Discussions in Breakout Groups

Results and Insights

Cooperation of KeY and Other Tools (Jonas Klamroth)

- JML as main exchange format
- how to store proofs?
  - Integration of witnesses
  - non-JML tools
    - give annotations that let KeY reconstruct proof
- encoding intermediate states as JML (or assume/assert statements)
- tools can interact on states of symbolic execution graphs
- theories are axiomatised differently for different tools -> source for communication issues?
- potential for exploiting different strengths of different tools

Trust in Proofs (Hans-Dieter Hiep)

“Who needs trust?”

Consistency

Meta-theory

Language core

Logic core

open-world: programs and theories

conservative/modular extensions

reloading proofs / reproducing

“seal of approval”: proof management

What taclets are used in a proof.

Verified Checker

"Why doesn't everyone use formal verification (yet) and what can we do about it?" (Joshua Bachmeier)

- Industry lives in different world, Rapid Development. Best case: TDD
- Benefits of FV are too indirect, but cost is immediate
- Even if engineers know about FV, management doesn’t know or care
- Industry often has more basic problems / flaws than correctness

- Start in education: engineers / SW-devs, students of practical fields must learn about FV in courses again & again
- Find pain-points of industry (time & money) and where FV can yield savings:
  - Where the error-case is expensive
  - Certification process of large appliances / systems
- Specification that is understandable by non-formal-experts and applicable to wide range of applications (avoid relearning)
- Error messages

<u>=> Shift reputation of FV from theoretical topic (scientists in ivory tower) to practical tool</u>

Future Application Domains (Eduard Kamburjan)

Application Domains

- CPS
- Databases
- Non-sequential programs

Modularity on the Logic and Dynamic Logic Level

Possible Combinations of State Logics and Different Dynamic Logics

Potential Inspirations

- Fibring
- SMT like

User Interface / Usability (Wolfram Pfeifer)

- Syntax highlighting and early feedback on syntax errors is important (and should be consistent with the view in KeY)
- Different interaction paradigms:
  - GUI point-and-click
  - Scripts
  - API
- Good error messages/feedback are important.
- Interaction features/paradigms/concepts need a clear requirements specification

AI for Verification (Thomas Baar)

- Short session with ChatGPT for “What is the JML invariant for …”
  - answer looked nice at the first glance but turned out to be crap when evaluated by experts
  - General observation: answers generally look appealing but (still) turn out to be wrong
- Leads to general question: “Can we trust AI-systems?”
  - Application scenarios can be split into two categories:
    - One cannot (or can hardly) verify/falsify the suggestions be AI-systems (e.g. classification traffic signs)
      - These scenarios are not discussed further by our group
    - One can verify/falsify the suggestions
      - This would be case for scenario “Please generate VeriFast-annotation for given program’
        - One would just start VeriFast for the suggested annotation
- Conclusion/Observation
  - Implementation code might be generated to a large extend in future -> nobody can trust implementation code any longer -> heavily increases the need for formal software verification :-)

We are in the advent of the wide usage of formal methods for establishing trust in implementation code co-autored by systems like ChapGPT!