# Introducció a la Programació Lògica

Mateu Villaret i Ausellé

18 d'abril de 2005

**Resum**

La programació lògica és un formalisme de còmput que ens permet d'una banda, utilitzant el llenguatge de la lògica, expressar coneixement, i d'altra banda mitjançant els sistemes d'inferència, manipular-lo i calcular.

Aquests apunts pretenen ser una aproximació a la programació lògica partint de les nocions bàsiques de la lògica com ara, interpretació, model, deducció... arribant a comprovar que demostrar és calcular.

# Índex

# 1 Introducció a la programació lògica

## 1.1 Què és la programació lògica?

La programació lògica es un formalisme de còmput que utilitza dos principis bàsics:

El **llenguatge lògic** per a expressar coneixement

La **inferència** per a manipular-lo

*Un programa lògic* per a resoldre un problema, *esdevé una transcripció dels fets i les regles de funcionament que sabem que són certes en referència a aquell problema i que permeten resoldre'l*, més que no pas un anar guiant a una màquina per quins són els passos que ha d'anar fent sobre un determinant estat per arribar a un estat final que sigui una resposta.

La clau és poder presentar unes quantes assumpcions, que expressarem mitjançant fets i regles, i poder calcular les conclusions mitjançant la inferència (SLD-resolution) sense haver de dir a la màquina com ha de fer aquesta inferència.

En la programació lògica es tracta de preguntar per la certesa d'un objectiu en funció d'una sèrie de regles i fets. Per exemple: podrem preguntar si la llista $[3, 2, 5]$ te alguna llista $X$ que n'és una ordenació, sempre i quant li haguem expressat correctament quines són les regles i fets que ens defineixen la relació: $Y$ *és_una_llista_ordenada_de* $Z$. Així, un **programa** seria un conjunt d'axiomes mentre que, un **còmput** seria una demostració d'una sentència objectiu.

De lògica n'hi ha de diferent expressivitats: lògica de proposicions, de primer ordre (o de predicats), d'ordre superior, ...

1. La lògica de proposicions només ens permet raonar sobre conjunts finits d'elements.

2. La lògica de primer ordre permet expressar propietats sobre conjunts infinits.

3. La lògica d'ordre superior, ens permet raonar també sobre funcions però no permet automatitzar tantes coses com a primer ordre.

**Exemple 1** *Expressivitat de* les lògiques

- *Per exemple, amb lògica proposicional podem fer raonaments del tipus:*

  *Ax 1:* Quan plou els carrers es mullen

  *Ax 2:* Plou

  *Posem per exemple que al fet de ploure el denotem per $P$ i que els carrers es mullin ho notarem per $Q$. La* formalització *dels axiomes quedaria com segueix:*

  $$\begin{array}{ll} \textit{Ax 1:} & P \rightarrow Q \\ \textit{Ax 2:} & P \end{array}$$

*de la qual s'en podria inferir $Q$.*

- *Clarament, amb lògica proposicional no es pot definir, ni raonar sobre, els naturals. Suposant que tenim una constant $0$, un símbol unari $s$, dos símbols binaris $+, \times$ i la igualtat $=$:*

$$
\begin{array}{l}
\ldots \\
\forall x + (x, 0) = x \\
\forall x \forall y + (x, s(y)) = s(+(x, y)) \\
\ldots \\
\forall P, [P(0) \wedge (\forall x(P(x) \rightarrow P(s(x))))] \rightarrow \forall y P(y)
\end{array}
$$

  *Com podem veure, els axiomes de la suma són de primer ordre doncs tenen quantificadors sobre variables que denoten objectes de la teoria, naturals. D'altra banda, l'últim axioma és d'ordre superior; fixem-nos que la quantificació també s'està fent sobre una variable de predicat $P$.*

La programació lògica que nosaltres veurem està basada en primer ordre (ProLog). Aquest llenguatge lògic és més proper al nostre pensament a l'hora de resoldre problemes que no pas l'estil Von Neuman. Si bé la lògica de segon ordre és més expressiva, no disposa, ni pot disposar, de cap sistema deductiu que permeti demostrar tot allò que és cert.

## 1.2 D'on ve?

Hilbert a principis del segle XX, pretenia fonamentar tota la matemàtica a partir dels axiomes de la lògica i de la teoria de conjunts, mitjançant un procediment mecànic. Això va fracassar: Gödel, Turing,... Al 1965 Robinson descriu la resolució i la unificació de primer ordre. Però no és fins als 70s que es construeix el Prolog (Colmerauer). Des de llavors, Prolog ha passat a ser un llenguatge utilitzat sobretot (però no només) en l'àmbit de la IA: llenguatge natural, sistemes experts, resolució de restriccions, etc. i en l'enginyeria del software, es pot fer servir per exemple per al prototipatge.

## 1.3 Avantatges

- Permet expressar coneixement d'una manera explícita independentment de l'arquitectura de la màquina, fent els programes més compactes, flexibles i entenedors.

- Els programes són més fàcils de manipular matemàticament, permetent relacionar-los millor amb els resultats, les especificacions i d'altres programes.

- El coneixement no depèn de com l'usis. Pots canviar d'arquitectura i d'estratègia de còmput. Pots aprendre'l a fer anar sense saber com

és l'ordinador ni quin serà l'ordre d'execució del programa. D'aquesta manera, l'equació:

Programa = lògica + control

esdevé:

Programa = LÒGICA + $_{\text{control}}$

## 1.4 Inconvenients

- Al principi del Prolog no hi havia massa suport per a coses com: tipatge, aritmètica, entrada - sortida, gràfics, manipulació de fitxers... Quan se li va afegir, era un tant farregós.
- La gestió de dades grans (una base de dades) s'ha de representar com un terme...
- Simular un funcionament procedural és molt costós.
- El Prolog no és pur.

## 1.5 Continguts

Introduirem 2 tipus de lògica: la proposicional i la de predicats.

Definirem una lògica com la unió de:

- Sintaxis: on descriurem quins símbols te el llenguatge i de quina manera els puc combinar per a obtenir *fórmules* correctes.
- Semàntica: on definirem les nocions d'*interpretació* (quins possibles conjunts de significats existeixen per als símbols) i de *satisfacció* (quan una interpretació satisfà una fórmula).

A més a més, una lògica pot tenir nocions sintàctiques de *deducció*, que ens permetran inferir conseqüències lògiques a partir de fórmules ja existents. És precisament aquesta noció la que dóna pas a les possibles utilitzacions pràctiques d'aquestes lògiques.

Començarem amb una lògica molt senzilla que ens servirà per introduir unes quantes nocions fonamentals i igualment útils per a la lògica de predicats.

# 2 LÒGICA PROPOSICIONAL

## 2.1 Sintaxis

Sigui $\mathcal{P}$ un conjunt de **símbols de predicat** $(p, q, r, \ldots)$. Una **fórmula** de lògica proposicional es defineix inductivament com:

- tot símbol de predicat és una fórmula
- si $F$ és una fórmula, llavors $\neg(F)$ és una fórmula
- si $F$ i $G$ són fórmules, llavors $(F \wedge G)$ i $(F \vee G)$ són fórmules
- res més és una fórmula

(els parèntesis no són sempre necessaris)

A part de les **connectives** $\neg$, $\wedge$, i $\vee$, sovint se'n fan servir d'altres com: $\rightarrow$, $\leftarrow$, i $\leftrightarrow$. Nosaltres no les hem introduït perquè es poden expressar amb les tres connectives anteriors.

## 2.2 Interpretació

Una **interpretació** $I$ per a un conjunt de símbols de predicat $\mathcal{P}$, és una funció $I : \mathcal{P} \rightarrow \{0, 1\}$, que assigna a cada símbol de predicat un valor de certesa o falsedat. La idea intuïtiva és que cada símbol de predicat representa un enunciat que pot ser cert o fals en certa situació del "món real".

## 2.3 Satisfacció

Sigui $I$ una interpretació i $F$ una fórmula. L'**avaluació** d'una fórmula qualsevol $F$ sota $I$, és una funció denotada $eval_I(F)$ que ens dóna un valor de $\{0, 1\}$. Es defineix com segueix:

- Si $F$ és un símbol de predicat $p$, $eval_I(F) = I(p)$
- $eval_I(F \wedge G) = 1$ sii $eval_I(F) = 1$ i $eval_I(G) = 1$
- $eval_I(F \vee G) = 1$ sii $eval_I(F) = 1$ o $eval_I(G) = 1$
- $eval_I(\neg(F)) = 1$ sii $eval_I(F) = 0$

Direm que $I$ **satisfà** $F$, $I \models F$, sii $eval_I(F) = 1$.

## 2.4 Nomenclatura

Ara podem introduir la següent nomenclatura bàsica que ens servirà també per a la lògica de predicats.

- Una interpretació $I$ és un **model** d'una fórmula $F$ si $I$ satisfà $F$, és a dir, si $I \models F$.
- Una fórmula $F$ és **satisfactible** si te algun model, per tant, si hi ha alguna interpretació que satisfaci $F$.
- Una fórmula $F$ és **insatisfactible** (o és una **contradicció**) si no te cap model, per tant si no hi ha cap interpretació que satisfaci $F$.
- Una fórmula $F$ és una **tautologia** (o es diu que és **vàlida**) si tota interpretació és model de $F$, és a dir, si tota interpretació satisfà $F$.
- Una fórmula $F$ és una **conseqüència lògica** d'una fórmula $G$ si tot model de $G$ també ho és de $F$, és a dir, si per a tota interpretació $I$ tal que $I \models G$ tenim també que $I \models F$. També es pot parlar d'**implicació lògica**. (Això es pot expressar com $G \models F$).
- Una fórmula $F$ és **lògicament equivalent** a una fórmula $G$ si tot model de $G$ també ho és de $F$ i viceversa, es a dir, si per a tota $I$ tenim: $I \models G$ sii $I \models F$.

Aquesta mena de propietats d'una fórmula (o conjunt de fórmules) són les que es plantegen en les diferents aplicacions pràctiques d'una lògica. En lògica proposicional totes aquestes preguntes són *decidibles*. La clau és que la llista de possibles interpretacions per a un conjunt finit de símbols de predicats és finita i que donada una interpretació $I$ i una fórmula $F$ es pot decidir si $I$ satisfà $F$.

Un altre aspecte a notar és la relació entre els "3 graus de veritat". Notem que:

**$F$ és vàlida sii $\neg F$ és insatisfactible**

| **Vàlides** | **Satisfactibles però no vàlides** | **Insatisfactibles** |
|---|---|---|
| $G$ | $F \quad \neg F$ | $\neg G$ |

Sigui $A \models B$ i $B \models A$, i sigui $H$ una fórmula que conté $A$ com a subfórmula. Sigui $H'$ la fórmula obtinguda substituint en $H$ les aparicions de $A$ per $B$, llavors: $H \models H'$ i $H' \models H$.

## 2.5 Taules de veritat

Una **taula de veritat** d'una fórmula determinada, visualitza el valor de veritat de la fórmula per a cadascuna de les possibles interpretacions dels seus símbols de predicats.

| $eval_I(p)$ | $eval_I(q)$ | $eval_I(\neg p)$ | $eval_I(p \wedge q)$ | $eval_I(p \vee q)$ |
|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 |

## 2.6 Equivalències de fórmules més comuns

| | | |
|---|---|---|
| $F \wedge F$ | $\equiv F$ | idempotència de $\wedge$ |
| $F \vee F$ | $\equiv F$ | idempotència de $\vee$ |
| $F \wedge G$ | $\equiv G \wedge F$ | commutativitat de $\wedge$ |
| $F \vee G$ | $\equiv G \vee F$ | commutativitat de $\vee$ |
| $F \wedge (G \wedge H)$ | $\equiv (F \wedge G) \wedge H$ | associativitat de $\wedge$ |
| $F \vee (G \vee H)$ | $\equiv (F \vee G) \vee H$ | associativitat de $\vee$ |
| $(F \wedge G) \vee H$ | $\equiv (F \vee H) \wedge (G \vee H)$ | distributivitat 1 |
| $(F \vee G) \wedge H$ | $\equiv (F \wedge H) \vee (G \wedge H)$ | distributivitat 2 |
| $F \wedge (F \vee G)$ | $\equiv F$ | absorció 1 |
| $F \vee (F \wedge G)$ | $\equiv F$ | absorció 2 |
| $\neg(F \wedge G)$ | $\equiv \neg F \vee \neg G$ | De Morgan 1 |
| $\neg(F \vee G)$ | $\equiv \neg F \wedge \neg G$ | De Morgan 2 |
| $\neg\neg F$ | $\equiv F$ | doble negació |

$A$ és vàlida sii $\forall B(A \wedge B)| = |B$ i $(A \vee B)| = |A$
$A$ és insatisfactible sii $\forall B(A \wedge B)| = |A$ i $(A \vee B)| = |B$

Aquestes equivalències ens permeten simplificar un tant la representació d'algunes fórmules, així, gràcies a l'associativitat del $\vee$ enlloc d'haver d'escriure $(p \vee q) \vee r$ o $p \vee (q \vee r)$ podrem estalviar-nos els parèntesis i escriure indistintament $p \vee q \vee r$.
Per a formalitzar els fets les regles i la presumpta conclusió, usarem el condicional $X \rightarrow Y$ que és lògicament equivalent a: $\neg X \vee Y$.

**Exemple 2** *Exemple de comprovació de conseqüència lògica*
*Si l'Antoni guanya llavors en Baltasar o en Carles són segons.*

$$p \rightarrow (q \vee r)$$

*Si en Baltasar és segon, llavors no guanya l'Antoni.*

$$q \rightarrow \neg p$$

*Si en Dimitri és segon, llavors no ho és en Carles.*

$$s \rightarrow \neg r$$

*L'Antoni ha guanyat.*

$$p$$

*Per tant, en Dimitri no és segon.*

$$\neg s$$

*La qüestió que ens plantegem és:*

$$\{p \rightarrow (q \vee r),\ q \rightarrow \neg p,\ s \rightarrow \neg r,\ p\} \models \neg s$$

*Per a veure si la presumpta conclusió és conseqüència lògica de les premisses, necessitem que tota interpretació que sigui model de les premisses, també ho sigui de la presumpta conclusió. Això ho podem comprovar construint la taula de veritat.*

| $p$ | $q$ | $r$ | $s$ | $p \rightarrow (q \vee r)$ | $q \rightarrow \neg p$ | $s \rightarrow \neg r$ | $\neg s$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | – | – | – | – | – | – | – |

*L'única interpretació que és model de totes les premisses (fila seleccionada), també ho és de la conclusió, per tant, podem concloure que efectivament n'és conseqüència lògica.*

Per saber si una fórmula és satisfactible, vàlida o insatisfactible, només cal construir la taula de veritat i comprovar què passa amb les interpretacions. D'altra banda, per automatitzar això aquest sistema no és massa eficient.

Nosaltres ens centrarem bàsicament en la noció d'implicació lògica i utilitzarem un sistema basat només en la sintaxi que ens serà fàcil d'automatitzar, és més, aquest sistema sintàctic s'avindrà amb la noció de veritat en teoria de models que ens interessa.

## 2.7 Regles d'inferència

Així com la *teoria de models* es basa en les interpretacions per a parlar de conseqüència lògica, des de la perspectiva de la *teoria de la demostració* el que es pretén és transformar *sintàcticament* les fórmules per obtenir-ne d'altres. Més concretament, l'objectiu és disposar d'un mecanisme sintàctic que ens permeti, mitjançant les **regles de deducció (o d'inferència)**, obtenir una fórmula (**conclusió**) a partir d'altres (**premisses**). Quan una fórmula $F_0$ es pot obtenir, mitjançant la regla de deducció R que estiguem fent servir, a partir d'un conjunt de fórmules $\{F_1, \ldots, F_n\}$, ho representarem així[1]:

$$\{F_1, \ldots, F_n\} \vdash_R F_0$$

A les fórmules de l'esquerra les anomenarem **axiomes** i a les fórmules que es poden obtenir a la banda dreta les anomenarem **teoremes**[2].

Algunes de les regles més conegudes i usades són:

- el *modus ponens*:
$$\{A, A \to B\} \vdash_{MP} B$$
- el *modus tollens*:
$$\{\neg A, B \to A\} \vdash_{MT} \neg B$$

Compte amb el popular però incorrecte (no sòlid), "*modus mistakens*":

$$\{B, A \to B\} \vdash_{MM} A$$

amb el qual es fan "argumentacions fraudulentes" com:

*Quan hi ha crisi s'apugen els impostos. Els impostos s'han apujat, per tant hi ha crisi.*

Per tal que una regla d'inferència R tingui "sentit" cal que allò que demostrem a partir dels axiomes sigui una conseqüència lògica d'aquests, és a dir, el que ens interessa es que:

$$\textit{si } \{F_1, \ldots, F_n\} \vdash_R F_0 \textit{ llavors } \{F_1, \ldots, F_n\} \models F_0$$

si això es veritat, direm que la regla és **sòlida**.

Direm que els axiomes juntament amb les regles d'inferència, formen un **sistema d'inferència**. Així, definirem una **demostració** o **derivació**, com a una seqüència de fórmules $\langle s_0, s_1, \ldots, s_n \rangle$ tal que $s_i$ és o bé un axioma o bé una fórmula que es pot obtenir utilitzant alguna regla del sistema d'inferència amb algun subconjunt de fórmules precedents. Els axiomes,

[1] Per extensió també usarem $\vdash_R$, si $R$ és un conjunt de regles o simplement $\vdash$ si no hi ha confusió possible

[2] Cal no confondre els teoremes d'una teoria amb els teoremes lògics

així com els teoremes que se'n poden derivar, formen el que en direm una **teoria**. Si una teoria conté tant la fórmula $F$ com $\neg F$, direm que la teoria és **inconsistent**.

El que ens interessarà d'un sistema d'inferència és que tingui suficient capacitat d'inferència per a poder derivar tot allò que és conseqüència lògica:

$$si\ \{F_1, \ldots, F_n\} \models F_0\ llavors\ \{F_1, \ldots, F_n\} \vdash F_0$$

en aquest cas, direm que el sistema d'inferència és **complet** [3].

## 2.8 Formes normals i clàusules

Un **literal** és o bé un símbol de predicat o bé un símbol de predicat negat: $p$ o $\neg p$. Una fórmula està en **forma normal conjuntiva (CNF)** si és una conjunció de disjuncions de literals:

$$(l_1^1 \vee l_2^1 \vee \ldots \vee l_{k_1}^1) \wedge \ldots \wedge (l_1^n \vee l_2^n \vee \ldots \vee l_{k_n}^n)$$

Una fórmula està en **forma normal disjuntiva (DNF)** si és una disjunció de conjuncions de literals:

$$(l_1^1 \wedge l_2^1 \wedge \ldots \wedge l_{k_1}^1) \vee \ldots \vee (l_1^n \wedge l_2^n \wedge \ldots \wedge l_{k_n}^n)$$

on cada $l_j^i$ és un literal.

Una **clàusula** és una disjunció de literals: $\neg p_1 \vee \ldots \vee \neg p_i \vee p_{i+1} \vee \ldots \vee p_n$ amb $n \geq 0$. De 1 a $i$ parlem de **literals negatius** mentre que la resta, de $i+1$ fins a $n$, són **literals positius**.

Una fórmula en CNF és una *conjunció de clàusules*. Per associativitat, commutativitat i idempotència de $\wedge$, aquesta conjunció de clàusules es pot veure equivalentment com a un *conjunt de clàusules*[4]. Semblantment, una clàusula es pot veure com un *conjunt de literals*. Hi ha una clàusula especial que consisteix en la disjunció de 0 literals i l'anomenarem: **clàusula buida**, representada per: $\square$. És el cas més senzill de fórmula insatisfactible ja que, com sabem, una interpretació $I$ satisfà una clàusula quan $I$ satisfà com a mínim un dels seus literals.

Tota fórmula es pot passar a **forma clausal** seguint els següents passos:

1. passar a fórmula escrita només per $\vee, \wedge, \neg$
2. passar les $\neg$ fins als símbols de predicats
3. aplicar distributivitat

[3] Per descomptat, ens interessa també que el sistema d'inferència es basi en regles sòlides

[4] Tant se val l'ordre i si estan repetides o no

## 2.9 Resolució

Tot seguit presentarem una regla de deducció que treballa amb clàusules. Quan parlem de la clàusula $C$ com a $p \vee D$, volem dir que la clàusula $C$ té un literal $p$ i que $D$ és la resta de disjuncions de literals de $C$.

Mitjançant la regla de deducció anomenada **resolució**, podem inferir a partir d'una clàusula $p \vee C$ i d'una altra $\neg p \vee D$, la nova clàusula $C \vee D$. Típicament es representa així:

$$\frac{p \vee C \qquad \neg p \vee D}{C \vee D} \quad \text{Resolució}$$

Si $S$ és un conjunt de clàusules, denotarem per $Res(S)$ el conjunt de totes les conclusions inferides per resolució amb premisses en $S$. Així, $Res^*(S)$ serà la clausura de $S$ sota resolució:

$$Res^*(S) = \bigcup_i S_i \qquad \text{on} \begin{cases} S_0 = S \\ S_{i+1} = S_i \cup Res(S_i) \end{cases}$$

Una regla de deducció $R$ sobre clàusules, és **refutacionalment completa** si, per a tot conjunt insatisfactible de clàusules $S$, la clàusula buida apareix a la clausura sota $R$ de $S$.

La resolució és refutacionalment completa, per tant:

si $S$ és un conjunt de clàusules insatisfactibles, llavors $\square \in Res^*(S)$

Fixem-nos que per arribar a la clàusula buida ens caldrà tenir un literal sol i el seu negat també sol. D'altra banda, el com seleccionar les clàusules de $S$ per tal d'arribar a $Res^*(S)$, admet força possibilitats.

Un sistema d'inferència basat en resolució és refutacionalment complet, però no és complet. Per exemple no podria deduir teoremes lògics ja que si no te premisses no pot tenir conclusions:

$$\models p \to (q \to p)$$

D'altra banda però, podríem utilitzar la seva completesa refutacional per a veure que aquesta fórmula és vàlida, negant-la i veient que és insatisfactible...

## 2.10 Implicació lògica a partir de la resolució

Atès que la resolució és refutacionalment completa, és fàcil veure que si jo vull comprovar si una fórmula $F$ és conseqüència lògica d'un conjunt d'axiomes $A = \{A_1, \ldots, A_n\}$ només cal que comprovi si $\{A, \neg F\} \vdash_{Res} \square$.

Tenim que:

$$\{A_1, \ldots, A_n\} \models F \text{ sii } \{A_1, \ldots, A_{n-1}\} \models A_n \to F$$

passant totes les premisses a l'altre costat, obtenim:

$$\models A_1 \to (A_2 \ldots \to (A_n \to F) \ldots)$$

i com que: $A \to (B \to C) \equiv (A \wedge B) \to C$, podem escriure:

$$\models (A_1 \wedge A_2 \wedge \ldots \wedge A_n) \to F$$

ara bé, comprovar que la fórmula de dalt és vàlida, és el mateix que comprovar que la fórmula de dalt negada és insatisfactible:

$$(A_1 \wedge A_2 \wedge \ldots \wedge A_n \wedge \neg F) \models \square$$

Per tant, com que la resolució és refutacionalment completa:

$$A \models F \text{ sii } \{A, \neg F\} \vdash_{Res} \square$$

## 2.11 Estratègies de Resolució

La manera més simple i per tant, computacionalment més costosa, d'aplicar la resolució per a trobar la clàusula buida a partir d'un conjunt de fórmules $S$, seria la **resolució sistemàtica**:

$$\begin{array}{lcll} Res^0(S) & = & S & \\ Res^1(S) & = & S \cup & \{ \text{ resolents de cada parell possible de clàusules de } S \} \\ \ldots & & & \\ Res^{n+1}(S) & = & Res^n(S) \cup & \{ \text{ resolents possibles de cada parell de clàusules de } Res^n(S) \} \end{array}$$

Com que $S$ és un conjunt finit de clàusules, té un conjunt finit de símbols de predicat, així doncs arribarà un punt en que $Res^k(S) = Res^{k+1}(S)$, per tant existirà alguna $i \leq k$ tal que $\square \in Res^i(S)$ si i només si $S$ és insatisfactible.

**Exemple 3** *Clausura de la resolució.*
*Sigui* $S = \{ \ p \vee q, \ p \vee \neg q, \ \neg p \vee r, \ \neg r \ \}$.
*És* $S$ *insatisfactible?*

$$\begin{array}{lcl} S_0 & = & S \\ S_1 & = & S_0 \cup \{ \ p, \ \neg p, \ q \vee r, \ \neg q \vee r \ \} \\ S_2 & = & S_1 \cup \{ \ p \vee r, \ q, \ \neg q, \ r, \ \square \} \\ S_3 & = & S_2 \end{array}$$

*Efectivament,* $S$ *és insatisfactible perquè la clàusula buida ja apareix a la segona iteració,* $S_2$.

Com podem veure aquest mètode és de força bruta i és, computacionalment parlant, molt costós: les clàusules poden anar creixent de mida i de nombre, per tant, cada vegada tenim més possibles parelles (fins que saturem el conjunt, és clar). L'objectiu és utilitzar estratègies que ens permetin evitar passos de resolució sense perdre la completesa refutacional.

Unes possibles millores serien:

- Si en una clàusula apareix un literal repetit, s'elimina tantes vegades com calgui fins que aparegui només una vegada.
- Eliminació de clàusules tautològiques: $(p \vee \neg p \vee q)$
- Eliminar clàusules repetides.
- Eliminar clàusules (subsumides) que contenen d'altres clàusules ja existents (en el sentit de teoria de conjunts). Això no afecta la completesa, ja que una clàusula i la fórmula resultant de fer la conjunció d'aquella amb altres que la subsumeixin, són lògicament equivalents. $(p \vee q) \wedge (p \vee q \vee r) \models (p \vee q)$ i $(p \vee q) \models (p \vee q) \wedge (p \vee q \vee r)$

Però aquests punts no tenen perquè millorar espectacularment la cerca de la clàusula buida.

## 2.12 Clàusules de Horn

Una **clàusula de Horn** és una clàusula que te com a màxim un literal positiu (afirmat), és a dir, que és de la forma

$$\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n \vee p$$

amb un únic literal $p$ positiu o

$$\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_n$$

sense cap literal positiu.

Quan en un pas de resolució, una de les premisses és sempre una **clàusula unitària** positiva (amb un sol literal positiu), l'anomenarem **resolució unitària**.

Aquest fet fa que les clàusules resolvents siguin més petites, millorant considerablement l'eficiència. La resolució unitària és refutacionalment completa per a clàusules de Horn.

En programació lògica la noció de clàusules de Horn és molt important, ja que típicament els programes es restringeixen a tres tipus de clàusules de Horn, anomenats:

- Fets: un sol literal positiu
- Regles: una clàusula amb un literal positiu i algun de negatiu
- Queries (preguntes): una clàusula amb només literals negatius.

## Exercicis

1. Quantes interpretacions possibles pot tenir una fórmula en funció de la cardinalitat de $\mathcal{P}$. I quantes avaluacions?
2. Inventeu una nova lògica on les interpretacions són funcions $I :\rightarrow \{0, 1, \perp\}$. $\perp$ denota la incertesa.
3. Demostreu que per a tota fórmula $F$ hi ha almenys una fórmula lògicament equivalent que està en CNF i una altra que està en DNF. (pista: obteniu les fórmules en CNF i en DNF a partir de les taules de veritat per a $F$)
4. Què es pot dir de la fórmula $F'$ resultant de canviar a $F$ una subfórmula $G$ per una d'equivalent $G'$.
5. Descriviu un procediment per a aconseguir, donada una fórmula $F$, una fórmula $F'$ en CNF que sigui lògicament equivalent. (pista: utilitzeu les equivalències de De Morgan i distributivitat)
6. Sigui $F$ una fórmula vàlida, $G$ una fórmula satisfactible (però no vàlida) i $H$ una fórmula insatisfactible. Digueu el que podem saber de les fórmules següents:

$$\neg F \quad \neg G \quad \neg H \quad F \wedge G \quad F \wedge H \quad G \wedge H \quad F \vee G \quad F \vee H \quad G \vee H$$

7. Demostreu que la resolució és correcta.
8. Demostreu que la resolució és refutacionalment completa.
9. Sigui $S$ un conjunt de clàusules. Demostreu que $S$ és satisfactible en cadascun del següents casos:
   (a) Tota clàusula de $S$ té algun literal positiu.
   (b) Tota clàusula de $S$ té algun literal negatiu.
   (c) Per a tot símbol de predicat $p$, es compleix que: o bé $p$ apareix només en literals positius a $S$, o bé apareix només en literals negatius a $S$.
10. Sigui $S$ un conjunt de clàusules de Horn. Demostreu que $S$ és satisfactible si no té cap clàusula que sigui unitària (és a dir, amb un únic literal i positiu).
11. Trobeu un exemple que provi que la resolució unitària no és completa per a clàusules generals.
12. Demostreu per resolució que el raonament de l'exemple 2 és lògicament correcte.

# 3 LÒGICA de PRIMER ORDRE

La lògica proposicional ens permet raonar correctament sobre teories de dominis finits i ens dóna els mecanismes necessaris per a extreure conclusions lògiques a partir d'aquestes. Per exemple, si sabem que si en Joan coneix a la Maria llavors la Maria coneix en Joan ($JCM \rightarrow MCJ$), i també sabem que en Joan coneix a la Maria ($JCM$), podem deduir (usant p.ex. Modus Ponens) que la Maria coneix a en Joan. Ara bé, si volem expressar el coneixement d'una manera més general com per exemple, afirmant que si una persona coneix a una segona llavors aquesta segona coneix a la primera, per a qualsevol dues persones, amb lògica proposicional no en fem prou.

La lògica de primer ordre ens permetrà arribar a aquest grau de representació de coneixement gràcies a un poder expressiu més elevat que ens permet relacionar els elements sobre els que estem definint la teoria.

Repetirem per a la lògica de primer ordre totes les definicions i tots els resultats vistos per a la lògica proposicional. Podrem veure el guany expressiu i la pèrdua de decidibilitat que això produeix.

## 3.1 Sintaxi

Per tal de guanyar poder expressiu, hem d'enriquir el llenguatge, passant a tenir:

- un conjunt de símbols de predicat $\mathcal{P} = p, q, r, \ldots, esparell, home, \ldots$
- un conjunt de símbols de funció $\mathcal{F} = f, g, h, \ldots, a, b, \ldots, successor, ...$
- un conjunt de símbols de variable $\mathcal{X} = X, Y, Z, \ldots$

Cada símbol de funció $f$ i cada símbol de predicat $p$, tenen associada una *aritat* (número d'arguments).

Un **terme** es defineix així:

- tota variable és un terme,
- $f(t_1, \ldots, t_n)$ és un terme si $t_1, \ldots, t_n$, són termes i $f$ és un símbol de funció d'aritat $n$,
- res més és un terme

Si $p$ és un símbol de predicat d'aritat $n$ i $t_1, \ldots, t_n$, són termes, llavors $p(t_1, \ldots, t_n)$ és un **àtom**. Cap altra expressió és un àtom.

Les **fórmules de primer** ordre es defineixen així:

- tot àtom és una fórmula,
- si $F$ i $G$ són fórmules, llavors $\neg(F), (F \wedge G)$ i $(F \vee G)$ són fórmules,

- si $F$ és una fórmula, llavors $\forall X\ F$ i $\exists X\ F$, són fórmules si $X$ apareix lliure (és a dir sense quantificar) a $F$,
- res més és una fórmula.

Els símbols $\forall$ i $\exists$ són els quantificadors **universal** i **existencial** respectivament.

Tant un terme, com una fórmula, si no tenen variables, se'ls anomena "**ground**".

## 3.2 Interpretació

Una **interpretació** $I$ en lògica de primer ordre està formada per:

- Un conjunt no buit $D$, anomenat **domini** de $I$,
- per cada símbol $f^n \in \mathcal{F}$, una funció $f_I : D \times \overset{n}{\ldots} \times D \mapsto D$,
- per cada símbol $p^n \in \mathcal{P}$, una funció $p_I : D \times \overset{n}{\ldots} \times D \mapsto \{0,1\}$.

Així, com a la lògica proposicional, una interpretació ens dona una manera d'interpretar (donar un significat) a cada símbol: ens diu de quin domini poden prendre valor les variables, ens diu com es comporten les funcions en el domini i ens diu quina funció de veritat té cada predicat.

## 3.3 Satisfacció

Sigui $I$ una interpretació. Extenem la noció d'interpretació a variables de manera que per a cada $X \in \mathcal{X}$ hi haurà un $X_I \in D$, és a dir, que les variables s'interpreten com un símbol de funció constant.

Així, l'**avaluació en una interpretació $I$ d'un terme** $t$, denotada per $eval_I(t)$ és una funció que per a cada terme dona un valor del domini $D$:

- si $X \in \mathcal{X}$ llavors $eval_I(X) = X_I$,
- si $f^n \in \mathcal{F}$ llavors $eval_I(f(t_1, \ldots, t_n)) = f_I(eval_I(t_1), \ldots, eval_I(t_n))$

Finalment, l'**avaluació en una interpretació $I$ d'una fórmula** $F$, denotada per $eval_I(F)$ és una funció que per cada fórmula dona un valor de $\{0,1\}$:

- si $p^n \in \mathcal{P}$ llavors $eval_I(p(t_1, \ldots, t_n)) = p_I(eval_I(t_1), \ldots,\ eval_I(t_n))$
- $eval_I(F \wedge G) = 1$ sii $eval_I(F) = 1$ i $eval_I(G) = 1$
- $eval_I(F \vee G) = 1$ sii $eval_I(F) = 1$ o $eval_I(G) = 1$

- $eval_I(\neg(F)) = 1$ sii $eval_I(F) = 0$
- $eval_I(\forall X\ F) = 1$ sii $eval_{I[X=d]}(F) = 1$ per a tot $d \in D$
- $eval_I(\exists X\ F) = 1$ sii $eval_{I[X=d]}(F) = 1$ per algun $d \in D$

on $I[X = d]$ és una interpretació $I'$ idèntica a $I$ excepte que $X_{I'} = d$.
Direm que $I$ **satisfà** $F$, notat per $I \models F$, sii $eval_I(F) = 1$.

Fixem-nos en que si tots els símbols de predicat que tenim són 0-aris, estem en lògica proposicional.

### 3.4 Algunes equivalències més

| | | |
|---|---|---|
| $\neg\forall X.H(X)$ | $\equiv \exists X.\neg H(X)$ | De Morgan 3 |
| $\neg\exists X.H(X)$ | $\equiv \forall X.\neg H(X)$ | De Morgan 4 |
| $(\forall X.F \wedge G)$ | $\equiv \forall X.(F \wedge G)$ | si $X$ no apareix a $G$ |
| $(\forall X.F \wedge \forall X.G)$ | $\equiv \forall X.(F \wedge G)$ | |
| $(\exists X.F \vee \exists X.G)$ | $\equiv \exists X.(F \vee G)$ | |

Compte amb les següents *no-equivalències*:

$$(\forall X.F \vee \forall X.G) \not\equiv \forall X.(F \vee G)$$

$$(\exists X.F \wedge \exists X.G) \not\equiv \exists X.(F \wedge G)$$

### 3.5 Exemples de formalització

Una vegada, coneixem la sintaxi i la noció de satisfacció en aquesta lògica, cal saber conceptualitzar bé "el món" del problema que volem tractar i formalitzar-lo. La conceptualització del món d'un problema no té per què tenir una única manera de fer-se, es tracta de trobar la manera que millor ens vagi per a resoldre el "nostre" problema (per exemple la conceptualització geocèntrica del món que tenia Aristòtil feia molt complicat explicar els moviments dels cossos celestes). El problema de l'adequació de conceptualització per a una correcta formalització no està resolt...

**Exemple 4** *Algunes frases (no necessàriament certes) sobre bolets i possibles formalitzacions ...*

*Tots els bolets púrpures són verinosos*

$$\forall X.(purpura(X) \wedge bolet(X)) \rightarrow verinos(X)$$
$$\forall X.(purpura(X) \rightarrow (bolet(X) \rightarrow verinos(X))$$
$$\forall X.(bolet(X) \rightarrow (pupura(X) \rightarrow verinos(X))$$

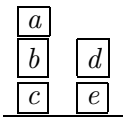
*Un bolet només és verinós si és púrpura*

$$\forall X.(bolet(X) \wedge verinos(X)) \rightarrow purpura(X)$$
$$\forall X.(bolet(X) \rightarrow (verinos(X) \rightarrow purpura(X))$$

*Cap bolet púrpura és verinós*

$$\forall X.\neg(purpura(X) \wedge bolet(X) \wedge verinos(X))$$
$$\neg(\exists X.(purpura(X) \wedge bolet(X) \wedge verinos(X)))$$

**Exemple 5** *Del món dels blocs:*




*Aquest món on tenim blocs 5 blocs amb nom formant dues piles, podem representar-lo mitjançant les interrelacions: sobre*$(\_,\_)$*, per_damunt*$(\_,\_)$*, sobre_taula*$(\_)$ *i dalt_de_tot*$(\_)$*:*

| | | | |
|---|---|---|---|
| $sobre(a,b)$ | $per\_damunt(a,b)$ | $sobre\_taula(c)$ | $dalt\_de\_tot(a)$ |
| $sobre(b,c)$ | $per\_damunt(b,c)$ | $sobre\_taula(e)$ | $dalt\_de\_tot(d)$ |
| $sobre(d,e)$ | $per\_damunt(a,c)$ | | |
| | $per\_damunt(d,e)$ | | |

*De fet podem expressar més fets (regles) a banda d'aquestes sentències simples:*

*si un bloc està a sobre d'un altre, llavors està per damunt*

$$\forall X.\forall Y.(sobre(X,Y) \rightarrow per\_damunt(X,Y))$$

*i també, la relació està per damunt és "transitiva":*

$$\forall X.\forall Y.\forall Z.(damunt(X,Y) \wedge damunt(Y,Z)) \rightarrow damunt(X,Z)$$

*si un bloc està a sobre d'un altre, llavors el primer no està sobre la taula i el segon no està a dalt de tot*

$$\forall X.\forall Y.(sobre(X,Y) \rightarrow (\neg sobre\_taula(X) \wedge \neg dalt\_de\_tot(Y))$$

*Fixem-nos que a partir de les sentències sobre*$(\_,\_)$ *i la regla 1, podem "deduir" totes les sentències: per_damunt*$(\_,\_)$*.*

**Exemple 6** *Sobre les llistes:*

*Suposant que tenim els termes:* $1, 2, 3, \ldots$*, definirem les llistes com a termes formats mitjançant els següents símbols de funció:*

$$\begin{array}{rl} [\,]: & \mapsto terme \\ \cdot: \quad terme \times terme & \mapsto terme \end{array}$$

*així, els termes:* $[\,]$, $1 \cdot 2 \cdot 3 \cdot [\,], 1 \cdot [\,] \cdot [\,] \cdot [\,]$, *són llistes.*
*En funció de com s'hagi format la llista, podem extreure'n informació:*
*Un terme és un element d'una llista?*

$$\forall X.\forall L.member(X, X \cdot L)$$
$$\forall X.\forall Y.\forall L.(member(X, L) \rightarrow member(X, Y \cdot L))$$

*Un terme és la llista resultant d'afegir dues llistes?*

$$\forall L.append([\,], L, L)$$
$$\forall X.\forall L.\forall M.\forall N.append(L, M, N) \rightarrow append(X \cdot L, M, X \cdot N)$$

### 3.6 Conseqüència lògica

Com hem vist anteriorment, en lògica proposicional és possible determinar la satisfactibilitat, la validesa i la implicació lògica senzillament mirant al conjunt de totes les interpretacions per als símbols de predicats que tinguem. Això ens proporciona procediments que sempre acaben degut a què hi ha un conjunt finit de combinacions.

En lògica de primer ordre, també tenim la noció d'interpretació, per tant, també ens podem fer les mateixes preguntes sobre fórmules. Per exemple: *La fórmula* $(\neg p(a) \wedge p(b))$ *és satisfactible?, és vàlida?*

Ens cal trobar una interpretació per a veure si la satisfà o no. Per exemple, aquesta sí:

$$D = \{*, \#\}$$
$$\{a_I = *, b_I = \#\}$$
$$\{p_I(*) = 0, p_I(\#) = 1\}$$

mentre que aquesta no:

$$D = \{*\}$$
$$\{a_I = *, b_I = *\}$$
$$\{p_I(*) = 1\}$$

Plantegem-nos ara el problema de la conseqüència lògica entre fórmules que tenen els següents símbols de predicat i els següents símbols de funció que ens permeten construir un nombre infinit de termes diferents:

$$\mathcal{P} = \{sum(\_, \_, \_)\}$$
$$\mathcal{F} = \{0, s(\_)\}$$

Si volem saber si:

$$sum(s(0), s(s(0)), s(s(s(0))))$$

és conseqüència lògica de:

$$\forall X.sum(X, 0, X)$$
$$\forall X \forall Y \forall Z.sum(X, Y, Z) \rightarrow sum(X, s(Y), s(Z))$$

és a dir:

$$\left\{ \begin{array}{l} \forall X.sum(X, 0, X), \\ (\forall X \forall Y \forall Z.sum(X, Y, Z) \rightarrow sum(X, s(Y), s(Z))) \end{array} \right\}$$
$$\models^{?}$$
$$sum(s(0), s(s(0)), s(s(s(0))))$$

hauríem de comprovar que tot model de les premisses ho és de la conclusió. Com ens ho podem fer si podem tenir infinites interpretacions?

Farem servir el mateix procediment que en lògica proposicional: neguem la conclusió, la passem a l'altra costat i intentem demostrar que és insatisfactible. Però, com podem demostrar que és insatisfactible?

## 3.7 Interpretació d'Herbrand

Nosaltres podem pensar en els dominis que vulguem, en particular, podríem pensar en un domini abstracte que consistiria només en el conjunt de termes diferents que puc fer mitjançant els símbols del llenguatge, l'anomenat **domini (o univers) d'Herbrand**. Per a l'exemple anterior, $D = \{0, s(0), s(s(0)), \dots\}$, és a dir, tots els termes *ground* que puguem construir utilitzant els símbols de $\mathcal{F}$.

Així una **interpretació d'Herbrand** per a la teoria anterior, que a més a més seria un **model** seria:

Per als termes:

$$\{0_I = 0, s_I(0) = s(0), s_I(s(0)) = s(s(0)), \dots\}$$

Per al predicat $sum(\_, \_, \_)$:

$$\begin{array}{ll} \{ \quad sum_I(0,0,0) = 1, & sum_I(0,0,s(0)) = 0 \\ \quad sum_I(s(0),0,0) = 0, & sum_I(s(0),0,s(0)) = 1 \\ \quad sum_I(0,s(0),0) = 0, & sum_I(0,s(0),s(0)) = 1 \\ \quad \dots & \} \end{array}$$

## 3.8 Procediments d'Herbrand

Un teorema molt important ens permet assegurar que:

| tot conjunt de clàusules és satisfactible<br>si i només si<br>te un model d'Herbrand |
|---|

Això, juntament amb el fet que disposem d'un algoritme que ens permet traduir tota fórmula $F$ a un conjunt de clàusules $S$ tal que $S$ és satisfactible si i només si $F$ ho és, ens permetrà utilitzar la resolució en lògica de predicats com a procediment sòlid i refutacionalment complet (encara que ara no tenim garantit l'acabament).

Ens interessa portar les nostres clàusules a la lògica proposicional on tot és més fàcil i ja ho coneixem. Per fer això és crucial saber que qualsevol model per a un conjunt de clàusules sobre algun domini del món real, necessàriament té un model d'Herbrand isomorf (idèntic en quant a estructura).

Per exemple, disposem dels següents símbols de funció i de predicats:

$$\begin{aligned} \mathcal{F} &= \{joan, logica, maria\} \\ \mathcal{P} &= \{li\_agrada(\_, \_)\} \end{aligned}$$

i tenim aquestes premisses:

$$\begin{aligned} &\forall X. li\_agrada(X, logica) \rightarrow li\_agrada(joan, X) \\ &li\_agrada(maria, logica) \end{aligned}$$

les fórmules ground que puc tenir són:

$$\begin{aligned} &li\_agrada(joan, logica) \rightarrow li\_agrada(joan, joan) \\ &li\_agrada(maria, logica) \rightarrow li\_agrada(joan, maria) \\ &li\_agrada(logica, logica) \rightarrow li\_agrada(joan, logica) \\ &li\_agrada(maria, logica) \end{aligned}$$

si ens hi fixem bé, podem veure que de fet el que tenim és expressable en lògica proposicional.

| | Forma clausal |
|---|---|
| $AJL \rightarrow AJJ$ | $\neg AJL \vee AJJ$ |
| $AML \rightarrow AJM$ | $\neg AML \vee AJM$ |
| $ALL \rightarrow AJL$ | $\neg ALL \vee AJL$ |
| $AML$ | $AML$ |

Podem demostrar que $AJM$ n'és una conseqüència lògica?
Als principis de la demostració automàtica es feien les demostracions de primer ordre seguint aquests passos:

1. Substituir les clàusules donades per la seva instanciació ground
2. Trobar la clàusula buida

Al darrera d'aquest procediment hi ha el teorema d'Herbrand, que es pot presentar de moltes formes però aquesta ens ve bé:

Un conjunt de clàusules $P \cup \{\neg A\}$ és insatisfactible
si i només si
algun subconjunt finit de la instanciació ground
de $P \cup \{\neg A\}$ és insatisfactible

Aquesta *finitud* necessària, ens permet pensar en que en un temps finit, si anem provant tots els possibles subconjunts, haurem de trobar-ne un d'insatisfactible si és que ho és el conjunt sencer. Hauríem de ser capaços d'ordenar:

$$Ground(P \cup \{\neg A\}) = \{g_1, g_2, \ldots\}$$

Necessàriament, si $P \models A$, ha d'existir una $k$ tal que de $\{g_1, g_2, \ldots, g_k\}$ sigui insatisfactible. Aquest test, per exemple, es podria fer amb taules de veritat. Evidentment, tota aquesta càrrega computacional no es duu a terme sinó que s'utilitza resolució. La resolució de primer ordre és sòlida i refutacionalment completa, per tant podem "traduir" el teorema d'Herbrand com segueix:

D'un conjunt de clàusules $P \cup \{\neg A\}$, en podem derivar $\square$
si i només si
se'n pot derivar $\square$ d'algun subconjunt finit de
la instanciació ground de $P \cup \{\neg A\}$

Per a poder aplicar resolució, necessitarem passar a forma normal conjuntiva.

### 3.9 Pas a CNF

Tota fórmula $F$ pot ser transformada en un conjunt (conjunció) de clàusules $S$ preservant la satisfactibilitat[5]:

- $F$ és satisfactible si i només si ho és $S$
- $S \models F$

Seguirem els següents passos:

1. eliminació dels símbols condicionals com: $\rightarrow$.
2. moure les negacions cap a dins, mitjançant:
   - $\neg(F \wedge G) \Longrightarrow (\neg F) \vee (\neg G)$
   - $\neg(F \vee G) \Longrightarrow (\neg F) \wedge (\neg G)$
   - $\neg\neg F \Longrightarrow F$

[5]Noteu que diem *preservant la satisfactibilitat*, ja que no s'obté necessàriament un conjunt de clàuluses lògicament equivalent

- $\neg\exists X.F \Longrightarrow \forall X.\neg F$
- $\neg\forall X.F \Longrightarrow \exists X.\neg F$

3. Eliminació de quantificadors existencials o **Skolemització**: es substitueix cada ocurrència de $\exists X$ que es troba en l'àmbit de les variables $Y_1, \ldots, Y_n$, quantificades universalment, per $f_X(Y_1, \ldots, Y_n)$, on $f_X$ és un símbol de funció nou.

   Per exemple:

   $$\forall X.(\neg huma(X) \vee \exists Y.es\_mare(Y, X))$$

   es traduiria a:

   $$\forall X.(\neg huma(X) \vee es\_mare(f_Y(X), X))$$

   on intuïtivament, $f_X$ denotaria la funció, "mare de".

   Aquest pas, si s'aplica, és el que pot fer perdre l'equivalència lògica. No podem saber a partir d'un existencial, quin element és el que satisfà la propietat.

4. Moviment de quantificadors universals cap enfora: els conflictes de nom en les variables (diferents variables que es diuen igual) s'eliminen mitjançant reanomenament; després es col·loquen els quantificadors al principi de la fórmula.

   Per exemple:

   $$(\forall X.P(X)) \vee (\forall X.Q(X))$$

   es traduiria primer a:

   $$(\forall X.P(X)) \vee (\forall Y.Q(Y))$$

   i finalment a:

   $$\forall X \forall Y.(P(X) \vee Q(Y))$$

5. Distribució de $\wedge$ sobre $\vee$, aplicant (exhaustivament) les regles:

   - $(F \wedge G) \vee H \Longrightarrow (F \vee H) \wedge (G \vee H)$
   - $F \vee (G \wedge H) \Longrightarrow (F \vee G) \wedge (F \vee H)$

Així, finalment, tindrem una expressió d'aquesta forma:

$$\forall X_1 \ldots \forall X_m (l_1^1 \vee l_2^1 \vee \ldots \vee l_{k_1}^1) \wedge \ldots \wedge (l_1^n \vee l_2^n \vee \ldots \vee l_{k_n}^n)$$

on cada $l_i^j$ és un literal (de primer ordre). Aquesta expressió pot ser vista com en conjunt (la conjunció) de $n$ clàusules:

$$\begin{array}{ll} \{ & l_1^1 \vee l_2^1 \vee \ldots \vee l_{k_1}^1 \\ & \ldots \\ & l_1^n \vee l_2^n \vee \ldots \vee l_{k_n}^n \quad \} \end{array}$$

on totes les variables estan universalment quantificades. El que definíem per a les clàusules en lògica proposicional, serveix també per aquí: clàusula buida, clàusules de Horn, ...

Per a poder aplicar resolució necessitàvem que dues clàusules compartissin un literal igual, l'una en positiu i l'altre en negatiu. Per aconseguir això en primer ordre, ens pot caler, per exemple, fer iguals dos termes amb variables.

## 3.10 Unificació

Una **substitució** és un conjunt de parells $\{X_1 = t_1, \ldots, X_n = t_n\}$, on les $X_i$'s són variables i els $t_i$'s són termes, i on $X_i \neq X_j$ per a $i \neq j$. **Aplicació d'una substitució** és: donada una substitució $\sigma = \{X_1 = t_1, \ldots, X_n = t_n\}$, i un àtom (o terme o clàusula) $A$, l'àtom $A\sigma$ (o $\sigma(A)$) és el resultat de substituir simultàniament cada variable $X_i$ en $A$ pel corresponent terme $t_i$.Per exemple:

$$f(X, h(X, Y), Z)\{X = g(a), Y = g(X), Z = b\} = f(g(a), h(g(a), g(X)), b)$$

Dos àtoms $A$ i $A'$ són **unificables** si existeix una substitució $\sigma$ tal que $A\sigma = A'\sigma$, és a dir, si existeixen valors per a les seves variables que els facin iguals, cas en el que $\sigma$ és anomenat **unificador**.
p.ex, el problema d'unificació:

$$A = f(X, h(X, a), Y) \doteq f(Z, Y, W) = A'$$

té solució:

$$\begin{aligned} \sigma &= \{X = c, Y = h(c, a), Z = c, W = h(c, a)\} \\ A\sigma &= f(\underline{c}, h(\underline{c}, a), \underline{h(c, a)}) = f(\underline{c}, \underline{h(c, a)}, \underline{h(c, a)}) = A'\sigma \end{aligned}$$

L'**unificador més general**: $\sigma = mgu(A, A')$, és únic (llevat de reanomenament de variables) i és tal que per a qualsevol altre unificador $\sigma'$ de $A$ i $A'$, podem trobar una substitució $\rho$ tal que $A\sigma' = (A\sigma)\rho$. Per a l'exemple anterior, el m.g.u. seria:

$$\rho = \{Y = h(X, a), Z = X, W = h(X, a)\}$$

$$A\rho = f(X, h(\underline{X}, a), \underline{h(X, a)}) = f(\underline{X}, \underline{h(X, a)}, \underline{h(X, a)}) = A'\rho$$

i $A\sigma$ es podria expressar com:

$$(A\rho)\{X = c\} = f(X, h(X, a), h(X, a))\{X = c\} = f(\underline{c}, h(\underline{c}, a), h(\underline{c}, a))$$

Un mètode habitual d'expressar algoritmes d'unificació és, en lloc de treballar sobre una sola equació $s \doteq t$, on es busca $mgu(s, t)$, treballar sobre un conjunt de parells de termes a unificar: $\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ per als que es busca el $mgu$ simultani, és a dir, que $s_i\sigma = t_i$ per a tot $i$ de 1 a $n$. Llavors es pot definir el següent conjunt de **regles de transformació**:

1. $P \cup \{t \doteq t\} \Longrightarrow P$
2. $P \cup \{f(t_1, \ldots, t_n) \doteq f(s_1, \ldots, s_n)\} \Longrightarrow P \cup \{t_1 \doteq s_1, \ldots, t_n \doteq s_n\}$
3. $P \cup \{f(t_1, \ldots, t_n) \doteq g(s_1, \ldots, s_n)\} \Longrightarrow$ FALLADA (si $f \neq g$)
4. $P \cup \{X \doteq t\} \Longrightarrow P\{X = t\} \cup \{X \doteq t\}$ si $X \in vars(P)$ i $X \notin vars(t)$
5. $P \cup \{X \doteq t\} \Longrightarrow$ FALLADA (si $X \in vars(t)$ i $X \neq t$ )

Finalment, quan ens trobem amb una equació del tipus $X \doteq Y$ tal que $X$ o $Y$ no apareixen a la resta d'equacions $P$, diem que l'equació ja està resolta i no aplicarem 4.

Així, veient per a cada parella quina "forma" te, podem "escollir" quina regla apliquem. Aquestes regles, transformen qualsevol problema d'unificació en un altre equivalent. Qualsevol problema al que no es puguin aplicar cap regla és un unificador més general de si mateix. El quina regla aplicar en cada moment defineix l'estratègia d'unificació. Qualsevol estratègia d'aplicació de les regles acaba. En funció de l'estratègia que tinguem, aconseguirem un algoritme d'unificació més o menys eficient, des d'un ordre lineal fins a un exponencial.

**Exemple 7** *Problema d'unificació:*

$$\{f(X, h(X, a), Y) \doteq f(Z, Y, W)\}$$

*per aplicació de* 2*,*

$$\{X \doteq Z, h(X, a) \doteq Y, Y \doteq W\}$$

*prenent* $P = \{X \doteq Z, Y \doteq W\}$ *apliquem* 4 *a* $h(X, a) \doteq Y$ *i obtenim:*

$$\{X \doteq Z, h(X, a) \doteq Y, h(X, a) \doteq W\}$$

*prenent* $P = \{h(X, a) \doteq Y, h(X, a) \doteq W\}$ *apliquem* 4 *a* $X \doteq Z$ *i obtenim:*

$$\{X \doteq Z, h(Z, a) \doteq W, h(Z, a) \doteq Y\}$$

*no cal que apliquem* 4 *a* $X \doteq Z$ *perquè seria només un "reanomenament de variables".*

### 3.11 Resolució i factorització

Ara ja estem en disposició de presentar les dues regles que ens serviran per a definir un càlcul de demostració per a clàusules de primer ordre, sòlid i refutacionalment complet:

1. **Resolució**. Siguin $A$ i $A'$ dos àtoms, i $C$ i $D$ dues clàusules

$$\frac{A \vee C \qquad \neg A' \vee D}{C\sigma \vee D\sigma} \text{ on } \sigma = mgu(A, A')$$

2. **Factorització**.Siguin $A$ i $A'$ dos àtoms i $C$ una clàusula

$$\frac{A \vee A' \vee C}{A\sigma \vee C\sigma} \text{ on } \sigma = mgu(A, A')$$

Així, per a demostrar que $F \models T$ només cal passar $F \wedge \neg T$ a forma clausal $(S)$ i trobar la clàusula buida mitjançant resolució i factorització. Per tant, també podríem definir un $ResFact^*(S)$ com ho hem fet per a la lògica proposicional:

$$\text{si } F \models T \text{ llavors } \square \in ResFact^*(S)$$

Cal tenir en compte, que cada vegada que agafem una clàusula del conjunt inicial per a fer un pas de resolució, reanomenarem les variables que tingui, amb noms de variable no utilitzats.

**Exemple 8** *Exemple de resolució*
*Així, ara podem demostrar que*

$$\begin{array}{l} \forall X.sum(X, 0, X) \\ (\forall X \forall Y \forall Z.sum(X, Y, Z) \rightarrow sum(X, s(Y), s(Z))) \\ \models^{?} \\ sum(s(0), s(s(0)), s(s(s(0)))) \end{array}$$

*Ara hem de passar a CNF. La transformació de la primera fórmula, és només l'eliminació dels universals:*

$$sum(X, 0, X)$$

*La transformació de la segona consisteix en primer eliminar el condicional:*

$$(\forall X \forall Y \forall Z. \neg sum(X, Y, Z) \vee sum(X, s(Y), s(Z))$$

*i després l'eliminació d'universals:*

$$\neg sum(X, Y, Z) \vee sum(X, s(Y), s(Z)))$$

*Així, el conjunt de clàusules a demostrar insatisfactible és:*

$$\begin{array}{ll} \{ & (1)\ sum(X, 0, X), \\ & (2)\ \neg sum(X, Y, Z) \vee sum(X, s(Y), s(Z)), \\ & (3)\ \neg sum(s(0), s(s(0)), s(s(s(0)))) \qquad \} \end{array}$$

*de* (2) *reanomenada* $(\neg sum(X_0, Y_0, Z_0) \vee sum(X_0, s(Y_0), s(Z_0)))$ *i* 3 *aplicant resolució i amb* $mgu(2, 3) = \{X_0 = s(0), Y_0 = s(0), Z_0 = s(s(0))\}$ *obtenim:*

$$(4)\ \neg sum(s(0), s(0), s(s(0)))$$

*de* (4) *i* (2) *reanomenada* $(\neg sum(X_1, Y_1, Z_1) \quad \vee sum(X_1, s(Y_1), s(Z_1)))$ *aplicant resolució i amb* $mgu(4, 3) = \{X_1 = s(0), Y_1 = 0, Z_1 = s(0)\}$ *obtenim:*

$$(5)\ \neg sum(s(0), 0, s(0))$$

*de (5) i (1) reanomenada* $(sum(X_2, 0, X_2))$ *obtindrem, aplicant resolució amb* $mgu(1, 5) = \{X_2 = s(0)\}$*:*

$$(6)\ \square$$

Imaginem-nos però, que tenim aquestes clàusules:

$$\{\ p(a), \neg p(X) \vee p(f(X))\ \}$$

i en volem derivar la clàusula buida mitjançant resolució. Com podem veure no acabaríem mai i és que les clàusules són satisfactibles.

## 3.12 Càlculs i respostes

De la mateixa manera que hem fet servir la resolució per a "validar" fórmules en una teoria (una suma en l'exemple), la podem fer servir per a calcular (en el nostre exemple calcular sumes).

$$\begin{array}{ll} \{ & (1)\ sum(X, 0, X), \\ & (2)\ \neg sum(X, Y, Z) \vee sum(X, s(Y), s(Z)), \\ & (3)\ \neg sum(s(0), s(s(0)), R) \vee resultat(R) \quad \} \end{array}$$

de (2) reanomenada $(\neg sum(X_0, Y_0, Z_0) \vee sum(X_0, s(Y_0), s(Z_0)))$ i (3) aplicant resolució i amb $mgu(2, 3) = \{X_0 = s(0), Y_0 = s(0), R = s(Z_0)\}$ obtenim:

$$(4)\ \neg sum(s(0), s(0), Z_0) \vee resultat(s(Z_0))$$

de (4) i (2) reanomenada ($\neg sum(X_1, Y_1, Z_1) \vee sum(X_1, s(Y_1), s(Z_1))$) aplicant resolució i amb $mgu(4, 3) = \{X_1 = s(0), Y_1 = 0, Z_0 = s(Z_1)\}$ obtenim:

$$(5)\ \neg sum(s(0), 0, Z_1) \vee resultat(s(s(Z_1)))$$

de (5) i (1) reanomenada ($sum(X_2, 0, X_2)$) aplicant resolució i amb $mgu(1, 5)$ $= \{X_2 = s(0), Z_1 = s(0)\}$ obtenim:

$$(6)\ resultat(s(s(s(0))))$$

Així hem vist com podem demanar per valors de variables tals que facin la teoria insatisfactible. Així hem passat de demostrar a calcular.

## 3.13 Linear resolution for Definite clauses with Selection-rule

Com podem veure, en els exemples hem aplicat els passos de resolució necessaris per a poder arribar a la solució, tot i que és fàcil notar que podem fer molts més passos de resolució. Per tant, com saber quins són els necessaris?

D'altra banda, si ens féssim preguntes com:

$$\neg sum(S1, S2, s(s(s(0)))) \vee resultat(S1, S2)$$

ens pot resultar interessant, poder trobar tots els possibles sumands, per tant necessitem poder trobar tots els camins fins a aquests.

La **resolució SLD** ens proporciona un mètode eficient i complet per a la refutació, per a clàusules de Horn. (Definite clauses)

Suposant que:

- d'una banda tenim un conjunt de clàusules de Horn que formen un **programa lògic**, a partir de **regles** (clàusules amb un literal positiu i algun de negatiu) i de **fets** (clàusules amb un sol literal i positiu)

$$P = \{A_1^1 \vee \neg A_1^2 \ldots \neg A_1^{n_1},\ \ldots\ , A_m^1 \vee \neg A_m^2 \ldots \neg A_m^{n_m}\}$$

- i d'altra banda en volem deduir un altre anomenat **objectiu** (clàusules que no tindran cap literal positiu)

$$G = \{\neg G_1 \vee \ldots \vee \neg G_k\}$$

la SLD resolució, consisteix en fer resolució seguint els següents passos: agafem la clàusula objectiu i escollim un àtom qualsevol a resoldre, per exemple $G_i$ i apliquem resolució amb la clàusula de $P$ que puguem, per exemple $A_k^1 \vee \neg A_k^2 \ldots \neg A_k^{n_k}$:

$$\frac{\neg G_1 \vee \ldots \vee \neg G_k \qquad A_k^1 \vee \neg A_k^2 \ldots \neg A_k^{n_k}}{(\neg G_1 \vee \ldots \vee \neg G_{i-1} \vee \neg A_k^2 \ldots \neg A_k^{n_k} \vee \neg G_{i+1} \vee \ldots \vee \neg G_k)\sigma}$$

on $\sigma = mgu(\neg G_i, A_k^1)$

Ara la clàusula resultant passa a ser l'objectiu i per a fer resolució haurem d'utilitzar una clàusula del programa. Així successivament fins a fallar o arribar a la clàusula buida.

El Prolog utilitza aquesta tècnica però té definit quin és el literal de l'objectiu que ha "d'atacar" cada vegada i quina regla ha d'utilitzar si pot.

1. Sempre el literal més a l'esquerra de l'objectiu.

2. Sempre la primera regla aplicable. (pèrdua de completesa)

Tot i ser completa la SLD-resolució per a clàusules de Horn, la seva implementació sobre ProLog no ho és ja que és determinista. El fet és que en explorar exhaustivament una regla, si l'espai de cerca d'aquesta és infinit, no encertarà a trobar la solució.

**Exemple 9** *Programa que no acaba en Prolog:*

$$\begin{array}{l} p(X) :- p(X). \\ p(X) :- q(X). \\ q(a). \\ ?\ p(Y). \end{array}$$

Això ens permet parlar de l'**arbre de resolució** i dels seus recorreguts en profunditat o en amplada. L'arrel seria l'objectiu, i cada branca, seria cada un dels resolent amb totes les possibles clàusules del programa. Una estratègia que ens permetés recórrer l'arbre amb amplada seria completa.

Com podrem veure, el ProLog acaba sacrificant certa "puresa" lògica en pro de l'eficiència. Així caldrà anar amb compte amb aspectes extra-lògics com ara l'ordre de les clàusules, el control de l'occur-check, l'assumpció de món tancat, ...

Per a veure les nocions bàsiques del llenguatge ProLog en en nostre entorn de GNUProLog, disposeu d'unes transparències a la meva plana web http://ima.udg.es/∼villaret i podeu consultar la bibliografia recomanada.

# 4 Exàmens d'anys anteriors

Exàmen parcial de **ProDe**

Nom i Cognoms:

1. Un professor despistat, ha intentat definir la suma de la següent manera:

$$sum(0, X, X)$$
$$\neg sum(X, Y, Z) \vee sum(X, s(Y), s(Z))$$

   - Perquè diem que s'ha despistat?
   - Troba dues maneres d'arreglar l'errada mitjançant un fet nou i mitjançant una regla nova.
   - Demostra que els nous conjunts de clàusules són satisfactibles.
   - Com definiries la resta? Comprova que

$$resta(s(s(0)), s(0), s(0))$$

2. Sigui $S$ un conjunt de clàusules de lògica proposicional. Explica perquè la clausura de $S$ sota resolució és un conjunt finit de clàusules. Demostra que això mateix és fals si $S$ és de primer ordre.

3. Sigui $F$ la fórmula $\forall X.P(X) \vee Q(X)$ i la fórmula $G : (\exists X.P(X)) \vee (\forall X.Q(X))$. Demostra per resolució que $G$ és conseqüència lògica de $F$.

4. Sigui $F$ la fórmula $\forall X.\exists Y.P(X, Y)$ i $G$, la fórmula $(\exists Y.\forall X P(X, Y))$.

   - És $F$ conseqüència lògica de $G$? Demostra-ho
   - És $G$ conseqüència lògica de $F$? Demostra-ho

Exàmen parcial de **ProDe**

Nom i Cognoms:

1. Resoleu utilitzant les regles d'unificació, el següent problema d'unificació:

$$f(f(f(f(Z_0, X_4), X_3), X_2), X_1) \doteq f(X_1, f(X_2, f(X_3, f(X_4, Z_1))))$$

   És un m.g.u.? Si ho és doneu-ne un que no ho sigui. Si no ho és trobeu-lo.

   Canviant només una variable obteniu un problem d'unificació on es produeixi un error d'occur-check.

2. Donada la següent teoria $T$ sobre llistes:

$$\forall L.append([\,], L, L)$$
$$\forall X.\forall L.\forall M.\forall N.append(L, M, N) \rightarrow append(X \cdot L, M, X \cdot N)$$

   demostreu per SLD-resolució que la fórmula $F$: $append(1 \cdot 2 \cdot [\,], 3 \cdot 4 \cdot [\,], 1 \cdot 2 \cdot 3 \cdot 4 \cdot [\,])$ nés conseqüència lògica.

   és $F \wedge T$ satisfactible? doneu-ne un model o demostreu que no ho és.

   és $F \wedge \neg T$ satisfactible? doneu-ne un model o demostreu que no ho és.

3. Sigui $S_1$ un conjunt de clàusules tals que cada clàusula només té literals positius o negatius. Que en podeu dir de la seva satisfactibilitat?

   Sigui $S_2$ un conjunt de clàusules unitaries (és a dir, amb un sol literal). Que en podeu dir de la seva satisfactibilitat?

   Sigui $S_3 = \{\forall X p(X) \wedge \neg p(X)\}$ que en podeu dir de la seva satisfactibilitat? Demostreu-ho.

# 5 Enunciats d'exercicis ProLog

1. Escriviu un predicat $fact(N, F)$ que significa: $F$ és el factorial del número natural $N$, per als casos:

   (a) quan ens donen instanciat un número natural $N$ i volem generar-ne el factorial $F$. Feu-ho de manera que funcioni correctament si es demanen més respostes, és a dir que falli.

   (b) quan ambdues variables no estan instanciades i vulguem generar tots els parells $N$, $F$.

   (c) quan el que ens ve instanciat és el factorial i vulguem calcular-ne la $N$.

   (d) quan tant $N$ com $F$ vinguin instanciats i vulguem respostes si/no.

   Finalment, feu-ne un que ens permeti tractar tots els casos d'abans. Utilitzeu $var(X)$ i $nonvar(X)$.

2. Escriviu un predicat $mcm(X, Y, M)$ que significa: $M$ és el mínim comú múltiple de $X$ i de $Y$. Ha de ser capaç de contestar correctament els següents casos:

   (a) quan ens venen instanciats els tres paràmetres, ha de dir si $M$ és o no el mcm de $X$ i de $Y$.

   (b) quan només ens donen instanciats $X$ i $Y$, ha de calcular-nos la $M$ correctament.

3. Escriviu un predicat $prod(L, P)$ que significa: $P$ és el producte dels elements de la llista d'enters $L$. Ha de poder tant generar $P$ com comprovar una $P$ donada.

4. Definir el predicat: $divisors(N, L)$ tal que donat un natural $N$, $L$ serà la llista de divisors de $N$ en ordre creixent. Així, si $N$ és 10, $L$ serà: $[1, 2, 5, 10]$. Ha de respondre $YES$ si una $L$ donada és la llista creixent dels divisors de un $N$ donat i $NO$ en cas contrari. També ha de poder general la llista quan li donen ja instanciada la $N$.

5. Definiu un predicat prolog $accepta(L)$, que és satisfà quan $L$ és una llista de la forma $a^n b^n c^n$ per a qualsevol $n > 0$. Permeteu també que la $L$ no vingui instanciada i que respongui sota demanda totes les paraules del llenguatge.

6. Definiu un predicat prolog $apareix2(L1, L2)$ que es satisfà quan tots els elements de $L1$ apareixen dues vegades seguides a $L2$.

7. Definir el predicat $take(L1, L2, N)$ tal que donats una llista $L1$ i un número $N$, $L2$ és la llista que conté els elements de $L1$ agrupats de $N$ en $N$. És a dir, $L2$ és la llista de llistes de $N$ elements cadascuna (menys potser la última) tal que al concatenar-les totes, obtenim la llista $L1$. Si donem com a dades $N$ i $L2$, $take(L1, L2, N)$ ha de tornar-nos $L1$. Semblantment, $take(L1, L2, N)$ ha de funcionar li les tres variables $L1, L2$ i $N$ estan instanciades.

   Si $L1$ és $[1, 2, 3, 4, 5]$ i $N$ és 2, llavors $take(L1, L2, N)$ ha de respondre $L2 = [[1, 2], [3, 4], [5]]$.

   Si $N$ és 2 i $L2$ és $[[1, 2], [3, 4], [5]]$ llavors $take(L1, L2, N)$ ha de respondre $L1 = [1, 2, 3, 4, 5]$.

   Si $L1$ és $[1, 2, 3, 4, 5]$ i $N$ és 2, llavors la resposta ha de ser $NO$.

8. Donant simplement les regles de derivació de la forma $d(E, X, C)$ podem definir un derivador simbòlic d'expressions matemàtiques. Per exemple, la regla de la derivada de la suma es definiria dient que: l-la derivada de $U + V$ amb respecte la variable $X$, és la suma de les derivades de $U$ i de $Y$":

$$d(U + V, X, U1 + V1) : -d(U, X, U1), d(V, X, V1)$$

   feu això per el major nombre de casos possibles: exponenciació, logaritmes, etc. Si cal, definiu-vos operadors nous, p. ex.

$$op(10, xfy, \hat{}).$$

   ens definiria un operador $\hat{}$ amb prioritat 10 i associativitat dreta, és a dir: $a \hat{} b \hat{} c = a \hat{} (b \hat{} c)$. Definir regles de simplificació el més general possibles per a les expressions que tornarà el derivador. P ex:

$$simplifica(X + 0, X).$$

$$simplifica(0 + X, X).$$

$$\ldots$$

9. Definir el predicat $ocur1(E, X)$ tal que donat un terme $E$ i una variable (sense instanciar) $X$, ens respongui afirmativament si la variable $X$ apareix al terme $E$ i no en cas contrari. Així,

$$ocur1(f(a, f(X, b), X) \Longrightarrow YES$$

$$ocur1(f(g(X), Y), Z) \Longrightarrow NO$$

...

Definir el predicat $cambiar\_totes(V, Subst, E1, E2)$ tal que donats $V$, $Subst$ i $E1, E2$ és el terme $E1$ on s'han substituït totes les aparicions de l'àtom $V$ pel terme $Subst$.

$$cambiar\_totes(a, b, f(a, f(b, h(b))), f(a, f(a, h(a)))) \Longrightarrow YES$$

10. Definir el predicat $daus(P, N, L)$ on la llista $L$ expressa una manera de sumar $P$ punts llençant $N$ daus. Així si $P$ és 5 i $N$ és 2, una solució de $L$ seria $[1, 4]$; la resta de solucions serien $[3, 2], [4, 1]$ i $[2, 3]$. Fixeu-vos que la longitud de les llistes és 2. $P$ i $N$ han d'estar inicialment instanciades.

11. En el joc de "xifres", el jugador disposa d'una llista $L$ de números enters i un objectiu $N$, que és un altre nombre enter. El jogador ha de trobar una manera d'obtenir $N$ a base de sumar, restar i multiplicar alguns nombres de la llista $L$. Es pot usar cada nombre tantes vegades com aparegui a la llista $L$.

    Per exemple, a $L = [4, 9, 8, 7, 100, 4]$ i $N = 380$, el programa hauria de respondre: $4 * (100 - 7) + 8, ((100 - 9) + 4) * 4, \ldots$

12. Tenim una aixeta d'aigua, un cubell de 5 litres i un altre de 8. Es pot vocar el contingut d'un cubell a un altre, omplir el cubell o buidar un cubell del tot. Escriure un programa Prolog que digui amb quina seqüència d'operacions podem obtenir exactament 4 litres d'aigua en el cubell de 8 litres. Resoleu el problema de manera general, és a dir, que es pugui resoldre qualsevol problema de l'estil "amb un primer cubell de $X$ litres i un segon de $Y$ litres, obtenir $Z$ litres exactes en el segon".

13. Una peça de dominó la representarem amb el terme $peca(num1, num2)$ on $num1$ i $num2$ són els valors de la peça. Construïu un predicat prolog $partida(X, L)$ que ens generi una llista $L$, de peces que representi una partida vàlida prenent la peça $X$ com a punt de partida segons les regles habituals del dominó.

14. Una jugada de pòquer la definim com a una llista de cinc cartes i una carta la representarem com al terme $carta(numero, pal)$ per exemple $carta(10, piques)$. Definiu un predicat prolog $poquer(J1, J2)$ que es satisfaci quan $J1$ sigui una jugada més alta que $J2$.

# 6 Bibliografia

*Mathematical logic for Computer Science*, Ben-Ari
*First-order logic and Automated Theorem Proving*, Fitting
*Essentials of Logic Programming*, C. J. Hogger
*Apunts de lògica*, Josep Humet
*Lógica de Primer Orden, Apuntes-Resumen* Robert Nieuwenhuis
*The Art of Prolog*, L. Sterling &E. Shapiro