Lightweight Diagramming for Lightweight Formal Methods: A Grounded Language Design (Artifact)

Siddhartha Prasad
Brown University, Providence, RI, USA

Ben Greenman
University of Utah, Salt Lake City, UT, USA

Tim Nelson
Brown University, Providence, RI, USA

Shriram Krishnamurthi
Brown University, Providence, RI, USA

Abstract

This artifact supports the paper "Lightweight Diagramming for Lightweight Formal Methods: A Grounded Language Design." It provides an implementation of Cope and Drag (`CnD`), a lightweight diagramming language grounded in cognitive principles and a corpus analysis of existing visualizations. `CnD` enables users to define diagrams using a small set of orthogonal primitives that capture essential domain structure while avoiding the complexity of writing full visualization programs. The artifact allows exploration of `CnD`'s implementation and reproduction of claims made in the accompanying paper.

2012 ACM Subject Classification Human-centered computing → Visualization toolkits; Software and its engineering → Constraint and logic languages; Software and its engineering → Specification languages

Keywords and phrases formal methods, diagramming, visualization, language design

Digital Object Identifier 10.4230/DARTS.11.2.12

Related Article Siddhartha Prasad, Ben Greenman, Tim Nelson, and Shriram Krishnamurthi, "Lightweight Diagramming for Lightweight Formal Methods: A Grounded Language Design", in 39th European Conference on Object-Oriented Programming (ECOOP 2025), LIPIcs, Vol. 333, pp. 26:1–26:29, 2025. `https://doi.org/10.4230/LIPIcs.ECOOP.2025.26`

Related Conference 39th European Conference on Object-Oriented Programming (ECOOP 2025), June 30–July 2, 2025, Bergen, Norway

Evaluation Policy The artifact has been evaluated as described in the ECOOP 2025 Call for Artifacts and the ACM Artifact Review and Badging Policy.

1 Content

This artifact includes the following files:

1. `artifact.pdf`: This document, which details how claims made in the paper can be reproduced.
2. `artifact-listings` : A directory which contains: `bt-error.cnd`, `fruit-error.cnd`, and `ring-lights-error.cnd`, `CnD` specs that can be used to verify the claims made in the paper's section 3.2 about unsatisfiable specifications. It also contains `fruit-icons.cnd`, a `CnD` spec that can be used to verify the claims made in the paper's section 5.2.2 about pictorial directives.
3. `cnd.tar` : A Docker tarball containing the `CnD` visualization tool. Instructions on how to load and run the tool are provided in section 8.
   a. `cnd-apple.tar` : A version of the same Docker image built for Apple silicon. If you are using an Apple silicon machine, please use this image instead of `cnd.tar`.
4. The `supp-cnd-and-data` directory, that contains `.cnd` layout files and corresponding alloy instance (`.xml`) files for each scenario visualized in the paper.

5. The `survey-instruments` directory, containing the following `.qsf` files, which are the Qualtrics survey instruments used in the user studies described in the paper's section 5.2:
   a. *Instance_Understanding.qsf*
   b. *Instance_Understanding_Diagrams.qsf*
   c. *Instance_Understanding_Icons.qsf*
6. The `survey-results` directory, which contains the questions and results of the studies in the paper's section 5.2 in CSV format. The directory contains the following files:
   a. `diagrams.csv`
   b. `icons.csv`
   c. `icons-ranking.csv`
   d. `badinstance.csv`
7. `prolificstudies.tar` : A Docker tarball containing scripts to confirm our experimental claims about these user studies.
   a. `prolificstudies-apple.tar` : A version of the same Docker image built for Apple silicon. If you are using an Apple silicon machine, please use this image instead of `cnd.tar`.

1.1 Required Software

In order to follow the instructions in this document, the reader will need to install the following software:

1. Docker : See `https://docs.docker.com/engine/install/` for installation instructions.
2. A web browser: `CnD` is a web application, and requires a web browser to run. We recommend using the latest version of Chrome or Firefox.

2 Scope

This artifact is intended to support the following experimental claims made in the accompanying paper: In particular, it enables reproduction of the following:

- The claim that `CnD` is able to produce a variety of interactive, constraint-based visualizations.
- The claims about unsatisfiable `CnD` specifications in the paper's section 3.2.
- The results of the user studies in the paper's section 5.2.
- The claims about the performance of `CnD` in the paper's section 5.3.

The artifact also includes Docker images for the `CnD` visualization tool and for reproducing the user study analyses, as well as layout and instance files used in the visualizations throughout the paper.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS).

The artifact is available at `https://zenodo.org/records/15351356`. The version of `CnD` presented in the accompanying paper is available, open source at GitHub release `https://github.com/sidprasad/copeanddrag/releases/tag/ecoop-25`, with associated commit hash `e36a2acc4e41ba6358e24ebf83128da641f508f5`.

4 Tested platforms

The artifact was tested on a Windows 10 machine with 16GB of memory, powered by a 2.30 GHz processor (AMD Ryzen 7), and 512 GB of disk space. Containers are built for amd64 architecture, but should be interoperable with arm64 architectures (e.g. Apple Silicon). We provide an arm64 equivalent container when necessary.

5 MD5 sum of the artifact

a6221177454f2cbbf678472986c1c9c2

6 Size of the artifact

2.3 GB

7 License

The artifact is available under license Creative Commons Attribution 4.0 International.

8 Paper Visualizations

The primary claim in the accompanying paper is that `CnD` is able to produce a variety of interactive, constraint-based visualizations. This section describes how to load paper examples in `CnD`, and then provides criteria by which the reader can evaluate the correctness of claims made about these visualizations.

1. Load the `CnD` Docker image by running the following command in a terminal:
   ```
   docker image load -i cnd.tar
   ```
   If you are using an Apple silicon machine, please use the following command instead:
   ```
   docker image load -i cnd-apple.tar
   ```
2. Start the `CnD` Docker container by running the following command in a terminal:
   ```
   docker run --rm -it -p 3000:3000 cnd:latest
   ```
   This will make `CnD` available at `http://localhost:3000`. If you would prefer to run `CnD` on a different port, run:
   ```
   docker run --rm -it -p <port of your choice>:3000 cnd:latest
   ```
3. A list of all paper examples should now be available at
   ```
   http://localhost:3000/example
   ```
   and specific examples can be accessed by either clicking on the example name or by navigating to:
   ```
   http://localhost:3000/example/<example_name>
   ```
   For instance, the **ab** visualization is available at `http://localhost:3000/example/ab`. Similarly, the error message shown in place of the **bt-dag** visualization (the paper's fig. 10) is available at `http://localhost:3000/example/bt-dag`.

Due to the nature of WebCola and D3 layout algorithms, the interactive visualizations may differ slightly from the images in the paper (e.g., link lengths may be larger or smaller). Crucially, however, every produced visualization is guaranteed to enforce the constraints in the accompanying `CnD` spec. We provide some concrete evaluation scenarios in sections 8.1 to 8.3.

The *Apply Layout* button on the top-right can be used to reapply the layout. We encourage the reader to play with the layout parameters to see how they affect the visualization.

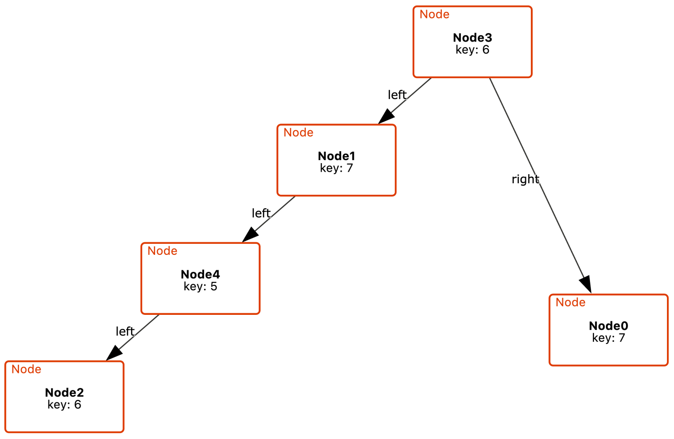

Figure 1 `CnD` visualization using orientation constraints. All nodes along the `left` field are laid below and to the left of their parent node, while nodes along the `right` field are placed to the right and below their parent node.

8.1 Verifying Orientation Constraints

The paper's section 3.1 describes how `CnD` can be used to enforce constraints of relative orientation between nodes. Load the **bt** example at `http://localhost:3000/example/bt`, and verify that it resembles fig. 1 (you may have to increase Diagram Compactness for a more compact layout).[1]

Verify that:

1. Tree nodes along the `left` relation are laid out to the bottom and left of their parents.
2. Tree nodes along the `right` relation are laid out to the bottom and right of their parents.
3. While you can drag nodes around, you cannot change their relative positions modulo these constraints.

In order to verify the claims made in the paper's section 3.2, paste the following unsatisfiable `CnD` spec (or copy it from `artifact-listings/bt-error.cnd`) into the editor and click the *Apply Layout* button:

```
constraints:
  - orientation: { field: left, directions: [right, left] }
directives:
  - attribute: { field: key }
  - flag: hideDisconnectedBuiltIns
```

Verify that this causes an error message to be displayed in terms of the directions `left` and `right`.

8.2 Verifying Cyclic Constraints

The paper's section 3.1 describes how `CnD` can be used to enforce cyclic constraints between nodes. Load the **ring-lights** example at `http://localhost:3000/example/ring-lights`, and click on the state labeled 1 in the Temporal Mini-Map. The image should now resemble fig. 2.

Verify that:

[1] The *Diagram Compactness* slider in the bottom left of the `CnD` controls panel can be used to adjust link lengths and node spacing.

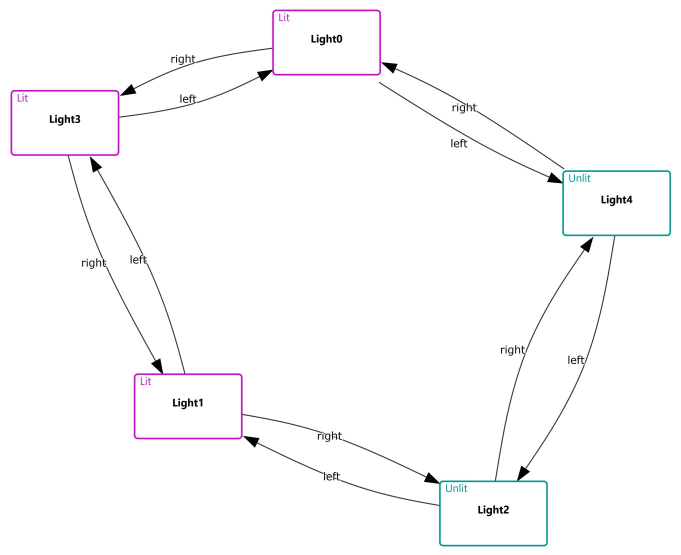

Figure 2 CnD visualization using a cyclic constraint on the `left` field.

1. The nodes are laid out in a cycle, with the `left` relation laid out in a clockwise direction.
2. While you can drag nodes around, you cannot change their relative positions in the cycle.

In order to verify claims in the paper's section 3.2, paste the following unsatisfiable CnD spec (or copy it from `artifact-listings/ring-lights-error.cnd`) into the editor and click the *Apply Layout* button:

```
constraints:
  - cyclic:
      field: left
      direction: clockwise
  - cyclic:
      field: right
      direction: clockwise
directives:
  - flag: hideDisconnected
```

And verify that you see an error stating that the instance being visualized is inconsistent with the layout constraints.

8.3 Verifying Grouping Constraints

The paper's section 3.1 describes how CnD can be used to enforce grouping constraints. Load the fruit example at `http://localhost:3000/example/fruit`. The image should now resemble fig. 3.

Verify that:

1. Nodes representing Fruit are placed in groups based on their position as denoted by the `fruit` relation.
2. While you can drag nodes around, you cannot change their groups.

In order to verify the claims made in the paper's section 3.2, paste the following unsatisfiable CnD spec into the editor (or copy it from `artifact-listings/fruit-error.cnd`) and click the *Apply Layout* button:

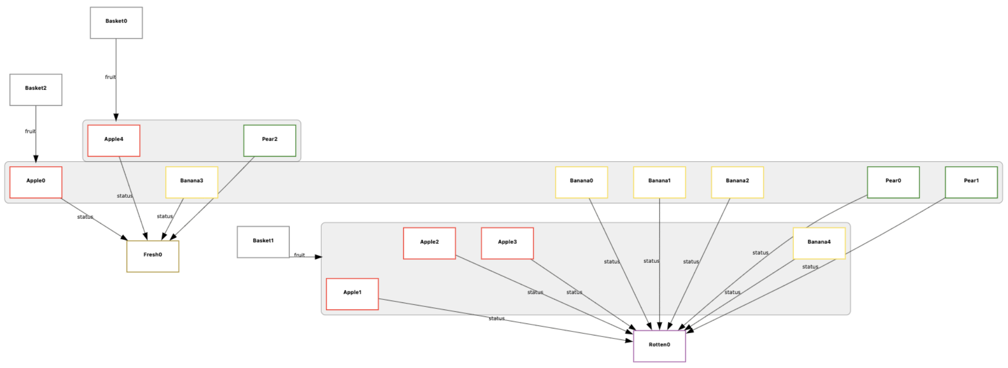

Figure 3 Fruit visualization, utilizing grouping constraints.

```
constraints:
  - group: {field: fruit, target : domain}
  - group: {field: status, target : range}
```

And verify that you see an error stating that the instance being visualized is inconsistent with the layout constraints.

8.4 Verifying Pictoral Directives

The paper's section 3.1 describes how CnD implements pictoral directives. Load the **fruit-icons** example at `http://localhost:3000/example/fruit-icons`.

Verify that each fruit is represented by a corresponding icon (i.e., Nodes of type Apple with Apple icons, Banana with Banana icons, and Pears with Pear icons).

Remove the Banana icon constraint by pasting in the following CnD spec (or copy it from `artifact-listings/fruit-icons.cnd`):

```
constraints:
  - group:
      field: fruit
      target: domain
directives:
  - flag: hideDisconnectedBuiltIns
  - color:
      sig: Banana
      value: '#FDDA0D'
  - color:
      sig: Apple
      value: red
  - color:
      sig: Pear
      value: green
  - color:
      sig: Basket
      value: grey
  - icon:
      sig: Apple
      icon:
        path: 'http://localhost:3000/img/apple.png'
        height: 70
        width: 70
  - icon:
      sig: Pear
      icon:
        path: 'http://localhost:3000/img/pear.png'
        height: 70
        width: 70
```

and verify that the Banana icon is no longer displayed.

9 Performance Data

The paper's section 5.3 makes claims about the performance of the CnD system when generating diagrams, emphasizing that the system generates diagrams within roughly 1 second at most. Table 1 provides the average time taken for each example diagram to be laid out by CnD. Each diagram was laid out 50 times on a machine with 16GB of memory, powered by a 2.30 GHz processor (AMD Ryzen 7).

Table 1 Time taken for Alloy instances to be laid out by CnD.

Scenario	Constraint Solving		Graph Layout		Total	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
ab	12.13 ms	3.07 ms	395.20 ms	11.22 ms	407.33 ms	11.46 ms
bt-dag	16.56 ms	20.38 ms	N/A	N/A	16.56 ms	20.38 ms
bt	19.24 ms	10.29 ms	325.93 ms	9.75 ms	345.17 ms	13.71 ms
cards	31.37 ms	7.73 ms	469.67 ms	11.50 ms	501.05 ms	14.23 ms
chord	29.60 ms	10.22 ms	955.25 ms	13.42 ms	984.85 ms	16.01 ms
face-inv	29.67 ms	7.72 ms	1032.47 ms	11.00 ms	1062.14 ms	13.53 ms
face	27.43 ms	8.45 ms	1007.28 ms	9.06 ms	1034.71 ms	12.03 ms
filesystem	30.30 ms	11.29 ms	456.80 ms	15.47 ms	487.10 ms	21.46 ms
fruit-icons	32.55 ms	10.29 ms	252.08 ms	14.03 ms	284.62 ms	19.32 ms
fruit	33.23 ms	9.59 ms	289.38 ms	268.37 ms	322.61 ms	268.90 ms
grid	26.68 ms	8.63 ms	446.16 ms	14.05 ms	472.84 ms	17.64 ms
phil-inv	44.63 ms	12.83 ms	457.94 ms	12.15 ms	502.57 ms	18.08 ms
phil	43.34 ms	13.34 ms	441.04 ms	10.46 ms	484.39 ms	17.50 ms
rc	47.96 ms	12.19 ms	193.81 ms	18.69 ms	241.77 ms	21.21 ms
ring-lights	43.54 ms	15.35 ms	492.47 ms	12.47 ms	536.02 ms	20.12 ms
ring-orientation	32.63 ms	9.96 ms	387.70 ms	12.42 ms	420.33 ms	17.88 ms
subway	22.98 ms	7.47 ms	411.93 ms	15.57 ms	434.89 ms	18.62 ms
ttt-inv	25.30 ms	8.35 ms	431.58 ms	13.10 ms	456.88 ms	18.35 ms
ttt	26.02 ms	8.51 ms	399.30 ms	11.78 ms	425.04 ms	15.47 ms
undirected-tree	19.27 ms	5.88 ms	84.46 ms	9.84 ms	103.76 ms	12.25 ms

The reader can test the performance of CnD in the Docker container by running the following command:

```
docker run --rm -it cnd:latest --benchmark <ex\_name> <times>
```

where `<ex_name>` is the name of the example to test, and `<times>` is the number of times the example should be run. For instance, to test the **bt** example over 10 runs, run:

```
docker run --rm -it cnd:latest --benchmark bt 10
```

To test *all* examples, run:

```
docker run --rm -it cnd:latest --benchmark --all <times>
```

While performance may differ depending on machine and docker resources, we believe that diagrams will be generated in roughly one second or less for all examples. **This does not mean that the benchmarking script will run in just a few seconds.** The benchmarking script may take a few minutes to run per example, since it repeatedly spins up a browser in the container to test layout performance. This issue may be further exacerbated if the machine running the container uses an architecture different from the container(linux/amd64).

If you need to stop the container in the midst of a run, we recommend using the `docker stop <container id>` command to stop the container cleanly. To identify the container id, use the `docker ps` command to list all running containers.

10 Study Instruments

The surveys in the paper's section 5.2 are provided in the `survey-instruments` directory, and are in the Qualtrics `.qsf` format. A QSF file can be imported into Qualtrics as detailed here:

`https://www.qualtrics.com/support/survey-platform/survey-module/survey-tools/import-and-export-surveys/`

Since this format is proprietary to Qualtrics, we also provide the survey questions and results in CSV format in the `survey-results` directory. The results of the surveys in the paper's section 5.2 are provided The files are as follows:

1. `diagrams.csv`: Results of the survey in the paper's section 5.2.1.
2. `icons.csv` and `icons-ranking.csv`: Results of the survey in the paper's section 5.2.2.
3. `badinstance.csv` : Results of the survey in the paper's section 5.2.3.

The experimental claims made in the paper's section 5.2 can be reproduced by first loading the Docker image in `prolificstudies.tar`.

```
docker image load -i prolificstudies.tar
```

If you are using an Apple silicon machine, please use the following command instead:

```
docker image load -i prolificstudies-apple.tar
```

10.1 Instance Understanding

```
docker run --rm -it prolificstudies diagrams
```

The first part of the **output** produces a table that substantiates the claims made in the paper's table 1 (we reproduce this table in table 2 for convenience).

The second part of the output substantiates the claims made in the paper's section 5.2.1 about the statistical significance of the results. (We restate these claims below for convenience.)

Table 2 Scenarios used to study the effectiveness of `CnD` constraints on spec understanding.

Scenario	Constraint	Correct Answer Percent		Mean Time on Task	
		default-graph	`CnD`	default-graph	`CnD`
cards	Cyclic	28 %	36 %	161.96s	128.81s
subway	Orientation	31 %	56 %	84.02s	121.73s
fruit	Grouping	69 %	74 %	60.15s	70.32s

Claim: Participants shown `CnD` diagrams were significantly more likely to get questions correct than those shown default-graph visualizations (62.75 % vs 48.04 % correct, $Z = 2.11$, $p < 0.05$), with a small-to-medium effect size ($d = 0.26$).

Output:

```
Proportions Correct: CnD: 0.6274509803921569, Default: 0.4803921568627451
Z-Score: 2.1127
p-Value: 0.03462
Cohen's D: 0.2632
```

10.2 Pictorial directives

```
docker run --rm -it prolificstudies icons
```

The output should substantiate the claims of statistical significance made in the paper's section 5.2.2 (reproduced below for convenience)

Output:

```
      73.33         86.67             68.55s            56.76s

Z-Score: 1.6
p-Value: 0.11

Timing Statistics (Raw Data):
Mean Time (Icons): 56.76s
Mean Time (Boxes): 68.55s

T-Test Results (Raw Data):
T-Statistic: -0.96
p-value: 0.34
```

Claim: Participants shown the diagram with pictorial directives answered questions correctly more often, and faster than those shown the diagram without them (86.67 % vs 73.33 % correct, mean time-on-task: 56.76s vs 68.55s). However, these were not statistically significant at a 95 % confidence level (correctness: $Z = 1.60$, $p = 0.11$, time-on task: $t = -0.96$, $p = 0.34$).

The second part of the output substantiates the ranking claims made in the paper's section 5.2.2:

Output:

```
Mean ranking, Default:  2.5666666666666667
Mean ranking, CnD Boxes:  2.0
Mean ranking, CnD Icon:  1.4333333333333333
Friedman test statistic: 19.266666666666665, p-value: 6.550832439925755e-05
            Default  CnD Boxes  CnD Icon
Default    1.000000   0.072064  0.000034
CnD Boxes  0.072064   1.000000  0.072064
CnD Icon   0.000034   0.072064  1.000000
```

Claim: Participants were then shown the fruit scenario with three different diagram styles: default-graph, default `CnD`, and `CnD` with pictorial directives, and asked to rank them in order of preference. Participants tended to rank the `CnD` with icons diagram highest (mean rank 1.45), followed by the `CnD` with default nodes (mean rank 2.00), and then the default-graph (mean rank 2.57). This difference in ranking was statistically significant (Friedman $Q = 19.27$, $p = 6.55e - 5$). A pairwise comparison showed that the ranking for `CnD` with icons was significantly higher than that of the default-graph ($p = 3.4e - 5$). Participants identified that the `CnD` with icons diagram conveyed the same information as `CnD` with default nodes, but still showed a slight preference for the former ($p = 0.07$).

10.3 Bad-Instances

```
docker run --rm -it prolificstudies badinstance
```

These should substantiate the claims of statistical significance made in the paper's section 5.2.3:

Output:

```
Proportion Correct (Default): 43.24%
Proportion Correct (CnD): 71.43%

Chi-squared Test Results:
Chi-squared: 4.73
p-value: 0.03
Degrees of Freedom: 1
```

Claim: Participants were significantly more likely to identify bad-instances and correctly explain *why* they were bad when shown `CnD` diagrams than when shown the default-graphs (71.43 % vs 43.24 %, $p = 0.03$, $\chi^2 = 4.73$).

11 Re-Usability

`CnD` is an open-source tool that serves as a substitute visualizer for Alloy and Forge. The version of `CnD` presented in the accompanying paper is available, open source at GitHub release `https://github.com/sidprasad/copeanddrag/releases/tag/ecoop-25`, with associated commit hash `e36a2acc4e41ba6358e24ebf83128da641f508f5`.

While this release includes a compiled version of the tool, a user can compile from source code by running:

```
npm run install
npm run build
npm run start
```

This requires an installation of Node.js and NPM, which can be found at `https://nodejs.org/en/`.

An interested reader can also find the latest version of `CnD` at `https://github.com/sidprasad/copeanddrag/` with up to date documentation at `https://sidprasad.github.io/copeanddrag/`.