Local Reasoning for Reconfigurable Distributed Systems

Emma Ahrens

Supervised by Dr. Radu Iosif[1]
and Prof. Dr. Joost-Pieter Katoen

March 2, 2021

[1] With support from Dr. Marius Bozga

Contents

Table of Contents

Hoare Logic

Verification

If C is a program, then we specify assertions P and Q and prove:

$$\{\ P\ \}\ C\ \{\ Q\ \}.$$

Separation Logic

- Extension of Hoare Logic

Separation Logic

- Extension of Hoare Logic
- Combines boolean ($\wedge$) and spatial ($*$) connectives

Separation Logic

- Extension of Hoare Logic
- Combines boolean ($\wedge$) and spatial ($*$) connectives
- Originally: Verification of programs with pointers

Separation Logic

- Extension of Hoare Logic
- Combines boolean ($\wedge$) and spatial ($*$) connectives
- Originally: Verification of programs with pointers
- *Abstract Separation Logic* for verification of programs on arbitrary resources

Separation Logic

- Extension of Hoare Logic
- Combines boolean ($\wedge$) and spatial ($*$) connectives
- Originally: Verification of programs with pointers
- *Abstract Separation Logic* for verification of programs on arbitrary resources
- Supports *local reasoning*

Separation Logic

- Extension of Hoare Logic
- Combines boolean ($\wedge$) and spatial ($*$) connectives
- Originally: Verification of programs with pointers
- *Abstract Separation Logic* for verification of programs on arbitrary resources
- Supports *local reasoning*

2	7	11	4
0	1	2	3

Separation Logic

- Extension of Hoare Logic
- Combines boolean ($\wedge$) and spatial ($*$) connectives
- Originally: Verification of programs with pointers
- *Abstract Separation Logic* for verification of programs on arbitrary resources
- Supports *local reasoning*

2	7	22	4
0	1	2	3

Separation Logic

- Extension of Hoare Logic
- Combines boolean ($\wedge$) and spatial ($*$) connectives
- Originally: Verification of programs with pointers
- *Abstract Separation Logic* for verification of programs on arbitrary resources
- Supports *local reasoning*

2	7	11	4		8	17		
0	1	2	3	4	5	6	7	8

Separation Logic

- Extension of Hoare Logic
- Combines boolean ($\wedge$) and spatial ($*$) connectives
- Originally: Verification of programs with pointers
- *Abstract Separation Logic* for verification of programs on arbitrary resources
- Supports *local reasoning*

2	7	22	4		8	17		
0	1	2	3	4	5	6	7	8

BIP

- Architecture description language for component-based distributed systems

BIP

- Architecture description language for component-based distributed systems
- Short for *behavior, interaction, priority*

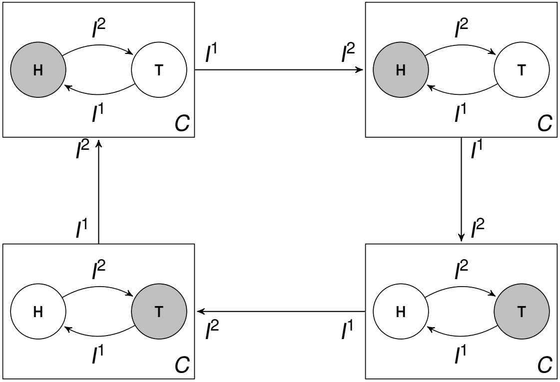
BIP

- Architecture description language for component-based distributed systems
- Short for *behavior, interaction, priority*
- Behavior represented by set of *components* that contain finite-state transition system and ports

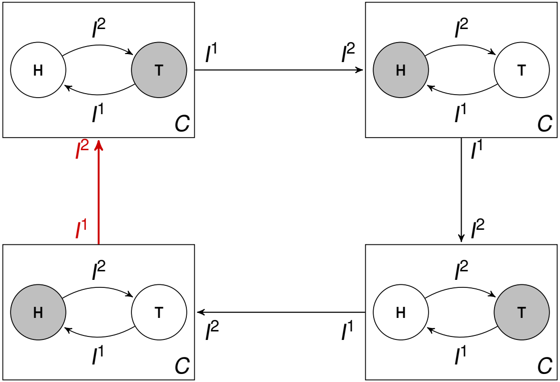
BIP

- Architecture description language for component-based distributed systems
- Short for *behavior, interaction, priority*
- Behavior represented by set of *components* that contain finite-state transition system and ports
- *Interactions* connect ports of components

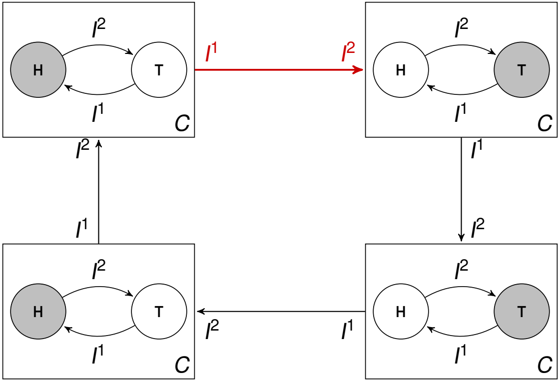
Token Ring



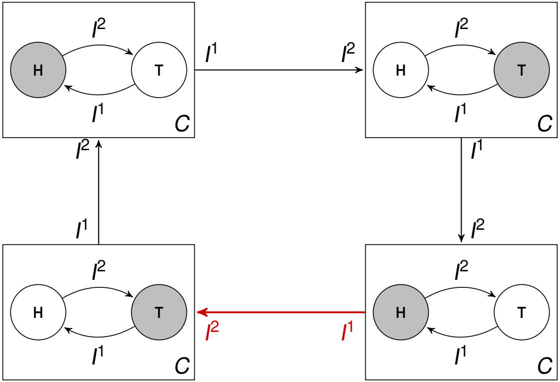
Token Ring



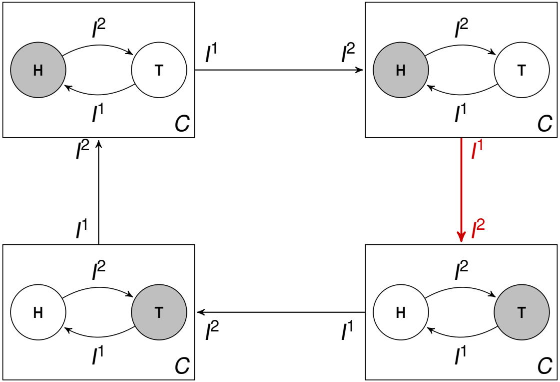
Token Ring



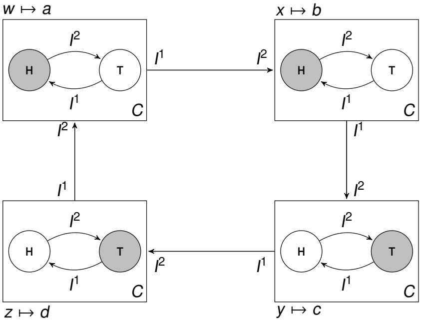
Token Ring



Token Ring

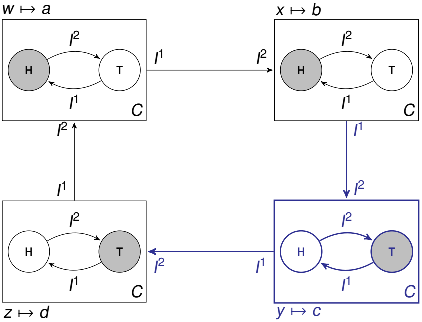


Verification of Reconfiguration Programs



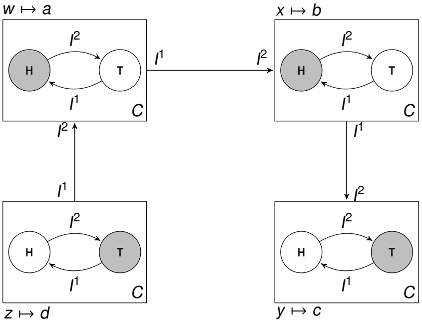

```
with  I(x,y) ∗ C(y) ∗ I(y,z) ∧ state(y,T)  do
   disconnect(I,y,z);
   disconnect(I,x,y);
   delete(C,y);
   connect(I,x,z)
```

Verification of Reconfiguration Programs



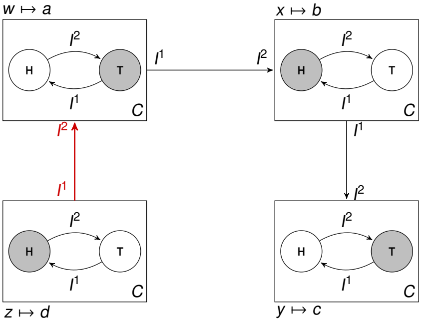

```
with I(x,y) ∗ C(y) ∗ I(y,z) ∧ state(y,T) do
    disconnect(I,y,z);
    disconnect(I,x,y);
    delete(C,y);
    connect(I,x,z)
```

Verification of Reconfiguration Programs



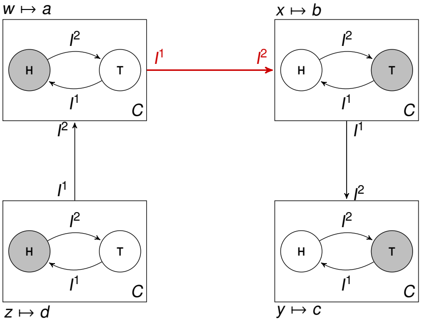

```
with  I(x,y) ∗ C(y) ∗ I(y,z) ∧ state(y,T)  do
  disconnect(I,y,z);
  disconnect(I,x,y);
  delete(C,y);
  connect(I,x,z)
```

Verification of Reconfiguration Programs



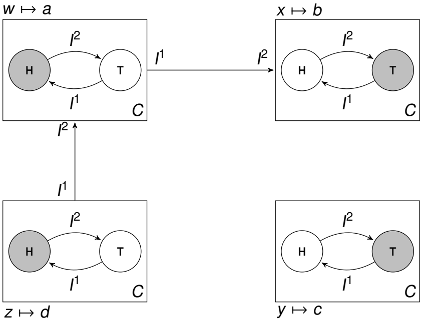

```
with I(x,y) ∗ C(y) ∗ I(y,z) ∧ state(y,T) do
    disconnect(I,y,z);
    disconnect(I,x,y);
    delete(C,y);
    connect(I,x,z)
```

Verification of Reconfiguration Programs



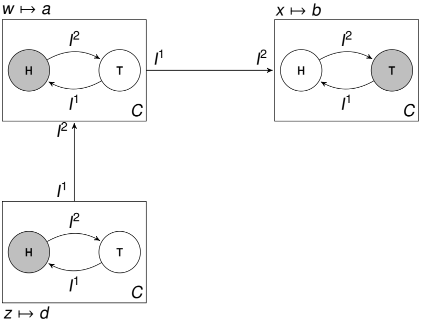

```
with I(x,y) * C(y) * I(y,z) ∧ state(y,T) do
  disconnect(I,y,z);
  disconnect(I,x,y);
  delete(C,y);
  connect(I,x,z)
```

Verification of Reconfiguration Programs




```
with  I(x,y) ∗ C(y) ∗ I(y,z) ∧ state(y,T)  do
   disconnect(I,y,z);
   disconnect(I,x,y);
   delete(C,y);
   connect(I,x,z)
```

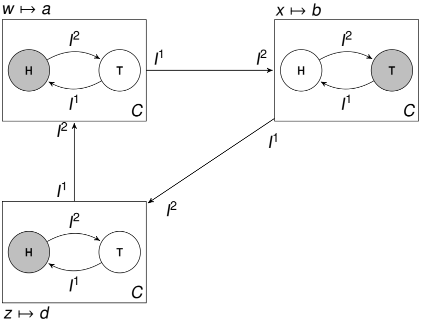
Verification of Reconfiguration Programs



with $I(x, y) * C(y) * I(y, z) \wedge \mathrm{state}(y, \mathrm{T})$ **do**

```
    disconnect(I,y,z);
    disconnect(I,x,y);
    delete(C,y);
    connect(I,x,z)
```

Verification of Reconfiguration Programs




```
with I(x,y) ∗ C(y) ∗ I(y,z) ∧ state(y,T) do
   disconnect(I,y,z);
   disconnect(I,x,y);
   delete(C,y);
   connect(I,x,z)
```

Verification of Reconfiguration Programs

Objectives

To verify reconfiguration programs using Hoare logic, we need to

- define *BIP Configurations*,

Verification of Reconfiguration Programs

Objectives

To verify reconfiguration programs using Hoare logic, we need to

- define *BIP Configurations*,
- define *Separation Logic* on BIP configurations,

Verification of Reconfiguration Programs

Objectives

To verify reconfiguration programs using Hoare logic, we need to

- define *BIP Configurations*,
- define *Separation Logic* on BIP configurations,
- define *Reconfiguration Language*, and

Verification of Reconfiguration Programs

Objectives

To verify reconfiguration programs using Hoare logic, we need to

- define *BIP Configurations*,
- define *Separation Logic* on BIP configurations,
- define *Reconfiguration Language*, and
- specify inference rules and prove their soundness.

Table of Contents

Signature

Definition

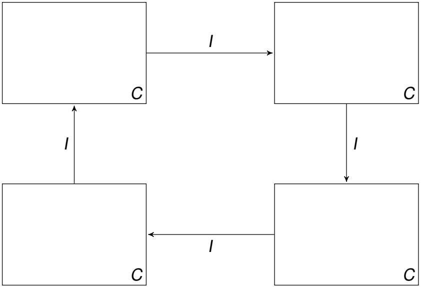
The *signature* of a BIP system is

$$\langle\, \mathcal{C}, \mathcal{I} \,\rangle = \langle\, C_1, \ldots, C_n, I_1, \ldots, I_m \,\rangle,$$

where

- $\mathcal{C} = \{C_1, \ldots, C_n\}$ is a finite set of component symbols with arity 1, and
- $\mathcal{I} = \{I_1, \ldots, I_m\}$ is a finite set of interaction symbols with arity $\alpha(I_j) \geq 2$ for each $I_j \in I$.

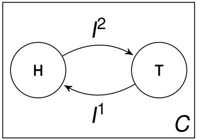
Signature



Token Ring

- Only one component type C and one interaction type I with arity $\alpha(I) = 2$.
- The signature is $\langle\ C, I\ \rangle$.

Components



Definition

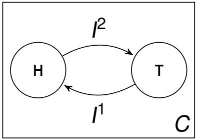
The *component type* $C_i \in C$ is associated to

$$(\mathbb{S}_i, \mathbb{P}_i, s_i^0, \rightsquigarrow_i),$$

where

- $\mathbb{S}_i$ is a finite set of states,

Components



Definition

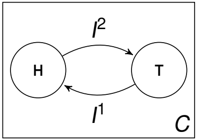
The *component type* $C_i \in C$ is associated to

$$(\mathbb{S}_i, \mathbb{P}_i, s_i^0, \leadsto_i),$$

where

- $\mathbb{S}_i$ is a finite set of states,
- $\mathbb{P}_i \subseteq \{I_j^\ell \mid I_j \in I,\ 1 \leq \ell \leq \alpha(j)\}$ is a finite set of ports,

Components



Definition

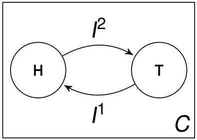
The *component type* $C_i \in C$ is associated to

$$(\mathbb{S}_i, \mathbb{P}_i, s_i^0, \rightsquigarrow_i),$$

where

- $\mathbb{S}_i$ is a finite set of states,
- $\mathbb{P}_i \subseteq \{l_j^\ell \mid l_j \in I,\ 1 \leq \ell \leq \alpha(j)\}$ is a finite set of ports,
- $s_i^0 \in \mathbb{S}_i$ is the initial state and

Components



Definition

The *component type* $C_i \in C$ is associated to

$$(\mathbb{S}_i, \mathbb{P}_i, s_i^0, \rightsquigarrow_i),$$

where

- $\mathbb{S}_i$ is a finite set of states,
- $\mathbb{P}_i \subseteq \{I_j^\ell \mid I_j \in I,\ 1 \leq \ell \leq \alpha(j)\}$ is a finite set of ports,
- $s_i^0 \in \mathbb{S}_i$ is the initial state and
- $\rightsquigarrow_i \subseteq \mathbb{S}_i \times \mathbb{P}_i \times \mathbb{S}_i$ is a finite set of transition rules.

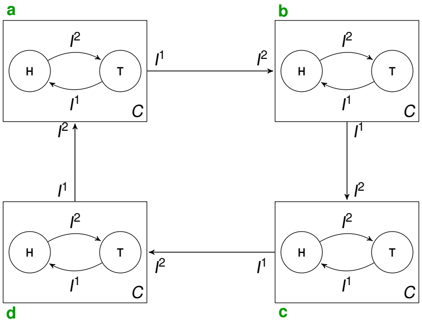
BIP System

Definition

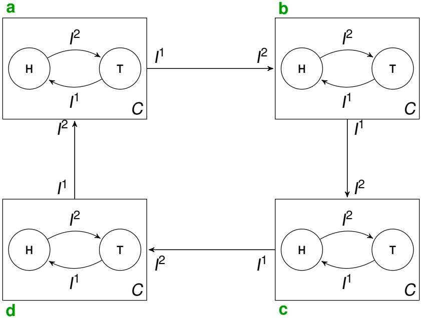
A *BIP system* is $\mathfrak{S} := \langle\, C_1^{\mathfrak{S}}, \ldots, C_n^{\mathfrak{S}}, I_1^{\mathfrak{S}}, \ldots, I_m^{\mathfrak{S}} \,\rangle$, where

- $C_i^{\mathfrak{S}} \subseteq \mathcal{U}$, $1 \leq i \leq n$, are relations over the universe $\mathcal{U}$ with arity 1, and
- $I_j^{\mathfrak{S}} \subseteq \mathcal{U}^{\alpha(j)}$, $1 \leq j \leq m$, are relations over the universe $\mathcal{U}$ with arity $\alpha(j)$.

Token Ring as BIP System



Token Ring as BIP System

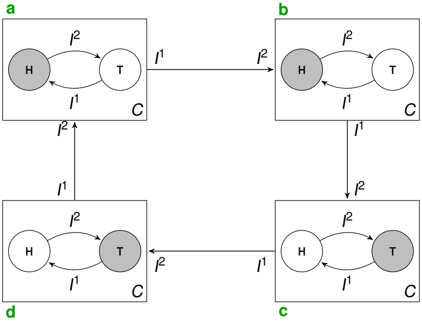


This system can be written as

$$\mathfrak{S} = \langle\ C^{\mathfrak{S}} = \{a, b, c, d\}, I^{\mathfrak{S}} = \{(a, b), (b, c), (c, d), (d, a)\}\ \rangle$$

for pairwise distinct elements $a, b, c, d \in \mathcal{U}$.

Token Ring as BIP System

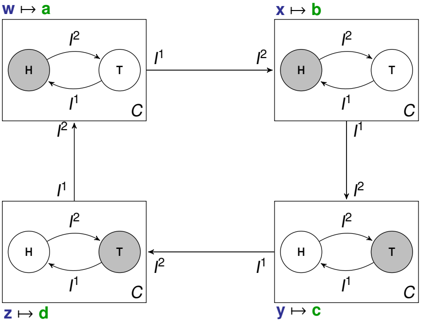


This system can be written as

$$\mathfrak{S} = \langle\ C^{\mathfrak{S}} = \{a, b, c, d\}, I^{\mathfrak{S}} = \{(a, b), (b, c), (c, d), (d, a)\}\ \rangle$$

for pairwise distinct elements $a, b, c, d \in \mathcal{U}$.

Token Ring as BIP System



This system can be written as

$$\mathfrak{S} = \langle\, C^{\mathfrak{S}} = \{a, b, c, d\}, I^{\mathfrak{S}} = \{(a, b), (b, c), (c, d), (d, a)\}\,\rangle$$

for pairwise distinct elements $a, b, c, d \in \mathcal{U}$.

BIP Configurations

Definition

Let $\mathbb{S} = \bigcup_{i=1}^{n} \mathbb{S}_i$. A *state snapshot* is a function

$$\varsigma : \mathcal{U} \times C \to \mathbb{S},$$

where $\varsigma(u, C_i) \in \mathbb{S}_i$ for every $1 \leq i \leq n$.

BIP Configurations

Definition

Let $\mathbb{S} = \bigcup_{i=1}^{n} \mathbb{S}_i$. A *state snapshot* is a function

$$\varsigma : \mathcal{U} \times C \rightarrow \mathbb{S},$$

where $\varsigma(u, C_i) \in \mathbb{S}_i$ for every $1 \leq i \leq n$.

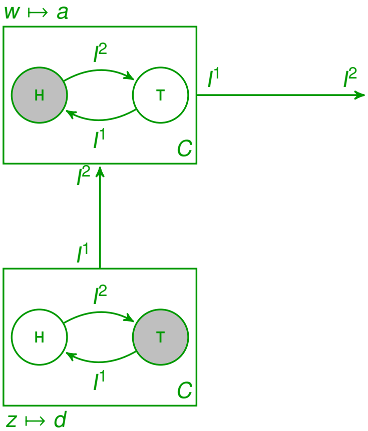
Definition

A *BIP Configuration* is a triple $(\mathfrak{S}, \varsigma, \nu)$, where

- $\mathfrak{S}$ is a BIP system,
- ς is a state snapshot, and
- $\nu : \mathcal{V} \rightarrow \mathcal{U}$ maps each variable to an element in the universe.

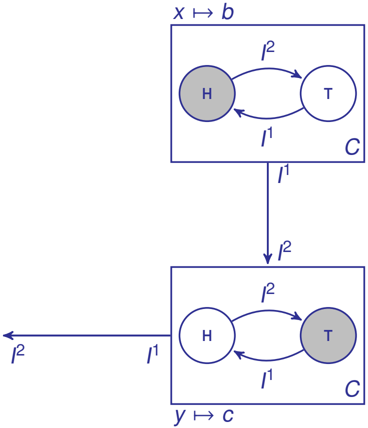
Set of configurations $\Sigma_{\langle\, C, \mathcal{I}\,\rangle}$.

Separation Algebra



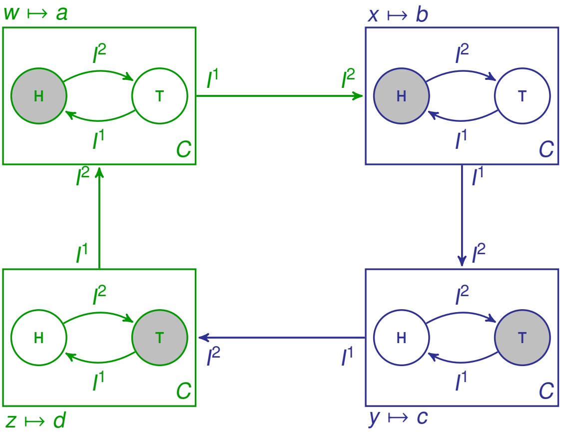
$$(\mathfrak{S}_0, \varsigma, \nu)$$

Separation Algebra



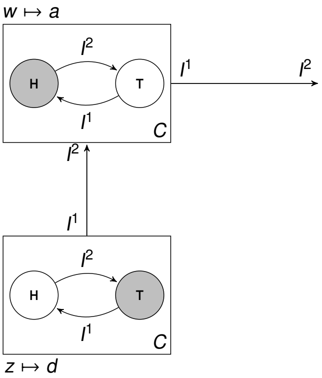
$$(\mathfrak{S}_1, \varsigma, \nu)$$

Separation Algebra

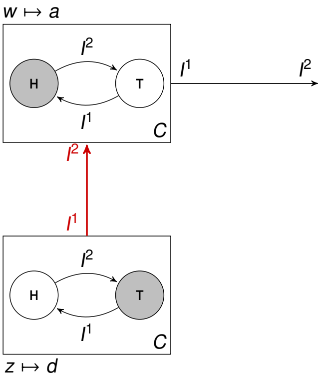


$$(\mathfrak{S}_0, \varsigma, \nu) \bullet (\mathfrak{S}_1, \varsigma, \nu)$$

Behavioral Semantics



Behavioral Semantics



Behavioral Semantics

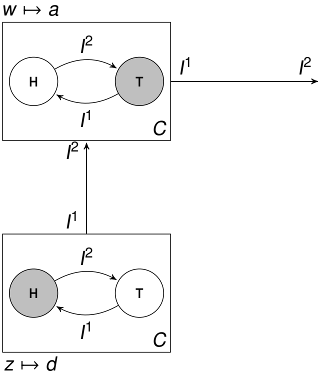
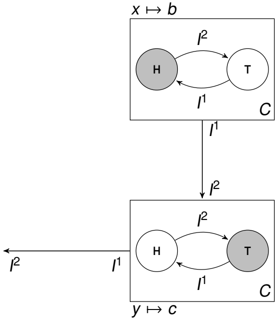


Table of Contents

Separation Logic on BIP

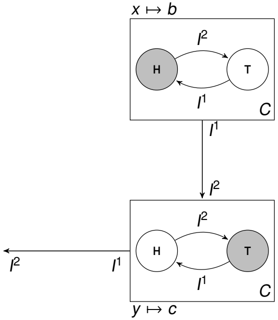
$$\phi ::= \mathtt{emp} \mid C_i(x) \mid I_j(x_1, \ldots, x_{\alpha(j)}) \mid \mathsf{state}(x, s) \mid A(t_1, \ldots, t_{\alpha(A)}) \mid \mathtt{true} \mid \neg\phi \mid \phi * \psi \mid \phi \wedge \psi \mid \exists x.\phi,$$

Semantics of Separation Logic on BIP



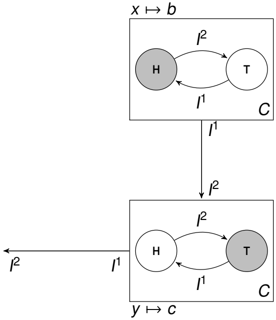
$$(\mathfrak{S}, \varsigma, \nu) \models C(x) * I(x, y) * C(y) * I(y, z)$$

Semantics of Separation Logic on BIP



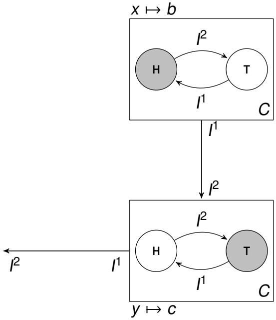
$$(\mathfrak{S}, \varsigma, \nu) \models C(x) * I(x, y) * C(y) * I(y, z)$$
$$(\mathfrak{S}, \varsigma, \nu) \not\models C(x)$$

Semantics of Separation Logic on BIP



$$(\mathfrak{S}, \varsigma, \nu) \models C(x) * I(x, y) * C(y) * I(y, z)$$
$$(\mathfrak{S}, \varsigma, \nu) \not\models C(x)$$
$$(\mathfrak{S}, \varsigma, \nu) \models C(x) * I(x, y) * \mathtt{true}$$

Semantics of Separation Logic on BIP



$$(\mathfrak{S}, \varsigma, \nu) \models C(x) * I(x, y) * C(y) * I(y, z)$$
$$(\mathfrak{S}, \varsigma, \nu) \not\models C(x)$$
$$(\mathfrak{S}, \varsigma, \nu) \models C(x) * I(x, y) * \mathtt{true}$$
$$(\mathfrak{S}, \varsigma, \nu) \models C(x) * I(x, y) * \mathtt{true} \wedge \mathrm{state}(x, \textsc{h}) \wedge \mathrm{state}(y, \textsc{t})$$

Inductive Predicates

$$\mathsf{chain}(x, x) \leftarrow \mathtt{emp},$$
$$\mathsf{chain}(x, z) \leftarrow \exists y \;.\; C(x) * I(x, y) * \mathsf{chain}(y, z)$$

Inductive Predicates

$$\mathsf{chain}(x, x) \leftarrow \mathtt{emp},$$
$$\mathsf{chain}(x, z) \leftarrow \exists y \;.\; C(x) * I(x, y) * \mathsf{chain}(y, z)$$

chain

Inductive Predicates

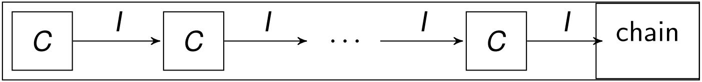
$$\mathsf{chain}(x, x) \leftarrow \mathtt{emp},$$
$$\mathsf{chain}(x, z) \leftarrow \exists y \,.\, C(x) * I(x, y) * \mathsf{chain}(y, z)$$

Inductive Predicates

$$\text{chain}(x, x) \leftarrow \mathtt{emp},$$
$$\text{chain}(x, z) \leftarrow \exists y\ .\ C(x) * I(x, y) * \text{chain}(y, z)$$

Inductive Predicates

$$\mathsf{chain}(x, x) \leftarrow \mathtt{emp},$$
$$\mathsf{chain}(x, z) \leftarrow \exists y \;.\; C(x) * I(x, y) * \mathsf{chain}(y, z)$$



Inductive Predicates

$$\mathsf{chain}(x, x) \leftarrow \mathtt{emp},$$
$$\mathsf{chain}(x, z) \leftarrow \exists y \,.\, C(x) * I(x, y) * \mathsf{chain}(y, z)$$

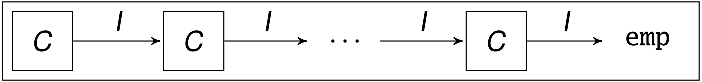


Table of Contents

Reconfiguration Language on BIP

$$
\begin{aligned}
\ell \ ::= \ & \texttt{new}(C_i, x) \mid \texttt{delete}(C_i, x) \mid \texttt{connect}(I_j, x_1, \ldots, x_{\alpha(j)}) \mid \\
& \texttt{disconnect}(I_j, x_1, \ldots, x_{\alpha(j)}) \mid \texttt{skip} \mid \\
& \texttt{when}\ \phi\ \texttt{do}\ \ell \mid \texttt{with}\ \psi\ \texttt{do}\ \ell \mid \ell; \ell' \mid \ell + \ell' \mid \ell^*
\end{aligned}
$$

Hoare Triple

Hoare Triple

For $P, Q \in \mathsf{SL}^{\mathsf{BIP}}_{\mathcal{R}}\langle\, \mathcal{C}, \mathcal{I}\,\rangle$, $\ell \in \mathcal{L}\langle\, \mathcal{C}, \mathcal{I}\,\rangle$:

$$\{\, P \,\}\ \ell\ \{\, Q \,\}.$$

Valid Hoare Triple

For all $(\mathfrak{S}, \varsigma, \nu) \in \Sigma_{\langle\, \mathcal{C}, \mathcal{I}\,\rangle}$

$$(\mathfrak{S}, \varsigma, \nu) \models P \quad \text{implies} \quad (\mathfrak{S}', \varsigma', \nu') \models Q$$

for all $(\mathfrak{S}', \varsigma', \nu') \in [\![\ell]\!](\mathfrak{S}, \varsigma, \nu)$.

Semantics of Reconfiguration Language on BIP

- $\texttt{new}(C_i, x)$
  $\Rightarrow \{ \texttt{emp} \} \; \texttt{new}(C_i, x) \; \{ C_i(x) \wedge \mathsf{state}(x, s_i^0) \}$

Semantics of Reconfiguration Language on BIP

- $\texttt{new}(C_i, x)$
  $\Rightarrow \{\ \texttt{emp}\ \}\ \texttt{new}(C_i, x)\ \{\ C_i(x) \wedge \mathsf{state}(x, s_i^0)\ \}$
- $\texttt{delete}(C_i, x)$
  $\Rightarrow \{\ C_i(x)\ \}\ \texttt{delete}(C_i, x)\ \{\ \texttt{emp}\ \}$

Semantics of Reconfiguration Language on BIP

- $\texttt{new}(C_i, x)$
  $\Rightarrow \{\ \texttt{emp}\ \}\ \texttt{new}(C_i, x)\ \{\ C_i(x) \wedge \mathsf{state}(x, s_i^0)\ \}$
- $\texttt{delete}(C_i, x)$
  $\Rightarrow \{\ C_i(x)\ \}\ \texttt{delete}(C_i, x)\ \{\ \texttt{emp}\ \}$
- $\texttt{connect}(I_j, x, y)$
  $\Rightarrow \{\ \texttt{emp}\ \}\ \texttt{connect}(I_j, x_1, \ldots, x_{\alpha(j)})\ \{\ I_j(x_1, \ldots, x_{\alpha(j)})\ \}$

Semantics of Reconfiguration Language on BIP

- $\texttt{new}(C_i, x)$
  $\Rightarrow \{ \text{ emp } \} \ \texttt{new}(C_i, x) \ \{ \ C_i(x) \wedge \text{state}(x, s_i^0) \ \}$
- $\texttt{delete}(C_i, x)$
  $\Rightarrow \{ \ C_i(x) \ \} \ \texttt{delete}(C_i, x) \ \{ \text{ emp } \}$
- $\texttt{connect}(I_j, x, y)$
  $\Rightarrow \{ \text{ emp } \} \ \texttt{connect}(I_j, x_1, \ldots, x_{\alpha(j)}) \ \{ \ I_j(x_1, \ldots, x_{\alpha(j)}) \ \}$
- $\texttt{disconnect}(I_j, x, y)$
  $\Rightarrow \{I_j(x_1, ..., x_{\alpha(j)})\} \ \texttt{disconnect}(I_j, x_1, ..., x_{\alpha(j)}) \ \{\text{emp}\}$

Semantics of Reconfiguration Language on BIP

- $\texttt{new}(C_i, x)$
  $\Rightarrow \{\text{ emp }\}\ \texttt{new}(C_i, x)\ \{\ C_i(x) \wedge \text{state}(x, s_i^0)\ \}$
- $\texttt{delete}(C_i, x)$
  $\Rightarrow \{\ C_i(x)\ \}\ \texttt{delete}(C_i, x)\ \{\text{ emp }\}$
- $\texttt{connect}(I_j, x, y)$
  $\Rightarrow \{\text{ emp }\}\ \texttt{connect}(I_j, x_1, \ldots, x_{\alpha(j)})\ \{\ I_j(x_1, \ldots, x_{\alpha(j)})\ \}$
- $\texttt{disconnect}(I_j, x, y)$
  $\Rightarrow \{I_j(x_1, ..., x_{\alpha(j)})\}\ \texttt{disconnect}(I_j, x_1, ..., x_{\alpha(j)})\ \{\text{emp}\}$
- $\texttt{skip}$
  $\Rightarrow \{\ P\ \}\ \texttt{skip}\ \{\ P\ \}$

Semantics of Reconfiguration Language on BIP

- when ϕ do ℓ

$$\Rightarrow \frac{\{\ P \wedge \phi\ \}\ \ell\ \{\ Q\ \}}{\{\ P\ \}\ \texttt{when}\ \phi\ \texttt{do}\ \ell\ \{\ Q\ \}}$$

Semantics of Reconfiguration Language on BIP

- $\texttt{when}\ \phi\ \texttt{do}\ \ell$

$$\Rightarrow \frac{\{\ P \wedge \phi\ \}\ \ell\ \{\ Q\ \}}{\{\ P\ \}\ \texttt{when}\ \phi\ \texttt{do}\ \ell\ \{\ Q\ \}}$$

- $\texttt{with}\ \psi\ \texttt{do}\ \ell$

$$\Rightarrow \frac{\{\ \exists y_1, \ldots, y_i.\ P \wedge \psi[x_1/y_1, \ldots, x_i/y_i] * \texttt{true}\ \}\ \ell\ \{\ Q\ \}}{\{\ P\ \}\ \texttt{with}\ \psi\ \texttt{do}\ \ell\ \{\ Q\ \},}$$

where $\{x_1, \ldots, x_i\} \subseteq \mathrm{fv}(\psi)$ and $y_1, \ldots, y_n \in \mathcal{V}$,

Structural Reconfiguration Rules

$$\frac{\{\ P\ \}\ \ell_0\ \{\ P'\ \} \qquad \{\ P'\ \}\ \mathtt{havoc}\ \{\ Q'\ \} \qquad \{\ Q'\ \}\ \ell_1\ \{\ Q\ \}}{\{\ P\ \}\ \ell_0; \ell_1\ \{\ Q\ \}},$$

Structural Reconfiguration Rules

$$\frac{\{\ P\ \}\ \ell_0\ \{\ P'\ \} \qquad \{\ P'\ \}\ \mathtt{havoc}\ \{\ Q'\ \} \qquad \{\ Q'\ \}\ \ell_1\ \{\ Q\ \}}{\{\ P\ \}\ \ell_0;\ell_1\ \{\ Q\ \}},$$

$$\frac{\{\ P\ \}\ \ell_0\ \{\ Q\ \} \qquad \{\ P\ \}\ \ell_1\ \{\ Q\ \}}{\{\ P\ \}\ \ell_0 + \ell_1\ \{\ Q\ \}},$$

Structural Reconfiguration Rules

$$\frac{\{\,P\,\}\ \ell_0\ \{\,P'\,\} \qquad \{\,P'\,\}\ \mathtt{havoc}\ \{\,Q'\,\} \qquad \{\,Q'\,\}\ \ell_1\ \{\,Q\,\}}{\{\,P\,\}\ \ell_0;\ell_1\ \{\,Q\,\}},$$

$$\frac{\{\,P\,\}\ \ell_0\ \{\,Q\,\} \qquad \{\,P\,\}\ \ell_1\ \{\,Q\,\}}{\{\,P\,\}\ \ell_0+\ell_1\ \{\,Q\,\}}, \qquad \frac{\{\,P\,\}\ \ell\ \{\,P\,\} \qquad \{\,P\,\}\ \mathtt{havoc}\ \{\,P\,\}}{\{\,P\,\}\ \ell^*\ \{\,P\,\}},$$

Frame Rule

$$\frac{\{\ P\ \}\ \ell\ \{\ Q\ \}}{\{\ P * F\ \}\ \ell\ \{\ Q * F\ \}}$$

where

- $\mathrm{Modifies}(\ell) \cap \mathrm{fv}(F) = \emptyset$,
- ℓ does not contain `with` ψ `do`, sequential composition, and the Kleene operator.

Frame Rule

$$\frac{\{\ P\ \}\ \ell\ \{\ Q\ \}}{\{\ P * F\ \}\ \ell\ \{\ Q * F\ \}}$$

where

- $\mathrm{Modifies}(\ell) \cap \mathrm{fv}(F) = \emptyset$,
- ℓ does not contain `with` ψ `do`, sequential composition, and the Kleene operator.

Theorem

The reconfiguration rules are sound.

Table of Contents

Havoc Triple

Havoc Triple

For $P, Q \in \mathrm{SL}_{\mathcal{R}}^{\mathrm{BIP}}\langle\, \mathcal{C}, \mathcal{I}\,\rangle$ and L is language over alphabet Σ:

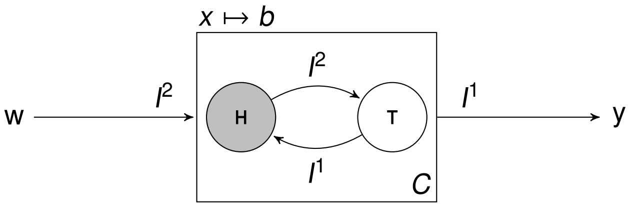
$$\{\, P \,\}\ L\ \{\, Q \,\}.$$

Valid Havoc Triple

For all $(\mathfrak{S}, \varsigma, \nu), (\mathfrak{S}, \varsigma', \nu) \in \Sigma_{\langle\, \mathcal{C}, \mathcal{I}\,\rangle}$

> $(\mathfrak{S}, \varsigma, \nu) \models P$ *and* $(\mathfrak{S}, \varsigma, \nu) \overset{w}{\rightsquigarrow}_o (\mathfrak{S}, \varsigma', \nu)$ *for some* $w \in L$ *implies* $(\mathfrak{S}, \varsigma', \nu) \models Q$.

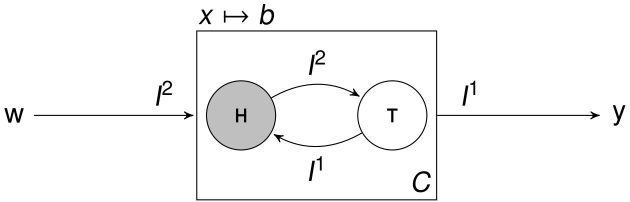
Examples of Havoc Triples



$$P := I(w, x) * C(x) * I(x, y)$$

$$\{\ P \wedge \text{state}(x, \text{H})\ \}\ \ I(w, x)\ \ \{\ P \wedge \text{state}(x, \text{T})\ \}$$

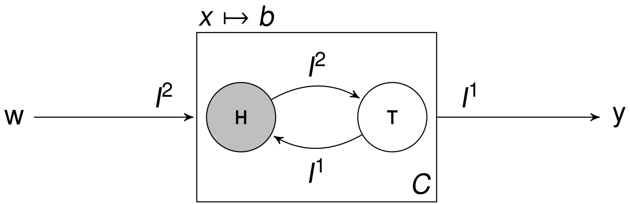
Examples of Havoc Triples



$$P := I(w, x) * C(x) * I(x, y)$$

$$\{\ P \wedge \text{state}(x, \text{H})\ \}\ \ I(w, x)\ \ \{\ P \wedge \text{state}(x, \text{T})\ \}$$
$$\{\ P \wedge \text{state}(x, \text{H})\ \}\ \ I(x, y)\ \ \{\ \texttt{false}\ \}$$

Examples of Havoc Triples



$$P := I(w,x) * C(x) * I(x,y)$$

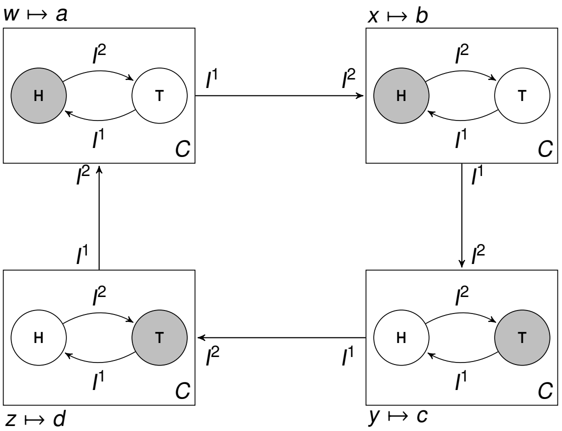
$$\{\ P \wedge \mathrm{state}(x, \textsc{h})\ \}\ \ I(w,x)\ \ \{\ P \wedge \mathrm{state}(x, \textsc{t})\ \}$$
$$\{\ P \wedge \mathrm{state}(x, \textsc{h})\ \}\ \ I(x,y)\ \ \{\ \mathtt{false}\ \}$$
$$\{\ P \wedge \mathrm{state}(x, \textsc{h})\ \}\ \ (I(w,x) \cdot I(x,y))^*\ \ \{\ P \wedge \mathrm{state}(x, \textsc{h})\ \}$$

Selection of Havoc Rules

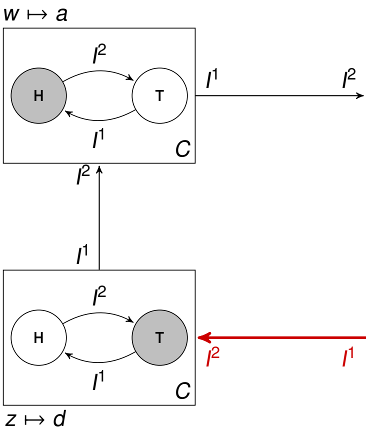
$$\frac{}{\{\ P\ \}\ \epsilon\ \{\ P\ \}}\ (\epsilon) \qquad \frac{\{\ P\ \}\ L_1\ \{\ Q\ \} \qquad \{\ Q\ \}\ L_2\ \{\ R\ \}}{\{\ P\ \}\ L_1 \cdot L_2\ \{\ R\ \}}\ (\cdot)$$

$$\frac{\{\ P\ \}\ L_1\ \{\ Q\ \} \qquad \{\ P\ \}\ L_2\ \{\ Q\ \}}{\{\ P\ \}\ L_1 \cup L_2\ \{\ R\ \}}\ (\cup) \qquad \frac{\{\ P\ \}\ L\ \{\ P\ \}}{\{\ P\ \}\ L^*\ \{\ P\ \}}\ (*)$$

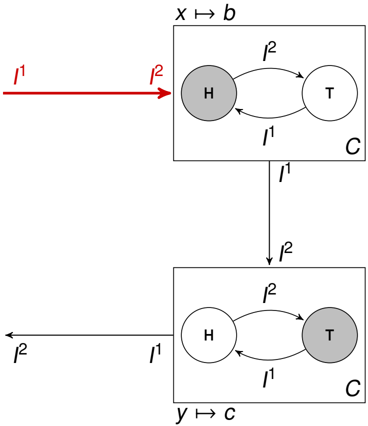
Frontier



Frontier



Frontier



Composition Rule

$$\frac{\{\, P_1 * \mathcal{F}\ (P_1, P_2)\}\ L_1\ \{\, Q_1 * \mathcal{F}\ (P_1, P_2)\} \qquad \{\, P_2 * \mathcal{F}\ (P_2, P_1)\}\ L_2\ \{\, Q_2 * \mathcal{F}\ (P_2, P_1)\}}{\{\, P_1 * P_2\,\}\ L_1 \bowtie L_2\ \{\, Q_1 * Q_2\,\}}\ (\bowtie)$$

Composition Rule

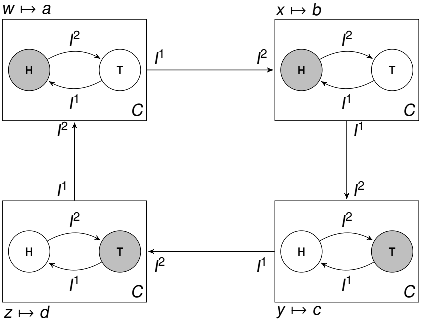
$$\frac{\{\ P_1 * \mathcal{F}\ (P_1, P_2)\}\ L_1\ \{\ Q_1 * \mathcal{F}\ (P_1, P_2)\} \quad \{\ P_2 * \mathcal{F}\ (P_2, P_1)\}\ L_2\ \{\ Q_2 * \mathcal{F}\ (P_2, P_1)\}}{\{\ P_1 * P_2\ \}\ L_1 \bowtie L_2\ \{\ Q_1 * Q_2\ \}}\ (\bowtie)$$

Theorem

The havoc rules are sound.

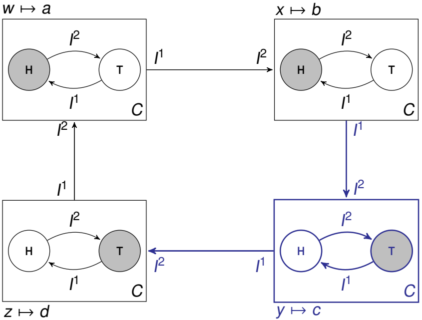
Table of Contents

Application on Token Ring




```
with I(x,y) ∗ C(y) ∗ I(y,z) ∧ state(y,T) do
    disconnect(I,y,z);
    disconnect(I,x,y);
    delete(C,y);
    connect(I,x,z)
```

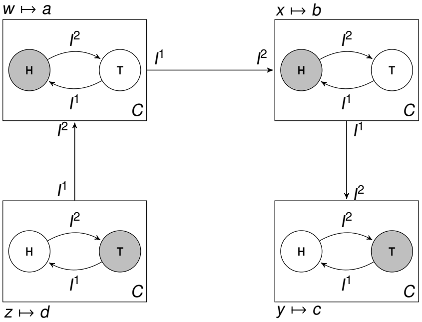
Application on Token Ring



with $l(x,y) * C(y) * l(y,z) \wedge \mathrm{state}(y,\mathrm{T})$ **do**

```
    disconnect(I,y,z);
    disconnect(I,x,y);
    delete(C,y);
    connect(I,x,z)
```

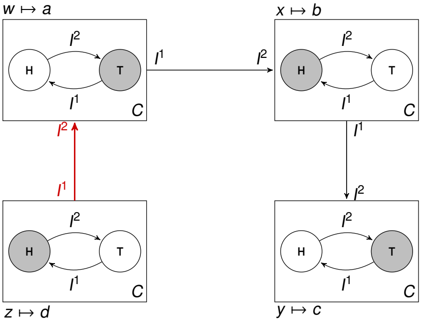
Application on Token Ring



with $I(x,y) * C(y) * I(y,z) \wedge \text{state}(y, \mathrm{T})$ **do**

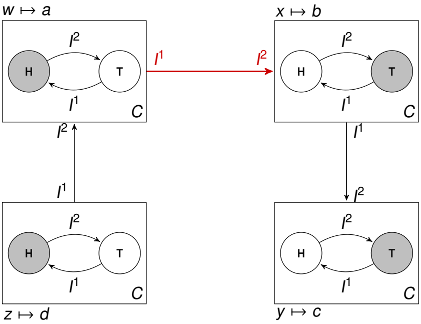
```
  disconnect(I,y,z);
  disconnect(I,x,y);
  delete(C,y);
  connect(I,x,z)
```

Application on Token Ring



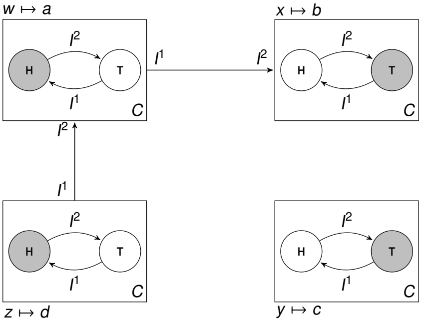

```
with I(x,y) * C(y) * I(y,z) ∧ state(y, T) do
    disconnect(I,y,z);
    disconnect(I,x,y);
    delete(C,y);
    connect(I,x,z)
```

Application on Token Ring



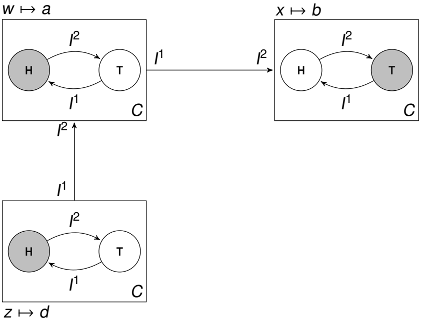

```
with I(x,y) * C(y) * I(y,z) ∧ state(y,T) do
    disconnect(I,y,z);
    disconnect(I,x,y);
    delete(C,y);
    connect(I,x,z)
```

Application on Token Ring




```
with I(x, y) ∗ C(y) ∗ I(y, z) ∧ state(y, T) do
   disconnect(I,y,z);
   disconnect(I,x,y);
   delete(C,y);
   connect(I,x,z)
```

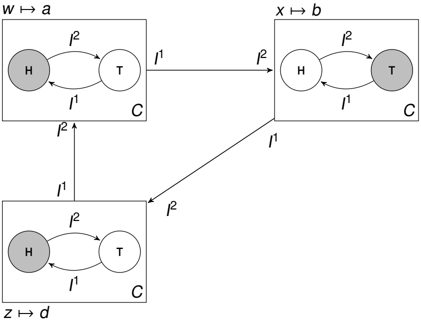
Application on Token Ring



with $I(x, y) * C(y) * I(y, z) \wedge \mathrm{state}(y, \mathrm{T})$ **do**

```
   disconnect(I,y,z);
   disconnect(I,x,y);
   delete(C,y);
   connect(I,x,z)
```

Application on Token Ring




```
with I(x,y) ∗ C(y) ∗ I(y,z) ∧ state(y,T) do
   disconnect(I,y,z);
   disconnect(I,x,y);
   delete(C,y);
   connect(I,x,z)
```

Correctness of Reconfiguration Program

Theorem

The reconfiguration program $P_{\texttt{delete}}$ is correct, meaning that

$$\{\ \mathsf{token\_ring}^{\top}(a)\ \}\ P_{\texttt{delete}}\ \{\ \mathsf{token\_ring}(a)\ \}.$$

Correctness of Reconfiguration Program

$F := [\mathsf{chain}^*(z, x, h-1, t) \wedge \mathsf{state}(x, \text{н})] \vee [\mathsf{chain}^*(z, x, h, t-1) \wedge \mathsf{state}(x, \text{т})]$

$\{ \mathsf{token\_ring}^{\text{T}}(a) \}$
$\{ \exists x, y, z.\ \ C(x) * I(x,y) * C(y) * I(y,z) * F \wedge \mathsf{state}(y, \text{т}) \}$
with $I(x,y) * C(y) * I(y,z) \wedge \mathsf{state}(y, \text{т})$ **do**

```
disconnect(I,y,z)
```

$\{ C(x) * I(x,y) * C(y) * F \wedge \mathsf{state}(y, \text{т}) \}$

```
havoc
```

$\{ C(x) * I(x,y) * C(y) * F \wedge \mathsf{state}(y, \text{т}) \}$

```
disconnect(I,x,y)
```

$\{ C(x) * C(y) * F \wedge \mathsf{state}(y, \text{т}) \}$

```
havoc
```

$\{ C(x) * C(y) * F \wedge \mathsf{state}(y, \text{т}) \}$

```
delete(C,y)
```

$\{ C(x) * F \}$

```
havoc
```

$\{ C(x) * F \}$

```
connect(I,x,z)
```

$\{ C(x) * I(x,z) * F \}$
$\{ \mathsf{token\_ring}(x) \}$

Conlusion

Achievements

- BIP Configurations
- Separation Logic on BIP
- Reconfiguration Language on BIP
- Inference rules
- Correctness of reconfiguration programs on token rings $\rightarrow$ the resulting configuration is still deadlock-free

Future Work

- Completeness of inference rules
- Apply on *dining philosophers* problem
- Proof correctness of reconfiguration programs for other systems
- Automatize proofs

References I

Ananda Basu, Saddek Bensalem, Marius Bozga, Jacques Combaz, Mohamad Jaber, Thanh-Hung Nguyen, and Joseph Sifakis.
Rigorous component-based system design using the BIP framework.
IEEE Softw., 28(3):41–48, 2011.

Marius Bozga and Radu Iosif.
Verifying safety properties of inductively defined parameterized systems.
CoRR, abs/2008.04160, 2020.

Cristiano Calcagno, Peter W. O'Hearn, and Hongseok Yang.
Local action and abstract separation logic.
In *22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, Proceedings*, pages 366–378. IEEE Computer Society, 2007.

References II

C. A. R. Hoare.
An axiomatic basis for computer programming.
Commun. ACM, 12(10):576–580, 1969.

Peter W. O'Hearn, John C. Reynolds, and Hongseok Yang.
Local reasoning about programs that alter data structures.
In Laurent Fribourg, editor, *Computer Science Logic, 15th International Workshop, CSL 2001. 10th Annual Conference of the EACSL, Paris, France, September 10-13, 2001, Proceedings*, volume 2142 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2001.

John C. Reynolds.
Separation logic: A logic for shared mutable data structures.
In *17th IEEE Symposium on Logic in Computer Science (LICS 2002), 22-25 July 2002, Copenhagen, Denmark, Proceedings*, pages 55–74. IEEE Computer Society, 2002.