Causal-consistent reversible debugging for Erlang

Ivan Lanese, University of Bologna/INRIA

Thanks to Pietro Lami, German Vidal and many others

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 778233

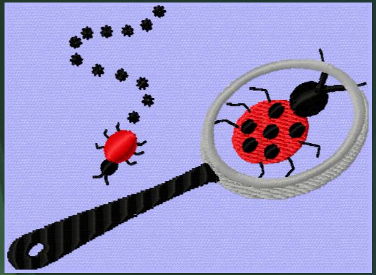
BEHAPI Behavioural Application Program Interfaces

Reversible computing

The possibility of executing a computation both in the standard, forward direction, and in the backward direction, going back to a past state

- In some areas systems are naturally reversible: biology, quantum computing, …
- In other areas making systems reversible can be useful: robotics, ***debugging***, reliability, ...

Reversibility for debugging



- Debugging amounts to find the wrong line of code (bug) causing a visible misbehavior
- The bug precedes and causes the misbehavior
- Quite natural to use reversibility to go back from the misbehavior to the bug
- Sequential reversible debugging is well understood
  - Gdb (since 2009), Microsoft time-travel debugger, ...

Debugging concurrent programs

- Concurrent reversible debugging not so developed
  - Most approaches just linearize the execution
  - Like a recorded movie, where you can go back and forward
  - Causal information is lost
- Can we exploit causal information?

Debugging and causality

- Standard debugging procedure:
  1) Observe an unexpected behavior
  2) Find in the code the instruction that caused it
  3) Correct the instruction
- Causal information can be used to drive step 2 above
- Debugging strategy: follow causality links backwards from the misbehavior to the bug

CauDEr

- Causal-consistent Debugger for Erlang
- Allows one to debug concurrent Erlang programs taking advantage of causality information
- Only an academic prototype
  - Supports a limited fragment of Erlang
  - Efficiency has never been considered
- … but can show what the approach can do on selected small examples

A simple example

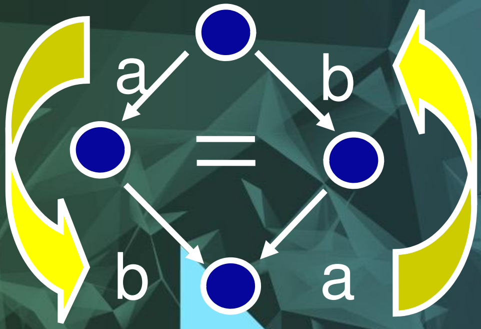
- A server allowing one to invoke both stateless and stateful mathematical services
- Services spawned on the first request
- All served requests are logged

Sequentiality vs concurrency

- Reversibility in a sequential setting:
  - recursively undo the last action
- In concurrent programs there is no uniquely defined last action
  - Which actions can be undone?
- We follow causal-consistent reversibility

Causal-consistent reversibility

- Causal dependencies must be respected
- First reverse the consequences, then the causes
- Independent actions are reversed independently



Causal-consistent debugging

- Allows one to explore a concurrent computation back and forward
- Any action can be undone provided its consequences have been undone beforehand
- The action to be undone can be selected by the user or by a scheduler
- But we can do better

How to follow causal links?

- If something wrong occurs, find the immediate causal link
- A variable has an unexpected value?
  → Undo its assignment (and inspect it)
- A message has an unexepcted content?
  → Undo its send (and inspect it)
- Either the inspected instruction contains the bug, or we need to iterate the procedure

The `roll` command

- We need a debugger command to perform such undos
- The `roll` command allows one to undo a selected past action, including all and only its consequences
- Minimal set of undos needed to undo the selected action without breaking causal dependencies

Conclusion

- Causal-consistent debugging allows one to explore concurrent computations back and forward
- The `roll` command allows one to follow causal dependencies from the visible misbehavior towards the bug
- CauDEr showcases our approach
- Still a lot of work to be done

Future perspective

- We plan to continue to work on CauDEr and the underlying theory
  - Supporting a larger fragment of the language
  - Understanding causality dependencies
  - Looking for further useful debugging commands
- We will be happy to have any feedback from you

Thanks!

Thanks!

Questions?

Additional resources

- CauDEr repositories:
  - Used: https://github.com/PietroLami/cauder
  - Stable: https://github.com/mistupv/cauder
- Relevant paper (and references therein):
  - I. Lanese, U. P. Schultz, I. Ulidowski: Reversible Computing in Debugging of Erlang Programs. IT Prof. 24(1) (2022)