KS i Programmeringsparadigm 2015, del 3: Formella språk och syntaxanalys
2015-12-01 08.15-10.00 med efterföljande kamraträttning

Inga hjälpmedel är tillåtna. Skriv svaren direkt på blanketten. Bonuspoäng från Syntaxlabbarna hösten 2015 kommer automatiskt att tillgodoräknas på denna KS. 12 poäng (utifrån max 20) krävs för godkänt. Sitt kvar ända till klockan 09.00. Då lämnar alla in samtidigt. Därefter tar kamraträttning vid.

1. (8 p)

   I den här uppgiften ska vi arbeta med reguljära uttryck för datum, på formatet "YYYY-MM-DD" t.ex. dagens datum 2015-12-01 eller 0042-02-01 (första februari år 42). Årtal ska alltså innehålla exakt 4 siffror, månader exakt 2 siffror, och dagar exakt 2 siffror. Som synes på exemplen används inledande nollor för att fylla ut om det behövs.

   Du får använda syntaktiskt socker för reguljära uttryck, men inte regex-funktionalitet som inte går att simulera med reguljära uttryck (typexempel bakåtreferenser).

   (a) Vi börjar med att ignorera hur många månader det finns, och hur många dagar det finns i varje månad. Skriv ett reguljärt uttryck för datum där månad och dag tillåts vara vilket tvåsiffrigt tal som helst (inklusive 00). (1p)

   .....

   .....

   (b) Nu finns det ju bara 12 månader på året, ingen månad är längre än 31 dagar, och varken månad eller dag nummer 00 existerar. Skriv ett reguljärt uttryck för datum där månad måste vara mellan 01 och 12, och dag måste vara mellan 01 och 31. (4p)

   .....

   .....

   (c) I verkligheten är ju reglerna för vad som är ett giltigt datum mer komplicerade. Månaderna har olika längd och vissa år är skottår. Vidare har olika kalendrar lite olika regler för detta.
   Är språket med datum på formatet "YYYY-MM-DD" (där år, månad och dag fortfarande måste vara decimala tal med 4, 2, och 2 siffror) alltid reguljärt, oavsett vilka möjliga eller omöjliga regler man kan tänkas hitta på för vilka av dessa strängar som är giltiga datum? Eller går det att hitta på en regel som gör att språket inte längre är reguljärt? Motivera ditt svar! (3p)

   .....

   .....

   .....

   .....

   .....

   .....

2. (9 p)

JSON är ett vanligt text-format för strukturerad data.[1] En JSON-fil kan se ut t.ex. så här:

```
{
  "labbar": ["F1", "F2", "L1", "L2", "S1", "S2", "Inet"],
  "KS": {"funkprog": 17, "logprog": 14, "syntax": 15},
  "godkänd": true }
```

Det finns fem datatyper på värden. Vi har de tre enkla datatyperna *strängar* (t.ex. `"labbar"` och `"S1"` ovan), tal (t.ex. `17` ovan), och Booleska värden (t.ex. `true` ovan).

Vidare finns *listor*, som är (möjligtvis tomma) komma-separerade listor med värden (av vilka som helst av de fem olika typerna), omgivna av hakparenteser, t.ex. listan med labbar ovan.

Slutligen finns *objekt*, som består av en (möjligtvis tom) komma-separerad lista med nyckel-värde-par, omgivna av måsvingar. Varje nyckel-värde-par består av ett sträng-värde (nyckeln), följt av ett kolon, följt av ett värde (av vilken som helst av de fem olika typerna).

En JSON-fil innehåller ett enda värde, av typen "objekt".

I listor och objekt kan typer blandas godtyckligt, och listor och objekt kan nästlas i varandra. Samma nyckel får förekomma mer än en gång i samma objekt. T.ex. är alltså följande en giltig JSON-fil.

```
{"x": [true, 23.2,
       {"a": false, "b": "c", "b": [47, 11], "b": { }},
       42, []] }
```

(a) Som ett första steg för att parsa JSON gör vi en lexikal analys där vi bryter upp en indata-fil i token-typerna "number", "string", "bool", samt de sex speciella tecknen '`{`', '`}`','`[`','`]`', '`,`' och '`:`'. Förklara vad som menas med detta. (2p)

.....

.....

.....

.....

.....

.....

.....

(b) Antag att vi har icke-slutsymboler List och Object för datatyperna lista och objekt. Använd dessa och token-typerna som nämns i (a) för att skriva en grammatikregel för en ny slutsymbol Value som representerar ett JSON-värde. (1p)

.....

.....

[1] Om du råkar vara bekant med JSON sedan tidigare så ignorerar vi några delar av JSON i den här uppgiften. Följ formatet så som det beskrivs i uppgiften.

(c) Skriv grammatikregler för att definiera icke-slutsymbolen List för datatypen lista. Du får använda de andra icke-slutsymbolerna som beskrivs i (b) samt slutsymbolerna från (a), och får definiera ytterligare icke-slutsymboler om du behöver dem. (2p)

.....

.....

.....

.....

(d) Skriv grammatikregler för att definiera icke-slutsymbolen Object för datatypen objekt. Du får använda de andra icke-slutsymbolerna som beskrivs i (b) samt slutsymbolerna från (a), och får definiera ytterligare icke-slutsymboler om du behöver dem. (3p)

.....

.....

.....

.....

.....

(e) Uppgifterna (b)-(d) ger tillsammans en fullständig grammatik för JSON-filer. Men vad ska startsymbolen vara? (1p)

.....

.....

3. (3 p)

Vilka av följande påståenden är korrekta?

Motiveringar krävs ej. För varje påstående kan du svara "sant", "falskt", eller "vet ej". "Vet ej" ger 0 poäng, ett korrekt sant/falskt-svar ger 1 poäng, och ett felaktigt svar ger -1 poäng (dock aldrig mindre än 0 poäng totalt på uppgiften).

(a) Kontextfria grammatiker kan beskriva alla språk som stackautomater (Push-Down Automaton, PDA) kan beskriva, **och** ytterligare några språk till.

.....

(b) En tvetydig grammatik är inte LL(1).

.....

(c) En delmängd av ett reguljärt språk är alltid reguljärt.

.....