Shared Representation

CS 347

Jiaju Ma

Announcements

Quiz after lecture today — pencils!

Previously

AI vs. IA

Direct manipulation vs. Agents

Mixed-initiative interaction

Human and machine should collaborate, but *over what substrate*?

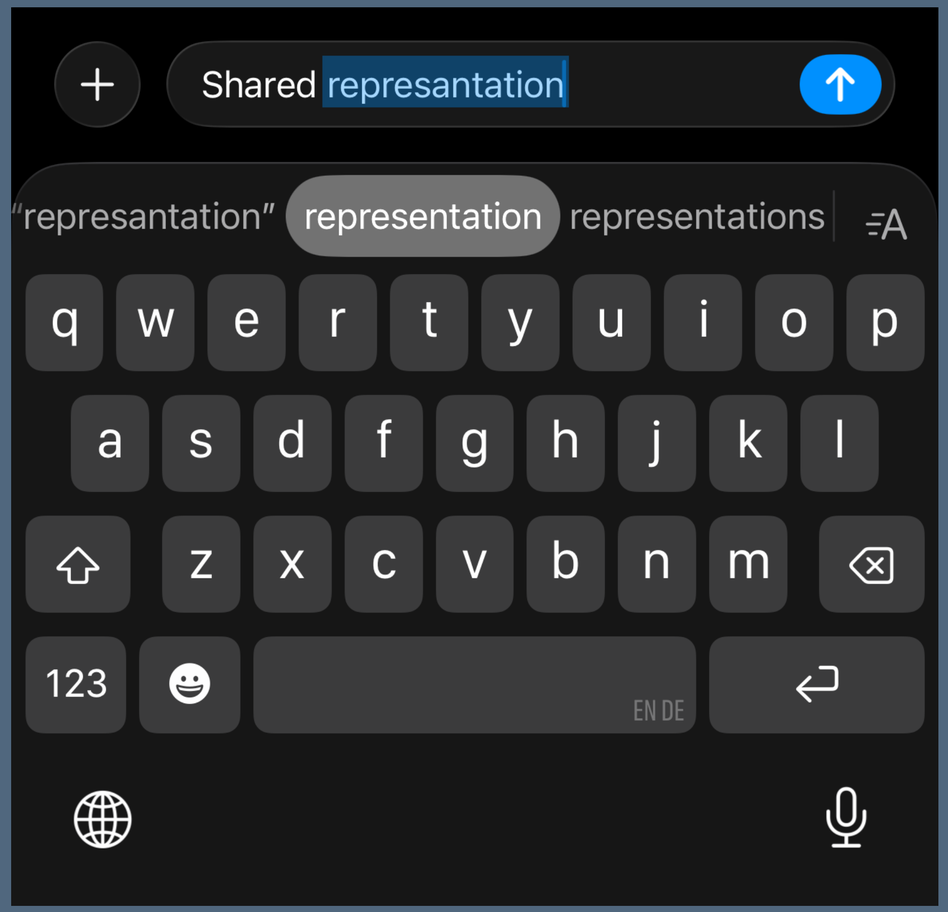
Agency plus Automation

[Heer 2019]

[Shared representation]

A medium that both human and machine can read, write, and modify

Autocorrect



Autocorrect

User intention — The user wants to spell a word correctly

User input — The user has typed a word that might contain typos

Machine output — The machine suggests the correct spelling

Shared representation — The word typed by the user

Designing shared representation

1. The shared representation can directly be *in the same format* as the user input and machine output

ShadowDraw

[Lee 2011]

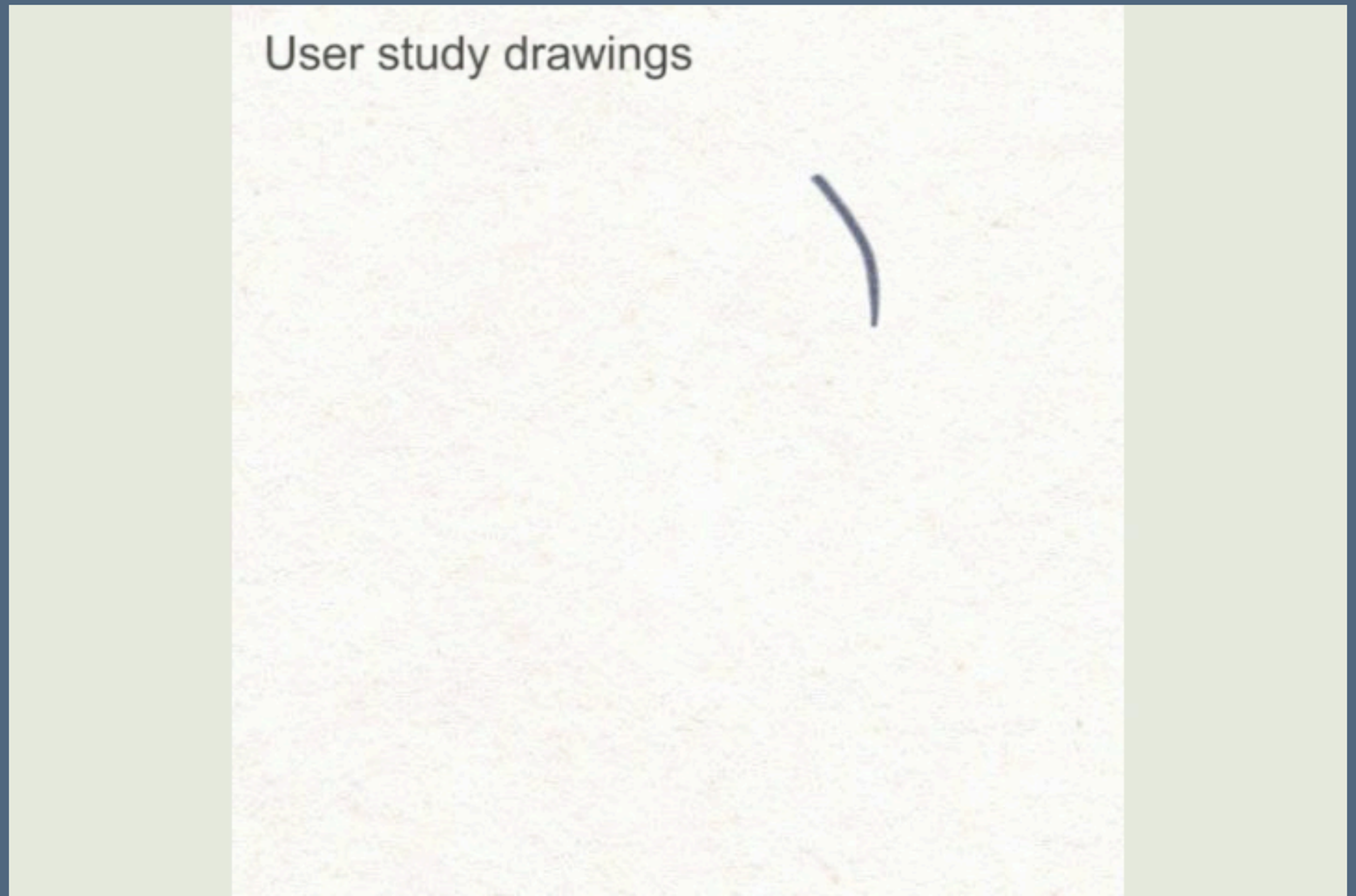


Image Analogies

[Hertzmann 2001]

User input

Machine output

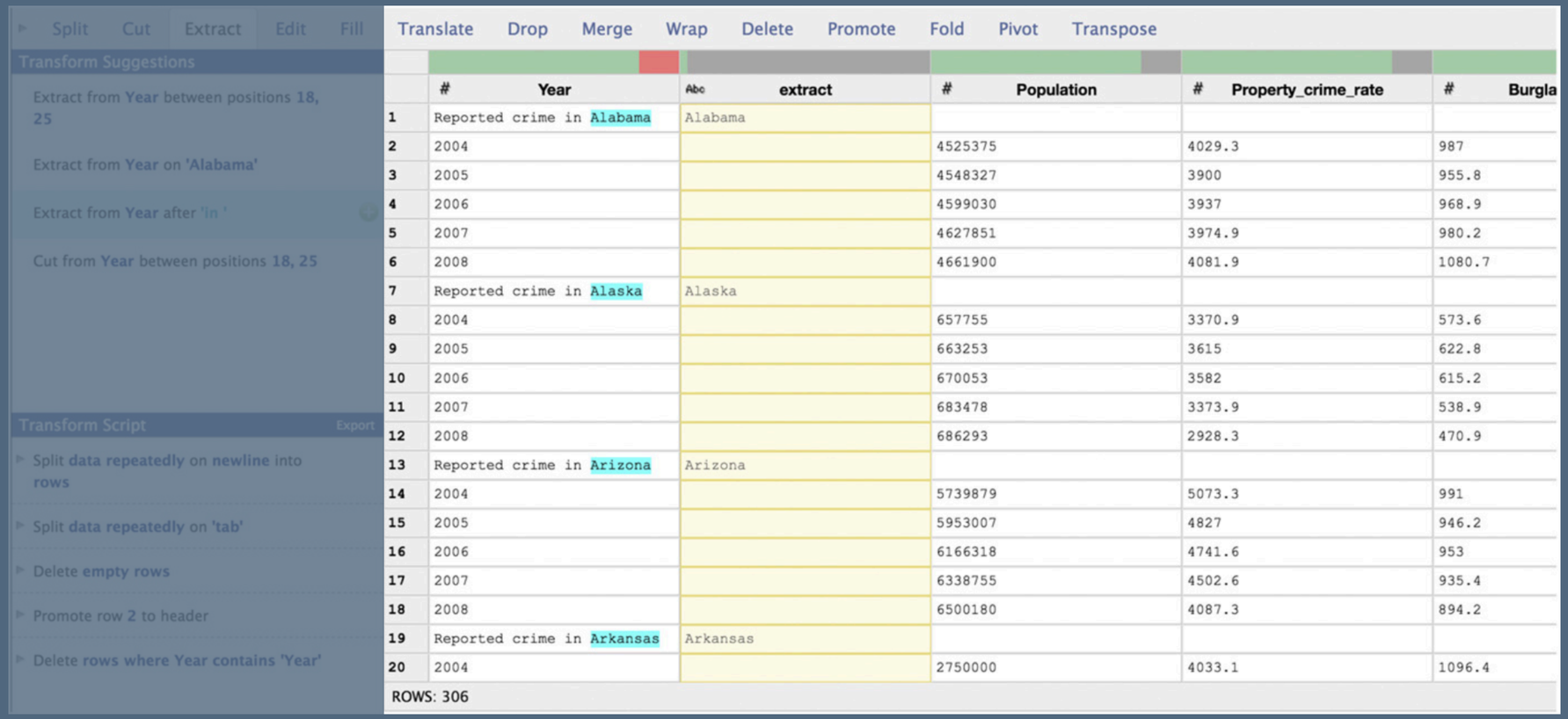
Designing shared representation

1. The shared representation can directly be *in the same format* as the user input and machine output

What if the input and the output are *not* in the same format?

Data Wrangler

[Heer 2019]

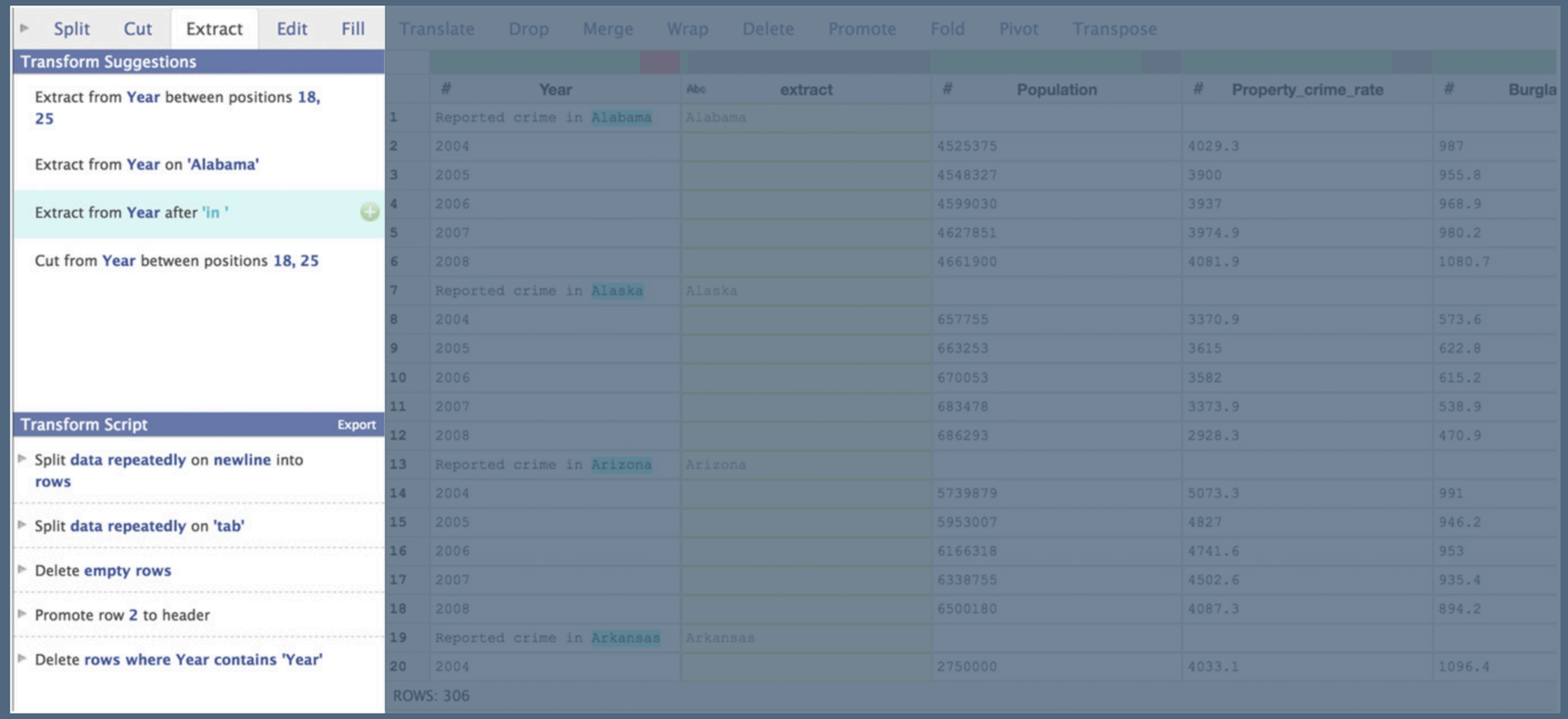


	# Year	Abc extract	# Population	# Property_crime_rate	# Burgla
1	Reported crime in Alabama	Alabama			
2	2004		4525375	4029.3	987
3	2005		4548327	3900	955.8
4	2006		4599030	3937	968.9
5	2007		4627851	3974.9	980.2
6	2008		4661900	4081.9	1080.7
7	Reported crime in Alaska	Alaska			
8	2004		657755	3370.9	573.6
9	2005		663253	3615	622.8
10	2006		670053	3582	615.2
11	2007		683478	3373.9	538.9
12	2008		686293	2928.3	470.9
13	Reported crime in Arizona	Arizona			
14	2004		5739879	5073.3	991
15	2005		5953007	4827	946.2
16	2006		6166318	4741.6	953
17	2007		6338755	4502.6	935.4
18	2008		6500180	4087.3	894.2
19	Reported crime in Arkansas	Arkansas			
20	2004		2750000	4033.1	1096.4

User input

Data Wrangler

[Heer 2019]



Machine output

Data Wrangler

[Heer 2019]

User intention — The user wants to transform data

User input — Data frame + selection

Machine output — The machine suggests potential transformations

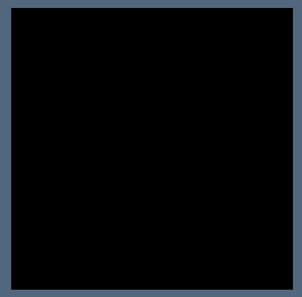
Shared representation — A DSL of data transformations

Designing shared representation

1. The shared representation can directly be *in the same format* as the user input and the machine output

2. The shared representation can be an intermediate layer between the user input and the machine output when 1. is not feasible

“three dogs in the image”



Is this what I asked for?

“three dogs in the image”

VPEval

[Cho 2023]

“three dogs in the image”

Visual Program

```
countEval(img, “dog”, “==3”)
```

VPEval

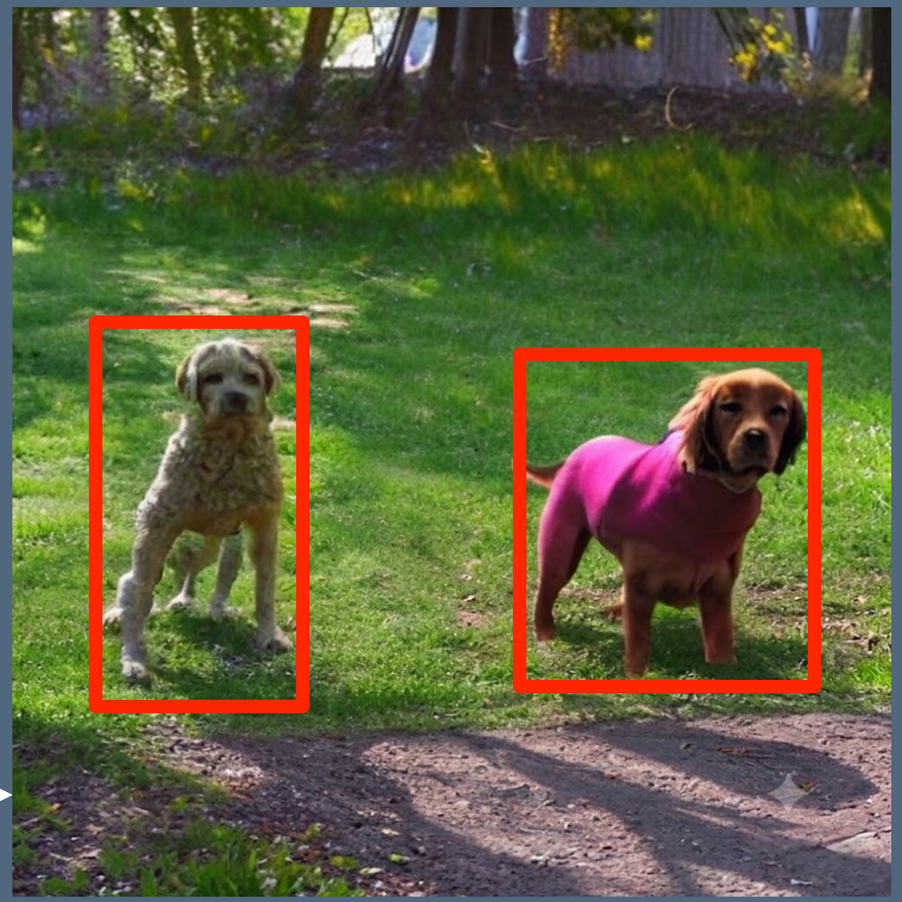
[Cho 2023]

"three dogs in the image"

Visual Program

```
countEval(img, "dog", "==3")
```

```
>>> countEval(img, "dog", "==3")
False
```



Program as shared representation

Encodes action space — Common set of actions a user can perform and the machine can search over

User understandable and editable — User can look at the program to reason about the output

Machine executable — Unambiguous, formal, directly executable

ViperGPT

[Menon 2023]

Query: how many muffins can each kid have for it to be fair?

Input Image:

ViperGPT

[Menon 2023]

Query: how many muffins can each kid have for it to be fair?

Code Generation

Input Image:

```
def execute_command(image)->str:
    image_patch = ImagePatch(image)
    muffin_patches = image_patch.find("muffin")
    n_muffins = len(muffin_patches)
    kid_patches = image_patch.find("kid")
    n_kids = len(kid_patches)
    result = str(n_muffins // n_kids )
    return result
```

ViperGPT

[Menon 2023]

Query: how many muffins can each kid have for it to be fair?

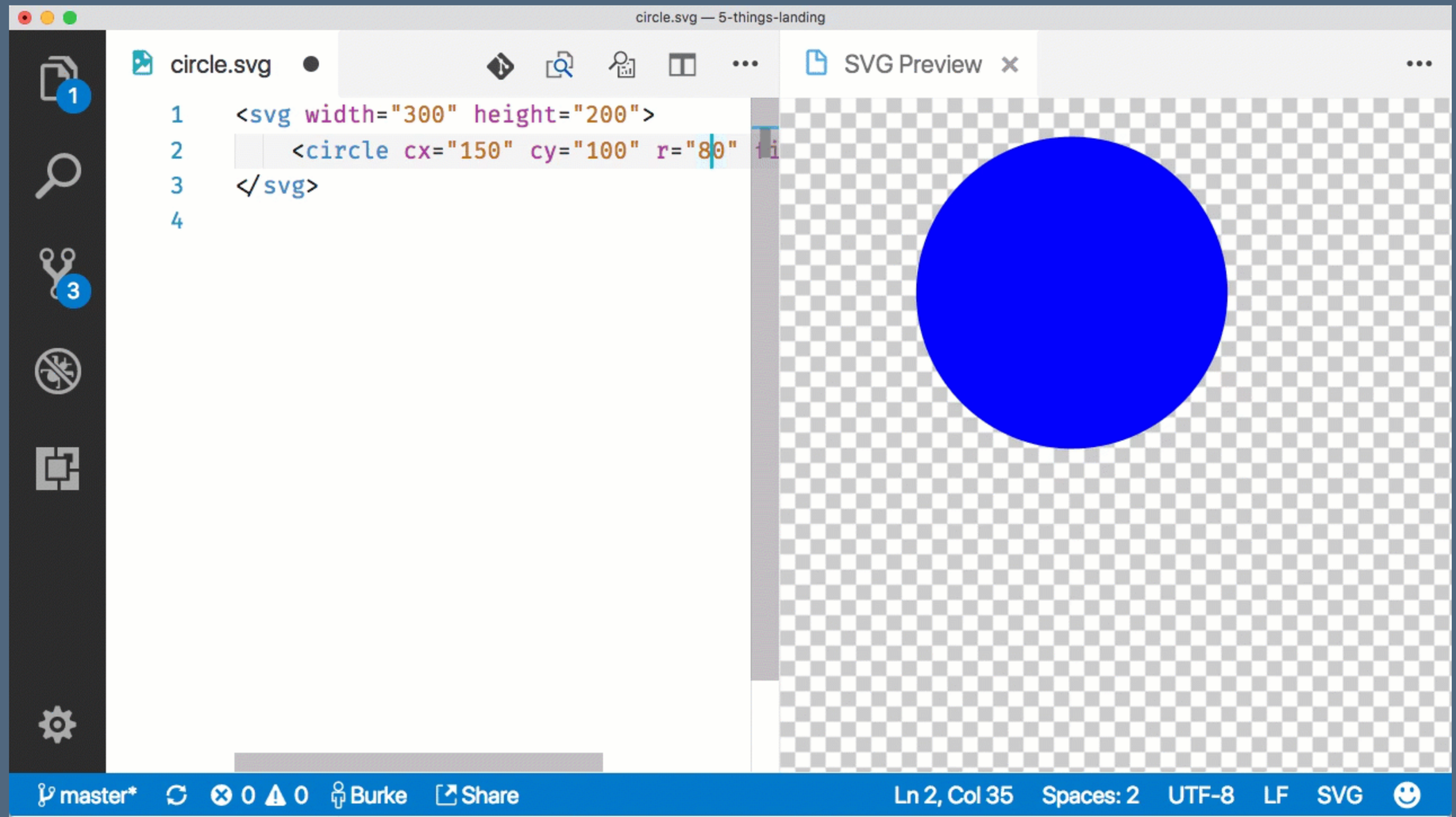
Code Generation

Input Image:

```
def execute_command(image)->str:
    image_patch = ImagePatch(image)
    muffin_patches = image_patch.find("muffin")
    n_muffins = len(muffin_patches)
    kid_patches = image_patch.find("kid")
    n_kids = len(kid_patches)
    result = str(n_muffins // n_kids )
    return result
```

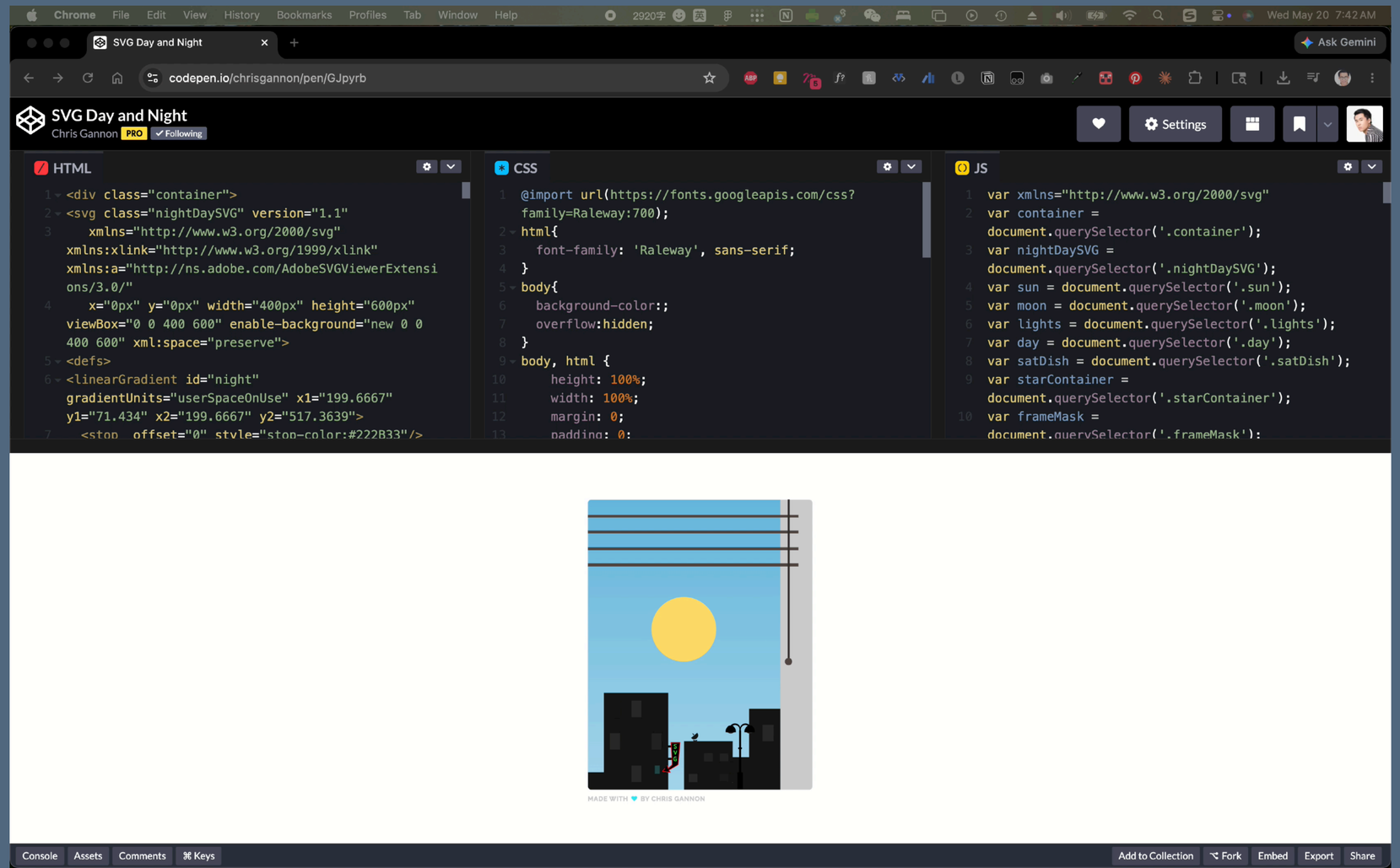
Image as program

[Holland 2018, Medium]



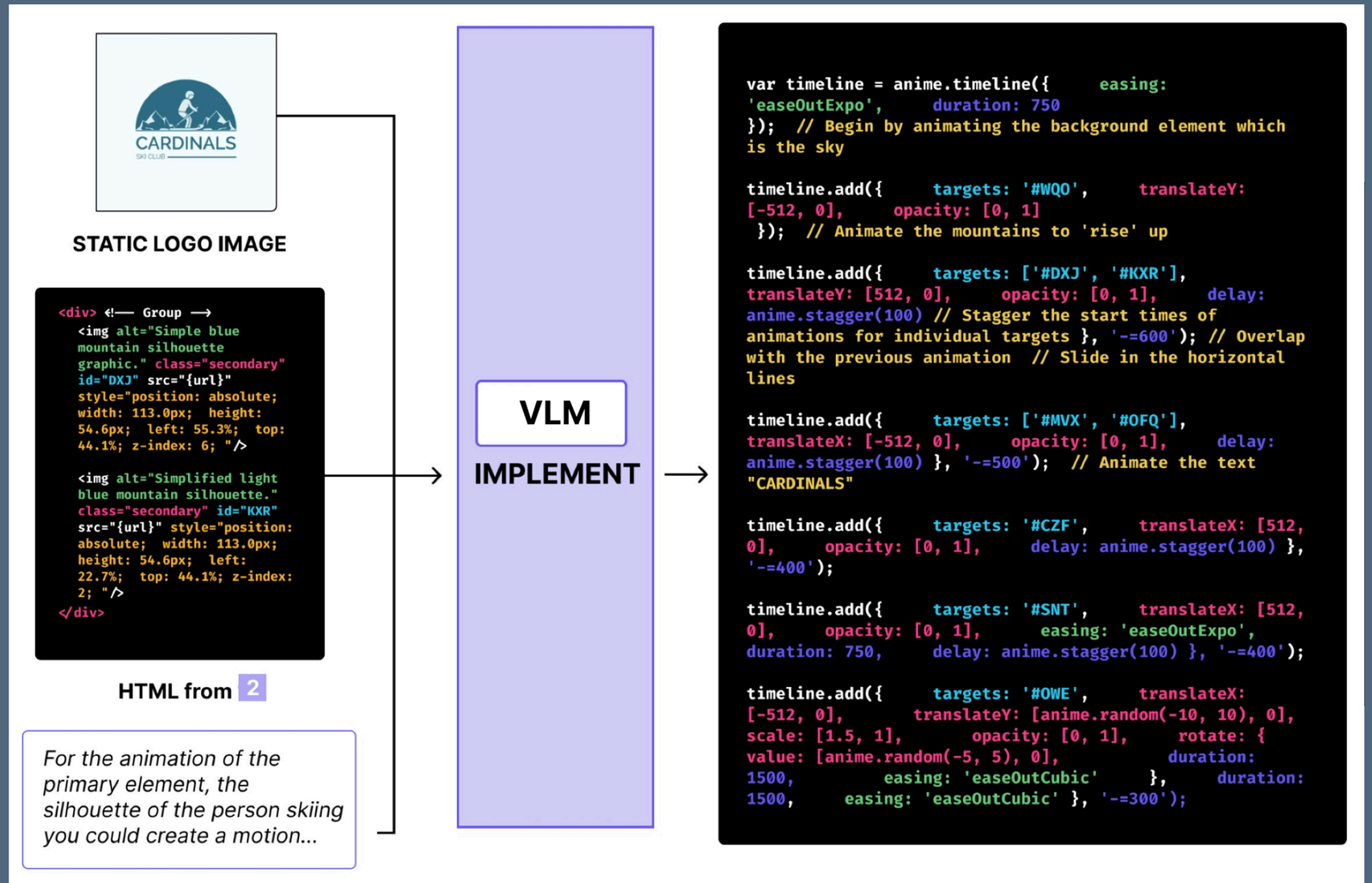
Animation as program

[Gannon 2017, CodePen]



LogoMotion

[Liu 2025]



Formal representation of natural language

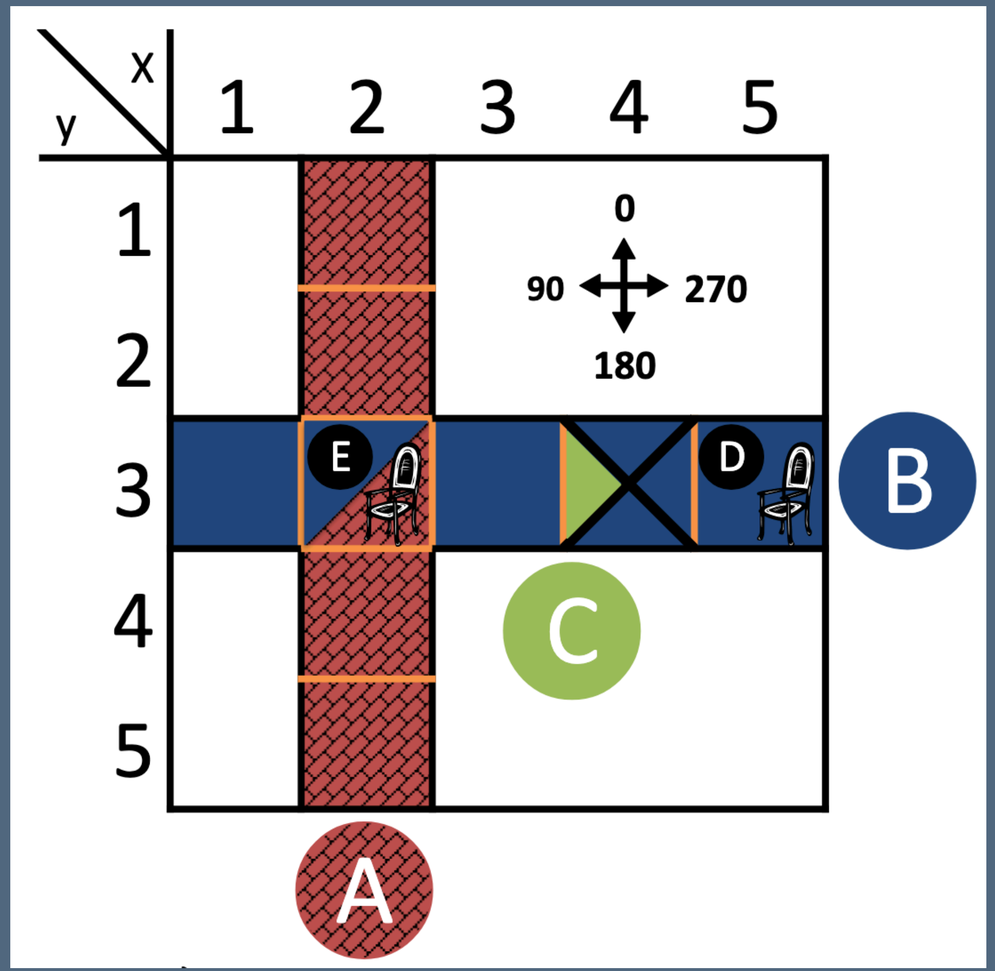
Natural language is ambiguous, underspecified, and hard for machines to reason over directly.

Long-standing approach in linguistics and AI: represent meaning as logical expressions with variables, predicates, and quantifiers.

"A red ball is on the table." → $\exists x.\ \text{ball}(x) \wedge \text{red}(x) \wedge \text{on}(x, \text{table})$

Instructions to Actions

[Artzi and Zettlemoyer 2013]



{D E}
(a) chair
$\lambda x.chair(x)$

{A B}
(b) hall
$\lambda x.hall(x)$

E
(c) the chair
$\iota x.chair(x)$

C
(d) you
you

{B}
(e) blue hall
$\lambda x.hall(x) \wedge blue(x)$

{E}
(f) chair in the intersection
$\lambda x.chair(x) \wedge$
$intersect(\iota y.junction(y), x)$

{A B E}
(g) in front of you
$\lambda x.in\_front\_of(you, x)$

Instructions to Actions

[Artzi and Zettlemoyer 2013]

move forward twice to the chair

$$\lambda a.move(a) \wedge dir(a, forward) \wedge len(a, 2) \wedge to(a, \iota x.chair(x))$$

at the corner turn left to face the blue hall

$$\lambda a.pre(a, \iota x.corner(x)) \wedge turn(a) \wedge dir(a, left) \wedge post(a, front(you, \iota x.blue(x) \wedge hall(x)))$$

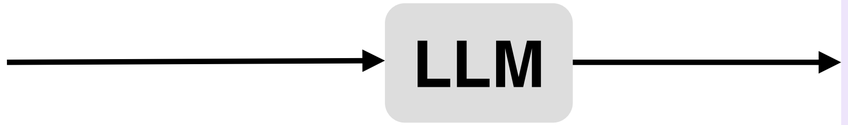
move to the chair in the third intersection

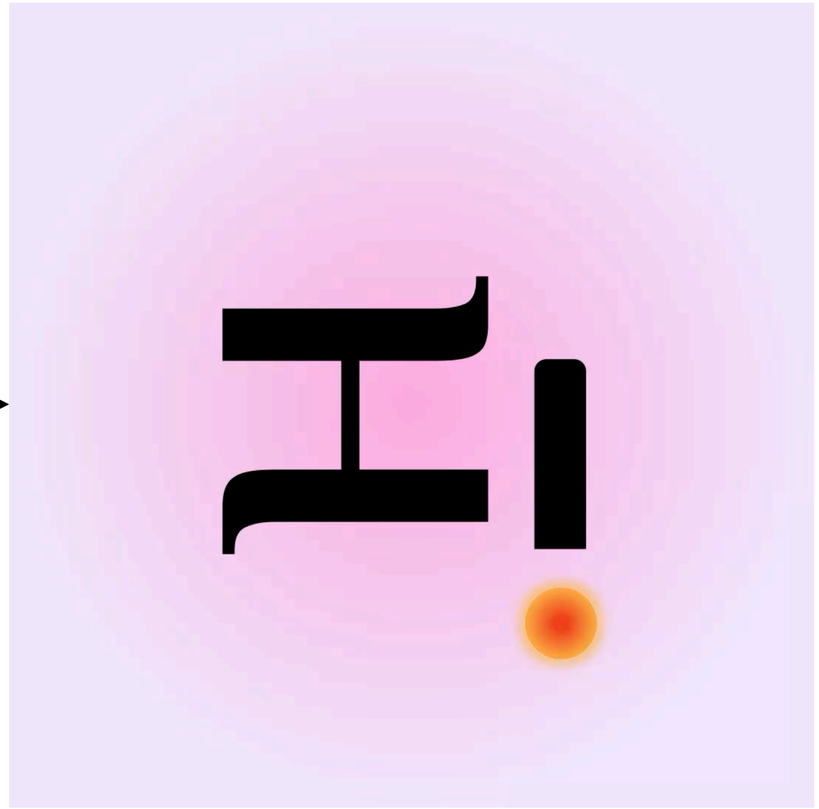
$$\lambda a.move(a) \wedge to(a, \iota x.sofa(x)) \wedge intersect(order(\lambda y.junction(y), frontdist, 3), x)$$

MoVer

[Ma and Agrawala 2025]

The orange circle translates above the rectangular shape. In the meantime, the letter H rotates 90 degrees clockwise.

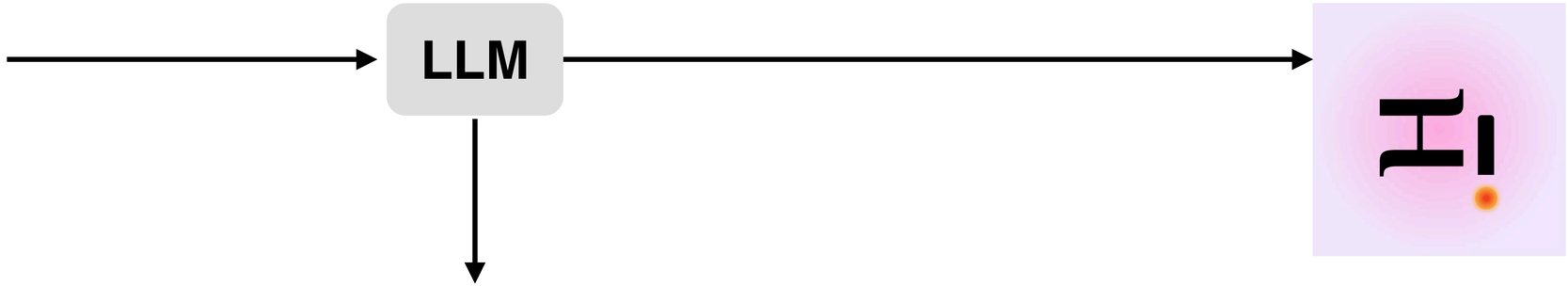




MoVer

[Ma and Agrawala 2025]

The orange circle translates above the rectangular shape. In the meantime, the letter H rotates 90 degrees clockwise.



o_1 = ιo.clr(o,"orange")∧shp(o,"circle")
o_2 = ιo.shp(o,"rectangle")
o_3 = ιo.id(o,"H")

m_1 = ιm.type(m,"trn")∧agt(m,o_1)∧post(m,top(o_1,o_2))
m_2 = ιm.type(m,"rot")∧agt(m,o_3)∧dir(m,"cw")∧mag(m,90)

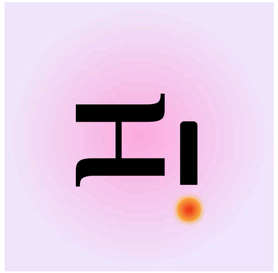
while(m_1,m_2)

MoVer

[Ma and Agrawala 2025]

The orange circle translates above the rectangular shape. In the meantime, the letter H rotates 90 degrees clockwise.

LLM



o_1 = ιo.clr(o,"orange")∧shp(o,"circle") — T

o_2 = ιo.shp(o,"rectangle") — T

o_3 = ιo.id(o,"H") — T

MoVer Execution Engine

m_1 = ιm.type(m,"trn")∧agt(m,o_1)∧post(m,top(o_1,o_2)) — F

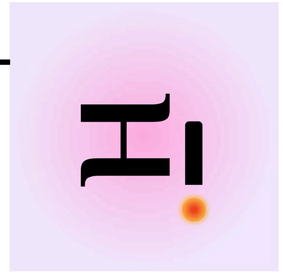
m_2 = ιm.type(m,"rot")∧agt(m,o_3)∧dir(m,"cw")∧mag(m,90) — T

while(m_1,m_2) — F

MoVer

[Ma and Agrawala 2025]

The orange circle translates above the rectangular shape. In the meantime, the letter H rotates 90 degrees clockwise.



LLM

o_1 = ιo.clr(o,"orange")∧shp(o,"circle") — T
o_2 = ιo.shp(o,"rectangle") — T
o_3 = ιo.id(o,"H") — T

MoVer Execution Engine

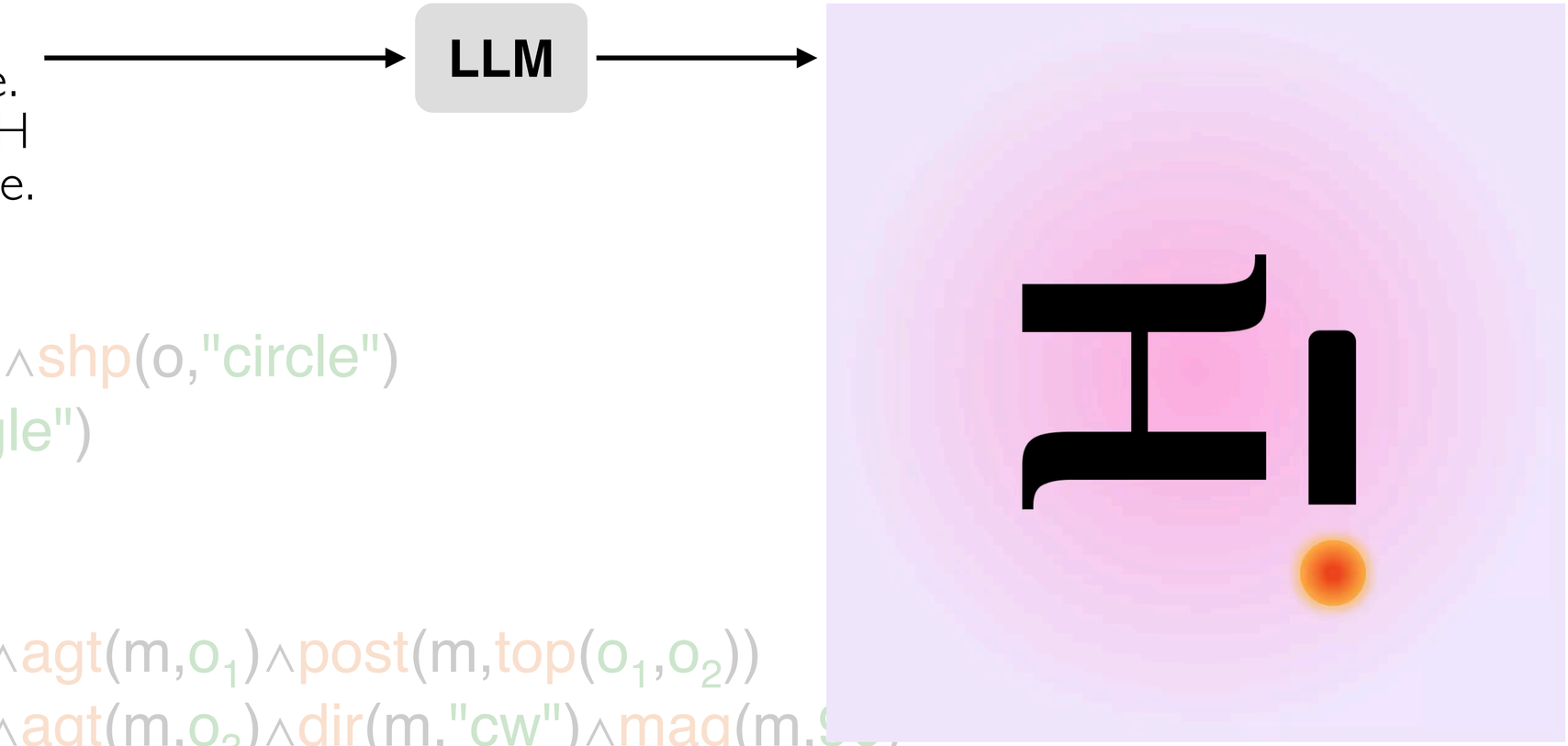
m_1 = ιm.type(m,"trn")∧agt(m,o_1)∧post(m,top(o_1,o_2)) — F
m_2 = ιm.type(m,"rot")∧agt(m,o_3)∧dir(m,"cw")∧mag(m,90) — T

while(m_1,m_2) — F

MoVer

[Ma and Agrawala 2025]

The orange circle translates above the rectangular shape. In the meantime, the letter H rotates 90 degrees clockwise.



o_1 = ιo.clr(o,"orange")∧shp(o,"circle")
o_2 = ιo.shp(o,"rectangle")
o_3 = ιo.id(o,"H")

m_1 = ιm.type(m,"trn")∧agt(m,o_1)∧post(m,top(o_1,o_2))
m_2 = ιm.type(m,"rot")∧agt(m,o_3)∧dir(m,"cw")∧mag(m,

while(m_1,m_2)

Designing shared representation

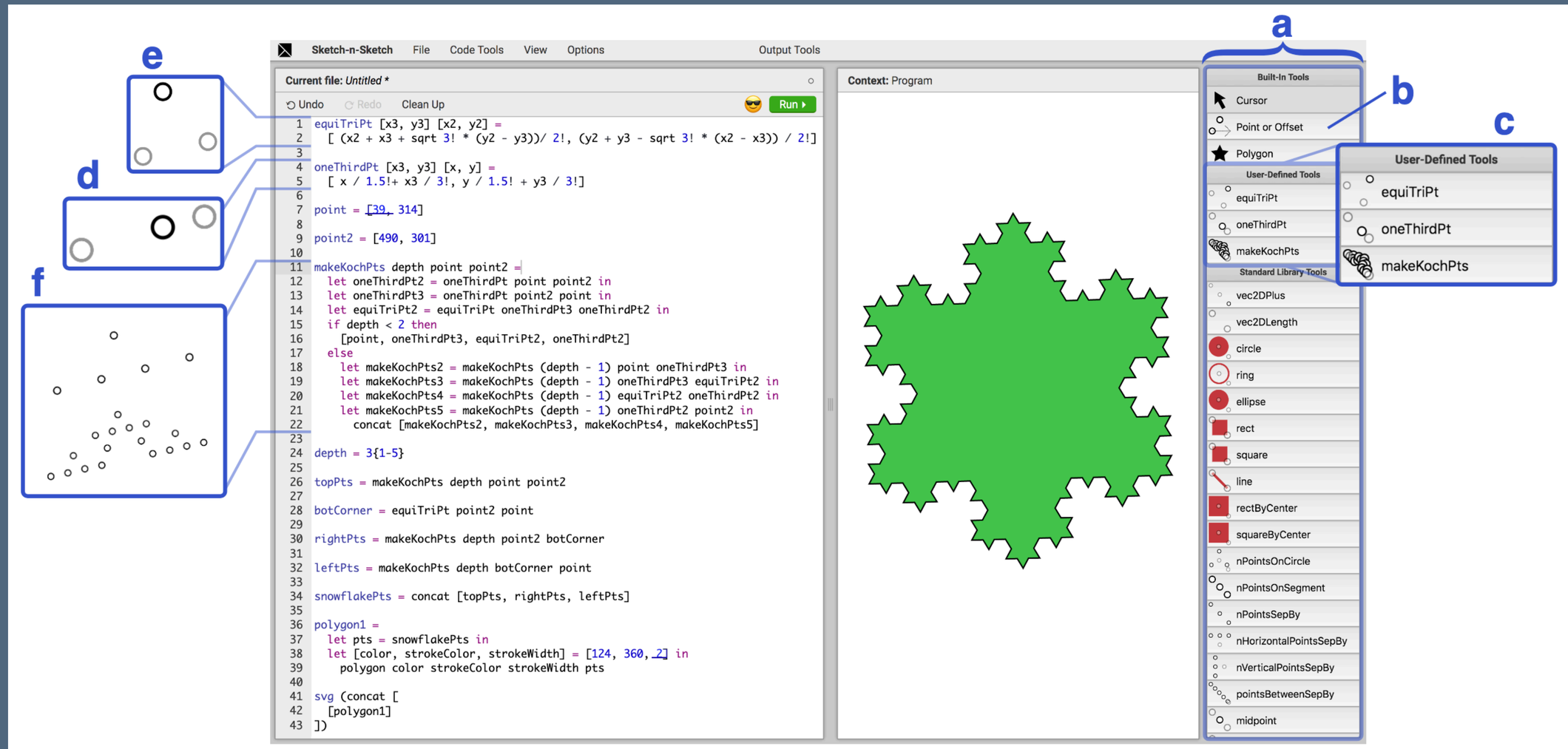
1. The shared representation can directly be *in the same format* as the user input and the machine output

2. The shared representation can be an intermediate layer between the user input and the machine output when 1. is not feasible

How can we help humans to *more efficiently interpret* these intermediate layers?

Sketch-n-Sketch

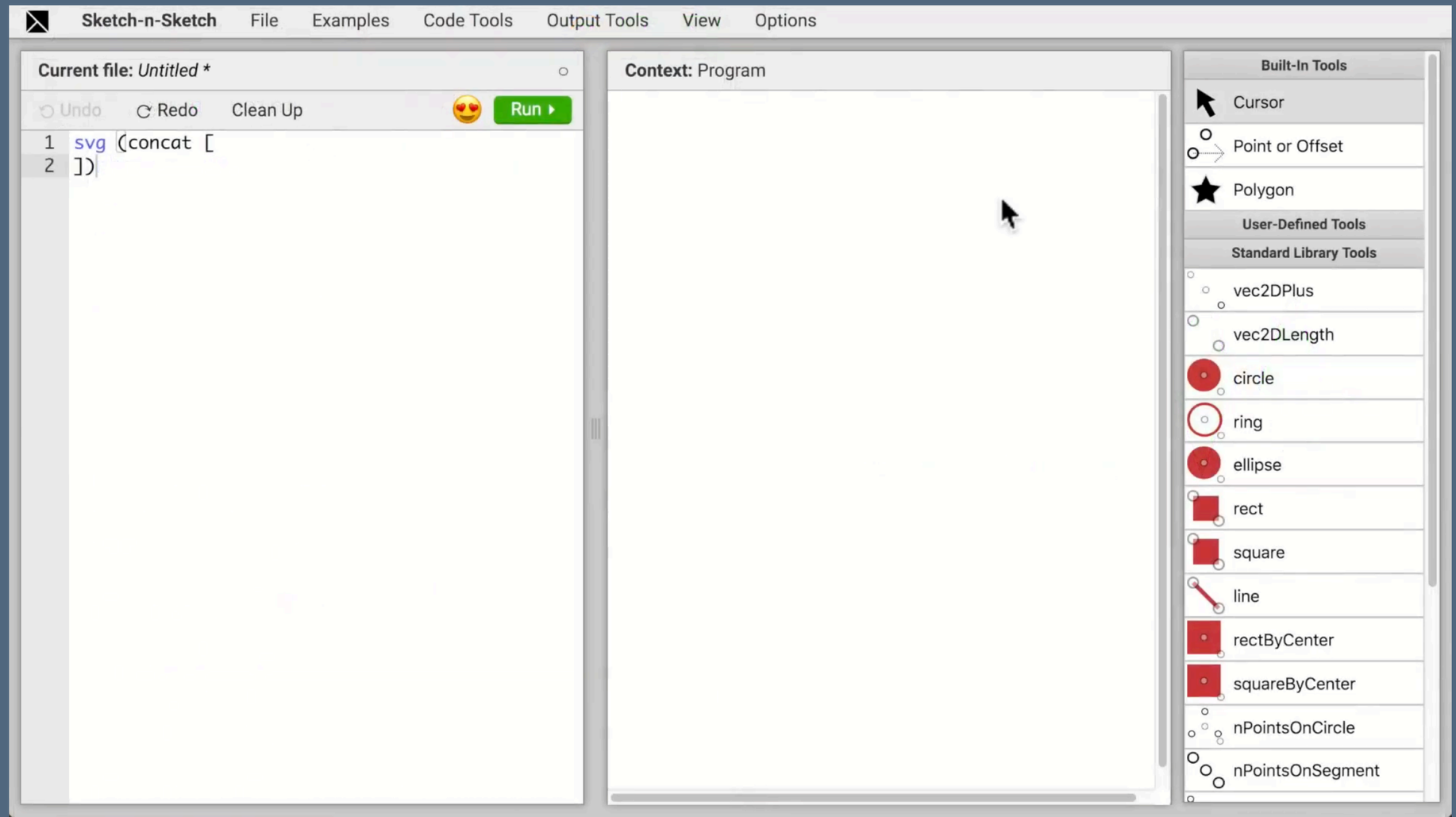
[Hempel 2019]




```
equiTriPt [x3, y3] [x2, y2] =
  [ (x2 + x3 + sqrt 3! * (y2 - y3))/ 2!, (y2 + y3 - sqrt 3! * (x2 - x3)) / 2!]

oneThirdPt [x3, y3] [x, y] =
  [ x / 1.5!+ x3 / 3!, y / 1.5! + y3 / 3!]

point = [39, 314]

point2 = [490, 301]

makeKochPts depth point point2 =
  let oneThirdPt2 = oneThirdPt point point2 in
  let oneThirdPt3 = oneThirdPt point2 point in
  let equiTriPt2 = equiTriPt oneThirdPt3 oneThirdPt2 in
  if depth < 2 then
    [point, oneThirdPt3, equiTriPt2, oneThirdPt2]
  else
    let makeKochPts2 = makeKochPts (depth - 1) point oneThirdPt3 in
    let makeKochPts3 = makeKochPts (depth - 1) oneThirdPt3 equiTriPt2 in
    let makeKochPts4 = makeKochPts (depth - 1) equiTriPt2 oneThirdPt2 in
    let makeKochPts5 = makeKochPts (depth - 1) oneThirdPt2 point2 in
      concat [makeKochPts2, makeKochPts3, makeKochPts4, makeKochPts5]

depth = 3{1-5}

topPts = makeKochPts depth point point2

botCorner = equiTriPt point2 point

rightPts = makeKochPts depth point2 botCorner

leftPts = makeKochPts depth botCorner point

snowflakePts = concat [topPts, rightPts, leftPts]

polygon1 =
  let pts = snowflakePts in
  let [color, strokeColor, strokeWidth] = [124, 360, 2] in
    polygon color strokeColor strokeWidth pts

svg (concat [
  [polygon1]
])
```

Sketch-n-Sketch

[Hempel 2019]



Designing shared representation

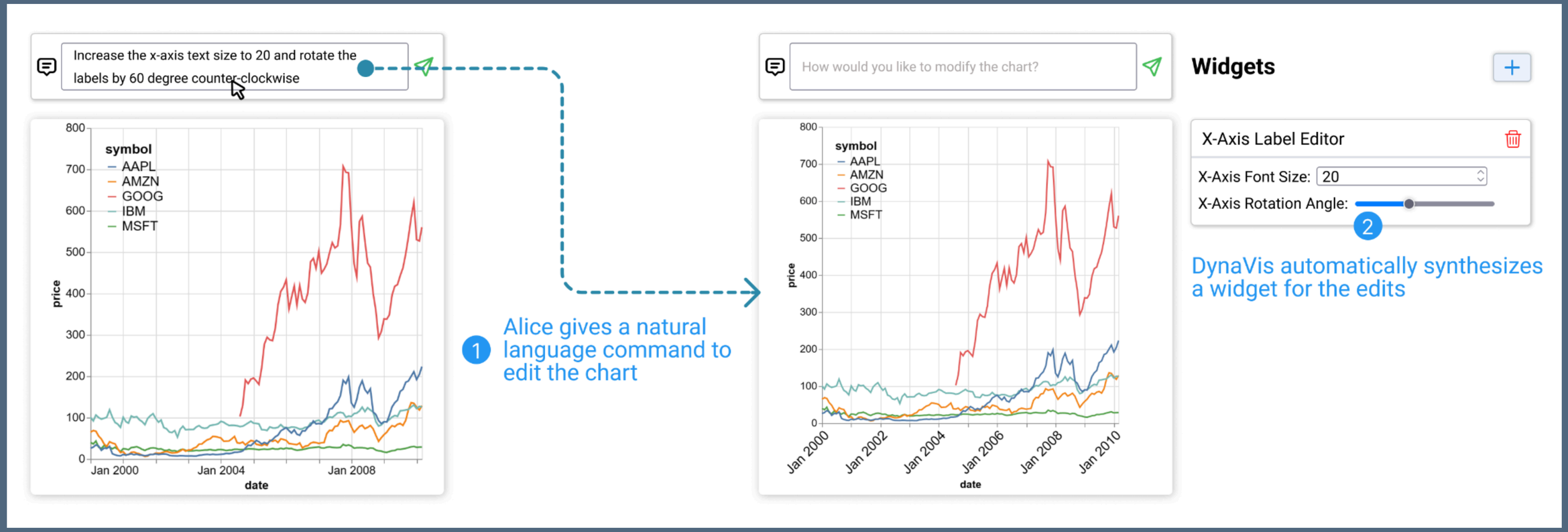
1. The shared representation can directly be *in the same format* as the user input and the machine output.

2. The shared representation can be an intermediate layer between the user input and the machine output when 1. is not feasible.

3. The shared representation can be designed to improve readability for the user while maintaining interpretability for the machine.

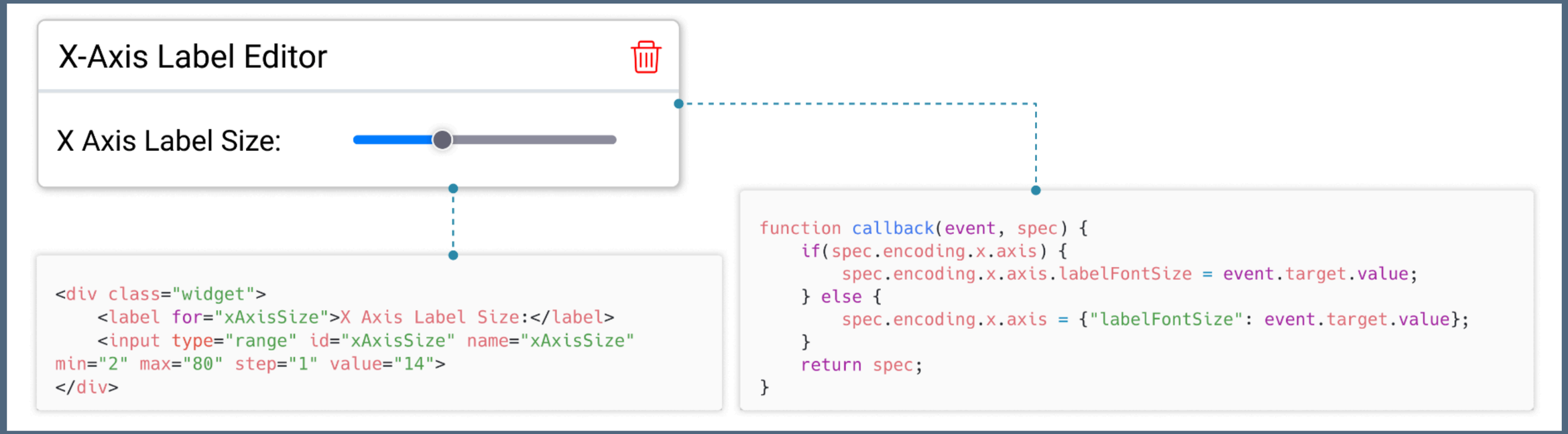
DynaVis

[Vaithilingam 2024]



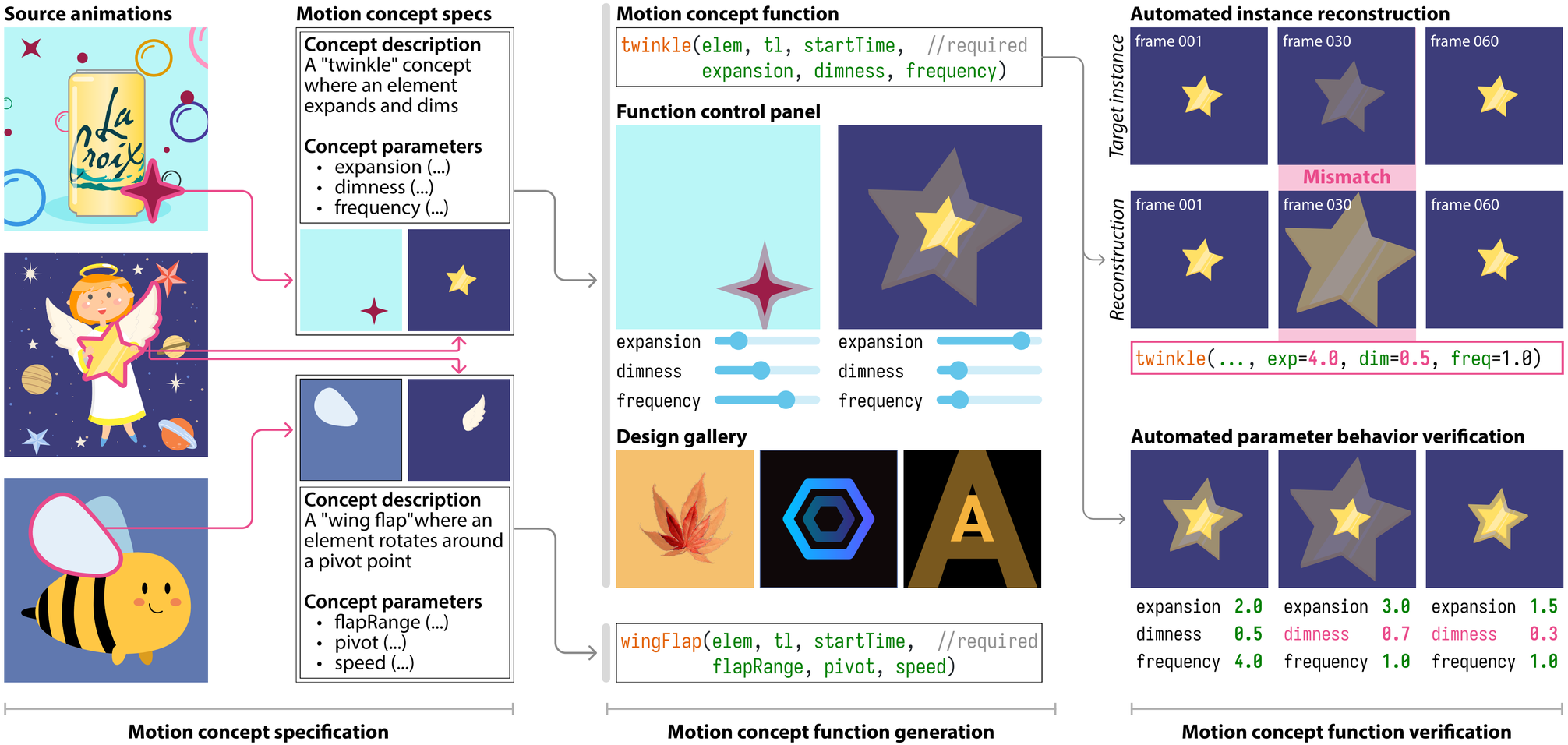
DynaVis

[Vaithilingam 2024]



VibeAnimation

[Ma 2026]



VibeAnimation

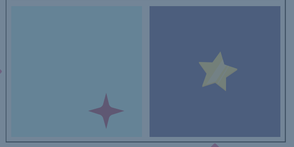
[Ma 2026]

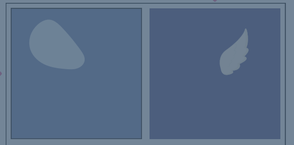
Source animations

Motion concept specs

Concept description
A "twinkle" concept where an element expands and dims

Concept parameters
- expansion (...)
- dimness (...)
- frequency (...)





Concept description
A "wing flap"where an element rotates around a pivot point

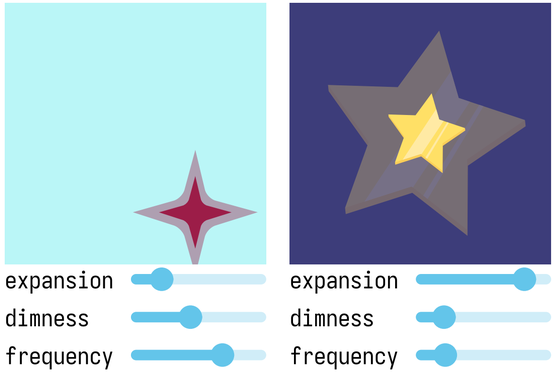
Concept parameters
- flapRange (...)
- pivot (...)
- speed (...)

Motion concept specification

Motion concept function

```
twinkle(elem, tl, startTime,  //required
        expansion, dimness, frequency)
```

Function control panel

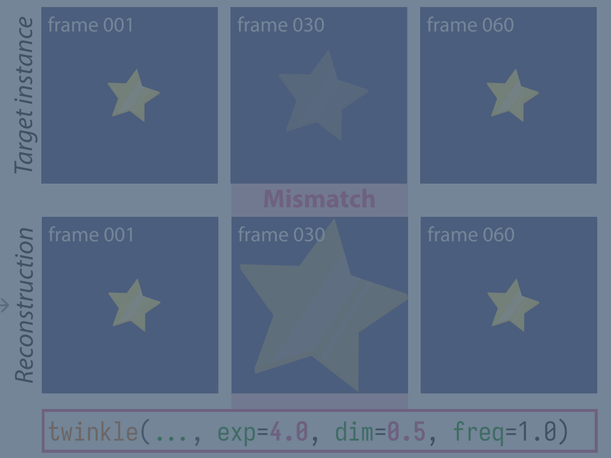


Design gallery

```
wingFlap(elem, tl, startTime,  //required
         flapRange, pivot, speed)
```

Motion concept function generation

Automated instance reconstruction




```
twinkle(..., exp=4.0, dim=0.5, freq=1.0)
```

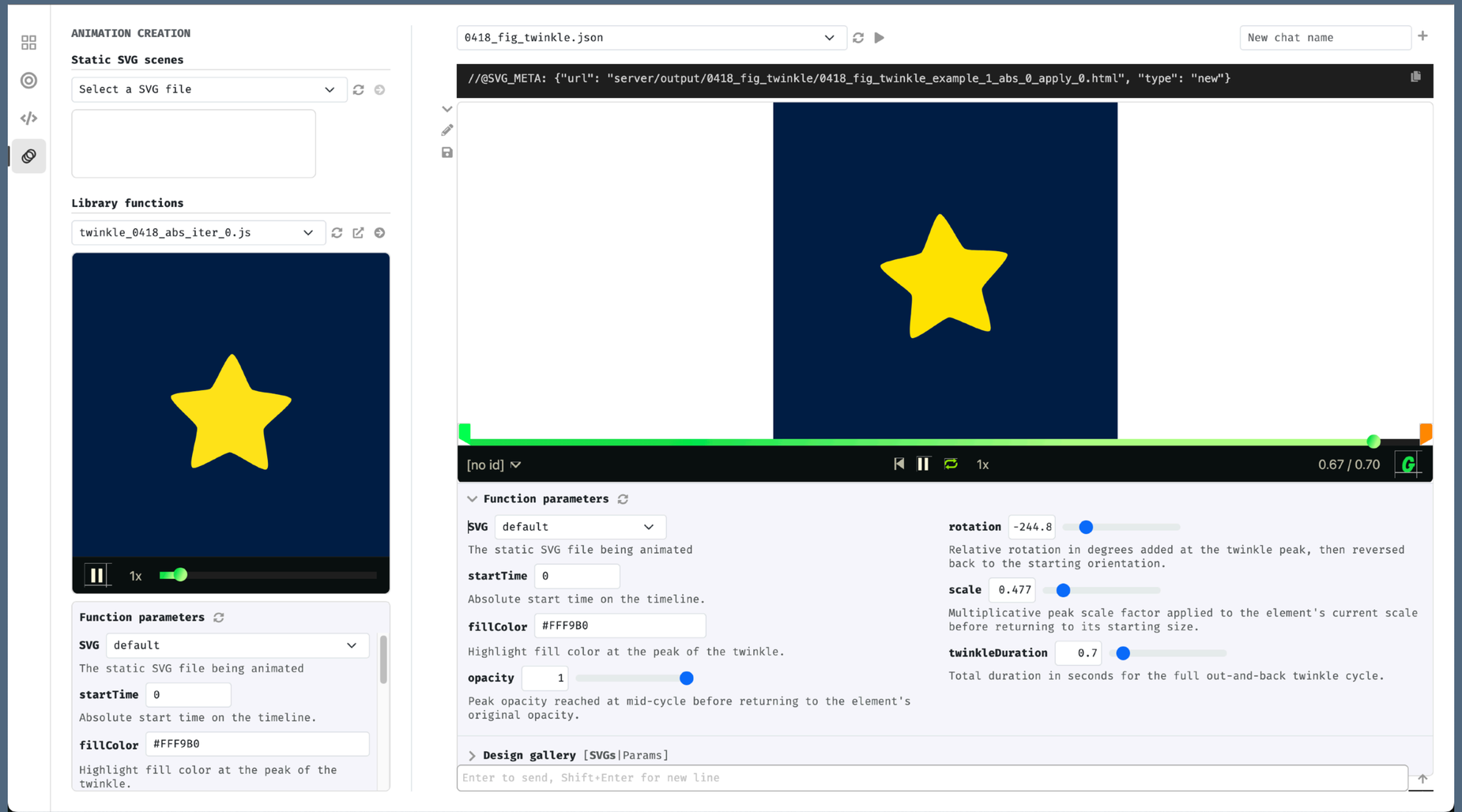
Automated parameter behavior verification

expansion 2.0	expansion 3.0	expansion 1.5
dimness 0.5	dimness 0.7	dimness 0.3
frequency 4.0	frequency 1.0	frequency 1.0

Motion concept function verification

VibeAnimation

[Ma 2026]



Summary

1. The shared representation can directly be *in the same format* as the user input and the machine output.

2. The shared representation can be an intermediate layer between the user input and the machine output when 1. is not feasible.

    - Programs are often being used as the intermediate layer

3. The shared representation can be designed to improve readability for the user while maintaining interpretability for the machine.

References

Heer, Jeffrey. "Agency plus automation: Designing artificial intelligence into interactive systems." Proceedings of the National Academy ofSciences 116.6 (2019): 1844-1850.

Horvitz, Eric. "Principles of mixed-initiative user interfaces." Proceedings of the SIGCHI conference on Human Factors in Computing Systems.

Maes, Pattie. "Agents that reduce work and information overload." Readings in human–computer interaction. Morgan Kaufmann, 1995. 811-821.

Shneiderman, Ben, and Pattie Maes. "Direct manipulation vs. interface agents." interactions 4.6 (1997): 42-61.

Shneiderman, Ben. Human-centered AI. Oxford University Press, 2022.

Lee, Yong Jae, C. Lawrence Zitnick, and Michael F. Cohen. "Shadowdraw: real-time user guidance for freehand drawing." ACM Transactions on Graphics (ToG) 30.4 (2011): 1-10.

Hertzmann, Aaron, et al. "Image analogies." Seminal Graphics Papers: Pushing the Boundaries, Volume 2. 2023. 557-570.

Cho, Jaemin, Abhay Zala, and Mohit Bansal. "Visual programming for step-by-step text-to-image generation and evaluation." Advances in Neural Information Processing Systems 36 (2023): 6048-6069.

Surís, Dídac, Sachit Menon, and Carl Vondrick. "Vipergpt: Visual inference via python execution for reasoning." Proceedings of the IEEE/CVF international conference on computer vision. 2023.

References

Liu, Vivian, et al. "Logomotion: Visually-grounded code synthesis for creating and editing animation." Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems. 2025.

Artzi, Yoav, and Luke Zettlemoyer. "Weakly supervised learning of semantic parsers for mapping instructions to actions." Transactions of the association for computational linguistics 1 (2013): 49-62.

Ma, Jiaju, and Maneesh Agrawala. "Mover: Motion verification for motion graphics animations." ACM Transactions on Graphics (TOG) 44.4 (2025): 1-17.

Hempel, Brian, Justin Lubin, and Ravi Chugh. "Sketch-n-sketch: Output-directed programming for svg." Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology. 2019.

Vaithilingam, Priyan, et al. "Dynavis: Dynamically synthesized ui widgets for visualization editing." Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems. 2024.

https://medium.com/free-code-camp/things-you-need-to-know-about-working-with-svg-in-vs-code-63be593444dd

https://codepen.io/chrisgannon/pen/GJpyrb