Proof checker

Celem projektu jest napisanie programu, który sprawdza poprawność dowodów formuł logicznych w systemie dedukcji naturalnej znanej z Logiki dla informatyków.

Dane wejściowe

Program wczytuje dane z pliku tekstowego. Plik wejściowy zawiera ciąg dowodów, z których każdy jest zapisany w następującym formacie:

```
goal modusPonens: A /\ (A => B) => B
proof
  [ A /\ (A => B) :
    A;
    A => B;
    B ];
  A /\ (A => B) => B
end.
```

Każda formuła rozpoczyna się od słowa kluczowego `goal` i ma swój identyfikator (w przykładzie `modusPonens`). Zapis dowodu jest zamknięty między słowami kluczowymi `proof` i `end`. Język zapisu dowodu przedstawiony jest w dalszej części specyfikacji.

Użyj programów `ocamllex` i `menhir` do wygenerowania leksera i parsera. Specyfikację parsera należy umieścić w pliku o rozszerzeniu .mly, a specyfikację leksera w pliku o rozszerzeniu .mll (wystarczy rozbudować dostarczone pliki lexer.mll i parser.mly).

Wynik działania programu

Program sprawdza poprawność każdego dowodu i wypisuje odpowiednią informację do pliku wyjściowego. W przypadku dowodów niepoprawnych program powinien zwrócić informację o lokalizacji/rodzaju błędu.

Nazwy pliku wejściowego oraz wyjściowego są zadane jako argumenty wywołania programu.

$$\frac{\texttt{A true} \quad \texttt{B true}}{\texttt{A} \land \texttt{B true}}(\text{andI}) \qquad \frac{\boxed{\begin{array}{c}\texttt{A}\\ \vdots \\ \texttt{B true}\end{array}}}{\texttt{A} \Rightarrow \texttt{B true}}(\text{impI}) \qquad \frac{\texttt{A true}}{\texttt{A} \lor \texttt{B true}}(\text{orI}_\text{l}) \qquad \frac{\texttt{B true}}{\texttt{A} \lor \texttt{B true}}(\text{orI}_\text{r})$$

$$\frac{}{\texttt{T true}}(\text{trueI}) \qquad \frac{\texttt{F true}}{\texttt{A true}}(\text{falseE})$$

$$\frac{\texttt{A} \land \texttt{B true}}{\texttt{A true}}(\text{andE}_\text{l}) \qquad \frac{\texttt{A} \land \texttt{B true}}{\texttt{B true}}(\text{andE}_\text{r}) \qquad \frac{\texttt{A true} \quad \texttt{A} \Rightarrow \texttt{B true}}{\texttt{B true}}(\text{impE})$$

$$\frac{\texttt{A} \lor \texttt{B true} \quad \boxed{\begin{array}{c}\texttt{A}\\ \vdots \\ \texttt{C true}\end{array}} \quad \boxed{\begin{array}{c}\texttt{B}\\ \vdots \\ \texttt{C true}\end{array}}}{\texttt{C true}}(\text{orE})$$

Rysunek 1: Reguły ND

Dedukcja naturalna

Rozważmy formuły rachunku zdań zbudowane ze zmiennych zdaniowych, stałych logicznych `T` i `F` oraz spójników koniunkcji `/\`, alternatywy `\/`, negacji $\sim$, implikacji `=>` i równoważności `<=>`.

Dowód w systemie dedukcji naturalnej można przedstawić jako drzewo, w którego korzeniu znajduje się dana formuła, a jego kolejne poziomy tworzone są według reguł wnioskowania przedstawionych na rys. 1, tzn. jeśli spełnione są przesłanki umieszczone nad kreską, to wówczas możemy wywieść prawdziwość formuły pod kreską. Fragmenty umieszczone w ramkach oznaczają wnioskowanie hipotetyczne:

$$\boxed{\begin{array}{c}\texttt{A}\\ \vdots \\ \texttt{B true}\end{array}}$$

oznacza, że zakładamy prawdziwość `A` i przy tym założeniu dowodzimy prawdziwości `B`. Przesłanka `A` nie jest dostępna na zewnątrz ramki.

Zauważ, że reguły na rys. 1 są podzielone na reguły wprowadzenia (andI, impI, orI, trueI) oraz eliminacji spójników (andE, impE, orE, falseE). Brakuje reguł dla spójników `<=>` i $\sim$ – te reguły należy napisać samodzielnie korzystając z faktu, że te spójniki da się wyrazić za pomocą pozostałych.

Uwaga: opisany system jest wystarczający dla dowodzenia w logice intucjonistycznej, ale można go rozszerzyć tak, by móc dowodzić formuł prawdziwych w logice klasycznej (np. $\texttt{A} \lor \sim\texttt{A}$).

Język zapisu dowodów

Zamiast zapisywać dowody w postaci drzew będziemy używać wygodniejszej (ale mniej precyzyjnej) notacji liniowej. Taka reprezentacja dowodu składa się z elementów rozdzielonych średnikiem. Każdy element dowodu jest albo pojedynczą formułą albo ramką. Wystąpienie pojedynczej formuły w dowodzie jest poprawne, jeśli jest konkluzją jednej z reguł wnioskowania, której wszystkie przesłanki są spełnione. Jeśli program napotka formułę, której nie da się wywieść w ten sposób, uznajemy dowód za niepoprawny.

Przykładowo dowód

```
proof
  A;
  A
end.
```

jest niepoprawny, ponieważ w pierwszej linii występuje formuła A, której prawdziwości nie umiemy uzasadnić bez dodatkowych przesłanek.

Drugi rodzaj elementu dowodu – ramkę – zapisujemy w nawiasach kwadratowych; składa się ona z pojedynczej formuły, którą traktujemy jako przesłankę, oraz dowodu, w którym można użyć tej przesłanki. Przesłanka nie jest dostępna poza ramką, w której została wprowadzona, ale ramki można zagnieżdżać.
Przykładowo dowód implikacji `A => A` można zapisać tak:

```
proof
  [ A : A ];
  A => A
end.
```

gdzie fragment `[ A :` wprowadza przesłankę `A`, z której możemy korzystać wewnątrz ramki - zatem drugie wystąpienie formuły `A` jest poprawne, ponieważ dysponujemy już przesłanką. Zauważmy, że cały dowód zawiera jeszcze wystąpienie formuły `A => A` i te dwa elementy odpowiadają zastosowaniu reguły wnioskowania impI.

Z kolei dowód formuły `A => B => A` można zapisać tak:

```
proof
  [ A : [ B : A];
         B => A];
    A => B => A
end.
```

Pierwsza ramka wprowadza przesłankę `A`, z której korzystamy wewnątrz drugiej ramki, która z kolei jest potrzebna do wprowadzenia drugiej przesłanki `B` (z której jednak nie korzystamy w tym dowodzie). Zauważ też, że dla formuły `A => A => A` możemy napisać dowód

```
proof
  [ A :
    [ A : A ];
      A => A ];
  A => A => A
end.
```

który jest poprawny, ale niejednoznaczny, bo nie wiadomo, z którego wprowadzenia przesłanki `A` korzystamy wewnątrz drugiej ramki.

Jeśli mamy dostępną przesłankę `A \/ B` i chcemy wywieść z niej `C`, dowód powinien zawierać fragment z dwoma ramkami wprowadzającymi odpowiednio `A` i `B` jako przesłanki jak w poniższym szkicu dowodu (co odpowiada zastosowaniu reguły orE):

```
proof
 ...
 A \/ B;
 [ A : ... C ];
 [ B : ... C ];
 C
end.
```

Warianty zadania

Po zaimplementowaniu wersji podstawowej rozważ możliwe rozszerzenia programu, takie jak:

1. Możliwość korzystania z aksjomatów oraz wcześniej udowodnionych formuł w dowodach występujących później w pliku. W tym celu należy m. in. wprowadzić nową notację dla aksjomatów i rozszerzyć język zapisu dowodów. Aby uzyskać logikę klasyczną można korzystać z dodatkowego aksjomatu albo dodać nową regułę wnioskowania.

2. Obsługa formuł logiki I rzędu.

3. Gdy już umiemy weryfikować dowody w logice I rzędu, można pokusić się o możliwość weryfikacji dowodów własności liczb naturalnych czy list.

4. Możemy sobie wyobrazić sytuację, w której użytkownik chciałby pominąć pewien fragment dowodu. Wtedy chcielibyśmy, żeby program umiał automatycznie wypełniać takie dziury (przynajmniej w niektórych sytuacjach) i zwracać użytkownikowi informację, czy i jak mu się to udało zrobić.

5. Opisany w tej specyfikacji format dowodu nie pozwala na jednoznaczne odtworzenie odpowiadającego mu drzewa wyprowadzenia wg reguł z rys. 1. Zaproponuj modyfikację programu, która umożliwiałaby generowanie drzewa dowodu na podstawie zadanego zapisu dowodu. Do zakodowania drzewa użyj termów rachunku lambda (rys. 2).

$$\frac{\mathtt{M} : \mathtt{A} \quad \mathtt{N} : \mathtt{B}}{(\mathtt{M}, \mathtt{N}) : \mathtt{A} \wedge \mathtt{B}}(\text{andI}) \qquad \frac{\boxed{\begin{array}{c} \mathtt{x} : \mathtt{A} \\ \vdots \\ \mathtt{M} : \mathtt{B} \end{array}}}{\lambda \mathtt{x}.\mathtt{M} : \mathtt{A} \Rightarrow \mathtt{B}}(\text{impI}) \qquad \frac{\mathtt{M} : \mathtt{A}}{\mathtt{inl}(\mathtt{M}) : \mathtt{A} \vee \mathtt{B}}(\text{orI}_{\text{l}}) \qquad \frac{\mathtt{N} : \mathtt{B}}{\mathtt{inr}(\mathtt{M}) : \mathtt{A} \vee \mathtt{B}}(\text{orI}_{\text{r}})$$

$$\frac{}{() : \top}(\text{trueI}) \qquad \frac{\mathtt{M} : \bot}{\mathtt{efq}(\mathtt{M}) : \mathtt{A}}(\text{falseE})$$

$$\frac{\mathtt{M} : \mathtt{A} \wedge \mathtt{B}}{\mathtt{fst}(\mathtt{M}) : \mathtt{A}}(\text{andE}_{\text{l}}) \qquad \frac{\mathtt{M} : \mathtt{A} \wedge \mathtt{B}}{\mathtt{snd}(\mathtt{M}) : \mathtt{B}}(\text{andE}_{\text{r}}) \qquad \frac{\mathtt{N} : \mathtt{A} \quad \mathtt{M} : \mathtt{A} \Rightarrow \mathtt{B}}{\mathtt{M}\,\mathtt{N} : \mathtt{B}}(\text{impE})$$

$$\frac{\mathtt{M} : \mathtt{A} \vee \mathtt{B} \quad \boxed{\begin{array}{c} \mathtt{x} : \mathtt{A} \\ \vdots \\ \mathtt{N} : \mathtt{C} \end{array}} \quad \boxed{\begin{array}{c} \mathtt{y} : \mathtt{B} \\ \vdots \\ \mathtt{P} : \mathtt{C} \end{array}}}{\mathtt{case}\ \mathtt{M}\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}).\mathtt{N} \mid \mathtt{inr}(\mathtt{y}).\mathtt{P} : \mathtt{C}}(\text{orE})$$

Rysunek 2: Kodowanie dowodów ND jako termów

Literatura

1. A. Madhavapeddy, J. Hickey, Y. Minsky, “Real World OCaml”.
2. M. Huth, M. Ryan, “Logic in Computer Science” – wyczerpująco o dedukcji naturalnej.