INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING
ICPP/2021/CHICAGO/USA
acm In-Cooperation
sighpc
AUGUST 9-12, 2021

ArchViMP - a Framework for Automatic Extraction of Concurrency-related Software Architectural Properties

Monireh Pourjafarian
mpourjaf@rhrk.uni-kl.de
Technical University of Kaiserslautern
Kaiserslautern, Germany
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Jasmin Jahić
jj542@cam.ac.uk
University of Cambridge
Cambridge, UK
UNIVERSITY OF CAMBRIDGE

INTRODUCTION AND MOTIVATION

Concurrent software is complex.

- There are many dependencies between threads

Why complexity is important?

- Implicit dependencies and high-coupling

Concurrency-related architectural properties:

- Data dependency
- Functionality

Problem: Comprehension of many dependencies between threads is hard.

Solution: Make understanding easier through a visual representation of analysis thread interdependencies based on the architecture principles.

Methodology and Contribution

Analyzing Communication Patterns → Defining Logical Rules → Creating Architectural Views

e.g.
- Producer-Consumer

e.g.
- Callee Function Rule

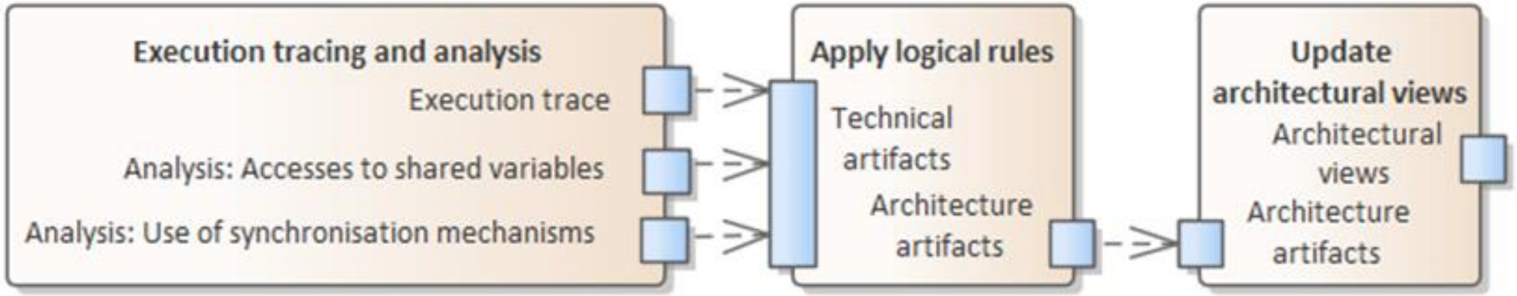
e.g.
- Logical View (EMP)

We made understanding of a concurrent software easier by providing:

- A set of abstraction rules that helps with reducing the number of elements and relations in screen
- A set of architectural views that helps the user to observe the software analysis from a different viewpoint

Methodology and Contribution

BOSMI: A Framework for Non-intrusive Monitoring and Testing of Embedded Multithreaded Software on the Logical Level



Overview of ArchViMP Architecture

Architectural Views for Multithreaded Programs (ArchviMP) is implemented as a web-based prototype
https://mpourjafarian.github.io/ArchViMP.github.io/

Logical Rules

- Technical data and components vs Logical data and components

1. The Data Structure Rule (DSR) - groups shared variables declared within the same data structure (e.g., union, struct).
2. The Entry Point Function Rule (EPFR) abstracts the functional behaviour of threads by grouping those with the same functionality.
3. The Thread Operation Rule (TOR) creates groups of shared variables based on the common operation types (Input, Output, and Process).
4. The Thread Data Access Rule (TASR) represents a combination of the EPFR and TOR rules to complete the behavioural representation of thread communication over shared variables.
5. The Callee Function Rule (CFR) abstracts logical data as a group of shared variables that have been accessed by a single or group of logical components within a callee function, by identifying function calls of the logical component and shared variable they accessed inside each callee function.
6. The Logical Decision Rule (LDR) - shared variables that are only accessed by an individual or group of logical components under a logical decision (not being a part of a logical decision but accessed after the decision is evaluated to true) are grouped as a logical data.
7. The Time Span Rule (TSR) - if threads access shared variables, they are then grouped according to the user-defined time intervals.

Architectural Views for Concurrency-related Software Properties

1. Context
2. Functional Concurrency View
   a. Functional Flow Sub-view
   b. Execution Control Flow Sub-view
3. Technical Concurrency View
   a. Logical Component Sub-view
   b. Data Structure Sub-view
   c. Data Type Sub-view
4. Timeline Concurrency View

Levels of Abstraction
starting from level 3 as the highest level to level 0 as the lowest level of abstraction.

Level 3: Defines the architectural view that illustrates direct access dependencies between group of logical components over group of shared variables (logical data).

Level 2: Describes the architectural perspectives that show how logical components interact with technical or logical data groups.

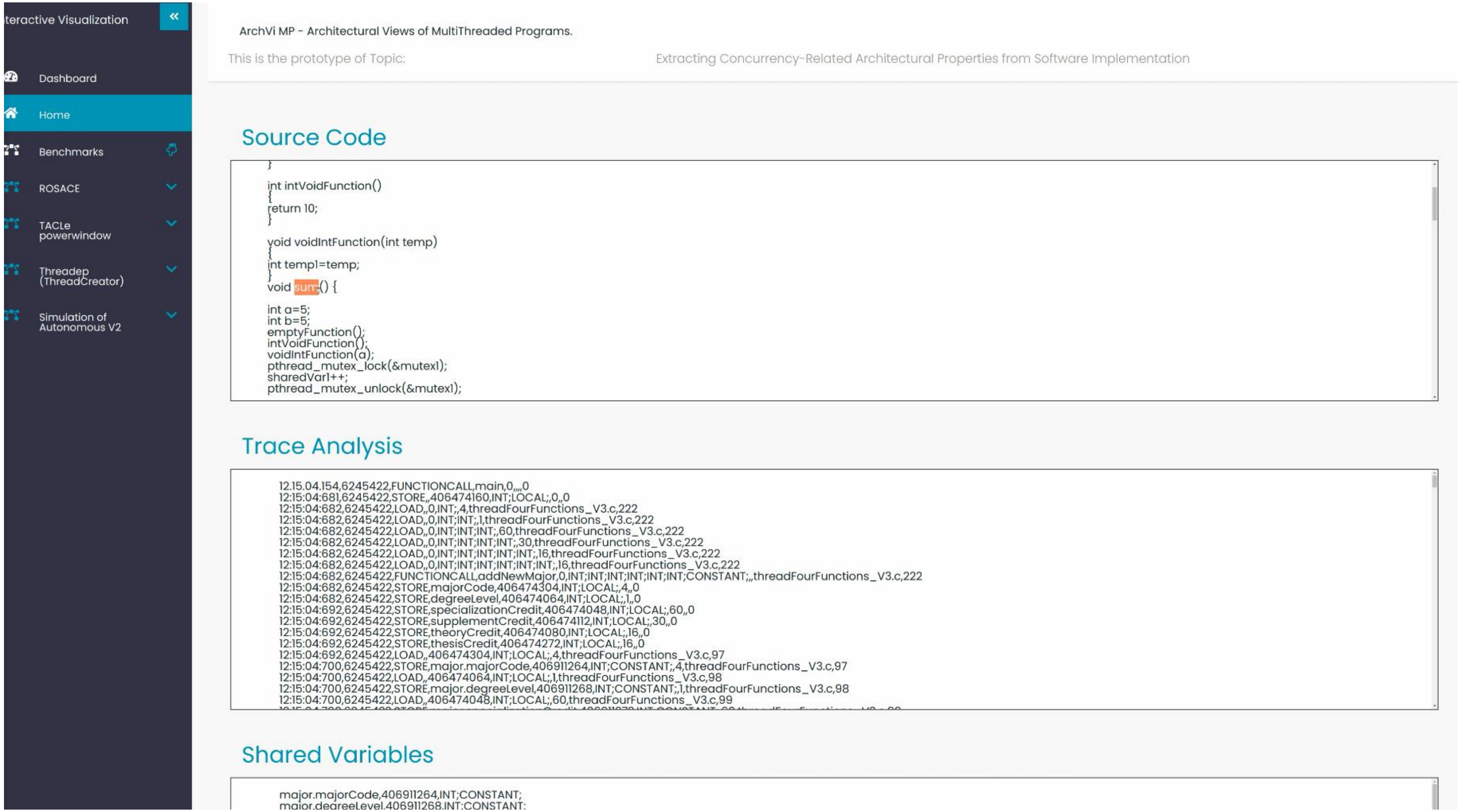
Level 1: Defines the architectural views that display the members of logical groups, either logical components or logical data.

Level 0: This level is the raw visualization of trace execution data which depicts only the technical elements of the concurrent software and raw connection between them (i.e., pure technical implementation details).

DEMO

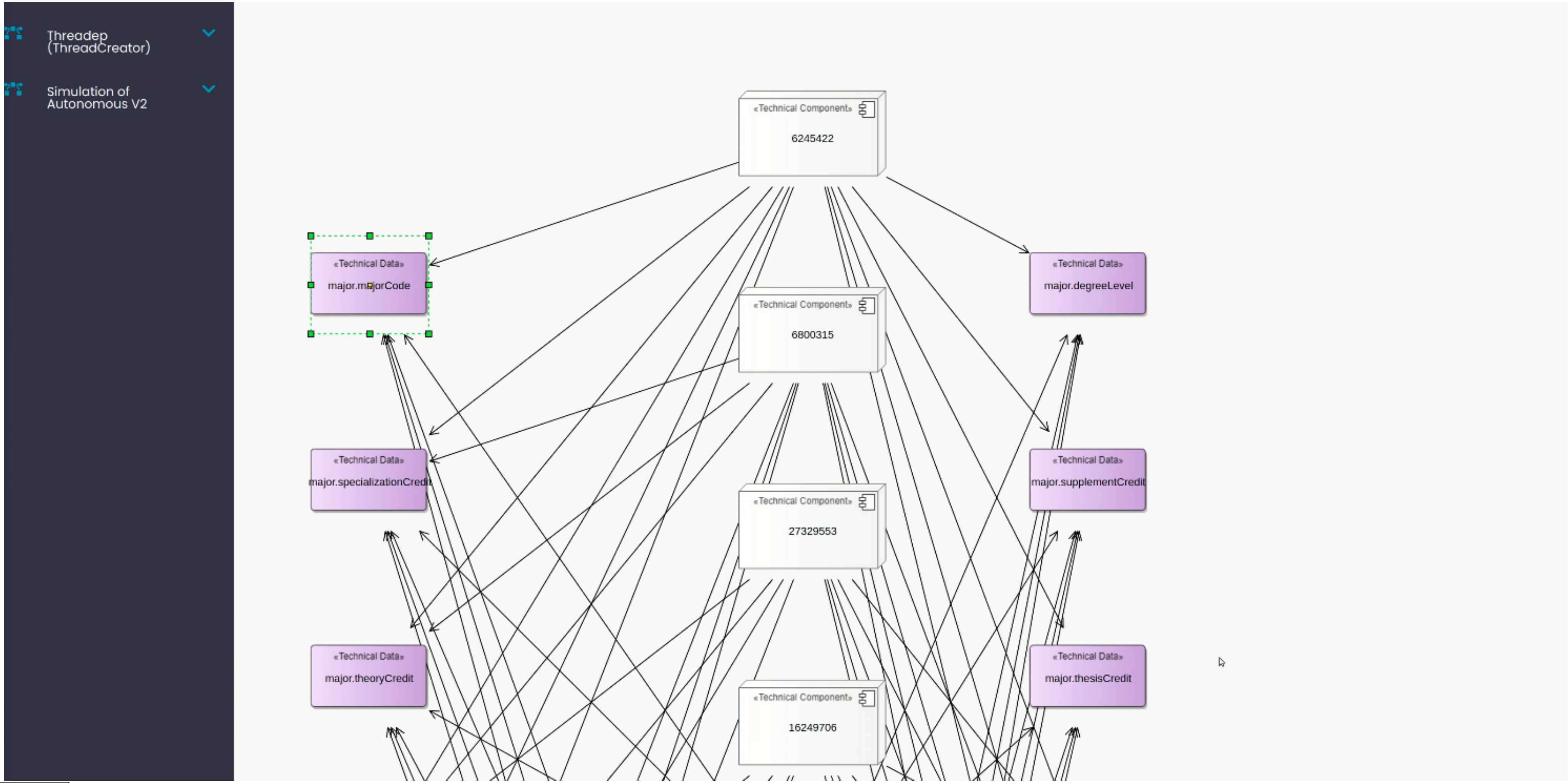
- **ArchViMP framework**

Example

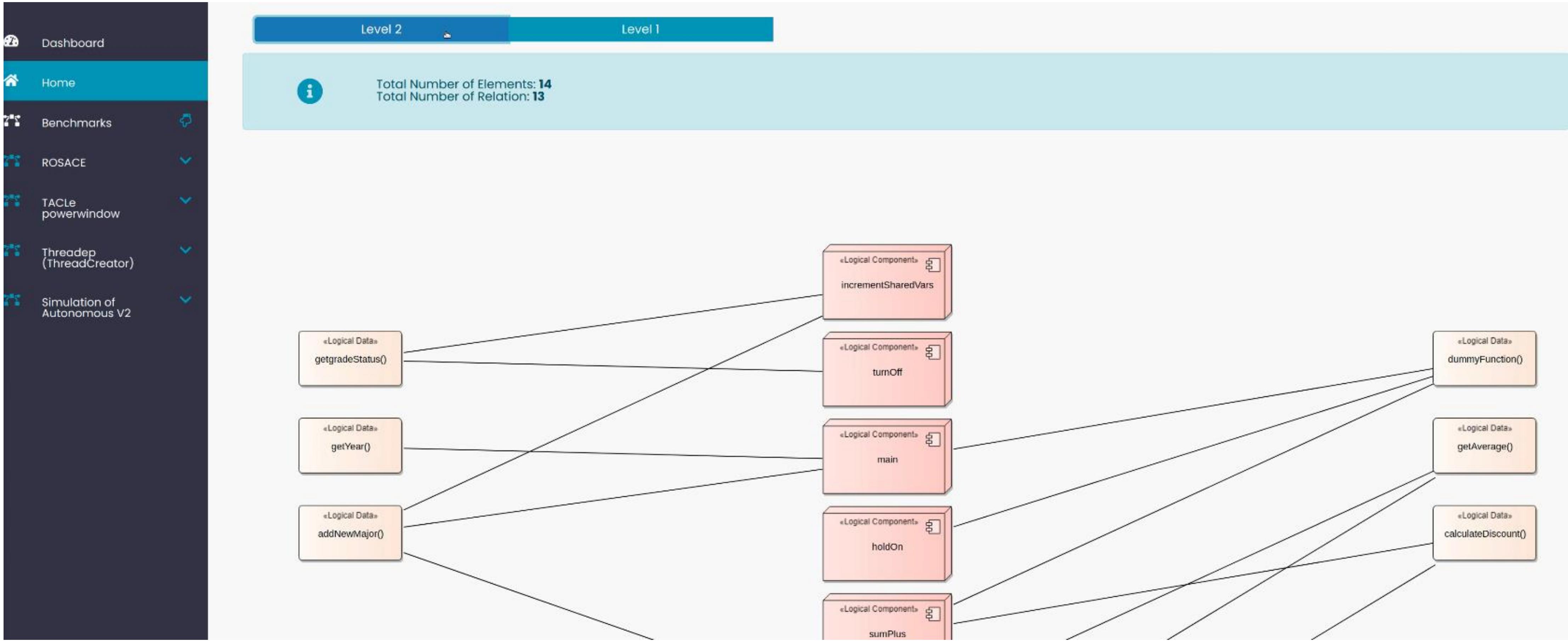


teractive Visualization
Dashboard
Home
Benchmarks
ROSACE
TACLe powerwindow
Threadep (ThreadCreator)
Simulation of Autonomous V2
ArchVi MP - Architectural Views of MultiThreaded Programs.
This is the prototype of Topic:
Extracting Concurrency-Related Architectural Properties from Software Implementation
Source Code
int intVoidFunction()
{
return 10;
}
void voidIntFunction(int temp)
{
int temp1=temp;
}
void sum() {
int a=5;
int b=5;
emptyFunction();
intVoidFunction();
voidIntFunction(a);
pthread_mutex_lock(&mutex1);
sharedVar1++;
pthread_mutex_unlock(&mutex1);
Trace Analysis
12.15.04.154,6245422,FUNCTIONCALL,main,0,,,,0
12:15:04:681,6245422,STORE,,406474160,INT;LOCAL;,0,,0
12:15:04:682,6245422,LOAD,,0,INT;,4,threadFourFunctions_V3.c,222
12:15:04:682,6245422,LOAD,,0,INT;INT;,1,threadFourFunctions_V3.c,222
12:15:04:682,6245422,LOAD,,0,INT;INT;INT;,60,threadFourFunctions_V3.c,222
12:15:04:682,6245422,LOAD,,0,INT;INT;INT;INT;,30,threadFourFunctions_V3.c,222
12:15:04:682,6245422,LOAD,,0,INT;INT;INT;INT;INT;,16,threadFourFunctions_V3.c,222
12:15:04:682,6245422,LOAD,,0,INT;INT;INT;INT;INT;INT;,16,threadFourFunctions_V3.c,222
12:15:04:682,6245422,FUNCTIONCALL,addNewMajor,0,INT;INT;INT;INT;INT;INT;CONSTANT;,,threadFourFunctions_V3.c,222
12:15:04:682,6245422,STORE,majorCode,406474304,INT;LOCAL;,4,,0
12:15:04:682,6245422,STORE,degreeLevel,406474064,INT;LOCAL;,1,,0
12:15:04:692,6245422,STORE,specializationCredit,406474048,INT;LOCAL;,60,,0
12:15:04:692,6245422,STORE,supplementCredit,406474112,INT;LOCAL;,30,,0
12:15:04:692,6245422,STORE,theoryCredit,406474080,INT;LOCAL;,16,,0
12:15:04:692,6245422,STORE,thesisCredit,406474272,INT;LOCAL;,16,,0
12:15:04:692,6245422,LOAD,,406474304,INT;LOCAL;,4,threadFourFunctions_V3.c,97
12:15:04:700,6245422,STORE,major.majorCode,406911264,INT;CONSTANT;,4,threadFourFunctions_V3.c,97
12:15:04:700,6245422,LOAD,,406474064,INT;LOCAL;,1,threadFourFunctions_V3.c,98
12:15:04:700,6245422,STORE,major.degreeLevel,406911268,INT;CONSTANT;,1,threadFourFunctions_V3.c,98
12:15:04:700,6245422,LOAD,,406474048,INT;LOCAL;,60,threadFourFunctions_V3.c,99
Shared Variables
major.majorCode,406911264,INT;CONSTANT;
major.degreeLevel,406911268,INT;CONSTANT;

Example


Threadep (ThreadCreator)
Simulation of Autonomous V2
«Technical Component»
6245422
«Technical Component»
6800315
«Technical Component»
27329553
«Technical Component»
16249706
«Technical Data»
major.majorCode
«Technical Data»
major.specializationCredit
«Technical Data»
major.theoryCredit
«Technical Data»
major.degreeLevel
«Technical Data»
major.supplementCredit
«Technical Data»
major.thesisCredit

Example


Level 2
Level 1
Dashboard
Home
Benchmarks
ROSACE
TACLe powerwindow
Threadep (ThreadCreator)
Simulation of Autonomous V2
Total Number of Elements: 14
Total Number of Relation: 13
«Logical Component» incrementSharedVars
«Logical Component» turnOff
«Logical Component» main
«Logical Component» holdOn
«Logical Component» sumPlus
«Logical Data» getgradeStatus()
«Logical Data» getYear()
«Logical Data» addNewMajor()
«Logical Data» dummyFunction()
«Logical Data» getAverage()
«Logical Data» calculateDiscount()

EVALUATION
Benchmarks

We have performed tests on several publicly available **benchmarks** that use POSIX threads.

1. ROSACE (Research Open-Source Avionics and Control Engineering)
2. TACLeBench (Timing Analysis on Code-Level) – powerwindow

Two self-developed Benchmarks

1. Threadep
2. SimulationofAutomotive

https://github.com/mahsa-poorjafari/SA_SimulationofAutomotive

EVALUATION
Assessment Criteria

- Qualitative Measure:
  - Adequacy of the abstraction level

- Quantitative Measure:
  - Percentage of the reduction in elements and relations achieved with ArchViMP

$$\boxed{\begin{array}{l} Element(El)/Relation(Re)\ Reduction\ Percentage = \\ \left(1 - \dfrac{number\ of\ El/Re\ in\ an\ abstract\ view}{number\ of\ El/Re\ in\ the\ raw\ representation}\right) * 100 \end{array}}$$

Results
Evaluation of benchmarks in architectural views

- Element Reduction Percentage (ERP)
- Relation Reduction Percentage (RRP)
- ROSACE (R)
- Threadep (TE)
- TACLeBench - powerwindow (TB)
- SimulationOfAutomotive (SA)
- No view (-)
- Logical Context (LC)
- Functional Flow Sub-view (FFSV)
- Execution Control Flow Sub-view (ECFSV)

Evaluation Parameter	R	TE	TB	SA
ERP - LC at level 3	-	-	45%	-
RRP - LC at level 3	-	-	60%	-
ERP - FFSV at level 2	50%	51.7%	63.6%	25%
RRP - FFSV at level 2	83.3%	88%	-	71%
ERP - ECFSV at level 2	60%	58.6%	-	79%
RRP - ECFSV at level 2	92%	92%	-	92%

Conclusion and Future work

- automatic visualization of the concurrency related behavior within multiple concurrent architectural views and
- reduction of the number of elements shown on a single screen, by using different viewpoints and abstraction levels.

- Further refinement and combination of logical rules to:

    - Cover more corner cases

    - Cover more dependencies

Thank you

Monireh Pourjafarian	Jasmin Jahić
mpourjaf@rhrk.uni-kl.de	jj542@cam.ac.uk
Technical University of Kaiserslautern	University of Cambridge
Kaiserslautern, Germany	Cambridge, UK