FMSoft
Lecture 14 — Wlp- and vc-friendly specifications in separation logic

(lecture version)

Tadeusz Litak

January 29, 2019

Informatik 8, FAU Erlangen-Nürnberg

Back to dynamic programs

- We have spent quite some time with weakest (liberal) preconditions

Back to dynamic programs

- We have spent quite some time with weakest (liberal) preconditions
- But all of this has been done for the simple language IMP

Back to dynamic programs

- We have spent quite some time with weakest (liberal) preconditions
- But all of this has been done for the simple language IMP
- It is time to return to DIMP and separation logic

Back to dynamic programs

- We have spent quite some time with weakest (liberal) preconditions
- But all of this has been done for the simple language IMP
- It is time to return to DIMP and separation logic
- We have not discussed semantics, so we will not try to do relative completeness

  and recall from the last lecture such results only go so far anyway

Back to dynamic programs

- We have spent quite some time with weakest (liberal) preconditions
- But all of this has been done for the simple language IMP
- It is time to return to DIMP and separation logic
- We have not discussed semantics, so we will not try to do relative completeness

  and recall from the last lecture such results only go so far anyway
- But even on a purely syntactic level, do we have all the pieces for computing a DIMP analogue of *wlp*?

Reminder of small axioms

- The small axiom for allocation:

  $\vdash \{(\mathtt{X}==i)\wedge \mathtt{emp}\}\ \mathtt{X}\ :=\ \mathtt{CONS}\ \overline{a}\ \ \{\mathtt{X}:\!\!\to \overline{a}[i/\mathtt{X}]\}$

- The small axiom for lookup:

  $\vdash \{(\mathtt{X}==i)\wedge(a :\!\!\to i')\}\ \mathtt{X}\ :=\ [a]\ \{(\mathtt{X}==i')\wedge(a[i/\mathtt{X}] :\!\!\to i')\}$

- The small axiom for mutation:

  $\vdash \{\exists i.a :\!\!\to \_\}\ [a]\ :=\ a'\ \ \{a :\!\!\to a'\}$

- The small axiom for deallocation:

  $\vdash \{\exists i.a :\!\!\to \_\}\ \mathtt{DISPOSE}\ a\ \ \{\mathtt{emp}\}$

- Small axioms indeed!
  - Precondition for CONS and postcondition for DISPOSE: empty heaplet
  - Postcondition for CONS: heaplet the size of $\overline{a}$
  - Everywhere else: heaplet size 1

Reminder of global specifications (via the frame rule)

- $\vdash \{(a :\!\!\rightarrow \_) * B\}$ DISPOSE a $\{B\}$
- $\vdash \{((X = i) \wedge (a :\!\!\rightarrow i')) * A[i'/X]\}$ X:=[a] $\{(a[i/X] :\!\!\rightarrow i') * A\}$
- $\vdash \{(X = i) \wedge A\}$ X:=CONS $\overline{a}$ $\{(X :\!\!\rightarrow \overline{a}[i/X]) * A[i/X])\}$
- $\vdash \{(a :\!\!\rightarrow \_) * B\}$ $[a] := a'$ $\{(a :\!\!\rightarrow a') * B\}$
- Only the one for deallocation is of the right form for computing weakest (liberal) preconditions!
- That is, it has an arbitrary assertion as a precondition
- If you recall though, we needed certain features of the boolean language to compute them

- We've had this bit in the definition of *wlp* for conditionals:

$$wlp(\texttt{IF}\ b\ \texttt{THEN}\ c_1\ \texttt{ELSE}\ c_2, B) :=$$
$$(b \wedge wlp(c_1, B)) \vee (\neg b \wedge wlp(c_2, B))$$

- We've had this bit in the definition of *wlp* for conditionals:

$$wlp(\texttt{IF}\ b\ \texttt{THEN}\ c_1\ \texttt{ELSE}\ c_2, B) :=$$
$$(b \wedge wlp(c_1, B)) \vee (\neg b \wedge wlp(c_2, B))$$

- We've also defined the sequence for while as follows

$$A_0^{b,B,c} := \top$$
$$A_{n+1}^{b,B,c} := (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$$

- We've had this bit in the definition of *wlp* for conditionals:

$$wlp(\texttt{IF}\ b\ \texttt{THEN}\ c_1\ \texttt{ELSE}\ c_2, B) :=$$
$$(b \wedge wlp(c_1, B)) \vee (\neg b \wedge wlp(c_2, B))$$

- We've also defined the sequence for while as follows

$$A_0^{b,B,c} := \top$$
$$A_{n+1}^{b,B,c} := (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$$

- Notice a pattern here?

- We've had this bit in the definition of *wlp* for conditionals:

$$wlp(\texttt{IF}\ b\ \texttt{THEN}\ c_1\ \texttt{ELSE}\ c_2, B) :=$$
$$(b \wedge wlp(c_1, B)) \vee (\neg b \wedge wlp(c_2, B))$$

- We've also defined the sequence for while as follows

$$A_0^{b,B,c} := \top$$
$$A_{n+1}^{b,B,c} := (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$$

- Notice a pattern here?
- Exercise: Show that

$$(b \wedge A) \vee (\neg b \wedge C) \equiv (b \to A) \wedge (\neg b \to C)$$

- We've had this bit in the definition of *wlp* for conditionals:

$$wlp(\texttt{IF}\ b\ \texttt{THEN}\ c_1\ \texttt{ELSE}\ c_2, B) :=$$
$$(b \wedge wlp(c_1, B)) \vee (\neg b \wedge wlp(c_2, B))$$

- We've also defined the sequence for while as follows

$$A_0^{b,B,c} := \top$$
$$A_{n+1}^{b,B,c} := (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$$

- Notice a pattern here?
- Exercise: Show that

$$(b \wedge A) \vee (\neg b \wedge C) \equiv (b \to A) \wedge (\neg b \to C)$$

- It is crucial that $\to$ is a (definable) implication well-playing with standard connectives

- We've had this bit in the definition of *wlp* for conditionals:

$$wlp(\texttt{IF}\ b\ \texttt{THEN}\ c_1\ \texttt{ELSE}\ c_2, B) :=$$
$$(b \wedge wlp(c_1, B)) \vee (\neg b \wedge wlp(c_2, B))$$

- We've also defined the sequence for while as follows

$$A_0^{b,B,c} := \top$$
$$A_{n+1}^{b,B,c} := (b \wedge wlp(c, A_n^{b,B,c})) \vee (\neg b \wedge B)$$

- Notice a pattern here?
- Exercise: Show that

$$(b \wedge A) \vee (\neg b \wedge C) \equiv (b \to A) \wedge (\neg b \to C)$$

- It is crucial that $\to$ is a (definable) implication well-playing with standard connectives
- But now we have $*$, which is a different connective and a different conjunction than those we know from GLoIn

- One way of seeing this is that $\wedge$ and $\to$ are natural partners:

  $A \wedge B$ provably implies C iff A provably implies $B \to C$

  Mathematicians speak of *adjoint pairs*
- Note that in particular we have $\vDash (A \wedge (A \to B)) \to B$
- Is there a partner connective like this for $*$?
- Yes, and we did mention it: it's the magic wand or separating implication $-\!\!*$

The magic wand $\mathrel{-\!\!*}$

- $\sigma, \mathfrak{h} \vDash^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t.

The magic wand $\mathrel{-\!\!*}$

- $\sigma, \mathfrak{h} \vDash^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t.
  - $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$

The magic wand $\mathrel{-\!\!*}$

- $\sigma, \mathfrak{h} \vDash^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t.
  - $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$
  - $\sigma, \mathfrak{h}' \vDash^I B_1$

The magic wand $\mathrel{-\!\!*}$

- $\sigma, \mathfrak{h} \vDash^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t.
  - $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$
  - $\sigma, \mathfrak{h}' \vDash^I B_1$

The magic wand $\mathrel{-\!\!*}$

- $\sigma, \mathfrak{h} \vDash^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t.
  - $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$
  - $\sigma, \mathfrak{h}' \vDash^I B_1$

  we have that $\sigma, \mathfrak{h} \cup \mathfrak{h}' \vDash^I B_2$
- Note that $\vDash (A * (A \mathrel{-\!\!*} B)) \to B$

The magic wand $\mathrel{-\!\!*}$

- $\sigma, \mathfrak{h} \models^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t.
  - $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$
  - $\sigma, \mathfrak{h}' \models^I B_1$

  we have that $\sigma, \mathfrak{h} \cup \mathfrak{h}' \models^I B_2$
- Note that $\models (A * (A \mathrel{-\!\!*} B)) \to B$
- The class of assertions of *(infinitary) separation logic with* $\mathrel{-\!\!*}$ (dynamic Hoare logic) SepAss is defined as

$$B_1, B_2 ::= \top \mid a_1 \texttt{ == } a_2 \mid a_1 \texttt{ < } a_2 \mid \neg B_1 \mid \bigwedge_{z \in \mathbb{Z}} B_z \mid$$

$$\texttt{emp} \mid a_1 \texttt{ :-> } a_2 \mid B_1 \texttt{ * } B_2 \mid B_1 \mathrel{-\!\!*} B_2 .$$

Benefits of not waving the magic wand around too often?

- Not present in the formalism of (Implicit) Dynamic Frames and frameworks based on IDF, in particular VeriFast

  developed (well!) at KU Leuven, often used in our tutorials

Benefits of not waving the magic wand around too often?

- Not present in the formalism of (Implicit) Dynamic Frames and frameworks based on IDF, in particular VeriFast

  developed (well!) at KU Leuven, often used in our tutorials
- Problematic for model checking: quantification over all possible extensions of heap(let)

Benefits of not waving the magic wand around too often?

- Not present in the formalism of (Implicit) Dynamic Frames and frameworks based on IDF, in particular VeriFast

  developed (well!) at KU Leuven, often used in our tutorials
- Problematic for model checking: quantification over all possible extensions of heap(let)
- A line of work on adjunction elimination

  Calcagno et al., *Adjunct elimination in Context Logic for trees*, Information and Computation 2010; Dawar et al., *Adjunct Elimination Through Games in Static Ambient Logic*, FSTTCS 2004; Lozes, *Adjuncts elimination in the static ambient logic*, EXPRESS 2003

Benefits of not waving the magic wand around too often?

- Not present in the formalism of (Implicit) Dynamic Frames and frameworks based on IDF, in particular VeriFast

  developed (well!) at KU Leuven, often used in our tutorials
- Problematic for model checking: quantification over all possible extensions of heap(let)
- A line of work on adjunction elimination

  Calcagno et al., *Adjunct elimination in Context Logic for trees*, Information and Computation 2010; Dawar et al., *Adjunct Elimination Through Games in Static Ambient Logic*, FSTTCS 2004; Lozes, *Adjuncts elimination in the static ambient logic*, EXPRESS 2003
- But used in framework based on proof assistants!

Benefits of not waving the magic wand around too often?

- Not present in the formalism of (Implicit) Dynamic Frames and frameworks based on IDF, in particular VeriFast

  developed (well!) at KU Leuven, often used in our tutorials
- Problematic for model checking: quantification over all possible extensions of heap(let)
- A line of work on adjunction elimination

  Calcagno et al., *Adjunct elimination in Context Logic for trees*, Information and Computation 2010; Dawar et al., *Adjunct Elimination Through Games in Static Ambient Logic*, FSTTCS 2004; Lozes, *Adjuncts elimination in the static ambient logic*, EXPRESS 2003
- But used in framework based on proof assistants!
  - Iris and its derivatives (Iron, RustBelt)

    Aarhus U., MPI-SWS, TU Delft, KU Leuven ...

Benefits of not waving the magic wand around too often?

- Not present in the formalism of (Implicit) Dynamic Frames and frameworks based on IDF, in particular VeriFast

  developed (well!) at KU Leuven, often used in our tutorials
- Problematic for model checking: quantification over all possible extensions of heap(let)
- A line of work on adjunction elimination

  Calcagno et al., *Adjunct elimination in Context Logic for trees*, Information and Computation 2010; Dawar et al., *Adjunct Elimination Through Games in Static Ambient Logic*, FSTTCS 2004; Lozes, *Adjuncts elimination in the static ambient logic*, EXPRESS 2003
- But used in framework based on proof assistants!
  - Iris and its derivatives (Iron, RustBelt)

    Aarhus U., MPI-SWS, TU Delft, KU Leuven ...
  - Verified Software Toolchain and its derivatives (CompCert)

    mostly Princeton, also INRIA, Galois.com ...

Mutation

$$\vdash \{(a :\!\!\rightarrow \_) * B\}\ [a] := a'\ \{(a :\!\!\rightarrow a') * B\}$$

Mutation

$$\vdash \{(a :\to \_) * B\}\ [a] := a'\ \{(a :\to a') * B\}$$

$$\frac{\vdash \{(a :\to \_) * ((a :\to a') \mathrel{-\!\!*} A)\}\ [a] := a'\ \{(a :\to a') * ((a :\to a') \mathrel{-\!\!*} A)\} \qquad \vDash (\alpha * (\alpha \mathrel{-\!\!*} \beta)) \to \beta}{\vdash \{(a :\to \_) * ((a :\to a') \mathrel{-\!\!*} A)\}\ [a] := a'\ \{A\}\,.}$$

Allocation: basic facts about quantifies and $-\!\!*$

$$\vdash \{(\mathtt{X} = i) \wedge A\}\ \mathtt{X} := \mathtt{CONS}\ \overline{a}\ \{(X :\!\!\rightarrow \overline{a}[i/\mathtt{X}]) * A[i/\mathtt{X}]\}$$

Allocation: basic facts about quantifies and $-\!\!*$

$$\vdash \{(\mathsf{X} = i) \wedge A\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{(X :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * A[i/\mathsf{X}]\}$$

$$\frac{\begin{array}{c}\vdash \{(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{(X :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}]\} \\ \vDash \beta \rightarrow \exists i'.(\mathsf{X} = i') \wedge \beta[i'/\mathsf{X}]\end{array}}{\vdash \{(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\}}$$

Allocation: basic facts about quantifies and $-\!\!*$

$$\vdash \{(\mathsf{X} = i) \wedge A\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{(X :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * A[i/\mathsf{X}])\}$$

$$\frac{\begin{array}{c}\vdash \{(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{(X :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\} \\ \vDash \beta \rightarrow \exists i'.(\mathsf{X} = i') \wedge \beta[i'/\mathsf{X}]\end{array}}{\begin{array}{c}\vdash \{(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\} \\ \vDash (\forall i'.\beta) \rightarrow \beta\end{array}}$$

$$\vdash \{\forall i'.(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\}$$

Allocation: basic facts about quantifies and $\twoheadrightarrow$

$$\vdash \{(\mathsf{X} = i) \wedge A\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{(X :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * A[i/\mathsf{X}]\}$$

$$\text{SUBST}\ \frac{\begin{array}{c}\vdash \{\forall i'.(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\} \\ \models (\forall i'.\beta) \to \beta \end{array}}{\vdash \{\forall i'.(\mathsf{X} = i) \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) \twoheadrightarrow A[i'/\mathsf{X}])\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) \twoheadrightarrow A[i'/\mathsf{X}]))\}}$$

$$\frac{\begin{array}{c}\vdash \{(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{(X :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}]\} \\ \models \beta \to \exists i'.(\mathsf{X} = i') \wedge \beta[i'/\mathsf{X}] \end{array}}{\begin{array}{c}\vdash \{(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\} \\ \models (\forall i'.\beta) \to \beta \end{array}}$$

Allocation: basic facts about quantifies and $\mathrel{-\!\!*}$

$$\vdash \{(\mathsf{X} = i) \wedge A\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{(X :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * A[i/\mathsf{X}])\}$$

$$\dfrac{\begin{array}{c}\vdash \{(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{(X :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\} \\ \vDash \beta \rightarrow \exists i'.(\mathsf{X} = i') \wedge \beta[i'/\mathsf{X}]\end{array}}{\dfrac{\begin{array}{c}\vdash \{(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\} \\ \vDash (\forall i'.\beta) \rightarrow \beta\end{array}}{\text{SUBST}\ \dfrac{\begin{array}{c}\vdash \{\forall i'.(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\} \\ \vDash \forall i'.\alpha \rightarrow \forall i'.((\mathsf{X} = i) \wedge \alpha[i/\mathsf{X}]) \qquad \vDash (\gamma \wedge \alpha * (\alpha \mathrel{-\!\!*} \beta)) \rightarrow \gamma \wedge \beta\end{array}}{\begin{array}{c}\vdash \{\forall i'.(\mathsf{X} = i) \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) \mathrel{-\!\!*} A[i'/\mathsf{X}])\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a} \\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) * ((i' :\!\!\rightarrow \overline{a}[i/\mathsf{X}]) \mathrel{-\!\!*} A[i'/\mathsf{X}]))\}\end{array}}}}$$

$$\vdash \{\forall i'.(i' :\!\!\rightarrow \overline{a}) \mathrel{-\!\!*} A[i'/\mathsf{X}])\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge A[i'/\mathsf{X}]\}$$

Allocation: basic facts about quantifies and $\mathrel{-\!\!*}$

$$\vdash \{(\mathsf{X} = i) \wedge A\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{(X :\!\!\to \overline{a}[i/\mathsf{X}]) * A[i/\mathsf{X}]\}$$

$$\text{SUBST}\ \frac{\begin{array}{c}\vdash \{\forall i'.(\mathsf{X} = i) \wedge \alpha\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\to \overline{a}[i/\mathsf{X}]) * \alpha[i/\mathsf{X}])\} \\ \models (\forall i'.\beta) \to \beta \end{array}}{\begin{array}{c}\vdash \{\forall i'.(\mathsf{X} = i) \wedge ((i' :\!\!\to \overline{a}[i/\mathsf{X}]) \mathrel{-\!\!*} A[i'/\mathsf{X}])\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a} \\ \{\exists i'.(\mathsf{X} = i') \wedge ((i' :\!\!\to \overline{a}[i/\mathsf{X}]) * ((i' :\!\!\to \overline{a}[i/\mathsf{X}]) \mathrel{-\!\!*} A[i'/\mathsf{X}]))\} \\ \models \forall i'.\alpha \to \forall i'.((\mathsf{X} = i) \wedge \alpha[i/\mathsf{X}]) \qquad \models (\gamma \wedge \alpha * (\alpha \mathrel{-\!\!*} \beta)) \to \gamma \wedge \beta \end{array}}$$

$$\frac{\begin{array}{c}\vdash \{\forall i'.(i' :\!\!\to \overline{a}) \mathrel{-\!\!*} A[i'/\mathsf{X}])\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{\exists i'.(\mathsf{X} = i') \wedge A[i'/\mathsf{X}]\} \\ \models (\exists i'.(\mathsf{X} = i') \wedge \alpha[i'/\mathsf{X}]) \to \alpha \end{array}}{\vdash \{\forall i'.(i' :\!\!\to \overline{a}) \mathrel{-\!\!*} A[i'/\mathsf{X}])\}\ \mathsf{X} := \mathsf{CONS}\,\overline{a}\ \{A\}}$$

For lookup we need something more:

- An observation regarding strictly exact assertions
- A missing rule regarding fresh variables in preconditions

Warm-up for strict exactness

- Recall: $\sigma, \mathfrak{h} \models^I B_1 * B_2$ if there exist $\mathfrak{h}_1, \mathfrak{h}_2$ s.t.
  - $\mathsf{dom}\mathfrak{h}_1 \cap \mathsf{dom}\mathfrak{h}_2 = \varnothing$
  - $\mathfrak{h}_1 \cup \mathfrak{h}_2 = \mathfrak{h}$
  - $\sigma, \mathfrak{h}_1 \models^I B_1$ and $\sigma, \mathfrak{h}_2 \models^I B_2$
- Recall: $\sigma, \mathfrak{h} \models^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t. $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$ and $\sigma, \mathfrak{h}' \models^I B_1$ we have that $\sigma, \mathfrak{h} \cup \mathfrak{h}' \models^I B_2$

Exercise:

- Show: $\models (A * (A \mathrel{-\!\!*} B)) \to B$

Warm-up for strict exactness

- Recall: $\sigma, \mathfrak{h} \models^I B_1 * B_2$ if there exist $\mathfrak{h}_1, \mathfrak{h}_2$ s.t.
  - $\mathsf{dom}\mathfrak{h}_1 \cap \mathsf{dom}\mathfrak{h}_2 = \varnothing$
  - $\mathfrak{h}_1 \cup \mathfrak{h}_2 = \mathfrak{h}$
  - $\sigma, \mathfrak{h}_1 \models^I B_1$ and $\sigma, \mathfrak{h}_2 \models^I B_2$
- Recall: $\sigma, \mathfrak{h} \models^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t. $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$ and $\sigma, \mathfrak{h}' \models^I B_1$ we have that $\sigma, \mathfrak{h} \cup \mathfrak{h}' \models^I B_2$

Exercise:

- Show: $\models (A * (A \mathrel{-\!\!*} B)) \to B$
- Deduce: $\models (A * (A \mathrel{-\!\!*} B)) \to (A * \texttt{true}) \wedge B$

Warm-up for strict exactness

- Recall: $\sigma, \mathfrak{h} \models^I B_1 * B_2$ if there exist $\mathfrak{h}_1, \mathfrak{h}_2$ s.t.
  - $\mathsf{dom}\mathfrak{h}_1 \cap \mathsf{dom}\mathfrak{h}_2 = \varnothing$
  - $\mathfrak{h}_1 \cup \mathfrak{h}_2 = \mathfrak{h}$
  - $\sigma, \mathfrak{h}_1 \models^I B_1$ and $\sigma, \mathfrak{h}_2 \models^I B_2$
- Recall: $\sigma, \mathfrak{h} \models^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t. $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$ and $\sigma, \mathfrak{h}' \models^I B_1$ we have that $\sigma, \mathfrak{h} \cup \mathfrak{h}' \models^I B_2$

Exercise:

- Show: $\models (A * (A \mathrel{-\!\!*} B)) \to B$
- Deduce: $\models (A * (A \mathrel{-\!\!*} B)) \to (A * \mathtt{true}) \wedge B$
- Disprove: $\models (A * \mathtt{true}) \wedge B \to (A * (A \mathrel{-\!\!*} B))$

Warm-up for strict exactness

- Recall: $\sigma, \mathfrak{h} \vDash^I B_1 * B_2$ if there exist $\mathfrak{h}_1, \mathfrak{h}_2$ s.t.
  - $\mathsf{dom}\mathfrak{h}_1 \cap \mathsf{dom}\mathfrak{h}_2 = \varnothing$
  - $\mathfrak{h}_1 \cup \mathfrak{h}_2 = \mathfrak{h}$
  - $\sigma, \mathfrak{h}_1 \vDash^I B_1$ and $\sigma, \mathfrak{h}_2 \vDash^I B_2$
- Recall: $\sigma, \mathfrak{h} \vDash^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t. $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$ and $\sigma, \mathfrak{h}' \vDash^I B_1$ we have that $\sigma, \mathfrak{h} \cup \mathfrak{h}' \vDash^I B_2$

Exercise:

- Show: $\vDash (A * (A \mathrel{-\!\!*} B)) \to B$
- Deduce: $\vDash (A * (A \mathrel{-\!\!*} B)) \to (A * \texttt{true}) \wedge B$
- Disprove: $\vDash (A * \texttt{true}) \wedge B \to (A * (A \mathrel{-\!\!*} B))$
- (Hint: $A := \neg\texttt{emp}, B := 0 :\!\to 0$)

Warm-up for strict exactness

- Recall: $\sigma, \mathfrak{h} \models^I B_1 * B_2$ if there exist $\mathfrak{h}_1, \mathfrak{h}_2$ s.t.
  - $\mathsf{dom}\mathfrak{h}_1 \cap \mathsf{dom}\mathfrak{h}_2 = \varnothing$
  - $\mathfrak{h}_1 \cup \mathfrak{h}_2 = \mathfrak{h}$
  - $\sigma, \mathfrak{h}_1 \models^I B_1$ and $\sigma, \mathfrak{h}_2 \models^I B_2$
- Recall: $\sigma, \mathfrak{h} \models^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t. $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$ and $\sigma, \mathfrak{h}' \models^I B_1$ we have that $\sigma, \mathfrak{h} \cup \mathfrak{h}' \models^I B_2$

Exercise:

- Show: $\models (A * (A \mathrel{-\!\!*} B)) \to B$
- Deduce: $\models (A * (A \mathrel{-\!\!*} B)) \to (A * \mathtt{true}) \wedge B$
- Disprove: $\models (A * \mathtt{true}) \wedge B \to (A * (A \mathrel{-\!\!*} B))$
- (Hint: $A := \neg\mathtt{emp}, B := 0 :\!\to 0$)
- While we're at it, disprove: $\models (A \wedge (A \mathrel{-\!\!*} B)) \to B$

Warm-up for strict exactness

- Recall: $\sigma, \mathfrak{h} \vDash^I B_1 * B_2$ if there exist $\mathfrak{h}_1, \mathfrak{h}_2$ s.t.
  - $\mathsf{dom}\mathfrak{h}_1 \cap \mathsf{dom}\mathfrak{h}_2 = \varnothing$
  - $\mathfrak{h}_1 \cup \mathfrak{h}_2 = \mathfrak{h}$
  - $\sigma, \mathfrak{h}_1 \vDash^I B_1$ and $\sigma, \mathfrak{h}_2 \vDash^I B_2$
- Recall: $\sigma, \mathfrak{h} \vDash^I B_1 \mathrel{-\!\!*} B_2$ if for all $\mathfrak{h}'$ s.t. $\mathsf{dom}\mathfrak{h} \cap \mathsf{dom}\mathfrak{h}' = \varnothing$ and $\sigma, \mathfrak{h}' \vDash^I B_1$ we have that $\sigma, \mathfrak{h} \cup \mathfrak{h}' \vDash^I B_2$

Exercise:

- Show: $\vDash (A * (A \mathrel{-\!\!*} B)) \to B$
- Deduce: $\vDash (A * (A \mathrel{-\!\!*} B)) \to (A * \mathtt{true}) \wedge B$
- Disprove: $\vDash (A * \mathtt{true}) \wedge B \to (A * (A \mathrel{-\!\!*} B))$
- (Hint: $A := \neg\mathtt{emp}, B := 0 :\!\to 0$)
- While we're at it, disprove: $\vDash (A \wedge (A \mathrel{-\!\!*} B)) \to B$
- (Hint: $A := 0 :\!\to 0$, $B := \mathtt{false}$)

- For some subclasses of SL expressions, we have more laws governing $-\!\!*$

- For some subclasses of SL expressions, we have more laws governing $-\!\!*$
- In particular, this applies to strictly exact assertions, isolated by Hongseok Yang

- For some subclasses of SL expressions, we have more laws governing $-\!\!*$
- In particular, this applies to strictly exact assertions, isolated by Hongseok Yang
- A is *strictly exact* iff for any σ, $\mathfrak{h}$ and $\mathfrak{h}'$ we have that $\mathfrak{h} = \mathfrak{h}'$ whenever $\sigma, \mathfrak{h} \models^I A$ and $\sigma, \mathfrak{h}' \models^I A$

- For some subclasses of SL expressions, we have more laws governing $-\!\!*$
- In particular, this applies to strictly exact assertions, isolated by Hongseok Yang
- A is *strictly exact* iff for any σ, $\mathfrak{h}$ and $\mathfrak{h}'$ we have that $\mathfrak{h} = \mathfrak{h}'$ whenever $\sigma, \mathfrak{h} \models^I A$ and $\sigma, \mathfrak{h}' \models^I A$

Fact
Assertions built from $:\!\!\rightarrow$ *-atoms using* $*$ *(i.e., descriptions of concrete finite heap(let)s!) are strictly exact.*

- For some subclasses of SL expressions, we have more laws governing $-\!\!*$
- In particular, this applies to strictly exact assertions, isolated by Hongseok Yang
- A is *strictly exact* iff for any σ, $\mathfrak{h}$ and $\mathfrak{h}'$ we have that $\mathfrak{h} = \mathfrak{h}'$ whenever $\sigma, \mathfrak{h} \models^I A$ and $\sigma, \mathfrak{h}' \models^I A$

Fact
Assertions built from $:\!\!\rightarrow$ *-atoms using* $*$ *(i.e., descriptions of concrete finite heap(let)s!) are strictly exact.*

- The importance of this concept for us:

- For some subclasses of SL expressions, we have more laws governing $\mathrel{-\!\!*}$
- In particular, this applies to strictly exact assertions, isolated by Hongseok Yang
- A is *strictly exact* iff for any σ, $\mathfrak{h}$ and $\mathfrak{h}'$ we have that $\mathfrak{h} = \mathfrak{h}'$ whenever $\sigma, \mathfrak{h} \models^I A$ and $\sigma, \mathfrak{h}' \models^I A$

Fact
Assertions built from $:\!\!\rightarrow$ *-atoms using* $*$ *(i.e., descriptions of concrete finite heap(let)s!) are strictly exact.*

- The importance of this concept for us:

Fact
Whenever A *is strictly exact and* B *is any assertion,*

$$\models (A * \mathtt{true}) \wedge B \rightarrow (A * (A \mathrel{-\!\!*} B))$$

- For some subclasses of SL expressions, we have more laws governing $-\!\!*$
- In particular, this applies to strictly exact assertions, isolated by Hongseok Yang
- A is *strictly exact* iff for any σ, $\mathfrak{h}$ and $\mathfrak{h}'$ we have that $\mathfrak{h} = \mathfrak{h}'$ whenever $\sigma, \mathfrak{h} \models^I A$ and $\sigma, \mathfrak{h}' \models^I A$

Fact
Assertions built from $:\!\!\rightarrow$ *-atoms using* $*$ *(i.e., descriptions of concrete finite heap(let)s!) are strictly exact.*

- The importance of this concept for us:

Fact
Whenever A *is strictly exact and* B *is any assertion,*

$$\models (A * \texttt{true}) \wedge B \rightarrow (A * (A -\!\!* B))$$

- Exercise: prove it!

- For some subclasses of SL expressions, we have more laws governing $-\!\!*$
- In particular, this applies to strictly exact assertions, isolated by Hongseok Yang
- A is *strictly exact* iff for any σ, $\mathfrak{h}$ and $\mathfrak{h}'$ we have that $\mathfrak{h} = \mathfrak{h}'$ whenever $\sigma, \mathfrak{h} \models^I A$ and $\sigma, \mathfrak{h}' \models^I A$

Fact
Assertions built from $:\!\!\rightarrow$ *-atoms using* $*$ *(i.e., descriptions of concrete finite heap(let)s!) are strictly exact.*

- The importance of this concept for us:

Fact
Whenever A *is strictly exact and* B *is any assertion,*

$$\models (A * \texttt{true}) \wedge B \rightarrow (A * (A -\!\!* B))$$

- Exercise: prove it!
- Exercise: $\neg\texttt{emp}$ obviously fails to be strictly exact.

Fresh variables in preconditions

- Logical and arithmetical reasoning is entirely delegated to the consequence rule

Fresh variables in preconditions

- Logical and arithmetical reasoning is entirely delegated to the consequence rule
- There is however a rule corresponding to a logical rule for quantifiers which needs to be stated separately:

$$\frac{\vdash \{A\}\ c\ \{B\} \qquad i \in \mathit{fresh}(B)}{\vdash \{A[a/i]\}\ c\ \{B\}}\ \textsc{VPrEl}$$

Fresh variables in preconditions

- Logical and arithmetical reasoning is entirely delegated to the consequence rule
- There is however a rule corresponding to a logical rule for quantifiers which needs to be stated separately:

$$\frac{\vdash \{A\}\ c\ \{B\} \qquad i \in \mathit{fresh}(B)}{\vdash \{A[a/i]\}\ c\ \{B\}}\ \textsc{VPrEl}$$

- Exercise: derive from this

$$\frac{\vdash \{\mathsf{X} = i \wedge A\}\ c\ \{B\} \qquad i \in \mathit{fresh}(B) \cap \mathit{fresh}(A)}{\vdash \{A\}\ c\ \{B\}}\ \textsc{VPrEl'}$$

Fresh variables in preconditions

- Logical and arithmetical reasoning is entirely delegated to the consequence rule
- There is however a rule corresponding to a logical rule for quantifiers which needs to be stated separately:

$$\frac{\vdash \{A\}\ c\ \{B\} \qquad i \in \mathit{fresh}(B)}{\vdash \{A[a/i]\}\ c\ \{B\}}\ \textsc{VPrEl}$$

- Exercise: derive from this

$$\frac{\vdash \{\mathsf{X} = i \wedge A\}\ c\ \{B\} \qquad i \in \mathit{fresh}(B) \cap \mathit{fresh}(A)}{\vdash \{A\}\ c\ \{B\}}\ \textsc{VPrEl'}$$

- If $\exists$ primitive, take still stronger (why?)

$$\frac{\vdash \{A\}\ c\ \{B\} \qquad i \in \mathit{fresh}(B)}{\vdash \{\exists i.A\}\ c\ \{B\}}\ \textsc{VPrEl}\exists$$

- Aside: is VPrEl$\exists$ derivable from VPrEl?

- Aside: is VPrEl$\exists$ derivable from VPrEl?
- Recall we encoded $\exists i.A$ as $\bigvee_{z \in \mathbb{Z}} A[\mathsf{z}/i]$

- Aside: is VPrEl$\exists$ derivable from VPrEl?
- Recall we encoded $\exists i.A$ as $\bigvee_{z \in \mathbb{Z}} A[\mathsf{z}/i]$
- For this, we need to directly postulate

$$\frac{\vdash \{A_i\}\, c\, \{B\} \qquad \text{for all } i \in \mathbb{Z}}{\vdash \{\bigvee_{i \in \mathbb{Z}} A_i\}\, c\, \{B\}}$$

- Aside: is VPrEl$\exists$ derivable from VPrEl?
- Recall we encoded $\exists i.A$ as $\bigvee_{z \in \mathbb{Z}} A[\mathsf{z}/i]$
- For this, we need to directly postulate

$$\frac{\vdash \{A_i\}\, c\, \{B\} \qquad \text{for all } i \in \mathbb{Z}}{\vdash \{\bigvee_{i \in \mathbb{Z}} A_i\}\, c\, \{B\}}$$

- O'Hearn, Yang and Reynolds: if the disjunction rule not postulated explicitly, even its finitary variant may happen to be admissible without being derivable

Dually, in postconditions ...

$$\frac{\vdash \{A\}\ c\ \{B\} \qquad i \in \mathit{fresh}(A)}{\vdash \{A\}\ c\ \{B[a/i]\}}\ \textsc{VPoEl}$$

$$\frac{\vdash \{A\}\ c\ \{\mathsf{X} = i \to B\} \qquad i \in \mathit{fresh}(A) \cap \mathit{fresh}(B)}{\vdash \{A\}\ c\ \{B\}}\ \textsc{VPoEl'}$$

$$\frac{\vdash \{A\}\ c\ \{B\} \qquad i \in \mathit{fresh}(A)}{\vdash \{A\}\ c\ \{\forall i.B\}}\ \textsc{VPoEl}\forall$$

$$\frac{\vdash \{A\}\ c\{B_i\} \qquad \text{for all } i \in \mathbb{Z}}{\vdash \{A\}\ c\{\bigwedge_{i\in\mathbb{Z}} B_i\}}$$

Lookup

$\vdash \{((\mathsf{X} = i) \wedge (a :\!\!\rightarrow i')) * A'[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{(a[i/\mathsf{X}] :\!\!\rightarrow i') * A'\}$

Pick i, i' fresh for A:

Lookup

$$\vdash \{((\mathsf{X} = i) \wedge (a :\!\rightarrow i'))\ast A'[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{(a[i/\mathsf{X}] :\!\rightarrow i')\ast A'\}$$

Pick i, i' fresh for A:

$$\frac{\begin{array}{c}\vdash \{((\mathsf{X} = i) \wedge (a :\!\rightarrow i'))\ast((a[i/\mathsf{X}] :\!\rightarrow i')\mathrel{-\!\!\ast} A[i'/\mathsf{X}])\}\\ \mathsf{X} := [a]\ \{(a[i/\mathsf{X}] :\!\rightarrow i')\ast((a[i/\mathsf{X}] :\!\rightarrow i')\mathrel{-\!\!\ast} A)\}\end{array}\qquad \begin{array}{l}\vDash (\mathsf{X} = i) \wedge (\beta\ast\alpha) \to ((\mathsf{X} = i) \wedge \beta)\ast\alpha[i/\mathsf{X}]\\ \vDash (\alpha'\ast(\alpha'\mathrel{-\!\!\ast}\beta')) \to \beta'\end{array}}{\vdash \{(\mathsf{X} = i) \wedge ((a :\!\rightarrow i')\ast((a :\!\rightarrow i')\mathrel{-\!\!\ast} A[i'/\mathsf{X}]))\}\ \mathsf{X} := [a]\ \{A\}}$$

Lookup

$$\vdash \{((\mathsf{X} = i) \wedge (a :\!\!\rightarrow i'))\ast A'[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{(a[i/\mathsf{X}] :\!\!\rightarrow i')\ast A'\}$$

Pick i, i' fresh for A:

$$\textsc{VPrEl'}\ \frac{\begin{array}{c}\vdash \{((\mathsf{X} = i) \wedge (a :\!\!\rightarrow i'))\ast((a[i/\mathsf{X}] :\!\!\rightarrow i') \mathrel{-\!\!\ast} A[i'/\mathsf{X}])\}\\ \mathsf{X} := [a]\ \{(a[i/\mathsf{X}] :\!\!\rightarrow i')\ast((a[i/\mathsf{X}] :\!\!\rightarrow i') \mathrel{-\!\!\ast} A)\}\\ \vDash (\mathsf{X} = i) \wedge (\beta\ast\alpha) \rightarrow ((\mathsf{X} = i) \wedge \beta)\ast\alpha[i/\mathsf{X}] \qquad \vDash (\alpha'\ast(\alpha' \mathrel{-\!\!\ast} \beta')) \rightarrow \beta'\end{array}}{\begin{array}{c}\vdash \{(\mathsf{X} = i) \wedge ((a :\!\!\rightarrow i')\ast((a :\!\!\rightarrow i') \mathrel{-\!\!\ast} A[i'/\mathsf{X}]))\}\ \mathsf{X} := [a]\ \{A\}\\ \hline \vdash \{(a :\!\!\rightarrow i')\ast((a :\!\!\rightarrow i') \mathrel{-\!\!\ast} A[i'/\mathsf{X}])\}\ \mathsf{X} := [a]\ \{A\}\end{array}}$$

Lookup

$$\vdash \{((\mathsf{X} = i) \wedge (a :\!\rightarrow i')) * A'[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{(a[i/\mathsf{X}] :\!\rightarrow i') * A'\}$$

Pick i, i' fresh for A:

$$\textsc{VPrEl'}\ \frac{\begin{array}{c}\vdash \{((\mathsf{X} = i) \wedge (a :\!\rightarrow i')) * ((a[i/\mathsf{X}] :\!\rightarrow i') \mathrel{-\!\!*} A[i'/\mathsf{X}])\}\ \mathsf{X} := [a]\ \{(a[i/\mathsf{X}] :\!\rightarrow i') * ((a[i/\mathsf{X}] :\!\rightarrow i') \mathrel{-\!\!*} A)\} \\ \vDash (\mathsf{X} = i) \wedge (\beta * \alpha) \rightarrow ((\mathsf{X} = i) \wedge \beta) * \alpha[i/\mathsf{X}] \qquad \vDash (\alpha' * (\alpha' \mathrel{-\!\!*} \beta')) \rightarrow \beta'\end{array}}{\vdash \{(\mathsf{X} = i) \wedge ((a :\!\rightarrow i') * ((a :\!\rightarrow i') \mathrel{-\!\!*} A[i'/\mathsf{X}]))\}\ \mathsf{X} := [a]\ \{A\}}$$

$$(*)\ \frac{\vDash (((a :\!\rightarrow i') * \mathtt{true}) \wedge A[i'/\mathsf{X}]) \rightarrow (a :\!\rightarrow i') * ((a :\!\rightarrow i') \mathrel{-\!\!*} A[i'/\mathsf{X}])}{\vdash \{((a :\!\rightarrow i') * \mathtt{true}) \wedge A[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{A\}}$$

The (*) step because of strict exactness!

Lookup

$$\vdash \{((\mathsf{X} = i) \wedge (a :\!\rightarrow i')) * A'[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{(a[i/\mathsf{X}] :\!\rightarrow i') * A'\}$$

Pick i, i' fresh for A:

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\vdash \{((\mathsf{X} = i) \wedge (a :\!\rightarrow i')) * ((a[i/\mathsf{X}] :\!\rightarrow i') \mathrel{-\!\!*} A[i'/\mathsf{X}])\}\\ \mathsf{X} := [a]\ \{(a[i/\mathsf{X}] :\!\rightarrow i') * ((a[i/\mathsf{X}] :\!\rightarrow i') \mathrel{-\!\!*} A)\}\end{array}}{\vdash \{(\mathsf{X} = i) \wedge ((a :\!\rightarrow i') * ((a :\!\rightarrow i') \mathrel{-\!\!*} A[i'/\mathsf{X}]))\}\ \mathsf{X} := [a]\ \{A\}}\ \text{VPREL'}}{\vdash \{(a :\!\rightarrow i') * ((a :\!\rightarrow i') \mathrel{-\!\!*} A[i'/\mathsf{X}])\}\ \mathsf{X} := [a]\ \{A\}}}{\vdash \{((a :\!\rightarrow i') * \mathtt{true}) \wedge A[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{A\}}\ (*)}{\vdash \{\exists i'.((a :\!\rightarrow i') * \mathtt{true}) \wedge A[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{A\}}\ \text{VPREL}\exists$$

$$\models (\mathsf{X} = i) \wedge (\beta * \alpha) \rightarrow ((\mathsf{X} = i) \wedge \beta) * \alpha[i/\mathsf{X}] \qquad \models (\alpha' * (\alpha' \mathrel{-\!\!*} \beta')) \rightarrow \beta'$$

$$\models (((a :\!\rightarrow i') * \mathtt{true}) \wedge A[i'/\mathsf{X}]) \rightarrow (a :\!\rightarrow i') * ((a :\!\rightarrow i') \mathrel{-\!\!*} A[i'/\mathsf{X}])$$

The (*) step because of strict exactness!

- Let us take stock ...

- Let us take stock ...
- Allocation:

$$\vdash \{\forall i'.(i' :\!\!\rightarrow \overline{a}) \mathrel{-\!\!*} A[i'/\mathsf{X}]\}\ \mathsf{X} := \mathtt{CONS}\,\overline{a}\ \{A\}$$

- Let us take stock ...
- Allocation:

$$\vdash \{\forall i'.(i' :\!\!\rightarrow \overline{a}) \mathrel{-\!\!*} A[i'/\mathtt{X}]\}\ \mathtt{X} := \mathtt{CONS}\ \overline{a}\ \{A\}$$

- Lookup:

$\vdash \{\exists i'.((a :\!\!\rightarrow i') * \mathtt{true}) \wedge A[i'/\mathtt{X}]\}\ \mathtt{X} := [a]\ \{A\}$

(i' fresh for A)

- Let us take stock ...
- Allocation:

$$\vdash \{\forall i'.(i' :\!\!\rightarrow \overline{a}) \mathrel{-\!\!*} A[i'/\mathtt{X}]\}\ \mathtt{X} := \mathtt{CONS}\ \overline{a}\ \{A\}$$

- Lookup:

$$\vdash \{\exists i'.((a :\!\!\rightarrow i') * \mathtt{true}) \wedge A[i'/\mathtt{X}]\}\ \mathtt{X} := [a]\ \{A\}$$

(i' fresh for A)

- Mutation:

$$\vdash \{(a :\!\!\rightarrow \_) * ((a :\!\!\rightarrow a') \mathrel{-\!\!*} A)\}\ [a] := a'\ \{A\}$$

- Let us take stock ...
- Allocation:

$$\vdash \{\forall i'.(i' :\!\!\rightarrow \overline{a}) \mathrel{-\!\!*} A[i'/\mathsf{X}]\}\ \mathsf{X} := \mathtt{CONS}\ \overline{a}\ \{A\}$$

- Lookup:

$$\vdash \{\exists i'.((a :\!\!\rightarrow i') * \mathtt{true}) \wedge A[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{A\}$$

(i' fresh for A)

- Mutation:

$$\vdash \{(a :\!\!\rightarrow \_) * ((a :\!\!\rightarrow a') \mathrel{-\!\!*} A)\}\ [a] := a'\ \{A\}$$

- Deallocation:

$$\vdash \{(a :\!\!\rightarrow \_) * A\}\ \mathtt{DISPOSE}\ a\ \{A\}$$

- Let us take stock ...
- Allocation:

$$\vdash \{\forall i'.(i' :\!\!\rightarrow \overline{a}) \mathrel{-\!\!*} A[i'/\mathsf{X}]\}\ \mathsf{X} := \mathtt{CONS}\ \overline{a}\ \{A\}$$

- Lookup:

$$\vdash \{\exists i'.((a :\!\!\rightarrow i') * \mathtt{true}) \wedge A[i'/\mathsf{X}]\}\ \mathsf{X} := [a]\ \{A\}$$

(i' fresh for A)

- Mutation:

$$\vdash \{(a :\!\!\rightarrow \_) * ((a :\!\!\rightarrow a') \mathrel{-\!\!*} A)\}\ [a] := a'\ \{A\}$$

- Deallocation:

$$\vdash \{(a :\!\!\rightarrow \_) * A\}\ \mathtt{DISPOSE}\ a\ \{A\}$$

- Just like in the case of ordinary IMP, we can compute weakest (liberal) preconditions