Forwards and Backwards in Separation Algebra

Peter Höfner
(joint work with C. Bannister and G. Klein)

June 2017

Separation Logic (Reynolds, O'Hearn et al.)

- Extension to Hoare Logic
- Based on Separation *Algebras* of abstract heaps
- Captures the notion of *disjointness* in the world

Separation Logic (Reynolds, O'Hearn et al.)

Motivation

- Pointer programs are hard to reason about

$$\{p \mapsto a\}$$

$$\text{delete } p$$

$$\{p \not\mapsto \_\}$$

The Frame Problem

Separation Logic (Reynolds, O'Hearn et al.)

Motivation

- Pointer programs are hard to reason about

$$\{p \mapsto a \wedge p' \mapsto b\}$$
$$\text{delete } p$$
$$\{p \not\mapsto \_ \wedge p' \mapsto b\}$$

The Frame Problem

Separation Logic (Reynolds, O'Hearn et al.)

Motivation

- Pointer programs are hard to reason about

$$\{p \mapsto a \wedge p' \mapsto b \wedge p \neq p'\}$$
$$\text{delete } p$$
$$\{p \not\mapsto \_ \wedge p' \mapsto b \wedge p \neq p'\}$$

The Frame Problem

Separation Logic (Reynolds, O'Hearn et al.)

Motivation

- $s, h \models P$
  where *s* is a store, *h* is a heap, and *P* is an *assertion* over the given store and heap

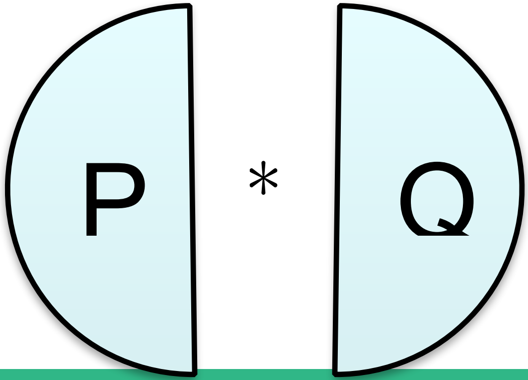
$$\begin{array}{ll} & s, h \models P * Q \\ \Leftrightarrow & \exists h_1, h_2.\ h_1 \perp h_2 \text{ and} \\ & h = h_1 \cup h_2 \text{ and } s, h_1 \models P \text{ and } s, h_2 \models Q \end{array}$$

Separation Logic (Reynolds, O'Hearn et al.)

Motivation

- $s, h \models P$
  where *s* is a store, *h* is a heap, and *P* is an *assertion* over the given store and heap

$$
\begin{array}{rl}
 & s, h \models P * Q \\
\Leftrightarrow & \exists h_1, h_2.\ h_1 \perp h_2 \text{ and} \\
 & h = h_1 \cup h_2 \text{ and } s, h_1 \models P \text{ and } s, h_2 \models Q
\end{array}
$$



Frame Rule

$$\frac{\{P\}\ C\ \{Q\}}{\{P * R\}\ C\ \{Q * R\}}\quad (mod(C) \cap fv(R) = \emptyset)$$

- R is the ‘Frame’
  - Extending an environment with a disjoint portion changes nothing
  - Local Reasoning
  - Compositional

Separation Algebras

- Separation logic can be lifted to algebra

- Allows abstract reasoning
- Transfers knowledge
- Ideal for interactive and automated theorem proving

Separation Algebras (Calcagno et al.)

- partial commutative monoid
  partial plus (+), and neutral element (0)

- h # h' captures the 'definedness' or partiality of (+)

- 0 is the empty heap

$$x + 0 = x \qquad x \# 0$$

$$s, h \models P * Q \;\Leftrightarrow\; \exists h_1, h_2.\; h_1 \# h_2 \wedge h = h_1 + h_2 \wedge P(h_1) \wedge Q(h_2)$$

Algebra of Assertions (Dang et al.)

- Set-based semantics

$$[\![p]\!] \Leftrightarrow \{(s,h) : s,h \models p\} \ .$$

$$\begin{array}{rcl} [\![p \ * \ q]\!] & = & [\![p]\!] \,\dot{\cup}\, [\![q]\!] \\ P \,\dot{\cup}\, Q & \Leftrightarrow & \{(s, h \cup h') : (s,h) \in P \wedge (s,h') \in Q \\ & & \wedge\ doms(h) \cap dom(h') = \emptyset\} \ . \end{array}$$

Separating Implication
Magic Wand

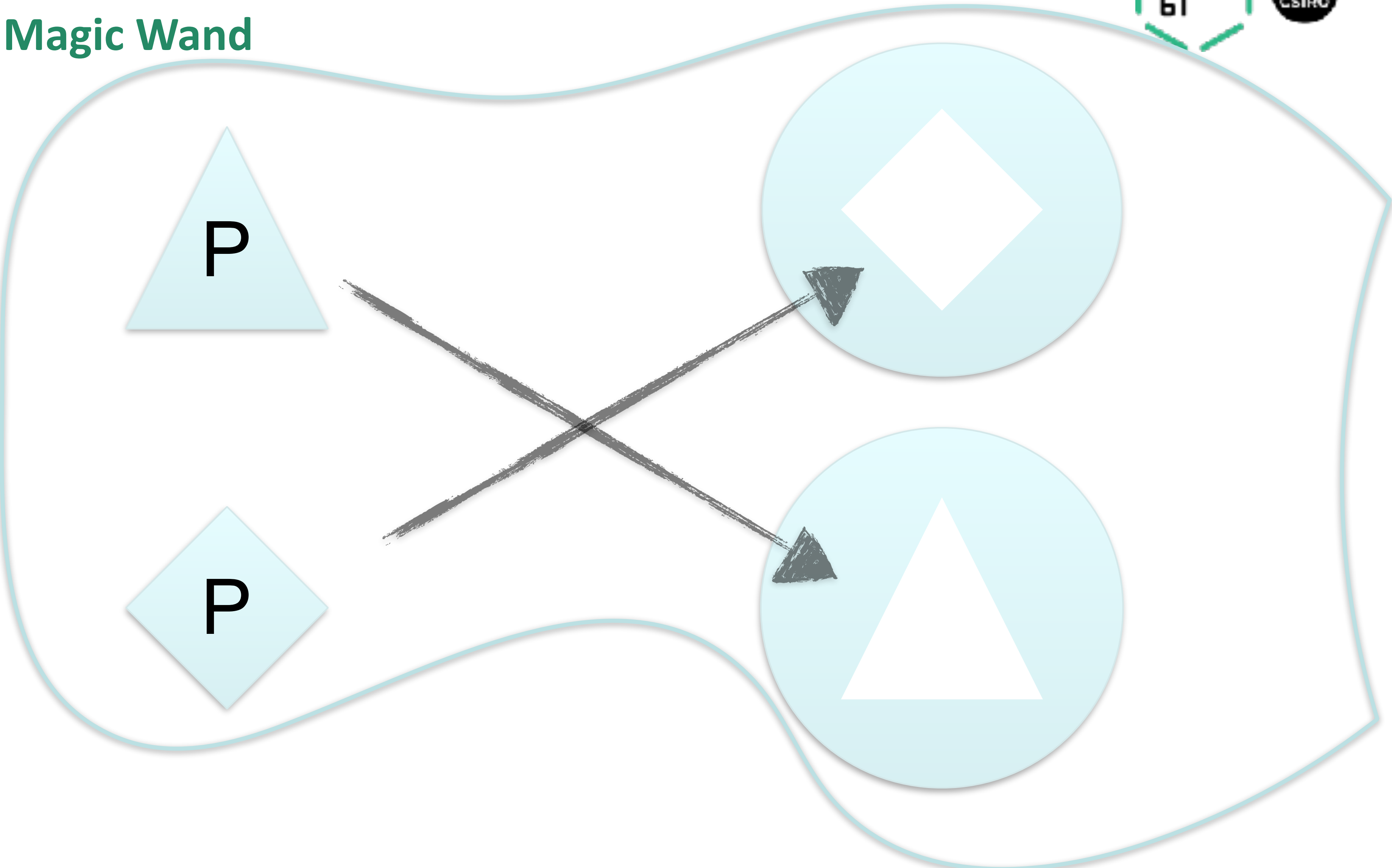
- Separating Implication $P \mathrel{-\!\!*} Q$
  - Extending by P produces Q over the combination

- Describes a *mapping* between heaps and 'holes'

$$s, h \models P \mathrel{-\!\!*} Q \quad \Leftrightarrow \quad \forall h'.\ (h' \perp h \text{ and } s, h' \models P) \\ \text{implies } s, h' \cup h \models Q$$

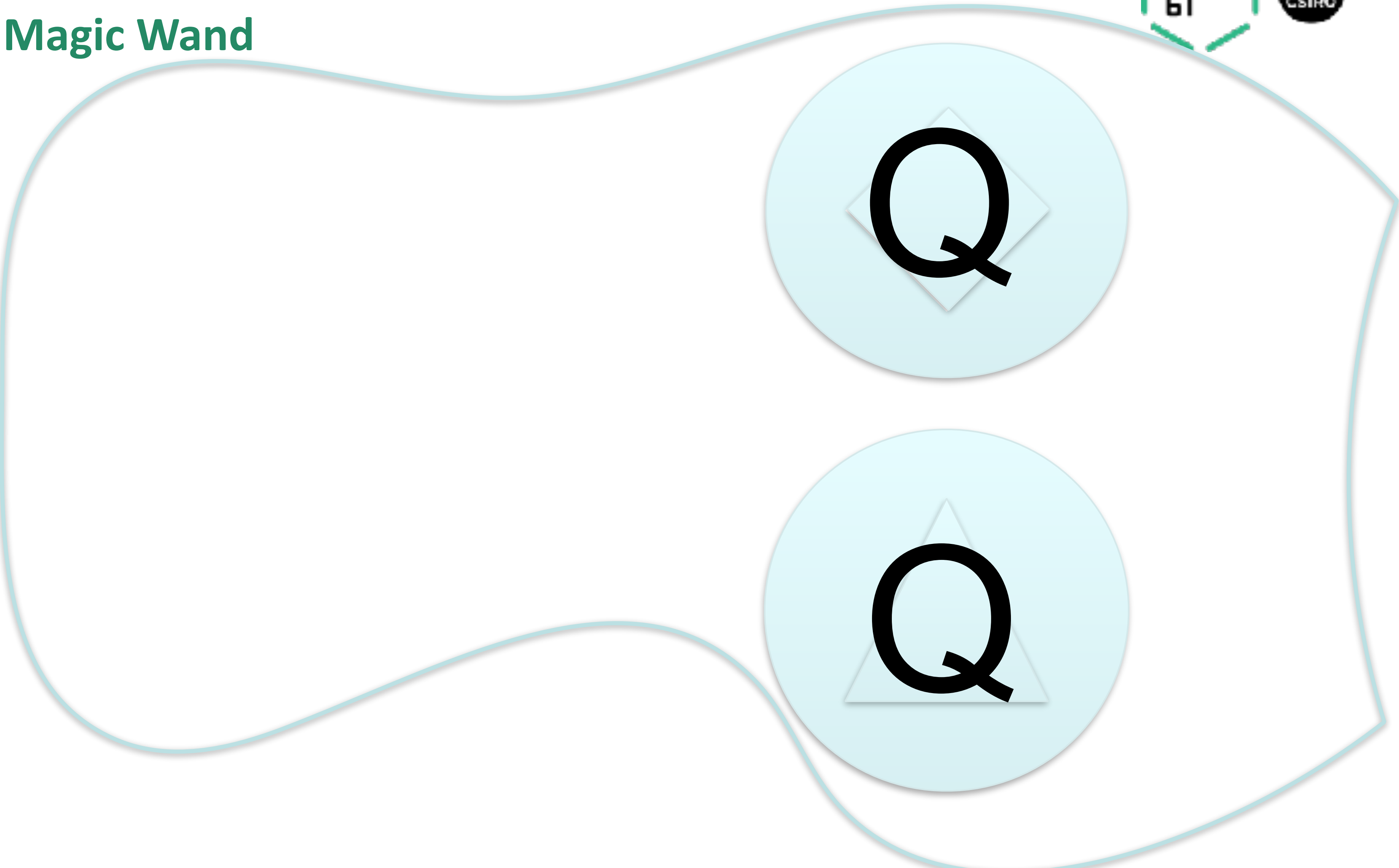
Separating Implication

Magic Wand



Separating Implication

Magic Wand



Conjunction version Implication

- Podus ponens

$$\frac{s, h \models Q * (Q \mathrel{-\!\!*} P)}{s, h \models P}$$

Conjunction version Implication

- Podus ponens

$$[\![Q * (Q -\!\!* P)]\!] \;\subseteq\; [\![P]\!]$$

Conjunction version Implication

- Podus ponens

$$Q * (Q \mathrel{-\!\!*} P) \;\Rightarrow\; P$$

Conjunction version Implication

- Podus ponens

$$Q * (Q \mathrel{-\!\!*} P) \;\Rightarrow\; P$$

- Currying/decurrying

$$(P * Q \;\Rightarrow\; R) \;\Leftrightarrow\; (P \;\Rightarrow\; Q \mathrel{-\!\!*} R)$$

Conjunction version Implication

- Podus ponens

$$Q * (Q \mathrel{-\!\!*} P) \;\Rightarrow\; P$$

- Currying/decurrying

$$(P * Q \;\Rightarrow\; R) \;\Leftrightarrow\; (P \;\Rightarrow\; Q \mathrel{-\!\!*} R)$$

Galois connection

Relationships between Operators

$$\begin{tikzcd} \boxed{P * Q} \arrow[rr, leftrightarrow, bend left, "\text{Galois}"] & & \boxed{P \mathrel{-\!\!*} Q} \end{tikzcd}$$

Backwards Reasoning

- Backward reasoning / reasoning in weakest precondition style
- for given postcondition Q and given program C, determine weakest precondition $wp(C, Q)$ such that
$$\{wp(C,Q)\}\ C\ \{Q\}$$
is valid Hoare triple
- but what about separation logic where frames occur?

$$\{P * R\}\ C\ \{Q * R\}$$

(problem with frame calculation)

Backwards Reasoning II

- from Galois connection we get

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \quad \Leftrightarrow \quad (\forall R.\ \{P * (Q \mathrel{-\!\!*} R)\}\ C\ \{R\})$$

- used to transform specifications
  for example

$$\{p \mapsto \_ * R\}\ \mathrm{set\_ptr}\ p\ v\ \{p \mapsto v * R\}$$

Backwards Reasoning II

- from Galois connection we get

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \quad \Leftrightarrow \quad (\forall R.\ \{P * (Q \mathrel{-\!\!*} R)\}\ C\ \{R\})$$

- used to transform specifications
  for example

$$\{p \mapsto \_ * (p \mapsto v \mathrel{-\!\!*} R)\}\ \mathrm{set\_ptr}\ p\ v\ \{R\}$$

Backwards Reasoning III

- Allows now full backwards reasoning without calculating the frame in every step
- Supported in Isabelle/HOL
- Easy patterns (alternation between implication and conjunction) allow automated simplifications

Forward Reasoning

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \quad \Leftrightarrow \quad (\forall R.\ \{R\}\ C\ \{??\})$$

Forward Reasoning II

- Ideal world

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \quad \Leftrightarrow \quad (\forall R.\ \{R\}\ C\ \{Q * (P \multimap\!\circledast\ R)\})$$

More Separation Logic

- there is another operator in the literature: septraction

$$s, h \models P \mathbin{-\!\!\circledast} Q$$

$$\Leftrightarrow \; \exists h_2.\, h \text{ subheap of } h_2 \text{ and } s, h_2 - h \models P \text{ and } s, h_2 \models Q$$

- algebraically:

$$P \mathbin{-\!\!\circledast} Q \;\Leftrightarrow\; \neg(P \mathbin{-\!\!*} (\neg Q))$$

Forward Reasoning II

- Ideal world seemingly impossible

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \quad \Leftrightarrow \quad (\forall R.\ \{R\}\ C\ \{Q * (P \multimap\!\circledast\ R)\})$$

- Can't describe what happens in case where precondition doesn't hold

$$\{emp\}\ \text{delete}\ p\ \{??\}$$

Forward Reasoning II

- Ideal world seemingly impossible

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \not\Leftrightarrow (\forall R.\ \{R\}\ C\ \{Q * (P \multimap\!\circledast\ R)\})$$

- Can't describe what happens in case where precondition doesn't hold

$$\{emp\}\ \text{delete}\ p\ \{??\}$$

Relationships between Operators

$$
\begin{tikzcd}
\boxed{P \ast Q} \arrow[r, leftrightarrow, bend left, "\text{Galois}"] & \boxed{P \mathrel{-\!\!\ast} Q} \arrow[d, leftrightarrow, "\text{dual}"'] \\
 & \boxed{P \mathrel{-\!\!\circledast} Q}
\end{tikzcd}
$$

Separating ‘Coimplication’

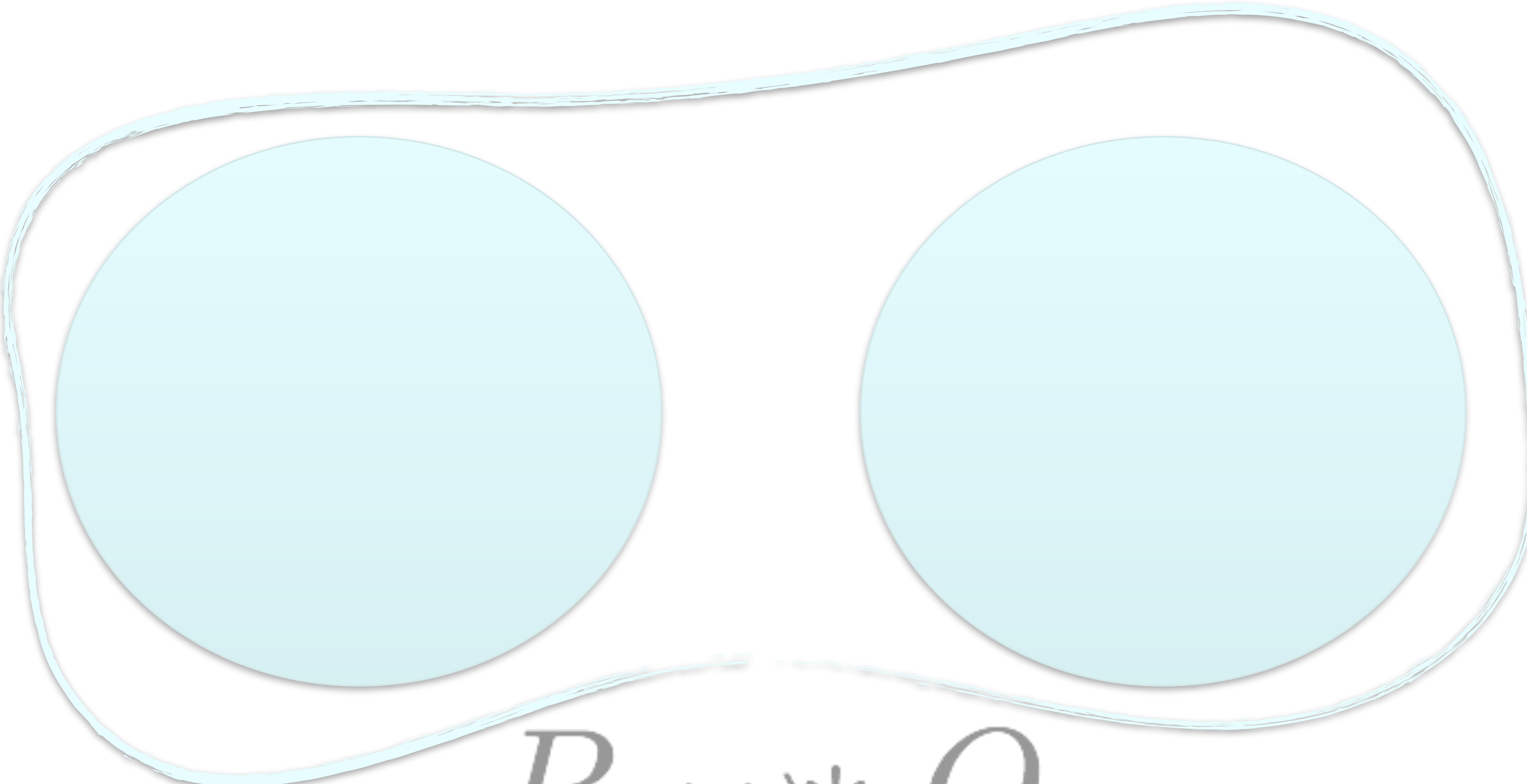
Magic Snake

- $$P \leadsto\!\!\ast\, Q \;\Leftrightarrow\; \neg(P \ast (\neg Q))$$

- Removing P produces Q over the reduction

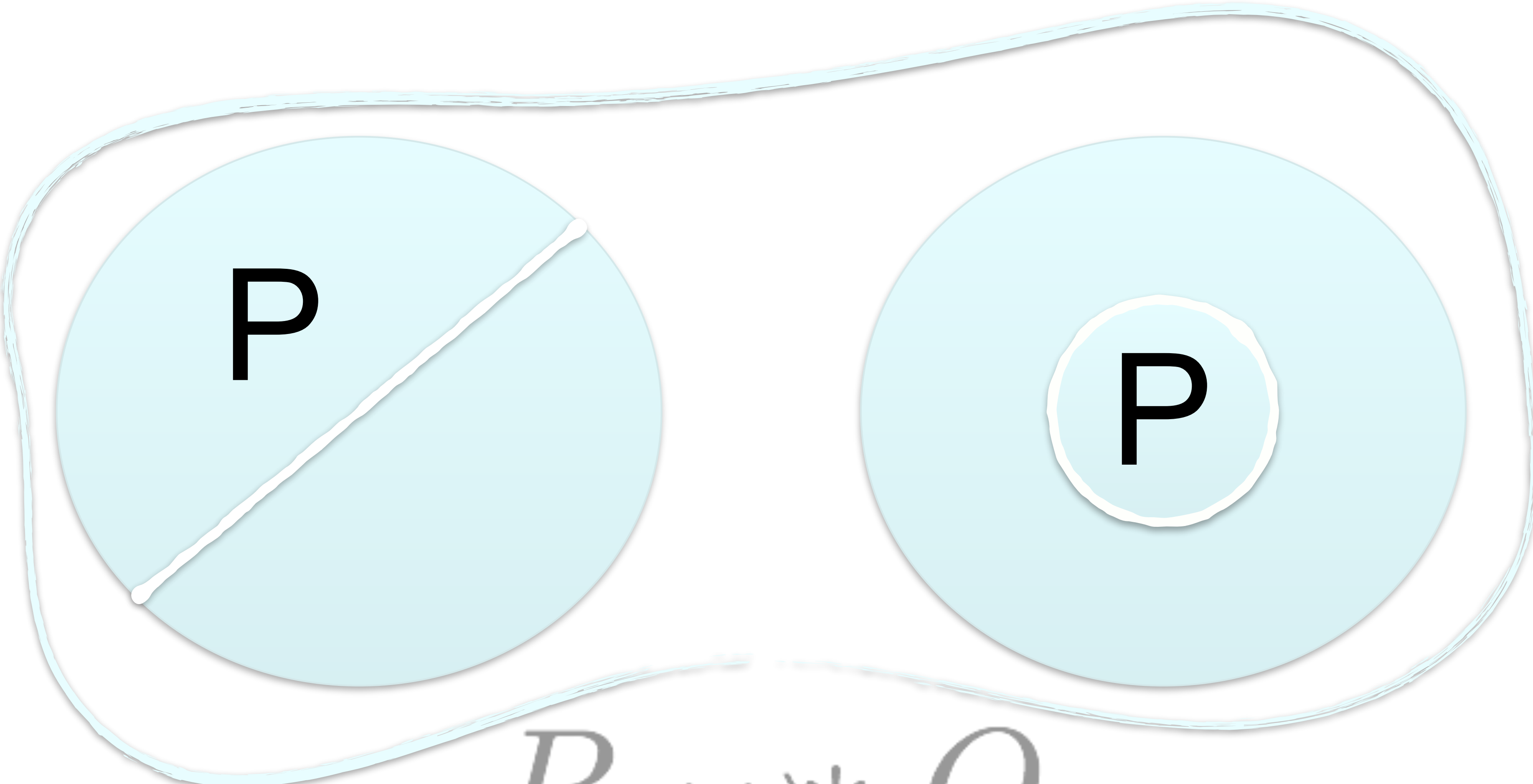
- Every time we can find a P in our heap, the rest of the heap is a Q

Separating Coimplication



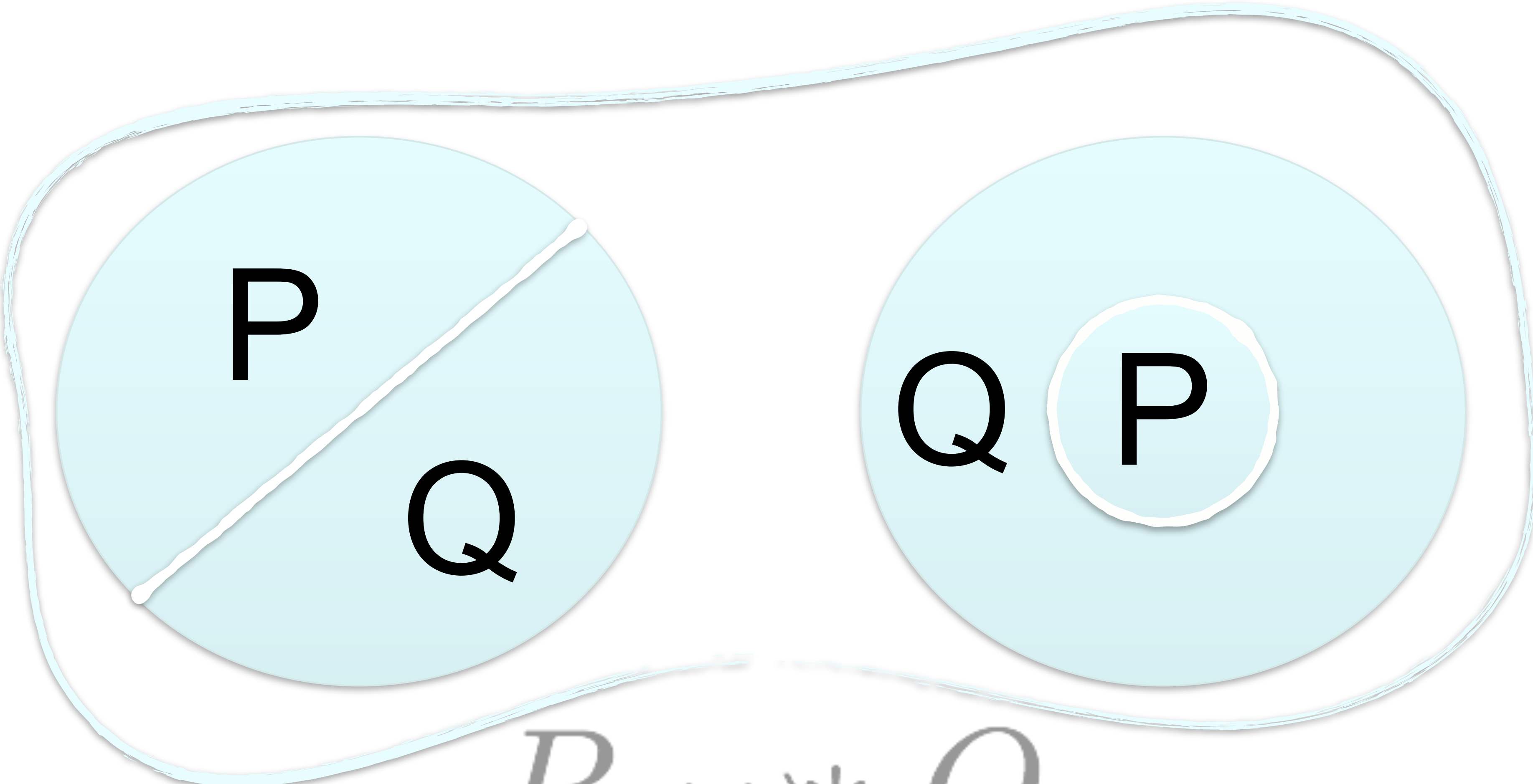
$$P \rightsquigarrow\!\!* \; Q$$

Separating Coimplication



$$P \leadsto\!\!* \; Q$$

Separating Coimplication



$$P \rightsquigarrow\!\!* \; Q$$

Separating ‘Coimplication’

Magic Snake

- $$P \leadsto\!\!\!* \, Q \iff \neg(P * (\neg Q))$$

Separating 'Coimplication'

Magic Snake

- $$P \leadsto\!\!* \, Q \;\Leftrightarrow\; \neg(P * (\neg Q))$$

$$(P \;-\!\!\circledast\; Q \;\Rightarrow\; R) \;\Leftrightarrow\; (Q \;\Rightarrow\; (P \leadsto\!\!* \, Q))$$

(Galois connection)

- many properties come for free from the Galois connection

Relationships between Operators

(complete)

$$
\begin{tikzcd}
P * Q \arrow[rr, leftrightarrow, bend left, "\text{Galois}"] \arrow[d, leftrightarrow, "\text{dual}"] & & P \mathrel{-\!\!*} Q \arrow[d, leftrightarrow, "\text{dual}"'] \\
P \mathrel{\rightsquigarrow\!\!*} Q \arrow[rr, leftrightarrow, bend right, "\text{Galois}"] & & P \mathrel{-\!\!\circledast} Q
\end{tikzcd}
$$

Specifications with Separating Coimplication

- P not satisfied by any subheap

$$P \leadsto\!\!\ast\, false$$

- specification of delete

$$\{p \mapsto \_ \leadsto\!\!\ast\, R\}\ \mathrm{delete}\ p\ \{R\}$$

Back to Forward Reasoning

- Ideal world seemingly impossible

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \ \not\Leftrightarrow\ (\forall R.\ \{R\}\ C\ \{Q * (P \multimap\!\circledast\ R)\})$$

- Relax specifications/requirements

$$\{P * R\}\ C\ \{Q * R\}$$

Back to Forward Reasoning

- Ideal world seemingly impossible

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \;\not\Leftrightarrow\; (\forall R.\ \{R\}\ C\ \{Q * (P \;-\!\!\circledast\; R)\})$$

- Relax specifications/requirements

$$\{P \rightsquigarrow\!\!* R\}\ C\ \{Q * R\}$$

Back to Forward Reasoning

- Ideal world seemingly impossible

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \ \not\Leftrightarrow\ (\forall R.\ \{R\}\ C\ \{Q * (P \multimap\!\circledast\ R)\})$$

- Relax specifications/requirements

$$\{P \rightsquigarrow\!\!* R\}\ C\ \{Q * R\}$$

- another example

$$\{p \mapsto \_ \rightsquigarrow\!\!* R\}\ \mathrm{set\_ptr}\ p\ v\ \{p \mapsto v * R\}$$

Forward Reasoning III

- Ideal world seemingly impossible

$$(\forall R.\ \{P * R\}\ C\ \{Q * R\}) \ \not\Leftrightarrow\ (\forall R.\ \{R\}\ C\ \{Q * (P \multimap\!\circledast\ R)\})$$

- By Galois connections and dualities we get a rule for forward reasoning

$$(\forall R.\ \{P \rightsquigarrow\!\!* R\}\ C\ \{Q * R\}) \ \Leftrightarrow\ (\forall R.\ \{R\}\ C\ \{Q * (P \multimap\!\circledast\ R)\})$$

Forwards Reasoning IV

- allows backwards reasoning without calculating the frame in every step
- supported in Isabelle/HOL
- easy patterns (alternation between implication and conjunction) allow automated simplifications

Forward Reasoning (Problems)

- we restricted ourselves to partial correctness
  - no problem for backwards reasoning
  - but for forward reasoning postcondition does not need to exist

- rules are only valid because we deal with partial correctness

$$\{P\}\ C\ \{Q\} \ \Leftrightarrow\ \forall s.\ P(s) \rightarrow (\forall s'.\ Some\ s' = (C\ s) \rightarrow Q(s'))$$

- if failure occurs anything is possible

$$\{p \not\mapsto \_\}\ \mathrm{set\_ptr}\ p\ v\ \{\mathrm{P{=}NP}\}$$

Unified Correctness

- introduce explicit failure state
- always describe what actually occurs

$$\{P\}\ C\ \{Q\} \iff \forall s.\ P(s) \to Q(C(s))$$

- requirements:
  - failed program execution stays failed

$$\{\text{fail}\}\ C\ \{\text{fail}\}$$

  - failure is separate from False
  - we can determine whether or not we succeeded

Extending the Model

- New Heap Model

  - Same as standard heap model, but we add a boolean flag for failure

$$[p \mapsto v, q \mapsto v'..] \to ([p \mapsto v, q \mapsto v'..], True)$$

$$(h, False) + (h', -) = (h + h', False)$$

- “Infinitely” many failure states
- New operators needed

Extending the Model I

- New Septraction operator for grabbing resources

Old

$$s, h \models P \;-\!\!\circledast\; Q$$

$$\Leftrightarrow \;\; \exists h_2.\, h \text{ subheap of } h_2 \text{ and } s, h_2 - h \models P \text{ and } s, h_2 \models Q$$

Extending the Model I

- New Septraction operator for grabbing resources

Old

$$s, h \models P \;-\!\circledast\; Q$$

$$\Leftrightarrow\ \exists h_2.\, h \text{ subheap of } h_2 \text{ and } s, h_2 - h \models P \text{ and } s, h_2 \models Q$$

$$\Leftrightarrow\ \exists h_1, h_2.\ \models P \text{ and } s, h_1 \models Q \text{ and}$$

$$h \perp h_1,\ h_2 = h + h_1$$

Extending the Model I

- New Septraction operator for grabbing resources

Old

$$s, h \models P \mathbin{-\!\circledast} Q$$

$$\Leftrightarrow \;\; \exists h_2.\, h \text{ subheap of } h_2 \text{ and } s, h_2 - h \models P \text{ and } s, h_2 \models Q$$

$$\Leftrightarrow \;\; \exists h_1, h_2.\, \models P \text{ and } s, h_1 \models Q \text{ and}$$

$$h \perp h_1,\; h_2 = h + h_1$$

$$s, h \models P \mathbin{-\!\circledast} Q$$

New

$$\Leftrightarrow \;\; \exists h_1, h_2.\, \models P \text{ and } s, h_1 \models Q \text{ and}$$

$$\textbf{if } \text{flag}(h) \textbf{ then } h \perp h_1,\; h_2 = h + h_1$$

$$\textbf{else } \text{flag}(h_2) \rightarrow (\text{flag}(h_1) \rightarrow h_1 \perp h_2) \wedge (\text{flag}(h) \rightarrow h \perp h_2)$$

Extending the Model II

- Desired properties are satisfied

- **Consuming**: If the resource is there, we succeed

$$s, h \models (p \mapsto v) \;-\!\!\circledast\; (p \mapsto v) \Rightarrow h = (\mathrm{emp}, true)$$

- **Collapsing:** Once crashed, remain crashed

$$s, h \models P \;-\!\!\circledast\; \text{`fail'} \;\Rightarrow\; h = (\_, false)$$

- **Paraconsistent:** Removing something that didn't exist yield failure

$$s, h \models p \mapsto \_ \;-\!\!\circledast\; \mathrm{emp} \;\Rightarrow\; h = (\_, false)$$

The Good and The Bad

- New operators satisfy Galois connections and dualities

$$
\begin{tikzcd}
P * Q \arrow[r, leftrightarrow, bend left, "\text{Galois}"] \arrow[d, leftrightarrow, "\text{dual}"] & P \mathrel{-\!\!*} Q \arrow[d, leftrightarrow, "\text{dual}"'] \\
P \mathrel{\rightsquigarrow\!\!*} Q \arrow[r, leftrightarrow, bend right, "\text{Galois}"'] & P \mathrel{-\!\!\circledast} Q
\end{tikzcd}
$$

- Separation algebra is identical to the ‘old’ in case of **no failure**

- In case of failure, associativity of separating conjunction is lost

The Good and The Bad

- New operators satisfy Galois connections and dualities

$$\begin{tikzcd} \boxed{P * Q} \arrow[rr, leftrightarrow, bend left, "\text{Galois}"] \arrow[d, leftrightarrow, "\text{dual}"] & & \boxed{P \mathrel{-\!\!*} Q} \arrow[d, leftrightarrow, "\text{dual}"'] \\ \boxed{P \mathrel{\rightsquigarrow\!\!*} Q} \arrow[rr, leftrightarrow, bend right, "\text{Galois}"'] & & \boxed{P \mathrel{-\!\!\circledast} Q} \end{tikzcd}$$

- Separation algebra is identical to the ‘old’ in case of **no failure**

- In case of failure, associativity of separating conjunction is lost

Is this natural? Is this problematic?

Conclusion

- Framework for
  backwards reasoning using weakest preconditions and
  forward reasoning using strongest postconditions for Partial and Unified Correctness

- Automation

- Basic examples demonstrated
  - e.g. Linked-List Reverse

Thank you

Data61
Peter Höfner

t +61 2 9490 5861
e peter.hoefner@data61.csiro.au
w www.data61.csiro.au