

Construct



Analyze



Repair



More reliable realization!



(a) Contract-Matched Interface Construction

A. Shared Semantic Contract

$a_t = (u, v)$

operation: go
argument: destination
type: room

Other operations:



LOOK



TAKE



OPEN

...

B. Surface Rendering

(GO, living room)



structured call

```
{  
  "tool": "go",  
  "destination": "living room"  
}
```

terminal cmd

```
> go --destination  
  "living room"
```

C. Decode to Shared Semantics



MCP surface



CLI surface



D_{MCP}

D_{CLI}

$\hat{a}_t : (\text{GO, living room})$

Shared Runtime



Validate



Compile



Execute

Same semantic action, different surface realizations.

(b) Local Sensitivity & Trajectory Propagation

1 Local Interface Sensitivity

Frozen matched context



state s_t



history h_t

restore exactly

Interface factorization

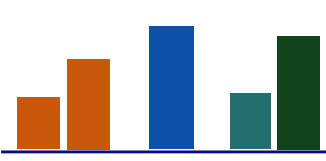
ϕ
(presentation)

R
(realization)

ψ
(feedback)

$2 \times 2 \times 2 = 8$ matched conditions

Response shift



Action Realization Admission

2 Paired Natural Trajectories

common prefix

t^*

MCP trajectory

CLI trajectory

Downstream analysis



History



State



Recovery

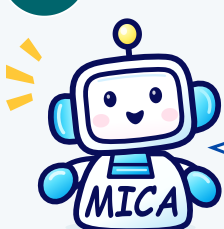


Cost

Local attribution at matched contexts, propagation analysis after t^* .

(c) DART: Commitment-Preserving Realization

1 Triggered Semantic Selection



(GO, living room)

2 Validate & Commit



$c_t = (k_t, \hat{a}_t)$



valid



commit

keep fixed!

3 Realize & Check



MCP realization
{ ... }



CLI realization
> go --destination living room



Match

failure

realization failure

4 Repair or Release → Execute

Valid(c_t)?

Yes



Targeted Repair
repair expression only

No



Release to Retry
ordinary policy

Execute



Environment

Repair the expression, not the decision.

Same Commitment
Repaired Realization!

Re-realize with same semantic commitment