ISOMORPHIC DATA TRANSFORMATION

Alessandro Coglio

Kestrel Institute

Assumptions

given isomorphic domains (see separate 'Domain Mappings' notes):

$A \underset{\alpha'}{\overset{\alpha}{\longleftrightarrow}} A'$ and optionally $G[A \underset{\alpha'}{\overset{\alpha}{\longleftrightarrow}} A']$

$B \underset{\beta'}{\overset{\beta}{\longleftrightarrow}} B'$ and optionally $G[B \underset{\beta'}{\overset{\beta}{\longleftrightarrow}} B']$

Non-Recursive Function

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, shift right, "\alpha"'] & B \arrow[d, shift right, "\beta"'] \\ A' \arrow[u, shift right, "\alpha'"'] \arrow[r, "f'"'] & B' \arrow[u, shift right, "\beta'"'] \end{tikzcd}$$

old function: $f(x) \triangleq e(x)$ $\qquad f: \mathcal{U} \to \mathcal{U}$

condition: $\boxed{fAB}$ $\quad x \in A \Rightarrow f(x) \in B$ — $f(A) \subseteq B$ — $f: A \to B$

new function: $f'(x') \triangleq \underline{\text{if}}\ x' \in A'\ \underline{\text{then}}\ \beta(e(\alpha'(x')))\ \underline{\text{else}}\ \ldots$

any value (irrelevant)

this wrapping test is not strictly necessary, but it uniforms recursive and non-recursive case

$\vdash \boxed{f'f}$ $\quad x' \in A' \Rightarrow f'(x') = \beta(f(\alpha'(x')))$

$x' \in A'$ $\quad f'(x') \underset{\delta_{f'}}{=} \beta(e(\alpha'(x'))) \underset{\delta_f}{=} \beta(f(\alpha'(x')))$

QED

$\vdash \boxed{f'A'B'}$ $\quad x' \in A' \Rightarrow f'(x') \in B'$ — $f'(A') \subseteq B'$ — $f': A' \to B'$

$x' \in A' \xrightarrow{\alpha'A'} \alpha'(x') \in A \xrightarrow{fAB} f(\alpha'(x')) \in B \xrightarrow{\beta B} \beta(f(\alpha'(x'))) \in B' \xrightarrow{f'f} f'(x') \in B'$

QED

$\vdash \boxed{ff'}$ $\quad x \in A \Rightarrow f(x) = \beta'(f'(\alpha(x)))$

$x \in A \xrightarrow{\alpha A} \alpha(x) \in A'$

$x \in A \xrightarrow[\alpha'\alpha]{} \alpha'(\alpha(x)) = x$

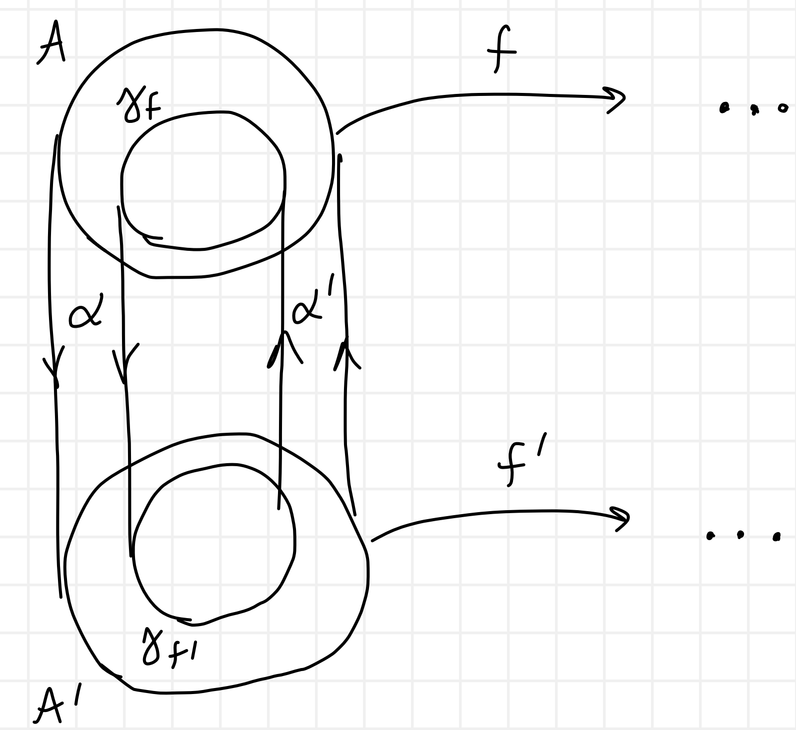
$\alpha(x) \in A' \xrightarrow[x' := \alpha(x)]{f'f} f'(\alpha(x)) = \beta(f(\alpha'(\alpha(x)))) = \beta(f(x))$

$x \in A \xrightarrow{fAB} f(x) \in B$

$\beta'(f'(\alpha(x))) = \beta'(\beta(f(x))) \underset{\beta'\beta}{=} f(x)$

QED

Guards for Non-Recursive Function



$\boxed{\surd f}$ $\gamma_{\gamma_f}(x) \wedge [\gamma_f(x) \Rightarrow \gamma_e(x)]$

condition: $\boxed{Gf}$ $\gamma_f(x) \Rightarrow x \in A$

$\gamma_{f'}(x') \triangleq [x' \in A' \wedge \gamma_f(\alpha'(x'))]$

$\vdash \gamma_f(x) \Rightarrow \gamma_{f'}(\alpha(x))$

$\gamma_f(x) \xrightarrow{Gf} x \in A \xrightarrow{\alpha'\alpha} \alpha'(\alpha(x)) = x$

$\delta_{\gamma_{f'}} \xrightarrow[x' := \alpha(x)]{} \gamma_{f'}(\alpha(x)) = \gamma_f(\alpha'(\alpha(x))) = \gamma_f(x)$

$\gamma_{f'}(\alpha(x))$

QED

$\vdash \gamma_{f'}(x') \Rightarrow \gamma_f(\alpha'(x))$

$\gamma_{f'}(x') \xrightarrow{\delta_{\gamma_{f'}}} x' \in A' \wedge \gamma_f(\alpha'(x'))$

QED

$\vdash \boxed{\surd f'}$

$\omega_{f'}(x') = \overset{GA'}{\cancel{\gamma_{A'}(x')}} \wedge [x' \in A' \Rightarrow \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \gamma_{\gamma_f}(\overset{\surd f}{\cancel{\alpha'(x')}})] \wedge$

$[x' \in A' \wedge \gamma_f(\alpha'(x')) \Rightarrow \overset{GA'}{\cancel{\gamma_{A'}(x')}} \wedge [x' \in A' \Rightarrow \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \overset{\surd f}{\cancel{\gamma_e(\alpha'(x'))}} \wedge \overset{G\beta}{\cancel{\gamma_\beta(e(\alpha'(x')))}}]$

$\wedge [x' \notin A' \Rightarrow \cancel{\surd \ldots}]]$

$\alpha' A'$

$\alpha'(x') \in A \xrightarrow{fAB} f(\alpha'(x')) \in B \xrightarrow{\delta_f} e(\alpha'(x')) \in B$

QED

Recursive Function

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, shift right, "\alpha"'] & B \arrow[d, shift right, "\beta"'] \\ A' \arrow[r, "f'"] \arrow[u, shift right, "\alpha'"'] & B' \arrow[u, shift right, "\beta'"'] \end{tikzcd}$$ $\Bigg\}$ same picture as non-recursive case

old function: $f(x) \triangleq \underline{\text{if}}\ a(x)\ \underline{\text{then}}\ b(x)\ \underline{\text{else}}\ c(x, f(d(x)))$ $\quad f: U \to U$

$\boxed{\tau_f}$ $\neg a(x) \Rightarrow \mu_f(d(x)) \prec_f \mu_f(x)$

conditions $\begin{cases} \boxed{fAB} & x \in A \Rightarrow f(x) \in B \quad \text{— as in non-recursive case} \\ \boxed{Ad} & x \in A \wedge \neg a(x) \Rightarrow d(x) \in A \quad \text{— recursive call preserves } A \end{cases}$

new function: $f'(x') \triangleq \underline{\text{if}}\ x' \in A'\ \underline{\text{then}} \left[\underline{\text{if}}\ a(\alpha'(x'))\ \underline{\text{then}}\ \beta(b(\alpha'(x')))\ \underline{\text{else}}\ \beta(c(\alpha'(x'), \beta'(f'(\alpha(d(\alpha'(x')))))))\right]\ \underline{\text{else}}\ \underbrace{\ldots}_{\text{any value (irrelevant)}}$

$\mu_{f'}(x') \triangleq \mu_f(\alpha'(x')) \qquad \prec_{f'} \triangleq \prec_f$

$\vdash \boxed{\tau_{f'}}$ $x' \in A' \wedge \neg a(\alpha'(x')) \Rightarrow \mu_{f'}(\alpha(d(\alpha'(x')))) \prec_{f'} \mu_{f'}(x')$ — f' terminates

$x' \in A' \xrightarrow{\alpha' A'} \alpha'(x') \in A \xrightarrow{Ad} d(\alpha'(x')) \in A \xrightarrow{\alpha'\alpha} \alpha'(\alpha(d(\alpha'(x')))) = d(\alpha'(x'))$

$\neg a(\alpha'(x'))$

$\mu_{f'}(\alpha(d(\alpha'(x')))) \overset{\delta_{\mu_{f'}}}{=\!=} \mu_f(\alpha'(\alpha(d(\alpha'(x'))))) = \mu_f(d(\alpha'(x'))) \prec_f \mu_f(\alpha'(x')) \overset{\delta_{\mu_{f'}}}{=\!=} \mu_{f'}(x')$

$\qquad \| \ \delta_{\prec_{f'}}$

$\qquad \prec_{f'}$

QED

$\vdash \boxed{f'f}$ $\quad x' \in A' \Rightarrow f'(x') = \beta(f(\alpha'(x')))$ — as in non-recursive case, with additional hypothesis

induct f'

base) $a(\alpha'(x')) \xrightarrow{\delta_{f'}} f'(x') = \beta(b(\alpha'(x')))$

$a(\alpha'(x')) \xrightarrow{\delta_f} f(\alpha'(x')) = b(\alpha'(x'))$

$\Rightarrow f'(x') = \beta(f(\alpha'(x')))$

step) $\neg a(\alpha'(x')) \xrightarrow{\delta_{f'}} f'(x') = \beta(c(\alpha'(x'), \beta'(f'(\alpha(d(\alpha'(x'))))))) \overset{IH}{=} \beta(c(\alpha'(x'), \beta'(\beta(f(\alpha'(\alpha(d(\alpha'(x'))))))))))$

$x' \in A' \xrightarrow{\alpha'A'} \alpha'(x') \in A \xrightarrow{Ad} d(\alpha'(x')) \in A \xrightarrow{\alpha A} \alpha(d(\alpha'(x'))) \in A'$

$d(\alpha'(x')) \in A \xrightarrow{\alpha'\alpha} \alpha'(\alpha(d(\alpha'(x')))) = d(\alpha'(x'))$

$d(\alpha'(x')) \in A \xrightarrow{fAB} f(d(\alpha'(x'))) \in B \xrightarrow{\beta'\beta} \beta'(\beta(f(d(\alpha'(x'))))) = f(d(\alpha'(x')))$

$\neg a(\alpha'(x')) \xrightarrow{\delta_f} f(\alpha'(x')) = c(\alpha'(x'), f(d(\alpha'(x'))))$

$= \beta(c(\alpha'(x'), \beta'(\beta(f(d(\alpha'(x'))))))))$

$= \beta(c(\alpha'(x')), f(d(\alpha'(x'))))$

$= \beta(f(\alpha'(x')))$

QED

$\vdash \boxed{f'A'B'}$ $\quad x' \in A' \Rightarrow f'(x') \in B'$ — as in non-recursive case; same proof

$\vdash \boxed{ff'}$ $\quad x \in A \Rightarrow f(x) = \beta'(f'(\alpha(x)))$ — as in non-recursive case; same proof

Guards for Recursive Function

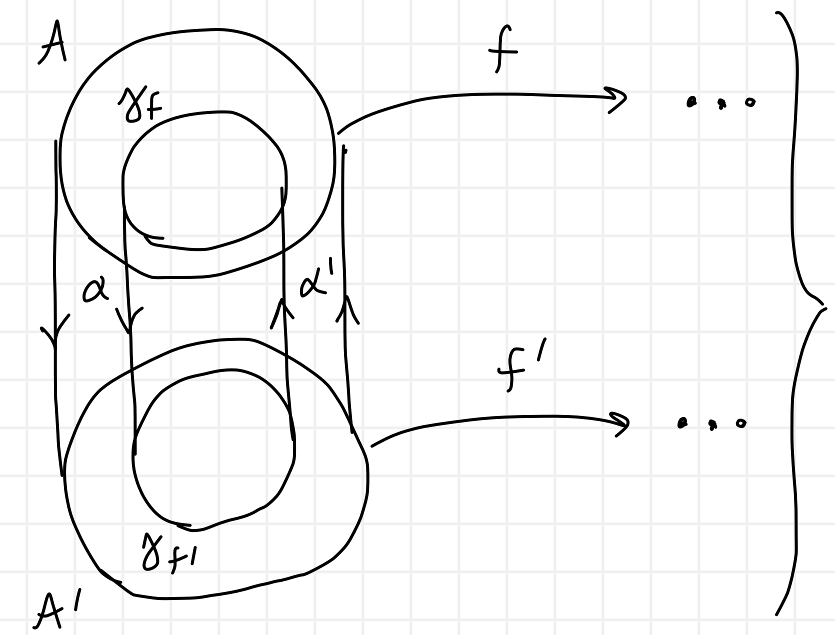
$\boxed{\sqrt{f}}$ $\gamma_{\gamma_f}(x) \wedge [\gamma_f(x) \Rightarrow \gamma_a(x) \wedge [a(x) \Rightarrow \gamma_b(x)] \wedge [\neg a(x) \Rightarrow \gamma_d(x) \wedge \gamma_f(d(x)) \wedge \gamma_c(x, f(d(x)))]]$

condition: $\boxed{Gf}$ $\gamma_f(x) \Rightarrow x \in A$

$\gamma_{f'}(x') \triangleq [x' \in A' \wedge \gamma_f(\alpha'(x'))]$ — as in non-recursive case

$\vdash \gamma_f(x) \Rightarrow \gamma_{f'}(\alpha(x))$ — as in non-recursive case; same proof

$\vdash \gamma_{f'}(x') \Rightarrow \gamma_f(\alpha'(x))$ — as in non-recursive case; same proof



same picture as non-recursive case

$\vdash \boxed{\sqrt{f'}}$

$\omega_{f'}(x') = \overset{GA'}{\cancel{\gamma_{A'}(x')}} \wedge [x' \in A' \Rightarrow \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \overset{\sqrt{f}}{\cancel{\gamma_{\gamma_f}(\alpha'(x'))}}] \wedge$

$[x' \in A' \wedge \gamma_f(\alpha'(x')) \Rightarrow \overset{GA'}{\cancel{\gamma_{A'}(x')}} \wedge$

$[x' \in A' \Rightarrow \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \overset{\sqrt{f}}{\cancel{\gamma_a(\alpha'(x'))}}] \wedge$

$[a(\alpha'(x')) \Rightarrow \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \overset{\sqrt{f}}{\cancel{\gamma_b(\alpha'(x'))}} \wedge \overset{GB}{\cancel{\gamma_B(b(\alpha'(x')))}}] \wedge$

$[\neg a(\alpha'(x')) \Rightarrow \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \overset{\sqrt{f}}{\cancel{\gamma_d(\alpha'(x'))}} \wedge \overset{G\alpha}{\cancel{\gamma_\alpha(d(\alpha'(x')))}} \wedge$

$\cancel{\alpha(d(\alpha'(x'))) \in A'} \wedge \overset{\sqrt{f}}{\cancel{\gamma_f(\alpha'(\alpha(d(\alpha'(x')))))}} \wedge$

$\overset{GB'}{\cancel{\gamma_{B'}(f'(\alpha(d(\alpha'(x')))))}} \wedge$

$\overset{\sqrt{f}}{\cancel{\gamma_c(\alpha'(x'), \beta'(f'(\alpha(d(\alpha'(x'))))))}} \wedge$

$\overset{GB}{\cancel{\gamma_B(c(\alpha'(x'), \underbrace{\beta'(f'(\alpha(d(\alpha'(x')))))}_{= f(d(\alpha'(x')))}))}}]] \wedge$

$[x \notin A' \Rightarrow \cancel{\sqrt{\dots}}]$

$\alpha'A'$: $\alpha'(x') \in A$

δ_f, fAB: $f(\alpha'(x')) = b(\alpha'(x')) \in B$

Ad: $d(\alpha'(x')) \in A$

$\alpha'\alpha$: $\alpha'(\alpha(d(\alpha'(x')))) = d(\alpha'(x'))$

αA: $\alpha(d(\alpha'(x'))) \in A'$

$f'A'B'$: $f'(\alpha(d(\alpha'(x')))) \in B'$

ff': $f(d(\alpha'(x'))) = \beta'(f'(\alpha(d(\alpha'(x')))))$

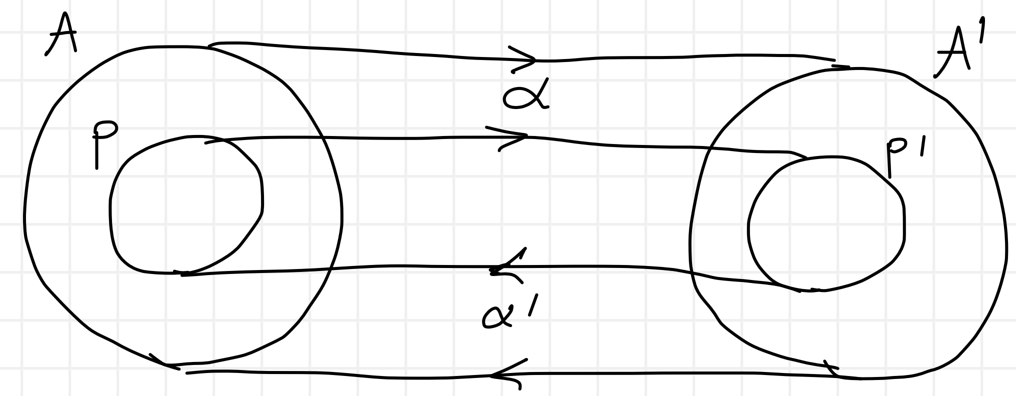
δ_f: $f(\alpha'(x')) = c(\alpha'(x'), f(d(\alpha'(x'))))$

fAB: $c(\alpha'(x'), f(d(\alpha'(x')))) \in B$

QED

Non-Recursive Predicate

old predicate: $p(x) \triangleq e(x)$ $\quad\quad p \subseteq \mathcal{U}$



condition: $\boxed{pA}$ $\quad p(x) \Rightarrow x \in A$ $\quad$ — $\quad p \subseteq A$

new predicate: $p'(x') \triangleq [x' \in A' \wedge e(\alpha'(x'))]$

$\vdash$ $\boxed{p'p}$ $\quad x' \in A' \Rightarrow p'(x') = p(\alpha'(x'))$ $\quad$ — as in function case, but without β

$x' \in A'$ $\quad p'(x') \overset{\delta p'}{=} [\cancel{x' \in A'} \wedge e(\alpha'(x'))]$

$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \| \; \delta_p$

$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad p(\alpha'(x'))$

QED

$\vdash$ $\boxed{p'A'}$ $\quad p'(x') \Rightarrow x' \in A'$ $\quad$ — $\quad p' \subseteq A'$

$p'(x') \overset{\delta p'}{=} x' \in A' \wedge \dots \rightarrow x' \in A'$

QED

$\vdash$ $\boxed{pp'}$ $\quad x \in A \Rightarrow p(x) = p'(\alpha(x))$ $\quad$ — as in function case, but without β

$x \in A \xrightarrow{\alpha'\alpha} \alpha'(\alpha(x)) = x$

$x \in A \xrightarrow{\alpha A} \alpha(x) \in A'$

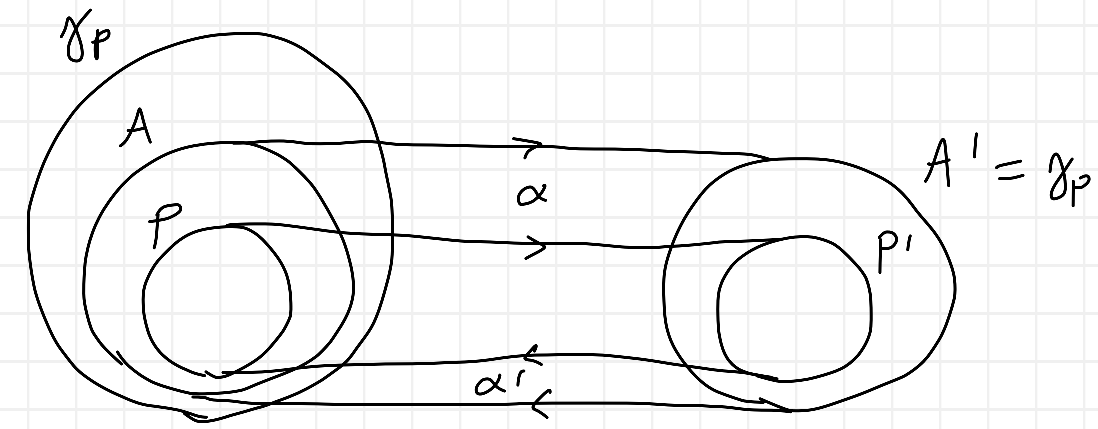
$\xrightarrow[x' := \alpha(x)]{p'p} p'(\alpha(x)) = p(\alpha'(\alpha(x))) = p(x)$

QED

Guards for Non-Recursive Predicate

$\boxed{\gamma P}$ $\gamma_{\gamma_P}(x) \wedge \left[\gamma_P(x) \Rightarrow \gamma_e(x)\right]$

condition: $\boxed{G_P}$ $x \in A \Rightarrow \gamma_P(x)$



$\gamma_{P'}(x') \triangleq x' \in A'$

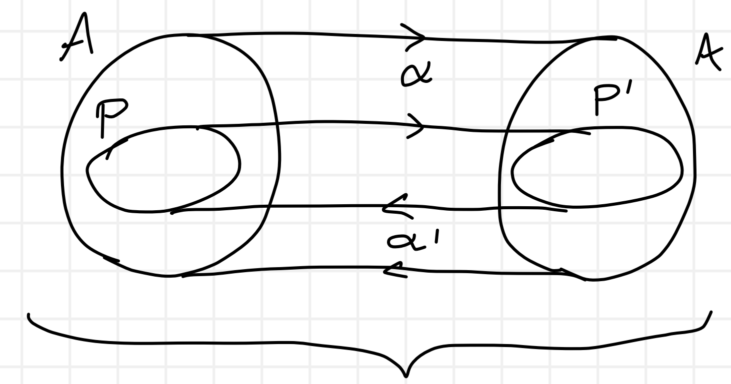
$\vdash \boxed{\gamma P'}$

$$\omega_{P'}(x') = \left[\overset{GA'}{\cancel{\gamma_{A'}(x')}} \wedge \left[x' \in A' \Rightarrow \overset{GA'}{\cancel{\gamma_{A'}(x')}} \wedge \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \overset{\gamma P}{\cancel{\gamma_e(\alpha'(x'))}}\right]\right]$$

$$x' \in A' \xrightarrow{\alpha' A'} \alpha'(x') \in A \xrightarrow{G_P} \gamma_P(\alpha'(x'))$$

QED

Recursive Predicate



same picture as non-recursive case

old predicate : $p(x) \triangleq \underline{\text{if}}\ a(x)\ \underline{\text{then}}\ b(x)\ \underline{\text{else}}\ c(x, p(d(x)))$ $\qquad p \subseteq \mathcal{U}$

$\boxed{\tau_p}$ $\quad \neg a(x) \Rightarrow \mu_p(d(x)) \prec_p \mu_p(x)$

conditions $\begin{cases} \boxed{pA} & p(x) \Rightarrow x \in A \\ \boxed{Ad} & x \in A \wedge \neg a(x) \Rightarrow d(x) \in A \end{cases}$

— as in non-recursive predicate case (for $\boxed{pA}$)

— as in recursive function case (for $\boxed{Ad}$)

new predicate : $p'(x') \triangleq x' \in A' \wedge \left[\underline{\text{if}}\ a(\alpha'(x'))\ \underline{\text{then}}\ b(\alpha'(x'))\ \underline{\text{else}}\ c(\alpha'(x'), p'(\alpha(d(\alpha'(x')))))\right]$

$\mu_{p'}(x') \triangleq \mu_p(\alpha'(x')) \qquad \prec_{p'} \triangleq \prec_p$

$\vdash \boxed{\tau_{p'}}$ $\quad x' \in A' \wedge \neg a(\alpha'(x')) \Longrightarrow \mu_{p'}(\alpha(d(\alpha'(x')))) \prec_{p'} \mu_{p'}(x')$ — p' terminates — same proof as recursive function case

$\vdash \boxed{p'p}$ $\quad x' \in A' \Rightarrow p'(x') = p(\alpha'(x'))$ — as in non-recursive predicate case

induct p'

base) $a(\alpha'(x')) \xrightarrow{\delta_{p'}} p'(x') = b(\alpha'(x'))$

$a(\alpha'(x')) \xrightarrow{\delta_p} p(\alpha'(x')) = b(\alpha'(x'))$

$\longrightarrow p'(x') = p(\alpha'(x'))$

step) $\neg a(\alpha'(x')) \xrightarrow{\delta_{p'}} p'(x') = c(\alpha'(x'), p'(\alpha(d(\alpha'(x'))))) \underset{\text{IH}}{=} c(\alpha'(x'), p(\alpha'(\alpha(d(\alpha'(x'))))))$

$x' \in A' \xrightarrow{\alpha' A'} \alpha'(x') \in A \xrightarrow{Ad} d(\alpha'(x')) \in A \xrightarrow{\alpha A} \alpha(d(\alpha'(x'))) \in A'$

$d(\alpha'(x')) \in A \xrightarrow{\alpha'\alpha} \alpha'(\alpha(d(\alpha'(x')))) = d(\alpha'(x'))$

$= c(\alpha'(x'), p(d(\alpha'(x'))))$

$\neg a(\alpha'(x')) \xrightarrow{\delta_p} p(\alpha'(x')) = c(\alpha'(x'), p(d(\alpha'(x'))))$

$= p(\alpha'(x'))$

QED

$\vdash \boxed{p'A'}$ $\quad p'(x') \Rightarrow x' \in A'$ — $p' \subseteq A'$ — as in non-recursive predicate case

$p'(x')$ $\qquad x' \notin A' \xrightarrow{\delta_{p'}} \neg p'(x') \longrightarrow$ impossible $\longrightarrow x' \in A$

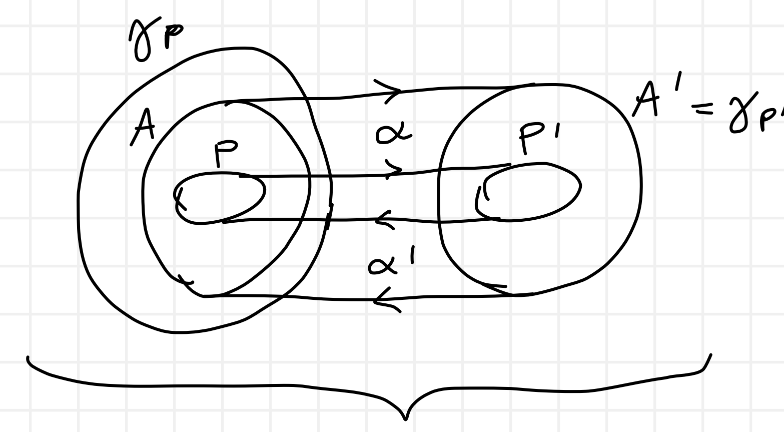
QED

$\vdash \boxed{pp'}$ $\quad x \in A \Rightarrow p(x) = p'(\alpha(x))$ — as in non-recursive predicate case ; same proof

Guards for Recursive Predicate

$\boxed{\gamma p}$ $\gamma_{\gamma_p}(x) \wedge [\gamma_p(x) \Rightarrow \gamma_a(x) \wedge [a(x) \Rightarrow \gamma_b(x)] \wedge [\neg a(x) \Rightarrow \gamma_d(x) \wedge \gamma_p(d(x)) \wedge \gamma_c(x, p(d(x)))]]$

condition: $\boxed{Gp}$ $x \in A \Rightarrow \gamma_p(x)$ — as in non-recursive predicate case



same picture as non-recursive case

$\gamma_{p'}(x') \triangleq x' \in A'$ — as in recurcive predicate case

$\vdash \boxed{\gamma p'}$

$\omega_{p'}(x') = \overset{GA'}{\cancel{\gamma_{A'}(x')}} \wedge$

$[x' \in A' \Rightarrow \overset{GA'}{\cancel{\gamma_{A'}(x')}} \wedge$

$[x' \in A' \Rightarrow \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \overset{\gamma p}{\cancel{\gamma_a(\alpha'(x'))}} \wedge$

$[a(\alpha'(x')) \Rightarrow \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \overset{\gamma p}{\cancel{\gamma_b(\alpha'(x'))}}] \wedge$

$[\neg a(\alpha'(x')) \Rightarrow \overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge$

$\overset{G\alpha'}{\cancel{\gamma_{\alpha'}(x')}} \wedge \overset{\gamma p}{\cancel{\gamma_d(\alpha'(x'))}} \wedge \overset{G\alpha}{\cancel{\gamma_\alpha(d(\alpha'(x')))}} \wedge$

$\overset{\gamma p}{\cancel{\gamma_c(\alpha'(x'), p'(\alpha(d(\alpha'(x')))))}}]]]$

$\xrightarrow{\alpha' A'} \alpha'(x') \in A \xrightarrow{Gp} \gamma_p(\alpha'(x'))$

$\xrightarrow{Ad} d(\alpha'(x')) \in A$

$\xrightarrow{\alpha'\alpha} \alpha'(\alpha(d(\alpha'(x')))) = d(\alpha'(x'))$

$\xrightarrow{\alpha A} \alpha(d(\alpha'(x'))) \in A' \wedge$

$p'(\alpha(d(\alpha'(x')))) = p(\alpha'(\alpha(d(\alpha'(x'))))) \xleftarrow{p'p} \alpha(d(\alpha'(x'))) \in A'$

$= p(d(\alpha'(x')))$

QED

Generalization to Tuples

$f : \mathcal{U}^n \to \mathcal{U}^m$ $\quad$ $p \subseteq \mathcal{U}^n$ $\quad$ $A \subseteq \mathcal{U}^n$ $\quad$ $B \subseteq \mathcal{U}^m$ $\quad$ $\alpha : \mathcal{U}^n \to \mathcal{U}^{n'}$ $\quad$ $\beta : \mathcal{U}^m \to \mathcal{U}^{m'}$

$f' : \mathcal{U}^{n'} \to \mathcal{U}^{m'}$ $\quad$ $p' \subseteq \mathcal{U}^{n'}$ $\quad$ $A' \subseteq \mathcal{U}^{n'}$ $\quad$ $B' \subseteq \mathcal{U}^{m'}$ $\quad$ $\alpha' : \mathcal{U}^{n'} \to \mathcal{U}^n$ $\quad$ $\beta' : \mathcal{U}^{m'} \to \mathcal{U}^{n'}$

straightforward, similar to 'Isomorphisms' notes

Compositional Establishment of Isomorphic Mappings on Tuples

partition old and new inputs into equal numbers of disjoint non-empty subsets:

$$\left.\begin{array}{llll} \{1,\dots,n\} = \{i_{1,1},\dots,i_{1,n_1}\} \uplus \dots \uplus \{i_{k,1},\dots,i_{k,n_k}\} & , & n_1>0,\dots,n_k>0 & , \quad n_1+\dots+n_k = n \geq 1 \\ \{1,\dots,n'\} = \{i'_{1,1},\dots,i'_{1,n'_1}\} \uplus \dots \uplus \{i'_{k,1},\dots,i'_{k,n'_k}\} & , & n'_1>0,\dots,n'_k>0 & , \quad n'_1+\dots+n'_k = n' \geq 1 \end{array}\right\} \text{same } k$$

establish isomorphic mappings between each pair of partitions:

$$A_1 \underset{\alpha'_1}{\overset{\alpha_1}{\longleftrightarrow}} A'_1 \;,\; \dots \;,\; A_k \underset{\alpha'_k}{\overset{\alpha_k}{\longleftrightarrow}} A'_k \quad , \quad A_1 \subseteq \mathcal{U}^{n_1},\dots, A_k \subseteq \mathcal{U}^{n_k} \quad , \quad A'_1 \subseteq \mathcal{U}^{n'_1},\dots,A'_k \subseteq \mathcal{U}^{n'_k}$$

combine the isomorphic mappings:

$$A \triangleq \{\langle x_1,\dots,x_n\rangle \in \mathcal{U}^n \mid \langle x_{i_{1,1}},\dots,x_{i_{1,n_1}}\rangle \in A_1 \wedge \dots \wedge \langle x_{i_{k,1}},\dots,x_{i_{k,n_k}}\rangle \in A_k\}$$

$$A' \triangleq \{\langle x'_1,\dots,x'_{n'}\rangle \in \mathcal{U}^{n'} \mid \langle x'_{i'_{1,1}},\dots,x'_{i'_{1,n'_1}}\rangle \in A'_1 \wedge \dots \wedge \langle x'_{i'_{k,1}},\dots x'_{i'_{k,n'_k}}\rangle \in A'_k\}$$

$$\left.\begin{array}{l} \alpha(x_1,\dots,x_n) \triangleq \langle \alpha_1(x_{i_{1,1}},\dots,x_{i_{1,n_1}}),\dots,\alpha_k(x_{i_{k,1}},\dots,x_{i_{k,n_k}})\rangle \\ \alpha'(x'_1,\dots,x'_{n'}) \triangleq \langle \alpha'_1(x'_{i'_{1,1}},\dots,x'_{i'_{1,n'_1}}),\dots,\alpha'_k(x'_{i'_{k,1}},\dots,x'_{i'_{k,n'_k}})\rangle \end{array}\right\} \text{flatten nested tuples}$$

do analogously for old and new outputs:

$$\left.\begin{array}{llll} \{1,\dots,m\} = \{j_{1,1},\dots,j_{1,m_1}\} \uplus \dots \uplus \{j_{h,1},\dots,j_{h,m_h}\} & , & m_1>0,\dots,m_h>0 & , \quad m_1+\dots+m_h = m \geq 1 \\ \{1,\dots,m'\} = \{j'_{1,1},\dots,j'_{1,m'_1}\} \uplus \dots \uplus \{j'_{h,1},\dots,j'_{h,m'_h}\} & , & m'_1>0,\dots,m'_h>0 & , \quad m'_1+\dots+m'_h = m' \geq 1 \end{array}\right\} \text{same } k$$

$$B_1 \underset{\beta'_1}{\overset{\beta_1}{\longleftrightarrow}} B'_1 \;,\; \dots \;,\; B_h \underset{\beta'_h}{\overset{\beta_h}{\longleftrightarrow}} B'_h \quad , \quad B_1 \subseteq \mathcal{U}^{m_1},\dots,B_h \subseteq \mathcal{U}^{m_h} \quad , \quad B'_1 \subseteq \mathcal{U}^{m'_1},\dots,B'_h \subseteq \mathcal{U}^{m'_h}$$

$$B \triangleq \{\langle y_1,\dots,y_m\rangle \in \mathcal{U}^m \mid \langle y_{j_{1,1}},\dots,y_{j_{1,m_1}}\rangle \in B_1 \wedge \dots \wedge \langle y_{j_{h,1}},\dots,y_{j_{h,m_h}}\rangle \in B_h\}$$

$$B' \triangleq \{\langle y'_1,\dots,y'_{m'}\rangle \in \mathcal{U}^{m'} \mid \langle y'_{j'_{1,1}},\dots,y'_{j'_{1,m'_1}}\rangle \in B'_1 \wedge \dots \wedge \langle y'_{j'_{h,1}},\dots,y'_{j'_{h,m'_h}}\rangle \in B'_h\}$$

$$\left.\begin{array}{l} \beta(y_1,\dots,y_m) \triangleq \langle \beta_1(y_{j_{1,1}},\dots,y_{j_{1,h}}),\dots,\beta_h(y_{j_{h,1}},\dots,y_{j_{h,m_h}})\rangle \\ \beta'(y'_1,\dots,y'_{m'}) \triangleq \langle \beta'_1(y'_{j'_{1,1}},\dots,y'_{j'_{1,h}}),\dots,\beta'_h(y'_{j'_{h,1}},\dots,y'_{j'_{h,m'_h}})\rangle \end{array}\right\} \text{flatten nested tuples}$$