Pràctica 1

Introducció

El llenguatge de programació escollit per a les pràctiques és Mathematica [Wolfram,91]. En les implementacions que es van a dur a terme, cal manejar objectes abstractes com són els autòmats i les gramàtiques, sobre els quals es realitzaran una sèrie de transformacions. Tant uns com les altres poden ser concebuts com a estructures més complexes realitzades sobre la base d'altres més senzilles. L'interès de les pràctiques no consisteix a aconseguir implementacions eficients, sinó a utilitzar un llenguatge que facilite la construcció d'aquests objectes abstractes i el maneig dels mateixos. Tenint en compte tots aquests aspectes, vegem a continuació algunes de les característiques de Mathematica que ho fan adequat per al seu ús en aquestes pràctiques.

Mathematica és un paquet de desenvolupament per a aplicacions de tipus general en què els aspectes de desenvolupament matemàtic, algebraic, numèric, simbòlic i gràfic juguen un paper preponderant. Les aplicacions de Mathematica comprenen pràcticament totes les àrees d'investigació i desenvolupament en investigacions científiques, d'enginyeria, econòmiques, d'arquitectura etc. En el marc d'aquestes àreees es poden utilitzar tant com a eina de treball com una eina docent per a aquelles matèries que comporten una alta càrrega de desenvolupament matemàtic.

Mathematica com a llenguatge de programació es diferencia des llenguatges de propòsit general, d'una banda per la seua habilitat en el tractament d'expressions matemàtiques, nombres, expressions simbòliques, etc., i per una altra en què és un llenguatge interpretat. Com a conseqüència de ser interpretat, un càlcul tarda més temps a executar-se que en un llenguatge compilat, no obstant açò, escriure un programa en Mathematica requereix una fracció del temps necessari per a escriure el mateix programa en un altre llenguatge, i el que és més important en l'opinió de l'equip docent, permet concentrar els esforços en els detalls conceptuals i no en els de implementació.

El sistema Mathematica és interactiu, la qual cosa significa que no cal compilar programes. En lloc d'açò, els càlculs es fan típicament executant les expressions necessàries, i per tant, els resultats intermedis d'aquestes avaluacions es poden vore immediatament. En aquesta primera pràctica es pretén introduir alguns conceptes elementals del llenguatge associat a l'aplicació Mathematica, així com el desenvolupament de programes relacionats amb cadenes i llenguatges formals utilitzant aquesta eina.

Abans de començar

La finestra de Mathematica és totalmente típica. No obstant, té algunes peculiaritats. Les més importants són:

- Els noms són distints si s'escriuen en majúscules o en minúscules.
- Per executar una instrucció cal teclejar **shift+intro**.
- Les instruccions es van numerant de manera automàtica i es disposa en **cel·les**. Cada instrucció du associada una cel·la d'entrada (In[..]) i una altra d'eixida (Out[..]). La seua disposició es pot vore en línies verticals a la dreta.

- Mentre dura l'execució d'una intrucció la línia de la cel·la d'entrada es manté doble. Si s'entra en un bucle indefinit es pot interrompre l'execució teclejant **Alt+.** i si no funciona s'executa **Quit Kernel** del menú **Evaluation**. La primera de les opciones només interromp l'execució en curs, mentre la segona anul·la totes les instruccions las de la sessión actual.

- En tot moment es pot demanar ajuda interactiva sobre la sintaxi d'una ordre escrivint el signe **?** seguit del nom de l'ordre (o part, acabat amb*****).

- Mathematica sagna el text de manera automàtica quan es realitzen els programes. Si el text està mal sagnat acostuma a ser per una sintaxi incorrecta.

Llistes en Mathematica

Una de les estructures de dades més importants en Mathematica (i, sense dubte la que nosaltre usarem més) és la llista. Es poden usar llistes tant per a les paraules dels llenguatges amb els que treballem, com per a les transicions dels autòmats que accepten aquests llenguatges.

Les llistes es representen entre claus i els seus elements estan separats per comes, per exemple: list1 = {a, b, {c, d}} assigna a la variable **list1** la llista amb primer element **a**, segon element **b** i tercer la llista {c, d}. L'element i-éssim d'una llista es referencia com nom[[i]]. Així, list1[[3]] és {c, d}. Observeu que els elements de la llista no cal que siguen homogenis.

Operacions amb llistes

Per estalviar temps començarem contruïnt llistes de manera automàtica. Amb la instrucció `l=Table[Random[Integer, {1, 9}], {i, 20}]` (**shift+intro**) assignem a la variable **l** una llista de 20 nombres enters aleatoris en el rang de 1 a 9 (naturalment es pot variar tant la quantitat com el rang).

Es demana construir dues llistes **l1** e **l2**, i provar les ordres següents:

Important: les ordres que es descriuen a continuació, excepte els que s'indica explícitament, no actualitzen les llistes. Per poder disposar del resultat cal assignar-lo a una variable.

- `Length[l1]`: Torna la longitud de la llista.

- `Join [l1, l2]`: Concatena dues llistes.

- `Union[l1, l2]`: Torna una llista amb els elements que es troben en l1 o l2 i els ordena.

- `Intersection[l1, l2]`: Torna una llista amb els elements comuns a l1 i l2

- `Complement[l1, l2]`: Torna una llista amb els elements de l1 que no estan en l2.

- `Sort[l1]`: Torna l1 ordenada de menor a major (no actualitza l1).

- `Reverse[l1]`: Torna el revers de l1.

- `RotateRight[l1]`: Torna l1 amb els elements desplaçats una posició a la dreta (l'último passa a ser el primer).
- `RotateLeft[l1]`: Idèntic a l'anterior però desplaçant cap a l'esquerra.
- `First[l1]`: Torna el primer element de la llista.
- `Rest[l1]`: Llista l1 sense el primer element.
- `Drop[l1, n]`: Torna la llista sense els primers n elements.
- `Take[l1, n]`: Torna els primers n elements de la llista.
- `Take[l1, {n1,n2}]`: Torna la subllista de $l1$ que conté els elements de la posició $n1$ a la $n2$.
- `Append[l1, x]`: Afegeix l'element x al final.
- `Prepend[l1, x]`: Afegeix l'element x al començament.
- `AppendTo[l1, x]`, `PrependTo[l1, x]`: Idèntiques a les anteriors però actualitzen la llista.
- `Position[l1,x]`: Torna una llista amb les posicions de x en l1.
- `MemberQ[l1,x]`: Torna True si x pertany a l1 i False en cas contrari.

La funció Cases

Per la seua importància, destaquem la funció següent:

`Cases[llista, patró]`: Torna una llista amb els elements de llista que concorden amb patró. El patró pot contenir el símbol `_` (subratllat), que se substitueix per qualsevol símbol.

Exemple `lista={{a,a},{b,a},{b,b},{a,b}}`
`Cases[lista,{a,_}]` torna {{a,a},{a,b}}.
`Cases[lista,{c,_}]` torna {} (llista buida).

Operadors lògics i relacionals

Són operadors que donen com a resultat True o False.
Són els següents:

- Negació `!`
- Conjunció `&&`
- Disjunció `||`
- Igualtat `==`

- No igualdtat `=!=`
- `>, <, >=, <=`.

Programació

Mathematica du incorporat un llenguatge de programació propi que permet incorporar funcions per realitzar tasques específiques al mateix nivell que les funcions predefenides.

Al ser un intèrpret, el mode de treball pot ser totalment interactiu; així, si executem l'expressió `For[i = 1, i < 10, i++, Print[i]]` s'escriuen en pantalla els dígits de l'1 al 9 (utilitzem la instrucció "forçom a exemple per indicar que la sintaxi en Mathematica és molt similar a la d'altres llenguatges de programació habituals).

Mòduls

El concepte de mòdul és totalment similar al de procedure en altres llenguatges de programació. El seu esquema genèric és:

`nom[paràmetres] : = Module [{`variables locals separades per comes`},`
Accions (separades per `;`);
`Return[nvar]` (si el mòdul torna un valor)
`]`

El mòdul, les variables i l'execució del mòdul amb les variables convé que estiguen en cel·les diferents.

Exemple

Mòdul que pren com entrada un nombre positiu n i torna la suma dels n primers nombres enters.

```
suma[n_Integer]:=Module[{i,suma1},
  suma1=0;
  For[i =1,i <= n,i++,suma1 = suma1 + i];
    Return[suma1];
  ]
```

Per a executar-lo s'escriuria per exemple suma[3] (que donarà com a resultat 6).

Estructures condicionals i de repetició

- **Condicional**: `If[condició, sentències-veritat, sentències-fals]`
- **Iteració fixa**: `For[començament, test, increment, sentències]`; S'avalua *començament* i s'executen *sentències* i *increment* fins que *test* falla.
- **Iteració** `While[condició, sentències]`; S'executa *sentències* mentre *condició* és certa.

Representació de cadenes i llenguatges finits

En el que segueix, una cadena es representa como una llista de símbols sobre un determinat alfabet. Així la cadena x=*abbaca*, es representarà com `{a,b,b,a,c,a}`; la cadena buida es representa com la llista de longitud 0, és a dir, `{}`. Un llenguatge finit és un conjunt finit de cadenes. Per tant, un llenguatge es representa com una llista, els elements de la qual (cadenes) són llistes. Per exemple, el llenguatge `L={abba, bb }`, es representarà com `{{a,b,b,a},{b,b}}`; el llenguatge buit es representa com `{}` el llenguatge que només té la cadena buida es representa com `{{}}`.

Exercicis

Exercici 1

Es demana escriure un mòdul Mathematica que amb entrada una cadena x i un símbol a, calcule $|x|_a$ (nombre d'ocurrències de a en x).

Exercici 2

Es demana escriure un mòdul *Mathematica* que amb entrada una cadena x i un enter positiu n, obtinga x^n (concatenació de la paraula x amb sí mateix n voltes).

Exercici 3

Es demana escriure un mòdul *Mathematica* que torne el conjunt de prefixos d'una cadena x.

Exercici 4

Es demana escriure un mòdul *Mathematica* que torne el conjunt de sufixos d'una cadena x.

Exercici 5

Es demana escriure un mòdul *Mathematica* que torne el conjunt de segments d'una cadena x.

Exercici 6

Es demana escriure un mòdul *Mathematica* que torne el producte de dos llenguatges finits donats.

Exemple: Donats $L_1 = \{a, bb, aba\}$ i $L_2 = \{\lambda, ba, bba\}$, el mòdul ha de tornar $\{a, bb, aba, abba, bbba, ababa, bbbba, ababba\}$.

Cal tenir en compte que, en aquestas pràctiques, la representació de les cadenes és mitjançant llistes de símbols i la representació de llenguatges considera llistes de cadenes, per tant, llistes de llistes.

Exercici 7

Es demana escriure un mòdul *Mathematica* que torne la unió de dos llenguatges finits donats.

Exemple: Donats $L_1 = \{a, bb, aba\}$ i $L_2 = \{\lambda, bb, bba\}$, el mòdul ha de tornar $\{\lambda, a, bb, aba, bba\}$.

Exercici 8

Es demana escriure un mòdul *Mathematica* que, amb entrada un llenguatge finit L, i un enter $n > 0$ calcule L^n.

Exemple: Donats $L_1 = \{a, bb, aba\}$ i $n = 2$, el mòdul ha de tornar $\{aa, abb, aaba, bba, bbbb, bbaba, abaa, ababb, abaaba\}$.

Exercici 9

Es demana escriure un mòdul *Mathematica* que, donada una cadena x sobre l'alfabet $\{a, b\}$ com entrada, torne $True$ si x conté un nombre parell de símbols a i almenys dos símbols b. En cas contrari el mòdul tornarà $False$.

Exemple: Donada $x = \{a, b, a, b, b, a, a, a, b, a, b, a, b, b, a, a, b, a, b, a, a, a, a, b, a, a, b, a\}$, el mòdul ha de tornar $False$

Exercici 10

Es demana escriure un mòdul *Mathematica* que, donada una cadena x sobre l'alfabet $\{a, b, c\}$ com entrada, torne $True$ si x conté un nombre parell de símbols b després de l'últim símbol c. En cas contrari el mòdul tornarà $False$.

Exemple: Donada $x_1 = \{a, b, a, c, b, b, a, a, a, b, c\}$, el mòdul ha de tornar $True$. Donada $x_2 = \{a, b, a, c, b, b, a, a, a, b, c, a, a, b, a, a, a\}$, el mòdul ha de tornar $False$. Donada $x_3 = \{a, b, a, b, b, a, a, a, b, a, a, a, a\}$ el mòdul ha de tornar $False$ (x_3 no conté el símbol c).

Exercici 11

Es diu que una cadena s es subcadena de x si s denota una seqüència de símbols que apareixen en x en el mateix ordre encara que no de forma consecutiva.

Es demana escriure un mòdul *Mathematica* que, donades dues cadenes x i s sobre l'alfabet $\{a, b\}$, torne $True$ si x conté la subcadena s. En cas contrari el mòdul tornarà $False$.

Exemple: Donada la cadena $x = \{a, b, a, a, b, a, a, b, a\}$, algunes subcadenes de x són $\{a, b, a\}$, $\{a, a, a\}$, $\{b, b, a\}$ o $\{b, b, b\}$. No obstant, $\{b, b, b, a, a\}$ no és subcadena de x.

Exercici 12

Es demana escriure un mòdul Mathematica que, donades dues cadenes x_n i x_m de longituds n i m, amb $n \leq m$, torne `False` si x_n no és un segment de x_m, o bé, cas que sí ho siga, la posició del primer símbol de x_n en x_m.

Exemple: Donades $x_n = \{b, a, a, b\}$ i $x_m = \{b, a, b, a, b, b, b, a, b, a, a, b, b, b, b, a, b, a, b, b, a, a, b, a\}$, el mòdul ha de tornar 9.

Exercici 13

Es demana escriure un mòdul Mathematica que, donats un conjut de cadenes M i una cadena x, torne el conjunt de posicions de x on es pot trobar una cadena de M com a segment, així com el segment trobat.

Exemple: Donats:

$$M = \{\{b, b\}, \{a, b, b, b\}, \{b, b, a, b\}, \{a, a, a, a\}\}$$

$$x = \{b, a, b, a, a, b, b, a, b, b, b, a, b, b, a, b, a, a, a, a, a, b, b, a, a, b, b, a, b, a\}$$

el mòdul ha de tornar:

$\{\{6, \{b, b\}\}, \{6, \{b, b, a, b\}\}, \{9, \{b, b\}\}, \{10, \{b, b\}\}, \{10, \{b, b, a, b\}\}, \{8, \{a, b, b, b\}\},$
$\{13, \{b, b\}\}, \{13, \{b, b, a, b\}\}, \{17, \{a, a, a, a\}\}, \{18, \{a, a, a, a\}\}, \{22, \{b, b\}\},$
$\{26, \{b, b\}\}, \{26, \{b, b, a, b\}\}\}$