tcpst todo

todo

February 1, 2023

1 intro

TODO:

- Something on IETF and RFCs
- Motivation behind formalising RFCs
- Session types (possibly how is this different to other formalisms?)

2 Background

TODO:

- some info on other formalisms and why STs are interesting, basically motivation for using STs
- ST - is there a point in explaining binary?
- MPST - is there a point in explaining classic vs less is more?
- could include syntax of less is more for context
- should mention that the tool works by translating ST context to mcrl2 checker (not sure if in too much detail)
- how are STs useful in PLs? Could include snippet here from my Rust lib as examples
- how are STs useful outside of PLs? I.e. that less is more has the model checking tool

3 Discussion/models

TODO:

- explain what properties we are checking for and what they mean
- pop3 model - we can see that its df, all versions of liveness and safe, could be non terminating which is fine

- tcp model - harder protocol, calculus cant describe it fully, not df hence lose over properties
- why its still useful
- why are we not using some calculus with time-outs? (tooling non existent, unclear what these languages mean by timeout)
- what other stuff is missing - what can we model using STs and what can we not?
- difference in how RFCs are designed and STs - non state based, continuation can make things not actually parallel (especially without async)
- what is not done by current ST theory? E.g. The ST concept of time mostly is based on a timeout - how do we describe things like in X time we can send at most Y segments?
- how easy is it to solve this discrepancy using some IR? (TODO: if this point is raised, should probably actually make such an IR?)
- how do we see people using STs for RFCs - should probably make a case for this being hidden away and generated automatically?
- could make some case for some syntax sugar suggestions and modify syntax to include this but this is a dubious contribution?

4 Future directions

TODO:

- balancing static and dynamic checks
- context free?
- a more ”practical” calculus
- parsing and code generation, running model checker on translated code (e.g. in Rust) rather than on sepp ST language
- feedback for RFC maker - how can we use this output to direct the document writing? Can we extract the exact point where something went wrong?
- e.g. we do something and the model is no longer safe, can we say why?
- if we want to include the binary ST library then could talk about how systems languages are suited for STs?
- translating ST type contexts to other model checkers?

5 Random bits that will go somewhere

$$
\begin{array}{rcl}
c & ::= & s[r] \colon S \\
r & ::= & v \\
S & ::= & r \oplus \{l_i(P_i).S_i\}_{i \in I} \mid r \& \{l_i(P_i).S_i\}_{i \in I} \mid \mu(t).(S) \mid t \mid end \\
P & ::= & \{TcpPayloadTypes\} \cup \{Pop3PayloadTypes\}
\end{array}
$$

We only need to consider a simplified subset of the π-calculus. Specifically, we do not consider that a session itself can be sent on a channel. This choice is made as the presented network protocols would not benefit from such a mechanism.

References

[1] Kenneth L. McMillan and Lenore D. Zuck. "Compositional Testing of Internet Protocols". In: 2019. DOI: 10.1109/SecDev.2019.00031.

[2] Stephen McQuistin et al. "Investigating Automatic Code Generation for Network Packet Parsing". In: 2021. DOI: 10.23919/IFIPNetworking52078.2021.9472829.

[3] Stephen McQuistin et al. "Parsing Protocol Standards to Parse Standard Protocols". In: 2020. DOI: 10.1145/3404868.3406671. URL: https://doi.org/10.1145/3404868.3406671.