Metalogika i Coq

sprawozdanie z projektu

Paweł Wieczorek

2010

---

1 Wstęp.

Niniejszy dokument opisuje prace nad studenckim projektem mającym na celu próbę formalizacji twierdzeń logiki matematycznej dotyczących samej logiki.

Prace nad formalizacją rachunku zdań opierały się głównie o książkę Rutkiewicza [1], prace nad formalizacją logiki pierwszego były dodatkowo wspierane przez książkę Grzegorczyka [2].

Tworząc skrypty kierowałem się ideą, aby dowodzić dużo drobnych lematów i korzystać w dużej mierze z automatyzacji przy dowodzeniu następnych oraz większych twierdzeń. W ten sposób uzyskałem łatwiejszy w modyfikacjach kod, co było bardzo pomocne gdy decydowałem się na modyfikację fundamentalnych definicji, oraz umożliwiało tworzenie zwięzłych dowodów większości lematów i twierdzeń.

Niestety niektóre fragmenty dowodów zostały przyjęte z góry (`admit`), również najbardziej *doniosła* część projektu - twierdzenie o istnieniu modelu (o pełności) dla logiki pierwszego rzędu - nie obeszła się bez tej taktyki.

Całość jest zawarta w paczce `metalogic`, która wewnętrznie jest podzielona na trzy mniejsze paczki `Pc`, `Fol`, `Util` oznaczające odpowiednio część dotyczącą rachunku zdań, logiki pierwszego rzędu oraz małą biblioteczkę z pomocniczymi lematami, dotyczącymi głównie zbiorów, niezwiązanymi tematycznie z projektem.

2 Przyjęty system dowodzenia.

Pracę nad formalizacją zacząłem od rachunku zdań. Używany przeze mnie system jest wzięty z książki [1], czyli są to poniższe aksjomaty i reguły wnioskowania, a sam język zawiera tylko implikację oraz negację[1]. Dla logiki pierwszego rzędu są dołożone jeszcze oba kwantyfikatory oraz symbol $\top$.

A1	$\varphi \to \phi \to \varphi$
A2	$(\varphi \to \phi \to \gamma) \to (\varphi \to \phi) \to \varphi \to \gamma$
A3	$\neg\varphi \to \varphi \to \gamma$
A4	$(\varphi \to \neg\varphi) \to \neg\varphi$
A5	$(\neg\varphi \to \varphi) \to \varphi$

[1] Autor nie podał źródła aksjomatów z jakich korzysta, ale w książce Grzegorczyka oraz innych źródłach wspomina się o systemie Łukasiewicza, który również jest nad ubogim językiem i ma bardzo podobne aksjomaty.

$$\frac{T \vdash \varphi \to \phi \qquad T \vdash \varphi}{T \vdash \phi}\ \text{MP}$$

Autor książki nie załączył wyżej wymienionych aksjomatów na sztywno do systemu wnioskowania. Posługiwał się zbiorem A_0 zawierającym wszystkie formuły o takich schematach, a twierdzenia formułował w następujący sposób: *„Jeżeli $A_0 \subseteq T$ to w T zachodzi twierdzenie o..."*. Pozwoliłem sobie na nieistotną modyfikację i dołączyłem zbiór A_0 na sztywno do systemu, bo ułatwiło mi to pracę.

Do pracy nad logiką pierwszego rzędu ostatecznie użyłem tego samego systemu oraz dołączyłem do niego aksjomaty i reguły wnioskowania dotyczące kwantyfikatorów z książki Grzegorczyka (system L^+). Było to dobrym kompromisem pomiędzy łatwym przeniesieniem wykonanych już prac (opierających się o stare aksjomaty i ubogi język) oraz korzystaniem z innej książki (z innymi aksjomatach). Ta synteza była wykonalna bo jedyna istotna różnica między systemami z tych dwóch książek jest taka, że Pan Grzegorczyk posługuje się bogatszym językiem i większym zbiorem aksjomatów. Nowe aksjomaty i reguły wyglądają następująco:

$$\begin{aligned} &\text{L1} \quad \varphi \to \exists x\, \varphi \\ &\text{L2} \quad \forall x\, \varphi \to \varphi \end{aligned}$$

$$\frac{T \vdash \phi \to \varphi}{T \vdash \exists x\, \phi \to \varphi}\ \text{R1} \qquad \frac{T \vdash \varphi \to \phi}{T \vdash \varphi \to \forall x\, \phi}\ \text{R2} \qquad \frac{T \vdash \varphi}{T \vdash \varphi[x \rightsquigarrow t]}\ \text{SB}$$

Reguła SB wymaga, aby term t był podstawialny w φ w miejsce zmiennej x. Reguły R1 i R2 wymagają, aby zmienna x nie występowała w formule φ.

We wspomnianej książce można znaleźć również definicję systemu bez reguł wnioskowania dotyczących kwantyfikatorów (system L - rozdział *Redukcja reguł kwantyfikatorowych do aksjomatów*). Pracę początkowo rozpocząłem z tym systemem, ale doszedłem do wniosku że nie rozumiem wszystkich szczegółów technicznych i się z niego wycofałem.

Dowody w używanym systemie dowodzenia są bardzo nieintuicyjne, są po prostu ciągiem syntaktycznych operacji, które bardziej przedstawiają spryt autora w stosowaniu dostępnych operacji na formułach, niż jakąś intuicję stojąca za dowodzonym twierdzeniem. Z drugiej strony mała liczba reguł wnioskowania powoduje uproszczenie dowodów meta twierdzeń.

Ponieważ książka Grzegorczyka opierała się o podobny system to mogłem przepisywać z niej dowody wewnątrz systemu, których brakowało mi w książce Rutkowskiego. Ciekawym zagadnieniem jest to, że posługiwanie się ubogim językiem, co miało upraszczać dowody, sprawiło mi dużo problemu przy dowodzie twierdzenia o dedukcji.

2.1 Definicje.

Język jakim się posługiwałem przez większość czasu nie zawierał symbolu $\top$, dołożyłem go pod koniec moich prac ponieważ był mi potrzebny sposób aby łatwo konstruować zdanie w dowolnym języku.

Język (a właściwie jego sygnatura) został zdefiniowany jako rekord opisujący typ reprezentujący symbole funkcyjne; funkcje zwracające ich arność; sposób rozstrzygania równości na symbolach; funkcje zwracająca symbol na podstawie numeru oraz dowód, że każdy symbol ma swój numer (oraz takie same pola dotyczące symboli relacyjnych).

Pola dotyczące numeracji mają zapewnić przeliczalność zbioru wszystkich formuł w danym języku. Niestety ale funkcje numerujące formuły nie zostały wykonane - ich istnienie zostało

założone. Powodem niedokończenia tej części projektu jest duża część pracy, którą planowałem wykonać na koniec, związaną z dowodzeniem twierdzeń opierających się o arytmetykę. Sam pomysł na numerację formuł jest natomiast łatwy do wykonania *na papierze* opierając się o funkcję pary Cantora.

```
Record FolLang := folLang
  {
      folFuncs : Set;
      folFuncEqDec: ∀ f1 f2 : folFuncs, {f1 = f2} + {f1 ≠ f2};
      folFuncAr: folFuncs → nat;
      folRels : Set;
      folRelAr: folRels → nat;
      folRelEqDec: ∀ r1 r2: folRels, {r1 = r2} + {r1 ≠ r2};
      folSomeConst: { c : folFuncs | folFuncAr c = 0 };
      folFuncNum: nat → folFuncs;
      folFuncCnt: ∀ f, ∃ n, folFuncNum n = f;
      folRelNum: nat → folRels;
      folRelCnt: ∀ r, ∃ n, folRelNum n = r
  }.
```

Definicja termów oraz formul nad danym językiem L jest następująca:

```
Inductive folTerm : Set :=
| folVar : string → folTerm
| folApp : ∀ f:folFuncs L, folTermList (folFuncAr L f) → folTerm

with folTermList : nat → Set :=
| folTnil : folTermList 0
| folTcons : ∀ n, folTerm → folTermList n → folTermList (S n)
.

Inductive folProp :=
| folTrue : folProp
| folAtom : ∀ r:folRels L, folTermList (folRelAr L r) → folProp
| folImpl : folProp → folProp → folProp
| folNeg : folProp → folProp
| folForall : string → folProp → folProp
| folExists : string → folProp → folProp
.
```

Dowód formuły w dostępnych źródłach był zdefiniowany jako skończony ciąg formuł, gdzie ostatni element ciągu to dowodzona formuła oraz każdy element jest aksjomatem albo wynika z elementów poprzednich za pomocą reguły wnioskowania. Posługiwanie się taką definicją dowodu było niewygodne i nie mogłem daleko się posunąć w pracach. Dopiero po zmianie definicji na drzewo, takie jak systemie naturalnej dedukcji, pojawiły się większe postępy w projekcie. Problem z poprzednimi definicjami dotyczył indukcji strukturalnej, mianowicie robiąc indukcje po dowodzie otrzymywałem nieprzydatne założenia indukcyjne.

```
Inductive folDeriv L (T:FolPropSet L) : folProp L → Type :=
 | folDerivAX1: ∀ a b, folDeriv T (FolAx1 a b)
 | folDerivAX2: ∀ a b c, folDeriv T (FolAx2 a b c)
 | folDerivAX3: ∀ a b, folDeriv T (FolAx3 a b)
```

| *folDerivAX4*: ∀ *a*, *folDeriv T* (*FolAx4 a*)
| *folDerivAX5*: ∀ *a*, *folDeriv T* (*FolAx5 a*)
| *folDerivL1*: ∀ *x a*, *folDeriv T* (*FolGxL1 x a*)
| *folDerivL2*: ∀ *x a*, *folDeriv T* (*FolGxL2 x a*)
| *folDerivSubst*: ∀ *x t a*, *FolSubstCorr L x t a* → *folDeriv T a* → *folDeriv T* (*a* [*x* ⇝ *t*])
| *folDerivT*: *folDeriv T* (*folTrue L*)

| *folDerivAX*: ∀ *p*, *In T p* → *folDeriv T p*
| *folDerivMP*: ∀ *p q*, *folDeriv T* (*q* ⟶ *p*) → *folDeriv T q* → *folDeriv T p*
| *folDerivR1*: ∀ *x p q*, *folProp_noFV L q x* → *folDeriv T* (*p* ⟶ *q*) → *folDeriv T* (([[*E x*]] *p*) ⟶ *q*)

Modele również zostały sformalizowane jako rekordy:

```
Record FolStruct := folStruct {
        folStrUniv : Type;
        folStrNonEmpty: inhabited folStrUniv;
        folStrFunc : ∀ (f : folFuncs L), nAry (folFuncAr L f) folStrUniv;
        folStrRel : ∀ (r : folRels L), nAryP (folRelAr L r) folStrUniv
    }
.
```

Jego pola zawierają typ reprezentujący uniwersum, dowód że typ jest zamieszkany oraz funkcje interpretujące symbole funkcyjne i relacyjne. Funkcje *nAry* i *nAryP* obliczają odpowiednio typ funkcji, najprościej je opisać poniższymi równaniami:

$$\begin{aligned} nAry\ n\ U &= nAryB\ n\ U\ U \\ nAryP\ n\ U &= nAryB\ n\ Prop\ U \\ nAryB\ n\ B\ U &= \underbrace{U \to \cdots \to U}_{n} \to B \end{aligned}$$

3 Najciekawsze fragmenty.

3.1 Dowodzenie twierdzeń wewnątrz systemu.

Do budowania dowodów wewnątrz rozważanego systemu zostały skonstruowane specjalne taktyki, dodatkowo dzięki notacji dla taktyk są one widoczne jako jedna taktyka, pobierająca polecenie jako swój argument, dla ich użytkownika.

Taktyki można opisać następującą składnią:

```
fol_pose d a AX p [env T]
fol_pose d a MP p [env T] from D2 D1
fol_pose d a SB p [env T] from D
fol_pose d a R1 p [env T] from D
fol_pose d a R2 p [env T] from D
fol_pose d a TH p [env T] [eauto]
fol_pose d a REN  [env T] from D
fol_pose QED
```

Taktyki nie operują na celu, lecz dodają zbudowane drzewa do środowiska - stąd też `pose` w nazwie. Pierwszy argument dla każdej taktyki to identyfikator jaki ma być nadany skonstruowanemu drzewu; drugi to identyfikator formuły w korzeniu tego drzewa.

Trzeci argument identyfikuje konkretną taktykę, z punktu widzenia użytkownika to nazwa reguły wnioskowania jaka ma być użyta. Czwarty argument to formuła jaka ma być w korzeniu drzewa. Opcjonalny argument `env T` to podanie teorii w jakiej konstruujemy dowód, musi być podany wtedy gdy taktyki nie mogą się same jego domyśleć na podstawie celu.

Taktyki `AX`, `MP`, `SB`, `R1`, `R2` oznaczają odpowiednio: podanie formuły będącej aksjomatem; użycie reguły wnioskowania, gdzie `D2` to identyfikator drzewa zawierającego dowód implikacji, a `D1` to identyfikator drzewa zawierającego dowód przesłanki; użycie reguły podstawiania oraz dwie reguły wprowadzające kwantyfikator, gdzie `D` oznacza jedyną przesłankę tych reguł.

Taktyka `REN` służy do przemianowania istniejącego już drzewa. Taktyka `TH` służy do wykorzystania jakiegoś już twierdzenia znajdującego się w bazie podpowiedzi, wewnętrznie wykorzystuje `auto` , lub `eauto` w przypadku podania opcjonalnego parametru. Można tą taktykę traktować jako wykorzystanie reguły wtórnej.

Taktyka `QED` jest opakowaniem na auto i została dodana w celach estetycznych, aby rozwiązywać cel do którego było potrzeba było skonstruować drzewo w rozważanym systemie.

Przykładowy dowód obrazujący użycie taktyk. Warto zwrócić uwagę, że choć reprezentuję teraz dowód jako drzewo to dzięki notacją udało się zachować oryginalny styl dowodu jako ciąg formuł. Dzięki czemu większość dowodów wewnątrz dowodzonych twierdzeń zgadza się co do wiersza z źródłem.

```
Theorem folDeriv_impl_perm: ∀ L T (p q r:folProp L),
        folDeriv T (p -> q -> r)
    → folDeriv T (q -> p -> r).
Proof.
intros L T p q r d1.
fol_pose d2 a2 AX ( (p -> q -> r) -> (p -> q) -> (p -> r)).
fol_pose d3 a3 MP ( (p -> q) -> (p -> r) ) from d2 d1.
fol_pose d4 a4 TH ( ((p -> q) -> p -> r) -> (q -> p -> q) -> q -> p -> r ).
fol_pose d5 a5 MP ((q -> (p -> q)) -> q -> p -> r) from d4 d3.
fol_pose d6 a6 AX (q -> p -> q).
fol_pose d7 a7 MP (q -> p -> r) from d5 d6.
fol_pose QED.
Qed.
```

3.2 Twierdzenie o zwartości.

Twierdzenie 1 (O zwartości)**.** *Zbiór formuł jest niesprzeczny wtedy i tylko gdy każdy jego skończony podzbiór jest niesprzeczny.*

Twierdzenie wykorzystuje w sobie, na pozór nietrywialne, pojęcie podzbioru skończonego. Obawiałem się, że takie twierdzenia wykorzystujące bezpośrednio pojęcia z teorii mnogości będą bardzo trudne w formalizacji. W przypadku tego twierdzenia okazało się przeciwnie, to pierwsze *większe* twierdzenie jakie udało mi się sformalizować, nawet gdy nie posługiwałem się dobrą definicją dowodu.

W całym projekcie posługiwałem się typem `Ensemble` jako odpowiednikiem pojęcia zbioru z teorii mnogości, bardzo dobrze go oddaje moim zdaniem. Pojęcie zbioru skończonego dla tego typu jest zdefiniowane już w bibliotece standardowej przez predykat `Finite`.

Poniżej załączam skrypt pokazujący jak dzięki używaniu wielu drobnych lematów i automatyzacji udało mi się utworzyć zwięzły skrypt dla tego twierdzenia.

```
Theorem FolCompactness: ∀ L (T:FolPropSet L),
 FolConsistent T ↔ (∀ S, Included S T → Finite S → FolConsistent S).
Proof.
intros.
split.
 intros; eauto using FolConsistent_subset.

intros.
intro IncT.

destruct IncT as [p incProof].
 destruct incProof as [Tp Tnp].

destruct FolProof_need_finite_axs2 with L T p (!p) as [C HC]; auto.
fol_extracthyps.
assert (FolInconsistent C) as incC.
∃ p; split; auto with metalogic_db.
destruct H with C; auto.
Qed.
```

3.3 Twierdzenie o dedukcji.

Pracę nad twierdzeniem o dedukcji były z początku głównym motorem moich prac nad projektem. Dzięki trudnościami w dowodzie tego twierdzenia dla rachunku zdań podjąłem, wspomnianą już, decyzję o reprezentacji drzewnej oraz skonstruowałem wspomniane taktyki do tworzenia dowodów wewnątrz systemu.

Przy rachunku zdań zauważyłem duże korzyści z korzystania ubogiego języka i małej liczby reguł wnioskowania. Sam dowód opiera się o indukcję strukturalną na drzewie dowodu, co dla używanego systemu oznaczało małą liczbę przypadków do rozważenia, a dzięki ubogiemu językowi konstruowanie dowodów wewnątrz systemu na potrzeby twierdzenia było proste.

Sytuacja była o wiele bardziej skomplikowana dla logiki pierwszego rzędu. Nie dość że pojawiły się nowe reguły wnioskowania, co zwiększało rozmiar dowodu, to dodatkowo skonstruowanie dowodów wewnątrz systemu wymagało użycia większej ilości twierdzeń wewnątrz systemu, jak np praw De Morgana dla kwantyfikatorów.

Odczułem tutaj negatywny wpływ prostoty systemu jakim się posługuję, ponieważ nie byłem w stanie skonstruować tych dowodów samodzielnie. Dowody większości twierdzeń znalazły się w książce Grzegorczyka, jednak mimo tego udogodnienia odczułem kolejny negatywny skutek, tym razem wynikający bezpośrednio z ubogiego języka.

Przypatrzmy się poniższym regułom, reguły importacji i eksportacji są w systemie jakim posługuje się Pan Grzegorczyk.

$$\frac{T \vdash \varphi \to \phi}{T \vdash \varphi \to \forall x\ \phi}\ \text{R2}$$

$$\frac{T \vdash \phi \to \varphi \to \gamma}{T \vdash \phi \wedge \varphi \to \gamma}\ \text{import} \qquad \frac{T \vdash \varphi \wedge \phi \to \gamma}{T \vdash \varphi \to \phi \to \gamma}\ \text{export}$$

Załóżmy teraz, że φ_1 oraz φ_2 są zdaniami oraz, że skonstruowaliśmy dowód twierdzenia $\varphi_1 \to \varphi_2 \to \phi$. Problem jaki napotkałem to konstrukcja dowodu twierdzenia $\varphi_1 \to \varphi_2 \to \forall x\ \phi$.

Reguła wnioskowania $R2$ pozwala skonstruować jedynie dowód $\varphi_1 \to \forall x\ (\varphi_2 \to \phi)$, dalej trzeba by skorzystać z kolejnych twierdzeń których dowody wewnątrz system znałem jedynie przy wykorzystaniu właśnie dowodzonego twierdzenia. W książce Grzegorczyka ten problem jest rozwiązany wykorzystując prawa importacji i eksportacji:

$$\dfrac{\dfrac{\dfrac{T \vdash \varphi_1 \to \varphi_2 \to \phi}{T \vdash \varphi_1 \wedge \varphi_2 \to \phi}\ \text{import}}{T \vdash \varphi_1 \wedge \varphi_2 \to \forall x\ \phi}\ \text{R2}}{T \vdash \varphi_1 \to \varphi_2 \to \forall x\ \phi}\ \text{export}$$

Zakodowanie koniunkcji wewnątrz ubogiego języka i dowiedzenie tych reguł nie okazało się trywialne do wykonania, wręcz przeciwnie. Domniemawszy, że system którym się posługuje ma związek z systemem Łukasiewicza udało mi się znaleźć dokument [3] zawierający dowody, które mogłem przepisać do mojego projektu.

Mimo pojawienia się wielu gotowych twierdzeń nadal nie byłem w stanie dowieść potrzebnych praw. Dowód udało mi się skonstruować dopiero gdy, zakodowałem koniunkcję za pomocą zakodowanej alternatywy, tzn nie $\varphi \wedge \phi \equiv \neg(\varphi \to \neg\phi)$, lecz $\varphi \wedge \phi \equiv \neg(\neg\varphi \vee \neg\phi)$, co dało ostatecznie $\neg(\neg\neg\varphi \to \neg\phi)$. Sam dowód udało mi się skonstruować w sposób przypadkowy.

Trudność w dowodzeniu banalnych twierdzeń wewnątrz tak prymitywnego systemu sprawia, że jestem ciekawy czy korzyści z tego płynące są rzeczywiście opłacalne.

3.4 Rozszerzanie do teorii zupełnej.

Twierdzenie 2. *Jęzeli T jest niesprzecznym zbiorem formuł to istnieje jego nadbziór, który jest niesprzeczny i zupełny.*

„Papierowy" dowód tego twierdzenia opiera się o następującą konstrukcję: Niech φ_n oznacza formułę, której numerem jest n.

$$\begin{aligned} T_0 &= T \\ T_{n+1} &= \begin{cases} T_n & : T_n \vdash \varphi_n \\ T_n \cup \{\neg\varphi_n\} & : T_n \nvdash \varphi_n \end{cases} \\ T^* &= \bigcup_{n\in\mathbb{N}} T_n \end{aligned}$$

Wystarczy pokazać, że każda teoria T_n jest niesprzeczna, oraz, że dla każdej formuły istnieje T_k, taka że dowodliwa jest formuła albo jej negacja. Wykorzystując te twierdzenia oraz fakt, że dowód wymaga tylko skończonej liczby aksjomatów, dowodzi się niesprzeczności i zupełności nieskończonej sumy tych teorii.

Teorio mnogościowy charakter tego dowodu sprawił mi trochę trudności na początku. Chociaż dowód można uznać za konstruktywny, bo został podany przepis jak otrzymać pożądany nadzbiór, to z punktu widzenia formalizacji w systemie Coq wymaga on użycia prawa wyłącznego środka. Problem z nieskończoną sumą jest praktycznie rozwiązany przez definicję `Ensemble`[2]. Definicja sumy rodziny zbiorów w teorii mnogości jest następująca:

$$x \in \bigcup \mathcal{R} \Leftrightarrow \exists R.R \in \mathcal{R} \wedge x \in R$$

[2] `Ensemble U: U -> Prop.`

Co można zinterpretować jako *Aby pokazać, że dany element należy do sumy, muszę wskazać zbiór z rodziny do którego on należy.* Podobnie interpretuję typ `Ensemble`, że term o takim typie reprezentuje jakiś zbiór na zasadzie, że jest funkcją, która mi mówi co muszę udowodnić aby pokazać że jej argument należy do reprezentowanego zbioru.

Wykorzystując te laickie intuicje można prosto przepisać do systemu Coq teoriomnogościową definicję sumy rodziny zbiorów:

```
Definition BigUnionR (R : Ensemble (Ensemble A)) : Ensemble A :=
 fun p ⇒ ∃ a, In _ R a ∧ In _ a p.
```

Nieskończona suma zbiorów nie była jedynym zadaniem przy formalizacji tego dowodu, kolejnym i trudniejszym był sposób konstruowania tych teorii. Moja pierwsza próba polegała na skonstruowaniu funkcji, która dla danego T oraz n skonstruuje po prostu zbiór T_n dodając skończoną liczbę - n - formuł do niego. Potrzeba użycia prawa wyłącznego środka nie pozwoliła mi skonstruować takiej funkcji, gdyż typy o typie `Prop` nie są obliczeniowe i nie mogłem robić eliminacji na termie `classic`.

Problem udało mi się rozwiązać konstruując indukcyjnie predykat $P(T, X, n)$, mówiący że X jest n-tym zbiorem z konstrukcji dla $T_0 = T$. Poniżej jego definicja w systemie Coq dla rachunku zdań. Wersja dla logiki pierwszego rzędu różni się jedynie tym, że rozważane są tylko formuły zamknięte oraz jest parametryzowana sygnaturą języka.

```
Inductive ExtConstr T : PcPropSet → nat → Prop :=
 | pcConstr0 :
     ExtConstr T T 0
 | pcConstrSnNP : ∀ T' n, let p := pcPropNum n in
     ExtConstr T T' n → (T' ⊬ p) → ExtConstr T (T' ∪ {!p}) (S n)
 | pcConstrSnP : ∀ T' n, let p := pcPropNum n in
     ExtConstr T T' n → (T' ⊢ p) → ExtConstr T T' (S n)
.
```

Dowód jest wykonany w całości, jedynie funkcje numerujące wszystkie formuły w danym języku są założone. Dla logiki pierwszego rzędu formuła numerująca wszystkie zdania, w oparciu o założoną funkcję dla wszystkich formuł, jest zrobiona.

3.5 Twierdzenia o istnieniu modelu oraz o pełności.

Za najbardziej doniosłą część projektu uważam niedokończone twierdzenie o istnieniu modelu (Twierdzenie Henkina). Ponieważ nazwy mogą być niejednoznaczne przytoczę twierdzenia.

Twierdzenie 3 (Twierdzenie o istnieniu modelu)**.** *Teoria T ma model wtedy i tylko wtedy gdy jest niesprzeczna.*

Twierdzenie 4 (Twierdzenie o pełności)**.** *Zdanie φ jest dowodliwe w teorii T wtedy i tylko wtedy, gdy jest spełnione w każdym modelu tej teorii.*

Dowód, że twierdzenie o pełności wynika z twierdzenia o istnieniu modelu jest skończony. Wykazałem też implikację w drugą stronę.

Dowód twierdzenia o istnieniu modelu nie jest kompletny. W „papierowym” dowodzie twierdzenia dąży się do skonstruowania modelu dla teorii, model buduje się ze stałych (zamkniętych) termów. W tym celu należy rozszerzyć teorię do teorii opisowo konstruktywnej oraz zupełnej. Opisowa konstruktywność oznacza, że każdy element jakiego istnienie możemy wywnioskować „ma swoją nazwę”, w postaci jakiegoś stałego termu.

Rozszerzanie wykonuje się w następujący sposób: Niech $\varphi_n(x_n)$ oznacza formułę z tylko jedną zmienną wolną x_n. Następnie indukcyjnie tworzy się nieskończony ciąg formuł Φ_n:

$$\Phi_n \equiv \exists x_n \varphi_n(x_n) \rightarrow \varphi_n(c)$$

Gdzie c to jedna ze stałych dodanych do języka, taka że nie występuje we wcześniej utworzonych formułach Φ oraz formule φ_n. Formuły Φ staną się potem aksjomatami rozszerzonej teorii, zapewniającymi że nowo dodane stałe są przydzielonymi „prywatnymi nazwami" do obiektów jakich istnienie możemy wywnioskować.

Aksjomaty i rozszerzoną teorię (nad rozszerzonym językiem) tworzymy za pomocą znanej już konstrukcji teoriomnogościowej.

$$\begin{aligned} T_0 &= T \\ T_{n+1} &= T_n \cup \{\Phi_n\} \\ T^* &= \bigcup_{n \in \mathbb{N}} T_n \end{aligned}$$

Następnie rozszerzamy T^* do teorii zupełnej $\mathcal{T}$.

Następnie tworzony jest model, którego uniwersum to termy stałe rozszerzonego języka. Symbole funkcyjne oraz relacyjne są interpretowane w następujący sposób:

$$\begin{aligned} f^{\mathfrak{A}}(x_1^{\mathfrak{A}}, \cdots, x_n^{\mathfrak{A}}) &\quad = \quad f(x_1, \cdots, x_n) \\ r^{\mathfrak{A}}(x_1^{\mathfrak{A}}, \cdots, x_n^{\mathfrak{A}}) &\quad \text{wtw} \quad \mathcal{T} \vdash r(x_1, \cdots, x_n) \end{aligned}$$

To znaczy, interpretacją symbolu funkcyjnego f jest funkcja zwracająca term reprezentujący aplikację danego symbolu funkcyjnego do termów reprezentujących jej argumenty. Natomiast relacja r jest spełniona dla takich argumentów wtedy i tylko wtedy gdy w rozszerzonej teorii jest dowodliwa odpowiednia formułą atomowa.

Poprawność konstrukcji weryfikuje twierdzenie

Twierdzenie 5. *Dla każdego zdania φ zachodzi:*

$$\mathfrak{A} \models \varphi \textit{ wtw } \mathcal{T} \vdash \varphi$$

Wracając do tematu projektu, nie udało mi się sformalizować etapu rozszerzania teorii do teorii opisowo konstruktywnej. Z samym twierdzeniem pracowałem zaczynając od końca, tzn zakładałem potrzebne pod-twierdzenia i pokazywałem twierdzenie, następnie powtarzałem tę metodę na pod twierdzeniach.

Udało mi się sformalizować nadawanie pożądanej interpretacji symbolom relacyjnym i funkcyjnym oraz konstrukcję uniwersum. Dowód poprawności konstrukcji również udało mi się skończyć poza przypadkiem bazowym i przypadkiem formuły rozpoczętej kwantyfikatorem uniwersalnym.

Przypadek bazowy okazał się zbyt trudny technicznie, sprowadza się właściwie do wykazania że nadałem interpretację symbolom taką jaką nadałem, ale wymaga to przejścia przez ogólną funkcję interpretującą aplikację.

Przypadek kwantyfikatora ogólnego pominąłem ze względów czasowych, ale jego założenie nie może prowadzić do sprzeczności gdyż skonstruowałem dowód dla kwantyfikatora egzystencjalnego i negacji.

Z ciekawych problemów związanych z formalizacją tego twierdzenia jeszcze jest różnica między sygnaturami języka wejściowego i języka rozszerzonego w trakcie dowodu, mianowicie na końcu otrzymujemy model nad rozszerzonym językiem, a twierdzenie postuluje istnienie modelu dla wejściowego języka. Jest to część dowodu zwykle pomijana, ze względu na intuicyjną oczywistość, w znanych mi wariantach papierowych.

Twierdzenie o modelu dla pod-języka jest wykonane w całości poza przypadkiem bazowym, gdzie trudne okazały się technikalia związane z typami zależnymi.

4 Fragmenty założone z góry oraz ograniczenia.

W kilkunastu miejscach pojawia się taktyka `admit`, znacznej większości twierdzeń gdzie się pojawiła jestem pewien że próba ich udowodnienia nie wykryłaby jakiejś usterki w mojej formalizacji, tzn jestem pewien że założenie tych twierdzeń nie prowadzi do sprzeczności. W większości ta taktyka dotyczy fragmentów dowódów, w których nie poradziłem sobie z typami zależnymi. Jedynym miejscem, w którym spodziewam się problemów to pierwsza część twierdzenia o istnieniu modelu. Z twierdzenia o rozszerzaniu do teorii zupełnej jestem pewien że sama konstrukcja teoriomnogościowa z etapu rozszerzania jest wykonalna. Obawiam się natomiast, że różne wymagane operacje na języku czy twierdzenia jego dotyczące mogłyby wymagac ulepszenia niektórych definicji w moim projekcie, a nawet mogłyby wykryć jakieś niedopatrzenie.

Trudno mi ocenić, czy nie popełniłem nigdzie błędu i który nie został wykryty dzięki taktyce `admit`, w szczególności, że projekt rozrósł się do ponad ośmiu tysięcy wierszy kodu.

Twierdzenie o istnieniu modelu, i co za tym idzie twierdzenie o pełności, są ograniczone do języków o przeliczalnym zbiorze wszystkich formuł. To ograniczenie nie pozwaliło mi na pracę nad dolnym twierdzeniem Lowenheima-Skolema, na którym mi bardzo zależało bo zostawia duże pole do nadinterpretacji. Sam dowód „papierowy" tego twierdzenia jest bardzo prosty i opiera się drugą wersję twierdzenia o zwartości, gdzie właściwość bycia niesprzecznym jest zastąpiona właściwością bycia spełnialnym. Tę wersję twierdzenia można dowieść za pomocą starego twierdzenia o zwartości oraz twierdzenia o istnieniu modelu.

Twierdzenie 6 (O zwartości (wersja modelowa))**.** *Zbiór formuł jest spełnialny wtedy i tylko wtedy gdy każdy jego skończony podzbiór jest spełnialny.*

Twierdzenie 7 (Lowenheim-Skolem - luźne sformułowanie)**.** *Jeżeli teoria T ma model mocy nieskończonej to również ma model każdej mocy nieskończonej.*

W dowodzie powyższego twierdzenia należy dodać zbiór nowych stałych do języka, o takiej mocy jakiej chcemy osiągnąć model. Definicja sygnatury języka w moim projekcie wymaga aby liczba symboli była przeliczalna.

Nie umiem zaplanować prac nad zniesieniem przytoczonego ograniczenia, nie potrafię tego zrobić również dla wersji papierowych.

W projekcie planowałem również wykonać pracę nad twierdzeniem Herbranda, ale nie starczyło już na to czasu. Prawdopodobnie również napotkałbym wiele trudności w dowodzie tego twierdzenia, gdyż zdaje się, że jest technicznie bardziej skomplikowane od twierdzenia o istnieniu modelu.

Literatura

[1] A. Rutkowski, *Elementy logiki matematycznej*, Biblioteczka Matematyczna, tom 35, WSiP, Warszawa 1978

[2] A. Grzegorczyk, *Zarys logiki matematycznej*, Biblioteka Matematyczna, tom 20, PWN, Warszawa 1973

[3] C. Ó Dúnlaing, *Completeness of some axioms of Lukasiewicz's: an exercise in problem-solving*, Trinity College, Dublin 2, Ireland, June 30, 1997
http://www.maths.tcd.ie/report_series/tcdmath/tcdm9705.ps