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

Cristian Sottile

20 de octubre de 2023

1 Avances

En este mes estuve siguiendo el libro PLFA de Wadler para interiorizarme con el asistente. Comparto el repositorio en el que estuve copiando las definiciones e implementando los ejercicios propuestos: `https://github.com/cfsottile/plfa`. Estoy en el capítulo de Cuantificadores, me quedan los capítulos de Decidibilidad y de Listas, y luego planeo empezar con la implementación del $\lambda-$cálculo simplemente tipado para poder probar la propiedad de progreso. No me topé con mayores dificultades aprendiendo Agda. Sí tuve que detenerme para intentar entender con cierta profundidad algunos aspectos, como el caso de expresar `rewrite` en términos de `with` (que finalmente no comprendí), la prueba de simetría de la igualdad de Leibniz que me desacomodó bastante las ideas, y algunos ejercicios de la parte de negación que no me salieron rápido. En general le dedico bastante a hacer los ejercicios así entiendo mejor, por eso avanzo lento. Anoté dos dudas que aprovecho para copiar a continuación.

2 Consultas

2.1 Ignorando parámetros sin éxito

Aclaro que usé este formato inusual de definir las funciones en el `where` solo para ver si funcionaba.

```
1  ⊎-distrib-× : ∀ {A B C : Set} → (A × B) ⊎ C ≲ (A ⊎ C) × (B ⊎ C)
2  ⊎-distrib-× {A} {B} {C} = record { to = to' ; from = from' ; from∘to = from∘to' }
3    where
4      to' : (A × B) ⊎ C → (A ⊎ C) × (B ⊎ C)
5      to' (inj₁ ⟨ a , b ⟩) = ⟨ inj₁ a , inj₁ b ⟩
6      to' (inj₂     c    ) = ⟨ inj₂ c , inj₂ c ⟩
7      from' : (A ⊎ C) × (B ⊎ C) → (A × B) ⊎ C
8      from' ⟨ inj₁ a , inj₁ b ⟩ = inj₁ ⟨ a , b ⟩
9      from' ⟨ _ , inj₂ c ⟩ = inj₂ c
10     from' ⟨ inj₂ c , _ ⟩ = inj₂ c
11     -- ¿Por qué me marca esto? Ah, debe ser porque no sabe si _ es inj₁ o inj₂. Y
12     -- cuando especifico inj₁ a , inj₂ c ... Ah, no. Porque entonces es un
13     -- problema también no saber si el _ de ⟨ inj₂ c , _ ⟩ es inj₁ o inj₂.
14     -- ¿Quizá hay algo con que el segundo elemento pueda depender del primero?
15     -- PREGUNTAR A MIGUEL
16     from∘to' : (x : (A × B) ⊎ C) → from' (to' x) ≡ x
17     from∘to' (inj₁ ⟨ a , b ⟩) = refl
18     from∘to' (inj₂     c    ) = refl
```

2.2 Normalización de goals pero no de términos

```
3   -- Preguntar a Miguel: cuándo reduce λx.fx a f como para dar el goal f ≡ f
2   -- Considerando que pedirle a Agda que normalice λ x → f x da el mismo término
1   η-→ : ∀ {A B : Set} (f : A → B) → (λ (x : A) → f x) ≡ f
128 η-→ f = {}0
1
1  - 5.5k 06-Connectives.agda  Agda  ⓐⓨⓚ  Git:main Mod
?0 : f ≡ f
```