گزارش پروژه اثبات قضیه Church-Rosser در coq

محسن لیاقت

۲ مرداد ۱۴۰۲

چکیده

در حساب λ بدون تایپ در برخی موارد می تواد یک ترم را به چند روش کاهش داد قضیه چرچ راسر بیان می کند که اگر یک ترم A در یک توالی از کاهش ها به ترم B و در یک توالی از کاهش ها به ترم C کاهش یابد ترم D به گونه ای موجود است که ترم های B و C پس از چند بار کاهش به ترم D کاهش یابند. این قضیه در سال ۱۹۳۶ توسط Church و Rosser اثبات شد.

ما در این پروژه سعی کردیم که اثبات فرمال این قضیه در coq بنویسیم.

این اثبات در چند فایل تقسیم شده تا مدیریت لم های مورد نیاز ساده تر باشد. در ادامه به تشریح هر یک از کار های انجام شده در این فایل ها می پردازیم.

استراتژی کلی برای اثبات این است که ابتدا اثبات کنیم یک محمول دو موضعی وجود دارد که reflexive transitive closure آن با β-reduction معادل است سپس اثبات کنیم که diamond property تحت reflexive transitive closure حفظ می شود و پس از آن اثبات کنیم که محمول معرفی شده دارای خاصیت diamond property است. (در بخش BR.v این خاصیت تعریف شده و اثبات هایی درباره آن آمده است.)

فهرست مطالب

۱.۰ کتاب خانه های استفاده شده در این پروژه :

Arith: در این کتابخانه یک سری ماژول برای عملیات های ریاضی تعریف شده که در هر یک از این ماژول ها تعریف عملیات مربوطه و اثبات برخی قضایای مربوطه آمده است.

Lia: این کتابخانه برای استفاده از تاکتیک lia استفاده می شود که این تاکتیک وظیفه خودکار کردن برخی اثبات های بدیهی اولیه را در باره اعداد طبیعی و صحیح بر عهده دارد.

List: این کتابخانه تعاریف مربوط به لیست و یک سری از توابع از لیست ها و برخی اثبات ها پیرامون این تعاریف را دارد

ListNotations: این کتابخانه شامل تعریف نوتیشن های استاندارد مورد استفاده در لیست ها است.

RelationClasses: این کتابخانه شامل تعریف برخی از کلاس های روابط و اثبات هایی پیرامون این کلاس ها است. برای مثال در این کتابخانه روابط هم ارزی و reflexive تعریف شده اند که در این پروژه از آن ها استفاده شده است.

Relation_Operators: این کتابخانه شامل تعریف یک سری اپراتور روی روابط و برخی اثبات های مربوط به آن ها است. برای مثال اپراتور های transitive closure و transitive reflexive closure در این کتابخانه تعریف شده اند.

۲.۰ Support.v

در این فایل قضایای مورد استفاده در باره منطق، لیست و اعداد طبیعی است.
در این فایل ابتدا یک لم بدیهی اثبات شده که بیان می کند بیشینه دو عدد با یکی از آن دو برابر است. سپس باتوجه به اینکه در COQ یک تابع existsb وجود دارد که دو پارامتر ورودی می گیرد که یکی از آنها یک تابع از تایپ دلخواه به بولین و دیگری یک لیست است و چک می کند که اگر مقدار تابع به ازای یکی از اعضای لیست true باشد مقدار true را بر می گرداند در غیر این صورت مقدار false را بر می گرداند. یک قضیه اثبات کردیم که این تابع مقدار false را بر می گرداند اگر و تنها اگر هیچ عضوی در لیست تابع ورودی اش را ارضا نکند به عبارت دیگر این تابع به ازای تابع f لیست l مقدار false را بر می گرداند اگر و تنها اگر گزاره زیر ارضا شود.(In یک محمول دو موضعی است که بیان می کند یک عضو در یک لیست قرار دارد.)

$$\neg \exists x, \mathrm{In}\, x\, l \wedge f\, x = true$$

سپس قضیه دیگر اثبات شده که بیان می کند بیشینه مقدار موجود در یک لیست بزرگتر یا مساوی همه اعضای آن لیست است.
سپس اثبات کردیم که هرگاه اعدد طبیعی مانند n داشته باشیم که بدانیم از همه اعضای یک لیست l بزرگ تر است آنگاه $\neg\mathrm{In}\ n\ l$ قابل اثبات است.
پس از آن نیز استقرا قوی روی اعداد طبیعی را اثبات کردیم.

۳.۰ BR.v

در این فایل یک سری قضایای مورد نیاز را در باره محمول های دو موضعی دلخواه اثبات کردیم.
در این فایل ابتدا تعریف diamond property را ارائه دادیم که در واقع یک محمول یک موضعی است که پارامتر آن یک محمول دو موضعی R روی تایپ دلخواه A است و این محمول (diamond property) برقرار است اگر و تنها اگر داشته باشیم .

$$\forall xyz : A, Rxy \to Rxz \to \exists w : A, Ryw \wedge Rzw$$

سپس تلاش کردیم تعریفی از transitive reflexive closure ارائه دهیم چون نیاز بود که طول دنباله مربوط به هر یک از اعضای آن را داشته باشیم تا بتوانیم روی آن استقرا بزنیم پس این تعریف را با کمک limited transitive reflexive closure تعریف کردیم که تعریف آن در همین فایل پیش از تعریف trans_clos(transitive reflexive closure) آمده است.

(limited transitive reflexive closure)limited_trans_clos: اپراتور دو موضعی که پارامتر اول آن یک محمول ۲ موضعی R روی تایپ A و یک عدد طبیعی n است و خروجی آن یک محمول دوو موضعی است که تنها زمانی برای ترم های x و y از تایپ A برقرار است که فرمول زیر اثبات شود.

$$\exists a : \mathbb{N} \to A, a0 = x \wedge an = y \wedge (\forall i, i < n \to R(a_i)(a_{(Si)})).$$

به عبارت دیگر تنها زمانی بر قرار است که دنباله ای به طول n موجود باشد که برای هر دو عضو متوالی آن این محمول برقرار شود و همچنین عضو اول آن x و عضو n ام آن y باشد.

trans_clos: یک اپراتور یک موضعی روی محمول های دو موضعی است که به صورت زیر تعریف می شود.

$$\text{trans\_clos}Rxy := \exists n : \mathbb{N}, \text{limited\_trans\_clos}Rnxy$$

پس از این تعاریف یک سری قضیه و لم اثبات می شود که به اثبات حفظ شدن diamond property تحت trans_clos ختم می شوند. این لم ها و قضایا به ترتیب به صورت زیر هستند.

- برای محمول دلخواه R که دارای diamond property است. و عدد طبیعی n داریم :

$$\forall xyz : A, Rxy \to \text{limited\_trans\_clos}Rnxz \to \exists w : A, Rzw \wedge \text{limited\_trans\_clos}Rnyw$$

اثبات این حکم با استقرا رو n است که برای $n = 0$ ساده است و برای $n = Sn'$ باید از فرض استقرا و diamond property استفاده کرد.

- برای محمول دلخواه R که دارای diamond property است. و عداد طبیعی n و m داریم :

$$\forall xyz : A, \text{limited\_trans\_clos}Rnxy \to \text{limited\_trans\_clos}Rmxz \to$$

$$\exists w : A, \text{limited\_trans\_clos}Rmyw \wedge \text{limited\_trans\_clos}Rnzw$$

که اثبات این حکم نیز با استقرا روی n و استفاده از قضیه قبلی است.

- این قضیه بیان می کند که trans_clos حافظ diamond property است. که اثبات آن با توجه به لم قبلی ساده است.

سپس یک محمول دو موضعی به نام subeqrel معرفی می کنیم که برای بیان subrelation به کار می رود و روی روابط دو موضعی تعریف می شود. تعریف دقیق تر این محمول به صورت زیر است.

$$subeqrel R R' := \forall xy : A, Rxy \to R'xy$$

از این به بعد این محمول را با $\subseteq$ نمایش می دهیم.
حال اثبات می کنیم که این محمول تحت trans__clos حفظ می شود. که اثبات آن با استقرا روی طول دنباله مربوطه در limited__trans__clos است.
پس از ان اثبات می کنیم که برای دو محمول دلخواه R و R' برای اثبات اینکه $\text{trans\_clos} R \subseteq \text{trans\_clos} R'$ باشد کافی است ثابت کنیم $R \subseteq \text{trans\_clos} R'$ است.
در نهایت برای راحت تر شدن اثبات در برخی دیگر از قضایا اثبات کردیم که تعریف ما از trans__clos تعریف کوک از clos__refl__trans را شامل می شود. که این تعریف بیان می کند برای محمول دلخواه R و برای x ، y دلخواه $\text{clos\_refl\_trans}\ Rxy$ بر قرار است اگر یکی از شروط زیر برقرار باشد.

- $x = y$
- Rxy
- $\exists z, \text{clos\_refl\_trans}\ Rxz \wedge \text{clos\_refl\_trans}\ Rzy$

۴.۰ Term.v

در این فایل تعریف ترم ها در حساب لامبدا بدون تایپ و توابع مورد نیاز از ترم ها به دیگر تایپ ها را ارائه می کنیم. سپس چند مورد را درباره آن ها اثبات می کنیم.

ابتدا atomic term را به صورت عدد طبیعی تعریف می کنیم این کار باعث می شود که تعداد شمارای نامتناهی atomic term داشته باشیم. سپس تعریف ترم را مطابق با تعریف کتاب ارائه دادیم پس از آن تابعی تعریف کردیم که لیست همه متغیر های آزاد یک ترم را بازگرداند این تابع به صورت بازگشتی تعریف شده که در صورتی که ترم تنها یک متغیر باشد یک لیست تک عضوی شامل آن متغیر را خروجی می دهد و در صورتی که ترم به صورت application دو ترم روی هم باشد لیست متغیر های آزاد آن ها را با هم concat می کند و در صورتی که ترم به صورت abstraction یک ترم باشد متغیری که bound شده را از لیست متغیر های آزاد آن را حذف می کند و نتیجه را خروجی می دهد. نام این تابع را free__vars گذاشتیم.
سپس تابع is__free را به این صورت تعریف کردیم که دو ورودی بگیرد که یکی از آن دو ترم T باشد و دیگری atomic term x و نتیجه محاسبه عبارت زیر را خروجی دهد.

existsb (fun x0 => x =? x0) (free_vars T)

fun x0 => x =? x0 تابعی است که از الگوریتم چک کردن تساوی دو عدد طبیعی استفاده می کند و چک می کند که آیا متغیرش با x برابر است یا نه.
سپس لمی را اثبات می کنیم که نشان می دهد اگر متغیری از همه متغیر های آزاد یک ترم بزرگ تر باشد آن متغیر درون آن ترم آزاد نیست.
پس از آن تعریف ارتفاع یک ترم را ارائه می دهیم که در واقع بیانگر ارتفاع درخت متناظر با آن ترم است و به صورت بازگشتی محاسبه می شود، به این صورت که ارتفاع متغیر ها صفر است ارتفاع abstraction یک ترم یکی واحد بیشتر از ارتفاع خود آن ترم است و ارتفاع application دو ترم یک واحد بیشتر از ماکسیمم ارتفاع آن دو ترم است.
این تعریف برای استقرا بعدا استفاده خواهد شد.

۵.۰ UCLP.v

این فایل شامل محمول هایی روی ترم ها و اثبات قضایای مربوط به آن ها است که در نهایت تلاش داشتیم اثبات قضیه چرچ راسر را نیز در فایل بیاوریم که متاسفانه موفق نشدیم.

در ابتدا جانشینی را به عنوان یک محمول ۴ موضعی تعریف می کنیم دلیل اینکه جانشینی را به عنوان محمول تعریف می کنیم این است که تعریفی که از جانشینی در کتاب ها آمده خوش تعریف نیست مگر آنکه به جای ترم ها از کلاس های هم ارزی ترم ها با توجه به α-equivalency استفاده کنیم و کوک (تا آنجا که من بلدم) چنین چیزی را نمی پذیرد از طرف دیگر اگر بخواهیم عضوی را از هر کلاس به صورت deterministic انتخاب کنیم کوک باز هم قبول نخواهد کرد زیرا برای کوک به دلیل انجام renaming قابل تشخیص نیست که پارامتری که این تابع بازگشتی دریافت می کند کوچک می شود و در نهایت محاسبه پایان می پذیرد.
از طرف دیگر با توجه به اینکه جانشینی با توجه به کلاس هم ارزی ترم ها در رابطه α-equivalency خوش تعریف است و از جانشینی در کاهش ترم ها استفاده می شود پس کل قضیه چرچ راسر با توجه به این کلاس های هم ارزی برقرار است پس ایرادی ندارد که ما تعریف خود را از جانشینی کمی تغییر دهیم به طوری که نتیجه آن عضوی از همان کلاس هم ارزی باشد که در تعریف های معمول است. در نهایت با توجه به همه این توضیحات جانشینی را به صورت زیر تعریف می کنیم.

```
Inductive substitution : atomic_term -> term -> term -> term -> Prop :=
  | subst_var1 (x : atomic_term ) ( T : term ) : substitution x x T T
  | subst_var2 { x y : atomic_term } ( T : term ) : x <> y -> substitution x y T y
  | subst_appl { x : atomic_term } { T1 T2 T1' T2' T : term } :
    substitution x T1 T T1' -> substitution x T2 T T2' -> substitution x ( appl T1 T2 ) T ( appl T1' T2' )

  | subst_abst1 { x : atomic_term } { T Tx : term } : substitution x ( abst x T ) Tx ( abst x T )

  | subst_abst2 { x y : atomic_term }{ T Ty T' : term } :
     x <> y -> is_free Ty x = false -> substitution y T Ty T' -> substitution y ( abst x T ) Ty ( abst x T' )

  | subst_abst3 { x y z : atomic_term } { T Ty Tz T' : term } :
    x <> y -> is_free Ty x = true -> is_free T z = false -> is_free Ty z = false -> substitution x T z Tz ->
    substitution y Tz Ty T' -> substitution y ( abst x T ) Ty ( abst z T' ).
```

که عملا تنها تفاوت آن با تعریف کتاب در این است که در حالتی که abstraction یک ترم دیگر باشد تا جای امکان renaming را انجام نمی دهد.
سپس اثبات می کنیم که با جانشینی یک متغیر به جای یک متغیر دیگر در یک ترم ارتفاع آن ترم تغییر نمی کند و اثبات این مورد را با استقرای قوی روی ارتفاع ترم دلخواه انجام داده ایم.
سپس تعریف استاندارد α-equivalency و β-reduction را در کوک پیاده سازی کردیم و اثبات کردیم که α-equivalency یک رابطه هم ارزی است. (از این پس α-equivalency را با $\sim$ نمایش می دهیم.) توجه کنید که هر دو مورد محمول های دو موضعی روی ترم ها هستند. دلیل اثبات رابطه هم ارزی بودن $\sim$ این است که بتوانیم از برخی تاکتیک های کوک برای اثبات احکام مختلف درباره آن استفاده کنیم.
سپس یک محمول دو موضعی دیگر به نام gbeta تعریف کردیم که قرار است اثبات کنیم transitive reflexive closure آن با transitive reflexive closure β-reduction معادل است و همین طور این رابطه دارای diamond property است. تعریف gbeta به صورت زیر است.

```
Inductive gbeta : term -> term -> Prop :=
  | gb_refl { t : term } : gbeta t t
  | gb_reduct ( t1 t1' t2 t2' T : term ){ x : atomic_term } : gbeta t1 t1' -> gbeta t2 t2' -> substitution x t1' t2' T -> gbeta ( appl ( abst x t1) t2 ) T
  | gb_abst { t t' : term }{ x : atomic_term } : gbeta t t' -> gbeta ( abst x t ) ( abst x t' )
  | gb_appl { t1 t2 t3 t4 : term } : gbeta t1 t2 -> gbeta t3 t4 -> gbeta ( appl t1 t3 ) ( appl t2 t4 ).
```

$$
\begin{array}{lcl}
P \twoheadrightarrow_l P & & \\
P \twoheadrightarrow_l P' & \Rightarrow & \lambda x.P \twoheadrightarrow_l \lambda x.P' \\
P \twoheadrightarrow_l P' \ \& \ Q \twoheadrightarrow_l Q' & \Rightarrow & P\,Q \twoheadrightarrow_l P'\,Q' \\
P \twoheadrightarrow_l P' \ \& \ Q \twoheadrightarrow_l Q' & \Rightarrow & (\lambda x.P)\,Q \twoheadrightarrow_l P'[x := Q']
\end{array}
$$

سپس اثبات کردیم که gbeta و β-reduction معادل اند به این معنا که $\forall xy, (\text{gbeta}\ xy) \leftrightarrow (x \to_\beta y)$. برای اثبات این حکم نیاز از چند قضیه و لم استفاده می کنیم که به ترتیب در زیر آمده اند.

- در ابتدا اثبات می کنیم که $\beta \subseteq$ gbeta برای اثبات این قضیه کافی است روی β استقرا بزنیم و نتیجه این قضیه این است که trans_clos$\beta \subseteq$ gbeta trans_clos

- حکم لم بعدی بیان می کند که $\forall XY : term,$ clos_refl_trans beta$XY \to \forall Z : term,$ clos_refl_trans beta$(XZ)(YZ)$. برای اثبات این حکم کافی است روی clos_refl_trans استقرا بزنیم.

- حکم لم بعدی بیان می کند که $\forall XY : term,$ clos_refl_trans beta$XY \to \forall Z : term,$ clos_refl_trans beta$(ZX)(ZY)$. برای اثبات این حکم کافی است روی clos_refl_trans استقرا بزنیم.

- حکم لم بعدی بیان می کند که $\forall MNPQ : term,$ clos_refl_trans beta$MN \to$ beta$PQ \to$ clos_refl_trans beta$(MP)(NQ)$. برای اثبات این حکم کافی است از دو لم قبل استفاده کنیم.

- حکم لم بعدی بیان می کند که $\forall XY : term,$ clos_refl_trans beta$XY \to \forall x : term,$ clos_refl_trans beta$(\lambda x.X)(\lambda x.Y)$. برای اثبات این حکم کافی است روی clos_refl_trans استقرا بزنیم.

- در نهایت می توانیم حکم مورد نظر یعنی اثبات معادل بودن trans_clos β-reduction و trans_clos gbeta را اثبات کنیم. ابتدا حکم را به دو بخش تقسیم می کنیم اثبات بخش اول که بیان می کند که trans_clos β-reduction $\subseteq$ gbeta trans_clos. با استفاده از این که trans_clos حافظ $\subseteq$ است و اینکه β-reduction $\subseteq$ gbeta است ساده است. برای اثبات طرف دوم نیز با توجه به قضیه ای که پیش تر اثبات کردیم کافی است اثبات کنیم که trans_clos β-reduction $\subseteq$ gbeta برقرار است برای این نیز کافی است مجددا با توجه به قضیه ای که پیش تر اثبات شده کافی است اثبات کنیم که clos_refl_trans β-reduction $\subseteq$ gbeta که اثبات این مورد نیز با استقرا روی gbeta و لم های قبلی ساده است.

۱.۵.۰ بخش پایانی:

اکنون تنها چیزی که برای اثبات قضیه چرچ راسر نیاز است این است که اثبات کنیم gbeta دارای diamond property است که برای این مورد نیز کافی است روی آن استقرا بزنیم که البته این راه نیاز دارد به دانستن اینکه gbeta تحت جانشینی حفظ می شود و برای اثبات این مورد کافی است روی جانشینی استقرا بزنیم که البته این راه نیز نیازمند این است که اثبات کنیم اگر A، B و C سه ترم دلخواه باشند و x و y نیز دو متغیر دلخواه باشند که $x \neq y$ و x در C آزاد نباشد آنگاه $A[x := B][y := C] \sim A[y := C][x := B[y := C]]$ و برای اثبات این مورد نیز کافی است روی ارتفاع A استقرا قوی بزنیم که البته در این راه نیاز داریم ثابت کنیم که به ازای هر دو ترم A و B و متغیر x ترم C به گونه ای وجود دارد که C حاصل جانشینی B در A به جای x شود. که از همه اثبات هایی که در بخش پایانی این بخش بیان کردم تا کنون فقط موفق شدم اثبات مورد آخر را در کوک پیاده سازی کنم و عملا بیشتر وقت من صرف تلاش هایی برای اثبات سایر لم های بخش پایانی شد که همگی به جز این راه که نیمه تمام مانده با شکست مواجه شدند.