Exercise 17

donderdag 16 februari 2023 20:27

(a) Place brackets to clarify the interpretation of the following ProVerif process:

```
!  in(net,x) ; in(net,y) | in(net,z) ; out(net,q)
```

```
[!  {in(net,x) ; in(net,y)}] | {in(net,z) ; out(net,q)}
```

Note that it must be the case that the bang (i.e. `!`) refers to everything on the left-hand side of the `|`; `!` indicates that an infinite/arbitrary number of processes may run in parallel. This process in its entirety does not make sense when something runs after the conclusion of the infinite/arbitrary number of processes. Furthermore, we have that `!` binds stronger than `|`.

(b) Translating between SPI and ProVerif (and vice versa)

SPI	ProVerif equivalent
`inp net x; stop`	`in(net,x).`
`P(x,y)== …`	`private free icP. … let P = in(icP, (x,y)); …`
`...P(x,y)...`	`private free icP. … out(icP, (x,y)); P …`

For the latter two, an initialization channel `icP` needs to be defined. This channel is used to input values into a process's variables.

Effectively, we are first (in the last line) placing values on this channel, to then immediately read them upon actually creating the process (in the middle line).

Note that this channel is specifically used for this (single) process, so that the values are guaranteed to end up in the 'called' process.

ProVerif	SPI equivalent
`in( net, (A,=B) ).`	`inp net x; split x is (A, x2); if x2=B then stop`
`let (=A,B,=M) = x in ...`	`split x in (x1, B, x3); if x1=A then if x3=M then …`
`query attacker:  s`	~~`(secret(s)|system)`~~ ~~where *system* is the overall process being checked~~ *replace all occurrences of* `new s; Q` *by* `new s; (secret(s)|Q)` *and check for safety for robust secrecy* *If s is free in the process, add* `|secret(s)` *to the whole process*