REFLEXIONANDO SOBRE LA SEMANTICA DEL PARALELO

Ejercicio 1 : Demostrar que $(S_1 \text{ par } S_2, s)$ y $(S_2 \text{ par } S_1, s)$ generan los mismos cómputos (módulo la reversión de los argumentos de par en la instrucción de cada configuración).

Ejercicio 2 : Demostrar que $(S \text{ par skip}, s)$ y (S, s) generan "prácticamente" los mismos cómputos (módulo la "extensión" de S a S par skip en los campos instrucción de las configuraciones donde "converja", y salvo la introducción de un paso adicional de cambio de instrucción, pero no de estado, en el primer caso).

Corolario 1 : S par skip, skip ; S y S ; skip, son equivalentes.

Ejercicio 3 : Considerad cualquier S de While equivalente a skip (aprovechad para precisar qué propiedad semántica caracteriza a estas sentencias, ¡pero no intentéis describir "extensionalmente" el conjunto que forman, pues es imposible!, ¿por qué?). Estudiar si S par S' será siempre equivalente a S par S' y a S' par S. ¿Seguirán siendo S y skip equivalentes tras extender While con par, y definir "adecuadamente" la correspondiente noción de equivalencia semántica? Tras todo esto se concluye una "nada agradable" "propiedad negativa" de la semántica operacional de par (considerando como tal la función S_{sos}) o equivalentemente de la equivalencia semántica introducida aquí arriba. ¿Qué propiedad? ¿Habría alguna forma de "corregir" esta "desagradable" situación?

Corolario 2 : Es imposible definir una semántica de paso grueso que cubra el paralelo, manteniendo como "valores producidos" los estados. (Para nota : Sin embargo sí que podríamos rizar el rizo y definir una semántica tal que "devolviera" valores mucho más complejos. ¿Cuáles? ¿cómo?)

Ejercicio 4 : Diremos que S_1 y S_2 son independientes cuando S_1 par S_2 sea equivalente a $S_1 ; S_2$ y a $S_2 ; S_1$. Buscar un criterio "razonable" (o sea, sencillo y al tiempo "bastante amplio") que caracterice una familia Ind de pares de procesos independientes. Demostrar que, en efecto, los pares de Ind están formados por procesos independientes.

Ejercicio 5 : Encontrar S_1 y S_2 tales que $\forall$ s $S_1 ; S_2$ y $S_2 ; S_1$ desde s terminan siempre, y sin embargo S_1 par S_2 podría no terminar.

Ejercicio 6 : Demostrar que si S_1 puede no terminar entonces S_1 par S_2 también puede no terminar. Encontrar algún ejemplo sencillo de S_1 tal, y una S_2 sencilla de manera que toda ejecución de S_1 par S_2 que nos deje como resto por ejecutar un cierto S_1' (pues la ejecución de S_2 se terminó), siempre dará lugar a cómputos completos terminados (o sea, S_1' no puede no terminar).

Breve introducción a la sincronización

Podríamos sincronizar la ejecución de los dos argumentos de un paralelo vía una variable turno $\in \{1,2\}$. Cuando queremos "parar" la ejecución de S_i introducimos while turno $\neq i$ do skip. Cuando un proceso quiere "pasarle el turno" al otro hace turno := 3 - i y se "espera a su turno", como dijimos antes. Naturalmente, bajo nuestra semántica además de los cómputos deseados se generan otros con "espera ocupada" infinita.

Como alternativa podríamos añadir una nueva instrucción de espera "en reposo" wait P, que no hace nada, hasta que P sea cierto, momento en que podría comportarse como skip. Definir su semántica.