Refinement Inference for Sequence Diagrams

Lunjin Lu & Dae-kyoo Kim

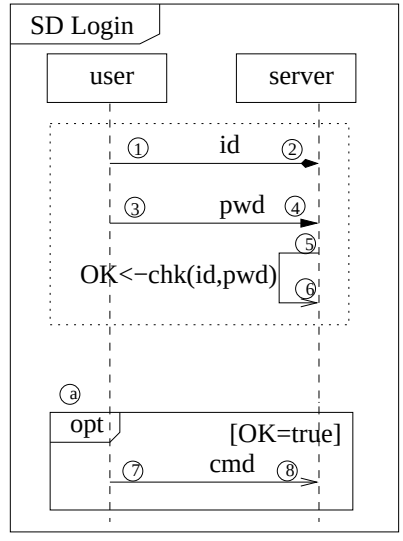
Oakland University

Outline

- Required versus Possible Behaviors
- Semantics and Refinement
- Refinement Inference Rules
- Mandatory Access Control
- Future Work

Sequence Diagrams

- Visual Specification and Design Language
- Lifelines: processes/objects
- Messages (Events)
- Fragments



Possible vs. Required Behaviors

- UML standard is not rigorous about formal semantics of sequence diagrams;
- An SD specifies a set of positive traces and a set of negative traces;
- Positive traces?
  - Possible traces: system may exhibit
  - Required traces: system must produce

Possible vs. Required Behaviors

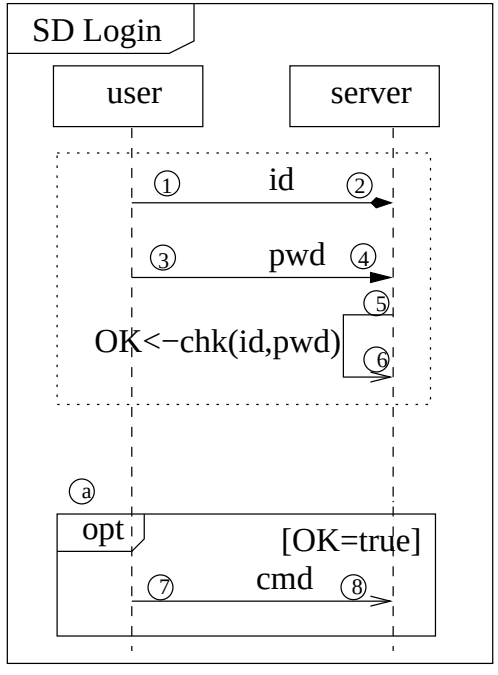
- Possible traces: useful for verification against safety properties such as secure information leakage;
- Required traces: useful for verification with respect to liveness properties such as a request being served eventually;
- UML SD models are often partial specifications, making required trace view more relevant;

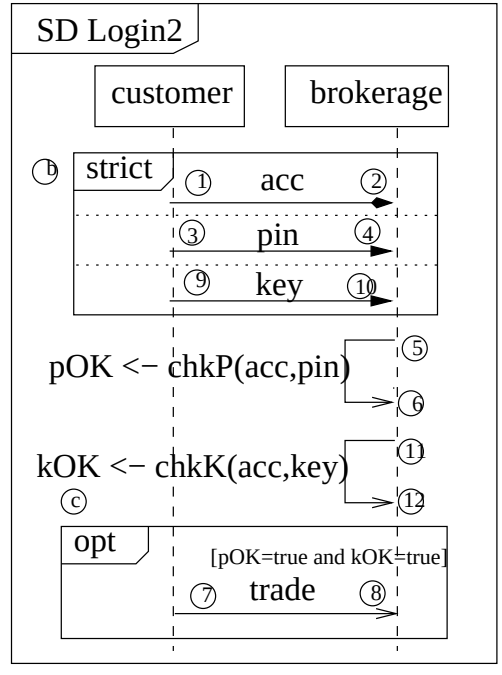
Refinement

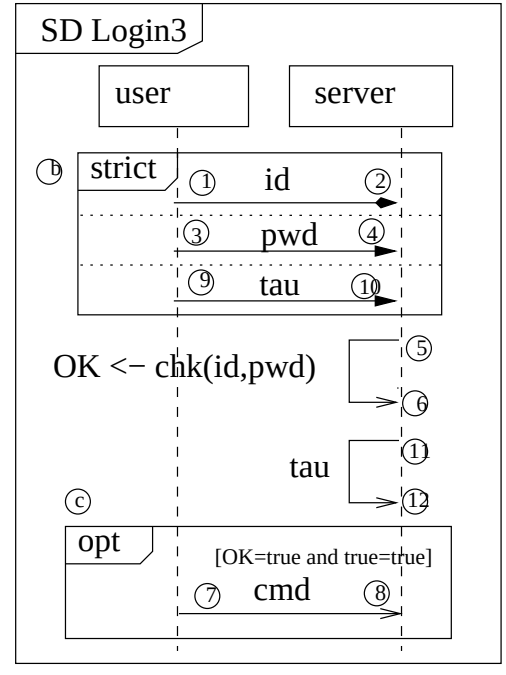
Refinement is commonly used in software development

- Model Driven Development
- Pattern-based Development
- Aspect-oriented Development

Refinement: Example

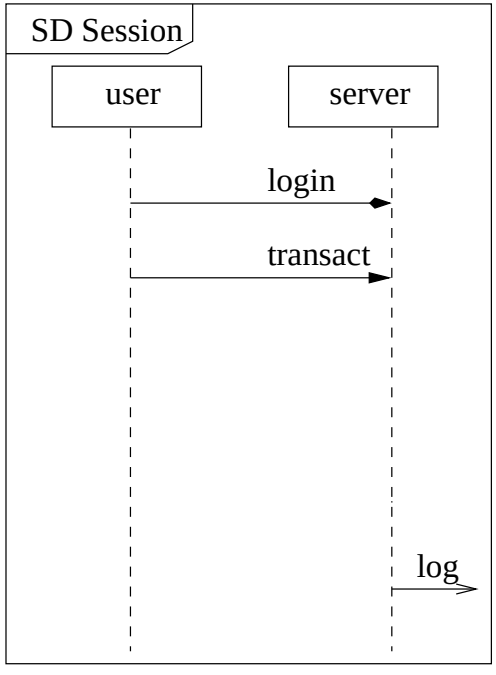


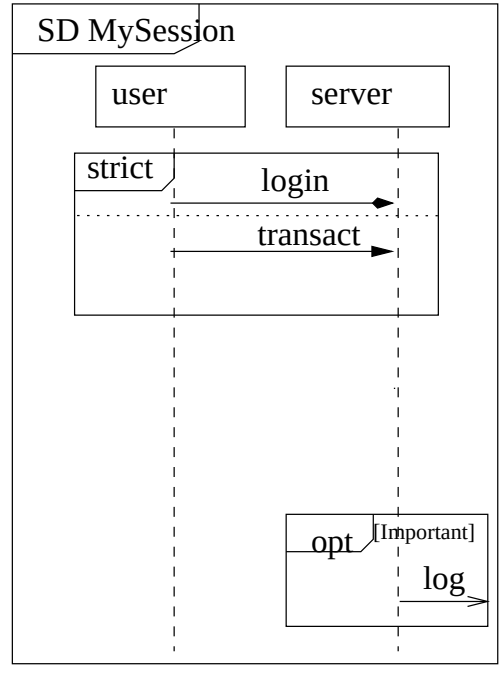




Click here for formal justification!

Refinement: Example



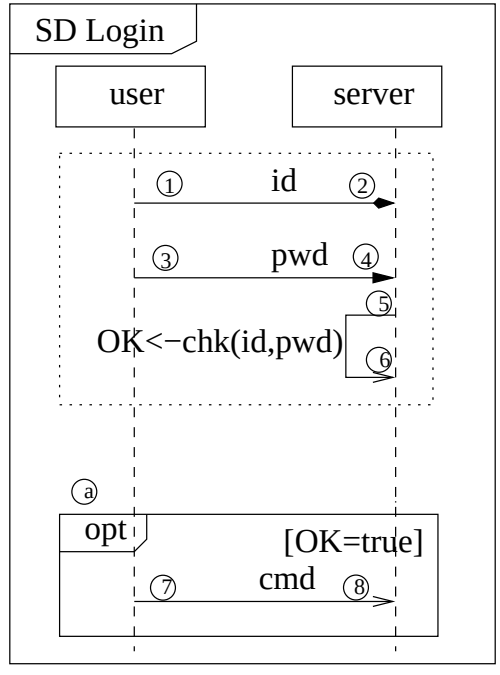


MySession refine Session only when Important=true!

Trace Semantics

- An SD specifies a set of positive (possible) traces and a negative (prohibited) traces:
  - Cengarle, Graubmann and Wagner 2006
  - Cengarle and Knapp 2004
  - Haugen etc. 2005 (if xalt is not considered)
  - Störrle 2003

Trace Semantics



t: !id ?id !pwd ?pwd !chk ?chk

t': !id !pwd ?id ?pwd !chk ?chk

t!cmd?cmd

t'!cmd?cmd

Trace Semantics

- An SD specifies a set of positive (possible) traces and a negative (prohibited) traces:
  - Cengarle, Graubmann and Wagner 2006
  - Cengarle and Knapp 2004
  - Haugen etc. 2005 (if xalt is not considered)
  - Störrle 2003
  - $[\![Login]\!] = \{t, t!cmd?cmd, t', t'!cmd?cmd\}$
- Refinement = eliminating positive traces and making them proscribed;
- An SD without positive traces refines every other SD!

Trace Semantics

- Do not distinguish required behaviors from optional behaviors as pointed out in Sengupta and Cleveland 06;
- Do not deal with critical regions adequately: either not defined for SDs with critical regions or lose substitutivity
- Ignore guard conditions, which compromises refinement reasoning of required behavior;
  - E.g. alt(c,!m,!n) refines !m, which is false

Abstract Syntax

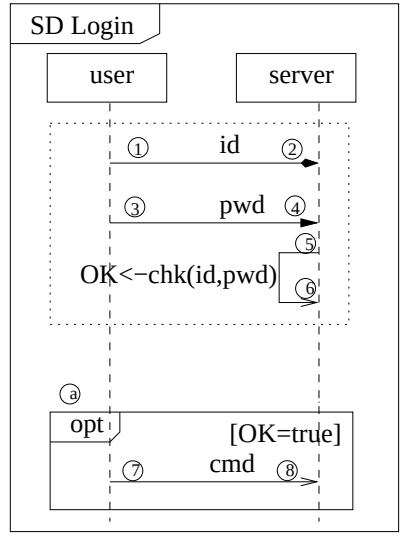
$$
\begin{aligned}
D ::= \quad & \tau \\
\mid \quad & e \\
\mid \quad & opt(c, D_1) \\
\mid \quad & alt(c, D_1, D_2) \\
\mid \quad & loop(c, D_1) \\
\mid \quad & critical(D_1) \\
\mid \quad & par(D_1, D_2) \\
\mid \quad & strict(D_1, D_2) \\
\mid \quad & seq(D_1, D_2) \\
\mid \quad & block(L, \iota, \twoheadrightarrow, \dashrightarrow)
\end{aligned}
$$

Operators not considered:

- ignore
- consider
- assert
- neg
- break

Abstract Syntax

$$
\begin{aligned}
Login &= block(\{1..6, a\}, \\
&\qquad \{i \mapsto e_i \mid 1 \leq i \leq 6\} \cup \{a \mapsto D_a\}, \twoheadrightarrow_0, \emptyset) \\
\twoheadrightarrow_0 &= \{\langle 1,2\rangle, \langle 1,3\rangle, \langle 3,4\rangle, \langle 2,4\rangle, \langle 4,5\rangle, \langle 5,6\rangle, \langle 6,a\rangle\} \\
D_a &= opt(OK = true, \\
&\qquad block(\{7,8\}, \{7 \mapsto e_7, 8 \mapsto e_8\}, \{\langle 7,8\rangle\}, \emptyset))
\end{aligned}
$$



Semantics

- Trace: sequence of events, guards and critical segments which are sequences of events and guards: $t[OK]!cmd?cmd$
- Obligation: a set of traces representing alternative minimum requirement on specified system

$$\{t[OK]!cmd?cmd, t[\neg OK]\}$$

- Meaning: a set of obligations at least one of which must be fulfilled.

$$\left\{ \begin{array}{l} \{t[OK]!cmd?cmd, t[\neg OK]\}, \\ \{t'[OK]!cmd?cmd, t'[\neg OK]\} \end{array} \right\}$$

Refinement: Issues

Refinement may require changes including

1. adding new lifelines and messages;
2. renaming lifelines and messages;
3. introducing new system variables and associated guard conditions;
4. replace one message (or event) by another knowing that the latter can simulate the former.

Solution is to parameterize refinement

- 1: a set of events $\mathcal{U}$;
- 2 and 3: a function ρ;
- 4: an oracle $\mathcal{S}$

Refinement: Definition

D_1 refines D_2 wrt $\mathcal{S}, \rho, \mathcal{U}$, denoted $D_1 \sqsupseteq_{\mathcal{S},\rho,\mathcal{U}} D_2$, iff both

1. $\rho(\mathcal{U}) \cap \{e_1 \mid \exists e_2 \in \mathbb{E}\mathrm{vt}(D_2).\langle e_1, e_2\rangle \in \mathcal{S}\} = \emptyset$, and
2. $\forall \mathcal{O}_1 \in [\![\rho(hide_{\mathcal{U}}(D_1)]\!].\exists \mathcal{O}_2 \in [\![D_2]\!].\forall t_2 \in \mathcal{O}_2^{-\circ}.\exists t_1 \in \mathcal{O}_1^{-\circ}.(t_1 \ltimes_{\mathcal{S}} t_2)$.

$\mathcal{O}^{-\circ}$ results from removing redundant decision points and redundant traces from $\mathcal{O}$

$t_1 \ltimes_{\mathcal{S}} t_2$ if t_1 can be obtained from t_2 weakening and/or dropping guards and protecting some of its sub-traces. Events can be replaced according to oracle $\mathcal{S}$

Refinement

Click here to see SDs for Login, Login2 and Login3.

$$Login3 = \rho \circ hide_{\mathcal{U}}(Login2)$$

$$[\![Login]\!] = \left\{ \begin{array}{l} \{t[OK]!cmd?cmd, t[\neg OK]\}, \\ \{t'[OK]!cmd?cmd, t'[\neg OK]\} \end{array} \right\}$$

$$[\![Login3]\!] = \{\{t[OK]!cmd?cmd, t[\neg OK]\}\}$$

Normalization

- $par(D_1, D_2) \Longrightarrow block(\{\ell_1, \ell_2\}, \{\ell_1 \mapsto D_1, \ell_2 \mapsto D_2\}, \emptyset, \emptyset)$
- $seq(D_1, D_2) \Longrightarrow block(\{\ell_1, \ell_2\}, \{\ell_1 \mapsto D_1, \ell_2 \mapsto D_2\}, \emptyset, \{\langle \ell_1, \ell_2 \rangle\})$
- $strict(D_1, D_2) \Longrightarrow block(\{\ell_1, \ell_2\}, \{\ell_1 \mapsto D_1, \ell_2 \mapsto D_2\}, \{\langle \ell_1, \ell_2 \rangle\}, \emptyset$

Normalization

- $block(L \cup \{\ell\}, \iota \cup \{\ell \mapsto D'\}, \twoheadrightarrow, \dashrightarrow) \Longrightarrow block(L'', \iota'', \twoheadrightarrow'', \dashrightarrow'')$

  where η is an invertible mapping η on $\mathbb{Lab}$ such that $D'_\eta = block(L', \iota', \twoheadrightarrow', \dashrightarrow')$ and $L \cap L' = \emptyset$, $L'' = L \cup L'$, $\iota'' = \iota \cup \iota'$, $\twoheadrightarrow'' = (\twoheadrightarrow \cup \twoheadrightarrow' \cup left(\twoheadrightarrow, \ell) \times min(\twoheadrightarrow') \cup \ max(\twoheadrightarrow') \times right(\twoheadrightarrow, \ell)) \cap (L'' \times L'')$ and $\dashrightarrow'' = (\dashrightarrow \cup \dashrightarrow' \cup left(\dashrightarrow, \ell) \times min(\dashrightarrow') \cup max(\dashrightarrow') \times right(\dashrightarrow, \ell)) \cap (L'' \times L'')$.

- Claim: Transformations are semantics-preserving.

Inference Rules

$$\frac{}{\rho,\mathcal{U}\vdash e_1 \rhd_{\mathcal{S}} e_2}\ \langle\rho(e_1),e_2\rangle\in\mathcal{S}\wedge\forall e_1'\in\mathcal{U}.\langle\rho(e_1'),e_2\rangle\notin\mathcal{S}$$

$$\frac{\rho,\mathcal{U}\vdash \iota_1(\ell_1)\rhd_{\mathcal{S}}\iota_2(\ell_1')\quad\cdots\quad\rho,\mathcal{U}\vdash\iota_1(\ell_n)\rhd_{\mathcal{S}}\iota(\ell_n')}{\rho,\mathcal{U}\vdash block(\{\ell_1..\ell_n\}\cup L,\iota_1,\twoheadrightarrow_1,\dashrightarrow_1)\rhd_{\mathcal{S}}\langle block(\{\ell_1'..\ell_n'\},\iota_2,\twoheadrightarrow_2,\dashrightarrow_2)}\ \kappa_1\wedge\kappa_2$$

$$\frac{\rho,\mathcal{U}\vdash\iota(\ell)\rhd_{\mathcal{S}}I}{\rho,\mathcal{U}\vdash block(\{\ell\}\cup L,\iota,\twoheadrightarrow,\dashrightarrow)\rhd_{\mathcal{S}}I}\ \kappa_2$$

$$\frac{}{\rho,\mathcal{U}\vdash H\rhd_{\mathcal{S}}\tau}\ \mathcal{U}\supseteq\mathbb{Evt}(H)$$

$$\kappa_1=\forall 1\le i,j\le n.\left(\begin{array}{c}((\ell_i'\twoheadrightarrow_2\ell_j')\Rightarrow(\ell_i\twoheadrightarrow_1^*\ell_j))\wedge\\((\ell_i'\dashrightarrow_2\ell_j')\Rightarrow((\ell_i\dashrightarrow_1^*\ell_j)\vee(\ell_i\twoheadrightarrow_1^*\ell_j)))\end{array}\right)$$

$$\kappa_2=\forall\ell\in L.\mathcal{U}\supseteq\mathbb{Evt}(\iota(\ell))$$

Inference Rules

$$\frac{\rho,\mathcal{U}\vdash H_1 \rhd_{\mathcal{S}} H_2 \qquad \rho,\mathcal{U}\vdash I_1 \rhd_{\mathcal{S}} I_2}{\rho,\mathcal{U}\vdash alt(C_1,H_1,I_1)\rhd_{\mathcal{S}} alt(C_2,H_2,I_2)}\; C_1 \Leftrightarrow C_2$$

$$\frac{\rho,\mathcal{U}\vdash H_1 \rhd_{\mathcal{S}} I_2 \qquad \rho,\mathcal{U}\vdash I_1 \rhd_{\mathcal{S}} H_2}{\rho,\mathcal{U}\vdash alt(C_1,H_1,I_1)\rhd_{\mathcal{S}} alt(C_2,H_2,I_2)}\; C_1 \Leftrightarrow \neg C_2$$

$$\frac{\rho,\mathcal{U}\vdash H \rhd_{\mathcal{S}} J}{\rho,\mathcal{U}\vdash alt(C_1,H,I)\rhd_{\mathcal{S}} opt(C_2,J)}\; (C_2 \Leftrightarrow C_1) \wedge (\mathcal{U} \supseteq \mathbb{E}\mathrm{vt}(I))$$

$$\frac{\rho,\mathcal{U}\vdash I \rhd_{\mathcal{S}} J}{\rho,\mathcal{U}\vdash alt(C_1,H,I)\rhd_{\mathcal{S}} opt(C_2,J)}\; (C_2 \Leftrightarrow \neg C_1) \wedge (\mathcal{U} \supseteq \mathbb{E}\mathrm{vt}(H))$$

$$\frac{\rho,\mathcal{U}\vdash J \rhd_{\mathcal{S}} H \qquad \rho,\mathcal{U}\vdash J \rhd_{\mathcal{S}} I}{\rho,\mathcal{U}\vdash J \rhd_{\mathcal{S}} alt(C,H,I)}$$

$$\frac{\rho,\mathcal{U}\vdash H \rhd_{\mathcal{S}} J \qquad \rho,\mathcal{U}\vdash I \rhd_{\mathcal{S}} J}{\rho,\mathcal{U}\vdash alt(C,H,I) \rhd_{\mathcal{S}} J}$$

Inference Rules

$$\frac{\rho, \mathcal{U} \vdash H_1 \rhd_{\mathcal{S}} H_2}{\rho, \mathcal{U} \vdash loop(C_1, H_1) \rhd_{\mathcal{S}} loop(C_2, H_2)}\ C_2 \Leftrightarrow C_1$$

$$\frac{\rho, \mathcal{U} \vdash H_1 \rhd_{\mathcal{S}} H_2}{\rho, \mathcal{U} \vdash loop(C_1, H_1) \rhd_{\mathcal{S}} opt(C_2, H_2)}\ C_2 \Leftrightarrow C_1$$

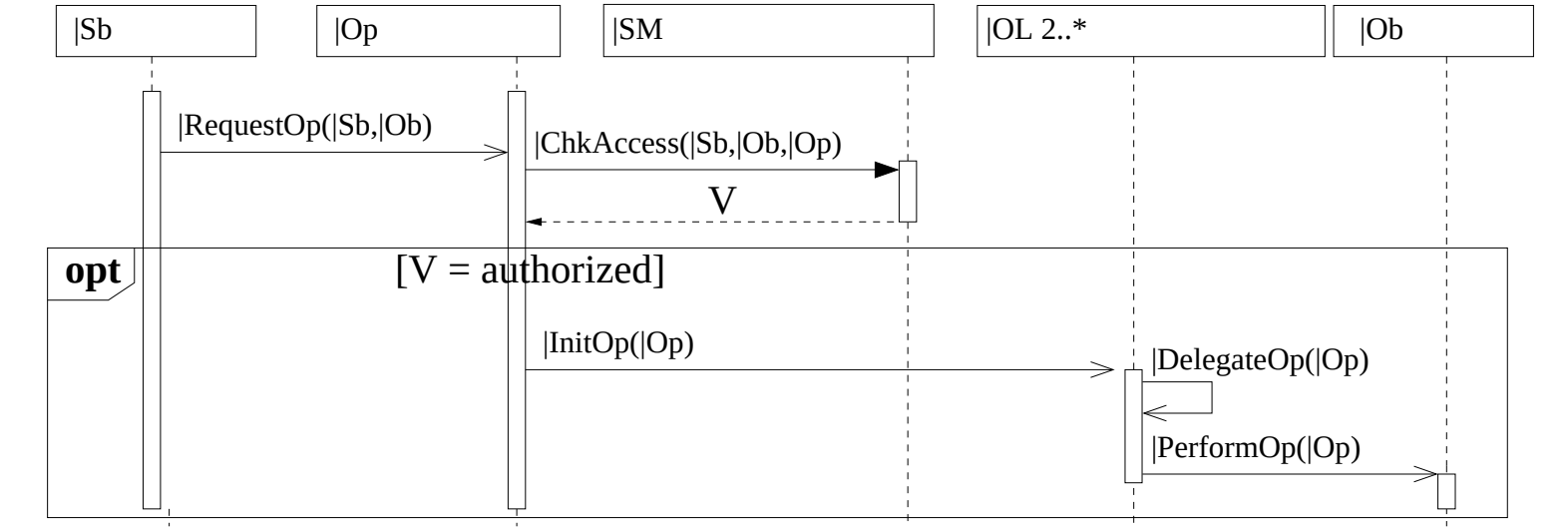
$$\frac{\rho, \mathcal{U} \vdash H_1 \rhd_{\mathcal{S}} H_2}{\rho, \mathcal{U} \vdash opt(C_1, H_1) \rhd_{\mathcal{S}} opt(C_2, H_2)}\ C_2 \Leftrightarrow C_1$$

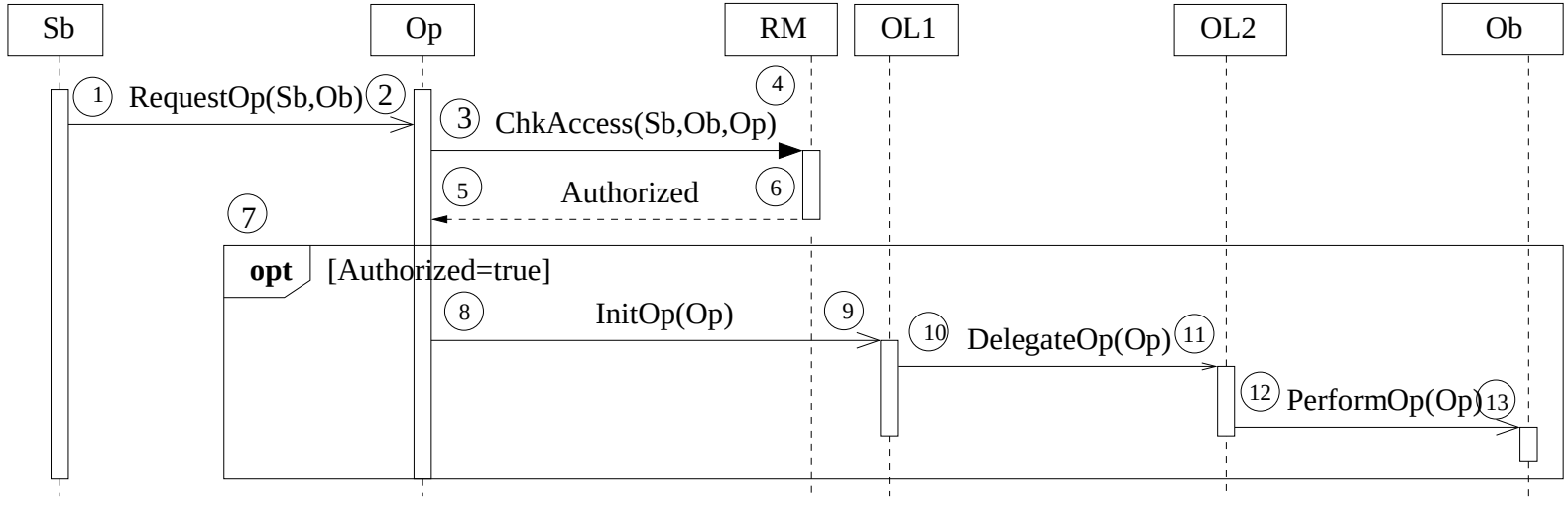
$$\frac{\rho, \mathcal{U} \vdash H \rhd_{\mathcal{S}} I}{\rho, \mathcal{U} \vdash critical(H) \rhd_{\mathcal{S}} critical(I)}$$

$$\frac{\rho, \mathcal{U} \vdash H \rhd_{\mathcal{S}} I}{\rho, \mathcal{U} \vdash critical(H) \rhd_{\mathcal{S}} I}$$

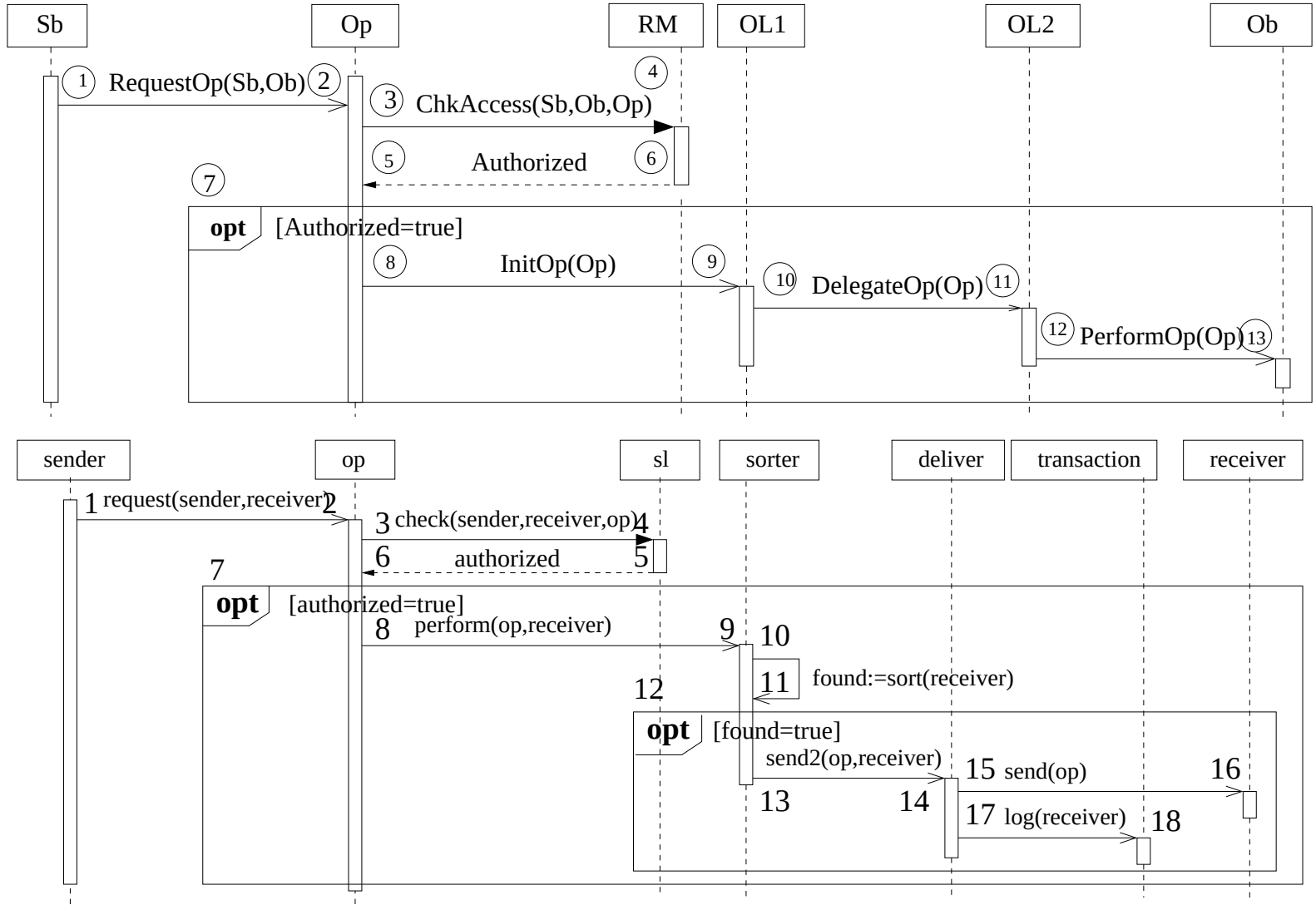
Claim: Inference system is sound.

Case Study: MAC IPS and Instance

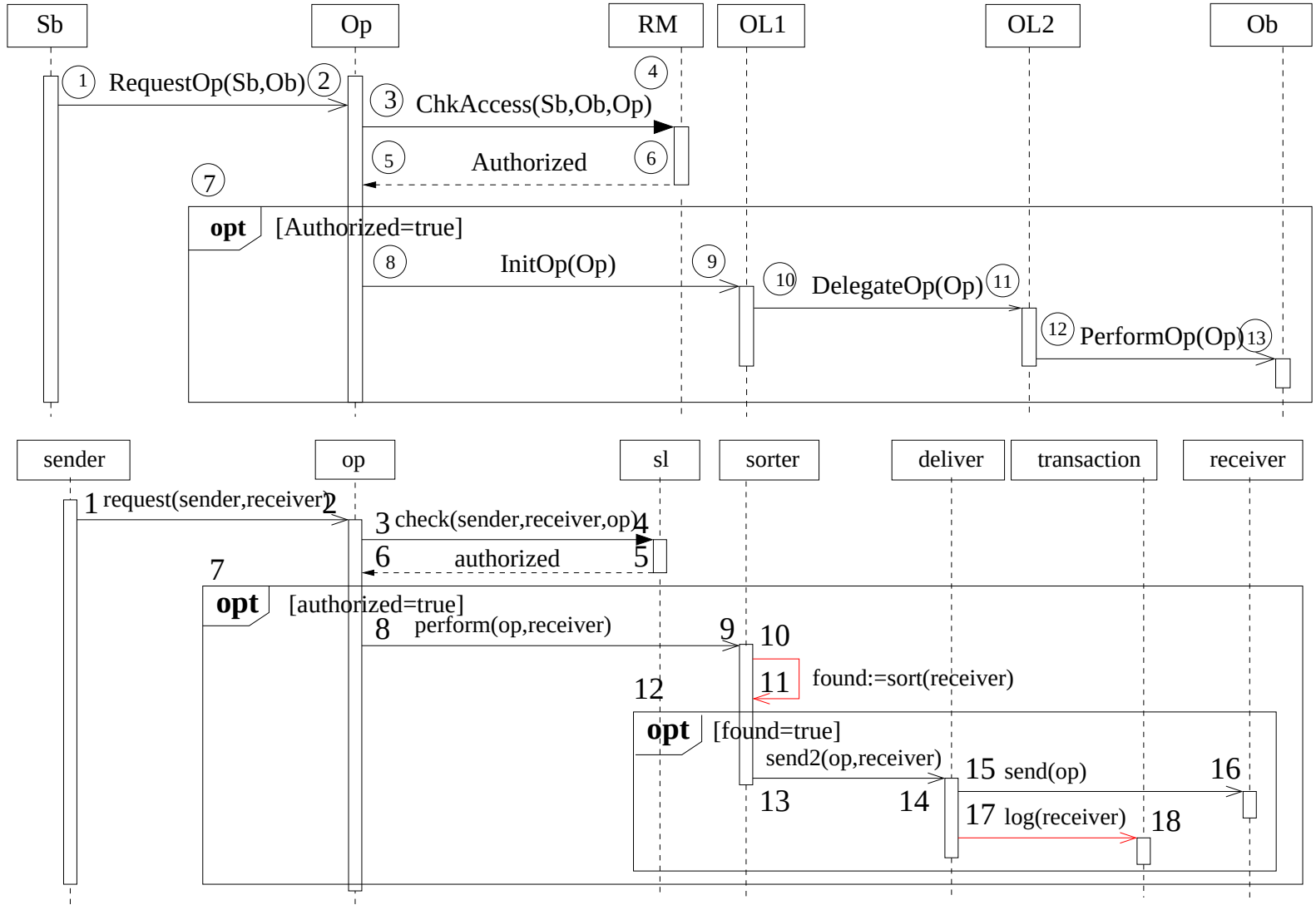




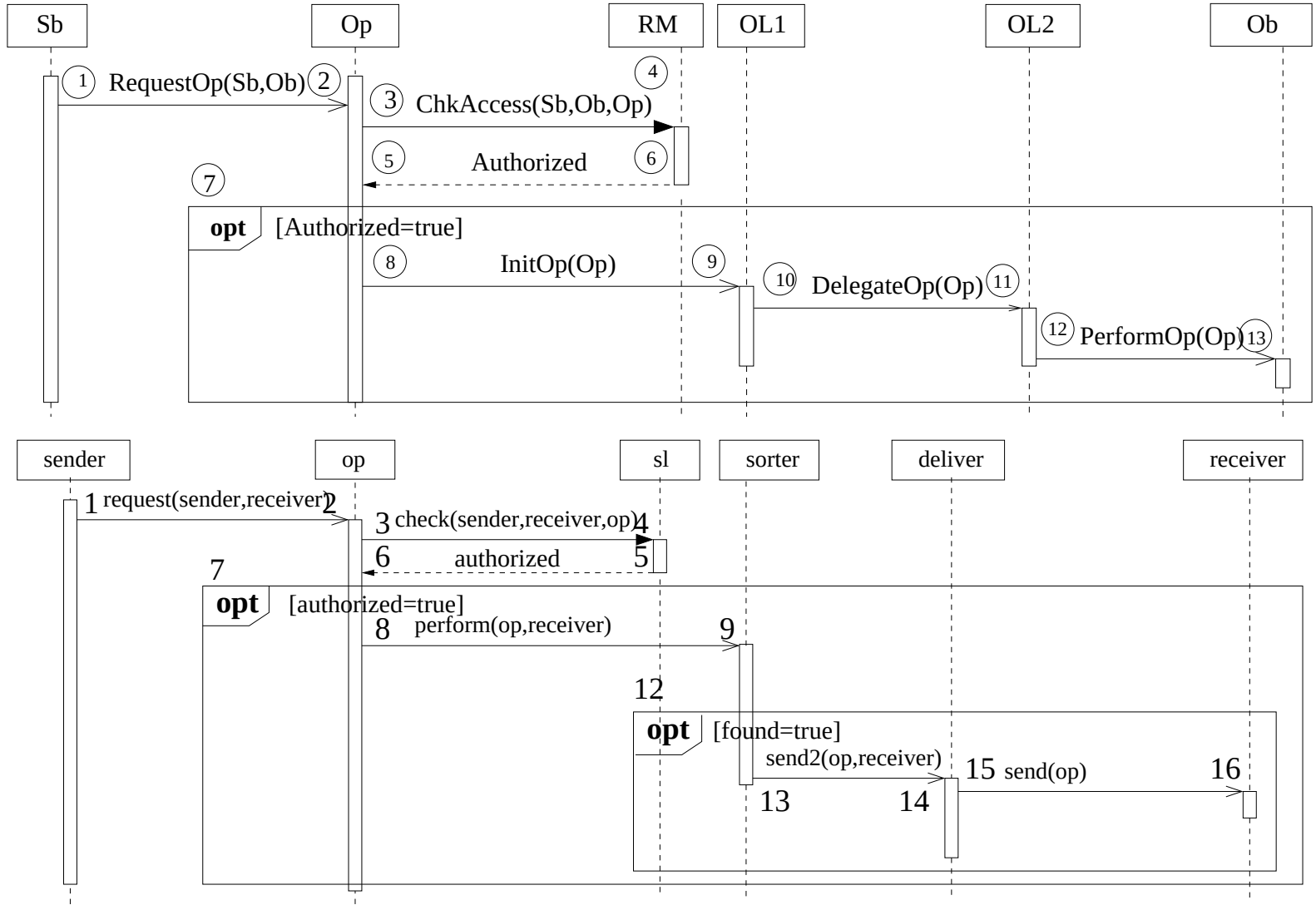
Case Study: MAC



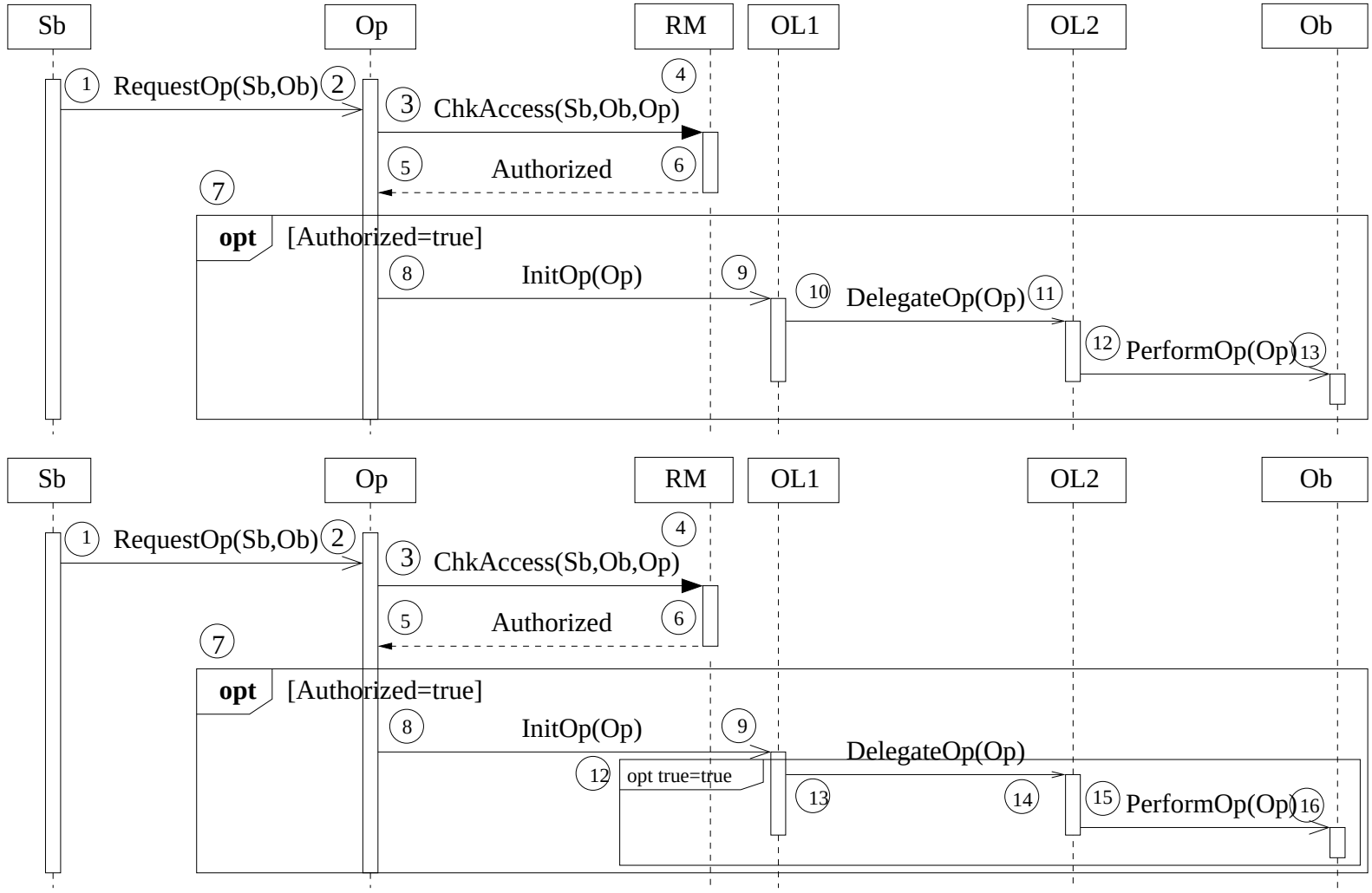
Case Study: MAC



Case Study: MAC



Case Study: MAC



Case Study: Hidden Events and Renaming

$$\mathcal{U} = \{e_{11}, e_{12}, e_{17}, e_{18}\}$$

$$\rho = \left\{ \begin{array}{c} request \mapsto RequestOp, check \mapsto ChkAccess, \\ perform \mapsto InitOp, \\ send2 \mapsto DelegateOp, send \mapsto PerformOp, \\ authorized \mapsto Authorized, sender \mapsto Sb, op \mapsto Op, \\ receiver \mapsto Ob, sl \mapsto SM, sorter \mapsto OL1, deliver \mapsto OL2 \end{array} \right\}$$

Future work

- Extention to the interaction operator *neg* and *assert*.
- Integration of SD refinement inference with class diagram refinement inference.
- Decidability of the refinement relation and completeness of the proposed refinement inference system.