3.5 Funcţia Error

Haskell are o funcţie integrată numită **error** al cărei tip este **String->a**. Această funcţie este oarecum ciudată: Tipul ei arată ca şi cum ar întoarce o valoare de tip polimorfic despre care nu ştie nimic, de vreme ce nu primeşte niciodată o valoare de acest tip ca argument!

De fapt, *este* o valoare comună tuturor tipurile: _|_. Într-tadevăr, semantic aceasta este exact valoarea care este întotdeauna întoarsă de **error** (Amintiti-vă că toate erorile au valoarea _|_). Oricum, ne putem aştepta ca o implementare rezonabilă să afişeze argumentul de tip String primit de **error** pentru scopuri de diagnosticare. Aşadar această funcţie este folositoare când dorim să terminăm un program sau când un parametru primit a fost greşit. De exemplu, actuala definiţie a funcţiei **head** luată din Standard Prelude este:

```
head  (x:xs)          =  x
head    [ ]           =  error "head{PreludeList}: head [ ]"
```

4 Expresii Case şi Pattern Matching(potrivire de şabloane)

Mai devreme am dat câteva exemple de potrivire de şabloane în definirea funcţiilor --- de exemplu **length** şi **fringe**. În această secţiune vom examina procesul de potrivire de şabloane mai detaliat (§3.17).8

Şabloanele nu sunt valori de primă clasă; există un set fixat de diferite tipuri de şabloane. Am văzut deja câteva exemple de şabloane de constructori de date; atât **length** cât şi **fringe** definite mai devreme sunt construite pe astfel de şabloane, primul pe constructorul unui tip integrat (liste), ultimul pe tipul definit de utilizator (arbori). Într-adevăr, potrivirea este permisă folosind constructori de orice tip, definiţi de utilizator sau nu. Acestea includ n-uple, stringuri, numere, caractere, etc. De exemplu, uitaţi o funcţie *contrived* care potriveşte un n-uplu de constante.

```
contrived  : :  ([a],  Char,  (Int,  Float),  String,  Bool)  -> Bool
contrived       ([ ],    'b',     (1, 2.0),      "hi",    True)   = False
```

Acest exemplu demonstrează deasemenea că încuibărirea şabloanelor este permisă (la o adâncime arbitrară).

Tehnic vorbind, *parametrii formali*9 sunt deasemenea sabloane doar că ei nu eşuează niciodată în potrivirea cu o valoare. Ca o consecinţă a potrivirii cu succes, parametrul formal este legat la valoarea cu care a fost potrivit. Din acest motiv şabloanele dintr-o ecuaţie nu pot avea mai multe apariţii ale aceluiaşi parametru formal (o proprietate numita linearitate §3.17,§3.3,§4.4.2).

Şabloanele de genul parametrilor formali care nu dau niciodată greş la potrivire se spune că sunt irefutabile, în contrast cu şabloanele refutabile care pot eşua la potrivire. Şablonul folosit in exemplul cu funcţia *contrived* de mai

8Potrivirea de şabloane în Haskell este diferită de cea întâlnită în limbajele de programare logice ca Prolog; în particular aceasta poate fi privită ca o potrivire „într-un singur sens”, în vreme ce Prologul permite potrivirea „în două sensuri” (prin unificare), cu urmărire implicită în mecanismul său de evaluare.

9Raportul le numeşte pe acestea variabile.

sus este refutabil. Mai sunt încă trei tipuri de şabloane irefutabile, iar două dintre ele le vom prezenta acum (celălalt îl vom amâna până la Secţiunea 4.4).

Şabloanele-@. Câteodată este convenabil să dăm un nume unui întreg şablon necesar pentru folosirea în partea dreaptă a unei ecuaţii. De exemplu, o funcţie care duplică primul element dintr-o listă poate fi scrisă aşa:

```
f (x : xs)          = x : x : xs
```

(Amintiţi-vă ca „:” asociază la dreapta.) Ţineti minte că x:xs apare şi ca şablon în partea stângă, dar şi ca expresie în partea dreaptă. Pentru a îmbunătăţi lizibilitatea programului, ar fi de preferat să scriem x:xs o singură dată, lucru pe care îl putem obţine folosind *şabloanele-@* după cum urmează:10

```
f s@(x : xs)        = x : s
```

Tehnic vorbind, şabloanele-@ întotdeauna reuşesc cu succes să se potrivească, deşi sub-şablonul (în acest caz x:xs) ar putea desigur să eşueze.

Jockeri. Altă situaţie des întâlnită este potrivirea cu o valoare despre care nu ne interesează nimic. De exemplu, funcţiile head şi tail definite în Secţiunea 2.1 pot fi rescrise astfel:

```
head (x : _)        = x
tail (_ : xs)       = xs
```

în care am făcut vizibil faptul că nu ne interesează care este o anumită parte a intrărilor.Fiecare jocker se potriveşte independent cu orice, dar în contrast cu un parametru formal, niciunul nu se leagă de nimic; din acest motiv pot exista mai mult de unul în tr-o ecuaţie.

4.1 Semantica potrivirii de şabloane

Până acum am discutat despre şabloanele individuale: sunt potrivite, cum unele sunt refutabile, altele sunt irefutabile, etc. Dar cum decurge procesul în ansamblu? În ce ordine sunt încercate potrivirile? Dar dacă nu reuşeşte niciuna? Această secţiune se adresează acestor întrebări.

Potrivirea şabloanelor *poate eşua,* poate *reuşi* sau *diverge*. O potrivire reuşită leagă parametrii formali din şablon de valori. Divergenţa apare când o valoare necesară şablonului conţine o eroare (_|_). Procesul de potrivire însuşi decurge de sus în jos, şi de la stânga la dreapta. Eşecul unui şablon oriunde într-o ecuaţie înseamnă eşecul întregii ecuaţii, şi apoi este încercată următoarea ecuaţie. Dacă toate ecuaţiile eşuează, valoarea funcţiei de aplicare este _|_, şi are ca rezultat o eroare în timpul execuţiei.

De exemplu, dacă [1, 2] este potrivit cu [0, **bot**], atunci 1 eşuează să se potrivească cu 0, aşa că rezultatul este o potrivire *eşuată*. (Amintiţi-vă că **bot**, definit mai devreme, este o variabilă legată de _|_). Dar daca [1, 2] este potrivit cu [bot, 0], atunci potrivirea lui 1 cu **bot** produce divergenţă (adică _|_).

8Potrivirea de şabloane în Haskell este diferită de cea întâlnită în limbajele de programare logice ca Prolog; în particular aceasta poate fi privită ca o potrivire „într-un singur sens”, în vreme ce Prologul permite potrivirea „în două sensuri” (prin unificare), cu urmărire implicită în mecanismul său de evaluare.

9Raportul le numeşte pe acestea variabile.

Cealaltă deviere de la acest set de reguli este că şabloanele (de nivel înalt - exterioare) pot avea de asemenea o gardă booleană, ca în această definiţie a unei funcţii ce ne furnizează semnul unui număr:

```
sign x | x > 0          = 1
       | x == 0         = 0
       | x < 0          = -1
```

Ţineţi minte că pentru acelaşi şablon pot fi date mai multe gărzi. Iar ca şi şabloanele, gărzile sunt evaluate de sus în jos. Prima care este evaluată la True (Adevărat) produce o potrivire cu succes şi alege astfel formula de calcul a rezultatului.

8Potrivirea de şabloane în Haskell este diferită de cea întâlnită în limbajele de programare logice ca Prolog; în particular aceasta poate fi privită ca o potrivire „într-un singur sens”, în vreme ce Prologul permite potrivirea „în două sensuri” (prin unificare), cu urmărire implicită în mecanismul său de evaluare.

9Raportul le numeşte pe acestea variabile.