ESOP 2018, Thessaloniki, Greece

Logical Reasoning for Disjoint Permissions

Xuan-Bach Le

Aquinas Hobor

NUS National University of Singapore

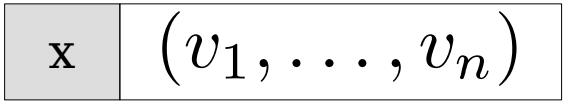
This talk is about ...

Permission reasoning in Concurrent Separation Logic

Some notations

- Mapsto predicate

$$x \mapsto (v_1, \cdots, v_n)$$



Some notations

- Mapsto predicate

$$x \mapsto (v_1, \cdots, v_n)$$

x	$(v_1, \ldots, v_n)$

- Fractional mapsto predicate

$$x \overset{\pi}{\mapsto} (v_1, \cdots, v_n)$$

x	$(v_1, \ldots, v_n, \pi)$

Some notations

- Mapsto predicate

$$x \mapsto (v_1, \cdots, v_n)$$

x	$(v_1, \ldots, v_n)$

- Fractional mapsto predicate

$$x \overset{\pi}{\mapsto} (v_1, \cdots, v_n)$$

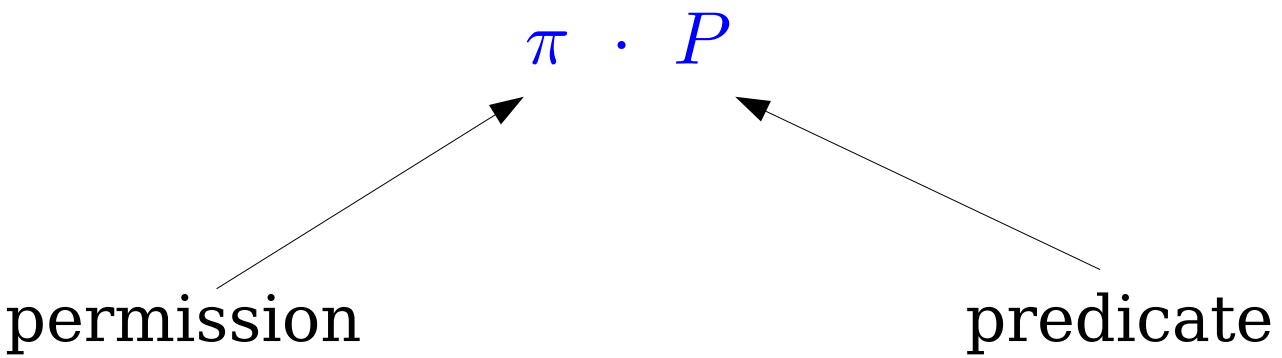
x	$(v_1, \ldots, v_n, \pi)$

- Disjoint conjunction

$$P \star Q$$

$$h \models P \star Q \overset{\text{def}}{=} \exists h_1, h_2.\ h = h_1 \uplus h_2 \wedge h_1 \models P \wedge h_2 \models Q$$

Predicate multiplication



Ownership reasoning

```
class Example {
   BinaryTree t;

   void shareTree{
     fork();
     readTree(t);
     wait();
    deallocate(t);
   }
}
```

Ownership reasoning

```
class Example {
    BinaryTree t;
    {tree(t)}
    void shareTree{
      fork();
      readTree(t);
      wait();
     deallocate(t);
    }
    {emp}
}
```

Parent process	Child process
`fork()`	
`readTree(t)`	`readTree(t)`
`wait()`	`wait()`
`deallocate(t)`	

Case study: Rational permissions

- Model:

$$\langle\ (0,1]\ \ ,\ \ +\ \rangle$$

- Examples:

$$x \overset{1/2}{\longmapsto} 1\ \ ,\ \ 0.4 \cdot tree(y)$$

- Combine permissions:

$$x \overset{0.3}{\longmapsto} 42\ \star\ x \overset{0.2}{\longmapsto} 42\ \ \dashv\vdash\ \ x \overset{0.5}{\longmapsto} 42$$

Reasoning with rational permissions (expected result)

Parent process

$\{tree(t)\}$

```
fork()

readTree(t)

wait()

deallocate(t)
```

$\{emp\}$

Child process

```
readTree(t)

wait()
```

Reasoning with rational permissions (expected result)

Parent process

$\{tree(t)\}$

$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$

`fork()`

`readTree(t)`

`wait()`

`deallocate(t)`

Child process

`readTree(t)`

`wait()`

Reasoning with rational permissions (expected result)

Parent process	Child process
$\{tree(t)\}$	
$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$	
`fork()`	
$\{\frac{1}{2} \cdot tree(t)\}$	$\{\frac{1}{2} \cdot tree(t)\}$
`readTree(t)`	`readTree(t)`
`wait()`	`wait()`
`deallocate(t)`	

Reasoning with rational permissions (expected result)

Parent process

$\{tree(t)\}$

$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$

`fork()`

$\{\frac{1}{2} \cdot tree(t)\}$

`readTree(t)`

$\{\frac{1}{2} \cdot tree(t)\}$

`wait()`

Child process

$\{\frac{1}{2} \cdot tree(t)\}$

`readTree(t)`

$\{\frac{1}{2} \cdot tree(t)\}$

`wait()`

`deallocate(t)`

Reasoning with rational permissions (expected result)

Parent process	Child process
$\{tree(t)\}$	
$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$	
`fork()`	
$\{\frac{1}{2} \cdot tree(t)\}$	$\{\frac{1}{2} \cdot tree(t)\}$
`readTree(t)`	`readTree(t)`
$\{\frac{1}{2} \cdot tree(t)\}$	$\{\frac{1}{2} \cdot tree(t)\}$
`wait()`	`wait()`
$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$	$\{emp\}$

`deallocate(t)`

Reasoning with rational permissions (expected result)

Parent process	Child process
$\{tree(t)\}$	
$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$	
`fork()`	
$\{\frac{1}{2} \cdot tree(t)\}$	$\{\frac{1}{2} \cdot tree(t)\}$
`readTree(t)`	`readTree(t)`
$\{\frac{1}{2} \cdot tree(t)\}$	$\{\frac{1}{2} \cdot tree(t)\}$
`wait()`	`wait()`
$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$	$\{emp\}$
$\{tree(t)\}$	
`deallocate(t)`	

Reasoning with rational permissions (expected result)

Parent process	Child process
$\{tree(t)\}$	
$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$	
`fork()`	
$\{\frac{1}{2} \cdot tree(t)\}$	$\{\frac{1}{2} \cdot tree(t)\}$
`readTree(t)`	`readTree(t)`
$\{\frac{1}{2} \cdot tree(t)\}$	$\{\frac{1}{2} \cdot tree(t)\}$
`wait()`	`wait()`
$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$	$\{emp\}$
$\{tree(t)\}$	
`deallocate(t)`	
$\{emp\}$	

Reasoning with rational permissions (expected result)

Parent process

$\{tree(t)\}$

$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$

`fork()`

$\{\frac{1}{2} \cdot tree(t)\}$

`readTree(t)`

$\{\frac{1}{2} \cdot tree(t)\}$

`wait()`

$\{\frac{1}{2} \cdot tree(t) \star \frac{1}{2} \cdot tree(t)\}$

$\{tree(t)\}$

`deallocate(t)`

$\{emp\}$

Child process

$\{\frac{1}{2} \cdot tree(t)\}$

`readTree(t)`

$\{\frac{1}{2} \cdot tree(t)\}$

`wait()`

$\{emp\}$

Shortcomming of Rationals

$$tree(t) \stackrel{\text{def}}{=} (t = 0 \wedge emp) \vee \Big(t \mapsto (t_1, t_2) \star tree(t_1) \star tree(t_2)\Big)$$

Shortcomming of Rationals

$$tree(t) \stackrel{\text{def}}{=} (t = 0 \wedge emp) \vee \left(t \mapsto (t_1, t_2) \star tree(t_1) \star tree(t_2)\right)$$

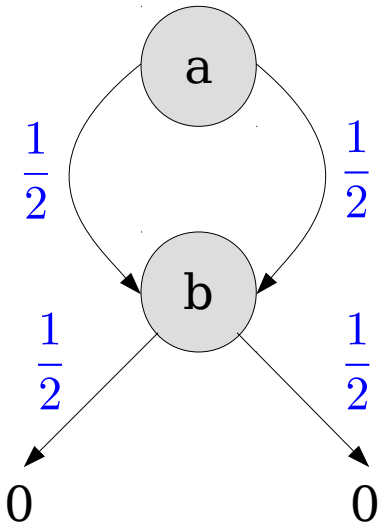
$$\frac{1}{2} \cdot tree(t) \stackrel{\text{def}}{=} (t = 0 \wedge emp) \vee \left(t \stackrel{1/2}{\longmapsto} (t_1, t_2) \star \frac{1}{2} \cdot tree(t_1) \star \frac{1}{2} \cdot tree(t_2)\right)$$

Shortcomming of Rationals

$$tree(t) \stackrel{\text{def}}{=} (t = 0 \wedge emp) \vee \left(t \mapsto (t_1, t_2) \star tree(t_1) \star tree(t_2)\right)$$

$$\frac{1}{2} \cdot tree(t) \stackrel{\text{def}}{=} (t = 0 \wedge emp) \vee \left(t \stackrel{1/2}{\longmapsto} (t_1, t_2) \star \frac{1}{2} \cdot tree(t_1) \star \frac{1}{2} \cdot tree(t_2)\right)$$

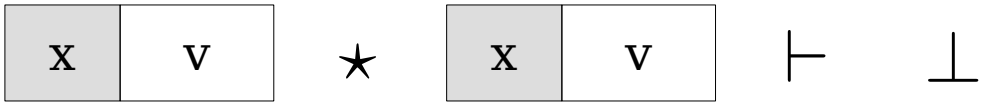
The latter is ~~always a tree~~ possibly a DAG!



Diagnosis

- Without permissions

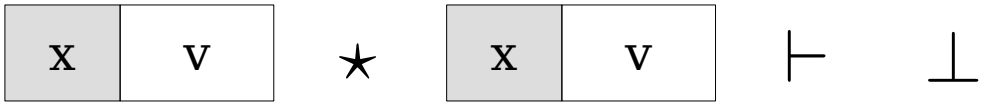
$x \mapsto v \ \star \ x \mapsto v$ is not satisfiable



Diagnosis

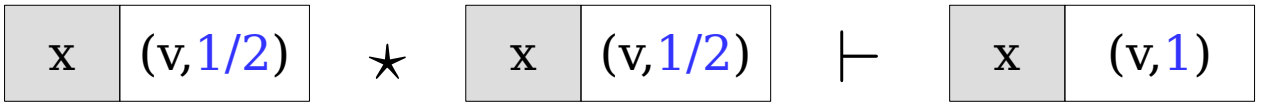
- Without permissions

$x \mapsto v \ \star \ x \mapsto v$ is not satisfiable



- With permissions

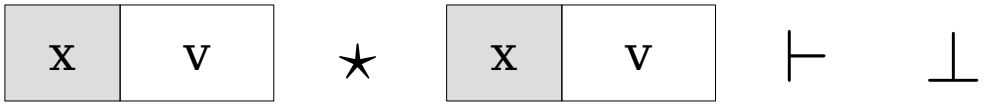
$x \overset{1/2}{\longmapsto} v \ \star \ x \overset{1/2}{\longmapsto} v$ is equivalent to $x \overset{1}{\longmapsto} v$



Diagnosis

- Without permissions

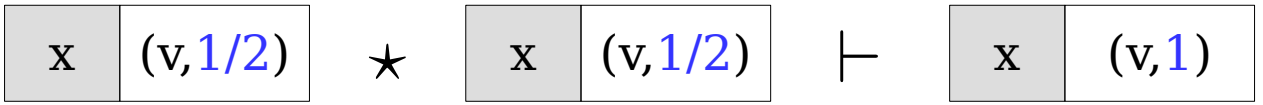
$x \mapsto v \ \star \ x \mapsto v$ is not satisfiable



- With permissions

Rational permissions fail to preserve the disjointness property of Separation Logic!

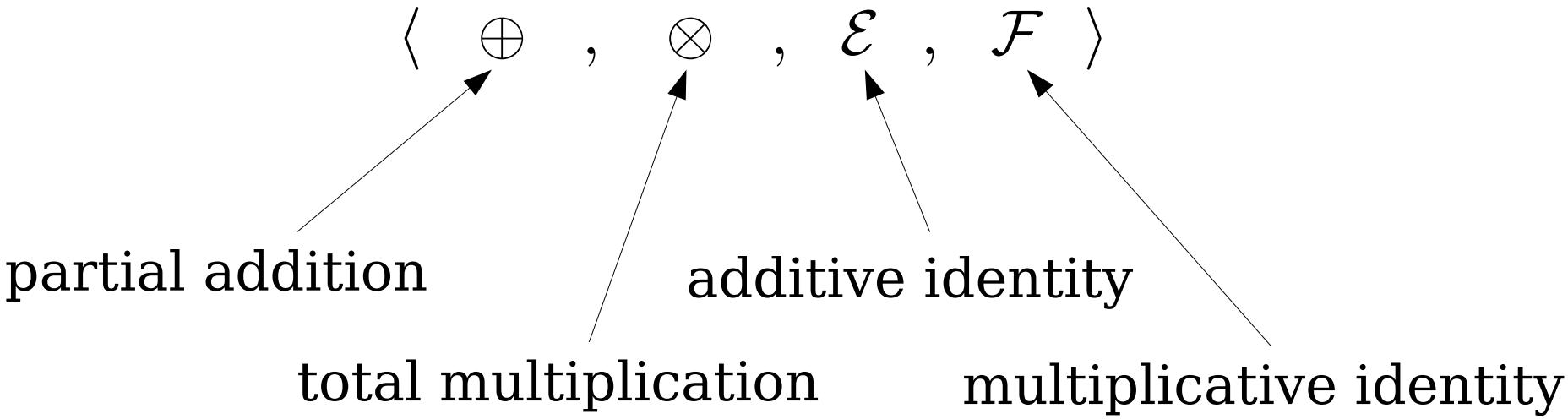
$x \overset{1/2}{\longmapsto} v \ \star \ x \overset{1/2}{\longmapsto} v$ is equivalent to $x \overset{1}{\mapsto} v$



This talk

1. Disjoint permission model
2. Inference systems
3. Disjoint permission analysis

Disjoint permission model



Disjoint permission model

- Axioms from semiring:
  - Commutativity over addition
  - Associativity over multiplication
  - Right distributivity of multiplication over addition
  - ...
- Disjointness axiom:

$$a \oplus a = b \Rightarrow a = \mathcal{E}$$

Disjoint permission model

- Axioms from semiring:
  - Commutativity over addition
  - Associativity over multiplication
  - Right distributivity of multiplication over addition
  - ...
- Disjointness axiom:

$$a \oplus a = b \Rightarrow a = \mathcal{E}$$

Not true with rationals!

$$0.2 + 0.2 = 0.4 \not\Rightarrow 0.2 = 0$$

Enable efficient bi-abduction reasoning

- Complete a partial entailment

$$A \star [??] \quad \vdash \quad B \star [??]$$

Enable efficient bi-abduction reasoning

- Complete a partial entailment

$$A \star [??] \quad \vdash \quad B \star [??]$$

- With disjoint permissions

$$x \overset{\pi}{\longmapsto} (x_1, x_2) \star \pi \cdot tree(x_1) \star [\pi \cdot tree(x_2)] \vdash \pi \cdot tree(x)$$

Enable efficient bi-abduction reasoning

- Complete a partial entailment

$$A \star [??] \quad \vdash \quad B \star [??]$$

- With disjoint permissions

$$x \overset{\pi}{\longmapsto} (x_1, x_2) \star \pi \cdot tree(x_1) \star [\pi \cdot tree(x_2)] \vdash \pi \cdot tree(x)$$

- Automatic tool ShareInfer

Roadmap

- Predicate multiplication with disjoint permissions
- Inference systems
- Disjoint permissions analysis

Overview of inference system

- 10/13 rules are bidirectional $A \dashv\vdash B$
- Some rules don't hold with rational permissions

$$\frac{P \vdash Q}{\pi \cdot P \vdash \pi \cdot Q}\ \textsc{Dot Pos} \qquad \frac{}{\pi \cdot \langle P \rangle \dashv\vdash \langle P \rangle}\ \textsc{Dot Pure} \qquad \frac{}{\pi \cdot (P \Rightarrow Q) \vdash (\pi \cdot P) \Rightarrow (\pi \cdot Q)}\ \textsc{Dot Impl}$$

$$\frac{}{\pi \cdot (P \wedge Q) \dashv\vdash (\pi \cdot P) \wedge (\pi \cdot Q)}\ \textsc{Dot Conj} \qquad \frac{}{\pi \cdot (P \vee Q) \dashv\vdash (\pi \cdot P) \vee (\pi \cdot Q)}\ \textsc{Dot Disj} \qquad \frac{}{\pi \cdot (\neg P) \vdash \neg \pi \cdot P}\ \textsc{Dot Neg}$$

$$\frac{\tau \neq \emptyset}{\pi \cdot \big(\forall x : \tau.\ P(x)\big) \dashv\vdash \forall x : \tau.\ \pi \cdot P(x)}\ \textsc{Dot Univ} \qquad \frac{}{\pi \cdot \big(\exists x : \tau.\ P(x)\big) \dashv\vdash \exists x : \tau.\ \pi \cdot P(x)}\ \textsc{Dot Exis}$$

$$\frac{}{\mathcal{F} \cdot P \dashv\vdash P}\ \textsc{Dot Full} \qquad \frac{}{\pi_1 \cdot (\pi_2 \cdot P) \dashv\vdash (\pi_1 \otimes \pi_2) \cdot P}\ \textsc{Dot Dot} \qquad \frac{}{\pi \cdot x \mapsto y \dashv\vdash x \overset{\pi}{\mapsto} y}\ \textsc{Dot MapsTo}$$

$$\frac{\mathrm{precise}(P)}{(\pi_1 \oplus \pi_2) \cdot P \dashv\vdash (\pi_1 \cdot P) \star (\pi_2 \cdot P)}\ \textsc{Dot Plus} \qquad \frac{P \vdash \mathsf{uniform}(\pi) \qquad Q \vdash \mathsf{uniform}(\pi)}{\pi \cdot (P \star Q) \dashv\vdash (\pi \cdot P) \star (\pi \cdot Q)}\ \textsc{Dot Star}$$

A closer look

$$\frac{}{\mathcal{F} \cdot P \dashv\vdash P}\ \begin{array}{l}\text{DOT}\\ \text{FULL}\end{array}$$

Initiate the sharing mechanism

A closer look

$$\frac{}{\pi_1 \cdot (\pi_2 \cdot P) \dashv\vdash (\pi_1 \otimes \pi_2) \cdot P}\ \begin{matrix}\text{DOT}\\ \text{DOT}\end{matrix}$$

Collapse nested permissions

A closer look

$$\frac{\text{precise}(P)}{(\pi_1 \oplus \pi_2) \cdot P \dashv\vdash (\pi_1 \cdot P) \star (\pi_2 \cdot P)} \begin{array}{l}\text{DOT}\\ \text{PLUS}\end{array}$$

Splitting permissions over predicate

precise(P): P cannot hold in 2 subheaps simultaneously

$$\text{precise}(P) \stackrel{\text{def}}{=} \forall h, h_1, h_2.\ h_1 \subseteq h_2 \Rightarrow h_2 \subseteq h \Rightarrow (h_1 \models P) \Rightarrow (h_2 \models P) \Rightarrow h_1 = h_2$$

A closer look

$$\frac{P \vdash \mathsf{uniform}(\pi') \qquad Q \vdash \mathsf{uniform}(\pi')}{\pi \cdot (P \star Q) \dashv\vdash (\pi \cdot P) \ \star \ (\pi \cdot Q)} \ \begin{matrix}\text{DOT}\\ \text{STAR}\end{matrix}$$

Distribute permissions over predicates

uniform(π'): all addresses have permission π'

$$x \overset{\pi'}{\longmapsto} 1 \star y \overset{\pi'}{\longmapsto} 2 \star z \overset{\pi'}{\longmapsto} 1 \quad , \quad \pi' \cdot tree(x)$$

Inductive Reasoning (honourable mention)

- Inference system for inductive reasoning
  - 8 inference rules
  - Induction over finiteness of fractional heap
  - Can prove side conditions precise, uniform
  - Implementation: ShareInfer tool

Example

Parent process

$\{tree(t)\}$

```
fork()
readTree(t)
wait()
```

Child process

```
readTree(t)
wait()
```

```
deallocate(t)
```

$\{emp\}$

Example

	Parent process	Child process
	$\{tree(t)\}$	
DotFull	$\{\mathcal{F} \cdot tree(t)\}$	
	`fork()`	
	`readTree(t)`	`readTree(t)`
	`wait()`	`wait()`
	`deallocate(t)`	

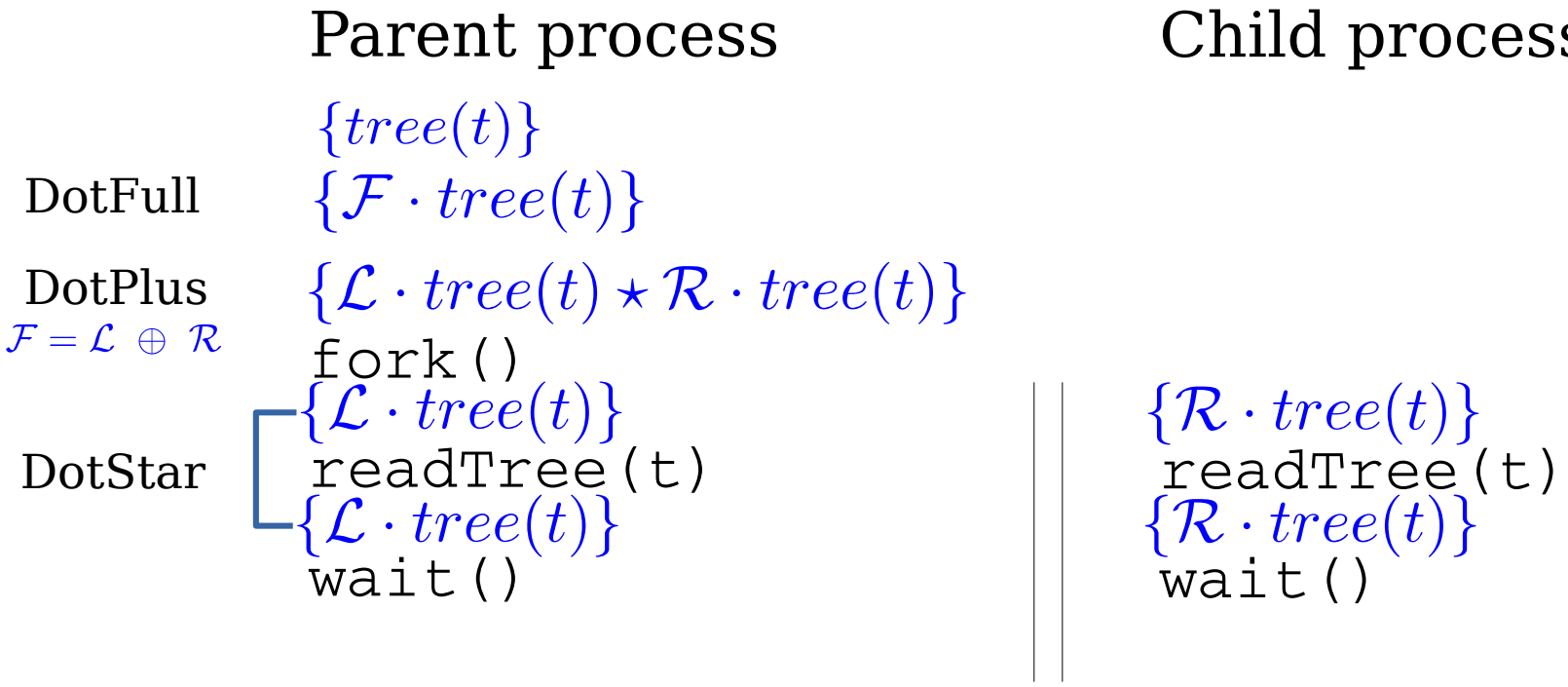
Example

	Parent process	Child process
	$\{tree(t)\}$	
DotFull	$\{\mathcal{F} \cdot tree(t)\}$	
DotPlus $\mathcal{F} = \mathcal{L} \oplus \mathcal{R}$	$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$	
	`fork()`	
	`readTree(t)`	`readTree(t)`
	`wait()`	`wait()`
	`deallocate(t)`	

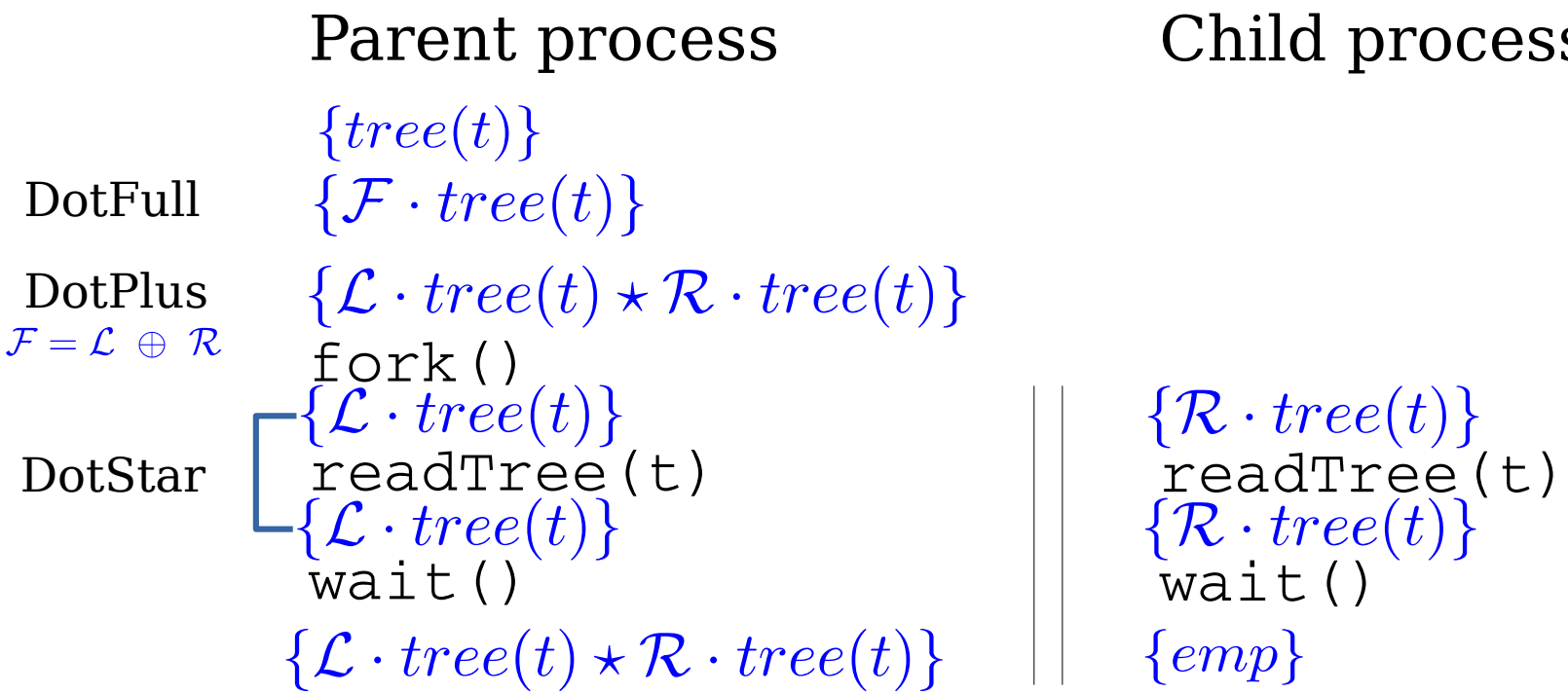
Example

	Parent process	Child process
	$\{tree(t)\}$	
DotFull	$\{\mathcal{F} \cdot tree(t)\}$	
DotPlus $\mathcal{F} = \mathcal{L} \oplus \mathcal{R}$	$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$	
	`fork()`	
	$\{\mathcal{L} \cdot tree(t)\}$	$\{\mathcal{R} \cdot tree(t)\}$
	`readTree(t)`	`readTree(t)`
	`wait()`	`wait()`
	`deallocate(t)`	

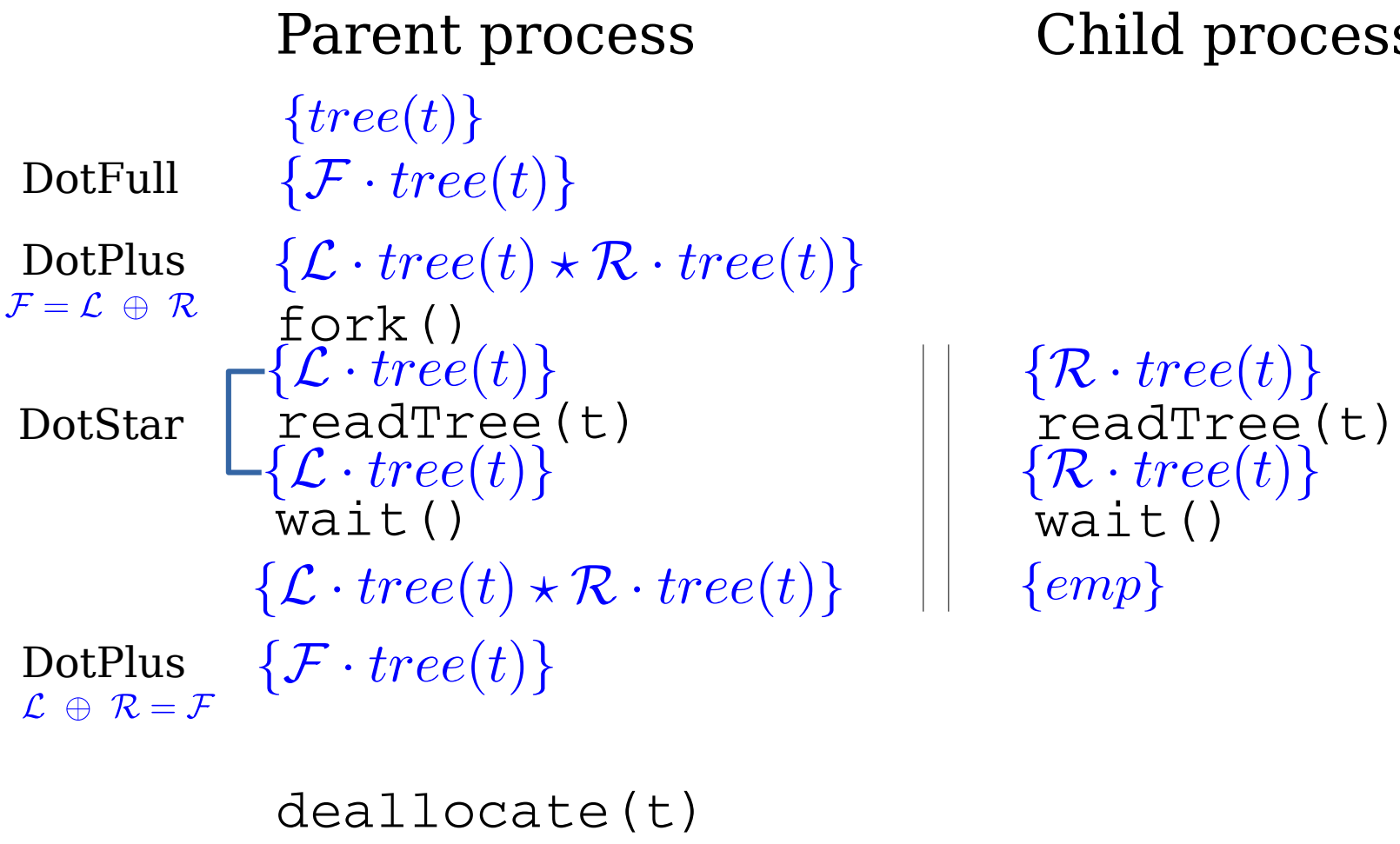
Example


Parent process
Child process
$\{tree(t)\}$
DotFull
$\{\mathcal{F} \cdot tree(t)\}$
DotPlus
$\mathcal{F} = \mathcal{L} \oplus \mathcal{R}$
$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$
fork()
$\{\mathcal{L} \cdot tree(t)\}$
$\{\mathcal{R} \cdot tree(t)\}$
DotStar
readTree(t)
readTree(t)
$\{\mathcal{L} \cdot tree(t)\}$
$\{\mathcal{R} \cdot tree(t)\}$
wait()
wait()
deallocate(t)

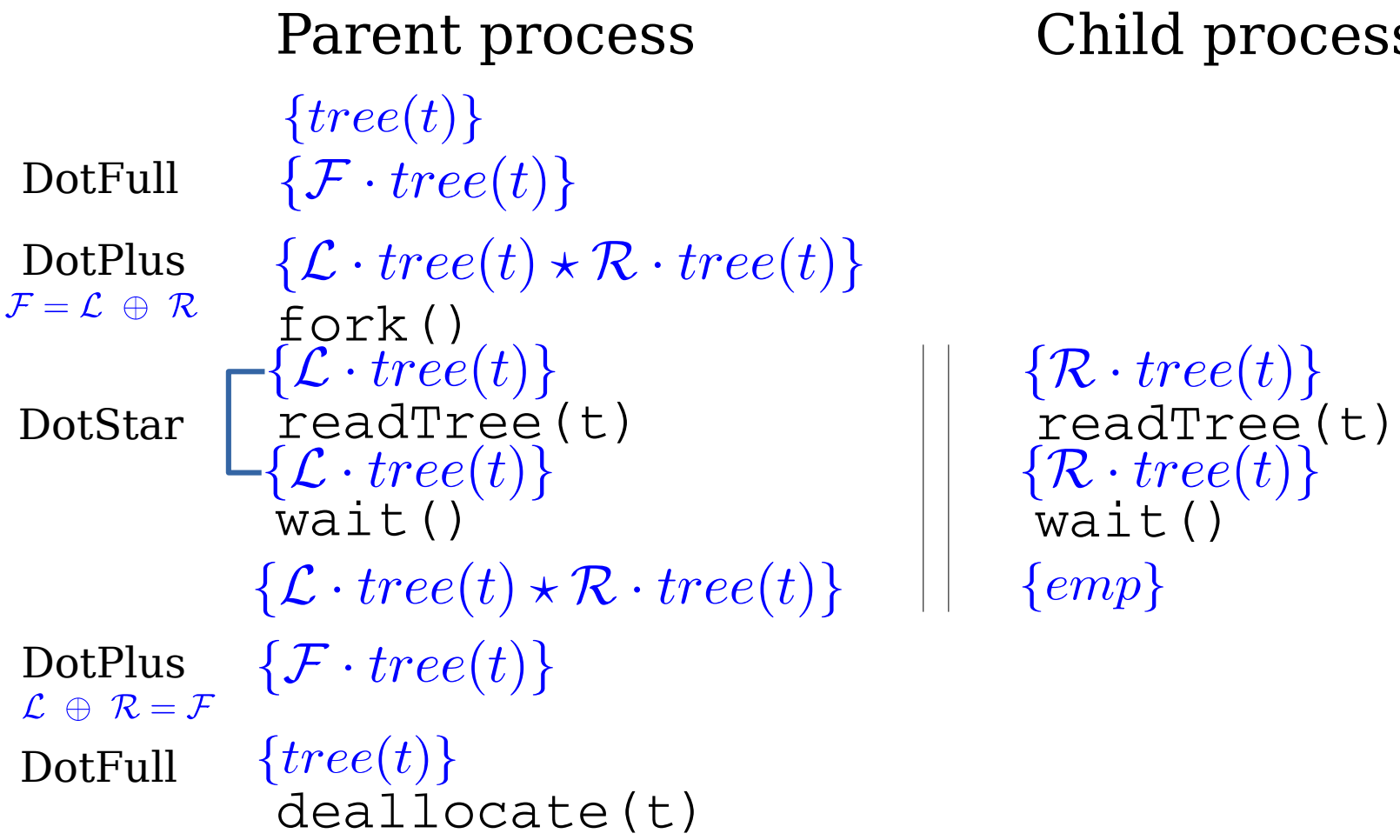
Example


Parent process
Child process
$\{tree(t)\}$
DotFull
$\{\mathcal{F} \cdot tree(t)\}$
DotPlus
$\mathcal{F} = \mathcal{L} \oplus \mathcal{R}$
$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$
fork()
$\{\mathcal{L} \cdot tree(t)\}$
$\{\mathcal{R} \cdot tree(t)\}$
DotStar
readTree(t)
readTree(t)
$\{\mathcal{L} \cdot tree(t)\}$
$\{\mathcal{R} \cdot tree(t)\}$
wait()
wait()
$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$
$\{emp\}$
deallocate(t)

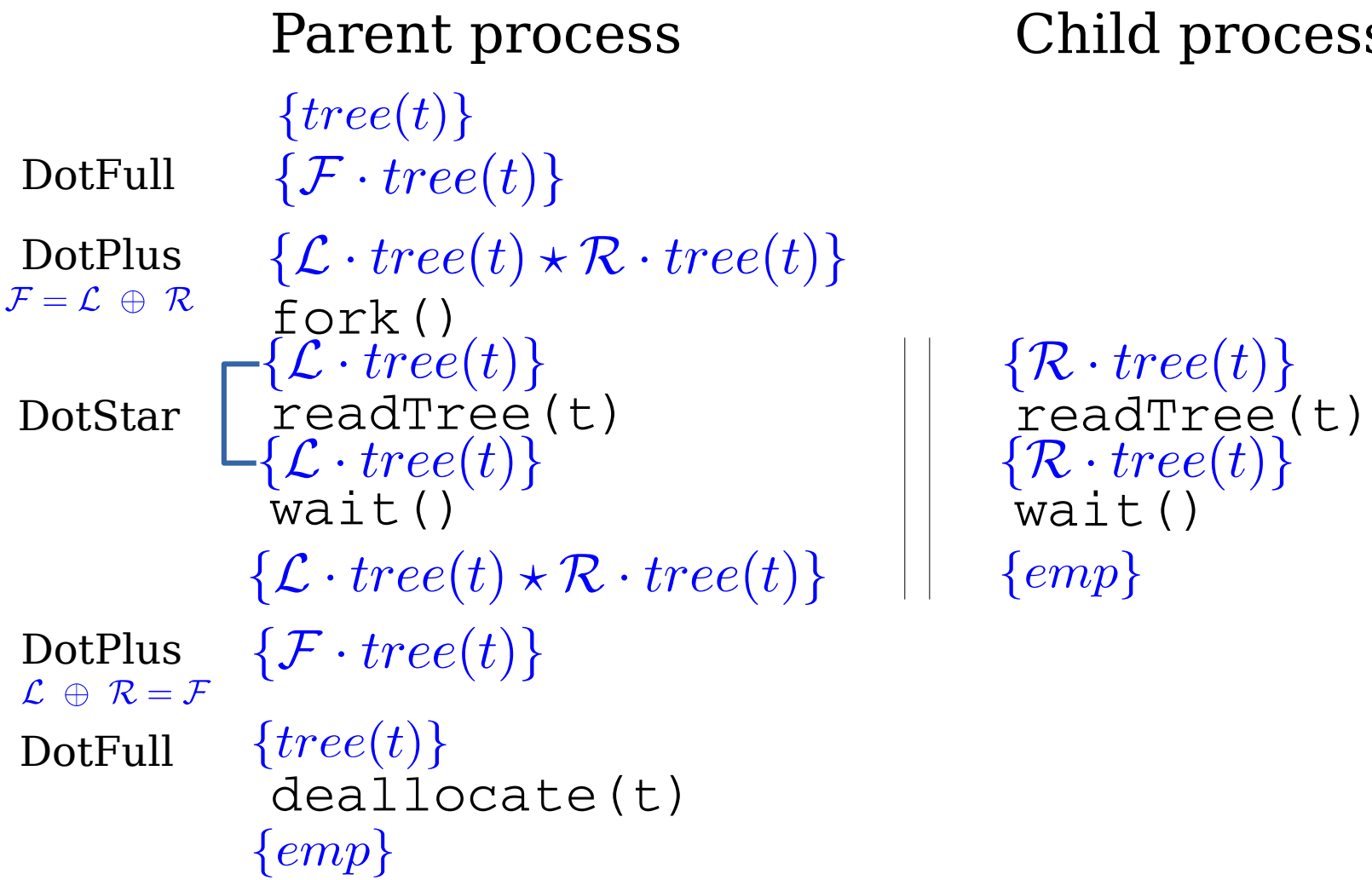
Example


Parent process
Child process
$\{tree(t)\}$
DotFull
$\{\mathcal{F} \cdot tree(t)\}$
DotPlus
$\mathcal{F} = \mathcal{L} \oplus \mathcal{R}$
$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$
fork()
$\{\mathcal{L} \cdot tree(t)\}$
readTree(t)
$\{\mathcal{R} \cdot tree(t)\}$
readTree(t)
DotStar
$\{\mathcal{L} \cdot tree(t)\}$
wait()
$\{\mathcal{R} \cdot tree(t)\}$
wait()
$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$
$\{emp\}$
DotPlus
$\mathcal{L} \oplus \mathcal{R} = \mathcal{F}$
$\{\mathcal{F} \cdot tree(t)\}$
deallocate(t)

Example


Parent process
Child process
$\{tree(t)\}$
DotFull
$\{\mathcal{F} \cdot tree(t)\}$
DotPlus
$\mathcal{F} = \mathcal{L} \oplus \mathcal{R}$
$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$
fork()
$\{\mathcal{L} \cdot tree(t)\}$
readTree(t)
$\{\mathcal{R} \cdot tree(t)\}$
readTree(t)
DotStar
$\{\mathcal{L} \cdot tree(t)\}$
wait()
$\{\mathcal{R} \cdot tree(t)\}$
wait()
$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$
$\{emp\}$
DotPlus
$\mathcal{L} \oplus \mathcal{R} = \mathcal{F}$
$\{\mathcal{F} \cdot tree(t)\}$
DotFull
$\{tree(t)\}$
deallocate(t)

Example


Parent process
Child process
$\{tree(t)\}$
DotFull
$\{\mathcal{F} \cdot tree(t)\}$
DotPlus
$\mathcal{F} = \mathcal{L} \oplus \mathcal{R}$
$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$
fork()
DotStar
$\{\mathcal{L} \cdot tree(t)\}$
readTree(t)
$\{\mathcal{L} \cdot tree(t)\}$
wait()
$\{\mathcal{L} \cdot tree(t) \star \mathcal{R} \cdot tree(t)\}$
$\{\mathcal{R} \cdot tree(t)\}$
readTree(t)
$\{\mathcal{R} \cdot tree(t)\}$
wait()
$\{emp\}$
DotPlus
$\mathcal{L} \oplus \mathcal{R} = \mathcal{F}$
$\{\mathcal{F} \cdot tree(t)\}$
DotFull
$\{tree(t)\}$
deallocate(t)
$\{emp\}$

Soundness

- Prove over fractional heap model:

$\mathcal{H}$: Address $\rightharpoonup$ Value $\times$ Permission

a_1	(v_1, π_1)
...	...
a_n	(v_n, π_n)

Soundness

- Prove over fractional heap model:

$$\mathcal{H} : \text{Address} \rightharpoonup \text{Value} \times \text{Permission}$$

a_1	(v_1, π_1)
...	...
a_n	(v_n, π_n)

- Heap multiplication:

$$\pi \cdot \begin{array}{|c|c|}\hline a_1 & (v_1, \pi_1) \\ \hline \ldots & \ldots \\ \hline a_n & (v_n, \pi_n) \\ \hline \end{array} \stackrel{\text{def}}{=} \begin{array}{|c|c|}\hline a_1 & (v_1, \pi \otimes \pi_1) \\ \hline \ldots & \ldots \\ \hline a_n & (v_n, \pi \otimes \pi_n) \\ \hline \end{array}$$

Soundness

- Prove over fractional heap model:

$$\mathcal{H} : \text{Address} \rightharpoonup \text{Value} \times \text{Permission}$$

a_1	(v_1, π_1)
...	...
a_n	(v_n, π_n)

- Heap multiplication:

$$\pi \cdot \begin{array}{|c|c|}\hline a_1 & (v_1, \pi_1) \\ \hline \ldots & \ldots \\ \hline a_n & (v_n, \pi_n) \\ \hline \end{array} \stackrel{\text{def}}{=} \begin{array}{|c|c|}\hline a_1 & (v_1, \pi \otimes \pi_1) \\ \hline \ldots & \ldots \\ \hline a_n & (v_n, \pi \otimes \pi_n) \\ \hline \end{array}$$

- Predicate multiplication:

$$h \models \pi \cdot P \stackrel{\text{def}}{=} \exists h'.\ h' \models P \wedge h = \pi \cdot h'$$

Soundness

disjoint permission axioms $\Rightarrow$ inference rules

Soundness

disjoint permission axioms $\Rightarrow$ inference rules

Example:

$\forall a, b, c.\ (a \otimes b) \otimes c = a \otimes (b \otimes c)$ $\Rightarrow$ $\dfrac{}{\pi_1 \cdot (\pi_2 \cdot P) \dashv\vdash (\pi_1 \otimes \pi_2) \cdot P}$ $\begin{matrix}\textsc{Dot}\\ \textsc{Dot}\end{matrix}$

Associativity of $\otimes$

Roadmap

- Predicate multiplication with disjoint permissions
- Inference systems
- Disjoint permissions analysis

Inference rules force permission axioms

inference rules $\Rightarrow$ disjoint permission axioms

Inference rules force permission axioms

inference rules $\Rightarrow$ disjoint permission axioms

Example:

$$\frac{}{\mathcal{F} \cdot P \dashv\vdash P}\ \begin{array}{l}\text{DOT}\\ \text{FULL}\end{array} \qquad \Rightarrow \quad \forall a.\ \mathcal{F} \otimes a = a$$

Inference rules force permission axioms

inference rules $\Rightarrow$ disjoint permission axioms

Example:

$$\frac{}{\mathcal{F} \cdot P \dashv\vdash P}\ \begin{matrix}\text{DOT}\\ \text{FULL}\end{matrix} \quad \Rightarrow \quad \forall a.\ \mathcal{F} \otimes a = a$$

Proof sketch: Let P be $x \overset{a}{\mapsto} 1$ and $h \models \mathcal{F} \cdot P$.

By definition, $h =$ | x | $(1, \mathcal{F} \otimes a)$ |

As $h \models P$, we also have $h =$ | x | $(1, a)$ |

Thus $\mathcal{F} \otimes a = a$. QED

Disjointness in a multiplicative setting

- Disjointness axiom (D)

$$a \oplus a = b \Rightarrow a = \mathcal{E}$$

- Left distributivity (LD)

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

- Right distributivity (RD)

$$(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$$

Disjointness in a multiplicative setting

- Disjointness axiom (D)

$$a \oplus a = b \Rightarrow a = \mathcal{E}$$

- Left distributivity (LD)

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

- Right distributivity (RD)

$$(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$$

D + LD + RD + other standard axioms $\Rightarrow$ Trivial models

Disjointness in a multiplicative setting

- Disjointness axiom (D)

$$a \oplus a = b \Rightarrow a = \mathcal{E}$$

- Left distributivity (LD)

$$a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$$

- Right distributivity (RD)

$$(b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a)$$

D + ~~LD~~ + RD + other standard axioms $\ni$ Good models

Disjointness in a multiplicative setting

Multiplicative left inverse (LI):

$$\forall a \exists a'.\ a' \otimes a = \mathcal{F}$$

Disjointness in a multiplicative setting

Multiplicative left inverse (LI):

$$\forall a \exists a'.\ a' \otimes a = \mathcal{F}$$

D + (LD or RD) + LI + other axioms $\Rightarrow$ Trivial models

Disjointness in a multiplicative setting

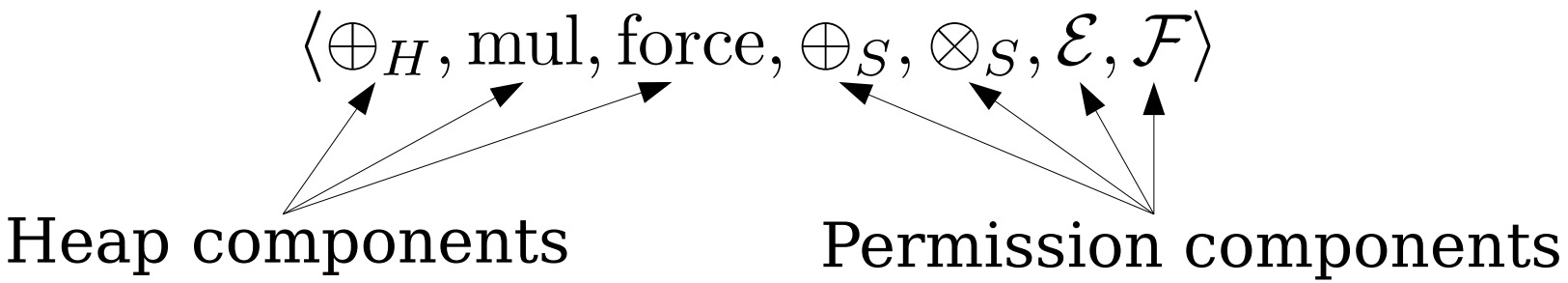
Multiplicative left inverse (LI):

$$\forall a \exists a'.\ a' \otimes a = \mathcal{F}$$

D + (LD or RD) + ~~LI~~ + other axioms $\ni$ Good models

Scaling separation algebra

Capture characteristics of fractional heaps



Scaling separation algebra

Capture characteristics of fractional heaps

$$\langle \oplus_H, \text{mul}, \text{force}, \oplus_S, \otimes_S, \mathcal{E}, \mathcal{F} \rangle$$

Heap join:

a_1	(v_1, π_1)
a_2	(v_2, π_2)

$\oplus_H$

a_1	(v_1, π_1')
a_3	(v_3, π_3)

$=$

a_1	$(v_1, \pi_1 \oplus \pi_1')$
a_2	(v_2, π_2)
a_3	(v_3, π_3)

Scaling separation algebra

Capture characteristics of fractional heaps

$$\langle \oplus_H, \text{mul}, \text{force}, \oplus_S, \otimes_S, \mathcal{E}, \mathcal{F} \rangle$$

Heap mul:

$\text{mul}(\ \pi\ ,$

a_1	(v_1, π_1)
...	...
a_n	(v_n, π_n)

$)\ \overset{\text{def}}{=}$

a_1	$(v_1, \pi \otimes \pi_1)$
...	...
a_n	$(v_n, \pi \otimes \pi_n)$

Scaling separation algebra

Capture characteristics of fractional heaps

$$\langle \oplus_H, \text{mul}, \text{force}, \oplus_S, \otimes_S, \mathcal{E}, \mathcal{F} \rangle$$

Heap force:

force(π , | a_1 | (v_1, π_1) |
|---|---|
| ... | ... |
| a_n | (v_n, π_n) |) $\overset{\text{def}}{=}$ | a_1 | (v_1, π) |
|---|---|
| ... | ... |
| a_n | (v_n, π) |

Scaling separation algebra

14 axioms for scaling separation algebra

$S_1.\ force(\pi, force(\pi', a)) = force(\pi, a)$ | $S_2.\ force(\pi, mul(\pi', a)) = force(\pi, a)$

$S_3.\ mul(\pi, force(\pi', a)) = force(\pi \otimes_S \pi', a)$ | $S_4.\ mul(\pi, mul(\pi', a)) = mul(\pi \otimes_S \pi', a)$

$S_5.\ identity(a) \Rightarrow force(\pi, a) = a$ | $S_6.\ a \subseteq_H force(\mathcal{F}, a)$

$S_7.\ \pi_1 \subseteq_S \pi_2 \Rightarrow force(\pi_1, a) \subseteq_H force(\pi_2, a)$ | $S_8.\ force(\pi, a) \oplus_H force(\pi, b) = c \Rightarrow force(\pi, c) = c$

$S_9.\ identity(a) \Rightarrow mul(\pi, a) = a$ | $S_{10}.\ mul(\mathcal{F}, a) = a$

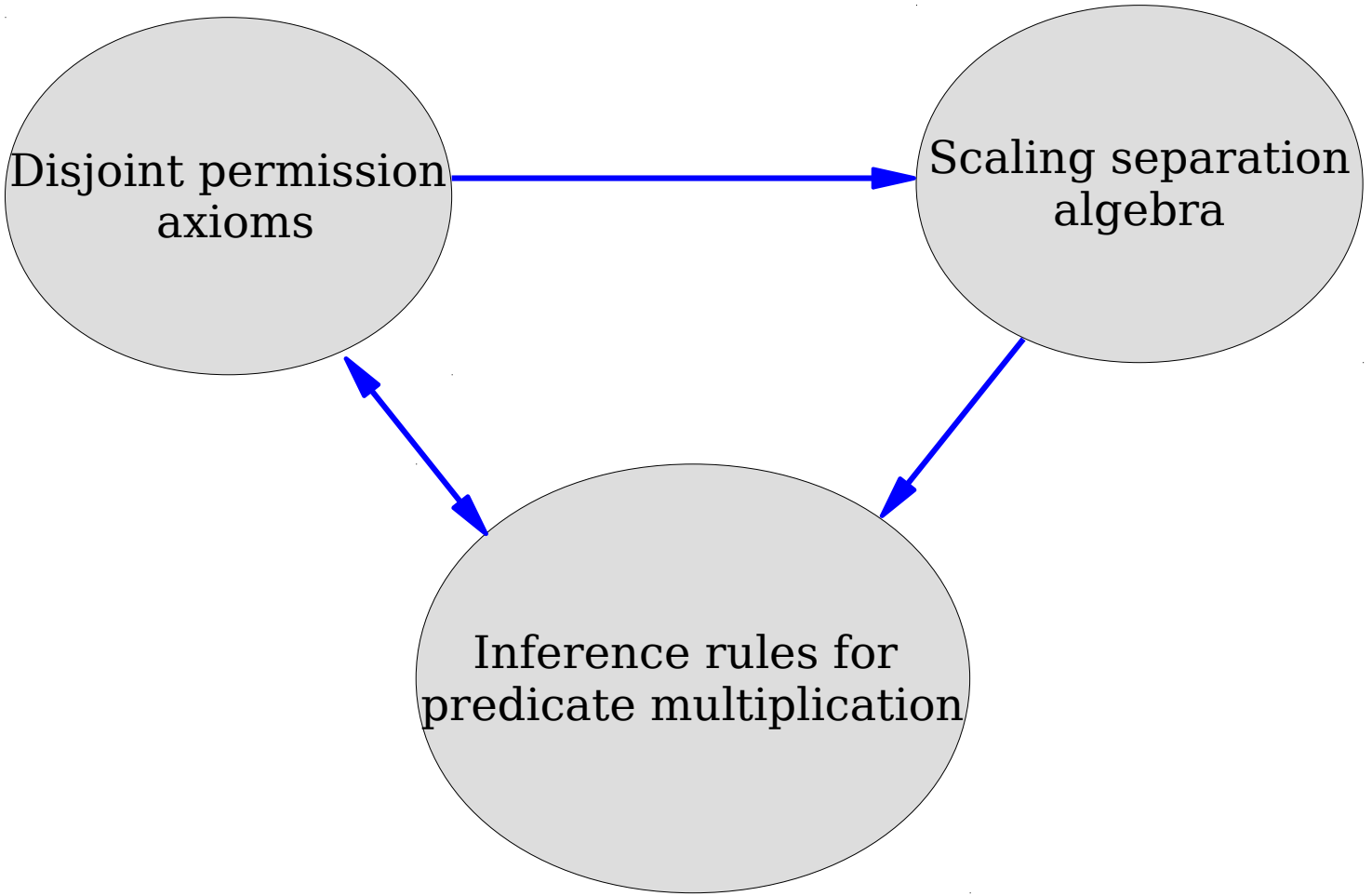
$S_{11}.\ mul(\pi, a_1) = mul(\pi, a_2) \Rightarrow a_1 = a_2$ | $S_{12}.\ mul(\pi, a) \subseteq_H a$

$S_{13}.\ \pi_1 \oplus_S \pi_2 = \pi_3 \Rightarrow \forall b, c. \big(\big(mul(\pi_1, b) \oplus_H mul(\pi_2, b) = c \big) \Leftrightarrow \big(c = mul(\pi_3, b) \big) \big)$

$S_{14}.\ force(\pi', a) \oplus_H force(\pi', b) = force(\pi', c) \quad \Leftrightarrow$
$mul\big(\pi, force(\pi', a)\big) \oplus_H mul\big(\pi, force(\pi', b)\big) = mul\big(\pi, force(\pi', c)\big)$

Axioms for fractional heaps

A graphical summary



Conclusion

- We proposed inference systems (with tool support) for predicate multiplication with disjoint permissions.
- Our soundness proof is verified in Coq using fractional heap model and Scaling Separation Algebra.
- We justified why certain properties of disjoint permissions cannot hold simultaneously.
- Future work: further investigation for permission algebra and Scaling Separation Algebra.

Thank you!