Dynamic Separation Logic

Frank de Boer, *Hans-Dieter Hiep*, Stijn de Gouw
hdh@cwi.nl

Leiden University (LIACS)
Centrum Wiskunde & Informatica (CWI)
the Netherlands

MFPS 2023

Overview

- First-order logic (FOL)

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness
- Hoare's logic $\{p\}S\{q\}$

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness

- Hoare's logic $\{p\}S\{q\}$
  - simple **while** programs
  - rich program semantics
  - relative completeness

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness
- Hoare's logic $\{p\}S\{q\}$
  - simple **while** programs
  - rich program semantics
  - relative completeness
- First-order dynamic logic

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness
- Hoare's logic $\{p\}S\{q\}$
  - simple **while** programs
  - rich program semantics
  - relative completeness
- First-order dynamic logic
  - modalities $[S]q$
  - embedding $p \rightarrow [S]q$
  - expressivity $\neg[S]\neg q$

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness
- Hoare's logic $\{p\}S\{q\}$
  - simple **while** programs
  - rich program semantics
  - relative completeness
- First-order dynamic logic
  - modalities $[S]q$
  - embedding $p \rightarrow [S]q$
  - expressivity $\neg[S]\neg q$
- Separation logic (SL)

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness
- Hoare's logic $\{p\}S\{q\}$
  - simple **while** programs
  - rich program semantics
  - relative completeness
- First-order dynamic logic
  - modalities $[S]q$
  - embedding $p \rightarrow [S]q$
  - expressivity $\neg[S]\neg q$
- Separation logic (SL)
  - separating conjunction $*$
  - magic wand $-\!*$
  - aliasing, footprints

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness
- Hoare's logic $\{p\}S\{q\}$
  - simple **while** programs
  - rich program semantics
  - relative completeness
- First-order dynamic logic
  - modalities $[S]q$
  - embedding $p \rightarrow [S]q$
  - expressivity $\neg[S]\neg q$
- Separation logic (SL)
  - separating conjunction $*$
  - magic wand $-\!*$
  - aliasing, footprints
- Reynolds' logic $\{p\}S\{q\}$

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness
- Separation logic (SL)
  - separating conjunction $*$
  - magic wand $-\!*$
  - aliasing, footprints
- Hoare's logic $\{p\}S\{q\}$
  - simple **while** programs
  - rich program semantics
  - relative completeness
- Reynolds' logic $\{p\}S\{q\}$
  - pointer programs
  - frame rule
  - relative completeness
- First-order dynamic logic
  - modalities $[S]q$
  - embedding $p \to [S]q$
  - expressivity $\neg[S]\neg q$

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness
- Separation logic (SL)
  - separating conjunction $*$
  - magic wand $-\!*$
  - aliasing, footprints
- Hoare's logic $\{p\}S\{q\}$
  - simple **while** programs
  - rich program semantics
  - relative completeness
- Reynolds' logic $\{p\}S\{q\}$
  - pointer programs
  - frame rule
  - relative completeness
- First-order dynamic logic
  - modalities $[S]q$
  - embedding $p \rightarrow [S]q$
  - expressivity $\neg[S]\neg q$
- Dynamic separation logic

Overview

- First-order logic (FOL)
  - rich proof theory
  - rich model theory
  - semantic completeness
- Separation logic (SL)
  - separating conjunction $*$
  - magic wand $-\!*$
  - aliasing, footprints
- Hoare's logic $\{p\}S\{q\}$
  - simple **while** programs
  - rich program semantics
  - relative completeness
- Reynolds' logic $\{p\}S\{q\}$
  - pointer programs
  - frame rule
  - relative completeness
- First-order dynamic logic
  - modalities $[S]q$
  - embedding $p \rightarrow [S]q$
  - expressivity $\neg[S]\neg q$
- Dynamic separation logic
  - this talk

Contributions

- Reynolds' logic has different axiomatizations such that $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$ (relative completeness: oracle, expressivity)

Contributions

- Reynolds' logic has different axiomatizations such that $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$ (relative completeness: oracle, expressivity)
  - Local axioms plus frame rule

Contributions

- Reynolds' logic has different axiomatizations such that $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$ (relative completeness: oracle, expressivity)
  - Local axioms plus frame rule
  - Global weakest precondition (WP) axiomatization

Contributions

- Reynolds' logic has different axiomatizations such that $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$ (relative completeness: oracle, expressivity)
  - Local axioms plus frame rule
  - Global weakest precondition (WP) axiomatization
  - ...

Contributions

- Reynolds' logic has different axiomatizations such that $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$ (relative completeness: oracle, expressivity)
  - Local axioms plus frame rule
  - Global weakest precondition (WP) axiomatization
  - ...
  - These do **not** analyze the logical structure of p or q

Contributions

- Reynolds' logic has different axiomatizations such that $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$ (relative completeness: oracle, expressivity)
  - Local axioms plus frame rule
  - Global weakest precondition (WP) axiomatization
  - ...
  - These do **not** analyze the logical structure of p or q
- This talk introduces Dynamic Separation Logic (DSL) $\models \{p\}\ S\ \{q\}$ if and only if $\models p \rightarrow [S]q$

Contributions

- Reynolds' logic has different axiomatizations such that $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$ (relative completeness: oracle, expressivity)
  - Local axioms plus frame rule
  - Global weakest precondition (WP) axiomatization
  - ...
  - These do **not** analyze the logical structure of p or q

- This talk introduces Dynamic Separation Logic (DSL) $\models \{p\}\ S\ \{q\}$ if and only if $\models p \rightarrow [S]q$
  - Axiomatization (useful for eliminating modalities)

Contributions

- Reynolds' logic has different axiomatizations such that
  $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$
  (relative completeness: oracle, expressivity)
  - Local axioms plus frame rule
  - Global weakest precondition (WP) axiomatization
  - ...
  - These do **not** analyze the logical structure of p or q

- This talk introduces Dynamic Separation Logic (DSL)
  $\models \{p\}\ S\ \{q\}$ if and only if $\models p \rightarrow [S]q$
  - Axiomatization (useful for eliminating modalities)
  - Weakest preconditions that **do** analyze logical structure

Contributions

- Reynolds' logic has different axiomatizations such that $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$ (relative completeness: oracle, expressivity)
  - Local axioms plus frame rule
  - Global weakest precondition (WP) axiomatization
  - ...
  - These do **not** analyze the logical structure of p or q
- This talk introduces Dynamic Separation Logic (DSL) $\models \{p\}\ S\ \{q\}$ if and only if $\models p \rightarrow [S]q$
  - Axiomatization (useful for eliminating modalities)
  - Weakest preconditions that **do** analyze logical structure
  - ...

Contributions

- Reynolds' logic has different axiomatizations such that $\models \{p\}\ S\ \{q\}$ if and only if $\vdash \{p\}\ S\ \{q\}$ (relative completeness: oracle, expressivity)
  - Local axioms plus frame rule
  - Global weakest precondition (WP) axiomatization
  - ...
  - These do **not** analyze the logical structure of p or q
- This talk introduces Dynamic Separation Logic (DSL) $\models \{p\}\ S\ \{q\}$ if and only if $\models p \rightarrow [S]q$
  - Axiomatization (useful for eliminating modalities)
  - Weakest preconditions that **do** analyze logical structure
  - ...
- Leads to surprising equivalences in Separation Logic

Separation logic (syntax)

Signature:
standard signature of arithmetic: $0, 1, +, \times, \leq$

Language:
$p, q ::= b \mid (e \hookrightarrow e') \mid p \wedge q \mid p \rightarrow q \mid \forall x p \mid p * q \mid p \mathrel{-\!\!*} q$

Separation logic (syntax)

Signature:
standard signature of arithmetic: $0, 1, +, \times, \leq$

Language:
$p, q ::= b \mid (e \hookrightarrow e') \mid p \wedge q \mid p \to q \mid \forall x p \mid p * q \mid p \mathrel{-\!\!*} q$

Derived notions:

- classical $\exists x$ and $\vee$
- $(e \hookrightarrow -)$ as $\exists x(e \hookrightarrow x)$, and **emp** as $\forall x(x \not\hookrightarrow -)$
- $(e \mapsto e')$ as $(e \hookrightarrow e') \wedge (\forall x.(x \hookrightarrow e) \to x = e)$

Separation logic (syntax)

Signature:
standard signature of arithmetic: $0, 1, +, \times, \leq$

Language:
$p, q ::= b \mid (e \hookrightarrow e') \mid p \wedge q \mid p \rightarrow q \mid \forall x p \mid p * q \mid p -\!\!* \; q$

Derived notions:

- classical $\exists x$ and $\vee$
- $(e \hookrightarrow -)$ as $\exists x (e \hookrightarrow x)$, and **emp** as $\forall x (x \not\hookrightarrow -)$
- $(e \mapsto e')$ as $(e \hookrightarrow e') \wedge (\forall x.(x \hookrightarrow e) \rightarrow x = e)$

Examples

- $(e \mapsto e') \rightarrow (e \hookrightarrow e')$ but $(e \hookrightarrow e') \not\rightarrow (e \mapsto e')$
- $(e \hookrightarrow e') \equiv (e \mapsto e') *$ **true**

Separation logic (semantics)

Interpretation:
$h, s \models p$, given heap $h : \mathbb{Z} \rightharpoonup_{\text{fin}} \mathbb{Z}$ and store $s : V \to \mathbb{Z}$

- Tarski-style, standard classical logic

Separation logic (semantics)

Interpretation:
$h, s \models p$, given heap $h : \mathbb{Z} \rightharpoonup_{\text{fin}} \mathbb{Z}$ and store $s : V \to \mathbb{Z}$

- Tarski-style, standard classical logic
- $h, s \models (e \hookrightarrow e')$ iff $s(e) \in dom(h)$ and $h(s(e)) = s(e')$
- $h, s \models p * q$ iff $h_1, s \models p$ and $h_2, s \models q$ for some $h_1 \uplus h_2 = h$
- $h, s \models p \mathrel{-\!\!*} q$ iff $h', s \models p$ implies $h \uplus h', s \models q$ for all $h' \perp h$

Separation logic (semantics)

Interpretation:

$h, s \models p$, given heap $h : \mathbb{Z} \rightharpoonup_{\text{fin}} \mathbb{Z}$ and store $s : V \rightarrow \mathbb{Z}$

- Tarski-style, standard classical logic
- $h, s \models (e \hookrightarrow e')$ iff $s(e) \in dom(h)$ and $h(s(e)) = s(e')$
- $h, s \models p * q$ iff $h_1, s \models p$ and $h_2, s \models q$ for some $h_1 \uplus h_2 = h$
- $h, s \models p \mathrel{-\!\!*} q$ iff $h', s \models p$ implies $h \uplus h', s \models q$ for all $h' \perp h$

Examples

- $(x \mapsto 1) \wedge (y \mapsto 1) \not\rightarrow (x \mapsto 1) * (y \mapsto 1)$

Separation logic (semantics)

Interpretation:
$h, s \models p$, given heap $h : \mathbb{Z} \rightharpoonup_{\text{fin}} \mathbb{Z}$ and store $s : V \to \mathbb{Z}$

- Tarski-style, standard classical logic
- $h, s \models (e \hookrightarrow e')$ iff $s(e) \in dom(h)$ and $h(s(e)) = s(e')$
- $h, s \models p * q$ iff $h_1, s \models p$ and $h_2, s \models q$ for some $h_1 \uplus h_2 = h$
- $h, s \models p \mathrel{-\!\!*} q$ iff $h', s \models p$ implies $h \uplus h', s \models q$ for all $h' \perp h$

Examples

- $(x \mapsto 1) \wedge (y \mapsto 1) \not\rightarrow (x \mapsto 1) * (y \mapsto 1)$
- $(x \hookrightarrow 1) * \mathbf{emp} \not\rightarrow (x \hookrightarrow 1) \wedge \mathbf{emp}$

Separation logic (semantics)

Interpretation:

$h, s \models p$, given heap $h : \mathbb{Z} \rightharpoonup_{\text{fin}} \mathbb{Z}$ and store $s : V \to \mathbb{Z}$

- Tarski-style, standard classical logic
- $h, s \models (e \hookrightarrow e')$ iff $s(e) \in dom(h)$ and $h(s(e)) = s(e')$
- $h, s \models p * q$ iff $h_1, s \models p$ and $h_2, s \models q$ for some $h_1 \uplus h_2 = h$
- $h, s \models p \mathrel{-\!\!*} q$ iff $h', s \models p$ implies $h \uplus h', s \models q$ for all $h' \perp h$

Examples

- $(x \mapsto 1) \wedge (y \mapsto 1) \not\to (x \mapsto 1) * (y \mapsto 1)$
- $(x \hookrightarrow 1) * \mathbf{emp} \not\to (x \hookrightarrow 1) \wedge \mathbf{emp}$
- $p * (p \mathrel{-\!\!*} q) \to q$

Pointer programs

Programming language:
$S ::= x := e \mid x := [e] \mid [x] := e \mid x := \textbf{cons}(e) \mid \textbf{dispose}(e) \mid \ldots$

Big-step operational semantics:
$(S, h, s) \Rightarrow (h', s')$ or $(S, h, s) \Rightarrow$ **fail** or neither

Pointer programs

Programming language:
$S ::= x := e \mid x := [e] \mid [x] := e \mid x := \textbf{cons}(e) \mid \textbf{dispose}(e) \mid \ldots$

Big-step operational semantics:
$(S, h, s) \Rightarrow (h', s')$ or $(S, h, s) \Rightarrow \textbf{fail}$ or neither

- $(x := [e], h, s) \Rightarrow (h, s[x := h(s(e))])$ if $s(e) \in dom(h)$
- $(x := [e], h, s) \Rightarrow \textbf{fail}$ if $s(e) \notin dom(h)$

Pointer programs

Programming language:
$S ::= x := e \mid x := [e] \mid [x] := e \mid x := \textbf{cons}(e) \mid \textbf{dispose}(e) \mid \ldots$

Big-step operational semantics:
$(S, h, s) \Rightarrow (h', s')$ or $(S, h, s) \Rightarrow \textbf{fail}$ or neither

- $(x := [e], h, s) \Rightarrow (h, s[x := h(s(e))])$ if $s(e) \in dom(h)$
- $(x := [e], h, s) \Rightarrow \textbf{fail}$ if $s(e) \notin dom(h)$
- $([x] := e, h, s) \Rightarrow (h[s(x) := s(e)], s)$ if $s(e) \in dom(h)$
- $([x] := e, h, s) \Rightarrow \textbf{fail}$ if $s(e) \notin dom(h)$

Pointer programs

Programming language:
$S ::= x := e \mid x := [e] \mid [x] := e \mid x := \mathbf{cons}(e) \mid \mathbf{dispose}(e) \mid \ldots$

Big-step operational semantics:
$(S, h, s) \Rightarrow (h', s')$ or $(S, h, s) \Rightarrow \mathbf{fail}$ or neither

- $(x := [e], h, s) \Rightarrow (h, s[x := h(s(e))])$ if $s(e) \in dom(h)$
- $(x := [e], h, s) \Rightarrow \mathbf{fail}$ if $s(e) \notin dom(h)$
- $([x] := e, h, s) \Rightarrow (h[s(x) := s(e)], s)$ if $s(e) \in dom(h)$
- $([x] := e, h, s) \Rightarrow \mathbf{fail}$ if $s(e) \notin dom(h)$
- $(x := \mathbf{cons}(e), h, s) \Rightarrow (h[n := s(e)], s[x := n])$ where $n \notin dom(h)$

Pointer programs

Programming language:
$S ::= x := e \mid x := [e] \mid [x] := e \mid x := \textbf{cons}(e) \mid \textbf{dispose}(e) \mid \ldots$

Big-step operational semantics:
$(S, h, s) \Rightarrow (h', s')$ or $(S, h, s) \Rightarrow \textbf{fail}$ or neither

- $(x := [e], h, s) \Rightarrow (h, s[x := h(s(e))])$ if $s(e) \in dom(h)$
- $(x := [e], h, s) \Rightarrow \textbf{fail}$ if $s(e) \notin dom(h)$
- $([x] := e, h, s) \Rightarrow (h[s(x) := s(e)], s)$ if $s(e) \in dom(h)$
- $([x] := e, h, s) \Rightarrow \textbf{fail}$ if $s(e) \notin dom(h)$
- $(x := \textbf{cons}(e), h, s) \Rightarrow (h[n := s(e)], s[x := n])$ where $n \notin dom(h)$
- $(\textbf{dispose}(x), h, s) \Rightarrow (h[s(x) := \bot], s)$ if $s(e) \in dom(h)$
- $(\textbf{dispose}(x), h, s) \Rightarrow \textbf{fail}$ if $s(e) \notin dom(h)$

Reynolds' logic

Strong partial correctness axiomatization:

- all rules and axioms of Hoare's logic

Reynolds' logic

Strong partial correctness axiomatization:

- all rules and axioms of Hoare's logic
- $\{\exists y.(e \hookrightarrow y) \wedge p[y/x]\}\ x := [e]\ \{p\}$
- $\{(x \mapsto -) * ((x \mapsto e) -\!\!* p)\}\ [x] := e\ \{p\}$
- $\{\forall x.(x \mapsto e) -\!\!* p\}\ x := \mathbf{cons}(e)\ \{p\}$ $\qquad (x \notin FV(e))$
- $\{(x \mapsto -) * p\}\ \mathbf{dispose}(x)\ \{p\}$

Reynolds' logic

Strong partial correctness axiomatization:

- all rules and axioms of Hoare's logic
- $\{\exists y.(e \hookrightarrow y) \wedge p[y/x]\}\ x := [e]\ \{p\}$
- $\{(x \mapsto -) * ((x \mapsto e) -\!\!* \ p)\}\ [x] := e\ \{p\}$
- $\{\forall x.(x \mapsto e) -\!\!* \ p\}\ x := \mathbf{cons}(e)\ \{p\}$ $\qquad (x \notin FV(e))$
- $\{(x \mapsto -) * p\}\ \mathbf{dispose}(x)\ \{p\}$
- the frame rule

$$\frac{\{p\}\ S\ \{q\}}{\{p * r\}\ S\ \{q * r\}}$$

Reynolds' logic

Strong partial correctness axiomatization:

- all rules and axioms of Hoare's logic
- $\{\exists y.(e \hookrightarrow y) \wedge p[y/x]\}\ x := [e]\ \{p\}$
- $\{(x \mapsto -) * ((x \mapsto e) \mathrel{-\!\!*} p)\}\ [x] := e\ \{p\}$
- $\{\forall x.(x \mapsto e) \mathrel{-\!\!*} p\}\ x := \mathbf{cons}(e)\ \{p\}$ $\qquad (x \notin FV(e))$
- $\{(x \mapsto -) * p\}\ \mathbf{dispose}(x)\ \{p\}$
- the frame rule

$$\frac{\{p\}\ S\ \{q\}}{\{p * r\}\ S\ \{q * r\}}$$

Soundness and relative completeness
(Bannister, Höfner, Klein 2018)
(Tatsuta, Chin, Al Ameen, 2019)

Dynamic separation logic

Language:

$p, q ::= b \mid (e \hookrightarrow e') \mid p \wedge q \mid p \rightarrow q \mid \forall x p \mid p * q \mid p \mathrel{-\!\!*} q \mid [S]p$

Interpretation:

- $h, s \models [S]p$ iff $(S, h, s) \not\Rightarrow$ **fail** and $(S, h, s) \Rightarrow (h', s')$ implies $h', s' \models p$

Fact

- $\models \{[S]q\}\ S\ \{q\}$
- $\models \{p\}\ S\ \{q\}$ *implies* $p \rightarrow [S]q$

Dynamic separation logic

Language:

$p, q ::= b \mid (e \hookrightarrow e') \mid p \wedge q \mid p \rightarrow q \mid \forall x p \mid p * q \mid p -\!\!* \, q \mid [S]p$

Interpretation:

- $h, s \models [S]p$ iff $(S, h, s) \not\Rightarrow$ **fail** and $(S, h, s) \Rightarrow (h', s')$ implies $h', s' \models p$

Fact

- $\models \{[S]q\}\ S\ \{q\}$
- $\models \{p\}\ S\ \{q\}$ *implies* $p \rightarrow [S]q$

Question. Can we analyze $[S]p$ compositionally in p?

Dynamic separation logic

Language:

$p, q ::= b \mid (e \hookrightarrow e') \mid p \wedge q \mid p \rightarrow q \mid \forall x p \mid p * q \mid p \mathrel{-\!\!*} q \mid [S]p$

Interpretation:

- $h, s \models [S]p$ iff $(S, h, s) \not\Rightarrow$ **fail** and $(S, h, s) \Rightarrow (h', s')$ implies $h', s' \models p$

Fact

- $\models \{[S]q\}\ S\ \{q\}$
- $\models \{p\}\ S\ \{q\}$ *implies* $p \rightarrow [S]q$

Question. Can we analyze $[S]p$ compositionally in p?
Answer. Yes, using equivalence axioms, allowing rewriting

Axiomatization

Introduce pseudo-instructions:

- $(\langle x \rangle := e, h, s) \Rightarrow (h[s(x) := s(e)], s)$ unconditionally
- $(\langle x \rangle := \bot, h, s) \Rightarrow (h[s(x) := \bot], s)$ unconditionally

Axiomatization

Introduce pseudo-instructions:

- $(\langle x \rangle := e, h, s) \Rightarrow (h[s(x) := s(e)], s)$ unconditionally
- $(\langle x \rangle := \bot, h, s) \Rightarrow (h[s(x) := \bot], s)$ unconditionally

$$[[x] := e]p \equiv (x \hookrightarrow -) \wedge [\langle x \rangle := e]p \qquad \text{(E6)}$$

$$[x := \mathbf{cons}(e)]p \equiv \forall x.(x \not\hookrightarrow -) \rightarrow [\langle x \rangle := e]p \qquad \text{(E7)}$$

$$[\mathbf{dispose}(x)]p \equiv (x \hookrightarrow -) \wedge [\langle x \rangle := \bot]p \qquad \text{(E8)}$$

Axiomatization

Introduce pseudo-instructions:

- $(\langle x \rangle := e, h, s) \Rightarrow (h[s(x) := s(e)], s)$ unconditionally
- $(\langle x \rangle := \bot, h, s) \Rightarrow (h[s(x) := \bot], s)$ unconditionally

$$[[x] := e]p \equiv (x \hookrightarrow -) \wedge [\langle x \rangle := e]p \quad \text{(E6)}$$

$$[x := \textbf{cons}(e)]p \equiv \forall x.(x \not\hookrightarrow -) \to [\langle x \rangle := e]p \quad \text{(E7)}$$

$$[\textbf{dispose}(x)]p \equiv (x \hookrightarrow -) \wedge [\langle x \rangle := \bot]p \quad \text{(E8)}$$

$$[\langle x \rangle := e]b \equiv b \quad \text{(E9)}$$

$$[\langle x \rangle := e](e' \hookrightarrow e'') \equiv (x = e' \wedge e'' = e) \vee (x \neq e' \wedge e' \hookrightarrow e'') \quad \text{(E10)}$$

$$[\langle x \rangle := e](p * q) \equiv ([\langle x \rangle := e]p * q') \vee (p' * [\langle x \rangle := e]q) \quad \text{(E11)}$$

$$[\langle x \rangle := e](p \mathrel{-\!\!*} q) \equiv p' \mathrel{-\!\!*} [\langle x \rangle := e]q \quad \text{(E12)}$$

where $p' = p \wedge (x \not\hookrightarrow -)$ and $q' = q \wedge (x \not\hookrightarrow -)$ and
$[\langle x \rangle := e]$ works like substitution for logical connectives (E1-3)

Dynamic separation logic

Introduce pseudo-instructions:

- $(\langle x \rangle := e, h, s) \Rightarrow (h[s(x) := s(e)], s)$ unconditionally
- $(\langle x \rangle := \bot, h, s) \Rightarrow (h[s(x) := \bot], s)$ unconditionally

$$[[x] := e]p \equiv (x \hookrightarrow -) \wedge [\langle x \rangle := e]p \quad \text{(E6)}$$

$$[x := \mathbf{cons}(e)]p \equiv \forall x.(x \not\hookrightarrow -) \rightarrow [\langle x \rangle := e]p \quad \text{(E7)}$$

$$[\mathbf{dispose}(x)]p \equiv (x \hookrightarrow -) \wedge [\langle x \rangle := \bot]p \quad \text{(E8)}$$

$$[\langle x \rangle := \bot]b \equiv b \quad \text{(E13)}$$

$$[\langle x \rangle := \bot](e \hookrightarrow e') \equiv (x \neq e \wedge (e \hookrightarrow e')) \quad \text{(E14)}$$

$$[\langle x \rangle := \bot](p * q) \equiv [\langle x \rangle := \bot]p * [\langle x \rangle := \bot]q \quad \text{(E15)}$$

$$[\langle x \rangle := \bot](p \mathrel{-\!\!*} q) \equiv (p' \mathrel{-\!\!*} [\langle x \rangle := \bot]q) \wedge \forall y.[\langle x \rangle := y]p \mathrel{-\!\!*} [\langle x \rangle := y]q \quad \text{(E16)}$$

where $p' = p \wedge (x \not\hookrightarrow -)$ and $[\langle x \rangle := \bot]$ works for $\wedge, \rightarrow, \forall$ (E1-3)

Surprising impact

$$\equiv$$

$$[[x] := 0](y \hookrightarrow z)$$

$$\equiv$$

Surprising impact

$$(x \mapsto -) * ((x \mapsto 0) \mathrel{-\!\!*} (y \hookrightarrow z))$$

$$\equiv$$

$$[[x] := 0](y \hookrightarrow z)$$

$$\equiv$$

Surprising impact

$$(x \mapsto -) * ((x \mapsto 0) \mathrel{-\!\!*} (y \hookrightarrow z))$$

$$\equiv$$

$$[[x] := 0](y \hookrightarrow z)$$

$$\equiv$$

$$(x \hookrightarrow -) \wedge ((y = x \wedge z = 0) \vee (y \neq x \wedge y \hookrightarrow z))$$

Surprising impact

$$(x \mapsto -) * ((x \mapsto 0) \mathrel{-\!\!*} (y \hookrightarrow z))$$

$$\equiv$$

$$[[x] := 0](y \hookrightarrow z)$$

$$\equiv$$

$$(x \hookrightarrow -) \wedge ((y = x \wedge z = 0) \vee (y \neq x \wedge y \hookrightarrow z))$$

- Bug in CVC4-SL, not equivalent in CVC5-SL (incomplete)
- Not provable in Iris, needs more axioms (incomplete)
- No proof known in VerCors / Viper (incomplete)

Contributions

- This talk has introduced Dynamic Separation Logic (DSL)
  - Axiomatization (useful for eliminating modalities)
  - Novel weakest preconditions axiomatization

Contributions

- This talk has introduced Dynamic Separation Logic (DSL)
  - Axiomatization (useful for eliminating modalities)
  - Novel weakest preconditions axiomatization
- Surprising equivalences

Contributions

- This talk has introduced Dynamic Separation Logic (DSL)
  - Axiomatization (useful for eliminating modalities)
  - Novel weakest preconditions axiomatization
- Surprising equivalences
- See the paper for more:
  - Expressiveness: completeness of local axioms **without** $-\!\!*$

Contributions

- This talk has introduced Dynamic Separation Logic (DSL)
  - Axiomatization (useful for eliminating modalities)
  - Novel weakest preconditions axiomatization
- Surprising equivalences
- See the paper for more:
  - Expressiveness: completeness of local axioms **without** $-\!\!*$
  - Robustness: strongest postcondition axiomatization

Contributions

- This talk has introduced Dynamic Separation Logic (DSL)
  - Axiomatization (useful for eliminating modalities)
  - Novel weakest preconditions axiomatization
- Surprising equivalences
- See the paper for more:
  - Expressiveness: completeness of local axioms **without** $-\!\!*$
  - Robustness: strongest postcondition axiomatization
  - Gracefulness: comparison to other axiomatizations

Contributions

- This talk has introduced Dynamic Separation Logic (DSL)
  - Axiomatization (useful for eliminating modalities)
  - Novel weakest preconditions axiomatization
- Surprising equivalences
- See the paper for more:
  - Expressiveness: completeness of local axioms **without** $-\!\!*$
  - Robustness: strongest postcondition axiomatization
  - Gracefulness: comparison to other axiomatizations
- **New Foundations for Separation Logic**

Contributions

- This talk has introduced Dynamic Separation Logic (DSL)
  - Axiomatization (useful for eliminating modalities)
  - Novel weakest preconditions axiomatization
- Surprising equivalences
- See the paper for more:
  - Expressiveness: completeness of local axioms **without** $-\!\!*$
  - Robustness: strongest postcondition axiomatization
  - Gracefulness: comparison to other axiomatizations
- **New Foundations for Separation Logic**
- Future work:
  - extending KeY (JavaDL) to use dynamic separation logic