Not quite synchronous enough

Compositional invariant proofs of reactive systems in an interactive theorem prover (Isabelle/HOL)

Timothy Bourke

INRIA Paris-Rocquencourt

École normale supérieure (DI)

`http://www.di.ens.fr/ParkasTeam.html`

(In collaboration with R.J. van Glabbeek and P. Höfner of UNSW/NICTA Sydney)

3 December 2014, SYNCHRON Workshop at Aussois

Context

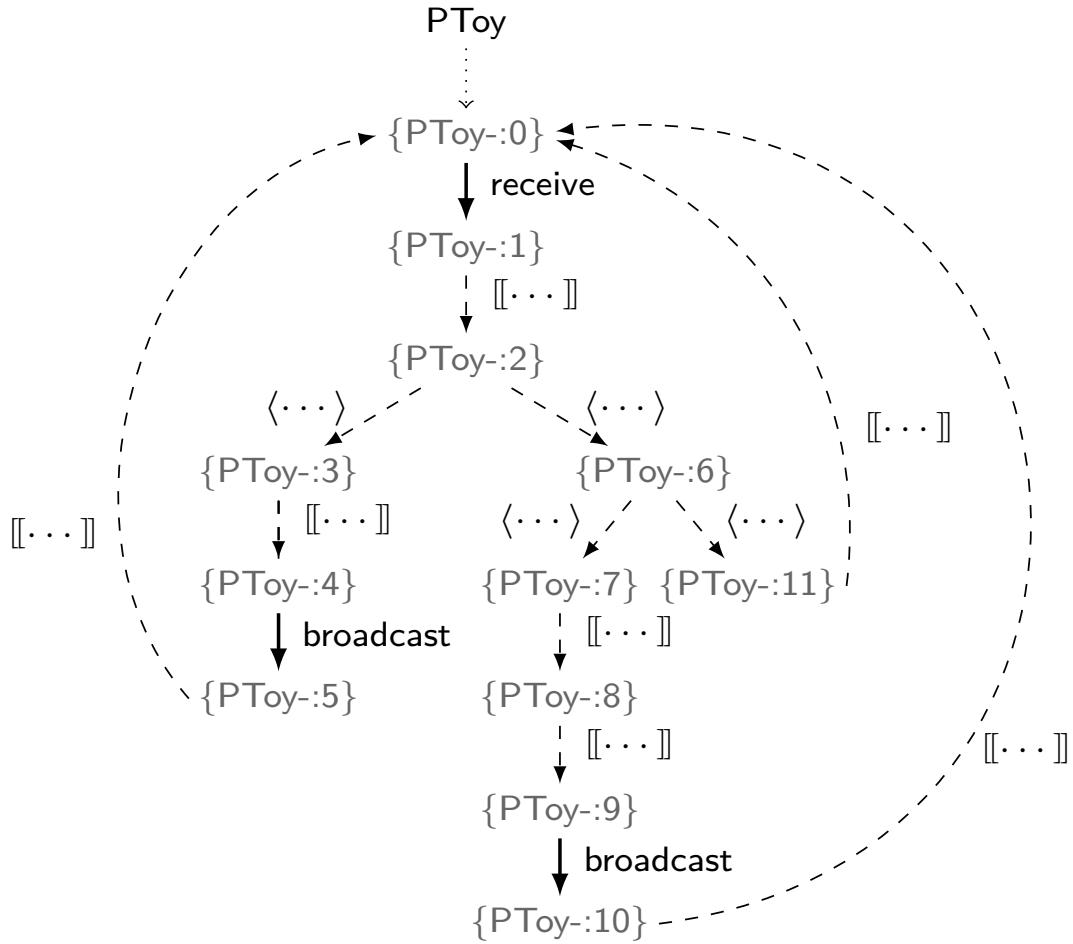
- My colleagues at NICTA in Sydney
  - developed the Algebra for Wireless Networks (AWN),
  - applied it to model the AODV (RFC 3561) routing protocol, and
  - verified 'loop freedom'—an important invariant.
- I thought it would fun to 'mechanize' this work:
  - Real-world reactive system;
  - Formal modelling in an interactive theorem prover.

Goals of this talk

- Show one way to model reactive systems and verify safety properties.
- Give some 'feeling' for how it works in Isabelle:
  - Extensible platform: syntax and tool integration.
  - Powerful: parallelism for proofs, access to SMT solvers.
  - Sound: LCF kernel and Higher Order Logic.
- Compare 'synchronous' and 'unsynchronous' rules.

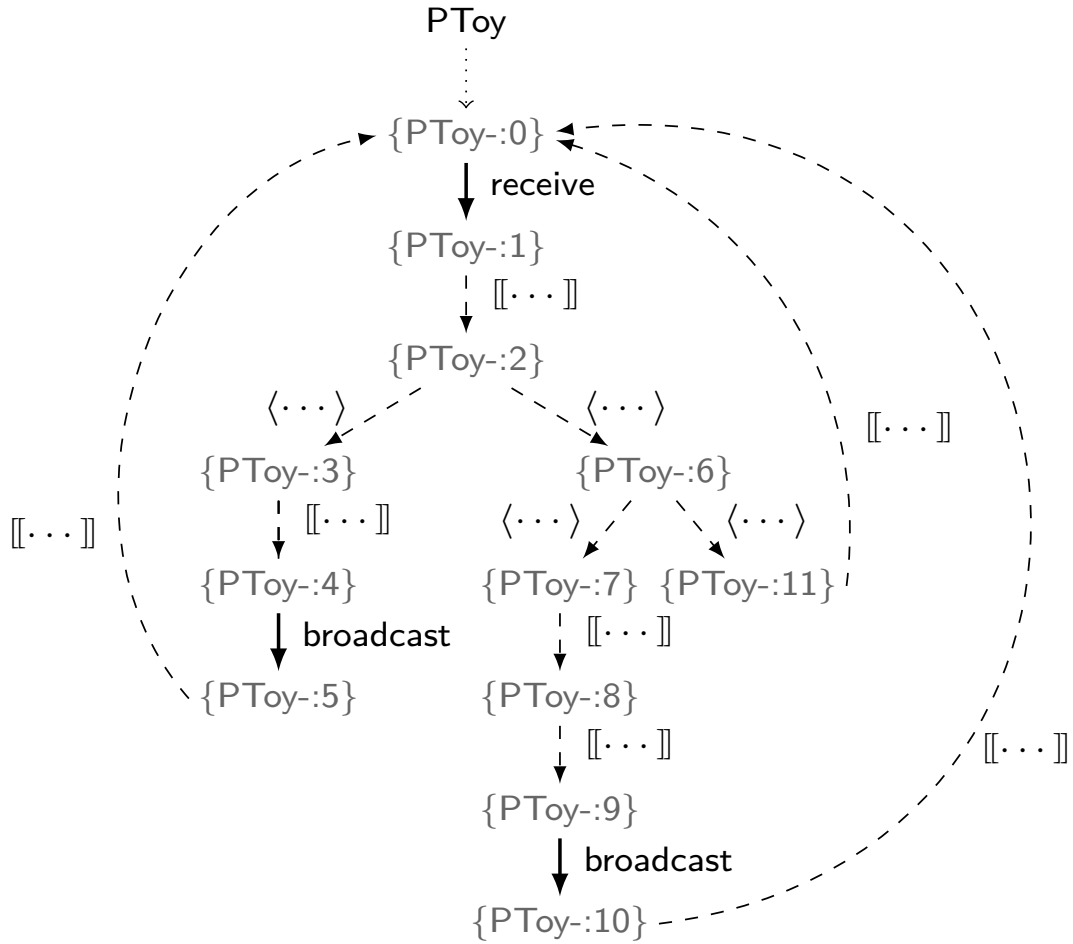
A sequential process specification

Γ_{Toy} 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}



A sequential process specification

Γ_{Toy} 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}



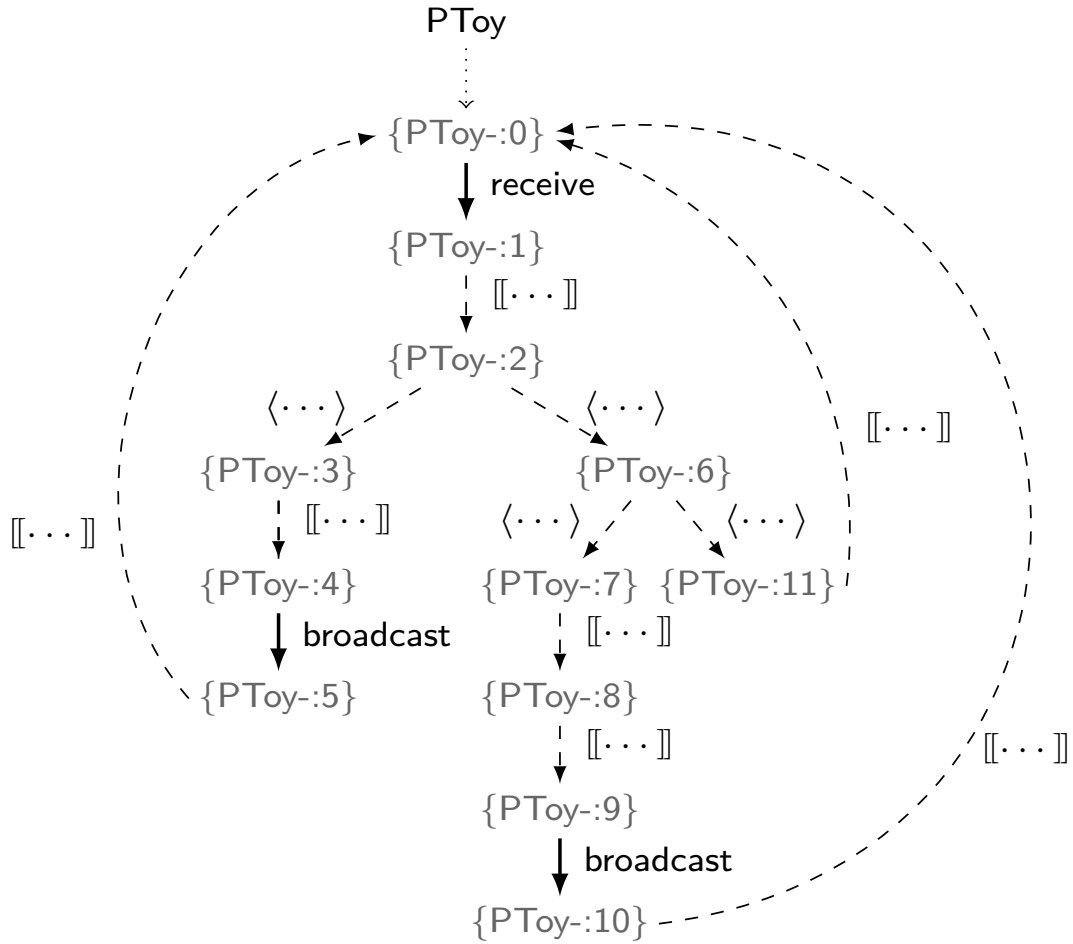
ptoy i = (|init = {(toy-init i, Γ_{Toy} PToy)}, trans = seqp-sos Γ_{Toy}|)

toy-init i =

(|id = i, no = 0, nhid = i, msg = SOME x. True, num = SOME x. True, sid = SOME x. True|)

A sequential process specification

Γ_{Toy} 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}



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

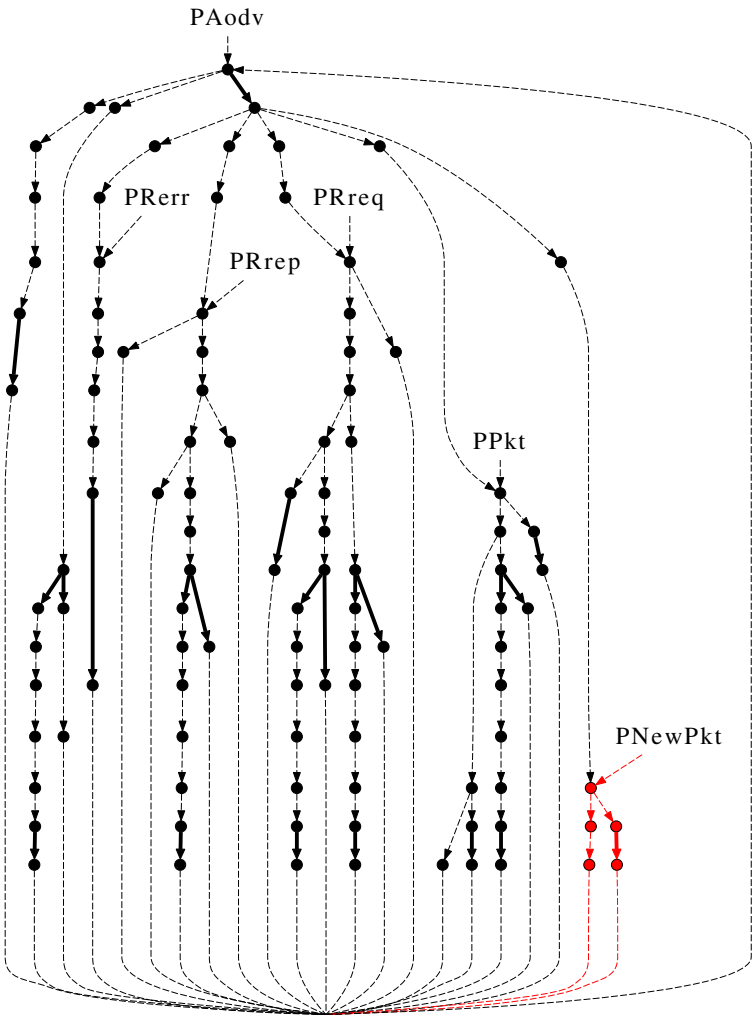
$$\text{ptoy i} = (|\text{init} = \{(\text{toy-init i},\ \Gamma_{\text{Toy}}\ \text{PToy})\},\ \text{trans} = \text{seqp-sos}\ \Gamma_{\text{Toy}}|)$$

$$((\xi,\ p),\ a,\ (\xi',\ p')) \in \textit{seqp-sos}\ \Gamma$$

toy-init i =

(|id = i, no = 0, nhid = i, msg = SOME x. True, num = SOME x. True, sid = SOME x. True|)

Modelling AODV: control state



Γ_{AODV} PNewPkt $=$ labelled PNewPkt (

$\langle \lambda \xi.$ if dip $\xi =$ ip ξ then $\{\xi\}$ else $\emptyset \rangle$

deliver(data) . $[\![$clear-locals$]\!]$ call(PAodv)

$\oplus$

$\langle \lambda \xi.$ if dip $\xi \neq$ ip ξ then $\{\xi\}$ else $\emptyset \rangle$

$[\![\lambda \xi.\ \xi(|$store := add (data ξ) (dip ξ) (store $\xi)|)]\!]$

$[\![$clear-locals$]\!]$ call(PAodv))

Stating invariant properties

Reachability

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

- Focus on invariants of states and steps
- Not necessary to reason over traces

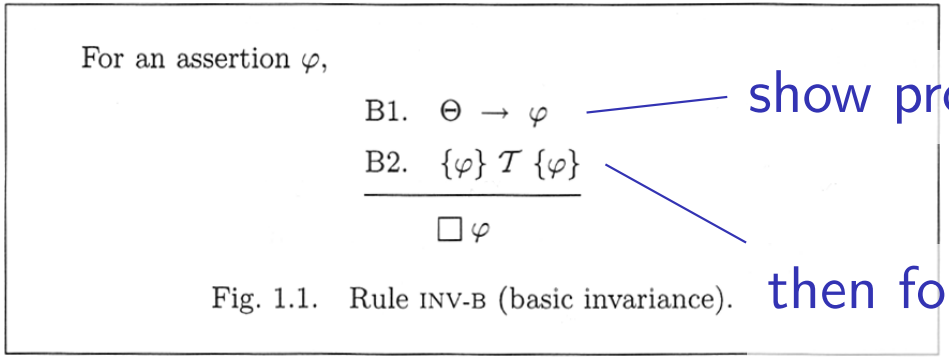
Invariants

$$A \Vdash (I \rightarrow)\ P = \forall s \in \mathsf{reachable}\ A\ I.\ P\ s$$

Step Invariants

$$A \Vdash\!\!\!\equiv (I \rightarrow)\ P = \forall a.\ I\ a \longrightarrow (\forall s \in \mathsf{reachable}\ A\ I.\ \forall s'.\ (s,\ a,\ s') \in \mathsf{trans}\ A \longrightarrow P\ (s,\ a,\ s'))$$

Showing invariance by induction (Floyd/Manna/Pnueli)



For an assertion φ,

$$\text{B1.}\quad \Theta \rightarrow \varphi$$
$$\text{B2.}\quad \{\varphi\}\ \mathcal{T}\ \{\varphi\}$$
$$\hline$$
$$\Box\varphi$$

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

show property of initial states

then for every (control) transition:

- assume the property of the pre state (φ)
- show the property of the post state (φ')

Showing invariance by induction (Floyd/Manna/Pnueli)

For an assertion φ,

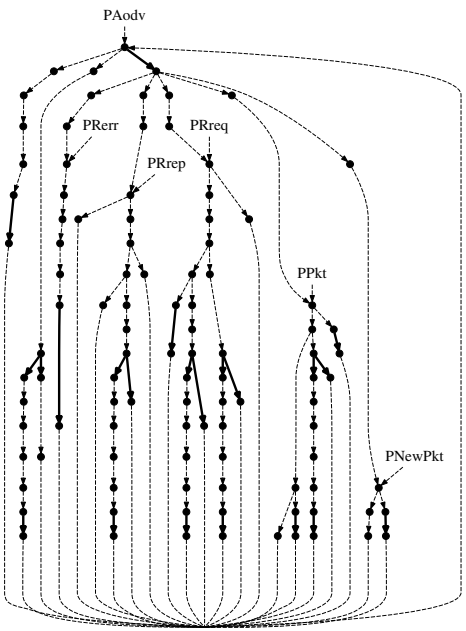
$$\frac{\text{B1. } \Theta \rightarrow \varphi \qquad \text{B2. } \{\varphi\}\ \mathcal{T}\ \{\varphi\}}{\Box\varphi}$$

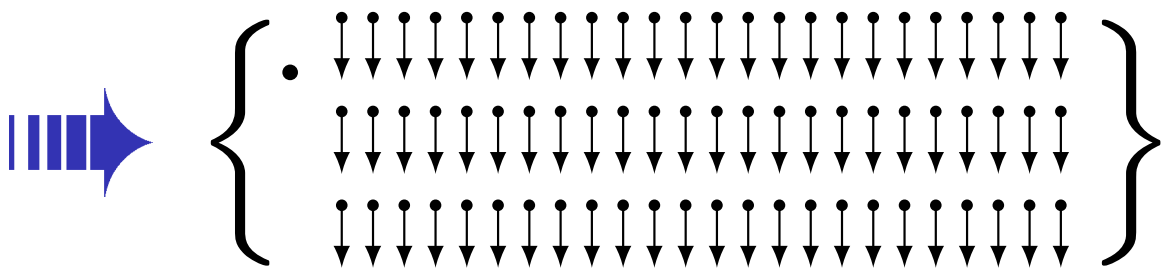
Fig. 1.1. Rule INV-B (basic invariance).

show property of initial states

then for every (control) transition:

- assume the property of the pre state (φ)
- show the property of the post state (φ')





Showing invariance by induction (Floyd/Manna/Pnueli)

For an assertion φ,

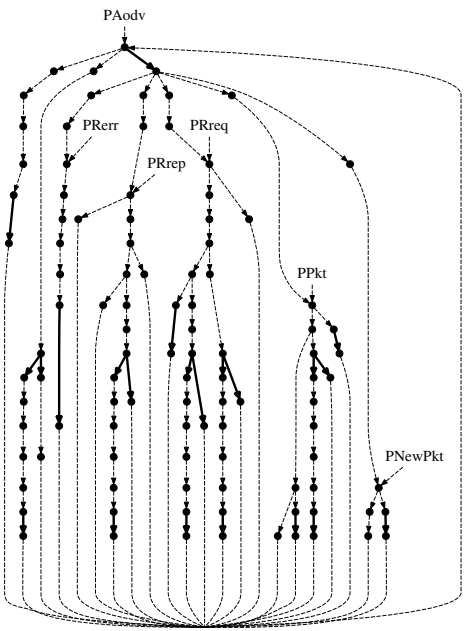
$$\frac{\text{B1. } \Theta \rightarrow \varphi \qquad \text{B2. } \{\varphi\}\ \mathcal{T}\ \{\varphi\}}{\Box\varphi}$$

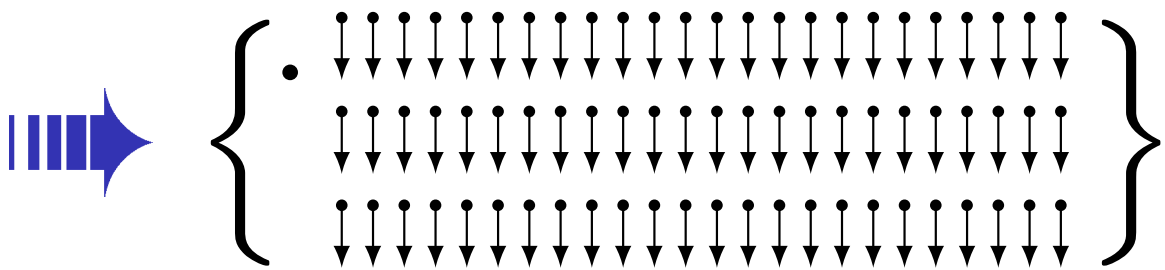
Fig. 1.1. Rule INV-B (basic invariance).

show property of initial states

then for every (control) transition:

- assume the property of the pre state (φ)
- show the property of the post state (φ')





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

Invariant and proof: example

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

$$\mathsf{ptoy\ i} \models \mathsf{onl}\ \Gamma_{\mathsf{Toy}}\ (\lambda(\xi, \mathsf{l}).\ \mathsf{l} \in \{\mathsf{PToy\text{-}:7..PToy\text{-}:9}\} \longrightarrow \mathsf{no}\ \xi \leqslant \mathsf{num}\ \xi)$$

Queueing

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

$$
\text{qmsg} = (|\text{init} = \{([], \Gamma_{\text{QMSG}}\ \text{Qmsg})\},\ \text{trans} = \text{seqp-sos}\ \Gamma_{\text{QMSG}}|)
$$

Queueing

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

$$\text{qmsg} = (|\text{init} = \{([], \Gamma_{\text{QMSG}}\ \text{Qmsg})\},\ \text{trans} = \text{seqp-sos}\ \Gamma_{\text{QMSG}}|)$$

$$\text{ptoy i} \langle\langle\ \text{qmsg}$$

$$\text{A} \langle\langle\ \text{B} = (|\text{init} = \text{init A} \times \text{init B},\ \text{trans} = \text{parp-sos (trans A) (trans B)}|)$$

Queueing

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

$$\text{qmsg} = (|\text{init} = \{([], \Gamma_{\text{QMSG}}\ \text{Qmsg})\},\ \text{trans} = \text{seqp-sos}\ \Gamma_{\text{QMSG}}|)$$

$$\text{ptoy i} \langle\langle\ \text{qmsg}$$

$$\text{A} \langle\langle\ \text{B} = (|\text{init} = \text{init A} \times \text{init B},\ \text{trans} = \text{parp-sos (trans A) (trans B)}|)$$

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

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

Networks of nodes

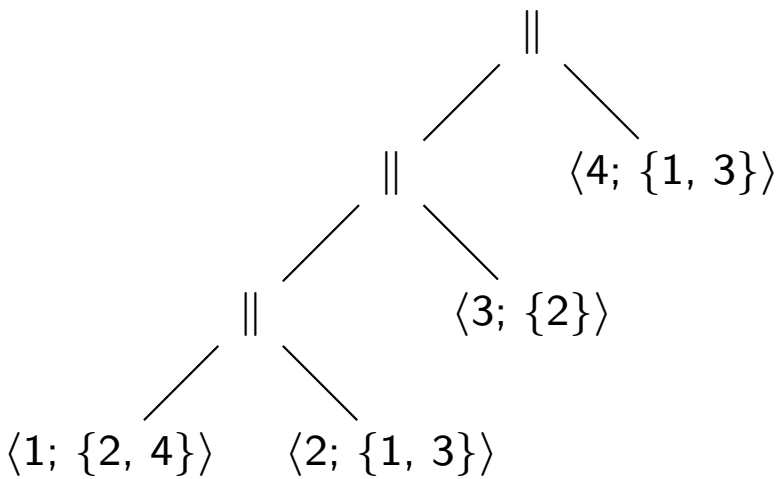
1. Annotate each node with its address and initial neighbours

$$\langle \mathsf{i} : \mathsf{np} : R_i \rangle = (\!|\mathsf{init} = \{\mathsf{s}^{\mathsf{i}}_{R_i} \mid \mathsf{s} \in \mathsf{init\ np}\},\ \mathsf{trans} = \text{node-sos}\ (\mathsf{trans\ np})|\!)$$

$$\frac{(\mathsf{s},\ \mathsf{groupcast\ D\ m},\ \mathsf{s'}) \in T_A}{(\mathsf{s}^{\mathsf{i}}_{R},\ (R \cap D){:}*\mathsf{cast}(\mathsf{m}),\ \mathsf{s'}^{\mathsf{i}}_{R}) \in \text{node-sos}\ T_A}$$

$$(\mathsf{s}^{\mathsf{i}}_{R},\ \mathsf{connect}(\mathsf{i},\ \mathsf{i'}),\ \mathsf{s}^{\mathsf{i}}_{R \cup \{\mathsf{i'}\}}) \in \text{node-sos}\ T_A$$

2. Express a network of nodes as an (arbitrary) term



$$\begin{aligned}
\mathsf{pnet\ np}\ \langle \mathsf{i};\ R_i \rangle &= \langle \mathsf{i} : \mathsf{np\ i} : R_i \rangle \\
\mathsf{pnet\ np}\ (\Psi_1 \parallel \Psi_2) &= (\!|\mathsf{init} = \{\mathsf{s}_1 \amalg \mathsf{s}_2 \mid \mathsf{s}_1 \in \mathsf{init}\ (\mathsf{pnet\ np}\ \Psi_1) \wedge \mathsf{s}_2 \in \mathsf{init}\ (\mathsf{pnet\ np}\ \Psi_2)\}, \\
&\qquad \mathsf{trans} = \text{pnet-sos}\ (\mathsf{trans}\ (\mathsf{pnet\ np}\ \Psi_1))\ (\mathsf{trans}\ (\mathsf{pnet\ np}\ \Psi_2))|\!)
\end{aligned}$$

Networks of nodes

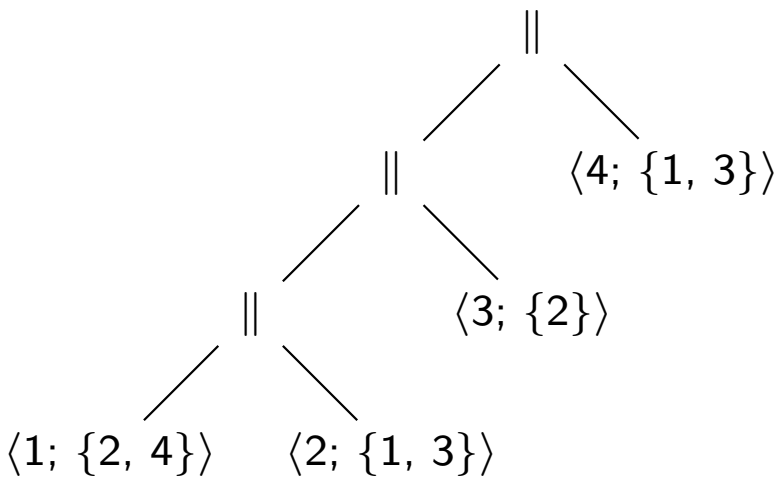
1. Annotate each node with its address and initial neighbours

$$\langle \mathsf{i} : \mathsf{np} : \mathsf{R}_i \rangle = (\!|\mathsf{init} = \{\mathsf{s}^{\mathsf{i}}_{\mathsf{R}_i} \mid \mathsf{s} \in \mathsf{init\ np}\},\ \mathsf{trans} = \mathsf{node\text{-}sos}\ (\mathsf{trans\ np})|\!)$$

$$\frac{(\mathsf{s},\ \mathsf{groupcast\ D\ m},\ \mathsf{s'}) \in \mathsf{T}_A}{(\mathsf{s}^{\mathsf{i}}_{\mathsf{R}},\ (\mathsf{R} \cap \mathsf{D}){:}*\mathsf{cast}(\mathsf{m}),\ \mathsf{s'}^{\mathsf{i}}_{\mathsf{R}}) \in \mathsf{node\text{-}sos}\ \mathsf{T}_A}$$

$$(\mathsf{s}^{\mathsf{i}}_{\mathsf{R}},\ \mathsf{connect}(\mathsf{i},\ \mathsf{i'}),\ \mathsf{s}^{\mathsf{i}}_{\mathsf{R} \cup \{\mathsf{i'}\}}) \in \mathsf{node\text{-}sos}\ \mathsf{T}_A$$

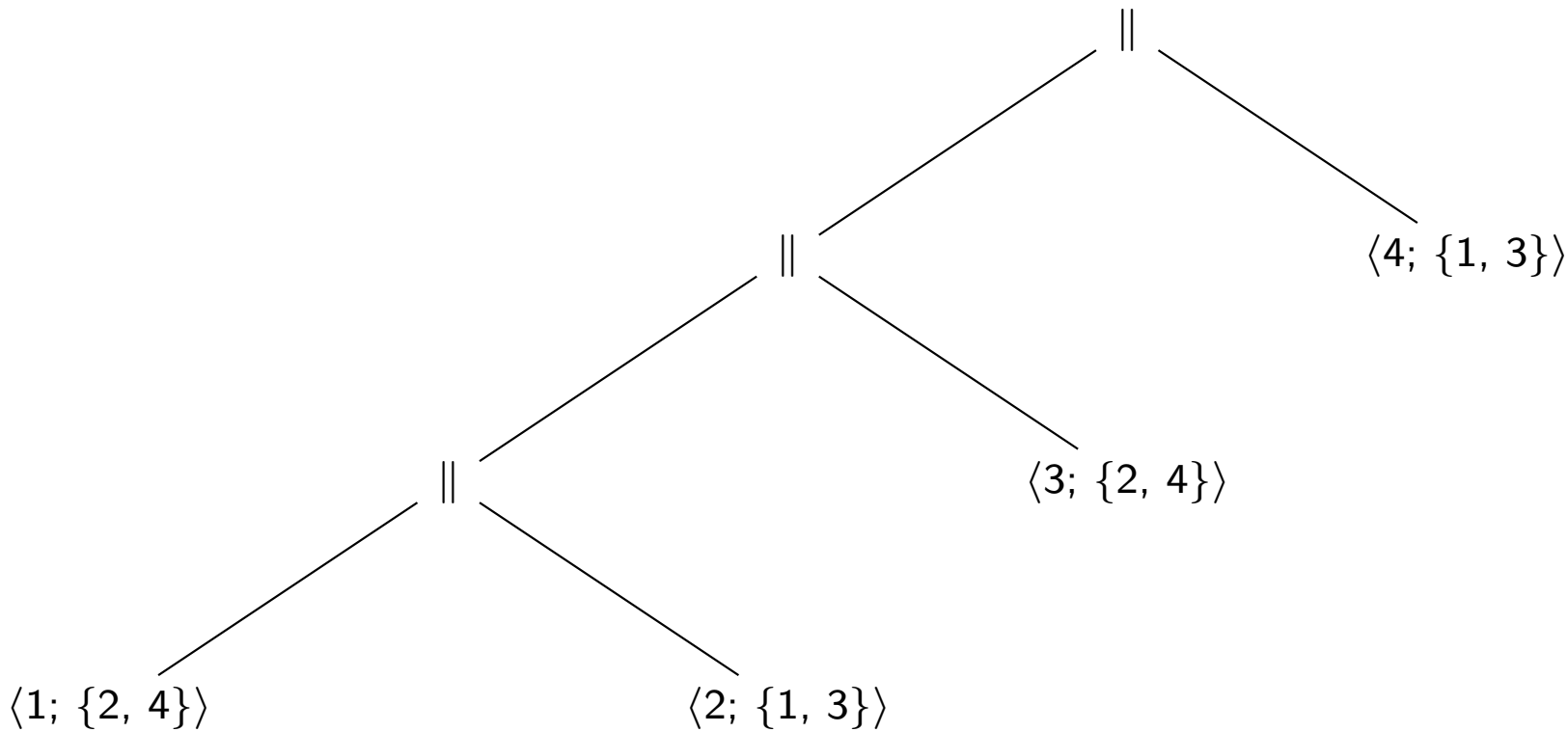
2. Express a network of nodes as an (arbitrary) term



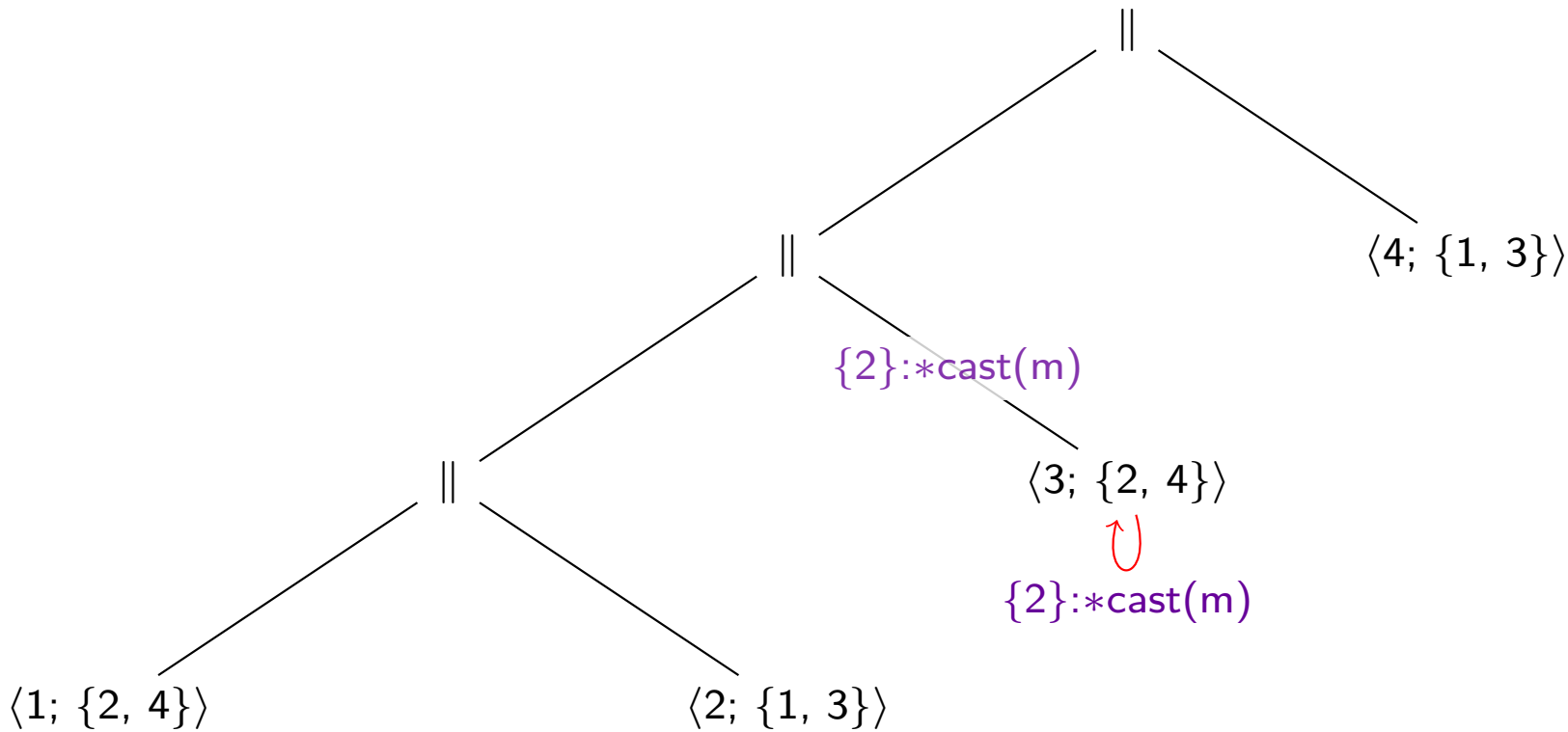
- No associativity / commutativity / bisimulation
- Not even induction over N nodes.
- Induction over arbitrary Ψ.

$$\begin{aligned}
\mathsf{pnet\ np}\ \langle \mathsf{i};\ \mathsf{R}_i \rangle &= \langle \mathsf{i} : \mathsf{np\ i} : \mathsf{R}_i \rangle \\
\mathsf{pnet\ np}\ (\Psi_1 \parallel \Psi_2) &= (\!|\mathsf{init} = \{\mathsf{s}_1 \amalg \mathsf{s}_2 \mid \mathsf{s}_1 \in \mathsf{init}\ (\mathsf{pnet\ np}\ \Psi_1) \wedge \mathsf{s}_2 \in \mathsf{init}\ (\mathsf{pnet\ np}\ \Psi_2)\}, \\
&\qquad \mathsf{trans} = \mathsf{pnet\text{-}sos}\ (\mathsf{trans}\ (\mathsf{pnet\ np}\ \Psi_1))\ (\mathsf{trans}\ (\mathsf{pnet\ np}\ \Psi_2))|\!)
\end{aligned}$$

Network communication

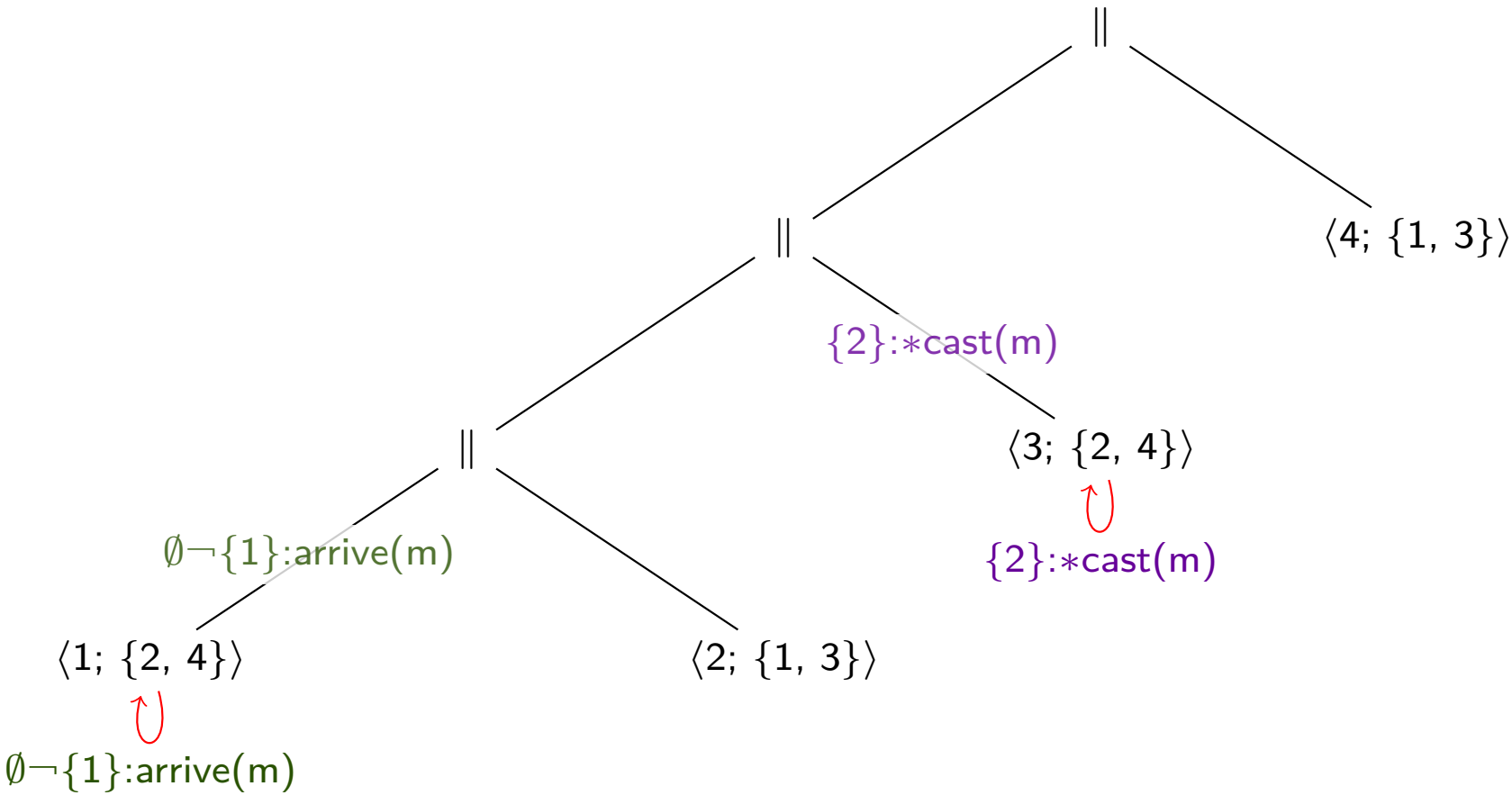


Network communication



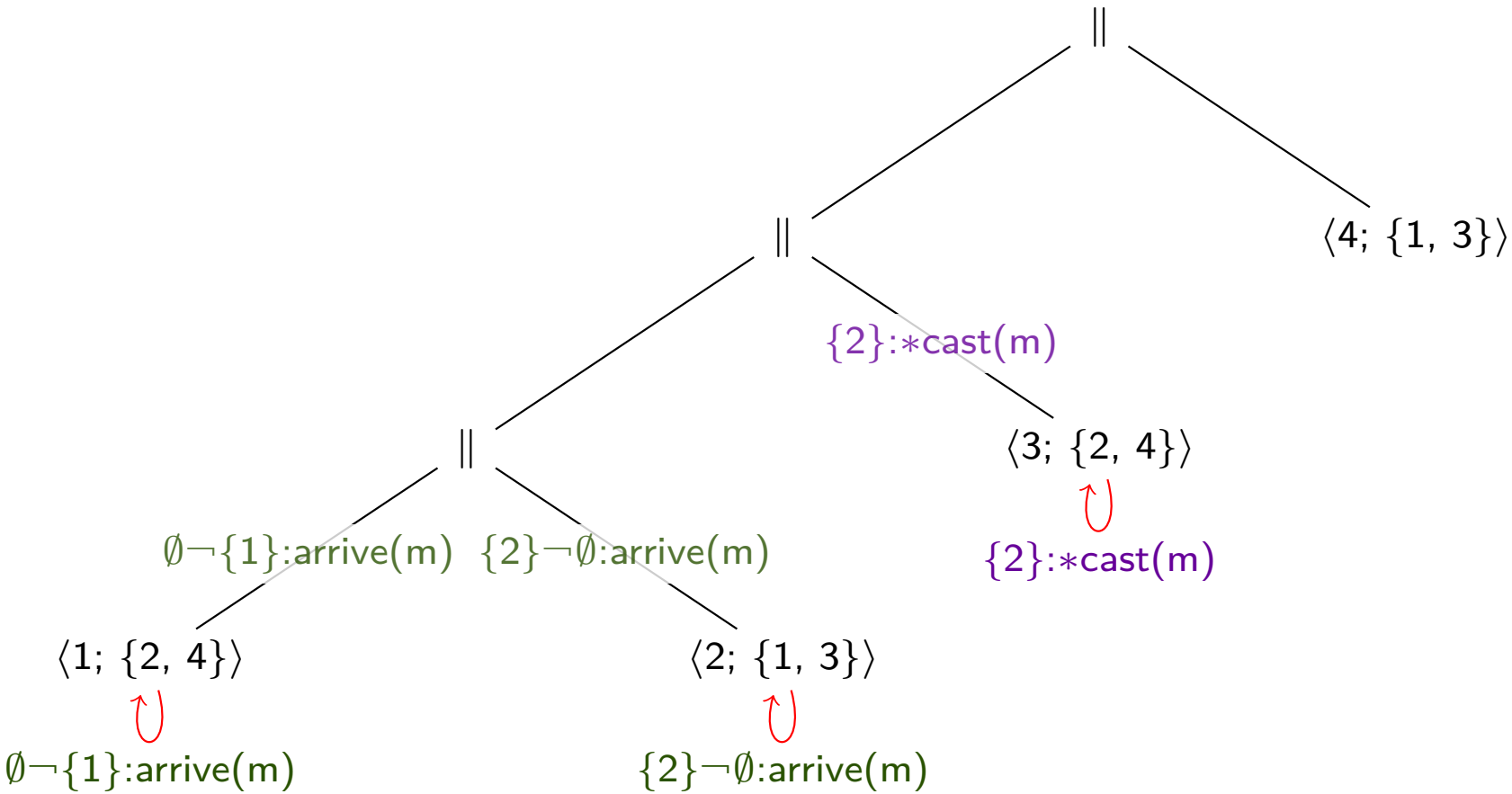
$$\frac{(s,\ \text{unicast dst m},\ s') \in T_A \qquad \text{dst} \in R}{(s_R^i,\ \{\text{dst}\}{:}{*}\text{cast(m)},\ s'^i_R) \in \text{node-sos } T_A}$$

Network communication



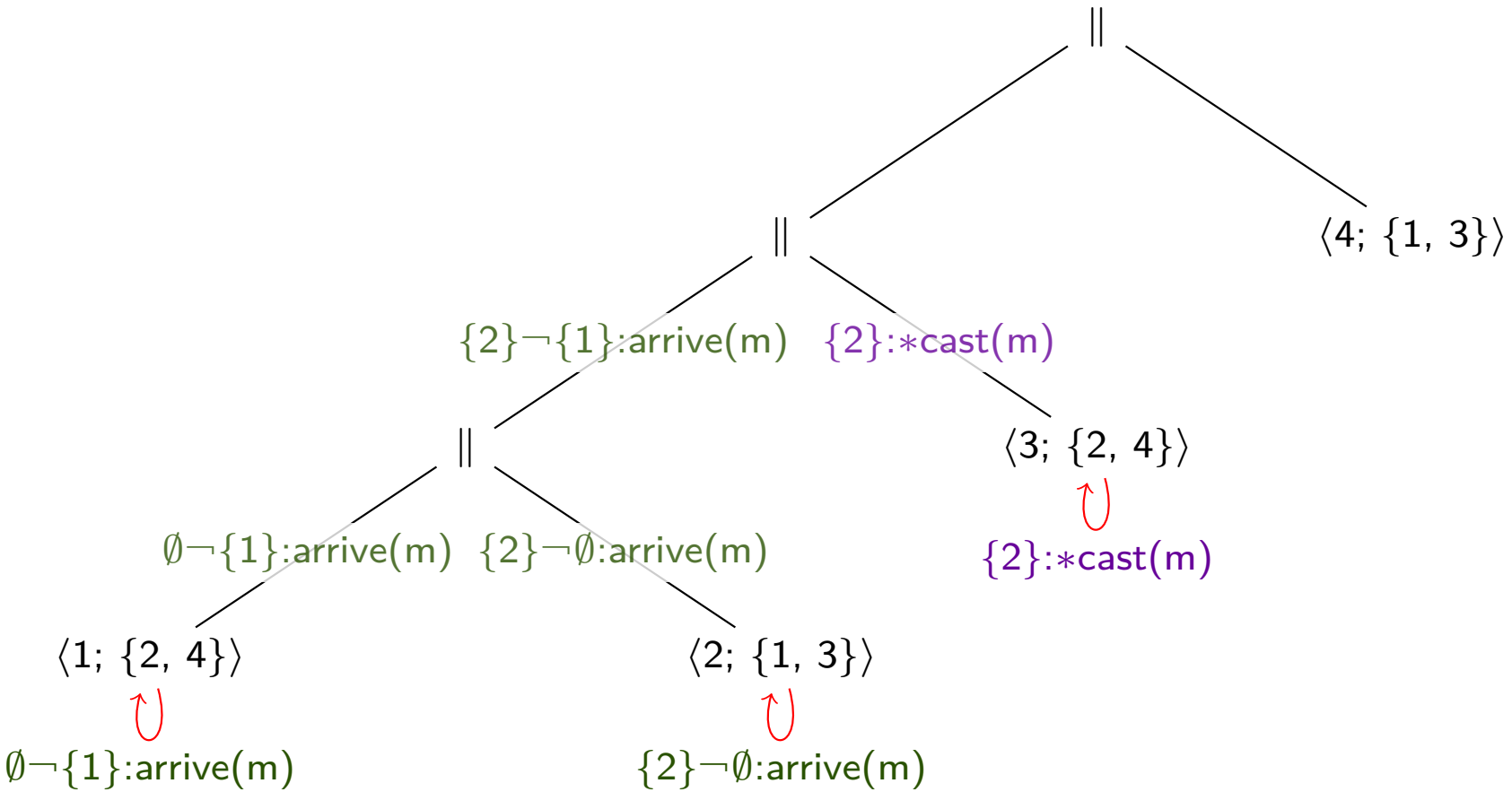
$$(s_R^i,\ \emptyset\neg\{i\}\text{:arrive(m)},\ s_R^i) \in \text{node-sos } T_A$$

Network communication



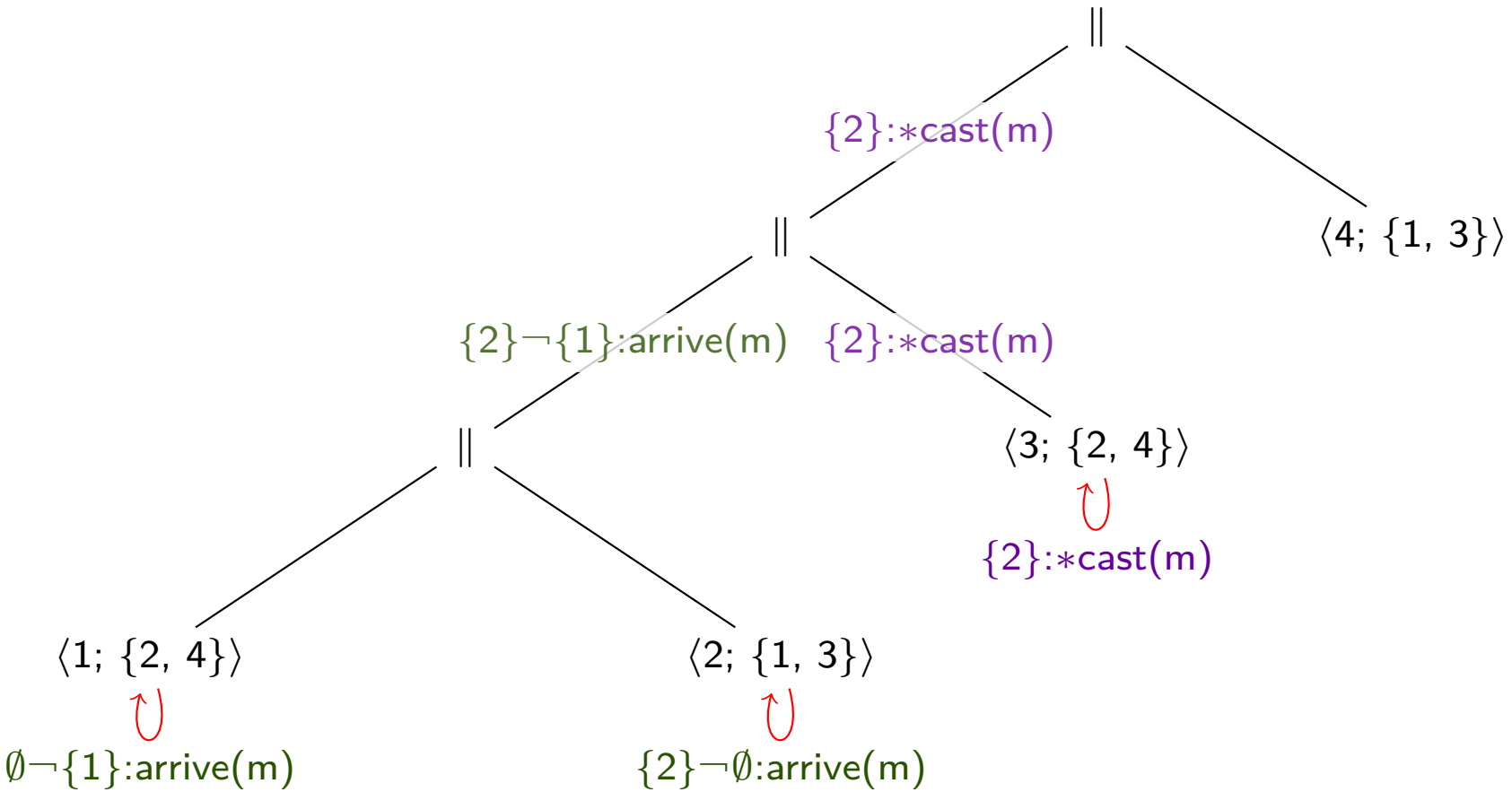
$$\frac{(s,\ \text{receive m},\ s') \in T_A}{(s_R^i,\ \{i\}\neg\emptyset\text{:arrive(m)},\ s'^{\,i}_R) \in \text{node-sos } T_A}$$

Network communication



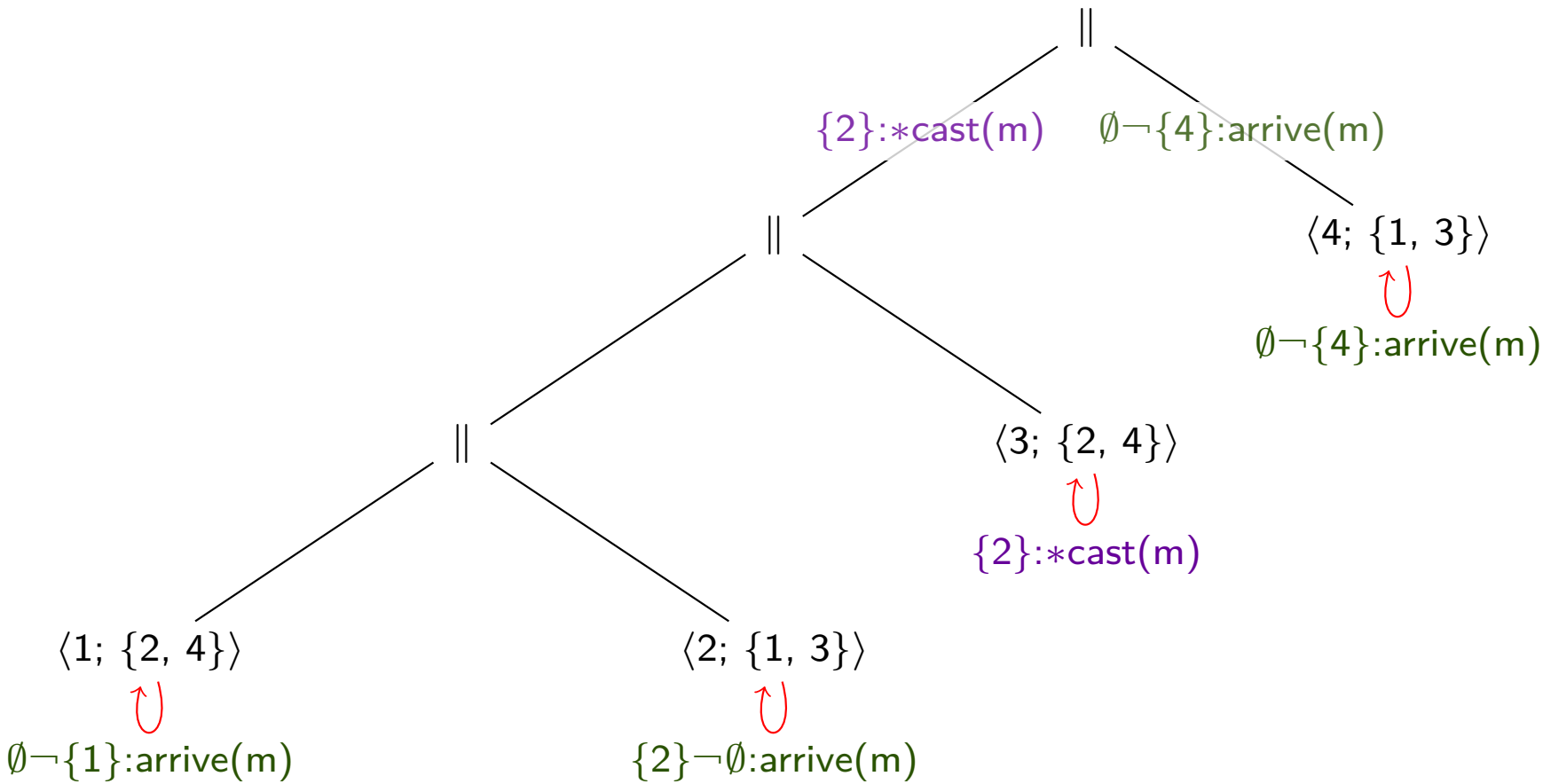
$$\frac{(s,\ H\neg K{:}\mathsf{arrive}(m),\ s') \in \mathsf{T}_A \qquad (t,\ H'\neg K'{:}\mathsf{arrive}(m),\ t') \in \mathsf{T}_B}{(s \parallel t,\ (H \cup H')\neg(K \cup K'){:}\mathsf{arrive}(m),\ s' \parallel t') \in \mathsf{pnet\text{-}sos}\ \mathsf{T}_A\ \mathsf{T}_B}$$

Network communication



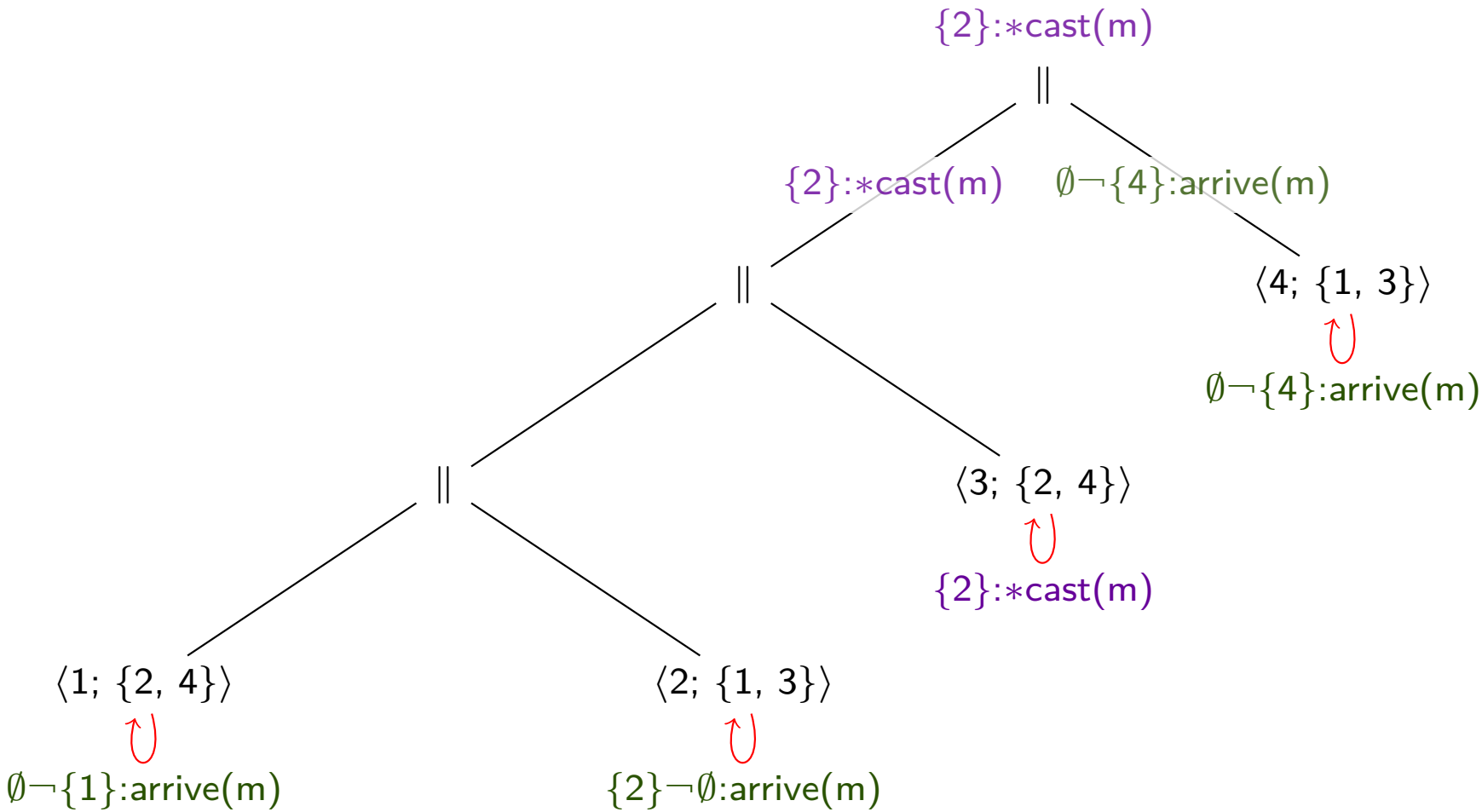
$$\frac{(s,\ H\neg K{:}\mathsf{arrive}(m),\ s') \in T_A \qquad (t,\ R{:}*\mathsf{cast}(m),\ t') \in T_B \qquad H \subseteq R \qquad K \cap R = \emptyset}{(s \parallel t,\ R{:}*\mathsf{cast}(m),\ s' \parallel t') \in \mathsf{pnet\text{-}sos}\ T_A\ T_B}$$

Network communication



$$(s_R^i,\ \emptyset\neg\{i\}\text{:arrive(m)},\ s_R^i) \in \text{node-sos}\ T_A$$

Network communication

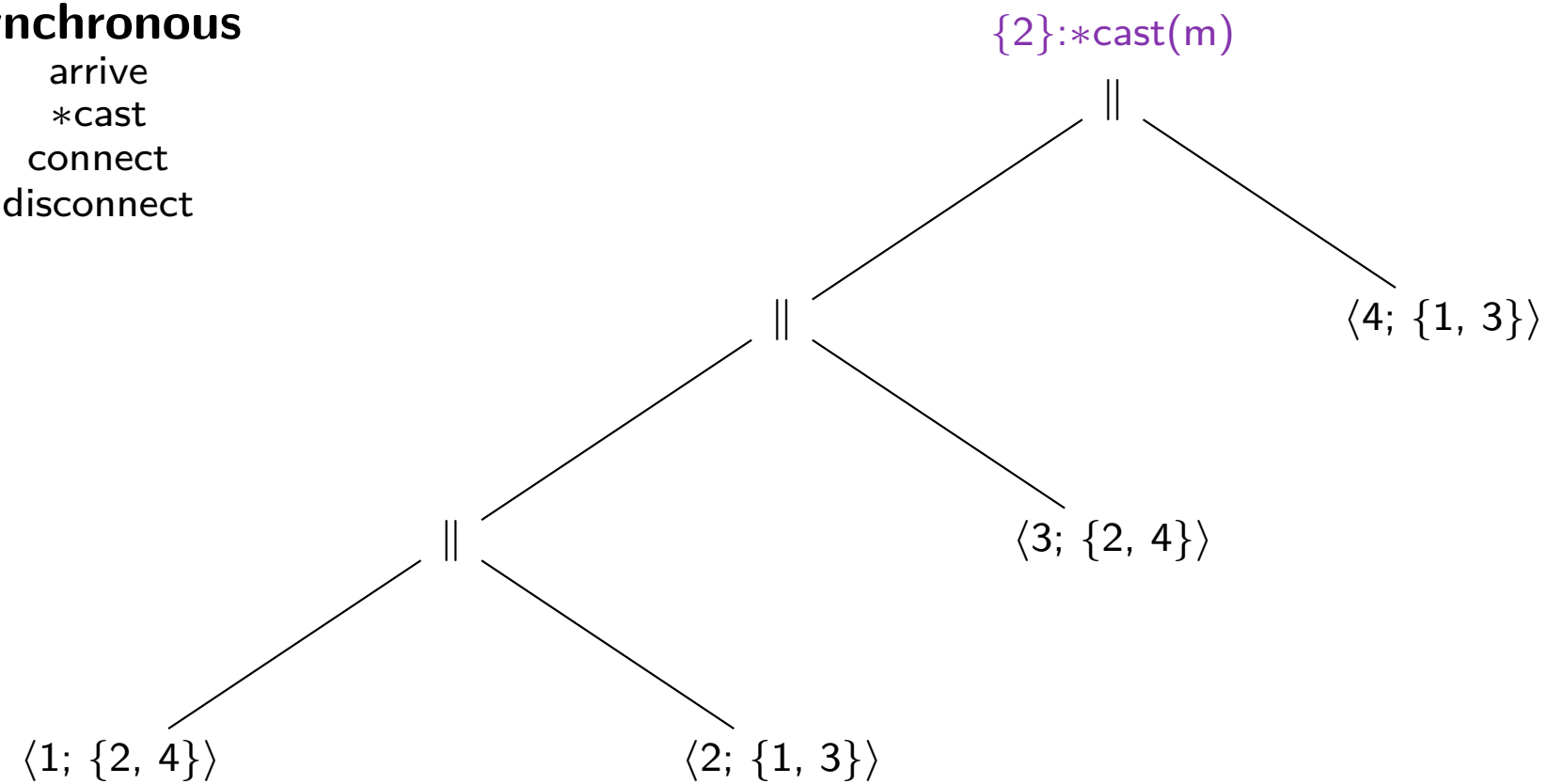


$$\frac{(s,\ R{:}{*}\mathsf{cast}(m),\ s') \in T_A \qquad (t,\ H\neg K{:}\mathsf{arrive}(m),\ t') \in T_B \qquad H \subseteq R \qquad K \cap R = \emptyset}{(s \parallel t,\ R{:}{*}\mathsf{cast}(m),\ s' \parallel t') \in \text{pnet-sos}\ T_A\ T_B}$$

Network communication

synchronous

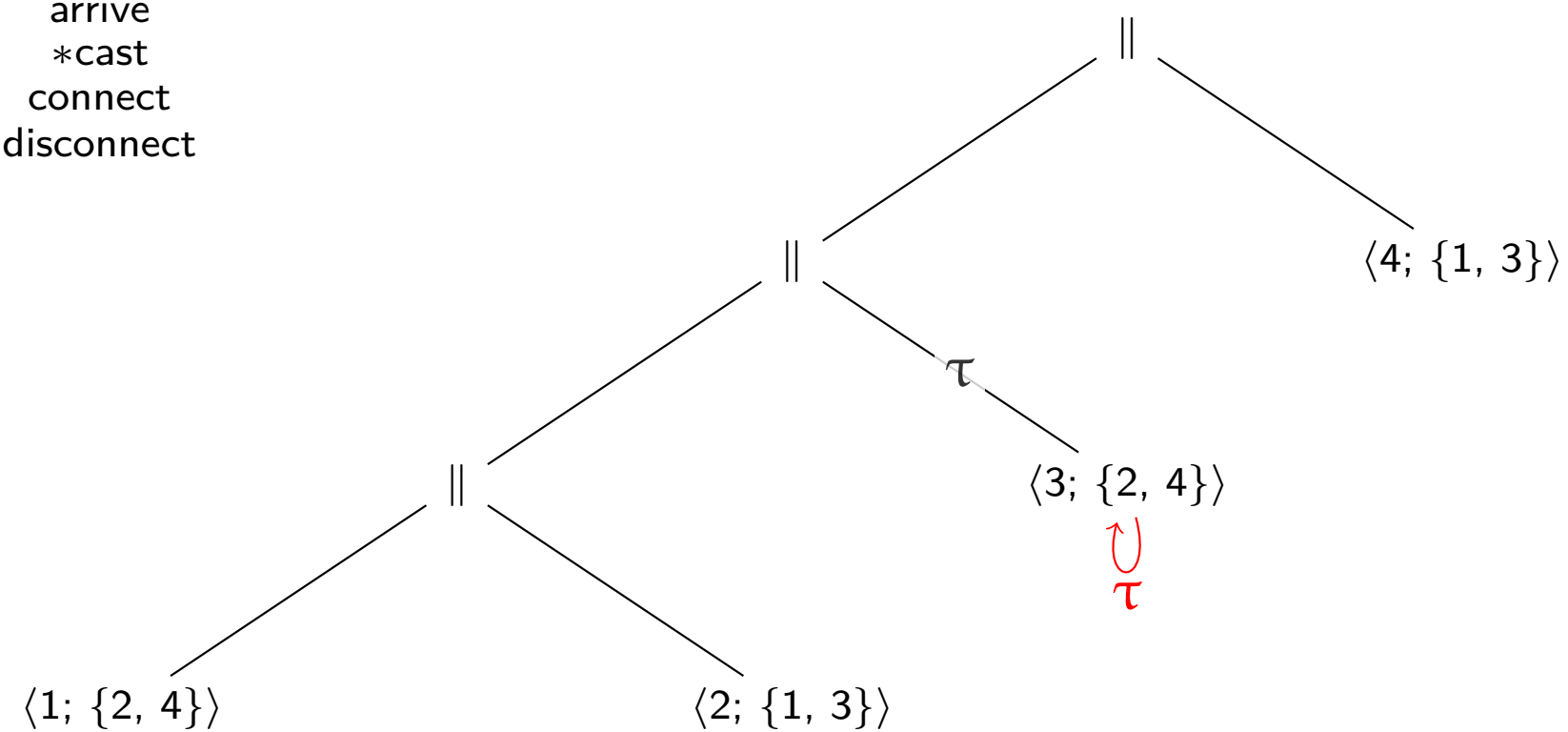
arrive
$*$cast
connect
disconnect



Network communication

synchronous

arrive
∗cast
connect
disconnect

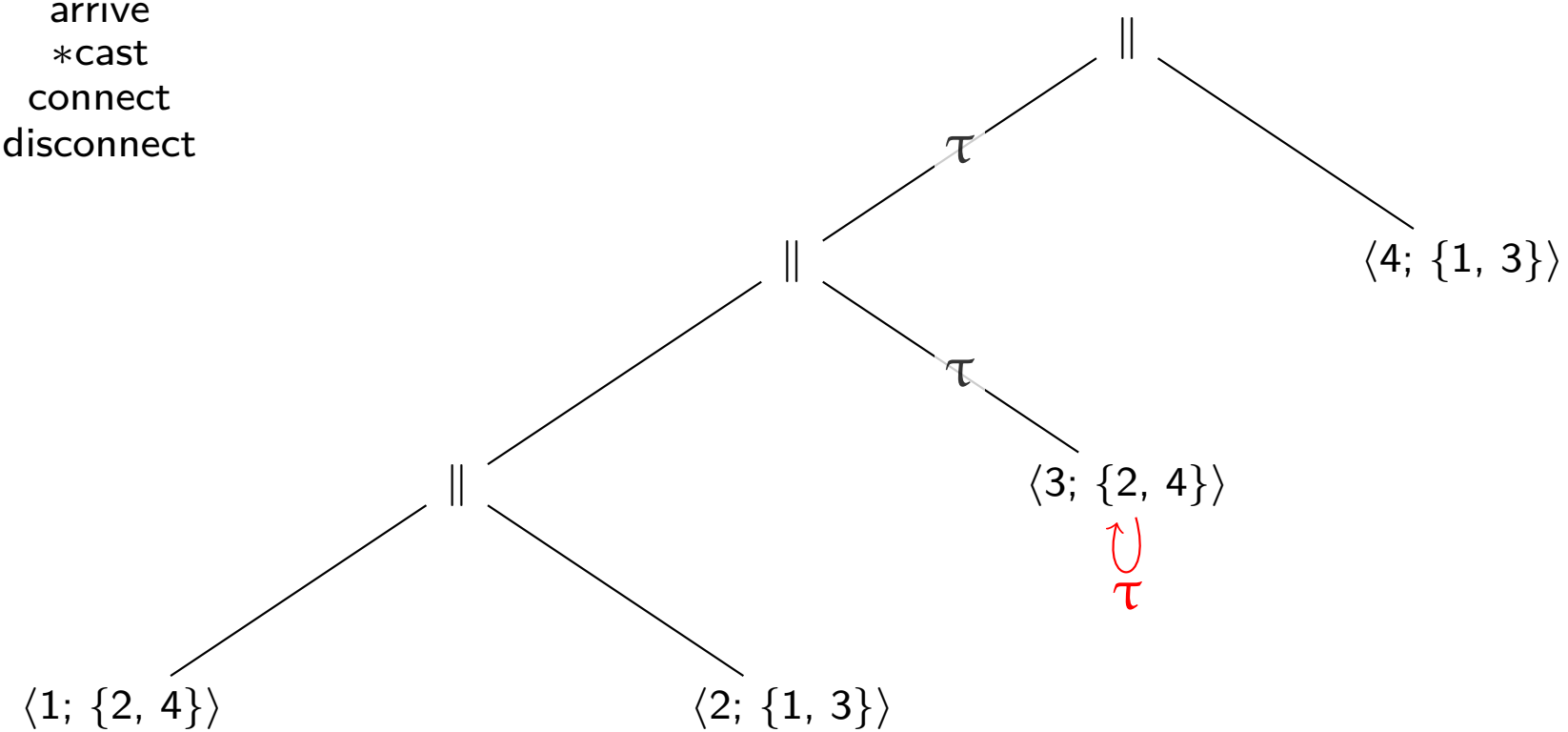


$$\frac{(s,\ \tau,\ s') \in \mathsf{T}_A}{(s^{\mathsf{i}}_{\mathsf{R}},\ \tau,\ s'^{\mathsf{i}}_{\mathsf{R}}) \in \text{node-sos}\ \mathsf{T}_A}$$

Network communication

synchronous

arrive
∗cast
connect
disconnect

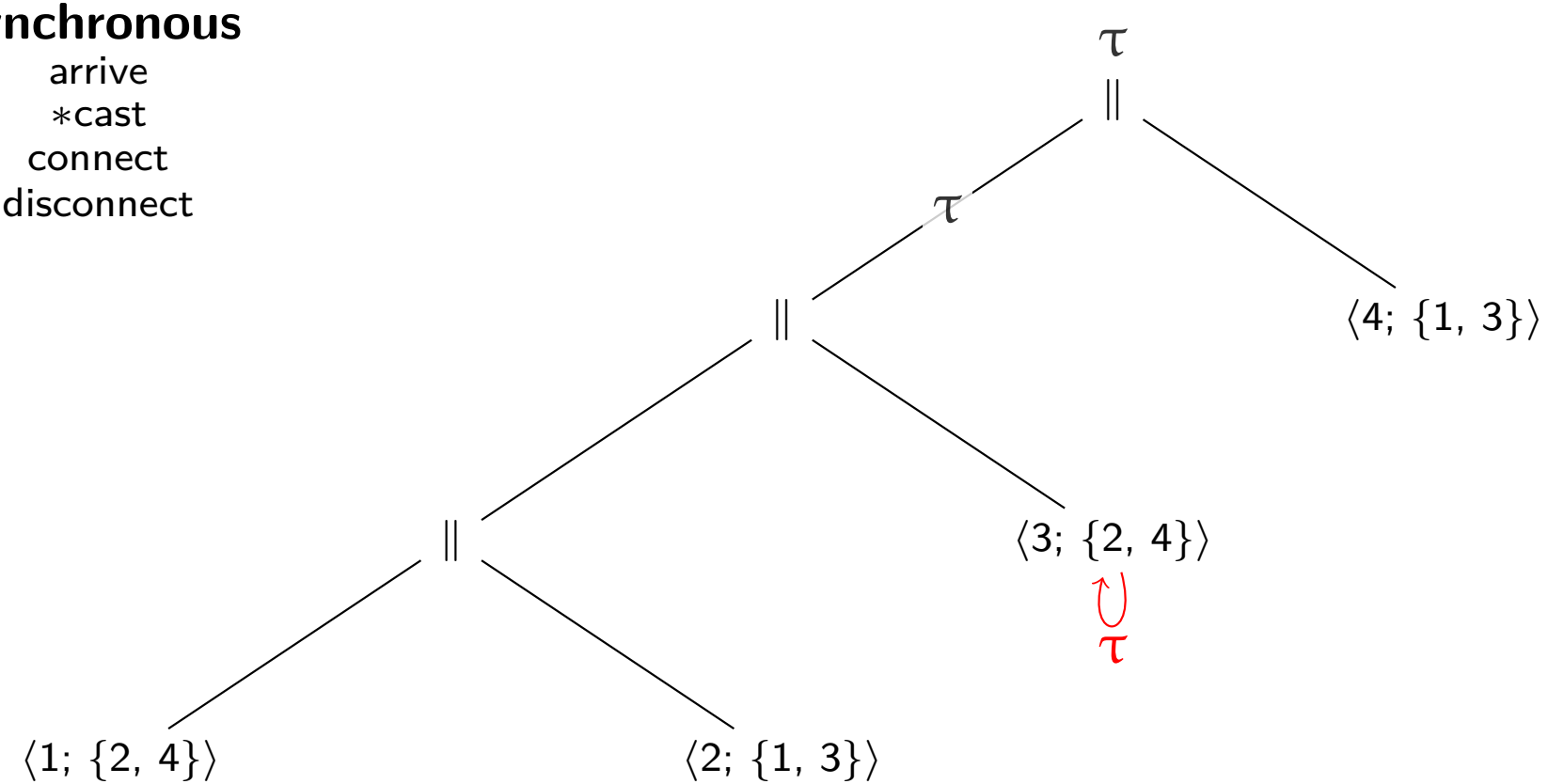


$$\frac{(t,\ \tau,\ t') \in T_B}{(s \parallel t,\ \tau,\ s \parallel t') \in \text{pnet-sos}\ T_A\ T_B}$$

Network communication

synchronous

arrive
∗cast
connect
disconnect



$$\frac{(s,\ \tau,\ s') \in T_A}{(s \parallel t,\ \tau,\ s' \parallel t) \in \text{pnet-sos}\ T_A\ T_B}$$

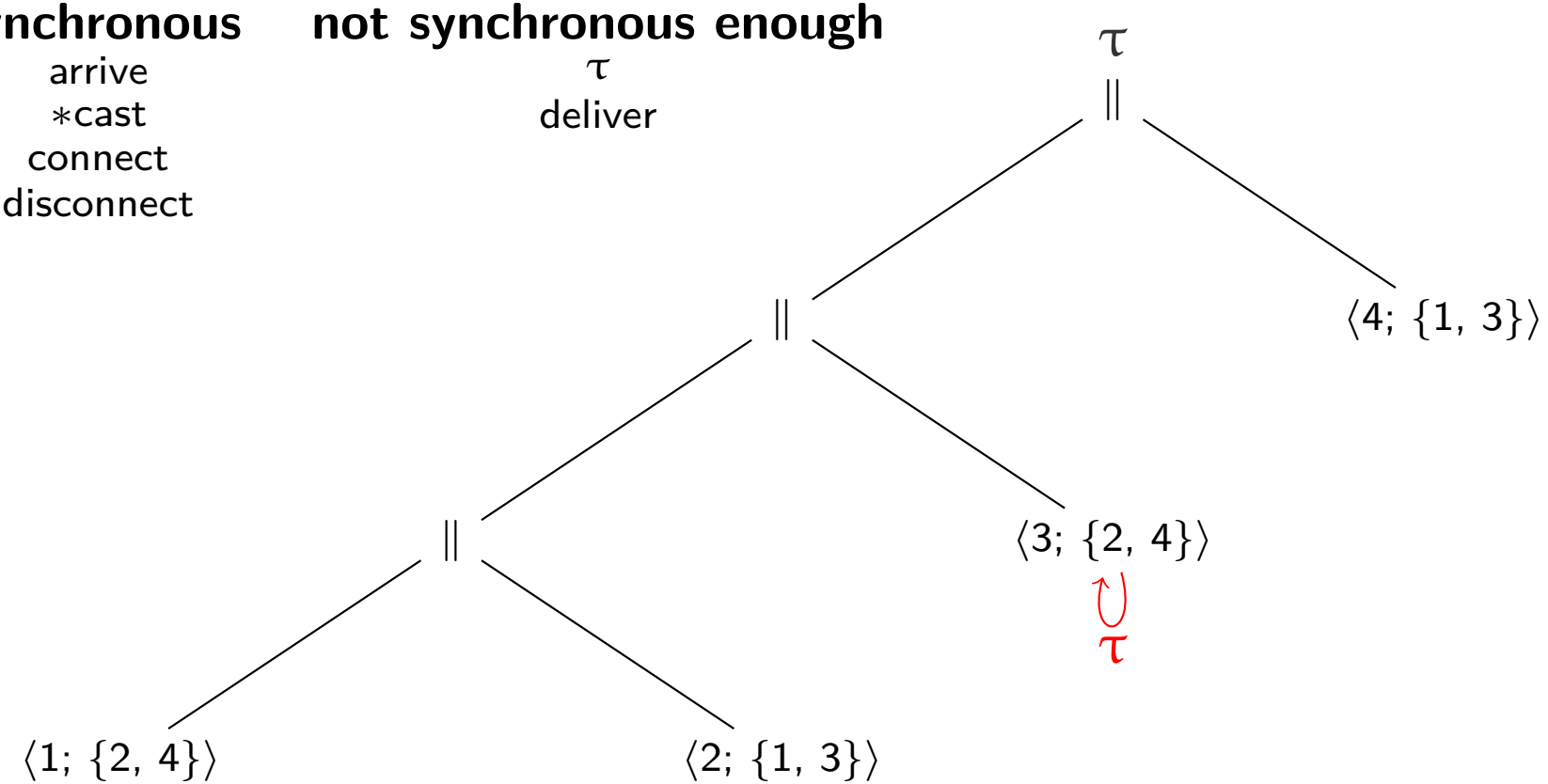
Network communication

synchronous

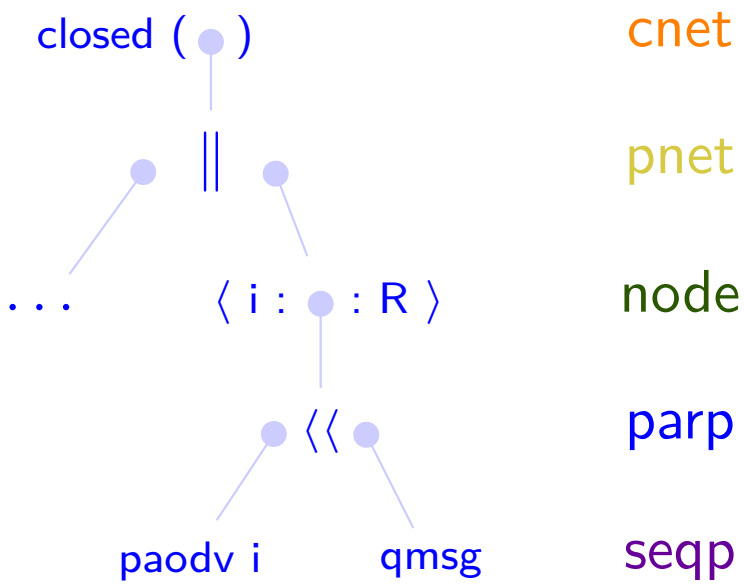
arrive
$*$cast
connect
disconnect

not synchronous enough

τ
deliver



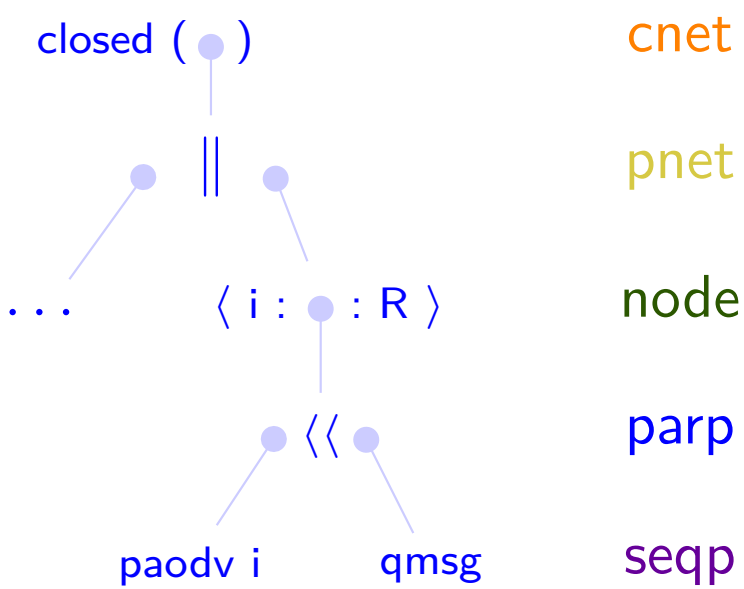
Mechanization of AWN



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

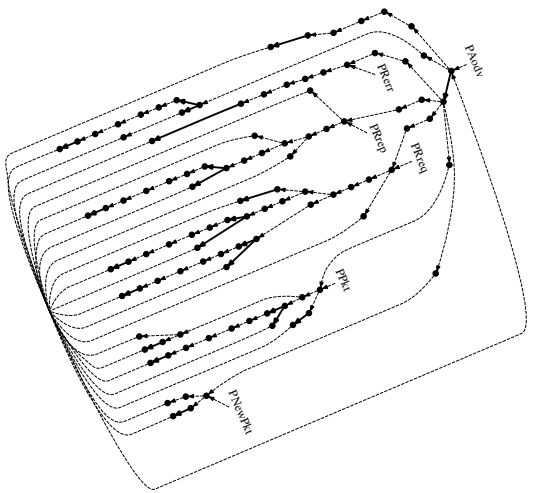
- Model all as automata
  (initial states and transitions)

Mechanization of AWN



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

- Model all as automata
  (initial states and transitions)



Network invariants

$$\text{ptoy i} \Vdash \text{onl } \Gamma_{\text{Toy}}\ (\lambda(\xi, \text{l}).\ \text{l} \in \{\text{PToy-:7..PToy-:9}\} \longrightarrow \text{no } \xi \leqslant \text{num } \xi)$$

$$\text{closed (pnet } (\lambda \text{i. ptoy i} \langle\langle \text{ qmsg)} \Psi) \Vdash \text{netglobal } (\lambda\sigma.\ \forall \text{i. no } (\sigma \text{ i}) \leqslant \text{no } (\sigma \text{ (nhid } (\sigma \text{ i}))))$$

- How to express such properties at the level of sequential nodes? (where we can verify them with Floyd/Manna/Pneuli)
- How to 'lift' the results to the full network model?
- Pencil-and-paper proof
  - Very detailed invariant proofs over sequential processes.
  - Less formal reasoning for other levels.

An 'open model' of AWN

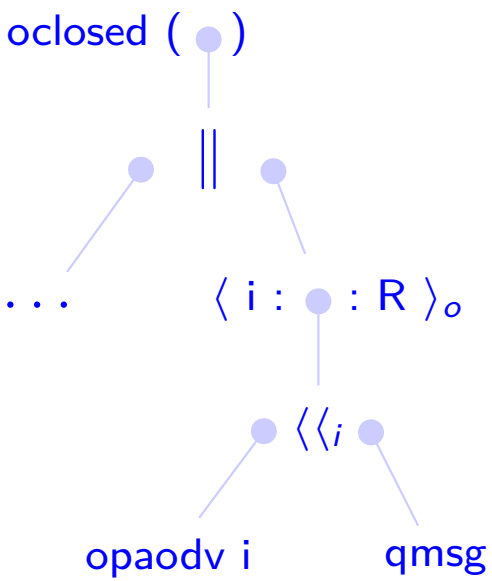
$$\xi :: \text{state}$$

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

Simple idea. Several technical consequences:

- Redefine all SOS rules.
- Redefine operators on automata.
- Redefine reachability.
- Lifting and transfer technicalities.

An ‘open model’ of AWN



ocnet

opnet

onode

oparp

oseqp

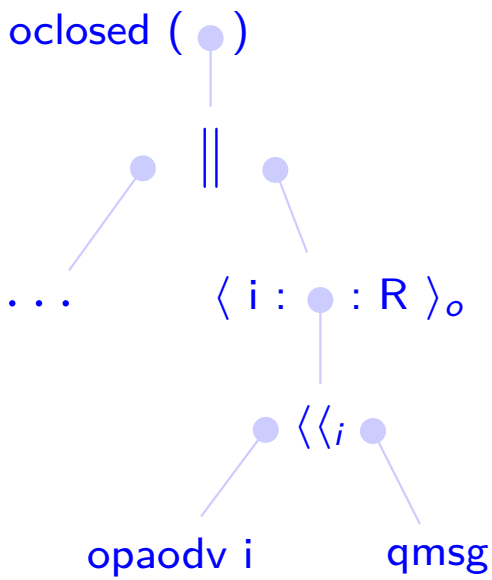
$\xi :: \text{state}$

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

Simple idea. Several technical consequences:

- Redefine all SOS rules.
- Redefine operators on automata.
- Redefine reachability.
- Lifting and transfer technicalities.

An ‘open model’ of AWN



ocnet

opnet

onode

oparp

oseqp

$\xi :: \text{state}$

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

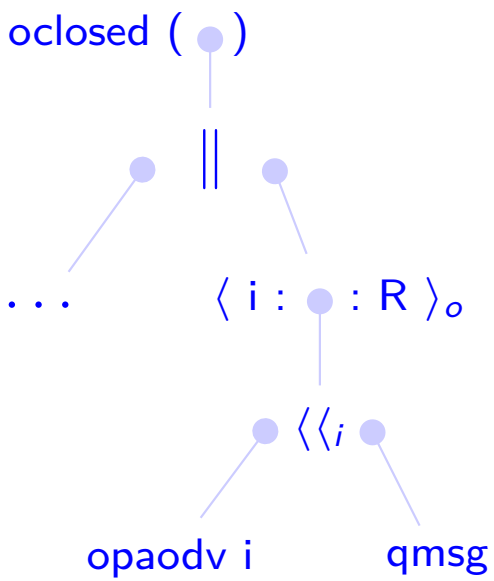
$$\text{optoy i} = (|\text{init} = \{(\text{toy-init}, \Gamma_{\text{Toy}}\ \text{PToy})\},\ \text{trans} = \textbf{o}\text{seqp-sos}\ \Gamma_{\text{Toy}}\ \text{i}|).$$

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

versus

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

An 'open model' of AWN



ocnet

opnet

onode

oparp

oseqp

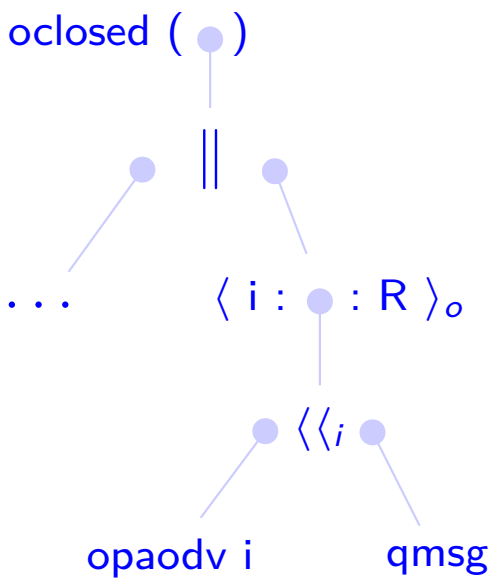
$\xi :: \text{state}$

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

$$\frac{((\sigma, s), \text{groupcast D m}, (\sigma', s')) \in T_A}{((\sigma, s^i_R), (R \cap D){:}*\text{cast}(m), (\sigma', s'^i_R)) \in \textbf{onode-sos}\ T_A}$$

$$\frac{((\sigma, s), \tau, (\sigma', s')) \in T_A \qquad \forall j \neq i.\ \sigma' j = \sigma j}{((\sigma, s^i_R), \tau, (\sigma', s'^i_R)) \in \textbf{onode-sos}\ T_A}$$

An 'open model' of AWN



ocnet

opnet

onode

oparp

oseqp

$\xi :: \text{state}$

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

$$\begin{aligned}
\text{opnet onp } \langle \text{i; } R_i \rangle &= \langle \text{i : onp i : } R_i \rangle_o \\
\text{opnet onp } (\Psi_1 \parallel \Psi_2) &= (\!| \text{init} = \{(\sigma, s_1 \shuffle s_2) \mid (\sigma, s_1) \in \text{init (opnet onp } \Psi_1) \\
&\qquad \wedge (\sigma, s_2) \in \text{init (opnet onp } \Psi_2) \\
&\qquad \wedge \text{net-ips } s_1 \cap \text{net-ips } s_2 = \emptyset\}, \\
&\qquad \text{trans} = \text{opnet-sos (trans (opnet onp } \Psi_1)) \text{ (trans (opnet onp } \Psi_2))|
\end{aligned}$$

$$\frac{((\sigma, s), H\neg K\text{:arrive(m)}, (\sigma', s')) \in T_A \qquad ((\sigma, t), H'\neg K'\text{:arrive(m)}, (\sigma', t')) \in T_B}{((\sigma, s \shuffle t), (H \cup H')\neg(K \cup K')\text{:arrive(m)}, (\sigma', s' \shuffle t')) \in \text{opnet-sos } T_A\ T_B}$$

Open invariants

Open reachability

$$\frac{(\sigma,\ s) \in \mathsf{init}\ A}{(\sigma,\ s) \in \mathsf{oreachable}\ A\ S\ U} \qquad \frac{(\sigma,\ s) \in \mathsf{oreachable}\ A\ S\ U \qquad U\ \sigma\ \sigma'}{(\sigma',\ s) \in \mathsf{oreachable}\ A\ S\ U}$$

$$\frac{(\sigma,\ s) \in \mathsf{oreachable}\ A\ S\ U \qquad ((\sigma,\ s),\ a,\ (\sigma',\ s')) \in \mathsf{trans}\ A \qquad S\ \sigma\ \sigma'\ a}{(\sigma',\ s') \in \mathsf{oreachable}\ A\ S\ U}$$

Open invariants

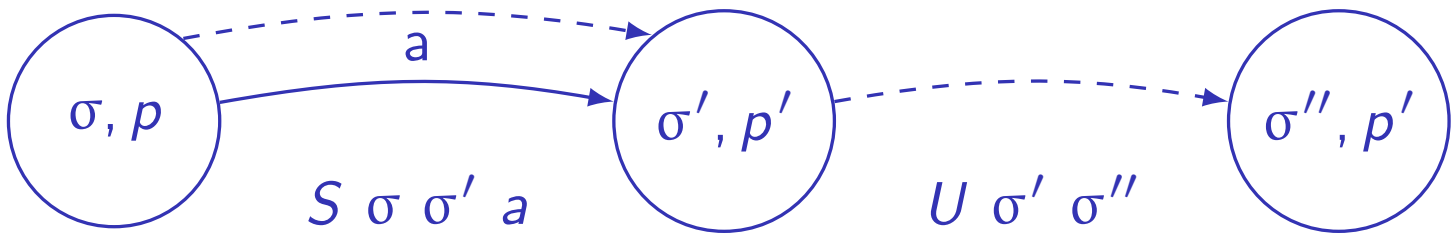
Open reachability

interleaving steps must satisfy U

$$\frac{(\sigma,\ s) \in \text{init A}}{(\sigma,\ s) \in \text{oreachable A S U}} \qquad \frac{(\sigma,\ s) \in \text{oreachable A S U} \qquad \text{U } \sigma\ \sigma'}{(\sigma',\ s) \in \text{oreachable A S U}}$$

$$\frac{(\sigma,\ s) \in \text{oreachable A S U} \qquad ((\sigma,\ s),\ a,\ (\sigma',\ s')) \in \text{trans A} \qquad \text{S } \sigma\ \sigma'\ a}{(\sigma',\ s') \in \text{oreachable A S U}}$$

'local' steps must satisfy S



Open invariants

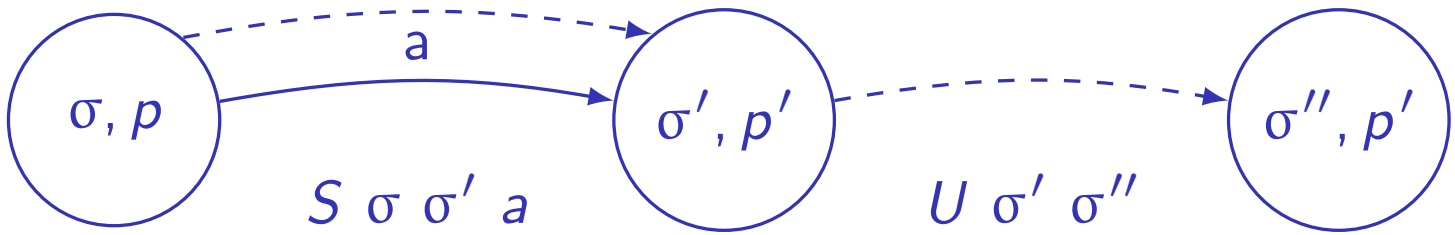
Open reachability

interleaving steps must satisfy U

$$\frac{(\sigma,\ s) \in \text{init A}}{(\sigma,\ s) \in \text{oreachable A S U}} \qquad \frac{(\sigma,\ s) \in \text{oreachable A S U} \qquad \text{U } \sigma\ \sigma'}{(\sigma',\ s) \in \text{oreachable A S U}}$$

$$\frac{(\sigma,\ s) \in \text{oreachable A S U} \qquad ((\sigma,\ s),\ a,\ (\sigma',\ s')) \in \text{trans A} \qquad \text{S } \sigma\ \sigma'\ a}{(\sigma',\ s') \in \text{oreachable A S U}}$$

'local' steps must satisfy S



other E N σ σ' = ∀ i. if i ∈ N then σ' i = σ i else E (σ i) (σ' i)

otherwith E N I σ σ' a = (∀ i. i ∉ N ⟶ E (σ i) (σ' i)) ∧ I σ a

Open invariants

Open reachability

interleaving steps must satisfy U

$$\frac{(\sigma, s) \in \text{init } A}{(\sigma, s) \in \text{oreachable } A\ S\ U} \qquad \frac{(\sigma, s) \in \text{oreachable } A\ S\ U \qquad U\ \sigma\ \sigma'}{(\sigma', s) \in \text{oreachable } A\ S\ U}$$

$$\frac{(\sigma, s) \in \text{oreachable } A\ S\ U \qquad ((\sigma, s), a, (\sigma', s')) \in \text{trans } A \qquad S\ \sigma\ \sigma'\ a}{(\sigma', s') \in \text{oreachable } A\ S\ U}$$

'local' steps must satisfy S

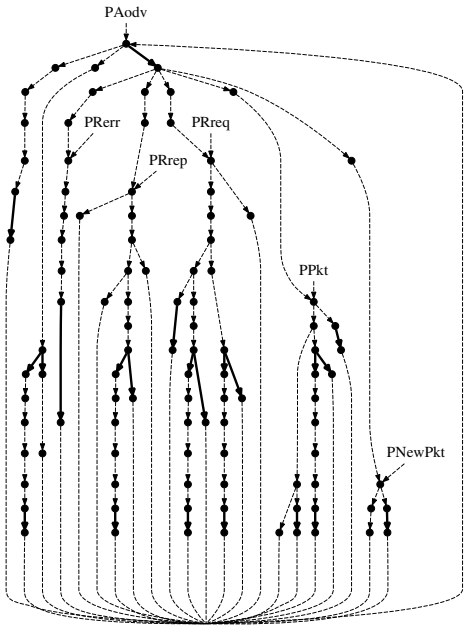
Open Invariants

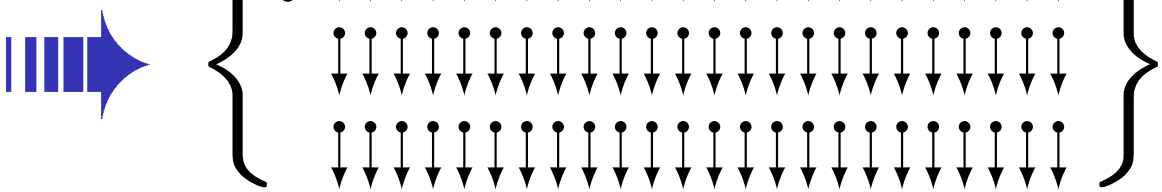
$$A \models (S, U \rightarrow)\ P = \forall s \in \text{oreachable } A\ S\ U.\ P\ s$$

Open Step Invariants

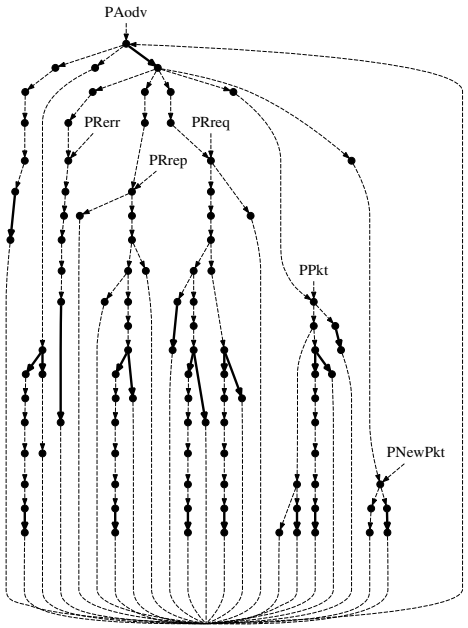
$$A \models\!\models (S, U \rightarrow)\ P =$$
$$\forall s \in \text{oreachable } A\ S\ U.\ \forall a\ s'.\ (s, a, s') \in \text{trans } A \wedge S\ (\text{fst } s)\ (\text{fst } s')\ a \longrightarrow P\ (s, a, s')$$

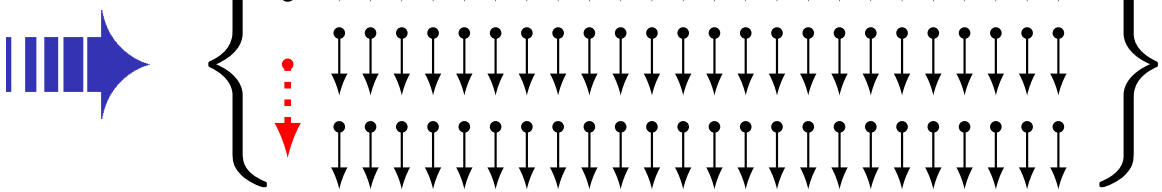
Showing invariance compositionally



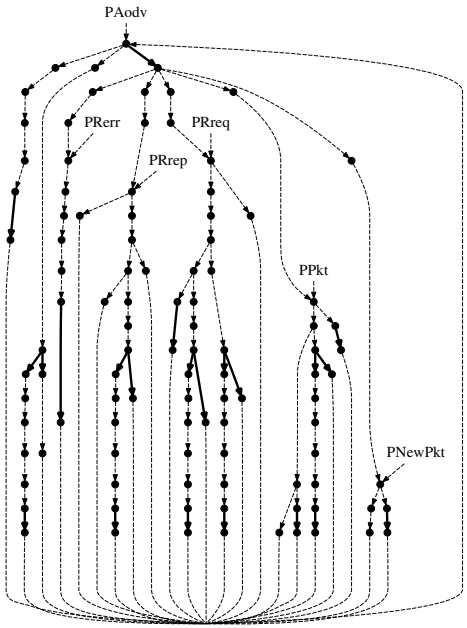


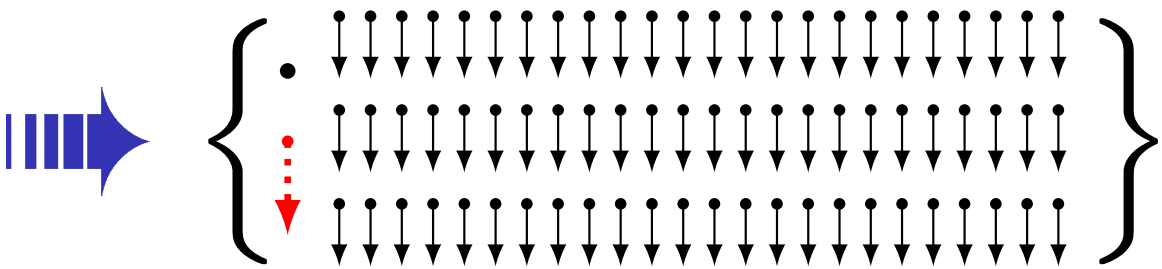
Showing invariance compositionally





Showing invariance compositionally





$$\frac{(\sigma,\ s) \in \text{init A}}{(\sigma,\ s) \in \text{oreachable A S U}} \qquad \frac{(\sigma,\ s) \in \text{oreachable A S U} \qquad \text{U } \sigma\ \sigma'}{(\sigma',\ s) \in \text{oreachable A S U}}$$

$$\frac{(\sigma,\ s) \in \text{oreachable A S U} \qquad ((\sigma,\ s),\ a,\ (\sigma',\ s')) \in \text{trans A} \qquad \text{S } \sigma\ \sigma'\ a}{(\sigma',\ s') \in \text{oreachable A S U}}$$

Lifting and transfer

cnet-sos

closed (pnet (λi. ptoy i ⟪ qmsg) Ψ) ⊨ P

ocnet-sos

pnet-sos

opnet-sos

node-sos

onode-sos

parp-sos

oparp-sos

seqp-sos

oseqp-sos

Lifting and transfer

cnet-sos

closed (pnet (λi. ptoy i $\langle\!\langle$ qmsg) Ψ) $\Vdash P$

ocnet-sos

pnet-sos

opnet-sos

node-sos

onode-sos

parp-sos

oparp-sos

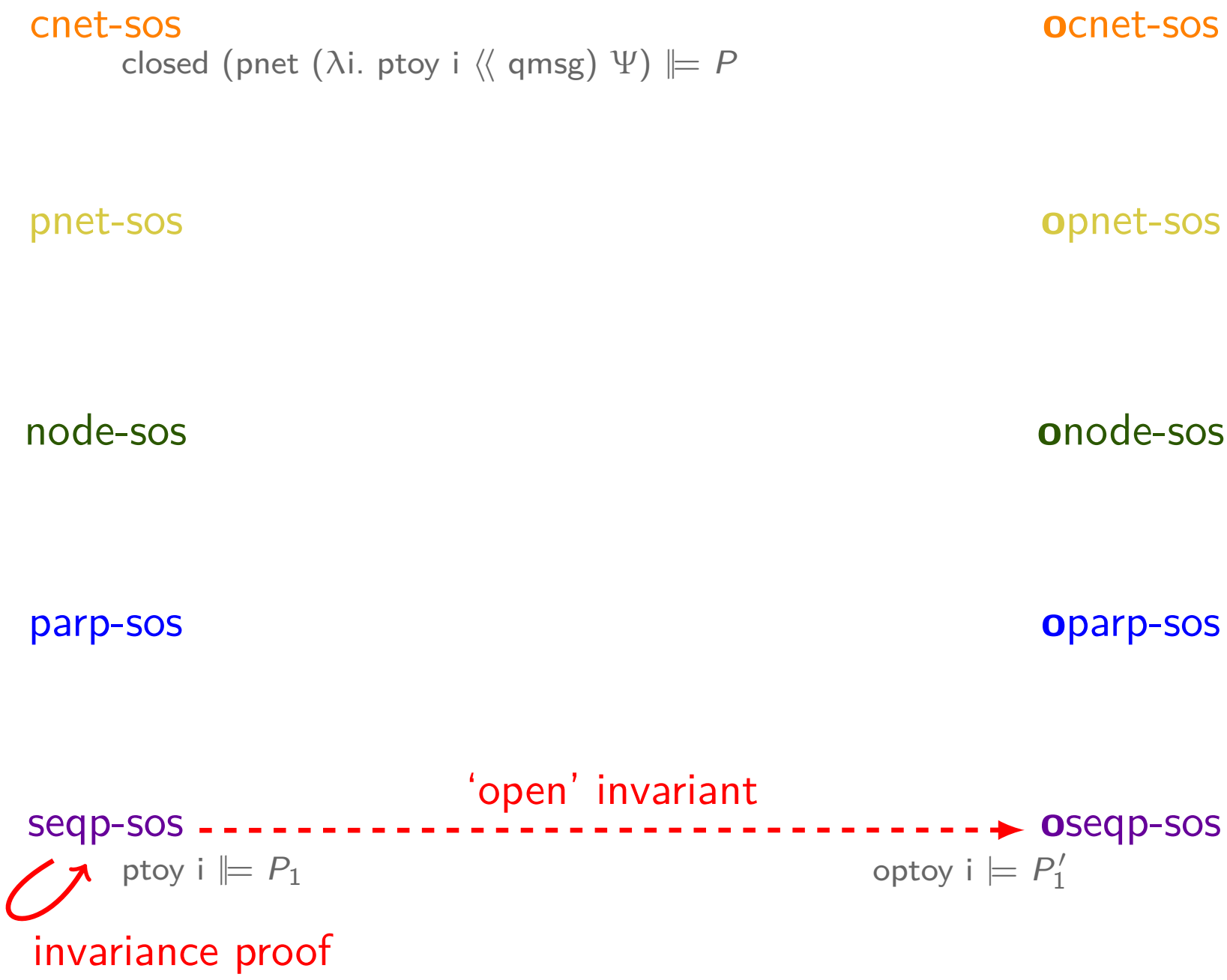
seqp-sos

ptoy i $\Vdash P_1$

oseqp-sos

invariance proof

Lifting and transfer



Lifting and transfer

cnet-sos

closed (pnet (λi. ptoy i ⟪ qmsg) Ψ) ⊨ P

ocnet-sos

pnet-sos

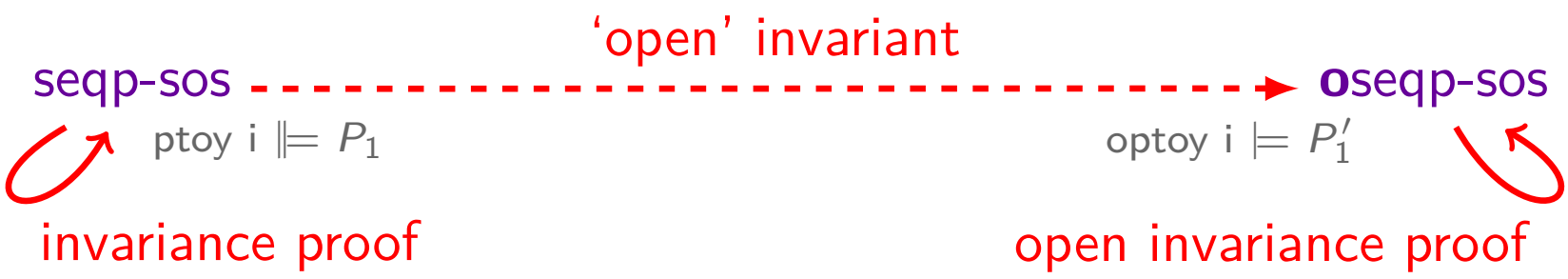
opnet-sos

node-sos

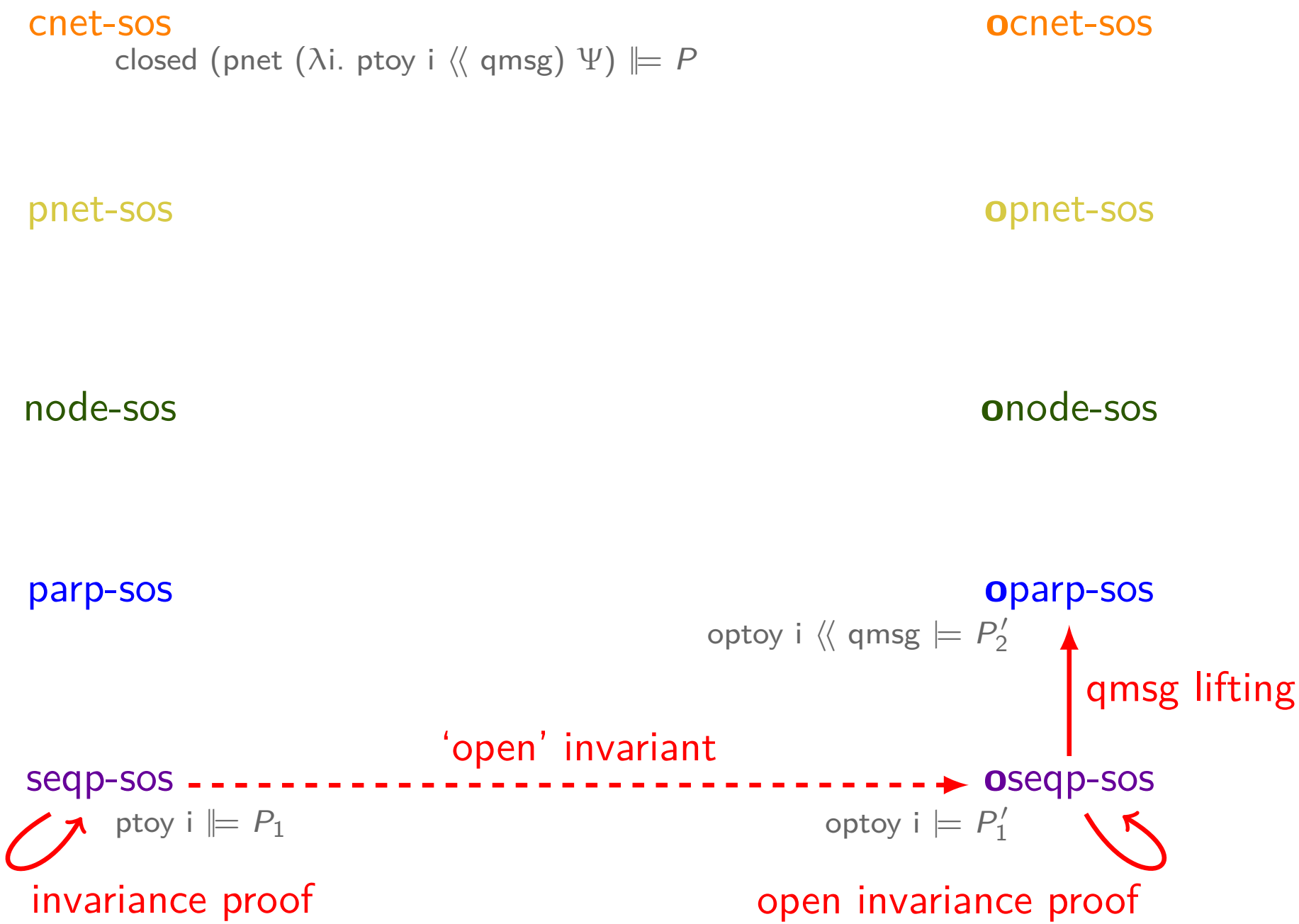
onode-sos

parp-sos

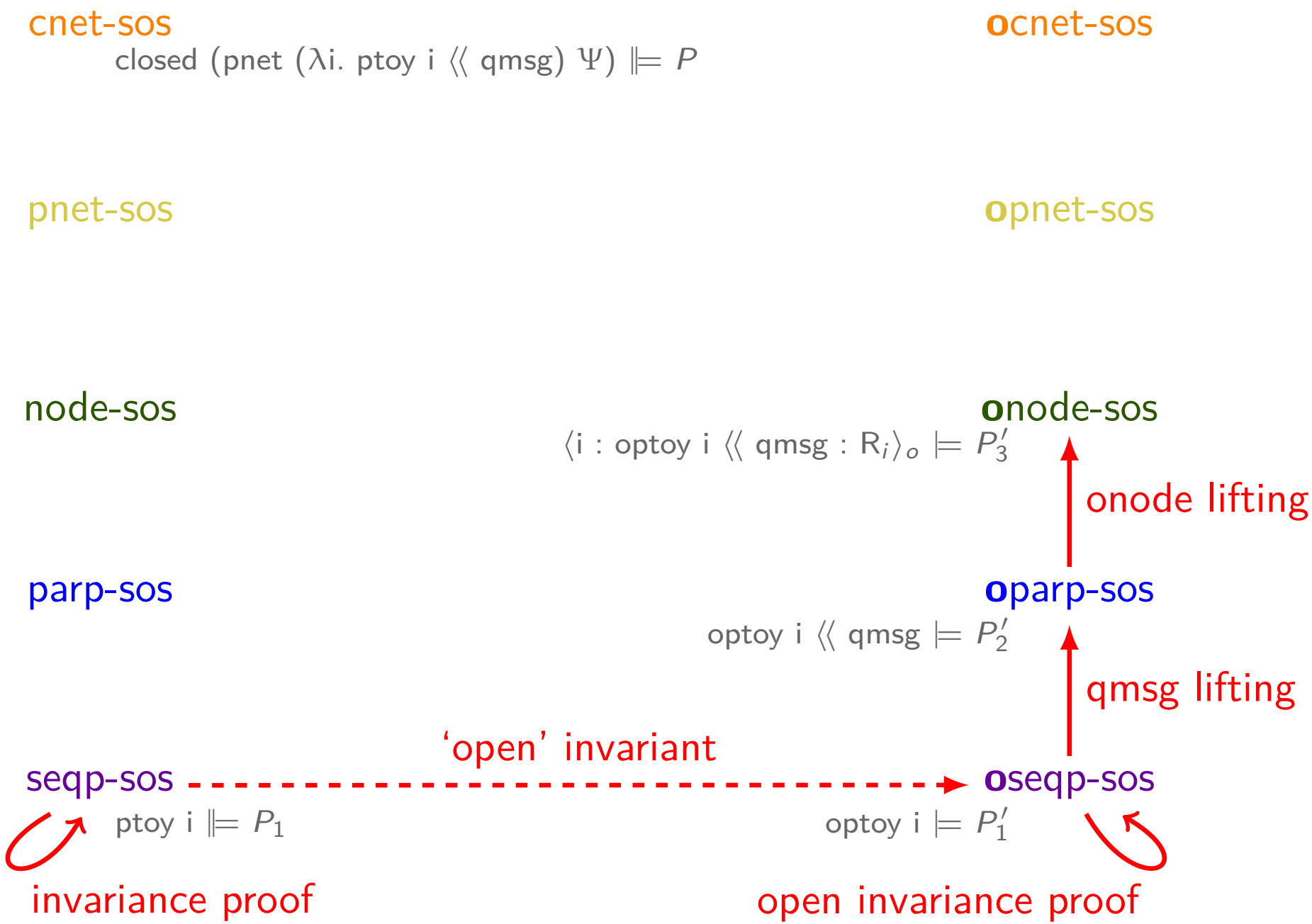
oparp-sos



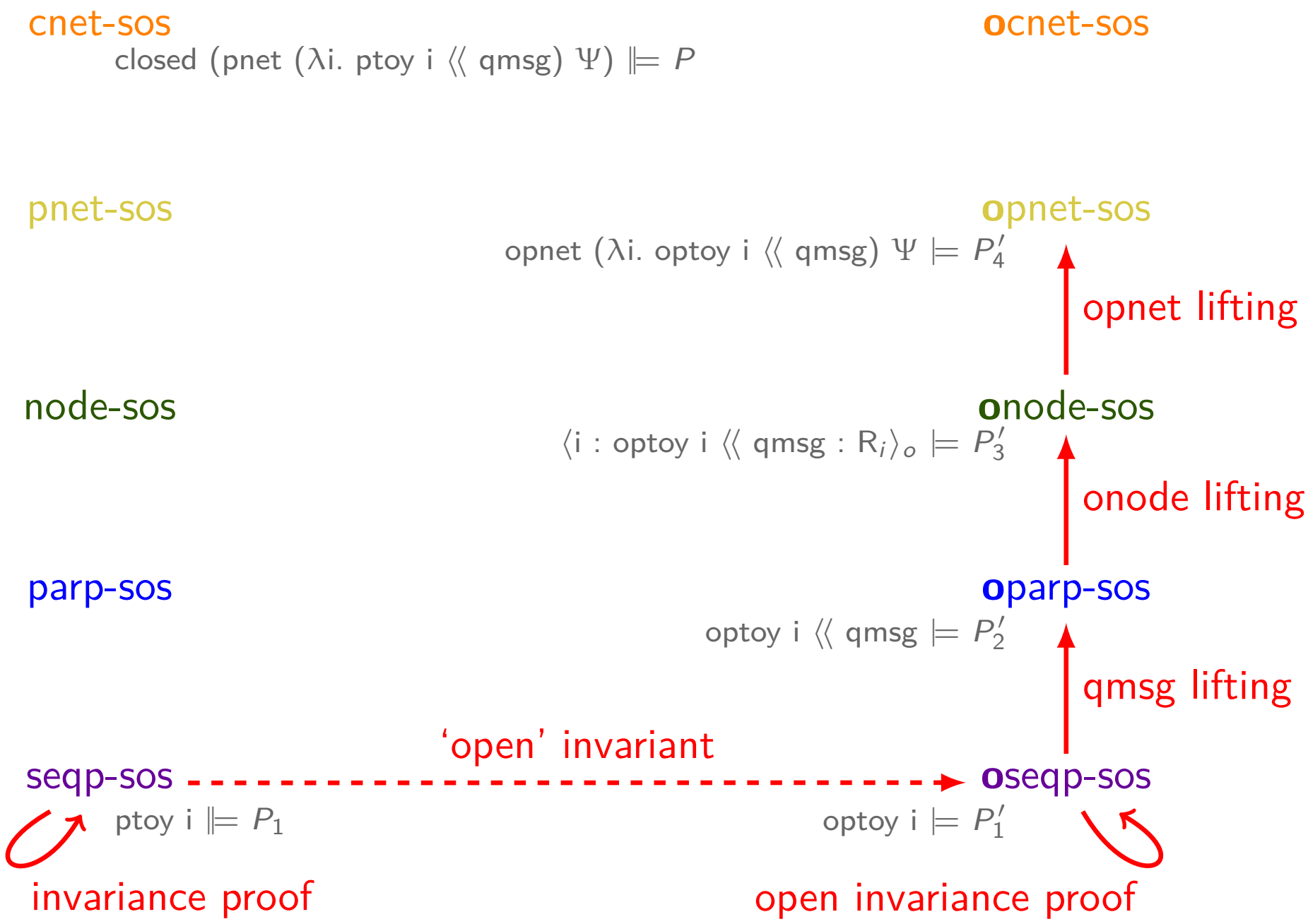
Lifting and transfer



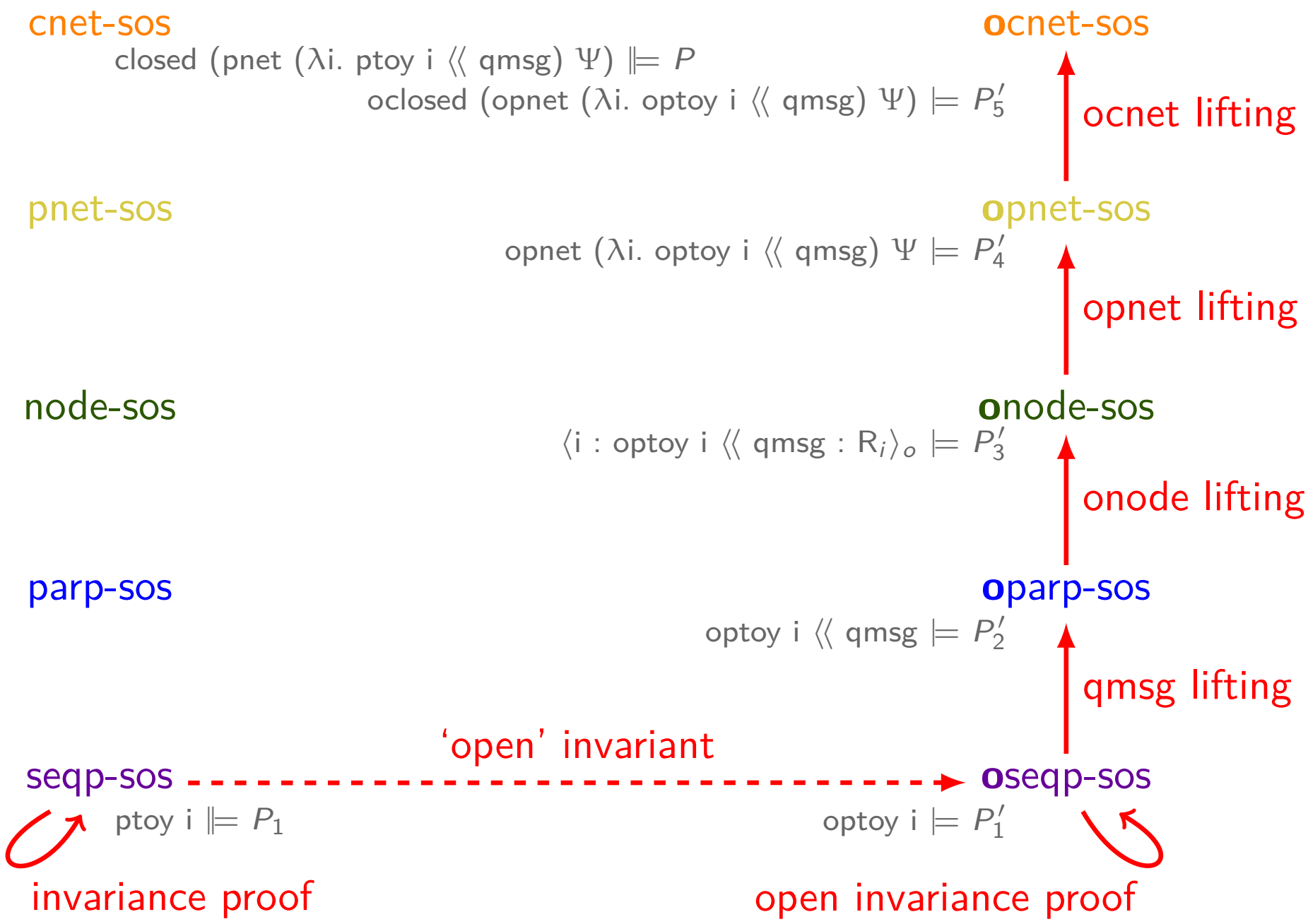
Lifting and transfer



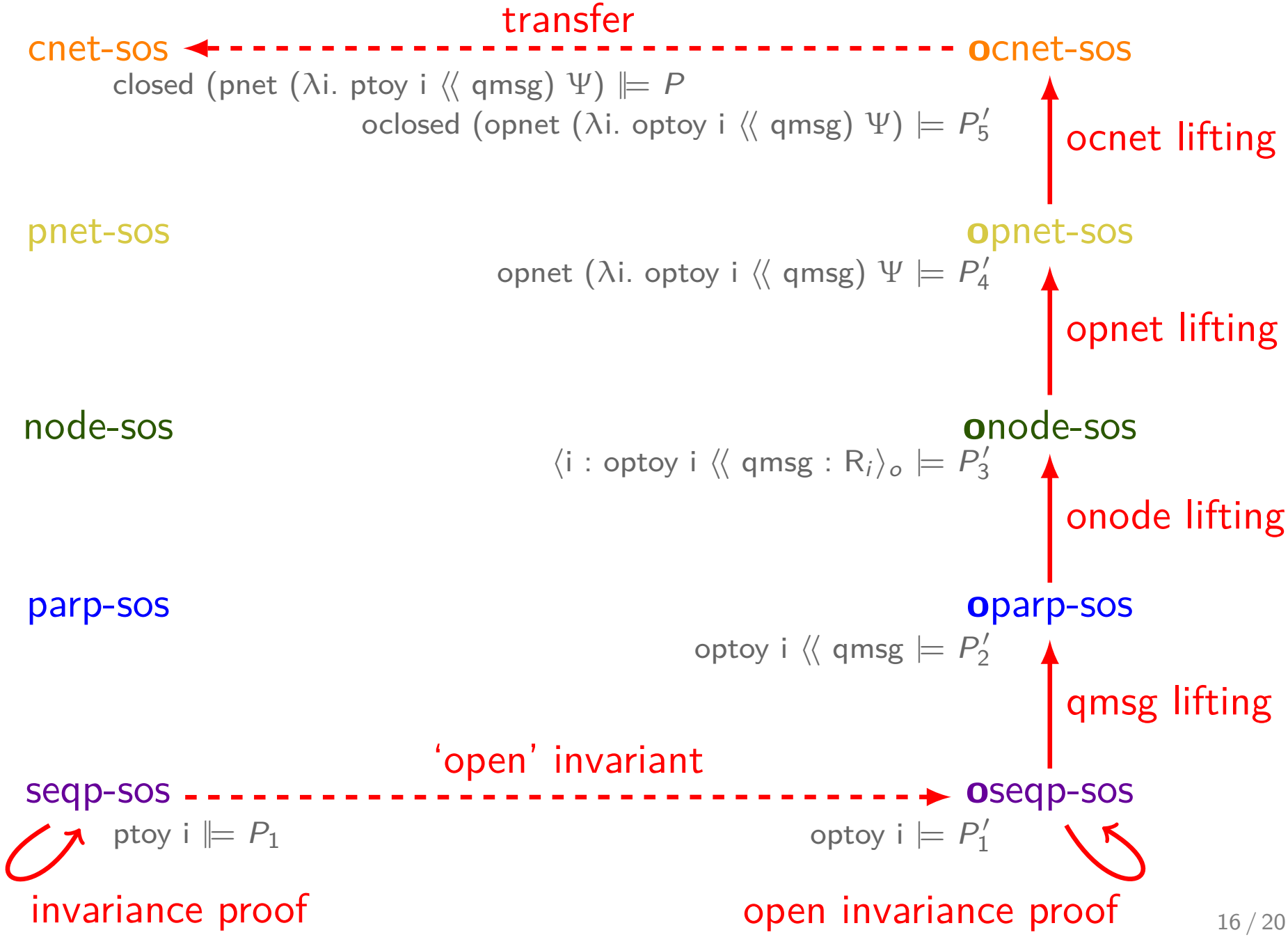
Lifting and transfer



Lifting and transfer



Lifting and transfer



Lifting for (open) networks

induction on oreachable

initial: easy
environment: easy
local: ...

S = otherwith E (net-tree-ips ($\Psi_1 \parallel \Psi_2$)) (oarrivemsg I)
U = other F (net-tree-ips ($\Psi_1 \parallel \Psi_2$))

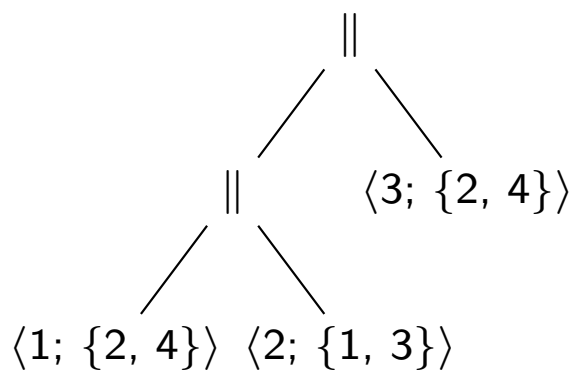
$(\sigma, s \parallel t) \in$ oreachable (opnet onp ($\Psi_1 \parallel \Psi_2$)) S U

$\Downarrow$

$(\sigma, s) \in$ oreachable (opnet onp Ψ_1) S_1 U_1

$(\sigma, t) \in$ oreachable (opnet onp Ψ_2) S_2 U_2

induction on term



⟨4; {1, 3}⟩

Lifting for (open) networks

induction on oreachable

initial: easy
environment: easy
local: ...

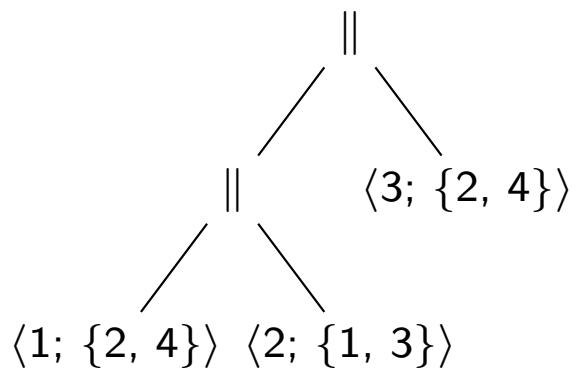
S = otherwith E (net-tree-ips ($\Psi_1 \parallel \Psi_2$)) (oarrivemsg I)
U = other F (net-tree-ips ($\Psi_1 \parallel \Psi_2$))

$(\sigma, s \shuffle t) \in$ oreachable (opnet onp ($\Psi_1 \parallel \Psi_2$)) S U

$\Downarrow$

$(\sigma, s) \in$ oreachable (opnet onp Ψ_1) S_1 U_1 $\quad$ $(\sigma, t) \in$ oreachable (opnet onp Ψ_2) S_2 U_2

induction on term



$\langle 4; \{1, 3\}\rangle$

- arrive/arrive (synchronous)
  - assumption from the environment
  - conjunction of constraints

$$\frac{((\sigma, s), H\neg K{:}\text{arrive}(m), (\sigma', s')) \in T_A \qquad ((\sigma, t), H'\neg K'{:}\text{arrive}(m), (\sigma', t')) \in T_B}{((\sigma, s \shuffle t), (H \cup H')\neg(K \cup K'){:}\text{arrive}(m), (\sigma', s' \shuffle t')) \in \text{opnet-sos } T_A\ T_B}$$

Lifting for (open) networks

induction on oreachable

initial: easy
environment: easy
local: ...

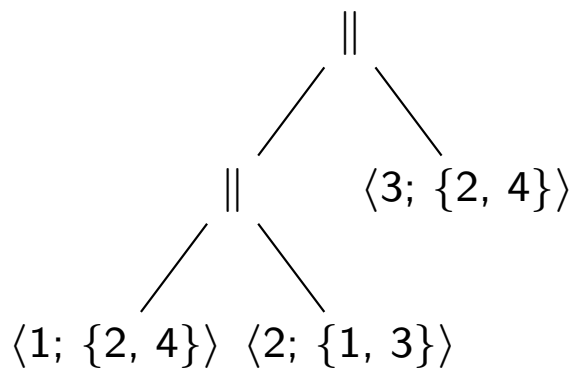
S = otherwith E (net-tree-ips ($\Psi_1 \parallel \Psi_2$)) (oarrivemsg I)
U = other F (net-tree-ips ($\Psi_1 \parallel \Psi_2$))

$$(\sigma,\ s \shuffle t) \in \text{oreachable (opnet onp } (\Psi_1 \parallel \Psi_2))\ S\ U$$

$$\Downarrow$$

$(\sigma,\ s) \in \text{oreachable (opnet onp } \Psi_1)\ S_1\ U_1$ $\qquad$ $(\sigma,\ t) \in \text{oreachable (opnet onp } \Psi_2)\ S_2\ U_2$

induction on term



$\langle 4;\ \{1,\ 3\}\rangle$

- arrive/arrive (synchronous)
- cast/arrive (synchronous)
  - assumption from cast
  - conjunction of constraints

$$\frac{((\sigma,\ s),\ R{:}{*}\text{cast}(m),\ (\sigma',\ s')) \in T_A \qquad ((\sigma,\ t),\ H\neg K{:}\text{arrive}(m),\ (\sigma',\ t')) \in T_B \qquad H \subseteq R \qquad K \cap R = \emptyset}{((\sigma,\ s \shuffle t),\ R{:}{*}\text{cast}(m),\ (\sigma',\ s' \shuffle t')) \in \text{opnet-sos } T_A\ T_B}$$

Lifting for (open) networks

induction on oreachable

initial: easy
environment: easy
local: ...

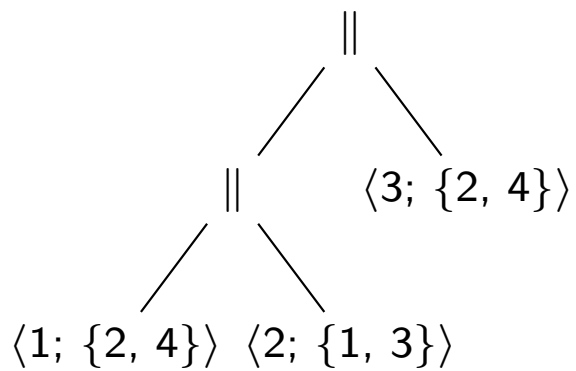
S = otherwith E (net-tree-ips ($\Psi_1 \parallel \Psi_2$)) (oarrivemsg I)
U = other F (net-tree-ips ($\Psi_1 \parallel \Psi_2$))

$$(\sigma,\ s \parallel t) \in \text{oreachable (opnet onp } (\Psi_1 \parallel \Psi_2))\ S\ U$$

$$\Downarrow$$

$(\sigma,\ s) \in \text{oreachable (opnet onp } \Psi_1)\ S_1\ U_1$ $\qquad$ $(\sigma,\ t) \in \text{oreachable (opnet onp } \Psi_2)\ S_2\ U_2$

induction on term



$\langle 4;\ \{1,\ 3\}\rangle$

- arrive/arrive (synchronous)
- cast/arrive (synchronous)
- tau (not synchronous enough)
  - encode constraints in SOS rule;
  - becomes an obligation during transfer.

$$\frac{((\sigma,\ s),\ \tau,\ (\sigma',\ s')) \in T_A \qquad \forall j \neq i.\ \sigma'\ j = \sigma\ j}{((\sigma,\ s_R^i),\ \tau,\ (\sigma',\ s'^i_R)) \in \textbf{onode-sos}\ T_A}$$

Transferring properties back to original model

If every step of np can be simulated by onp
and wf-net-tree Ψ
and oclosed (opnet onp Ψ) $\models$ (λ- - -. True, U $\rightarrow$) ($\lambda(\sigma$, -). P σ)

---

then closed (pnet np Ψ) $\Vdash$ netglobal np sr P

- Induction on reachable effectively gives simulation proof
- For each action: induction up from nodes
- Justify 'assumptions' encoded in the open semantics

- The final result is stated in terms of the original model
  —the 'open' model is used only in the proof

Conclusions

- Interactive compositional invariant proofs of a reactive system.
- The basic meta-theory is classic and readily mechanized.
  - Beneficial to focus on a concrete verification task;
  - No real process algebra.
- We worked harder to show network invariants.
  - 'Open' model with encoded assumptions;
  - Intermediate constructions only needed for the proof.
  - *Not* arbitrary compositions.
- Proofs published in the Archive of Formal Proofs
  - `http://afp.sourceforge.net/entries/AWN.shtml`
  - `http://afp.sourceforge.net/entries/AODV.shtml`