## Realizing Bar Induction ($^{*}26.3c$) a la Kleene FIM p. 107–109

Kleene assumes that function $\pi$ realizes the hypotheses of axiom schema $^{*}26.3$, Bar Induction – discussed on pages 48–57. Schema $^{*}26.3$ applies to the universal spread, and from it theorem $^{*}26.4$ applies to general spreads. See class handouts and web page for Kleene's account in detail. These notes summarize and simplify.

Bar Induction (for universal spread) $^{*}26.3c$:

$$\forall\alpha\, \exists!\, x\, R(\bar\alpha(x))\ \&\ \forall a\,[\,Seq(a)\ \&\ R(a) \supset A(a)\,]\ \&$$
$$\forall a\,[\,Seq(a)\ \&\ \forall s\, A(a * 2^{s+1}) \supset A(a)\,] \supset A(1)$$

Assume that function $\pi$ realizes the hypothesis, then

$(\pi)_{0,0}$ realizes $\forall\alpha\,\exists!\,x\,R(\bar\alpha(x))$, i.e. $\forall\alpha\,\exists x\,[\,R(\bar\alpha(x))\ \&\ \forall y\,(R(\bar\alpha(y)) \supset x = y)\,]$

Let $bar = \{(\pi)_{0,0}\}$ the index of the recursive function.

$(\pi)_{0,1}$ realizes $\forall a\,[\,Seq(a)\ \&\ R(a) \supset A(a)\,]$

Let $base = \{(\pi)_{0,1}\}$

$(\pi)_1$ realizes $\forall a\,[\,Seq\ \&\ \forall s\, A(a*2^{s+1}) \supset A(a)\,]$

Let $ind = \{(\pi)_1\}$

(1) $bar[\alpha]_1$ realizes $R(\bar\alpha(x))\ \&\ \forall y\,(R(\bar\alpha(y)) \supset x = y)$
for $x = (bar[\alpha](0))_0$ (see p. 96 for explanation of $bar[\alpha](0)$, a trivial detail, can also use $bar[\alpha]_0$ for simplicity.)

(2) for each $a$, $\rho_0, \rho_1$ where $\rho_0$ realizes-a $Seq(a)$ and $\rho_1$ realizes-a $R(a)$ then $base[a]\langle\rho_0,\rho_1\rangle$ realizes-a $A(a)$
(We use $\langle\rho_0,\rho_1\rangle$ for the pair instead of $[\rho_0,\rho_1]$.)

Realizing Bar Induction ($^{*}26.3c$) continued 2

(3) For each $a, \rho_0, \rho_1$ if $\rho_0$ realizes-a $Seq(a)$ and for each $s$ $\{\rho_1\}[s]$ realizes-a, $s$ $A(a * 2^{s+1})$ then $\{ind[\alpha]\}\langle\rho_0,\rho_1\rangle$ realizes-a $A(a)$

(4) In Kleene this is a trivial realizer for $x=y$ and is not critical.

(5) Here we define a recursive predicate $R_1(a)$ to identify the immediately secured nodes, the base case of Bar Induction.

$$R_1(a) \simeq a = \bar{\alpha}_1(x_1) \text{ for } \alpha_1 = \lambda(t.\ a_t \dot{-} 1)$$
$$\text{and } x_1 = (bar[\alpha_1](0))_0 \quad (\text{recall (1) above}).$$

(6) In this item Kleene deals with more elementary "assembly language" coding that is not critical.

(7) Define informally the tree $S_1^{\pi}$ of sequence numbers barred by the predicate $R$. Kleene uses this informal dependent function type to characterize the bar-ind ~~per~~ "combinator" he seeks, he calls it $\eta[\pi, a]$. We will call it bar-ind(a).

$$\boxed{\begin{array}{l} a \in S_1^{\pi} \to \{\eta(\pi,a) \text{ realizes-a } A(a)\} \\ a \in S_1^{\pi} \to \{\text{bar-ind}(a) \text{ realizes } A(a)\} \end{array}}$$

He uses ~~a~~ informal bar induction to justify the $\eta(\pi,a)$ (for us bar-ind) realizer. This argument is quite easy to follow, and is not repeated here. We go to the definition of bar-ind.

# Realizing Bar Induction continued - 3

We add item (8) defining the bar induction combinator.
Kleene defines $\eta(\pi, a, u)$ as a way of combining the base case realizer with the induction case through the argument $u$. He simply defines $h(\pi, a) = \lambda(u. \eta(\pi, a, u))$.

We let sq realize Seq(a) for any a.

$$\begin{aligned} \text{bar\_ind}(a,u) == \;& \underline{\text{if}}\ R_1(a)\ \underline{\text{then}}\ (\text{base}[a] \langle sq, \text{bar}[\alpha, ] \rangle)_{1,0}\, u \\ & \underline{\text{else}}\ \ \text{ind}[a] \langle sq, \lambda(s. \lambda(t. \text{bar\_ind}(a * 2^{s+1}, t))) \rangle\, u \end{aligned}$$

If we leave out the trivial realizer of Seq then the shape of the realizer is more transparent.

$$\begin{aligned} \text{bar\_ind}(a,u) == \;& \underline{\text{if}}\ R_1(a)\ \underline{\text{then}}\ \ \text{base}[a] (\text{bar}[\alpha, ])_{1,0}\, u \\ & \underline{\text{else}}\ \ \text{ind}[a] (\lambda(s. \lambda(t. \text{bar\_ind}(a * 2^{s+1}, t))))\, u \end{aligned}$$

$$\text{bar\_ind}(a) = \lambda(u. \text{bar\_ind}(a,u))$$

Kleene uses informal bar induction to prove that this realizer has the right type, thus terminates.