DeepMind

Computer-Aided Design as Language

Yaroslav Ganin

In collaboration with **Sergey Bartunov**, **Yujia Li**, **Ethan Keller** (Onshape), **Stefano Saliceti**

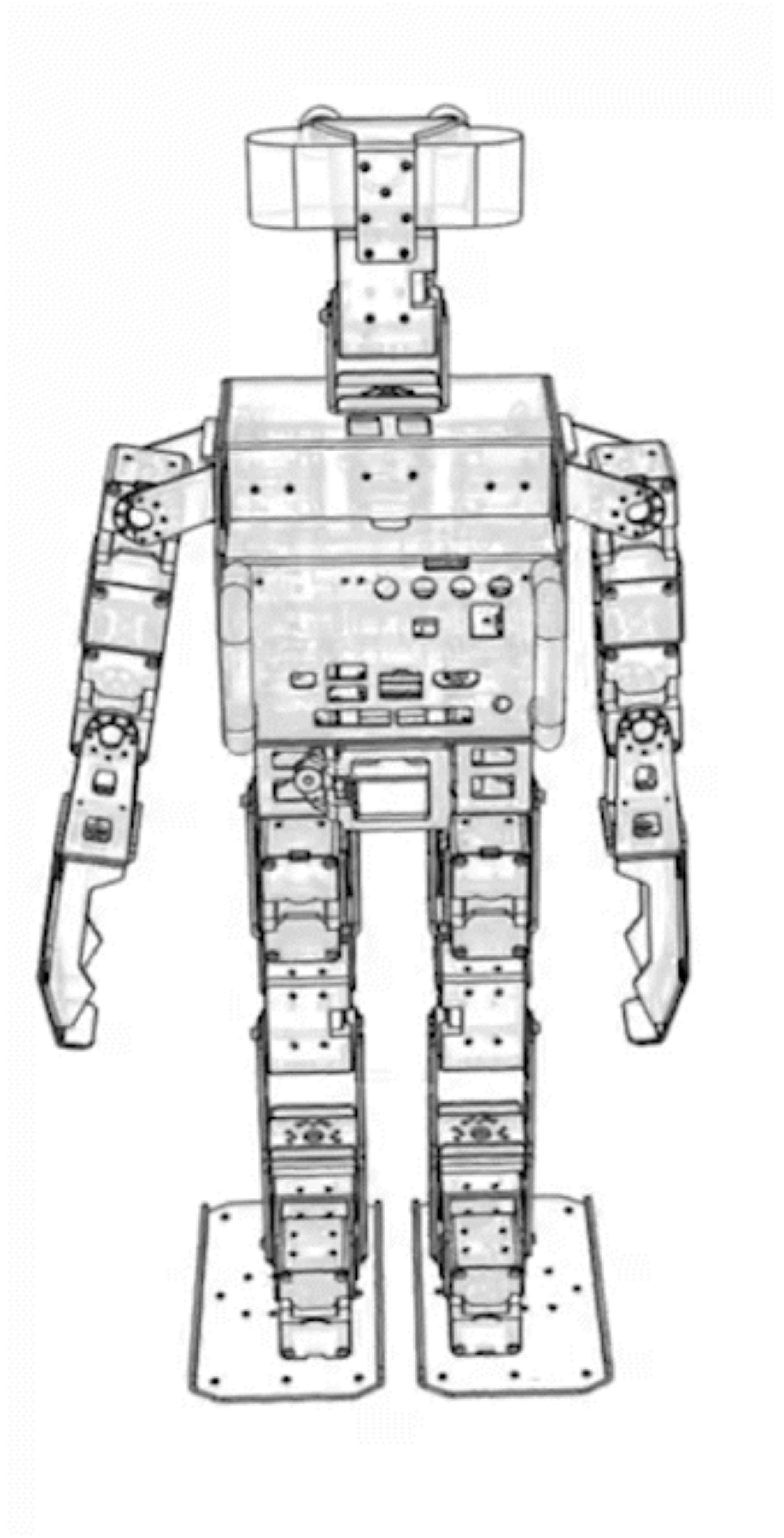
NeurIPS 2021

Skykettle

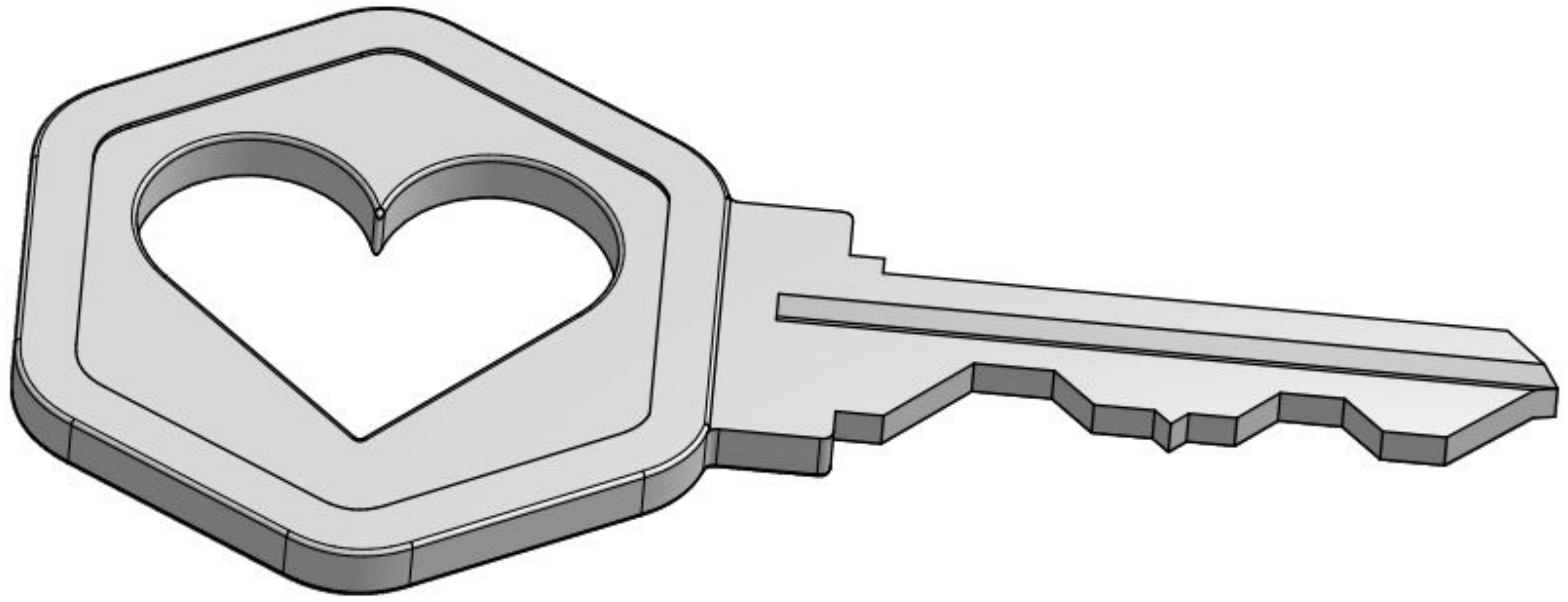
What and why

- **CAD is everywhere:** cars, coffee mugs, robots, cell phones
- CAD engineering is **hard** — mastering the craft requires **years of experience** and training; CAD software is constantly improving but the **learning curve** is still **steep**
- An ML-powered **generative model** would have a lot of potential to **make** the engineer's **life much easier** and boost their productivity

  Other examples: reward-driven design; priors for man-made objects
- It's an **interesting challenge** —
  the domain is **complex** and **underexplored**

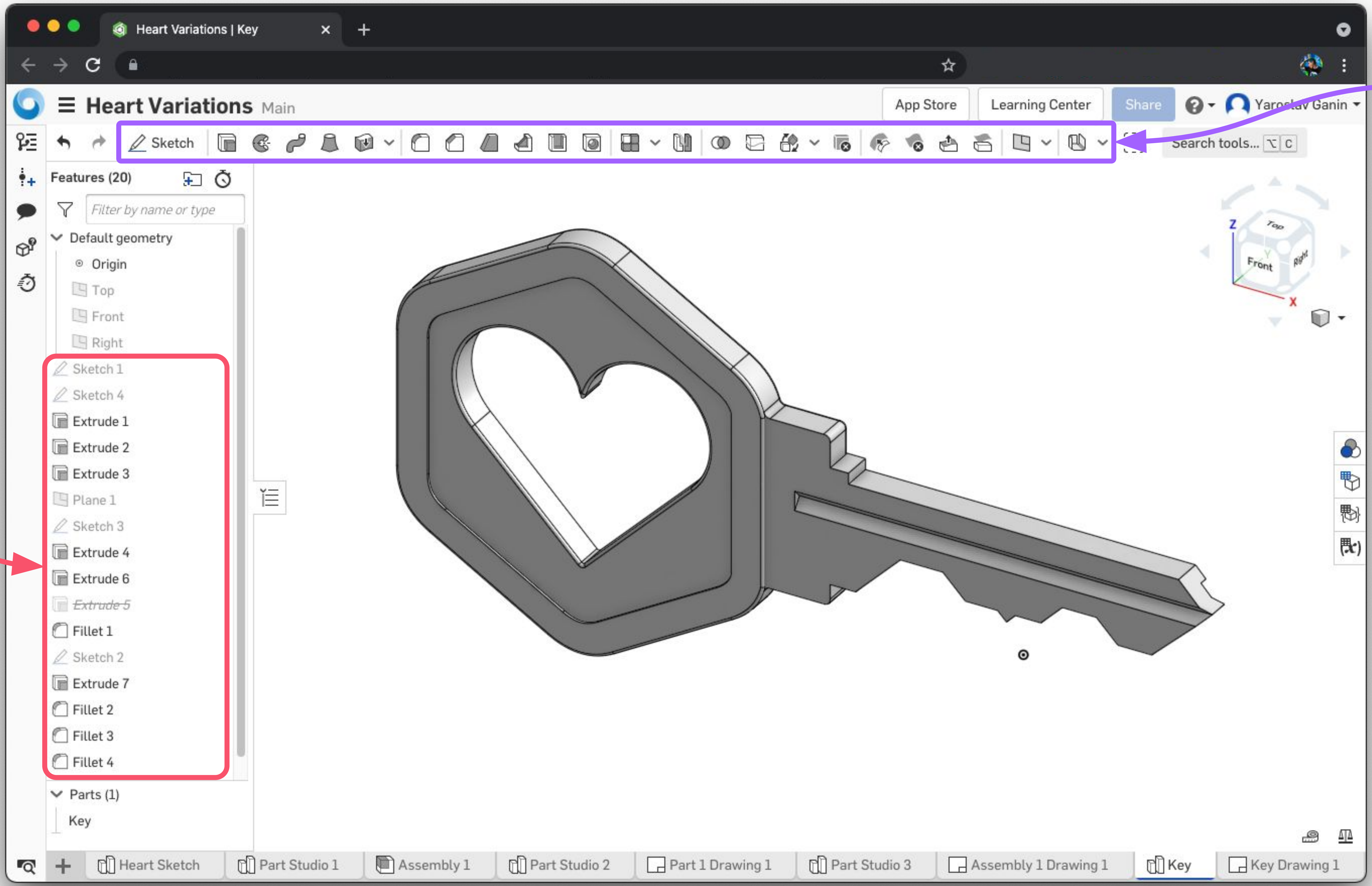


Anatomy of CAD



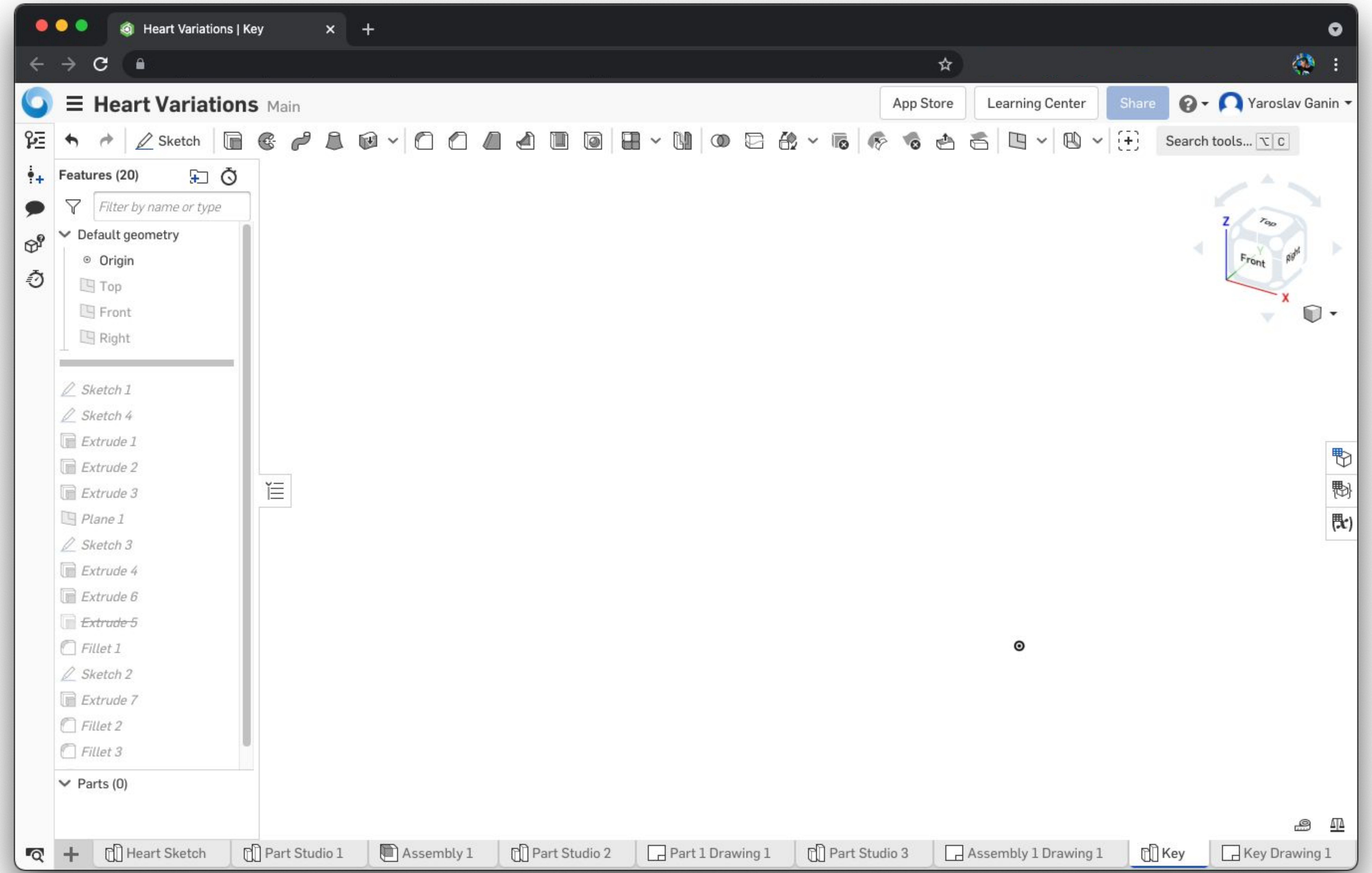
Anatomy of CAD

Sequence of modeling **actions** (features)

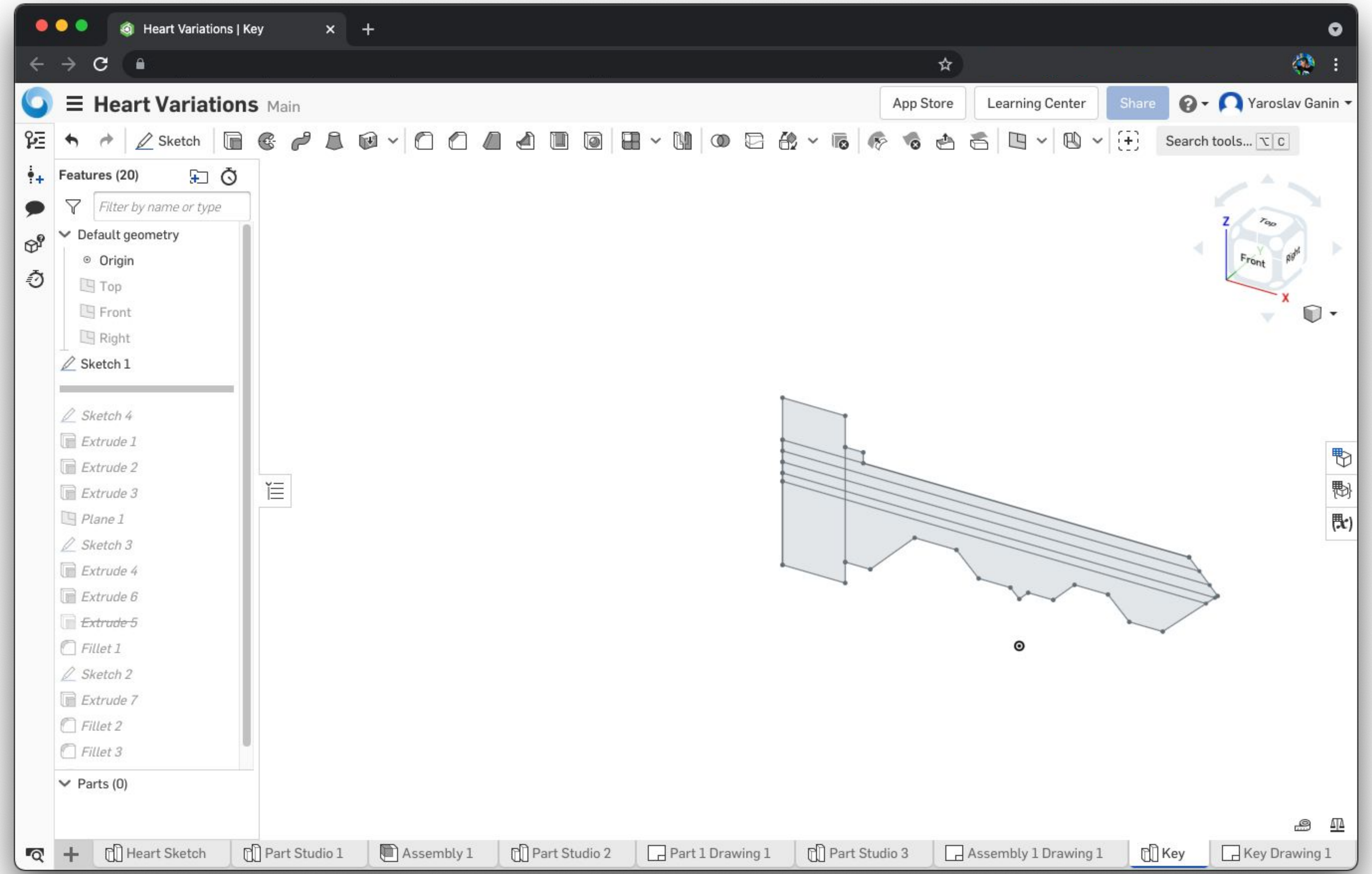


Available actions

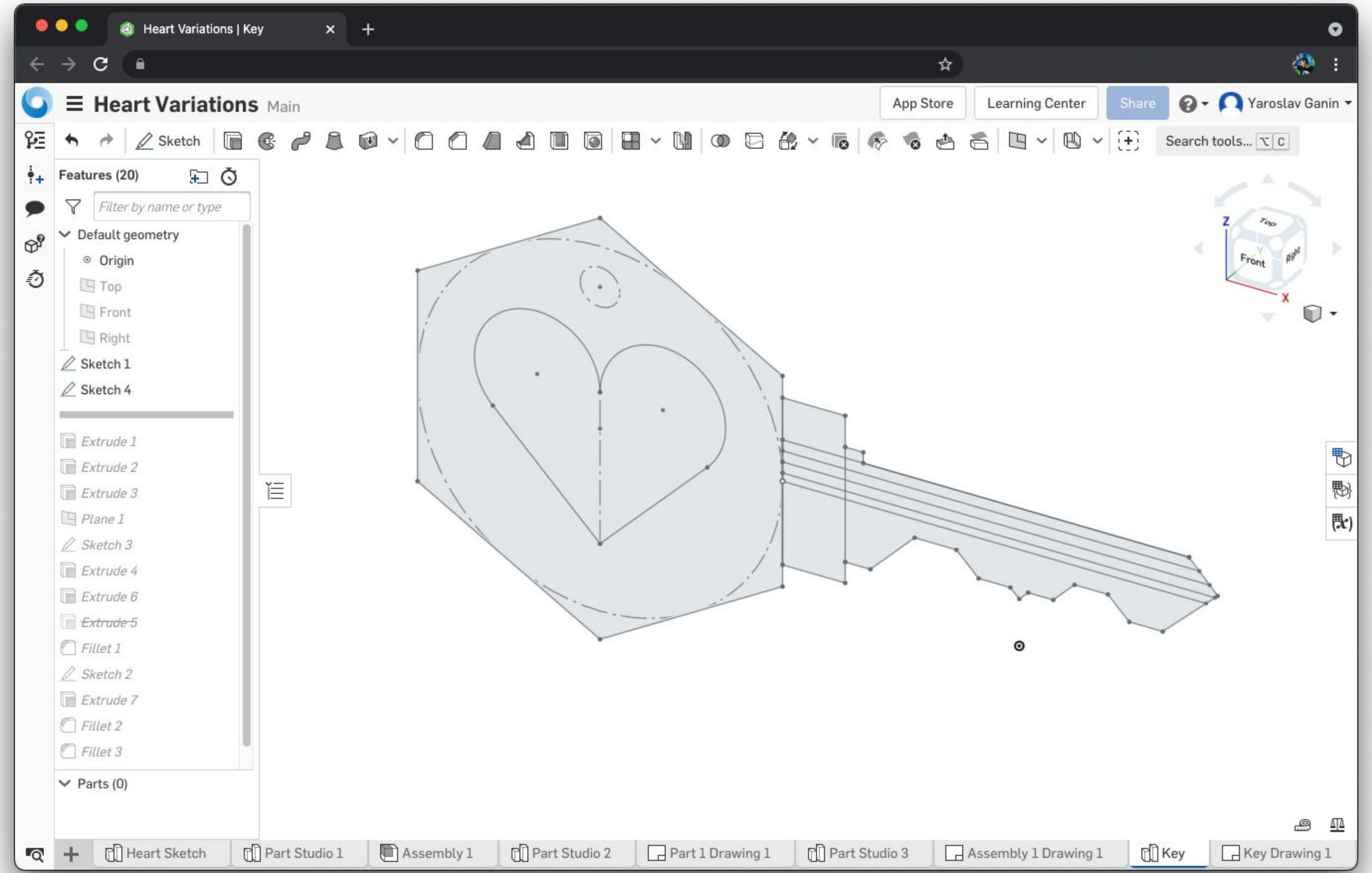
Anatomy of CAD



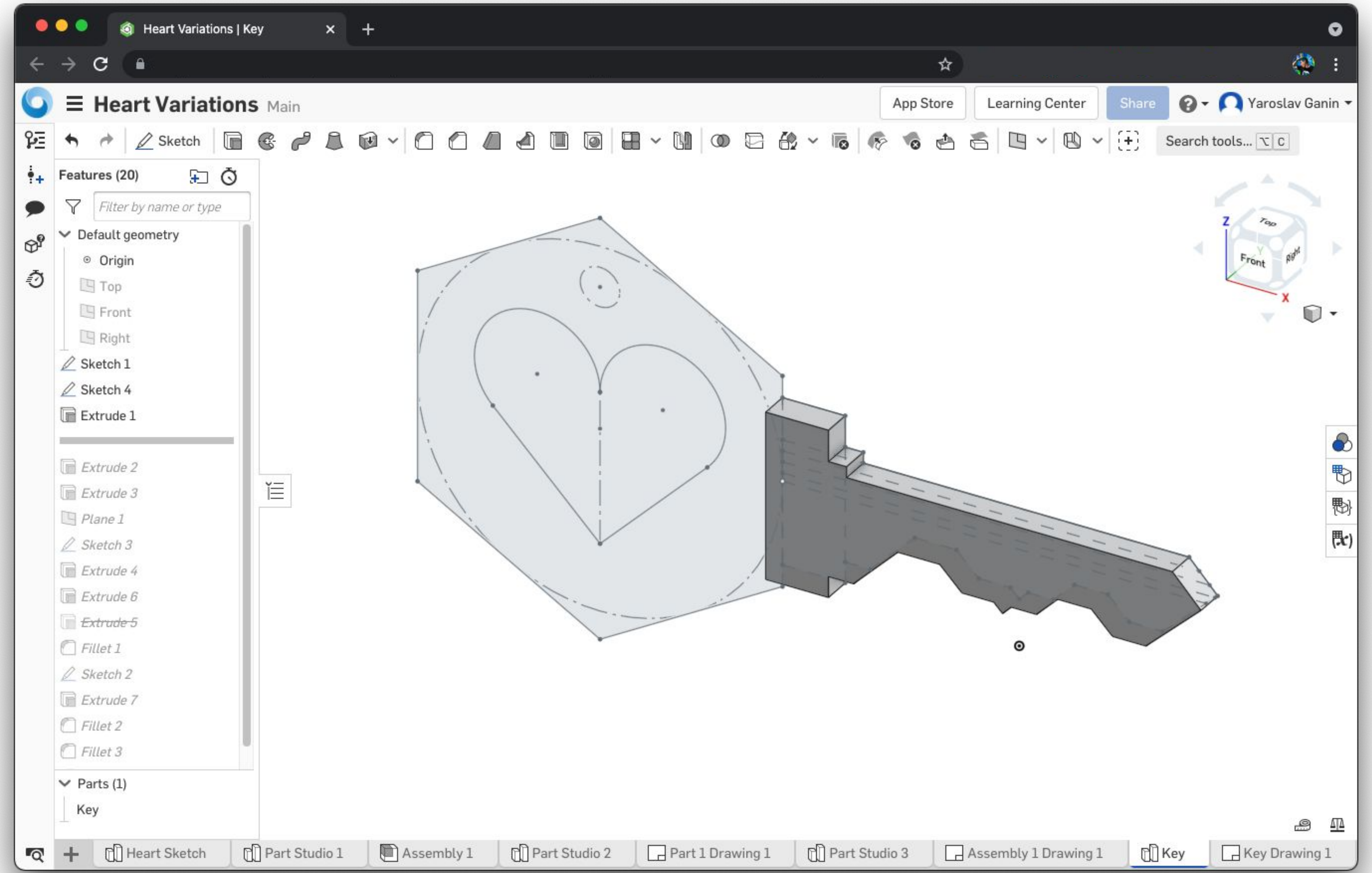
Anatomy of CAD



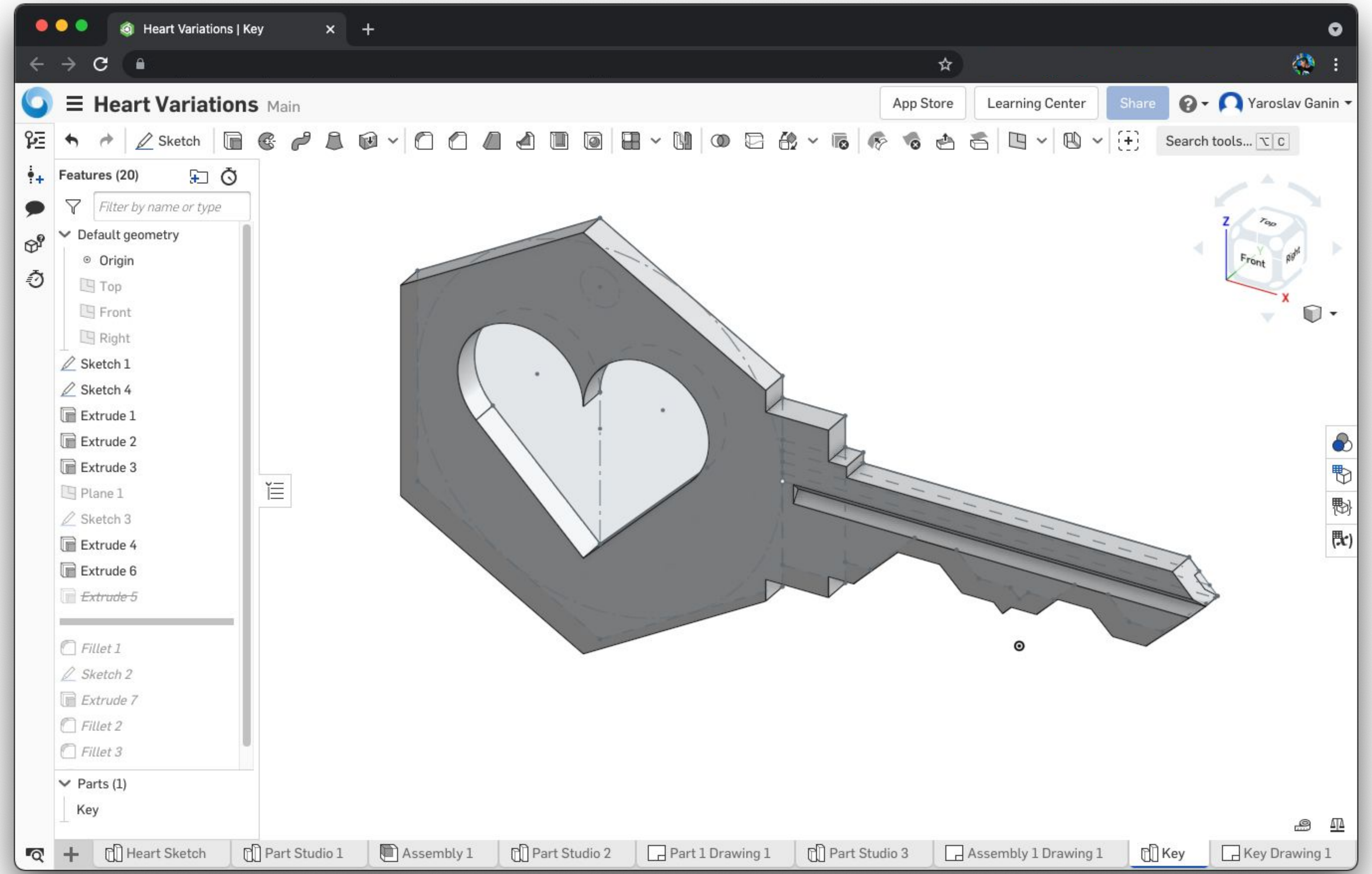
Anatomy of CAD



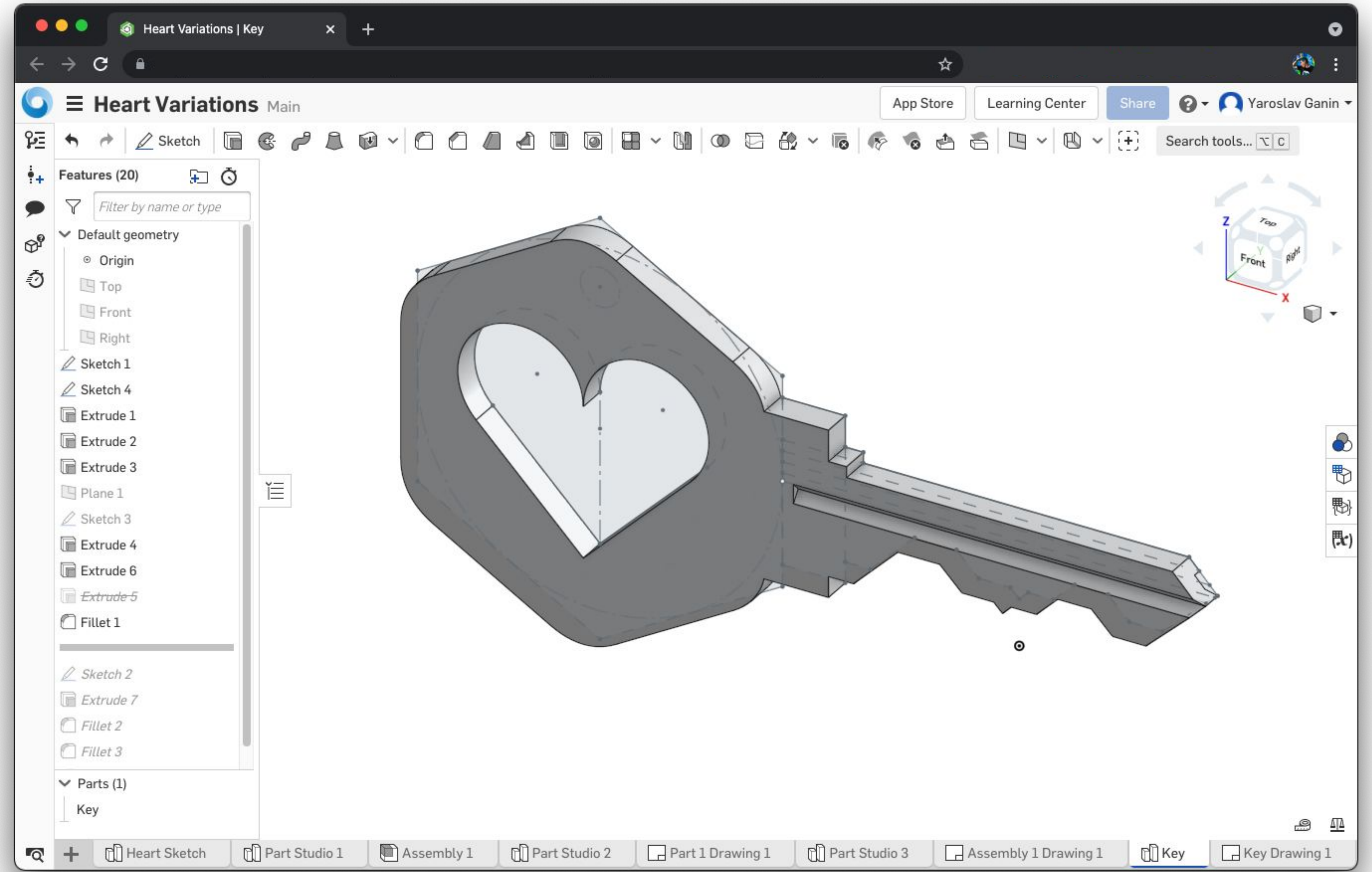
Anatomy of CAD



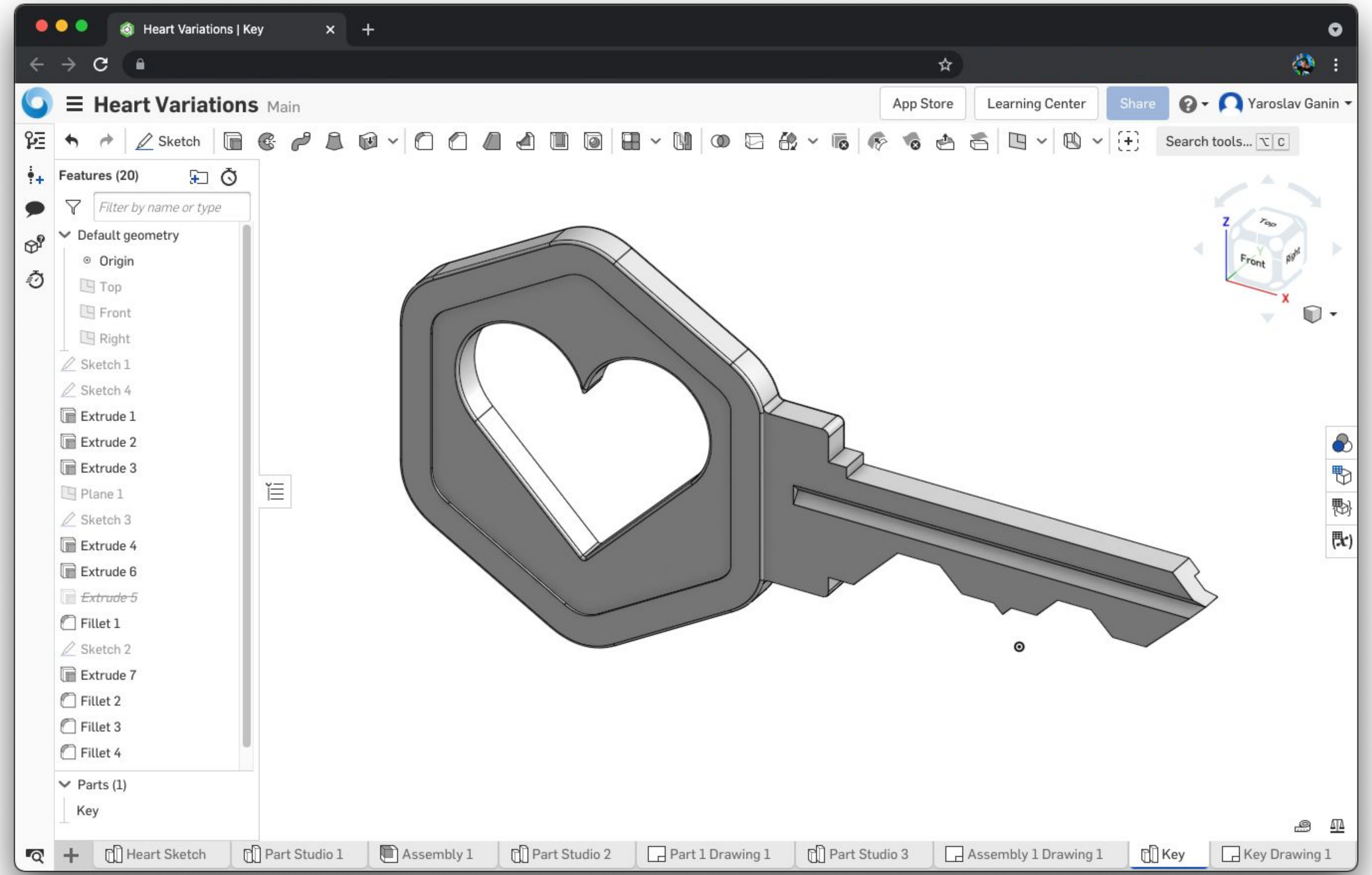
Anatomy of CAD



Anatomy of CAD

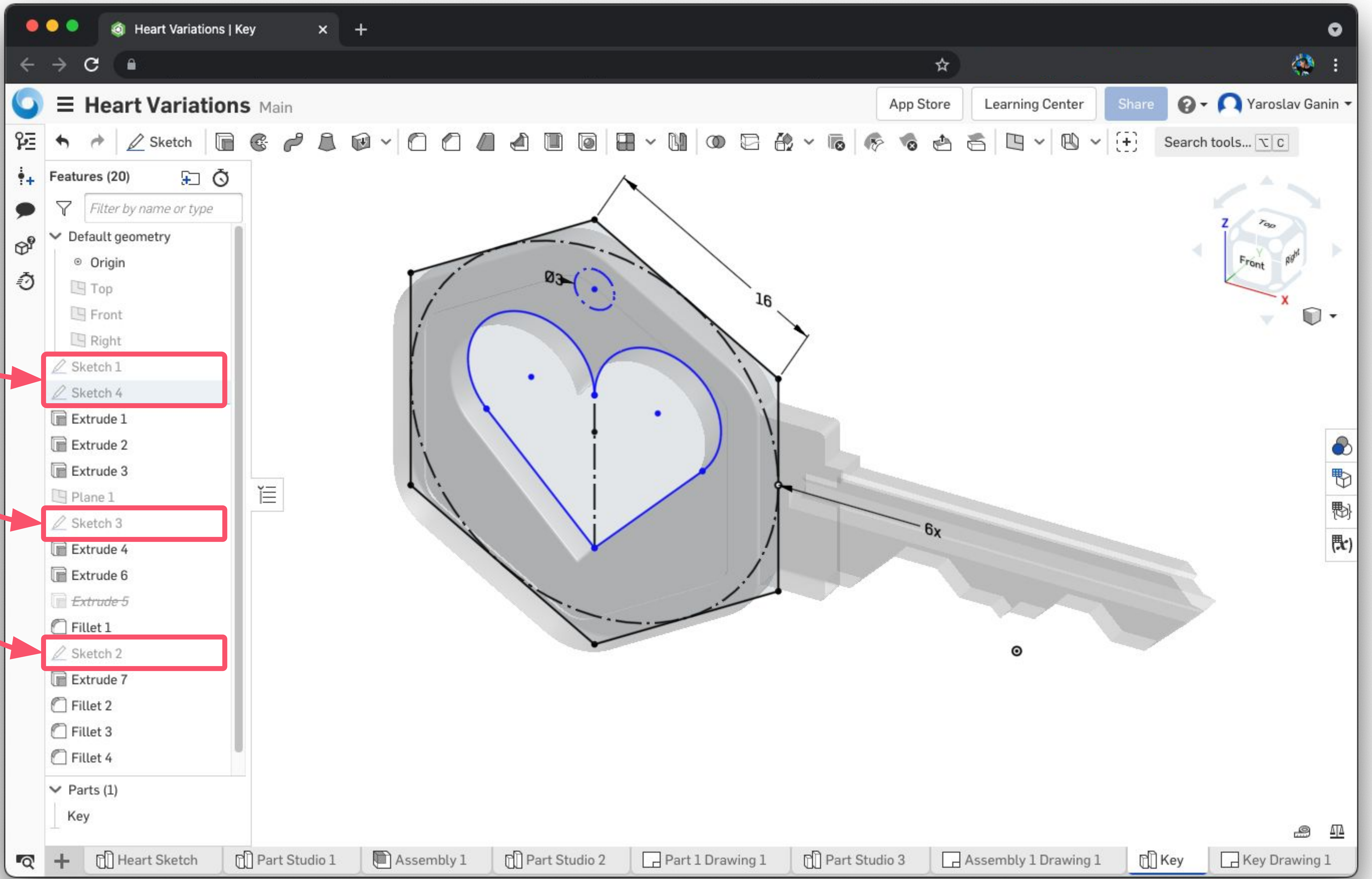


Anatomy of CAD



Anatomy of CAD

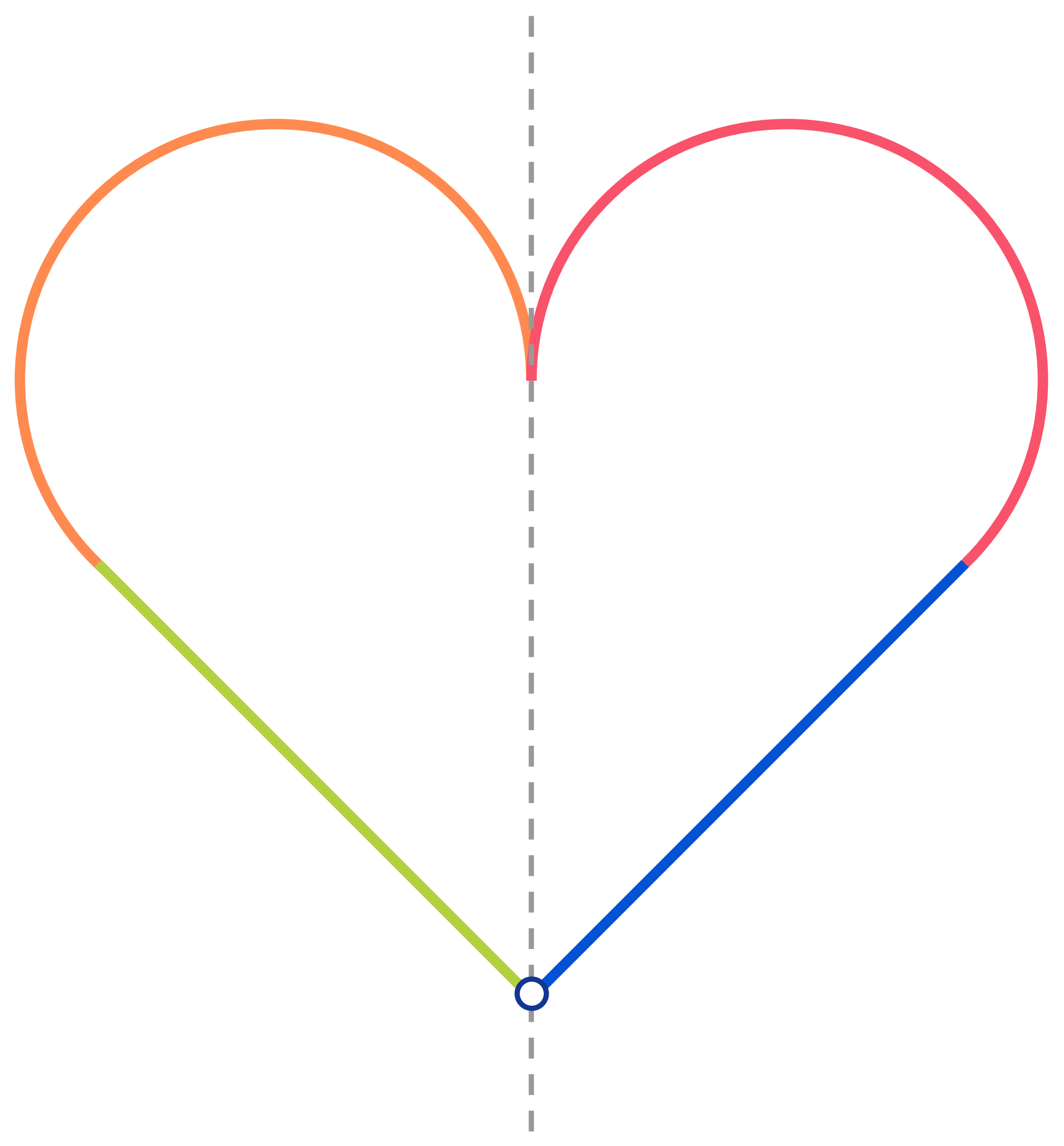
Our focus in this work



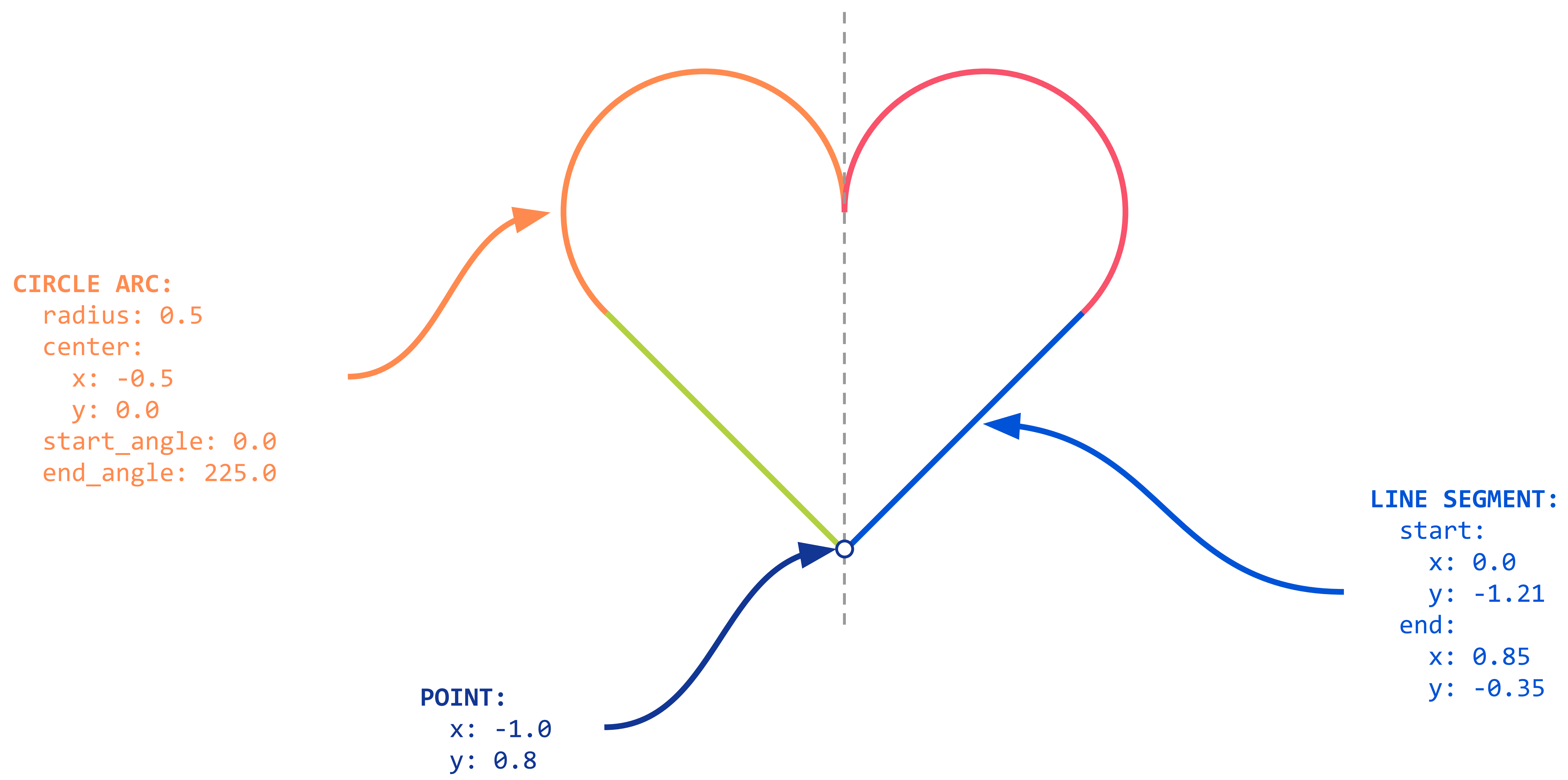
Sketching

- A **drawing on a plane**. Think straightedge and compass construction on steroids.
- **Integral part** of any 3D model design (**more than 50%** of the modeling effort is spent on creating sketches). Usually several sketches per shape.
- Handled by a dedicated server-side **solver** which receives **geometric entities** and **constraints** (as **sets**).

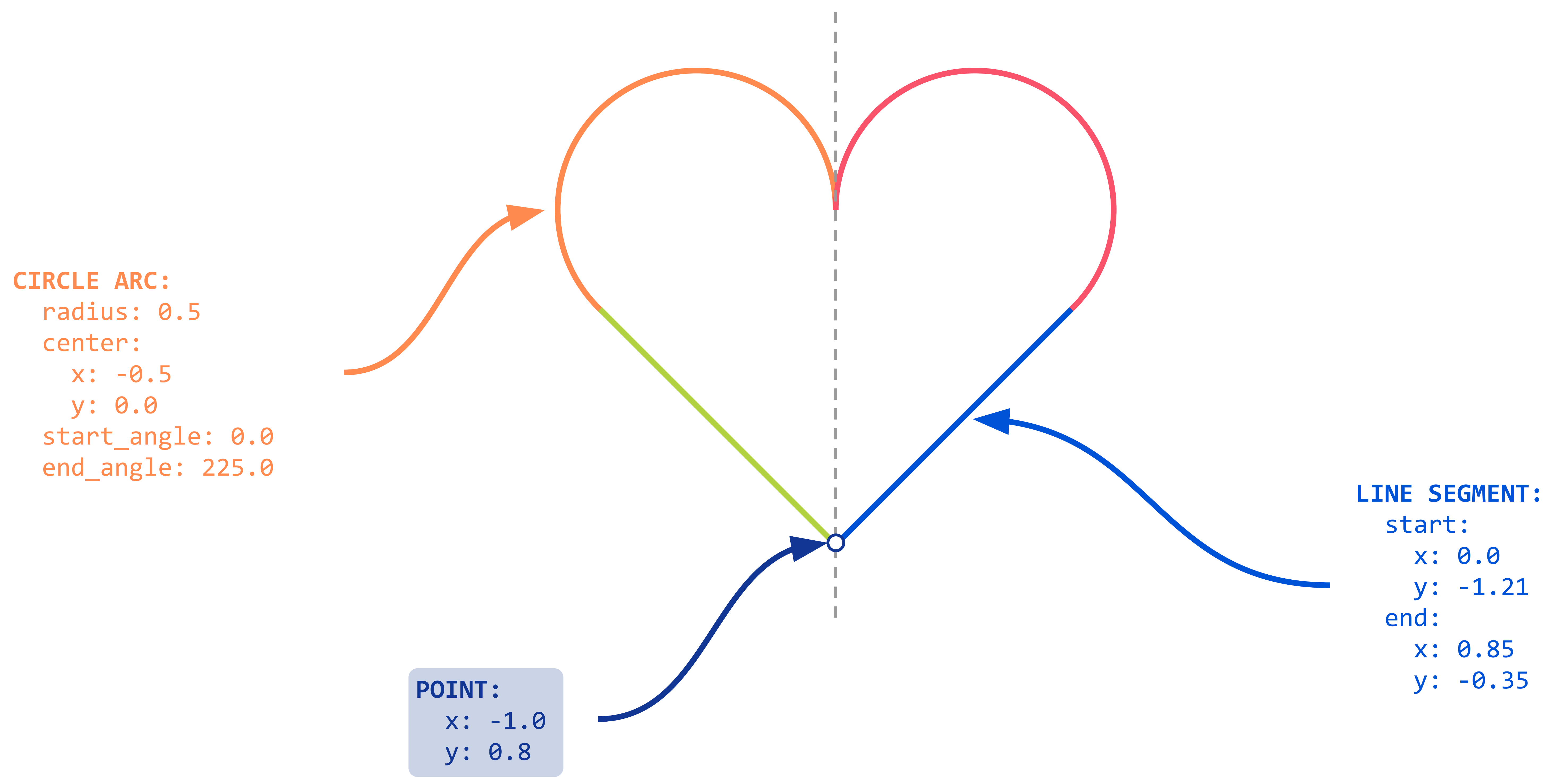
Sketching



Entities


CIRCLE ARC:
radius: 0.5
center:
x: -0.5
y: 0.0
start_angle: 0.0
end_angle: 225.0
LINE SEGMENT:
start:
x: 0.0
y: -1.21
end:
x: 0.85
y: -0.35
POINT:
x: -1.0
y: 0.8

Entities

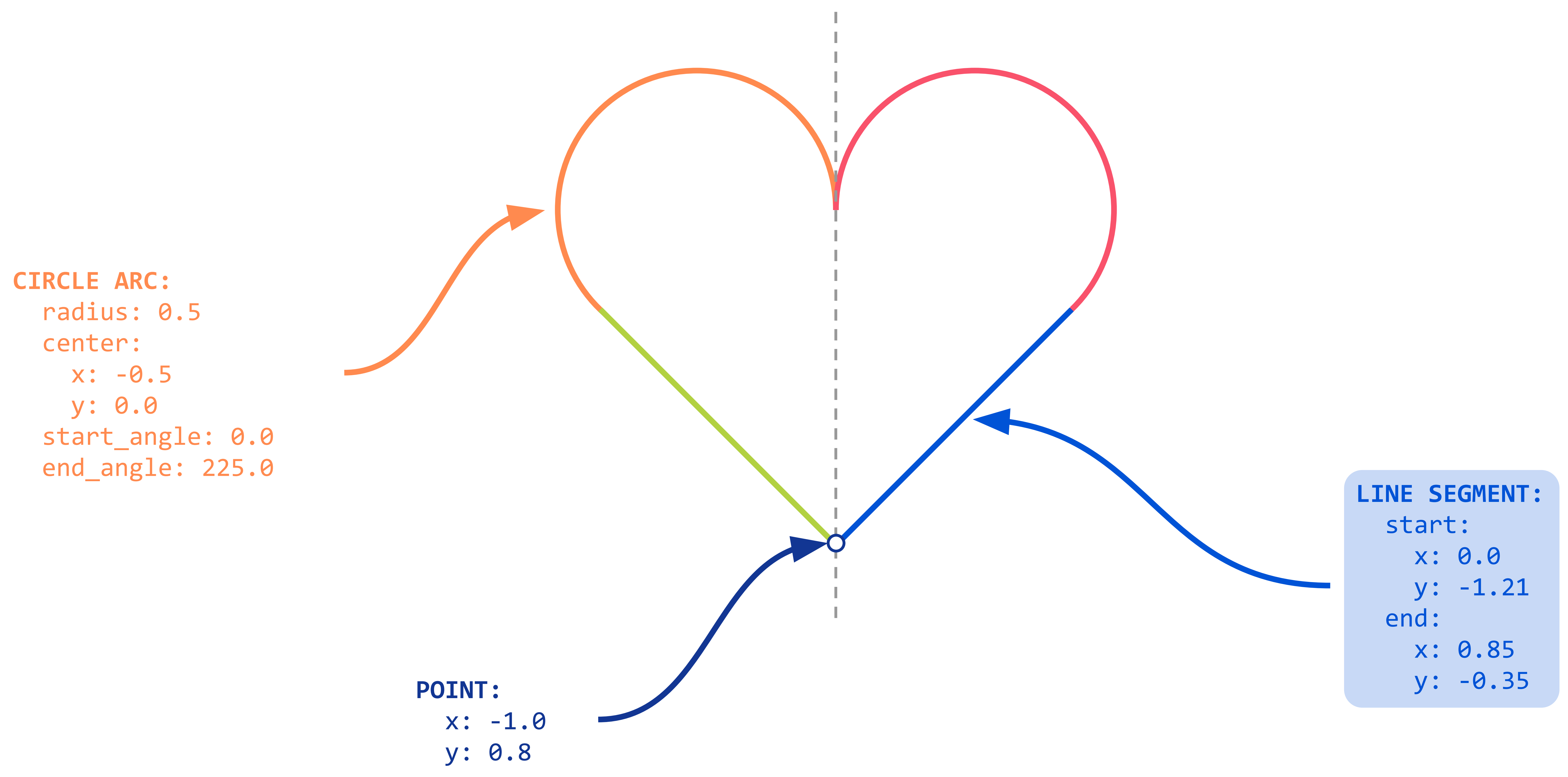


```
CIRCLE ARC:
  radius: 0.5
  center:
    x: -0.5
    y: 0.0
  start_angle: 0.0
  end_angle: 225.0
```

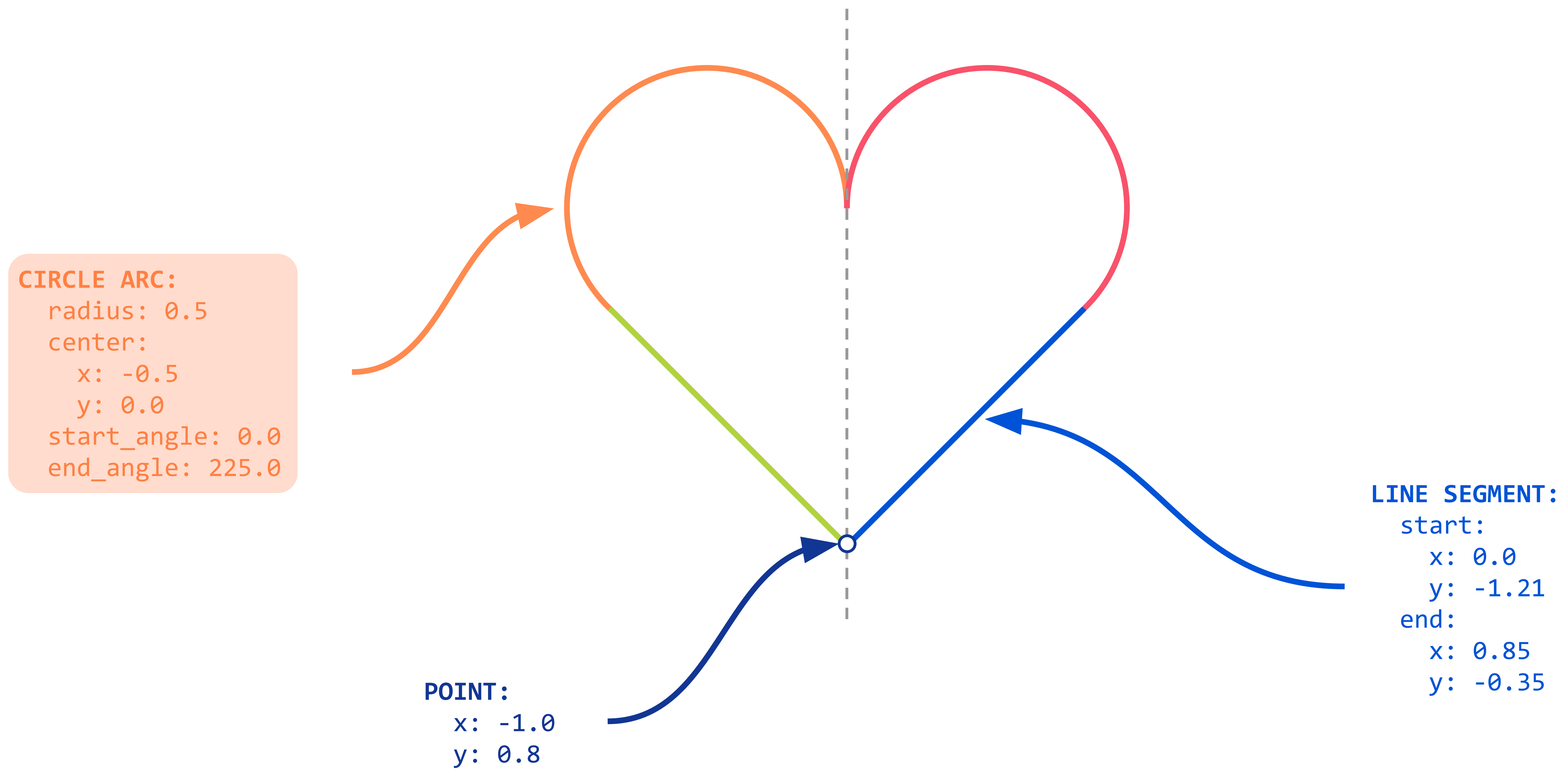
```
POINT:
  x: -1.0
  y: 0.8
```

```
LINE SEGMENT:
  start:
    x: 0.0
    y: -1.21
  end:
    x: 0.85
    y: -0.35
```

Entities


CIRCLE ARC:
radius: 0.5
center:
x: -0.5
y: 0.0
start_angle: 0.0
end_angle: 225.0
POINT:
x: -1.0
y: 0.8
LINE SEGMENT:
start:
x: 0.0
y: -1.21
end:
x: 0.85
y: -0.35

Entities

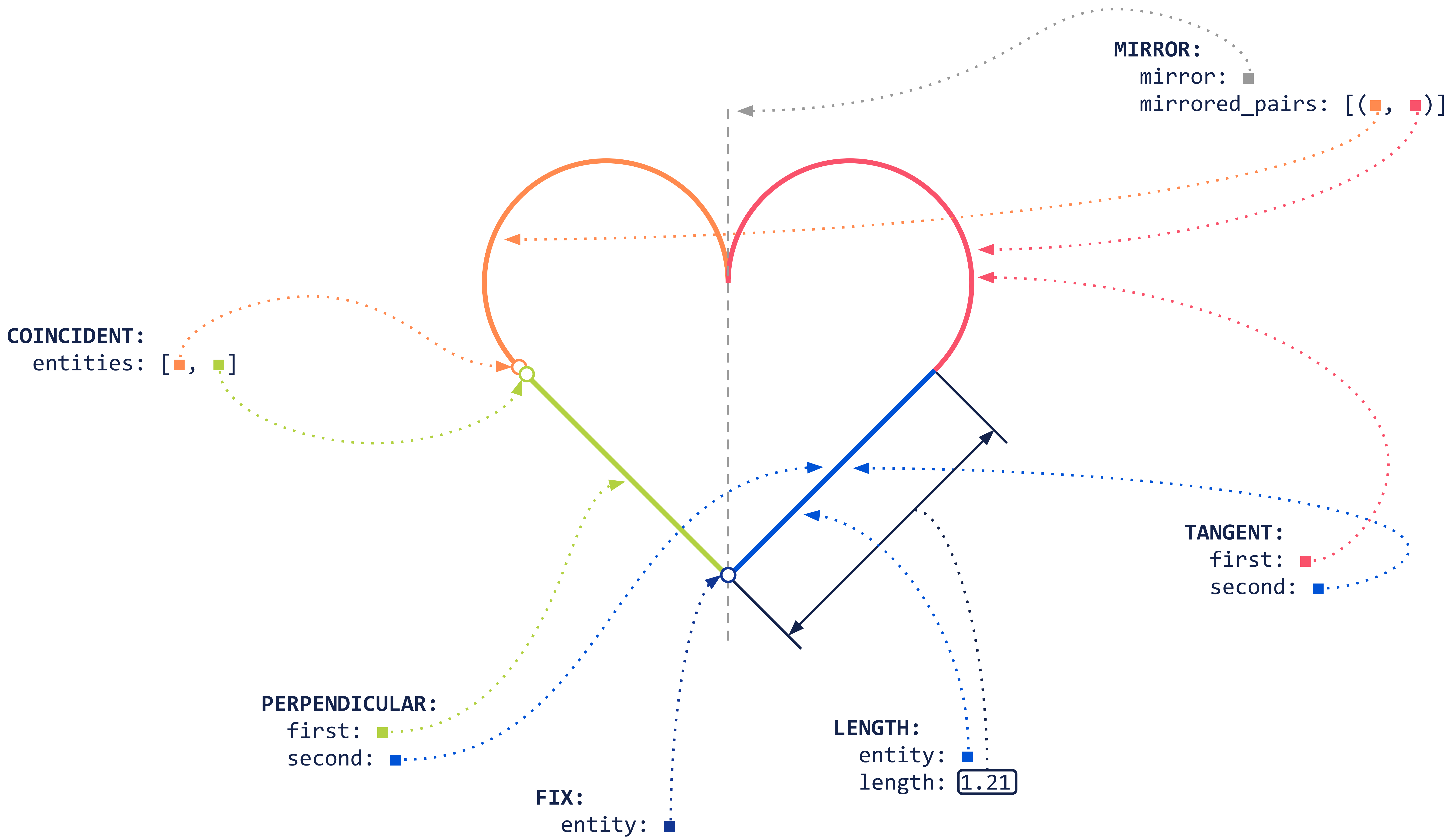


```
CIRCLE ARC:
  radius: 0.5
  center:
    x: -0.5
    y: 0.0
  start_angle: 0.0
  end_angle: 225.0
```

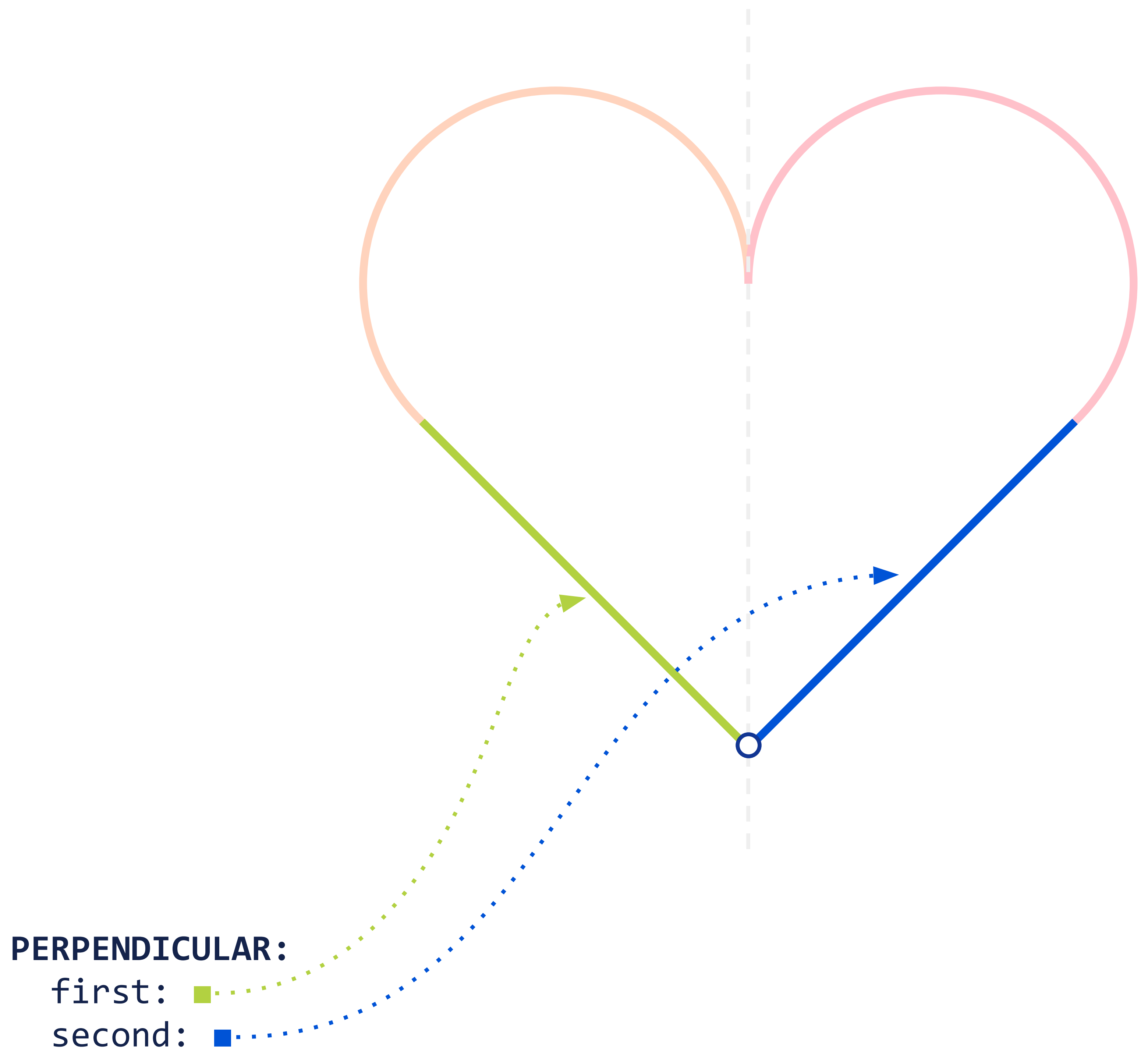
```
POINT:
  x: -1.0
  y: 0.8
```

```
LINE SEGMENT:
  start:
    x: 0.0
    y: -1.21
  end:
    x: 0.85
    y: -0.35
```

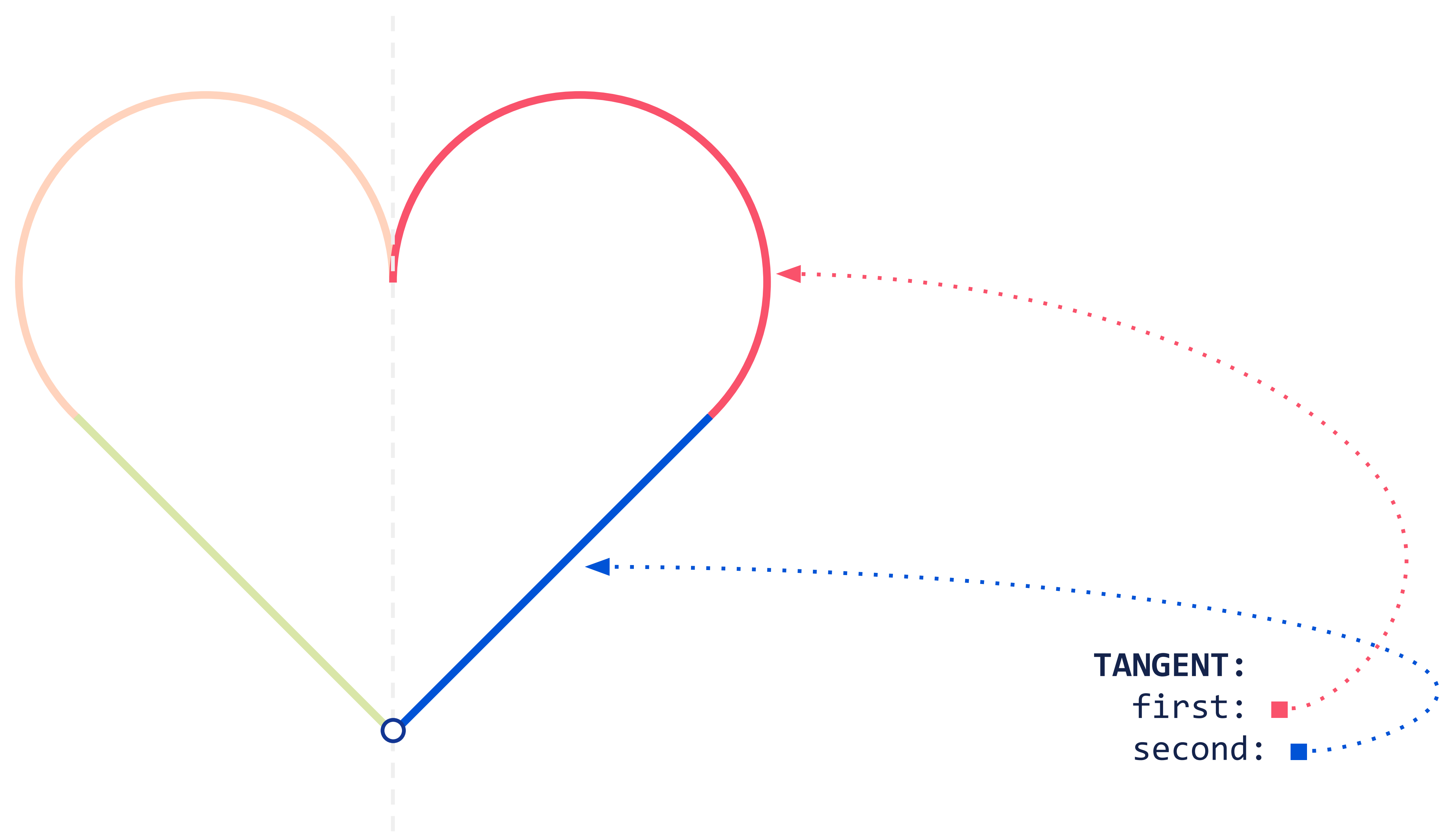
Constraints



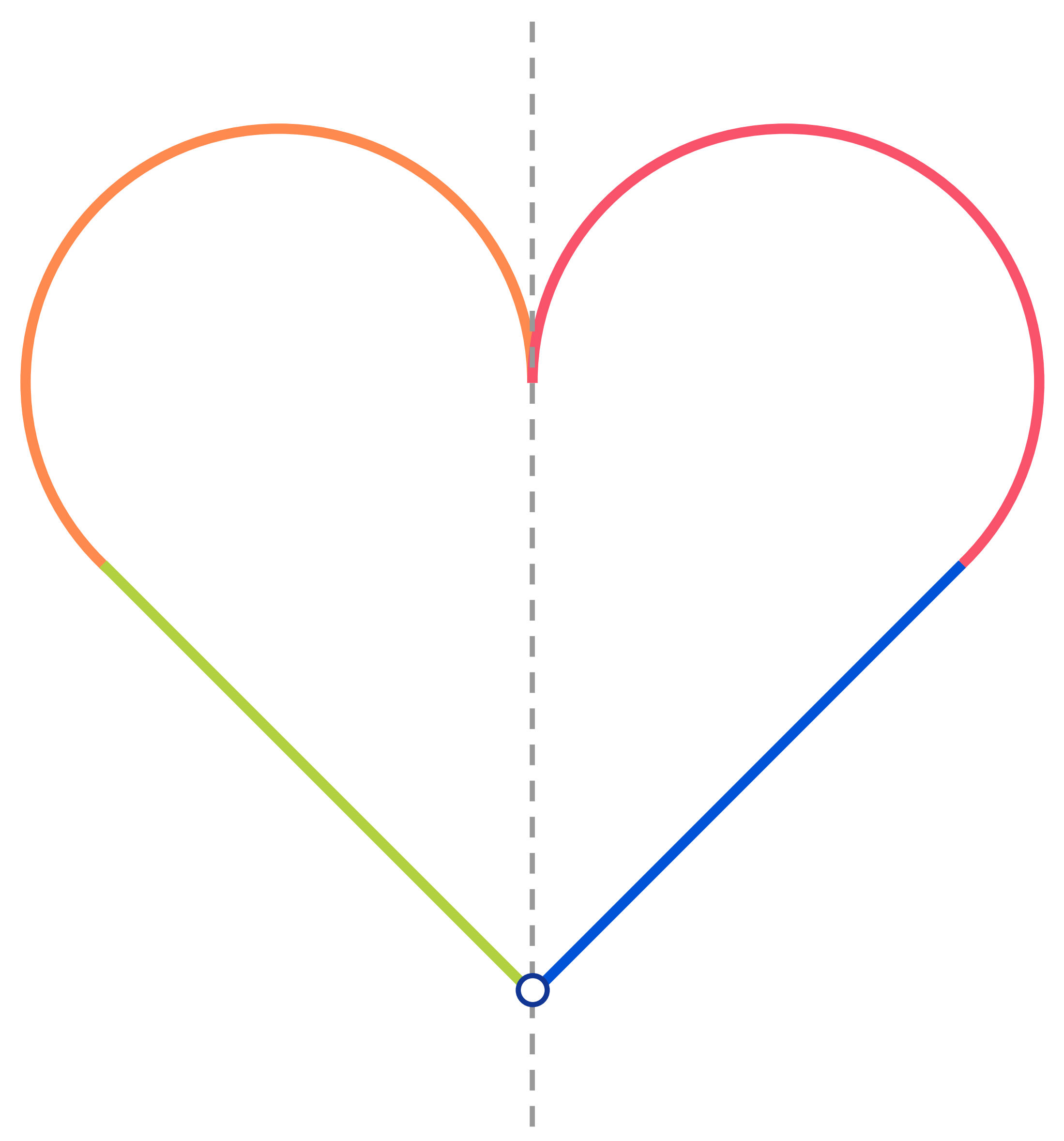
Constraints



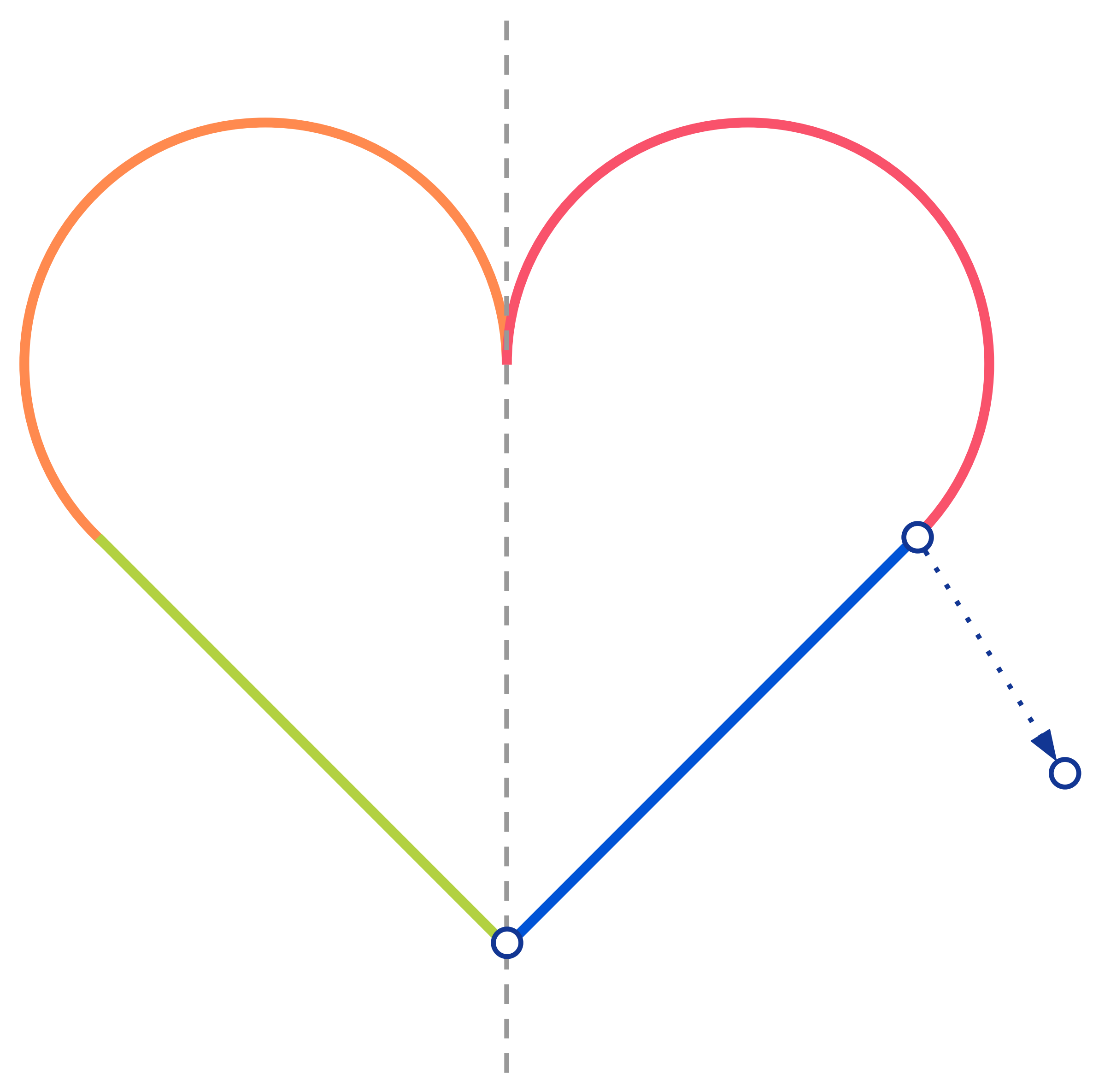
Constraints



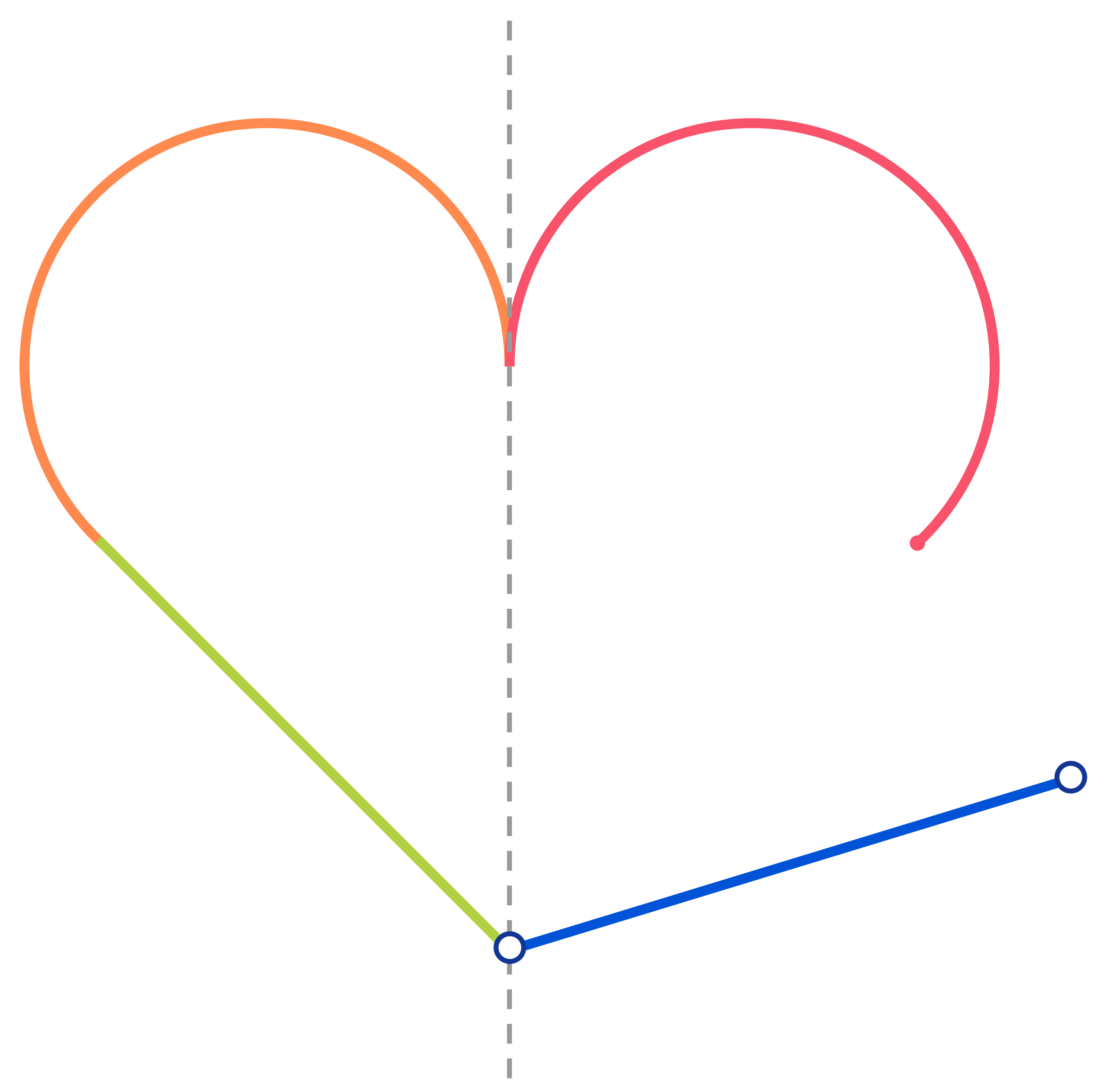
Constraints



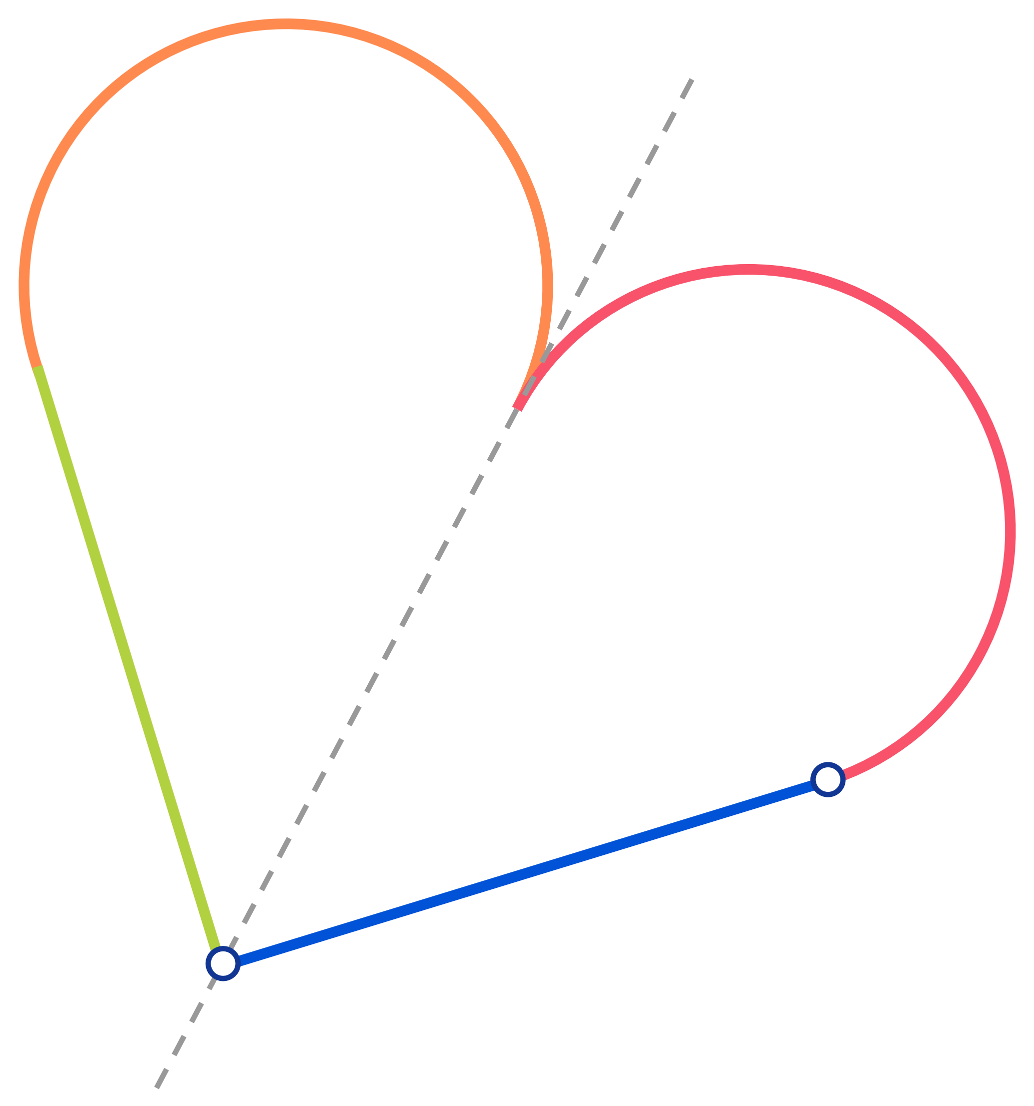
Constraints



Constraints



Constraints



Proto specification

```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}

message CircleArcEntity {
  bool is_construction = 1;
  Vector center = 2;
  double radius = 3;
  message ArcParams {
    Vector direction = 1;
    bool is_clockwise = 2;
    double start_angle = 3;
    double end_angle = 4;
  }
  oneof additional_params {
    google.protobuf.Empty circle_params = 4;
    ArcParams arc_params = 5;
  }
}
```

```
message DistanceConstraint {
  uint32 first = 1 [is_pointer = true];
  uint32 second = 2 [is_pointer = true];
  enum Direction {
    HORIZONTAL = 0;
    VERTICAL = 1;
    MINIMUM = 2;
  }
  Direction direction = 3;
  double length = 4;
  enum Alignment {
    ALIGNED = 0;
    ANTI_ALIGNED = 1;
  }
  enum HalfSpace {
    NOT_AVAILABLE = 0;
    LEFT = 1;
    RIGHT = 2;
  }
  message HalfSpaceParams {
    HalfSpace half_space_first = 1;
    HalfSpace half_space_second = 2;
  }
  oneof additional_params {
    option handler = "select_distance_params";
    Alignment alignment = 5;
    HalfSpaceParams half_space_params = 6;
  }
}
```

Proto specification

```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}

message CircleArcEntity {
  bool is_construction = 1;
  Vector center = 2;
  double radius = 3;
  message ArcParams {
    Vector direction = 1;
    bool is_clockwise = 2;
    double start_angle = 3;
    double end_angle = 4;
  }
  oneof additional_params {
    google.protobuf.Empty circle_params = 4;
    ArcParams arc_params = 5;
  }
}
```

```
message DistanceConstraint {
  uint32 first = 1 [is_pointer = true];
  uint32 second = 2 [is_pointer = true];
  enum Direction {
    HORIZONTAL = 0;
    VERTICAL = 1;
    MINIMUM = 2;
  }
  Direction direction = 3;
  double length = 4;
  enum Alignment {
    ALIGNED = 0;
    ANTI_ALIGNED = 1;
  }
  enum HalfSpace {
    NOT_AVAILABLE = 0;
    LEFT = 1;
    RIGHT = 2;
  }
  message HalfSpaceParams {
    HalfSpace half_space_first = 1;
    HalfSpace half_space_second = 2;
  }
  oneof additional_params {
    option handler = "select_distance_params";
    Alignment alignment = 5;
    HalfSpaceParams half_space_params = 6;
  }
}
```

Proto specification

```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}

message CircleArcEntity {
  bool is_construction = 1;
  Vector center = 2;
  double radius = 3;
  message ArcParams {
    Vector direction = 1;
    bool is_clockwise = 2;
    double start_angle = 3;
    double end_angle = 4;
  }
  oneof additional_params {
    google.protobuf.Empty circle_params = 4;
    ArcParams arc_params = 5;
  }
}
```

```
message DistanceConstraint {
  uint32 first = 1 [is_pointer = true];
  uint32 second = 2 [is_pointer = true];
  enum Direction {
    HORIZONTAL = 0;
    VERTICAL = 1;
    MINIMUM = 2;
  }
  Direction direction = 3;
  double length = 4;
  enum Alignment {
    ALIGNED = 0;
    ANTI_ALIGNED = 1;
  }
  enum HalfSpace {
    NOT_AVAILABLE = 0;
    LEFT = 1;
    RIGHT = 2;
  }
  message HalfSpaceParams {
    HalfSpace half_space_first = 1;
    HalfSpace half_space_second = 2;
  }
  oneof additional_params {
    option handler = "select_distance_params";
    Alignment alignment = 5;
    HalfSpaceParams half_space_params = 6;
  }
}
```

Proto specification

```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}

message CircleArcEntity {
  bool is_construction = 1;
  Vector center = 2;
  double radius = 3;
  message ArcParams {
    Vector direction = 1;
    bool is_clockwise = 2;
    double start_angle = 3;
    double end_angle = 4;
  }
  oneof additional_params {
    google.protobuf.Empty circle_params = 4;
    ArcParams arc_params = 5;
  }
}
```

```
message DistanceConstraint {
  uint32 first = 1 [is_pointer = true];
  uint32 second = 2 [is_pointer = true];
  enum Direction {
    HORIZONTAL = 0;
    VERTICAL = 1;
    MINIMUM = 2;
  }
  Direction direction = 3;
  double length = 4;
  enum Alignment {
    ALIGNED = 0;
    ANTI_ALIGNED = 1;
  }
  enum HalfSpace {
    NOT_AVAILABLE = 0;
    LEFT = 1;
    RIGHT = 2;
  }
  message HalfSpaceParams {
    HalfSpace half_space_first = 1;
    HalfSpace half_space_second = 2;
  }
  oneof additional_params {
    option handler = "select_distance_params";
    Alignment alignment = 5;
    HalfSpaceParams half_space_params = 6;
  }
}
```

Proto specification

```
message Entity {
  oneof kind {
    PointEntity point_entity = 1;
    LineEntity line_entity = 2;
    // ...
  }
}

message Constraint {
  oneof kind {
    FixConstraint fix_constraint = 1;
    CoincidentConstraint coincident_constraint = 2;
    // ...
    LengthConstraint length_constraint = 10;
    // ...
  }
}
```

```
message Object {
  oneof kind {
    Entity entity = 1;
    Constraint constraint = 2;
  }
}

message Sequence {
  repeated Object objects = 1;
}
```

Proto specification

```
message Entity {
  oneof kind {
    PointEntity point_entity = 1;
    LineEntity line_entity = 2;
    // ...
  }
}

message Constraint {
  oneof kind {
    FixConstraint fix_constraint = 1;
    CoincidentConstraint coincident_constraint = 2;
    // ...
    LengthConstraint length_constraint = 10;
    // ...
  }
}
```

```
message Object {
  oneof kind {
    Entity entity = 1;
    Constraint constraint = 2;
  }
}
```

```
message Sequence {
  repeated Object objects = 1;
}
```

A sketch is simply a **sequence** of **objects** each of which is either an **entity** or a **constraint**

Proto specification

```
message Entity {
  oneof kind {
    PointEntity point_entity = 1;
    LineEntity line_entity = 2;
    // ...
  }
}

message Constraint {
  oneof kind {
    FixConstraint fix_constraint = 1;
    CoincidentConstraint coincident_constraint = 2;
    // ...
    LengthConstraint length_constraint = 10;
    // ...
  }
}
```

```
message Object {
  oneof kind {
    Entity entity = 1;
    Constraint constraint = 2;
  }
}

message Sequence {
  repeated Object objects = 1;
}
```

Proto specification

```
message Entity {
  oneof kind {
    PointEntity point_entity = 1;
    LineEntity line_entity = 2;
    // ...
  }
}

message Constraint {
  oneof kind {
    FixConstraint fix_constraint = 1;
    CoincidentConstraint coincident_constraint = 2;
    // ...
    LengthConstraint length_constraint = 10;
    // ...
  }
}
```

```
message Object {
  oneof kind {
    Entity entity = 1;
    Constraint constraint = 2;
  }
}

message Sequence {
  repeated Object objects = 1;
}
```

Proto specification

```
message Entity {
  oneof kind {
    PointEntity point_entity = 1;
    LineEntity line_entity = 2;
    // ...
  }
}

message Constraint {
  oneof kind {
    FixConstraint fix_constraint = 1;
    CoincidentConstraint coincident_constraint = 2;
    // ...
    LengthConstraint length_constraint = 10;
    // ...
  }
}
```

```
message Object {
  oneof kind {
    Entity entity = 1;
    Constraint constraint = 2;
  }
}

message Sequence {
  repeated Object objects = 1;
}
```

Proto specification

```
message Entity {
  oneof kind {
    PointEntity point_entity = 1;
    LineEntity line_entity = 2;
    // ...
  }
}

message Constraint {
  oneof kind {
    FixConstraint fix_constraint = 1;
    CoincidentConstraint coincident_constraint = 2;
    // ...
    LengthConstraint length_constraint = 10;
    // ...
  }
}
```

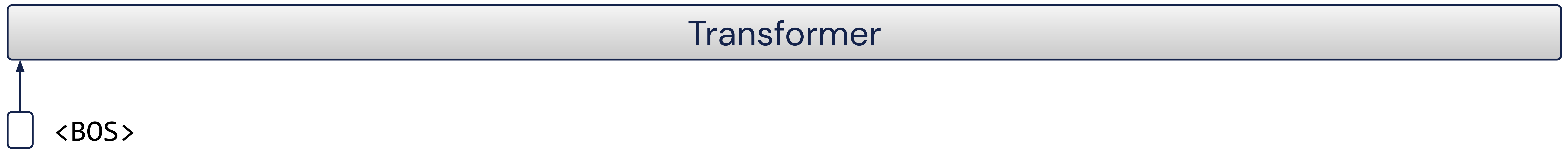
```
message Object {
  oneof kind {
    Entity entity = 1;
    Constraint constraint = 2;
  }
}

message Sequence {
  repeated Object objects = 1;
}
```

Interpreter-guided generation

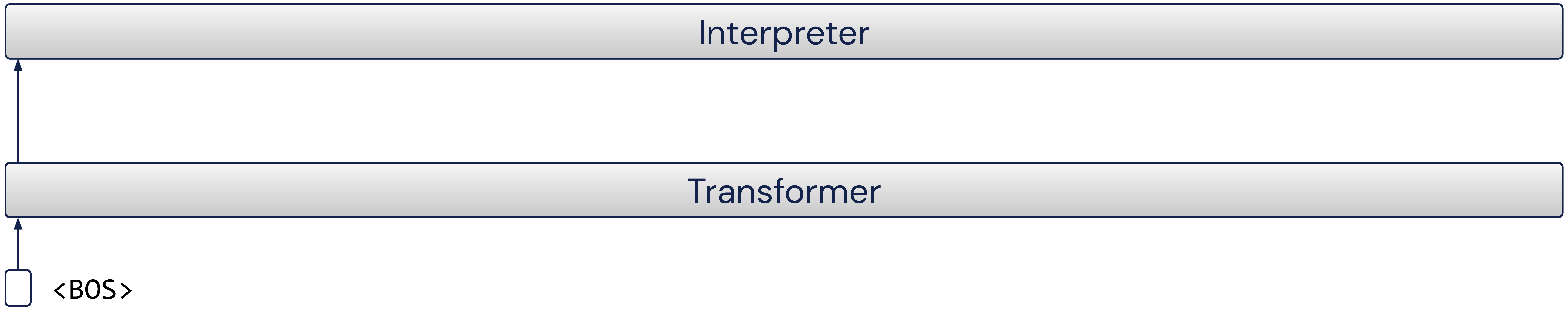
Interpreter-guided generation

▯ `<BOS>`

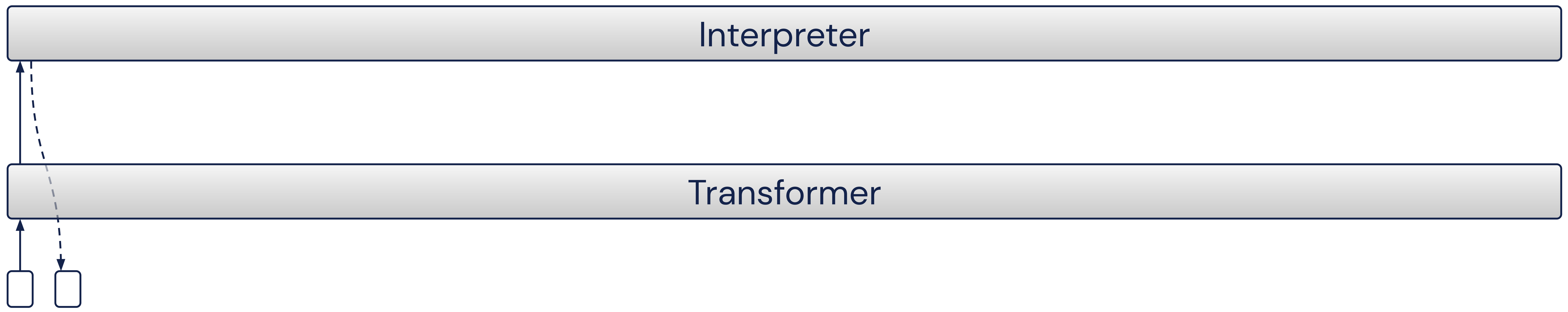
Interpreter-guided generation



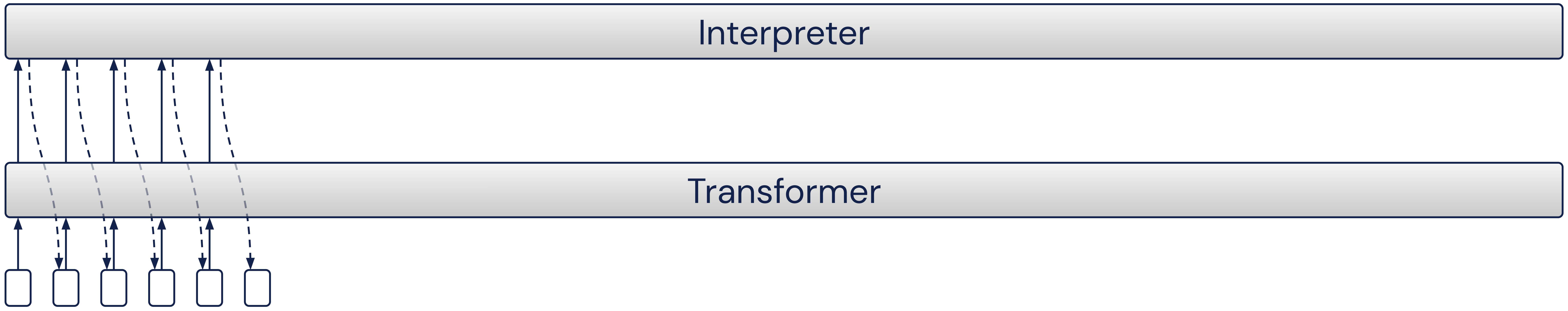
Interpreter-guided generation



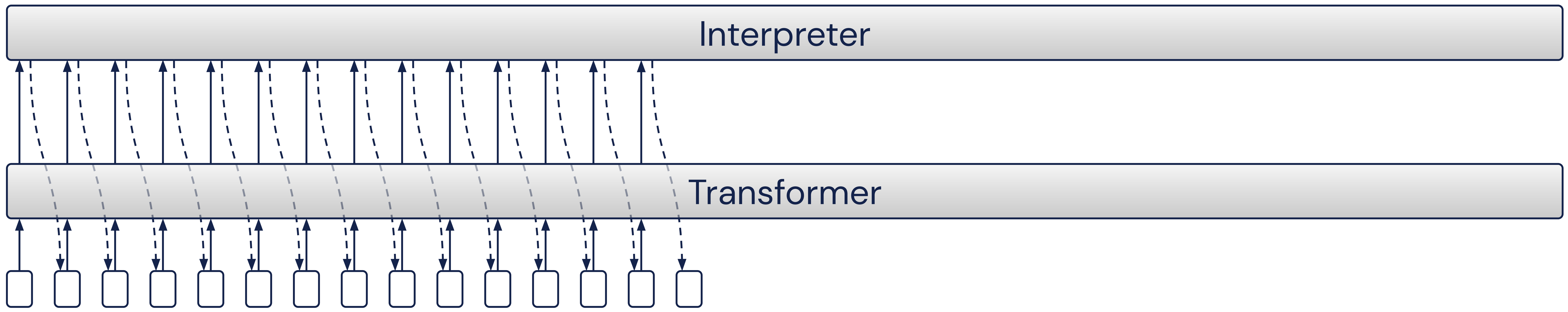
Interpreter-guided generation

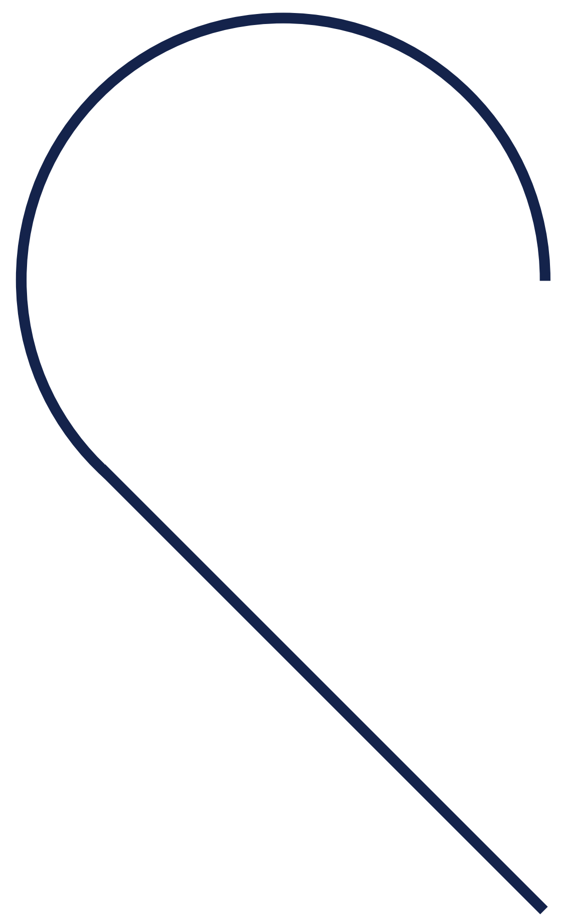


Interpreter-guided generation

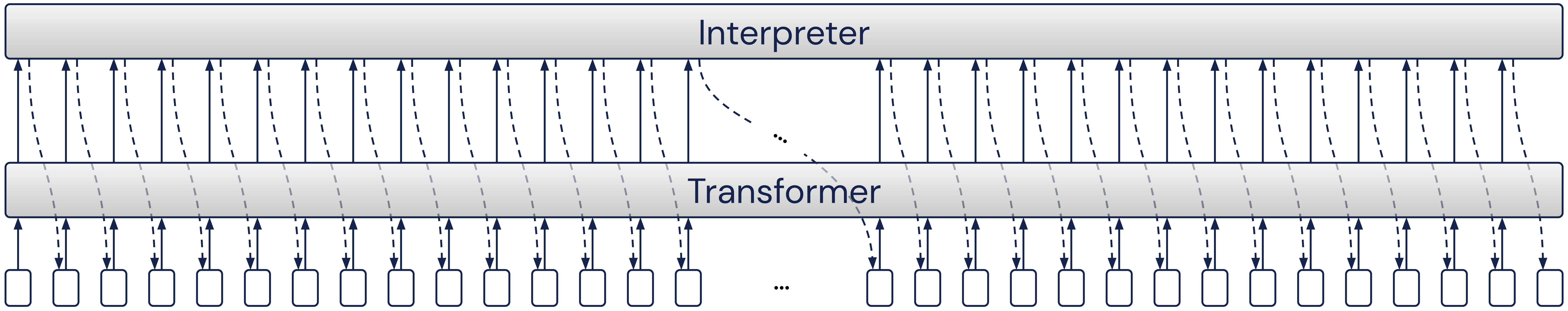


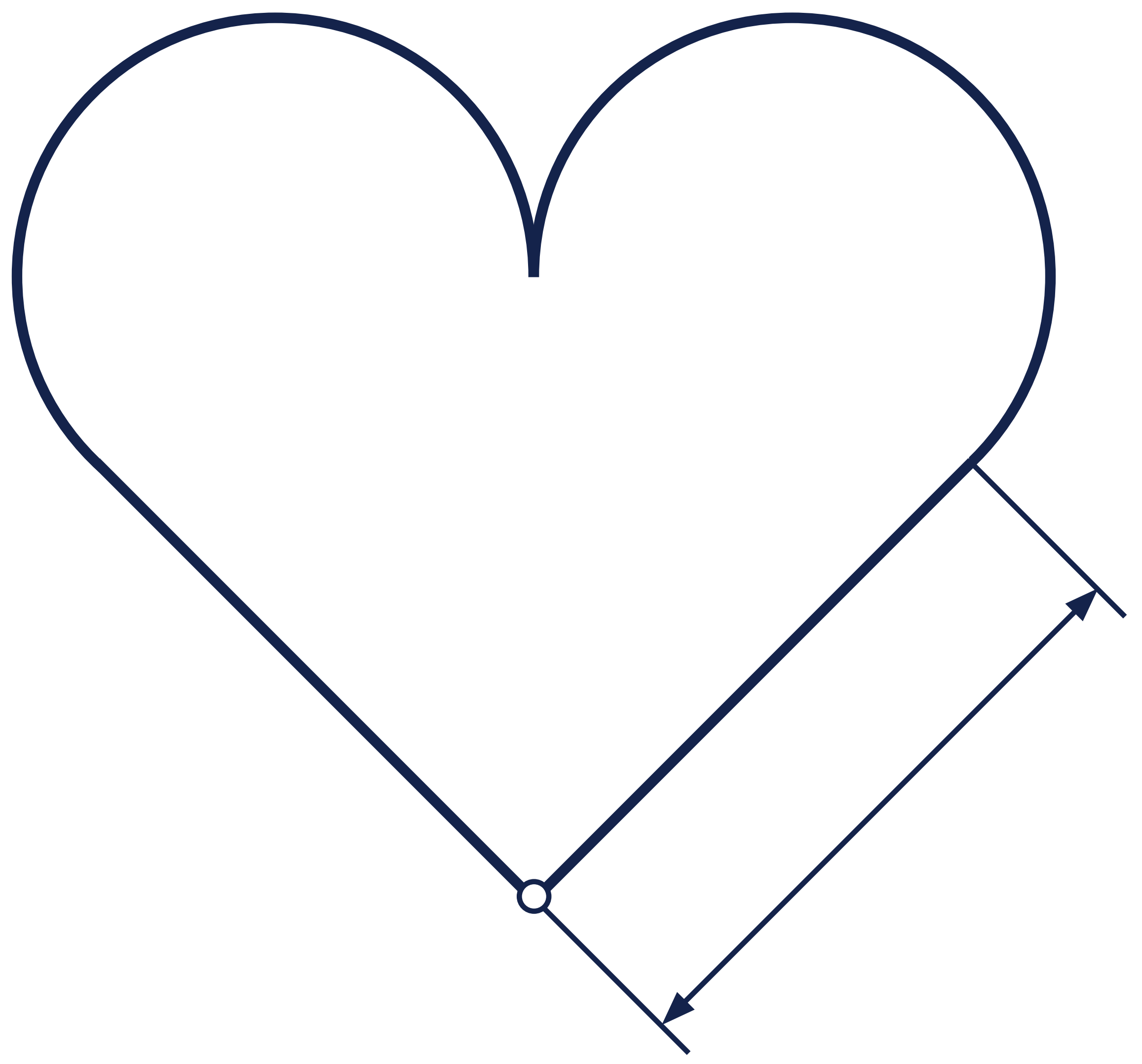
Interpreter-guided generation



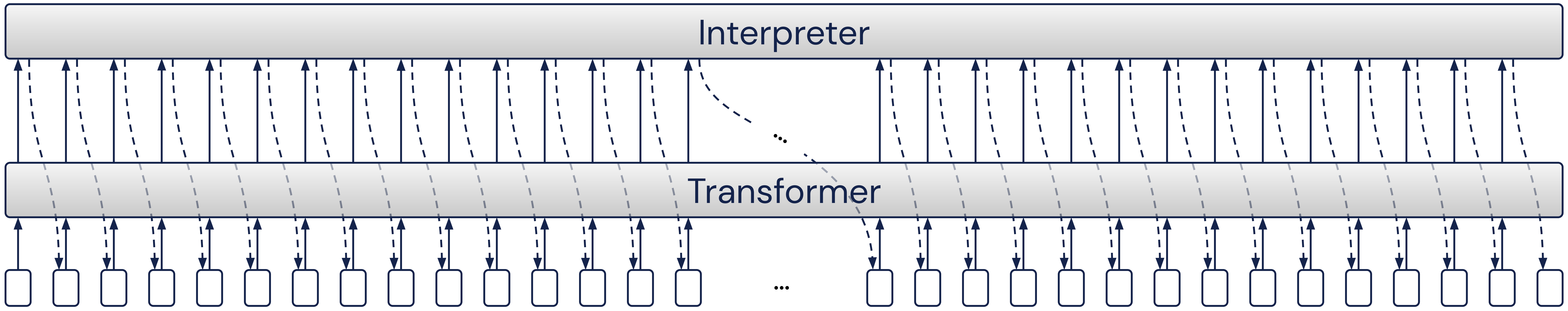


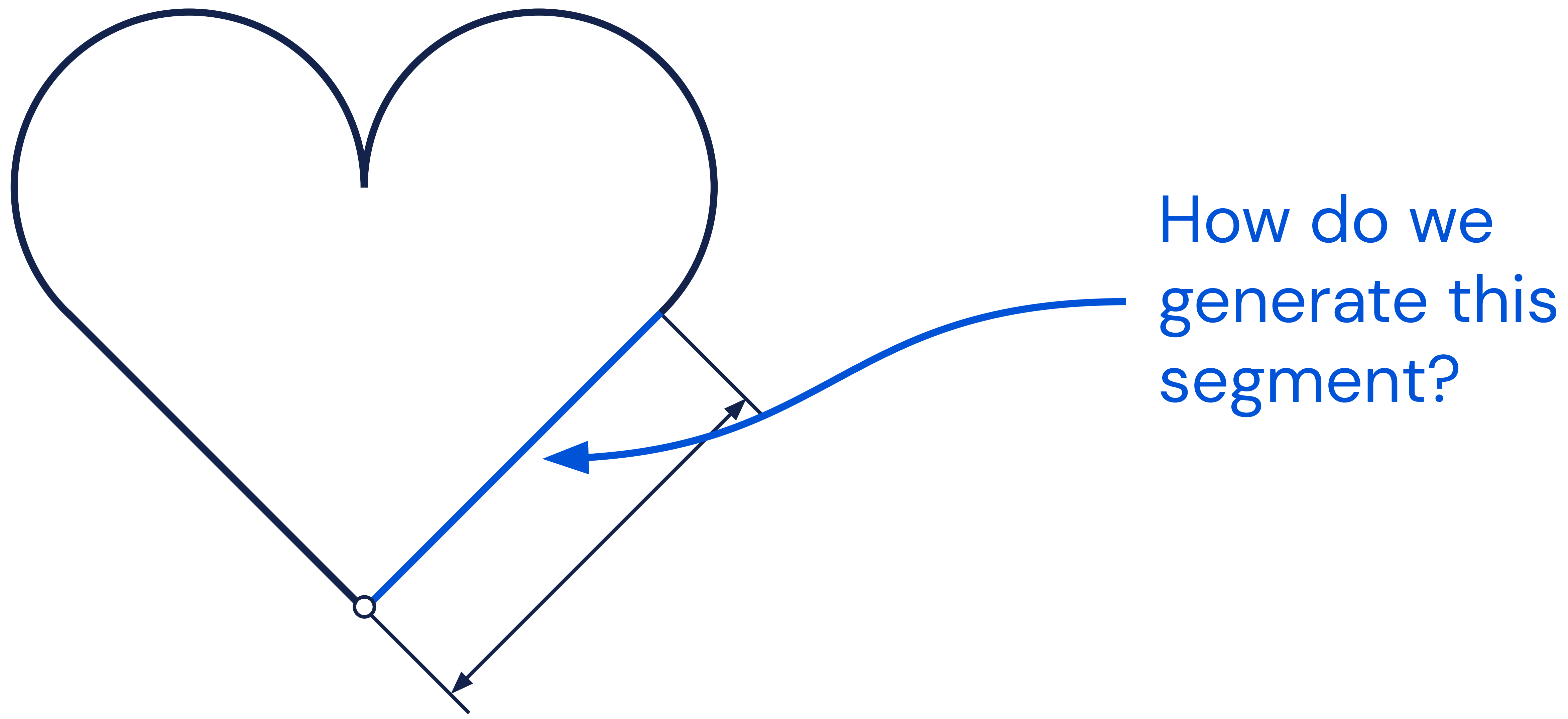
Interpreter-guided generation





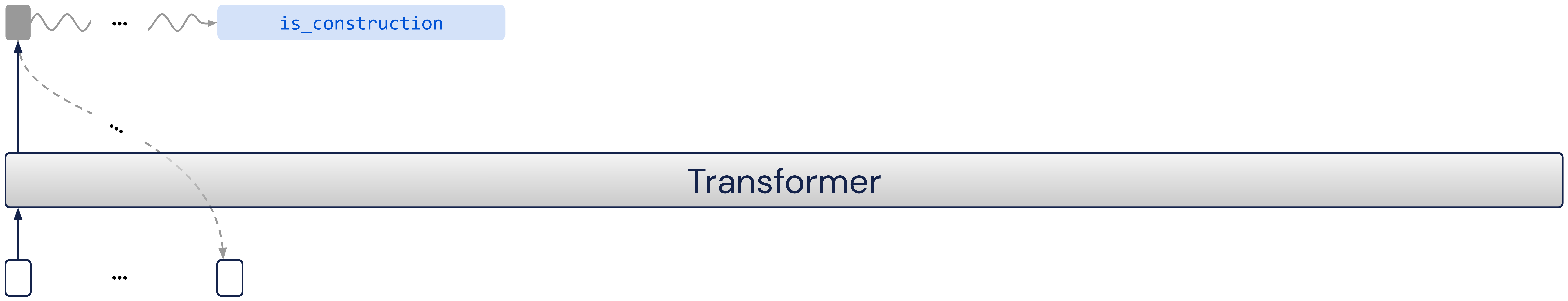
Interpreter-guided generation



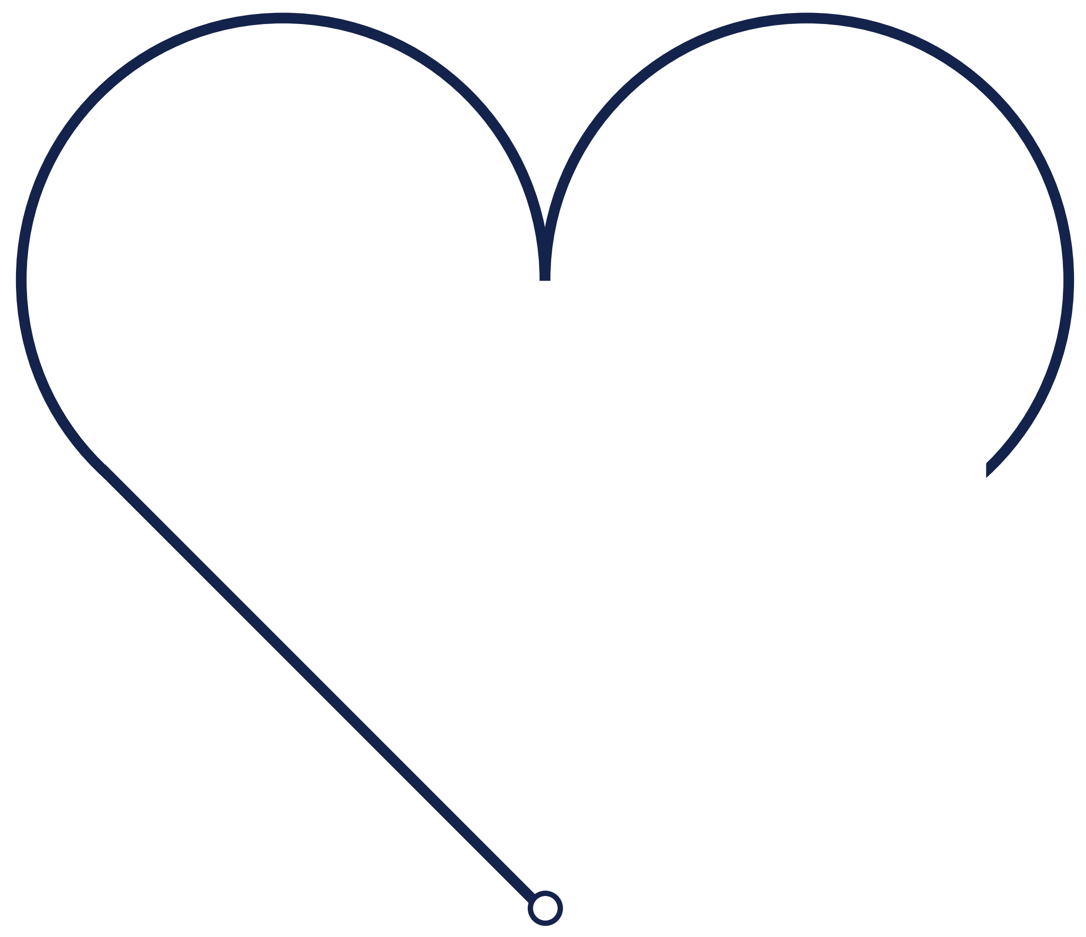


Interpreter-guided generation

Private & Confidential

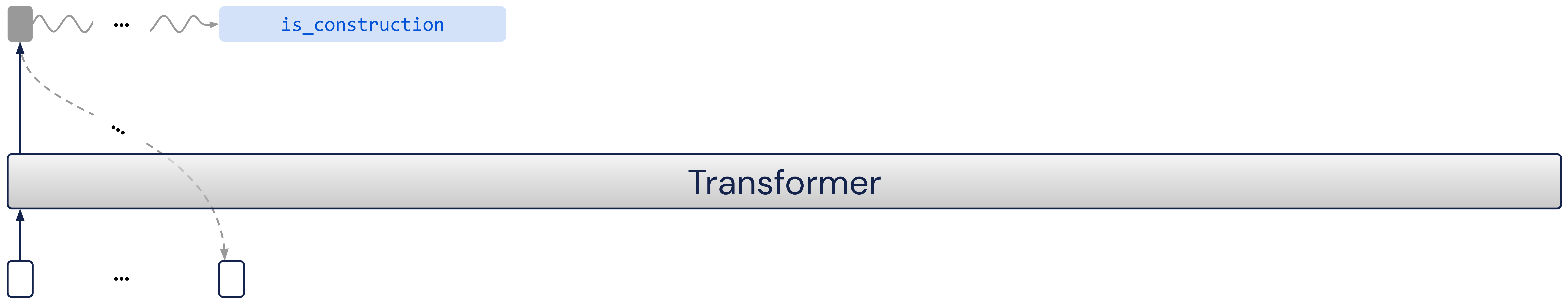


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```

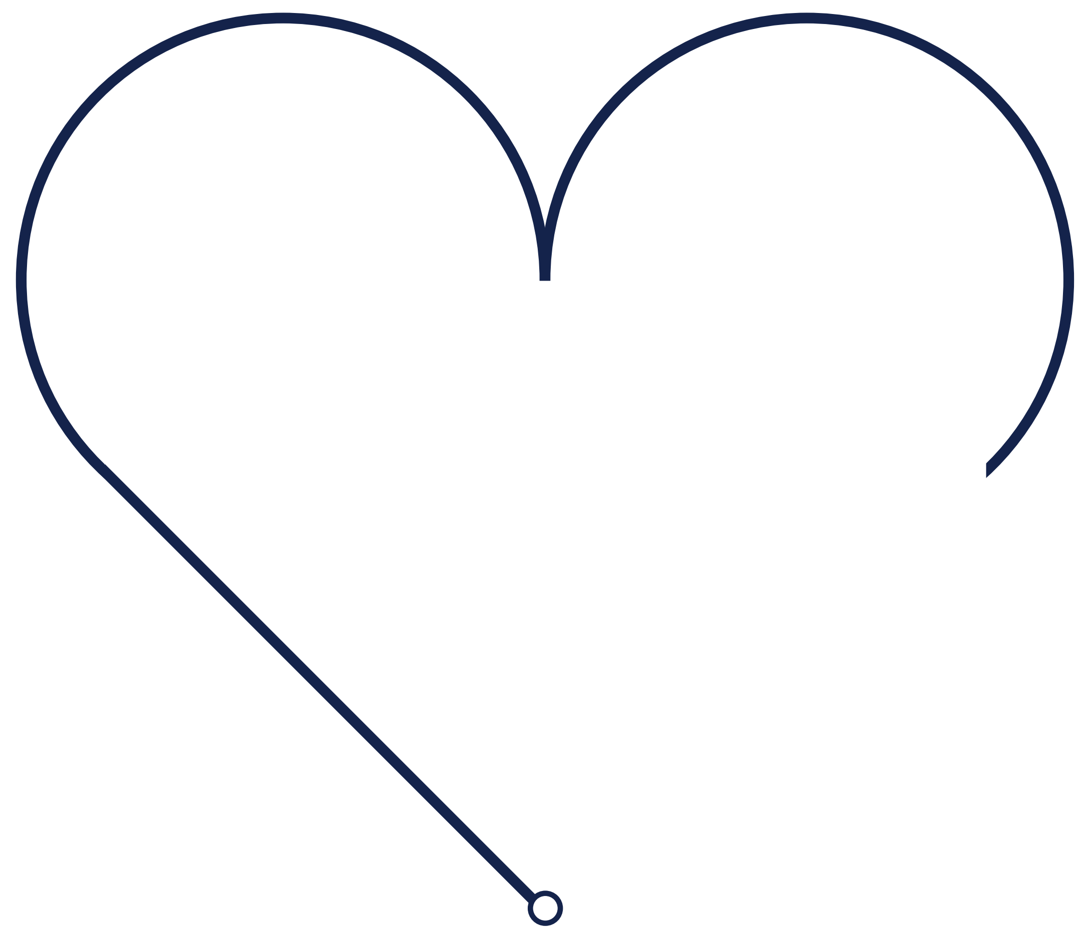


```
LINE SEGMENT:
  is_construction:
```

Interpreter-guided generation

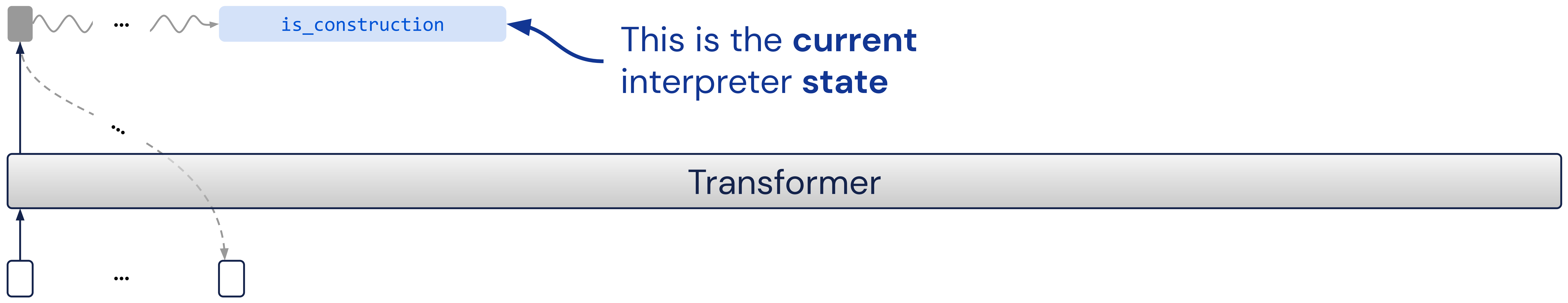


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```



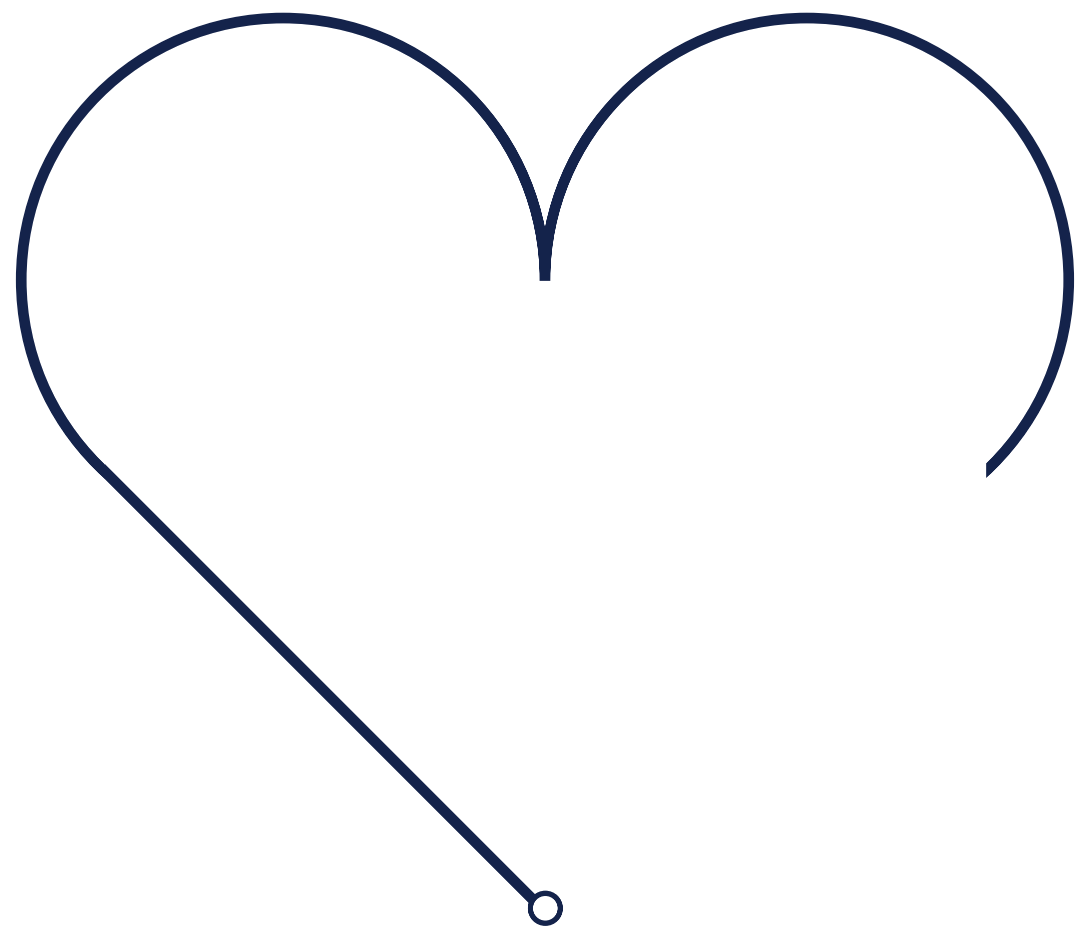
```
LINE SEGMENT:
  is_construction:
```

Interpreter-guided generation



This is the **current** interpreter **state**

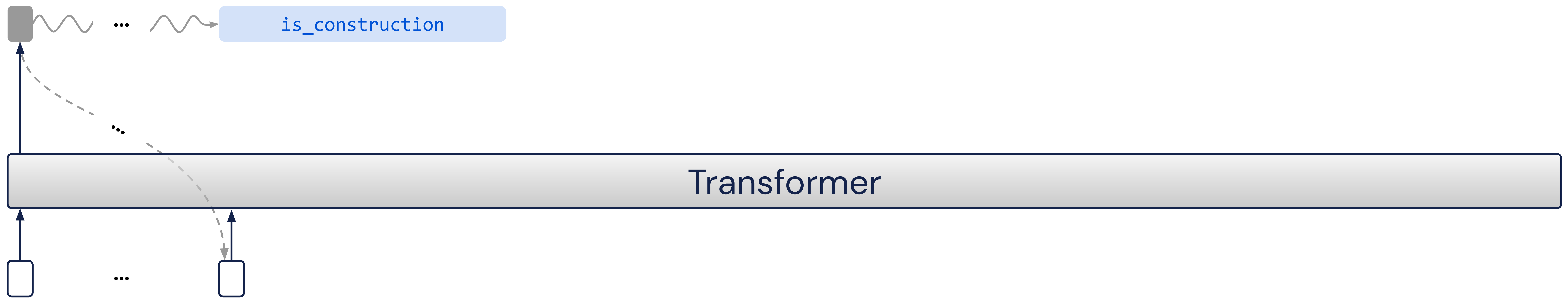
```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```



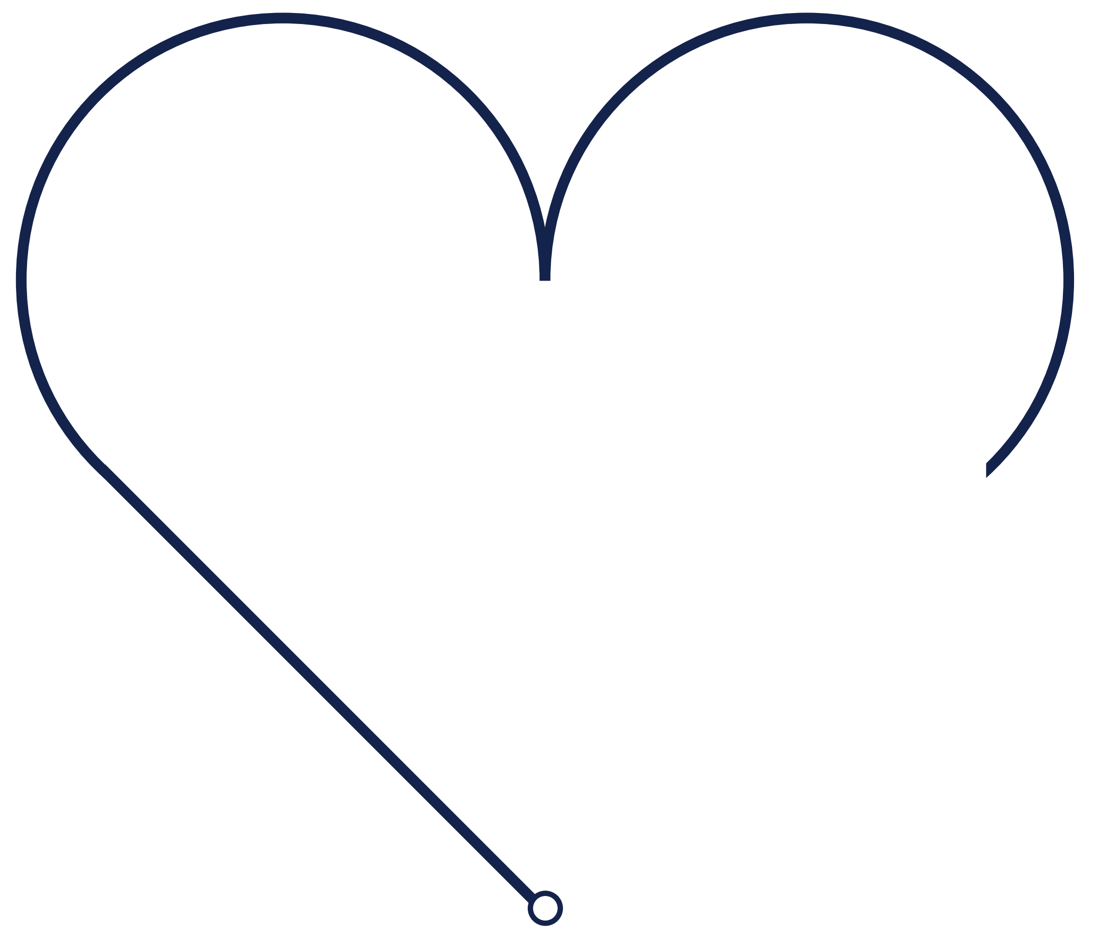
```
LINE SEGMENT:
  is_construction:
```

Interpreter-guided generation

Private & Confidential

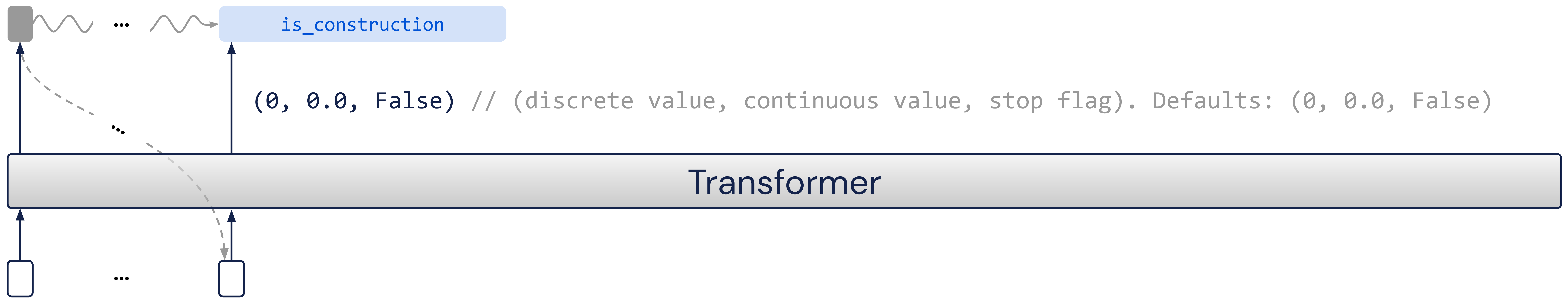


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```



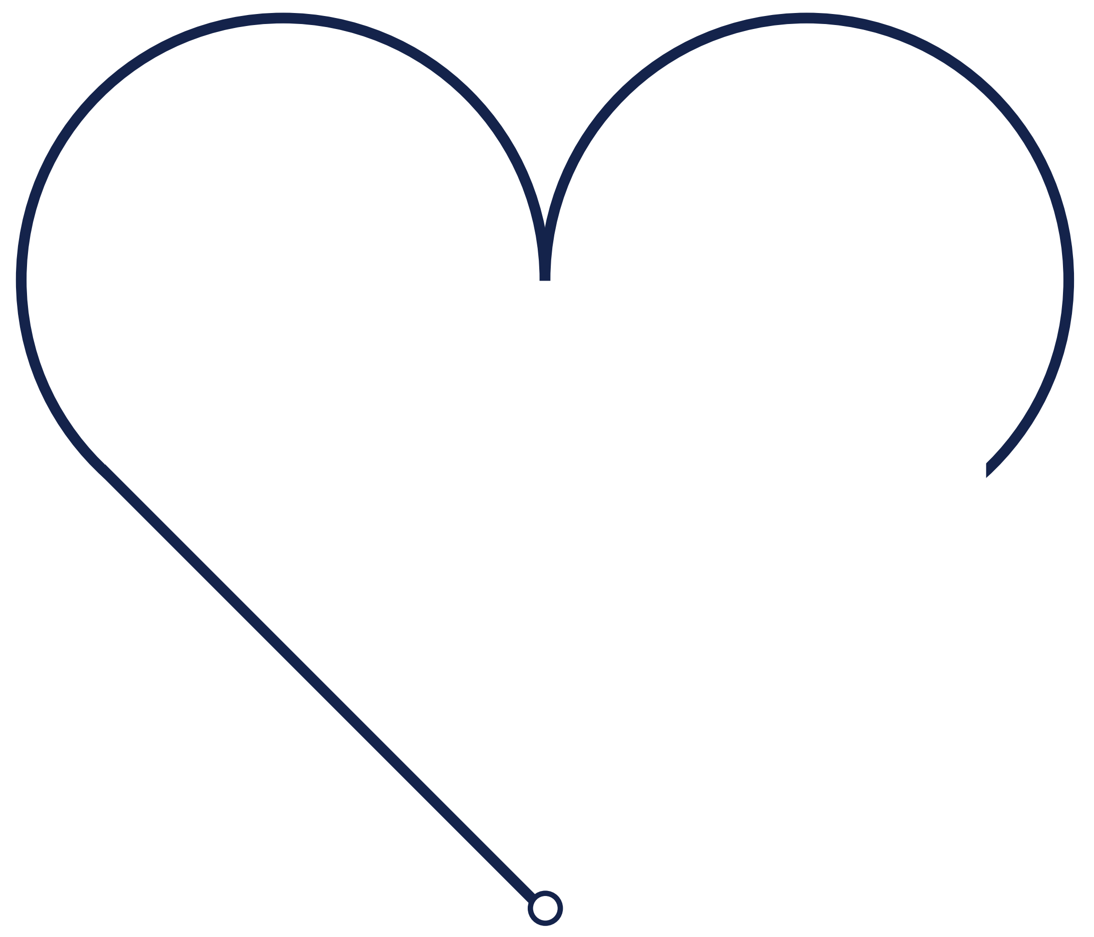
```
LINE SEGMENT:
  is_construction:
```

Interpreter-guided generation



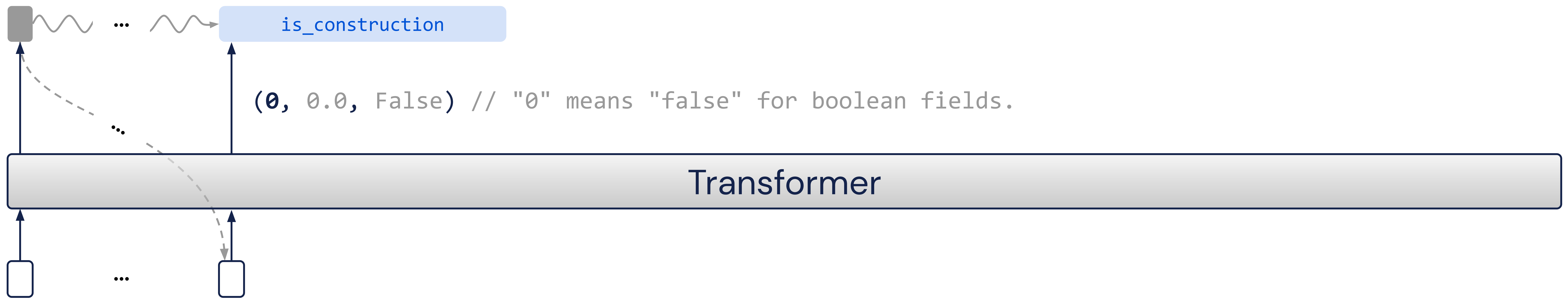
```
(0, 0.0, False) // (discrete value, continuous value, stop flag). Defaults: (0, 0.0, False)
```

```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```



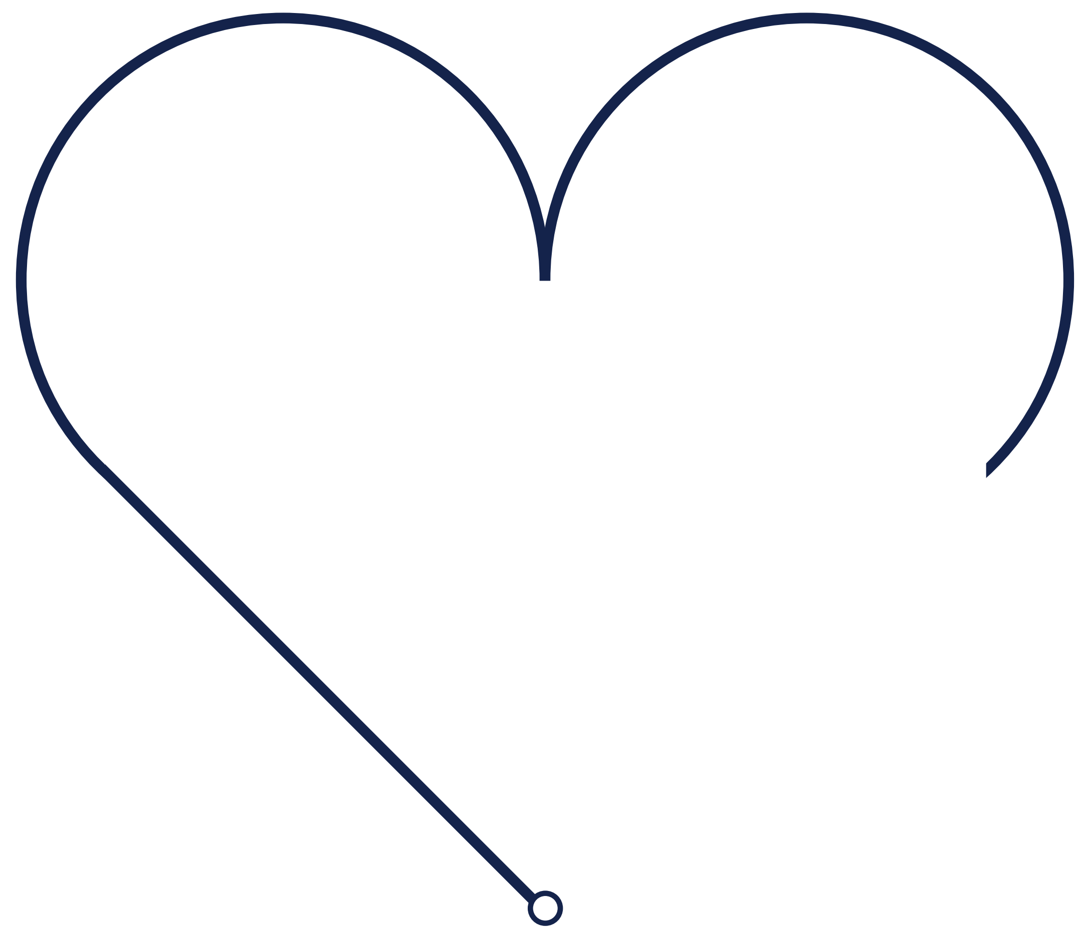
```
LINE SEGMENT:
  is_construction:
```

Interpreter-guided generation



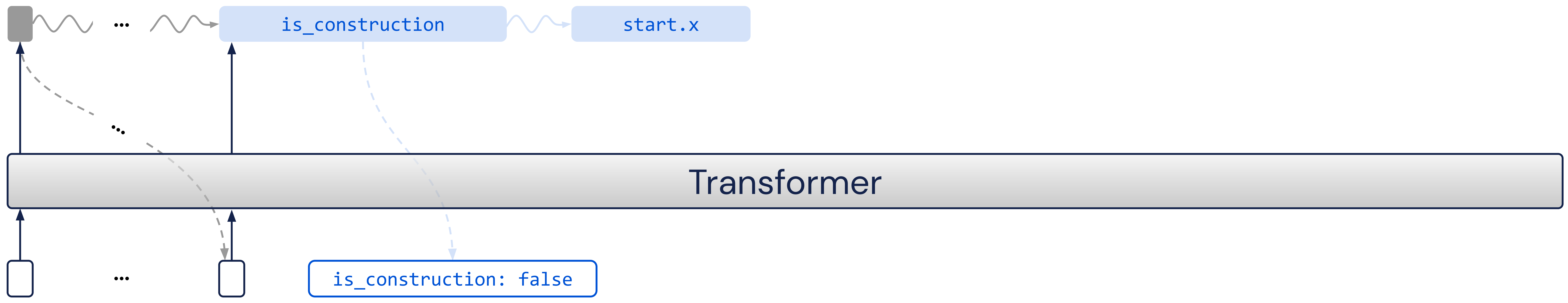
```
(0, 0.0, False) // "0" means "false" for boolean fields.
```

```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```

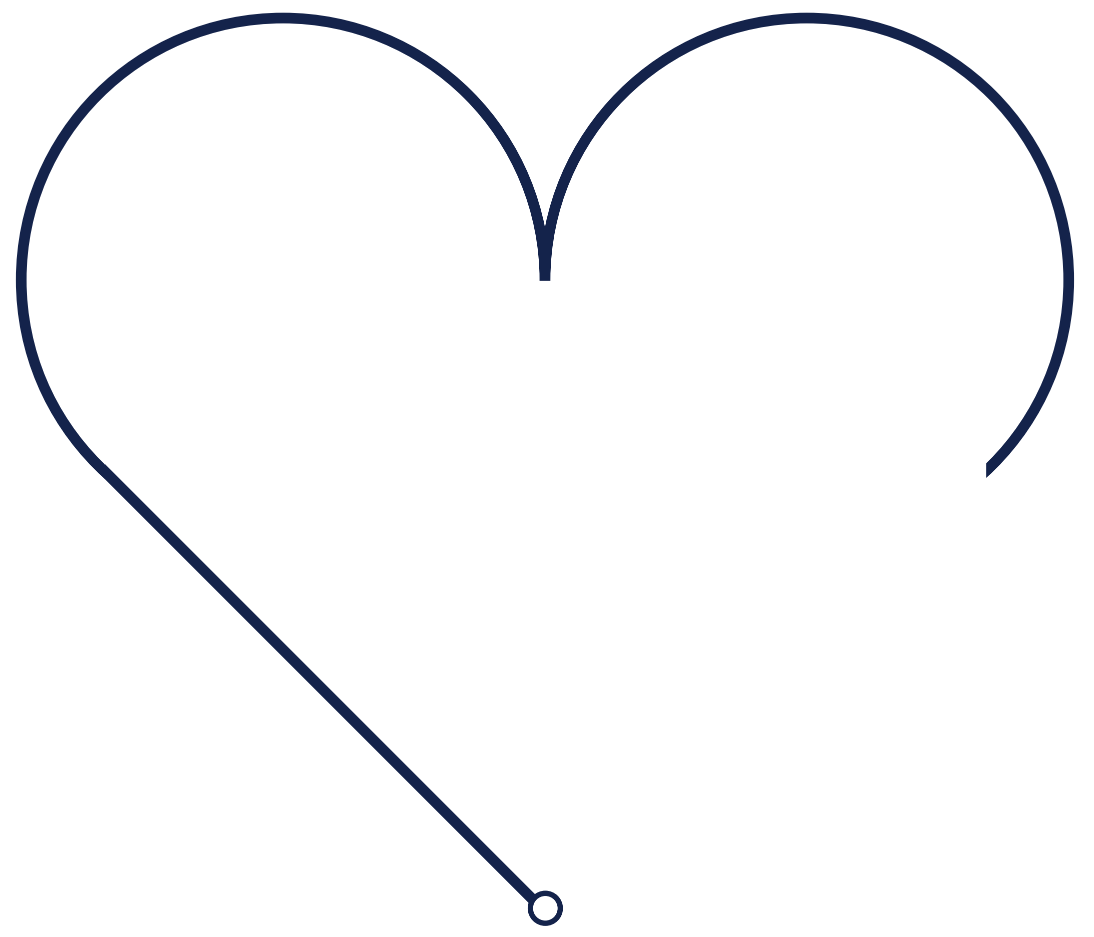


```
LINE SEGMENT:
  is_construction:
```

Interpreter-guided generation

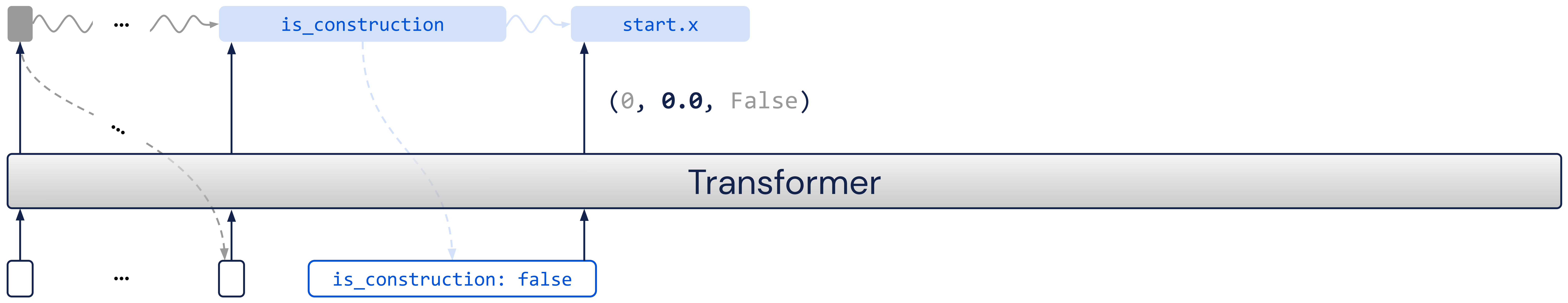


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```

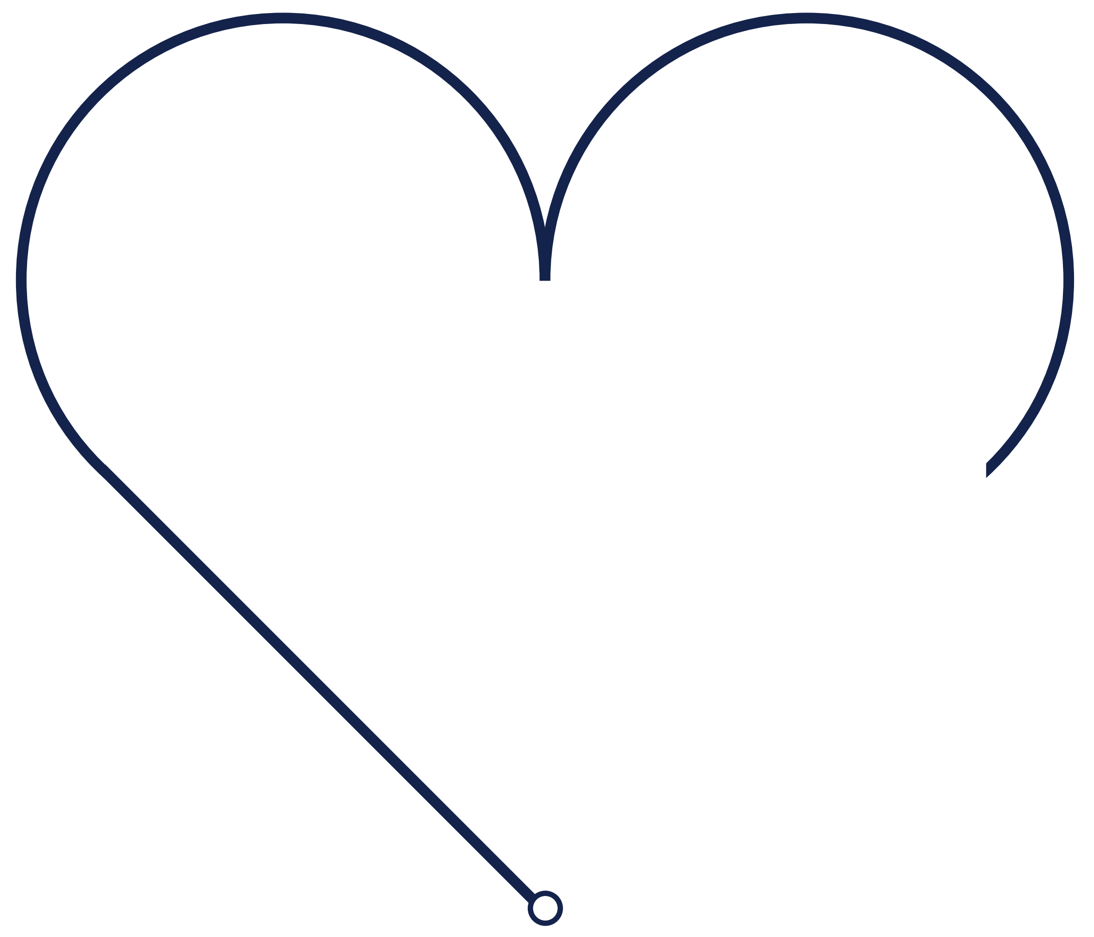


```
LINE SEGMENT:
  is_construction: false
  start:
    x:
```

Interpreter-guided generation

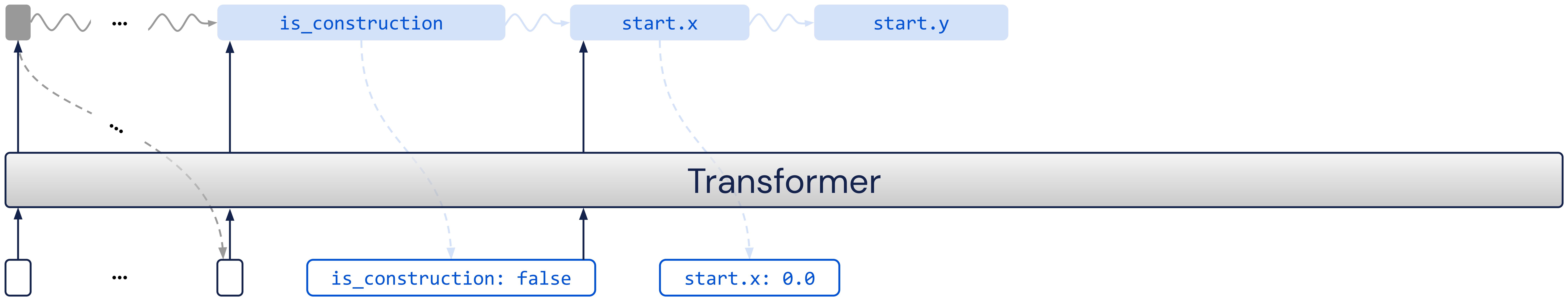


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```

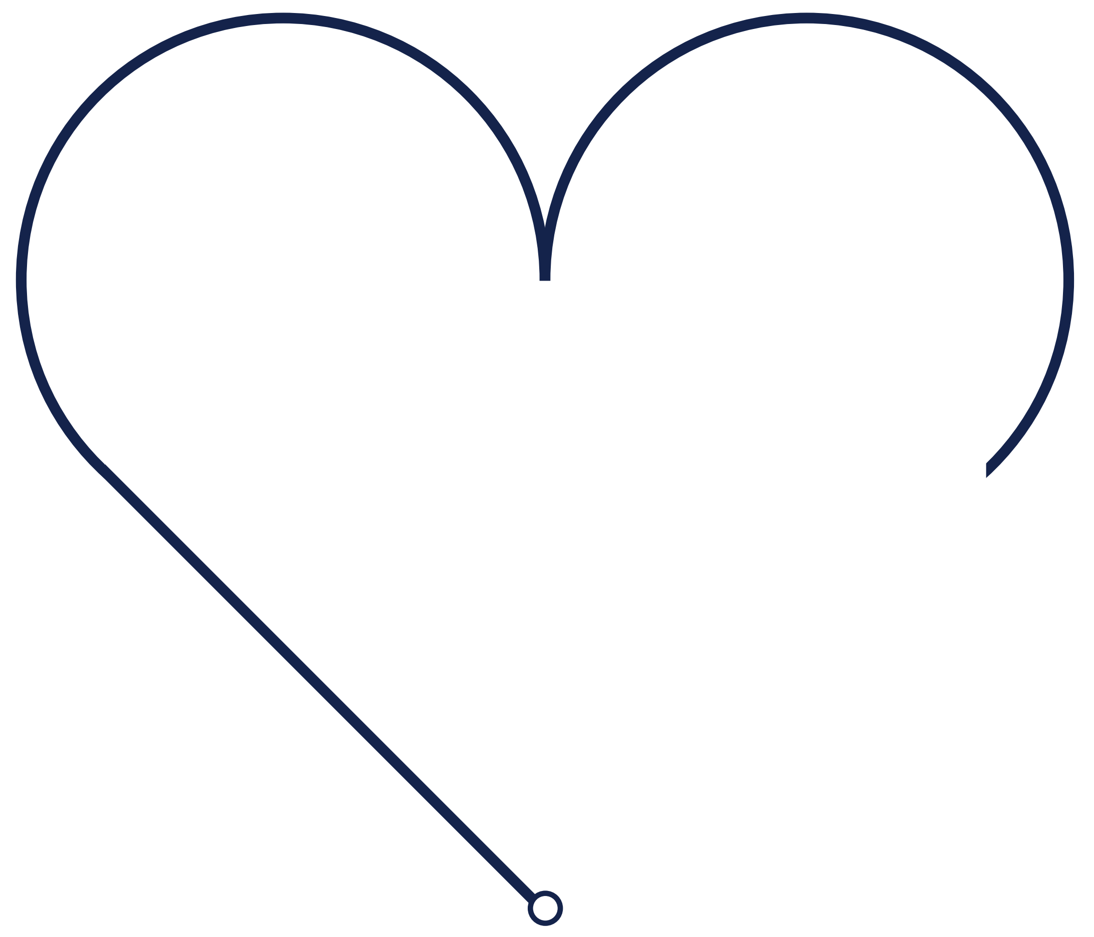


```
LINE SEGMENT:
  is_construction: false
  start:
    x:
```

Interpreter-guided generation

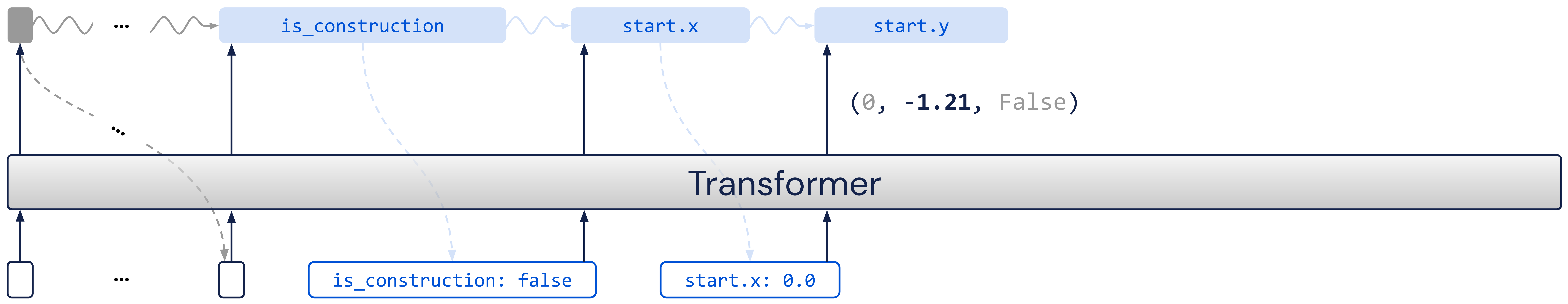


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```

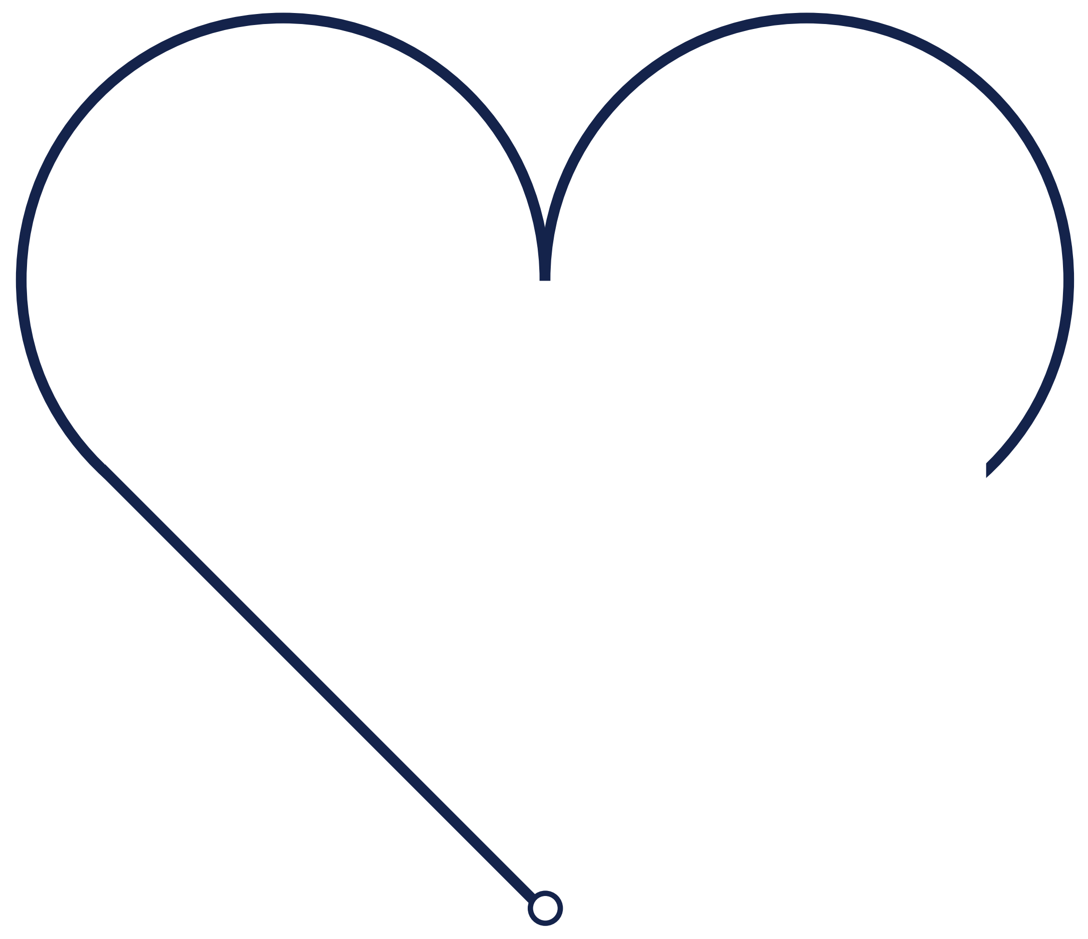


```
LINE SEGMENT:
  is_construction: false
  start:
    x: 0.0
    y:
```

Interpreter-guided generation

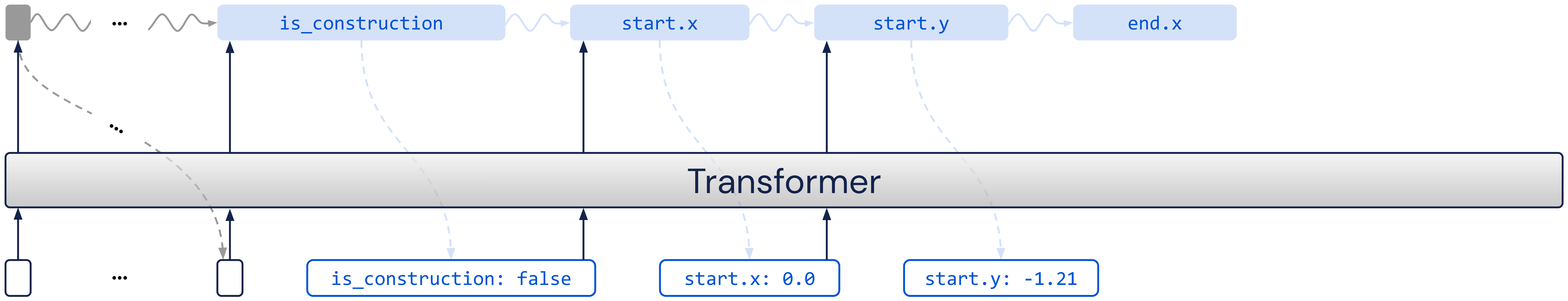


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```

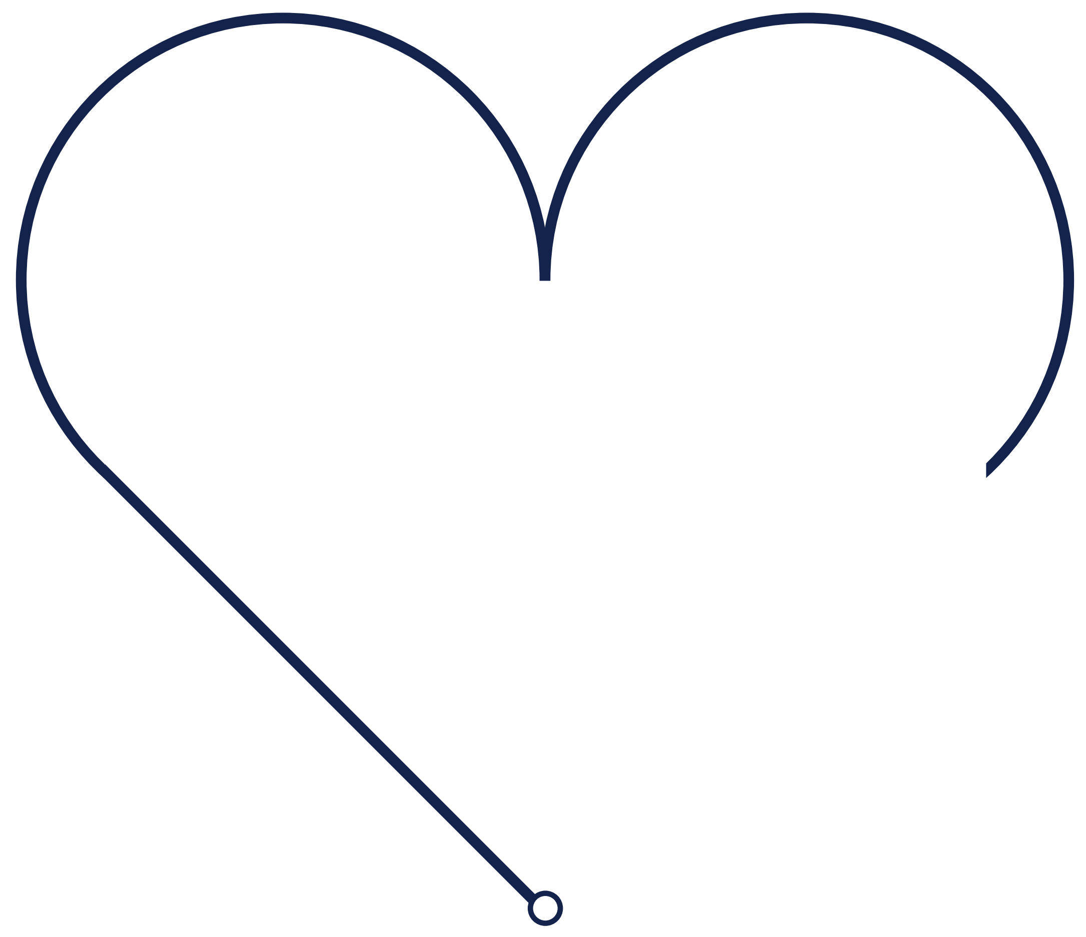


```
LINE SEGMENT:
  is_construction: false
  start:
    x: 0.0
    y:
```

Interpreter-guided generation

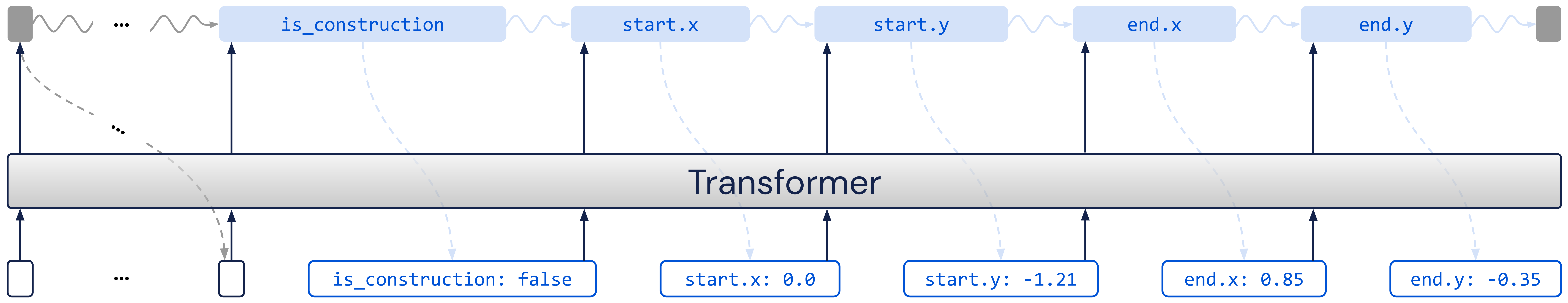


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```

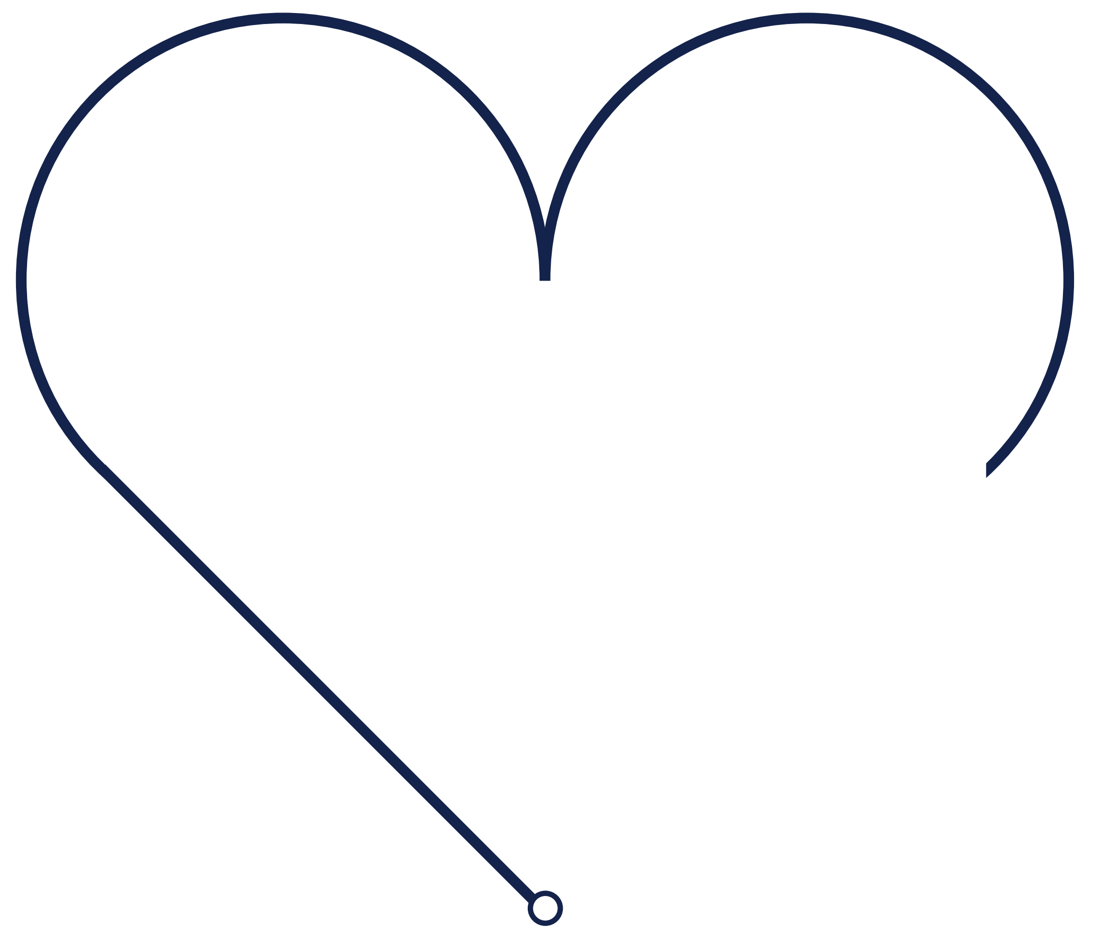


```
LINE SEGMENT:
  is_construction: false
  start:
    x: 0.0
    y: -1.21
  end:
    x:
```

Interpreter-guided generation

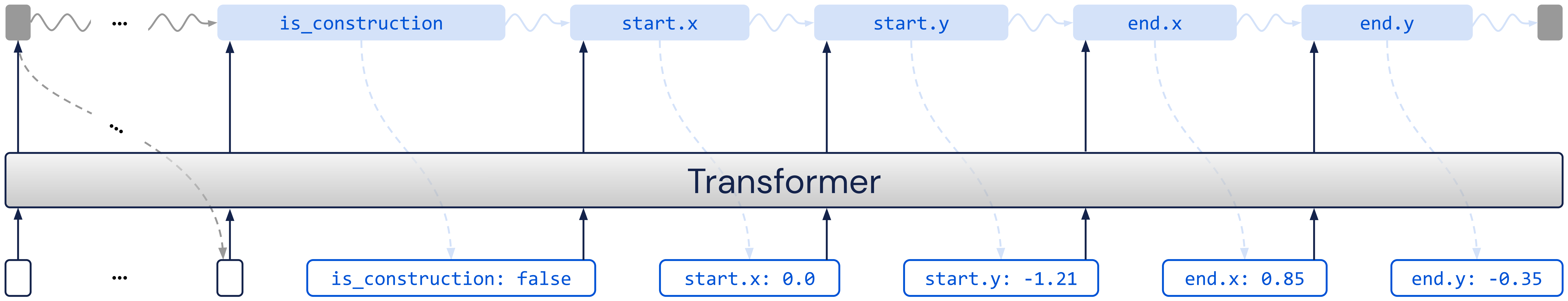


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```

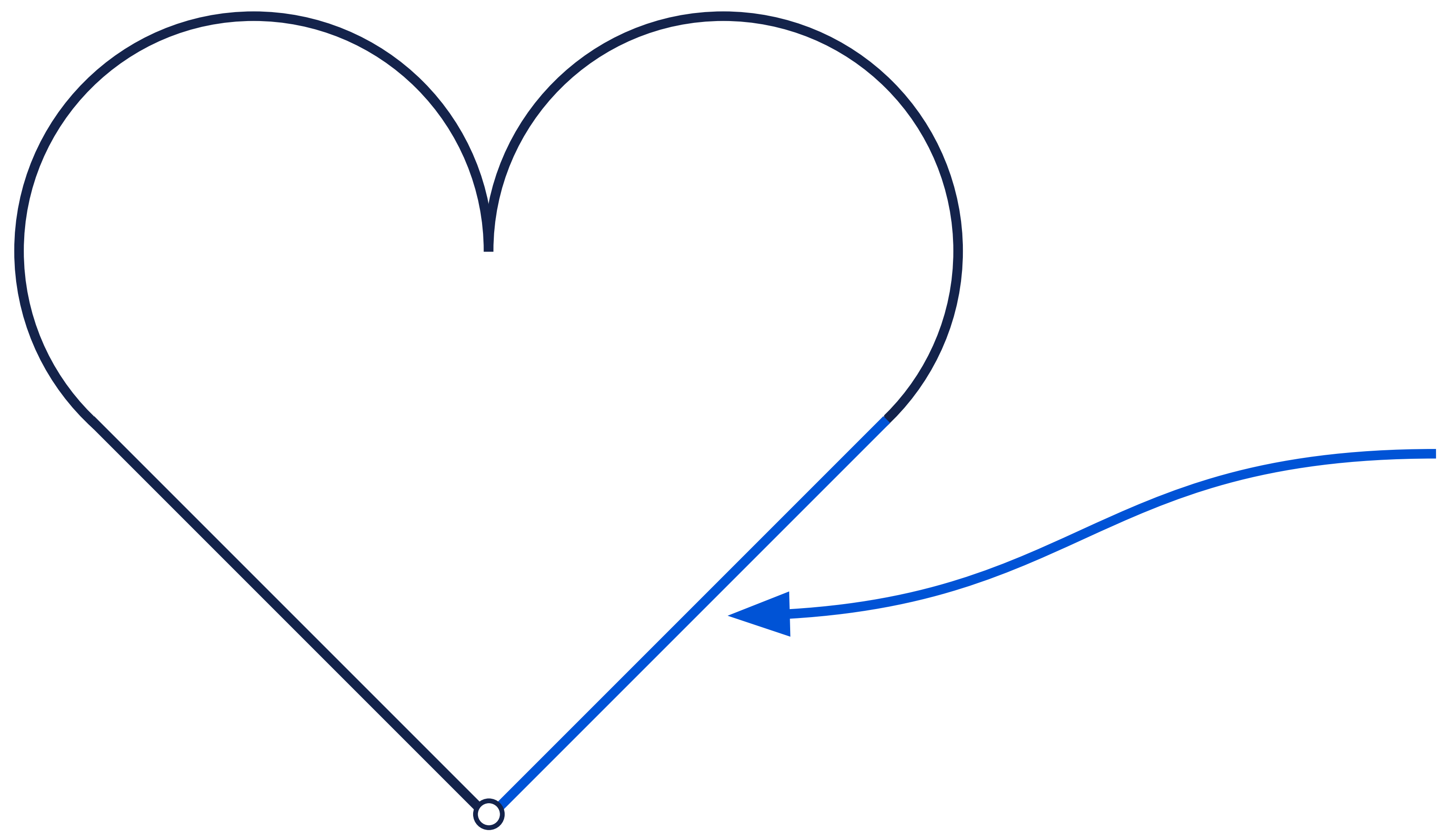


```
LINE SEGMENT:
  is_construction: false
  start:
    x: 0.0
    y: -1.21
  end:
    x: 0.85
    y: -0.35
```

Interpreter-guided generation

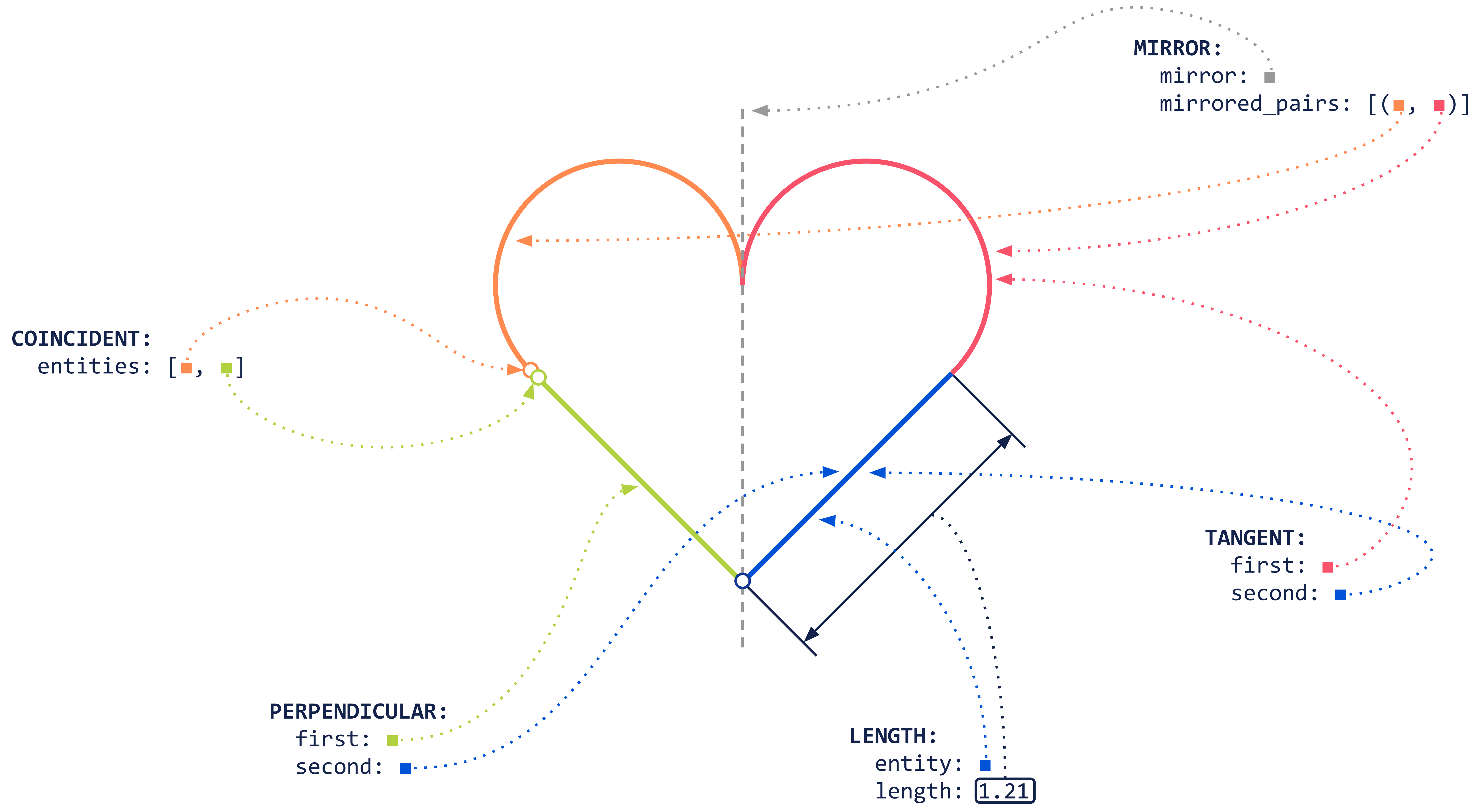


```
message Vector {
  double x = 1;
  double y = 2;
}

message LineEntity {
  bool is_construction = 1;
  Vector start = 2;
  Vector end = 3;
}
```

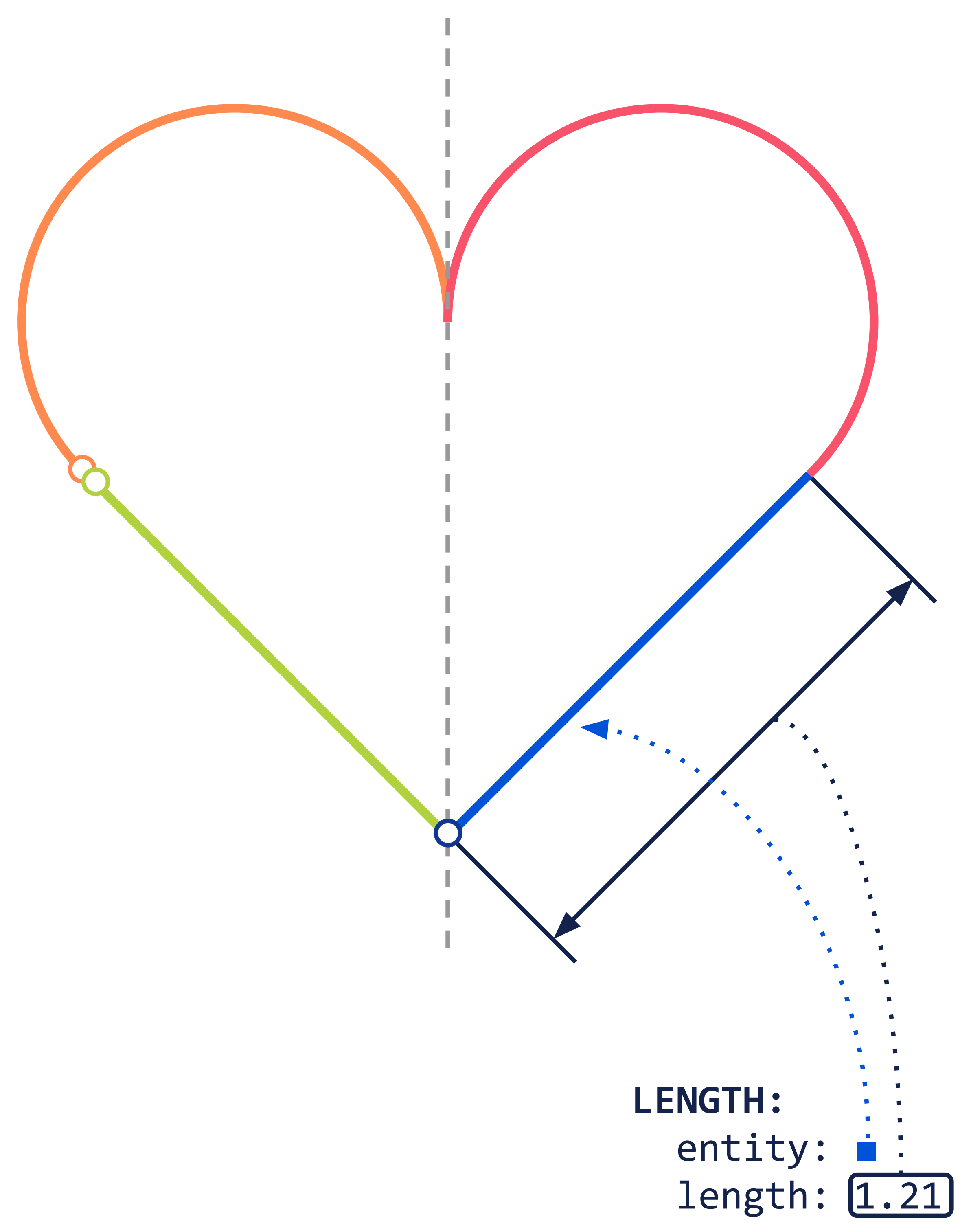


```
LINE SEGMENT:
  is_construction: false
  start:
    x: 0.0
    y: -1.21
  end:
    x: 0.85
    y: -0.35
```

Constraints

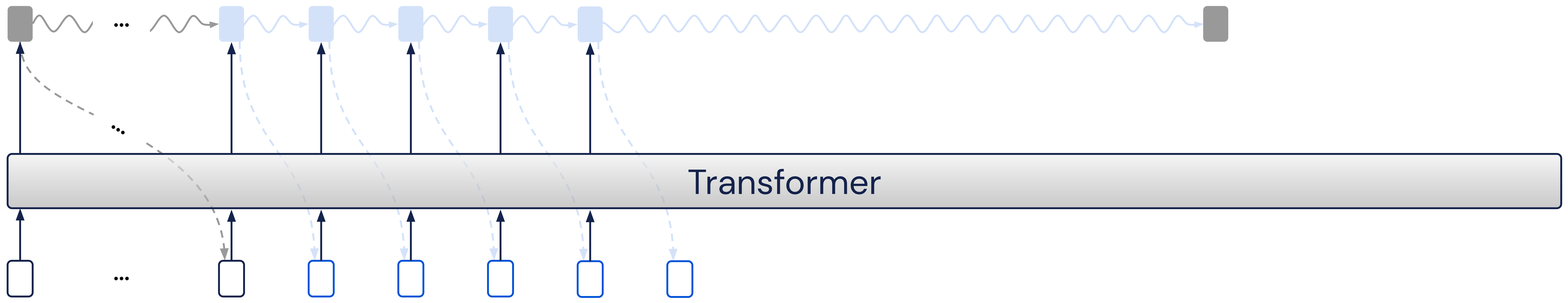


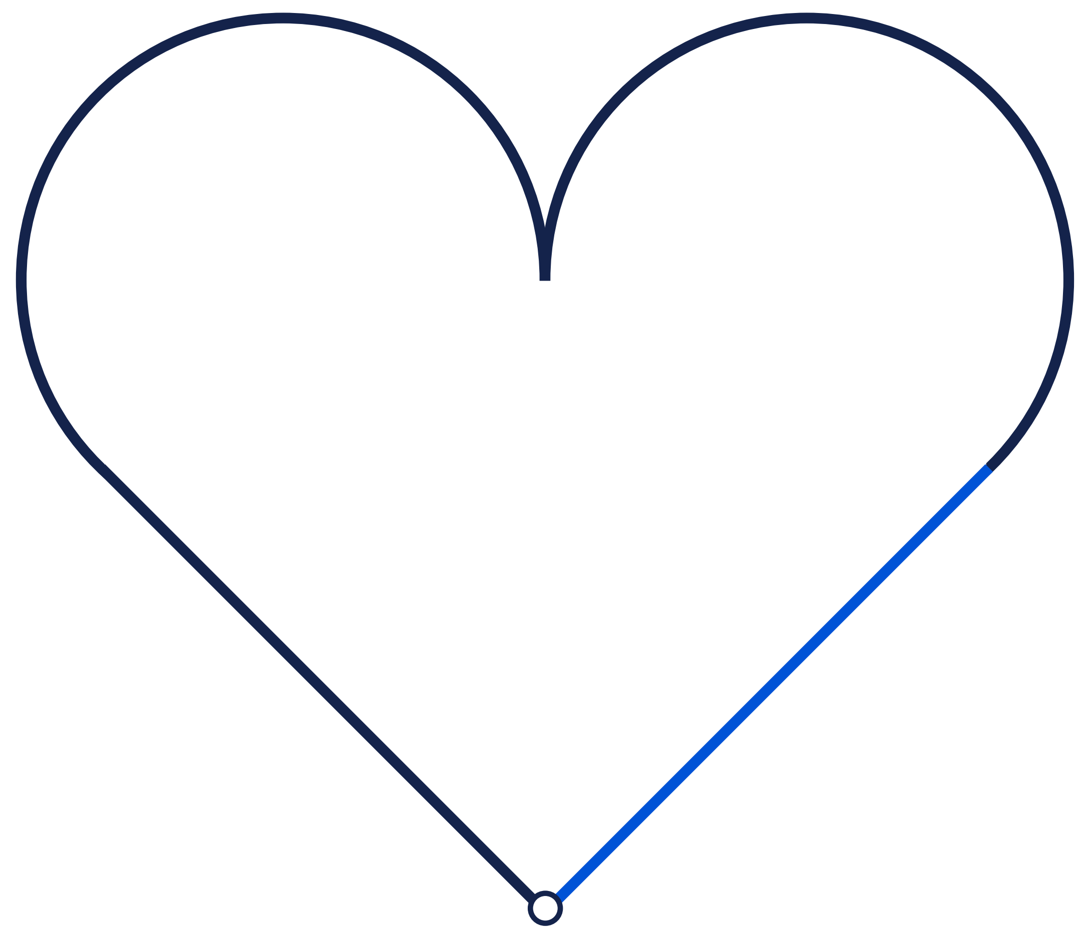
Constraints



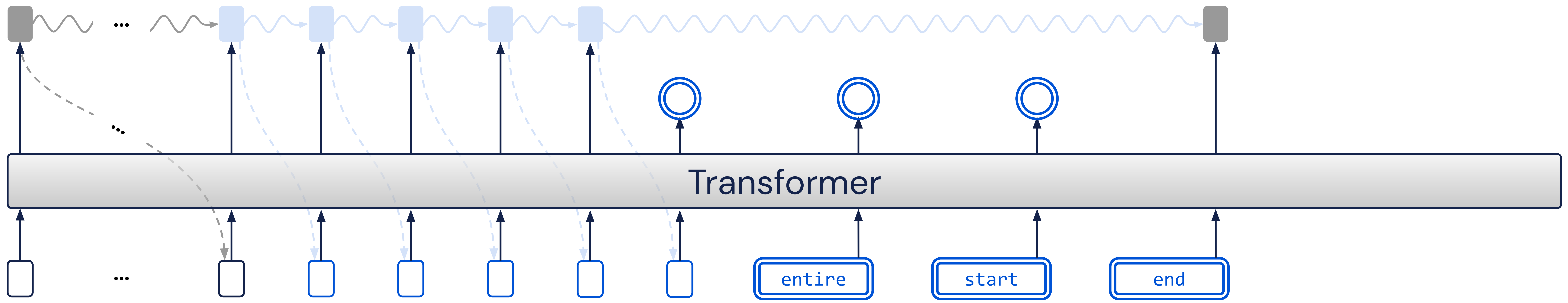
```
LENGTH:
  entity: ■
  length: 1.21
```

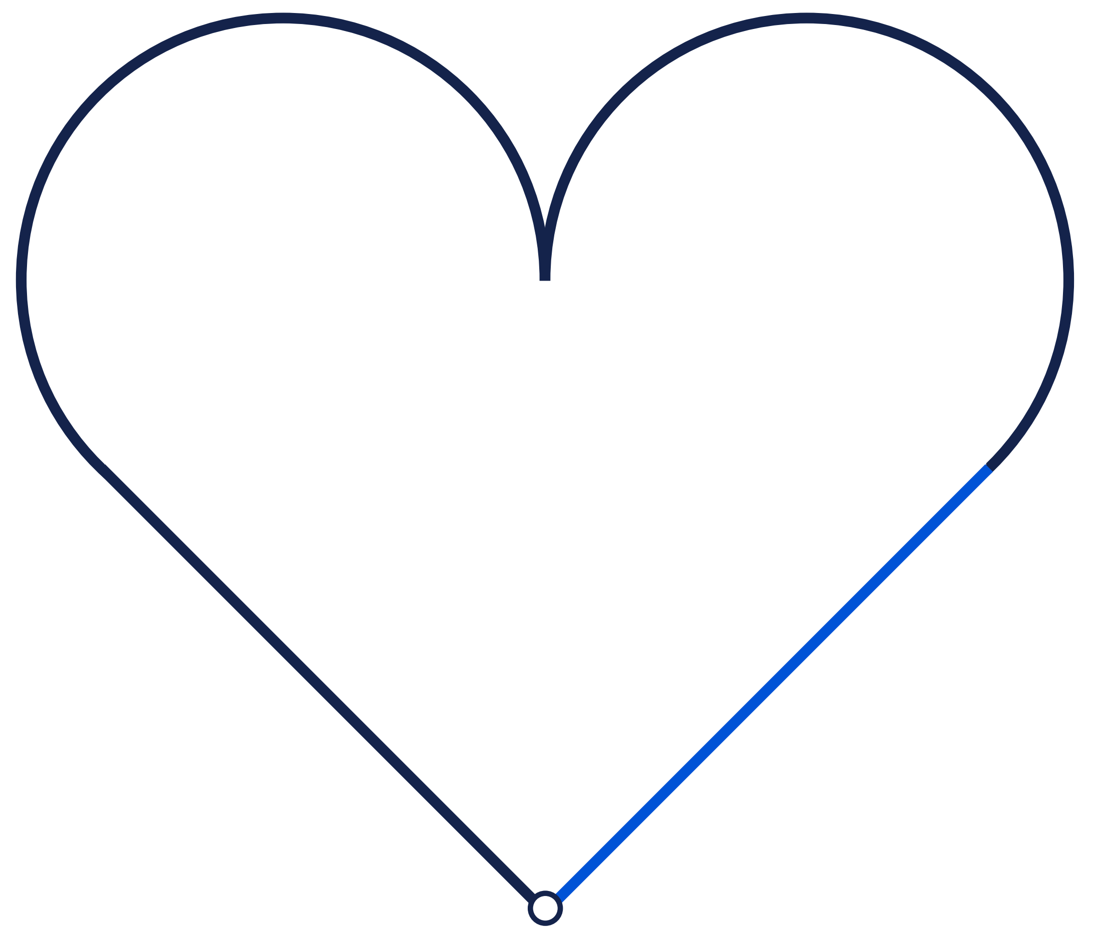
Interpreter-guided generation



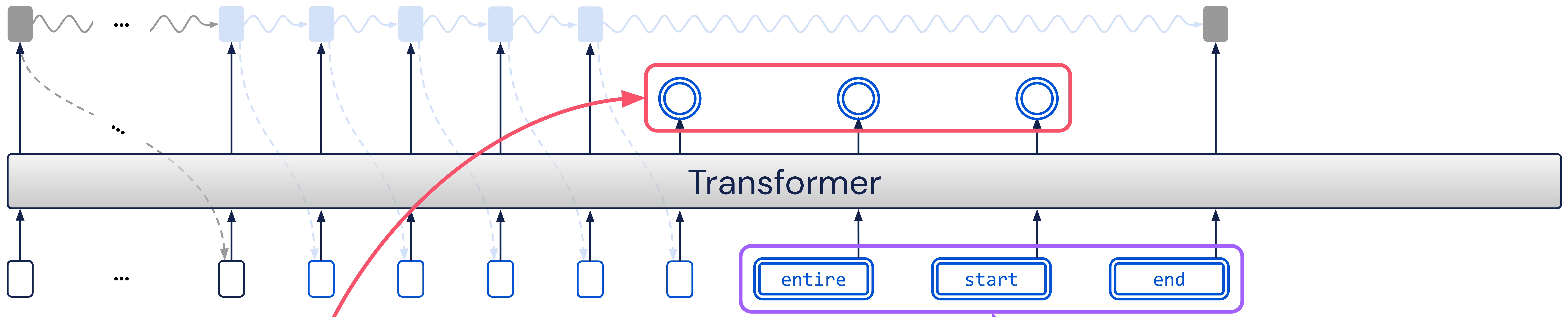


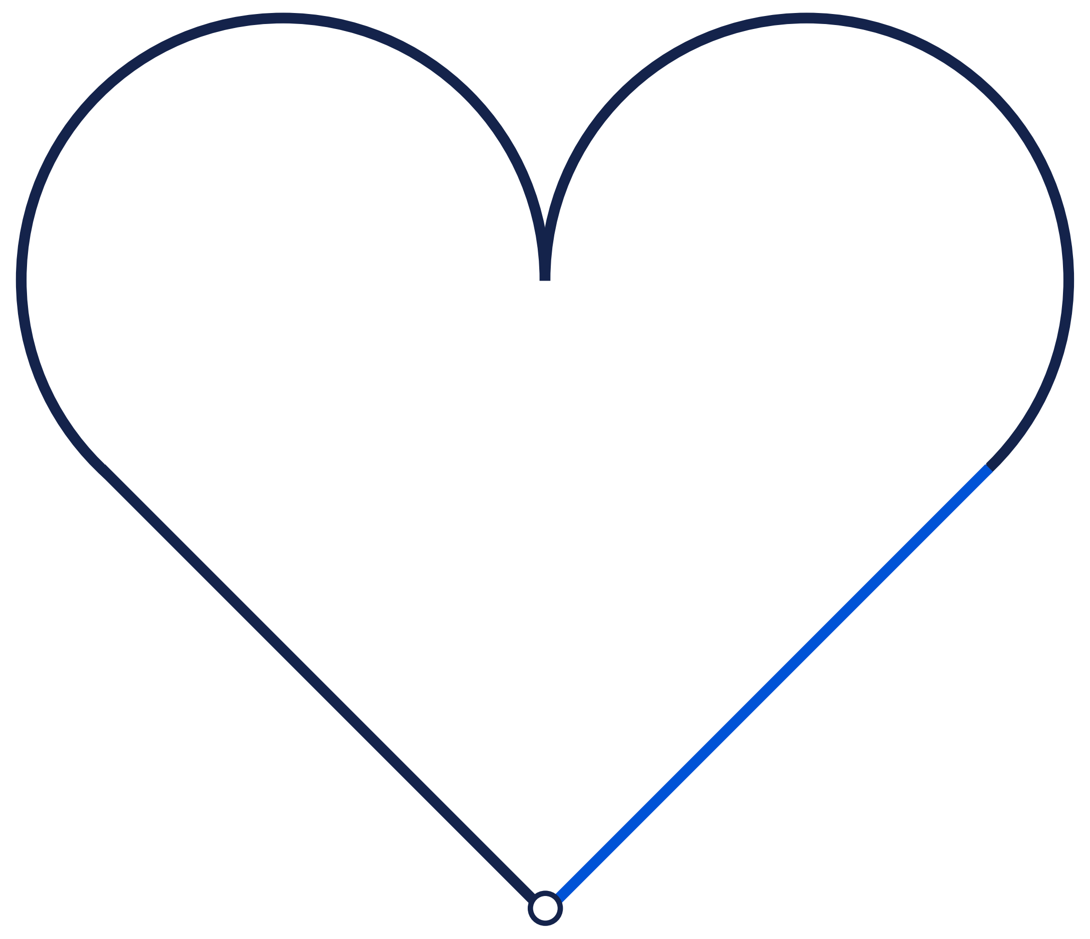
Interpreter-guided generation





Interpreter-guided generation

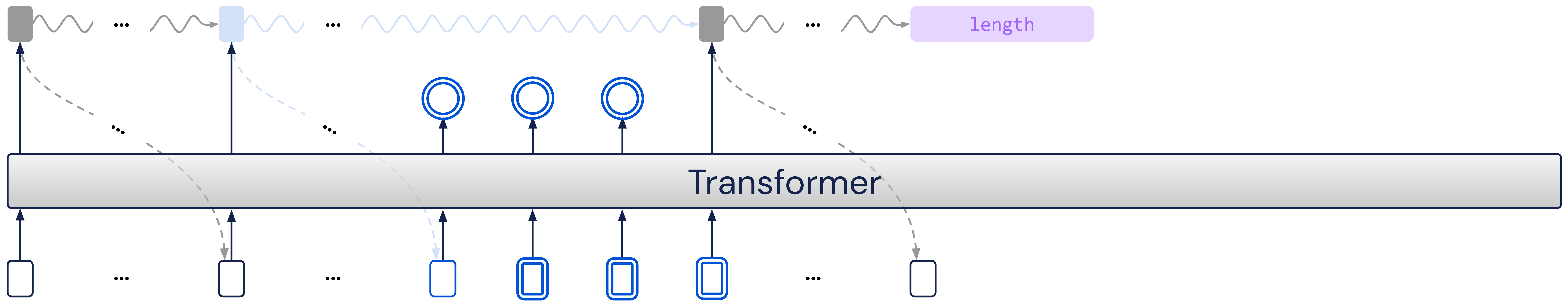




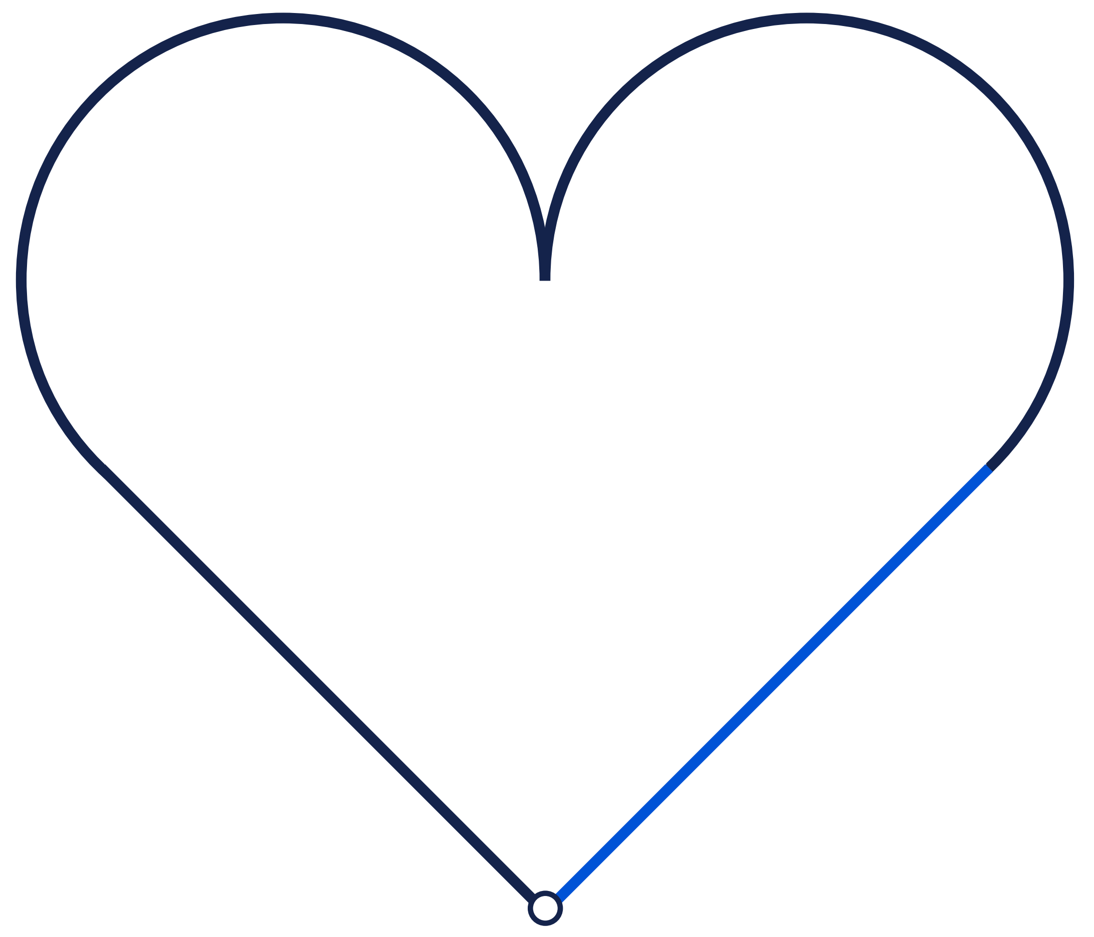
Representations of different line **parts**

These are **not predicted** – they are known in advance for every object type

Interpreter-guided generation



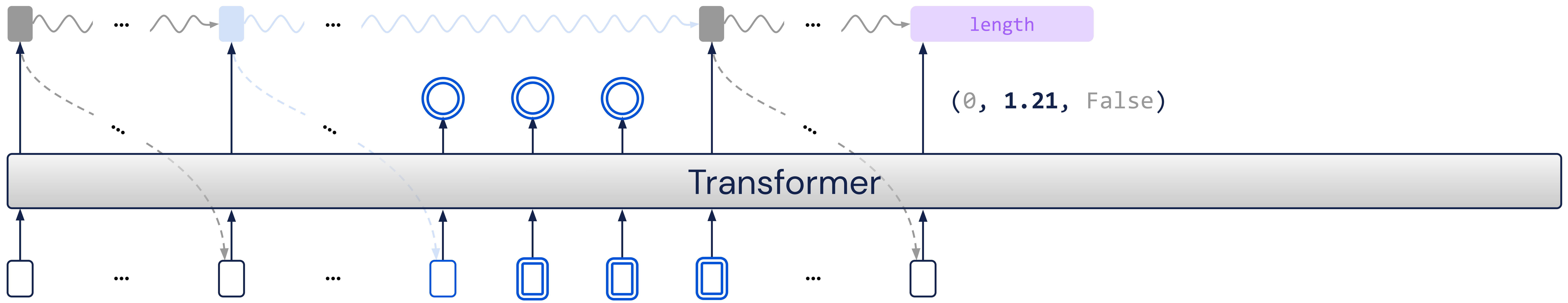
```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```

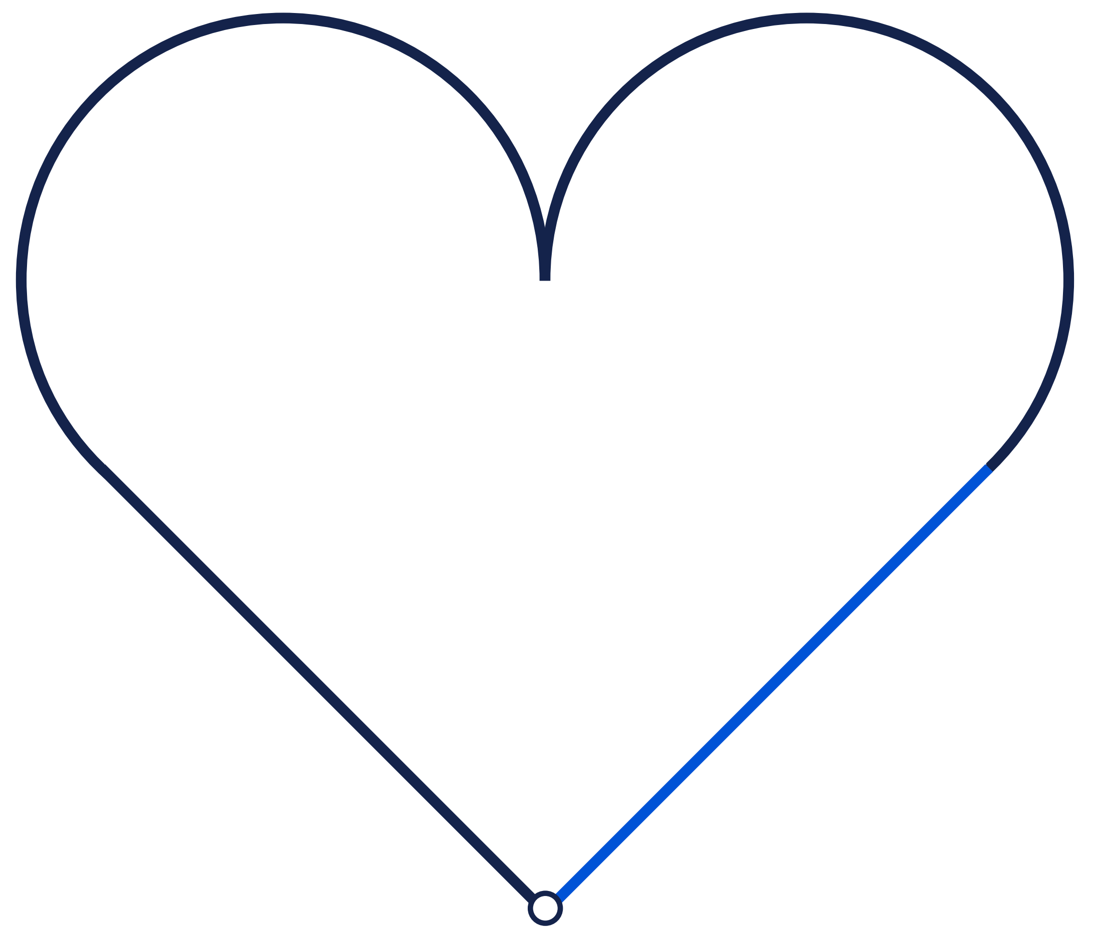


```
LENGTH:
  length:
```

Interpreter-guided generation

Private & Confidential

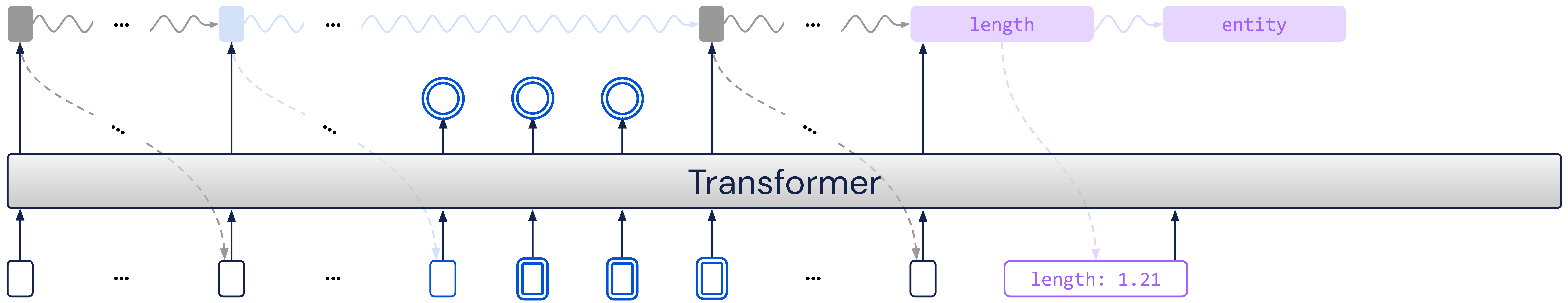


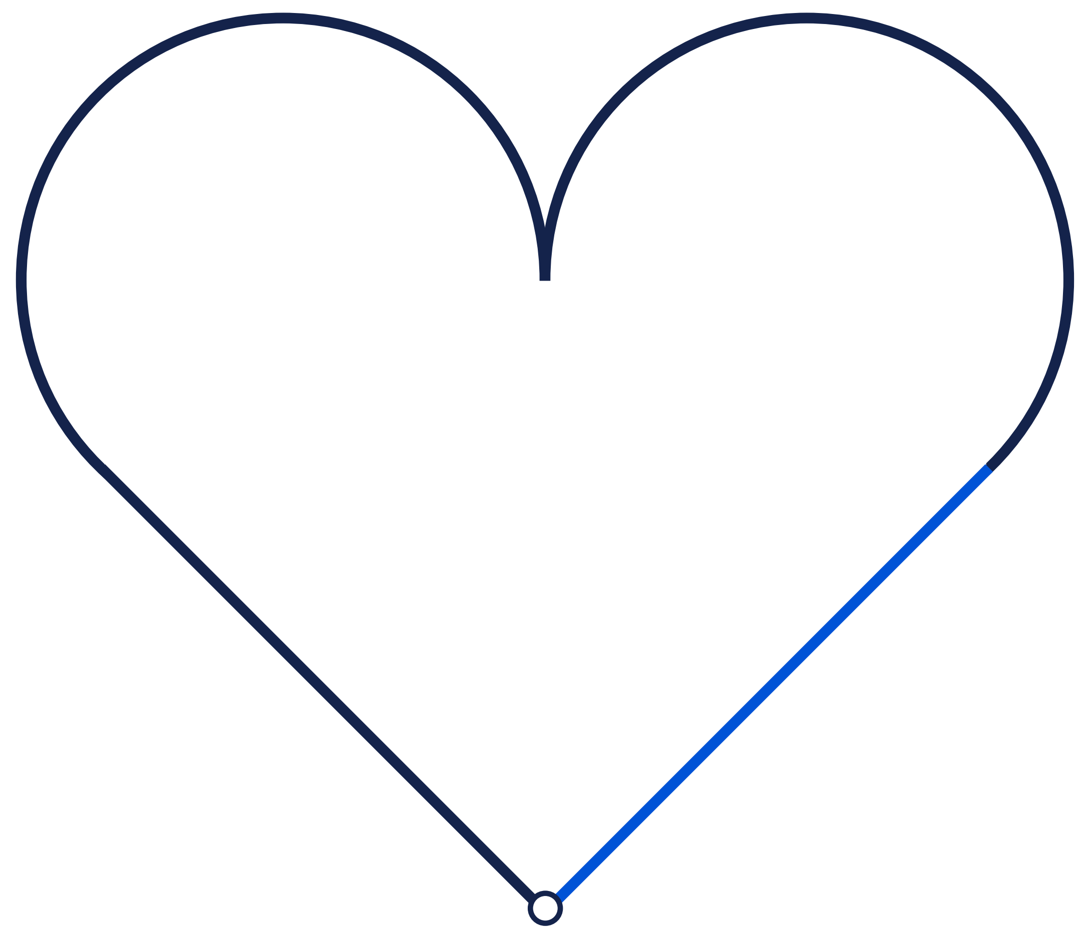


```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```

```
LENGTH:
  length:
```

Interpreter-guided generation

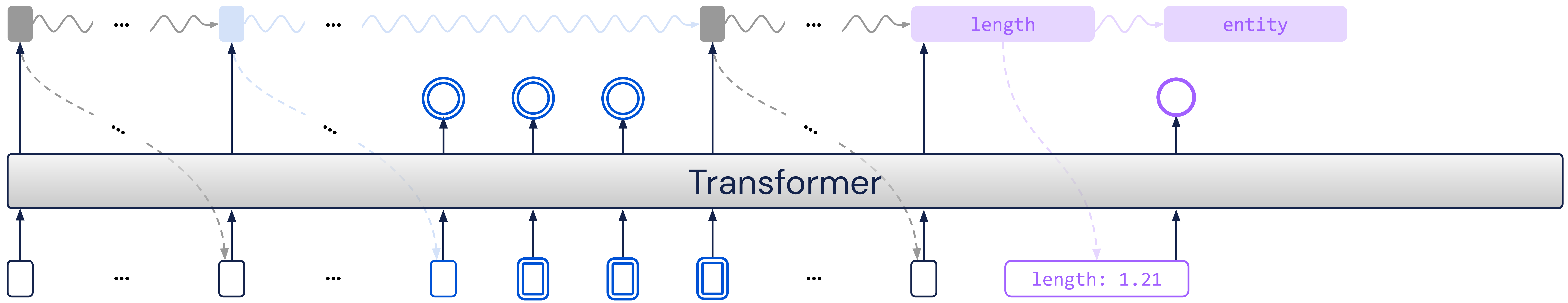




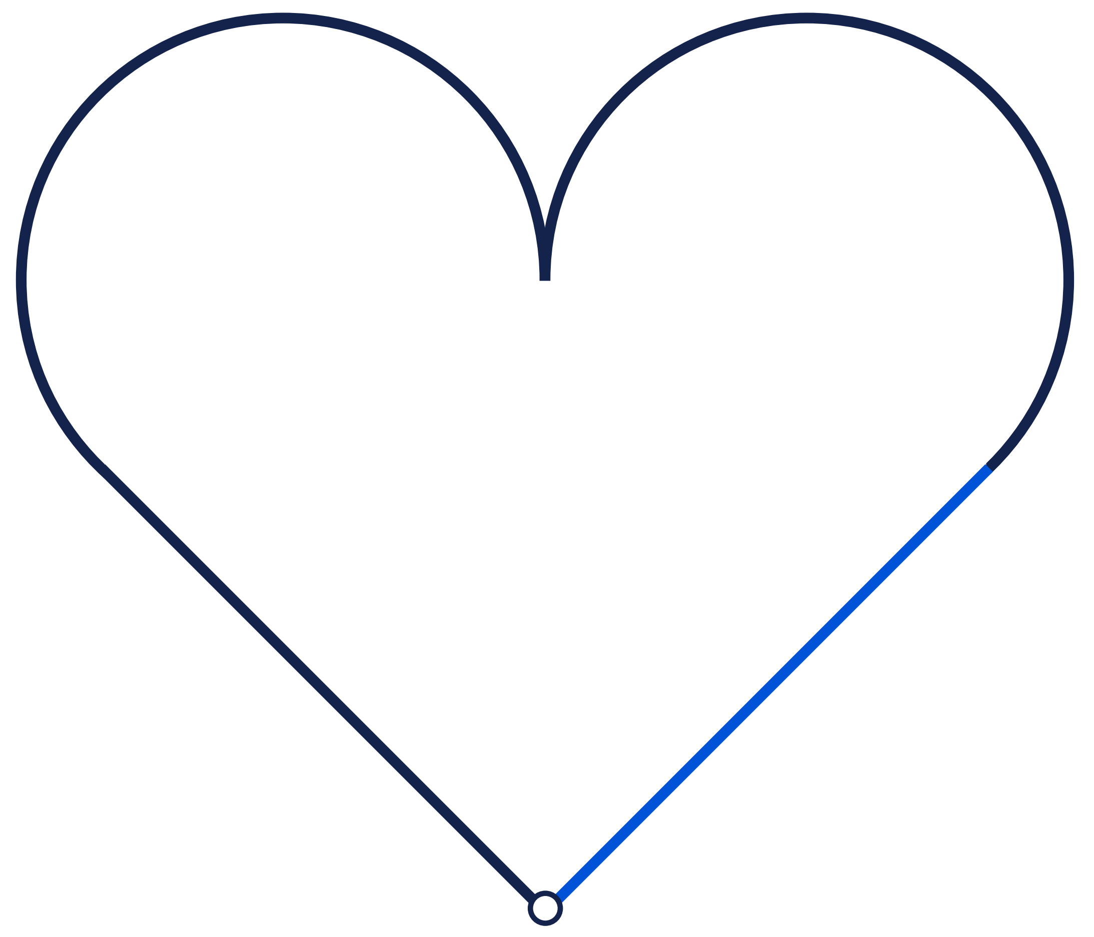
```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```

```
LENGTH:
  length: 1.21
  entity:
```

Interpreter-guided generation

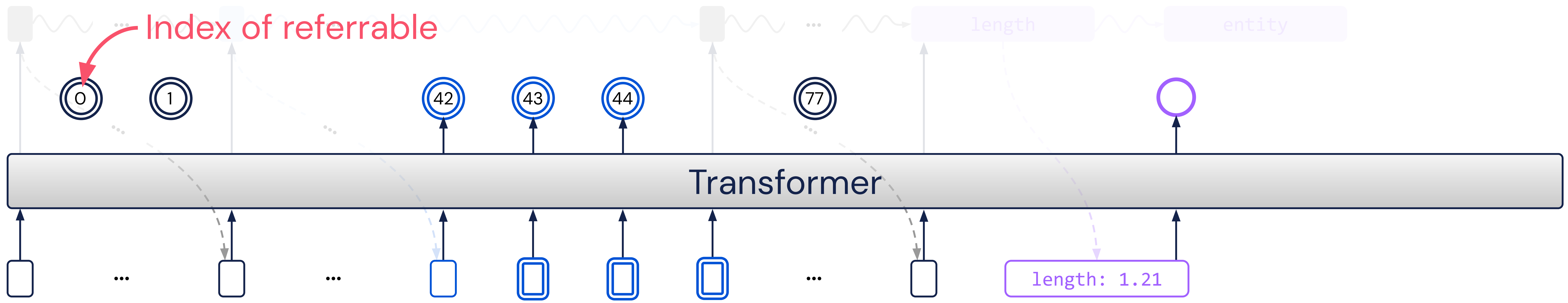


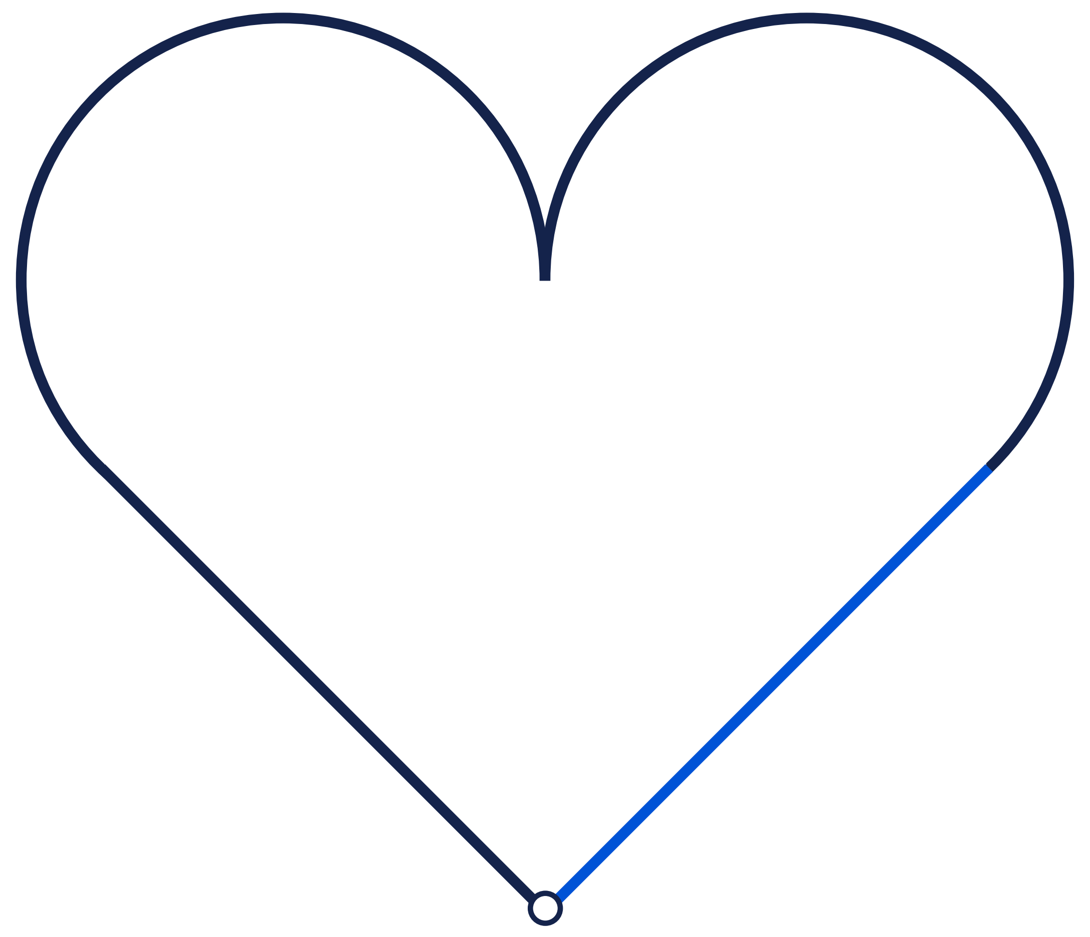
```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```



```
LENGTH:
  length: 1.21
  entity:
```

Interpreter-guided generation

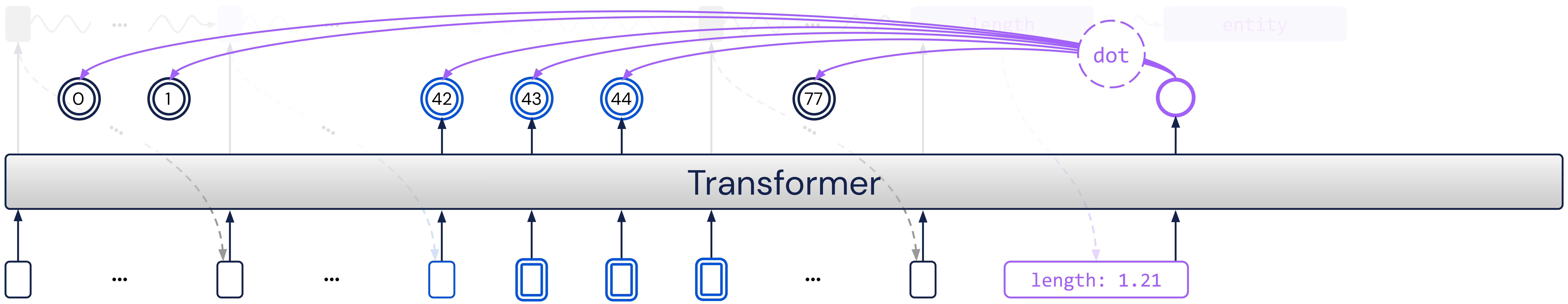


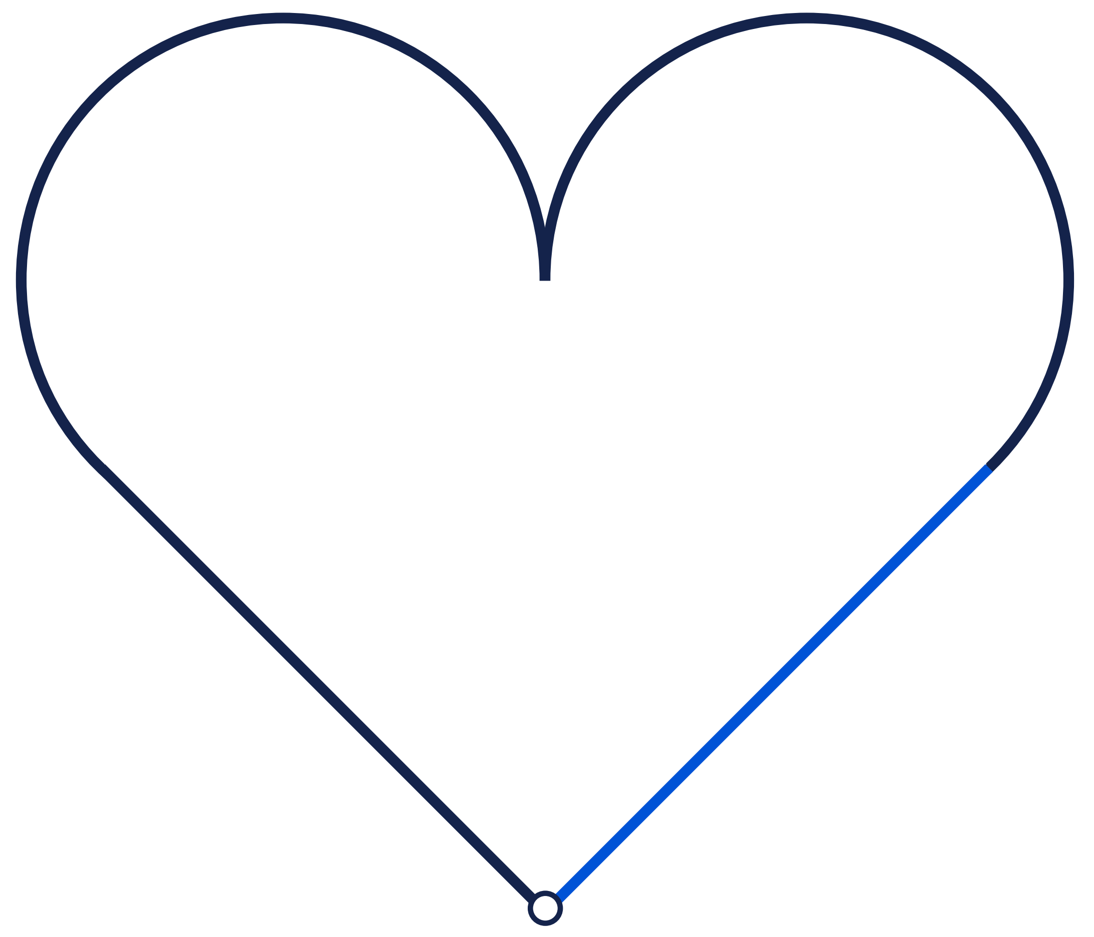


```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```

```
LENGTH:
  length: 1.21
  entity:
```

Interpreter-guided generation

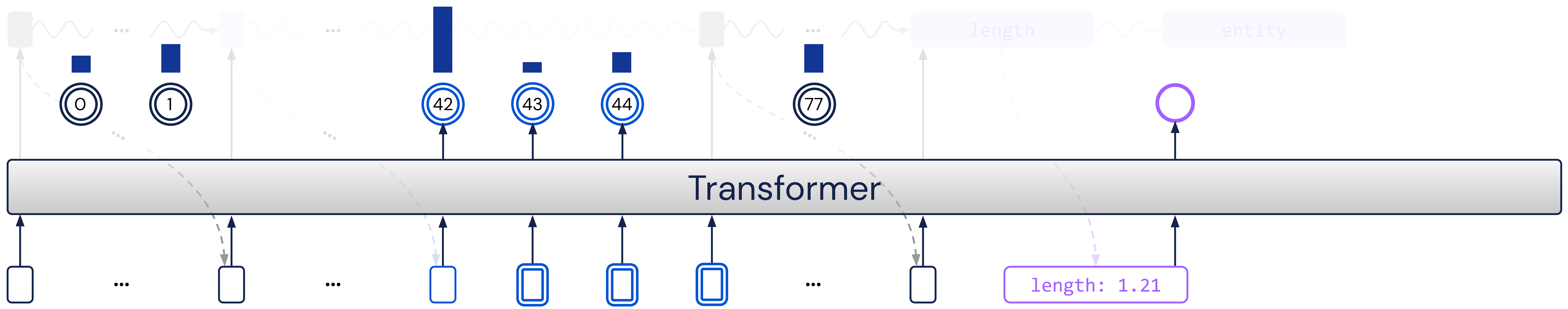


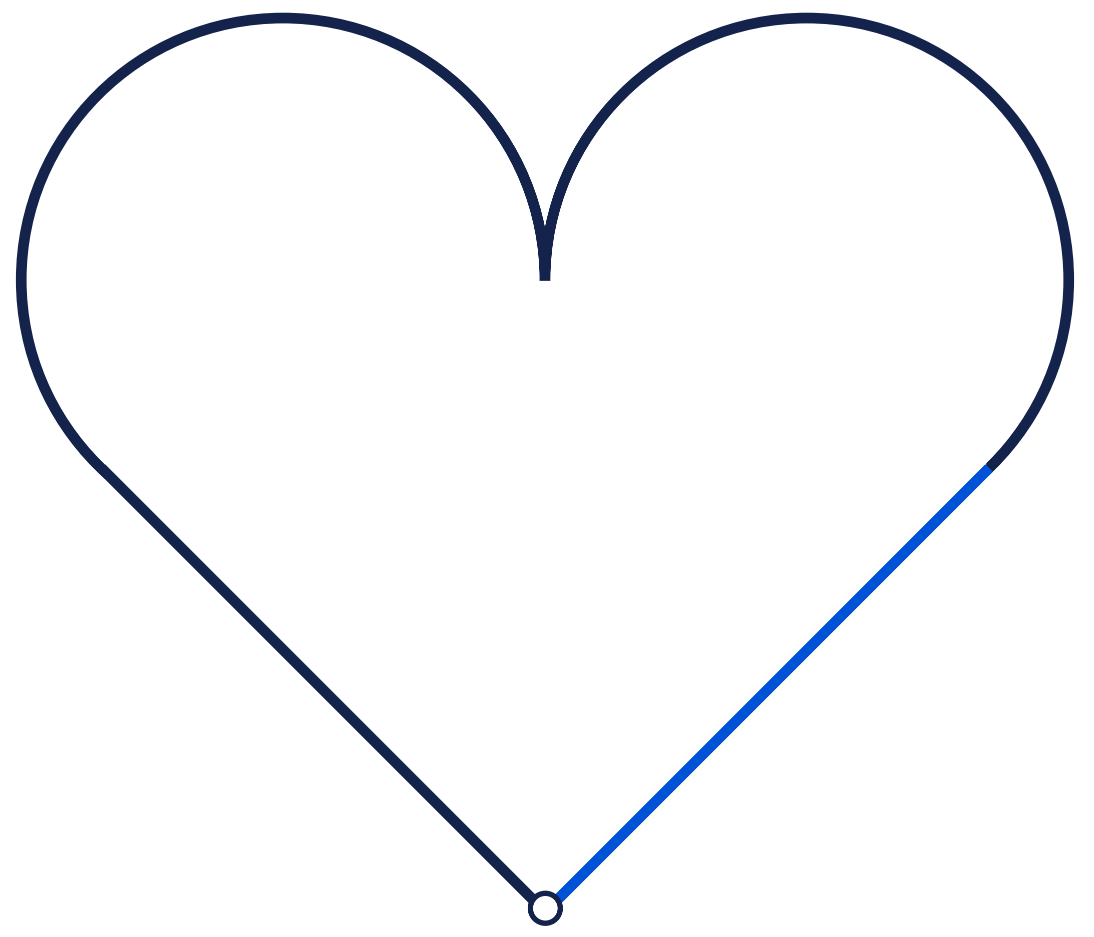


```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```

```
LENGTH:
  length: 1.21
  entity:
```

Interpreter-guided generation

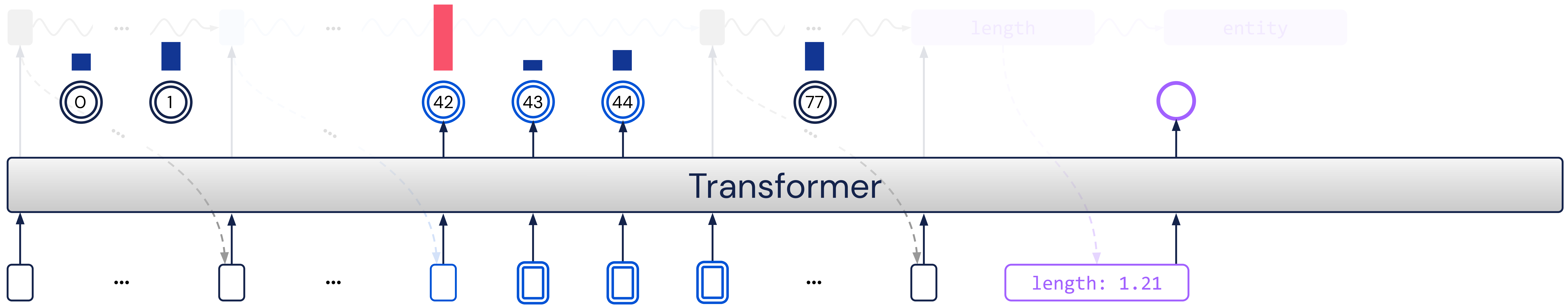


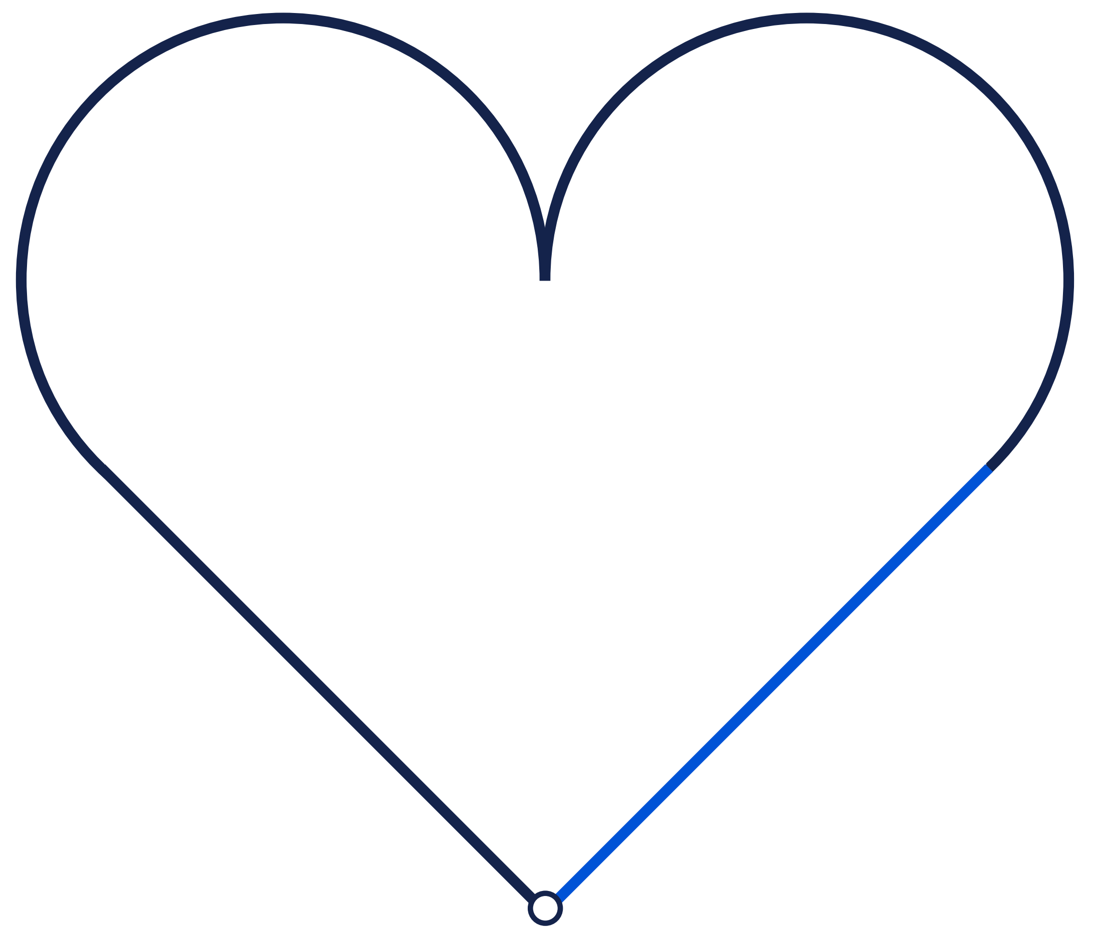


```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```

```
LENGTH:
  length: 1.21
  entity:
```

Interpreter-guided generation

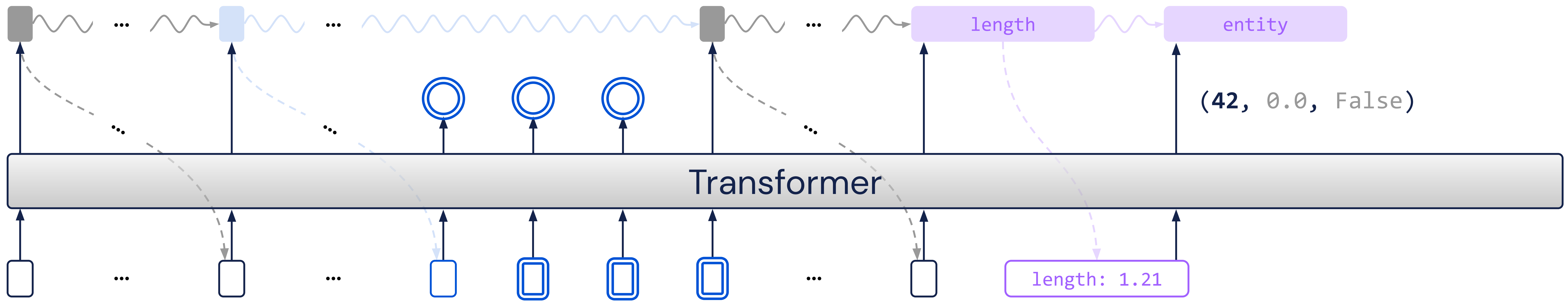




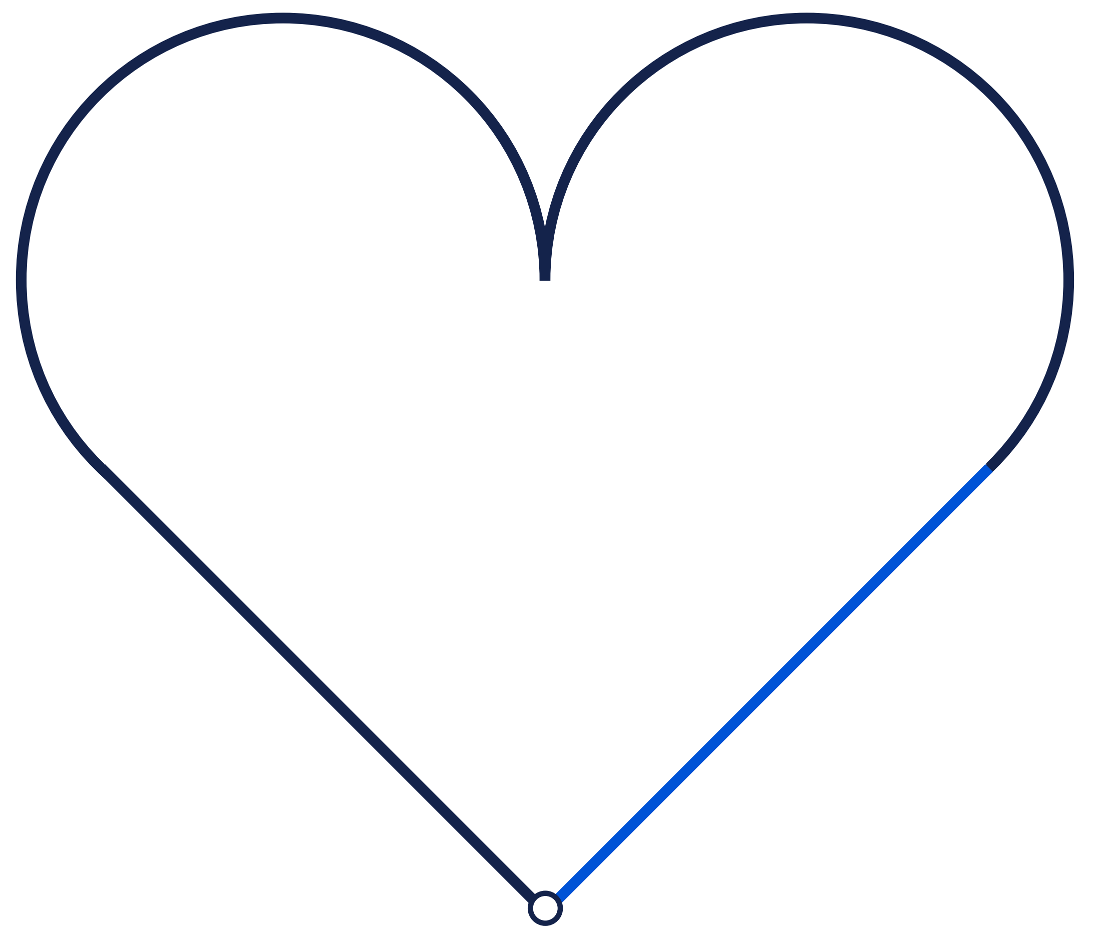
```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```

```
LENGTH:
  length: 1.21
  entity:
```

Interpreter-guided generation

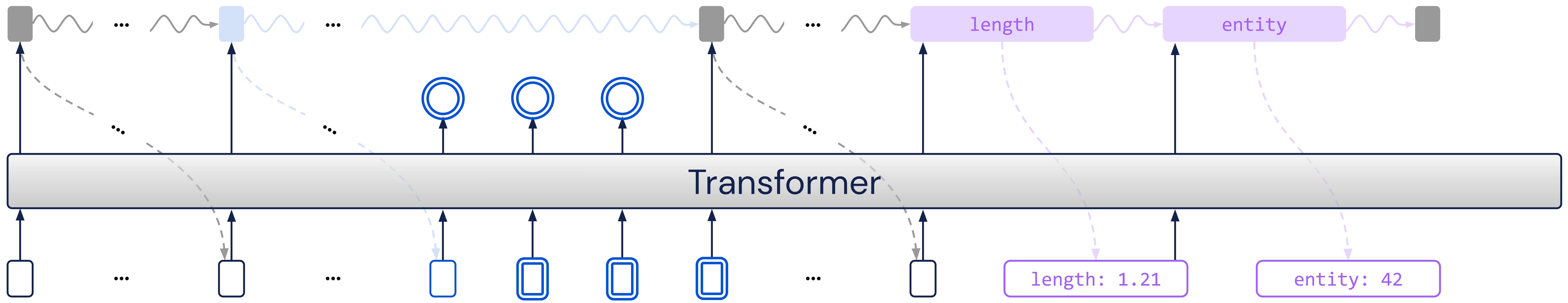


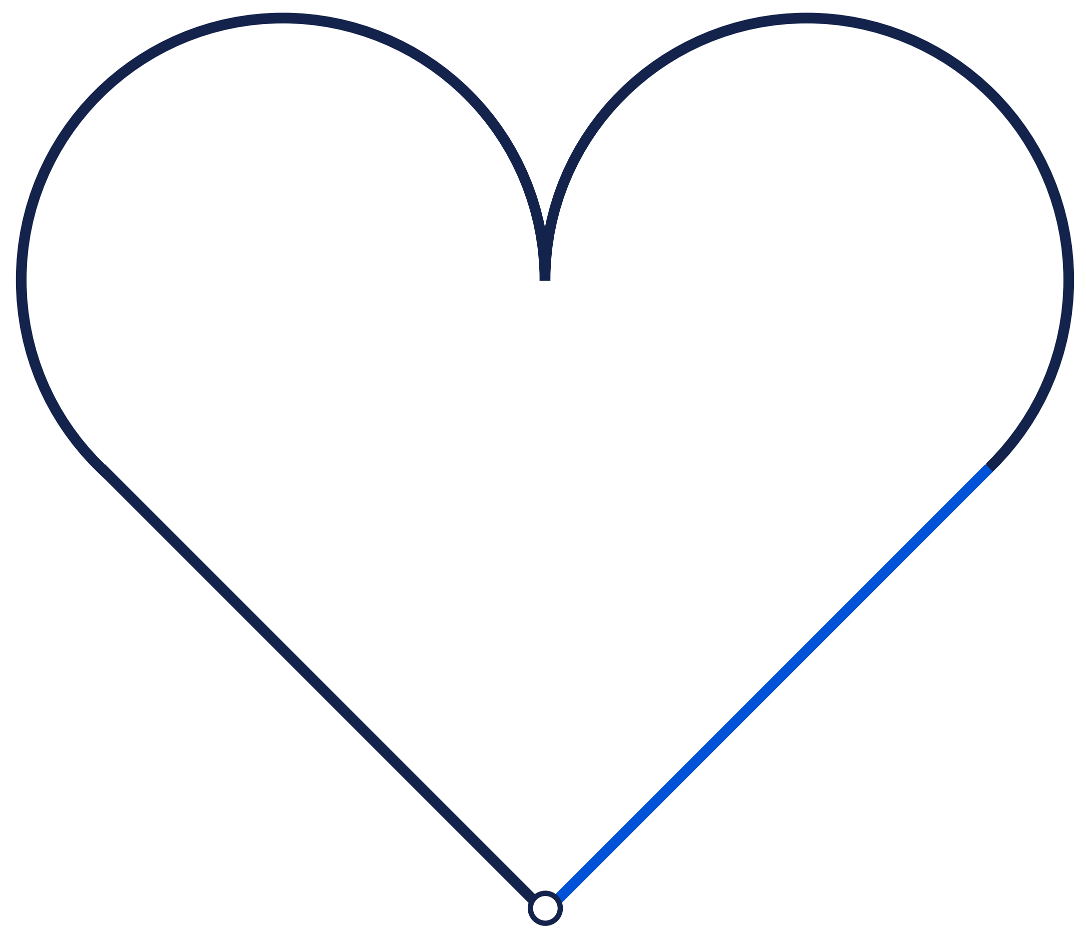
```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```



```
LENGTH:
  length: 1.21
  entity:
```

Interpreter-guided generation

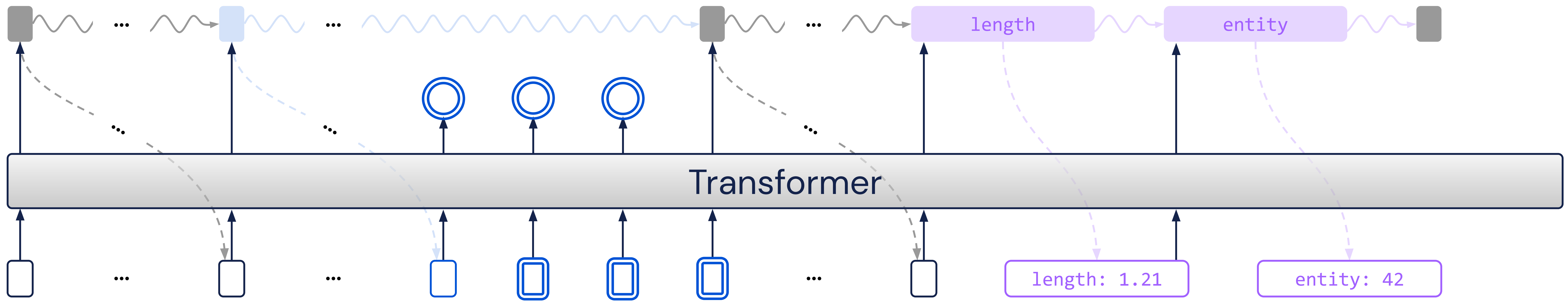




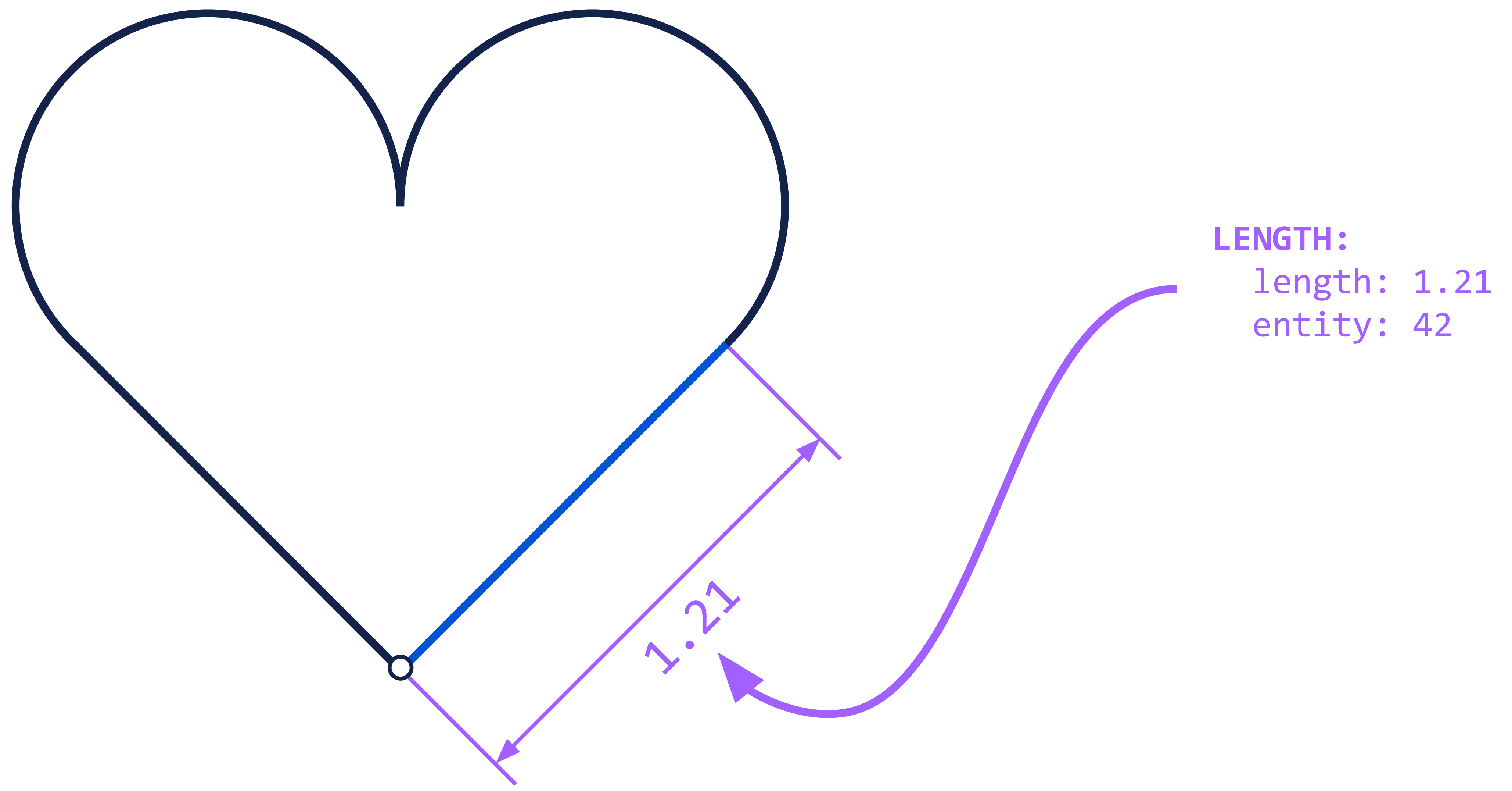
```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```

```
LENGTH:
  length: 1.21
  entity: 42
```

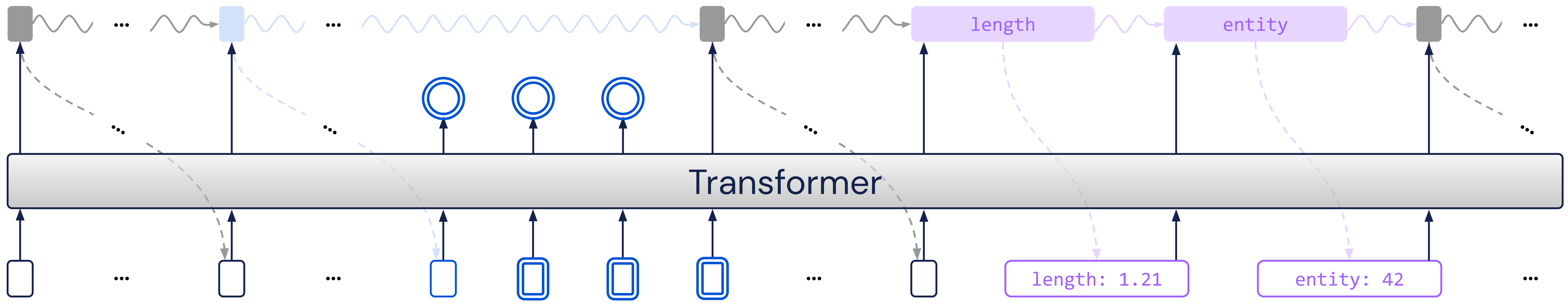
Interpreter-guided generation



```
message LengthConstraint {
  double length = 1;
  uint32 entity = 2 [is_pointer = true];
}
```



Interpreter-guided generation



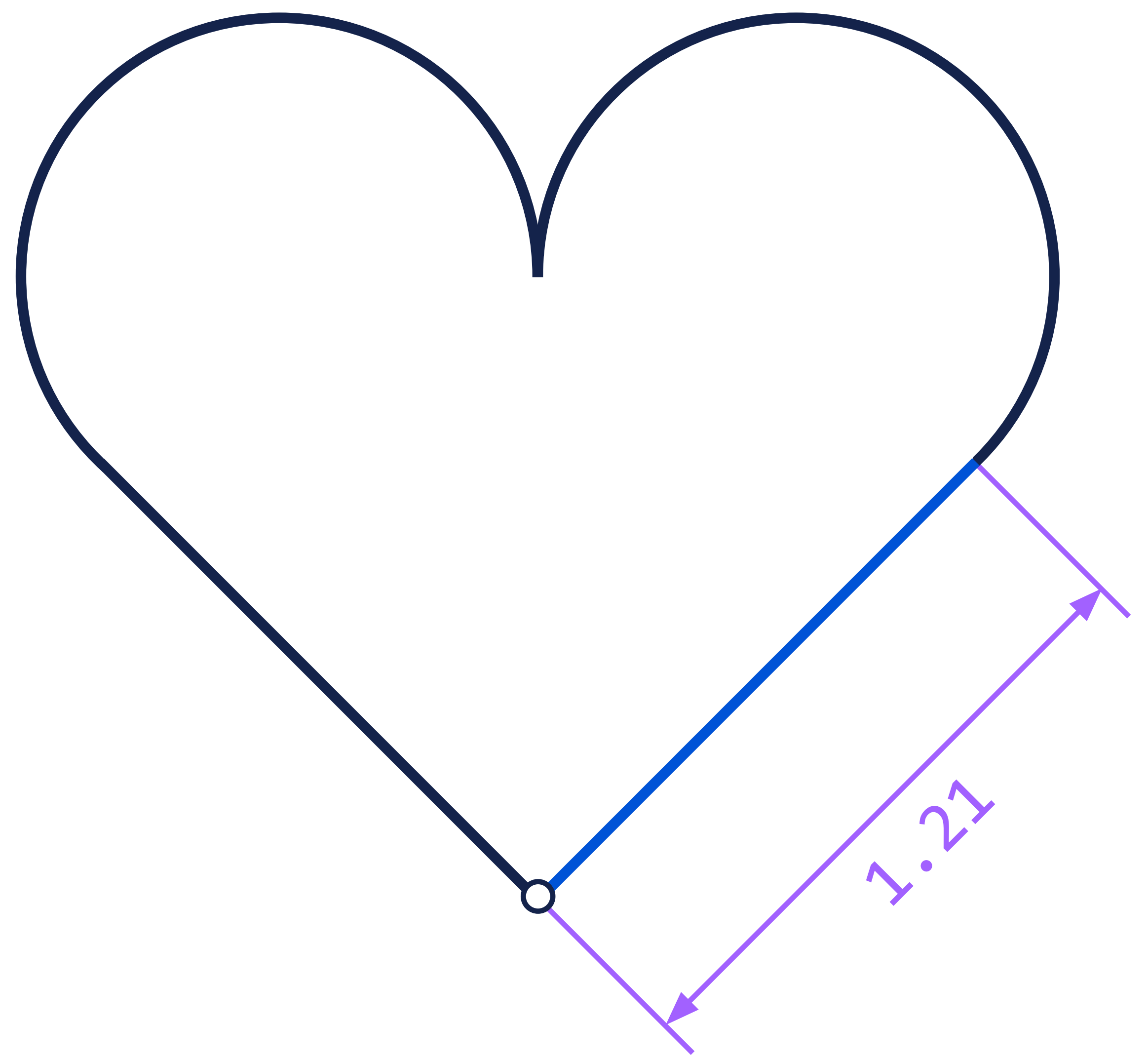


Image to sketch

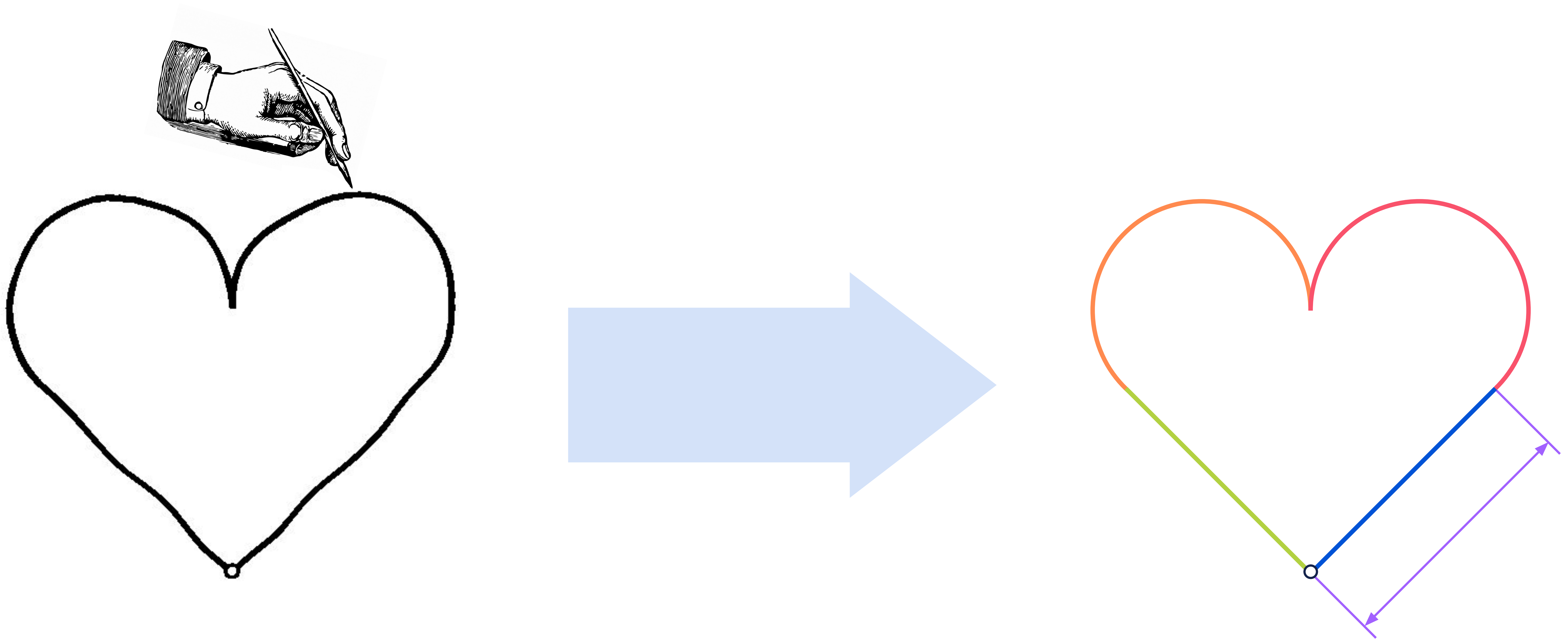
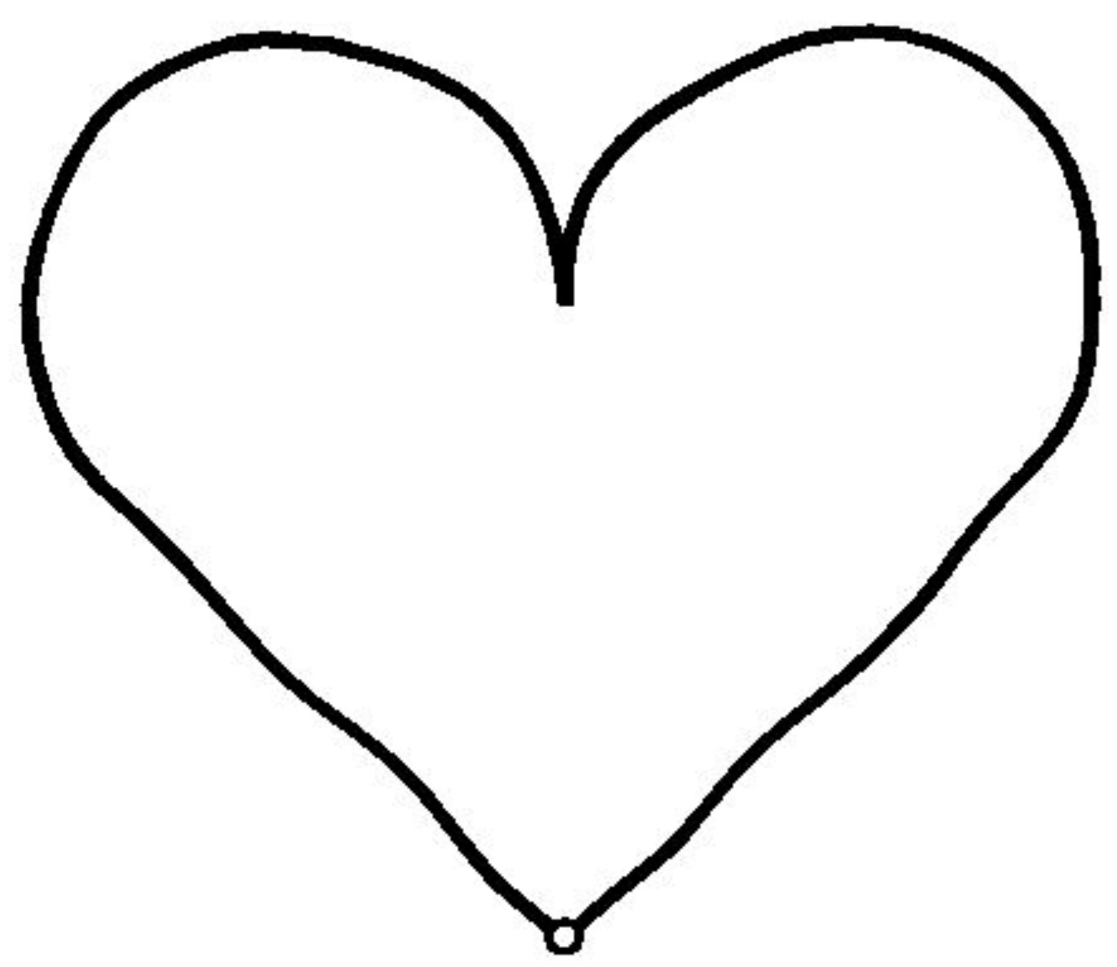


Image to sketch



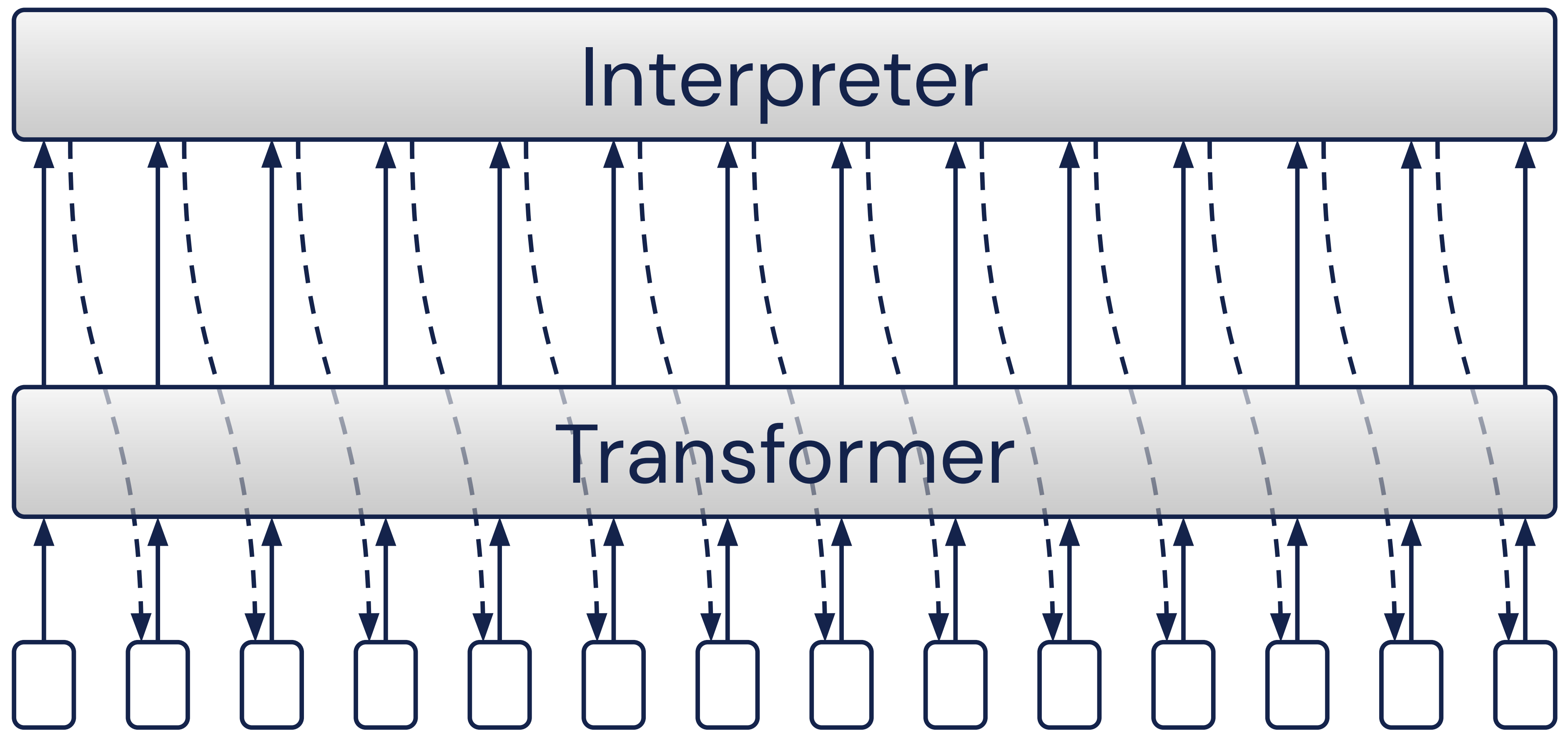
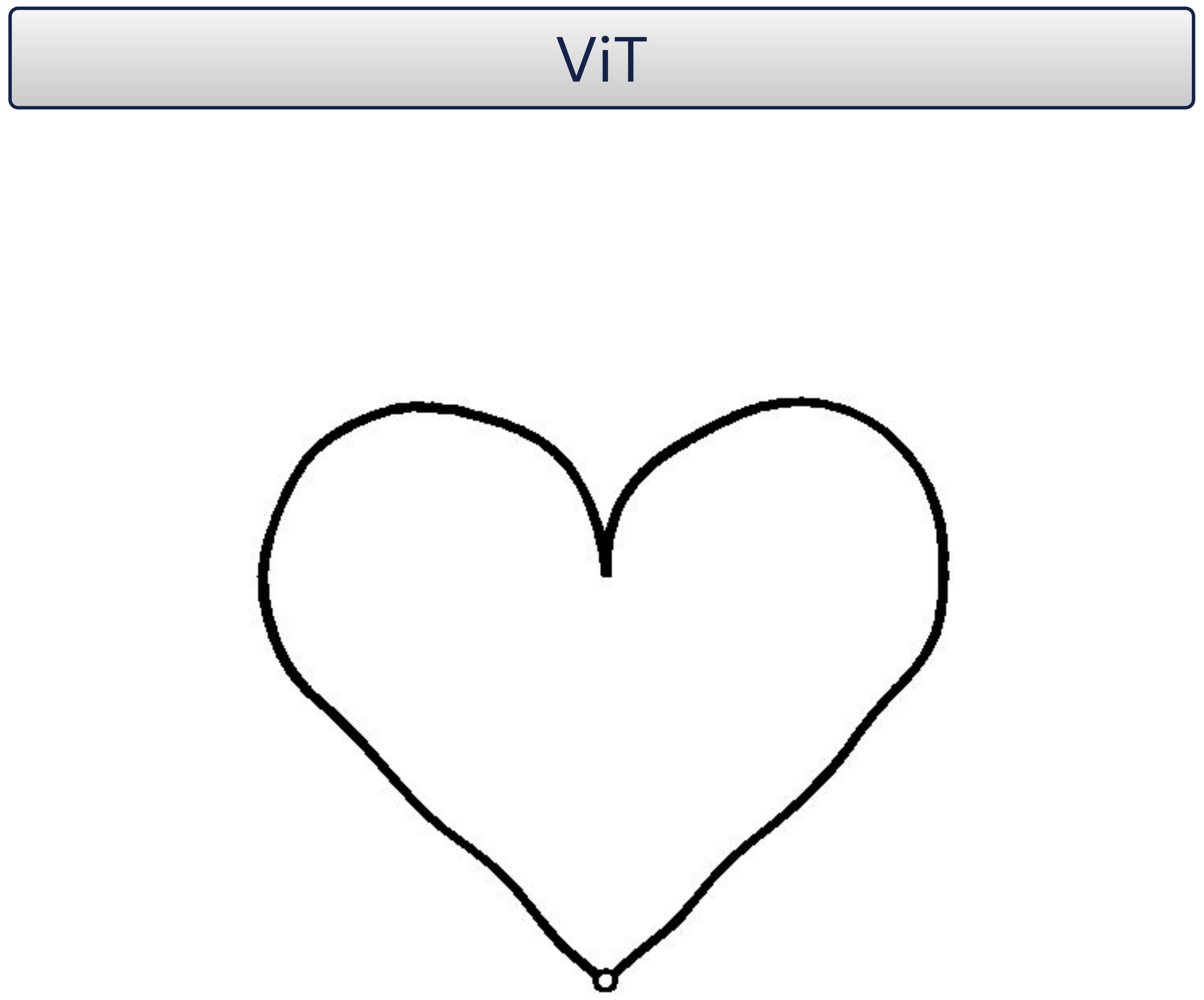


Image to sketch



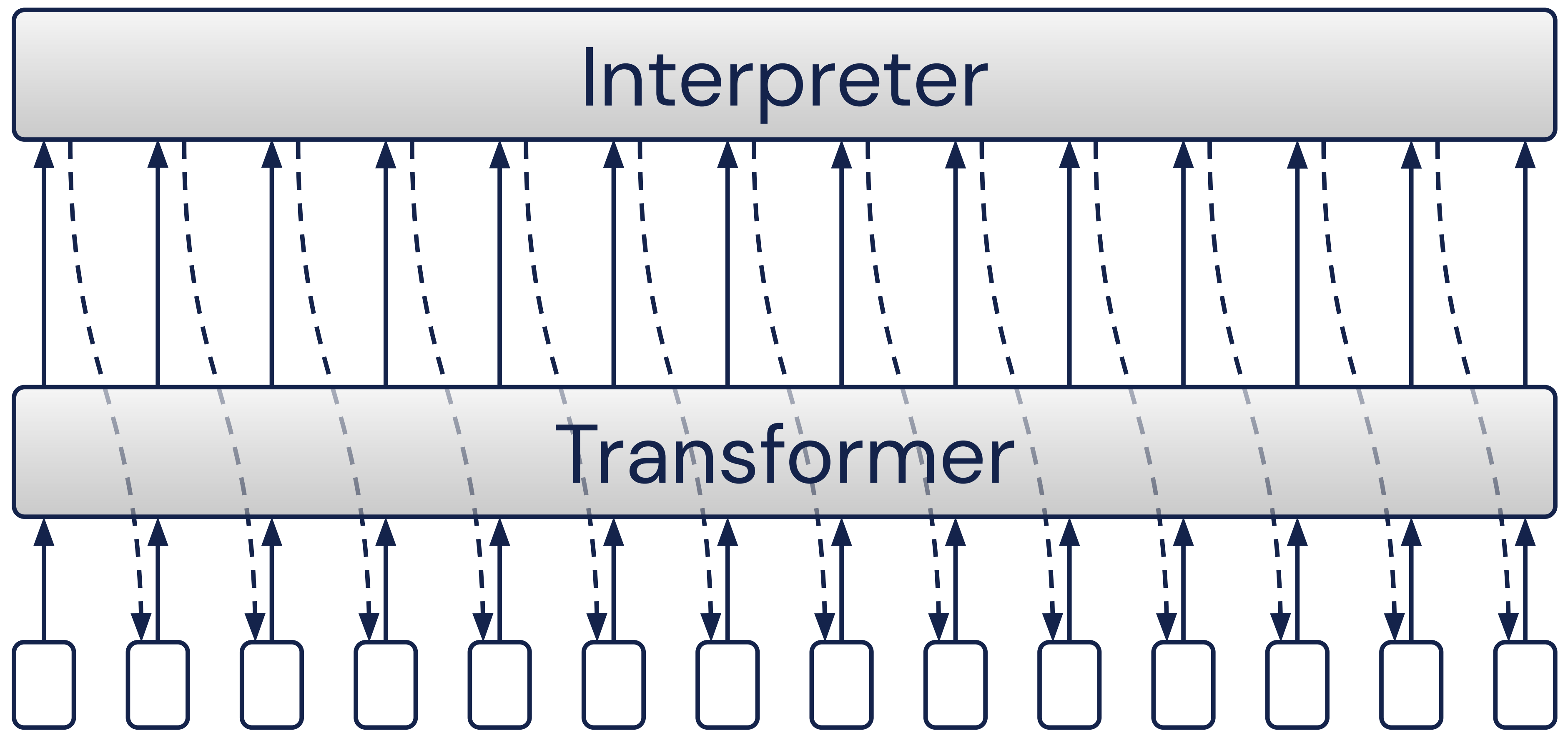
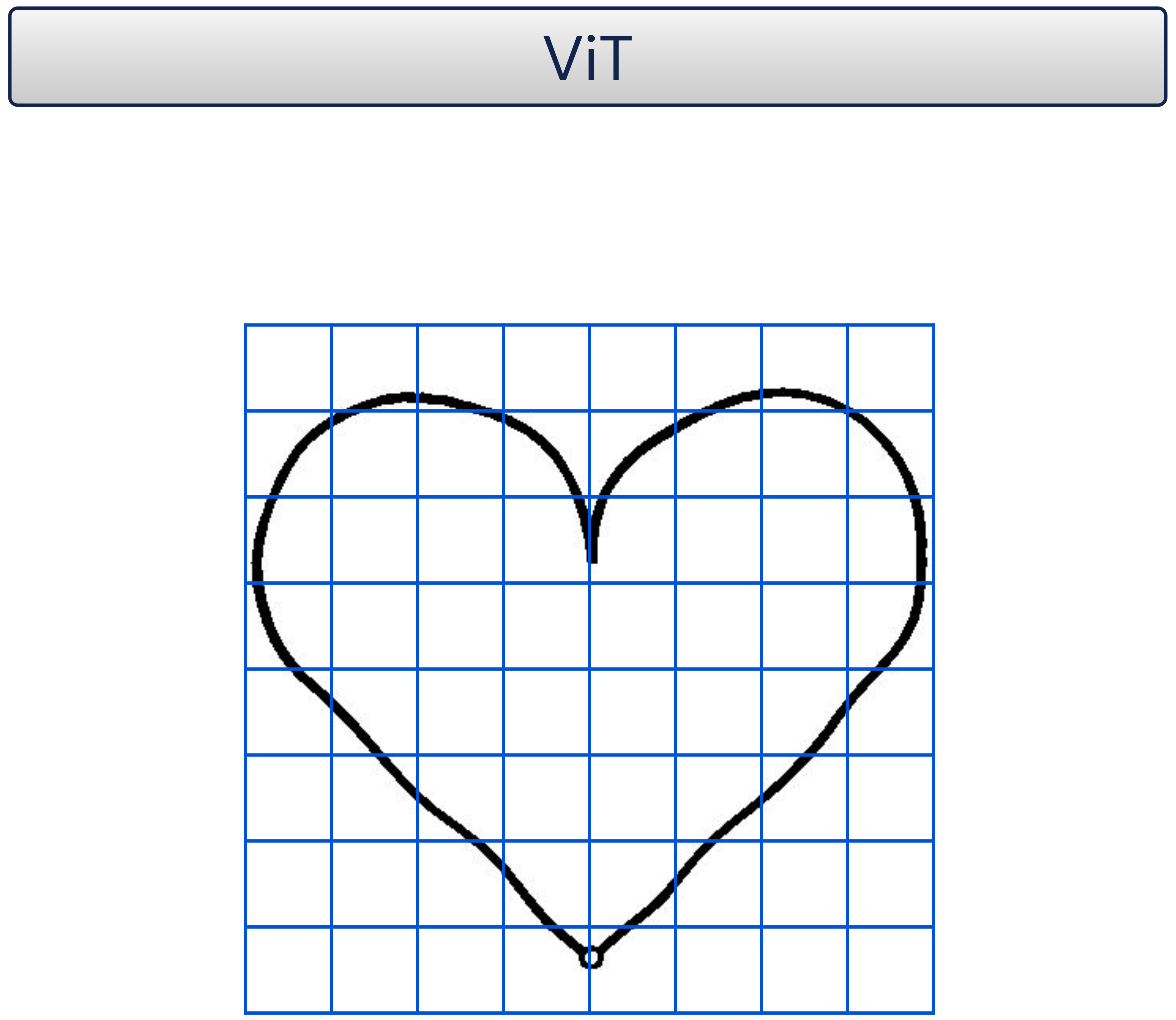


Image to sketch



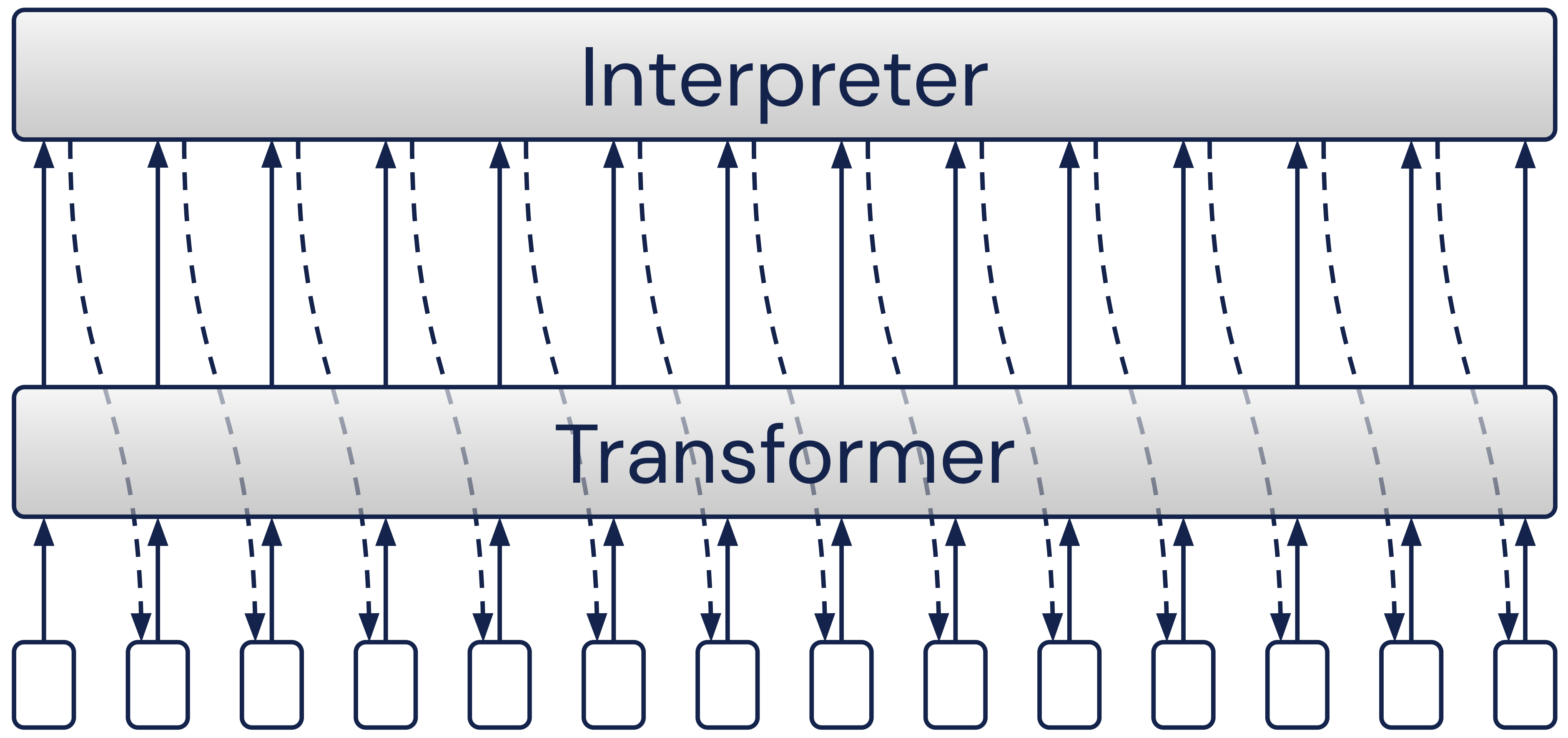


Image to sketch

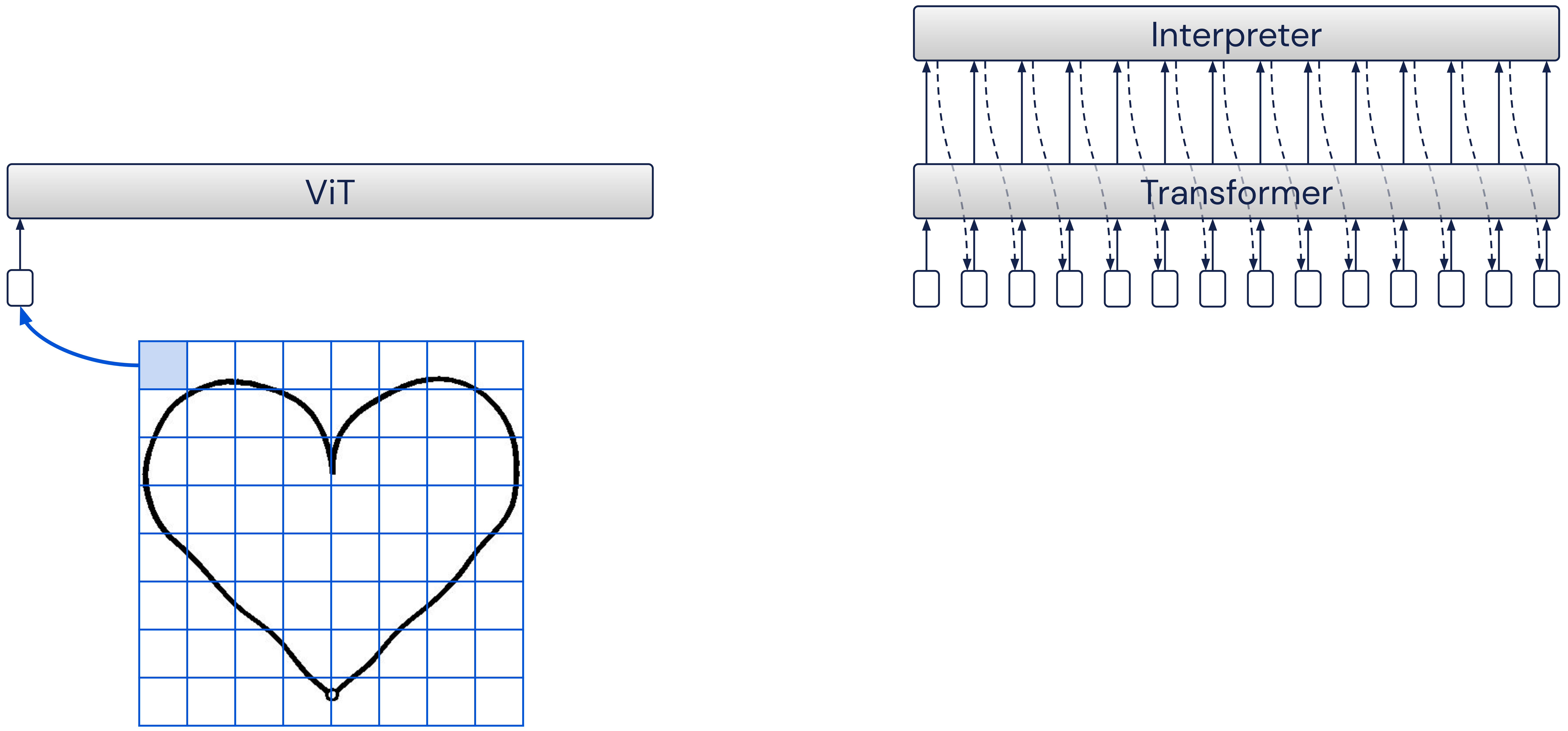
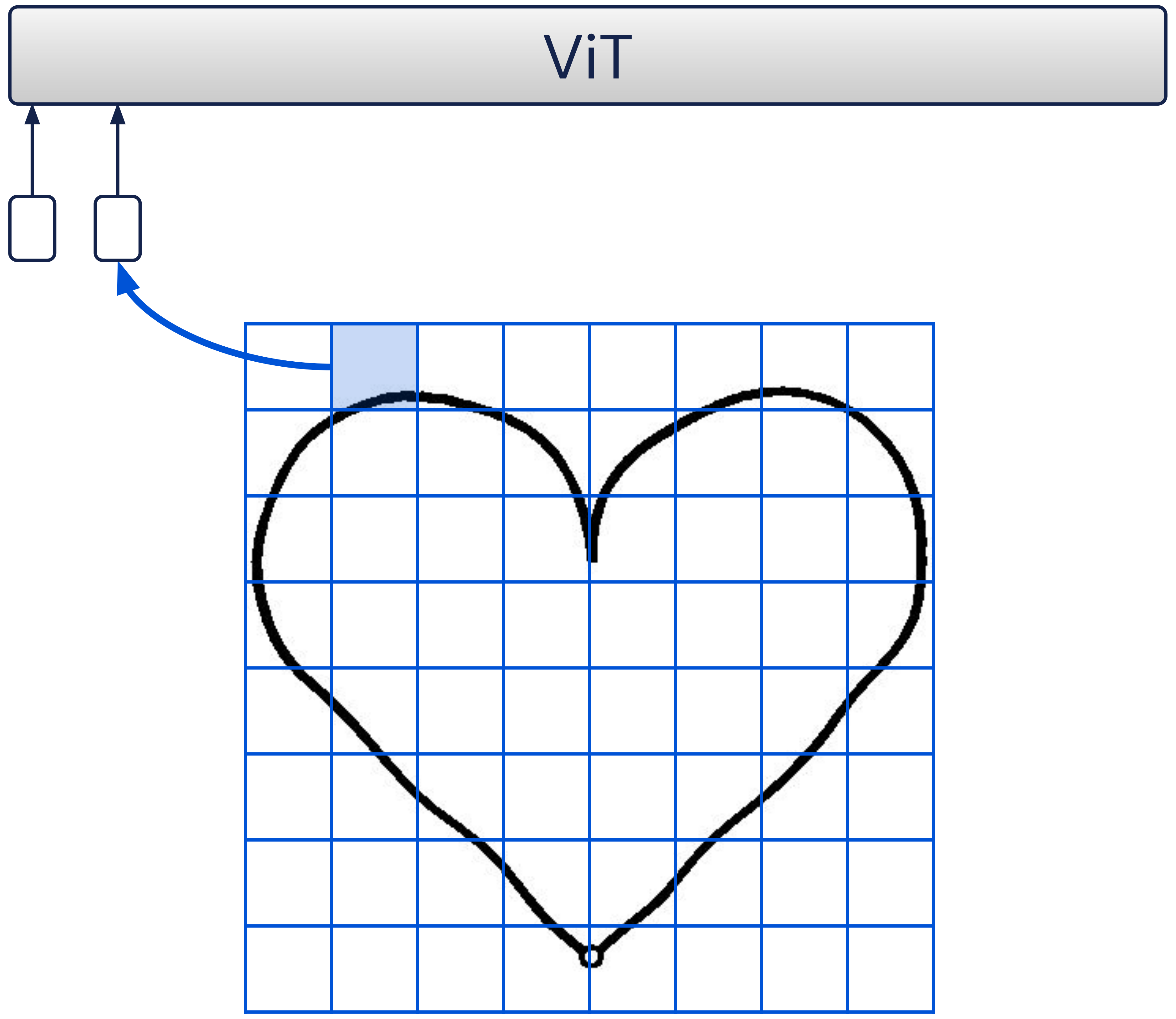


Image to sketch



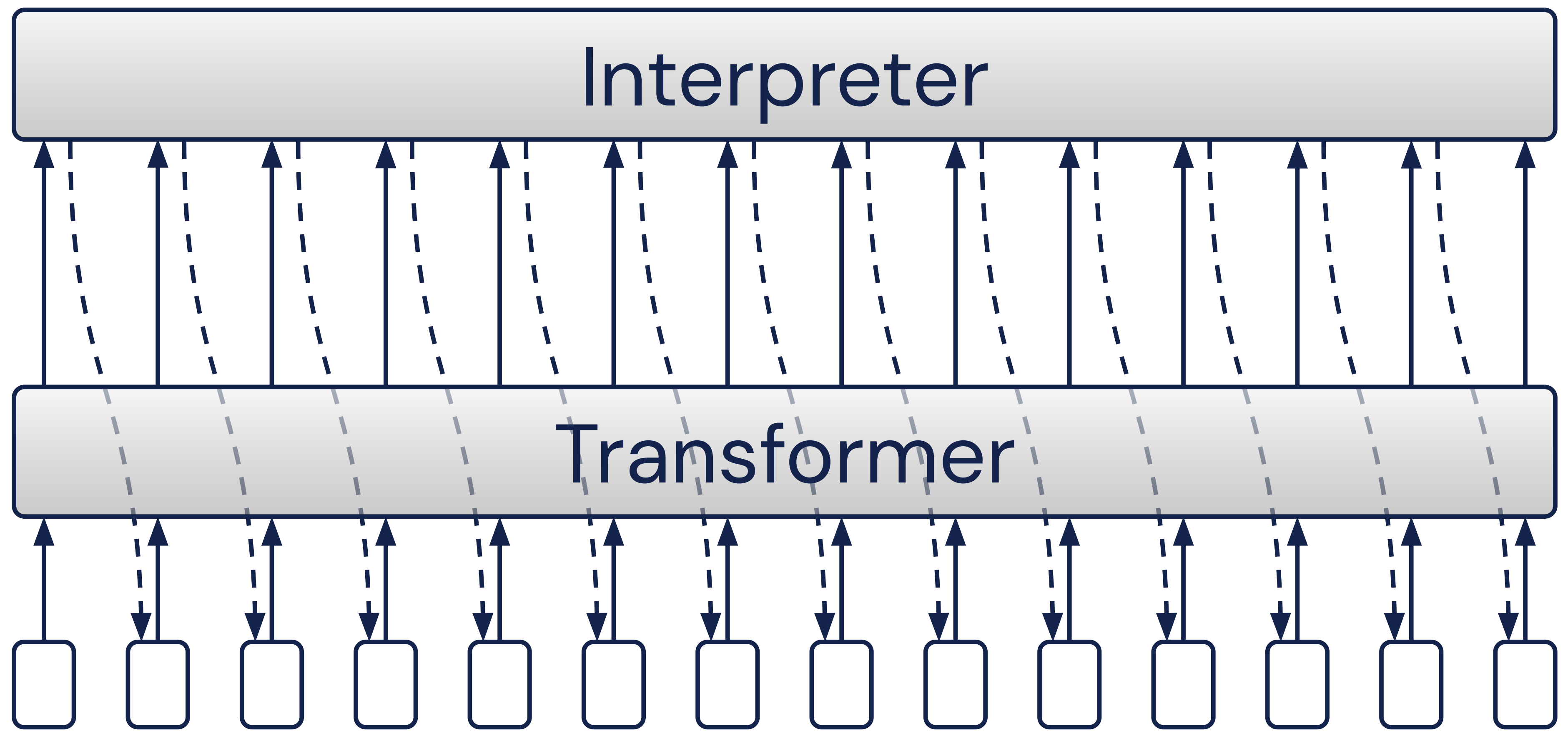
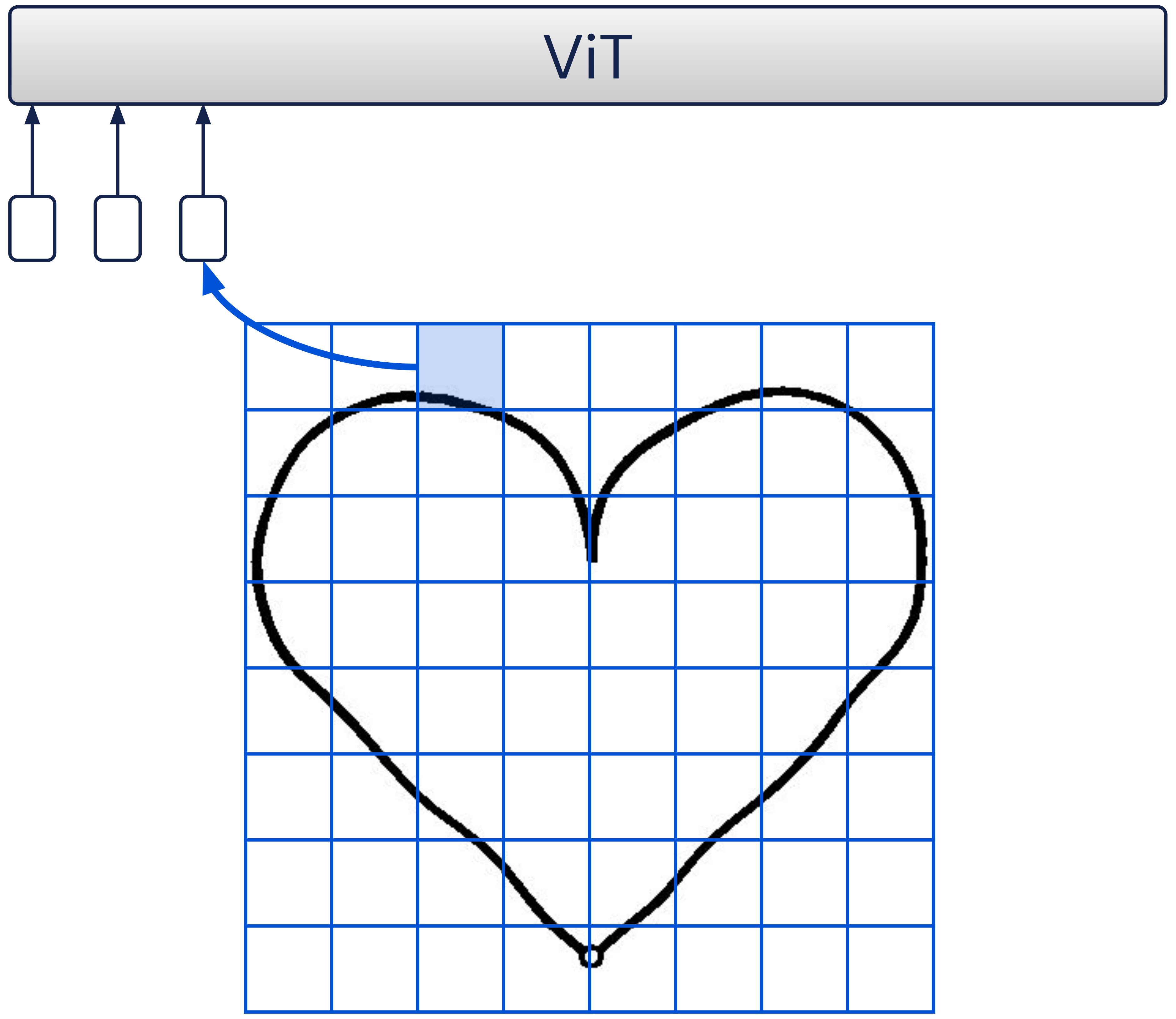


Image to sketch



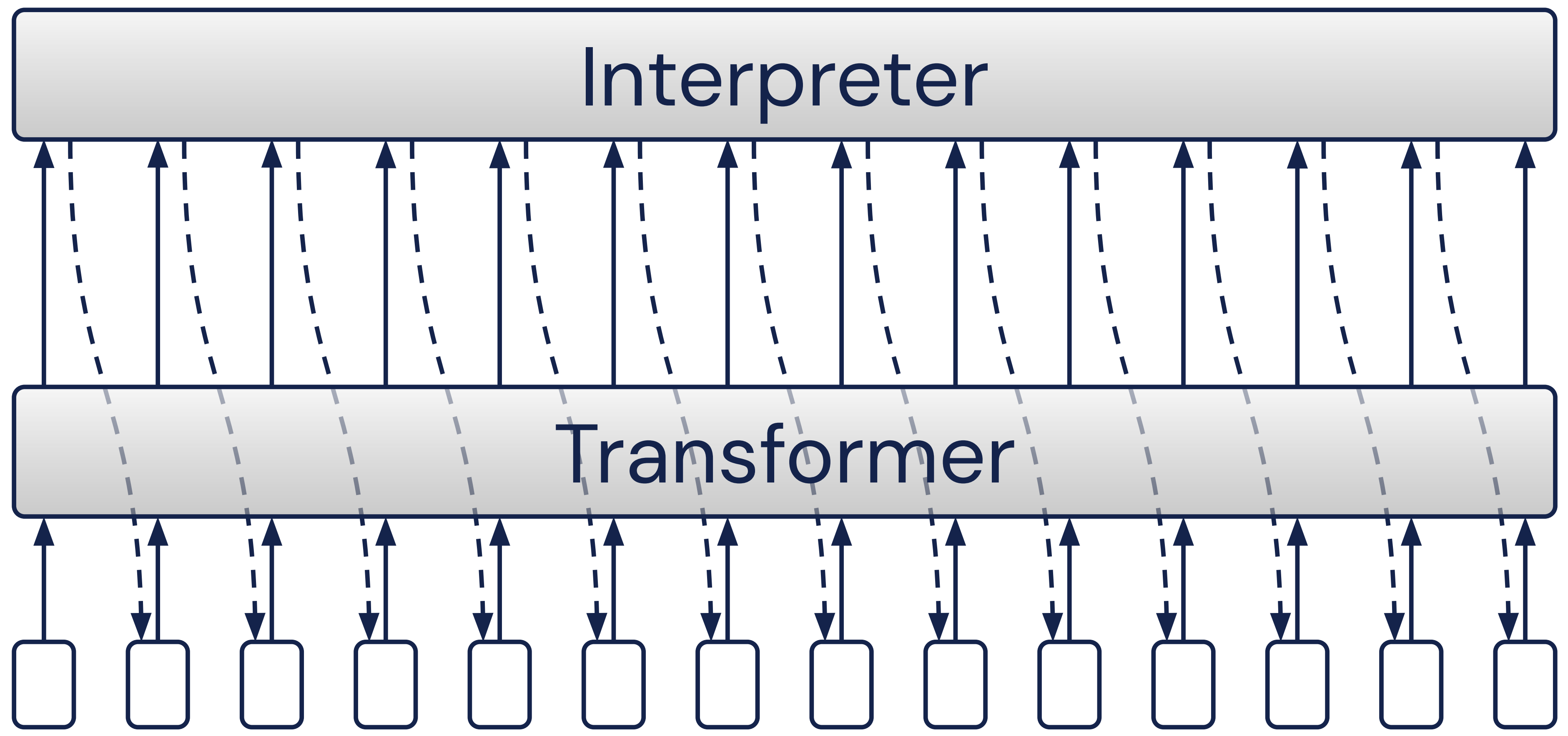
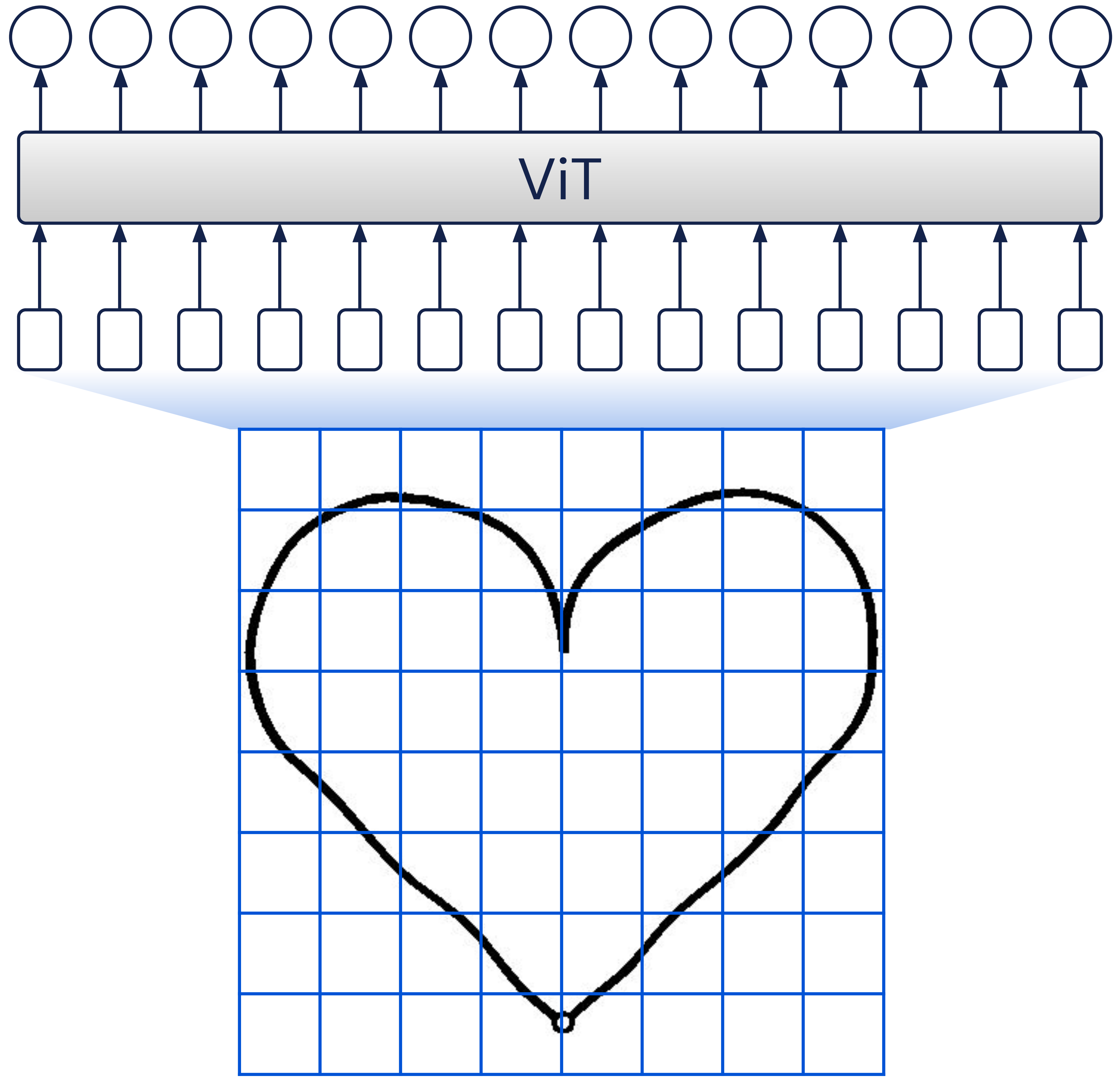


Image to sketch



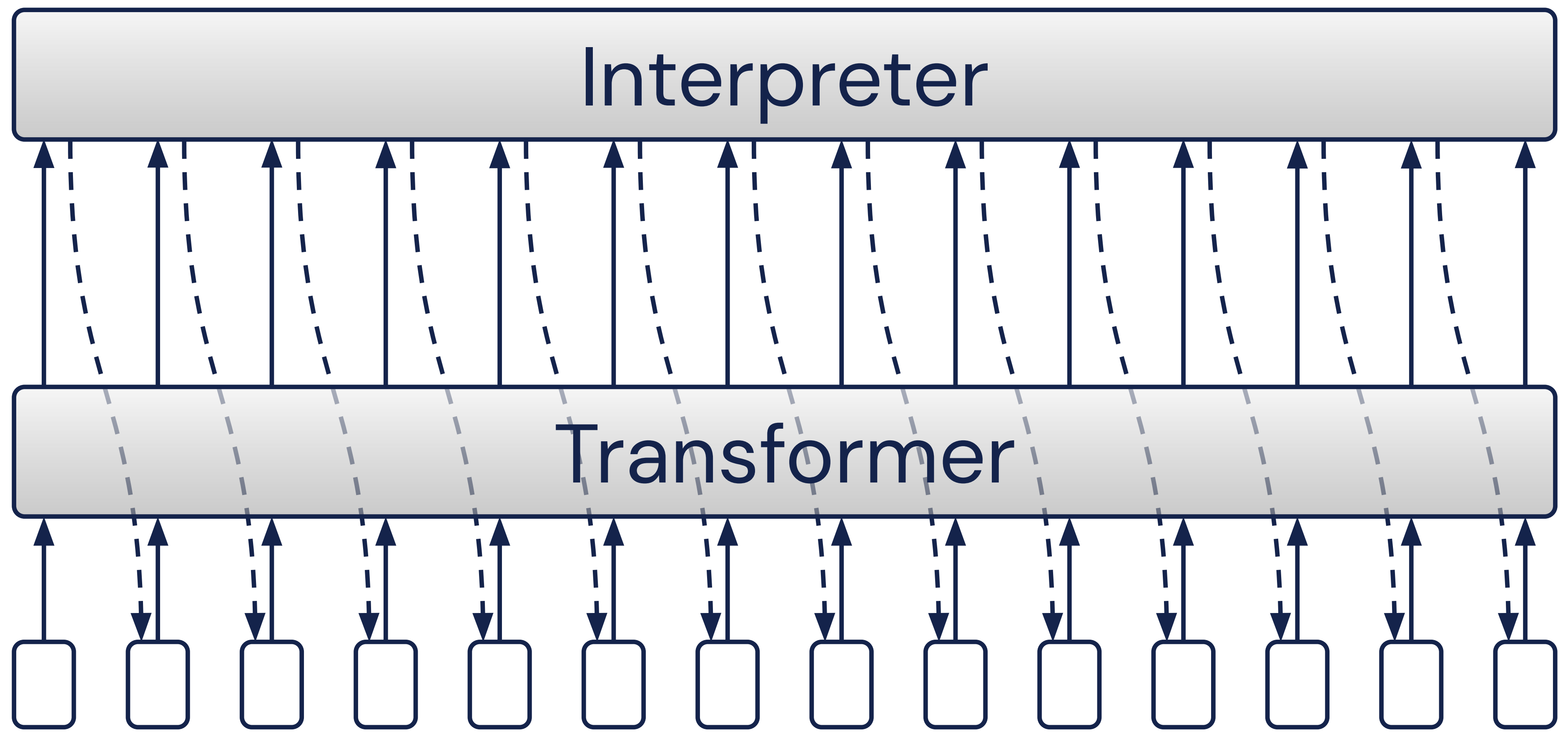
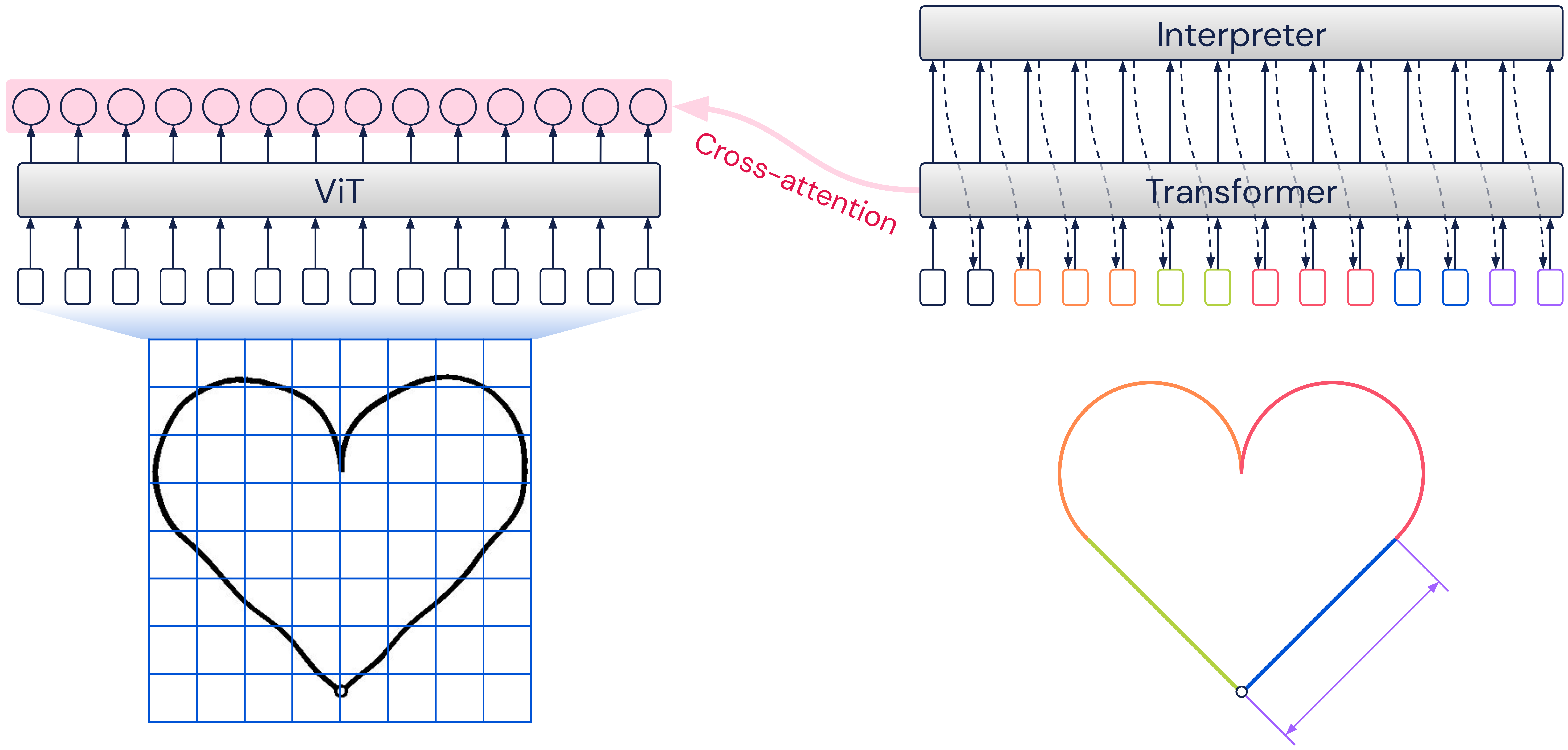


Image to sketch



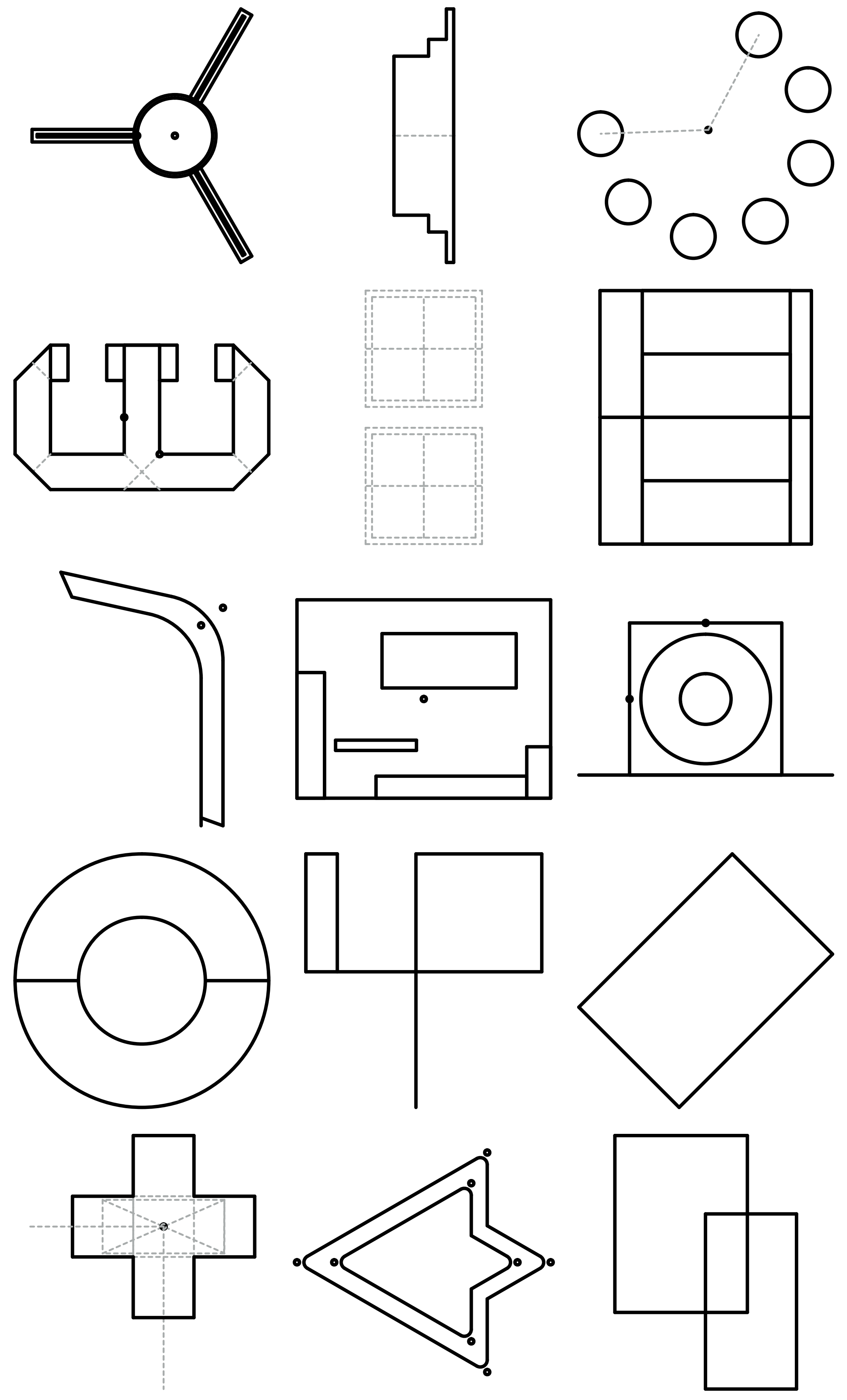
Dataset

We scraped about **6TB** of CAD data from **Onshape**
- 3D actions
- Sketches

A significant level of **duplication** so we had to apply **non-trivial filtering**

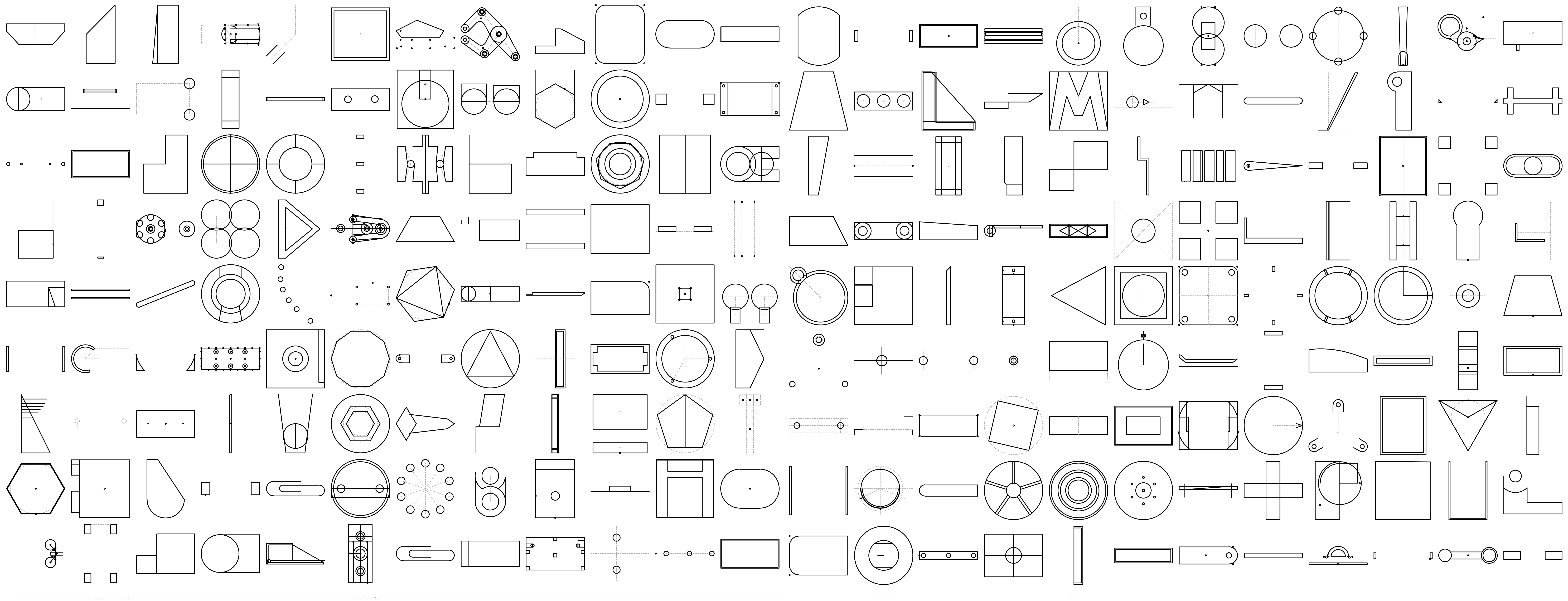
The stats (random split):
- **4,656,607 training** examples
- **50,000** both for the **validation** and the **test** set

Made **available** for public use

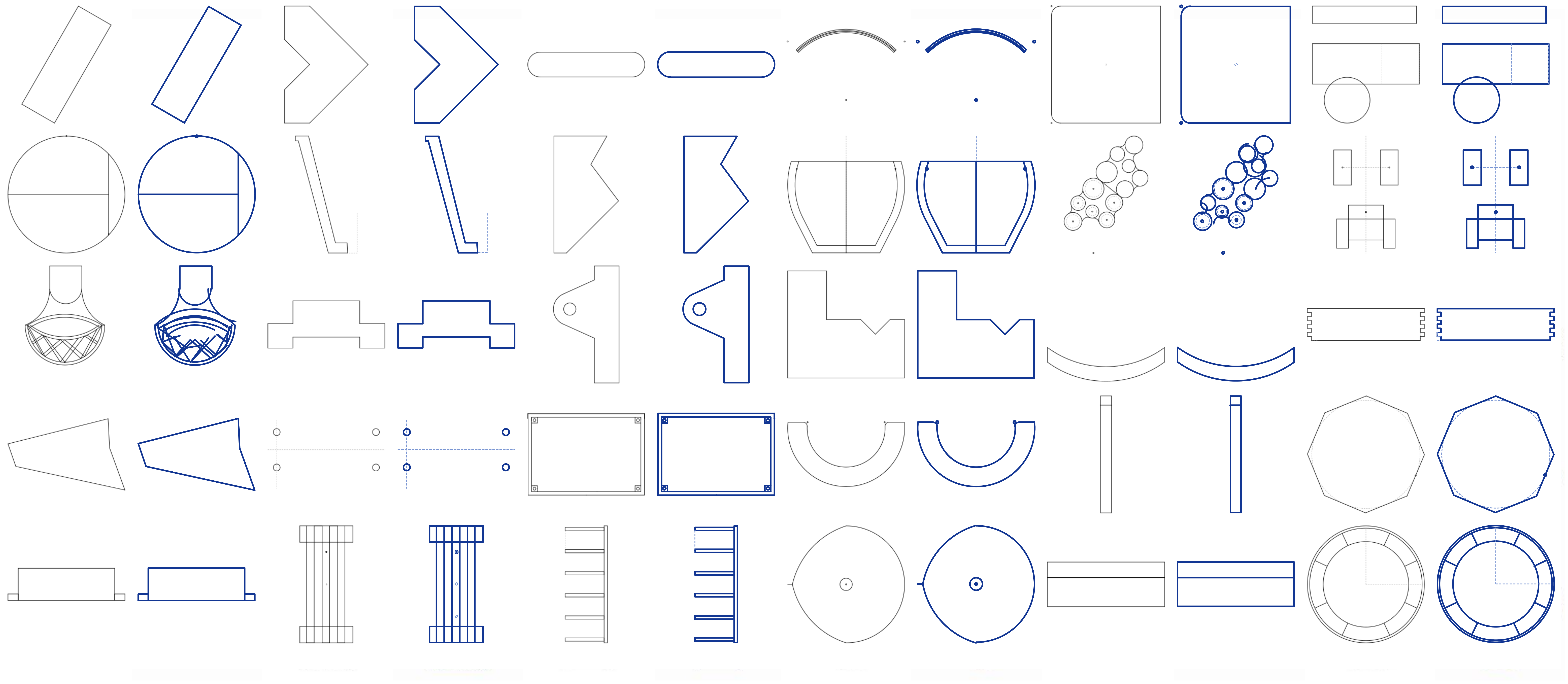


Results: unconditional samples

Results: unconditional samples

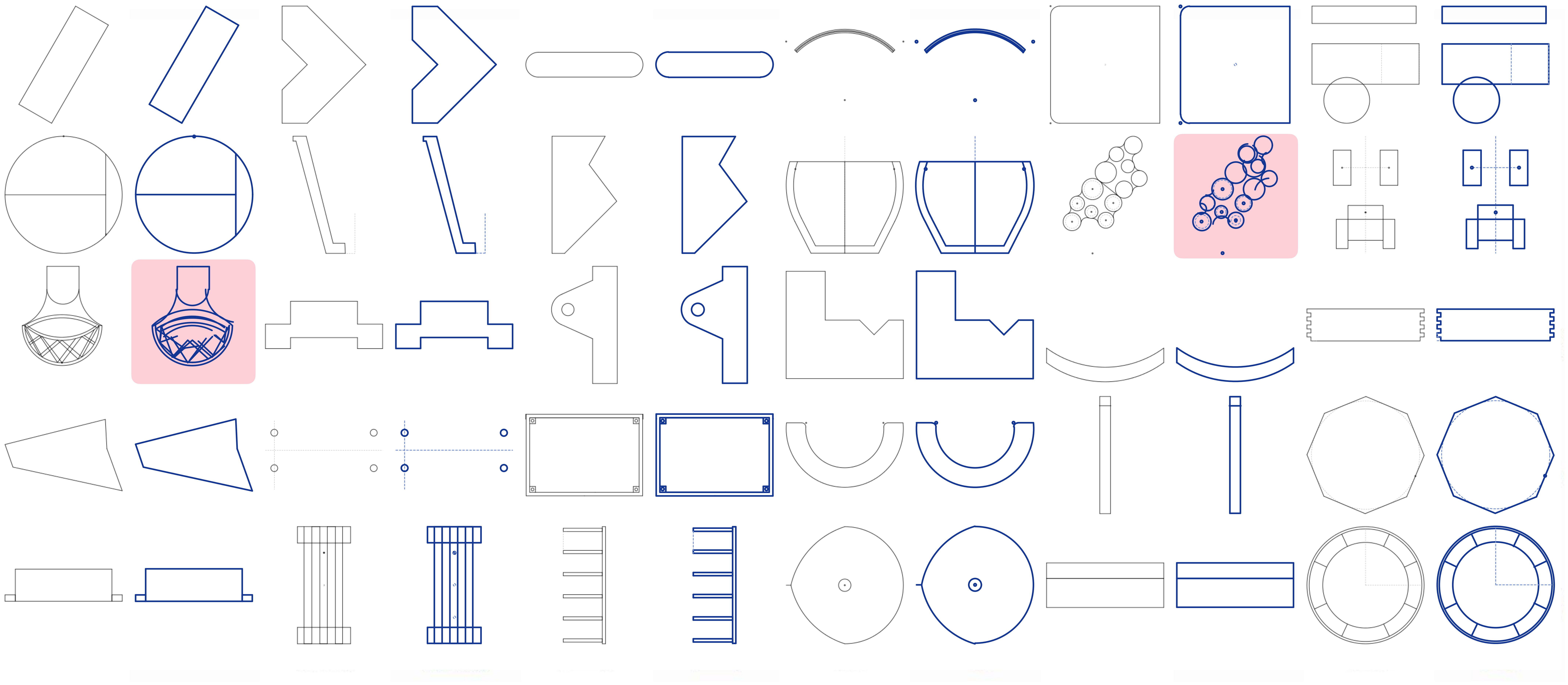


Results: image to sketch



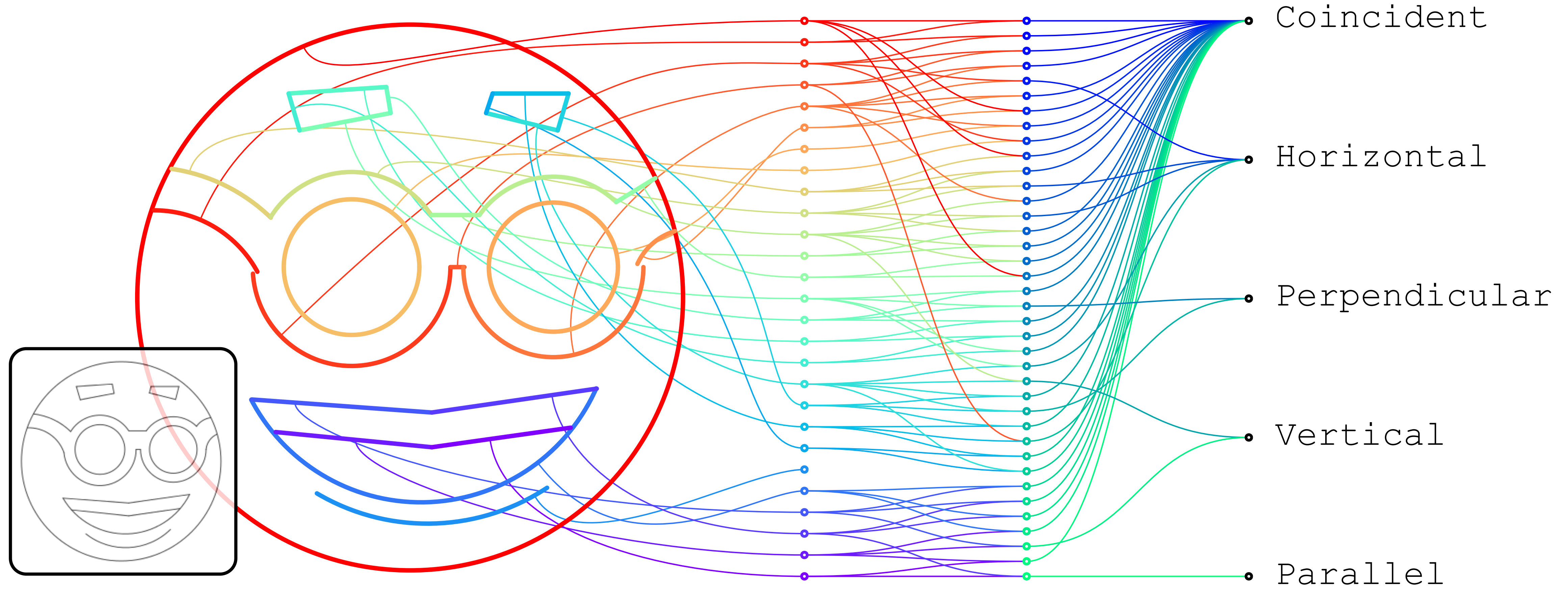
● Input bitmap ● Output sketch

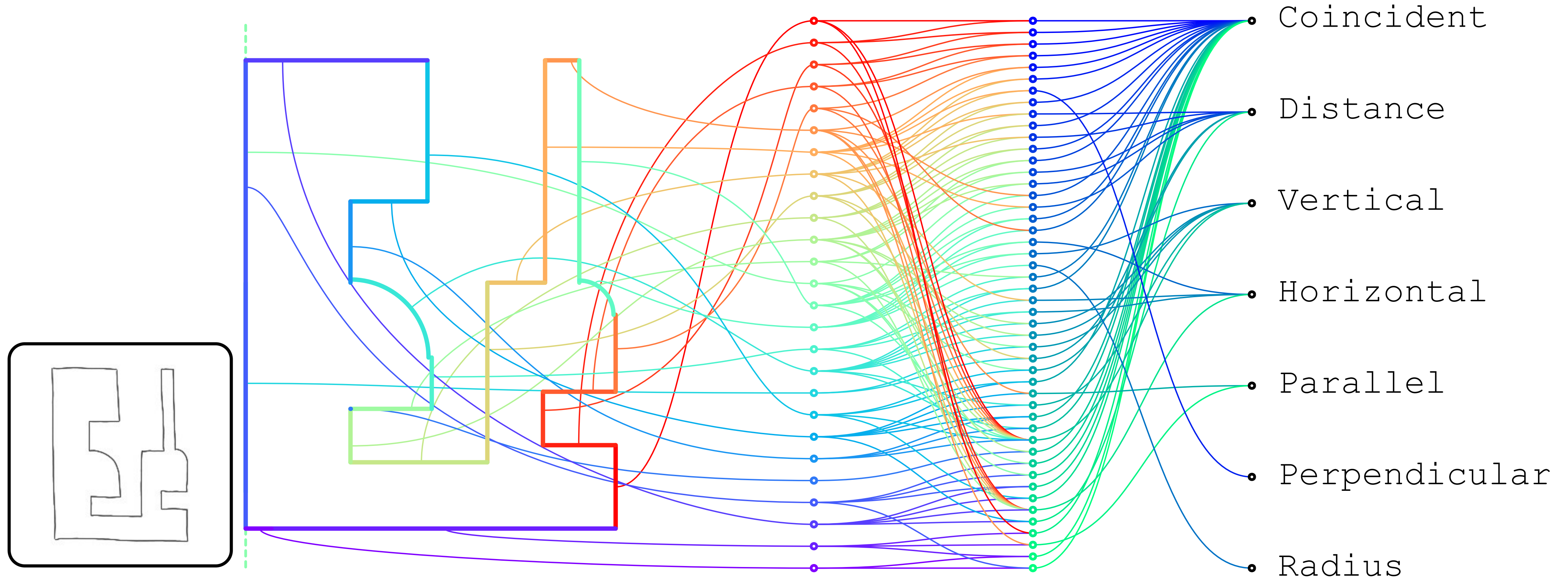
Results: image to sketch

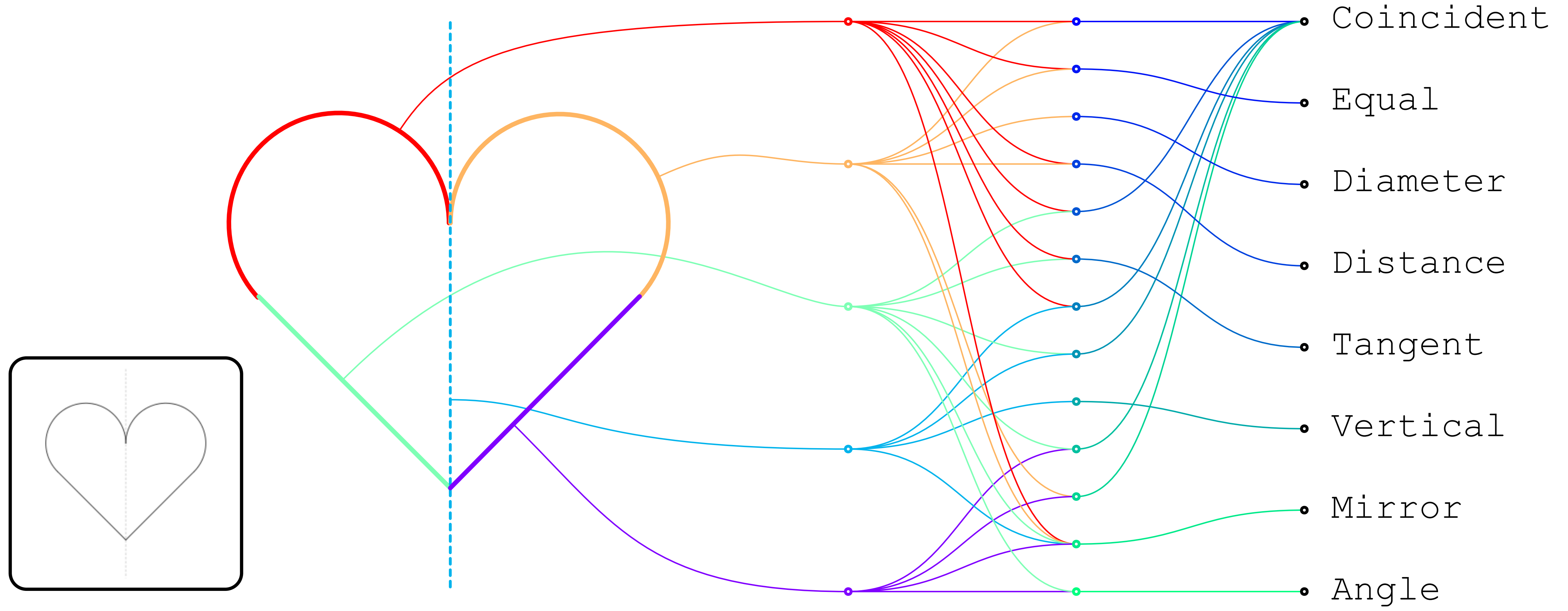


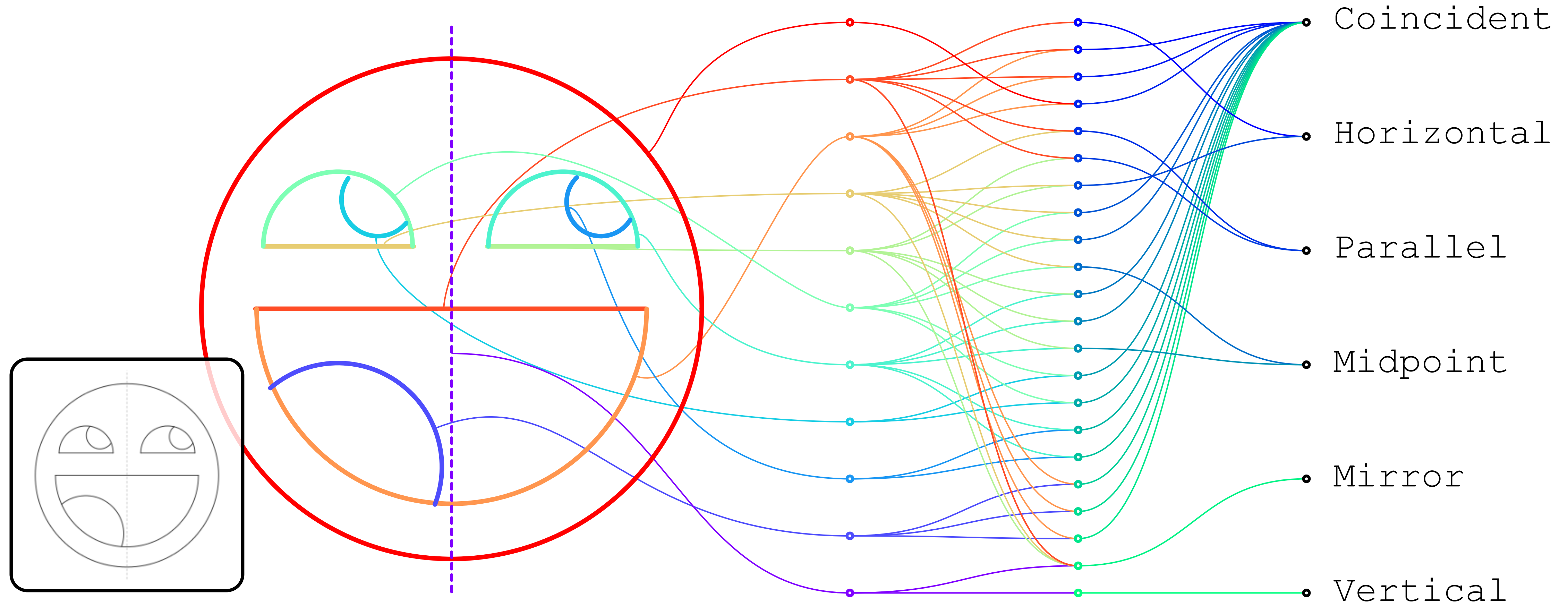
● Input bitmap ● Output sketch

Results: out of distribution

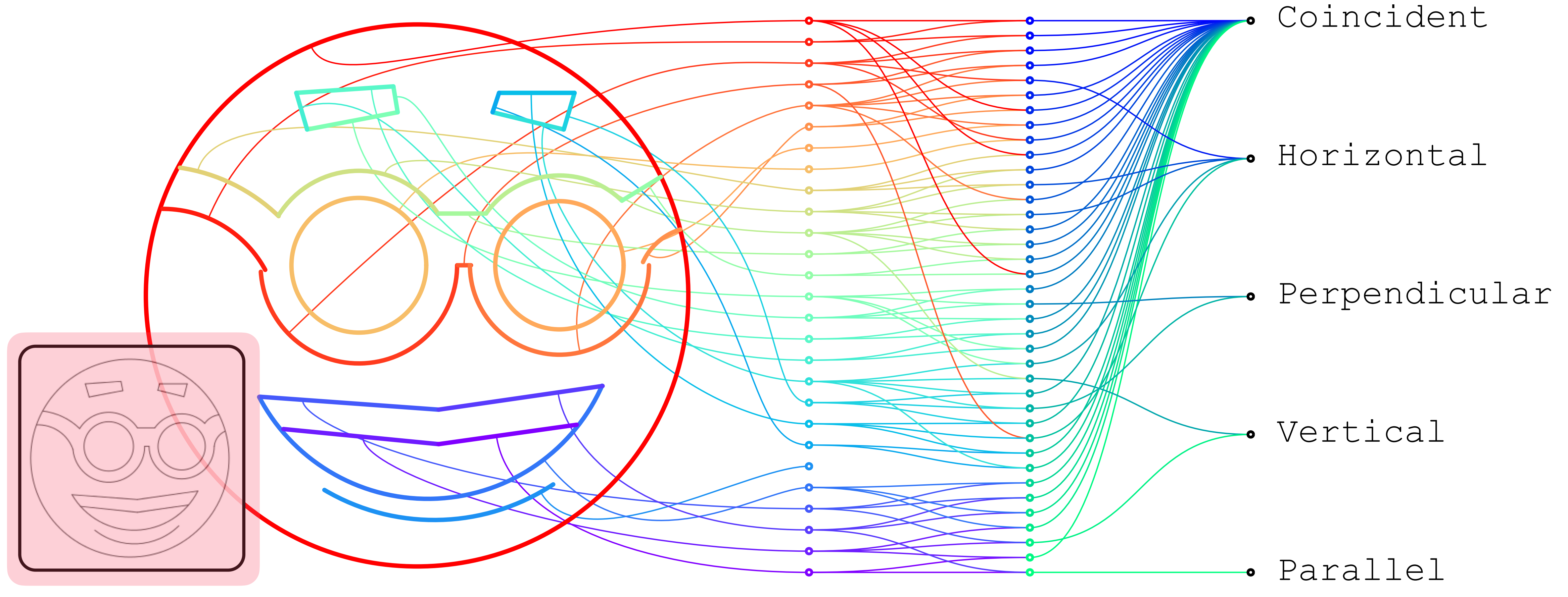


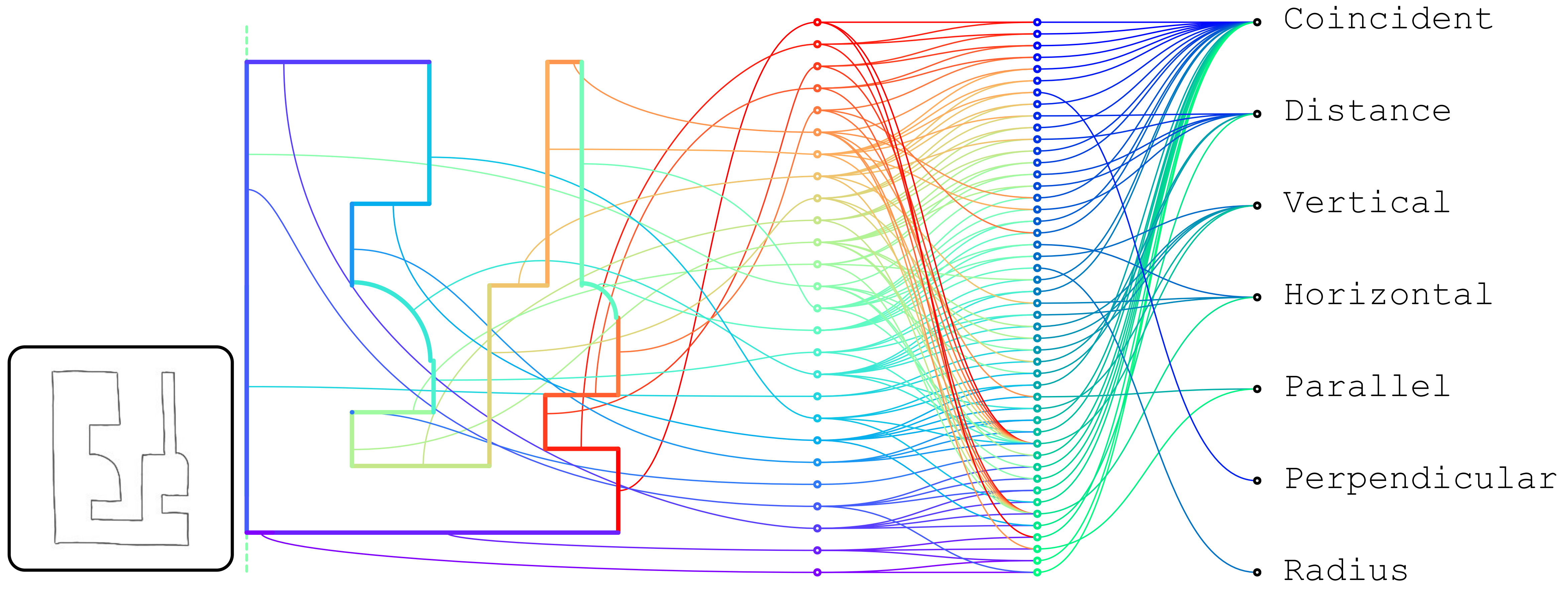


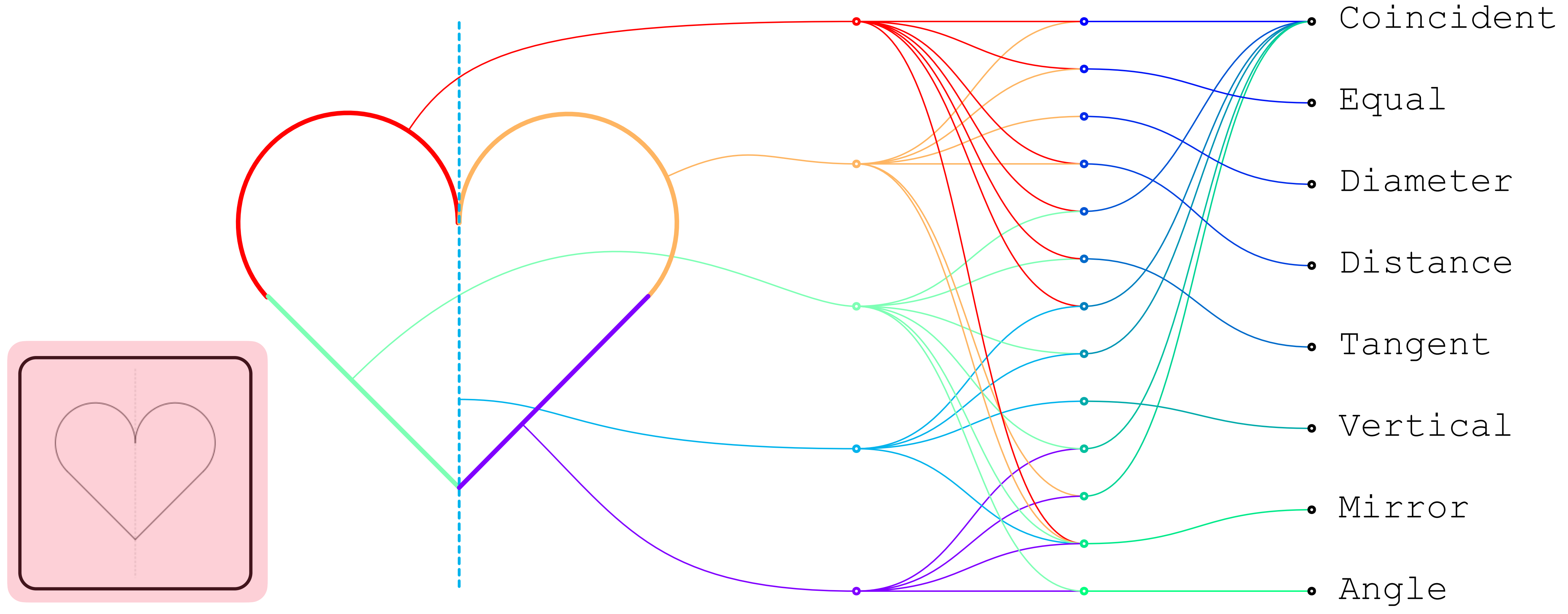


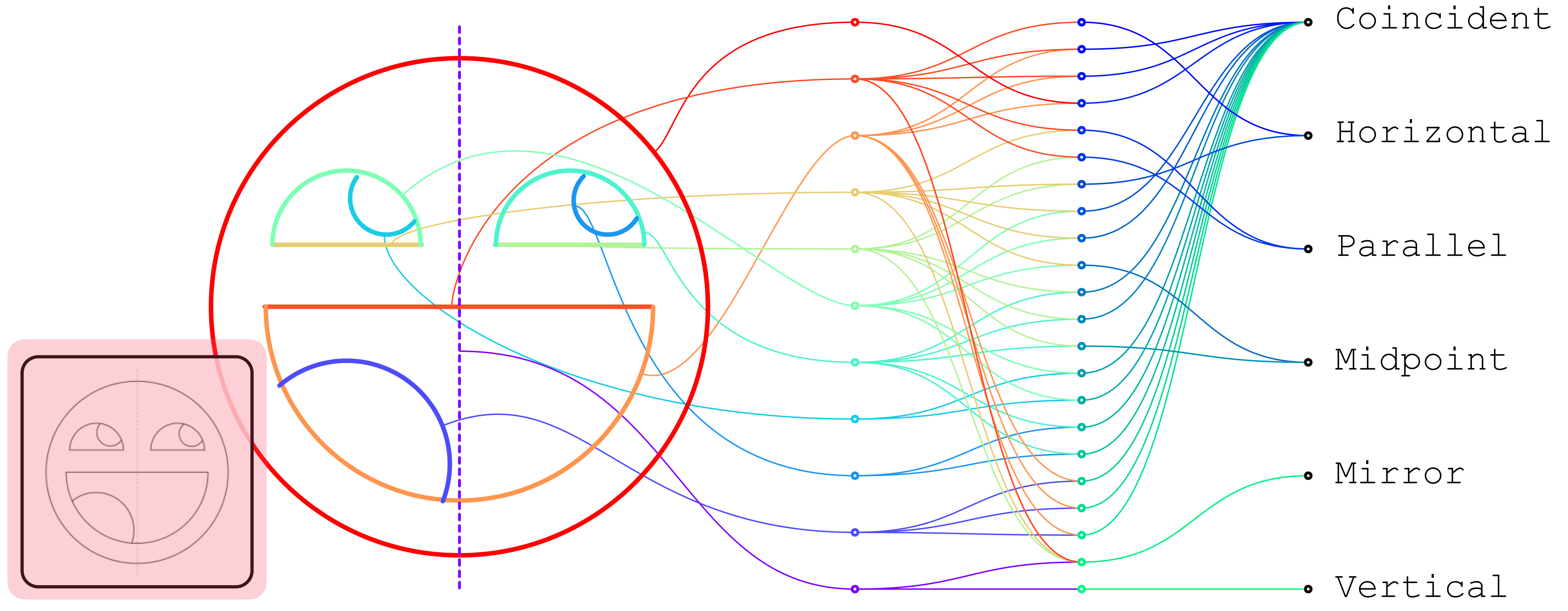


Results: out of distribution

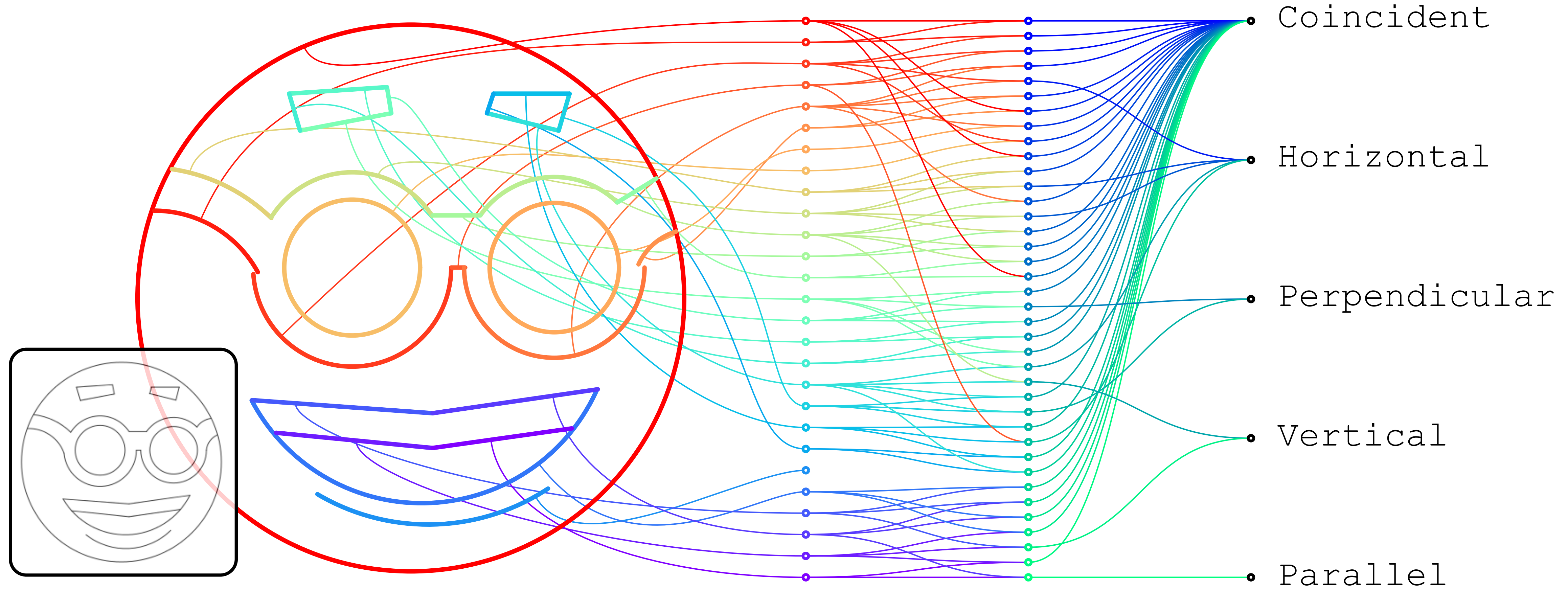


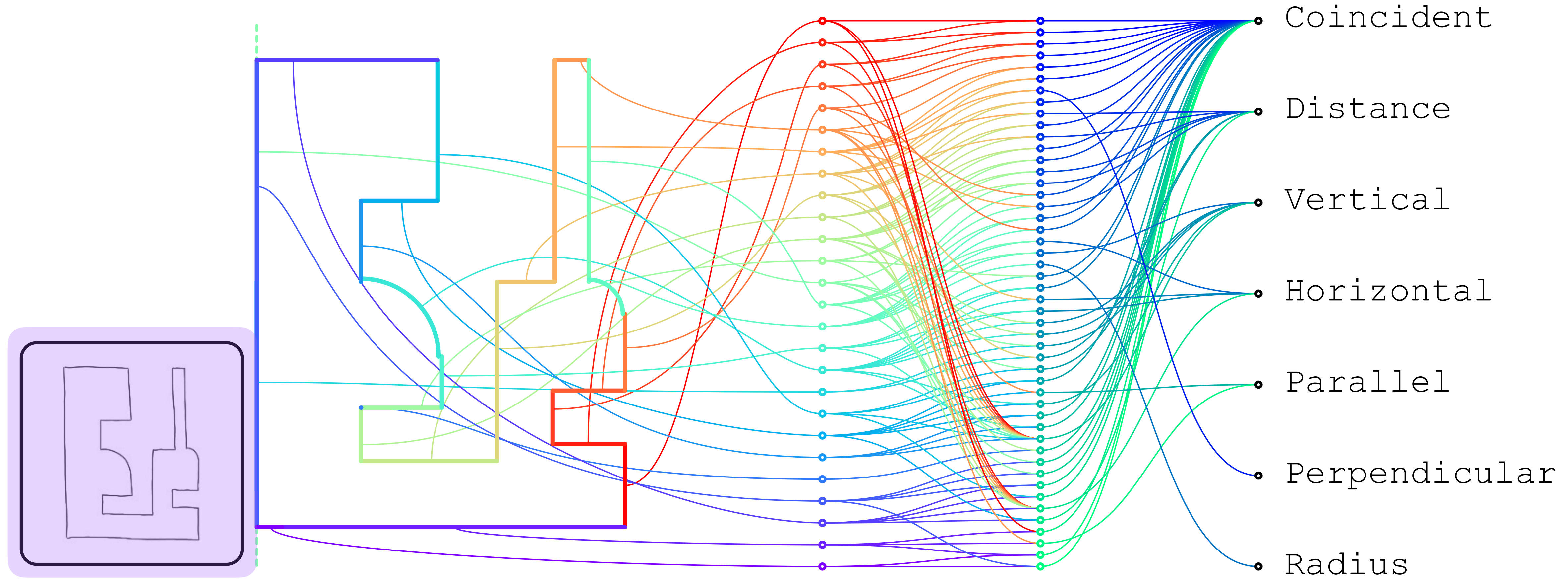


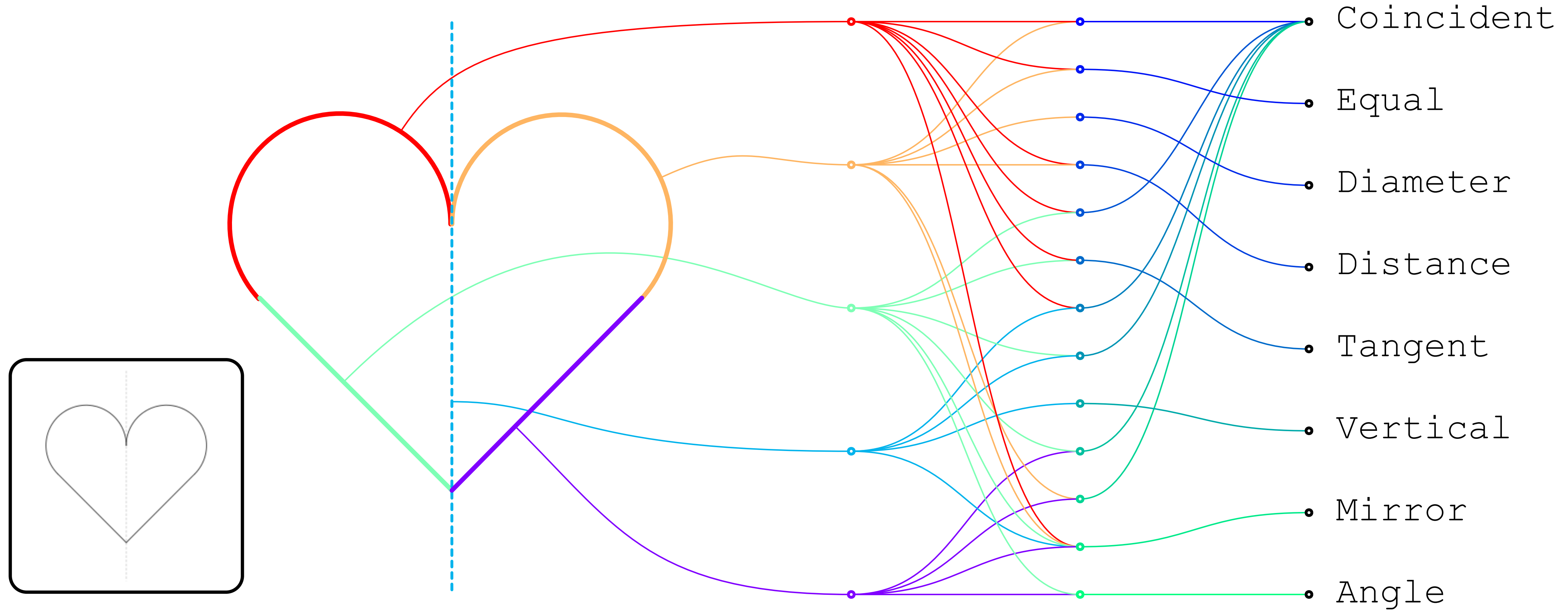


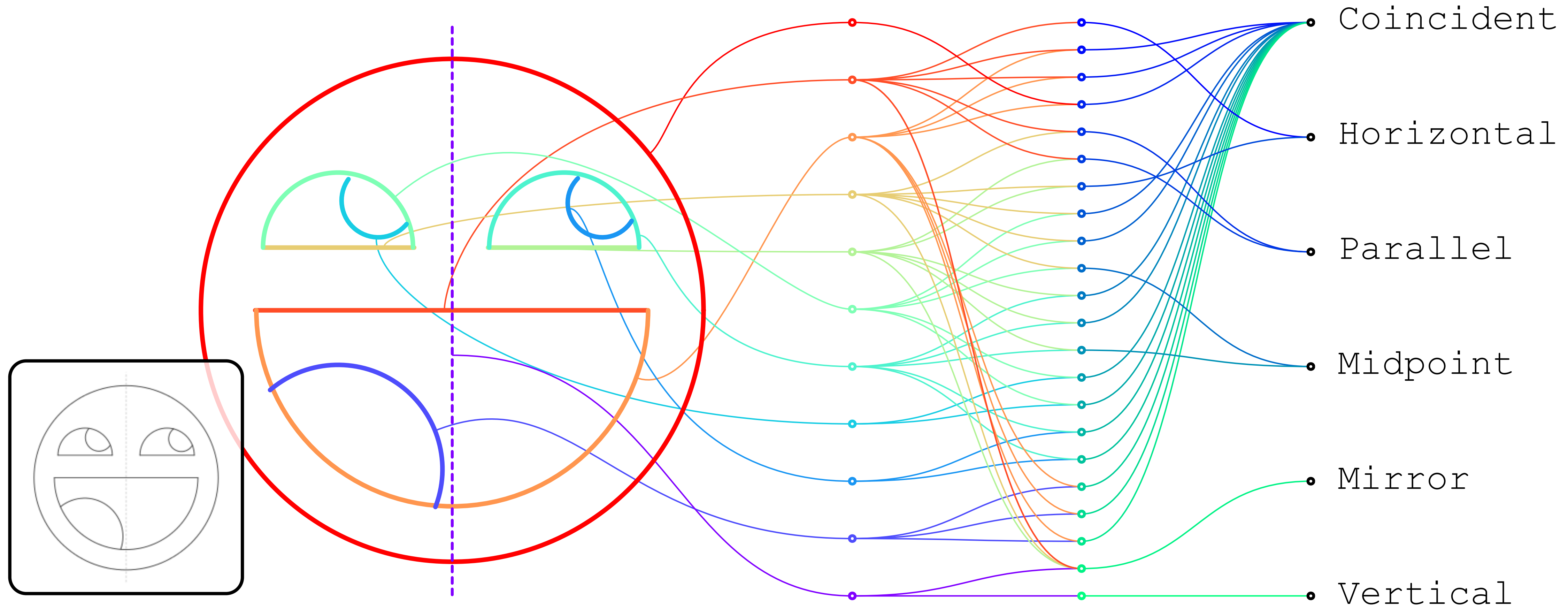


Results: out of distribution









Results: likelihoods

Model	Sequence	Avg bits per object	Avg bits per sketch
Step-independent uniform predictions	unordered bytes	112.23	3683.56
	unordered triplets	25.34	847.52
Vanilla Transformer on protos as raw bytes	concatenated	4.381	132.621
	interleaved	4.252	127.495
Interpreter-guided Transformer	concatenated	4.218	127.913
	interleaved	**4.103**	**123.213**
Image to sketch model	interleaved	1.570	53.730

Results: conditional model evaluation

Metric	Nearest neighbour baseline	Our model
Chamfer distance	2.97	0.59
# of entities discrepancy	4.67	1.10
# of constraints discrepancy	9.82	5.33

Private & Confidential

Related work

	Data	Entities	Constraints	3D
SketchGraphs (Seff et al., 2020) • Mostly advertises the dataset (a subset of what we use) • A GraphNet-base baseline that predicts entity types (not parameters)	Onshape (unfiltered)	✅ (no params)	✅ (only simple)	❌
CurveGen & TurtleGen (Willis et al., 2021) • A variation of PolyGen: generate vertices and then group them into curves • A Transformer for "turtle"-style graphics	Onshape (filtered)	✅	❌	❌
SketchGen (Para et al., 2021) • Most similar to our work • PolyGen-style generation: separate models for entities and constraints	Onshape (filtered)	✅	✅ (only simple)	❌
Fusion 360 Gallery (Willis et al., 2020) • Reconstruction of B-Reps • GraphNet predicting extrusions of profiles from the target shape	Fusion 360 (much smaller)	❌	❌	✅ (extrusions)
DeepCAD (Wu et al., 2021) • Generates sketch entities and extrusions (3D but very limited) • Transformer-based autoencoder for compound actions	Onshape (unfiltered)	✅	❌	✅ (extrusions)

Limitations

→ We support **only** a **subset** of entity and constraint **types**

Some constraints are difficult to implement or depend on external entities

→ The **dataset** is **biased** towards **simpler** sketches

→ The model is **not aware** of the sketch **solver**

Some generated sketches are unsolvable

→ At its core, it's a **standard autoregressive** language model

- Can't fix own mistakes
- Not permutation invariant

→ **Image-to-sketch** specific limitations

- Training bitmaps are synthetic renders rather than real drawings or scanned blueprints
- Struggles with fine details (probably due to patch-based perception)

→ **Loss of precision** due to 8-bit quantization

→ **Slow inference** due to device-host memory transfers at every step

Summary

→ A **general-purpose** framework for synthesizing **structured objects**

→ **Key ideas**

- Represent objects as **Protocol Buffer** messages
- Use a **state machine** to guide a Transformer-based **language model**
- Use **Pointer Nets** to refer to synthesized items

→ **Works well** for 2D **CAD** sketches

→ **Dataset is available:**
https://github.com/deepmind/deepmind-research/tree/master/cadl