Introducere în semantica denotațională

— raport de cercetare, anul 1, semestrul II —

Adrian Manea
coordonator: Conf. dr. Denisa Diaconescu

Cuprins

1 Introducere

Lucrarea de față continuă studiul corespondenței Curry–Howard–Lambek, expusă pe scurt într-o prezentare de ansamblu în semestrul anterior. De data aceasta, ne concentrăm asupra noțiunilor corespunzătoare *semanticii denotaționale a limbajelor de programare* și introducem cîteva dintre uneltele matematice utile pentru a studia *teoria domeniilor*. Vom insista pe mulțimile parțial ordonate complete și stricte, cărora le vom justifica utilitatea în specificații recursive în topologia Scott (cf. Scott and Strachey (1971)) și pe care le vom lega și de categorii cartezian închise.

În particular, prezentarea de față pune accentul pe partea computațională a corespondenței Curry–Howard–Lambek, prin semantica denotațională, dar totodată evidențiază și formalismul teoriei categoriilor pentru aceste scopuri, formalism care, conform Lambek (1989), reprezintă un mediu excelent pentru interpretarea sistemelor logice deductive.

2 Semantica punctelor fixe

2.1 Motivație

Conform Schmidt (2009), *semantica denotațională* își propune să pornească de la specificații sintactice ale limbajelor, în varianta simplă folosindu-se forma Backus-Naur (BNF) și să asocieze operații (în general, funcții), care să arate înțelesul expresiilor din sintaxă. Totodată, este de dorit uneori ca semantica denotațională să țină cont și de aspecte *operaționale*, adică cele privitoare la implementarea limbajului și a compilatoarelor.

Deoarece accentul cade pe funcții, dar și ținîndu-se cont de aspectele operaționale, semantica denotațională are un *caracter compozițional*. În acest sens, devine extrem de importantă specificarea matematică a așa-numitelor *algebre semantice*, care constau în domeniile de definiție și de valori ale funcțiilor implementate și funcțiile propriu-zise (de arități diferite, deci în particular includem și operații binare, unare și funcții de mai multe variabile).

Dana Scott (Scott and Strachey (1971)) a arătat, într-o serie de articole publicate în anii '70-'80 că, pentru majoritatea necesităților „practice", i.e. a limbajelor deja existente, implementate sau teoretice, sînt necesare constrîngeri speciale asupra naturii mulțimilor ce devin domenii semantice, dar și asupra funcțiilor dintre ele. Astfel, dacă structuri, instrucțiuni și operații aritmetice simple se pot defini fără constrîngeri speciale asupra mulțimilor și funcțiilor, în afara proprietăților de tip morfism (i.e. compunerea funcțiilor să fie compatibilă cu operațiile), recursivitatea omniprezentă este cea care ridică probleme.

În particular, *combinatorii din λ-calcul* de forma $\lambda x.xx$ sînt cei care ridică probleme, prin auto-aplicare.

În continuare, vom prelua din expunerile Abramsky and Jung (1994) și Schmidt (2009) pentru a justifica și exemplifica semantica funcțiilor recursive și a domeniilor necesare pentru a le susține.

2.2 Ecuația de punct fix

Presupunem că avem o definiție recursivă, de forma:

$$X = \ldots X \ldots,$$

căreia vrem să-i asociem un sens (semantică), i.e. o metodă de interpretare, care să fie compatibilă cu restul limbajului din care ar face parte această expresie. În particular, vom fi interesați de proprietățile compoziționale, deci semantica va trebui să fie dată funcțional.

Presupunem că ecuația este formulată într-un domeniu D, deci căutăm un element $d \in D$ astfel încît, făcînd o substituție convenabilă în X cu d, să obținem o expresie cu sens. Membrul drept al unei expresii recursive are o formă funcțională, de tipul $f(d)$,

iar membrul stîng trebuie să fie un rezultat, pentru ca expresia să returneze ceva, deci sîntem conduși la o *ecuație de punct fix*, de forma generală:

$$f(d) = d, \quad d \in D,$$

unde urmează să definim funcția f și proprietățile domeniului D.

În plus, pentru generalitatea teoriei, avem nevoie de o metodă canonică sau măcar generală de a găsi puncte fixe pentru funcții (cu anumite proprietăți), definite între domenii de un anumit fel.

Ajungem astfel la ecuația fundamentală rezolvată de Scott în semantica denotațională, anume:

$$D \simeq [D \to D].$$

Această ecuație afirmă că sîntem în căutarea unei corespondențe biunivoce între elementele „spațiului de funcții" ($[D \to D]$) și elementele lui D, care pot fi rezultate ale definițiilor recursive pentru aceste funcții.

Ecuația a fost formulată și rezolvată de Scott în cazul unor domenii înzestrate cu o topologie discretă, numită *topologia Scott*, iar funcțiile din $[D \to D]$ au o proprietate de continuitate în raport cu această topologie.

Discuția detaliată asupra topologiei Scott depășește cadrul acestei expuneri, dar vom folosi cele de mai sus ca motivație pentru studiul unor proprietăți ale domeniilor semantice, dintre care cea mai importantă este *ordonarea parțială*. De fapt, structura de poset a domeniului D este punctul de pornire pentru definiția generală a topologiei Scott, deci putem considera cele ce urmează drept o introducere elementară în acest subiect.

Observație 2.1: Deoarece tema aleasă urmărește și legătura cu teoria categoriilor, facem observația că ecuația domeniilor Scott,

$$D \simeq [D \to D]$$

poate fi formulată și în context categorial. Fie $\mathcal{C}$ o categorie. Spațiul de funcții $[D \to D]$ poate fi gîndit ca un (bi)functor $F : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$.

Atunci, ecuația Scott ajunge la găsirea acelor categorii $\mathcal{C}$ și obiecte $D \in \mathcal{C}$ astfel încît $D \simeq F(D, D)$.

Vom detalia acest punct de vedere categorial într-o oarecare măsură în §3.

2.3 Exemplu: Semantica factorialului

Necesitatea folosirii domeniilor înzestrate cu o ordine parțială poate fi văzută printr-un exemplu, preluat din Schmidt (2009), §6.

Fie `Nat` domeniul numerelor naturale, $\texttt{Nat}_\perp = \texttt{Nat} \cup \{\perp\}$ și funcția definită prin:

```
fac n = λn.n equals zero → one
          □ n times (fac n minus one)
```

Am folosit sintaxa preluată (și adaptată) din Schmidt (2009), iar notațiile în litere pentru constante și funcții elementare arată faptul că ele pot fi definite separat, în așa-numitele domenii semantice primitive, cum este cazul `Nat`. Detalii se găsesc, de exemplu, în Schmidt (2009), §3. În continuare, însă, vom face abuz de notație și vom folosi simbolurile obișnuite.

Evaluînd funcția `fac` pe argumente concrete, observăm că avem 3 cazuri posibile:

- nicio explicitare (eng. *unfolding*) — caz imposibil, pe care îl notăm cu $\texttt{fac}_0$ și care are graficul $\emptyset$ (formal, este $\Gamma\texttt{fac}_0 = \{(\emptyset, \emptyset)\}$);
- o explicitare, pentru $n = 0$, caz pe care îl notăm cu $\texttt{fac}_1$, care are graficul $\Gamma\texttt{fac}_1 = \{(0, 1)\}$;
- $i + 1$ explicitări, pentru i apeluri recursive, caz pe care îl notăm cu $\texttt{fac}_i$ și care, pentru i fixat, are graficul $\Gamma\texttt{fac}_i = \{(0, 1), (1, 1), \ldots, (i, i!)\}$.

Aceste grafice sînt ordonate parțial cu incluziunea (admițînd că $\{(\emptyset, \emptyset)\}$ este submulțime pentru orice mulțime de perechi. În mod evident, ele modelează cazuri particulare ale funcției `fac`, deci:

$$\Gamma\texttt{fac} \supseteq \bigcup_{i \geq 0} \Gamma\texttt{fac}_i.$$

Dar are loc și incluziunea reciprocă: dacă o pereche $(a, b) \in \Gamma\texttt{fac}$, ea s-a obținut printr-un anumit număr de explicitări, deci există i cu $(a, b) \in \Gamma\texttt{fac}_i$. Rezultă că avem egalitate:

$$\Gamma\texttt{fac} = \bigcup_{i \geq 0} \Gamma\texttt{fac}_i.$$

Aceste rezultat stă la baza semanticii pentru puncte fixe, care afirmă exact că înțelesul total al unei funcții recursive este dat de cazurile sale particulare. De aceea, putem extrage și o reprezentare nerecursivă pentru fiecare $\texttt{fac}_i$:

$$\begin{aligned}
\texttt{fac}_i &: \texttt{Nat} \to \texttt{Nat}_\perp, \quad i \geq 0 \\
\texttt{fac}_0 &= \lambda \texttt{n}.\perp \\
\texttt{fac}_{i+1} &= \lambda \texttt{n.n} \texttt{ = } 0 \to 1 \,\square\, \texttt{n} \cdot \texttt{fac}_i(n-1), \quad i \geq 1,
\end{aligned}$$

ceea ce ne conduce chiar la funcționala:

$$\begin{aligned}
&\texttt{F} : (\texttt{Nat} \to \texttt{Nat}_\perp) \to (\texttt{Nat} \to \texttt{Nat}_\perp) \\
&\texttt{F} = \lambda \texttt{f} \lambda \texttt{n.n} \texttt{ = } 0 \to 1 \,\square\, \texttt{n} \cdot \texttt{f(n-1)},
\end{aligned}$$

cu $\texttt{fac}_{i+1} = \texttt{F fac}_i$.

Rezultă:

$$\Gamma\texttt{fac} = \bigcup_{i \geq 0} \Gamma\texttt{F}^i\emptyset,$$

unde $\emptyset = \lambda \mathtt{n}.\,\bot$ și mai mult,

$$\Gamma \mathtt{F\ fac} = \Gamma \mathtt{fac},$$

deci `F fac` = `fac`, de unde rezultă că `fac` este un punct fix pentru `F`.

Dar, dacă pornim cu funcția:

$$\begin{aligned} &\mathtt{q} : \mathtt{Nat} \to \mathtt{Nat}_\bot \\ &\mathtt{q} = \lambda \mathtt{n.n{=}0} \to 1 \,\square\, \mathtt{q(n{+}1)}, \end{aligned}$$

se poate arăta (Schmidt (2009), pp. 97-98) că ecuația `Qq = q` are o infinitate de soluții, unde `Q` este funcționala corespunzătoare:

$$\begin{aligned} &\mathtt{Q} : (\mathtt{Nat} \to \mathtt{Nat}_\bot) \to (\mathtt{Nat} \to \mathtt{Nat}_\bot) \\ &\mathtt{Q} = \lambda \mathtt{g} \lambda \mathtt{n.n{=}0} \to 1 \,\square\, \mathtt{g(n{+}1)}. \end{aligned}$$

Avem nevoie, așadar, de *cel mai mic punct fix*, un motiv suplimentar de a lucra cu structuri de poset, atît pe D, cît și pe $[D \to D]$. Mai facem observația că pe $[D \to D]$ vom lucra cu *definiții extensionale*, adică:

$$f \geq g \Leftrightarrow \Gamma f \subseteq \Gamma g.$$

2.4 Domenii parțial ordonate

Fără a intra în detalii, rezultatele esențiale care arată utilitatea poset-urilor pentru semantica definițiilor recursive urmează.

Definiţie 2.1: Fie $(D, \sqsubseteq)$ o mulțime parțial ordonată.

(1) O submulțime $X \subseteq D$ se numește *lanț* dacă $X \neq \emptyset$ și pentru orice $a, b \in X$, $a \sqsubseteq a$ sau $b \sqsubseteq a$.

(2) Poset-ul $(D, \sqsubseteq)$ se numește *complet* (eng. *cpo*) dacă orice lanț în D are cea mai mică margine superioară în D.

(3) D se numește *punctat* sau *strict* (eng. *pcpo/scpo*) dacă este complet și D însuși are un cel mai mic element.

(4) Fie A, B două cpo. O funcție $f : A \to B$ se numește *monotonă* (crescătoare) dacă respectă ordinea, i.e. $a_1 \sqsubseteq a_2$ în A implică $f(a_1) \sqsubseteq f(a_2)$ în B, relațiile de ordine parțială fiind cele ale domeniilor respective.

(5) O funcție monotonă $f : A \to B$ se numește *continuă* dacă pentru orice lanț $X \subseteq A$ are loc:

$$f\left(\bigsqcup X\right) = \bigsqcup \{f(x) \mid x \in X\},$$

unde $\sqcup = \sup$.

Observaţie 2.2: (1) Adesea, este foarte ușor de transformat un cpo într-unul strict, prin reuniune cu un cel mai mic element $\{\bot\}$.

(2) Noțiunile de mai sus pot fi definite și mai riguros, într-o topologie, numită *topologia Scott* (detalii în Abramsky and Jung (1994)).

Rezultatul fundamental care justifică utilitatea celor de mai sus în semantica definițiilor recursive este:

Teoremă 2.1: *Fie D un pcpo și $F : D \to D$ o funcție continuă. Atunci F are un cel mai mic punct fix:*

$$\mathit{fix}F = \bigsqcup\{F^i \bot \mid i \geq 0\}.$$

Demonstraţie. Calculăm succesiv:

$$\begin{aligned}
F\mathtt{fix}F &= F\left(\bigsqcup\{F^i \bot \mid i \geq 0\}\right) \\
&= \bigsqcup\left\{F(F^i \bot) \mid i \geq 0\right\} \quad \text{(continutate)} \\
&= \bigsqcup\left\{F^i \bot \mid i \geq 1\right\} \\
&= \bigsqcup\left\{F^i \bot \mid i \geq 0\right\} \\
&= \mathtt{fix}F.
\end{aligned}$$

Penultimul pas este justificat de $F^0 \bot = \bot \sqsubseteq F\bot$, din definiția celui mai mic element, $\bot$.

Fie acum e un alt punct fix. Rezultă:

$$\bot \sqsubseteq e \Rightarrow F^i \bot \sqsubseteq F^i e = e, \forall i \Rightarrow \sqcup\left\{F^i \bot \mid i \geq 0\right\} \sqsubseteq e \Rightarrow \mathtt{fix}F \sqsubseteq e.$$

□

Putem acum formula:

Definiţie 2.2: În condițiile și cu notațiile de mai sus, semantica specificației recursive $f = Ff$ este $\mathtt{fix}F$, cel mai mic punct fix al lui F.

O observație importantă pentru semantica unor limbaje de programare simple este: se poate arăta că, dacă D_1, D_2 sînt pcpo, atunci și domeniile compuse $D_1 + D_2$, $D_1 \times D_2$, $[D_1 \to D_2]$ sînt pcpo, iar constructorii sînt operații continue.

3 Categorii cartezian închise

Prezentarea de mai jos urmează Barr and Wells (1990) și arată cum domeniile semantice introduse mai sus pot fi interpretate în contextul categoriilor cartezian închise.

Începem cu o observație simplă: mulțimea cpo stricte și cu funcții continue și stricte (i.e. $f \perp = \perp$) formează o categorie, care este o subcategorie a cazului nestrict.

Vom vedea cum cazul nestrict ne permite, de fapt, să lucrăm cu categorii cartezian închise. Totodată, prin aceasta vom putea lega teoria domeniilor de tip cpo de semantica λ-calculului cu tipuri simple, asociat canonic unei categorii cartezian închise.

Definiţie 3.1: Fie $\mathcal{C}$ o categorie. Ea se numește *cartezian închisă* (CCC) dacă:

(CCC1) are un obiect terminal 1;

(CCC2) orice pereche de obiecte $A, B \in \mathcal{C}$ are un produs direct $A \times B$, împreună cu proiecțiile canonice:

$$A \xleftarrow{p_1} A \times B \xrightarrow{p_2} B;$$

(CCC3) pentru orice obiecte $A, B \in \mathcal{C}$, există un obiect $[A \to B]$ și o săgeată

$$\texttt{eval} : [A \to B] \times A \to B,$$

cu proprietatea de universalitate: pentru orice săgeată $f : C \times A \to B$, există o unică săgeată $\lambda f : C \to [A \to B]$ astfel încît compunerea:

$$C \times A \xrightarrow{\lambda f \times A} [A \to B] \times A \xrightarrow{\texttt{eval}} B$$

să fie exact f.

Cu alte cuvinte, avem diagrama comutativă din figura 1.

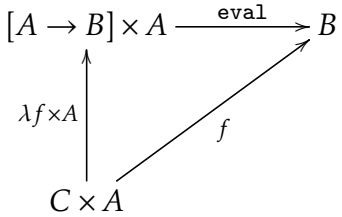

Figura 1: Proprietatea de universalitate a obiectului $[A \to B]$

Obiectul $[A \to B]$ se mai numește *exponențială*, se mai notează B^A, iar A se numește *exponent*.

Continuăm cu cîteva exemple esențiale.

Categoria mulțimilor, `Set` este cartezian închisă, cu structura $[A \to B] = \mathrm{Func}(A, B)$, mulțimea funcțiilor $A \to B$, iar săgeata `eval` este:

$$\texttt{eval} : [A \to B] \times A \to B, \quad \texttt{eval}(f)(a) = f(a).$$

Orice **algebră Boole** B este CCC. Mai întîi, ea este un poset, deci are o categorie asociată canonic $\mathcal{C}(B)$, cu cel mult o săgeată între oricare două elemente (definită dacă și numai dacă elementele sînt comparabile).

B are un obiect terminal, 1, iar produsul este $\wedge$.

Pentru structura de CCC, definim acum $[a \to b] = \neg a \vee b$. Existența săgeții `eval` înseamnă $[a \to b] \wedge a \leq b$, deci, folosind proprietatea de latice distributivă, obținem succesiv:

$$\begin{aligned}
[a \to b] \wedge a &= (\neg a \vee b) \wedge a \\
&= (\neg a \wedge a) \vee (b \wedge a) \\
&= 0 \vee (b \wedge a) \\
&= b \wedge a \leq b.
\end{aligned}$$

Existența săgeții $\lambda f : c \to [a \to b]$ asociate unei săgeți $f : c \wedge a \to b$ înseamnă, de fapt:

$$c \wedge a \leq b \Rightarrow c \leq [a \to b].$$

Să remarcăm că unicitatea va fi automată, deoarece în orice poset există cel mult o săgeată între oricare două obiecte.

Mai departe, presupunem $c \wedge a \leq b$. Rezultă:

$$\begin{aligned}
c &= c \wedge 1 \\
&= c \wedge (a \vee \neg a) \\
&= (c \wedge a) \vee (c \wedge \neg a) \\
&\leq b \vee (c \wedge \neg a) \\
&\leq b \vee \neg a \\
&= [a \to b].
\end{aligned}$$

Categoria cpo cu funcții continue este CCC, dar **subcategoria strictă nu este**.

Demonstrația este destul de tehnică și folosește esențialmente detalii privitoare la continuitate în topologia Scott, arătîndu-se faptul că strictețea nu se păstrează obligatoriu în proprietatea de universalitate a obiectului exponențial. Detalii se pot găsi în Reynolds (2000), pp. 37-39.

Următoarele două exemple sînt prezentate nu tocmai riguros, dar ele pot servi drept punct de plecare pentru o interpretare formală.

Putem privi un **limbaj simplu de programare funcțională ca alcătuind o categorie**. Obiectele sînt tipurile, presupunem că fiecare tip are o operație `id`, iar funcțiile sînt morfisme.

Dacă vrem ca acest limbaj să descrie o CCC, trebuie să ținem cont că:

- obiectul exponențial `[A → B]` va fi obiectul de funcții `func :: A -> B`. Axiomele CCC impun „funcții clasa I", adică obiectul `[A → B]` să se comporte ca orice alt obiect din categorie, i.e. tip, i.e. să poată fi luat ca argument de funcții;
- proprietatea de universalitate a CCC poate ridica probleme de implementare, mai ales în forma generalizată cu produse n-are.

Conform Lambek (1989), o categorie poate fi gîndită ca un **sistem deductiv**, obiectele fiind formule, iar morfismele fiind demonstrații.

Dacă vrem ca această categorie să fie cartezian închisă, obținem reguli de deducție binecunoscute.

Fie A, B formule. Atunci $[A \to B]$ ar trebui să fie tot o formulă, pe care o putem gîndi ca pe o implicație. Atunci `eval` este o demonstrație care deduce B din A și $[A \to B]$. Ea joacă, de fapt, rolul *modus ponens*.

Proprietatea de universalitate pornește cu $f : C \times A \to B$, adică o deducție a lui B din C și A (intuitiv, dar și formal, este chiar conjuncția) și produce o deducție $\lambda f : C \to [A \to B]$, care se mai numește *regula detașării*.

Mai multe detalii se pot găsi în Lambek (1989) și un rezumat succint în notițele din semestrul anterior.

4 Concluzii și perspective

În această prezentare, am oferit cîteva detalii privitoare la aplicații concrete ale corespondenței de tip Curry–Howard–Lambek în semantica denotațională.

Fiecare dintre cele trei ramuri ale corespondenței fac obiectul studiului meu ulterior, iar cîteva referințe bibliografice de urmărit sînt:

- îmbinarea teoriei domeniilor în special cu λ-calculul, dar și cu categoriile cartezian închise este detaliată excelent în Amadio and Curien (1996);
- aspecte logice fundamentale, atît în ce privește logica clasică de ordinul întîi, dar și trecerea către sisteme logice moderne, cu aplicații în programarea funcțională sînt conținute în Girard (2013) și Girard (1989);

- „traducerea“ logicii clasice, via algebre universale, către teoria categoriilor este prezentată succint, dar suficient de riguros în Pitts (1995).

Desigur, în afară de aceste referințe punctuale, rămîn de interes deosebit monografiile:

- Barendregt (1985) și Barendregt et al. (2010) pentru toate noțiunile fundamentale privitoare la λ-calcul, varianta tipizată sau nu;
- Jacobs (1999), pentru aspecte esențiale privitoare la interpretarea logicii în categorii;
- Lambek and Scott (1994), care pune laolaltă contribuțiile lui J. Lambek privitoare la interpretarea categoriilor drept sisteme deductive;
- Sørensen and Urzyczyn (2006), în care este sumarizată corespondență Curry–Howard, plină de detalii și demonstrații complete ale acestei legături.

Bibliografie

Abramsky, S. and Jung, A. (1994). Domain theory. In Gabbay, D., editor, *Handbook of Logic in Computer Science*, volume 3, chapter 1, pages 2–169. Clarendon Press.

Amadio, R. and Curien, P.-L. (1996). *Domains and Lambda-Calculi*. Cambridge.

Barendregt, H. (1985). *The Lambda Calculus, Its Syntax and Semantics*. North Holland.

Barendregt, H., Dekkers, W., and Statman, R. (2010). *Lambda Calculus with Types*. Cambridge University Press.

Barr, M. and Wells, C. (1990). *Category Theory for Computing Science*. Prentice Hall.

Girard, J.-Y. (1989). *Proofs and Types*. Cambridge University Press.

Girard, J.-Y. (2013). *The Blind Spot: Lectures on Logic*. European Mathematical Society.

Jacobs, B. (1999). *Categorical Logic and Type Theory*. North Holland.

Lambek, J. (1989). On some connections between logic and category theory. *Studia Logica*, 48(3):269–278.

Lambek, J. and Scott, P. J. (1994). *Introduction to Higher Order Categorical Logic*. Cambridge University Press.

Pitts, A. (1995). Categorial logic. In Gabbay, D., editor, *Handbook of Logic in Computer Science*, volume 5, chapter 2, pages 39–129. Clarendon Press.

Reynolds, J. (2000). Categories and cpos. disponibil online. notițe de curs.

Schmidt, D. A. (2009). *Denotational Semantics*. Brown & Co. disponibilă online.

Scott, D. and Strachey, C. (1971). *Towards a mathematical semantics for computer languages*. Oxford Computing Lab.

Sørensen, M. H. and Urzyczyn, P. (2006). *Lectures on the Curry-Howard Isomorphism*. Elsevier.