Laborator - Corespondenta Curry-Howard

Scop: implementarea sistemului de deductie naturala a logicii propozitionale, in Haskell.

Caracterizam un sistem logic prin doua elemente:

- sintaxa: demonstratia, teorema...
- semantica: adevar, validitate, satsfiabilitate...

In cazul logicii propozitionale, cele doua notiuni corespund. Daca φ este adevarat (semantic), atunci φ este demonstrabil (este teorema) - rezultatul este dat de teorema de completitudine.

Notatii:

- sintaxa: $\vdash \varphi$ (este o teorema a LP). $\Gamma \vdash \varphi$ (este o teorema in ipotezele de deductie din Γ - φ este o Γ-teorema)
- semantica: $\models \varphi$ (este o tautologie). $\Gamma \models \varphi$ (φ este o Γ-tautologie).

Semantica - tabele de adevar, este costisitoare din punct de vedere computational;
Sintaxa - sistemul de deductie Hilbert.

$(A_1)\varphi \to (\psi \to \varphi)$
$(A_2)(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi))$
$(A_3)(\neg\psi \to \neg\varphi) \to (\varphi \to \psi)$

Regula de deductie in logica propozitionala este *modus ponens*:

$$\frac{\varphi \quad \varphi \to \psi}{\psi}(MP)$$

In continuare, in loc sa lucram in sistemul Hilbert, vom lucra in sistemul de deductie naturala pentru logica propozitionala:

- este un sistem deductiv, construit doar din reguli de deductie pentru toti conectorii logici (negatia, implicatia, disjunctia, conjunctia si echivalenta);
- este un sistem echivalent cu sistemul Hilbert, dar este mai expresiv.

Sistemul de deductie naturala pentru logica propozitionala

- este un sistem expresiv, pentru care avem reguli de deductie pentru toti conectorii logici;
- regulile de deductie sunt impartite intre reguli de introducere si reguli de eliminare.

1. Conjunctia

Regula de introducere pentru conjunctie ne spune ca daca avem doua teoreme φ si ψ, atunci avem o demonstratie pentru teorema $\varphi \wedge \psi$.

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi}(\wedge_i)$$

Regulile de eliminare sunt proiectiile pe fiecare dintre cei doi conjuncti.

$$\frac{\varphi \wedge \psi}{\varphi}(\wedge e_l) \qquad \frac{\varphi \wedge \psi}{\psi}(\wedge e_r)$$

2. Implicatia

Regula de introducere este regula naturala. De exemplu, pentru a putea demonstra ca $\varphi \to \psi$, tehnica obisnuita este:

- presupunem ca φ este demonstrabil;
- aratam, in aceasta ipoteza, ca ψ este demonstrabil.

$$\frac{\varphi \to \ldots \to \ldots \to \psi}{\varphi \to \psi}(\to i)$$

Regula de eliminare este *modus ponens*:

$$\frac{\varphi \quad \varphi \to \psi}{\psi}(\to e)(\equiv MP)$$

Disjunctia

Regulile de introducere respecta caracterul semantic, adica un adevar in disjunctie cu orice ramane tot un adevar. ($1 \vee 0 = 1, 1 \vee 1 = 1$).

$$\frac{\varphi}{\varphi \vee \psi}(\vee i_l) \quad \frac{\psi}{\varphi \vee \psi}(\vee i_r)$$

In cazul regulii de eliminare, daca stim ca $\varphi \vee \psi$ este demonstrabil, nu stim sigur care dintre φ sau ψ este demonstrabil. Eliminarea se face pe cazuri:

- presupunem ca φ este demonstrabil, si aratam ca din φ se deduce χ;
- presupunem ca ψ este demonstrabil, si aratam ca din ψ se deduce χ.

In acest caz, indiferent de care dintre cele doua era demonstrabil, ajungem la acelasi χ.

$$\frac{\varphi \vee \psi \quad (\varphi \to \chi) \quad (\psi \to \chi)}{\chi}(\vee e)$$

4. Negatia

Notam falsul cu $\bot$, respectiv adevarul cu $\top$.

$$\frac{\bot}{\varphi}(\bot e) \quad \frac{\varphi \wedge \neg\varphi}{\bot}(\neg i)$$

Natural, se poate construi principiul RAA (reducerea la absurd):

$$\frac{(\varphi \to ... \to \bot)}{\neg\varphi}(RAA)$$

5. Echivalenta

Regulile se deduc automat, din faptul ca echivalenta se construieste din implicatii si o conjunctie.

$$\varphi \iff \psi \equiv (\varphi \to \psi) \wedge (\psi \to \varphi)$$

$$\frac{(\varphi \to ... \to \psi) \quad (\psi \to ... \to \varphi)}{\varphi \iff \psi}(\iff i)$$

$$\frac{\varphi \iff \psi \quad \varphi}{\psi}(\iff e_l) \quad \frac{\varphi \iff \psi \quad \psi}{\varphi}(\iff e_r)$$

Exemple de demonstratii in deductia naturala

Demonstrati ca urmatorii **secventi** sunt valizi.

1. $p \wedge q, r \wedge s \vdash p \wedge s$

Trebuie sa demonstram $p \wedge s$, pornind de la ipotezele de deductie $p \wedge q$, respectiv $r \wedge s$.

Echivalent, se mai putea scrie:
$\vdash (p \wedge q) \rightarrow (r \wedge s) \rightarrow (p \wedge s)$

In logica propozitionala exista **teorema deductiei**, care spune ca
$\vdash \varphi \rightarrow \psi$ e acelasi lucru cu $\{\varphi\} \vdash \psi$

In implementare, folosim a doua forma, cea cu implicatii

```
exercise :: And p q -> And r s -> And p s
exercise = undefined
```

Demonstratia este una pe linii, ca in sistemul Hilbert. Mereu, pe primele linii, se scriu ipotezele de deductie. Structura unei linii este (numar) formula (explicatie cum am ajuns la aceasta formula).

Demonstratie:
(1) $p \wedge q$ (ipoteza)
(2) $r \wedge s$ (ipoteza)
(3) p ($\wedge e_l$) (1)
(4) s ($\wedge e_r$) (2)
(5) $p \wedge s$ ($\wedge i$) (3) (4)

Am demonstrat ca secventul este valid.

2. $p \wedge (q \vee s) \vdash (p \wedge q) \vee (p \wedge s)$

Demonstratie:
(1) $p \wedge (q \vee s)$ (ipoteza)
(2) p ($\wedge e_l$) (1)
(3) $q \vee s$ ($\wedge e_r$) (1)

(4) | q (asumptie = ipoteza temporara)
(5) | $p \wedge q$ ($\wedge i$) (2) (4)
(6) | $(p \wedge q) \vee (p \wedge s)$ ($\vee i_l$) (5)

(7) | s (asumptie)
(8) | $p \wedge s$ ($\wedge i$) (2) (7)
(9) | $(p \wedge q) \vee (p \wedge s)$ ($\vee i_r$) (8)

(10) $(p \wedge q) \vee (p \wedge s)$ $(\vee e)$ (3) (4-6) (7-9)

Am demonstrat ca secventul este valid.

3. $\neg p \vee \neg q \vdash \neg(p \wedge q)$

Demonstratie:
(1) $\neg p \vee \neg q$ (ipoteza)
(2) | $p \wedge q$ (asumptie)
(3) | p ($\wedge e_l$) (2)
(4) | q ($\wedge e_r$) (2)
(5) | | $\neg p$ (asumptie)
(6) | | $\bot$ ($\neg i$) (3) (5)
---
(7) | | $\neg q$ (asumptie)
(8) | | $\bot$ ($\neg i$) (4) (7)
(9) | $\bot$ ($\vee e$) (1) (5-6) (7-8)
(10) $\neg(p \wedge q)$ (RAA) (2-9)

Am demonstrat ca secventul este unul valid.

Implementarea corespondentei Curry-Howard in Haskell

Avem urmatoarele echivalente:

- demonstratiile vor fi chiar programele;
- implicatia e chiar functia din Haskell; $\varphi \to \psi$ inseamna ca avem o functie de la φ la ψ;
- pentru restul tipurilor:
    - False ($\bot$) - tipul void;
    - True ($\top$) - tipul unit;
    - And ($\wedge$) - tip produs;
    - Or ($\vee$) - tip suma;

- Not ($\neg$) - va fi definita prin implicatie si False: $\neg\varphi \equiv \varphi \to \bot$
- Iff ($\iff$) - se va construi din And si implicatie.

```
data False                                  -- empty type, void

data True = True                    -- unit type

data And a b = And { proj1 :: a, proj2 :: b }

data Or a b = Left a
        | Right b

type Not a = a -> False

type Iff a b = And (a -> b) (b -> a)
```

Specificam regulile de introducere si de eliminare pentru aceste tipuri.

```
trueIntro :: True
trueIntro = True

falseElim :: False -> b
falseElim x = case x of

implElim :: (a -> b) -> a -> b
implElim f = f

implIntro :: (a -> b) -> a -> b
implIntro f = f

andIntro :: a -> b -> And a b
andIntro = And

andElimL :: And a b -> a
andElimL = proj1

andElimR :: And a b -> b
andElimR = proj2

orIntroL :: a -> Or a b
orIntroL = Left
```

```haskell
orIntroR :: b -> Or a b
orIntroR = Right

orElim :: (a -> c) -> (b -> c) -> Or a b -> c
orElim fac fbc or = case or of
        Left a -> fac c
        Right b -> fbc b

notIntro :: (forall p. a -> p) -> Not a
notIntro f = f

notElim :: p -> Not p -> c
notElim p np = falseElim (np p)

iffIntro :: (a -> b) -> (b -> a) -> Iff a b
iffIntro fab fba = andIntro fab fba

iffElimL :: a -> Iff a b -> b
iffElimL a iff = (andElimL iff) a

iffElimR :: b -> Iff a b -> a
iffElimR b iff = (andElimR iff) b

deMorgan1 :: And (Not p) (Not q) -> Not (Or p q)
deMorgan1 = undefined

deMorgan2 :: Not (Or p q) -> And (Not p) (Not q)
deMorgan2 = undefined

deMorgan3 :: Or (Not p) (Not q) -> Not (And p q)
deMorgan3 = undefined
```

In continuare, in cadrul acestui laborator definim un sistem de axiome pentru un fragment de logica intuitionista a logicii propozitionale. Logica intuitionista este logica in care nu tin principiul tertului exclus (nu stim $\varphi \vee \neg\varphi$ si nu tine nici eliminarea dublei negatii, adica nu avem o echivalenta intre φ si $\neg\neg\varphi$).

Avem 10 axiome si vrem sa implementam demonstratii pentru aceste axiome in sistemul de deductie naturala definit.

```
ax1 :: a -> b -> a
ax1 = implIntro (\a -> implIntro (\b -> a))

ax2 :: (a -> b) -> (a -> (b -> c)) -> a -> c
ax2 = implIntro (\f ->                          -- f = a -> b
                implIntro (\g ->                        -- g = a -> (b -> c)
                        implIntro (\a ->                -- a = a
                                implElim (implElim g a) (implElim f a))))
                                                -- b -> c       b
                                                                -- c
```

Lucram a doua axioma.

$$\vdash (a \to b) \to (a \to (b \to c)) \to a \to c$$

Demonstratie:
(1) $a \to b$ (ipoteza)
(2) $a \to (b \to c)$ (ipoteza)
(3) a (ipoteza)
(4) b ($\to e$) (1) (3)
(5) $b \to c$ ($\to e$) (2) (3)
(6) c ($\to e$) (4) (5)

```
ax3 :: a -> b -> And a b
ax3 = implIntro (\a -> implIntro (\b -> andIntro a b))

ax4 :: And a b -> a
ax4 = implIntro (\ab -> andElimL ab)
```

Tema:

- de terminat de definit toate axiomele din acest fragment - Lab07 din https://tinyurl.com/flp2023-materials;
- de demonstrat ca, daca in logica tine principiul tertului exclus, atunci este demonstrabila eliminarea dublei negatii:

$$(p \vee \neg p) \vdash \neg\neg p \to p$$

```
excludedMiddleImplDoubleNeg :: Or a (Not a) -> (Not (Not a) -> a)
```

```
excludedMiddleImplDoubleNeg = undefined
```

- de demonstrat regulile lui de Morgan (1, 2, 3).

Trimiteti tema pe mail la bogdan.macovei.fmi@gmail.com.
Pentru restul temelor, daca nu le-ati trimis, le mai puteti trimite pana la finalul saptamanii viitoare.