Decentralized Runtime Verification

Building Blocks: Components, Observations, Specification and Monitors [5,4]

A decentralized system contains multiple components. Components behavior is abstracted as observations.

The observation \( G(\text{alarm}) \) means, true indicates that the alarm is triggered.

A specification is a user-provided formal description of the correct behavior of the system. This is used to synthesize and integrate monitors into the system.

Specifications can be defined using automata, Linear-time Temporal Logic (LTL), or other formalisms.

Monitors are responsible for checking whether the current execution of the system complies with the specification. One or more monitors are attached to components. Monitors receive observations, do some processing and communicate with other monitors.

Monitoring API

An API for Common Monitoring Activities

- Parsing and managing specifications and traces.
- Datastructures for storing observations and monitor state.
- Creating, accessing and Instrumenting measures into the execution.
- Deploying and setting up components, monitors, and associating monitors to specifications.
- Specifying high level API for monitoring and communication between monitors.

Simple and Extensible Formats

THEMIS uses XML for specifications. Specifications are passed to your algorithm. Your algorithm is responsible of parsing and setting up the monitors appropriately.

1. Design

Decentralized RV Algorithms

Design new algorithms

Variants of existing algorithms

Refinements of existing algorithms

Measures

Create measures by instrumentation (AspectJ)

Use existing API for measures

Measures target all algorithms using the API

2. Execute

Simulation Monitor a trace using an algorithm

Visualization

Basic topology and communication visualization

Experiment

An experiment is a reproducible set of parameters, specifications, and algorithms

3. Analyze

Flexible Measures are stored in a database

Use any third party tools for analysis

Modular Instrumented at runtime using AspectJ

Existing API and classes to extend

Reusable Measures apply to different algorithms

Experiments can re-use new measures

THEMIS

A Tool for Decentralized Monitoring

Antoine Hokayem    Yliès Falcone

first.last@univ-grenoble-alpes.fr

Univ. Grenoble Alpes, LIG, Inria, CNRS, F-38000 Grenoble, France

References


Use Cases

Designing New Algorithms

THEMIS makes it easy to prototype and incrementally design new algorithms. Common tasks such as paving automata and LTL, setting up monitors, and communication are managed by the framework.

Optimizing Existing Algorithms

Using the experiment tool and the existing measures, designing new variants of algorithms can be easily return in a reproducible environment. New measures can be added to enrich the comparison, which will also apply to the older versions.

Comparing Decentralized RV Algorithms

The monitoring API can be used to compare different algorithms [1]. This is done by analyzing the usage of the same datastructures or API calls (such as communication).

THEMIS

Designing, Analyzing, and Comparing Decentralized RV Algorithms

THEMIS is a tool to facilitate the design, development, and analysis of decentralized monitoring algorithms. It is developed using Java and AspectJ.

It consists of a library and command-line tools. THEMIS provides an API, data structures and measures for decentralized monitoring.

These building blocks can be reused or extended to modify existing algorithms, design new more intricate algorithms, and elaborate new approaches to assess existing algorithms.

The theoretical aspects can be found in [3].