Formalización de la propiedad de Progreso para el $\lambda-$cálculo simplemente tipado

Cristian Sottile

23 de noviembre de 2023

1 Informe

Continué avanzando con la lectura del libro de PLFA hasta terminar la primera parte sobre *logical foundations*. Luego, como desde un principio tenía intenciones de aprender las técnicas clásicas de formalización de cálculo lambda en Agda, no intenté descularlas por mi propia cuenta, sino que continué leyendo la segunda parte del libro sobre *programming language foundations*. En particular me interesaba la parte de índices de de Bruijn, sobre los que escuché muchas veces pero nunca indagué. Pero una vez resueltos los problemas de substitución, la propiedad de progreso se cae de madura. Consideré que más allá de las horas dedicadas, no estaba cumplido el objetivo propuesto por el curso de intentar pensar cómo definir y demostrar propiedades de una manera autónoma, ya que si bien el curso tiene ejercicios que me resultaron complicados, la propuesta es siempre guiada y las definiciones son casi siempre provistas. Así que intenté lo siguiente:

1. Formalizar un teorema de un paper que presentamos este año
2. Darle alguna vuelta más a la propiedad de progreso

En las siguientes secciones describo cómo procedí con ambas ideas.

2 Formalizando la medida $\mathcal{W}$

El trabajo mencionado es "*Two decreasing measures for simply typed lambda calculus*", en donde construimos dos medidas decreciente para los términos del cálculo lambda simplemente tipado. Para ello definimos un cálculo auxiliar $\lambda^{\mathtt{G}}$ en donde no hay borrado: los argumentos "descartados" se guardan en una estructura auxiliar que nombramos memoria. Me enfoqué en una de estas dos medidas, llamada $\mathcal{W}$, cuya intuición es que a cada paso de reducción se agranda la memoria. Luego dados dos términos t y s tales que t reduce a s en un paso R, la forma normal de t tendrá más memorias que la de s. Como es posible obtener la forma normal en $\lambda^{\mathtt{G}}$ mediante dos operaciones de "simplificación" definidas por recursión sobre los términos y los grados de los redexes, definimos la medida $\mathcal{W}$ para t como la cantidad de memorias de la forma normal de t.

Para formalizar tomé las definiciones con tipos intrínsecos con las que había estado trabajando, di definiciones similares para las construcciones y operaciones nuevas del cálculo auxiliar $\lambda^{\mathtt{G}}$, postulé las propiedades del cálculo, di las definiciones correspondientes a la medida decreciente $\mathcal{W}$, y finalmente enuncié el teorema:

$$t \to s \implies \mathcal{W}(t) > \mathcal{W}(s)$$

Intenté continuar demostrando el teorema y los lemas postulados, y rápidamente me di cuenta de que aún no adquirí la independencia suficiente como para resolver las dificultades que iban surgiendo.

Por ejemplo:

1. $\lambda^{\mathtt{G}}$ introduce un nuevo constructor de términos que agregar una memoria a cualquier término: $t\langle s\rangle$. Luego un término puede tener varias memorias. En el trabajo presentamos la regla beta "a distancia", donde nos salteamos las memorias para efectuar el paso de reducción. Esto nos permite, dada cualquier aplicación, decir si constituye un redex y particular cuál es su grado; algo que necesitamos para las operaciones de simplificación. No me resulta claro cómo dar esta regla beta a distancia en Agda. Pensé en algo como una lista para las memorias, pero los términos pueden tener distinto tipo y no sé si eso no sería un problema para el tipado.
2. Di bastantes vueltas con cómo ver que un término es o no una abstracción, que es algo relevante para los lemas de simplificación, al principio definiendo formas de comparar términos, hasta que recordé que en varias partes en el libro Wadler sobrecarga constructores para indicar que cumplen ciertas propiedades, así que definí el tipo de dato is-λ cuyo único constructor coincide con el de términos.
3. Dado que queremos probar una propiedad del cálculo lambda simplemente tipado y utilizamos el cálculo auxiliar, hacen falta algunas propiedades de conversión de términos entre un sistema y otro. Uno de ellos es que $(to_{\mathtt{G}}t)[(to_{\mathtt{G}}s)/x]$ es igual a $to_{\mathtt{G}}(t[s/x])$. Intenté demostrarlo pero tuve problemas con las premisas para la substitución y desistí.

3 Más progreso

Con las definiciones del libro se prueba progreso para un cálculo con naturales, punto fijo y estrategia *weak call-by-value*. Pensé en utilizar el cálculo lambda simplemente tipado, sacar la estrategia y considerar reducción *strong*. Los primeros dos items fueron triviales, el segundo no porque la reducción bajo lambda extiende el contexto y no puedo usar la hipótesis inductiva. Tuve la idea de agregar un parámetro adicional en una función auxiliar donde, para cada variable del contexto, construyo una cadena de reducciones que me permite llegar hasta el término que la reemplazará. Si la variable está (o puede estar) en una posición de aplicación, entonces el siguiente paso será su cadena asociada que la reemplazará por una función y efectuará un paso beta. Si la variable no está en una posición de aplicación, entonces no hay problema. Si todas las variables que estén en posición de aplicación en una lambda tienen una cadena de reducción asociada para asegurar progreso, entonces esperaría poder demostrar la propiedad. Esto es una idea y no llegué a intentar materializarla, la estuve pensando hoy mismo. Una duda que tuve es qué tipo tendría una lista con cadenas de reducción que involucran distintos términos.

4 Entrega

Entrego los archivos correspondientes a:

1. Las definiciones, las propiedades y (casi todos los) ejercicios de los capítulos 1 a 13 del libro PLFA
2. La formalización de la prueba de progreso (que a la vez subsume a la de preservación de tipos) del cálculo lambda simplemente tipado a la Church con reducción weak y sin estrategia
3. El intento de formalización de la medida $\mathcal{W}$