Malwa: A Tool for Fully Automated Model Inference of Instrumented Web Applications

enhancing

ChatGPT in the Loop

Alexander Bainczyk, Marco Krumrey, Daniel Busch, Frederik Gossen, Alnis Murtovi, Gerrit Nolte, Sami Mitwalli, Maximilian Schlüter, **Bernhard Steffen**

06.04.2024

tu dortmund university

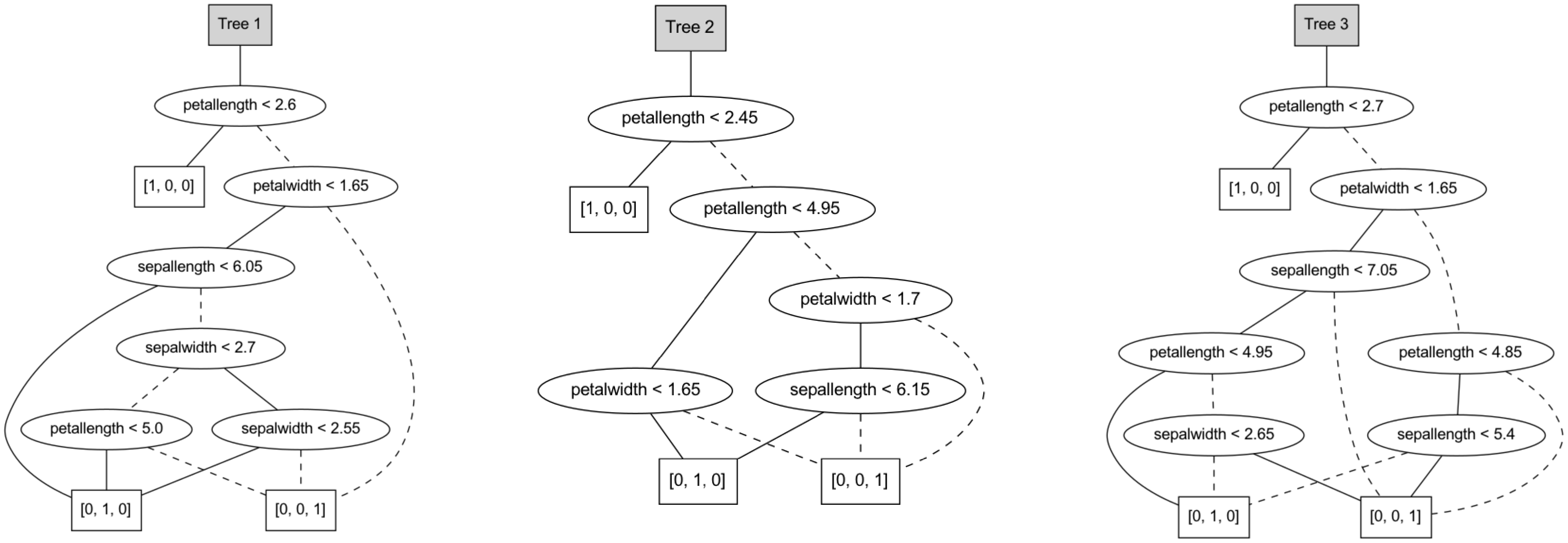
Overview

- **Brief Personal History**
  - Verification and Explanation: Concepts and Scalability
    - Random Forests
    - Deep Neural Networks
- **AI-Assisted Programming**
  - LLMs as part of Language-Driven (Softwareware) Engineering
- **Malwa: A Tool for Fully Automated Model Inference**
- **Conclusions and Perspectives**

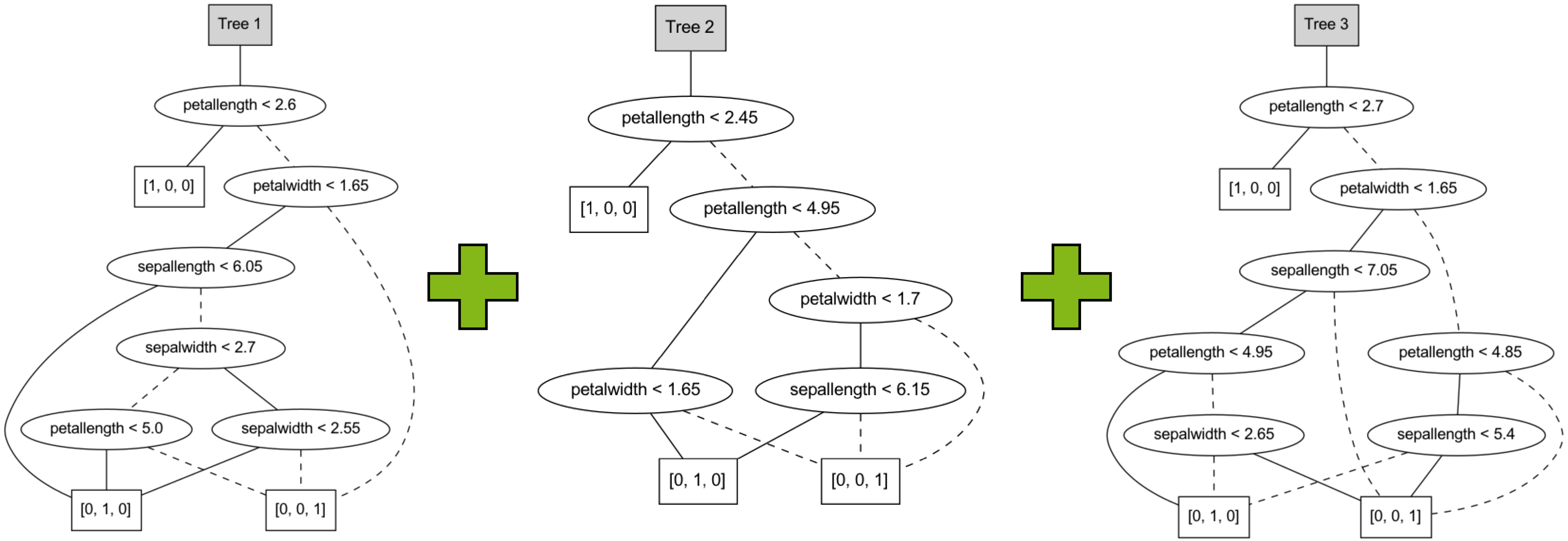
Random Forests

- **Aggregation**
  - The Power of Algebrais Decidion Diagrams
- **Explanation**
  - Abstraction and such
- **Verification**
  - Pre/Post-Verification as Infeasible Paths Reduction

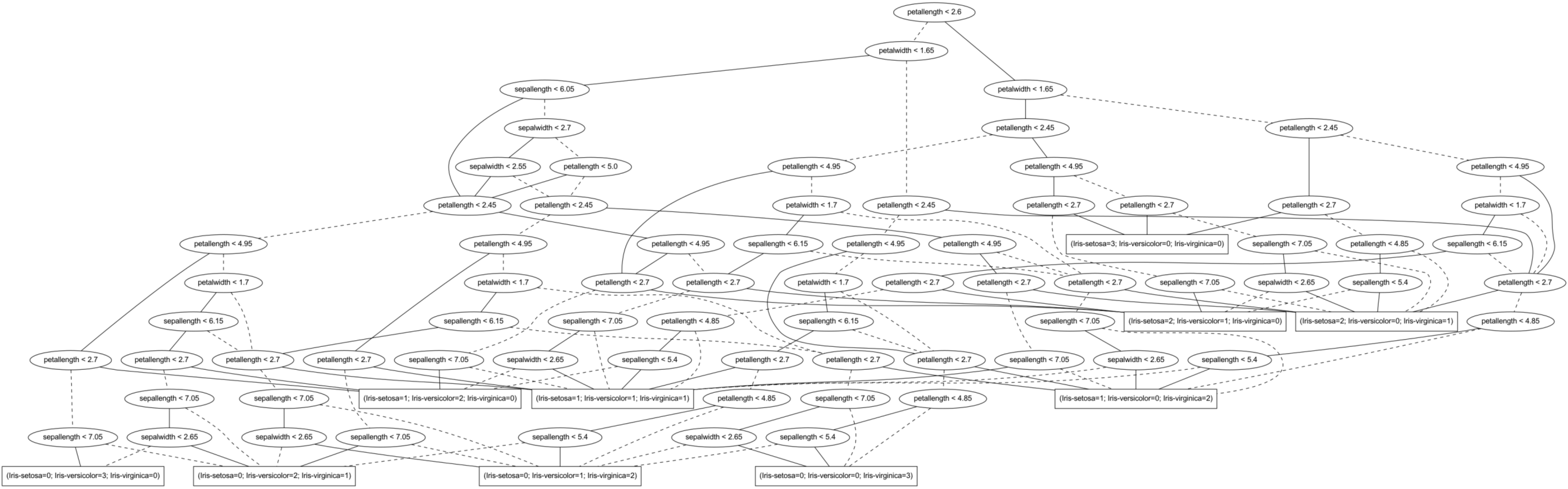
Decision Trees → Algebraic Decision Diagrams



Algebraic Aggregation

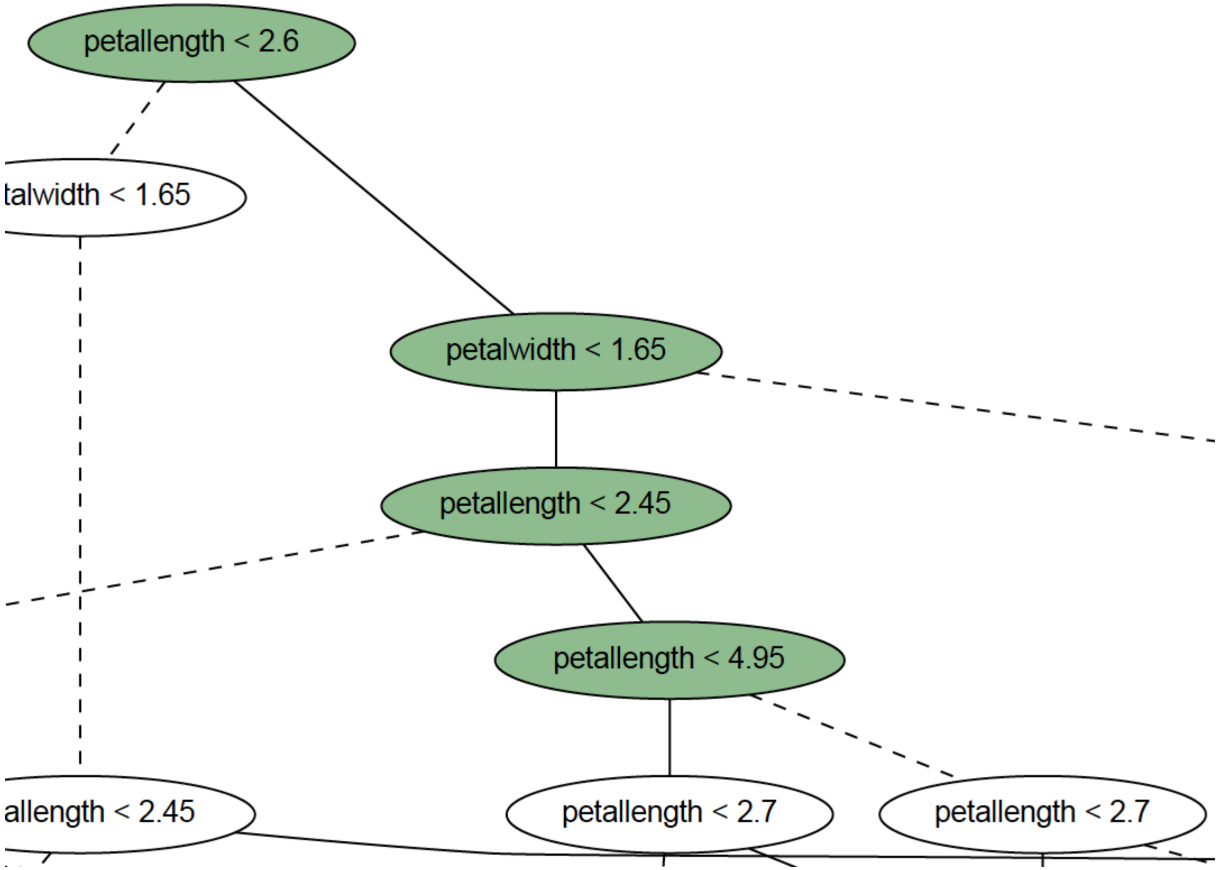


The Aggregate

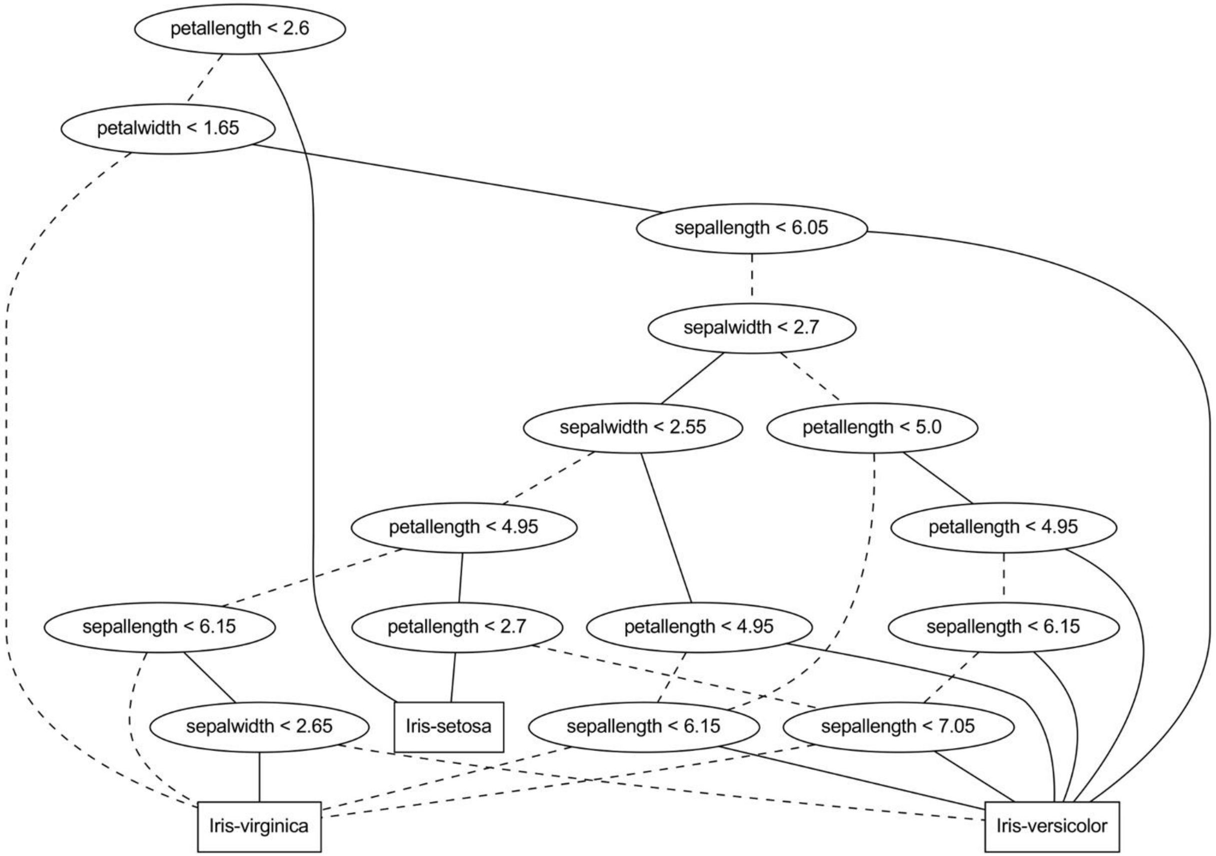


Infeasible Path Elimination

- Predicates are not independent of each other
- E.g.: petallength < 2.6 ⇒ petallength < 4.95
- Significant reduction of size and depth

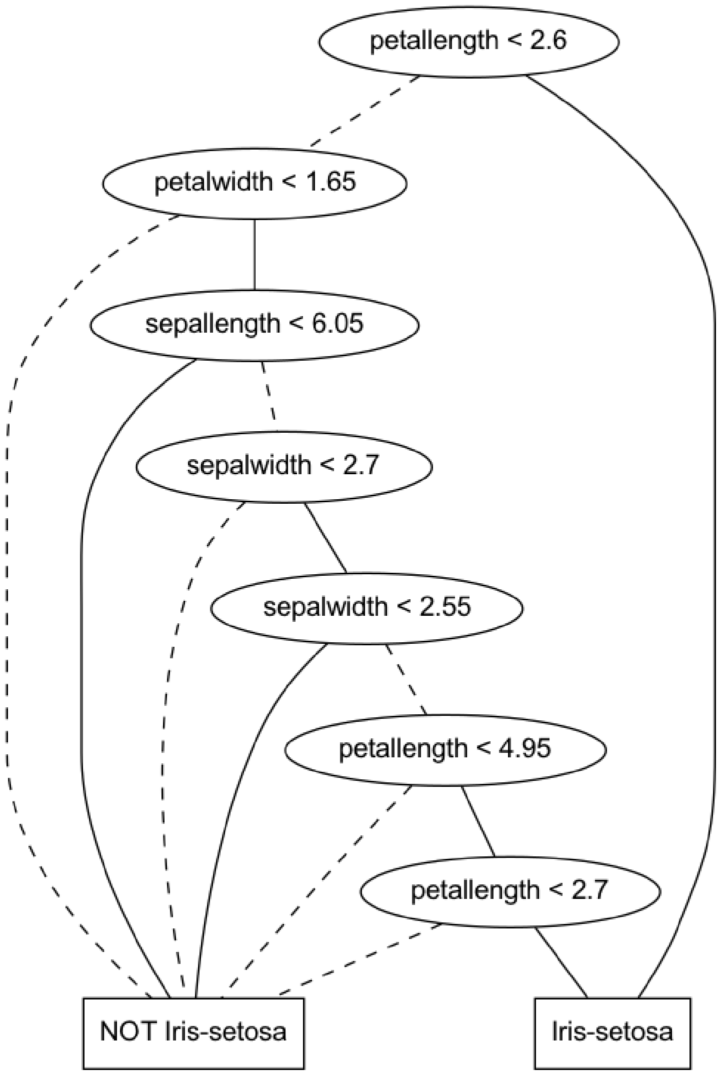


Model Explanation (18 nodes)



Class Characterization

- Given: Class **c**
- Restrict model explanation to part that is relevant for class **c**
- **BDD**: Class **c** vs. all other classes



Outcome Explanation

petallength ≥ 2.6

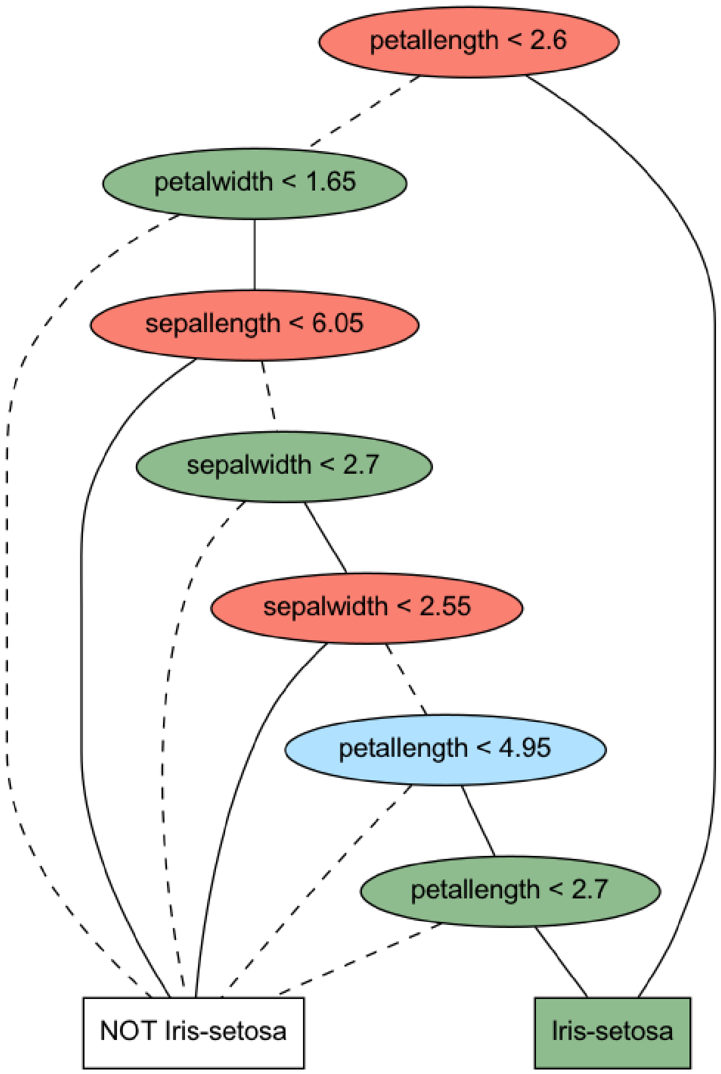
⋀ petalwidth < 1.65

⋀ sepallength ≥ 6.05

⋀ sepalwidth < 2.7

⋀ sepalwidth ≥ 2.55

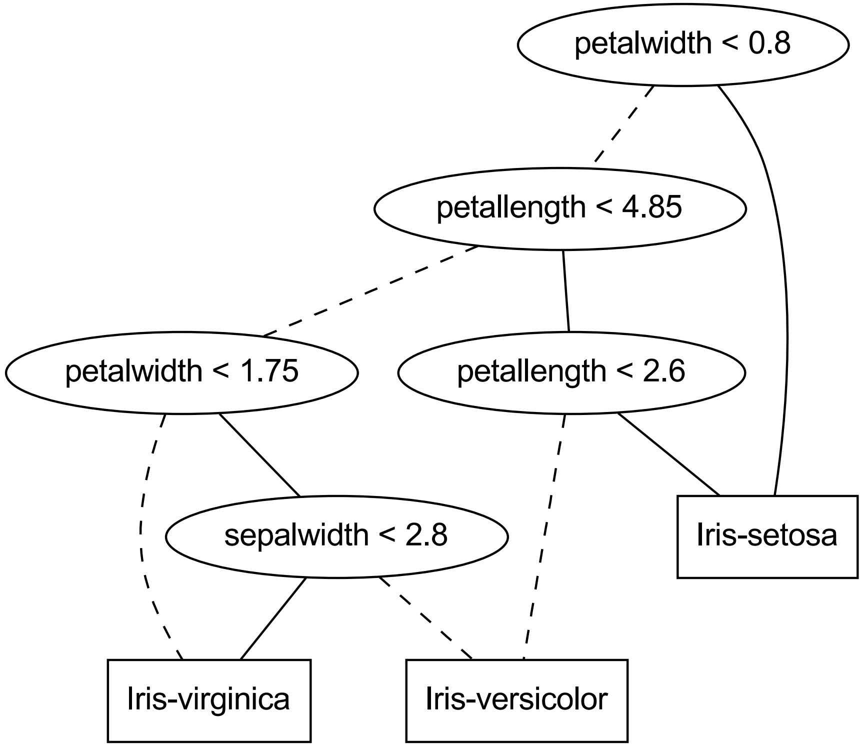
⋀ petallength < 2.7



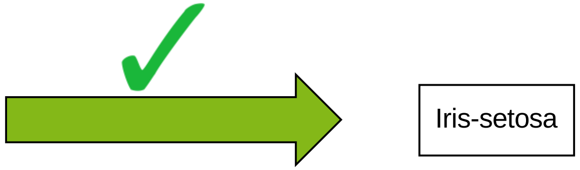
Pre/Post Forest Verification **for Free!!!**

- Input: x
- Random Forest: f
- Precondition: ϕ
- Postcondition: ψ
- Verify: $\forall x.\, \phi(x) \Rightarrow \psi(f(x))$
- Robustness: $\forall x'.\, distance(x, x') < \epsilon \Rightarrow f(x) = f(x')$
- Chebyshev distance: $D_{Chebyshev} = max_i\,(|\, x_i - y_i\,|)$

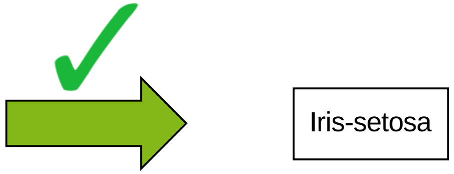
Example:



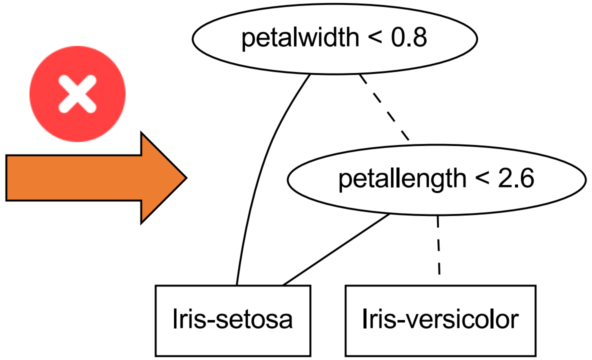
- Input 1:
  - petalwidth = 0.75
  - petallength = 2.45



- Input 2 (ϵ = 0.1):
  - 0.65 ≤ petalwidth ≤ 0.85
  - 2.35 ≤ petallength ≤ 2.55



- Input 3 (ϵ = 0.2):
  - 0.55 ≤ petalwidth ≤ 0.95
  - 2.25 ≤ petallength ≤ 2.65



Forest Gump

https://demo.forest-gump.k8s.ls-5.de/

ADD-Lib

https://gitlab.com/scce/add-lib

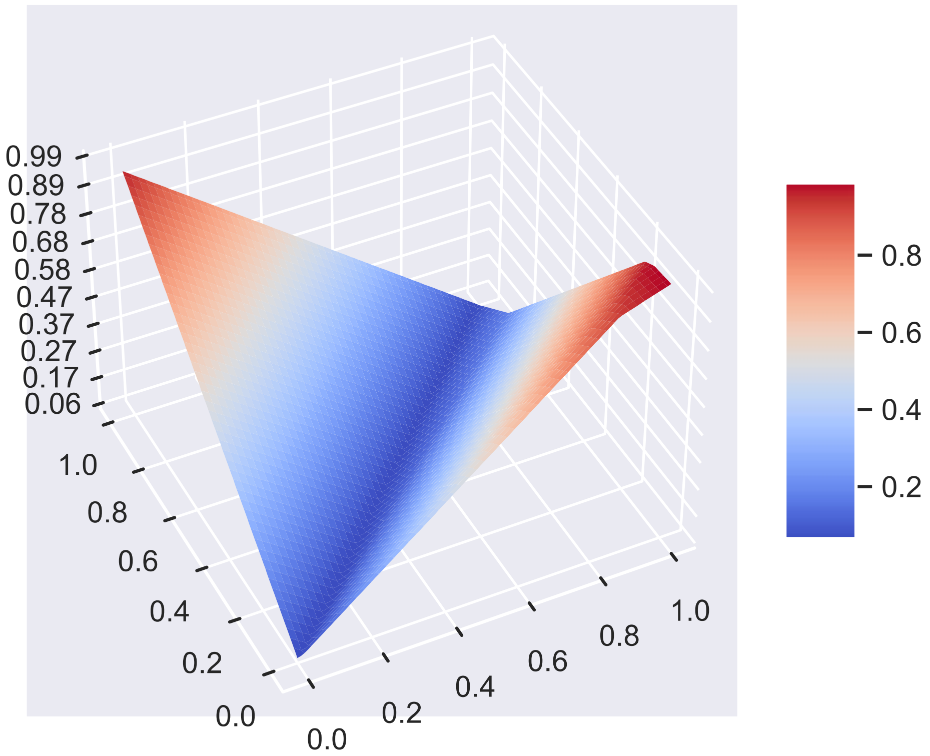
Overview

- **Brief Personal History**
  - Verification and Explanation: Concepts and Scalability
    - Random Forests
    - Deep Neural Networks
- **AI-Assisted Programming**
  - LLMs as part of Language-Driven (Softwareware) Engineering
- **Malwa: A Tool for Fully Automated Model Inference**
- **Conclusions and Perspectives**

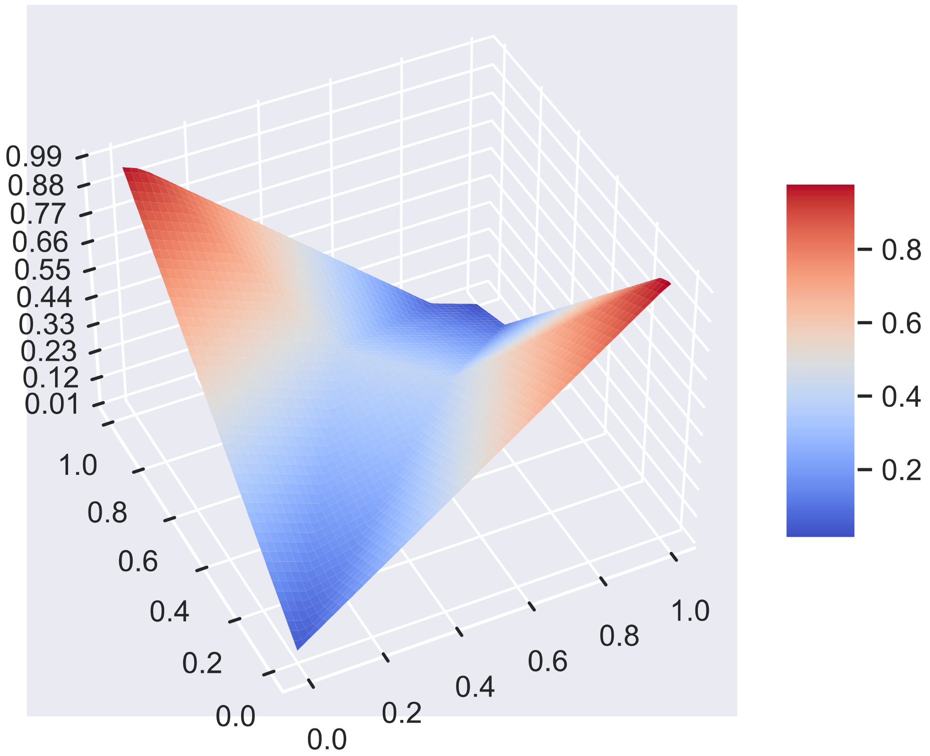
Deep Neural (ReLU) Networks

- **Transfer**
  - **Everything** can be done here as well
  - We have a **Richer** Algebra (DNN Composition)
  - Variable Ordering hurts
  - Essentially we are Dealing with **Trees**
  - There is a solid Scalability Wall
- **DNN Equivalence up to Epsilon**
- **Visual Verification via Concolic Execution**

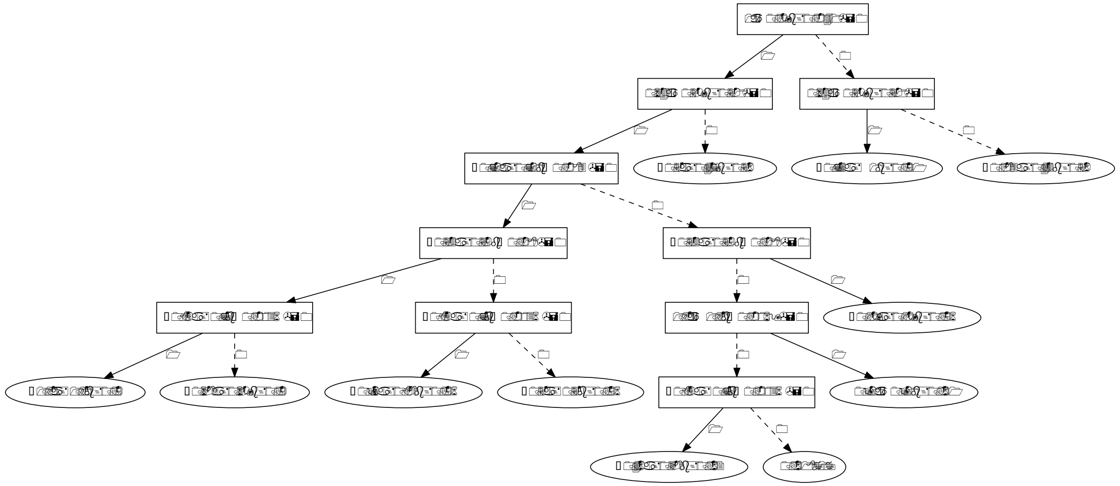
Learning a PLNN – Solution I

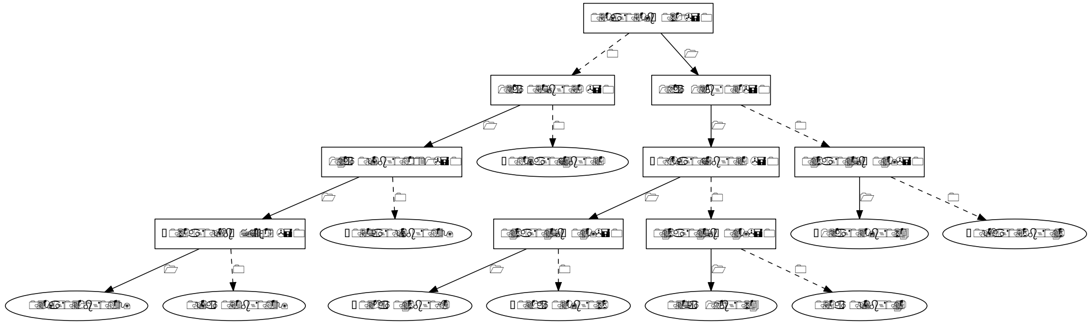


Learning a PLNN – Solution II

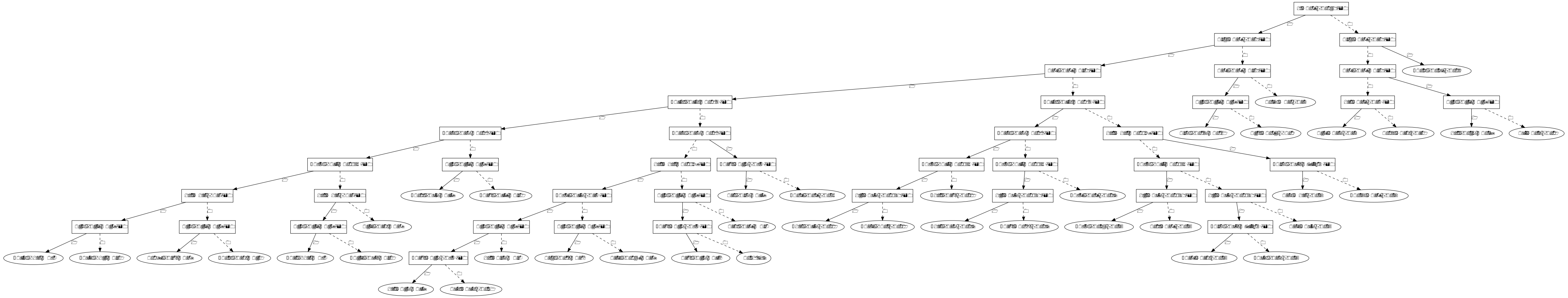


Analyzing PLNN – Algebraic Approach

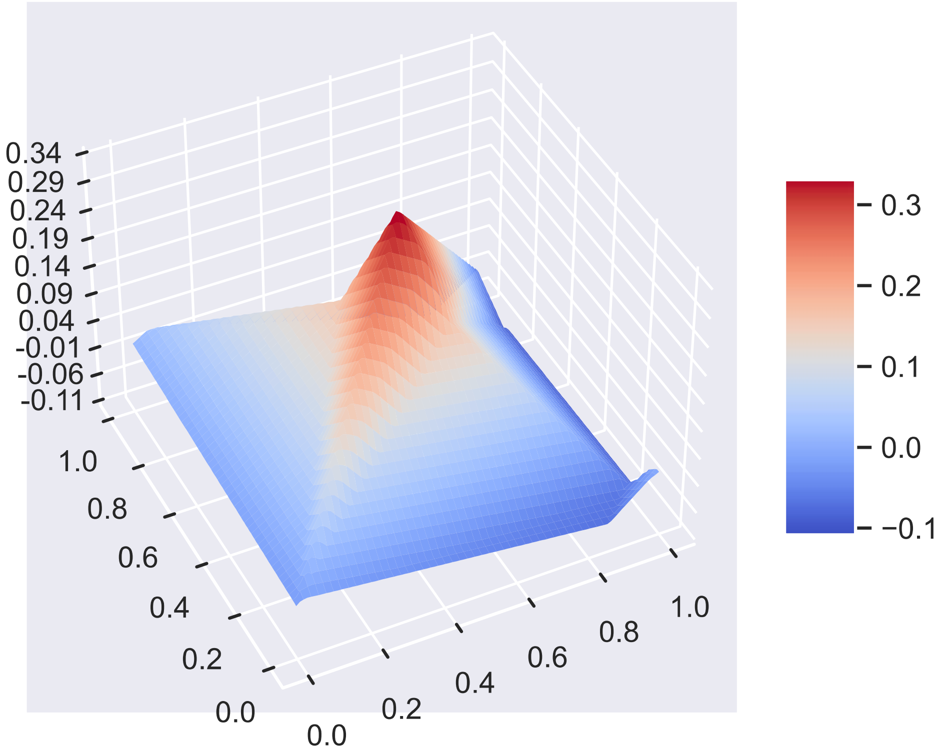




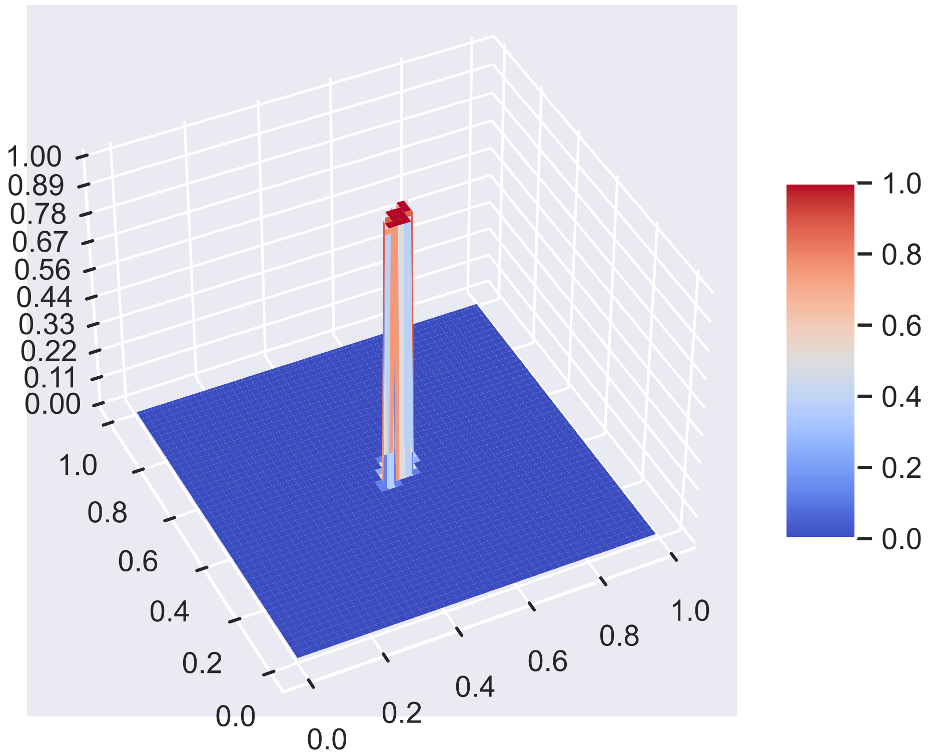
Analyzing PLNN – The Difference



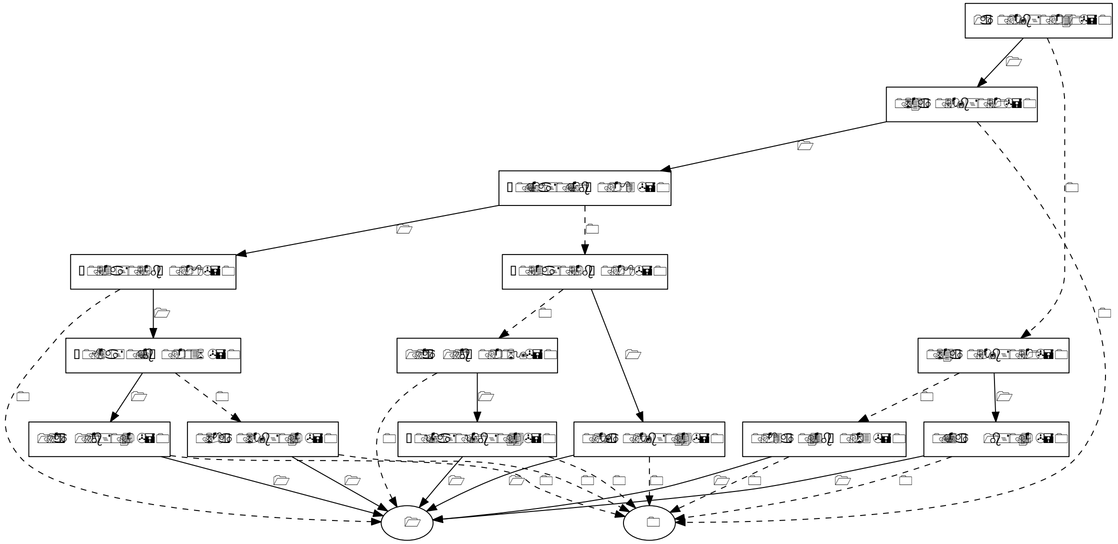
Analyzing PLNN – The Difference

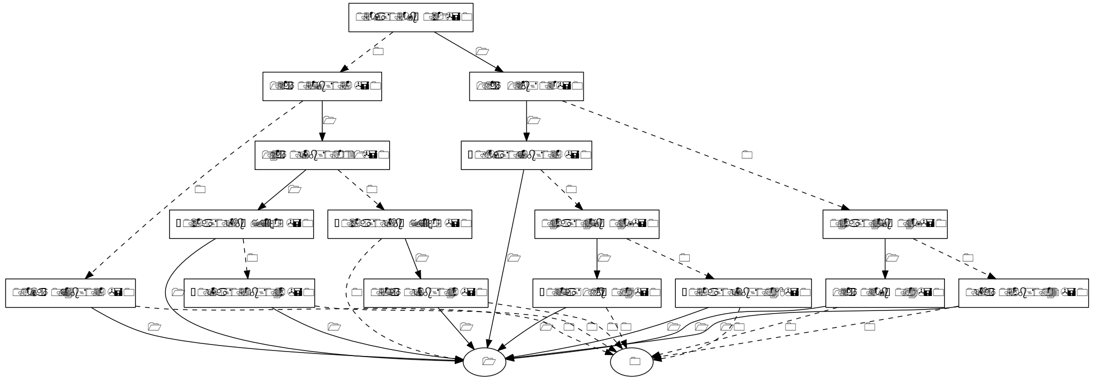


Analyzing PLNN – Equivalence up to

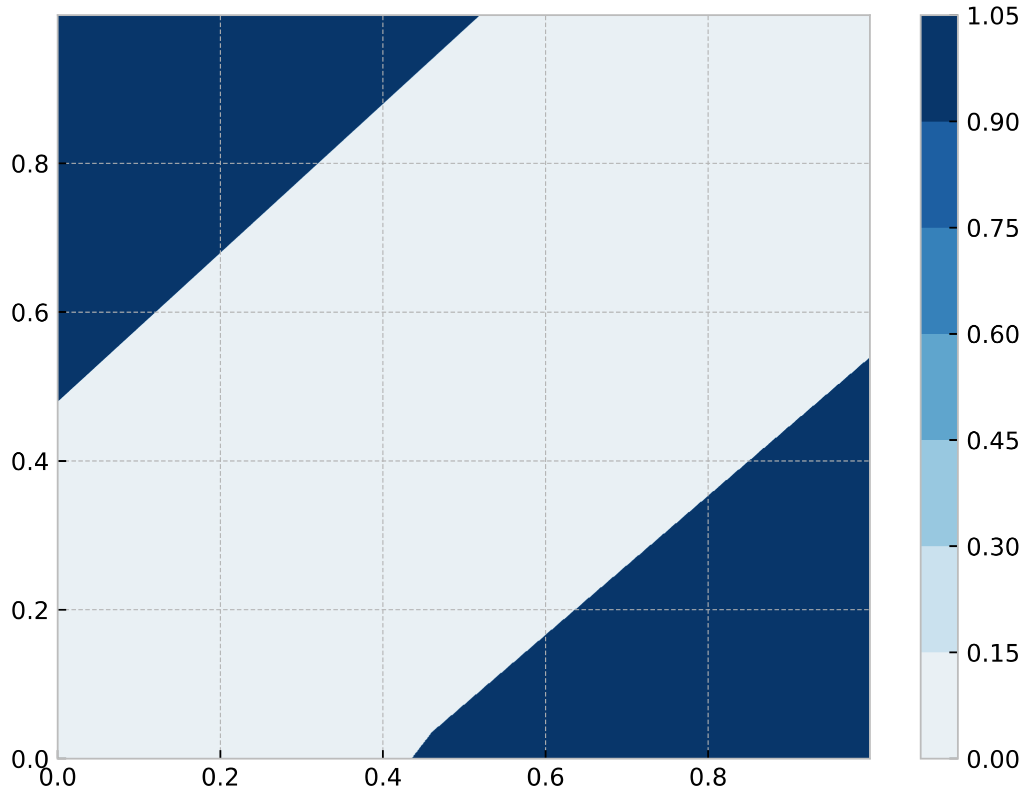


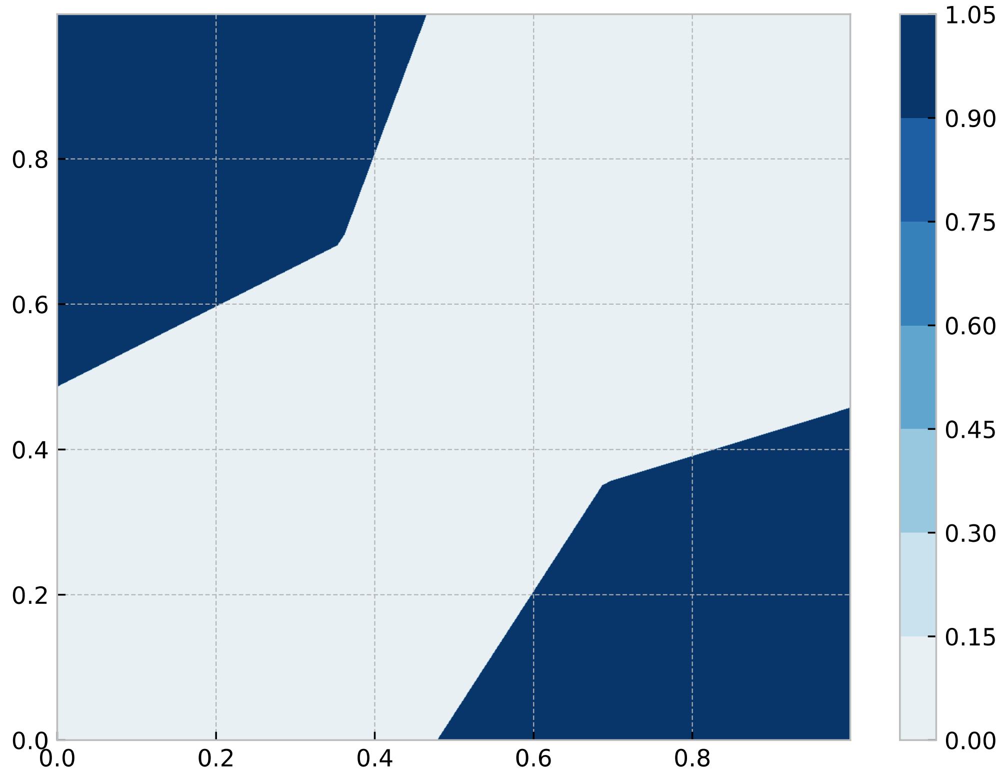
Classification: Requires a threshold



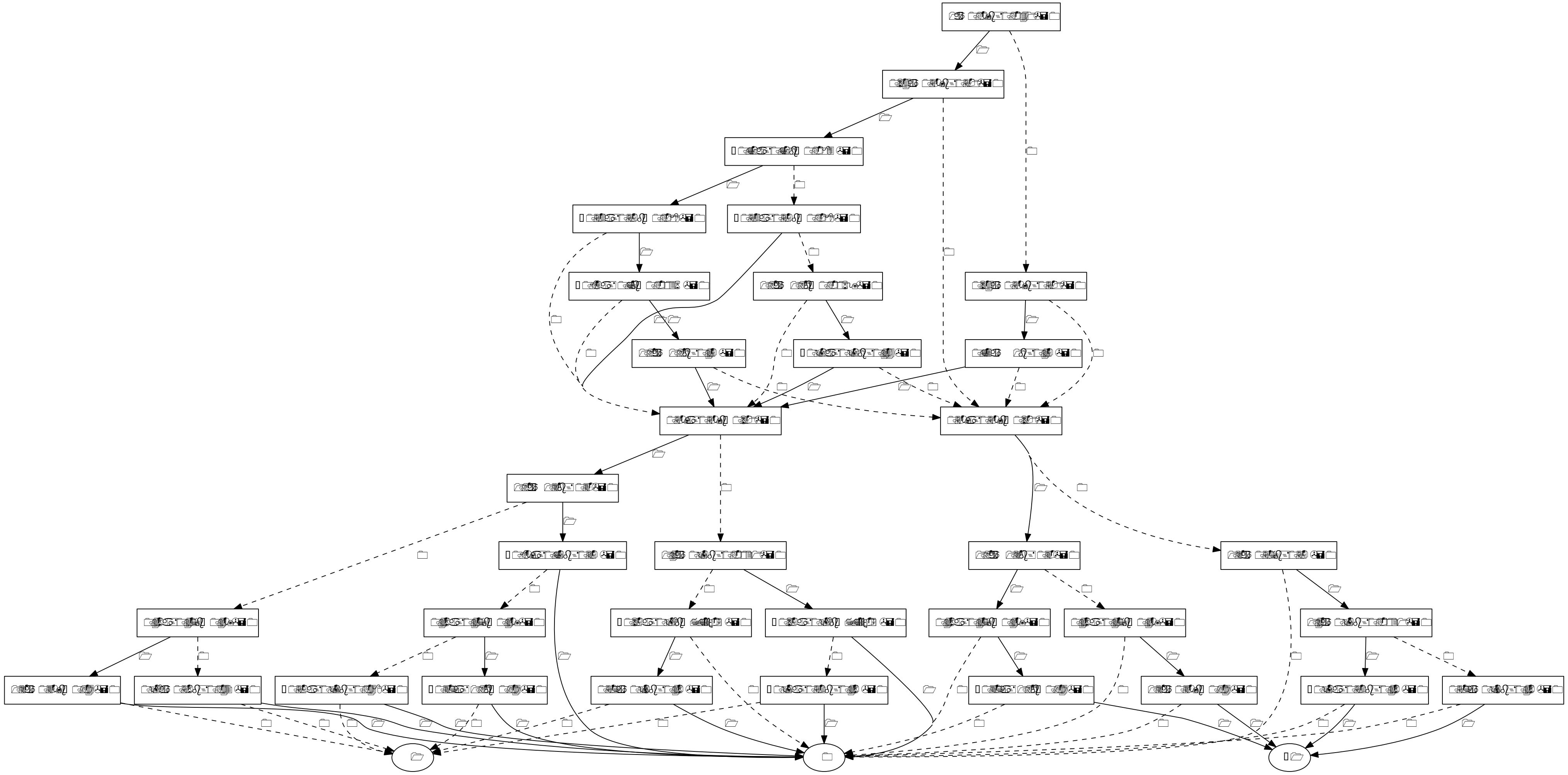


Classification

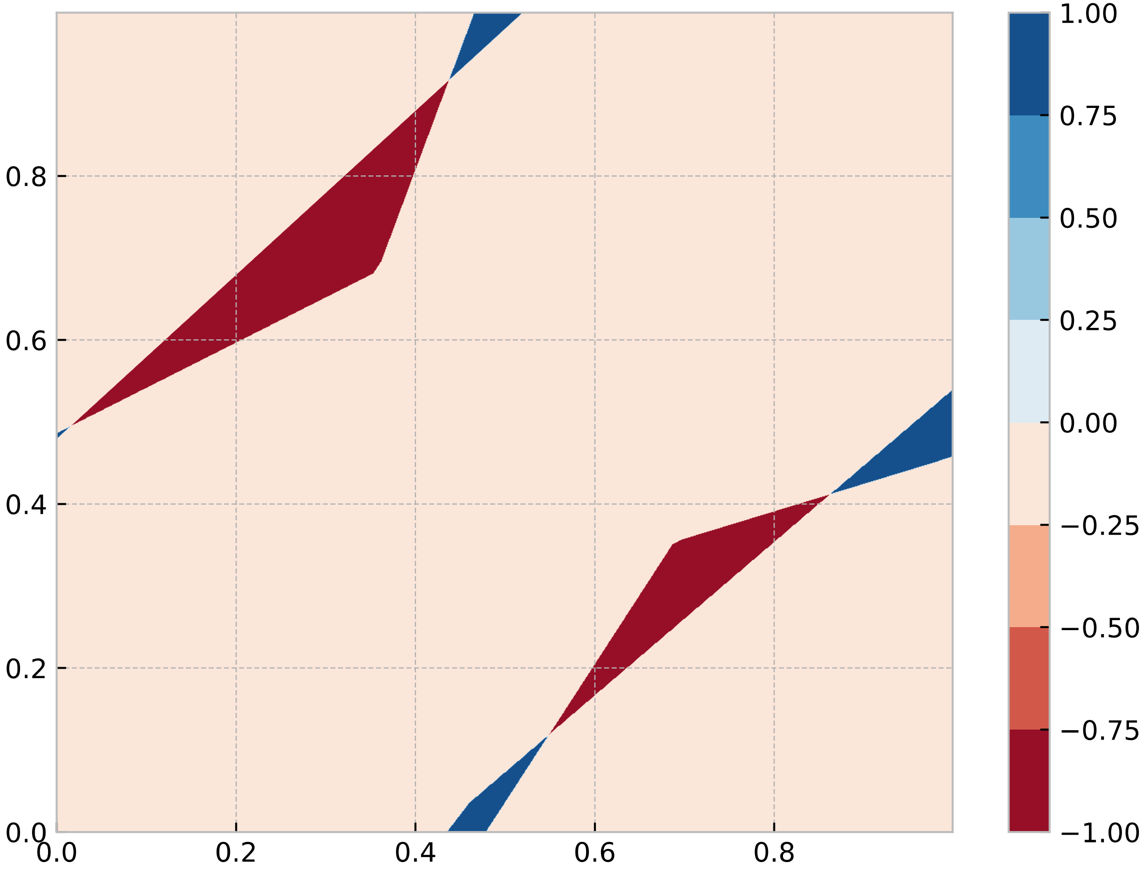




Classification



Classification



Robustness Verification (also Pre/Post)

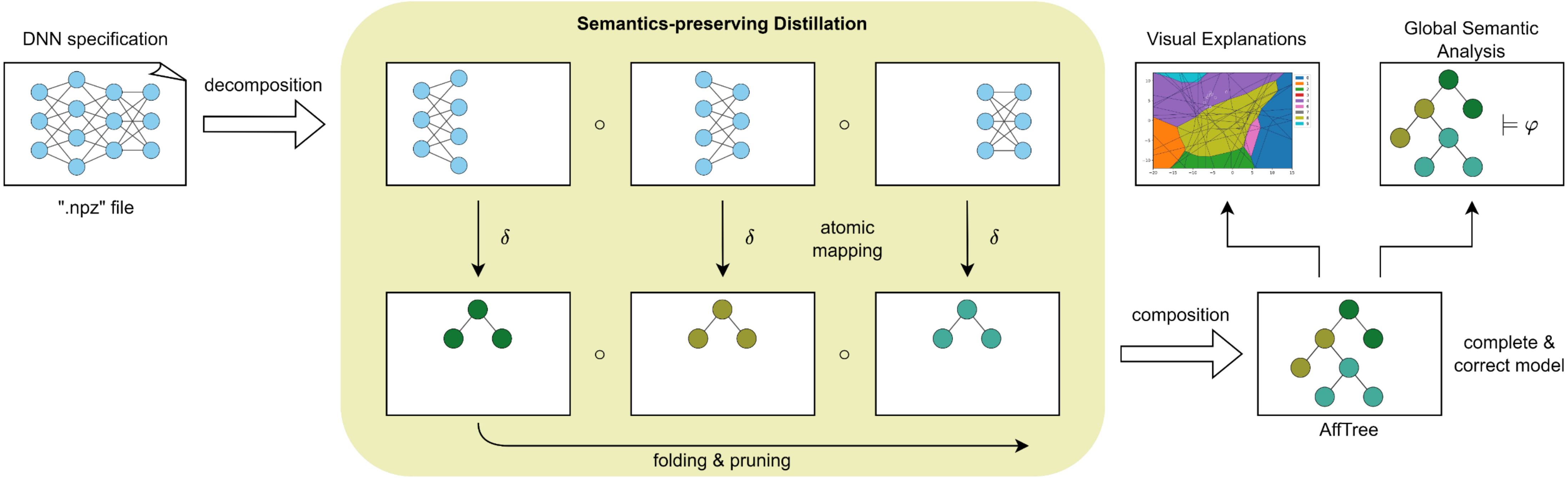
- **Abstract interpretation** of our Typed Affince Decisions Structure (**TADS**)
- **Precise:** Convex Polyhedra   (the result of ReLU)
- **Rough:** Hypercubes
- **Better:** Zonotopes
- **Powerful:** Star Sets

We also hit the **VNN-Wall**

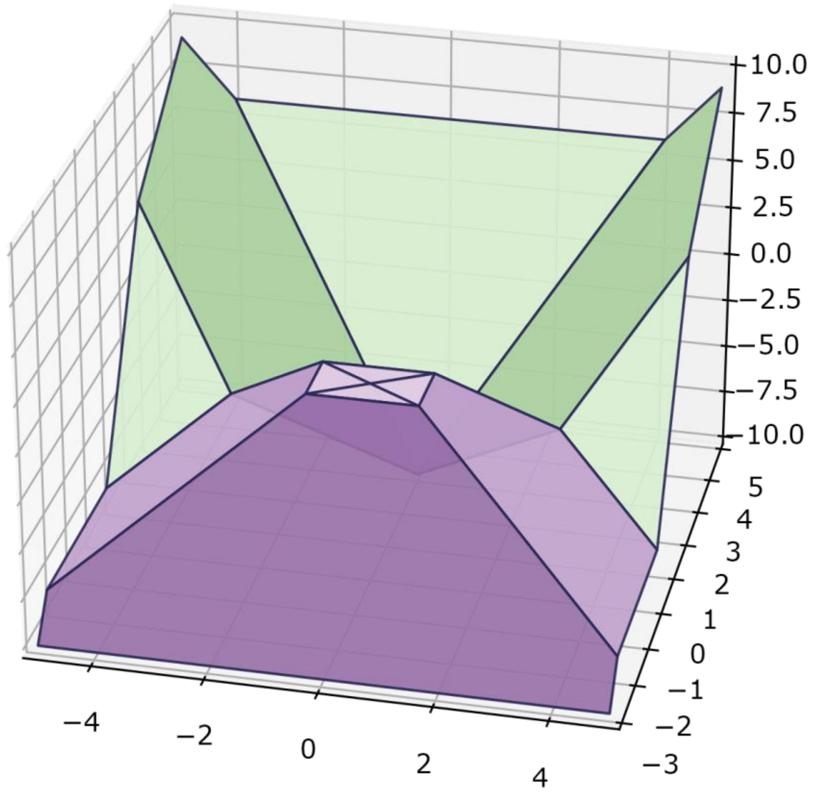
Our Open Sources Library (RUST):

- https://github.com/Conturing/affinitree

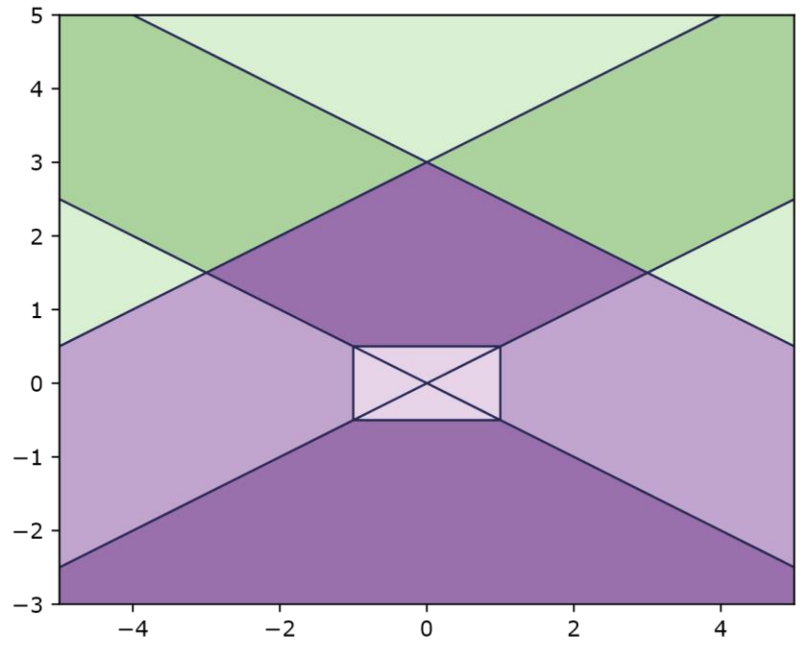
Visual Validation



Piece-wise Linear Functions (PWL)

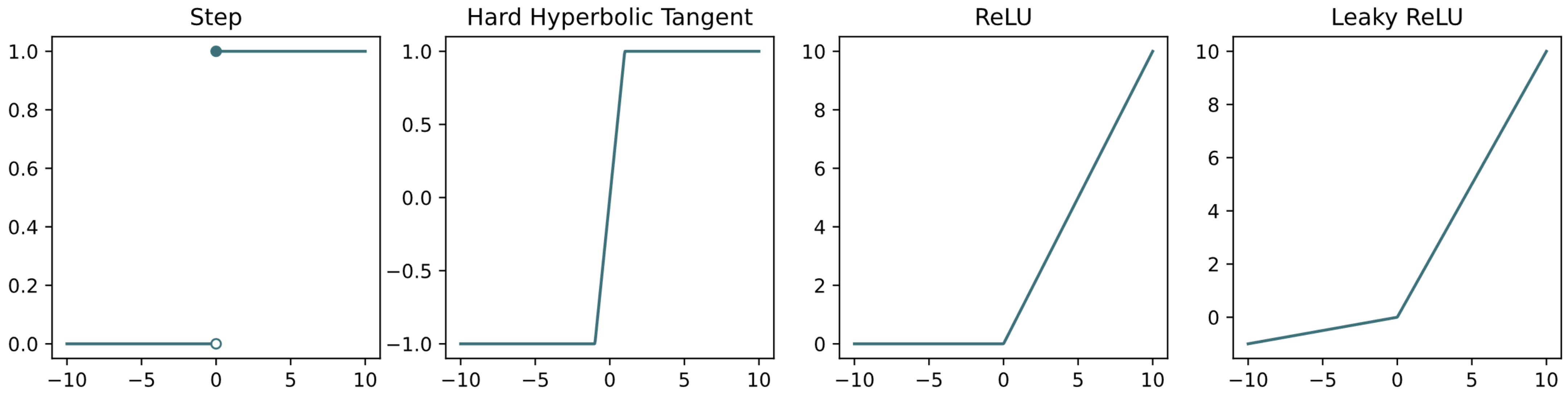


Decision Surface

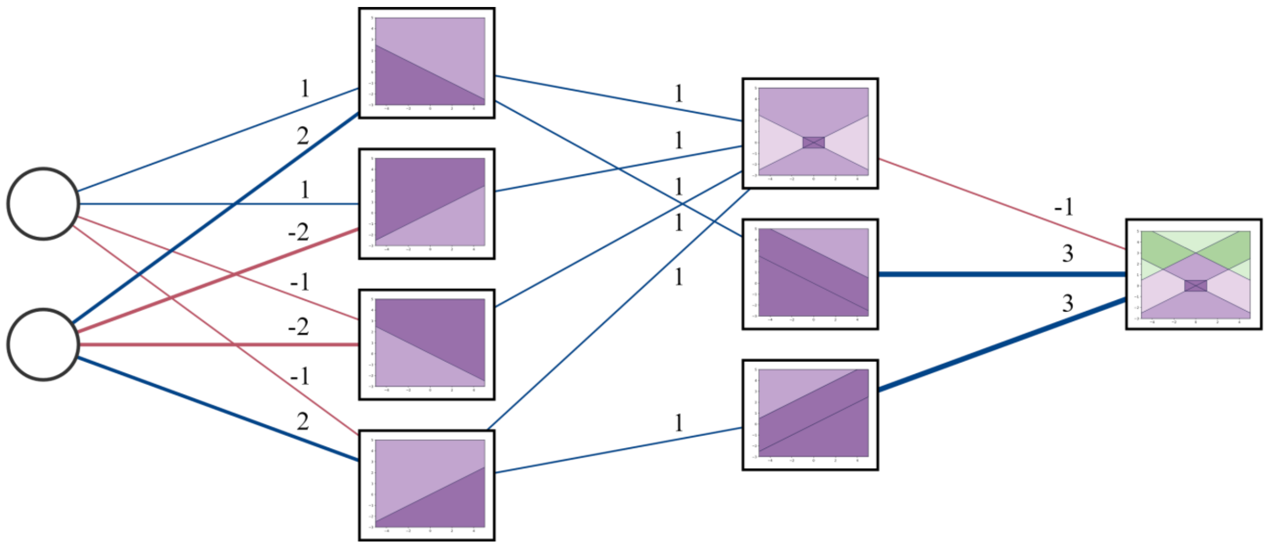


Decision Boundaries

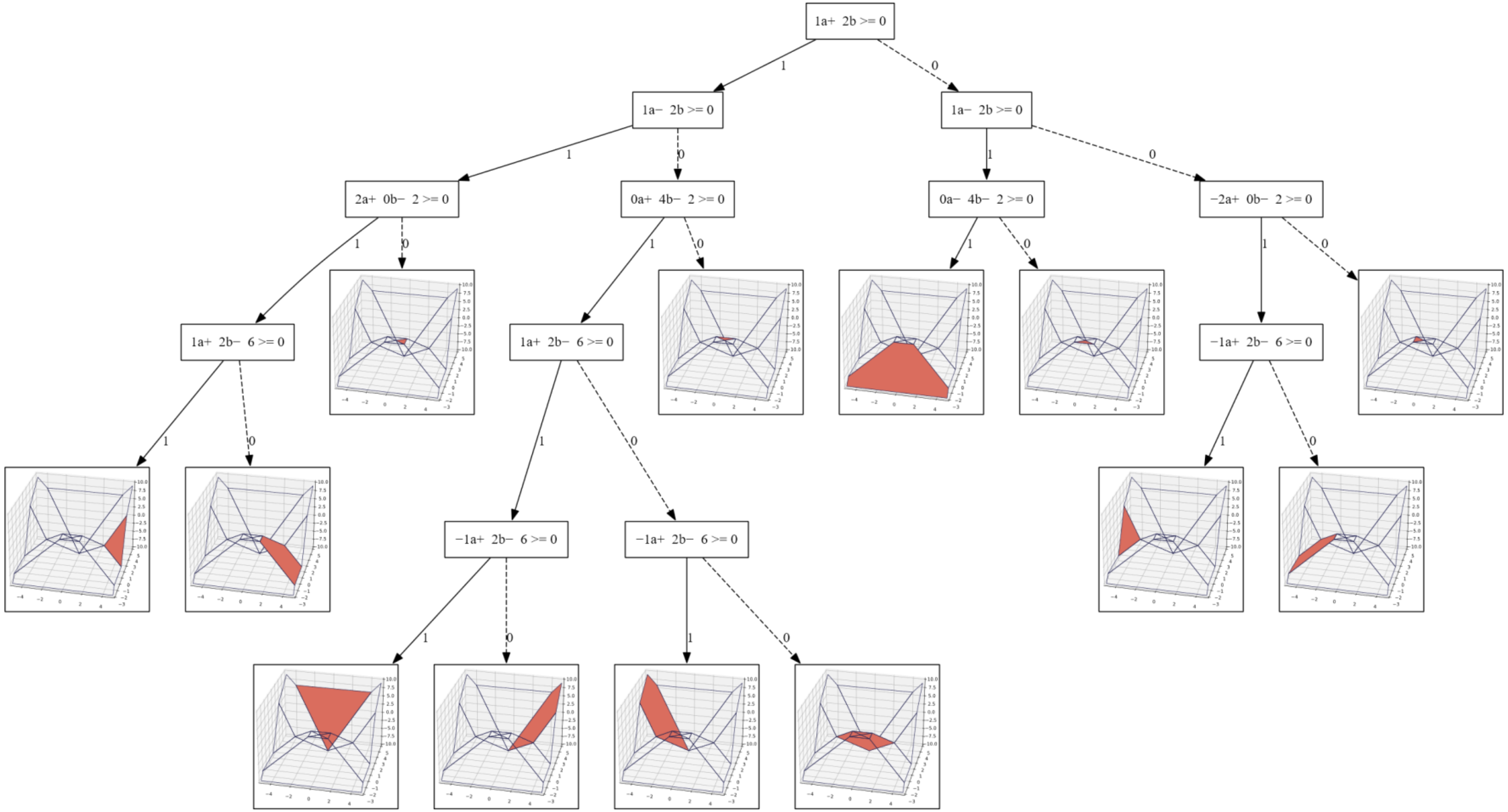
PWL Activation Functions



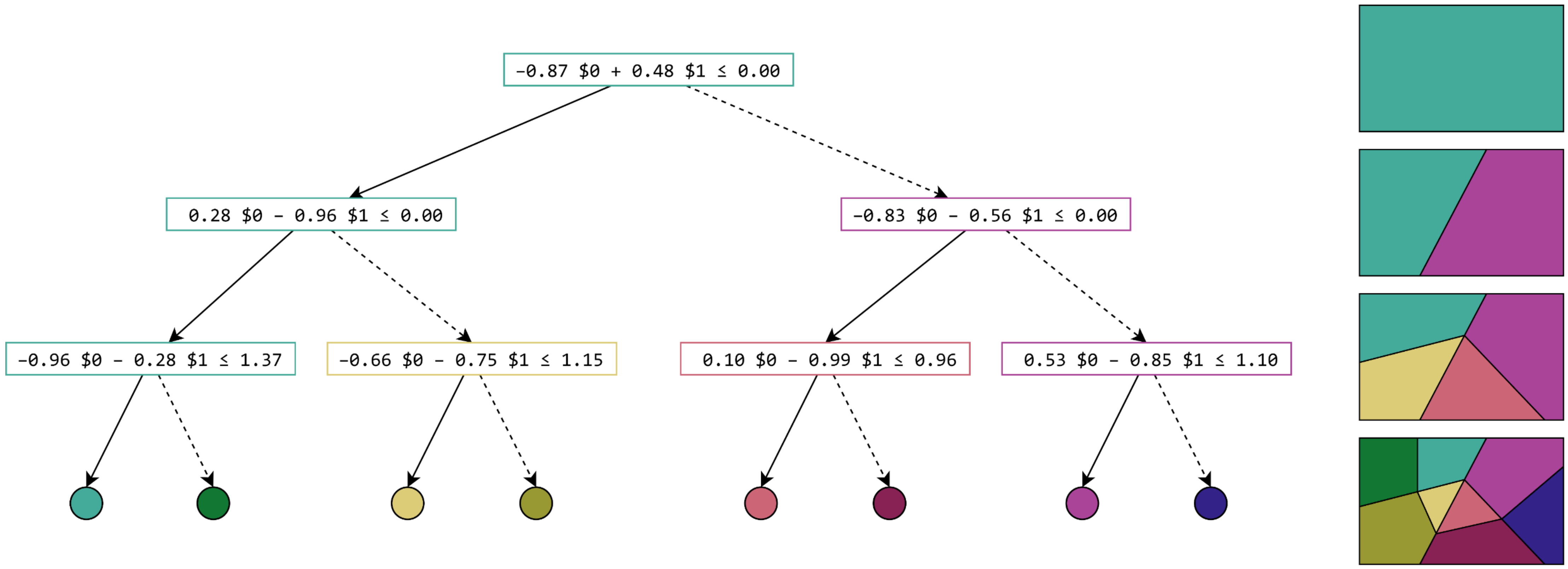
PWL Neural Networks



AffTree Example – Image (PWL)



AffTree Example – Preimage Partition

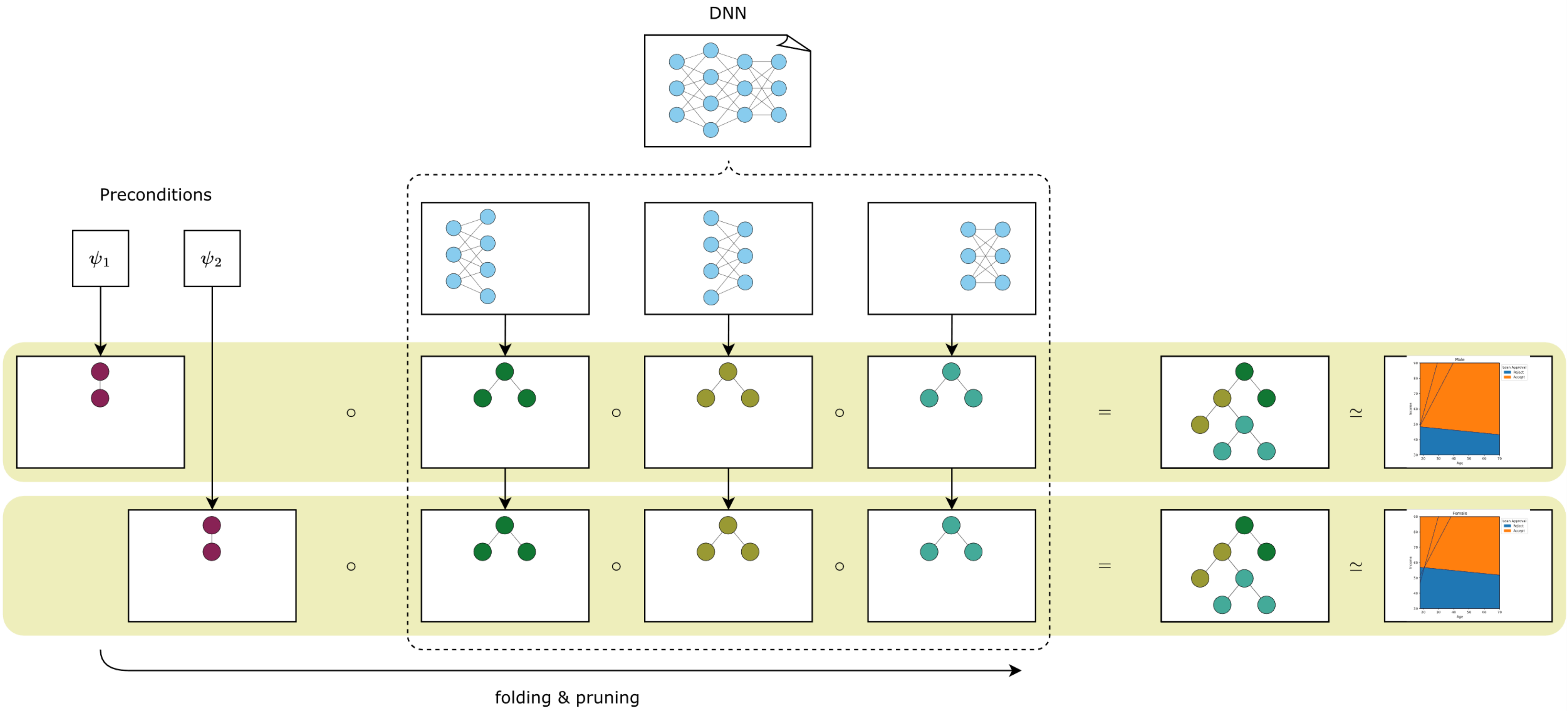


Fairness Example

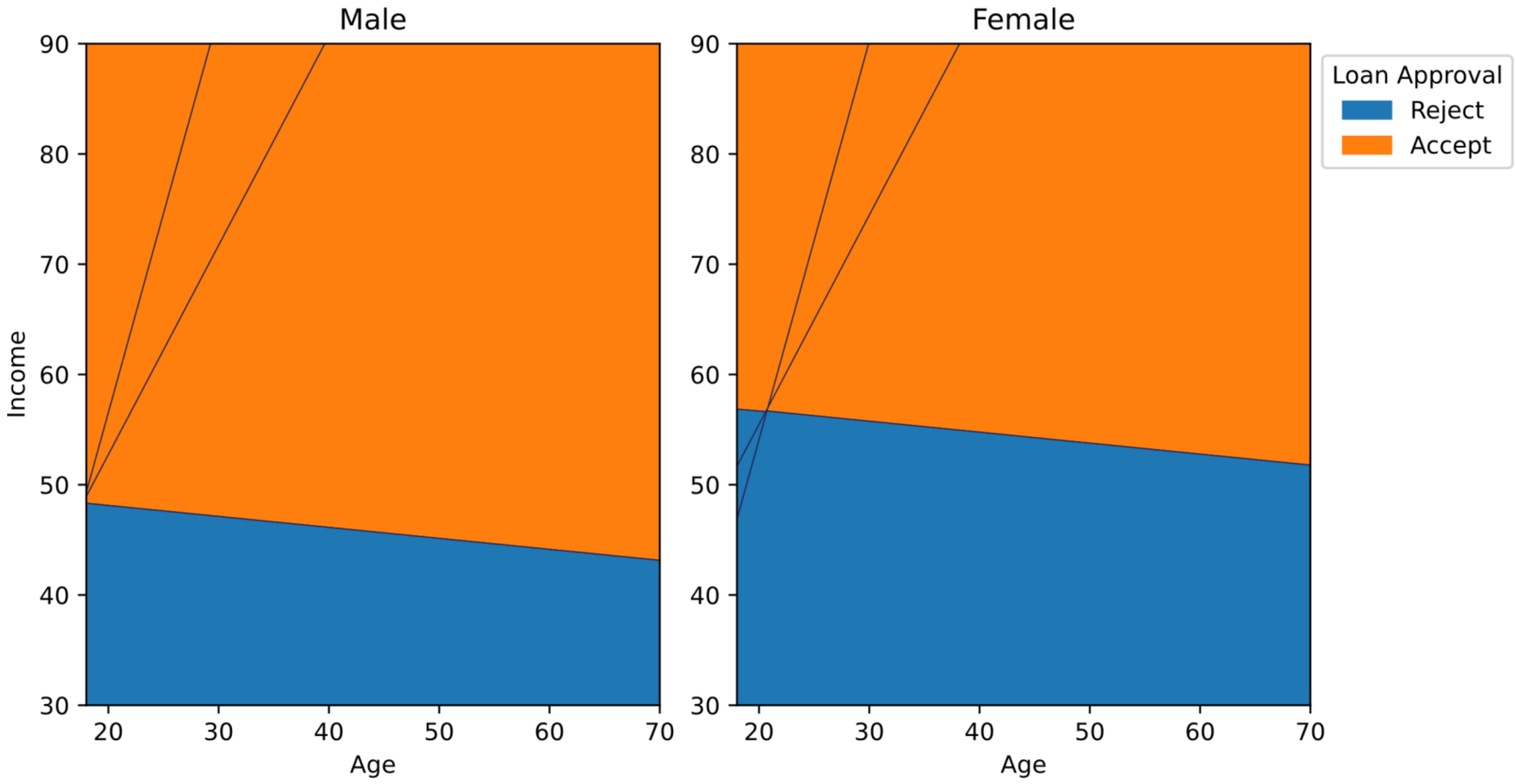
- Scenario: Loan Approval
- Features: Yearly Income, Age, Gender
- ML model: Deep Neural Network
- Trained on biased dataset
  - In reality not always know

- Question: Is the DNN biased?

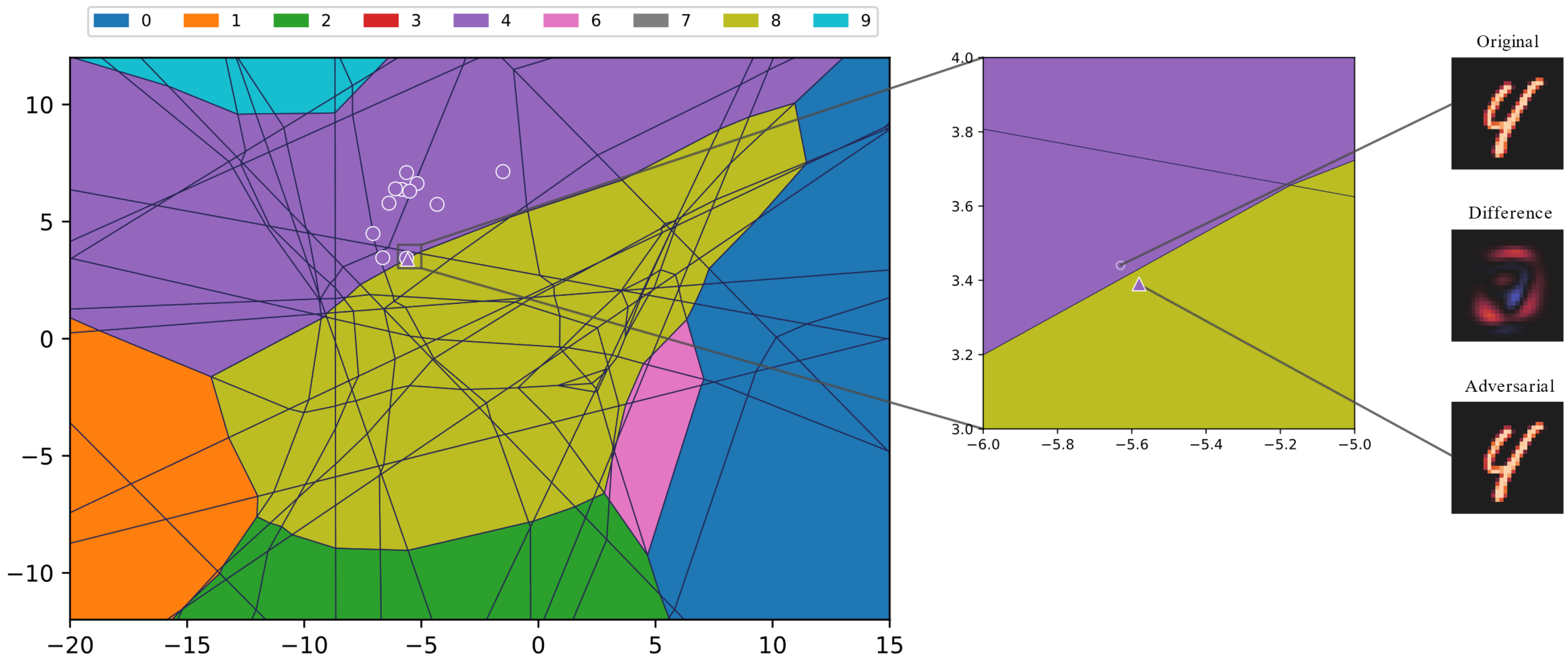
Technique – Slicing / Concolic Execution



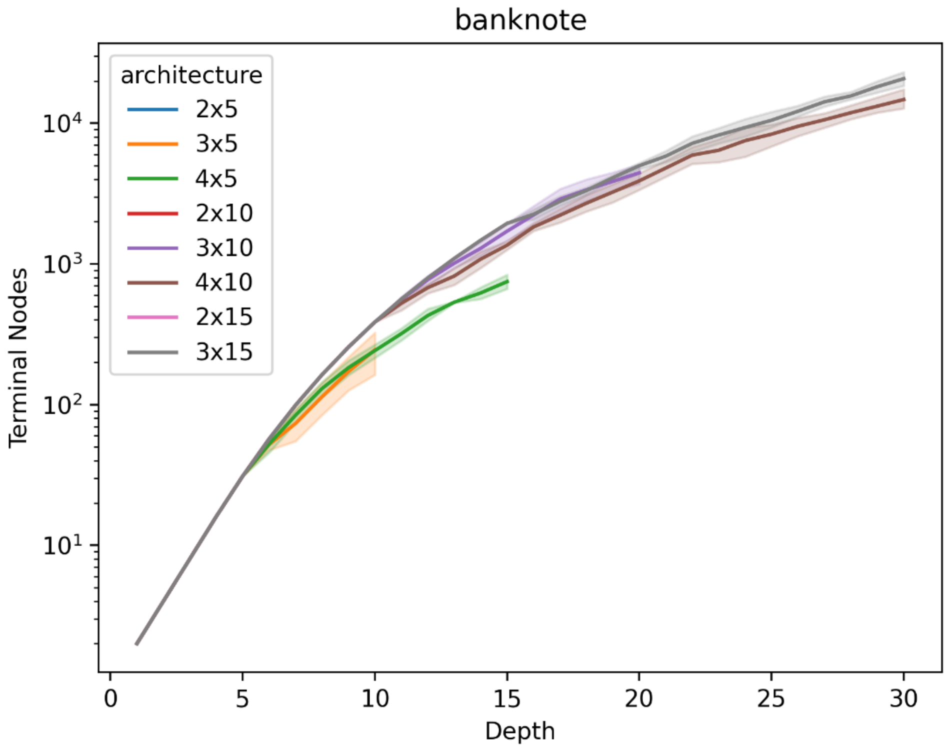
Fairness – Decision Boundaries

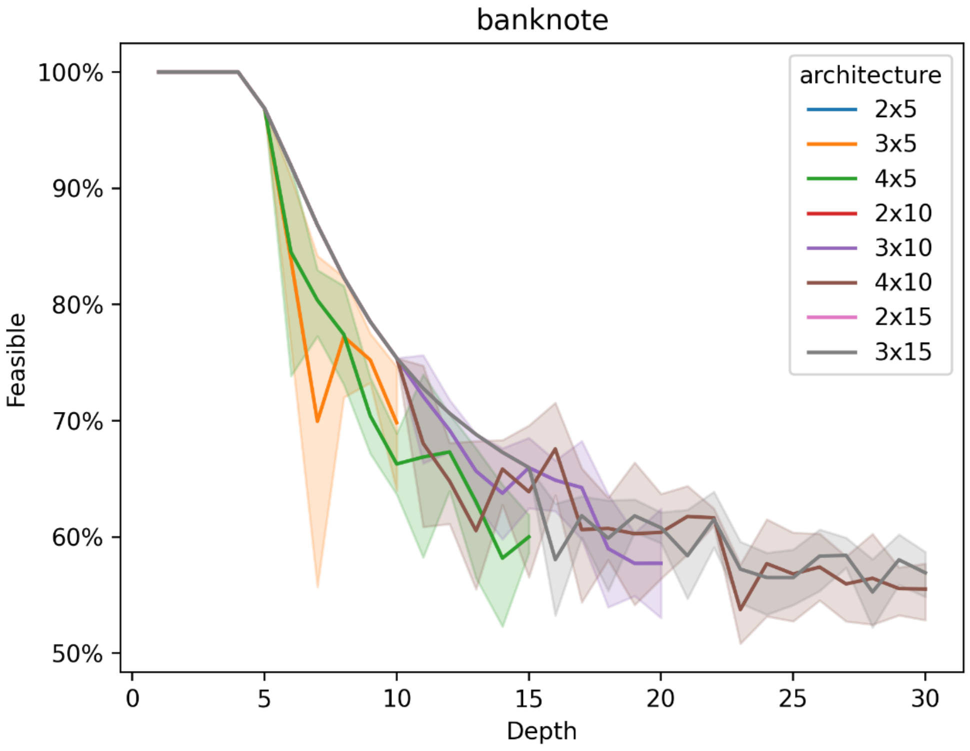


Adversarial Examples



Pruning Infeasible Paths





Overview

- **Brief Personal History**
  - Verification and Explanation: Concepts and Scalability
    - Random Forests
    - Deep Neural Networks
- **AI-Assisted Programming**
  - LLMs as part of Language-Driven (Softwareware) Engineering
- **Malwa: A Tool for Fully Automated Model Inference**
- **Conclusions and Perspectives**

Language-Driven Engineering (LDE)

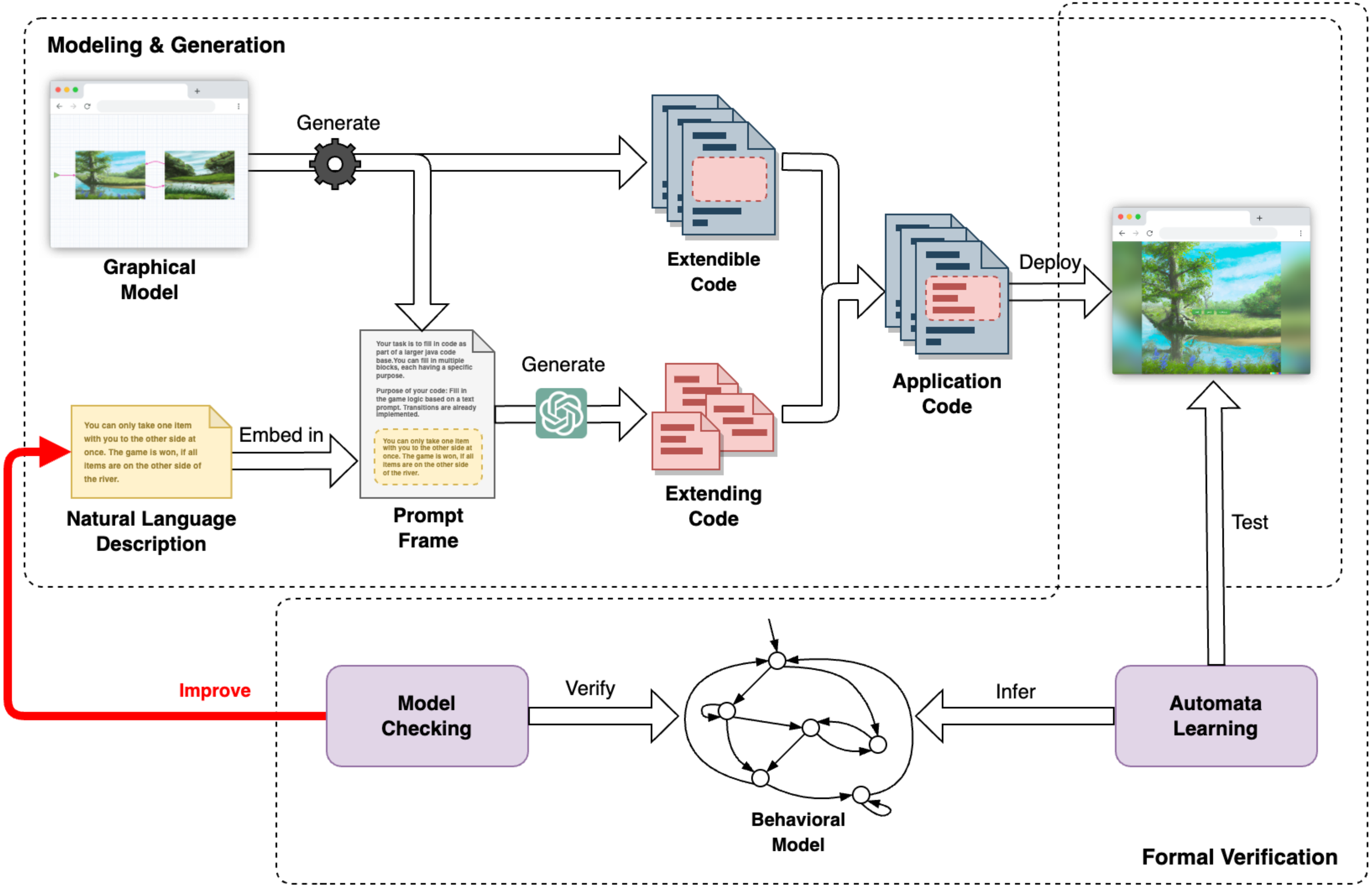
1. **Use of Domain-Specific Languages (DSLs):**
   - Tailored syntax and semantics for particular application domains.
   - Enhanced expressiveness for domain experts not familiar with traditional progamming languages
2. **Modeling and Code Generation:**
   - Automatic generation of code from high-level specifications.
   - Use of models as primary artifacts of the development process.
3. **Abstraction and Automation:**
   - High-level abstractions to simplify complex systems.
   - Automation of routine and error-prone tasks.
4. **Collaboration between Domain Experts and Developers:**
   - Enhanced communication through shared DSLs.
   - Domain experts can contribute directly to software development.
5. **Iterative and Incremental Development:**
   - Rapid prototyping and iterative refinement.
   - Continuous validation of models against domain requirements.

Main Goals

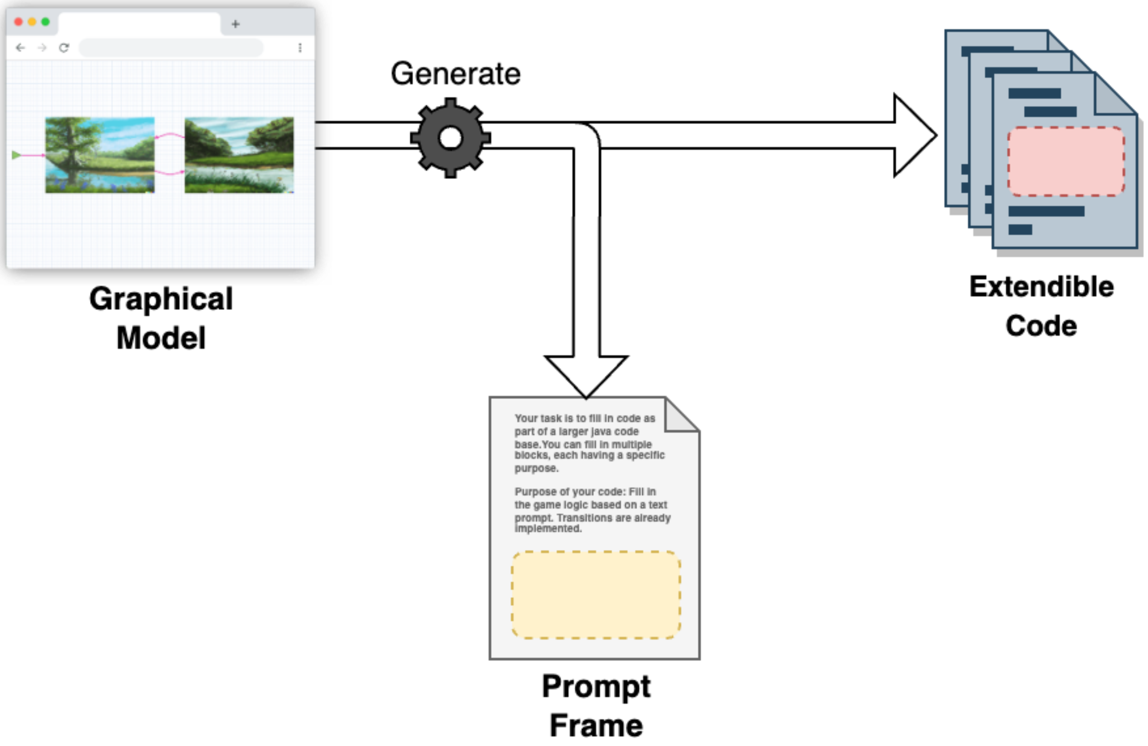
1. Low Code/No Code System Development
2. DSL-supported Mindsets
3. Formal Methods-Based Control
4. Now also with **(Domain-Specific) Natural Languages**

Viewpoint:

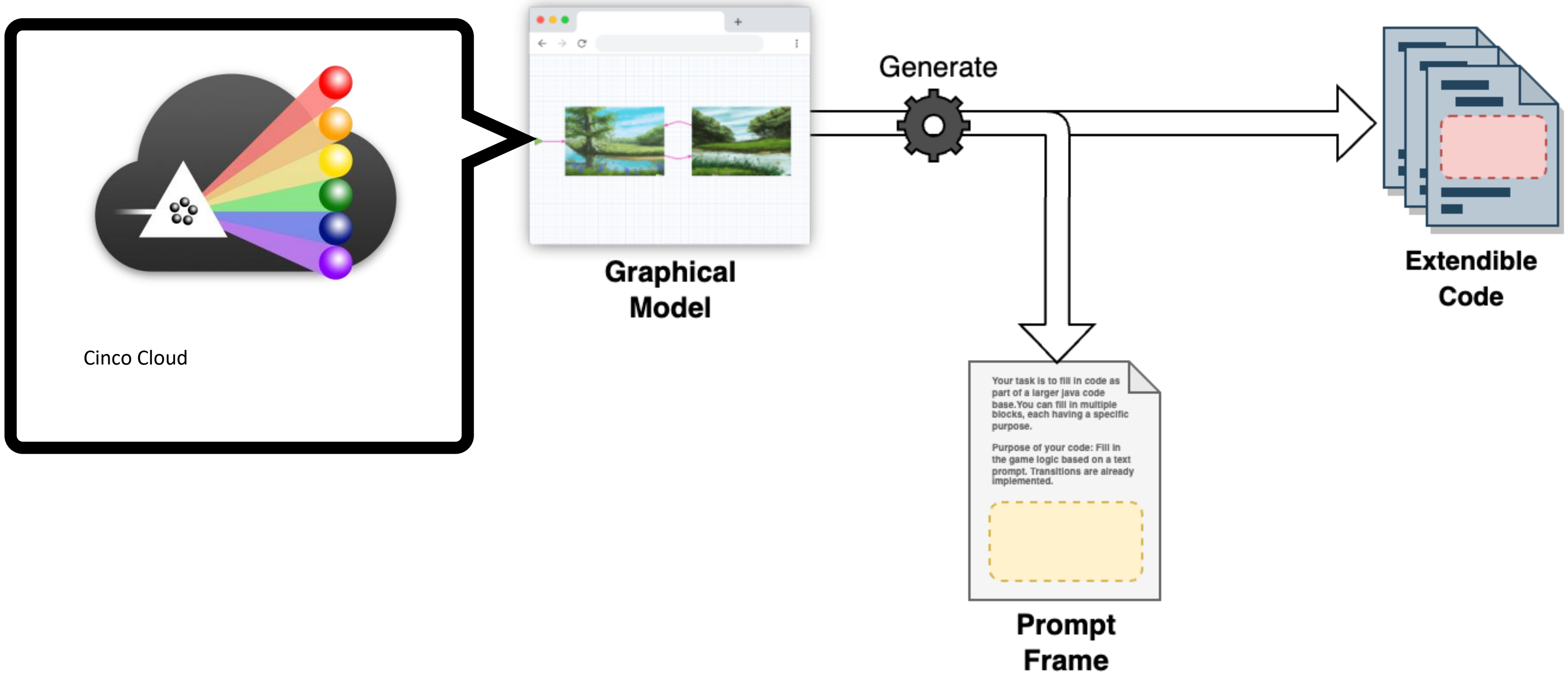
LLMs are considered Programmers!

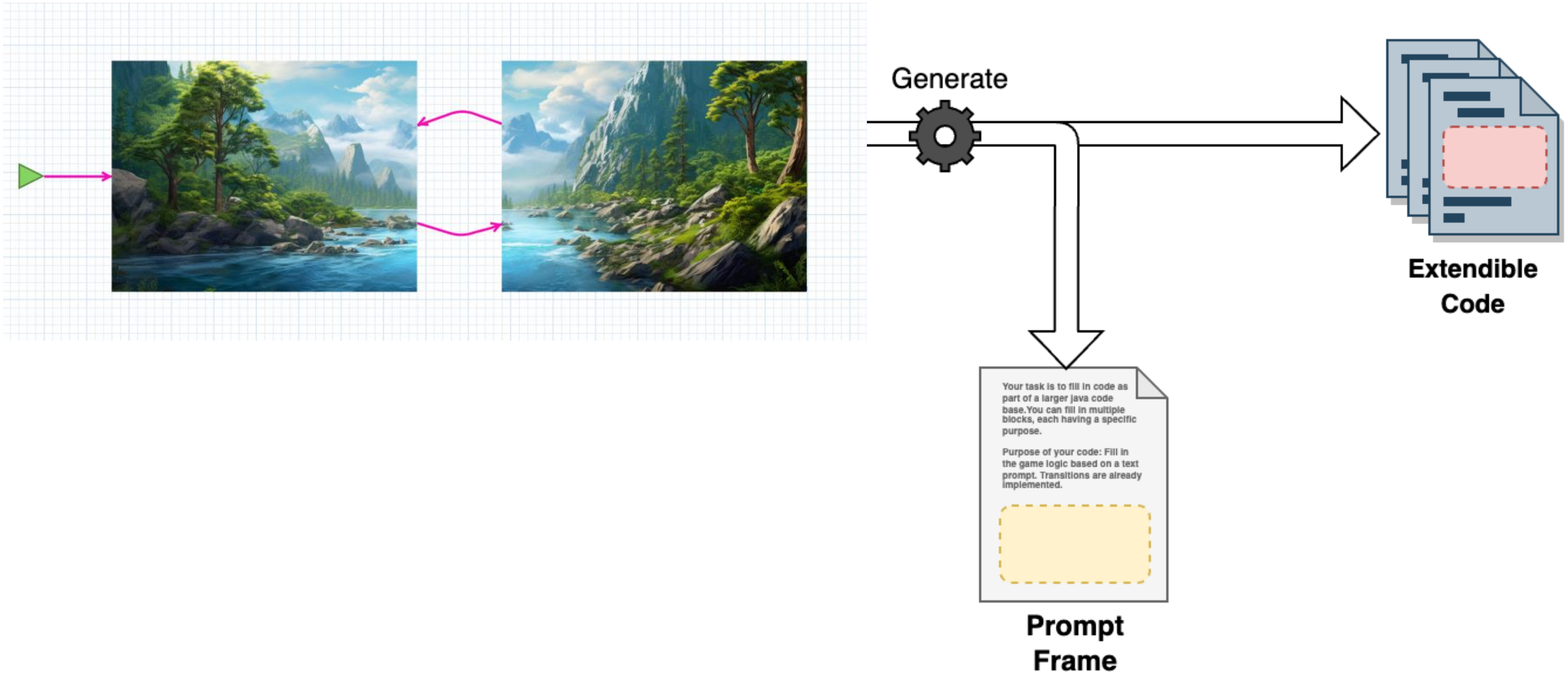


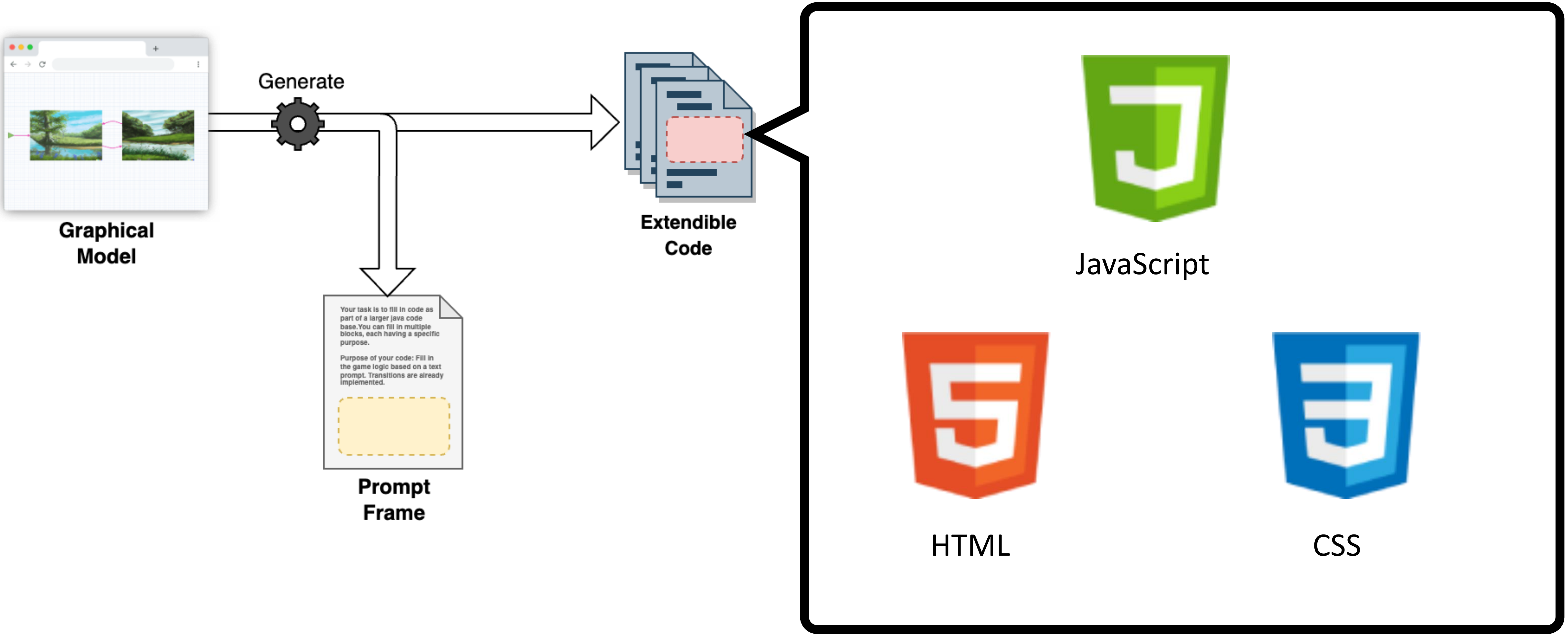
Modeling & Generation
Generate
Graphical Model
Extendible Code
Application Code
Deploy
Your task is to fill in code as part of a larger java code base. You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Generate
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Embed in
Natural Language Description
Prompt Frame
Extending Code
Test
Improve
Model Checking
Verify
Infer
Automata Learning
Behavioral Model
Formal Verification



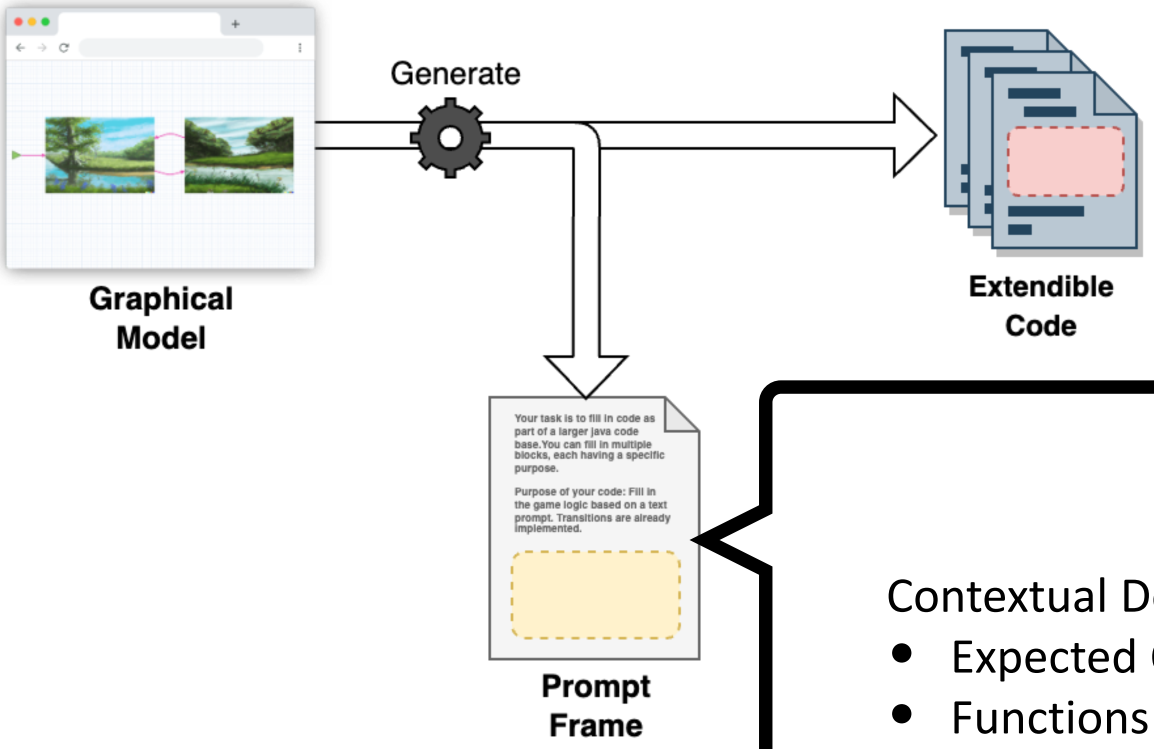
Generate
Graphical Model
Extendible Code
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
Prompt Frame


Cinco Cloud
Graphical Model
Generate
Extendible Code
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
Prompt Frame


Generate
Extendible Code
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
Prompt Frame

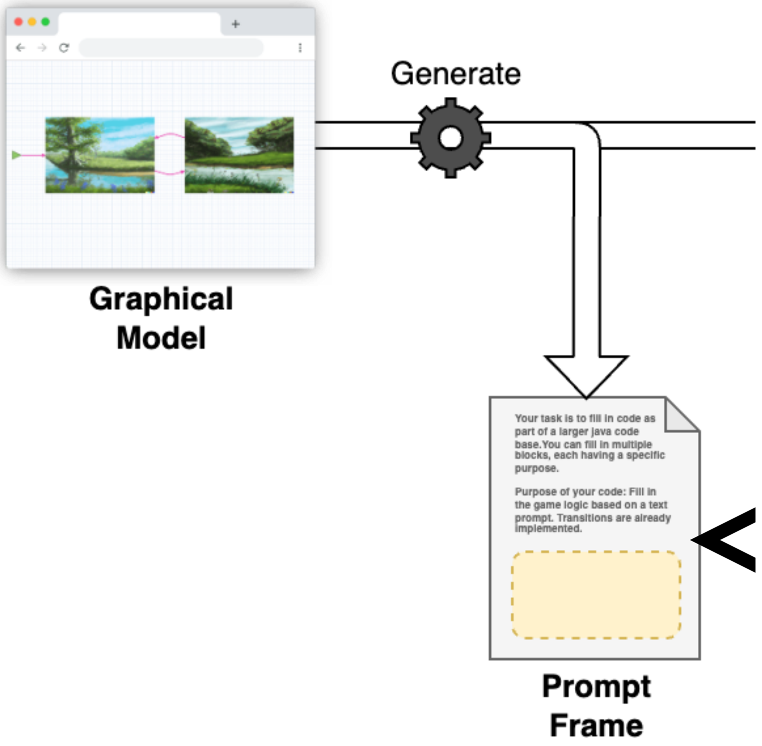


Generate
Graphical Model
Extendible Code
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
Prompt Frame
JavaScript
HTML
CSS

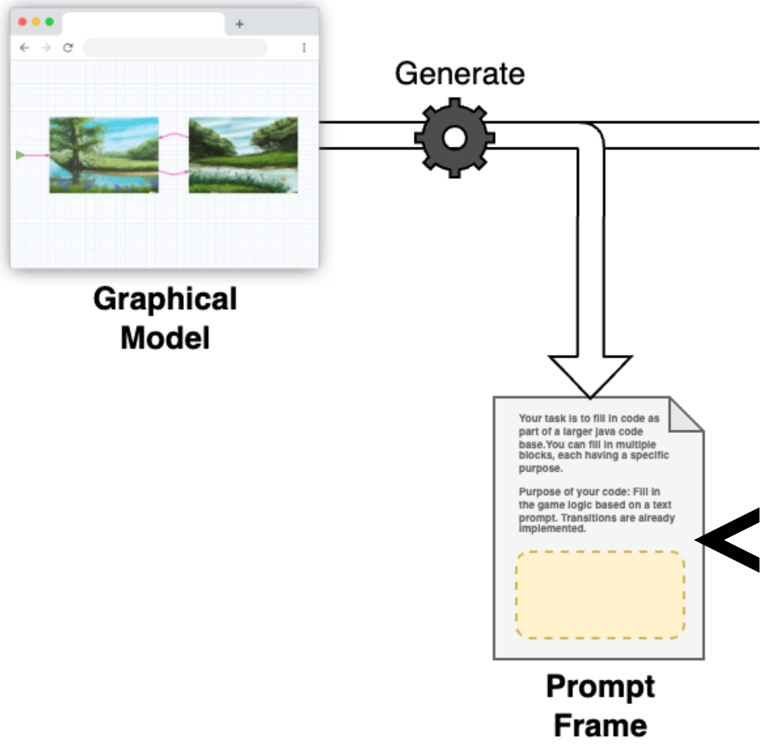


Contextual Domain Information:

- Expected Output
- Functions to implement
- Functions/variables available



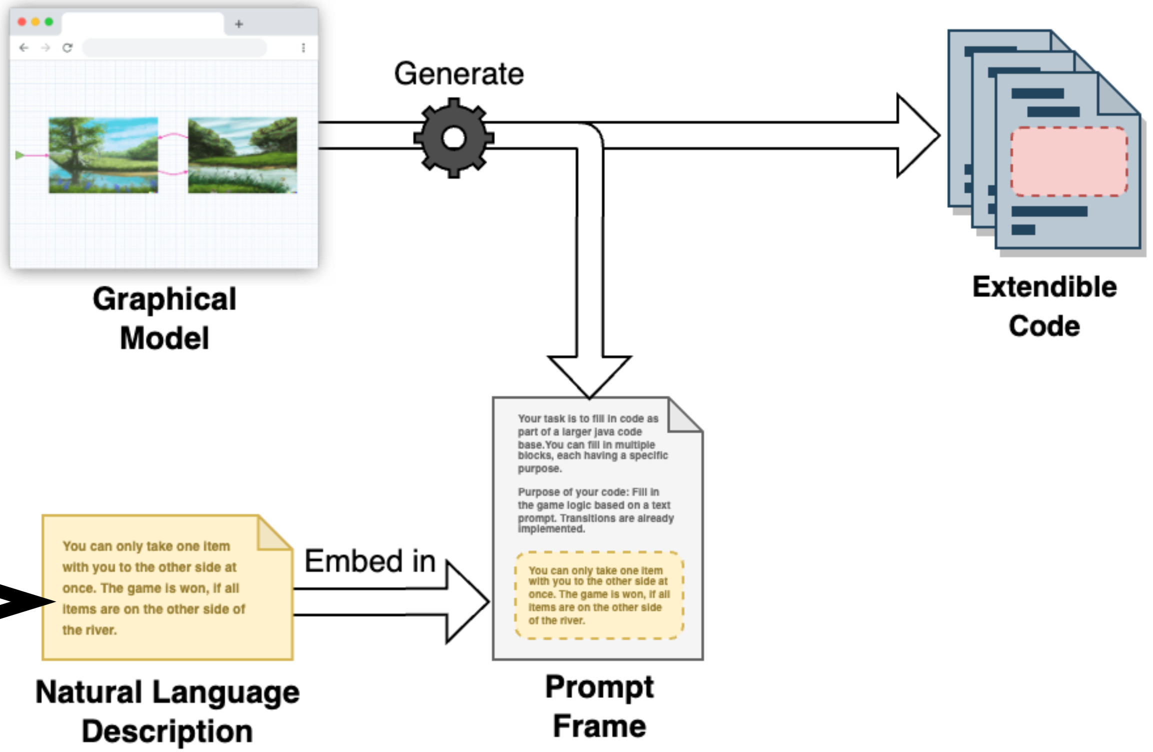

```
Your task is to fill in code as part of a larger JavaScript code base.

You can fill in multiple blocks, each having a specific purpose.

Larger context for code: a point-and-click adventure with state
    transitions. All code that you write is part of a game with
    attributes states=["leftRiver", "rightRiver"] and currentState
    which holds the currently visited river side and is therefore one
    of either values of states. These attributes are already written
    and you can safely assume that the rest of the code works as
    intended.

Purpose of your code: Fill in the game logic based on a text prompt.
    Game objects beside states and currentState should be objects
    having a name, and potentially multiple transition objects that
    contain a screen property which is the name the transition can be
    triggered on, and a function property which is the transition
    function for this screen. Game objects are considered present in a
    state if they possess the currentScreen property of the state.

The code blocks for you to implement:

// [...], see Listing 2

Prompt: // [...], see Listing 3

Answer as follows: Write down ONLY the filled in code blocks with the
    code that you seem fit. Add comments if you want but do NOT explain
    anything about the code, your answer should ONLY contain javascript
    code.
```

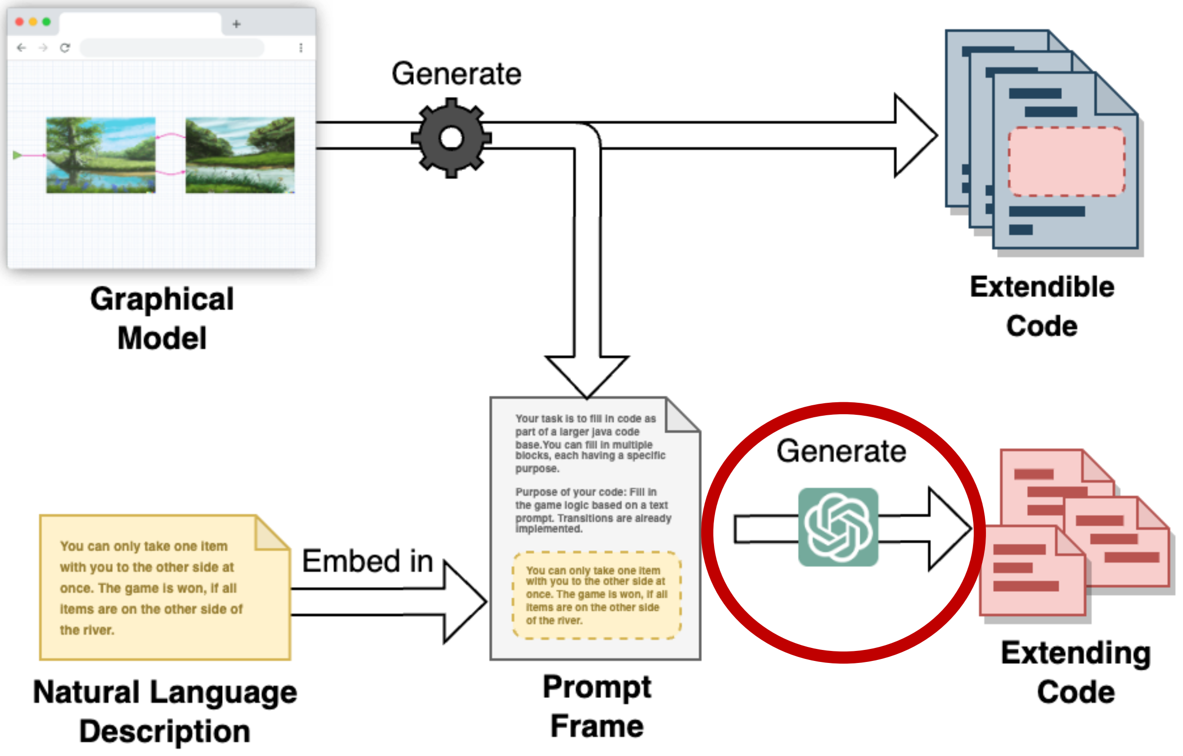



```
Your task is to fill in code as part of a larger JavaScript code base.

You can fill in multiple blocks, each having a specific purpose.

Larger context for code: a point-and-click adventure with state
```

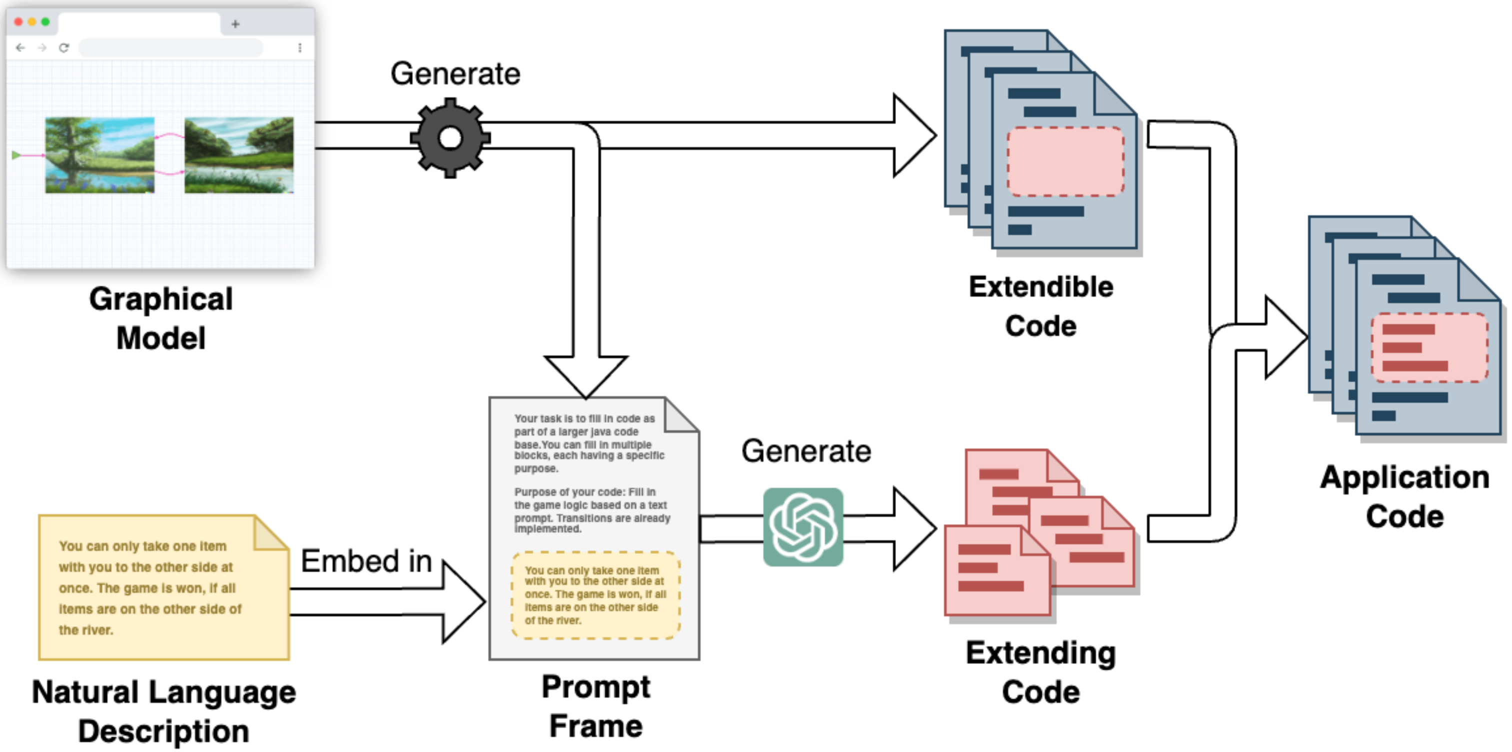
```
function initVariables() {
  // state objects should be of the following form
  // this.gameObjects = [
  //   {
  //     name: 'someName',
  //     currentScreen: 'someState',
  //     transitions: [
  //       {
  //         screen: 'someState',
  //         function: () => ()
  //       }
  //     ],
  //   }
  // ]
  // they can possess multiple transitions and are only
  // rendered on screens they have transitions for

  this.gameObjects = []; // fill in
}

function checkWin() { // fill in }

function checkLoss() { // fill in }
```

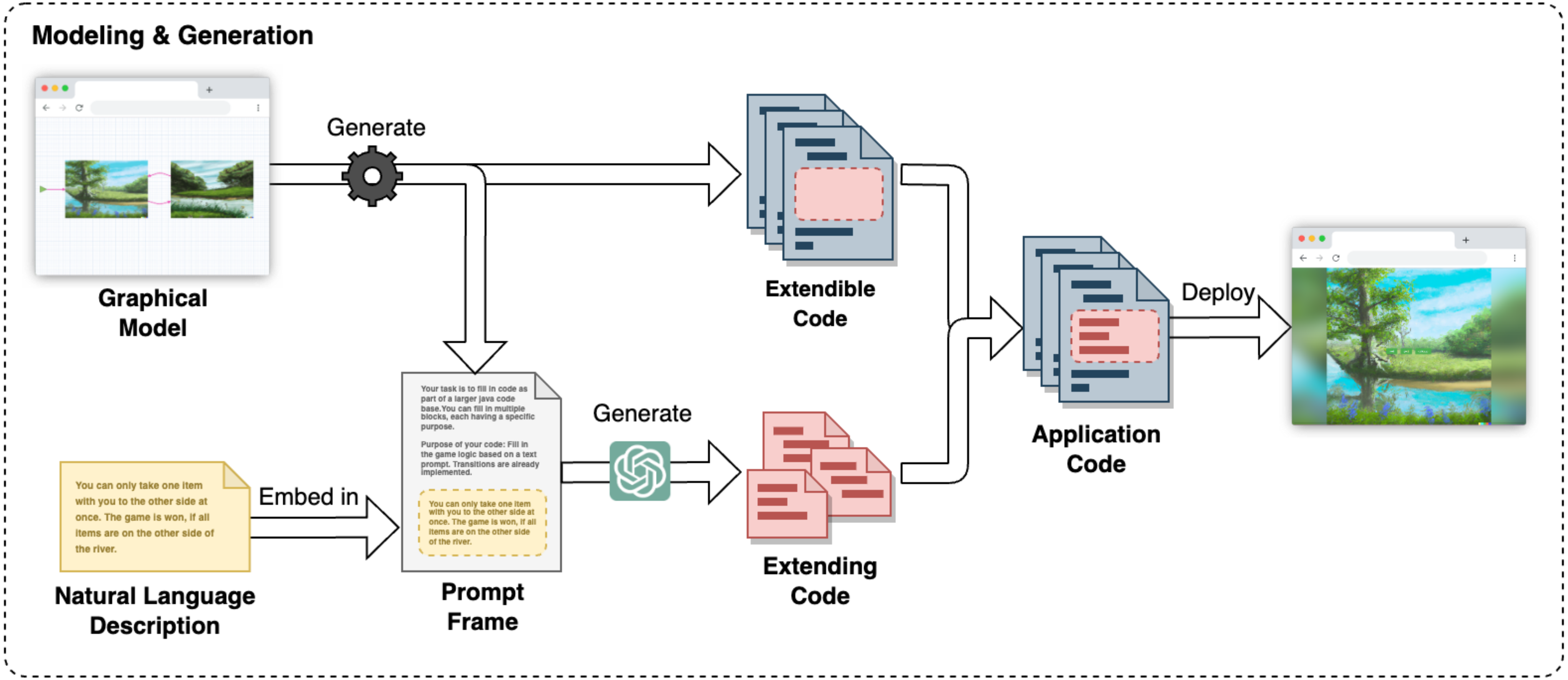
On the left side of the river there are a wolf, a goat, and a cabbage.
The game is won when each object is moved to the right side of the river.
The game is lost if the wolf and the goat are on the same side of the river, while the player is on the other side, or if the cabbage and the goat are on the same side of the river while the player is on the other side.

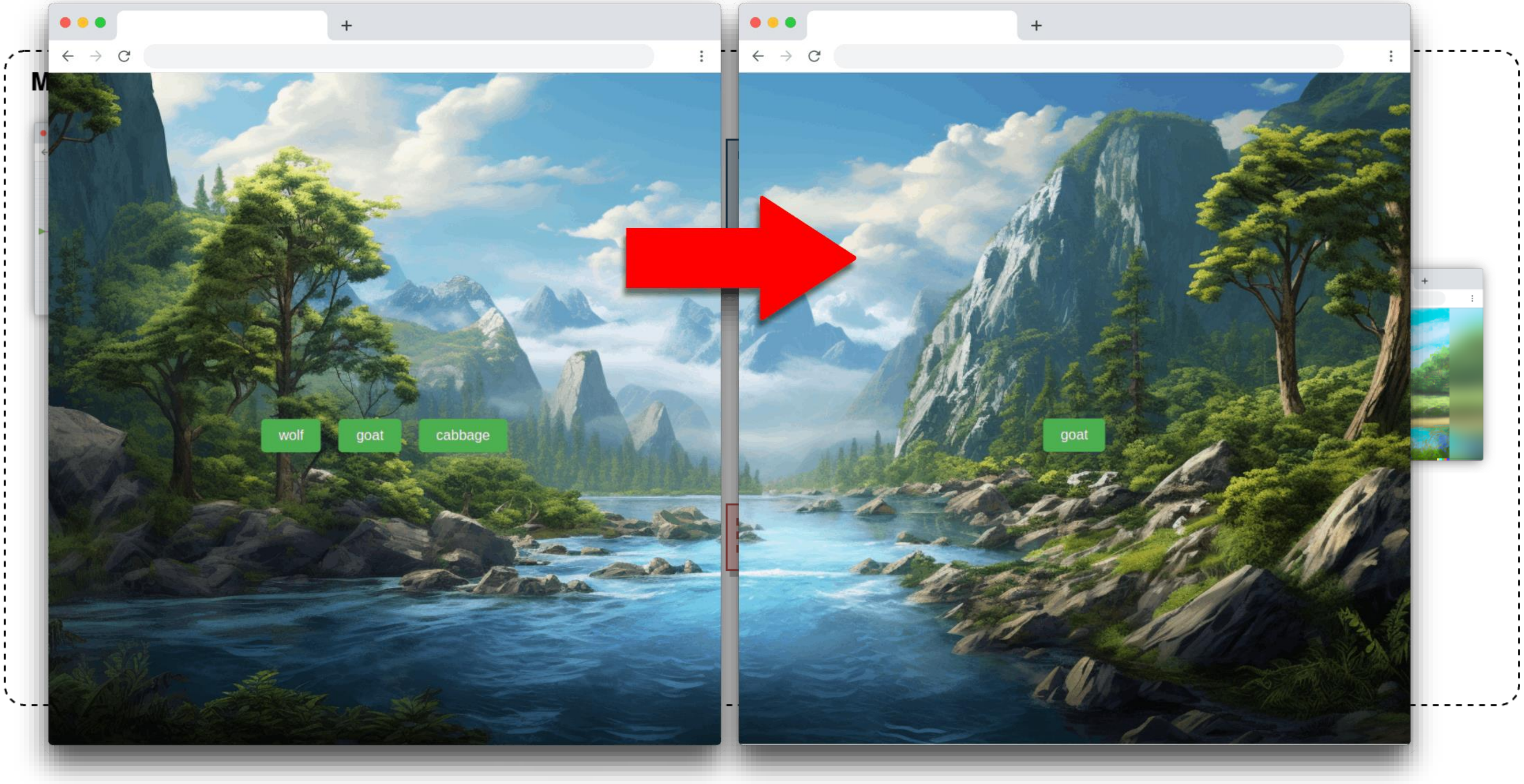


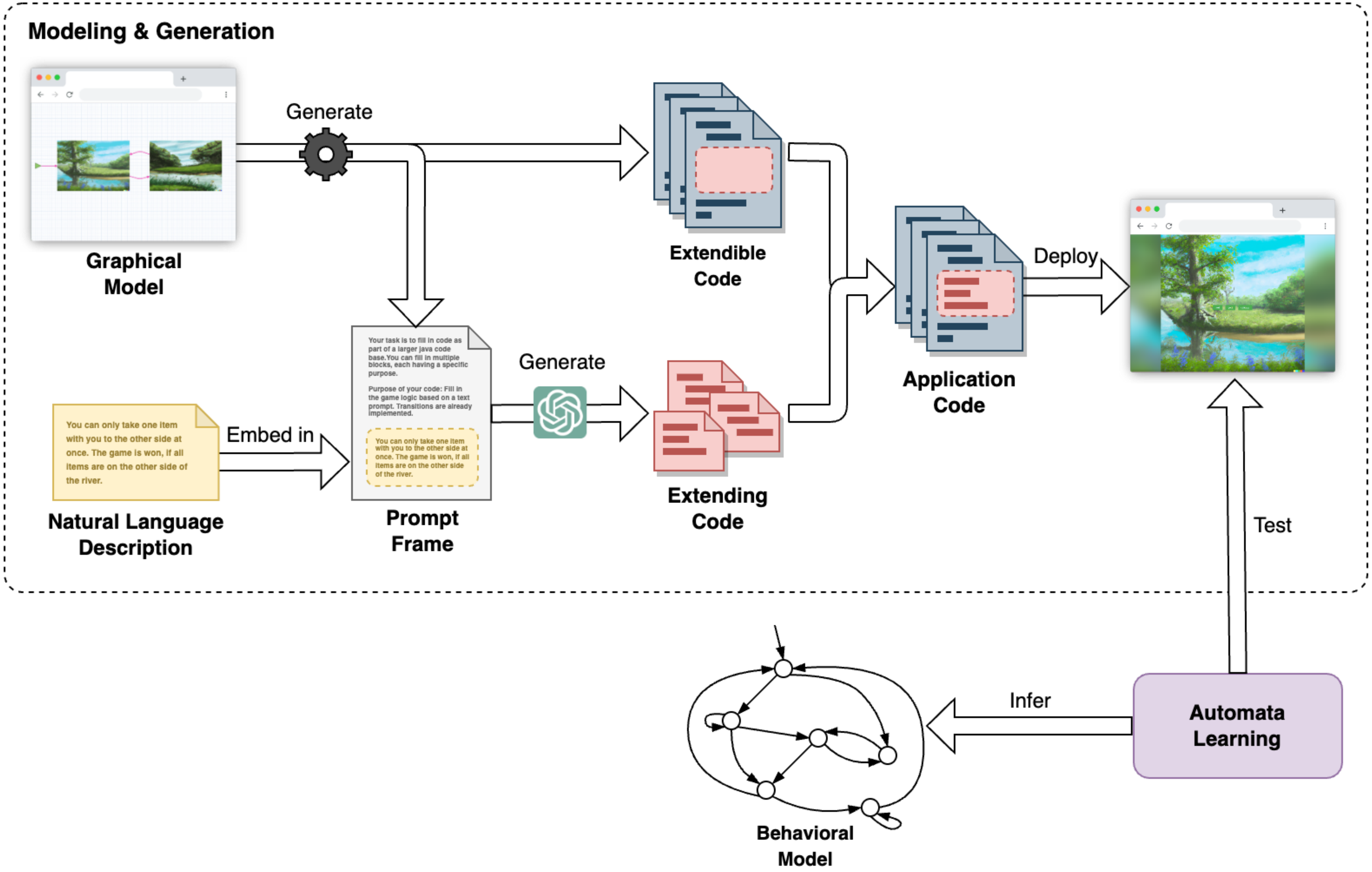


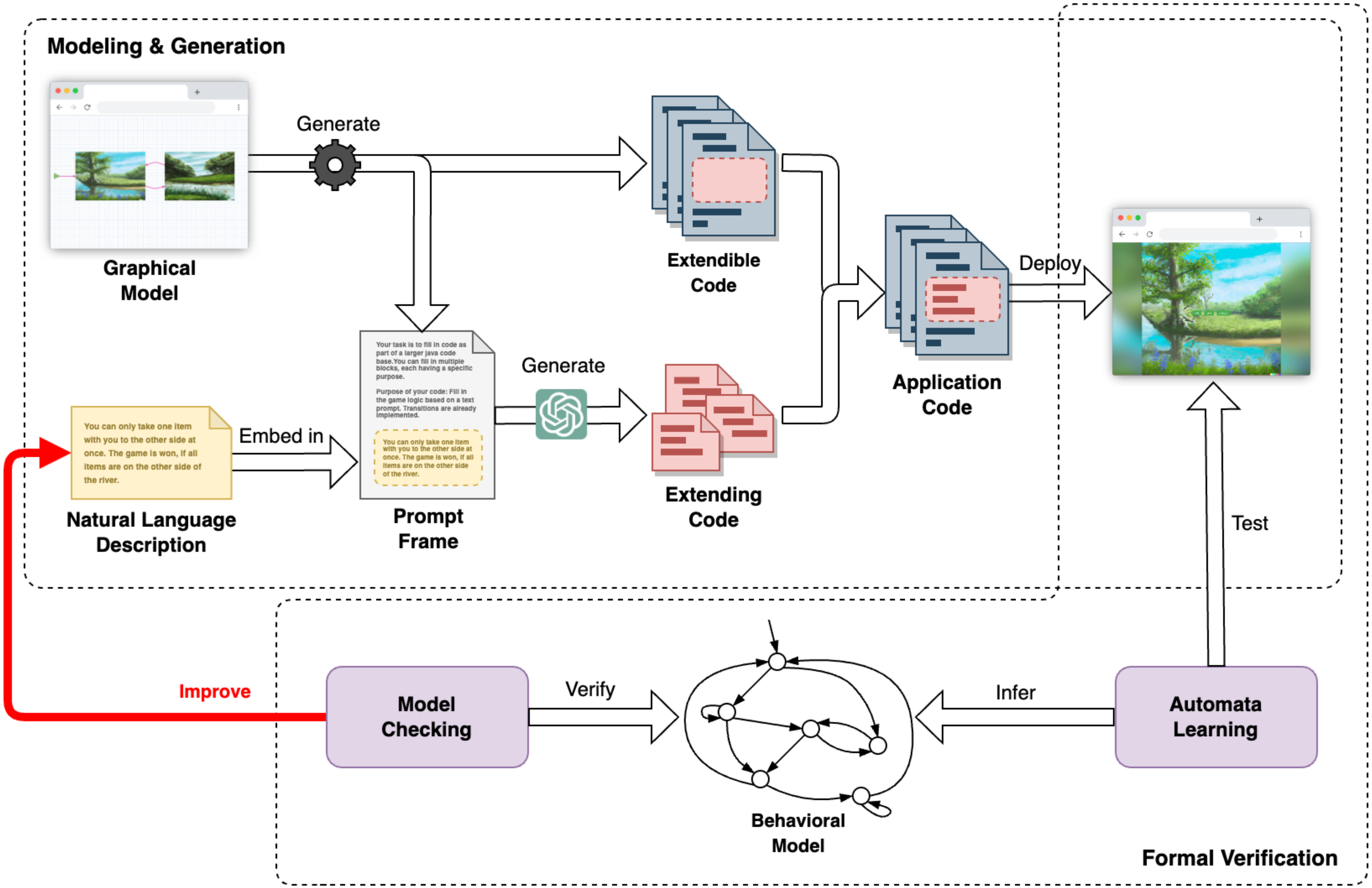
Generate
Graphical Model
Extendible Code
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Embed in
Generate
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Natural Language Description
Prompt Frame
Extending Code

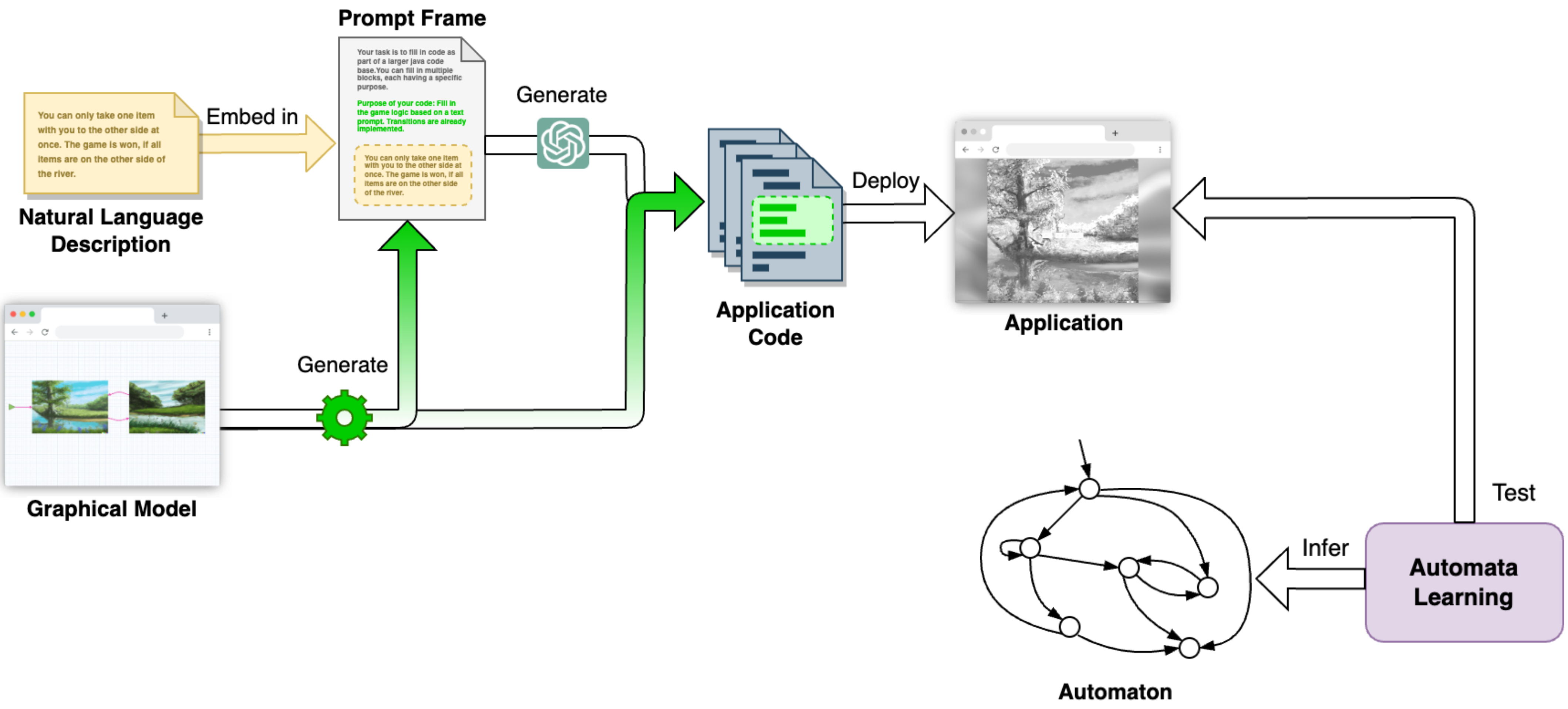

Generate
Graphical Model
Extendible Code
Application Code
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Generate
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Embed in
Natural Language Description
Prompt Frame
Extending Code


Modeling & Generation
Generate
Graphical Model
Extendible Code
Application Code
Deploy
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Generate
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Embed in
Natural Language Description
Prompt Frame
Extending Code

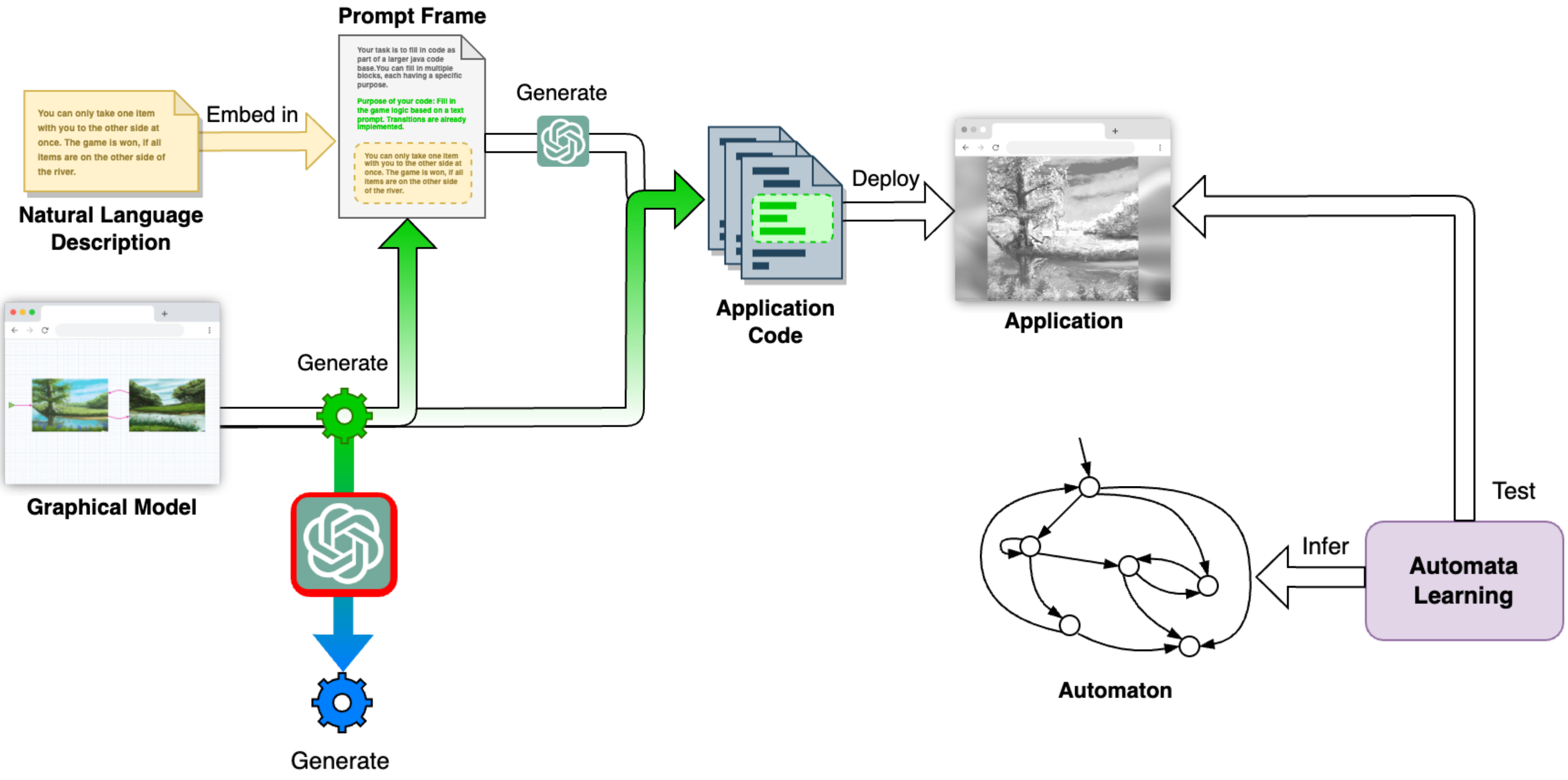

wolf
goat
cabbage
goat

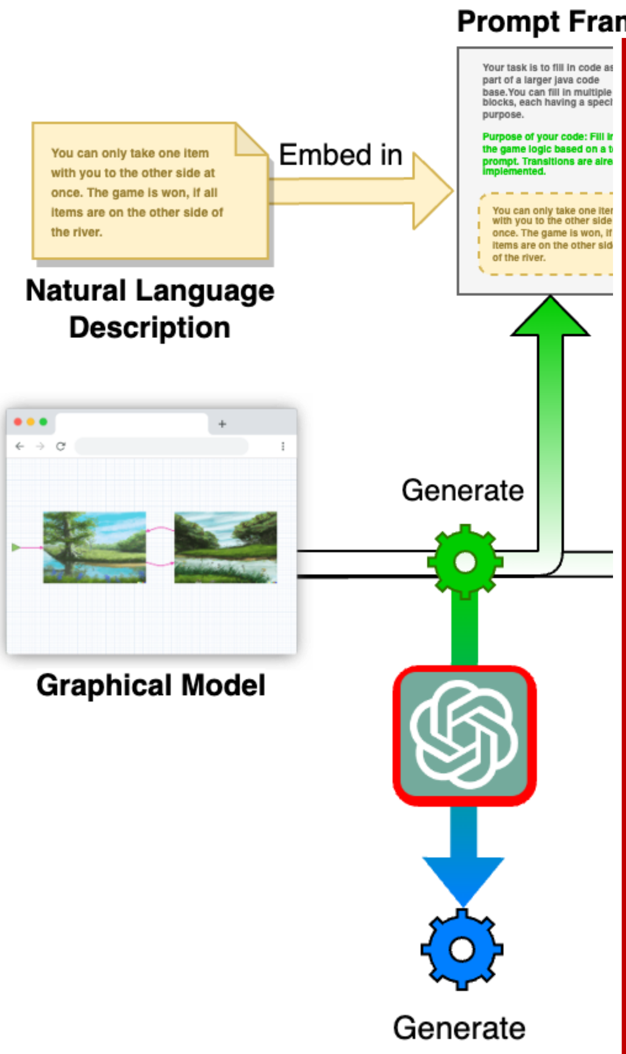

Modeling & Generation
Generate
Graphical Model
Extendible Code
Application Code
Deploy
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Generate
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Embed in
Natural Language Description
Prompt Frame
Extending Code
Test
Infer
Automata Learning
Behavioral Model


Modeling & Generation
Generate
Graphical Model
Extendible Code
Application Code
Deploy
Natural Language Description
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Embed in
Prompt Frame
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
Generate
Extending Code
Test
Improve
Model Checking
Verify
Behavioral Model
Infer
Automata Learning
Formal Verification



Daniel Busch et al. “ChatGPT in the Loop – A Natural Language Extension for Domain-Specific Modeling Languages”. In: Lecture Notes of Computer Science. Vol. 14380. Springer, 2023.

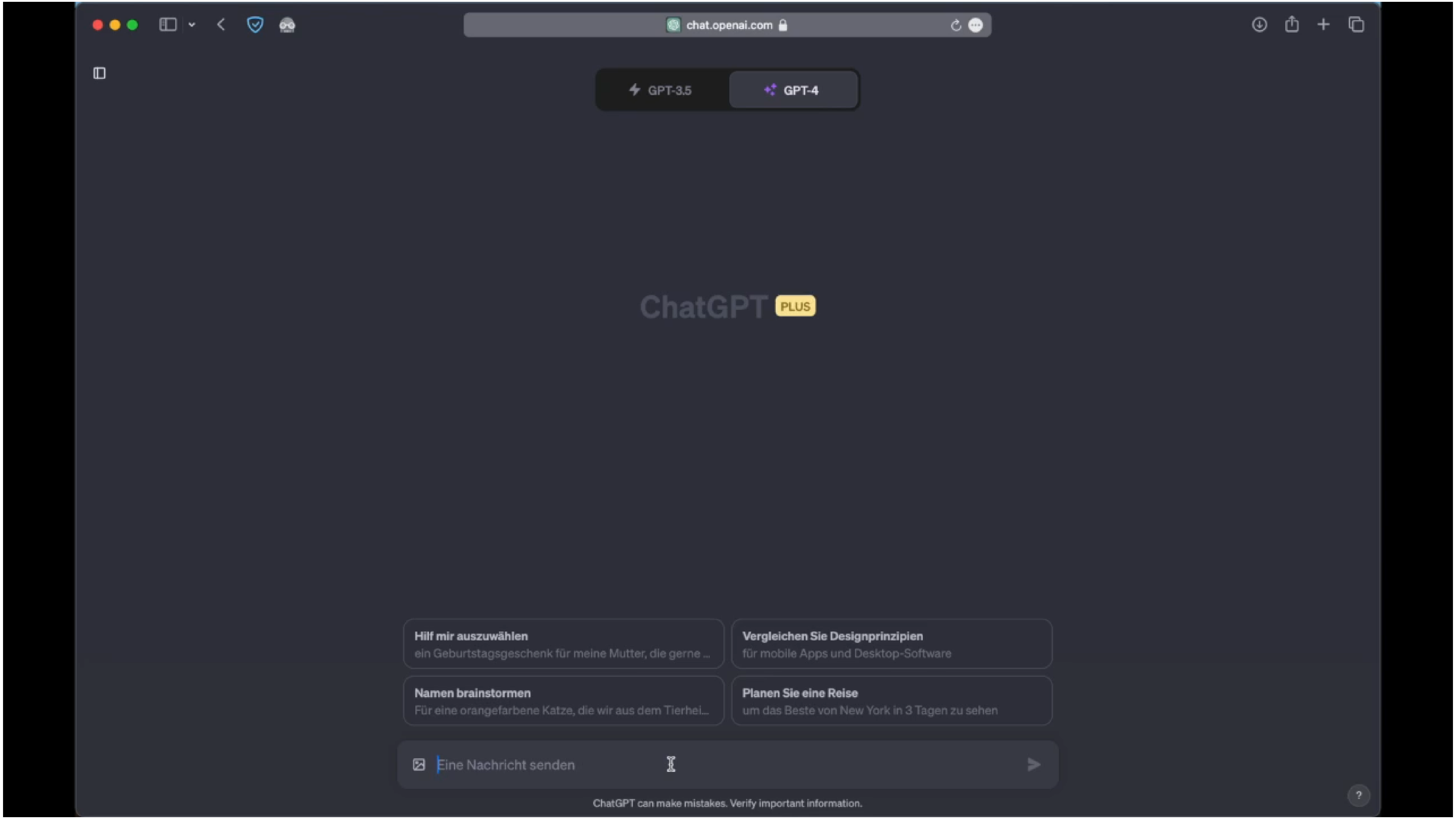

Prompt Frame
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Embed in
Natural Language Description
Generate
Application Code
Deploy
Application
Graphical Model
Generate
Generate
Automaton
Infer
Test
Automata Learning



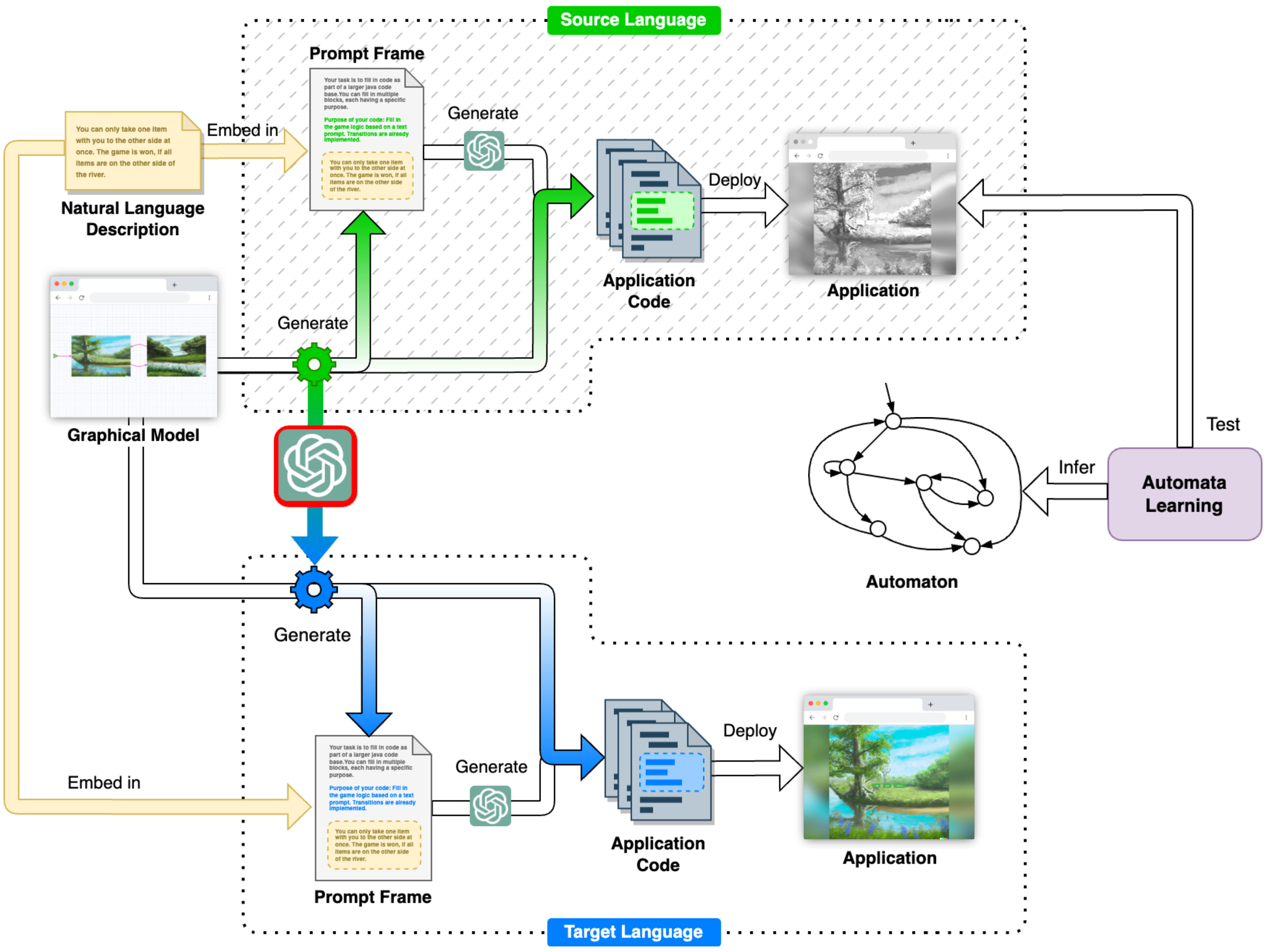
You are provided with prompt frames. The prompt frame is wrapped into "BEGIN PROMPT FRAME" and "END PROMPT FRAME". The prompt frame includes ALL text AND code. These prompt frames should be used for yourself to provide you with information to get a desired code output for an input scenario.

Your overall task will be to modify the given prompt frame so that you output a modified prompt frame for another programming language instead of the given prompt frame.

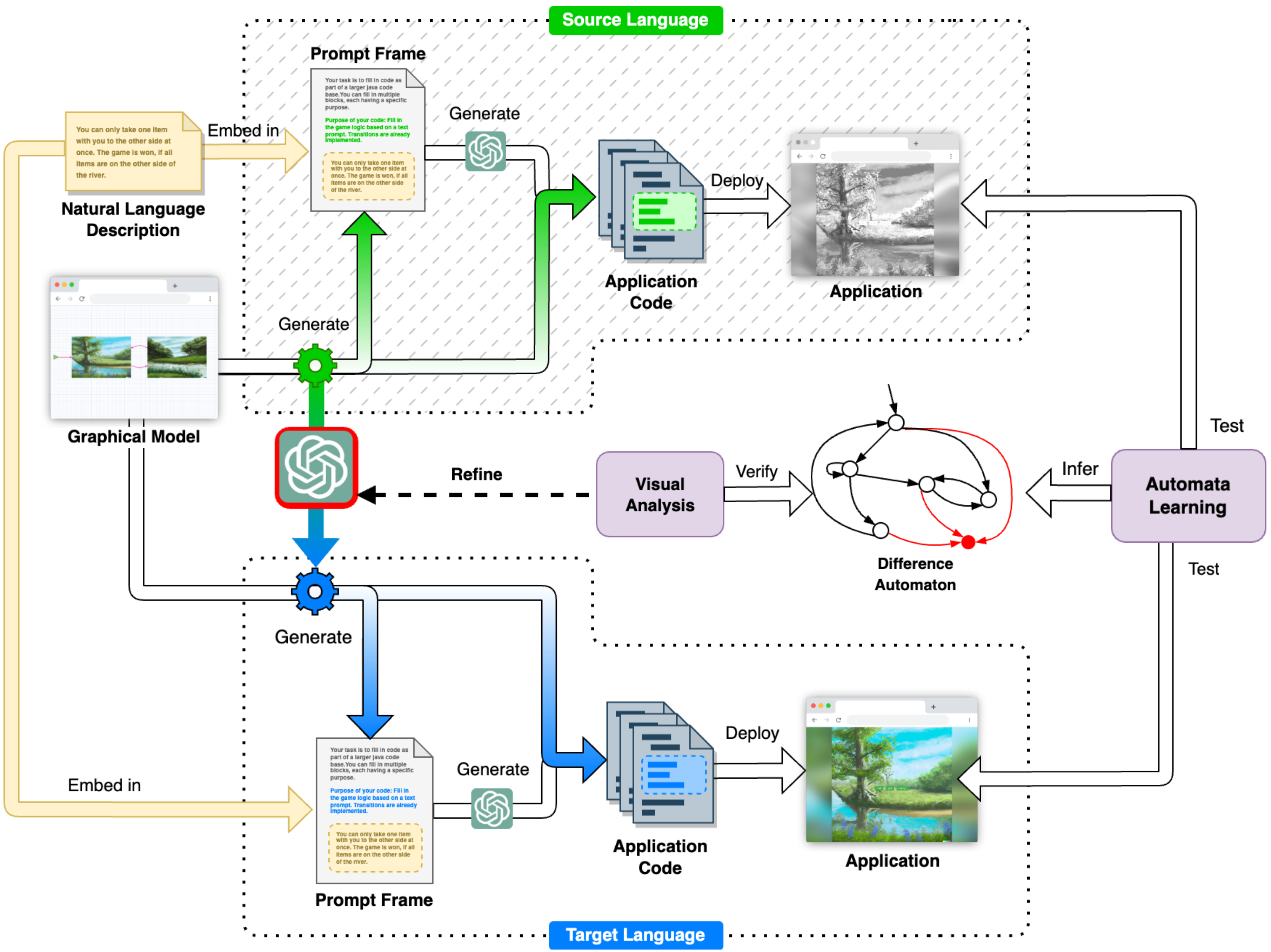
Answer only as follows in two interactions:
1. First, output only the programming language for which the given prompt frame seems to be made, and ask the user which programming language you should migrate the prompt frame to.
2. After receiving the user's answer, display only the migrated prompt frame and no additional text.



chat.openai.com
GPT-3.5
GPT-4
ChatGPT PLUS
Hilf mir auszuwählen
ein Geburtstagsgeschenk für meine Mutter, die gerne ...
Vergleichen Sie Designprinzipien
für mobile Apps und Desktop-Software
Namen brainstormen
Für eine orangefarbene Katze, die wir aus dem Tierhei...
Planen Sie eine Reise
um das Beste von New York in 3 Tagen zu sehen
Eine Nachricht senden
ChatGPT can make mistakes. Verify important information.

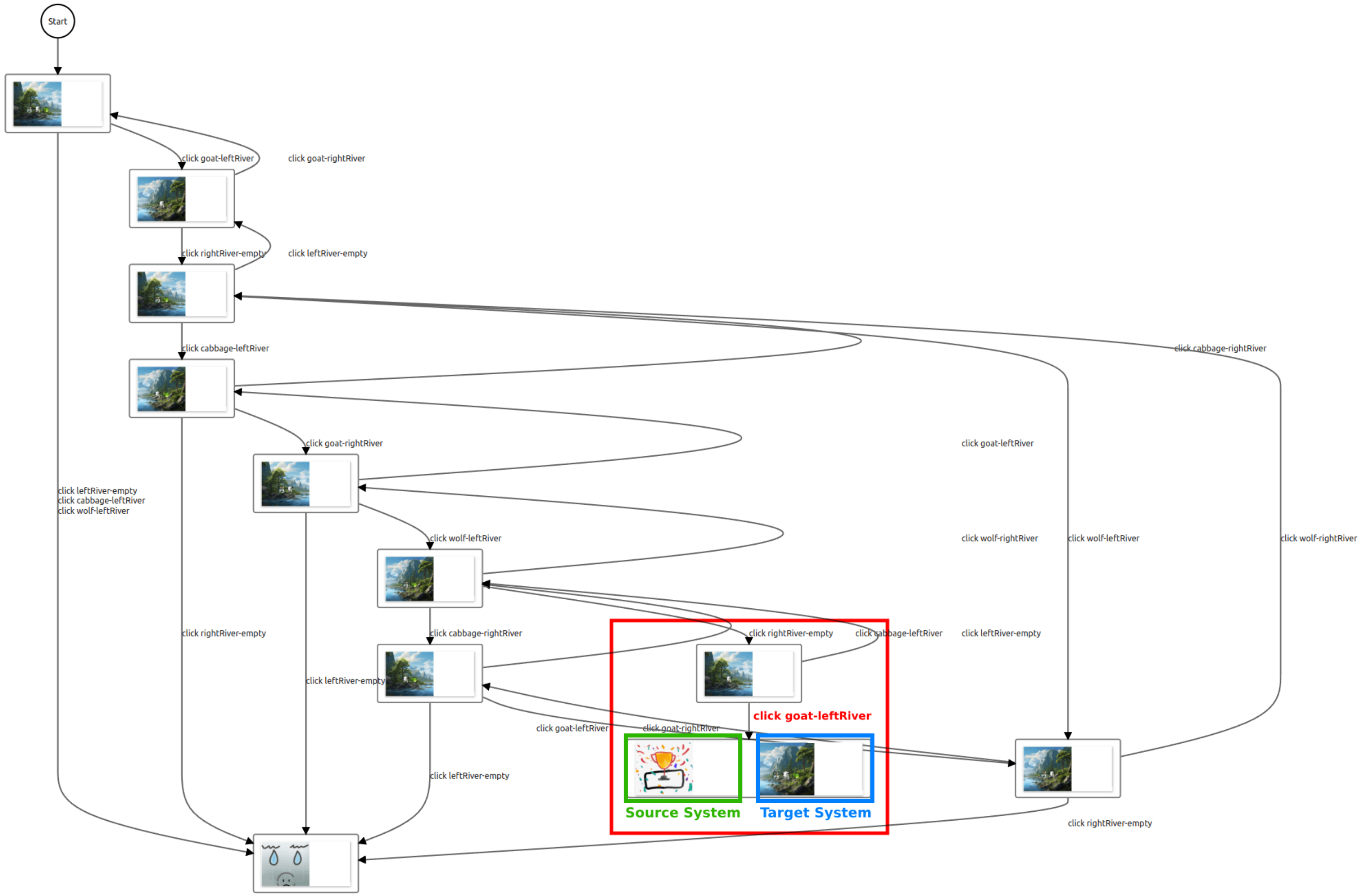


Source Language
Prompt Frame
Your task is to fill in code as part of a larger java code base.You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Embed in
Generate
Deploy
Natural Language Description
Application Code
Application
Graphical Model
Test
Infer
Automata Learning
Automaton
Generate
Embed in
Generate
Deploy
Application Code
Application
Prompt Frame
Target Language



Source Language
Prompt Frame
Your task is to fill in code as part of a larger java code base. You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Embed in
Natural Language Description
Generate
Application Code
Deploy
Application
Graphical Model
Refine
Visual Analysis
Verify
Difference Automaton
Infer
Automata Learning
Test
Test
Generate
Generate
Embed in
Prompt Frame
Application Code
Deploy
Application
Target Language

Difference Automaton

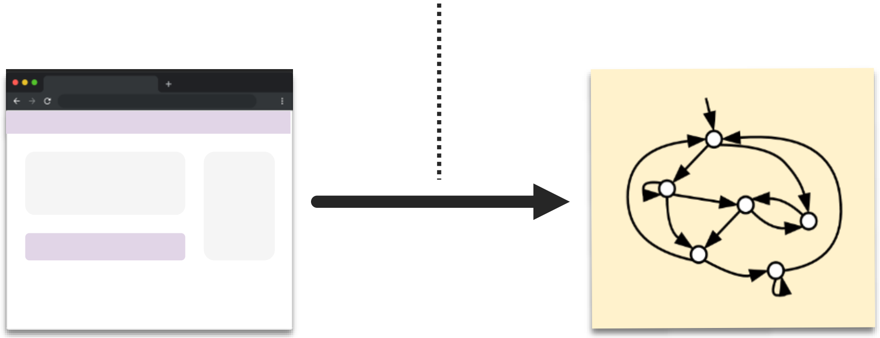


Overview

- **Brief Personal History**
  - Verification and Explanation: Concepts and Scalability
    - Random Forests
    - Deep Neural Networks
- **AI-Assisted Programming**
  - LLMs as part of Language-Driven (Softwareware) Engineering
- **Malwa: A Tool for Fully Automated Model Inference**
- **Conclusions and Perspectives**

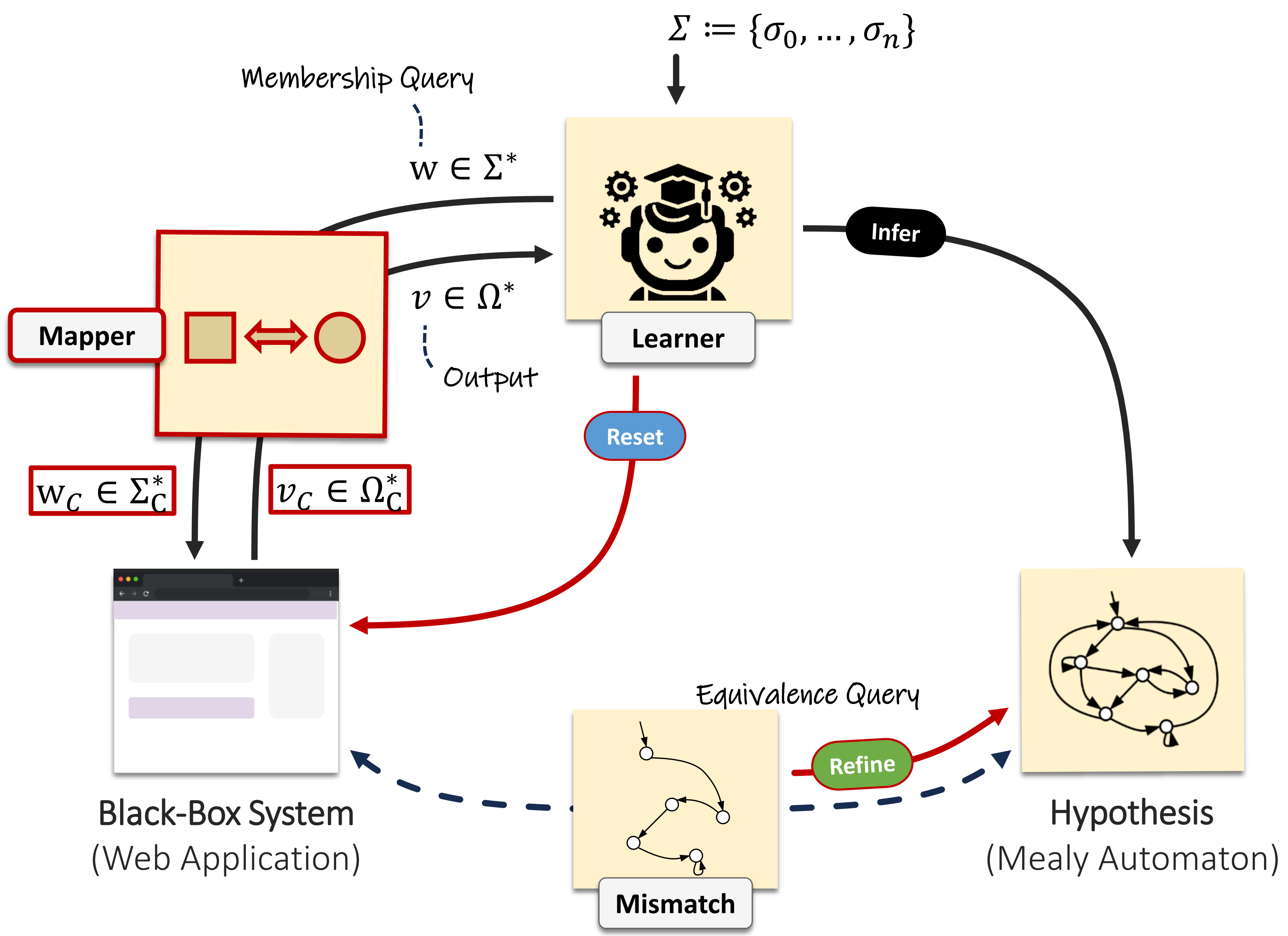
Automata Learning

Active Automata Learning

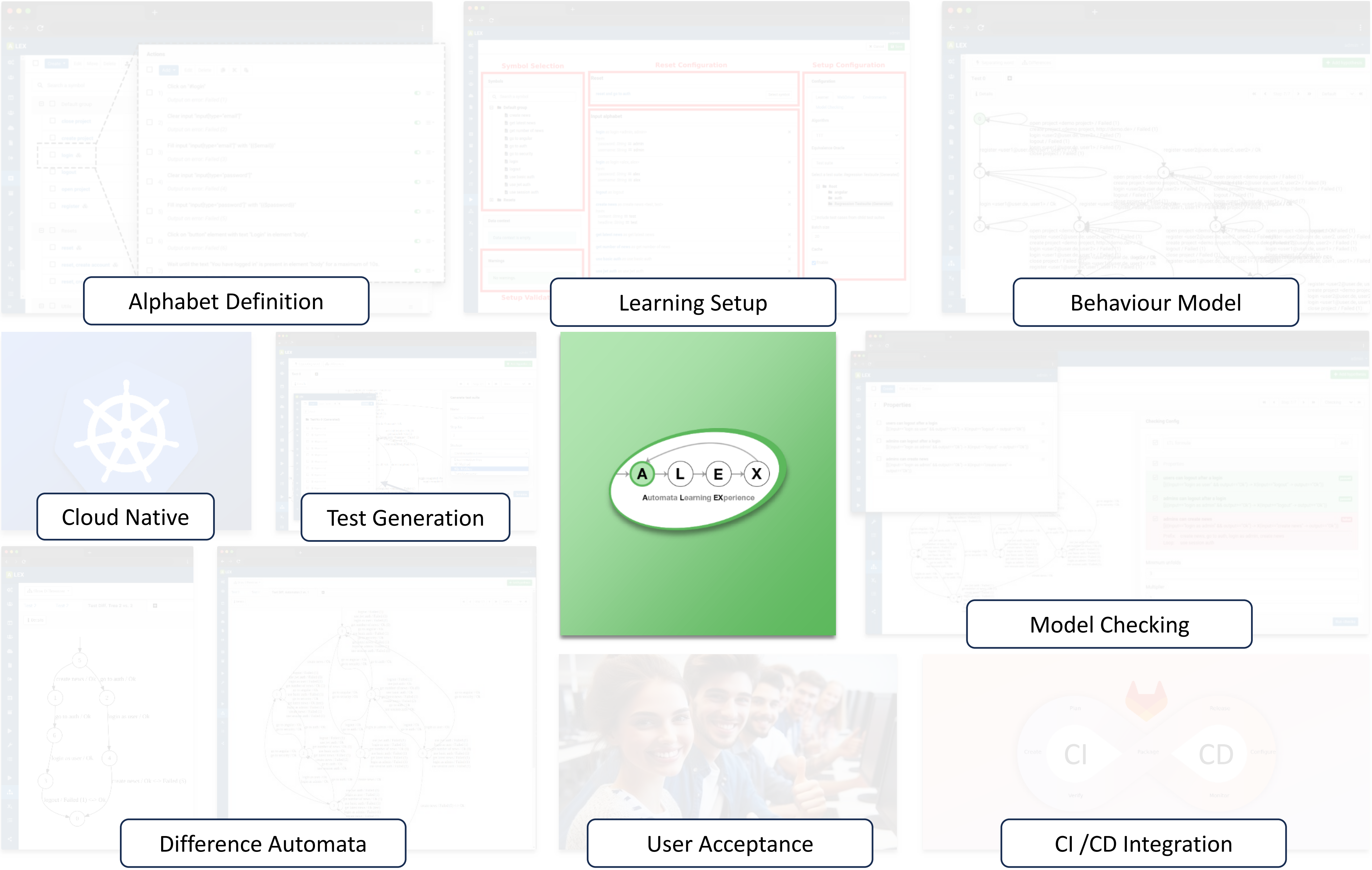


Black-Box System
(Web Application)

Model
(Mealy/Moore Automaton)



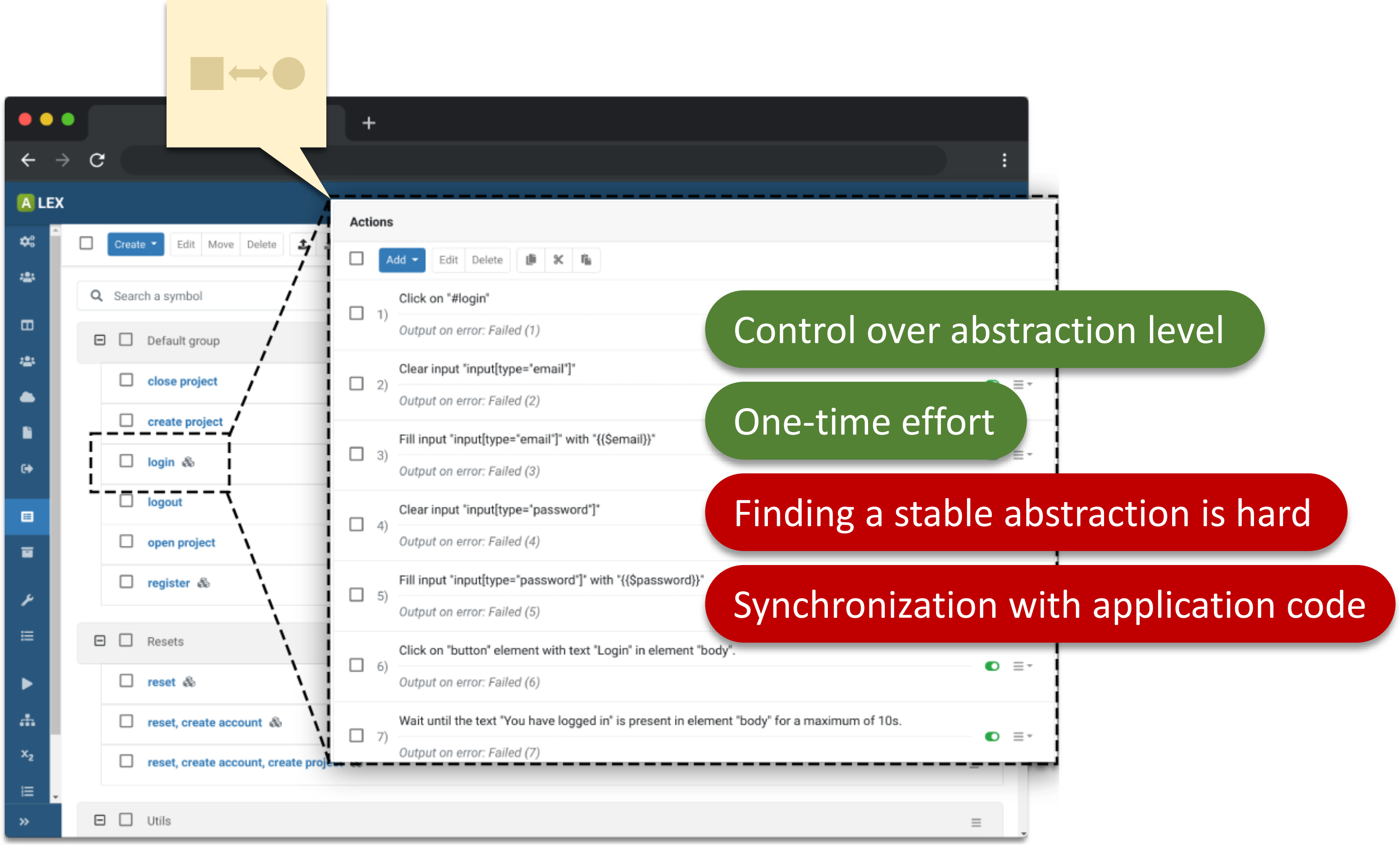
$\Sigma := \{\sigma_0, \ldots, \sigma_n\}$
Membership Query
$w \in \Sigma^*$
$v \in \Omega^*$
Output
Learner
Infer
Mapper
Reset
$w_C \in \Sigma_C^*$
$v_C \in \Omega_C^*$
Equivalence Query
Refine
Mismatch
Black-Box System
(Web Application)
Hypothesis
(Mealy Automaton)



Alphabet Definition
Learning Setup
Behaviour Model
Cloud Native
Test Generation
A L E X
Automata Learning EXperience
Model Checking
Difference Automata
User Acceptance
CI /CD Integration

Alphabet Definition

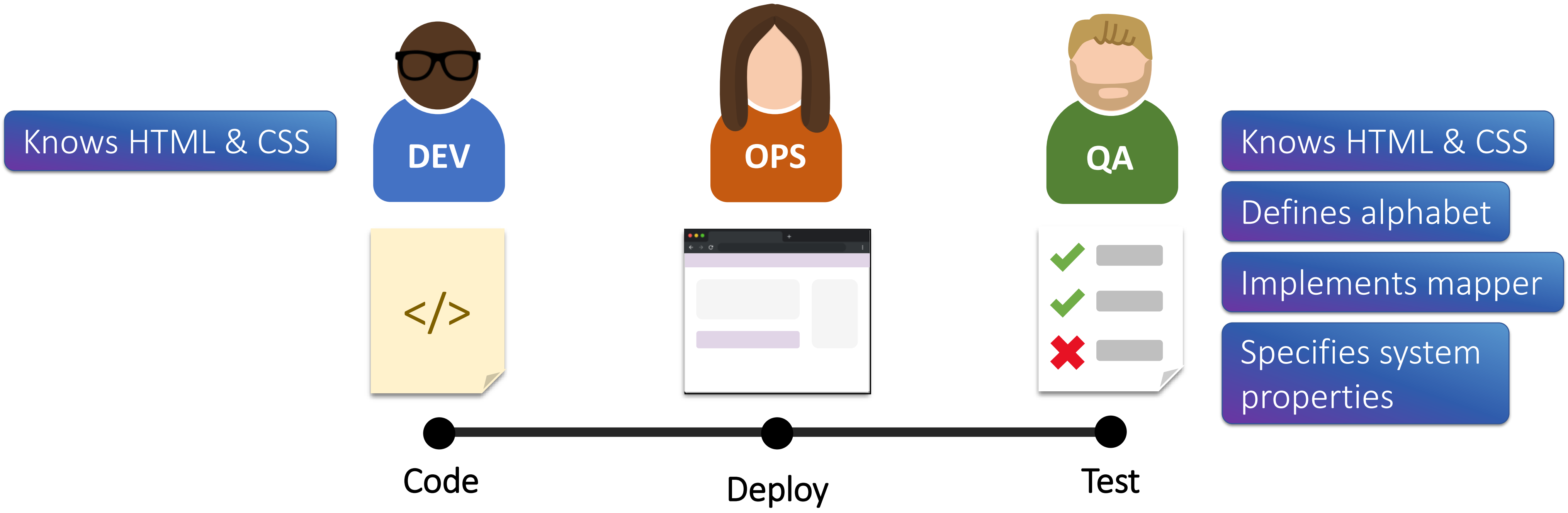
Where does the input alphabet come from?



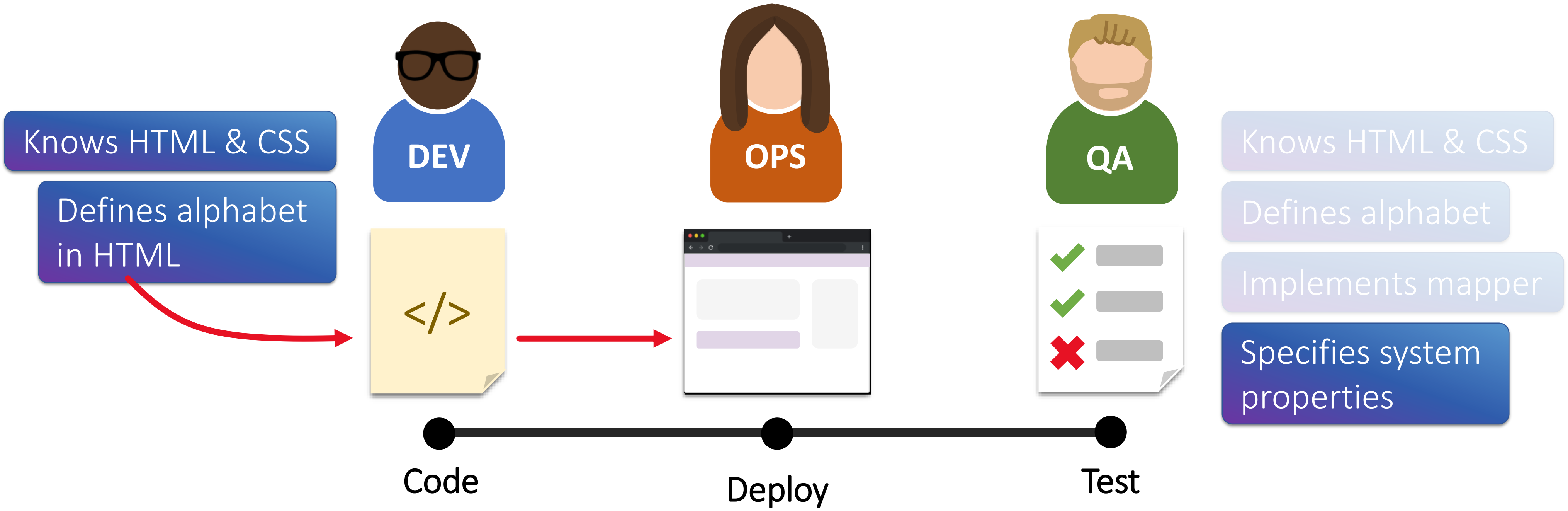
Learnability-by-Design

“Align alphabet modeling with application development.”

Status Quo


Knows HTML & CSS
DEV
OPS
QA
</>
Knows HTML & CSS
Defines alphabet
Implements mapper
Specifies system properties
Code
Deploy
Test

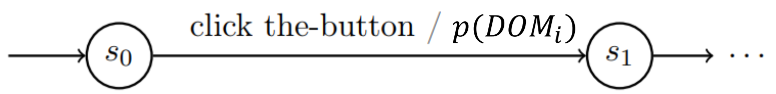
Left Shift



Knows HTML & CSS
Defines alphabet in HTML
DEV
</>
OPS
QA
Knows HTML & CSS
Defines alphabet
Implements mapper
Specifies system properties
Code
Deploy
Test

Instrumentation

```
<button data-lbd-action="Click"
        data-lbd-name="the-button">
  Click Me
</button>
```



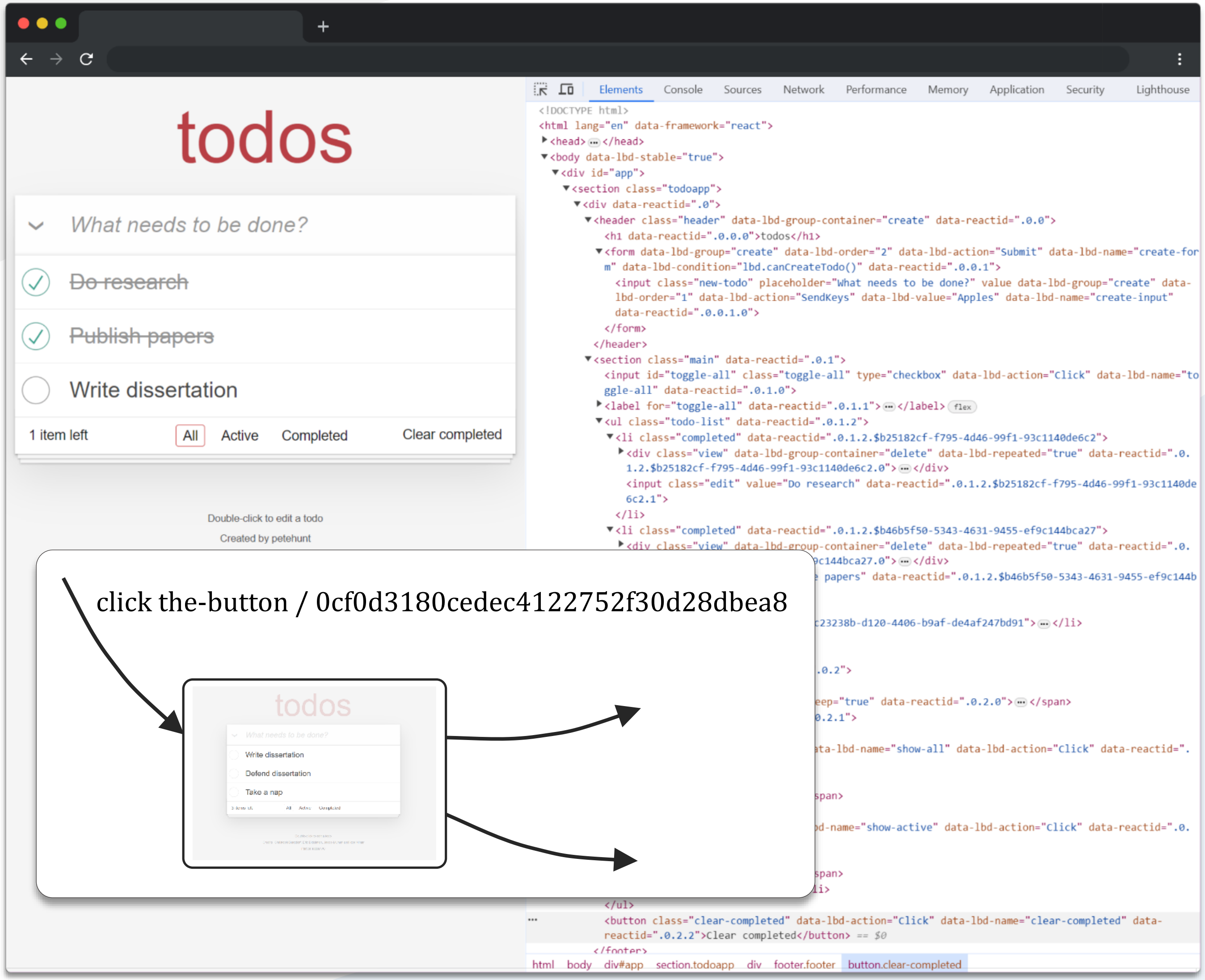
Make interaction points visible

Control semantic of input alphabets

Projected DOM represents system output

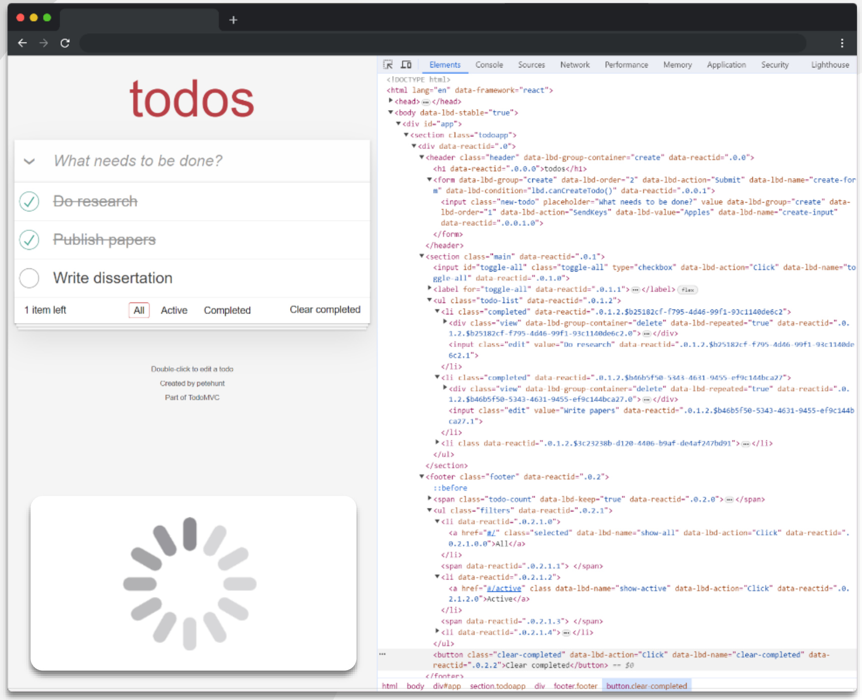
```
<span data-lbd-keep>
  Keep me, please.
</span>
```

Control DOM projection



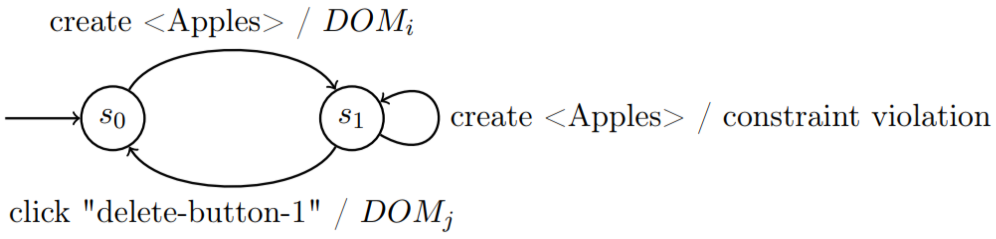
```html
<body data-lbd-stable="true">
  <!-- ... -->
</body>
```

Control quiescence

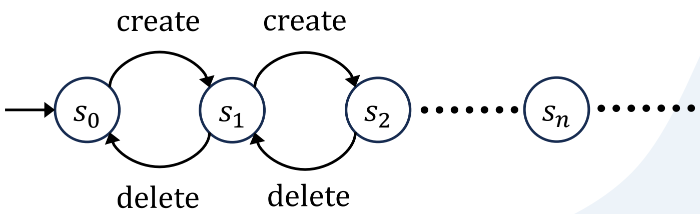


instrumented HTML

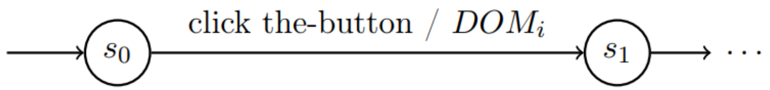
```html
<button data-lbd-condition="canExecute()"
        data-lbd-action="Click"
        data-lbd-name="the-button">
  Click me
</button>
```



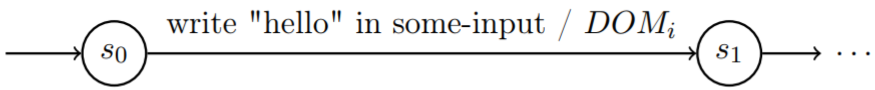
Control model approximations



```html
<button data-lbd-action="Click"
        data-lbd-name="the-button">
  Click Me
</button>
```




```html
<input data-lbd-action="SendKeys"
       data-lbd-value="hello"
       data-lbd-name="some-input">
```



Supports external datasets

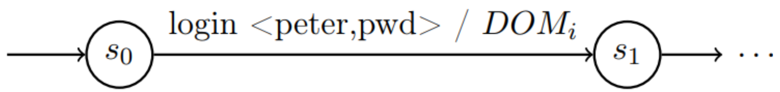
```html
<body data-lbd-stable="true">
  <!-- ... -->
</body>
```

Control quiescense

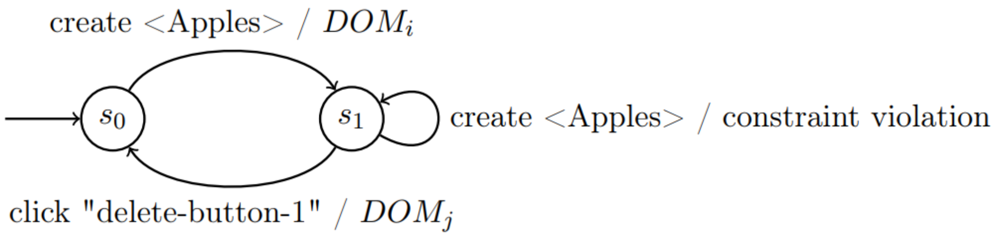
```html
<span data-lbd-keep>
  Keep me, please.
</span>
```

Control the output alphabet

```html
<form data-lbd-group-container="login">
  <input type="email"
         data-lbd-group="login"
         data-lbd-order="1" ... />
  <input type="password"
         data-lbd-group="login"
         data-lbd-order="2" ... />
  <button data-lbd-group="login"
          data-lbd-order="3" ... >
    Login
  </button>
</form>
```



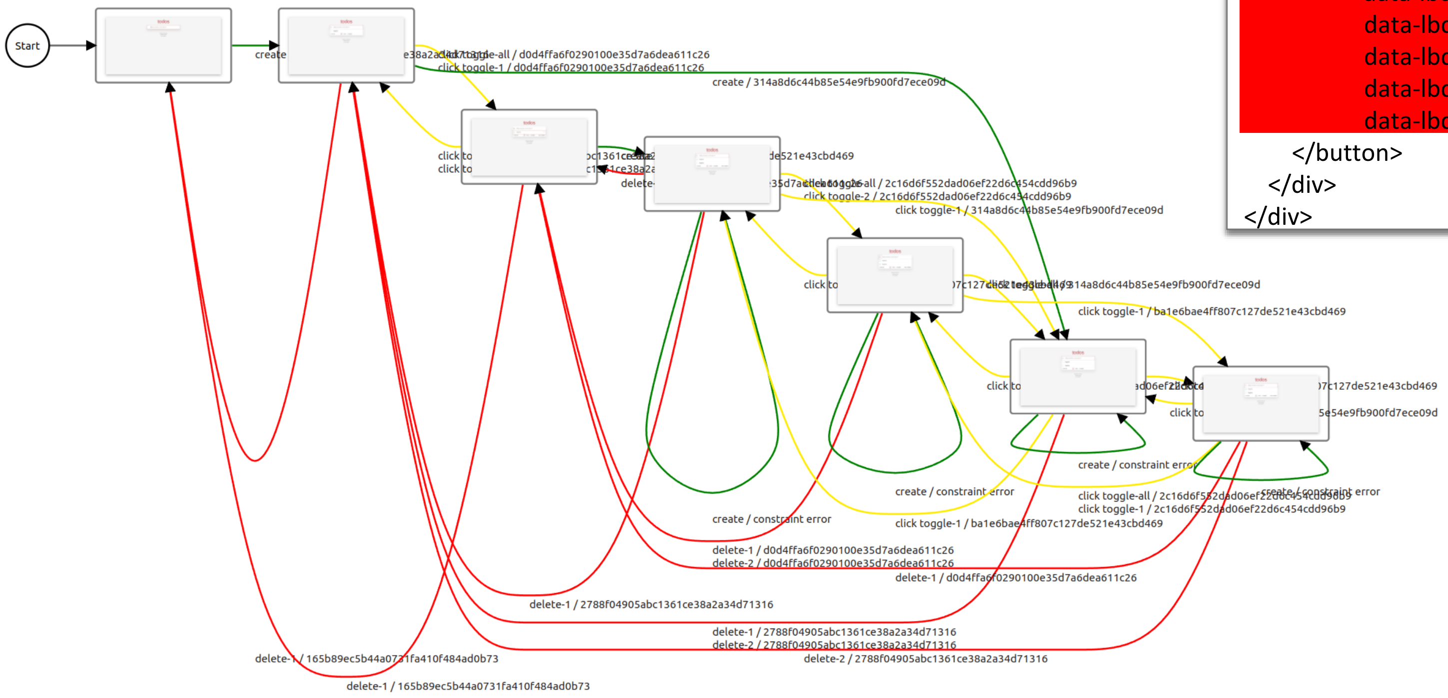

```html
<button data-lbd-condition="canExecute()"
        data-lbd-action="Click"
        data-lbd-name="the-button">
  Click me
</button>
```



Control approximations


```
<input data-lbd-action="Click"
   data-lbd-name="toggle-all">
<!-- ... -->
<div data-lbd-group-container="delete"
   data-lbd-repeated>
  <div data-lbd-name="todo"
     data-lbd-action="Hover"
     data-lbd-group="delete"
     data-lbd-order="1"
     data-lbd-repeated>
     <input type="checkbox"
         data-lbd-name="toggle"
         data-lbd-action="Click"
         data-lbd-repeated>
      <label>Apples</label>
      <button
         data-lbd-name="delete-button"
         data-lbd-action="Click"
         data-lbd-group="delete"
         data-lbd-repeated
         data-lbd-order="2">
     </button>
  </div>
</div>
```

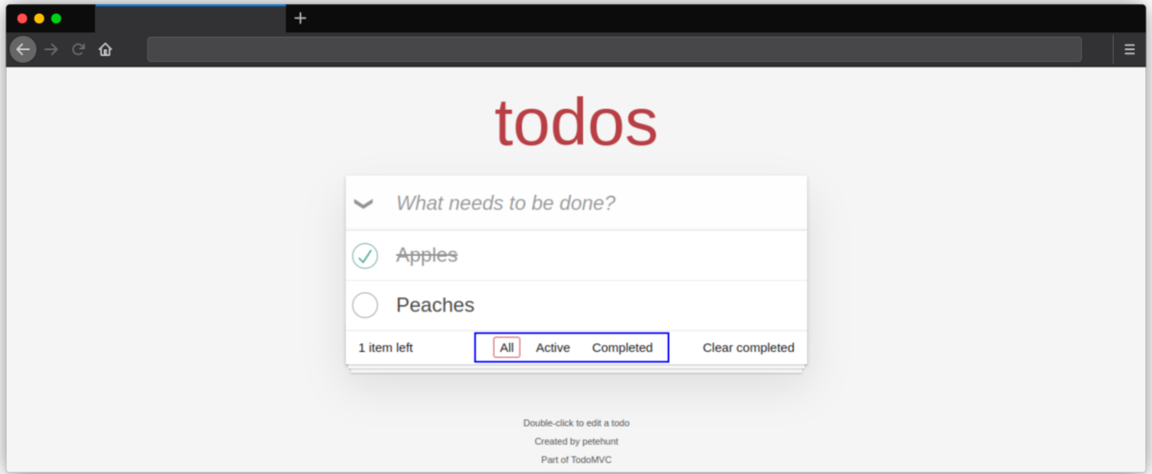
```
<header data-lbd-group-container="create">
  <h1>todos</h1>
  <form
     data-lbd-group="create"
     data-lbd-order="2"
     data-lbd-action="Submit"
     data-lbd-name="create-form"
     data-lbd-condition="lbd.canCreate()">
    <input
       class="new-todo"
       placeholder="What needs to be done?"
       data-lbd-group="create"
       data-lbd-order="1"
       data-lbd-action="SendKeys"
       data-lbd-value="Apples"
       data-lbd-name="create-input">
  </form>
</header>
```



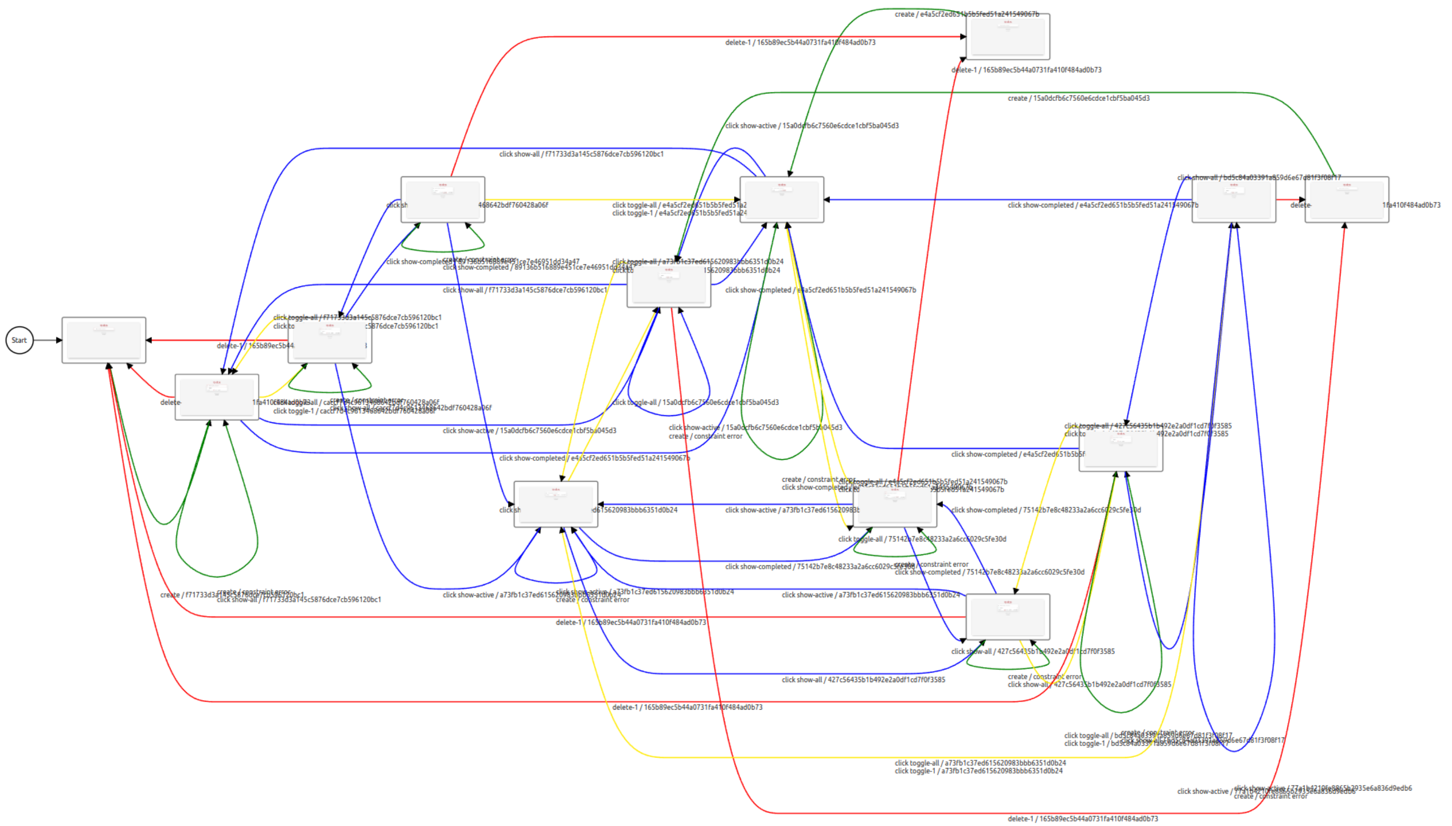
Create transition

Toggle transition

Delete transition




```
<a
   data-lbd-name="show-all"
   data-lbd-action="Click">All
</a>
<a
   data-lbd-name="show-active"
   data-lbd-action="Click">Active
</a>
<a
   data-lbd-name="show-completed"
   data-lbd-action="Click">Completed
</a>
```



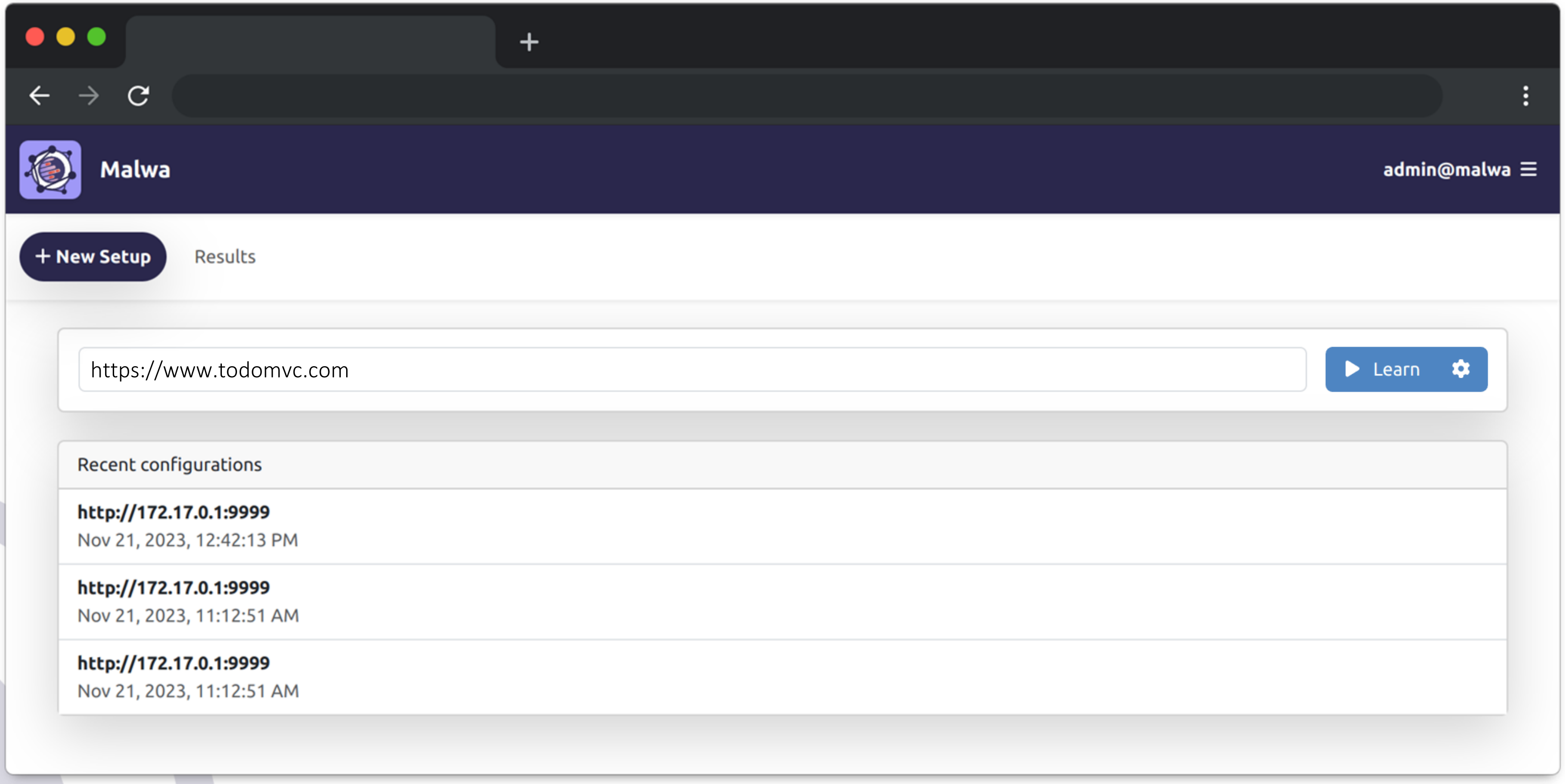
Create transition

Toggle transition

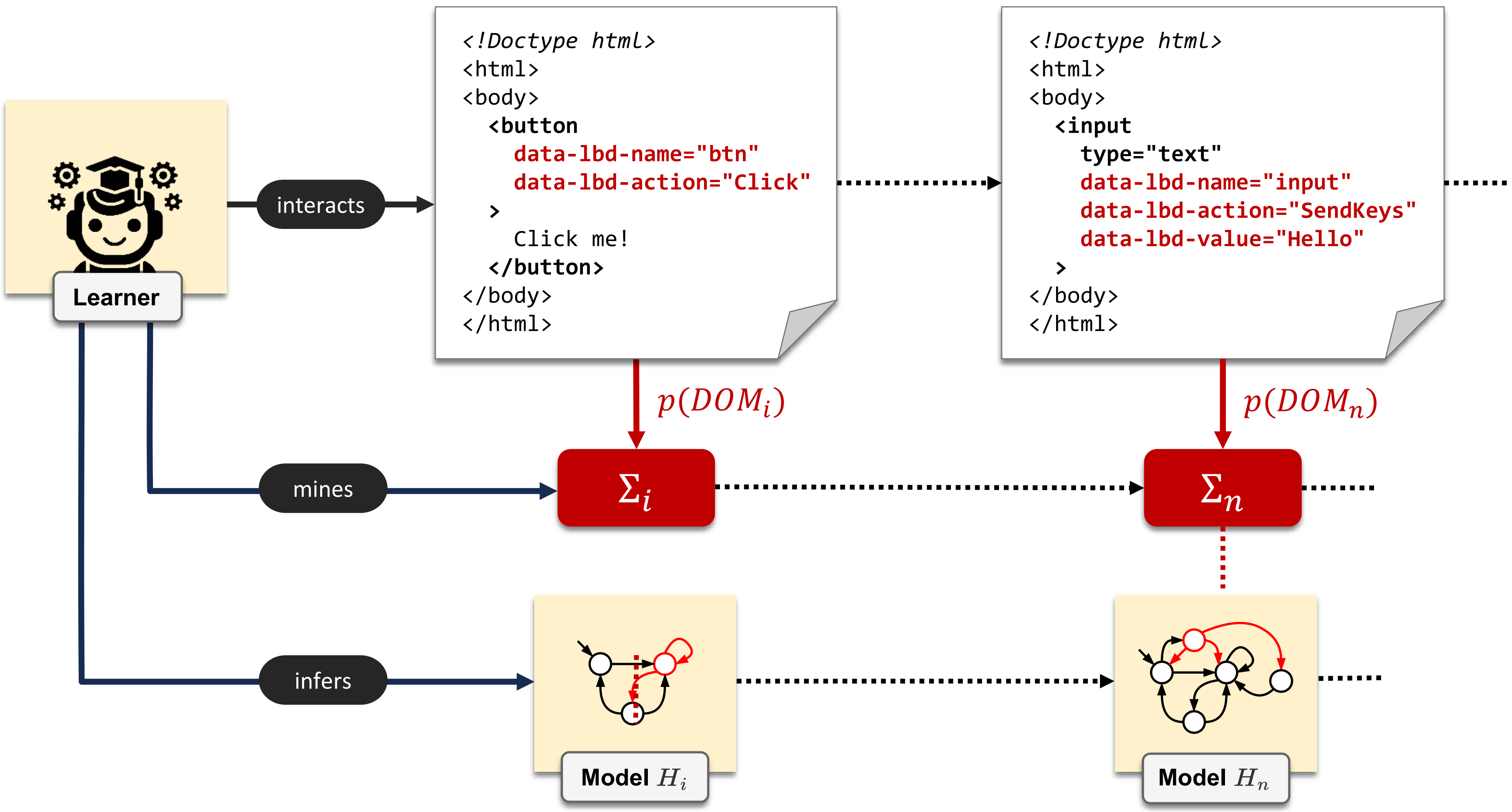
Delete transition

Filter View transition

Malwa: A Tool for Learnability by Design

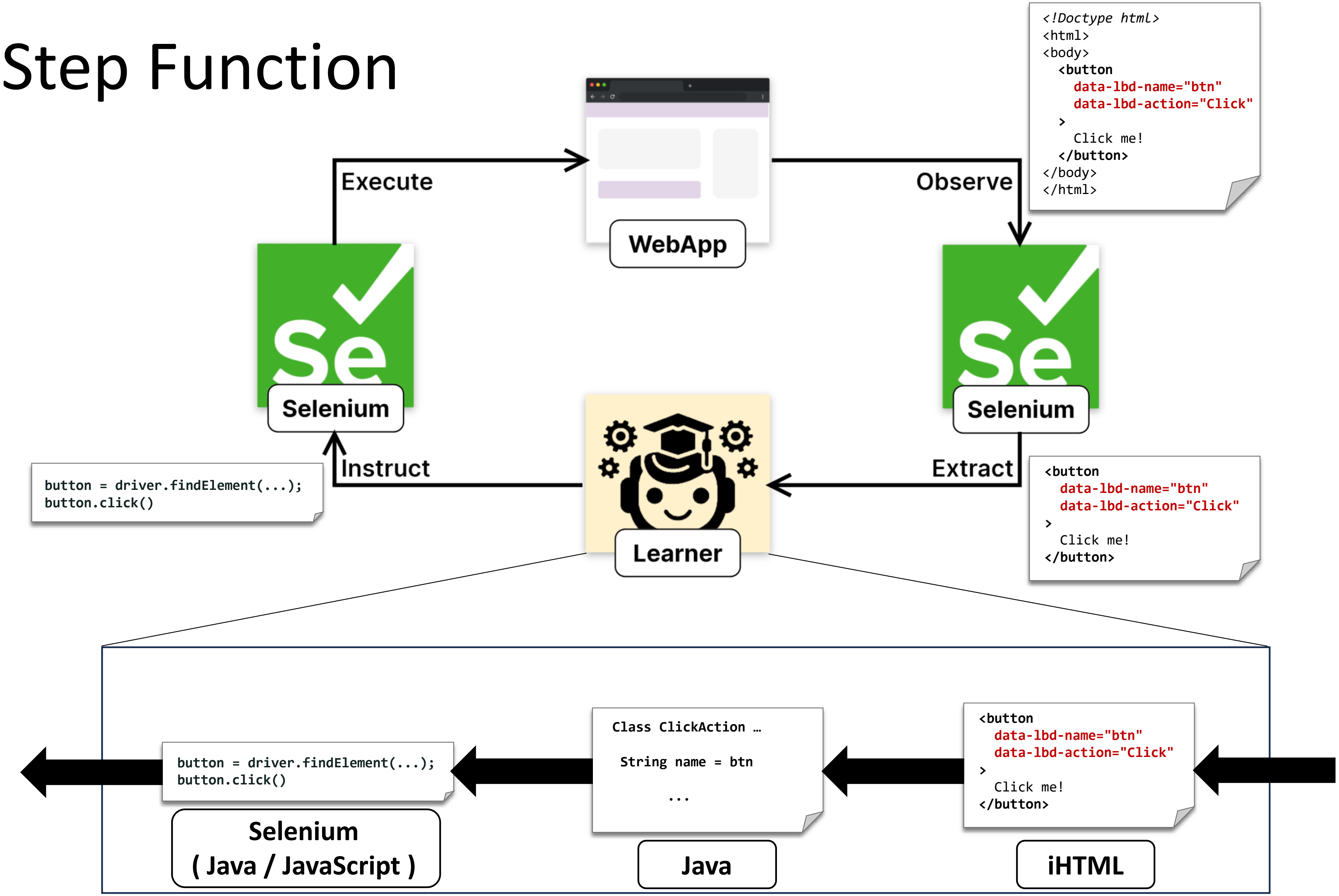


Malwa in Use

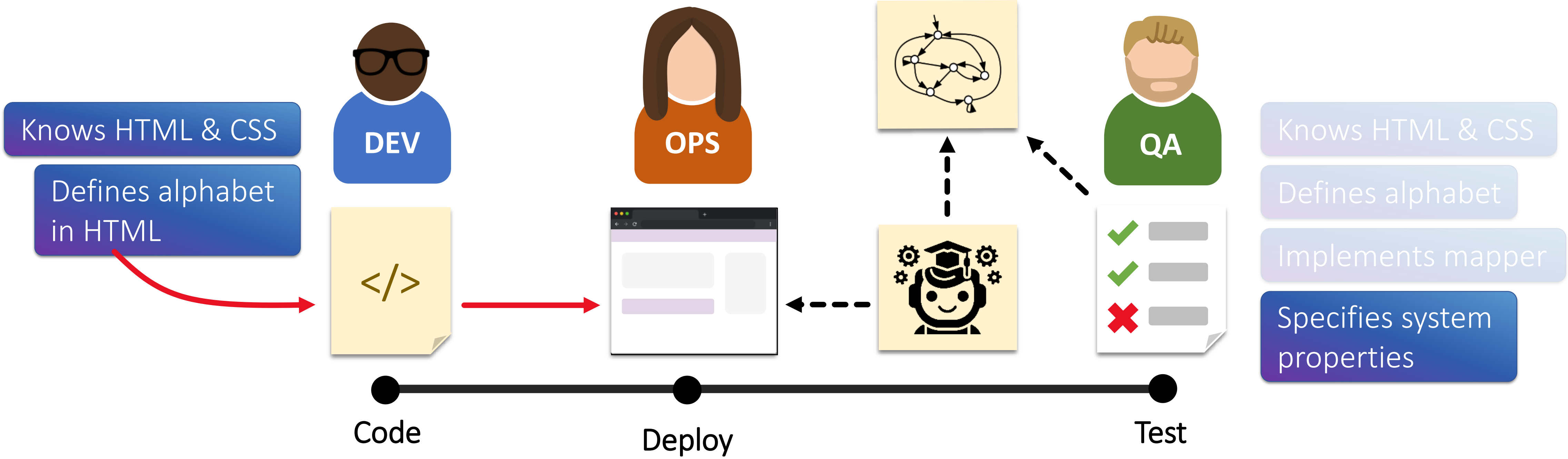


Learner
interacts
mines
infers
<!Doctype html>
<html>
<body>
<button
data-lbd-name="btn"
data-lbd-action="Click"
>
Click me!
</button>
</body>
</html>
<!Doctype html>
<html>
<body>
<input
type="text"
data-lbd-name="input"
data-lbd-action="SendKeys"
data-lbd-value="Hello"
>
</body>
</html>
$p(DOM_i)$
$p(DOM_n)$
Σ_i
Σ_n
Model H_i
Model H_n

The Step Function



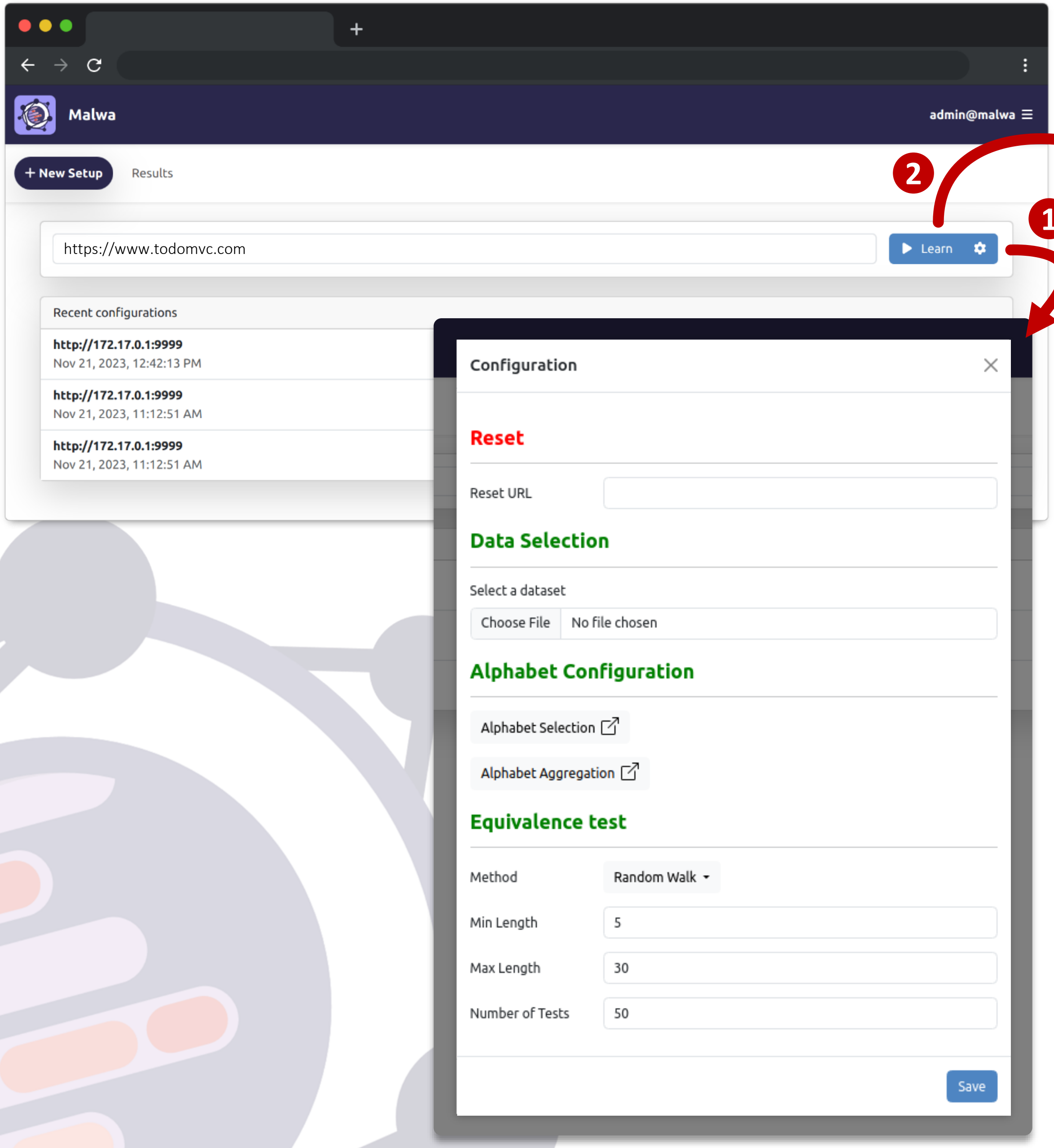
Continuous Improvement

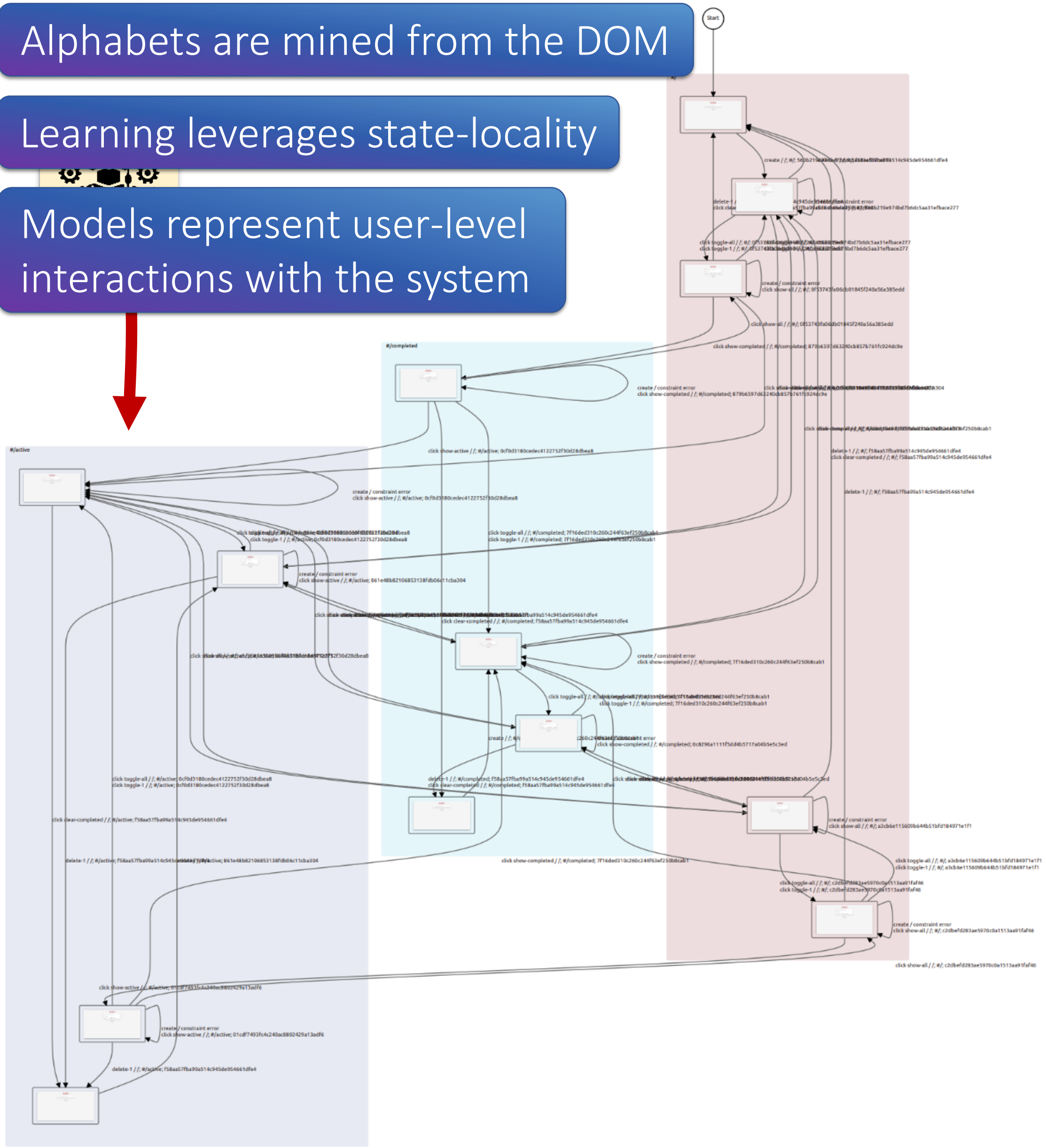


Alphabets are mined from the DOM

Learning leverages state-locality

Models represent user-level interactions with the system

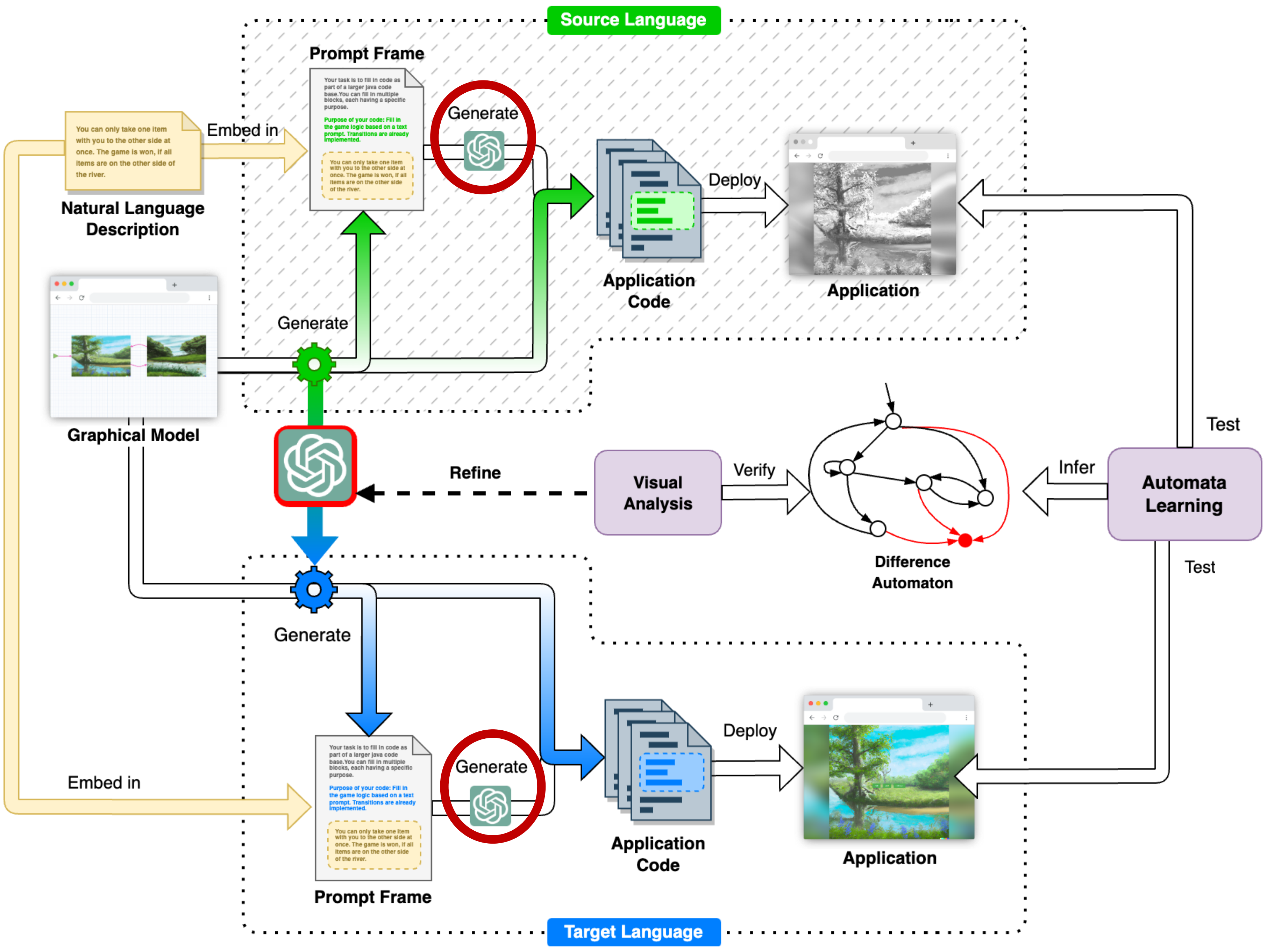




Conclusions and Perspectives

- Formal Methods are Fun and Effective
- **Decision Trees** are harmless
- **LLMs** are better considered **Partners** than **Tools**:
  - Verify their Contributions!
- We have to explore where their Strengths and Limitations are
- **Automata Learning** is an effective Control Methodology
- QA must be integral Part of Development
- **Automation** must be increased!

- **Max Tegmark:** https://youtu.be/xUNx_PxNHrY?si=mqMBbURa9QZo_yUg



Source Language
Prompt Frame
Your task is to fill in code as part of a larger java code base. You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Generate
Embed in
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Natural Language Description
Deploy
Application Code
Application
Generate
Graphical Model
Test
Refine
Visual Analysis
Verify
Infer
Automata Learning
Difference Automaton
Test
Generate
Deploy
Embed in
Your task is to fill in code as part of a larger java code base. You can fill in multiple blocks, each having a specific purpose.
Purpose of your code: Fill in the game logic based on a text prompt. Transitions are already implemented.
You can only take one item with you to the other side at once. The game is won, if all items are on the other side of the river.
Generate
Application Code
Application
Prompt Frame
Target Language

Links

Forest Gump

https://demo.forest-gump.k8s.ls-5.de/

ADD-Lib

https://gitlab.com/scce/add-lib

Our Open Sources Library (RUST): https://github.com/Conturing/affinitree

Busch et al.: ChatGPT in the Loop: A Natural Language Extension for Domain-Specific Modeling Languages

Busch, Bainczyk, and Steffen: Towards LLM-based System Migration in Language-Driven Engineering [to appear]

http://cinco.cloud/