1.1) Aplicando [block_{ns}] obtenemos s' con $s'\,fx = 1$ y upd_p genera env'_p con $env'_p\,fact = F$ "cuerpo" de la declaración de fact. Finalmente "calculamos" $env'_p \vdash \langle call\ fact, s'\rangle \to s''$

Aplicando [call^{rec}_{ns}] llegamos a $env'_p \vdash \langle F, s'\rangle \to s''$

Tenemos $s'x = sx \in \mathbb{N}$ y vamos a demostrar que $\boxed{s''\,fx = (sx)!}$ por inducción sobre sx

- $sx = 0$ Aplicando [if^{ff}_{ns}] obtenemos $s''\,fx = 1 = 0! = (sx)!$
- $sx = n > 0$ Aplicando [if^{tt}_{ns}] y [ass_{ns}] llegamos a s''' con $s'''\,fx = n$, $s'''x = sx - 1 = n-1$, y "nos queda" $env'_p \vdash \langle call\ fact, s'''\rangle \to s''$, lo que nos lleva a $env'_p \vdash \langle F, s'''\rangle \to s''$. Y como $s'''x < sx = n$ podemos aplicar la h.i para tratar de llegar a $s''\,fx = (sx)!$, pero "nos damos cuenta" de que "sólo llegaríamos" a $s''\,fx = (sx-1)!$, que obviamente NO es lo que queríamos. Pero no es que "esté mal" sino que debemos cambiar la tesis (y por tanto la hipótesis) de inducción, que se convierte en $s''\,fx = s'''\,fx * (s'''x)!$

  y ahora sí obtenemos la prueba por inducción deseada ya que tomando $C_1 \equiv (fx := fx * x\ ;\ x := x-1)$ obtenemos que $s'''\,fx * (s'''x)!$ es un "invariante" ya que $x! = (x-1)! * x$.

1.2) Evidentemente utilizaremos la correspondiente regla [call^{rec}_{ns}]. Bajo alcance dinámico "en cuanto" env'_p acumula el significado de fact, como F, este "pervive" durante todas las llamadas recursivas. Esto no sería obviamente así si tuviesemos alcance estático y utilizáramos [call_{ns}], ya que la llamada recursiva utilizaría el entorno env_p, que a saber lo que decía de fact. Pero al pasar a usar [call^{rec}_{ns}] cada llamada (recursiva) "redefine" en env_p el significado de fact como F, y aunque se "arrastra" env_p como entorno para las llamadas dentro de F, en el caso de esas llamadas recursivas nos encontramos con F como "significado" de fact, y por tanto no se utilizaría nunca $env_p\ p$.

¡Cuidado! En las reglas de Table 3.4. env_p y env'_p van justo al revés que en mi notación aquí. Aunque una vez más os recuerdo que los nombres no significan NADA, y por tanto podríamos "eliminar este problema" simplemente intercambiando $env_p\ p$ y $env'_p\ p$ en "todas las partes" de estas reglas, la razón que "justifica" la "discrepancia" entre las notaciones es que en las reglas se define el significado de cada instrucción (por ejemplo call p) partiendo del entorno vigente env_p, "utilizando" un entorno anterior env_p, mientras que mis explicaciones "intuitivas" siguen "el flujo de ejecución", por lo que env_p era el "entorno inicial" y env'_p el obtenido tras declarar env'_p.

Una vez hemos "conseguido" que F sea el "significado" de todas las llamadas recursivas, el "cálculo" de la semántica sería idéntico a cuando utilizamos alcance dinámico, y por tanto obtendríamos del mismo modo el mismo resultado.

2.1) En realidad lo que se nos pide es calcular el efecto que tendrían ulteriores llamadas a fact en el bloque en que está declarado (pues si no "éste" fact no podría ser "llamado").
La declaración de fact volvería a ligar fact a su cuerpo F en el correspondiente entorno env'_p. Y ahora una llamada con $s\,x \in \mathbb{N}$ nos exigiría aplicar $[\text{block}_{ns}]$ que genera s'' con $s''xl = s\,x$ tras lo que si $s\,x < 2$ la aplicación de $[\text{if}^{ff}_{ns}]$ nos llevaría al estado "final" s' con $s'\,fx = 1 = (s''\,xl)!\ (= (s\,x)!)$ lo que constituiría los dos casos bases de nuestra inducción.

En efecto, cuando $sx = n \geq 2$ la aplicación de $[\text{if}^{tt}_{ns}]$ genera
s''' con $s'''x = sx - 1$ ($< sx$), por lo que podremos asumir (por inducción sobre el valor de x en el estado vigente) que la llamada a fact "computa" en fx el valor $(s'''x)! = (n-1)!$
Con lo que aplicando $[\text{ass}_{ns}]$ terminamos la "ejecución" del bloque interno obteniendo s' con $\boxed{s'fx = s'''fx * s''xl = (n-1)! * n = n!}$

Observación: Es el hecho de que la recursión no sea final lo que hace que esta inducción sea mucho más sencilla que en el Ejercicio 1.

2.2) De nuevo, siempre que apliquemos $[\text{call}^{rec}_{ns}]$, la semántica de F2 bajo alcance estático va a coincidir con la dinámica ya que NO hay llamadas a ningún otro procedimiento, con lo que toda llamada recursiva a fact se hará con un entorno en el que fact tiene como "valor" su cuerpo F a ejecutar sobre el entorno de procedimientos "original". Con lo que la demostración de que fact calcula $(sx)!$ en fx sería idéntica a la del apartado anterior.

2.3) Asumiendo que la memoria sobre la que trabajamos "funciona bien" (o sea se van generando siempre posiciones nuevas) y que nuestras dos variables "globales" x y fx "no comparten" memoria ($env_V\, x \neq env_V\, fx$), de nuevo todo funciona exactamente igual que bajo alcance dinámico ya que la única variable adicional que se utiliza en fac se crea (en el bloque) y NO se utiliza en las llamadas recursivas anidadas porque en ellas se utiliza la "nueva" variable xl declarada al principio del bloque que constituye el cuerpo F de fact. De modo que "intuitivamente" la "pila virtual" que generaban los "bloques anidados (vía recursión) con la regla para alcance dinámico "iría en paralelo" con la "pila de direcciones virtual" que sigue generando la regla [block_{ns}] para alcance estático (<u>Nota</u>: recuérdese que este "efecto pila" lo veíamos "al salirnos del bloque" al aplicar [comp_{ns}] para pasar a "la instrucción siguiente", en la Table 3.3.

<u>Observación</u>: En realidad este fenómeno de "pila virtual" se da en los tres apartados, pero al tiempo en ninguno de ellos es necesario "visualizarlo" en la demostración por inducción, ya que en cada paso inductivo se trabaja sólo con la "primera" $x\ell$ "en la base de la pila", sobre la que llamada call frac "generaría" y luego "vaciaría" la pila (¡ por hipótesis de inducción ! (de forma implícita)), por lo que el razonamiento inductivo sólo ve esta (última) $x\ell$.

3.1) La inicialización del bloque genera un estado, al que llamaremos s, en el que $s\ xc = s\ x = n \in \mathbb{N}$ y $s\ fx = 1$.
De nuevo consideramos el efecto que tendría una llamada call fact_raro
Aplicando [call^{rec}_{ns}] conllevaría la "ejecución" del cuerpo de la declaración de fact_raro, Fraro, que tendríamos en env_p fact-raro.
Empezaríamos pasando a s_1 con $s_1\ fx = s\ fx * s\ x = 1 * n = n$.
Ahora, si $s_1\ xc = s\ x = n < 2$, [if^{ff}_{ns}] nos hace terminar con $s' = s_1$
Mientras que si $n \geq 2$, [if^{tt}_{ns}] nos lleva a aplicar [block_{ns}] sobre el bloque interno B, empezando por pasar a s_{B1} con $s_{B1}\ x = n-1$ y luego a s_{B2} con $s_{B2}\ xc = n-1$, para llamar "recursivamente" a fact_raro que asumiremos que "produce" un estado s_{B3} (que cuando lleguemos al final de Fraro esta vez, trataremos de ver qué cumpliría, para intentar luego demostrar por inducción

que ello en efecto es así).

Ahora, para terminar la ejecución de Fraro nos queda la última asignación, que "volvería a multiplicar" fx por "algo", ¿por cuanto? Procede mirar un ejemplo sencillo, pero en el que lleguemos aquí. Obviamente, el más sencillo es $n=2$.

Según hemos visto, $s_{B2}\,xc = 1$ y $s_{B2}\,fx = 2$, al tiempo que $s_{B2}\,x = 1$. De modo que la llamada call fact va a comportarse como una llamada con $n=1$, que hemos visto que "mantenía" el valor de fx, y también los de x y xc. Así que en este caso $s_{B3} = s_{B2}$, y la asignación final a fx "vuelve a multiplicar fx por 2 (ya que $s_{B3}\,x = 1$), con lo que obtendríamos $s'\,fx = s_{B3}\,fx * 2 = 2*2 = 4$.

Así que esto tiene "toda la pinta" de que estamos haciendo dos veces cada multiplicación necesaria para calcular n!, con lo que obtendríamos siempre $\boxed{s'\,fx = (s\,x)!^2 = n!^2.}$

Nos queda comprobar por inducción que este es el caso.

Vamos a necesitar además asegurarnos de que siempre $s_{B3}\,x = s_{B2}\,x$, para que así $s_{B3}\,x + 1 = (s_{B2}\,xc - 1) + 1 = s_{B2}\,xc$

Es aquí donde va a jugar un papel la "astuta redeclaración" de x con la que comienza el bloque B. Es fácil ver que call fact_raro mantiene el valor de la x ya que F.raro no asigna nunca ningún valor a "esa" x ya que la inicialización de x en B corresponde a "una nueva x", de manera que la combinación de [blockns] y [compns] hace que "al salir" de B se "recupere" el valor de $s_{B2}\,x$, aunque "dentro de B" haya ido cambiando.

Completemos con detalle la demostración por inducción:

Afirmamos que fact_raro transforma s en s' con

$$\left\{\begin{array}{l} s'\,fx = (s\,fx) * (s\,xc)!^2 \text{, de modo que, en particular} \\ (\text{siempre que } s\,x = s\,xc) \end{array}\right.$$

una llamada con $s\,fx = 1$ "calculará $s'fx = (s\,xc)!^2 \, (= (s\,x)!^2)$
Repasando la distinción de casos que hicimos antes. En efecto, cuando $s\,x_1 = 1$, obteníamos $s'fx = s\,fx = s\,fx * 1!^2$.
Dejamos ahora fuera el caso $s\,x = 0$ en el que obtendríamos $s'fx = 0 \neq s\,fx \cdot 0!^2$ (ya que "tomamos" $0! = 1$), pero a este caso no llegaremos nunca ya que la base de la inducción va a ser $s\,x = 1$.
Nos queda el "caso inductivo" $s\,x \geq 2$, y como vimos $s_1\,fx = s\,fx * s\,x$, lo que nos llevará después a

$$\begin{cases} s_{B2}\,fx = s\,fx * s\,x \ ; \ s_{B2}\,x = s\,xc - 1 \ (= s\,x - 1) \ ; \\ s_{B2}\,xc = s\,xc - 1 \ (= s\,x - 1) \end{cases}$$

Así que podemos aplicar la h.i. al hacer la llamada call fact_raro, ya que $s_{B2}\,x = s_{B2}\,xc < s\,x$. Ello nos lleva a
$s_{B3}\,fx = s_{B2}\,fx * (s_{B2}\,x)!^2$; $s_{B3}\,x = s_{B2}\,x$; $s_{B3}\,xc = s_{B2}\,xc$.
Y finalmente aplicamos [assns] para llegar a $s'\,fx = s_{B3}\,fx * (s_{B3}x + 1)$
Aplicamos todas las igualdades anteriores para escribir $s'fx$ en función de $s\,fx$, obteniendo $s'fx = (s\,fx * s\,x) * (s\,x - 1)!^2 * ((s\,x - 1) + 1)$
que "simplificado" nos da

$$\boxed{s'\,fx = s\,fx * (s\,x)!^2}$$ c.q.d.

3.2) Me voy a centrar en lo que en este caso va a generar las diferencias entre el alcance estático y el dinámico. Se trata, naturalmente del uso de x en la primera instrucción de Fraro. Bajo alcance estático el env_V que maneja fact_raro es el previo, así que cuando llegamos a esa primera instrucción al hacer cualquier llamada a fact_raro estaríamos utilizando "la x original" que se corresponde con la que "aparece" en el estado original s. Así que si llamamos a fact_raro con $s\,x = n \geq 2$ en cada llamada recursiva "anidada" vamos a ir multiplicando $s\,fx$ "siempre" por $s\,x$.

En cambio el flujo de llamadas recursivas "anidadas" está controlado por la "variable global" xc, por lo que se van a seguir anidando sx llamadas en las que "en cambio" se está "utilizando" la "variable local" x declarada en B que ya vimos que (intuitivamente) se maneja de igual forma bajo alcance estático que bajo alcance dinámico. En definitiva el "flujo global" va a seguir siendo el mismo que bajo alcance dinámico, pero como las multiplicaciones en la primera instrucción de Fraso son siempre por sx, lo que obtendríamos partiendo de $s\,fx = 1$ sería

$s'\,fx = (sx)! * (sx)^{sx}$, y si $s\,fx$ fuera arbitrario

$\boxed{s'\,fx = s\,fx * (sx)! * (sx)^{sx}}$, lo que se "comprobaría" formalmente viendo que la aplicación de las nuevas reglas genera "el mismo cómputo que antes", pero con el citado cambio en el factor de la primera multiplicación que es el que va a generar el factor $(sx)^{sx}$ en el "resultado" $s'\,fx$.

<u>Observaciones</u>: Bajo alcance dinámico hemos visto que las dos variables x y xc están "completamente acopladas", por lo que podríamos decir que xc "nos sobra", lo cual ciertamente es verdad. Podemos por tanto eliminar por completo xc sustituyendola donde aparece por x, y eliminar la instrucción $xc := xc - 1$ para obtener un bloque F3' más sencillo que sería en efecto equivalente.
En cambio bajo alcance estático al tener las dos variables ha sido fácil mostrar la diferencia con el caso dinámico al "desacoplarse" la variable x "de fuera de B". Y esta diferenciación entre x y xc es también imprescindible si queremos conseguir bajo alcance dinámico una "función" con el mismo efecto que F3. Si ahora consideráramos F3' el efecto sería "nefasto" pues "todo el tiempo" inicializaríamos la variable "local" x con el mismo valor $sx - 1$, y a partir de ahí <u>ninguna llamada</u> con $sx \geq 2$ <u>terminaría jamás</u>.