Modelling and Verification of Protocols for Wireless Networks

(Lecture 6)

Peter Höfner

(Lecture at University of Twente, Jan/Feb 2017)

www.data61.csiro.au

last update: Feb 2, 2017

Contents of this Lecture

What should you have learnt

- What’s Isabelle/HOL
- Encoding AWN/AODV in Isabelle
  - problems and challenges
  - overall structure

Isabelle/HOL

Isabelle

- generic interactive proof assistant
  - generic:
    not specialised to one particular logic
  - interactive:
    more than just yes/no, you can interactively guide the system
  - proof assistant:
    helps to explore, find, and maintain proofs
    allows mathematical formulas to be expressed in a formal language
- main application is the formalisation of mathematical proofs and in particular formal verification

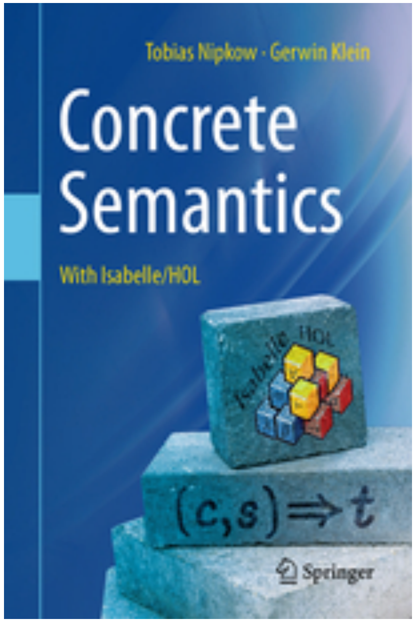
- originally developed at the U Cambridge and TU München
  - now includes numerous contributions, including Data61
- developed since the 80s

Isabelle/HOL

- most widespread instance of Isabelle
- provides a higher-order logic theorem proving environment
- includes powerful specification tools, e.g. for (co)datatypes, (co)inductive definitions and recursive functions with complex pattern matching.
- Proofs are often conducted in the structured proof language Isar
  - allows for proof text naturally understandable for humans

Isabelle - An Introduction

- Concrete Semantics
  by G. Klein and T. Nipkow



- it's available online
  http://concrete-semantics.org/

If I use Isabelle, it’s correct?

- NO,
  - implementation/specification could be faulty
  - logic could be inconsistent
  - theorem could mean something else

but assurance is increased

AWN, AODV, and Loop Freedom

- Why bother?
  - Can such a ‘manual’ proof be trusted (over time)?
  - The coarse structure of the proof is much looser than the fine details (i.e., the individual invariants)
  - Formalising it turns out to be an interesting challenge

  - Reuse the development
    - Changes to/variants of AODV
    - Development of new protocols
    - Provide a formal specification for verifying implementations?

AWN in Isabelle

Theorem: Loop Freedom

closed (pnet (λi. paodv i ⟨⟨ qmsg) n) ||= netglobal (λσ. ∀dip. irrefl ((rt-graph σ dip)$^{+}$))

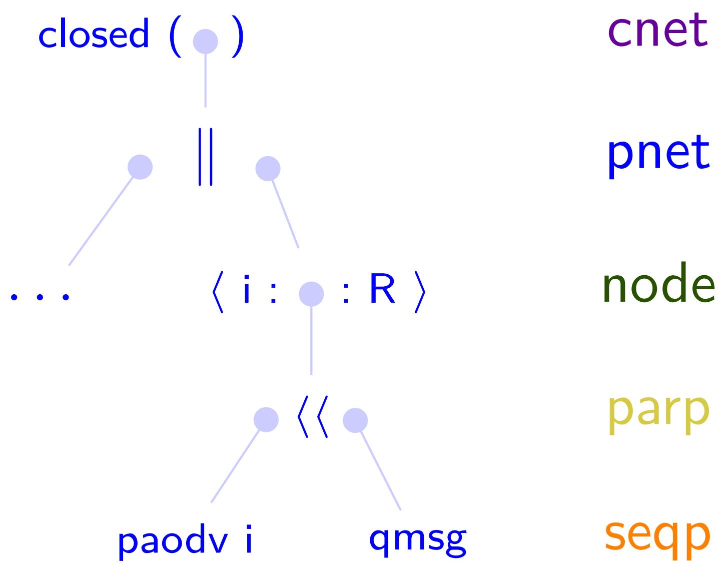
description of the network

lemma net_nhop_quality_increases:
  assumes "wf_net_tree n"
  shows "closed (pnet (λi. paodv i ⟨⟨ qmsg) n) ||= netglobal
    (λσ. ∀i dip. let nhip = the (nhop (rt (σ i)) dip)
      in dip ∈ vD (rt (σ i)) ∩ vD (rt (σ nhip)) ∧ nhip ≠ dip
      ⟶ (rt (σ i)) ⊏⇘dip⇙ (rt (σ nhip)))"

invariant

$$dip \in \mathtt{vD}_N^{ip} \cap \mathtt{vD}_N^{nhip} \;\wedge\; nhip \neq dip \;\Rightarrow\; \xi_N^{ip}(\mathtt{rt}) \sqsubset_{dip} \xi_N^{nhip}(\mathtt{rt})$$

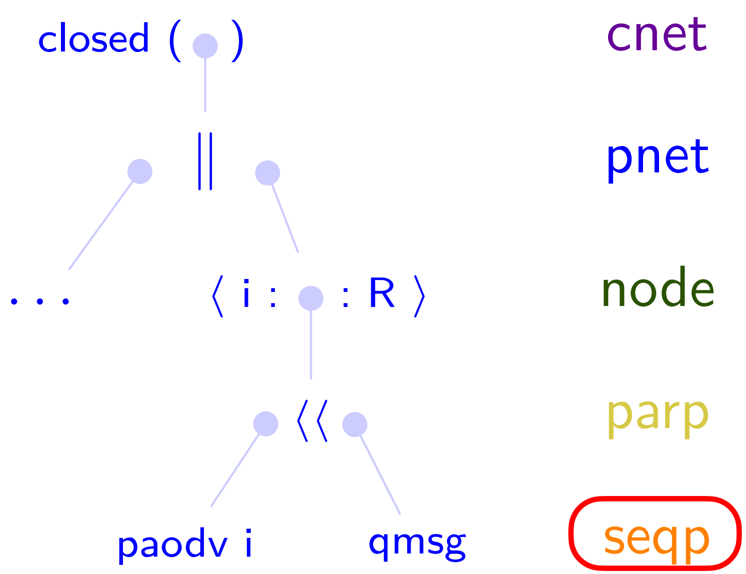
Mechanising AWN



- AWN: layered process algebra
- SOS rules for each ‘operator’
- Layers transform lower layers

- Model all as automata
(initial states and transitions)

Mechanising AWN



- AWN: layered process algebra
- SOS rules for each ‘operator’
- Layers transform lower layers

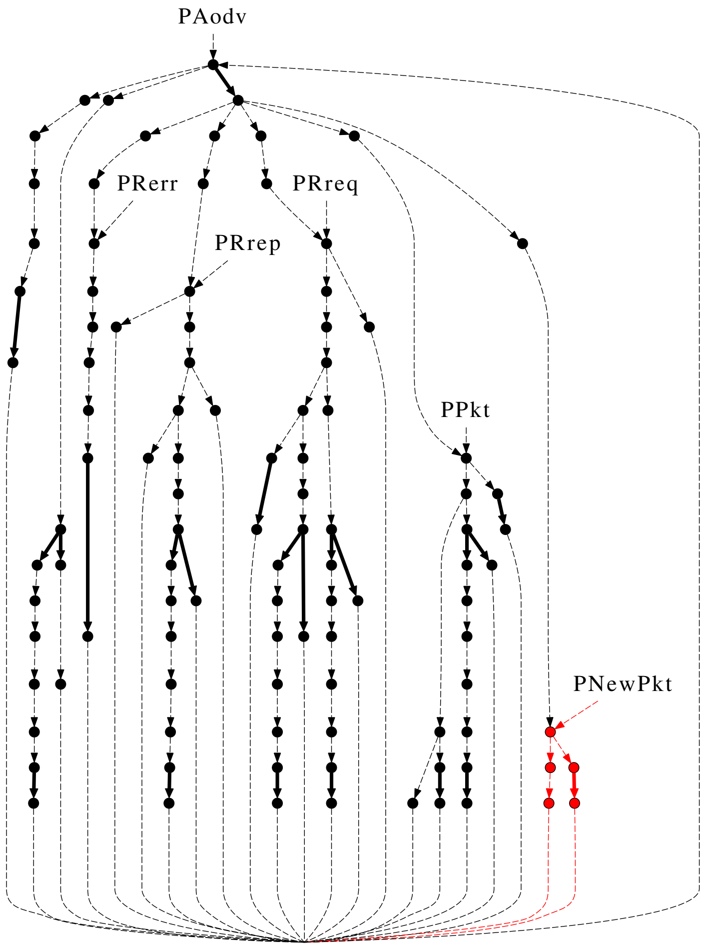
- Model all as automata (initial states and transitions)

$\text{paodv i} = (\!|\text{init} = \{(\text{aodv-init i}, \Gamma_{\text{AODV}}\ \text{PAodv})\}, \text{trans} = \text{seqp-sos}\ \Gamma_{\text{AODV}}|\!)$

$$((\xi, \{l\}\text{groupcast}(\text{ips}, \text{ms}) \,.\, p), \text{groupcast}\ (\text{ips}\ \xi)\ (\text{ms}\ \xi), (\xi, p)) \in \text{seqp-sos}\ \Gamma$$

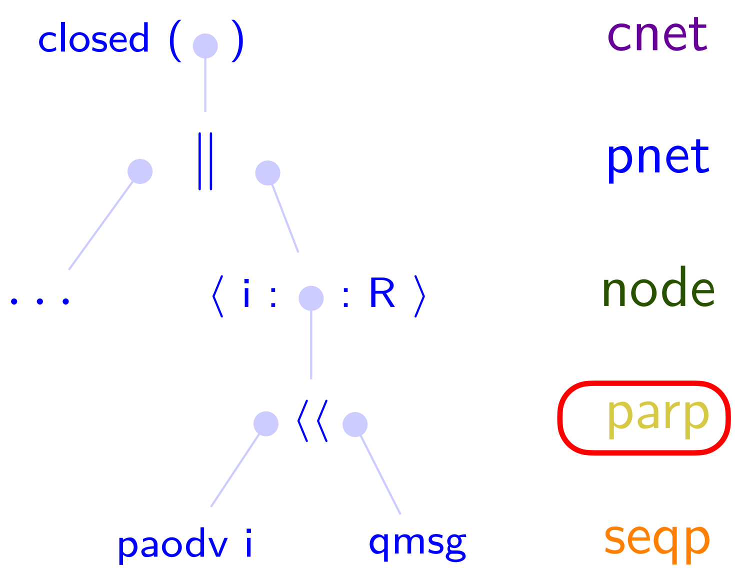
$$\frac{\xi' = \text{fa}\ \xi}{((\xi, \{l\}[\![\text{fa}]\!]\ p), \tau, (\xi', p)) \in \text{seqp-sos}\ \Gamma}$$

Mechanising AWN



Γ_{AODV} PNewPkt $=$ labelled PNewPkt (
$\langle \lambda\xi.$ if dip ξ $=$ ip ξ then $\{\xi\}$ else $\emptyset\rangle$
deliver(data) . $[\![$clear-locals$]\!]$ call(PAodv)
$\oplus$
$\langle \lambda\xi.$ if dip ξ $\neq$ ip ξ then $\{\xi\}$ else $\emptyset\rangle$
$[\![\lambda\xi.\ \xi(\!|$store := add (data ξ) (dip ξ) (store ξ)$|\!)]\!]$
$[\![$clear-locals$]\!]$ call(PAodv))

Mechanising AWN



- AWN: layered process algebra
- SOS rules for each 'operator'
- Layers transform lower layers

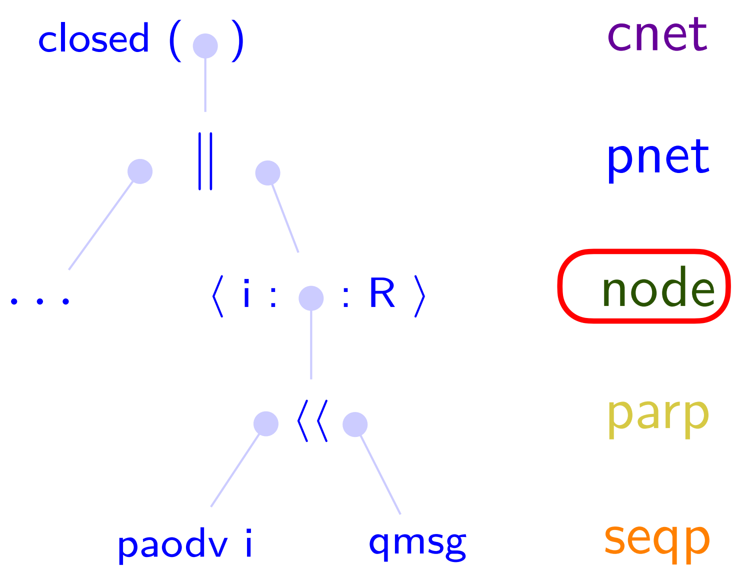
- Model all as automata (initial states and transitions)

$$s \langle\langle\, t \equiv (\!| \text{init} = \text{init } s \times \text{init } t,\ \text{trans} = \text{parp-sos } (\text{trans } s)\ (\text{trans } t) |\!)$$

$$\frac{(s, a, s') \in S \qquad \bigwedge m.\ a \neq \text{receive } m}{((s, t), a, (s', t)) \in \text{parp-sos } S\ T} \qquad \frac{(t, a, t') \in T \qquad \bigwedge m.\ a \neq \text{send } m}{((s, t), a, (s, t')) \in \text{parp-sos } S\ T}$$

$$\frac{(s, \text{receive } m, s') \in S \qquad (t, \text{send } m, t') \in T}{((s, t), \tau, (s', t')) \in \text{parp-sos } S\ T}$$

Mechanising AWN

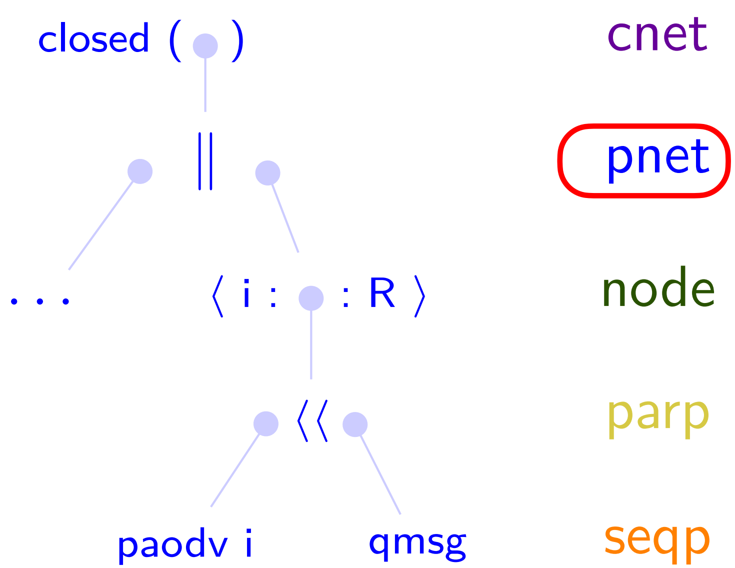


- AWN: layered process algebra
- SOS rules for each ‘operator’
- Layers transform lower layers
- Model all as automata
(initial states and transitions)

$$\langle i : S : R \rangle \equiv (\!| \text{init} = \{s_R^i \mid s \in \text{init } S\},\ \text{trans} = \text{node-sos } (\text{trans } S) |\!)$$

$$\frac{(s,\ \text{groupcast } D\ m,\ s') \in S}{(s_R^i,\ (R \cap D){:}{*}\text{cast}(m),\ s'^{\,i}_R) \in \text{node-sos } S} \qquad (P_R^i,\ \text{connect}(i, i'),\ P^i_{R \cup \{i'\}}) \in \text{node-sos } S$$

Mechanising AWN



- AWN: layered process algebra
- SOS rules for each ‘operator’
- Layers transform lower layers

- Model all as automata (initial states and transitions)

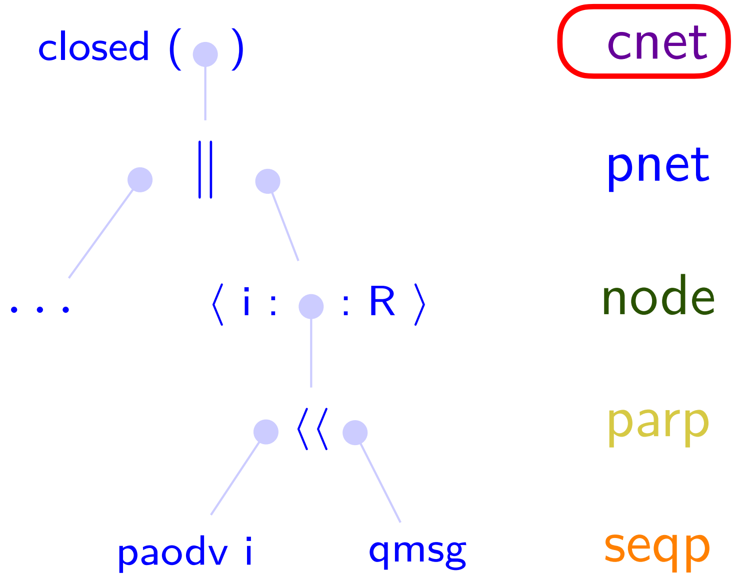
$$\text{pnet np } \langle i; R\rangle \;=\; \langle i : \text{np } i : R\rangle$$

$$\text{pnet np } (p_1 \parallel p_2) \;=\; (\!|\text{init} = \{s1 \shortparallel s2 \mid s1 \in \text{init (pnet np } p_1) \wedge s2 \in \text{init (pnet np } p_2)\},$$
$$\text{trans} = \text{pnet-sos (trans (pnet np } p_1)) \text{ (trans (pnet np } p_2))|\!)$$

$$\frac{(s, \tau, s') \in S}{(s \shortparallel t, \tau, s' \shortparallel t) \in \text{pnet-sos } S\ T}$$

$$\frac{(s, R{:}{*}cast(m), s') \in S \qquad (t, H\neg K{:}arrive(m), t') \in T \qquad H \subseteq R \qquad K \cap R = \emptyset}{(s \shortparallel t, R{:}{*}cast(m), s' \shortparallel t') \in \text{pnet-sos } S\ T}$$

Mechanising AWN



- AWN: layered process algebra
- SOS rules for each 'operator'
- Layers transform lower layers

- Model all as automata (initial states and transitions)

Limitations

- allowed are only processes of the form

$$(P \langle\langle\, \mathtt{qmsg}) \parallel (P \langle\langle\, \mathtt{qmsg}) \| \ldots$$

An Example

$$
\begin{array}{llr}
\mathsf{PToy} = & (\ \mathsf{receive}(\lambda \mathsf{msg}'\ \xi.\ \xi\ (\!|\ \mathsf{msg} := \mathsf{msg}'\ |\!)). & \{\mathsf{PToy}\text{-}{:}0\}\\
& [\![\lambda\xi.\ \xi\ (\!|\mathsf{nhid} := \mathsf{id}\ \xi|\!)]\!] & \{\mathsf{PToy}\text{-}{:}1\}\\
& (\quad \langle \mathsf{is\text{-}newpkt}\rangle & \{\mathsf{PToy}\text{-}{:}2\}\\
& \quad [\![\lambda\xi.\ \xi\ (\!|\mathsf{no} := \mathsf{max}\ (\mathsf{no}\ \xi)\ (\mathsf{num}\ \xi)|\!)]\!] & \{\mathsf{PToy}\text{-}{:}3\}\\
& \quad \mathsf{broadcast}(\lambda\xi.\ \mathsf{pkt}(\mathsf{no}\ \xi,\ \mathsf{id}\ \xi)). & \{\mathsf{PToy}\text{-}{:}4\}\\
& \quad [\![\mathsf{clear\text{-}locals}]\!]\ \mathsf{call}(\mathsf{PToy}) & \{\mathsf{PToy}\text{-}{:}5\}\\
& \oplus\ \langle \mathsf{is\text{-}pkt}\rangle & \{\mathsf{PToy}\text{-}{:}2\}\\
& \quad (\ \ \langle\lambda\xi.\ \mathsf{if}\ \mathsf{num}\ \xi \geq \mathsf{no}\ \xi\ \mathsf{then}\ \{\xi\}\ \mathsf{else}\ \emptyset\rangle & \{\mathsf{PToy}\text{-}{:}6\}\\
& \qquad [\![\lambda\xi.\ \xi\ (\!|\mathsf{no} := \mathsf{num}\ \xi|\!)]\!] & \{\mathsf{PToy}\text{-}{:}7\}\\
& \qquad [\![\lambda\xi.\ \xi\ (\!|\mathsf{nhid} := \mathsf{sid}\ \xi|\!)]\!] & \{\mathsf{PToy}\text{-}{:}8\}\\
& \qquad \mathsf{broadcast}(\lambda\xi.\ \mathsf{pkt}(\mathsf{no}\ \xi,\ \mathsf{id}\ \xi)). & \{\mathsf{PToy}\text{-}{:}9\}\\
& \qquad [\![\mathsf{clear\text{-}locals}]\!]\ \mathsf{call}(\mathsf{PToy}) & \{\mathsf{PToy}\text{-}{:}10\}\\
& \quad \oplus\ \langle\lambda\xi.\ \mathsf{if}\ \mathsf{num}\ \xi < \mathsf{no}\ \xi\ \mathsf{then}\ \{\xi\}\ \mathsf{else}\ \emptyset\rangle & \{\mathsf{PToy}\text{-}{:}6\}\\
& \qquad [\![\mathsf{clear\text{-}locals}]\!]\ \mathsf{call}(\mathsf{PToy})))) & \{\mathsf{PToy}\text{-}{:}11\}
\end{array}
$$

An Example

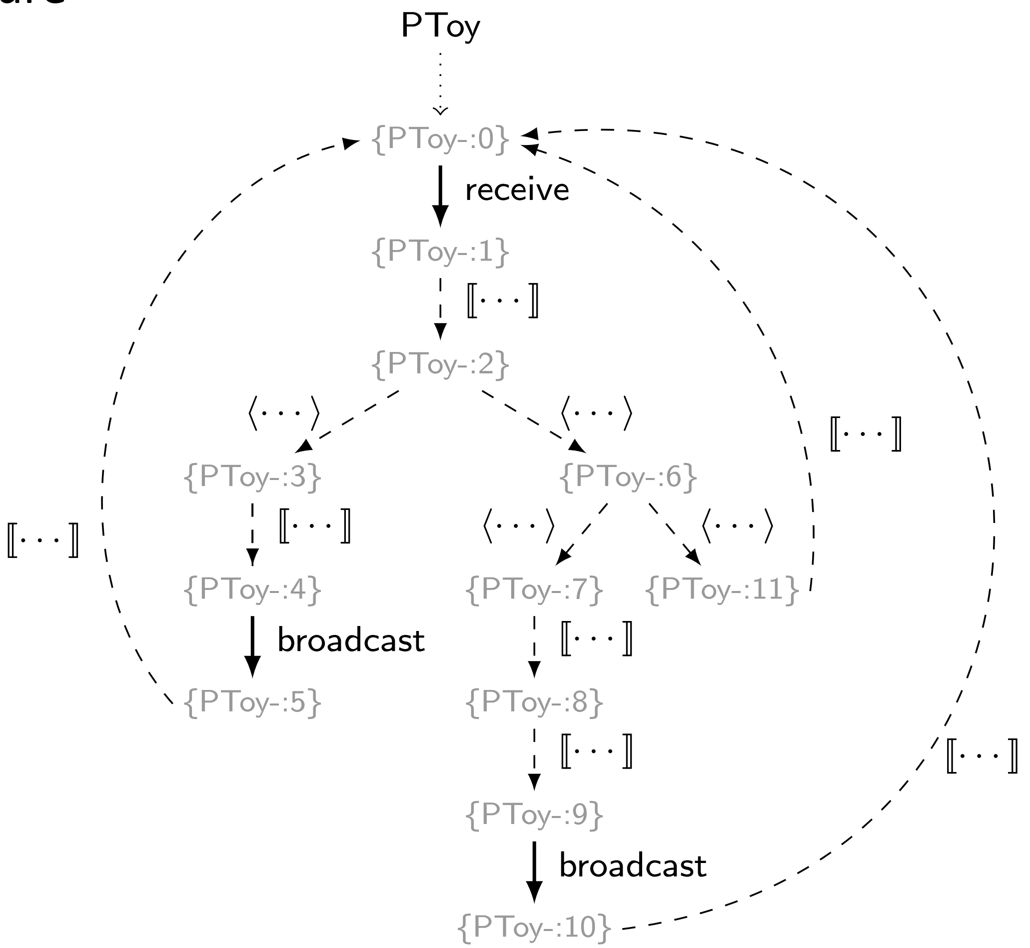
- the "magic" assignment of AWN needs to be implemented

$$\begin{array}{rcl} \textsf{is-newpkt}\ \xi & = & \textsf{case msg}\ \xi\ \textsf{of} \\ & & \quad \textsf{Pkt d sid} \Rightarrow\ \emptyset \\ & & \mid \textsf{Newpkt d dst} \Rightarrow \{\xi(\!|\textsf{num} := \textsf{d}|\!)\} \end{array}$$

- another difference is "clear locals"

An Example

- Control Structure



In-built Message Queue

$$
\begin{array}{lll}
\mathsf{Qmsg} = \mathsf{Qmsg}\ (& \mathsf{receive}(\lambda \mathsf{msg}\ \mathsf{msgs}.\ \mathsf{msgs}\ @\ [\mathsf{msg}])\ .\ \mathsf{call}(\mathsf{Qmsg}) & \{\mathsf{Qmsg\text{-}:0}\} \\
& \oplus \langle \lambda \mathsf{msgs}.\ \mathsf{if}\ \mathsf{msgs} \neq []\ \mathsf{then}\ \{\mathsf{msgs}\}\ \mathsf{else}\ \emptyset \rangle & \{\mathsf{Qmsg\text{-}:0}\} \\
& \quad (\quad \mathsf{send}(\lambda \mathsf{msgs}.\ \mathsf{hd}\ \mathsf{msgs})\ . & \{\mathsf{Qmsg\text{-}:1}\} \\
& \qquad [\![\lambda \mathsf{msgs}.\ \mathsf{tl}\ \mathsf{msgs}]\!]\ \mathsf{call}(\mathsf{Qmsg}) & \{\mathsf{Qmsg\text{-}:2}\} \\
& \quad \oplus\ \mathsf{receive}(\lambda \mathsf{msg}\ \mathsf{msgs}.\ \mathsf{msgs}\ @\ [\mathsf{msg}])\ .\ \mathsf{call}(\mathsf{Qmsg}))) & \{\mathsf{Qmsg\text{-}:1}\}
\end{array}
$$

Mechanising Properties

Properties

- so far *only invariants* supported
  (no reasoning over traces/paths necessary)

- reachability

$$\frac{s \in \text{init } A}{s \in \text{reachable } A \ I} \qquad \frac{s \in \text{reachable } A \ I \quad (s, a, s') \in \text{trans } A \quad I \ a}{s' \in \text{reachable } A \ I}$$

For an assertion φ,

$$\frac{\begin{array}{ll}\text{B1.} & \Theta \rightarrow \varphi \\ \text{B2.} & \{\varphi\}\ \mathcal{T}\ \{\varphi\}\end{array}}{\Box\varphi}$$

Fig. 1.1. Rule INV-B (basic invariance).

An Example

```
PToy = ( receive(λmsg' ξ. ξ (| msg := msg' |)).                    {PToy-:0}
         [[λξ. ξ (|nhid := id ξ|)]]                                {PToy-:1}
         (   ⟨is-newpkt⟩                                           {PToy-:2}
             [[λξ. ξ (|no := max (no ξ) (num ξ)|)]]                {PToy-:3}
             broadcast(λξ. pkt(no ξ, id ξ)).                       {PToy-:4}
             [[clear-locals]] call(PToy)                           {PToy-:5}
          ⊕ ⟨is-pkt⟩                                               {PToy-:2}
             (  ⟨λξ. if num ξ ≥ no ξ then {ξ} else ∅⟩              {PToy-:6}
                 [[λξ. ξ (|no := num ξ|)]]                         {PToy-:7}
                 [[λξ. ξ (|nhid := sid ξ|)]]                       {PToy-:8}
                 broadcast(λξ. pkt(no ξ, id ξ)).                   {PToy-:9}
                 [[clear-locals]] call(PToy)                       {PToy-:10}
              ⊕ ⟨λξ. if num ξ < no ξ then {ξ} else ∅⟩              {PToy-:6}
                 [[clear-locals]] call(PToy))))                    {PToy-:11}
```

State Invariant:

$$\text{ptoy i} \Vdash \text{onl } \Gamma_{\text{Toy}}\ (\lambda(\xi, \text{l}).\ \text{l} \in \{\text{PToy-:2..PToy-:8}\} \longrightarrow \text{nhid } \xi = \text{state.id } \xi)$$

Definition (invariance) Given an automaton A and an assumption I, a predicate P is *(state) invariant*, denoted $\mathsf{A} \Vdash (\mathsf{I} \rightarrow)\ \mathsf{P}$, iff $\forall \mathsf{s} \in \text{reachable A I. P s}$.

An Example

$$
\begin{array}{lll}
\text{PToy} = & (\ \text{receive}(\lambda \text{msg}'\ \xi.\ \xi\ (\!|\ \text{msg} := \text{msg}'\ |\!)). & \{\text{PToy-:0}\}\\
& [\![\lambda\xi.\ \xi\ (\!|\text{nhid} := \text{id}\ \xi|\!)]\!] & \{\text{PToy-:1}\}\\
& (\quad \langle\text{is-newpkt}\rangle & \{\text{PToy-:2}\}\\
& \quad [\![\lambda\xi.\ \xi\ (\!|\text{no} := \max\ (\text{no}\ \xi)\ (\text{num}\ \xi)|\!)]\!] & \{\text{PToy-:3}\}\\
& \quad \text{broadcast}(\lambda\xi.\ \text{pkt}(\text{no}\ \xi,\ \text{id}\ \xi)). & \{\text{PToy-:4}\}\\
& \quad [\![\text{clear-locals}]\!]\ \text{call}(\text{PToy}) & \{\text{PToy-:5}\}\\
& \oplus\ \langle\text{is-pkt}\rangle & \{\text{PToy-:2}\}\\
& \quad (\ \ \langle\lambda\xi.\ \text{if num}\ \xi \geq \text{no}\ \xi\ \text{then}\ \{\xi\}\ \text{else}\ \emptyset\rangle & \{\text{PToy-:6}\}\\
& \quad\quad [\![\lambda\xi.\ \xi\ (\!|\text{no} := \text{num}\ \xi|\!)]\!] & \{\text{PToy-:7}\}\\
& \quad\quad [\![\lambda\xi.\ \xi\ (\!|\text{nhid} := \text{sid}\ \xi|\!)]\!] & \{\text{PToy-:8}\}\\
& \quad\quad \text{broadcast}(\lambda\xi.\ \text{pkt}(\text{no}\ \xi,\ \text{id}\ \xi)). & \{\text{PToy-:9}\}\\
& \quad\quad [\![\text{clear-locals}]\!]\ \text{call}(\text{PToy}) & \{\text{PToy-:10}\}\\
& \quad \oplus\ \langle\lambda\xi.\ \text{if num}\ \xi < \text{no}\ \xi\ \text{then}\ \{\xi\}\ \text{else}\ \emptyset\rangle & \{\text{PToy-:6}\}\\
& \quad\quad [\![\text{clear-locals}]\!]\ \text{call}(\text{PToy})))) & \{\text{PToy-:11}\}
\end{array}
$$

Step Invariant:

$$\text{ptoy i} \models (\lambda((\xi, -), -, (\xi', -)).\ \text{no}\ \xi \leq \text{no}\ \xi')$$

Definition (step invariance) Given an automaton A and an assumption I, a predicate P is *step invariant*, denoted $A \models (I \rightarrow) P$, iff

$$\forall a.\ I\ a \longrightarrow (\forall s \in \text{reachable A I}.\ \forall s'.\ (s, a, s') \in \text{trans A} \longrightarrow P\ (s, a, s')).$$

Lecture 5: Auxiliary Invariants for AODV

- All routing table entries have a hop count greater or equal than 1.

$$(*,*,*,*,hops,*,*) \in \xi_N^{ip}(\mathtt{rt}) \;\Rightarrow\; hops \geq 1$$

just some decoration to identify node, and state of the network

- Whenever an originator sequence number is sent as part of a *route request* message, it is known, i.e., it is greater or equal than 1.

$$N \xrightarrow{R{:}*\mathbf{cast}(\mathtt{rreq}(*,*,*,*,*,*,osn_c,*))}_{ip} N' \;\Rightarrow\; osn_c \geq 1$$

- Whenever a destination sequence number is sent as part of a *route reply* message, it is known, i.e., it is greater or equal than 1.

$$N \xrightarrow{R{:}*\mathbf{cast}(\mathtt{rrep}(*,*,dsn_c,*,*))}_{ip} N' \;\Rightarrow\; dsn_c \geq 1$$

Inter-Node invariants

- examples are loop freedom

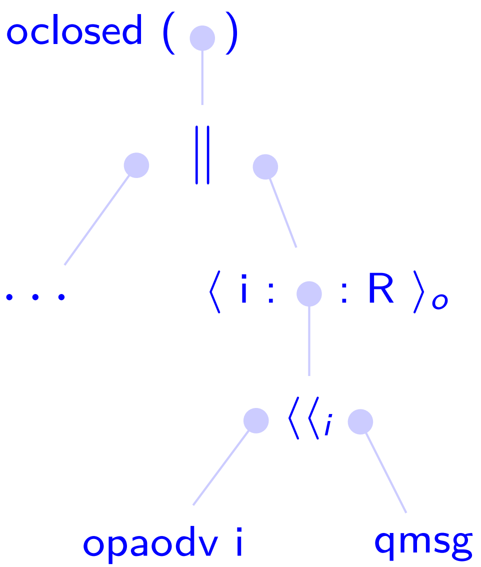
$$dip \in \mathtt{vD}_N^{ip} \cap \mathtt{vD}_N^{nhip} \;\wedge\; nhip \neq dip \;\Rightarrow\; \xi_N^{ip}(\mathtt{rt}) \sqsubset_{dip} \xi_N^{nhip}(\mathtt{rt})$$

or

$$\mathsf{closed}\ (\mathsf{pnet}\ (\lambda \mathsf{i}.\ \mathsf{ptoy}\ \mathsf{i} \ \langle\langle\ \mathsf{qmsg})\ \Psi) \ \Vert\!\!= \quad \mathsf{netglobal}\ (\lambda\sigma.\ \forall \mathsf{i}.\ \mathsf{no}\ (\sigma\ \mathsf{i}) \leq \mathsf{no}\ (\sigma\ (\mathsf{nhid}\ (\sigma\ \mathsf{i}))))$$

AWN in Isabelle (2)

An “open” Model



ocnet

opnet

onode

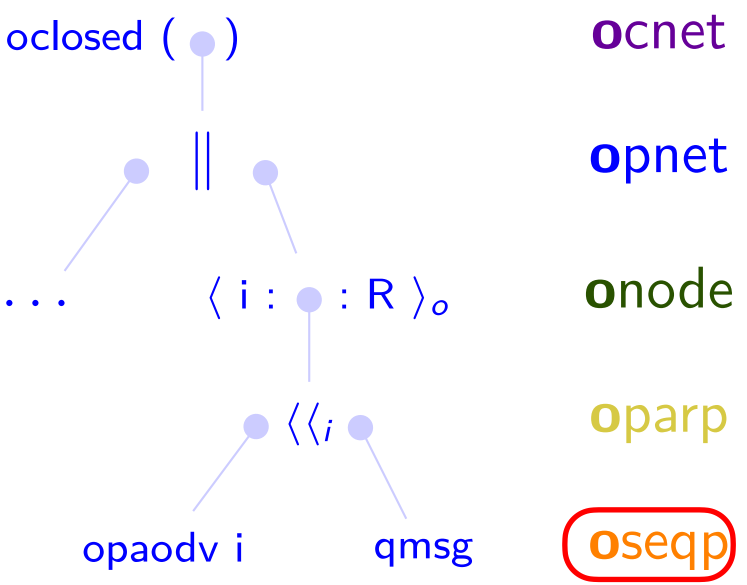
oparp

oseqp

$$\xi :: \text{state}$$

$$\sigma :: \text{ip} \Rightarrow \text{state}$$

An “open” Model

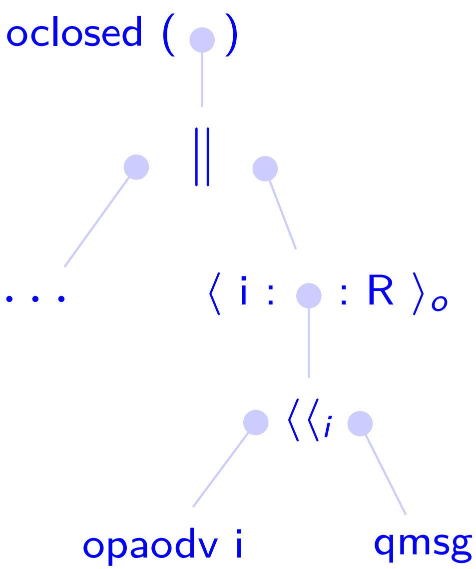


$$\frac{\sigma'\ i = \mathsf{fa}\ (\sigma\ i)}{((\sigma,\ \{\mathsf{l}\}[\![\mathsf{fa}]\!]\ \mathsf{p}),\ \tau,\ (\sigma',\ \mathsf{p}))\ \in\ \mathbf{o}\mathsf{seqp\text{-}sos}\ \Gamma\ \mathsf{i}}$$

versus

$$\frac{\xi' = \mathsf{fa}\ \xi}{((\xi,\ \{\mathsf{l}\}[\![\mathsf{fa}]\!]\ \mathsf{p}),\ \tau,\ (\xi',\ \mathsf{p}))\ \in\ \mathsf{seqp\text{-}sos}\ \Gamma}$$

An "open" Model



ocnet

opnet

onode

oparp

oseqp

$$\frac{((\sigma, P), \tau, (\sigma', P')) \in S}{((\sigma, P^{i}_{R}), \tau, (\sigma', P'^{i}_{R})) \in \textbf{o}\text{node-sos}\ S}$$

More lifting

- Definition of invariance need to be lifted to the open model
  - taking all other nodes into account, etc.
  - make assumptions about environment (e.g. message correct content) added as another condition (which need to be proven later)
  - pretty complicated

Examples

- Toy Protocol

$$\text{optoy i} \models (\text{otherwith nos-inc } \{i\} \text{ (orecvmsg msg-ok), other nos-inc } \{i\} \rightarrow)$$
$$(\lambda(\sigma, \text{-}).\ \text{no } (\sigma\ i) \leq \text{no } (\sigma\ (\text{nhid } (\sigma\ i))))\ ,$$

- AODV

$$\text{opaodv i} \models (\text{otherwith (op=) } \{i\} \text{ (orecvmsg } (\lambda \sigma\ m.\ \text{msg-fresh } \sigma\ m \wedge \text{msg-zhops m})),$$
$$\text{other quality-increases } \{i\} \rightarrow) \text{ onl } \Gamma_P\ (\lambda(\sigma, \text{-}).$$
$$\forall \text{dip. let nhip} = \text{the (nhop (rt } (\sigma\ i)) \text{ dip) in}$$
$$\text{dip} \in \text{vD (rt } (\sigma\ i)) \cap \text{vD (rt } (\sigma \text{ nhip})) \wedge \text{nhip} \neq \text{dip} \rightarrow \text{rt } (\sigma\ i) \sqsubset_{dip} \text{rt } (\sigma \text{ nhip}))$$

Overall Proof structure

cnet-sos ← - - - transfer - - - **o**cnet-sos

closed (pnet (λi. paodv i $\langle\langle$ qmsg) n) $\Vvdash P$

oclosed (opnet (λi. opaodv i $\langle\langle$ qmsg) n) $\Vvdash P'_5$

↑ lift

pnet-sos **o**pnet-sos

opnet (λi. opaodv i $\langle\langle$ qmsg) n $\Vvdash P'_4$

↑ lift

node-sos **o**node-sos

$\langle$i : opaodv i $\langle\langle$ qmsg : R$_i\rangle_o \Vvdash P'_3$

↑ lift

parp-sos **o**parp-sos

opaodv i $\langle\langle$ qmsg $\Vvdash P'_2$

↑ lift

seqp-sos - - - transfer - - - → **o**seqp-sos

paodv i $\Vvdash P_1$

opaodv i $\Vvdash P'_1$

Summary

- the 'open' model is used only in the proof
- was more complicated as anticipated
- fully mechanised
- no liveness yet

- Advantages
  - proof certificate
  - ideal for analysing variants
    (replay proof)

References

- T. Bourke, R.J. van Glabbeek, P. Höfner: *Mechanizing a Process Algebra for Network Protocols*. In Journal of Automated Reasoning 56(3):309-341, Springer, 2016. doi: 10.1007/s10817-015-9358-9