Prolog - Apple enheds filtrering

Af Zhentao Wei 3.S - Københavns mediegymnasium

Kravsspecifikation

Som det første i vores projekt har vi defineret krav for programmet, som programmet skal opnå. Det er vigtigt at opstille specifikke krav for et program før implantationen, for at holde orden og for at holde overblik for forventningerne af programmet. Uden en kravsspecifikation før implantationen kan skabe uorden i arbejdes processen. Nedenunder er alle vores opstillet kravsspecifikation over for programmet.

- Implementeret med Prolog
- Udnytter fordelene af prolog
- Simpelt

Design

Logisk programmering er en form af programmeringssprog, hvilket er baseret på logisk matematik. Et eksempel på en af disse programmeringssproge baseret på logik er Prolog. Disse programmeringssproge er fuldstændigt datadrevet, hvilket betyder at det er deterministisk ude fra dataet som den får. Fordi den er deterministisk, så er det altid det samme svar programmet resulterer i.[1]

Logisk programmeringssproge er ofte opbygget af 2 dele. Den første del er "head" delen, som definerer alt data'en/fakta'en for programmet. Den anden del er "body" sektionen hvor alle regler bliver defineret. Denne body del er ikke nødvendigt at inkludere for programmeringssprogets funktionalitet og derfor i nogle tilfælde ikke inkluderet i nogle programmer.[2]

Logisk programmering forsker sammenhængende imellem data. Altså, deres relationer mellem 2 eller flere data punkter. Et eksempel fremhævet af bogen Matematiske Horisonter ""København er i Danmark" og "Danmark er i Skandinavien"." I dette stykke viser de eksemplet med at købehavn er i Danmark, og Danmark er i Skandinavien. Udefra det er det nemt at observere at København også ligger i Skandinavien. Det er det som logisk programmering specialiserer til; relationer imellem data. Derfor er logisk programmering data-orienteret og relationen mellem data baseret.[3]

Problemformulering

Hvordan kan man finde en tilpassende Apple enhed?

Idebeskrivelse

Vi har valgt et program der kan udvælge alle enheder der passer til de parametre, som vi sender til programmet. Grundet til dette valg er for dens simplicitet, hvilket gør at dette program passer perfekt for alle kravene i kravsspecifikationen. Programmet kommer til at give os alle enheder i databasen, eller filtrer for år udgivet, eller filtrer for pris, eller filtrer for begge.

Illustrationer

Dette er et diagram overfor data'en, hvilket vi har indsamlet fra ChatGPT som eksempel data.

iphone_info	
PK	device name
	year released
	price

ipad_info	
PK	device name
	year released
	price

[1] https://www.linode.com og https://www.compute.dtu.dk/om-os/matematiske-horisonter
[2] https://www.linode.com og https://www.compute.dtu.dk/om-os/matematiske-horisonter
[3] https://www.linode.com og https://www.compute.dtu.dk/om-os/matematiske-horisonter

Implementering

Vores kode er overfladisk komprimeret af 6 dele, hvilket er implementeret i det logiske programmeringssprog Prolog og eksekveret med Swish. Disse dele er beskrevet i dybere detaljer i de neste følgende tekster.

```
% iPhone
iphone_info(iphone_5, 2012, 199).
iphone_info(iphone_5s, 2013, 199).
iphone_info(iphone_6, 2014, 199).
iphone_info(iphone_6_plus, 2014, 299).
iphone_info(iphone_6s, 2015, 199).
iphone_info(iphone_6s_plus, 2015, 299).
iphone_info(iphone_7, 2016, 199).
iphone_info(iphone_7_plus, 2016, 299).
iphone_info(iphone_8, 2017, 699).
iphone_info(iphone_8_plus, 2017, 799).
iphone_info(iphone_x, 2017, 999).
iphone_info(iphone_xr, 2018, 749).
iphone_info(iphone_xs, 2018, 999).
iphone_info(iphone_xs_max, 2018, 1099).
iphone_info(iphone_11, 2019, 699).
iphone_info(iphone_11_pro, 2019, 999).
iphone_info(iphone_11_pro_max, 2019, 1099).
iphone_info(iphone_se_2nd_gen, 2020, 399).
iphone_info(iphone_12_mini, 2020, 699).
iphone_info(iphone_12, 2020, 799).
iphone_info(iphone_12_pro, 2020, 999).
iphone_info(iphone_12_pro_max, 2020, 1099).
iphone_info(iphone_13_mini, 2021, 699).
iphone_info(iphone_13, 2021, 799).
iphone_info(iphone_13_pro, 2021, 999).
iphone_info(iphone_13_pro_max, 2021, 1099).

% iPad
ipad_info(ipad_4, 2012, 499).
ipad_info(ipad_air, 2013, 499).
ipad_info(ipad_mini_2, 2013, 399).
ipad_info(ipad_mini_3, 2014, 399).
ipad_info(ipad_air_2, 2014, 499).
ipad_info(ipad_mini_4, 2015, 399).
ipad_info(ipad_pro_12_9, 2015, 799).
ipad_info(ipad_pro_9_7, 2016, 599).
ipad_info(ipad_5th_gen, 2017, 329).
ipad_info(ipad_pro_10_5, 2017, 649).
ipad_info(ipad_6th_gen, 2018, 329).
ipad_info(ipad_mini_5th_gen, 2019, 399).
ipad_info(ipad_air_3rd_gen, 2019, 499).
ipad_info(ipad_7th_gen, 2019, 329).
ipad_info(ipad_pro_11, 2018, 799).
ipad_info(ipad_pro_12_9_2nd_gen, 2017, 799).
ipad_info(ipad_pro_12_9_3rd_gen, 2018, 999).
ipad_info(ipad_8th_gen, 2020, 329).
ipad_info(ipad_air_4th_gen, 2020, 599).
ipad_info(ipad_pro_11_2nd_gen, 2020, 799).
ipad_info(ipad_pro_12_9_4th_gen, 2020, 999).
```

iPhone data

I header sektionen i logisk programmering opbygget af fakta, hvilket senere vil blive brugt i bodysektionen. Alle disse opskrevet fakta opbygger en salgs af database med relationer mellem navn, årstal, og pris. I dette tilfælde er en database over iPhone.

IPad data

I forlængelse til iPhone data'en, så er der også en del af fakta af iPads i header sektionen. Her bliver der defineret fakta i samme format lige som iPhone faktaene, men her er der defineret data om IPads.

Kombiner data

```
54 get_device_data(Device, Year, Price) :-
55     iphone_info(Device, Year, Price);
56     ipad_info(Device, Year, Price).
```

I forbindelse af de to forrige header fakta sektioner, er de kombineret her i dette stykke af kode for at nemmere gøre filtrerings kode stykket. For når de begge kombineres til samme format, så er det muligt at bruge den samme algoritme for filtreringsprocessen. Dette stykke af kode og resten er kendt som bodysektionen af et logisk programmeringssprogram. I bodysektionen er dataet fra det forrige brugt for udredninger og for at skabe nyt data, afhængigt af det forrige data.

```
58 filter_devices_year(YearMin, YearMax, Device) :-
59     get_device_data(Device, Year, _),
60     (YearMin == -1; Year >= YearMin),
61     (YearMax == -1; Year =< YearMax).
62
63 filter_devices_price(PriceMin, PriceMax, Device) :-
64     get_device_data(Device, _, Price),
65     (PriceMin == -1; Price >= PriceMin),
66     (PriceMax == -1; Price =< PriceMax).
```

Filtreringskoden

Med hjælp af det kombineret kode og som nævnt i den forrige tekst, så filtrere vi alle enhederne for pris og/eller udgivelses år. Som der kan ses, er der også implementeret et annullere værdi, hvilket betyder hvis filteret er sat til -1, så vil den ikke filtrere efter den. Eller med andre ord, så er det muligt at filtrere for kun enheder større end en givet pris. Dette bliver vis senere i en af test-casene.

Kombiner filterne

Det forrige stykke af kode Indholt 2 filtre, den første var for at filteret efter udgivelses år og det anden stykke filtreret i forhold til enhedens pris. Disse 2 filtre funktionerer begge selvstændigt, men i dette stykke er de begge kombineret. I funktionen "search_device" så kan man både filetere efter pris og udgivelsesår. Dette er implementeret for at forbinde hele programmet sammen.

```
68 % Get devices from price and year range
69 search_device(PriceMin, PriceMax, YearMin, YearMax, Device) :-
70     filter_devices_year(YearMin, YearMax, Device),
71     filter_devices_price(PriceMin, PriceMax, Device).
```

Dokumentationen

Til sidst har vi lavet et lille stykke af kommenterende kode for at vise brugeren hvordan brugen af programmet virker. Dette er gjort for at brugeren nemmere kan gennemskue brugen af programmet.

```
74 % Usage: search_device(200, 500, 2015, 2019, X)
75 % 1: price min
76 % 2: price max
77 % 3: Year min
78 % 4: Year max
79
80 % Use "-1" for ignore
```

Test

Vi har valgt at opstille test-cases og teste dem. Test-case metoden er brugt for at teste elementer af et produkt, for at se om alt fungerer som forventet eller der er fejl i programmet. Hver test-case indeholder en ID, beskrivelse, trin, forventet resultater, reelle resultater og bedømmelse.

ID'en er brugt for at holde styr på den specifikke test-case. Dette er nødvendigt for tilfælde med 2 eller flere test-cases der er næsten magen til.

Beskrivelsen er en kort forklaring af formålet af test-casen.

Test trinene er en beskrivelse af alle trin involveret i casen. Dette er beskrevet med dybe detaljer for gentagelighedens skyld. For vis der er fejl i produktet, så er det vigtigt at kunne genskabe problemet flere gange.

Forventet resultater..... (Du forstår. Behøver ikke at forklare det til dig.)

Reelle resultater...

Bedømmelse... **Pass/Fail**...

Sammen med alle de kategorier der skabes en effektiv og dybgående test af vores produkt, hvilket i dette tilfælde er koden lavet i Prolog og kørt med Swish.

Test-cases

Test case ID	Test case beskrivelse	Test trin	Forventet resultater	Resultater	Pass/ Fail
1	Ingen filter	1. Input ”search_device(-1, -1, -1, -1, X)” 2. Kør koden	Alle enheder bliver vist	Alle enheder	Pass
2	Budget på 300	1. Input ”search_device(-1, 300, -1, -1, X)” 2. Kør koden	Alle enheder billigere end 300	Alle enheder billigere end 300	Pass
3	Vil bruge 300	1. Input ”search_device(300, -1, -1, -1, X)” 2. Kør koden	Alle enheder dyre end 300	Alle enheder dyre end 300	Pass
4	Vil have enheder før 2017	1. Input ”search_device(-1, -1, -1, 2017, X)” 2. Kør koden	Alle enheder før 2017	Alle enheder før 2017	Pass
5	Vil have efter/i 2020	1. Input ”search_device(-1, -1, 2020, -1, X)” 2. Kør koden	Alle enheder efter/i 2020	Alle enheder efter/i 2020	Pass
6	Specifikt år og pris	1. Input ”search_device(999, 999, 2020, 2020, X)” 2. Kør koden	ipad_pro_12_9_4th_gen	ipad_pro_12_9_4th_gen	Pass

Diskussion af test-cases

Udefra test-case skemaet, så klaret Prolog programmet det fremragende og udearbejdet 6 ud af 6 pass. Alle scenarier blev testet og viste sig ingen problemer, og derfor kan vi gå videre med vores produkt.

Konklusion

For at konkludere, så har vi lavet er program for at søge efter Apple enheder med optionelle filter efter pris og efter udgivelses år. I starten af teksten blev der opstillet de grundlæggende krav for produktet. Derefter blev der forklaret om de overfladiske principper i logisk programmering, så som header og body, og data-orienteret programmeringssprog. Senere blev der refereret til bogen "Matematiske Horisonter" for dens simple eksempel af lande og deres relationer. Igennem implantationsdelen er der skrevet om de 6 dele af programmet. Denne implantation gjorde det muligt at filtrere imellem pris og udgivelses af Apple produkter. Tilsidst blev der opstillet test-cases, hvor alle test-cases bestod deres test, hvilket betyder at vores produkt virker præcist som forventet.