An Extensible Invariant Exchange Format for \textsc{CPAchecker}

Nico Weise
Joint work with Dirk Beyer

LMU Munich, Germany
Motivation

- Cooperating actors
  - use different algorithms,
  - use different formalisms,
  - are written in different languages,
  - ...

Writing custom exchange formats for every use case is not practical!
Current Solution: Witnesses Automata

1 int x;
2 int y = x;
3 while (x < 100) {
4     x++;
5     y++;
6 }
7
8 assert (x == y);
Current Solution: Witnesses Automata

<node id="N2">
  <data key="invariant">( x == y )</data>
  <data key="invariant.scope">main</data>
</node>
<edge source="N1" target="N2">
  <data key="enterLoopHead">true</data>
  <data key="startline">2</data>
  <data key="endline">2</data>
</edge>
Problems with the Current Format

- Mapping to CFA-locations requires an analysis run.
- GraphML is not suitable for arbitrary data.
- Hard to interpret for humans.
New Artifact Exchange Format

- Structured with YAML.
- Files contain an array of entries.
- *Entry-type* determines schema of the entry.
Example: Loop Invariants

- entry_type: loop_invariant
  metadata:
    # ...
  location:
    # ...
  loop_invariant:
    # ...
Example: Loop Invariants (metadata)

```plaintext
format_version: 0.1
uuid: 91023a0f-9f45-4385-88c4-1152ade45537
creation_time: 2021-05-05T15:18:43+02:00
producer:
    name: CPAchecker
    version: 2.0.1-svn
    configuration: svcomp21−−04−kInduction
description: ...
command_line: ...
task:
    input_files:
        - multivar_1-1.c
input_file_hashes:
    multivar_1-1.c: 511f45a...
specification: CHECK( ... )
data_model: ILP32
language: C
```
Example: Loop Invariants (location)

file_name: multivar_1-1.c
file_hash: 511f45a...
line: 22
column: 0
function: main
Example: Loop Invariants (loop_invariant)

string: \((x \geq 1024U) \land (x \leq 4294967295U) \land (y == x)\)
type: assertion
format: C
Implementation In CPAchecker: Export

- InvariantExportAlgorithm (uses InvariantGenerators):
  1. Poll invariants (i.e. C-Expression and CFANode) from generator.
  2. Compute FileLocations of all StatementEdges leaving the node.
  3. Write Expression to disk for each FileLocation.
Implementation In CPAchecker: Import

- New package (implements InvariantGenerator):
  1. Load invariants from disk when requested.
  2. Get CFANodes from file location (heuristically).
  3. Parse C-Expression (heuristic declaration resolution).
  4. Return through InvariantSupplier.
Applications

- Human-Supported Verification
  - e.g. CPAchecker can continuously read invariants from disk.

- Cooperation
  1. tool1 produces sv-correctness-witnesses,
  2. CPAchecker converts them to new format,
  3. tool2 consumes the invariants by writing assertions into the program files.
Applications

▶ Invariant Store
▶ Invariant generators put invariants into the store,
▶ validators certify correctness of the invariants,
▶ verifiers use (certified) invariants to prove correctness.
Example: Loop Invariant Certificates

- `entry_type: loop_invariant_certificate`

  `metadata:`
  
  # ...

  `target:`
  
  # ...

  `certification:`
  
  # ...
Example: Loop Invariant Certificates (metadata)

```
format_version: 0.1
uuid: 91023a0f-9f45-4385-88c4-1152ade45537
creation_time: 2021-05-05T15:18:43+02:00
producer:
  name: CPAchecker
  version: 2.0.1-svn
  configuration: svcomp21--04-kInduction
  description: ...
  command_line: ...
```
Example: Loop Invariant Certificates (target)

\begin{itemize}
  \item \texttt{uuid}: 91023a0f-9f45-4385-88c4-1152ade45537
  \item \texttt{type}: loop-invariant
  \item \texttt{file_hash}: 622g56b...
\end{itemize}
Example: Loop Invariant Certificates (certification)

string: confirmed
type: verdict
format: confirmed | rejected
Applications

- Cooperation for Trust
  - Intuitively: If several tools confirm a result, it is more trustworthy.
  - Idea: Increase trust by only using trustworthy intermediate results, i.e. certified invariants.

![Diagram of trust cooperation]
New format for exchanging artifacts (currently only loop invariants):

- human-readable,
- easy to parse, and
- extandable.

Implemented in CPAchecker.

Description on GitHub.

Next steps: provide a linter, extend the description, introduce format to SV-COMP.