

Rule2DRC: Benchmarking LLM Agents for DRC Script Synthesis with Execution-Guided Test Generation

Jinuk Kim Junsoo Byun Donghwi Hwang Seong-Jin Park Hyun Oh Song

Seoul National University · Samsung Electronics

ICML 2026

github.com/snu-mlab/Rule2DRC

Design rule checking is a manual bottleneck

Before a chip can be manufactured, its layout must satisfy **thousands of geometric design rules**.

- ▶ A foundry ships each rule as (i) a natural-language spec and (ii) an executable **DRC script**; a DRC engine runs the script on a layout and reports violations.
- ▶ Writing these scripts by hand needs a domain-specific language (KLayout, SVRF) *and* process expertise — and must be redone for every new node.

Natural-language rule r_i

Pad must be inside M4 and enclosed by M4 by at least $0.850\ \mu\text{m}$.



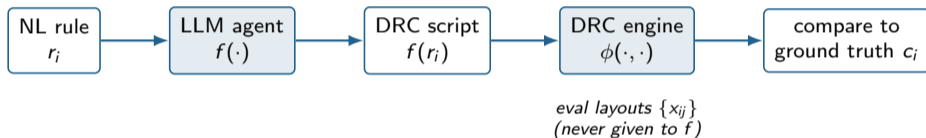
DRC script c_i (KLayout)

```
pad.enclosed(m4, 0.850.um)
  .output("PAD_M4_ENC", ...)
(pad - m4)
  .output("PAD_M4_ENC", ...)
```

Translating NL rules \rightarrow correct DRC scripts is the bottleneck we target with LLM agents.

Task: natural-language rule \rightarrow executable DRC script

An agent f sees **only** the NL rule r_i and outputs a script $f(r_i)$. We score it by **execution**, not code similarity.



- ▶ $\phi(x, c) \in \{0, 1\}$: does script c flag a violation on layout x ?
- ▶ **Success rate** — generated and ground-truth outputs match on *every* eval layout.
- ▶ **Error rate** — fraction of scripts that hit a compile / runtime error.

Prior benchmarks: small, code-similarity, closed

Benchmark	Test rules	Test layouts	Execution eval	No eval layouts as input	Open source
DRC-SG (2022/23)	200	–	✗	✓	✗
AST-Guided SVRF (2025)	74	–	✗	✓	✗
DRC-Coder (2025)	7	29	✓	✗	✗
Rule2DRC (Ours)	1000	13921	✓	✓	✓

Code similarity misranks correct scripts.

One rule has many syntactically different but *behaviorally identical* implementations:

`nw.interacting(...)` ✓

`hv_ndiff.separation(...)` ✓

`hv_ndiff.sep(...)` ✓

same rule \Rightarrow code sim. ✗, execution ✓

DRC-Coder uses execution, but needs ground-truth–labeled eval layouts *as agent input* — unrealistic in practice.

Rule2DRC: 1,000 tasks, 13,921 layouts, open-source

An order of magnitude larger than prior work, with execution-based scoring on KLayout / GDSII.

- ▶ **310** real rules extracted from the [SkyWater130 PDK](#), with hand-built corner-case layouts.
- ▶ **490** synthetic [multi-constraint](#) rules — chained, multi-layer operations for advanced (≤ 7 nm) nodes.
- ▶ **200** synthetic rules targeting [under-used grammar](#) — broadens DRC operator coverage.

Every task ships both passing and violating layouts

Including *hard negatives* that fail by the smallest margin (down to 1 nm) and topological corner cases. On average 13.9 eval layouts/task; 184 unique DRC methods across the benchmark.

Qualitative examples: one task per category

Task 0253

Natural Language Rule

n+_diff inside HVI must be inside hvntm and enclosed by hvntm by $\geq 0.185 \mu\text{m}$, excluding shapes overlapping areaid.ce.

Ground Truth DRC Script

```
... ndiff_in_hvi
= ndiff & hvi

hvntm.enclosing(
ndiff_in_hvi - ce,
0.185.um).outside(ce)
.output("HVNTM_NDIFF_...",
"hvntm enclosure...")

(ndiff_in_hvi - hvntm
- ce).output(
"HVNTM_NDIFF...",
"n+_diff inside HVI
...")
```

(a)

Task 0511

Natural Language Rule

This problem checks a min-spacing rule using a deep, multi-stage logic cascade.

Stage 1 builds the NWell reference set by selecting only those NWell polygons that are completely contained inside the areaid.ce region.

Stage 2 builds an allowed checking window by starting from the areaid.ce region and removing a POLY keepout zone created by expanding POLY by $0.30 \mu\text{m}$.

Stage 3 selects the candidate LVTN shapes by keeping only those LVTN polygons that touch or overlap the allowed checking window from Stage 2.

Stage 4 further filters the candidates to "dash-like" Manhattan shapes: only rectilinear candidates whose shorter bounding-box dimension is below $0.40 \mu\text{m}$ and whose longer bounding-box dimension is at most $1.20 \mu\text{m}$ remain...

Ground Truth DRC Script

```
... S1 = nwell.inside(ce)
poly_ko = poly.sized(t_poly_keepout)
S2 = ce - poly_ko
S3 = lvtm.interacting(S2)
S4 = S3.drc(rectilinear & (bbox_min < t_dash_w_max) &
(bbox_max <= t_dash_l_max)) ...
```

(b)

Task 0806

Natural Language Rule

This check looks at HV Nwell polygons and first classifies some of them as "sparse".

Sparseness is determined by computing, for each polygon, the ratio of the area of its bounding box to the polygon's own area; this ratio is one for a solid rectangle and increases when the polygon has cutouts or large empty regions inside its bounding box.

Only HV Nwell polygons whose ratio is greater than 1.25 are considered sparse...

Ground Truth DRC Script

```
... hvnw_sparse =
hvnw.drc(area_ratio >
1.25) ...
```

(c)

A (rule, script) pair from each category: (a) real SkyWater130 PDK rule, (b) synthetic multi-constraint, (c) synthetic niche-grammar.

Best-of- N : the hard part is *picking* the right script

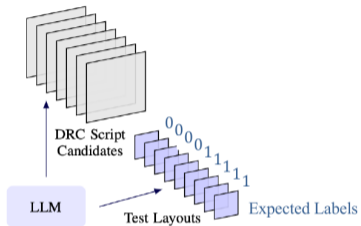
Sample N candidate scripts per task, then select the correct one.

- ▶ No ground-truth labels at selection time \Rightarrow the agent must **generate its own tests**.
- ▶ Weak tests leave many candidates with **identical outputs** — correct and incorrect scripts hide in the same cluster.

Where prior selectors fall short

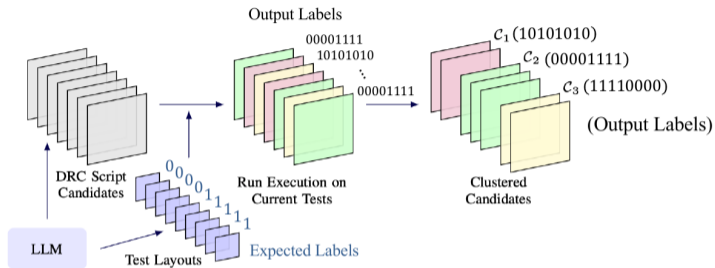
- ▶ **CodeMonkey** prunes to the top-3 early — can discard a candidate that only becomes distinguishable after more tests.
- ▶ **S*** spends tests on candidates that are *already* separated, leaving indistinguishable groups unexplored.

SplitTester: split indistinguishable clusters with targeted tests



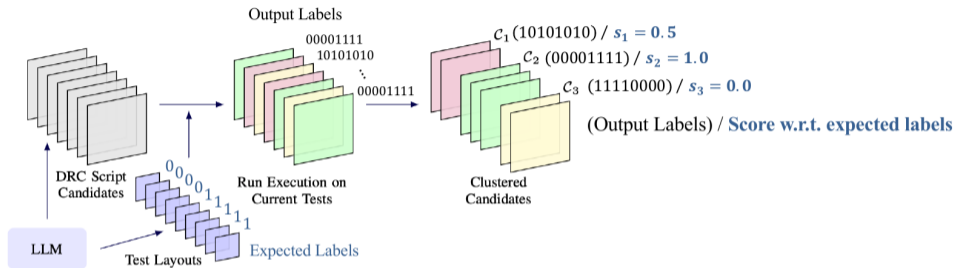
1. Generate test layouts + expected labels.

SplitTester: split indistinguishable clusters with targeted tests



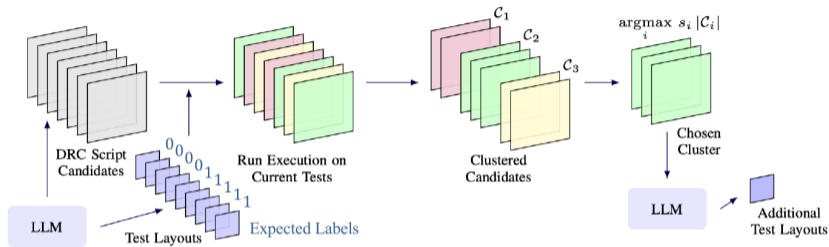
1. Generate test layouts + expected labels.
2. Run all candidates; cluster by identical outputs.

SplitTester: split indistinguishable clusters with targeted tests



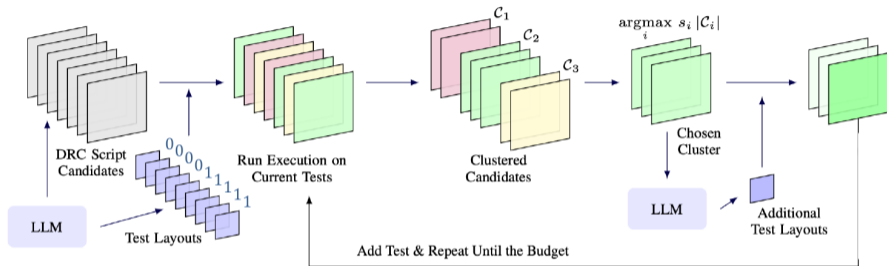
1. Generate test layouts + expected labels.
2. Run all candidates; cluster by identical outputs.
3. Score each cluster vs. expected labels (s_i).

SplitTester: split indistinguishable clusters with targeted tests



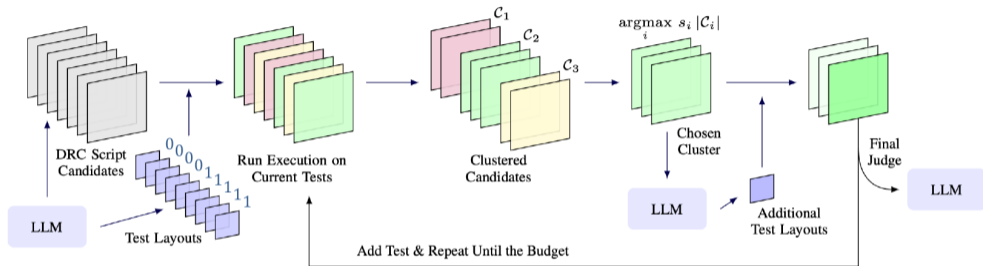
1. Generate test layouts + expected labels.
2. Run all candidates; cluster by identical outputs.
3. Score each cluster vs. expected labels (s_i).
4. Split the large, high-scoring cluster ($\text{argmax}_i s_i |C_i|$) with new tests.

SplitTester: split indistinguishable clusters with targeted tests



1. Generate test layouts + expected labels.
2. Run all candidates; cluster by identical outputs.
3. Score each cluster vs. expected labels (s_i).
4. Split the large, high-scoring cluster ($\text{arg max}_i s_i |C_i|$) with new tests.
5. Re-cluster; repeat until the test budget is spent.

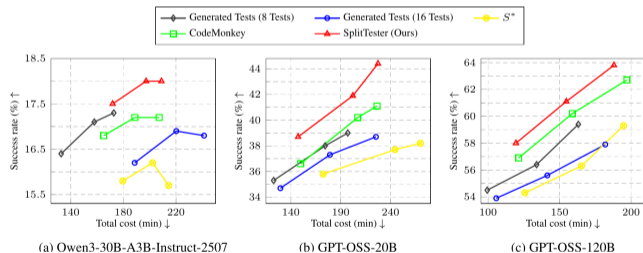
SplitTester: split indistinguishable clusters with targeted tests



1. Generate test layouts + expected labels.
2. Run all candidates; cluster by identical outputs.
3. Score each cluster vs. expected labels (s_i).
4. Split the large, high-scoring cluster ($\text{argmax}_i s_i |C_i|$) with new tests.
5. Re-cluster; repeat until the test budget is spent.
6. Judge LLM picks the best of the top-3.

SplitTester wins across models and budgets

Selector (Best-of-20)	Qwen3-30B-Instruct		GPT-OSS-20B		GPT-OSS-120B	
	Succ.↑	Err.↓	Succ.↑	Err.↓	Succ.↑	Err.↓
Single sample	14.1	61.9	16.9	66.9	32.5	48.5
CodeMonkey (best baseline)	17.2	38.6	41.1	14.1	62.7	4.2
SplitTester (Ours)	18.0	35.1	44.4	12.6	63.8	4.1
Oracle@20 (upper bound)	23.7	29.5	53.8	11.6	72.1	3.7



Pareto-optimal under runtime. **Context:** in-context KLayout API docs add +20%p pass@1 / +40%p pass@20 (used in all runs).

Conclusion

- ▶ **Rule2DRC** — the first **large-scale** (1,000 tasks / 13,921 layouts), **execution-based**, **open-source** benchmark for NL-rule → DRC script synthesis.
- ▶ **SplitTester** — clusters candidates by behavior and generates tests that **split previously indistinguishable clusters**, giving the best Best-of- N selection across three models while staying on the Pareto frontier.
- ▶ Benchmark, evaluation pipeline, and methods are open-sourced.

Thank you!

`github.com/snu-ml1ab/Rule2DRC`