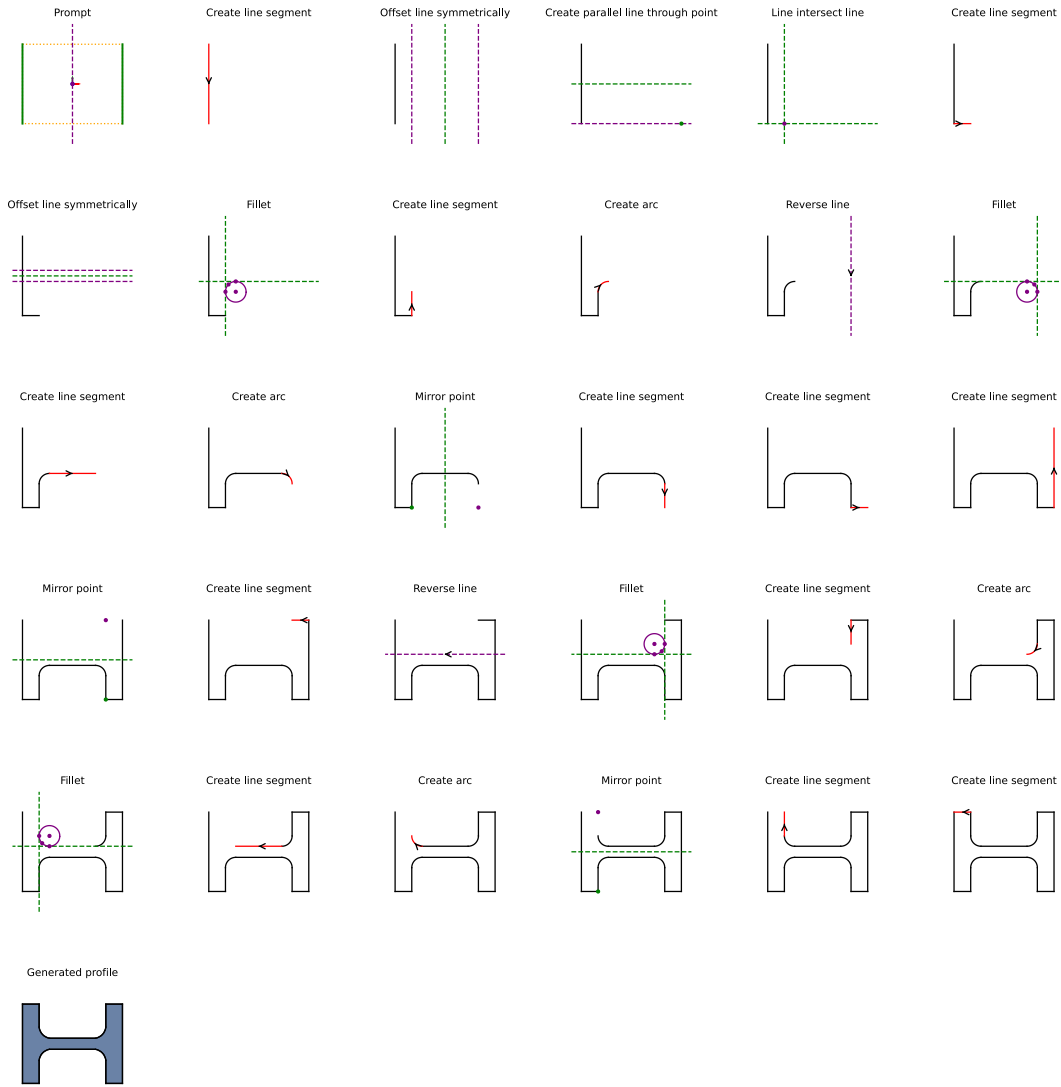


Figure 12: Construction sequence for an I-beam shape. Notice how the vertical symmetry line is defined, but the central position of the dimension datum causes a horizontal symmetry line to be implied.

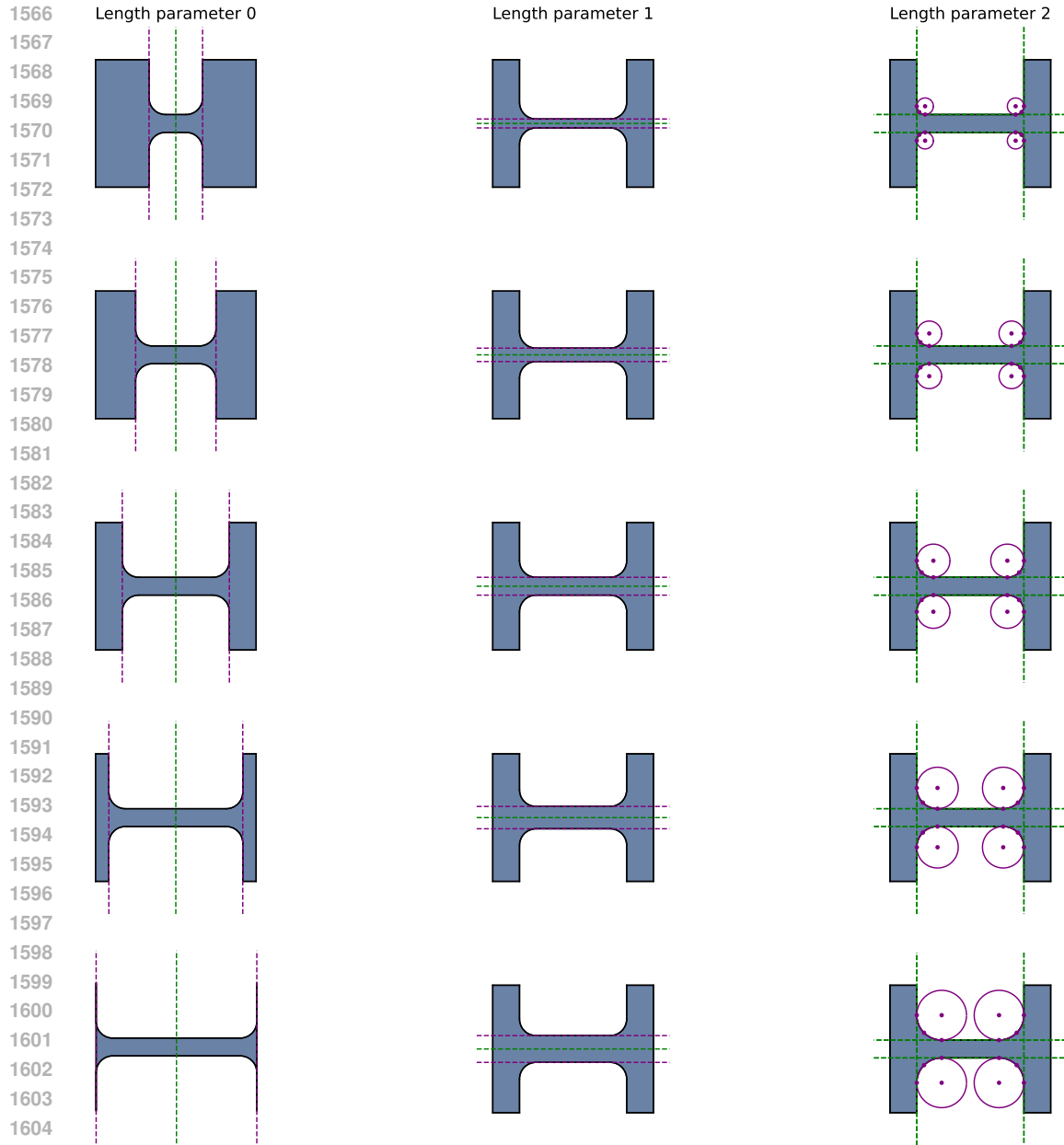


Figure 13: Parametric variations for the I-beam. The profile is parameterized with three values which allow control over the width of the two side bars, the width of the central beam and the radii of the four fillet arcs.

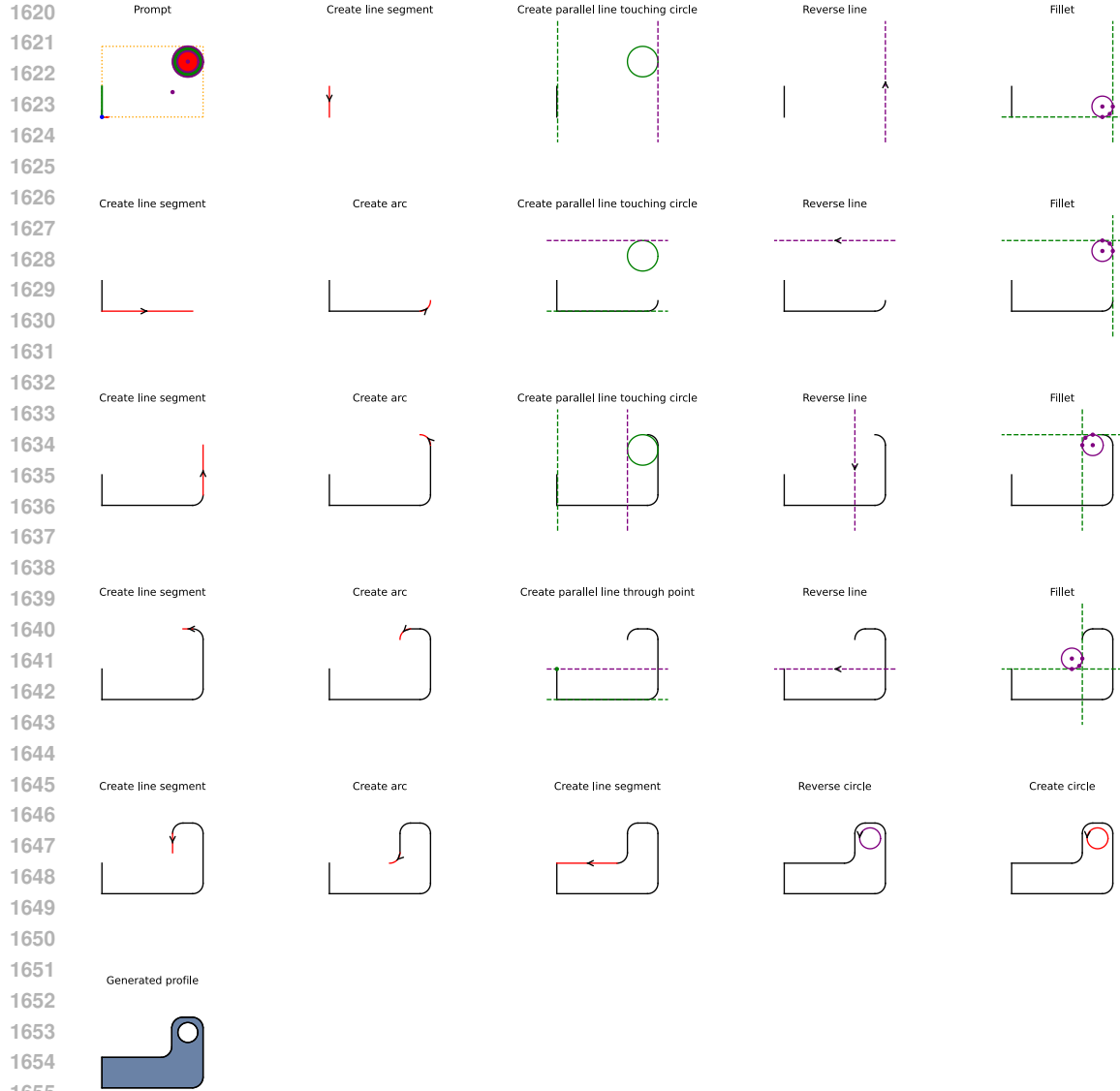


Figure 14: An example of a construction sequence with no free parameters. Notice how the vertical lines are created by offsetting the axis lines until they touch the clearance disk. The horizontal lines are derived from the end points of the line segment provided in the prompt and the dimension datum.

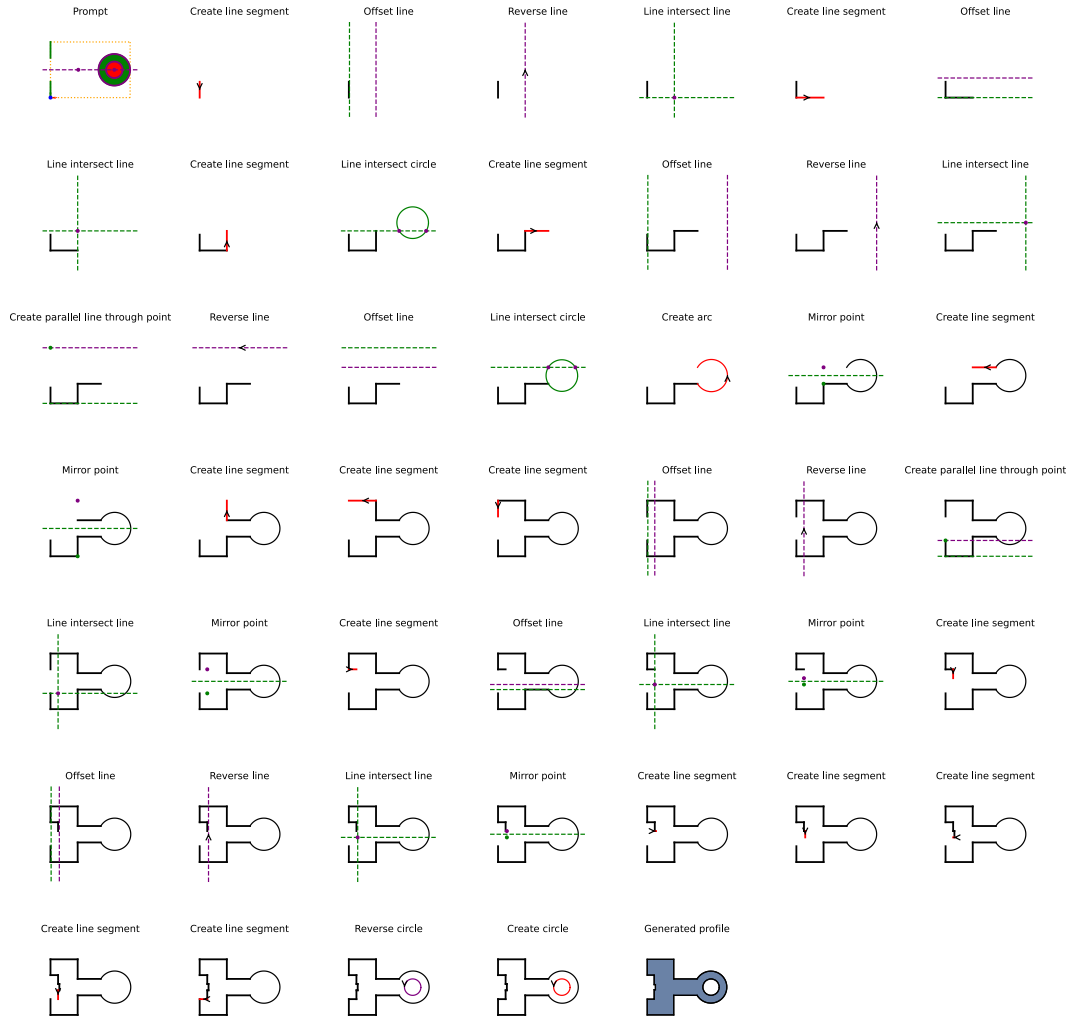


Figure 15: A construction sequence for a more complex shape.

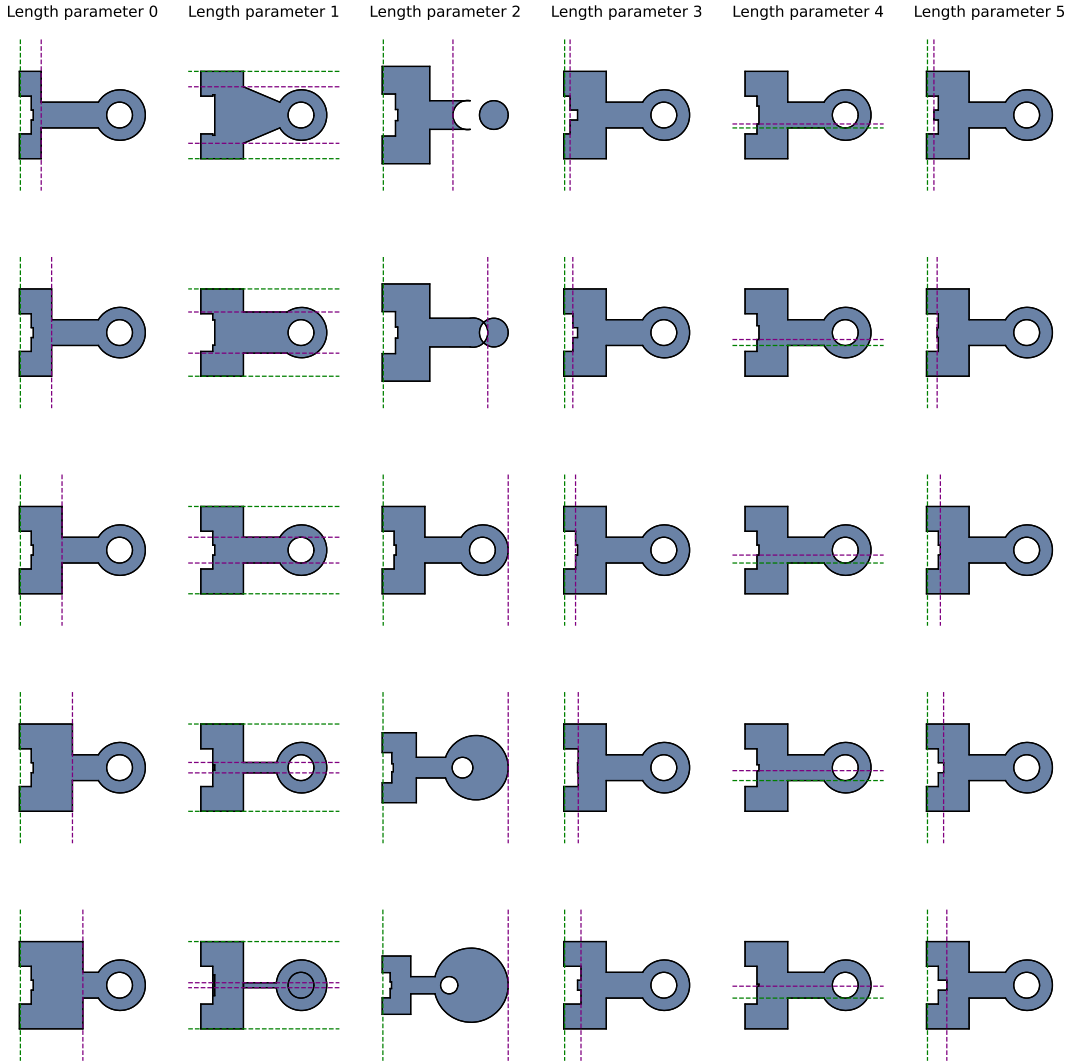


Figure 16: Parametric variations for the complex shape. We see that the parameterization is sensible in most cases, however this example does show two failure modes. When parameter 1 causes the line segments to move to a position where they fail to intersect with the circle then this operation fails. The points are not updated causing the introduction of sloped lines. Parameter 2, defines a line offset which is intended to be at the right hand side of the shape. This line is intersected with the center line to build the mid point of the outer circular arc. When this line is moved, the outer arc and inner circle are no longer concentric, revealing a problem with the parameterization. Techniques which improve the parametric behavior of the resulting profiles are left to further work.