Математичка гимназија

МАТУРСКИ РАД

- из математике -

Интерактивни доказивач теорема

Ученик:
Игор Енги IVд

Ментор:
др Зоран Петрић

Београд, мај 2021.

Садржај

0 Увод

Главни циљ овог рада је да уведемо теорију која се користи за прављење и развој интерактивних доказивача теорема, као и куцање програма. Интерактивни доказивач теорема је програм који може проверити математички доказ. Корисник пише код, који представља доказ неке теореме, а програм проверава да ли је резоновање било тачно.

Основа многих модерних доказивача, као што су Coq и Lean, је ламбда рачун. Ламбда рачун има своје терме који се сви истовремено могу посматрати и као функције. Прво уводимо ламбда рачун без типова и доказујемо његова својства, а затим сваком терму ламбда рачуна придружујемо тип. Тај тип ће мало ограничити како можемо да градимо терме, али то и желимо. Испоставља се да ако можемо наћи терм било ког типа, онда можемо доказати било коју теорему, што свакако не желимо. Својство строге нормализације терама (да се ни један терм не може бесконачно редуковати) играће кључну улогу у доказу конзистентности система.

На крају, укратко описујемо имплементацију програма и представљамо неколико доказа који су компјутерски проверени.

Такође, желео бих да се захвалим свом ментору и професору геометрије, Зорану Петрићу, на великој помоћи у откривању и разумевању ове области, као и у писању рада.

1 Ламбда рачун

1.1 Ламбда рачун без типова

Ламбда рачун први је увео Алонзо Черч. Идеја му је била да на неки начин опише израчунавање неког програма. Овде ћемо навести дефиницију и нека основна својства овог система. Занимљиво је да ће он бити Тјуринг комплетан, мада се овом чињеницом не бавимо овде.

Дефиниција 1. Скуп свих терма, Λ, дефинисан је индуктивно:

1. Пребројиво бесконачно променљивих: $v_1, v_2, \cdots \in \Lambda$,
2. Ако је x променљива, а $M \in \Lambda$, онда је $\lambda x.M \in \Lambda$,
3. Ако су $M, N \in \Lambda$, онда је $(M\ N) \in \Lambda$.

Друго правило зовемо апстракција, а треће примена. Апстракција има улогу прављења функције од терма, док ће примене омогућавати да применимо ту функцију (када уведемо $\beta-$редукцију, биће јасније).

Нотација. Дефинишемо неколико правила како би нам било лакше да записујемо терме.

- Када је терм $(M\ N)$ добијен применом, можемо склонити спољне заграде, па добијамо $M\ N$.
- Сматрамо да примена асоцира на лево, односно $A\ B\ C$ је исто што и $((A\ B)\ C)$.
- У случају да имамо $\lambda x.A\ B$, сматрамо да је то $\lambda x.(A\ B)$, а не $(\lambda x.A\ B)$.

Пример 1. Функцију $f(x) = x^2-3$ можемо представити преко терма $\lambda x.x^2-3$. Можемо чак рећи: $\lambda x.(\lambda a.x^2 - a)\ 3$ (3 ће да се замени уместо a преко $\beta-$редукције коју ћемо касније дефинисати). Сада ако желимо да видимо

колико је $f(5)$, урадимо примену и добијемо $(\lambda x.x^2 - 3)\ 5 = 5^2 - 3 = 22$ (5 се поново заменило уместо x преко $\beta-$редукције). У овом примеру смо претпоставили да нам је степеновање и одузимање дефинисано преко неких терма - иначе $\lambda x.x^2-3$ уопште није терм. Касније ћемо видети како је ово могуће урадити.

Још истичемо да терми који представљају функције немају ограничен домен. Домен им је цео скуп Λ. Могућност за функције са одређеним доменом добићемо у ламбда рачуну са *типовима*.

Пример 2. Терм $I = \lambda x.x$ представља идентичко пресликавање. Терме $T = \lambda x.\lambda y.x$ и $F = \lambda x.\lambda y.y$ ћемо звати тачно и нетачно, редом (то је само конвенција). Терми $(y\ \lambda z.x)$ или $(x\ y)$ такође постоје, али нису „затворени" , то јест променљиве x и y су слободне у њима.

Дефиниција 2. Подтерми терма добијеног апстракцијом $\lambda x.M$ су M и сви подтерми од M. Подтерми терма добијеног применом $(M\ N)$ су M, N, сви подтерми од M и сви подтерми од N. Једини подтерм променљиве x је x.

Сада уводимо слободне и везане променљиве терма. Са $FV(M)$ означавамо скуп слободних променљивих терма M, а са $BV(M)$ скуп везаних променљивих терма M.

Дефиниција 3. Скуп слободних и везаних променљивих дефинишемо индуктивно:

- Ако је терм x променљива онда је $FV(x) = \{x\}$, а $BV(x) = \emptyset$;
- Ако је $\lambda x.M$ терм добијен апстракцијом, онда је $FV(\lambda x.M) = FV(M)\setminus \{x\}$, а $BV(\lambda x.M) = BV(M) \cup \{x\}$;
- Ако је $M\ N$ терм добијен применом, онда је $FV(M\ N) = FV(M) \cup FV(N)$, а $BV(M\ N) = BV(M) \cup BV(N)$.

Додатно, дефинишемо $VAR(A) = FV(A) \cup BV(A)$ за сваки терм A.

Важно је назначити да $FV(M) \cap BV(M)$ не мора бити празан скуп, на пример у терму $M = x\ \lambda x.x$ имамо да је $FV(M) = \{x\}$ и $BV(M) = \{x\}$ јер је прво појављивање променљиве x заправо слободно, а друго је везано. Онда ћемо у терму $\lambda x.M$ имати $FV(\lambda x.M) = \emptyset$ и $BV(\lambda x.M) = \{x\}$. Приметимо да је овде променљива x везана два пута - то је дозвољено.

1.1.1 $\alpha-$конверзија

У овом поглављу навешћемо и доказаћемо основна својства $\alpha-$конверзије, односно преименовања везаних променљивих. Интуитивно, желимо да искажемо једнакост неких терма. На пример, нема разлога да правимо разлику између $\lambda x.x$ и $\lambda y.y$.

Са $M[x/N]$ означавамо терм који је добијен тако што је свако слободно појављивање променљиве x у терму M замењено термом N.

Дефиниција 4. За сваки терм N и променљиву x, дефинишемо операцију $[x/N]$ над термима индуктивно:

- $x[x/N] = N$,
- $y[x/N] = y$ (за $y \neq x$),
- $(\lambda x.M)[x/N] = \lambda x.M$,
- $(\lambda y.M)[x/N] = \lambda y.(M[x/N])$ (за $y \neq x$),
- $(A\ B)[x/N] = A[x/N]\ B[x/N]$.

Лема 1. Важи следеће:

- $M[x/x] = M$.
- $M[x/N] = M$ када $x \notin FV(M)$.
- $M[x/y][y/N] = M[x/N]$ када $y \notin FV(M)$.
- $M[x/X][y/Y] = M[y/Y][x/X[y/Y]]$ када $x \notin FV(Y)$.
- Ако $x \in FV(M)$, онда $VAR(M[x/N]) = VAR(M) \cup VAR(N)$ када $x \in BV(M)$, односно $VAR(M[x/N]) = (VAR(M) \setminus \{x\}) \cup VAR(N)$ када $x \notin BV(M)$.

Доказ. Доказаћемо само треће својство, остала се такође доказују индукцијом по сложености терма M. Можемо претпоставити $y \neq x$. За базу нам треба $x[x/y][y/N] = x[x/N] = N$ (применом дефиниције важи) и $z[x/y][y/N] = z[x/N] = z$ за $z \neq x, y$ (поново само дефиниција). Даље,

- $(\lambda x.M)[x/y][y/N] = (\lambda x.M)[y/N] = \lambda x.(M[y/N]) = \lambda x.M$ (јер је $y \notin FV(M)$).
- $(\lambda z.M)[x/y][y/N] = (\lambda z.(M[x/y]))[y/N] = \lambda z.(M[x/y][y/N]) = \lambda z.(M[x/N]) = (\lambda z.M)[x/N]$

- $(A\ B)[x/y][y/N] = (A[x/y][y/N]\ B[x/y][y/N]) = (A[x/N]\ B[x/N]) = (A\ B)[x/N]$.

□

Дефиниција 5. За терме A и B дефинишемо $\alpha-$конверзију индуктивно. $A \sim_\alpha B$ важи ако:

- $A = (C\ D)$, $B = (C\ D')$ и $D \sim_\alpha D'$,
- $A = (C\ D)$, $B = (C'\ D)$ и $C \sim_\alpha C'$,
- $A = \lambda x.C$, $B = \lambda x.D$ и $C \sim_\alpha D$ или
- $A = \lambda x.C$, $B = \lambda y.(C[x/y])$ и $y \notin FV(C)$.

Лема 2. Релација $\sim_\alpha$ је симетрична.

Доказ. Доказујемо $A \sim_\alpha B \implies B \sim_\alpha A$ индукцијом по сложености терма A. Ако је A примена довољна нам је једна индуктивна претпоставка. Ако је $A = \lambda x.C$ апстракција, имамо 2 случаја за B.

1. $B = \lambda x.D$, па је $C \sim_\alpha D$, а по индуктивној претпоставци $D \sim_\alpha C$ и $B \sim_\alpha A$.
2. $B = \lambda y.(C[x/y])$ и $y \notin FV(C)$. Сада $x \notin FV(C[x/y])$, па је $B \sim_\alpha \lambda x.(C[x/y][y/x]) = \lambda x.C = A$.

□

Дефиниција 6. $\equiv_\alpha$ је рефлексивно-транзитивно затворење релације $\sim_\alpha$.

Лема 3. $\equiv_\alpha$ је релација еквиваленције.

Доказ. Директно из дефиниције и леме 2. □

Лема 4. Ако је $A \equiv_\alpha B$, онда важи нешто од следећег:

- A и B су променљиве и $A = B$,
- $A = (M\ N)$, $B = (M'\ N')$ и $M \equiv_\alpha M'$, $N \equiv_\alpha N'$,
- $A = \lambda x.C$, $B = \lambda y.D$ и $C[x/y] \equiv_\alpha D$ и још $y \notin FV(C)$, $x \notin FV(D)$.

Доказ. Из дефиниције релација $\sim_\alpha$ и $\equiv_\alpha$. □

Теорема 1. Ако је A терм и X произвољан коначан скуп променљивих, онда постоји терм A' такав да је $A \equiv_\alpha A'$ и $BV(A') \cap X = \emptyset$.

Доказ. Индукцијом по сложености терма A. Ако је A променљива, онда $BV(A) = \emptyset$. Ако је $A = (M\ N)$ примена, онда по индуктивној претпоставци постоје M' и N' такви да је $M \equiv_\alpha M', N \equiv_\alpha N', BV(M') \cap X = \emptyset$ и $BV(N') \cap X = \emptyset$, па је $(M'\ N') \equiv_\alpha A$ тражени терм. Ако је $A = \lambda x.M$ апстракција, можемо по индуктивној претпоставци да нађемо терм $M' \equiv_\alpha M$ такав да је $BV(M') \cap (X \cup \{x\}) = \emptyset$. Сада одаберемо променљиву t тако да $t \notin X \cup VAR(M')$ (оба та скупа су коначна, па t заиста постоји). Тражени терм ће бити $\lambda t.M'[x/t]$. Можемо проверити да је $M'[x/t] \equiv_\alpha M[x/t]$, одакле ће следити $\lambda t.M'[x/t] \equiv_\alpha \lambda t.M[x/t] \equiv_\alpha \lambda x.M$ (због избора променљиве t немамо проблеме). □

Претходно својство доказали смо зато што ћемо касније имплементирати функцију која мења имена променљивих, по потреби.

У будуће, када напишемо терм A, мислићемо на класу еквиваленције релације $\equiv_\alpha$ у којој је A. Поистовећујемо све терме који су исти по тој релацији, односно све терме који се разликују само у именима везаних променљивих.

1.1.2 $\beta-$редукција

$\beta-$редукција ће нам омогућити „израчунавање" терма, или примењивање функције.

Терм зовемо редексом ако је облика $(\lambda x.M)\ N$.

Дефиниција 7. За терме A и B пишемо $A \to_\beta B$ ако и само ако важи нешто од следећег:

- $A \equiv (\lambda x.M)\ N$ је редекс и $B \equiv M[x/N]$, где је N узето тако су његове везане променљиве различите од свих променљивих терма M;
- $A \equiv \lambda x.M$, $B \equiv \lambda x.N$ и $M \to_\beta N$;
- $A \equiv (M\ N)$, $B \equiv (M\ N')$ и $N \to_\beta N'$;
- $A \equiv (M\ N)$, $B \equiv (M'\ N)$ и $M \to_\beta M'$.

Другим речима, $A \to_\beta B$ ако смо могли од A да добијемо B применом једног правила $\beta-$редукције.

Пример 3.

$$(\lambda x.x)\ t \to_\beta t$$

$$(\lambda x.y)\ t \to_\beta y$$

$$(\lambda x.(\lambda a.x^2 - a)\ 3)\ 5 \to_\beta (\lambda a.5^2 - a)\ 3 \to_\beta 5^2 - 3 = 22$$

Дефиниција 8. Релација $\twoheadrightarrow_\beta$ је рефлексивно-транзитивно затворење релације $\rightarrow_\beta$.

Последица 1. Ако $A \twoheadrightarrow_\beta B$ онда постоји n и терми $A_1 \equiv A, A_2, \ldots, A_n \equiv B$ такви да је $A_i \rightarrow_\beta A_{i+1}$ за свако $0 < i < n$.

Следећа теорема први пут је доказана у [4], а ми ћемо представити доказ из [7].

Теорема 2 (Черч-Росер)**.** Ако су A, B и C терми тако да важи $A \twoheadrightarrow_\beta B$, $A \twoheadrightarrow_\beta C$, онда постоји терм D тако да $B \twoheadrightarrow_\beta D$ и $C \twoheadrightarrow_\beta D$. Другим речима $\twoheadrightarrow_\beta$ задовољава „својство ромба" .

Доказ. Посматрајмо индуктивно дефинисану релацију $\Rightarrow_\beta$:

- $M \Rightarrow_\beta M$;
- Ако $M \Rightarrow_\beta M'$, онда $\lambda x.M \Rightarrow_\beta \lambda x.M'$;
- Ако $M \Rightarrow_\beta M'$ и $N \Rightarrow_\beta N'$, онда $(M\ N) \Rightarrow_\beta (M'\ N')$;
- Ако $M \Rightarrow_\beta M'$ и $N \Rightarrow_\beta N'$, онда $(\lambda x.M)\ N \Rightarrow_\beta M'[x/N']$.

Можемо доказати индукцијом да $A \Rightarrow_\beta B \implies A \twoheadrightarrow_\beta B$. Такође можемо закључити да $A \rightarrow_\beta B \implies A \Rightarrow_\beta B$. Сада имамо да је рефлексивно-транзитивно затворење релације $\Rightarrow_\beta$ заправо $\twoheadrightarrow_\beta$.

Можда није одмах јасно зашто је $\Rightarrow_\beta$ слабија од $\twoheadrightarrow_\beta$. Узмимо за пример терм $A = (\lambda x.x\ x)\ \lambda x.x$. Имамо да је $A \Rightarrow_\beta (\lambda x.x)\ (\lambda x.x)$ и имамо $A \twoheadrightarrow_\beta \lambda x.x$, али немамо $A \Rightarrow_\beta \lambda x.x$.

Доказаћемо још једно својство пре него што наставимо:

Лема 5. $M \Rightarrow_\beta M', N \Rightarrow_\beta N' \implies M[x/N] \Rightarrow_\beta M'[x/N']$ (x није везана у M).

Доказ. Индукцијом по сложености терма M. Ако је $M = x$ променљива, онда је $M[x/N] = N \Rightarrow_\beta N' = M'[x/N']$. Иначе имамо случајеве када је M апстракција или примена. Једини занимљив случај је када је $M = (\lambda y.M_1)\ M_2$ редекс и $M' = M_1'[y/M_2']$. Можемо узети y тако да $y \notin FV(N')$. Сада је $M'[x/N'] = M_1'[x/N'][y/(M_2'[x/N'])]$. По индуктивној претпоставци имамо да је $M_1[x/N] \Rightarrow_\beta M_1'[x/N']$ и $M_2[x/N] \Rightarrow_\beta M_2'[x/N']$, па по дефиницији имамо да је $M[x/N] = (\lambda y.(M_1[x/N]))\ (M_2[x/N]) \Rightarrow_\beta (\lambda y.(M_1'[x/N']))\ (M_2'[x/N']) \Rightarrow_\beta M_1'[x/N'][y/(M_2'[x/N'])] = M'[x/N']$. □

Доказаћемо да за сваки терм M постоји терм M^*, такав да за све N важи $M \Rightarrow_\beta N \implies N \Rightarrow_\beta M^*$. Дефинишимо операцију * индуктивно на следећи начин:

- За променљиве $x^* = x$;
- $(\lambda x.M)^* = \lambda x.M^*$;
- Ако $(M\ N)$ није редекс, онда $(M\ N)^* = (M^*\ N^*)$;
- За редексе $((\lambda x.M)\ N)^* = M^*[x/N^*]$.

На неки начин, M^* представља истовремено редуковање свих редекса који постоје у M (али добијени терм може да садржи редексе).

Индукцијом по M доказујемо да за свако N важи $M \Rightarrow_\beta N \implies N \Rightarrow_\beta M^*$. Ако је M променљива јасно је да важи. Сада раздвајамо по случајевима за сложеније M:

- Ако је $M = \lambda x.M_1$ апстракција, онда из дефиниције $\Rightarrow_\beta$, добијамо да је и $N = \lambda x.N_1$ за неко N_1 тако да $M_1 \Rightarrow_\beta N_1$. Сада је по индуктивној претпоставци $N_1 \Rightarrow_\beta M_1^*$, па по дефиницији имамо $\lambda x.N_1 \Rightarrow_\beta \lambda x.M_1^* = M^*$;
- Ако је $M = (M_1\ M_2)$ и M није редекс, слично као у прошлом случају имаћемо $N = (N_1\ N_2)$ где важи $M_1 \Rightarrow_\beta N_1$ и $M_2 \Rightarrow_\beta N_2$. Одатле ће следити $N \Rightarrow_\beta M^*$;
- Ако $M = (\lambda x.M_1)\ M_2$ јесте редекс, онда је $M^* = M_1^*[x/M_2^*]$ и постоје две могућности за N:
  - $N = (\lambda x.N_1)\ N_2$, где је $M_1 \Rightarrow_\beta N_1$ и $M_2 \Rightarrow_\beta N_2$. По индуктивној претпоставци имамо да је $N_1 \Rightarrow_\beta M_1^*$ и $N_2 \Rightarrow_\beta M_2^*$. Сада је по дефиницији $N \Rightarrow_\beta M_1^*[x/M_2^*] = M^*$
  - $N = N_1[x/N_2]$ за неке N_1 и N_2, где важи $M_1 \Rightarrow_\beta N_1$ и $M_2 \Rightarrow_\beta N_2$. По индуктивној претпоставци је $N_1 \Rightarrow_\beta M_1^*$ и $N_2 \Rightarrow_\beta M_2^*$. По леми 5 имамо да је $N \Rightarrow_\beta M_1^*[x/M_2^*] = M^*$.

Сада релација $\Rightarrow_\beta$ задовољава својство ромба. Остатак доказа теореме 2 је само тврђење да ако произвољна релација задовољава својство ромба, онда га и њено транзитивно раширење задовољава. Дајемо само скицу.

Претпоставимо да је $A \twoheadrightarrow_\beta B$ и $A \twoheadrightarrow_\beta C$. Како је $\twoheadrightarrow_\beta$ транзитивно затворење релације $\Rightarrow_\beta$, имамо да постоје $A_{1,1} = A, A_{2,1}, ..., A_{n,1} = B$ и $A_{1,1} = A, A_{1,2}, ..., A_{1,m} = C$ тако да је $A_{i,1} \Rightarrow_\beta A_{i+1,1}$ и $A_{1,j} \Rightarrow_\beta A_{1,j+1}$. Сада ће нам својство које смо доказали за $\Rightarrow_\beta$ направити мрежу терма $A_{i,j}$ у чијим ће се ћошковима налазити терми A, B, C и $D = A_{n,m}$. Одатле ће бити јасно да важи $B \twoheadrightarrow_\beta D$ и $C \twoheadrightarrow_\beta D$. □

Дефиниција 9. Кажемо да је терм A у нормалној форми ако ниједан његов подтерм није редекс. Додатно, нормална форма терма X је терм A (који је у нормалној форми), ако је $X \twoheadrightarrow_\beta A$.

Нормална форма не мора увек да постоји, на пример код терма $(\lambda x.x\ x)\ (\lambda x.x\ x)$ колико год $\beta-$редукција да урадимо, неће се променити.

Теорема 3 (Јединственост нормалне форме)**.** Ако терм X има нормалну форму, онда је она јединствена.

Доказ. Претпоставимо да су A и A' нормалне форме терма X. Тада је $X \twoheadrightarrow_\beta A$ и $X \twoheadrightarrow_\beta A'$. По теореми 2 имамо да постоји терм Y тако да $A \twoheadrightarrow_\beta Y$ и $A' \twoheadrightarrow_\beta Y$. Како су и A и A' у нормалној форми, не смеју садржати редексе, односно мора важити $A = Y = A'$. □

Дефиниција 10. Терм x је нормализибилан ако постоји низ $\beta-$редукција који доводи до нормалне форме. Терм x је строго нормализибилан ако не постоји бесконачан низ $\beta-$редукција (дакле сваки низ $\beta-$редукција у неком тренутку доводи до нормалне форме).

Дефиниција 11. $=_\beta$ је најмања релација еквиваленције која је надскуп релације $\twoheadrightarrow_\beta$.

Теорема 4 (Конфлуенција)**.** Ако је $M =_\beta N$, онда постоји R тако да $M \twoheadrightarrow_\beta R$ и $N \twoheadrightarrow_\beta R$.

Доказ. Ако је $M =_\beta N$, онда постоји низ $A_1 = M, A_2, \ldots, A_n = N$ тако да за свако $0 < i < n$ важи $A_i \twoheadrightarrow_\beta A_{i+1}$ или $A_{i+1} \twoheadrightarrow_\beta A_i$. Индукцијом по $i \geq 1$ показујемо да постоји R тако да $A_1 \twoheadrightarrow_\beta R$ и $A_i \twoheadrightarrow_\beta R$. Када правимо корак, нетривијалан случај је када $A_i \twoheadrightarrow_\beta A_{i+1}$. Тада по индуктивној хипотези узмемо R тако да $A_1 \twoheadrightarrow_\beta R$ и $A_i \twoheadrightarrow_\beta R$, а онда по теореми 2 узмемо R' тако да $R \twoheadrightarrow_\beta R'$ и $A_{i+1} \twoheadrightarrow_\beta R'$. □

Навешћемо неколико примера како бисмо илустровали $\beta-$редукцију, мада нам ови примери неће много значити касније.

Пример 4 (Черчово кодирање природних бројева)**.** Узимамо природне бројеве да буду облика $\lambda f.\lambda x.f^n\ x$, односно:

$$0 = \lambda f.\lambda x.x$$
$$1 = \lambda f.\lambda x.f\ x$$
$$2 = \lambda f.\lambda x.f\ (f\ x)$$
$$\ldots$$

Можемо дефинисати и операције на овим бројевима. Приметимо да је $m = \lambda f.\lambda x.m\ f\ x$ за сваки број m. Ако променимо f или x можемо добити разне ствари. Да бисмо добили $m + n$ у терму m ћемо „померити" x за n примена функције f:

$$p = \lambda m.\lambda n.\lambda f.\lambda x.m\ f\ (n\ f\ x).$$

Можемо проверити да се рецимо $p\ 2\ 2$ редукује у 4:

$$\begin{aligned} p\ 2\ 2 &\twoheadrightarrow_\beta \lambda f.\lambda x.2\ f\ (2\ f\ x) \\ &\twoheadrightarrow_\beta \lambda f.\lambda x.2\ f\ (f\ (f\ x)) \\ &\twoheadrightarrow_\beta \lambda f.\lambda x.f\ (f\ (f\ (f\ x))). \end{aligned}$$

Раније смо рекли да ће нам тачно и нетачно бити терми $T = \lambda x.\lambda y.x$ и $F = \lambda x.\lambda y.y$ редом. Да ли можемо дефинисати терм *iszero* за који важи $iszero\ 0 \twoheadrightarrow_\beta T$ и $iszero\ n \twoheadrightarrow_\beta F$ за $n > 0$? Можемо:

$$iszero = \lambda n.\lambda f.\lambda x.n\ (\lambda u.x)\ f.$$

Добра је вежба проверити да ово заправо ради. Трик је био да ако се f појави у n, онда поставимо да f буде функција која увек враћа x. Преко овога можемо добити и терм *isone*, али је поступак мало компликованији јер морамо да проверимо да ли се f појављује тачно једном. Уведимо следеће терме:

$$\begin{aligned} A &= \lambda a.\lambda b.\lambda c.a \\ B &= \lambda a.\lambda b.\lambda c.b \\ C &= \lambda a.\lambda b.\lambda c.c. \end{aligned}$$

Нека је $f = \lambda x.x\ B\ C\ C$. Приметимо да је $f\ A =_\beta B$, $f\ B =_\beta C$ и $f\ C =_\beta C$. Ако је n природан број, онда је $B =_\beta n\ (\lambda x.x\ B\ C\ C)\ A$ ако и само ако је $n = 1$. Коначно, добијамо:

$$isone = \lambda n.n\ (\lambda x.x\ B\ C\ C)\ A\ F\ T\ F$$

Пример 5 (Рекурзија)**.** Посматрајмо терм $Y = \lambda f.(\lambda x.f\ (x\ x))\ (\lambda x.f\ (x\ x))$. Ако га применимо на неко g имамо следеће:

$$\begin{aligned} Y\ g &\twoheadrightarrow_\beta (\lambda x.g\ (x\ x))\ (\lambda x.g\ (x\ x)) \\ &\twoheadrightarrow_\beta g\ ((\lambda x.g\ (x\ x))\ (\lambda x.g\ (x\ x))) \end{aligned}$$

$$=_\beta g\ (Y\ \ g)$$

Ово нам омогућава да дефинишемо рекурзију и ово нам даје занимљив бесконачан спуст. Овај терм може имати нормалну форму, рецимо за $g = \lambda u.\lambda v.v$ имамо да је $Y\ \ g \twoheadrightarrow_\beta \lambda v.v$. Ипак, није тачно да се сваки низ $\beta-$редукција завршава у нормалној форми, па овај терм није строго нормализибилан.

1.2 Ламбда рачуни са типовима

1.2.1 Дефиниција PTS

Желимо да термима ламбда рачуна придружимо типове. Пре, у примени $(m\ n)$, нисмо тражили никакве посебне услове за n, док ћемо сада захтевати да оно има одређен облик, односно тип (тај тип ће зависити од терма m). Касније ћемо видети да одређене типове можемо поистоветити са исказима у некој логици, а терме који имају те типове са доказима тих исказа. Постоји више начина да ово урадимо, па ћемо дефинисати целу класу ламбда рачуна са типовима које ћемо звати PTS (*Pure type system*). Они су се први пут појавили, у мало другачијем облику, у [2].

Дефиниција 12. PTS спецификација је триплет скупова $(\mathcal{S}, \mathcal{A}, \mathcal{R})$, где је $\mathcal{S}$ скуп сорти, $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ скуп аксиома и $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S} \times \mathcal{S}$ скуп правила.

Свака спецификација S ће нам индуковати PTS λS.

Дефиниција 13. Псеудотерми у $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$ дефинисани су индуктивно на следећи начин:

- Променљиве $x_1, x_2, \ldots$ су псеудотерми,
- Сорте $s \in \mathcal{S}$ су псеудотерми,
- За псеудотерме T и M и променљиву x имамо да је $\lambda x : T.M$ псеудотерм ($\lambda-$апстракција)
- За псеудотерме T и M и променљиву x имамо да је $\Pi x : T.M$ псеудотерм ($\Pi-$апстракција)
- За псеудотерме M и N, $(M\ N)$ је псеудотерм (примена).

Ова дефиниција псеудотерма је слична дефиницији терма у обичном ламбда рачуну од раније. $\lambda x.M$ прешло је у $\lambda x : T.M$, добили смо нову апстракцију и добили смо сорте. $\Pi-$апстракције биће терми који ће бити типови $\lambda-$апстракција, а сорте ће бити „типови типова" односно типови $\Pi-$апстракција.

Слободне и везане променљиве дефинисане су исто као раније - једина разлика је што ће сада и Π да везује променљиве. Такође, у $\lambda x : T.M$, рачунамо и променљиве из T и променљиве из M (али x није везано у T, већ само у M). Слично за Π.

$\alpha-$конверзија и $\beta-$редукција за $\lambda-$апстракције раде исто као раније (код $\beta-$редукције, тип у редексу се „брише"). За $\Pi-$апстракцију, дефинишемо само $\alpha-$конверзију (која наравно у случају $\Pi x : T.M$, не мења слободна x у T). Сва својства која смо до сада доказали за обичан ламбда рачун преносиће се, па их нећемо поново доказивати. Оно што ће бити ново је да ће за неке PTS сваки терм имати нормалну форму. Ово својство биће нам кључно да докажемо да систем није контрадикторан.

Пре него што дефинишемо правила за доделу типова, прво дефинишемо псеудоконтексте и секвенте.

Дефиниција 14. За променљиву x и псеудотерм T, израз $x : T$ зовемо декларацијом. Кажемо да x има тип T.

Дефиниција 15. За псеудотерме A и B, израз $A : B$ зовемо доделом. Кажемо да A има тип B.

Дефиниција 16. *Псеудоконтекст* је (уређена) листа декларација: $x_1 : T_1, x_2 : T_2, \ldots, x_n : T_n$.

Дефиниција 17. Домен псеудоконтекста $\Gamma = x_1 : A_1, x_2, A_2, \ldots, x_n : A_n$ је $dom(\Gamma) = \{x_1, x_2, \ldots, x_n\}$.

Дефиниција 18. $\Gamma \vdash A : B$ је *секвент*, где је Γ псеудоконтекст, а $A : B$ је додела.

За *извођење* секвената, користићемо *правила извођења*. Она су дата у следећој форми:

$$\frac{P_1 \quad P_2 \quad \ldots \quad P_n}{P}\ r$$

Овде су $P_1 \ldots P_n$ премисе, док је P последица правила чије је име r. За наше потребе, последица може бити само секвент (облика $\Gamma \vdash A : B$), док премисе могу бити облика $a \in A$, $a \notin A$ (где је A неки скуп) или секвенти.

Од сада, па надаље сматрамо да радимо у општем PTS, $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R})$, осим уколико не нагласимо другачије.

Дефиниција 19. У наредном сматрамо да је Γ произвољан псеудоконтекст, x је променљива, A, B, C, D су псеудотерми, а s, s_1, s_2, s_3 су произвољне сорте. Правила извођења су:

$$\frac{(s_1, s_2) \in \mathcal{A}}{\vdash s_1 : s_2}\ \text{аксиом}$$

$$\frac{\Gamma \vdash A : s \quad x \notin dom(\Gamma)}{\Gamma, x : A \vdash x : A} \text{ почетак}$$

$$\frac{\Gamma \vdash A : s \quad \Gamma \vdash B : C \quad x \notin dom(\Gamma)}{\Gamma, x : A \vdash B : C} \text{ слабљење}$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2 \quad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash \Pi x : A.B : s_3} \ \Pi\text{−правило}$$

$$\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash C : s_2 \quad \Gamma, x : A \vdash B : C}{\Gamma \vdash \lambda x : A.B : \Pi x : A.C} \ \lambda\text{−правило}$$

$$\frac{\Gamma \vdash A : \Pi x : C.D \quad \Gamma \vdash B : C}{\Gamma \vdash (A\ B) : D[x/B]} \text{ примена}$$

$$\frac{\Gamma \vdash B : C \quad \Gamma \vdash C' : s \quad C =_\beta C'}{\Gamma \vdash B : C'} \text{ конверзија}$$

Даћемо кратко појашњење за свако правило редом, јер она на први поглед не делују толико интуитивно.

- Правило аксиом и скуп $\mathcal{A}$ даје нам типове за сорте (који су увек поново сорте). Приметимо да једна сорта не мора имати јединствени тип, у зависности од скупа $\mathcal{A}$.
- Правило почетак нам омогућава да у контекст додамо нову декларацију. Можемо додати само декларације уколико је тип у декларацији у некој сорти.
- Правило слабљење такође нам омогућава да у контекст додамо нову декларацију, али да не променимо већ добијену доделу.
- У Π−правилу је дефинисано који тип ће имати Π−апстракције. Тај тип зависи од скупа $\mathcal{R}$. Као и у правилу аксиом, тип Π−апстракције не мора бити јединствен.
- У λ−правилу дефинишемо која је тачно Π−апстракција тип λ−апстракције.
- У правилу примена дефинишемо тип примене. Можемо приметити да, када бисмо имали дефинисану β−редукцију за Π−апстракције, онда би тај тип био β−једнак псеудотерму $(\Pi x : C.D)\ B$. У правилу примена смо одмах узели редуковану форму баш да не бисмо имали потребе да дефинишемо β−редукцију за Π−апстракције.

- Конверзија омогућава да поистоветимо све секвенте који добијају доделе са $\beta-$једнаким типовима.

Дефиниција 20. *Извођење* секвента $\Gamma \vdash A : B$ је дрво чији су чворови правила извођења, листови су аксиоме, а корен је $\Gamma \vdash A : B$.

Дефиниција 21. Секвент $\Gamma \vdash A : B$ је изводив ако постоји његово изводјење.

Дефиниција 22. Псеудоконтекст Γ је контекст ако постоји изводив секвент $\Gamma \vdash A : B$ за неке псеудотерме A и B.

Дефиниција 23. Псеудотерм A је терм ако постоји изводив секвент $\Gamma \vdash A : B$ или $\Gamma \vdash B : A$ за неки псеудоконтекст Γ и псеудотерм B.

Пример 6. Узмимо PTS $\lambda(\{s\}, \{(s,s)\}, \{(s,s,s)\})$. Као прво, у сваком контексту Γ ће секвент $\Gamma \vdash s : s$ бити изводив (применом правила почетак и више правила слабљење). Сада је и секвент $\Gamma, x : s \vdash s : s$ изводив, па применом $\Pi-$правила добијамо да је $\Gamma \vdash \Pi x : s.s : s$ изводив за сваки контекст Γ. Слично је и $\Gamma \vdash \Pi x : s.x$ изводив, али $\Gamma \vdash \Pi x : x.s$ никада није изводив. Можемо извести да терм $\lambda x : s.x$ има тип $\Pi x : s.s$. Испоставиће се да у овом PTS, сваки терм има јединствен тип (до на $\beta-$редукцију).

Пример 7. Нека је сада PTS у коме радимо дат са $\lambda(\{s_1, s_2, s_3\}, \{(s_1, s_2), (s_1, s_3\}, \emptyset)$. Одмах је јасно да су секвенти $\vdash s_1 : s_2$ и $\vdash s_1 : s_3$ изводиви, али тада тип од s_1 није јединствен. Такође, због празног скупа $\mathcal{R}$, нећемо моћи да урадимо готово ништа сем примене прва три правила (аксиома, почетак и слабљење). Заиста, $\Pi-$правило захтева тројке из скупа $\mathcal{R}$, док примена и $\lambda-$правило захтевају постојање терма са $\Pi-$апстракцијом.

1.2.2 Основна својства PTS

Овде ћемо навести важна тврђења која се односе на све PTS. Нека ћемо оставити без доказа јер они нису ништа епохално. Пуни докази и још нека својства могу се пронаћи у [6].

Лема 6. За контекст $\Gamma = x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$,

- $FV(B) \cup FV(C) \subseteq \{x_1, \ldots, x_n\}$,
- $x_i = x_j \implies i = j$,
- $\Gamma \vdash a : A$ за све $(a, A) \in \mathcal{A}$,
- $\Gamma \vdash x_i : A_i$ за све $i \leq n$.

Прво својство нам каже да су се све слободне промењиве у термима морале појавити у контексту (иначе извођење не би било правилно). Друго својство је последица тога да у правилима извођења почетак и слабљење захтевамо $x \notin dom(\Gamma)$. Треће и четврто се могу доказати применом правила слабљења и правила за аксиоме.

Лема 7 (Супституција)**.** Нека су Γ_1 и $\Gamma_1, y : A, \Gamma_2$ контексти, A, B, C, D терми и y променљива. Такође нека су секвенти $\Gamma_1 \vdash D : A$ и $\Gamma_1, y : A, \Gamma_2 \vdash B : C$ изводиви. Тада је и $\Gamma_1, \Gamma_2[y/D] \vdash B[y/D] : C[y/D]$ изводив.[1]

Кључно у претходној леми је да смо променљиву типа A заменили термом типа A и изведени секвент је остао изводив.

Доказ. Индукцијом по извођењу секвента $\Gamma_1, y : A, \Gamma_2 \vdash B : C$ под претпоставком да имамо $\Gamma_1 \vdash D : A$. Ако је Γ_2 празан и последње правило је почетак, онда мора бити $B = y$ и $C \equiv A$. Готови смо јер је $y[y/D] = D$. Ако је Γ_2 празан и последње правило је слабљење онда важи $\Gamma_1 \vdash B : C$ По леми 6, $y \notin FV(B) \cup FV(C)$. Тада је $B[y/D] : C[y/D] = B : C$.

Ако је последње правило наслеђивања било $\Pi-$правило, $\lambda-$правило, конверзија или је Γ_2 празно и последње правило је било почетак или слабљење, из индуктивне претпоставке и примене правила добићемо оно што нам треба. Рецимо, за $\lambda-$правило:

$$\frac{\Gamma \vdash T : s_1 \quad \Gamma, x : T \vdash M : s_2 \quad \Gamma, x : T \vdash R : M}{\Gamma \vdash \lambda x : T.R : \Pi x : T.M} \ \lambda-\text{правило}$$

У претходном је $\Gamma = \Gamma_1, y : A, \Gamma_2$, а $B = \lambda x : T.R$ и $C = \Pi x : T.M$. По индуктивној претпоставци имамо изводиве секвенте:

- $\Gamma_1, \Gamma_2[y/D] \vdash T[y/D] : s_1$,
- $\Gamma_1, \Gamma_2[y/D], x : T[y/D] \vdash M[y/D] : s_2$ и
- $\Gamma_1, \Gamma_2[y/D] \vdash R[y/D] : M[y/D]$.

. Применом $\lambda-$правила добијамо

$$\Gamma_1, \Gamma_2[y/D] \vdash \lambda x : T[y/D].R[y/D] : \Pi x : T[y/D].M[y/D]$$

.

Остао је само случај када је последње правило примена. Нека је $B = B_1\ B_2$ и $C = C_2[x/B_2]$. Још важи:

[1] Када напишемо $\Gamma_2[y/D]$ мислимо на (псеудо)контекст добијен када се у сваком типу који се појављује у Γ_2 слободна променљива y замени термом D.

$$\frac{\Gamma \vdash B_1 : \Pi x : C_1.C_2 \quad \Gamma \vdash B_2 : C_1}{\Gamma \vdash B_1\ B_2 : C_2[x/B_2]} \text{ примена}$$

Слично као у прошлом случају, по индуктивној претпоставци и једном правилу примене добијамо да је $\Gamma_1, \Gamma_2[y/D] \vdash B[y/D] : C_2[y/D][x/B_2[y/D]]$. Можемо претпоставити да $x \neq y$ и $x \notin FV(D)$, па је $C_2[y/D][x/B_2[y/D]] = C_2[x/B_2][y/D] = C[y/D]$. Тиме је индукција готова.

□

Лема 8 (Проширење)**.** Ако су Γ и Γ_1 контексти, где је $\Gamma \subseteq \Gamma_1$ (дакле, скуп декларација Γ је подскуп скупа декларација Γ_1) и још је $\Gamma \vdash B : C$, онда је и $\Gamma_1 \vdash B : C$.

Лема 9. Нека је $\Gamma = x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n$ контекст и нека су M, N, P терми. Тада важе следеће импликације:

- $\Gamma \vdash s : P$, где је $s \in \mathcal{S} \implies P =_\beta s_1 \in \mathcal{S} \wedge (s, s_1) \in \mathcal{A}$ за неко s_1,
- $\Gamma \vdash x : P$, где је x променљива $\implies \exists i \leq n.x = x_i$ и $P =_\beta A_i$ и $1x_1, \ldots x_{i-1} \vdash A_i : s$ за неко $s \in \mathcal{S}$,
- $\Gamma \vdash \Pi x : M.N : P \implies \Gamma \vdash M : s_1$ и $\Gamma, x : M \vdash N : s_2$ и $P =_\beta s_3$ за неко $(s_1, s_2, s_3) \in \mathcal{R}$,
- $\Gamma \vdash \lambda x : M.N : P \implies \Gamma \vdash M : s_1$ и $\Gamma, x : M \vdash B : s_2$ и $\Gamma, x : M \vdash N : B$ и $\Gamma \vdash \Pi x : M.B : s_3$ и $P =_\beta \Pi x : M.B$ за неки терм B и неко правило $(s_1, s_2, s_3) \in \mathcal{R}$,
- $\Gamma \vdash M\ N : P \implies \Gamma \vdash M : \Pi x : A.B$ и $\Gamma \vdash N : A$ и $P =_\beta B[x/N]$ за неке терме A и B и неку променљиву x,
- $\Gamma \vdash M : P \implies \exists s \in \mathcal{S}.(P = s \vee \Gamma \vdash P : s)$.

Ову лему ћемо такође оставити без доказа. Када погледамо правила постаје интуитивно јасно зашто она важи. Рецимо, терм облика $\lambda x : M.N$ могли смо добити само ако су већ били испуњени неки услови, на пример да је $M : s_1$. Тип тог терма је увек прво нека $\Pi-$апстракција, па је зато $P =_\beta \Pi x : M.B$ (морамо ставити $=_\beta$ јер смо тип можда променили конверзијом).

Лема 10 (Пермутација)**.** Ако су Γ_1 и $\Gamma_1, x : A, y : B, \Gamma_2$ контексти, а P и Q терми и важи:

$$\Gamma_1, x : A, y : B, \Gamma_2 \vdash P : Q$$

и још је $\Gamma_1 \vdash B : s$ за неко $s \in \mathcal{S}$, онда је:

$$\Gamma_1, y : B, x : A, \Gamma_2 \vdash P : Q.$$

Због чињенице да је $\Gamma_1 \vdash B : s$, знамо да је $\Gamma_1, y : B, x : A$ заиста валидан контекст. Користећи лему 8 о проширењу, можемо добити да је $\Gamma_1, y : B, x : A, \Gamma_2$ такође валидан контекст. Тада је, поново по истој леми, и $\Gamma_1, y : B, x : A, \Gamma_2 \vdash P : Q$.

Лема 11 (Редуковање)**.** Нека су су Γ и Γ' контексти, B, B' и C терми и нека је $\Gamma \vdash B : C$. Ако је $\Gamma \twoheadrightarrow_\beta \Gamma'$ и $B \twoheadrightarrow_\beta B'$ онда важи:

- $\Gamma \vdash B' : C$,
- $\Gamma' \vdash B : C$.

Доказ. Доказујемо оба тврђења једном индукцијом, по дубини стабла извођења секвента $\Gamma \vdash B : C$. Можемо претпоставити да се десила тачно једна $\beta-$редукција.

Нека се $\beta-$редукција десила у контексту. Уколико је последње примењено правило $\lambda-$правило, $\Pi-$правило, примена или конверзија, по индуктивној хипотези смо доказали да $\Gamma' \vdash B : C$. Уколико је последње правило почетак или слабљење и редукција се није десила у последњој декларацији у Γ, такође смо готови по индуктивној претпоставци. Нека је $\Gamma = \Gamma_1, x : A$ и $\Gamma' = \Gamma_1, x : A'$ где је $A \to_\beta A'$ и нека је последње правило било почетак. По леми 9, $\Gamma_1 \vdash A : s$ за неко $s \in \mathcal{S}$. Индуктивни корак нам каже да је тада и $\Gamma_1 \vdash A' : s$, па применом правила почетак добијамо $\Gamma_1, x : A' \vdash x : A'$. Применом правила конверзије добијамо $\Gamma' \vdash x : A$. Случај када је последње правило било слабљење се ради слично, само нам не треба конверзија на крају.

Сада разматрамо шта се дешава ако се $\beta-$редукција десила у терму B. Уколико је последње правило било $\lambda-$правило, $\Pi-$правило, конверзија, почетак или слабљење завршавамо применом индуктивне претпоставке. Разматрајмо случај када је последње правило била примена. Тада је $B = B_1\ B_2$, $C = C_2[x/B_2]$, где је $\Gamma \vdash B_1 : \Pi x : C_1.C_2$ и $\Gamma \vdash B_2 : C_1$ (лема 9). Уколико се редукција десила унутар B_1 или B_2, завршавамо применом индуктивног корака. Претпоставимо да је онда $B_1 = \lambda x : C_1.B_3$, где је $\Gamma \vdash B_3 : C_2'$ ($C_2' =_\beta C_2$) и да је $B' = B_3[x/B_2]$. Применом леме 7 на $\Gamma \vdash B_3 : C_2'$ добијамо $\Gamma \vdash B_3[x/B_2] : C_2'[x/B_2]$. Још једном конверзијом, добијамо: $\Gamma \vdash B_3[x/B_2] : C_2[x/B_2]$. □

Претходна тврђења важе за све PTS. Нажалост, тип неког терма у једном контексту није јединствено одређен, али ако додамо један једноставан услов биће.

Дефиниција 24. PTS је функцијски ако је $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$ функција и $\mathcal{R} \subseteq (\mathcal{S} \times \mathcal{S}) \times \mathcal{S}$ функција.[2]

Лема 12 (Јединственост доделе типова функцијских PTS)**.** У функцијском PTS, за контекст Γ и терме B, C и C' где важи $\Gamma \vdash B : C$ и $\Gamma \vdash B : C'$ је $C =_\beta C'$.

Једино место где смо заправо могли суштински да променимо тип неког терма је применом правила аксиоме или $\Pi-$правила, али функцијски PTS нам то не дозвољава.

Лема 13 (Јачање са редукцијом за функцијске PTS)**.** Нека је $\Gamma = \Gamma_1, x : A, \Gamma_2$ контекст, B и C су терми за које важи $\Gamma \vdash B : C$, а x је променљива и важи $x \notin FV(\Gamma_2) \cup FV(B)$. Тада постоји терм C' за који $\Gamma_1, \Gamma_2 \vdash B : C'$ и $C \twoheadrightarrow_\beta C'$.

Интуиција иза овог тврђења је да би терм B требао да има јединствен тип у контексту који има све његове слободне променљиве. Тада, додавање декларације $x : A$ не би требало ишта да промени (дакле, C не би требало да зависи од x). Ипак, x може да се појави у C, али ће се изгубити неким $\beta-$редукцијама.

Доказ. Радимо индукцију по извођењу секвента $\Gamma \vdash B : C$. Занимљиви случајеви су када је последњи корак $\lambda-$правило и примена.

Рецимо да је $B = \lambda y : B_1.B_2$, $C = \Pi y : B_1.C_2$. Тада је $\Gamma \vdash B_1 : s_1$, $\Gamma, y : B_1 \vdash C_2 : s_2$ и $\Gamma, y : B_1 \vdash B_2 : C_2$. По индуктивној претпоставци је $\Gamma_1, \Gamma_2 \vdash B_1 : s_1$ и $\Gamma_1, \Gamma_2, y : B_1 \vdash B_2 : C_2'$ за неко $C_2 \twoheadrightarrow_\beta C_2'$.

По леми 9, $C_2' =_\beta s \in \mathcal{S}$ и $s : s_2 \in \mathcal{A}$ или $C_2' : s \in \mathcal{A}$ и $s = s_2$ по јединствености типова. Применом $\lambda-$правила, добијамо $\Gamma_1, \Gamma_2 \vdash \lambda y : B_1.B_2 : \Pi y : B_1.C_2'$.

Нека је сада $B = B_1\ B_2$ и $C = C_2[y/B_2]$. Како је последње правило примена, $\Gamma \vdash B_1 : \Pi y : C_1.C_2$ и $\Gamma \vdash B_2 : C_1$. Уз индуктивну претпоставку, Черч-Росера и једну конверзију, добијамо да постоје $C_1 \twoheadrightarrow_\beta C_1'$ и $C_2 \twoheadrightarrow_\beta C_2'$ тако да је $\Gamma_1, \Gamma_2 \vdash B_1 : \Pi y : C_1'.C_2'$ и $\Gamma \vdash B_2 : C_1'$. Сада правилом примене добијамо $\Gamma_1, \Gamma_2 \vdash B_1\ B_2 : C_2'[y/B_2]$. Тиме је индукција готова.

□

Последица 2 (Јачање за функцијски PTS)**.** У функцијском PTS, нека су $\Gamma_1, x : A, \Gamma_2$ контекст, B и C променљиве и важи $x \notin FV(\Gamma_2) \cup FV(B) \cup FV(C)$. Важи још да $\Gamma_1, x : A, \Gamma_2 \vdash B : C$. Тада је $\Gamma_1, \Gamma_2 \vdash B : C$.

[2] **Не** тражимо да је домен функције $\mathcal{A}$ заправо скуп $\mathcal{S}$ нити да је домен скупа $\mathcal{R}$ заправо скуп $\mathcal{S} \times \mathcal{S}$, само да су домени подскупови тих скупова (редом).

Доказ. Из леме 13 знамо да $\Gamma_1, \Gamma_2 \vdash B : C'$ где је $C \twoheadrightarrow_\beta C'$. Из леме 9 је $C = s \in \mathcal{S}$ или $\Gamma_1, x : A, \Gamma_2 \vdash C : s$. У првом случају је $C = C'$. У другом случају је $\Gamma_1, \Gamma_2 \vdash C : s$ по леми 13, па смо готови уз једну конверзију. □

1.2.3 Својства неких PTS

За потребе овог поглавља, узећемо да је скуп сорти $\mathcal{S} = \{*, \Box\}$, а скуп аксиома $\mathcal{A} = \{(*, \Box)\}$ (односно $\mathcal{A} = \{* : \Box\}$). Интуитивно гледано, $*$ ће нам глумити универзум логичких исказа. Зато ћемо и користити ознаку Prop уместо $*$ када причамо о исказима у овом систему.

Како бисмо мало олакшали запис, увешћемо још нотације. За $\Pi x : A.B$ пишемо $A \to B$ када $x \notin FV(B)$. По договору, итерирана $\to$ је асоцирана удесно, што значи да $A \to B \to C = A \to (B \to C)$.

Систем F

Како бисмо стекли интуицију за рад у неком PTS, представићемо овде Систем F. У овом делу, даваћемо више примера него доказа (део тих примера преузет је из [9]). Касније ћемо се вратити доказивању теорема о другим системима (прво једним једноставнијим од Система F, а затим и рачуну конструкција који ће бити кључан за наш програм).

Систем F (често и $\lambda 2$) биће PTS са скупом правила $\mathcal{R}_F = \{(*, *, *), (\Box, *, *)\}$.

Овакав скуп правила омогућава $\Pi-$апстракције облика $\Pi a : A.B$ где су $A : *$ и $B : *$ (правило $(*, *, *)$), али и облика $\Pi X : *.Y$ где је $Y : *$ (правило $(\Box, *, *)$).

Шта ово значи за логику која ће бити изоморфна нашем систему? Сетимо се да је $*$ = Prop тип предиката или исказа. $\Pi-$апстракција глумиће нам квантификатор $\forall$. Тип $\Pi a : A.B$ преводимо као „За сваки доказ a исказа A, важи B" . Генерално, B неће зависити од a, па ће ово бити проста импликација и записиваћемо $A \to B$. С друге стране, тип $\Pi X : \text{Prop}.Y$ каже нам да за сваки исказ X важи исказ Y. Ово је, дакле, квантификатор другог реда јер прича о својству неког исказа.

Пример 8. $\Pi A : \text{Prop}.A$ је тип који би се у логици превео као „За сваки исказ A важи A" . Ово, наравно, не сме бити тачно иначе је наш систем контрадикторан. Баш зато овај тип зовемо контрадикција и означаваћемо га са False. Рецимо да је $f : \text{False}$ неки доказ контрадикције. Тада је $A : \text{Prop} \vdash (f\ A) : A$, односно $(f\ A)$ је доказ исказа A за било који исказ. Ово одговара правилу „све следи из контрадикције" природне дедукције.

Пример 9. Терм $\lambda A : \text{Prop}.\lambda a : A.a$ има тип $\Pi A : \text{Prop}.\Pi a : A.A$ (односно $\Pi A : \text{Prop}.A \to A$) у сваком контексту. Овај тип одговараће реченици „За сваки исказ A, $A \implies A$," која је, наравно, увек тачна.

Још једна јака страна овог система биће могућност прављења типова попут Bool, парова, природних бројева... Сада ћемо илустровати то.

Узећемо да је $\text{Bool} = \Pi X : *.X \to X \to X$. Означимо два терма тог типа са:

$$T = \lambda X : *.\lambda x : X.\lambda y : X.x$$

$$F = \lambda X : *.\lambda x : X\lambda y : X.y$$

Тврдимо, без целог доказа, да су то једина два терма тог типа (у празном контексту). Заиста, ако посматрамо само терме у нормалној форми, морамо да урадимо 3 $\lambda-$апстракције ($\lambda X : *$, $\lambda x : X$ и $\lambda y : X$) и на крају морамо навести нешто типа X, а једино су нам на располагању x и y. Терме који нису у нормалној форми решавамо строгом нормализацијом (коју нећемо доказати за овај систем, мада ће она важити).

Рецимо да је B типа $Bool$, а u и v неки терми типа U. Да ли постоји начин да разликујемо када је $B = T$, а када је $B = F$? Прецизније, да ли можемо направити „if-else" команду која враћа u када је $B = T$, а v када је $B = F$? Испоставља се да је то веома једноставно ако погледамо дефиницију ових терма. Оно што тражимо је $B\ U\ u\ v$. Посматрајмо бета редукцију овог терма када је $B = T$:

$$\begin{aligned} T\ U\ u\ v &= (\lambda X : *.\lambda x : X.\lambda y : X.x)\ U\ u\ v \\ &\twoheadrightarrow_\beta (\lambda x : U.\lambda y : U.x)\ u\ v \\ &\twoheadrightarrow_\beta (\lambda y : U.u)\ v \\ &\twoheadrightarrow_\beta u \end{aligned}$$

Слично ће бити када је $B = F$:

$$\begin{aligned} F\ U\ u\ v &= (\lambda X : *.\lambda x : X.\lambda y : X.y)\ U\ u\ v \\ &\twoheadrightarrow_\beta (\lambda x : U.\lambda y : U.y)\ u\ v \\ &\twoheadrightarrow_\beta (\lambda y : U.y)\ v \\ &\twoheadrightarrow_\beta v \end{aligned}$$

Још један веома значајан тип биће тип природних бројева. Пробаћемо да мотивишемо како бисмо дошли до тог типа. Стандардни начин да их дефинишемо је да имамо објекат 0 и функцију $S : \mathbb{N} \to \mathbb{N}$, а сваки број ће

бити облика $S(S(\dots(S(0))\dots))$. У духу ламбда рачуна рекли бисмо да је:

$$\mathbb{N} : *, 0 : \mathbb{N}, S : \mathbb{N} \to \mathbb{N} \vdash S\ (S\ (\dots(S\ 0)\dots)) : \mathbb{N}.$$

Ако сада „испразнимо" контекст, добијамо:

$$\vdash \lambda X : *.\lambda o : X.\lambda s : X \to X.s\ (s\ (\dots(s\ o)\dots)) : \Pi X : *.X \to (X \to X) \to X.$$

Из овог разлога узећемо да је тип природних бројева баш $\mathrm{Int} = \Pi X : *.X \to (X \to X) \to X$. Поново, без доказа, тврдимо да ће једини терми да буду облика $\lambda X : *.\lambda o : X.\lambda s : X \to X.s\ (s\ (\dots(s\ o)\dots))$ и њих ћемо звати природним бројевима. Број n биће представљен оним термом у коме се s понавља n пута, дакле:

$$\begin{aligned} 0 &= \lambda X : *.\lambda o : X.\lambda s : X \to X.o \\ 1 &= \lambda X : *.\lambda o : X.\lambda s : X \to X.s\ o \\ 2 &= \lambda X : *.\lambda o : X.\lambda s : X \to X.s\ (s\ o) \\ &\dots \end{aligned}$$

Са овако дефинисаним природним бројевима моћи ћемо да дефинишемо и операције сабирања и множења. Посматрајмо следећи терм:

$$\lambda X : *.\lambda o : X.\lambda s : X \to X.n\ X\ (m\ X\ o\ s)\ s,$$

где су m и n неки бројеви типа Int. Добијени терм одговараће терму који одговара броју $n + m$. Идеја је била да у терму n где се s понавља n пута, померимо почетак (то јест o) за m:

$$m\ X\ o\ s \twoheadrightarrow_\beta s^m(o)$$

$$n\ X\ s^m(o)\ s \twoheadrightarrow_\beta s^n(s^m(o)) \equiv s^{n+m}(o).$$

Пример 10. Узмимо $n = 1$ и $m = 2$ и посматрајмо редукцију нашег терма (игноришући апстракције на почетку):

$$\begin{gathered} n\ X\ ((\lambda X_2 : *.\lambda o_2 : X_2.\lambda s_2 : X_2 \to X_2.s_2\ (s_2\ o_2))\ X\ o\ s)\ s \twoheadrightarrow_\beta n\ X\ (s\ (s\ o))\ s \\ (\lambda X_1 : *.\lambda o_1 : X_1.\lambda s_1 : X_1 \to X_1.s_1\ o_1)\ X\ (s\ (s\ o))\ s \twoheadrightarrow_\beta s\ (s\ (s\ o)). \end{gathered}$$

Када вратимо апстракције ово ће бити терм који одговара броју 3.

Можемо добити и затворени терм који представља сабирање:

$$p = \lambda x : \mathrm{Int}.\lambda y : \mathrm{Int}.\lambda X : *.\lambda o : X.\lambda s : X \to X.x\ X\ (y\ X\ o\ s)\ s.$$

Сличном идејом можемо добити и множење. Овај пут, уместо да мењамо o, мењаћемо s. За бројеве m и n, њихов производ можемо представити као:

$$\lambda X : *.\lambda o : X.\lambda s : X.n\ X\ o\ (\lambda x : X.m\ X\ x\ s).$$

Терм $\lambda x : X.m\ X\ x\ s$ представља функцију која за неко x враћа његов $m-$ти следбеник, па када ту функцију применимо n пута на o заиста добијамо $mn-$ти следбеник броја o.

Нажалост, нећемо моћи да поставимо неке теореме о овим бројевима. Тип исказа о неком $m : \mathrm{Int}$ би био $\mathrm{Int} \to \mathrm{Prop}$. Да бисмо формирали такав терм, потребно нам је правило $(*, \Box, s)$ за неку сорту s. Овиме ћемо се бавити касније, а за сада ћемо склонити правило $(\Box, *, *)$ како бисмо доказали нека својства о том, једноставнијем, систему.

Ламбда рачун са простим типовима

Дефиниција 25. Ламбда рачун са простим типовима дат је спецификацијом $\lambda(\mathcal{S}, \mathcal{A}, \mathcal{R}_{\lambda^\to})$, где је $\mathcal{R}_{\lambda^\to} = \{(*, *, *)\}$. Означавамо га још са $\lambda^\to$.

Овај систем први пут је увео Черч [5].

Пошто нам је скуп правила тако ограничен, једине $\Pi-$апстракције у $\lambda^\to$ биће облика $\Pi x : A.B$ где је $A : \mathrm{Prop}$ и $B : \mathrm{Prop}$. Самим тим, биће немогуће да B зависи од x, па је $\Pi x : A.B = A \to B$, одакле и долази ознака $\lambda^\to$. Касније дајемо и доказе за ова тврђења.

Испоставља се да ће једини изводив секвент који има празан контекст бити $\vdash * : \Box$. То нам већ показује колико је овај систем слабији од Система F, али у непразном контексту ипак можемо нешто урадити.

Теорема 5. У контексту $A : \mathrm{Prop}, B : \mathrm{Prop}$ важи:

$$A \to (A \to B) \to B$$

(или $A \implies (A \implies B) \implies B$ за свака два исказа A и B).

Доказ. Потребно је само да нађемо терм који има наведени тип. За почетак, претпоставимо да имамо доказ од A и доказ од $A \to B$ и означимо их са a и ab редом. Дакле $a : A$ и $ab : A \to B$. Сада лако можемо наћи терм типа B једном применом: $(ab\ a) : B$ (овај терм нам илуструје модус-поненс). Када све то спојимо у једно, добијамо $\lambda a : A.\lambda ab : A \to B.ab\ a : A \to (A \to B) \to B$. $\square$

Нисмо могли рећи „За свако $A : \mathrm{Prop}$ и за свако $B : \mathrm{Prop}$ важи $A \to (A \to B) \to B$" зато што би се то превело у тип $\Pi A : \mathrm{Prop}.\Pi B : \mathrm{Prop}.A \to (A \to B) \to B$ који нам није дозвољен нашим скупом правила (у Систему F је дозвољен правилом $(\Box, *, *)$).

Могућности овог система нису велике. Једини искази које ћемо моћи да посматрамо у њему биће они који се граде из искључиво импликација. Ипак, баш због своје ограничене моћи, $\lambda^{\to}$ биће први систем у коме ћемо моћи да докажемо јаку нормализацију. Оно што следи је доказ строге нормализације за $\lambda^{\to}$, који је преузет из [13].

Лема 14. Једини терм типа $\Box$ је $*$.

Доказ. Прво, приметимо да не можемо увести променљиву типа $\Box$. По правилима за одређивање типа, једина могућност остаје нека сорта односно $*$. □

Дефиниција 26. Терм P је *тип* у $\lambda^{\to}$ када је $\Gamma \vdash P : *$ за неки контекст Γ.

Лема 15. Сваки тип је променљива или је облика $\Pi x : A.B$ за неке типове A и B.

Доказ. По правилима извођења, сви терми типа $*$ су сорте, променљиве или $\Pi x : A.B$. Не постоји сорта чији је тип $*$, па остају променљиве и $\Pi-$апстракције. Како је једино дозвољено правило $(*, *, *)$, мора бити $A : *$ и $B : *$. □

Лема 16. Нека је T тип. Тада је T променљива или је $T = A \to B$ за неке типове A и B.

Доказ. Индукцијом по сложености терма T, делећи на случајеве по леми 15. □

Последица 3. Типови су строго нормализујући.

За доказ строге нормализације за све терме, наивна индукција по термима нажалост не пролази. Потребно је да значајно ојачамо индуктивну претпоставку. Прво ћемо за сваки тип T увести скуп јако нормализујућих терма $L(T)$. Затим ћемо доказати да за свако $t : T$ важи да када у t супституишемо слободне променљиве неким термима добијамо елемент скупа $L(T)$. Специјално, идентичком супституцијом тих променљивих добићемо да је $t \in L(T)$, то јест да је строго нормализабилан.

Дефиниција 27. Скуп SN је скуп строго нормализујућих терама.

Дефиниција 28. За сваки тип A уводимо скуп $L(A)$ који је дефинисан индуктивно:

- За променљиве P, $L(P)$ је скуп свих строго нормализујућих терма p за које постоји контекст Γ тако да $\Gamma \vdash p : P$,
- За $A \to B$, $L(A \to B) = \{m \mid \forall n \in L(A).(m\ n) \in L(B)\}$.

Лема 17.

$$L(P) \subseteq SN.$$

Доказ. Индукцијом по P. Ако је P променљива, из дефиниције $L(P) \subseteq SN$. Нека је $P = A \to B$. Сада је $L(A \to B) = \{m \mid \forall n \in L(A).(m\ n) \in L(B)\}$. Како је $(m\ n) \in L(B)$ за свако $m \in L(A \to B)$ и $n \in L(A)$, можемо закључити да је $(m\ n)$ строго нормализујућ. Одатле следи да је и m строго нормализујућ и доказ је завршен. □

Лема 18.

$$m \in L(P) \wedge m \twoheadrightarrow_\beta m' \implies m' \in L(P).$$

Доказ. Индукцијом по P. Ако је P променљива, онда је $m \in SN$. Тада је и $m' \in SN$ и $m' : P$, па је по дефиницији $m' \in L(P)$. Ако је $P = A \to B$, добијамо да за свако $n \in L(A)$ важи $(m\ n) \in L(B)$. По индуктивној претпоставци је и $(m'\ n) \in L(B)$, па добијамо да је $m' \in L(A \to B)$. □

Следећа лема каже да можемо да „обрнемо" једну $\beta-$редукцију и останемо у скупу $L(T)$, што ће нам касније дати могућност да докажемо да су $\lambda-$абстракције у одговарајућем скупу $L(A \to B)$.

Лема 19.

$$\forall t.(\forall t'.t \to_\beta t' \implies t' \in L(T)) \implies t \in L(T)$$

Доказ. Индукцијом по T. Ако је T променљива, онда је сваки t' добијен једном редукцијом од t јако нормализујућ терм типа T, па је и t јако нормализујућ терм типа T. По дефиницији, $t \in L(T)$.

Ако је $T = A \to B$ ствари се мало компликују. По индуктивној претпоставци имамо:

$$\forall t.(\forall t'.t \to_\beta t' \implies t' \in L(B)) \implies t \in L(B). \tag{1.1}$$

Фиксирајмо неко m за које важи $\forall n.m \to_\beta n \implies n \in L(A \to B)$. Желимо да докажемо $m \in L(A \to B)$. То је еквивалентно са $\forall n \in L(A).(m\ n) \in L(B)$. По индуктивној претпоставци, довољно је доказати:

$$\forall n \in L(A).\forall w.(m\ n) \to_\beta w \implies w \in L(B).$$

За ово тврђење користимо индукцију по низу бета редукција од n: знамо да је $n \in L(A)$, па је $n \in SN$, то јест сваки низ редукција се завршава у неком n_0. Доказујемо да је:

$$\forall w.(m\ n^*) \to_\beta w \implies w \in L(B)$$

за $n \twoheadrightarrow_\beta n^*$.

По леми 18 је $n^* \in L(A)$. Посматратрајмо најдужи низ редукција од n^* и нека је он дужине N. Ако је $N = 0$ онда је $n^* = n_0$, па имамо да је $w = (m'\ n^*)$ где је $m \to_\beta m'$. То нам даје да је $m' \in L(A \to B)$, па је по дефиницији тог скупа $(m'\ n^*) \in L(B)$. Даље индукцијом по N. Ако је $w = (m'\ n^*)$ доказ ради исто као у бази. Сада је $w = (m\ n')$ где је $n^* \to_\beta n'$. Највећи број редукција од n' до n_0 је свакако мањи од N, па по индуктивној претпоставци имамо:

$$\forall w'.w \to_\beta w' \implies w' \in L(B).$$

Сада је, по индуктивној претпоставци 1.1 примењеној на w, и $w \in L(B)$ чиме смо завршили обе индукције. □

Лема 20. Ако за свако $a \in L(A)$ важи $b[x/a] \in L(B)$, онда је $\lambda x : A.b \in L(A \to B)$.

Доказ. Желимо да докажемо да је за свако $a \in L(A)$: $(\lambda x : A.b)\ a \in L(B)$. Знамо да је $x \in L(A)$, па је $b[x/x] \in L(B)$, то јест $b \in L(B)$, па је $b \in SN$. Такође је $a \in SN$. Сада радимо индукцију по збиру максималног броја редукција a и b и доказујемо:

$$\forall w.(\lambda x : A.b)\ a \to_\beta w \implies w \in L(B).$$

Ако је $w = b[x/a]$ по услову леме имамо да је $w \in L(B)$. Ако су a и b у нормалној форми, мора бити $w = b[x/a]$, па смо овиме добили базни случај. Ако је $w = (\lambda x : A.b')\ a$ где је $b \to_\beta b'$, по индуктивној претпоставци имамо да је:

$$\forall w'.w \to_\beta w' \implies w' \in L(B)$$

што по леми 19 даје $w \in L(B)$. Слично можемо урадити случај када је $w = (\lambda x : A.b)\ a'$ где је $a \to_\beta a'$.

Сада, уз примену леме 19 добијамо $(\lambda x : A.b)\ a \in L(B)$. □

Коначно, долазимо до тврђења које смо поменули након последице 3.

Теорема 6. Нека је $FV(b) = \{x_1, x_2, \ldots, x_n\}$, $a_i \in L(A_i)$ и $x_i : A_i \in \Gamma$ за $i = 1, 2, \ldots, n$. Тада $\Gamma \vdash b : B \implies b[x_1/a_1][x_2/a_2]\ldots[x_n/a_n] \in L(B)$.

Доказ. Индукцијом по структури b:

- Ако је $b = x_1$, онда је $\Gamma \vdash b : A_1$ и $b[x_1/a_1] = a_1 \in L(A_1)$.
- Ако је $b = (m\ n)$ мора бити $\Gamma \vdash m : A \to B$ и $\Gamma \vdash n : A$ за неки тип A. По индуктивној претпоставци је $m' = m[x_1/a_1][x_2/a_2]\dots[x_n/a_n] \in L(A \to B)$ и $n' = n[x_1/a_1][x_2/a_2]\dots[x_n/a_n] \in L(A)$, па је по дефиницији скупа $L(A \to B)$ и $(m'\ n') \in L(B)$. Онда је и $(m\ n)[x_1/a_1][x_2/a_2]\dots[x_n/a_n] \in L(B)$.
- Ако је $b = \lambda x : A.m$, онда $\Gamma, x : A \vdash m : M$ и $B = A \to M$. Сада, по индуктивној претпоставци, за свако $a \in L(A)$ важи

  $$m[x_1/a_1][x_2/a_2]\dots[x_n/a_n][x/a] \in L(M).$$

  По леми 20 добијамо да је $\lambda x : A.(m[x_1/a_1][x_2/a_2]\dots[x_n/a_n]) \in L(B)$. Како се променљиве x_i не могу појављивати у типу A, овај терм је идентичан терму $b[x_1/a_1][x_2/a_2]\dots[x_n/a_n]$.

Тиме је индукција готова. □

Теорема 7. Сваки терм m је строго нормализујућ.

Доказ. Рецимо да је $\Gamma \vdash m : M$. Сада је, по теореми 6,

$$m[x_1/x_1][x_2/x_2]\dots[x_n/x_n] \in L(M),$$

где су $x_1, x_2, \dots, x_n$ слободне променљиве у m. Дакле, $m \in L(M)$, па је $m \in SN$. □

Рачун конструкција

Рачун конструкција (понекад λC) је најкомплекснији PTS који ћемо овде разматрати. Увео га је Кокан [11] и његова варијанта чини основу за доказивач Coq. Дат је скупом правила $\mathcal{R}_C = \{(*,*,*), (\Box,*,*), (*\Box,\Box), (\Box,\Box,\Box)\}$. Дакле, практично све П− апстракције су нам дозвољене и не морамо више да пазимо на то.

Испоставља се да, упркос томе што је овај систем толико богат, он и даље задовољава стронгу нормализацију. Наводимо ово тврђење без доказа, који се може пронаћи у [6].

Теорема 8. Сви терми рачуна конструкција су строго нормализујући.

Ово својство биће кључно у доказу следећег.

Теорема 9. Секвент $\vdash f : \Pi A : *.A$ није изводив.

Тип $\Pi A : *.A$ представља тип False што смо навели раније. Ако би постојало f : False, онда би сваки исказ био доказив, јер овај терм заиста говори „за сваки исказ A постоји терм типа A" .

Доказ. Претпоставимо супротно. По теореми 8, можемо претпоставити да је f у нормалној форми (дакле, не садржи редексе). Такође, f нема слободних променљивих јер је контекст празан. Сада је $f = \lambda A : *.f'$, где је $A : * \vdash f' : A$ и једина слободна променљива у f' је A. Индукцијом по структури терма f' доказујемо да ово није могуће. Ако је $f' = A$, онда је $A : * \vdash A : A$. По јединствености типова, мора бити $A =_\beta *$, што није могуће. Ако је $f' = \lambda x : X.Y$, тада је $A : * \vdash \lambda x : X.y : \Pi x : X.Y$, па је поново по јединствености типова $\Pi x : X.Y =_\beta A$ што опет није могуће. Ако је $f' = M\ N$, онда је $A : * \vdash M : \Pi x : X.A$. Како је M у нормалној форми, M је или $\lambda-$апстракција, или је $M = A$. Први случај даје $M = \lambda x : X.f''$ где је $A : * \vdash f'' : A$ што није могуће по индуктивној претпоставци. Други случај даје $\Pi x : X.A =_\beta *$, што поново није могуће. Коначно, ако је f' нека $\Pi-$апстракција, онда је $A : * \vdash f' : s$ где је s сорта, па је $A =_\beta s$, што поново није могуће. [8] □

Рачун конструкција биће основа за наш програм који треба да проверава теореме. Сада ћемо у пар примера илустровати како то ради.

Пример 11. Рецимо да желимо да дефинишемо природне бројеве. Узећемо тип природних бројева $nat : *$ и то прво стављамо у контекст. Разлог што их стављамо у $*$ је што заиста немамо другу сорту у коју бисмо могли да их сместимо (□ нема свој тип, па самим тим не може садржати променљиве). Даље, желимо да преведемо Пеанове аксиоме на рачун конструкција. Прво нам треба једнакост, која још није нигде дефинисана - такозвану Лајбницову једнакост.[3] Идеја је да ако су два броја иста, онда у сваком исказу можемо један да заменимо са другим. Објекат који представља Лајбницову једнакост биће $eq : nat \to nat \to \mathrm{Prop}$. Дакле, она узима 2 природна броја и враћа исказ (о њима). Сада додајемо прве аксиоме, које гласе:

$$eq_i : \Pi x : nat.eq\ x\ x,$$

$$eq_e : \Pi P : nat \to \mathrm{Prop}.\Pi m : nat.\Pi n : nat.(eq\ m\ n) \to P\ m \to P\ n.$$

Прва од ових је сасвим јасна - каже да је сваки број x једнак самом себи. Друга аксиома каже да за сваки предикат типа $nat \to \mathrm{Prop}$ и све

[3]Ова једнакост је различита од $=_\beta$. Релацији $=_\beta$ заправо нећемо моћи да „приступимо" када доказујемо теореме, она је ту за проверу исправности доказа.

природне бројеве m и n, ако су m и n једнаки и важи $P\ m$ онда важи и $P\ n$. Ово ће нам омогућити да докажемо нека основна својства, рецимо да је једнакост комутативна. То тврђење представљено је типом $\Pi m : nat.\Pi n : nat.(eq\ m\ n) \to (eq\ n\ m)$. Терм који доказује дато тврђење почеће са $\lambda m : nat.\lambda n : nat.\lambda p : (eq\ m\ n)$. где је p заправо претпоставка да су m и n исти. Сада желимо да применимо eq_e да добијемо $eq\ n\ m$. За P узећемо $\lambda x.eq\ x\ m$. Терм који доказује комутативност биће:

$$eq_c = \lambda m : nat.\lambda n : nat.\lambda p : (eq\ m\ n).eq_e\ (\lambda x.eq\ x\ n)\ m\ n\ (eq_i\ m)$$

Заиста, $P\ m =_\beta eq\ m\ m$ је доказано термом $(eq_i\ m)$ одакле следи $P\ n =_\beta eq\ n\ m$. Лепота овога је што ми рачунару не морамо ништа да објашњавамо - ми само унесемо терм за који кажемо да је доказ, а његов задатак је да израчуна тип тог терма. У будућности можемо користити eq_c као скраћеницу за наш доказ. Такође, писаћемо $x = y$ као скраћеницу за $eq\ x\ y$.

Сада бисмо да додамо нулу и функцију следбеника: $0 : nat$, $s : nat \to nat$. Аксиоме које везују ова два објекта биће следеће:

$$snotzero : \Pi n : nat.s\ n = 0 \to \text{False}$$
$$sinj : \Pi x : nat.\Pi y : nat.s\ x = s\ y \to x = y$$

Горе се појавило $p \to \text{False}$ где је $p : \text{Prop}$ неки исказ. Ово заправо представља негацију - ако важи p, добијамо контрадикцију.

Остало је још додати индукцију и операције. Индукцију додајемо на следећи начин:

$$ind : \Pi P : nat \to \text{Prop}.P\ 0 \to (\Pi n : nat.P\ n \to P\ (s\ n)) \to \Pi n : nat.P\ n.$$

Операције $+$ и $\cdot$ нећемо разматрати овде зато што терми већ постају прилично велики.

Овим приступом можемо урадити много. Можемо додати логичке операторе $\wedge$ и $\vee$, квантификатор $\exists$ и искључење трећег: $\Pi p : \text{Prop}.p \vee (p \to \text{False})$ (које није доказиво у празном контексту). Ипак, постоје неки проблеми.

Системи већи од рачуна конструкција

Први проблем који се појављује када користимо рачун конструкција је што ми наше скупове објеката стављамо у $*$, иако они немају везе са исказима. Последица тога је да ћемо понекад моћи да докажемо више

ствари него у, рецимо, логици вишег реда са истим контекстом. Решење би било додати сорту која ће бити сорта објеката и први кандидат нам је $\Box$. Ако бисмо додали аксиому $\Box : \Box$, могли бисмо да уводимо промењљиве типа $\Box$. Нажалост, такав систем већ не задовољава строгу нормализацију. Не само то, већ постоји терм типа False (наравно, тај терм није у нормалној форми). [10]

Други начин је додати сорте „изнад" $\Box$. Неки доказивачи теорема узимају овај приступ - имају бесконачну хијерархију сорти Type_i где је $\mathrm{Type}_i : \mathrm{Type}_{i+1}$ и $\mathrm{Prop} : \mathrm{Type}_0$. Тада су сорте изнад Prop намењене за објекте, док Prop држи само исказе. Још једна могућност је додати сорту $\triangle$ и аксиому $\triangle : \Box$ са одговарајућим скупом правила. Размишљао сам о томе, али нисам успео да докажем да ће такави системи бити конзистентни, па сам одлучио да се задржим на рачуну конструкција.

Неки доказивачи имају и додатне типове - такозване индуктивне типове. Идеја је да се у типу дефинишу само конструктори, а систем сам дода шта је потребно. На пример, код природних бројева у примеру 11, ми бисмо само дефинисали $0 : nat$ и $s : nat \to nat$, а систем би по одређеном алгоритму закључио шта све треба да дода.

2 Имплементација

У овом поглављу ћемо укратко описати имплементацију интерактивног доказивача и дати пар примера неких доказа.

Језик који је коришћен за имплементацију је C#. Главне класе у програму су LambdaTerm, PTSDefinition, Context и TypingRules.

LambdaTerm

Задатак ове класе је да од низа карктера направи дрво које представља терме ламбда рачуна (са типовима). Још један, много тежи задатак, је да уради сва преименовања променљивих, проверава једнакост терма (преко $\beta-$нормалне форме) и пази да немамо „илегалне" терме. Ова класа је најбитнија јер се у њој одвијају све трансформације терма и цео програм зависи од ње. Зато је куцана са посебном пажњом и тражи веома специфичан формат у коме се терми уносе, да сигурно не би дошло до грешке.

PTSDefinition

Класа у којој се дефинише PTS, односно њој дајемо скуп сорти, аксиома и правила. Ова класа не би била потребна када бисмо хтели да се ограничимо само на рачун конструкција, али овако је много лакше пребацити се на неки други PTS и добити резултате у њему.

Context **и** TypingRules

Ове две класе заједно чувају контекст и проверавају тип сваког терма. Провера типа урађена је рекурзивно и дрво извођења типа се не изводи експлицитно, мада је увек могуће реконструисати га.

Ове четири класе заједно нам дају све што је у теорији потребно да ми доказујемо неке теореме. Ипак, за човека је готово немогуће да уноси сваки терм сам за себе. Докази неких теорема могу имати и хиљаде карактера, па нам је потребан неки начин да генеришемо сав тај код.

Зато је имплементиран систем у коме је могуће писати функције које ће превести текст који човек напише у одговарајуће терме. Тај систем се у многоме базира на ламбда рачуну без типова, где нам чињеница да је он Тјуринг комплетан даје велику слободу.

Пример 12. Представићемо код који је повезан са примером 11.

```
axiom nat : Prop;
axiom eq : nat->nat->Prop;
infix eq =;
axiom eq_intro : #n:nat.n=n;
axiom eq_elim : #A:nat->Prop.#x:nat.#y:nat.x=y->(A x)->(A y);
```

представља П−апстракцију док ће @ представљати λ−апстракцију. Команда axiom a:A додаје у контекст декларацију a:A, док infix eq = каже да сви x=y треба да се претворе у *eq x y*. Следећи код је доказ комутативности за једнакост.

```
theorem eq_comm : #x:nat.#y:nat.x=y->y=x
{
    let x:nat{
        let y:nat{
            let k:x=y{
                eq_elim @n:nat.n=x x y k (eq_intro x);
            }
        }
    }
}
```

Команда let a:A додаје у контекст декларацију a:A али само у блоку ограниченом витичастим заградама. Све let команде се склањају λ−правилом и проверава се тип добијеног терма. Ако је тај тип исти као онај наведен у теореми, онда се додаје правило које сваки пут када се напише eq_comm замени терм који доказује комутативност. Цео терм излгеда овако:

```
@x:nat.@y:nat.@k:((eq x) y).
(((((eq_elim @n:nat.((eq n) x)) x) y) k) (eq_intro x))
```

Рецимо да имамо терм који доказује p:x=y. Можемо дефинисати функције bin_left p и bin_right p које враћају редом x и y. Сада можемо из p одмах добити доказ за y=x дефинисањем још једне функције:

```
func make_eq_comm t=eq_comm (bin_left t) (bin_right t) t;
```

У овој линији func узима име функције и параметре пре знака =. Затим иде код у коме се евентуално појављују ти параметри. Када у будућности напишемо make_eq_comm p добићемо одмах доказ од y=x, уместо да пишемо сваки пут eq_comm x y p. Ово је посебно корисно када су x и y неки сложенији изрази.

Наводимо сада и доказ транзитивности.

```
theorem eq_trans : #x:nat.#y:nat.#z:nat.x=y->y=z->x=z
{
    let x:nat{
        let y:nat {
            let z:nat {
                let a:x=y{
                    let b:y=z{
                        eq_elim (eq x) y z b a;
                    }
                }
            }
        }
    }
}
```

Поново можемо увести скраћеницу као за комутативност:

```
func make_eq_trans l1 l2=
eq_trans (bin_left l1) (bin_right l1) (bin_right l2) l1 l2;
```

Навешћемо још дефиницију сабирања и доказаћемо асоцијативност. Дужина кода који генерише тај доказ нам говори да вероватно има доста простора за аутоматизацију. С друге стране, терм који доказује асоцијативност има скоро 2000 карактера, тако да смо ипак себи доста олакшали посао.

```
axiom plus : nat->nat->nat;
infix plus +;
axiom plus_succ : #a:nat.#b:nat.a+(succ b)=(succ a+b);
axiom plus_zero : #a:nat.a+zero=a;
theorem a_plus_one : #a:nat.a+(succ zero)=(succ a);
```

Теорему a_plus_one навели смо овде без доказа. Сада следи доказ асоцијативности. У доказу се појављује још пар ствари о којима нисмо дискутовали, мада њихова имена дају сасвим добро објашњење о томе шта се дешава. Још нешто што користимо су угњеждене теореме. Када наведемо теорему унутар теореме и докажемо је, онда је можемо користити у остатку доказа.

```
theorem plus_assoc : #a:nat.#b:nat.#c:nat.(a+b)+c=a+(b+c)
{
    let a:nat{
        let b:nat{
            theorem l1 : (a+b)+zero=a+(b+zero)
            {
                theorem l11 : b+zero=b{plus_zero b;}
                theorem l12 : a+(b+zero)=a+b{
                    apply_f_to_eq @x:nat.a+x l11;
                }
                theorem l13 : a+b=(a+b)+zero
                {eq_comm (a+b)+zero a+b (plus_zero a+b);}
                theorem l14 : a+(b+zero)=(a+b)+zero
                {eq_trans a+(b+zero) a+b (a+b)+zero l12 l13;}
                make_eq_comm l14;
            }
            theorem l2 : #c:nat.(a+b)+c=a+(b+c)->
                                (a+b)+(succ c)=a+(b+(succ c))
            {
                let c:nat{
                  let h : (a+b)+c=a+(b+c){
                    theorem l21 : (a+b)+(succ c)=(succ (a+b)+c)
                    {plus_succ a+b c;}
                    theorem l22 : a+(b+(succ c))=a+(succ b+c)
                    {apply_f_to_eq @x:nat.a+x (plus_succ b c);}
                    theorem l23 : a+(b+(succ c))=(succ a+(b+c))
                    {make_eq_trans l22 (plus_succ a b+c);}
                    theorem l24 : (a+b)+(succ c)=(succ (a+b)+c)
                    {plus_succ a+b c;}
                    theorem l25 : (succ (a+b)+c)=(succ a+(b+c))
                    {apply_succ_to_eq h;}
                    theorem l26 : (a+b)+(succ c)=(succ a+(b+c))
                    {make_eq_trans l24 l25;}
                    make_eq_trans l26 (make_eq_comm l23);
                  }
                }
            }
            ind @c:nat.(a+b)+c=a+(b+c) l1 l2;
        }
    }
}
```

Цео код са још неколико теорема и свим дефиницијама може се наћи у фајлу који је приложен под именом proof.txt.

Пример 13. Раније смо поменули да додавање аксиоме □ : □ руши конзистентност нашег система. Ипак, ми у програму можемо дефинисати такав систем и проверити да заиста постоји терм типа False. Дакле, у наредном ћемо радити у рачуну конструкција са □ : □ као додатном аксиомом. Следећи код даје терм наведен у [14], који програм тачно проверава као доказ контрадикције у овом систему.

```
func false=#r:Prop.r;
func not r=r->false;
func p A=A->Prop;
func U=#x:Type.(((p (p x))->x)->(p (p x)));
func tau t=@X:Type.@f:((p (p X))->X).
                    @P:(p X).(t @x:U.(P (f ((x X) f))));
func sigma s=((s U) @t:(p (p U)).(tau t));

func Delta=@y:U.(not #P:(p U).
                      (((sigma y) P)->(P (tau (sigma y)))));
func Omega=reduced (tau @P:(p U).#x:U.(((sigma x) P)->(P x)));

func fp=(@0:#P:(p U).((#x:U.(((sigma x) P)->(P x)))->(P Omega)).
(((0 Delta) @x:U.@2:((sigma x) Delta).@3:#P:(p U).
(((sigma x) P)->(P (tau (sigma x)))).(((3 Delta) 2)
@P:(p U).(3 @y:U.(P (tau (sigma y))))))
@P:(p U).(0 @y:U.(P (tau (sigma y)))))
@P:(p U).@1:#x:U.(((sigma x) P)->(P x)).((1 Omega)
@x:U.(1 (tau (sigma x)))));

theorem f : false
{
    fp;
}
```

3 Закључак

Дефинисали смо ламбда рачуне са и без типова и доказали многа њихова својства. Најбитнији докази били су докази Черч-Росер теореме о јединствености $\beta-$нормалне форме и доказ строге нормализације за ламбда рачун са једноставним типовима. Показали смо да строга нормализација, заједно са Черч-Росером, даје конзистентност система, у смислу да је тип $\Pi A : *.A$ празан. Представили смо рачун конструкција и на основу њега смо направили програм који омогућава доказивање теорема са аутоматском провером.

Постоји доста простора за напредак. Већ смо поменули проблеме рачуна конструкција и могуће поправке. Такође, што се саме имплементације тиче, свакако има много ствари које би могле да се додају као на пример аутоматизовање доказа који су чисто алгоритамски.

Литература

[1] H. P. Barendregt. *The Lambda Calculus, its syntax and semantics* 1984.

[2] H. P. Barendregt. *Introduction to generalized type systems* 1991.

[3] H. P. Barendregt. *Typed lambda calculi* 1992.

[4] A. Church, J. B. Rosser. *Some properties of conversion.* 1936. https://www.ams.org/journals/tran/1936-039-03/S0002-9947-1936-1501858-0/S0002-9947-1936-1501858-0.pdf

[5] A. Church. *A formulation of the simple theory of types* 1940. https://www.jstor.org/stable/2266170

[6] H. Geuvers, M.-J. Nederhof *Modular proof of strong normalization for the calculus of constructions* 1991. https://www.researchgate.net/publication/220676876_Modular_Proof_of_Strong_Normalization_for_the_Calculus_of_Constructions

[7] M. Takahashi. *Parallel Reductions in λ-Calculus* 1995. https://www.sciencedirect.com/science/article/pii/S0747717189800458

[8] Z. Luo. *An Extended Calculus of Constructions* 1990. https://era.ed.ac.uk/bitstream/handle/1842/12487/Luo1990.Pdf

[9] J.-Y. Girard, Y. Lafont, P. Taylor. *Proofs and Types* 1989. https://www.paultaylor.eu/stable/prot.pdf

[10] J.-Y. Girard. *Interprétation fonctionnelle et Élimination des coupures de l'arithmétique d'ordre supérieur* 1972.

[11] T. Coquand, G. Huet. *The Calculus of Constructions* 1988. https://www.sciencedirect.com/science/article/pii/0890540188900053

[12] The Coq development team. *The Coq Proof Assistant documentation* https://coq.inria.fr/documentation

[13] Marco Benini. *Mathematical Logic, part 5: strong normalisation of the simple theory of types*, video lecture. https://www.youtube.com/watch?v=nI5wFfhrE7k

[14] A. J. C. Hurkens. *A simplification of Girard's paradox* 1995. https://www.cs.cmu.edu/~kw/scans/hurkens95tlca.pdf