LECTURE 3:

LOGICS FOR CORRECTNESS ASSERTIONS

Concrete atomic actions: assignments

- Recall Environments (assignments)]
  Take vocab $V = (\mathbf{0}; \mathbf{s}, +, \times; <)$

  - "state" $=$ environment (mapping Variables $\mapsto \mathbb{N}$)
  - "action" $=$ update $\vec{x} := \vec{\mathbf{t}}$

- Given vocab V and V-structure $\mathcal{S}$,
  the assignment PD structure over $\mathcal{S}$, $\mathcal{K}(\mathcal{S})$:
- States $=$ environments over $\mathcal{S}$, i.e. functions $\eta : Vrb \rightarrow |\mathcal{S}|$
- Actions $=$ assignments over V: $\vec{x} := \vec{\mathbf{t}}$, ($\vec{\mathbf{t}}$ V-terms.
- Propositional identifiers $=$ quntifier-free V-formulas.
- $\xrightarrow{\mathbf{x} := \mathbf{t}}$ is the function that maps η to $[a/\mathbf{x}]\eta$, where $a = [\![\mathbf{t}]\!]_{\mathcal{S},\eta}$.

Programs over vocabulary V

- Regular programs over V:
  regular programs with
  - V-assignments as atomic action-identifiers
  - quantifier-free V-formulas as tests
- In BNF format:

$$
\begin{array}{rcll}
\mathit{Vrb} & \ni & \mathbf{x} & \\
\mathit{Trm}(V) & \ni & \mathbf{t} & \\
\mathit{QFFml}(V) & \ni & \boldsymbol{\chi} & \quad \textbf{(the qf } V\textbf{-formulas)} \\
\mathit{RPrg}(V) & \ni & \boldsymbol{\alpha}, \boldsymbol{\beta} & ::= \ (\vec{x} := \vec{\mathbf{t}}) \ \mid \ (\boldsymbol{\alpha}; \boldsymbol{\beta}) \ \mid \ (\boldsymbol{\alpha} \cup \boldsymbol{\beta}) \ \mid \ \boldsymbol{\alpha}^* \ \mid \ ?\boldsymbol{\chi}
\end{array}
$$

- Guarded Iterative Programs over V:
  - action-identifiers = V-assignments
  - guards = quantifier-free V-formulas
- Semantics: For a fixed V-structure $\mathcal{S}$,
  programs define transitions between environments: $\eta \xrightarrow{\alpha,\mathcal{S}} \eta'$.
- Note: Transitions depend on structure, since value of terms does.

First Order Dynamic Logic (Flat Version)

- Fix vocab V
- Programs over V (regular-program or GIP version).
  (No "rich tests").
- Formulas generated using quantifiers as well as $[\alpha]$, for α a program over V).
- Semantics:

$$\mathcal{S}, \eta \models [\alpha]\varphi \qquad \equiv_{\text{df}} \qquad \mathcal{S}, \eta' \models \varphi \quad \textbf{whenever } \eta \xrightarrow{\alpha,\mathcal{S}} \eta'$$

Partial and total correctness assertions

- Partial-correctness assertion (PCA):
  DL formula of the form $\psi \to [\alpha]\varphi$
  with φ, ψ first order
- Abbreviation: $\psi[\alpha]\varphi$
- The premise ψ = the pre-condition
- φ = the post-condition
- Example: Take $\mathcal{S} = (\mathbb{Z}, \mathbf{0}, \mathbf{s})$,
  $\alpha = \mathtt{while}\ x \neq \mathbf{0}\ \mathtt{do}\ x := \mathbf{s}(x)$
  $\eta \models \top[\alpha]\bot$ iff α runs indefinitely
  i.e. iff $\eta(x) \geq 0$

Total correctness assertions

- Partial-correctness assertion (PCA):
  DL formula of the form $\psi \rightarrow \langle \alpha \rangle \varphi$
  with φ, ψ first order
- Abbreviation: $\psi \langle \alpha \rangle \varphi$
- Pre-condition, post-condition.
- Expresible as the negation of a PCA, but not as a PCA.
  (Box and diamond have the same polarity.)
- If $\mathcal{S} \models \top \langle \alpha \rangle \varphi$ then α terminates for all input.
  Hence "total".

Hoare logics: deductive calculi for PCAs

- Not proving PCAs within dynamic logic,
  but within a calculus that refers only to PCAs.
- Desirable trait, but artificial for broader programming languages.
- Has plagued development of appropriate logics for complex imperative languages.

A Hoare Logic for regular programs

- Given vocab V.
  V-theory Γ (axioms are closed FO formulas).
- Define calculus $H_{reg}(\mathbf{\Gamma})$:

Assignment $\quad \{\mathbf{t}/\mathbf{x}\}\varphi \;\; [\mathbf{x} := \mathbf{t}] \;\; \varphi$

Composition $$\frac{\psi\ [\alpha]\ \chi \qquad \chi\ [\beta]\ \varphi}{\psi\ [\alpha;\beta]\ \varphi}$$

(χ **is the** *intermediate assertion*)

Branching $$\frac{\psi\ [\alpha]\ \varphi \qquad \psi\ [\beta]\ \varphi}{\psi\ [\alpha \cup \beta]\ \varphi}$$

Iteration $$\frac{\varphi\ [\alpha]\ \varphi}{\varphi\ [\alpha^*]\ \varphi}$$

Test $$\frac{\chi \rightarrow \varphi}{\top [?\chi] \varphi} \qquad \chi \text{ quantifier-free}$$

Logic rules

- Merging PCS'a with the background theory Γ:
-

$$\textbf{Pre-consequence} \qquad \frac{\psi\ [\alpha]\ \varphi \qquad \Gamma \vdash \psi' \to \psi}{\psi'\ [\alpha]\ \varphi}$$

$$\textbf{Post-consequence} \qquad \frac{\psi\ [\alpha]\ \varphi \qquad \Gamma \vdash \varphi \to \varphi'}{\psi\ [\alpha]\ \varphi'}$$

- Combined Consequence Rule:

$$\frac{\Gamma \vdash \varphi \to \varphi' \qquad \psi\ [\alpha]\ \varphi \qquad \Gamma \vdash \psi' \to \psi}{\psi'\ [\alpha]\ \varphi'}$$

- This works "accidentally".
  For more complex PLs need additional rules, e.g.

$$\frac{\psi\ [\alpha]\ \varphi_1 \qquad \psi\ [\alpha]\ \varphi_2}{\psi\ [\alpha]\ \varphi_1 \wedge \varphi_2}$$

A Hoare Logic for guarded iterative programs

- Define $H_{guard}(\mathbf{\Gamma})$
- Same rules for Assignment, Composition, and Consequence.

- Rules for guarded constructs:

Skip	$\varphi\ [\mathtt{skip}]\ \varphi$
Abort	$\varphi\ [\mathtt{abort}]\ \bot$
Cases	$\dfrac{(\psi \wedge \chi)\ [\alpha]\ \varphi \qquad (\psi \wedge \neg\chi)\ [\beta]\ \varphi}{\psi\ [\ \mathtt{if}\ \chi\ \mathtt{then}\ \alpha\ \mathtt{else}\ \beta]\ \varphi}$
Iteration	$\dfrac{(\varphi \wedge \chi)\ [\alpha]\ \varphi}{\varphi\ [\mathtt{while}\ \chi\ \ \mathtt{do}\ \ \alpha]\ (\varphi \wedge \neg\chi)}$

- Notations: $\mathbf{\Gamma} \vdash_H \psi\ [\alpha]\ \varphi$, and similarly for GIPs.

Trivial examples of proofs

- $t = \mathbf{0}\ \ [x := t]\ \ x = \mathbf{0}$
- $x+1 = a\ \ [x := x+1]\ \ x = a$
- $$\frac{x = a \wedge y = b\ \ [x := y] x = b \wedge y = b \qquad x = b \wedge y = b\ \ [y := x]\ \ x = b \wedge y = b}{x = a \wedge y = b\ \ [x := y;\ y := x]\ \ x = b \wedge y = b}$$
- $x = a \wedge y = b\ \ [x, y := y, x]\ \ x = b \wedge y = a$
- $$\frac{\Gamma \vdash (x \geq 0) \rightarrow (x+1 > 0) \qquad x+1 > 0\ \ [x := x+1]\ \ x = a}{x \geq 0\ \ [x := x+1]\ \ x > 0}$$

  Here we stipulate that Γ proves the needed implication.

Example for branching

$$\frac{\mathcal{D}_0 \qquad\qquad \mathcal{D}_1}{(x = a \land y = b)\ \ [\texttt{if}\ \ x > y\ \ \texttt{then}\ \ z := x\ \ \texttt{else}\ \ z := y]\ \ z = \max(a, b)}$$

where $\mathcal{D}_0$ is

$$\frac{\Gamma \vdash (x = a \land y = b \land x > y) \to x = \max(a, b) \qquad\qquad x = \max(a, b)[z := x]\ \ z = \max(a, b)}{(x = a \land y = b \land x > y)\ \ [z := x]\ \ z = \max(a, b)}$$

and $\mathcal{D}_1$ is

$$\frac{\Gamma \vdash (x = a \land y = b \land \neg(x > y)) \to y = \max(a, b) \qquad\qquad y = \max(a, b)[z := y]\ \ z = \max(a, b)}{(x = a \land y = b \land \neg(x > y))\ \ [z := y]\ \ z = \max(a, b)}$$

Example for Iteration

$$\dfrac{\dfrac{\vdash \top \rightarrow 0+n\cdot a = n\cdot a \qquad (0+n\cdot a = n\cdot a)[x,k:=0,n]\ \ (x+k\cdot a = n\cdot a)}{\top\ \ [x,k:=0,n]\ \ x+k\cdot a = n\cdot a} \qquad \mathcal{D}}{\top\ \ [x,k:=0,n;\quad \texttt{while}\ k\neq 0\ \texttt{do}\ x,k:=x\cdot a,k-1\ \texttt{od}\ ]\ \ x=n\cdot a}$$

where, writing W for the program proved,

A for $\quad x+k\cdot a = n\cdot a \rightarrow x\cdot a+(k-1)\cdot a = n\cdot a$

and B for $\quad (x+k\cdot a = n\cdot a)\wedge\neg(k\neq 0)\rightarrow x=n\cdot a$,

$\mathcal{D}$ is

$$\dfrac{\dfrac{\dfrac{\vdash A \qquad x\cdot a+(k-1)\cdot a = n\cdot a\ \ [x,k:=x\cdot a,k-1]\ \ x+k\cdot a=n\cdot a}{x+k\cdot a = n\cdot a\ \ [x,k:=x\cdot a,k-1]\ \ x+k\cdot a = n\cdot a}}{x+k\cdot a=n\cdot a\ \ [W]\ \ x+k\cdot a=n\cdot a\wedge\neg(k\neq 0)} \qquad \vdash B}{x+k\cdot a=n\cdot a\ \ [W]\ \ x=n\cdot a}$$

Soundness of H

- Both deductive calculi are Sound:

  If $\Gamma \vdash_H \psi[\alpha]\varphi$, then $\Gamma \models \psi[\alpha]\varphi$

- The rules of the calculus for GIPs are derivable from the rules for regular programs.
- Note the role of Γ (which was not realized historically).
- Completeness will be discussed later.

Are Hoare Logics practical?

- Hoare Logic proofs are cumbersome.
- Much repetition:

$$\frac{\psi\ [\alpha]\ \varphi \qquad \psi\ [\beta]\ \varphi}{\psi\ [\alpha \cup \beta]\ \varphi}$$

Programs duplicated, formulas triplicated!

Annotated imperative programs

- Fully annotated programs (FAPs) (of Γ):

Skip

$\{\varphi\}\,\texttt{skip}\,\{\varphi\}$ is a FAP.

Assignment

$\{\{\mathbf{t}/\mathbf{x}\}\varphi\}\ \mathbf{x} := \mathbf{t}\ \{\varphi\}$ is a FAP.

Composition

If $\alpha\{\varphi\}$ and $\{\varphi\}\beta$ are FAPs,
then so is $\alpha\{\varphi\}\,;\beta$.

Branching

If $\alpha' = \{\varphi \wedge \chi\}\alpha\{\psi\}$ and $\beta' = \{\varphi \wedge \neg\chi\}\beta\{\psi\}$ are FAPs,

then so is

$\{\varphi\}(\texttt{if}\ \chi\ \texttt{then}\ \alpha'\ \texttt{else}\ \beta'\ \texttt{fi})\,\{\psi\}$.

Iteration

If $\alpha' = \{\varphi \wedge \chi\}\alpha\{\varphi\}$ is a FAP,

then so is $\{\varphi\}(\texttt{while}\ \chi\ \texttt{do}\ \alpha'\ \texttt{od})\{\varphi \wedge \neg\chi\}$.

Consequence

Suppose $\Gamma \vdash \varphi \rightarrow \psi$.

If $\alpha\{\varphi\}$ is a FAP,

then so is $\alpha\{\varphi\}\{\psi\}$;

if $\{\psi\}\alpha$ is a FAP,

then so is $\{\varphi\}\{\psi\}\alpha$.

Examples: straight-line programs

- $\{x \geq 0\}$
  $\{x + 1 > 0\}$
  $x := x + 1$
  $\{x > 0\}$

- $\{x > y\}$
  $\{x + 1 > y + 1\}$
  $x := x + 1$
  $\{x > y + 1\};$
  $y := y + 1$
  $\{x > y\}$

- $\{x = a \wedge y = b\}$
  $x, y := y, x$
  $\{y = a \wedge x = b\}$

- $\{x = a \wedge y = b\}$
  $z := x$
  $\{z = a \wedge y = b\};$
  $x := y$
  $\{z = a \wedge x = b\};$
  $y := z$
  $\{y = a \wedge x = b\}$

- $\{Even(x) \wedge x \cdot y + z = a \cdot b\}$
  $\{(x \div 2) \cdot (y + y) + z = a \cdot b\}$
  $x := x \div 2$
  $\{x \cdot (y + y) + z = a \cdot b\};$
  $y := y + y$
  $\{x \cdot y + z = a \cdot b\}$

- $\{i = j^k\};$
  $\{i \cdot j = j^{k+1}\}$
  $k := k + 1;$
  $\{i \cdot j = j^k\}$
  $i := i \cdot j$
  $\{i = j^k\}$

- $\{Even(k) \wedge y \cdot z^k = c\};$
  $\{y \cdot (z \cdot z)^{k \div 2} = c\}$
  $k := k \div 2;$
  $\{y \cdot (z \cdot z)^k = c\}$
  $z := z \cdot z$
  $\{y \cdot z^k = c\}$

Examples: Branching programs

- $\{\top\}$
  $($`if` $x \geq y$ `then`
  $\{x \geq y\}$
  $\{x = x \wedge x \geq y\}$
  $z := x$
  $\{z = x \wedge z \geq y\}$
  $\{z = \max(x, y)\}$
  else
  $\{\neg(x \geq y)\}$
  $\{y = y \wedge x < y\}$
  $z := y$
  $\{z = y \wedge x < z\}$
  $\{z = \max(x, y)\}$
  fi
  $\{z = \max(x, y)\}$

- $\{y \cdot z^k = c\}$
  (if $\neg Even(k)$ then
  $\{y \cdot z^k = c \wedge \neg Even(k)\}$
  $\{(y \cdot z) \cdot z^{k-1} = c \wedge Even(k-1)\}$
  $k, y := k-1, y \cdot z;$
  $\{y \cdot z^k = c \wedge Even(k)\}$
  else
  $\{y \cdot z^k = c \wedge \neg\neg Even(k)\};$
  $\{y \cdot z^k = c \wedge Even(k)\}$
  skip
  $\{y \cdot z^k = c \wedge Even(k)\}$
  $\{Even(k) \wedge y \cdot z^k = c\}$

Inessential assertions

- Full annotation is also a drag: formulas still repeat.
- Essential are the first and last assertions of a FAP α: the pre-condition and post-condition. respectively.
- Also important is each invariant in a FAP: assertion that precedes a **while** identifier.
- For a program without iteration, a FAP can be obtained from the post-condition!
- For a single assignment the pre-condition determined uniquely by the post-condition.
- For branching program:
  Given FAPs $\{\varphi\}\beta\{\psi\}$ and $\{\varphi'\}\gamma\{\psi\}$,
  let $\xi = (\varphi \wedge \chi) \vee (\varphi' \wedge \neg\chi)$. Then $\vdash \xi \wedge \chi \rightarrow \varphi$ and $\vdash \xi \wedge \neg\chi \rightarrow \varphi'$. So

$$
\begin{array}{lll}
\{\xi\} & & \\
\text{if } \chi & \text{then} & \\
& \{\xi \wedge \chi\}\{\varphi\} & \\
& \beta & \\
& \{\psi\} & \\
\text{else} & & \\
& \{\xi \wedge \neg\chi\}\{\varphi'\} & \\
& \gamma & \\
& \{\psi\} & \\
\text{fi} & & \\
\{\psi\} & &
\end{array}
$$

is a FAP.

(Minimally) Annotated programs

- The *invariants* in a FAP cannot be derived mechanically.
- Assertion is essential if it is an invariant or the program's post-condition.
- An annotated program is obtained from a FAP by deleting any number of non-essential assertions.
- A minimally annotated program (MAP) is obtained from a FAP by deleting *all* non-essential assertions.
- reduct of an annotated program α is the minimally-annotated program $Red(\alpha)$ obtained by deleting all non-essential assertions in α.

Filling gaps in annotated programs

Define F that sends a MAP $\alpha\{\psi\}$ to a FAP $\{\varphi\}\bar{\alpha}\{\psi\}$, of which $\alpha\{\psi\}$ is a reduct.

Definition of F by recurrence on α.

- α is skip. Let $F(\alpha\{\psi\})$ be $\{\psi\}\mathrm{skip}\{\psi\}$.
- α is an assignment $\mathbf{x} := \mathbf{t}$. Let $F(\alpha\{\psi\})$ be $\{\{\mathbf{t}/\mathbf{x}\}\psi\}\ (\mathbf{x} := \mathbf{t})\ \{\psi\}$.
- α is $\beta;\gamma$. If $F(\gamma\{\psi\})$ is $\{\xi\}\gamma\{\psi\}$, and $F(\beta\{\xi\})$ is $\{\varphi\}\beta\{\xi\}$, then let $F(\alpha\{\psi\})$ be $\{\varphi\}\beta\{\xi\};\gamma\{\psi\}$.

- α is $\text{if } \chi \text{ then } \beta \text{ else } \gamma$. If $F(\beta\{\psi\})$ is $\{\xi\}\beta\{\psi\}$, and $F(\gamma\{\psi\})$ is $\{\xi'\}\gamma\{\psi\}$, then let ζ abbreviate $(\xi \wedge \chi) \vee (\xi' \wedge \neg\chi)$, and define $F(\alpha, \psi)$ to be

$$
\begin{array}{ll}
\{\zeta\} & \\
\text{if } \chi & \text{then} \\
 & \{\zeta \wedge \chi\} \\
 & \{\xi\} \\
 & \beta \\
 & \{\psi\} \\
\text{else} & \\
 & \{\zeta \wedge \neg\chi\} \\
 & \{\xi'\} \\
 & \gamma \\
 & \{\psi\} \\
\text{fi} & \\
\{\psi\} &
\end{array}
$$

- α is $\{\varphi\}$ while χ do β end . If $F(\beta\{\varphi\})$ is $\{\xi\}\beta\{\varphi\}$ then let $F(\alpha\{\psi\})$ be

$$
\begin{array}{ll}
\{\varphi\} & \\
\text{while } \chi & \text{do} \\
 & \{\varphi \wedge \chi\} \\
 & \{\xi\} \\
 & \beta \\
 & \{\varphi\} \\
\text{od} & \\
\{\varphi \wedge \neg\chi\} & \\
\{\psi\} &
\end{array}
$$

The Minimal Annotation Theorem

THEOREM. Given FAP $\{\varphi_0\}\alpha_0\{\psi_0\}$ and $\alpha\{\psi_0\} = \mathrm{R}(\{\varphi_0\}\alpha_0\{\psi_0\})$. Suppose $\Gamma \vdash \psi_0 \rightarrow \psi$. Then

1. $\mathrm{F}(\alpha\{\psi\})$ is a FAP,
2. $\mathrm{R}(\mathrm{F}(\alpha\{\psi\})) = \alpha\{\psi\}$, and
3. if φ is the pre-condition of $\mathrm{F}(\alpha\{\psi\})$, then $\Gamma \vdash \varphi_0 \rightarrow \varphi$.

Proof. Induction on α.

- α_0 is `skip`. All parts trivial.
- α_0 is an assignment $\mathbf{x} := \mathbf{t}$.
  Then φ_0 is $\{\mathbf{t}/\mathbf{x}\}\psi_0$, and φ is $\{\mathbf{t}/\mathbf{x}\}\psi$.
  (1) and (2) are immediate.
  Since we assume $\Gamma \vdash \psi_0 \rightarrow \psi$, and Γ has no free variables,
  $\Gamma \vdash .\{\mathbf{t}/\mathbf{x}\}\psi_0 \rightarrow \{\mathbf{t}/\mathbf{x}\}\psi$, ie $\Gamma \vdash .\varphi_0 \rightarrow \varphi$, proving (3).

- α_0 is $\beta_0\{\xi_0\};\gamma_0$.
  Then $\alpha = \beta;\gamma$, where
  $\gamma = \mathrm{R}(\{\xi_0\}\gamma_0\{\psi_0\})$,
  and $\beta = \mathrm{R}(\{\varphi_0\}\beta_0\{\xi_0\})$.
  By definition, $\mathrm{F}(\alpha\{\psi\})$ is of the form $\{\varphi\}\bar{\beta}\{\xi\};\bar{\gamma}\{\psi\}$.
  (1) and (2) are immediate by IH.
  By IH applied to γ_0, $\Gamma \vdash \xi_0 \rightarrow \xi$,
  so by IH for β_0, $\Gamma \vdash \varphi_0 \rightarrow \varphi$.

- α_0 is

$$
\begin{array}{ll}
\text{if } \chi & \text{then} \\
 & \{\varphi_0 \wedge \chi\} \\
 & \beta_0 \\
 & \{\psi_0\} \\
\text{else} & \\
 & \{\varphi_0 \wedge \neg\chi\} \\
 & \gamma_0 \\
 & \{\psi_0\} \\
\text{fi} &
\end{array}
$$

Then α is if χ then β else γ,
where $\beta = \mathrm{Red}(\{\varphi_0 \wedge \chi\}\beta_0\{\psi_0\})$
and $\gamma = \mathrm{Red}(\{\varphi_0 \wedge \neg\chi\}\gamma_0\{\psi_0\})$.
Also, $\mathrm{F}(\alpha\{\psi\})$ is of the form

$$\{\zeta\} \text{ if } \chi \text{ then } \{\zeta \wedge \chi\}\xi\bar{\beta}\ \{\psi\} \text{ else } \{\zeta \wedge \neg\chi\}\xi'\gamma\{\psi\} \text{ fi },$$

where $\zeta \equiv (\xi \wedge \chi) \vee (\xi' \wedge \neg\chi)$, $\{\xi\}\bar{\beta}\{\psi\}$ is $\mathrm{F}(\beta\{\psi\})$,
and $\{\xi'\}\bar{\gamma}\{\psi\}$ is $\mathrm{F}(\gamma\{\psi\})$.
By IH these two are correct FAPs, and we have
$\Gamma \vdash \varphi_0 \wedge \chi \to \xi$ and $\Gamma \vdash \varphi_0 \wedge \neg\chi \to \xi'$.
Thus, $\mathrm{F}(\alpha\{\psi\})$ is a correct FAP, and by propositional logic $\Gamma \vdash \varphi_0 \to \zeta$.

- α_0 is

$$\begin{array}{rl}
\{\varphi_0\} & \\
\texttt{while}\ \chi & \texttt{do} \\
& \{\varphi \wedge \chi\} \\
& \beta_0 \\
& \{\varphi\} \\
\texttt{od} & \\
\{\varphi \wedge \neg\chi\} &
\end{array}$$

$\mathrm{F}(\alpha\{\psi\})$ is then of the form

$$\begin{array}{rl}
\{\varphi_0\} & \\
\texttt{while}\ \chi & \texttt{do} \\
& \{\varphi_0 \wedge \chi\} \\
& \bar{\beta} \\
& \{\varphi_0\} \\
\texttt{od} & \\
\{\varphi_0 \wedge \neg\chi\} & \\
\{\psi\} &
\end{array}$$

Here φ is identical to φ_0,
(loop invariants are preserved throughout). By IH $\{\varphi_0 \wedge \chi\}\bar{\beta}\{\varphi_0\}$ is a correct FAP.
Also, the stated assumption is that $\Gamma \vdash \varphi_0 \wedge \neg\chi \to \psi$.
Hence $\mathrm{F}(\alpha_0\{\psi\})$ is a correct FAP.
The pre-conditions of α and $\mathrm{F}(\alpha_0\{\psi\})$ are identical, so (3) is trivial.

- α_0 is $\{\varphi_0\}\{\xi_0\}\beta_0\{\psi_0\}$,
  where $\Gamma \vdash \varphi_0 \to \xi_0$.
  If $\beta =_{\mathrm{df}} \mathrm{Red}(\beta_0)$, then $\alpha = \beta$,
  so by IH $\mathrm{F}(\alpha\{\psi\}) = \mathrm{F}(\beta\{\psi\})$ is a correct FAP, and (2) holds as well.

Finally, by IH $\Gamma \vdash \xi_0 \to \varphi$,
so $\Gamma \vdash \varphi_0 \to \varphi$.

$\dashv$