

---

---

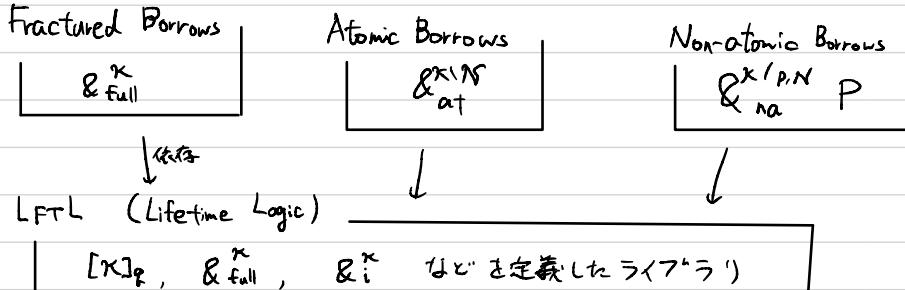
---

---

---



# Chapter 11



上3つを borrow

Frac

Atomic

Non-atomic

◦ persistent persistent ( $\&_{\text{frac}}^{\kappa} \emptyset$ ) persistent ( $\&_{\text{at}}^{\kappa/\text{N}} P$ ) persistent ( $\&_{\text{na}}^{\kappa/\text{p.N}} P$ )

◦ share  $\&_{\text{full}}^{\kappa} \Phi(1) \not\equiv_{N_{\text{lit}}} \&_{\text{frac}}^{\kappa} \emptyset$   $(N \# N_{\text{lit}}) * \&_{\text{full}}^{\kappa} P \not\equiv \&_{\text{at}}^{\kappa/\text{N}} P$   $\&_{\text{full}}^{\kappa} P \not\equiv \&_{\text{na}}^{\kappa/\text{p.N}} P$

$\&_{\text{full}}^{\kappa}$  が  $\&_{\text{frac}}^{\kappa}$  より強制的。

◦ monotone

$$\frac{\kappa' \subseteq \kappa}{\&_{\text{frac}}^{\kappa} \emptyset \rightarrow \&_{\text{frac}}^{\kappa'} \emptyset}$$

$$\frac{\kappa' \subseteq \kappa}{\&_{\text{at}}^{\kappa/\text{N}} P \rightarrow \&_{\text{at}}^{\kappa'/\text{N}} P}$$

$$\frac{\kappa' \subseteq \kappa, N \subseteq \kappa'}{\&_{\text{na}}^{\kappa/\text{p.N}} P \rightarrow \&_{\text{na}}^{\kappa'/\text{p.N}} P}$$

◦ ACC

$$\begin{array}{c} \text{LFTL-FRACT-ACC} \\ \Box \forall q_1, q_2. \Phi(q_1 + q_2) ** \Phi(q_1) * \Phi(q_2) \\ \&_{\text{frac}}^{\kappa} \Phi \rightarrow ([\kappa]_q \propto_{N_{\text{lit}}} q'. \triangleright \Phi(q')) \end{array}$$

LFTL-AT-ACC-TOK

$$\&_{\text{at}}^{\kappa/\text{N}} P \rightarrow ([\kappa]_q \propto_{N_{\text{lit}}, N} N_{\text{lit}} \triangleright P)$$

LFTL-NA-ACC

$$\&_{\text{na}}^{\kappa/\text{p.N}} P \rightarrow ([\kappa]_q * [N_{\text{lit}} : p, N] \propto_{N_{\text{lit}}, N} \triangleright P)$$

LFTL-FRACT-ACC-ATOMIC

$$\&_{\text{frac}}^{\kappa} \Phi \rightarrow (\text{True} \propto_{N_{\text{lit}}} \emptyset, b, q. \text{ifthenelse}(b, \triangleright \Phi(q), [\dagger \kappa]))$$

Lifetime taken おまけ  $\triangleright P$  は一時的 (= 得意なもん)。

indexed borrow  $\&_i^\kappa P$  (lifetime logic  $\infty_{\text{Borrow}}$  primitive & borrow)

LFTL-BOR-IDX

$$\&_{\text{full}}^\kappa P ** \exists i. \&_i^\kappa P * [\text{Bor} : i]_1$$

LFTL-IDX-PERSIST

$$\text{persistent}(\&_i^\kappa P)$$

LFTL-IDX-TIMELESS

$$\text{timeless}([\text{Bor} : i]_q)$$

LFTL-IDX-IFF

$$\triangleright \square(P ** Q) \\ \frac{}{\&_i^\kappa P * \&_i^\kappa Q}$$

LFTL-IDX-SHORTEN

$$\frac{\kappa' \sqsubseteq \kappa}{\&_i^\kappa P * \&_i^{\kappa'} P}$$

LFTL-IDX-FRACT

$$[\text{Bor} : i]_{q+q'} ** [\text{Bor} : i]_q * [\text{Bor} : i]_{q'}$$

LFTL-IDX-ACC

$$\&_i^\kappa P * ([\text{Bor} : i]_1 * [\kappa]_q \propto_{N_{\text{ft}}} \triangleright P)$$

LFTL-IDX-ACC-ATOMIC

$$\&_i^\kappa P * ([\text{Bor} : i]_q \stackrel{N_{\text{ft}} \propto \emptyset}{=} b. \text{ifthenelse}(b, \triangleright P, [\dagger \kappa]))$$

LFTL-IDX-BOR-UNNEST

$$\&_i^\kappa P * \&_{\text{full}}^{\kappa'} ([\text{Bor} : i]_1) \equiv_{N_{\text{ft}}} \&_{\text{full}}^{\kappa \sqcap \kappa'} P$$

Borrow と 2 種類の従事

1. persistent
2.  $\&_{\text{full}}^\kappa P$  の  $([\text{Bor} : i]_1, \text{共用})$  得た人。
3.  $\kappa' \sqsubseteq \kappa$  のとき lifetime  $\in \kappa'$  の  $\&_i^\kappa P$  の人。
4. 2 つとも accesser

Cancellative Invariant の種類

$$\&_i^\kappa P \Leftarrow C_{\text{Inv}}^{r, N}(P)$$

$$[\text{Bor} : i]_q \Leftarrow [C_{\text{Inv}} : r]_q$$

$$\circ \&_{\text{full}}^\kappa P \xrightarrow{*} \exists i. \underbrace{\&_i^\kappa P * [\text{Bor} : i]_1}_{\text{persistent}} \quad \underbrace{[\text{Bor} : i]_1}_{\text{timeless}}$$

$$\circ \&_i^\kappa P \longrightarrow ([\text{Bor} : i]_1 * [x]_q \propto_{N_{\text{ft}}} \triangleright P)$$

$$\bullet \triangleright P \Rightarrow_{\epsilon} \exists r. \underbrace{C_{\text{Inv}}^{r, N}(P)}_{\text{persistent}} * \underbrace{[C_{\text{Inv}} : r]_1}_{\text{timeless}}$$

$$\bullet \frac{N \subseteq E}{C_{\text{Inv}}^{r, N}(P) \vdash [C_{\text{Inv}} : r]_q \propto_{E^{\text{Inv}}} \triangleright P}$$

CINV-TIMELESS	CINV-PERSISTENT	CINV-SPLIT
$\text{timeless}([\text{CInv} : \gamma]_q)$	$\text{persistent}(\text{CInv}^{\gamma, \mathcal{N}}(P))$	$[\text{CInv} : \gamma]_{q_1+q_2} \Leftrightarrow [\text{CInv} : \gamma]_{q_1} * [\text{CInv} : \gamma]_{q_2}$
CINV-VALID	CINV-ALLOC	CINV-CANCEL
$[\text{CInv} : \gamma]_q \Rightarrow q \leq 1$	$\triangleright P \Rightarrow_{\mathcal{E}} \exists \gamma. [\text{CInv} : \gamma]_1 * \text{CInv}^{\gamma, \mathcal{N}}(P)$	$\frac{\mathcal{N} \subseteq \mathcal{E}}{\text{CInv}^{\gamma, \mathcal{N}}(P) * [\text{CInv} : \gamma]_1 \Rightarrow_{\mathcal{E}} \triangleright P}$
VS-CINV		
	$\frac{\triangleright P * Q_1 \Rightarrow_{\mathcal{E} \setminus \mathcal{N}} \triangleright P * Q_2 \quad \mathcal{N} \subseteq \mathcal{E}}{\text{CInv}^{\gamma, \mathcal{N}}(P) * [\text{CInv} : \gamma]_q * Q_1 \Rightarrow_{\mathcal{E}} \text{CInv}^{\gamma, \mathcal{N}}(P) * [\text{CInv} : \gamma]_q * Q_2}$	
HOARE-CINV		
	$\frac{\{\triangleright P * Q_1\} e \{\triangleright P * Q_2\}_{\mathcal{E} \setminus \mathcal{N}} \quad \text{atomic}(e) \quad \mathcal{N} \subseteq \mathcal{E}}{\left\{ \text{CInv}^{\gamma, \mathcal{N}}(P) * [\text{CInv} : \gamma]_q * Q_1 \right\} e \left\{ \text{CInv}^{\gamma, \mathcal{N}}(P) * [\text{CInv} : \gamma]_q * Q_2 \right\}_{\mathcal{E}}}$	

## 5.2 Cancellable invariants

- Cancellable invariants: 一時的に share される invariant で、のちに取り除かれるもの。
- (Fractional points-to と似たように、) この場合は、**fractional token** が invariant に equip できる。(考えられる、的な意味か?)
- Access する権利は分割できるが、cancel するためには full ownership が必要。
- $\text{CInv}^{\gamma, \mathcal{N}}(P)$ : 命題 P に対しての cancellable invariant の存在を表現する。
  - $\mathcal{N}$ : 名前空間 (なので、気にしなくていいと思われる)
  - $\gamma$ : ghost identifier。token と invariant を結びつける。
- $[\text{CInv} : \gamma]_q$ : 有理数  $q$  の分の token の所有権。 $[\dots]$  が token。

11.5.1

- Atomic borrow

$$\&_{\text{at}}^{\kappa/N} P := \exists_i (N = N_{\text{left}}) * \&_i^\kappa P * \boxed{[Bor:i]_1}^N$$

P・借用が作成した 全体の借用が "N" に  
locateされている。

→ Nを持っている ⇔ 借りOK.

11.5.2

- Fractured borrow

$$\&_{\text{frac}}^\kappa \emptyset := \exists_{K', r, i, (\kappa \subseteq K')} * \boxed{\exists_{q_i} [Bor:i]_{q_i}} * \&_i^\kappa I_{\emptyset}(K')$$

借用の一部が "allocated" している。

$$I_{\emptyset}(K') := \exists_{q_i} \emptyset(q_i) * \boxed{q_i} * (q_i = (\vee [\kappa']_{i-1}))$$

重複の一部を借用している

(残っていた  $\&_{\text{full}}^\kappa$  on rule)

Primitive rules.

LFTL-BOR-MERGE

$$\&_{\text{full}}^\kappa P * \&_{\text{full}}^\kappa Q \Rightarrow_{N_{\text{lit}}} \&_{\text{full}}^\kappa (P * Q)$$

LFTL-BOR-FAKE

$$[\dagger \kappa] \Rightarrow_{N_{\text{lit}}} \&_{\text{full}}^\kappa P$$

LFTL-BOR-ACC-STRONG

$$\&_{\text{full}}^\kappa P * [\kappa]_q \Rightarrow_{N_{\text{lit}}} \exists \kappa'. \kappa \sqsubseteq \kappa' * \triangleright P * \left( \forall Q. \triangleright (\triangleright Q * [\dagger \kappa']) \Rightarrow_{\emptyset} \triangleright P \right) * \triangleright Q \Rightarrow_{N_{\text{lit}}} \&_{\text{full}}^{\kappa'} Q * [\kappa]_q$$

LFTL-BOR-ACC-ATOMIC-STRONG

$$\&_{\text{full}}^\kappa P \stackrel{N_{\text{lit}}}{\Rightarrow} \emptyset \left( \exists \kappa'. \kappa \sqsubseteq \kappa' * \triangleright P * \left( \forall Q. \triangleright (\triangleright Q * [\dagger \kappa']) \Rightarrow_{\emptyset} \triangleright P \right) * \triangleright Q \stackrel{\emptyset}{\Rightarrow} \&_{\text{full}}^{\kappa'} Q \right) \vee \\ ([\dagger \kappa] * \stackrel{N_{\text{lit}}}{\Rightarrow} \text{True})$$

Derived rules.

LFTL-BOR-UNNEST

$$\&_{\text{full}}^{\kappa'} \&_{\text{full}}^\kappa P \Rightarrow_{N_{\text{lit}}} \&_{\text{full}}^{\kappa' \sqcap \kappa} P$$

11.5.3 つまり証明。

## 11.6 Implementing the lifetime logic. (without reborrowing)

Iris で Lifetime Logic の実装をする (  $\&_{full}^k$ ,  $\&_i^k$ ,  $[x]_q$  など )  
を実装する話.

まずは、目的に合うような ... の resource algebra を作ってみる  
する話.

( 著者による reborrowing ができない version でやる )

Lifetime を何で表現するか.

基本的にはライフタイム  $K$  はただの identifier と扱えれば良い。  
( ghost state もみたい。) ライフタイム のため identifier で可算個用意すれば十分に思える。

( が ) L, lifetime は intersection である。

これを表現するため、atomic と lifetime の multiset で lifetime を表現する。

$(K = \{A_1, A_2, \dots, A_n\})$  は  $A_1 \sim A_n$  の intersection )

$\uparrow$   
 $K \cap K \neq K$  でなければならない。

$[K \cap K]_q \leftarrow [K]_q * [K]_q$  でいい。

すなはち、atomic と lifetime の集合 atLft ( $|ALft| = N_0$ ) は  $\neq L$ 、  
lifetime 全体 Lft は atomic と lifetime の multiset 全体の集合

$$Lft = \{ \{A_1, \dots, A_n\} \mid n \in \mathbb{N}, A_i \in atLft \}$$

## Global ghost state

Iris は primitive ルールは  $\Gamma \vdash \Delta \vdash t \rightsquigarrow \text{Own}(M)$  のみ  
 が持つときは  $\Gamma \vdash \Delta \vdash M$  と,  
 "ghost state は" allocate する heap 全体"

すなはち

$$M := \prod_{i \in I} N \xrightarrow{\text{fin}} M_i \quad (M_i \text{ は RA})$$

巨大な RA を持つときは,  
 global ghost state (§4.3)

似たような置き方をする。

"各 lifetime に対する ghost state を割り当てる heap"

の RA

$$\text{ILft} := \text{Auth}(\text{Lft} \xrightarrow{\text{fin}} \text{Ag}(N \times N))$$

( $K$  に対する  $\forall i$  と  $\forall i$  は変換する "share" である)

で、

"各 atomic lifetime に対する, "生む" が生む" の" "

を表す RA

$$\text{ALft} := \text{Auth}(\text{atLft} \xrightarrow{\text{fin}} \text{Frac} \dashv (\text{ }))$$

$\underbrace{\text{atLft}}_{\text{atomic lifetime}} \dashv \underbrace{\text{Frac}}_{\text{RA}}$   
 one-shot RA.

## Lifetime token

$$[K]_q := \bigotimes_{\Lambda \in K} \left[ \circ \boxed{\Lambda \leftarrow \text{inl}(\alpha)} \right]^{\alpha}$$

$$[+K] := \exists \Lambda \in K. \left[ \circ \boxed{\Lambda \leftarrow \text{inr}(\beta)} \right]^{\beta}$$

$\text{Auth}(M)$  は §4.4.2 で定義した RA.

$a \in M := \text{ALft}$ ,  $\bullet a$  のは,  $a$  a fragment を持つこと, とある。  
( $\leftrightarrow$  authoritative element)

$[\Lambda \leftarrow \text{inh}(\varrho)]$  は,  $\Lambda$  は  $\text{inh}(\varrho)$  が allocate されていて, 他に無の heap.  
つまり,  $\Lambda$  は  $\varrho$  だけ生きている, といふこと。  
 $\rightarrow$  fragment こと,  $\Lambda$  が 1 生きている ことを表す。

すなはち,  $[K]_{\varrho}$  は, 各  $\Lambda \in K$  が  $\varrho$  以上生きていること。

逆に,  $[+K]$  は, ある  $\Lambda \in K$  が死んでしまった。

## NameSpace

$N_{\text{eff}}$  &  $3\gamma$  に分解。

$N_{\text{mgmt}}$  : main management invariant

$N_{\text{bor}}$  : borrowed resources

$N_{\text{inh}}$  : inheritance

$\gamma_i$  : ILft が割り当てられた location

$\gamma_a$  : ALft

## Global invariant.

$LftLInv := \exists A \in ALft, I \in ILft. \quad [A]^{r_a} * [I]^{r_i}$   
 $* \bigwedge_{x \in \text{dom} I} LftInv(A, x)$

$LftLCtx := \boxed{LftInv}^{N_{\text{mgmt}}}$

この 2つは, 常に成立し続けるといふもの。

$LftInv(A, K)$  : "A ∈ ALft かつ K は正しく状況である"

$LftAliveIn(A, K)$  :=  $\forall \lambda \in K, \exists g. A(\lambda) = \text{inv}(g)$

$LftDeadIn(A, K)$  :=  $\exists \lambda \in K. A(\lambda) = \text{invr}()$

$LftInv(A, K) := LftAliveIn(A, K) * LftAliveIn(A, K)$

$\vee LftDeadIn(A, K) * LftDeadIn(A, K)$

borrow はどうあるか:

$\&_i^x, \&_{full}^x, [x]_e$  と PEs の x<sup>th</sup> 目的  $t_e^x, t_e$ .

・  $\&_i^x \in CInv^{T,N}$  は似ているという話があった.

・ borrow が作られるとき =  $\triangleright P \equiv_{N_{full}} \&_{full}^x P * ([+x] \equiv_{N_{full}} \triangleright P)$

↳ 何らかの P を含む invariant を作っている.

さて、同じ lifetime で複数の命題を管理せらる。

↳  $P_0 * P_1 * \dots * P_n$  など.

複数の命題の中のいくつかだけを取り出したりしたい → "slice"

↓  
box というものを使う.

box

$\text{Box}(N, P, f) \quad = \quad \text{BoxSlice}(N, Q, \iota)$

$\text{Box}(N, P, f) \quad N: \text{name space}$

$P: \text{prop}$

$f: N \xrightarrow{\text{fin}} \text{slice} \rightarrow \{\text{empty}, \text{full}\}$

Name space  $\mathcal{N}$  ~ total resource P a box & t's,

各 slice は a 状態は  $f(\iota)$ .

full  $\leftarrow$  2 slice の 1つは box は  $\lambda, \gamma$ 's.

empty  $\leftarrow$  2 slice が given out.

BoxSlice( $\mathcal{N}, Q, \iota$ )

$\mathcal{N}$ : n.s.,  $Q$ : prop,  $\iota \in \mathbb{N}$ .

サイズ  $\iota$  は  $Q$  を manage す。

$BorSt := \text{in} \mid \text{open}(q : Frac)$  (state of a borrow box)

$BorBox := \text{Auth}(\mathbb{N} \xrightarrow{\text{fin}} Ag(BorSt) \times Frac)$  (per-lifetime ghost state to manage borrow box;  $\gamma_{bor}$ )

$InhBox := \text{Auth}(\wp^{\text{fin}}(\mathbb{N}))$  (per-lifetime ghost state to manage inheritance box;  $\gamma_{inh}$ )

ons.

$\text{OwnBor}(\kappa, a) := \exists \gamma_{bor}. \circ[\kappa \leftarrow \gamma_{bor}, \_, \_]^{\gamma_i} * [a : BorBox]^{\gamma_{bor}}$

$\text{OwnInh}(\kappa, a) := \exists \gamma_{inh}. \circ[\kappa \leftarrow \_, \_, \gamma_{inh}]^{\gamma_i} * [a : InhBox]^{\gamma_{inh}}$

$[\text{Bor} : (\kappa', \iota)]_q := \text{OwnBor}(\kappa', \circ(\iota \leftarrow (\text{in}, q)))$

$\&_{(\kappa', \iota)}^{\kappa} P := \kappa \sqsubseteq \kappa' * \text{BoxSlice}(\mathcal{N}_{bor}, P, \iota)$

||

$\&_i^{\kappa} \quad (i \in \mathbb{N} \times \mathbb{N})$

$\text{RawBor}(\kappa, P) := \exists \iota. \text{BoxSlice}(\mathcal{N}_{bor}, P, \iota) * \text{OwnBor}(\kappa, \circ(\iota \leftarrow (\text{in}, 1)))$

$\&_{\text{full}}^{\kappa} P := \exists \kappa'. \kappa \sqsubseteq \kappa' * \text{RawBor}(\kappa', P)$

Basic proof rules.

<b>BOX-CREATE</b> $\downarrow$ $\text{True} \Rightarrow_{\mathcal{N}} \text{Box}(\mathcal{N}, \text{True}, \emptyset)$	<span style="color: blue;">又々復無</span> <b>SLICE-PERSIST</b> $\text{persistent}(\text{BoxSlice}(\mathcal{N}, Q, \iota))$
<b>SLICE-INSERT-EMPTY</b> $\triangleright^b \text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \exists \iota \notin \text{dom}(f). \text{BoxSlice}(\mathcal{N}, Q, \iota) * \triangleright^b \text{Box}(\mathcal{N}, P * Q, f[\iota \leftarrow \text{empty}])$	
<b>SLICE-DELETE-EMPTY</b> $\frac{\text{SLICE-FILL} \quad f(\iota) = \text{empty}}{\text{BoxSlice}(\mathcal{N}, Q, \iota) \vdash \triangleright^b \text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \exists P'. \triangleright^b (\triangleright(P = (P' * Q)) * \text{Box}(\mathcal{N}, P', f[\iota \leftarrow \perp]))}$	
<b>SLICE-FILL</b> $\frac{f(\iota) = \text{empty}}{\text{BoxSlice}(\mathcal{N}, Q, \iota) \vdash \triangleright^b Q * \triangleright \text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \triangleright^b \text{Box}(\mathcal{N}, P, f[\iota \leftarrow \text{full}])}$	
<b>SLICE-EMPTY</b> $\frac{f(\iota) = \text{full}}{\text{BoxSlice}(\mathcal{N}, Q, \iota) \vdash \triangleright^b \text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \triangleright Q * \triangleright^b \text{Box}(\mathcal{N}, P, f[\iota \leftarrow \text{empty}])}$	
<b>BOX-FILL</b> $\frac{\forall \iota \in \text{dom}(f). f(\iota) = \text{empty}}{\triangleright P * \text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \text{Box}(\mathcal{N}, P, f[\iota \leftarrow \text{full} \mid \iota \in \text{dom}(f)])}$	
<b>BOX-EMPTY</b> $\frac{\forall \iota \in \text{dom}(f). f(\iota) = \text{full}}{\text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \triangleright P * \text{Box}(\mathcal{N}, P, f[\iota \leftarrow \text{empty} \mid \iota \in \text{dom}(f)])}$	

Derived proof rules.

<b>SLICE-INSERT-FULL</b> $\triangleright Q * \triangleright^b \text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \exists \iota \notin \text{dom}(f). \square \text{BoxSlice}(\mathcal{N}, Q, \iota) * \triangleright^b \text{Box}(\mathcal{N}, P * Q, f[\iota \leftarrow \text{full}])$	
<b>SLICE-DELETE-FULL</b> $\frac{f(\iota) = \text{full}}{\text{BoxSlice}(\mathcal{N}, Q, \iota) \vdash \triangleright^b \text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \triangleright Q * \exists P'. \triangleright^b (\triangleright(P = P' * Q) * \text{Box}(\mathcal{N}, P', f[\iota \leftarrow \perp]))}$	
<b>SLICE-SPLIT</b> $\frac{f(\iota) = s}{\text{BoxSlice}(\mathcal{N}, Q_1 * Q_2, \iota) \vdash \triangleright^b \text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \exists \iota_1 \notin \text{dom}(f), \iota_2 \notin \text{dom}(f). \iota_1 \neq \iota_2 * \square \text{BoxSlice}(\mathcal{N}, Q_1, \iota_1) * \square \text{BoxSlice}(\mathcal{N}, Q_2, \iota_2) * \triangleright^b \text{Box}(\mathcal{N}, P, f[\iota \leftarrow \perp][\iota_1 \leftarrow s][\iota_2 \leftarrow s])}$	
<b>SLICE-MERGE</b> $\frac{\iota_1 \neq \iota_2 \quad f(\iota_1) = f(\iota_2) = s}{\text{BoxSlice}(\mathcal{N}, Q_1, \iota_1), \text{BoxSlice}(\mathcal{N}, Q_2, \iota_2) \vdash \triangleright^b \text{Box}(\mathcal{N}, P, f) \Rightarrow_{\mathcal{N}} \exists \iota \notin \text{dom}(f) \setminus \{\iota_1, \iota_2\}. \square \text{BoxSlice}(\mathcal{N}, Q_1 * Q_2, \iota) * \triangleright^b \text{Box}(\mathcal{N}, P, f[\iota_1 \leftarrow \perp][\iota_2 \leftarrow \perp][\iota \leftarrow s])}$	