INF-121 Obligatorisk oppgave 2, H-2015

Fremgangsmåte

- Løsningen leveres elektronisk til en katalog "Oblig-2" under "Innlevering/Vurderingsmappe/" på kursets hjemmeside.
- Løsningen skal leveres i en zip fil med navn som "FORNAVN.ETTERNAVN.zip".
- Innleveringsfristen er ***MANDAG, 16 November, kl.12:00***. Det blir *ikke* mulig å forbedre karakteren – man må få innleveringen godkjent ved første forsøk.
- Eneste akseptable formatet er ren tekstfil (bruk helst kun ASCII tegn), som inneholder kjørbar kode. Koden som ikke kan kjøres gir et stryk/F. (***Ditt navn***, samt eventuelle tilleggsopplysning/-forklaring ***skal stå som kommentarer*** øverst i programteksten.)
- Kommenter ikke-åpenbare "triks" i koden din (uten å gjenta oppgave-teksten).
- Kommenter alle eventuelle presiseringer du har gjort i oppgaveteksten.
- Når oppgaven ber deg om å lage et predikat eller konstant med et gitt navn, er det ***vesentlig*** at du gir den akkurat det navnet/den typen.
- Tilnærmingsmåte og forslagene kan diskuteres med hverandre, men hver enkel må levere inn en uavhengig løsning som ikke må ha noen identiske deler med andres. Løsninger som har slike identiske deler blir vurdert som stryk/F. Hva som teller som 'identisk' bestemmes av lærer.
- Les nøye *hele* oppgavesettet før du begynner å løse enkelte deloppgaver.

  Oppgaven har flere deler, hvor de senere bygger på de foregående. Hvis du besvarer hele oppgaven er resultatet et program, og dette kan leveres inn uten å ha egne filer for de første deloppgavene. Men dette programmet må virke! Hvis du har jobbet med en deloppgave uten å ha fått den helt til, så legg den ved i en separat fil slik at vi kan evaluere det du har fått til. Skriv i en innledende kommentar, hvilke deloppgaver du har besvart.
- Spørsmål om oppgaven kan stilles på gruppen: **Mandag/Torsdag, 9/12 november**.
- Det er ingen forelesninger igjen, med mindre det kommer konkrete spørsmål. Disse sendes til `michal@ii.uib.no` innen fredag 20 november. Blir det behov for det, så møtes vi igjen 26 eller 27 november.
- Gjennomgang av løsninger på gruppen: Mandag/Torsdag, 23/26 november.

1 Enkel parsing (for E)

Vi betrakter følgende gramatikken G, med terminalalfabetet $\{a, b, c\}$ og startsymbolet S

S := aA | cS

A := bB

B := cS | aA | ϵ – ϵ betegner den tomme strengen.

Beskriv med ord strenger som er med i dette språket og definer Prolog predikat `acc(+S)` som holder hvis og bare hvis S er en liste med a'er, b'er og c'er som representerer en streng i dette språket.

2 Evaluering av prefiks uttrykk (for E)

Vi betrakter prefiks uttrykk, gitt som lister, der hvert element i en liste er et enkelt symbol: et siffer, eller en binær aritmetsik operasjon +,*,-,/. F.eks., representerer liste [+,*,2,3,-,6,2] uttrykket $+(*(2,3),-(6,2))$, dvs. $(2*3)+(6-2)$, med verdi 10. Implementer Prolog predikat `ev(+L,R)` som, gitt en slik (korrekt) liste L, returnerer verdien av uttrykket i R. For eksempelet over skal man ha:

```
?- ev([+,*,2,3,-,6,2],R).
R = 10.
```

Tilfredstillende løsning til oppgavene 1 og 2 kan være tilstrekkelig for E.

Problemstilling

Resten av oppgaven består av flere deler som går ut på å programmere Prolog predikat

`pl(+N,+Ant,+Type,S)`

som holder dersom S er en plassering av Ant(allet) sjakkbrikker av Type (som springer, løper, el.l.) på $N \times N$ brettet, slik at ingen av brikkene slår hverandre. Plasseringen S angis som liste av par $i : j$, $0 < i, j \leq N$ som identifiserer posisjoner der brikkene er plassert. Vi bruker følgende typer av brikker:

`spr` – for springer

`lpr` – for løper

`trn` – for tårn

`drn` – for dronnning

I første delen, Oppgaver 3 og 4, skal du implementere kun en variant av predikatet `pl`, som kan anta at også det siste argumentet er instansiert ved kall, dvs. slik at programmet kun sjekker om en oppgitt S er en riktig plassering. Dersom du besvarer senere oppgaver (som krever at programmet kan generere alle løsninger), trenger du ikke skrive en separat løsning til Oppgavene 3 og 4.

For noen typer brikker, kan man formulere noen generelle løsningstyper (som funksjon av N og brikketypen), men det er ikke lov å programmere slike. Dvs. programmet ditt må faktisk søke etter løsninger og, ideelt, bør kunne brukes også for andre typer brikker (med egendefinerte lovlige trekk, andre enn i sjakk).

3 For D

Implementer predikatet

`pl(+N,+Ant,+Type,+S)`

for Type `spr`, dvs. springer. F.eks., skal følgende gjelde:

```
?- pl(4,8,spr,[1:2,1:4,2:1,2:3,3:2,3:4,4:1,4:3]).
```

```
true.
?- pl(4,8,spr,[1:2,1:3,2:1,2:3,3:2,3:4,4:1,4:3]).
false.
```

Argumentet `Ant` er ikke nødvendig i denne og neste oppgaven (da den tilsvarer lengden av listen `S`), men vi beholder det for kompatibilitet med oppgaven 5.

NB! Det er lov å innføre noen antakelser om at forslaget `S` er gitt på en bestemt form, f.eks. sortert på en spesiell måte. Gjør du slike antakelser, må de spesifiseres tydelig som en kommentar i besvarelsen.

Hverken her eller i senere oppgaver trenger du å ta hensyn til noen "symmetrier" eller "rotasjoner" mellom forskjellige løsninger. F.eks., løsningen til 8 springere over, kan sees på som en rotasjon (med 90° med klokke) av løsningen [1:1,1:3,2:2,2:4,3:1,3:3,4:2,4:4]. Vi ser bort fra det og betrakter dem som to forskjellige løsninger.

4 For C eller bedre

Implementer predikatet

```
pl(+N,+Ant,+Type,+S)
```

for alle andre typer, dvs. `trn,lpr,drn`.

Hint: Bruk helst typenavn som navn for passende predikater, som kan kalles ved Prolog høyreordens predikat `call` på følgende måte:

```
pred(...P) :- ...,call(P, A1,...,Ak),...
```

der P instansieres til det aktuelle predikatet (`spr`, osv.) mens A1...Ak til dets argumenter utfra den aktuelle konteksten. Har du definert et predikat `spr` med k argumenter

```
spr(Arg1,...,Argk) :- ...
```

vil `pred(...spr)` føre til et kall `spr(A1,...,Ak)`, der `call(spr,A1...Ak)` utføres.

5 For B eller bedre

Implementer predikatet

```
pl(+N,+Ant,+Type,S)
```

som kan generere mulige løsninger for en oppgitt brettstørrelse N, Ant(all) og Type brikker. Implementer også predikatet

```
nrpl(+N,+Ant,+Type,X)
```

som holder dersom X er antallet forskjellige løsninger S generert ved et kall `pl(N,Ant,Type,S)`.

For A må løsningen være ikke bare korrekt, men også perfekt, f.eks., må ikke gi noen duplikate løsninger, må kunne utvides til andre typer brikker (ved å legge til definisjoner for nye predikater, tilsvarende `spr,drn`, osv., med andre regler for lovlige trekk), og må være rimelig effektiv slik at man kan kjøre noen muligheter på brett større enn 5×5. (Tidsbruk, eller antall inferenser, kan sjekkes ved å utføre spørring omgitt av `time(...)`. F.eks. `time(nrpl(5,12,spr,S))` bør ikke overstige et par-tre CPU sekunder, mens `time(nrpl(6,18,spr,S))` bør ikke overstige et par-tre CPU minutter. Å telle antall løsninger til det klassiske problemet, `time(nrpl(8,8,drn,S))`, bør klares under 20, kanskje tom. under 10 CPU sekunder.)