Danmarks Tekniske Universitet

Bachelor Projekt

TITLE

Jonas Dahl Larsen (s205829)

2. marts 2024

Indhold

1 Introduktion

I 2023 valgte Danmarks Tekniske Universitet at anvende Python som et hjælpeværktøj i deres grundlæggende matematikkursus "01001 Matematik 1a (Polyteknisk grundlag)". Python er et af de mest anvendte programmeringssprog [1], kun overgået af to sprog, der primært bruges sammen til at udvikle hjemmesider. Derfor har Python, med sit dynamiske skrevne sprog og en række matematiske programudvidelser som SymPy [2], været et oplagt valg som programmeringssprog til det grundlæggende matematikkursus tilbudt af DTU.

Projektet vil undersøge, hvordan et funktionsprogrammeringssprog, kan gavne de studerendes forståelse af de grundlæggende matematiske koncepter. Formålet er at guide læseren gennem opbygningen af en række funktionsprogrammer baseret på grundlæggende universitetsmatematik og dermed illustrere anvendelser. Projektet beskriver en generel struktur til opbygning og anvendelse af et funktions programmeringsprogram. Der tages udgangspunkt i F# [3], men beskrivelserne af programmerne vil også kunne anvendes i lignende funktionsprogrammeringssprog.

Rapporten begynder med at forklare nogle Fundamentale koncepter inden for funktionsprogrammering samt metoder til validering af programmerne.

2 Fundamentale koncepter

2.1 Introduktion til Funktions Programmering

Det er forventet af læseren har kendskab til programmering, der gives derfor kun en kort beskrivelse af syntax og notationen, så læser ikke bekendt med F# kan forstå de eksempler der løbende vil forkomme i rappoten.

$$f(n) = \begin{cases} 1 & n = 0 \\ n \cdot f(n-1) & n > 0 \\ \text{undefined} & n < 0 \end{cases} \tag{1}$$

Vi begynder derfor med at betragte funktionen for fakultet 1, et eksempel på en implementering i F# er givet i 1 som kan sammelignes med Python kode 2, da Python og pseudokode er næsten det samme.

```
// Fakultet i F#
let rec factorial n =
    match n with
    | 0              -> 1
    | x when x > 0   -> x * factorial (x - 1)
    | _              -> failwith "Negative argument"
```

Listing 1: Eksempel på Fakultet i F#

[1] https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/
[2] https://www.sympy.org/en/index.html
[3] https://en.wikipedia.org/wiki/F_Sharp_(programming_language)

```python
// Fakultet i Python
def factorial(n):
    if n == 0:
        return 1
    elif n > 0:
        return n * factorial(n - 1)
    else:
        raise ValueError("Negative argument")
```

Listing 2: Eksempel på Fakultet i Python

I F# anvendes let til at definere en ny variabel eller, i dette tilfælde, en funktion kaldet factorial. Næste nøgleord er rec, hvilket indikerer, at funktionen er rekursiv. Funktionen tager et input-argument n, og i linje 3 starter et match-udtryk. Her er n vores udtryk, og efter with begynder en række mønstre, som udtrykket forsøger at matche på, separeret med '|'.

Et match på et mønster er ikke det samme som '==' kendt fra andre programmeringssprog. I linje 4 forsøger den at tildele værdien af n til 0, og dette lykkes kun, hvis n er 0. Hvis n ikke er 0 og dermed ikke matcher linje 4, vil den forsøge at matche det næste mønster. Her står der kun x, da der ikke yderligere specificeres om netop x, vil det altid lykkes, og x bliver tildelt værdien af n svarende til $[x \mapsto n]$. Derefter skal betingelserne efter when være opfyldt. Hvis de er det, kan mønsteret matche på den givne linje. Derefter eksekverer den og returnerer koden efter '$\rightarrow$', hvor den samtidig har adgang til den nyligt tildelte værdi af x. Det sidste mønster på linje 6 anvender '_' som mønster. Dette betyder, at det kan matche alle udtryk, men vi er ikke interesseret i at anvende værdien. I dette tilfælde kan det tredje mønster på linje 6 kun køres, når n er negativ.

Bemærk hvordan en funktion i F# altid returnere sidste kørte linje af en funktion. Svarende til man ikke skrev return i linje 4 og 6 af Python koden2, men værdierne stadig blev returneret.

2.1.1 Typer

Alle funktioner har en type i F#, typen for factorial1 er $int \rightarrow int$. Det er ikke muligt at kalde funktionen med et argument der ikke er af typen int. Typen for funktionen skrives som $Factorial : int \rightarrow int$. Vi kan dermed formulere følgende omkring typer[4]:

$$f : T_1 \rightarrow T_2$$
$$fe : T_2 \iff e : T_1$$

Er en funktion kaldt med et argument der ikke matcher funktionen type, gives en fejlmeddelelse. Derudover kan en type også være bestående at en tuple af typer.

$$f : T_1 * T_2 * .. * T_n \rightarrow T_{n+1}$$
$$f(e_1, e_2, .., e_n) : T_{n+1} \iff e_1 : T_1 \wedge e_2 : T_2 \wedge .. \wedge e_n : T_n$$

En tuple som kun består af to typer, kaldes et par.
I F# er det ikke nødvendigt at anvende parenteser som i andre programmeringssprog. De vil

[4] s14 FPU F#

derfor kun anvendes hvor det er nødvendige, gennem rapporten. Når man laver en tuple er det nødvendigt. Givet en funktion $g : T_1 \to T_2 \to T_3$ betyder den tager to udtryk af typen T_1 og T_2 hvor evalueringen af funktionen giver T_3

2.2 Property-based testing