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
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 Algebraic Decision 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 \implies 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
\land \text{ petalwidth} < 1.65
\land \text{ sepallength} ≥ 6.05
\land \text{ sepalwidth} < 2.7
\land \text{ sepalwidth} ≥ 2.55
\land \text{ petallength} < 2.7
Pre/Post Forest Verification for Free!!!

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

- Input 1:
  - petalwidth = 0.75
  - petallength = 2.45
- Input 2 ($\varepsilon = 0.1$):
  - $0.65 \leq$ petalwidth $\leq 0.85$
  - $2.35 \leq$ petallength $\leq 2.55$
- Input 3 ($\varepsilon = 0.2$):
  - $0.55 \leq$ petalwidth $\leq 0.95$
  - $2.25 \leq$ petallength $\leq 2.65$

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
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](https://github.com/Conturing/affinitree)
Visual Validation
Piece-wise Linear Functions (PWL)
PWL Activation Functions

- **Step**
- **Hard Hyperbolic Tangent**
- **ReLU**
- **Leaky ReLU**
PWL Neural Networks
AffTree Example – Image (PWL)
AffTree Example – Preimage Partition

-0.87 $0 - 0.48 $1 \leq 0.00$

0.28 $0 - 0.96 $1 \leq 0.00$

-0.83 $0 - 0.56 $1 \leq 0.00$

-0.96 $0 - 0.28 $1 \leq 1.37$

-0.66 $0 - 0.75 $1 \leq 1.15$

0.10 $0 - 0.99 $1 \leq 0.96$

0.53 $0 - 0.85 $1 \leq 1.10
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
1. Use of Domain-Specific Languages (DSLs):
   • Tailored syntax and semantics for particular application domains.
   • Enhanced expressiveness for domain experts not familiar with traditional programming 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!**
Extendible Code

Prompt Frame

Your task is to fill in code as part of the larger codebase. This code fills in the missing context, which forms a complete purpose.

Purpose of your code is to...

this goes in to be filled in a text document. Suggestions are already...
Your task is to fill in code as per the given code snippet. You will be given a prompt with a text description. Your code should be based on this text.

Purpose of your code is to fill in the given code snippet.

Prompts are already filled in. Your task is to fill in the code snippet.

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 transitions that need to be implemented. Fill out the block of code with:

```javascript
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.
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 try to use the items you have in the game to help you win. The game works if all items are on the other side of the river.
Modeling & Generation

Graphical Model

Generate

Extendible Code

Deploy

Natural Language Description

Prompt Frame

Generate

Extending Code

Application Code

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.

Your task is to fill a container with water. The container weighs 10 kg, with 10 kg of water it weighs 20 kg.

Purpose of your code is to make the game simple based on input parameters. Suggestions are welcome.
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.
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)
Black-Box System (Web Application)

Hypothesis (Mealy Automaton)

Mapper

Learner

Infer

Reset

Equivalence Query

Mismatch

$\Sigma := \{\sigma_0, ..., \sigma_n\}$

$w \in \Sigma^*$

$v \in \Omega^*$

$w_C \in \Sigma_C^*$

$v_C \in \Omega_C^*$

Membership Query

Output

$\mathcal{H}$

$\mathcal{C}$
Alphabet Definition

Control over abstraction level
One-time effort
Finding a stable abstraction is hard
Synchronization with application code

Where does the input alphabet come from?
Learnability-by-Design

“Align alphabet modeling with application development.”
Status Quo

Knows HTML & CSS

DEV

<>

Code

OPS

Deploy

QA

Knows HTML & CSS
Defines alphabet
Implements mapper
Specifies system properties

Test
Left Shift

Knows HTML & CSS
Defines alphabet in HTML

<>

Defines alphabet
Implements mapper

Knows HTML & CSS
Defines alphabet
Implements mapper
Specifies system properties

Code
Deploy
Test
Instrumentation

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

- Make interaction points visible
- Control semantic of input alphabets
- Projected DOM represents system output

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

```
click the-button / 0cf0d3180cedec4122752f30d28dbea8
```
1. `<body data-lbd-stable="true">`
2. `<!-- ... -->`
3. `</body>`
Supports external datasets

Control quiescence

Control the output alphabet

Control approximations
Create transition

Toggle transition

Delete transition
Malwa: A Tool for Learnability by Design
Malwa in Use

\[ \sum_{i} p(DOM_{i}) \]

\[ \sum_{n} p(DOM_{n}) \]

\[ \text{Model } H_{i} \]

\[ \text{Model } H_{n} \]
Continuous Improvement

- **Knows HTML & CSS**
- **Defines alphabet in HTML**
- **OPS**
- **QA**
- **Code**
- **Deploy**
- **Test**

- **Knows HTML & CSS**
- **Defines alphabet**
- **Implements mapper**
- **Specifies system properties**
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_PxNHRrY?si=mqMBbURa9QZo_yUg
Busch, Bainczyk, and Steffen: Towards LLM-based System Migration in Language-Driven Engineering [to appear]

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

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

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

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

http://cinco.cloud/