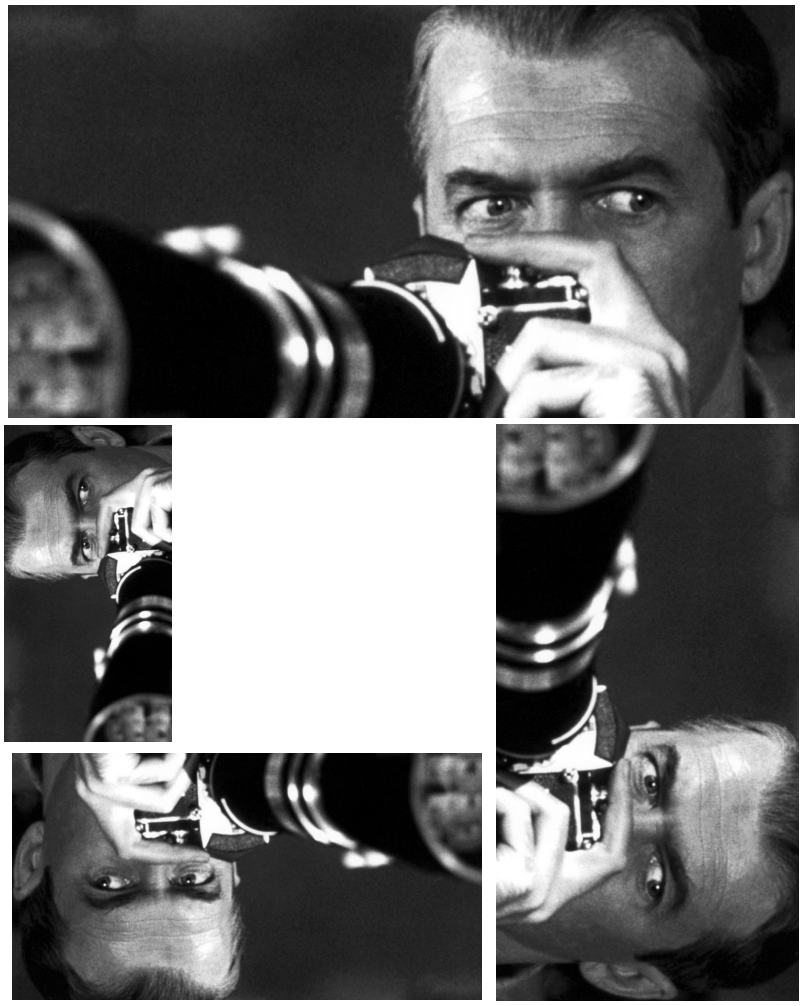
Matemàtica Discreta i Lògica

Teoria i, sobretot, problemes



Francesc J. Ferri

Versió 2.4
Agost 2022

Matemàtica Discreta i Lògica

Teoria i, sobretot, problemes

Francesc J. Ferri
Universitat de València

4 d'agost de 2022

Versió 2.4

Aquest llibre ha rebut un ajut del Servei de Política Lingüística de la Universitat de València en la convocatòria 2020 d'incentius per a la qualitat lingüística en l'elaboració de materials docents.

Em dic Llop. Solucione problemes.

 Agost 2022

Índex

Pròleg

L'assignatura *34666 - Matemàtica discreta i lògica* s'imparteix en primer curs d'Enginyeria Informàtica a la Universitat de València. És una assignatura introductòria que hauria de servir de pont entre conceptes matemàtics, lògics i computacionals, importants a l'hora d'assolir certes competències transversals com ara la capacitat per a representar informació i per a resoldre problemes.

En l'assignatura hi ha quatre blocs principals: combinatòria, lògica, recursió, i estructures gràfiques i arborescents. Encara que en queden fora molts continguts típics de cursos i llibres de matemàtica discreta (com ara resolució de recurrències, codificació, teoria de nombres, etc.), també és veritat que de qualsevol dels quatre blocs es podria impartir una assignatura sencera (encara que no en primer curs).

El tret principal del present manual és la col·lecció de problemes. Es tracta majorment de problemes que han sigut utilitzats en exàmens o en exercicis pràctics. Molts admeten solucions obertes, ramificacions i extensions que poden resultar interessants per a la comprensió dels conceptes relacionats.

Per això s'ha reduït la teoria a l'enumeració de conceptes importants, i s'han estès les solucions de problemes seleccionats perquè il·lustren i expliquen la part teòrica corresponent.

El material no s'hauria d'utilitzar com a apunts complets i autocontinguts, sinó com un resum de conceptes que cal ampliar en altres fonts. Tampoc no s'haurien de considerar les solucions com a respostes tipus que cal estudiar. Al contrari, la millor manera d'aprofitar el material consisteix a intentar seriosament la resolució de cada problema abans de mirar la solució.

El llibre s'ha concebut com un projecte obert, de manera que la col·lecció de problemes puga anar creixent. La idea és que estiga disponible en línia de manera oberta i que es puga navegar fàcilment entre les seues parts.

Finalment, val a dir que aquest projecte s'ha beneficiat d'una manera o d'una altra de material (concret) previ, discussions (discretes) i reflexions (lògiques) amb diversos companys del departament amb qui he tingut el plaer de treballar. Principalment, Fernando Barber, Ignacio García, Sergio Casas, Miguel Lozano i Salva Moreno.

Burjassot, 26 de febrer de 2020.

Índex de Problemes

1. Combinatòria

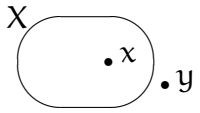
1.1 Conjunts, seqüències i aplicacions

1.1.1 Conjunts

Un **conjunt** és una agrupació o col·lecció de zero o més elements d'algun tipus (per exemple nombres enters, reals, complexos, lletres, o fins i tot altres conjunts) de manera que no importa l'ordre i els elements no es poden repetir.

Un **multiconjunt** és un conjunt en què els elements es poden repetir un nombre arbitrari de vegades.

Un element x **pertany** a un conjunt, X, si és un dels que formen la col·lecció. Ho escrivim com a $x \in X$. Si un element y no pertany a X, s'escriu $y \notin X$. Aquesta situació la representem gràficament mitjançant **diagrames de Venn** com a

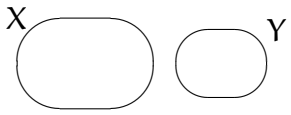


Si tots els elements d'un conjunt X estan en el conjunt Y, diem que Y **conté** X o que X **és contingut** en Y, i ho escrivim com a $X \subseteq Y$. També diem que X és un **subconjunt** de Y.

De la mateixa manera, diem que Y és un **superconjunt** de X. Gràficament

Si $X \subseteq Y$ i $Y \subseteq X$, aleshores necessàriament $X = Y$. Si $X \subseteq Y$ i $X \neq Y$, diem que X és un **subconjunt estricte** de Y. O també que **està estrictament contingut** en Y o que Y **conté estrictament** X. També es pot dir que X és un **subconjunt propi** de, o que **està pròpiament contingut** en Y.

Si cap dels elements d'un conjunt X pertany a un altre conjunt Y (i per tant, també al revés) es diu que X i Y, són **conjunts disjunts**.



De la mateixa manera, ens referim a qualsevol col·lecció de conjunts com a **disjunts** o **mútuament disjunts** si són tots ells disjunts dos a dos.

El **conjunt buit**, $\emptyset$, és un conjunt que no conté cap element. Com a conseqüència, $\emptyset$ és subconjunt de qualsevol altre conjunt. Fins i tot d'ell mateix.

Anomenem **univers** un conjunt que és superconjunt de tots els conjunts en un determinat context.

Per a denotar conjunts utilitzem claus com en

$$A = \{a, b, c\}$$

on hem definit el conjunt A per **extensió** (enumerant tots els seus elements).

També podem definir un conjunt per **comprensió** si especifiquem alguna propietat que ha de complir un element (de l'univers) si i només si està en el conjunt. Per exemple,

$$B = \{x \in \mathbb{N} \mid x = 2k\},$$

es llegeix com "el conjunt B és format per tots aquells nombres naturals tals que es poden escriure com a $2k$ (per a algun k, s'entén)".

En algunes ocasions podem definir conjunts informalment com en $\mathbb{N} = \{1, 2, 3, \ldots\}$. A banda dels **naturals**, $\mathbb{N}$, altres conjunts concrets que utilitzarem sovint són:

$\mathbb{N}$	naturals	$\{1, 2, 3, \ldots\}$
$\mathbb{N}_k$	k-naturals	$\{1, 2, 3, \ldots, k\}$
$\mathbb{Z}$	enters	$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$
$\mathbb{Z}_k$	k-enters	$\{-k, \ldots, -2, -1, 0, 1, 2, \ldots, k\}$
$\mathbb{Z}^+$	enters no negatius	$\{0, 1, 2, \ldots\}$
$\mathbb{Z}_k^+$	k-enters no negatius	$\{0, 1, 2, \ldots, k\}$
$\mathbb{Z}_1^+$	univers binari	$\{0, 1\}$
$\mathbb{Q}$	racionals	$\{p/q \mid p \in \mathbb{Z}, q \in \mathbb{N}\}$
$\mathbb{R}$	reals	{punts de la recta real}
$[0, 1]$	interval unitari tancat	$\{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$
$(0, 1)$	interval unitari obert	$\{x \in \mathbb{R} \mid 0 < x < 1\}$

Donat un conjunt X, anomenem **conjunt potència** de X el conjunt format per tots els seus subconjunts,

$$\mathcal{P}(X) = 2^X = \{Y \mid Y \subseteq X\}.$$

Per exemple, el conjunt potència de $\mathbb{Z}_1^+$ és

$$\{\emptyset, \{0\}, \{1\}, \{0,1\}\}$$

La **cardinalitat** d'un conjunt X, que escrivim com a $|X|$ o com a $\mathrm{card}(X)$, és el nombre d'elements que conté. Si $X \subseteq Y$ aleshores $|X| \leq |Y|$. La cardinalitat d'un conjunt és infinita si conté infinits elements.

Una col·lecció (o conjunt) de conjunts, $\mathcal{B} = \{B_1, \ldots, B_k\}$, s'anomena **partició** d'un conjunt A si tots ells són subconjunts **disjunts no buits** de A i tot element de A pertany també a algun dels subconjunts B_i. Els conjunts B_i s'anomenen **blocs** de la partició $\mathcal{B}$. Com a cas especial, es pot acceptar que $\{\emptyset\}$ és una partició de $\emptyset$.

Donades dues particions, $\mathcal{B}, \mathcal{C}$, d'un mateix conjunt A, es diu que $\mathcal{C}$ és un **refinament** de $\mathcal{B}$ si tot bloc de $\mathcal{C}$ és subconjunt d'algun bloc de $\mathcal{B}$.

En el següent exemple, $\mathcal{B}$, és una partició (en 3 blocs) de $\mathbb{Z}$, i $\mathcal{C}$ és un refinament (amb 5 blocs) de $\mathcal{B}$.

$\mathcal{B} = \{\mathbb{N}, \{0\}, \{n \mid -n \in \mathbb{N}\}\}$,
$\mathcal{C} = \{\{n \mid n = 2k, k \in \mathbb{N}\}, \{n \mid n = 2k-1, k \in \mathbb{N}\}, \{0\}, \{n \mid n = -2k, k \in \mathbb{N}\}, \{n \mid n = 1-2k, k \in \mathbb{N}\}\}$,

1.1.2 Operacions sobre conjunts

La **unió** de dos conjunts, A, B, és formada pels elements que estan en A o en B. S'escriu $A \cup B$. Formalment,

$$A \cup B = \{x \mid x \in A \vee x \in B\}.$$

La **intersecció** de dos conjunts, A, B, és formada pels elements que estan en A i en B. S'escriu $A \cap B$. Formalment,

$$A \cap B = \{x \mid x \in A \wedge x \in B\}.$$

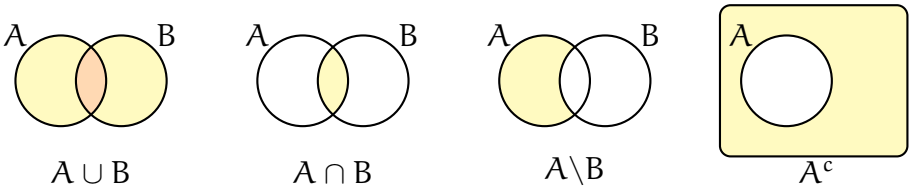
La **diferència conjuntista** entre A i B, és formada pels elements de A que no estan en B. S'escriu $A \backslash B$, o també $A - B$. Formalment,

$$A \backslash B = \{x \mid x \in A \wedge x \notin B\}.$$

El **complement** del conjunt A (respecte de l'univers U) es defineix com a $\overline{A} = A^c = U \backslash A$. Formalment,

$$A^c = \{x \mid x \notin A\}.$$

Aquestes quatre operacions, gràficament es poden representar mitjançant diagrames de Venn de la següent manera.



1.1.3 Seqüències i tuples

Una **seqüència** és una enumeració ordenada de zero o més elements normalment del mateix tipus. Una **tupla** és un element del **producte cartesià** de zero o més conjunts.

Una seqüència (homogènia) és una tupla sobre un únic conjunt d'elements. No distingirem entre seqüències i tuples que escriurem com a $(x_1, x_2, \ldots)$.

Si $x_i \in X_i$ per a i=1,2,..., k, aleshores

$$(x_1, x_2, \ldots, x_k) \in X_1 \times X_2 \times \cdots \times X_k.$$

També podem escriure que

$$X_1 \times X_2 \times \cdots \times X_k = \{(x_1, x_2, \ldots, x_k) \mid x_i \in X_i, i = 1, 2, \ldots, k\}.$$

Una tupla de k elements s'anomena k-tupla. El valor de k és la **dimensió**, **grandària** o **talla** de la tupla. Un **parell** és una 2-tupla. Una **terna**, **trio** o **tripleta** és una 3-tupla. De vegades, anomenarem **vector** qualsevol k-tupla de nombres (reals, en principi).

$$(x_1, x_2, \ldots, x_k) \in \mathbb{R}^k.$$

Si estem considerant conjunts sobre un determinat univers els elements del qual es poden numerar, $U = \{e_1, e_2, \ldots, e_n\}$, aleshores podem definir per a cada conjunt, $A \subseteq U$, el seu **vector característic** com a

$$\chi_A = (x_1, x_2, \ldots, x_n),$$

on

$$x_i = \begin{cases} 1 & \text{si } e_i \in A \\ 0 & \text{si } e_i \notin A \end{cases}$$

Per exemple, els vectors característics dels conjunts $\emptyset$ i $\{2, 3\}$ sobre l'univers $\mathbb{N}_6$ són $(0, 0, 0, 0, 0, 0)$ i $(0, 1, 1, 0, 0, 0)$, respectivament.

Els *vectors* característics sobre universos infinits són de dimensió infinita. Per exemple, $(0, 1, 0, 1, 0, \ldots)$ representaria el subconjunt format pels nombres parells de $\mathbb{N}$.

1.1.4 Relacions i aplicacions

Una **relació** (binària), R, entre els conjunts A i B és un subconjunt del producte cartesià $A \times B$. És a dir, un conjunt de parells formats per un element de A i un altre de B. O siga,

$$R \subseteq A \times B.$$

En el cas particular que $B = A$ diem que R és una relació en el conjunt A.

Diem que un element $a \in A$ està relacionat amb un element $b \in B$ segons R i ho escrivim com a aRb si i només si

$$(a, b) \in R \subseteq A \times B.$$

En cas contrari els elements no estan relacionats i escrivim $a \not R b$. Es poden definir relacions entre un nombre k d'elements diferent de 2. Les anomenem **relacions k-àries**. L'enter k és l'**aritat** de la relació.

Una relació que siga $R = \emptyset$ (ningú es relaciona amb ningú) o $R = A \times B$ (tots es relacionen amb tots) s'anomena **trivial**.

Una relació entre dos conjunts A i B en què almenys un element de A està relacionat amb almenys un element de B s'anomena **correspondència**.

Com que una relació (binària) no és més que un subconjunt del producte cartesià de dos conjunts, sempre és possible definir el vector característic d'una relació a partir de qualsevol enumeració d'aquest producte.

Per exemple, si considerem la igualtat mòdul 2 entre els conjunts $\mathbb{Z}_2^+$ i $\mathbb{Z}_3^+$,

$$n R_2 m \quad \textbf{sii} \quad \text{mod}\,(n, 2) = \quad \text{mod}\,(m, 2)$$

i l'enumeració

$$(0,0), (0,1), (0,2), (0,3), (1,0), \ldots, (2,2), (2,3)$$

també anomenada enumeració lèxico-gràfica, el corresponent vector característic de R_2 seria

$$(1,0,1,0,0,1,0,1,1,0,1,0)$$

No obtstant això, en el cas de relacions, resulta més natural i interessant representar aquest vector com una matriu, $\chi_R = [x_{i,j}]$, que anomenarem **matriu característica** de la relació $R \subseteq A \times B$, que es pot definir com

$$x_{i,j} = \begin{cases} 1 & \text{si } a_i R b_j \\ 0 & \text{si } a_i \not R b_j \end{cases}$$

on

$$A = \{a_1, \ldots, a_n\}, \; B = \{b_1, \ldots, b_m\}$$

Per exemple, la matriu característica de la relació R_2 anterior seria

$$\chi_{R_2} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Una **relació d'equivalència** és una relació binària en un conjunt A que és **reflexiva** (aRa per a qualsevol a), **simètrica** (si aRb aleshores bRa per a qualsevol parell d'elements), i **transitiva** (si aRb i bRc aleshores aRc per a qualsevol terna d'elements).

Exemples de relacions d'equivalència serien la igualtat, $R_=$, o la igualtat mòdul 2, R_2, definides sobre un únic conjunt. Per exemple, sobre el conjunt $\mathbb{Z}_3^+$, les matrius característiques d'aquestes relacions serien

$$\chi_{R_=} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \chi_{R_2} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Donada una relació binària d'equivalència R sobre un conjunt A, anomenem **classe d'equivalència** tot subconjunt B de A que compleix

$$aRb \textbf{ sii } a \in B, b \in B$$

És a dir, B conté **tots** els elements de A que estan relacionats entre ells.

La notació $[a]_R$ significa classe d'equivalència de R en A que conté l'element $a \in A$. Per tant, donada una classe d'equivalència B, es té que

$$[a]_R = B, \forall a \in B$$

El subíndex es pot eliminar si no pot haver-hi confusió.

S'anomena **conjunt quocient** del conjunt A i la relació R, o simplement quocient de la relació, i se denota com a A/R, el conjunt format per **totes** les classes d'equivalència induïdes per R en A.

$$A/R = \{[a] \mid a \in A\}$$

Els conjunts quocient corresponents a les relacions $R_=$ i R_2 definides anteriorment serien

$$\mathbb{Z}_3^+/R_= = \{\{0\},\{1\},\{2\},\{3\}\} \qquad \mathbb{Z}_3^+/R_2 = \{\{0,2\},\{1,3\}\}$$

Si considerem la relació R_2 sobre el conjunt infinit $\mathbb{Z}^+$ tindríem també un conjunt quocient de cardinalitat 2,

$$\mathbb{Z}^+/R_2 = \{[0],[1]\}$$

on les classes d'equivalència $[0]$ i $[1]$ contenen els enters parells i senars, respectivament.

Una **relació d'ordre** és una relació binària en un conjunt A que és **reflexiva**, **antisimètrica** (si aRb i bRa aleshores $a = b$ per a qualsevol parell), i **transitiva**.

Serien exemples la relació $\leq$ entre nombres o la relació $\subseteq$ entre conjunts. Per exemple, les relacions $R_{\leq}$ en $\mathbb{Z}_3^+$ i $R_{\subseteq}$ en $2^{\mathbb{Z}_1^+} = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$ serien relacions d'ordre i tindrien com a matrius característiques,

$$\chi_{R_{\leq}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \chi_{R_{\subseteq}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

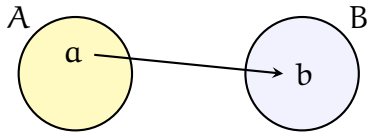
En canvi, les relacions estrictes $<$ i $\subset$ no són relacions d'ordre ja que no es compleix la propietat **reflexiva**.

Una **aplicació**, f, del conjunt A al conjunt B és una relació on **cada** element de A està relacionat amb un **únic** element de B. El conjunt A s'anomena **conjunt de partida** o **domini** de f, $\mathrm{Dom}(f)$, i el conjunt B s'anomena **conjunt d'arribada** o **codomini** de f, $\mathrm{Cod}(f)$. Per indicar que f és una aplicació de A a B escrivim

$$f : A \to B$$

Si $a \in A$ està relacionat amb $b \in B$ segons f escrivim $f(a) = b$ en lloc de afb. Diem que b és la **imatge** de a o, equivalentment, que a és **antiimatge** de b. El **conjunt imatge** o **rang** de f, $\mathrm{Im}(f)$, és el subconjunt de B format per aquells elements que tenen antiimatge en A.

Gràficament, dibuixem una fletxa des de a fins a b per a indicar que $f(a) = b$.



Aquesta mateixa representació gràfica es pot fer servir també per a **relacions** i per a **correspondències** (es dibuixa una fletxa de a a b si aRb).

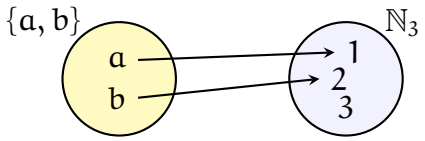
Una aplicació és **injectiva** si tot element del conjunt imatge té, com a molt, una única antiimatge. Una aplicació és **suprajectiva** o **exhaustiva** si no hi ha elements en el conjunt imatge sense antiimatge. Diem que una aplicació és **bijectiva** si és

alhora **injectiva** i **exhaustiva**.

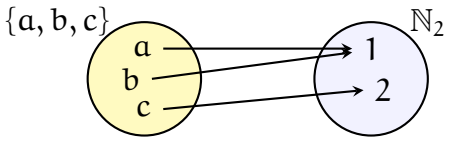
Tot element del domini d'una aplicació f ha de tenir imatge i aquesta ha de ser única (per definició d'aplicació). Els elements del codomini poden no tenir antiimatge, tenir una única antiimatge o tenir-ne més d'una. Tots en tindran una o cap si l'aplicació és injectiva. I tots en tindran una o més si f és exhaustiva.

Els elements de $\mathrm{Im}(f)$ han de tenir exactament una única antiimatge si f és bijectiva o simplement injectiva. I una o més si és exhaustiva.

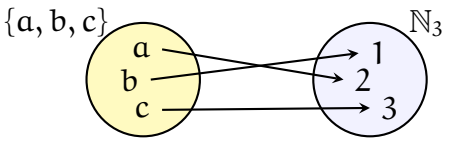
Alguns exemples d'aplicacions sobre conjunts petits es mostren gràficament a continuació.



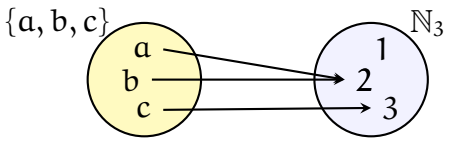
injectiva no exhaustiva



exhaustiva no injectiva



bijectiva



ni injectiva ni exhaustiva

La **composició** de les aplicacions $f : A \to B$ i $g : B \to C$, és una aplicació $f \circ g : A \to C$ de manera que $f \circ g(a) = g(f(a))$ per a tot $a \in A$.

Donada una aplicació $f : A \to B$, definim la **inversa**, f^{-1}, per a qualsevol element $y \in B$ com a

$$f^{-1}(y) = \{x \in A \mid \ f(x) = y\}.$$

Aquesta inversa es pot veure com una relació entre B i A que no és necessàriament una aplicació. De fet, la **relació inversa** es pot definir per a qualsevol relació exactament de la mateixa manera:

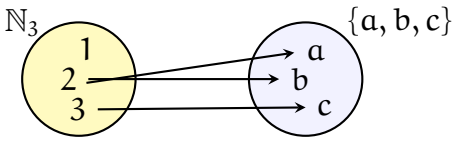
Donada una relació, $R \subseteq A \times B$, la **relació inversa**, $R^{-1} \subseteq B \times A$, es defineix de manera que per a tot parell d'elements, $a \in A$, $b \in B$,

$$bR^{-1}a \quad \textbf{sii} \quad aRb,$$

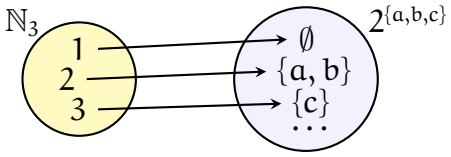
Aquesta relació inversa es pot representar gràficament a partir de la representació gràfica de la relació (directa) només invertint el sentit de les fletxes.

En el cas particular de les aplicacions, la inversa també es pot veure com una aplicació entre B i el conjunt potència de A, $\mathcal{P}(A)$. Per exemple, la inversa de l'últim

dels quatre exemples anteriors es podria veure com a una relació, $f^{-1} \subseteq \mathbb{N}_3 \times \{a, b, c\}$, o també com una aplicació, $f^{-1} : \mathbb{N}_3 \longrightarrow 2^{\{a,b,c\}}$.



correspondència (relació), no aplicació



aplicació no exhaustiva

Donada qualsevol aplicació $f : A \to B$ no exhaustiva, es pot definir trivialment una altra aplicació exhaustiva només canviant el codomini pel conjunt imatge de f. És a dir, $f' : A \to B' \subseteq B$ on $B' = \mathrm{Im}(f) = \{y \in B \mid f(x) = y \text{ per a algun } x\}$.

La inversa de l'últim exemple serà exhaustiva si considerem el conjunt $\{\emptyset, \{a, b\}, \{c\}\}$ com a codomini en lloc de la totalitat del conjunt potència.

Donada qualsevol aplicació $f : A \to B$ **injectiva**, sempre podrem definir la seua **aplicació inversa**, $f^{-1} : \mathrm{Im}(f) \to A$ com a $f^{-1}(y) = x$ si i només si $f(x) = y$.

1.1.5 Aplicacions i cardinalitat

A partir de les definicions es pot arribar a una relació entre les cardinalitats dels dominis i codominis, i el tipus d'aplicació que hi ha entre ells.

Diem que $|A| \leq |B|$ **sii** $\exists f : A \to B$ injectiva.

Diem que $|A| \geq |B|$ **sii** $\exists f : A \to B$ exhaustiva.

Diem que $|A| = |B|$ **sii** $\exists f : A \to B$ bijectiva (**regla de la bijecció**)

Un conjunt té **cardinalitat** k si es pot trobar una bijecció entre ell i el conjunt $\mathbb{N}_k$.

1.2 Comptatge de conjunts d'elements

Hi ha una sèrie de regles que permeten relacionar les cardinalitats de conjunts diferents de manera que pot resultar relativament senzill deduir la cardinalitat d'un conjunt a partir de la cardinalitat de l'altre.

regla de la bijecció: el cardinal de dos conjunts entre els quals existeix una bijecció és el mateix.

regla de la divisió: si hi ha una aplicació exhaustiva $f : A \to B$ de manera que tot element de B té exactament k antiimatges (aplicació k a 1), aleshores $|A| = k|B|$.

principi de les caixes: Si $|A| > |B|$ aleshores per a qualsevol aplicació, $f : A \to B$, almenys un element de B ha de tenir més d'una antiimatge. Equivalentment, almenys 2 elements de A han de tenir la mateixa imatge.

principi de les caixes generalitzat: Si $|A| > k|B|$ aleshores per a qualsevol aplicació, $f : A \to B$, almenys un element de B ha de tenir més de k antiimatges. Equivalentment, almenys $k + 1$ elements de A han de tenir la mateixa imatge.

regla del producte: el cardinal del producte cartesià de dos conjunts és el producte dels seus cardinals, $|A \times B| = |A| \cdot |B|$.

regla de la suma: el cardinal de la unió de dos conjunts disjunts és la suma de les seues cardinalitats.

principi d'inclusió-exclusió o **regla de la suma generalitzada**: el cardinal de la unió de dos conjunts és la suma de les seues cardinalitats menys la cardinalitat de la seua intersecció.

1.3 Variacions, permutacions i combinacions

El conjunt de les **variacions amb repetició de m elements agafats de n en n**, VR_m^n, és el conjunt format per totes les possibles n-tuples de $\mathbb{N}_m$. O siga, $\mathbb{N}_m^n$. El seu cardinal s'obté, per tant, com a

$$|VR_m^n| = |\mathbb{N}_m|^n = m^n$$

en aplicar el **principi del producte**.

A causa del principi de la bijecció, identificarem VR_m^n amb n-tuples sobre qualsevol altre conjunt de cardinalitat m. També de vegades abusarem de la notació i usarem VR_m^n per a referir-nos en realitat al seu cardinal.

El conjunt de les **variacions sense repetició de m elements agafats de n en n**, V_m^n, és el subconjunt de VR_m^n que conté només les n-tuples sense elements repetits. El seu cardinal s'obté també mitjançant la regla del producte com a

$$|V_m^n| = |\mathbb{N}_m| \cdot (|\mathbb{N}_m| - 1) \cdots (|\mathbb{N}_m| - n + 1) = m^{\underline{n}}$$

on $m^{\underline{n}} = m(m-1)\cdots(m-n+1)$ és la **potència decreixent n-èsima de m** (o també **factorial decreixent d'ordre n de m**) sempre que $n \leq m$. En el cas en què $n > m$ el cardinal de V_m^n és òbviament zero.

El conjunt de les **permutacions de m elements**, P_m, és el conjunt V_m^m. És a dir, totes les m-tuples de $\mathbb{N}_m$ sense elements repetits. Aquest conjunt es correspon amb totes les maneres possibles d'ordenar els m primers nombres naturals (o qualsevol altre conjunt de cardinalitat m). El cardinal de P_m és determinat per

$$|P_m| = |V_m^m| = m^{\underline{m}} = m!$$

Alternativament, una permutació dels m elements d'un conjunt A es pot veure com una aplicació bijectiva de A en ell mateix. O altrament dit, hi ha una bijecció entre el conjunt P_m i el conjunt de totes les aplicacions bijectives de la forma $f : A \longrightarrow A$ si $|A| = m$.

El conjunt de les **permutacions amb repetició de m elements que es repeteixen $(k_1, k_2, \ldots, k_m)$ vegades**, $PR_{(k_1,k_2,\ldots,k_m)}$, és el conjunt format per les $(k_1 + k_2 + \cdots + k_m)$-tuples de $\mathbb{N}_m$ que contenen k_i repeticions de l'element i, per a $i = 1, \ldots, m$.

El cardinal de $PR_{(k_1,k_2,\ldots,k_m)}$ es pot obtenir a partir del cardinal de les permutacions de $(k_1 + k_2 + \cdots + k_m)$ elements, $P_{(k_1+k_2+\cdots+k_m)}$, en aplicar el **principi de la divisió** com a

$$|PR_{(k_1,k_2,\ldots,k_m)}| = \frac{(k_1 + k_2 + \cdots + k_m)!}{k_1!\, k_2! \cdots k_m!} = \frac{(\sum_{i=1}^m k_i)!}{\prod_{i=1}^m k_i!} = (k_1, \ldots, k_m)!$$

on $(k_1, \ldots, k_m)!$ és el que es coneix com a **coeficients multinomials**

Per a justificar l'anterior podem raonar de la següent manera. Suposem que les k_i diferents ocurrències de cada element, i, en cada permutació amb repetició de $PR_{(k_1,k_2,\ldots,k_m)}$ es pogueren distingir. En aquest cas, el cardinal que cerquem seria

$$(\sum_{i=1}^m k_i)!$$

Per exemple, les permutacions amb repetició, $PR_{(3,2,1)}$ sobre el conjunt $\{a, b, c\}$ les podríem identificar amb les permutacions, P_6 sobre el conjunt $\{a_1, a_2, a_3, b_1, b_2, c_1\}$. Si considerem una de les permutacions com per exemple

$$(c_1, b_2, a_3, a_1, b_1, a_2),$$

veiem que es poden permutar les a de $P_3 = 6$ maneres, les b de $P_2 = 2$ maneres i les c de $P_1 = 1$ manera, sense que canvie la permutació amb repetició.
En general, la diferència entre distingir o no distingir entre les k_i ocurrències de l'element i és que cada tupla del segon cas es correspon amb $k_i!$ tuples del primer cas, que serien totes les maneres de permutar les k_i ocurrències allà on foren. Com que això es pot fer amb els m elements alhora, hi haurà una correspondència $\prod_{i=1}^{m} k_i!$ a 1 entre els conjunts $P_{(k_1+k_2+\cdots+k_m)}$ i $PR_{(k_1,k_2,\ldots,k_m)}$, per la qual cosa es pot aplicar el **principi de la divisió**.

El conjunt de les **combinacions de m elements agafats de n en n**, C_m^n, és format per tots els subconjunts de $\mathbb{N}_m$ de cardinalitat n i es correspon amb totes les maneres de triar n elements d'entre m possibles sense que importe l'ordre.

Si pensem en els vectors característics dels conjunts que formen C_m^n, veiem que es tracta de m-tuples en $\mathbb{Z}_1^+ = \{0, 1\}$ que contenen exactament n uns i $m - n$ zeros. Per tant, també hi ha una bijecció entre C_m^n i $PR_{(m-n,n)}$, per la qual cosa podem obtenir la cardinalitat de C_m^n com a

$$|C_m^n| = \frac{m!}{n!(m-n)!} = \binom{m}{n}$$

on $\binom{m}{n}$ representa el que es coneix com a **coeficient binomial** o **nombre combinatori** d'ordre (m, n) i es llegeix com "m sobre n".

Algunes propietats interessants dels coeficients binomials són

$$\binom{m}{n} = \binom{m}{m-n} \qquad \sum_{k=1}^{m} \binom{m}{k} = 2^m \qquad \binom{m}{n} = \binom{m-1}{n} + \binom{m-1}{n-1}$$

La primera és trivial. La segona es pot deduir a partir del **principi de la suma** i del fet que hi ha una bijecció entre $\mathcal{P}(\mathbb{N}_m)$ (subconjunts de $\mathbb{N}_m$) i VR_2^m (els seus vectors característics). La tercera és un poc més complicada. La seua deducció així com els casos particulars es deixen com a exercici.

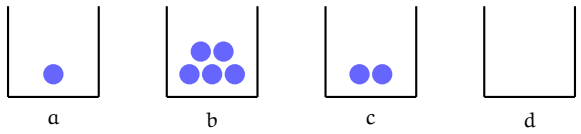
El conjunt de les **combinacions amb repetició de m elements agafats de n en n**, CR_m^n, és format per tots els multiconjunts sobre $\mathbb{N}_m$ de cardinalitat n i es correspon amb totes les maneres de triar n elements d'entre m possibles sense que importe l'ordre, però amb l'opció de poder elegir el mateix element més d'una vegada.

Una bijecció interessant és que CR_m^n es correspon també amb les differents formes de distribuir n objectes indistingibles entre m contenidors il·limitats i distingibles (o

numerats). Donat un multiconjunt particular, cada contenidor és un element de $\mathbb{N}_m$ i els objectes que conté són les vegades que aquest element apareix en el multiconjunt.

Per exemple, la figura següent il·lustra un cas particular de multiconjunts de 4 elements.

$\{a, b, b, b, b, b, c, c\} \quad \leftrightarrow$



Una altra bijecció més interessant encara entre CR_m^n i un subconjunt de $\mathbb{Z}_1^+$ és la següent. Representem el nombre d'ocurrències de l'i-èsim element, k_i, en unari. És a dir, com una k_i-tupla d'uns, $(\underbrace{1, 1, \ldots, 1}_{k_i \text{vegades}})$. Si concatenem les m k_i-tuples, però les separem amb $m-1$ zeros tenim una representació que és única per a tots els possibles multiconjunts de talla $n = k_1 + k_2 + \cdots + k_m$,

$$(\underbrace{1, 1, \ldots, 1}_{k_1}, 0, \underbrace{1, 1, \ldots, 1}_{k_2}, 0, \ldots, 0, \underbrace{1, 1, \ldots, 1}_{k_m}).$$

Això vol dir que podem representar qualsevol multiconjunt de cardinalitat n com una seqüència de $\mathbb{Z}_1^+$ que continga exactament n uns i $m-1$ zeros. I de la mateixa manera, donada qualsevol cadena amb n uns i $m-1$ zeros, es pot interpretar com un multiconjunt de m elements on cada un es pot repetir zero o més vegades però la cardinalitat és n.

En altres paraules, hi ha una bijecció entre les $(n+m-1)$-tuples de $\mathbb{Z}_1^+$ amb n uns i els conjunts de $n+m-1$ elements de talla n, per la qual cosa es pot obtenir el cardinal de CR_m^n com a

$$|CR_m^n| = |C_{n+m-1}^n| = \binom{n+m-1}{n} = \left(\!\!\binom{m}{n}\!\!\right)$$

on $\left(\!\binom{m}{n}\!\right)$ és el que es coneix com a **coeficient multiconjunt**.

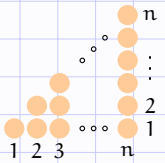
1.4 Tècniques visuals de comptatge

Una forma interessant d'aplicar la regla de la bijecció consisteix a fer-ho entre descripcions visuals o diagramàtiques del que es vol comptar. Exemples interessants són alguns sumatoris que apareixen sovint en computació.

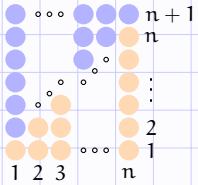
Suma dels n primers enters

$$S_1(n) = \sum_{i=1}^{n} i = 1 + 2 + \cdots + n$$

Podem representar cada terme del sumatori com una pila de boletes d'altures 1, 2, 3, ... i col·locar-les en fila



de manera que el valor del sumatori es correspon amb el nombre de boletes. En altres paraules, es pot establir una bijecció entre el valor del sumatori per a cada n i el nombre de boletes en n piles de boletes.
Però si considerem dues vegades el mateix sumatori i girem la representació gràfica 180º i les ajuntem, tenim que



A partir de la representació gràfica queda clar que

$$2S_1(n) = n \cdot (n+1)$$

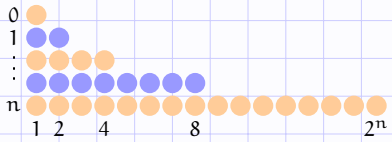
d'on s'obté

$$S_1(n) = \frac{n}{2}(n+1)$$

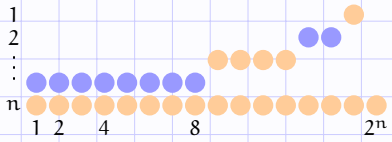
Suma dels $n+1$ dobles successius

$$S_d(n) = \sum_{i=0}^{n} 2^i = 1 + 2 + 4 + \cdots + 2^n$$

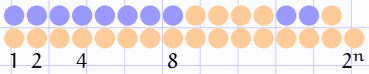
Representarem ara cada un dels $n+1$ termes del sumatori com una **fila** de boletes de colors alterns i les apilarem



Si ara desplacem les files de la següent manera



i les compactem totes en només dues files



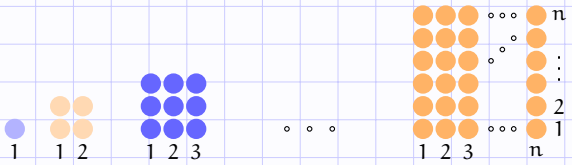
observem clarament que el nombre total de boles és

$$S_d(n) = 2 \cdot 2^n - 1$$

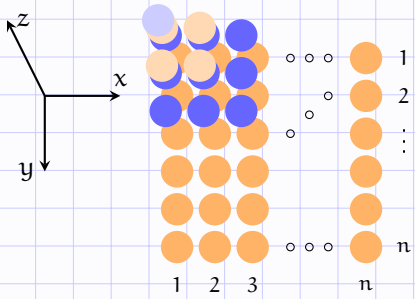
Suma dels n primers quadrats

$$S_2(n) = \sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2$$

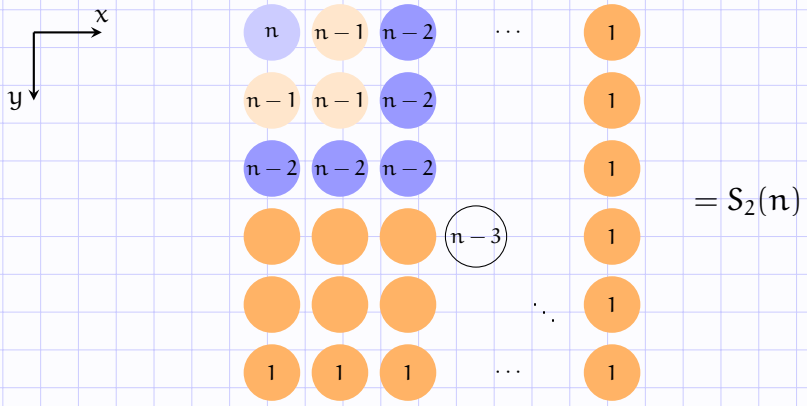
Representem el terme i-èsim del sumatori com una quadrícula de $i \times i$ boletes



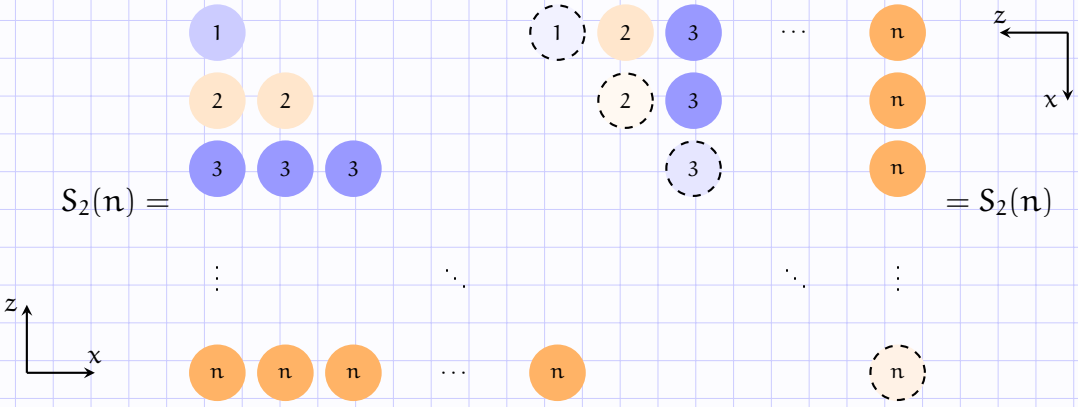
i les apilem totes formant una espècie de piràmide de manera que el seu vèrtex estiga en el cantó superior esquerre.



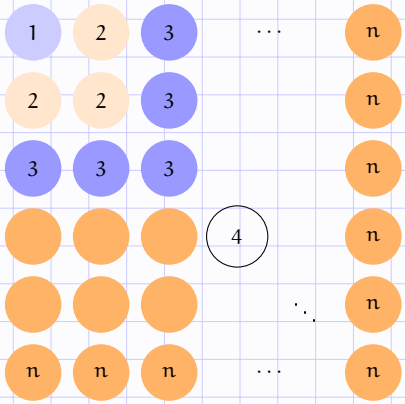
Si mirem aquesta piràmide des de la direcció z i apuntem en cada cercle el nombre de boles en cada posició (x, y), tenim



I si fem el mateix però mirant la piràmide des de la direcció y. I, a part, copiem aquest mateix resultat, però canviant files per columnes, obtenim



Si superposem ara les dues quadrícules després d'eliminar la diagonal que estava marcada la suma de la qual és exactament $S_1(n)$, arribem a



$$= 2 \cdot S_2(n) - S_1(n)$$

Si superposem ara a aquesta quadrícula de $n \times n$ boles numerades (o piles de boles) a la que es veu des de direcció z, obtenim una quadrícula de $n \times n$ piles de $n+1$ boles (o boles numerades on totes valen exactament $n+1$). De manera equivalent, podem escriure

$$n \times n \times (n+1) = S_2(n) + 2 \cdot S_2(n) - S_1(n)$$

d'on s'obté que

$$3S_2(n) = n^2(n+1) - S_1(n) = (2n+1)S_1(n)$$

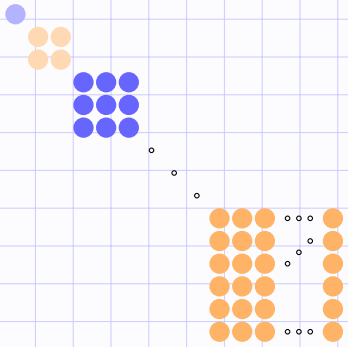
i, per tant,

$$S_2(n) = \frac{2n+1}{3} S_1(n) = \frac{n}{6}(n+1)(2n+1)$$

Suma dels n primers cubs

$$S_3(n) = \sum_{i=1}^{n} i^3 = 1^3 + 2^3 + \cdots + n^3$$

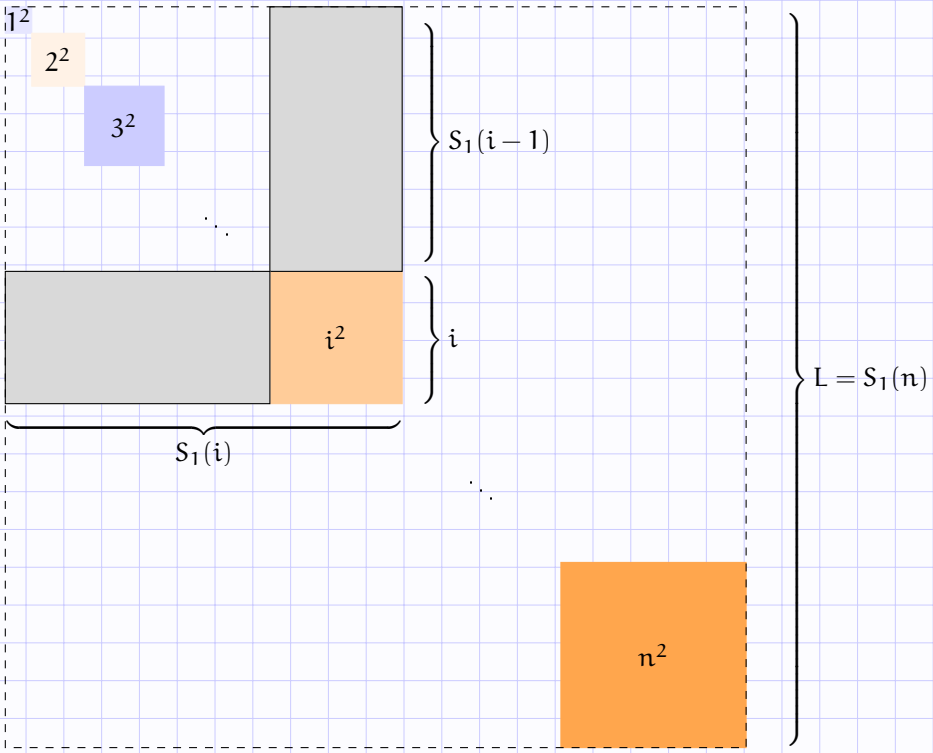
Lamentablement no és fàcil generalitzar els procediments anteriors per a aquest sumatori.
Considerarem la representació gràfica dels termes del sumatori anterior, $S_2(n)$, en la pàgina 15 i tornarem a dibuixar-los però en diagonal.



És clar que podem inscriure la representació anterior en una quadrícula de talla $L \times L$ on

$$L = 1 + 2 + 3 + \cdots + n = S_1(n).$$

Representarem els termes com a àrees i associarem al terme i-èsim l'àrea rectangular que està exactament damunt i també la que queda a la seua esquerra (pintades en gris en la figura).



L'àrea associada a l'i-èsim terme aleshores és i^2 més dues vegades l'àrea grisa que és

$$iS_1(i-1)$$

per la qual cosa la i-èsima àrea és

$$i^2 + 2iS_1(i-1) = i^2 + i^2(i-1) = i^3$$

O siga, que a cada terme li estem associant una àrea que és exactament igual a l'i-èsim cub. I com que la suma d'aquests termes estesos per una banda ha de ser igual a la suma dels cubs, però per l'altra ha de ser igual a l'àrea total de la quadrícula, que és L^2, resulta que

$$S_3(n) = S_1(n)^2 = \frac{n^2}{4}(n+1)^2$$

1.5 Problemes resolts i comentats

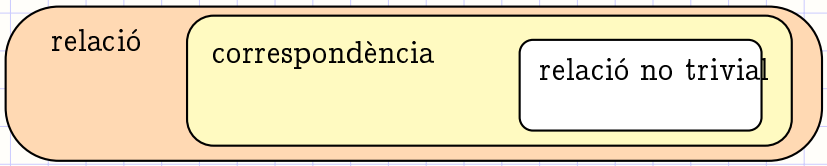
Problema 1.1:[correspondenciaRelacio] És el mateix **correspondència** que **relació no trivial**?

La pregunta són en realitat dues: 1) Tota relació no trivial és correspondència? i 2) Tota correspondència és relació no trivial?

La resposta a 1) és **sí**, ja que almenys algú ha d'estar relacionat amb algú. Si no, seria trivial. La resposta a 2) és **no**, perquè una correspondència en què tots es relacionen amb tots és trivial per definició.

Com a conseqüència, correspondència és més general que relació no trivial. Tota relació no trivial és correspondència, però no al revés.

La situació es mostra gràficament en la figura.



Donats un parell de conjunts, A, B, l'única relació entre ells que no és correspondència és la relació buida, $\emptyset$. I l'única correspondència que és relació trivial és la seua relació complementària, $A \times B$.

Problema 1.2:[unioDeTres] Quina és la cardinalitat de $A \cup B$ en funció de les cardinalitats de A i B? I la de $A \cup B \cup C$?

Pel principi d'inclusió-exclusió tenim que

$$|A \cup B| = |A| + |B| - |A \cap B|$$

Si apliquem l'anterior dues vegades, s'arriba a

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C|$$

Problema 1.3:[dabaleArroz]
La frase

"Dábale arroz a la zorra el abad"

és una frase palindròmica (les lletres són les mateixes si es llegeixen d'esquerra a dreta o de dreta a esquerra) formada per 7 paraules i 25 lletres (sense comptar espais).

a) Si no considerem espais, quantes frases palindròmiques diferents podem formar amb les mateixes lletres?

b) I si considerem espais?

c) I quantes frases palindròmiques de 7 paraules podríem formar?

d) I si només considerem paraules de 4 i 5 lletres?

e) I si considerem paraules de 3, 4 i 5 lletres?

a) La lletra central de la frase és la ℓ i és l'única que apareix un nombre imparell de vegades. Per tant, qualsevol frase palindròmica amb les mateixes lletres haurà de tenir-la en el centre.

A cada costat de la lletra central tenim 8 lletres diferents, $(d, a, b, e, r, o, z, \ell)$, que es repeteixen $(1, 4, 1, 1, 2, 1, 1, 1)$ vegades, respectivament. 12 en total.

La posició central ha quedat fixada i les lletres de la segona meitat han de ser les mateixes que les de la primera però en sentit invers.

Per tot això, el nombre de frases palindròmiques que es poden formar és determinat per les **permutacions amb repetició de 8 elements que es repeteixen $(4, 2, 1, 1, 1, 1, 1, 1)$ vegades**.

$$S_a = |PR_{(4,2,1,1,1,1,1,1)}| = \frac{12!}{4! \cdot 2! \cdot (1!)^6} = \frac{11!}{4} = \boxed{9\,979\,200}$$

b) Si considerem espais (entre qualssevol de les 25 lletres), resulta que hi ha 24 possibles posicions on els podem posar (o no). I les formes diferents de posar espais és determinada aleshores per les **variacions amb repetició de 2 elements agafats de 24 en 24**.

$$E = |VR_2^{24}| = 2^{24} = 16\,777\,216$$

I pel **principi del producte**, el total de frases palindròmiques amb espais serà

$$S_b = \boxed{E \times S_a} = 2^{22} \cdot 11! = \boxed{167\,423\,193\,907\,200}$$

c) Si hi ha 7 paraules és perquè hi ha exactament 6 espais que haurem de posar en algunes de les 24 possibles posicions. Per tant, en la fórmula anterior haurem de canviar les variacions amb repetició per **combinacions sense repetició de 24 elements agafats de 6 en 6**.

$$E_6 = |C_{24}^6| = \binom{24}{6} = \frac{24^{\underline{6}}}{6!} = \frac{\cancel{24} \cdot 23 \cdot 22 \cdot 21 \cdot \cancel{20}^{\,2} \cdot 19}{\cancel{6} \cdot \cancel{5} \cdot \cancel{4} \cdot 3 \cdot \cancel{2} \cdot 1} = 23 \cdot 22 \cdot 7 \cdot 2 \cdot 19 = 134\,596$$

Amb la qual cosa i també pel **principi del producte** obtenim

$$S_c = \boxed{E_6 \times S_a} = 23 \cdot 22 \cdot 7 \cdot 2 \cdot 19 \cdot \frac{11!}{4} = 23 \cdot 11 \cdot 7 \cdot 1 \cdot 19 \cdot 11! = \boxed{1\;343\,160\,403\,200}$$

d) En total hi ha 25 lletres. Per tant les úniques opcions són i) 5 paraules de 5 lletres, ii) 5 paraules de 4 lletres i 1 de 5.

En el cas i) només hi ha **una** opció per a col·locar els espais: cada 5 lletres.

En el cas ii) tenim tantes opcions com posicions relatives puga ocupar la paraula de 5 lletres dins la seqüència de les 6 paraules. És a dir, **sis** opcions.

En general, si tinguérem n paraules d'un tipus i m paraules de l'altre, les possibilitats (per als espais) serien determinades per les **permutacions amb repetició de 2 elements que es repeteixen n i m vegades.** En particular, per a cada un dels casos tenim

$$S_i = S_a \times PR_{(5,0)} = S_a$$
$$S_{ii} = S_a \times PR_{(5,1)} = S_a \cdot \frac{6!}{5! \cdot 1!} = 6S_a$$

I en aplicar el **principi de la suma** s'obté que

$$S_d = S_i + S_{ii} = \boxed{7S_a} = \boxed{69\,854\,400}$$

e) El raonament és exactament el mateix però amb més casos. En la primera columna de la taula següent s'indiquen els casos. Cada paraula es representa mitjançant un dígit que indica quantes lletres té. Els casos i) i ii) anteriors es marquen com a superíndexs.

55555 *i)		$PR_{(5,0,0)} = \frac{5!}{5!} = 1$
555 4 33		$PR_{(3,1,2)} = \frac{6!}{2!\,3!} = 6 \cdot 5 \cdot 4/2 = 60$
55 . . .	55 444 3	$PR_{(2,3,1)} = 60$
	55 33333	$PR_{(2,0,5)} = \frac{7!}{2!\,5!} = 7 \cdot 6/2 = 21$
5 . . .	5 44444 *ii)	$PR_{(1,5,0)} = \frac{6!}{5!} = 6$
	5 44 3333	$PR_{(1,2,4)} = \frac{7!}{2!\,4!} = 7 \cdot 6 \cdot 5/2 = 105$
4444 333		$PR_{(0,4,3)} = \frac{7!}{4!\,3!} = 7 \cdot 6 \cdot 5/3! = 35$
4 3333 333		$PR_{(0,1,7)} = \frac{8!}{7!} = 8$

Aplicant el **principi de la suma** i el **principi del producte** a cada un dels 8 casos, igual que en l'apartat anterior s'arriba a

$$S_e = S_a \times (1 + 60 + 60 + 21 + 6 + 105 + 35 + 8) = 296\,S_a = 2\,953\,843\,200$$

Problema 1.4:[futbetDutxes] 10 amics juguen a futbet els divendres.

a) De quantes maneres es podrien distribuir en 2 equips de 5?

b) Si hi ha 7 dels 10 amics que no estan disposats a jugar de porter, de quantes maneres es podran fer els equips?

c) I si dels 3 possibles porters n'hi ha 2 que no volen jugar en el mateix equip, de quantes maneres es podran distribuir?

d) I què passaria si els 2 que no volen jugar són dels 7 que no volen ser porters?

Dels 10 amics n'hi ha 6 que en acabar es dutxen al mateix temps en alguna de les dues dutxes col·lectives, C1 i C2, que hi ha al pavelló.

e) De quantes maneres es poden dutxar?

f) En el mateix pavelló construiran 3 dutxes més, però individuals (I1, I2, I3). De quantes maneres es podran dutxar els 6 (al mateix temps)?

g) I si, en lloc de 3 dutxes individuals, en construïren una triple (T1), on podrien cabre fins a tres persones?

a) Són 10 amics. I en total hi ha $|C_{10}^5| = \binom{10}{5}$ subconjunts diferents de grandària 5.

Una vegada format un possible equip, la resta d'amics fins a 10 (el complementari) formen l'altre.

Però cada possible partició en 2 equips es comptarà dues vegades, ja que cada equip i el seu complementari formen part dels $\binom{10}{5}$ subconjunts. Per tant, en aplicar el **principi de la divisió**, el cardinal del conjunt de totes les possibles particions vàlides, S_{10}, és determinat per

$$S_{10} = \frac{\binom{10}{5}}{2} = \frac{10^{2} \cdot 9 \cdot 8 \cdot 7 \cdot 6}{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} \cdot \frac{1}{2} = 9 \cdot 7 \cdot 2 = \boxed{126}.$$

b) Només hi ha, doncs, 3 possibles porters. I hi ha d'haver un porter en cada equip. Per tant, aquests 3 amics s'han de repartir entre els 2 equips i això es pot fer de $\binom{3}{2} = \binom{3}{1} = 3$ maneres possibles.

A banda, els 7 amics romanents s'han de distribuir entre els dos equips: 4 amb el porter que està sol i 3 amb els 2 porters. Ara no hi ha confusió ni repetició perquè les grandàries són diferents, i aquesta quantitat és $\binom{7}{4} = \binom{7}{3}$.

En aplicar el **principi de la multiplicació**, ja que la selecció dels porters és independent de la dels jugadors no porters, el nombre de possibles particions en aquest cas és determinat per

$$S_{(7,3)} = \binom{3}{1} \times \binom{7}{3} = 3 \cdot \frac{7 \cdot 6 \cdot 5}{3 \cdot 2 \cdot 1} = 3 \cdot 35 = \boxed{105}.$$

Alternativament, també haguérem pogut comptar les particions impossibles en què els 3 possibles porters estigueren junts en el mateix equip. Aquesta quantitat seria

$$S_3 = \binom{7}{5} = \binom{7}{2} = \frac{7 \cdot 6}{2!} = 21,$$

per la qual cosa les particions possibles s'obtindrien a partir de S_{10} i del **principi de la suma** com a

$$S_{(7,3)} = S_{10} - S_3 = \boxed{105}.$$

c) En el cas en què 2 dels possibles porters no vulguen jugar junts, les $\binom{3}{1}$ possibilitats de repartir els porters de l'apartat anterior queden reduïdes a 2 (amb qui dels 2 s'ajunta el tercer porter).

A partir d'ací el càlcul és exactament el mateix.

$$S_{(7,2,1)} = 2 \times \binom{7}{3} = 2 \cdot 35 = \boxed{70}.$$

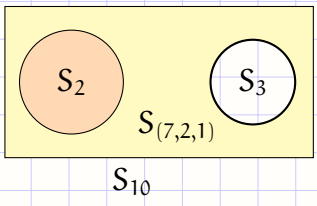
Alternativament, podríem calcular quantes de les particions en $S_{(7,3)}$ tenen junts els dos porters incompatibles (i l'altre porter en l'altre equip). Això correspon a fixar els porters i repartir els restants i seria

$$S_2 = \binom{7}{4} = \binom{7}{3} = 35.$$

Per tant, a partir de $S_{(7,3)}$ i en aplicar el **principi de la suma** es tindria que

$$S_{(7,2,1)} = S_{(7,3)} - S_2 = 105 - 35 = \boxed{70}.$$

El conjunt de totes les particions dels 10 amics en dos equips que estan involucrades en els apartats anteriors es pot representar gràficament en el següent diagrama de Venn.



La part en color es correspon amb $S_{(7,3)} = S_{(7,2,1)} \cup S_2$, i els tres subconjunts representats dins de S_{10} són subconjunts disjunts. (Estem abusant de la notació i referint-nos indistintament a conjunts i els seus cardinals).

d) Si els 2 que no volen jugar junts són dels no porters, tornem al raonament de l'apartat b). Les $\binom{3}{1}$ maneres de repartir els porters s'han de combinar ara amb les $|P_2| = 2$ maneres de repartir aquests 2 amics i finalment amb les maneres de distribuir els romanents (3 a un equip i 2 a l'altre).

$$S_{(5,2,3)} = \binom{3}{1} \times 2! \times \binom{5}{2} = 3 \cdot 2 \cdot \frac{5 \cdot 4}{2 \cdot 1} = \boxed{60}.$$

La identificació de les configuracions prohibides del càlcula alternatiu resulta ara un poc més complexa. D'una banda hi ha $\binom{3}{1}$ maneres de repartir els porters, però els 2 jugadors poden anar o be amb l'equip de 2 porters (amb la qual cosa faltaria repartir 1 i 4 en cada equip) o be amb l'equip que només en té 1 (amb la qual cosa faltaria repartir-ne 3 i 2). Aleshores i mitjançant el **principi de la suma** es té que

$$S_2' = \binom{3}{1}\left[\binom{5}{1} + \binom{5}{2}\right] = 45.$$

En aplicar novament el **principi de la suma** com en els casos anteriors s'arriba doncs a

$$S_{(5,2,3)} = S_{(7,3)} - S_2' = 105 - 45 = \boxed{60}.$$

e) Tal com es fa referència a les dutxes col·lectives, hauria de quedar clar que les dues dutxes **són distingibles** i de **capacitat il·limitada**.

Pert tant, cada un dels 6 amics que es dutxen només ha de decidir en quina de les dues dutxes col·lectives disponibles entra. O siga,

$$C_6 = |VR_2^6| = 2^6 = \boxed{64}.$$

f) Si hi haguera 3 dutxes individuals i **distingibles**, cada un dels 6 podria decidir utilitzar-ne alguna o no. Però sempre que no estiguera ja ocupada.

Per tant, caldrà considerar casos quant a l'ocupació de les dutxes individuals: Si ningú es dutxa en les individuals, $O_0 = \binom{3}{0} = 1$. Si només se n'utilitza una d'individual, hi ha 3 possibilitats, $O_1 = \binom{3}{1} = 3$. Si se n'utilitzen dues, $O_2 = \binom{3}{2} = 3$. I en el cas que s'ompliren les tres, $O_3 = \binom{3}{3} = 1$.

En resum, les diferents possibilitats d'ocupar les dutxes individuals es poden representar agrupades en quatre casos com s'indica en la següent taula.

i	I1 I2 I3	O_i
0	000	C_3^0
1	001 010 100	C_3^1
2	011 101 110	C_3^2
3	111	C_3^3

I es compleix que

$$\sum_{i=0}^{3} O_i = \sum_{i=0}^{3} |C_3^i| = \sum_{i=0}^{3} \binom{3}{i} = 2^3 = |VR_2^3| = 8.$$

Ara caldrà analitzar de manera independent per a cada cas de quantes maneres es poden omplir les dutxes individuals i les col·lectives.

En el cas O_0 només hi ha una manera d'ocupar les individuals, $I_0 = 1$, els 6 amics van a les col·lectives i es dutxarien de C_6 maneres possibles tal i com s'ha calculat en l'apartat anterior.

En el cas O_1, un dels 6 amics ocuparia la individual (de $I_1 = 6$ maneres perquè hi ha 6 possibles amics) i els 5 romanents anirien a les col·lectives de $C_5 = |VR_2^5|$ maneres possibles.

El mateix passaria en el cas O_2 però ara hi hauria

$$I_2 = |V_6^2| = 6^{\underline{2}} = 30$$

maneres possibles d'ocupar les dues individuals i els 4 romanents anirien a les col·lectives de $C_4 = |VR_2^4|$ maneres possibles.

Per últim, en el cas O_3, les individuals s'omplirien també de $6^{\underline{3}} = 120$ maneres possibles i les col·lectives de $C_3 = |VR_2^3|$ maneres possibles.

Podem completar ara la taula anterior amb tota la informació.

casos	individuals			col·lectives
	dutxes		persones	
i	I1 I2 I3	O_i	I_i	C_i
		C_3^i	V_6^i	VR_2^{6-i}
0	000	$\binom{3}{0}=1$	$6^{\underline{0}}=1$	2^6
1	001 010 100	$\binom{3}{1}=3$	$6^{\underline{1}}=6$	2^5
2	011 101 110	$\binom{3}{2}=3$	$6^{\underline{2}}=30$	2^4
3	111	$\binom{3}{3}=1$	$6^{\underline{3}}=120$	2^3

Per a obtenir el resultat final caldrà sumar els 4 possibles casos d'ocupació de les individuals (**principi de la suma**) i en cada cas multiplicar les possibilitats d'ocupació per les formes d'omplir les individuals, i també per les formes d'omplir les col·lectives (**principi del producte**).

$$\sum_{i=0}^{3} O_i \cdot I_i \cdot C_{6-i} = \sum_{i=0}^{3} |C_i^3| \cdot |V_6^i| \cdot |VR_2^{6-i}| = \sum_{i=0}^{3} \binom{3}{i} \cdot 6^{\underline{i}} \cdot 2^{6-i} =$$

$$= \underbrace{1 \cdot 1 \cdot 2^6}_{64} + \underbrace{3 \cdot 6 \cdot 2^5}_{576} + \underbrace{3 \cdot 6 \cdot 5 \cdot 2^4}_{1440} + \underbrace{1 \cdot 6 \cdot 5 \cdot 4 \cdot 2^3}_{960} =$$

$$= (1+9)2^6 + (90+60)2^4 = 640 + 880 = \boxed{3040}.$$

g) Si canviem les 3 dutxes individuals (i **distingibles**) per una única dutxa triple (o equivalentment 3 dutxes individuals **indistingibles**), l'anàlisi per casos anterior continua sent vàlida però ara només hi ha una forma d'ocupació. És a dir, $O_i = 1$ per a tot i. I la forma d'elegir les persones que van a la triple vindrà donada per **combinacions sense repetició** en lloc de **variacions sense repetició**. És a dir, $I_i = |C_6^i|$ per a tot i.

Tot plegat, el nombre total de possibilitats seria ara

$$\sum_{i=0}^{3} I_i \cdot C_{6-i} = \sum_{i=0}^{3} 1 \cdot \binom{6}{i} \cdot 2^{6-i} = \underbrace{1 \cdot 2^6}_{64} + \underbrace{6 \cdot 2^5}_{192} + \underbrace{15 \cdot 2^4}_{240} + \underbrace{20 \cdot 2^3}_{160} = \boxed{656}.$$

Problema 1.5:[gabiesAnimals] Una botiga d'animals té 5 gàbies numerades i grans per a ocells. Suposant que no sabem distingir 2 ocells de la mateixa espècie,

a) de quantes maneres es poden distribuir 1 lloro, 1 cotorra, i un tucà?

b) I si foren 2 lloros, 3 cotorres i 1 tucà?

c) I 7 tucans?

Si ens diuen que les gàbies són grans és perquè hi cabran tots els ocells que vulguem posar-hi. I com que estan numerades, això significa que són **distingibles**.

També és important remarcar que sabem distingir espècies diferents, però no diferents individus de la mateixa espècie.

a) En aquest cas tenim 3 ocells **distingibles** (perquè són d'espècies diferents) cada un dels quals pot estar en qualsevol de les 5 gàbies.

El lloro el podem col·locar en qualsevol de les 5 gàbies. Això són 5 possibilitats.

De la mateixa manera, tenim altres 5 possibilitats per a col·locar la cotorra i 5 més per al tucà.

I en aplicar el **principi del producte** arribaríem a un total de $5 \cdot 5 \cdot 5 = 125$ possibilitats per a distribuir els tres.

De manera equivalent, podem pensar que a cada un dels 3 ocells li hem d'assignar un d'entre 5 nombres possibles. O siga, **variacions amb repetició de 5 elements agafats de 3 en 3**, VR_5^3.

$$S_a = |VR_5^3| = 5^3 = \boxed{125}.$$

c) Respondrem primer a aquest apartat que és més senzill.

La diferència fonamental respecte al primer apartat, a banda que són 7 ocells en lloc de 3, és que ara són ocells **indistingibles**.

Distribuir n ocells (distingibles) en m gàbies numerades ja hem vist en l'apartat anterior que es correspon amb variacions amb repetició de m elements agafats de n en n, VR_m^n.

Però si els ocells són indistingibles, el problema equival a distribuir n boles iguals en m caixes numerades. O també, a considerar multiconjunts de m gàbies de cardinalitat n. És a dir, **combinacions amb repetició de m elements agafats de n en n**.

En el nostre cas particular, tenim que

$$S_c = CR_m^n = CR_5^7 = \left(\!\!\binom{5}{7}\!\!\right) = C_{5+7-1}^7 = \binom{11}{7} = \frac{11\cdot 10\cdot 9\cdot \cancel{8}}{\cancel{4}\cdot 3\cdot \cancel{2}\cdot 1} = \boxed{330}.$$

b) Si ara cal distribuir n_1 ocells d'una espècie i n_2 ocells d'una altra, podem fer-ho **independentment** i aplicar el **principi del producte**.

Aleshores, les diferents maneres de repartir els 2 lloros, les 3 cotorres i el tucà es poden calcular com a

$$S_b = CR_5^2 \times CR_5^3 \times CR_5^1 = \left(\!\!\binom{5}{2}\!\!\right) \times \left(\!\!\binom{5}{3}\!\!\right) \times \left(\!\!\binom{5}{1}\!\!\right) = \binom{6}{2} \times \binom{7}{3} \times \binom{5}{1} = 15\cdot 35\cdot 5 = \boxed{2625}.$$

Podem veure que d'aquesta manera obtindríem també el resultat del primer apartat on teníem un lloro, una cotorra i un tucà, i per tant

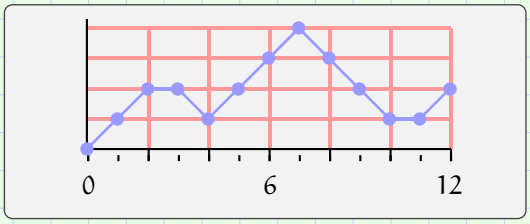
$$S_a = CR_5^1 \times CR_5^1 \times CR_5^1 = \boxed{125}.$$

Problema 1.6: [senderisme] Considera rutes de senderisme en què a cada quilòmetre es pot pujar o baixar 10 m o planar (no canviar d'altitud).

a) Quants perfils de ruta diferents de 12 km pot haver-hi amb desnivell positiu acumulat (suma dels metres pujats) de 60 m?

b) I quants de 12 km amb desnivell positiu i negatiu acumulats de 60 i 40m, respectivament?

Com a exemple, en la figura es mostra un perfil de 12 km amb desnivells positiu i negatiu acumulats de 60 i 40 metres, respectivament.



Des del punt de vista del seu perfil, podem representar les diferents rutes de m quilòmetres com a seqüències de longitud m formades per elements del conjunt $\{+1,-1,0\}$ (pujar, baixar i planar).

Per exemple, la ruta de la figura de l'exemple es podria representar com a

$$(+1,+1,0,-1,+1,+1,+1,-1,-1,-1,0,+1).$$

El nombre total de rutes de longitud m (en km), S_m, seria aleshores determinat per les **variacions amb repetició de 3 elements agafats de m en m**, VR_3^m.

Per tant tenim en el cas de l'exemple que

$$S_{12} = VR_3^{12} = 3^{12}.$$

a) Si dels m quilòmetres de la ruta, n han de ser de pujada, la qual cosa implica un desnivell positiu acumulat de $10n$ metres, això significa que els restants $m-n$ quilòmetres de la ruta han de ser neutres o de baixada.

Aleshores, d'una banda haurem de decidir de quantes maneres es poden col·locar els quilòmetres de pujada dins de la ruta.

I d'altra banda haurem de decidir si cada un dels restants quilòmetres és de baixada o neutre.

La quantitat de casos total serà, doncs, determinada pel **principi del producte**.

Els n quilòmetres de pujada dins dels m de la ruta es poden elegir de $\binom{m}{n}$ maneres. O, en altres paraules, es corresponen amb **combinacions de m elements agafats de n en n**.

I les maneres possibles d'etiquetar els $m-n$ romanents com a baixar o planar es correspon amb les **variacions amb repetició de 2 elements agafats de $m-n$ en $m-n$**.

El nombre total de rutes de m quilòmetres amb desnivell acumulat de $10n$ metres serà, doncs,

$$S_{(m,n)} = C_m^n \times VR_2^{m-n} = \binom{m}{n} \cdot 2^{m-n}.$$

En particular, el nombre de rutes de 12 km amb 60 m de desnivell positiu acumulat serien

$$S_{(12,6)} = \binom{12}{6} \cdot 2^6 = \frac{\cancel{20}^{\,2} \cdot 19 \cdot \cancel{18}^{\,2} \cdot 17 \cdot \cancel{16}^{\,2} \cdot \cancel{15} \cdots \cancel{12}^{\,2} \cdot 11}{\cancel{10} \cdot \cancel{9} \cdots \cancel{6} \cdot \cancel{5} \cdot 4 \cdot \cancel{3} \cdot 2 \cdot 1} \; 2^6 =$$

$$= 19 \cdot 17 \cdot 13 \cdot 11 \cdot 2^8 = 46189 \cdot 256 = \boxed{11\,824\,384}.$$

b) Ara en les rutes de m quilòmetres, n han de ser de pujada i k han de ser de baixada per a tenir desnivells positiu i negatiu acumulats de $10n$ i $10k$, respectivament.

Per a comptar totes les possibilitats, hem de seleccionar ara k quilòmetres de baixada d'entre els $m-n$ que no són de pujada.

I aplicarem el **principi del producte** igual que abans, amb la qual cosa el nombre total de rutes de m quilòmetres amb $10n$ metres i $10k$ metres de pujada i baixada acumulada seran

$$S_{(m,n,k)} = C_m^n \times C_{m-n}^k = \binom{m}{n} \cdot \binom{m-n}{k}.$$

En particular, el nombre de rutes de 12 km amb 60 m i 40 m de desnivells positiu i negatiu acumulats serien

$$S_{(12,6,4)} = \binom{12}{6} \cdot \binom{6}{4} = \frac{\cancel{12}\cdot 11\cdot 10\cdot 9\cdot \cancel{8}^{\,2}\cdot 7}{6!} \frac{6!}{\cancel{2!}\;\cancel{4}\cdot 3\cdot \cancel{2}\cdot 1} = 11\cdot 10\cdot 9\cdot 7\cdot 2 = \boxed{13\,860}.$$

Problema 1.7:[meitatsSuccessives] Intenta trobar un argument visual per a calcular el següent sumatori en què cada terme no és un nombre enter sinó una fracció de la

unitat.

$$\sum_{i=1}^{n} \frac{1}{2^i} = \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^n}$$

Potser la forma més senzilla d'obtenir un resultat per a sumatori de l'enunciat és relacionar-lo amb la suma dels dobles successius, $S_d(n)$, que s'ha vist en la pàgina 15.

Primer anomenem $S_m(n)$ el resultat, ho multipliquem tot per 2^n i obtenim

$$2^n S_m(n) = \sum_{i=1}^{n} 2^{n-i} = \sum_{j=0}^{n-1} 2^j + 2^n - 2^n = S_d(n) - 2^n,$$

on hem fet un canvi d'índex, hem sumat i restat 2^n i hem introduït $S_d(n)$, amb la qual cosa arribem a

$$2^n S_m(n) = 2 \cdot 2^n - 1 - 2^n,$$

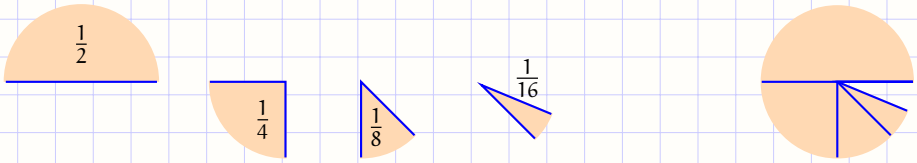
des d'on s'obté que el resultat del sumatori és

$$S_m(n) = 1 - \frac{1}{2^n}.$$

No obstant això, hi ha un argument visual molt senzill que permet obtenir fàcilment el mateix resultat.

Primer pensem en la unitat (una unitat de qualsevol cosa) com un cercle. Pensem, per exemple, en una pizza.

El nostre sumatori consisteix a sumar la meitat de la unitat (de la pizza) amb la meitat de la meitat, i amb la meitat de la meitat de la meitat, i així successivament. De manera gràfica,



La figura il·lustra el sumatori amb quatre termes. Si observem el dibuix de la dreta on estan "sumades" totes les porcions veiem que el total és igual a la unitat **menys** una porció que és exactament igual que l'últim terme del sumatori. És a dir, en el cas de la figura tindríem

$$1 - \frac{1}{16}.$$

I aquesta relació és certa per a qualsevol nombre de termes en el sumatori ja que cada nou terme que s'afegeix ocupa la meitat del romanent i el nou romanent acaba sent també de la mateixa grandària.

Per tant podem deduir l'expressió correcta per al sumatori,

$$\sum_{i=1}^{n} \frac{1}{2^i} = 1 - \frac{1}{2^n}.$$

1.6 Problemes proposats

Problema 1.8:[progressioAritmetica] El sumatori $S_1(n)$ de la pàgina 14 és un cas particular de suma dels termes d'una progressió aritmètica. Una seqüència de valors, $(a_1, a_2, \ldots, a_n)$, diem que és una **progressió aritmètica** si per a tot $i = 2, \ldots, n$ es compleix que

$$a_i = a_{i-1} + d,$$

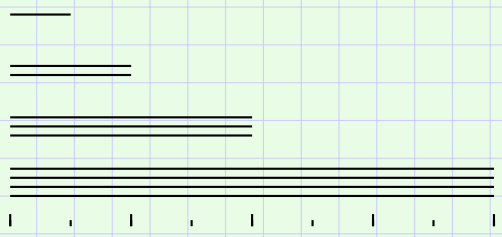
on d és l'anomenada **diferència** entre dos termes qualssevol de la progressió.
Estén l'argument visual utilitzat per obtenir la solució de $S_1(n)$ per a la suma de qualsevol progressió aritmètica.

La solució és

$$\frac{n}{2}(a_1 + a_n) = n\left(a_1 + \frac{n-1}{2}d\right).$$

Problema 1.9:[mesDoblesSuccessius] Intenta trobar un argument visual per calcular el següent sumatori en què cada terme es correspon amb una quantitat que va duplicant-se multiplicada per un factor (enter) que creix amb l'índex del sumatori. En el dibuix que s'adjunta com a ajut, cada terme del sumatori es representa com a i barres horitzontals de longitud 2^{i-1}.

$$\sum_{i=1}^{n} i2^{i-1} = 1 + 2 \cdot 2 + 3 \cdot 4 + 4 \cdot 8 + \cdots + n \cdot 2^{n-1}.$$



La solució és

$$n2^n - 2^n + 1.$$

Problema 1.10:[flannagan]
Flannagan és un jugador trampós que sempre té un as de cors en la mànega i l'habilitat d'intercanviar-lo per qualsevol de les 5 cartes de la seua mà sense que ningú se n'adone. Recorda que a la baralla hi ha 13 valors i 4 pals.

a) De quantes maneres diferents pot obtenir un pòquer (4 cartes amb el mateix valor)?

b) I un *full* (3 cartes amb un valor i 2 amb un altre)?

c) Calcula els dos apartats anteriors per a un jugador honrat.

Les solucions per a cada un dels apartats són:

a) $624 + 1128 = 1752$,

b) $3744 + 15840 = 19584$,

c_a) 624,

c_b) 3744.

Problema 1.11:[discretaliaMatricules] Els Estats Units de Discretàlia (EUD) han decidit utilitzar per als cotxes matrícules de 5 dígits d'entre els 10 possibles: 0123456789. Però cada estat ha afegit condicions a les seues matrícules:

a) A Variolina del Nord (VN) han decidit usar matrícules que tinguen un màxim de 3 dígits diferents. (consell: prova primer amb 2).

b) A Nova Vèrtex (NV) només volen matrícules capicues.

c) A Grafifòrnia (GF) exigeixen matrícules sense zeros a l'esquerra.

d) A Arcansas (o Aristansas, AR) acceptaran matrícules que siguen vàlides a Nova Vèrtex o a Grafifòrnia.

e) A Ojaio (o Disjuncijaio, OJ) només accepten matrícules que siguen vàlides a Nova Vèrtex i a Grafifòrnia (curiosament).

f) A Existàlia (EX) no es posa cap condició.

g) I a Connexicut (CX), aprofitant que els dígits 0, 6, 8 i 9 es poden girar 180 graus, només accepten matrícules que siguen *girables* sense que canvie el número.

Digues quantes matrícules diferents pot haver-hi en cada estat, raonant amb detall l'obtenció de les respostes.
A continuació es mostren alguns exemples de matrícules vàlides i no valides en cada estat.

	a) VN	b) NV	c) GF	d) AR	e) OJ	f) EX	g) CX
si	E-13233	E-13531	E-73210	b∨c	b∧c	-	E-96896 = 96896-∃
no	E-13244	E-11112	E-00700			-	E-00088 ≠ 88000-∃

Les solucions per a cada un dels apartats són:

a) VN	b) NV	c) GF	d) AR	e) OJ	f) EX	g) CX
19360	1000	90000	90100	900	100000	32

Problema 1.12: [futbol]
Suposem un equip de futbol que juga en tres línies. És a dir, amb 4 **defenses** (2-5), 4 **migcampistes** (6-9) i 2 **davanters** (10-11). Anomenem, camí directe a gol una jugada que parteix del porter (1) i passa per un jugador de cada una de les tres línies i acaba en gol.

a) Quants camins directes a gol diferents poden haver-hi?

b) De quantes maneres es poden distribuir els 10 jugadors de camp en les 3 línies si no importa quina posició ocupen dins de la línea?

c) I si sí que importa?

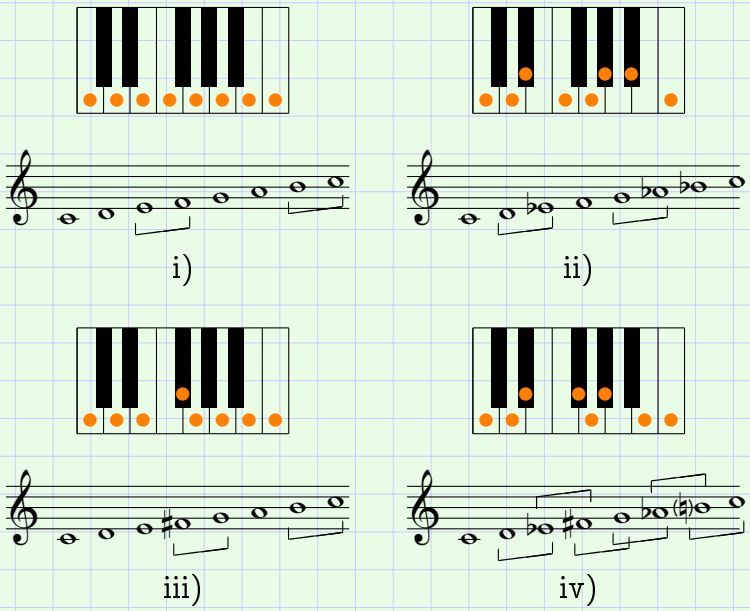
d) Aquesta forma de jugar s'anomena 1-4-4-2. Quantes altres formes de jugar poden haver amb 3 línies (no buides)?

e) I si no restringim el nombre de línies, però no considerem línies amb menys de 2 jugadors?

Les solucions per a cada un dels apartats són:

a)	b)	c)	d)	e)
32	3150	3628800	36	34

Problema 1.13:[escales]
Definim una escala musical heptatònica simple com una seqüència de 7+1 notes separades per un interval de **segona major** (2 semitons) o de **segona menor** (1 semitò) i que cobreixen exactament una octava.
Una octava conté 13 notes i 12 semitons que les separen: els 4 dibuixos de tecles de piano mostren les 13 notes consecutives, separades per semitons que cobreixen l'octava que comença en Do i acaba en Do.
En els 3 primers apartats de la figura hi ha 3 exemples d'escales amb notació musical i marcades sobre el teclat amb cercles. L'interval d'una nota a la següent pot ser d'1 o de 2 semitons (en notació musical es marquen per sota les segones menors). Com a curiositat, les escales mostrades són i) l'escala major, ii) l'escala menor, iii) l'escala Lídia.
L'apartat iv) mostra l'escala menor oriental que no seria vàlida (segons la definició anterior) perquè conté 2 intervals de **segona augmentada** (3 semitons), que es marquen per dalt.



a) Quantes escales heptatòniques simples diferents es poden formar a partir d'una nota?

b) I si considerem els 3 tipus de segones com en l'exemple iv)?

c) Què passaria si consideràrem **qualssevol** intervals?

d) I si considerem escales pentatòniques i/o hexatòniques (5 i/o 6 notes, respectivament)?

Les solucions per a cada un dels apartats són:

a)	b)	c)	d)	
21	266	462	330	462

Problema 1.14:[comptaInclusionsSubconj]
Considera un conjunt A de 4 elements qualssevol i el seu conjunt potència 2^A (tots els seus possibles subconjunts). Considera també la relació binària, R, definida per a tot $\alpha, \beta \subseteq A$ com

$$\alpha R \beta \quad \textbf{sii} \quad \alpha \subset \beta, \neg\exists \gamma \subseteq A : \alpha \subset \gamma \subset \beta$$

Alguns exemples de la relació R són

$$\{a_1, a_3\} R \{a_1, a_2, a_3\}, \quad \{a_1, a_3\} R A, \quad \{a_1, a_3\} \not R \{a_1, a_3\}, \quad \{a_1, a_3\} \not R \{a_1, a_2, a_4\}.$$

a) Quin és el cardinal de R? O en altres paraules, quin és el nombre de parells d'elements de A que estan relacionats segons R?

b) Quin és el cardinal de la relació $\subseteq$?

c) Generalitza els resultats anteriors per a conjunts A de cardinalitat n.

Les solucions per a cada un dels apartats són:

a)	b)	c)	
42	81	$n2^{n-1}$	3^n

■

2. Lògica

2.1 Proposicions i equivalències

Una **proposició** és una afirmació, un fet, o qualsevol altra frase que pot ser **certa** o **falsa**. Per exemple, "els rucs volen".

Els valors **vertader**, V, i **fals**, F, són les (úniques) **constants en lògica proposicional**.

Farem servir **variables proposicionals** per a referir-nos a proposicions determinades. Les escriurem normalment $p, q, r, \ldots$

És possible construir noves proposicions fent servir **connectives** o **operadors lògics**. En particular, considerarem la **negació**, $\neg$, la **conjunció**, $\wedge$, la **disjunció**, $\vee$, la **implicació**, $\Rightarrow$, i la **coimplicació**, $\Leftrightarrow$.

Definició recursiva

1. Vertader (V) i Fals (F) són proposicions.
2. Qualssevol variables proposicionals, $p, q, r, \ldots$ són proposicions.
3. Si p és proposició, $\neg p$ també ho és.
4. Si p i q són proposicions, $p \wedge q$, $p \vee q$, $p \Rightarrow q$ i $p \Leftrightarrow q$, també ho són.

Les proposicions definides només segons 1 i 2 s'anomenen **proposicions simples**. La resta de proposicions s'anomenen **proposicions compostes** o també **expressions (proposicionals)**.

En una **expressió** amb diversos operadors, sempre s'interpreten aquests d'esquerra a dreta i segons l'ordre de preferència o **prioritat** donat per la seqüència $(\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow)$.

Si es vol canviar l'ordre d'actuació dels operadors en una expressió, es poden

utilitzar els parèntesis. Per exemple, les dues expressions següents són equivalents.

$$p \vee q \vee \neg r \Rightarrow r \wedge p \vee r \Leftrightarrow q \qquad\qquad (((p \vee q) \vee (\neg r)) \Rightarrow ((r \wedge p) \vee r)) \Leftrightarrow q$$

En la pràctica escriurem algunes altres connectives lògiques tal i com es recull en la següent taula.

connectiva	equivalència	nom
$p \Leftarrow q$	$q \Rightarrow p$	implicació oposada o recíproca
$p \not\Leftrightarrow q$	$\neg(p \Leftrightarrow q)$	**disjunció exclusiva**
$p \uparrow q$	$\neg(p \wedge q)$	conjunció oposada (operador **NAND**)
$p \downarrow q$	$\neg(p \vee q)$	disjunció oposada (operador **NOR**)

A una expressió amb n variables diferents, li corresponen $|V_2^n| = 2^n$ possibles valors de veritat en funció que cada una de les variables siga vertadera o falsa. Aquesta informació en forma de taula rep el nom de **taula de veritat** de l'expressió.

Les taules de veritat corresponents a les connectives bàsiques en lògica proposicional són:

p	q	$\neg q$	$p \wedge q$	$p \vee q$	$p \Rightarrow q$	$p \Leftrightarrow q$
V	V	F	V	V	V	V
V	F	V	F	V	F	F
F	V		F	V	V	F
F	F		F	F	V	V

Diem que dues expressions, e_1 i e_2, són **equivalents** si tenen la mateixa taula de veritat. S'escriu

$$e_1 \equiv e_2.$$

Una expressió que sempre és vertadera s'anomena **tautologia**. Una expressió que sempre és falsa s'anomena **contradicció**.

2.1.1 Equivalències amb disjuncions i conjuncions

Llei del tercer exclòs

$$p \vee \neg p \equiv V$$

Llei de contradicció

$$p \wedge \neg p \equiv F$$

Lleis d'identitat (elements neutres)

$$p \vee F \equiv p \qquad p \wedge V \equiv p$$

Lleis de dominació

$$p \vee V \equiv V \qquad p \wedge F \equiv F$$

Lleis d'idempotència

$$p \vee p \equiv p \qquad p \wedge p \equiv p$$

Llei de doble negació

$$\neg\neg p \equiv \neg(\neg p) \equiv p$$

Lleis commutatives

$$p \vee q \equiv q \vee p \qquad p \wedge q \equiv q \wedge p$$

Lleis associatives

$$(p \vee q) \vee r \equiv p \vee (q \vee r) \qquad (p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$$

Lleis distributives (factor comú)

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r) \qquad p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$$

Lleis de De Morgan

$$\neg(p \vee q) \equiv \neg p \wedge \neg q \qquad \neg(p \wedge q) \equiv \neg p \vee \neg q$$

Lleis d'absorció de la conjunció/disjunció

$$(p \vee q) \wedge p \equiv p \qquad (p \wedge q) \vee p \equiv p$$

Una variable proposicional o la seua negació s'anomena **literal**. Una **clàusula disjuntiva** o simplement **clàusula** és qualsevol disjunció de literals.

$$(\ell_1 \vee \ell_2 \vee \cdots \vee \ell_k)$$

Anàlogament, una **clàusula conjuntiva** és qualsevol conjunció de literals.

$$(\ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_k)$$

Un tipus de clàusules que seran importants més endavant són les anomenades **clàusules de Horn** que són aquelles disjuncions de literals on no pot haver-hi més d'un literal no negat.

$$\neg p \vee q \vee \neg r \qquad\qquad \neg p \vee \neg q$$

2.1.2 Equivalències bàsiques amb implicacions

Definició de la implicació

$$p \Rightarrow q \equiv \neg p \vee q$$

Definició de la coimplicació

$$p \Leftrightarrow q \equiv (p \Rightarrow q) \wedge (q \Rightarrow p)$$

Contraposició lògica

$$p \Rightarrow q \equiv \neg q \Rightarrow \neg p$$

2.1.3 Algunes propietats interessants de les implicacions

Distributivitat per l'esquerra respecte de la disjunció

$$p \Rightarrow (q \vee r) \equiv (p \Rightarrow q) \vee (p \Rightarrow r)$$

$$p \Rightarrow (q \vee r) \overset{\text{(def.)}}{\equiv} \neg p \vee (q \vee r) \overset{\text{(idemp.)}}{\equiv} \neg p \vee q \vee \neg p \vee r \overset{\text{(def.)}}{\equiv} (p \Rightarrow q) \vee (p \Rightarrow r)$$

Distributivitat per l'esquerra respecte de la conjunció

$$p \Rightarrow (q \wedge r) \equiv (p \Rightarrow q) \wedge (p \Rightarrow r)$$

$$p \Rightarrow (q \wedge r) \overset{(\text{def.})}{\equiv} \neg p \vee (q \wedge r) \overset{(\text{distr.})}{\equiv} (\neg p \vee q) \wedge (\neg p \vee r) \overset{(\text{def.})}{\equiv} (p \Rightarrow q) \wedge (p \Rightarrow r)$$

Pseudodistributivitat per la dreta respecte de la disjunció/conjunció

Es pot traure factor comú per la dreta però canviant conjuncions per disjuncions.

$$(p \vee q) \Rightarrow r \equiv (p \Rightarrow r) \wedge (q \Rightarrow r)$$

$$\begin{aligned}(p \vee q) \Rightarrow r &\overset{(\text{def.})}{\equiv} \neg(p \vee q) \vee r \overset{(\text{Morgan})}{\equiv} (\neg p \wedge \neg q) \vee r \overset{(\text{distr.})}{\equiv} \\ &\equiv (\neg p \vee r) \wedge (\neg q \vee r) \overset{(\text{def.})}{\equiv} (p \Rightarrow r) \wedge (q \Rightarrow r)\end{aligned}$$

Pseudodistributivitat per la dreta respecte de la conjunció/disjunció

Es pot traure factor comú per la dreta però canviant disjuncions per conjuncions.

$$(p \wedge q) \Rightarrow r \equiv (p \Rightarrow r) \vee (q \Rightarrow r)$$

$$\begin{aligned}(p \wedge q) \Rightarrow r &\overset{(\text{def.})}{\equiv} \neg(p \wedge q) \vee r \overset{(\text{Morgan})}{\equiv} (\neg p \vee \neg q) \vee r \overset{(\text{idemp.})}{\equiv} \\ &\equiv \neg p \vee r \vee \neg q \vee r \overset{(\text{def.})}{\equiv} (p \Rightarrow r) \vee (q \Rightarrow r)\end{aligned}$$

Les **clàusules de Horn** amb exactament un literal positiu (no negat) s'anomenen **clàusules definides** i es poden escriure com una implicació l'antecedent de la qual és una conjunció de literals positius. En altres paraules,

$$\begin{aligned}\neg p_1 \vee \cdots \vee \neg p_k \vee q \equiv (\neg p_1 \vee q) \vee \cdots \vee (\neg p_k \vee q) \equiv \\ \equiv (p_1 \Rightarrow q) \vee \cdots \vee (p_k \Rightarrow q) \overset{(\text{pdistr})}{\equiv} (p_1 \wedge \cdots \wedge p_k) \Rightarrow q.\end{aligned}$$

2.2 Implicacions, deduccions i inferència

En una implicació, $p \Rightarrow q$, la proposició a l'esquerra, p, rep el nom d'**antecedent** i la proposició a la dreta és el **conseqüent**.

És important adonar-se que la implicació sempre és certa si l'antecedent és fals.

Això vol dir que per a comprovar la veracitat d'una implicació, només cal plantejar-se el cas en què l'antecedent és vertader. Si en aquest cas el conseqüent també és vertader, la implicació serà certa.

Alternativament, podem veure el conseqüent com una proposició que és certa quan l'antecedent ho és. En altres paraules, podem deduir la veritat de q a partir de la veritat de p (sempre que siga cert que $p \Rightarrow q$).

En general, diem que una expressió, Q, es pot **deduir** a partir d'una altra expressió, P, si Q és certa quan P ho és. En aquest context, P rep el nom de **premissa** i Q s'anomena **conclusió semàntica** o simplement **conclusió**. Si P és una conjunció de dos o més subexpressions, $P = P_1, P_2, \ldots$, totes les P_i s'anomenen **premisses**.

Quan una conclusió, Q, es dedueix d'un conjunt de premisses, $P_1, P_2, \ldots, P_k$, escrivim

$$P_1, P_2, \ldots, P_k \vdash Q$$

Aquesta expressió s'anomena també **teorema**. El conjunt de premisses es denomina també **hipòtesi** i la conclusió s'anomena **tesi**. Del fet de comprovar que la conclusió es pot deduir a partir de les premisses se'n diu també **demostració**.

Si $P \Rightarrow Q$ aleshores $P \vdash Q$, i al revés

O altrament expressat,

$$P \vdash Q \text{ sii } P \Rightarrow Q$$

A partir de la taula de veritat de $P \Rightarrow Q$ és obvi que Q és cert si P ho és. I, per tant,

$$P \vdash Q.$$

De la mateixa manera, quan $P \vdash Q$, l'únic cas prohibit és que P siga vertader i Q fals. I això coincideix exactament amb la taula de veritat de $P \Rightarrow Q$.

2.2.1 Algunes propietats de $\vdash$

Propietat reflexiva

$$p \vdash p$$

Evident.

Propietat antisimètrica

$$p \equiv q \textbf{ si } p \vdash q \wedge q \vdash p$$

Si q ha de ser cert quan p ho és i al revés, necessàriament hauran de ser tots dos certs o tots dos falsos. És a dir,

$$p \equiv q.$$

Una conseqüència immediata de l'anterior i de la propietat que relaciona deducció i implicació és que

$$P \equiv Q \textbf{ sii } P \Leftrightarrow Q$$

Propietat transitiva

$$p \vdash r \textbf{ si } p \vdash q \wedge q \vdash r$$

Quan p és cert, q ho ha de ser. I r també, perquè $q \vdash r$. Aleshores serà també cert que

$$p \vdash r.$$

A conseqüència de les tres propietats anteriors, la relació $\vdash$ és una relació binària d'ordre dins de les expressions en lògica proposicional.

Fites inferior i superior

$$F \vdash p \vdash V$$

La demostració és òbvia a partir de la definició de $\vdash$.
Quan escrivim $p \vdash V$ estem dient que qualsevol premissa p és bona per a deduir una cosa que és certa independentment de p.
Quan escrivim $F \vdash p$ estem expressant que sempre es pot deduir qualsevol cosa quan la premissa no es compleix mai. Aquest cas concret s'anomena **inferència inconsistent** perquè en realitat no estem deduint res.

L'expressió $F \vdash p$ no s'hauria de confondre amb l'expressió $\vdash p$ que és sovint utilitzada per a indicar que p és una tautologia. O, equivalentment, que la veritat de

p es pot deduir a partir d'un conjunt buit de premisses.

De fet, la conjunció d'un conjunt buit de premisses és V (l'element neutre de $\wedge$) per la qual cosa les expressions $\vdash p$ i $V \vdash p$ són equivalents i signifiquen que $p \equiv V$.

$P \Rightarrow Q$ en llenguatge natural

Hi ha moltes maneres de referir-se a la implicació en la parla informal. Heus-ne ací algunes:

- si P aleshores/llavors Q.
- P implica Q.
- P, llavors/aleshores Q.
- quan P, Q.
- Q si P.
- no P o Q (només en alguns casos com per exemple: "No cantes o plourà")

$P \Leftarrow Q$ en llenguatge natural

També hi ha maneres informals d'expressar la implicació contrària:

- només si P aleshores Q.
- Q només si P.
- només quan P, Q.

$P \Leftrightarrow Q$ en llenguatge natural

De la mateixa manera, la coimplicació (en aquest cas sempre podem intercanviar P i Q):

- si i només si P aleshores Q.
- P si i només si Q.
- P **sii** Q.

2.2.2 Regles d'inferència estàndard

Hi ha alguns esquemes que permeten deduir expressions a partir de premisses amb una certa forma. Això és el que es coneix com a **regles d'inferència**. Algunes són més que òbvies, mentre que altres requereixen un poc més d'atenció.

Totes tenen el seu origen en la lògica clàssica i el seu objectiu és que es puguen combinar per a dur a terme deduccions més complexes.

Una manera alternativa i més gràfica d'expressar una regla $P_1, P_2, \ldots \vdash Q$ és

$$\begin{array}{c} P_1 \\ P_2 \\ \vdots \\ \hline Q \end{array}$$

És a dir, una línia per a cada una de les premisses i per a la conclusió. D'aquesta manera és fàcil aplicar noves regles usant premisses o conclusions anteriors. Només cal identificar clarament cada una de les línies perquè quede clar quina regla s'aplica i sobre quines premisses per a l'obtenció de cada nova línia.

EC. Eliminació de la conjunció

$$\frac{p \wedge q}{p} \qquad \frac{p \wedge q}{q}$$

Aquesta regla s'obté directament a partir de la taula de veritat de la conjunció.

IC. Introducció de la conjunció

$$\begin{array}{c} p \\ q \\ \hline p \wedge q \end{array}$$

Més que d'una regla, es tracta d'expressar el mateix de dues maneres diferents.

$$p, q \equiv p \wedge q$$

La deducció en el sentit contrari es pot obtenir molt fàcilment a partir de l'aplicació consecutiva de les dues regles **EC** sobre $p \wedge q$.

ID. Introducció de la disjunció

$$\frac{p}{p \vee q} \qquad \frac{q}{p \vee q}$$

Molt similar a la regla **EC**. També s'obté directament a partir de la taula de veritat de la disjunció.

EN. Eliminació de la negació

$$\frac{\neg\neg p}{p}$$

És en realitat una equivalència (llei de doble negació).

IN. Introducció de la negació, o també reducció a l'absurd

$$\frac{\begin{array}{c}[p]\\ \vdots \\ F\end{array}}{\neg p}$$

És una manera molt usual de raonar. Suposem certa una expressió, p, i ho indiquem mitjançant claudàtors. Si a partir d'ací aconseguim deduir F, és a dir, arribem a una contradicció, és perquè p ha de ser fals i aleshores $\neg p$ és cert.

ED. Eliminació de la disjunció, o també demostració per casos

$$\frac{\begin{array}{c}p \vee q\\ [p]\\ \vdots \\ r\\ [q]\\ \vdots \\ r\end{array}}{r}$$

És també molt usual i es pot estendre a disjuncions de tres o més proposicions. Si en suposar per separat cada una de les opcions en una disjunció, s'arriba a la mateixa conclusió, aleshores aquesta conclusió es pot deduir en general.

EI. Eliminació de la implicació, o també modus ponens

$$\frac{p \Rightarrow q \qquad p}{q}$$

És una de les formes més clàssiques de raonar. El nom complet és *modus ponendo ponens*: mode que en afirmar afirma.
Si una implicació és certa i també ho és el seu antecedent, aleshores ho serà també el conseqüent.

II. Introducció de la implicació, o també demostració de la implicació

$$\frac{\begin{array}{c}[p] \\ \vdots \\ q\end{array}}{p \Rightarrow q}$$

Si suposem cert p, i a partir d'això arribem a q aleshores serà cert que

$$p \Rightarrow q.$$

Les vuit regles d'inferència estàndard (introducció/eliminació de la negació/conjunció/disjunció/implicació) són les que farem servir per a dur a terme deduccions i demostracions més complexes.

No obstant això, hi ha altres regles d'inferència clàssiques tan famoses o més que les que nosaltres anomenem estàndard. A continuació n'enunciem algunes i les deduirem a partir de les estàndard.

modus tollens

$$\frac{p \Rightarrow q \qquad \neg q}{\neg p}$$

És com el modus ponens, però ara neguem el conseqüent per a demostrar la falsedat de l'antecedent. El nom complet és *modus tollendo tollens*: mode que en negar nega.
Demostrarem la deducció,

$$p \Rightarrow q, \neg q \vdash \neg p,$$

primer mitjançant regles estàndard:

1)	[p]	comença **IN**, suposem el contrari del que volem deduir ...
2)	$p \Rightarrow q$	... premissa 1
3)	q	... **EI** (modus ponens) en 1), 2)
4)	$\neg q$	... premissa 2
5)	$q \wedge \neg q \equiv F$	... **IC** en 3), 4)
6)	$\neg p$	acaba **IN** en 1)

Alternativament, podem aplicar la **contraposició lògica** (l'última de les equivalències notables en la pàgina 44) per a modificar la implicació i aleshores demostrar-ho només aplicant **EI** (modus ponens).

modus tollendo ponens o també sil·logisme disjuntiu

$$\frac{p \vee q \quad \neg p}{q} \qquad \frac{p \vee q \quad \neg q}{p}$$

modus tollendo ponens: mode que en negar afirma. Aquesta regla d'inferència s'il·lustra en l'exercici 2.6

modus ponendo tollens

$$\frac{\neg(p \wedge q) \quad p}{\neg q} \qquad \frac{\neg(p \wedge q) \quad q}{\neg p}$$

modus ponendo tollens: mode que en negar afirma. Aquesta regla es pot obtenir a partir de l'anterior només canviant cada proposició per la seua negació.

sil·logisme hipotètic

$$\frac{p \Rightarrow q \quad q \Rightarrow r}{p \Rightarrow r}$$

Aquesta regla d'inferència s'obté a partir de la transitivitat de la implicació. Utilitzant regles d'inferència seria:

1)	[p]	comença **II**, suposem l'antecedent
2)	$p \Rightarrow q$	... premissa 1
3)	q	... **EI** en 1), 2)
4)	$q \Rightarrow r$	... premissa 2
5)	r	... **EI** en 3), 4)
6)	$p \Rightarrow r$	acaba **II** en 1)

Llei d'absorció

$$\frac{p \Rightarrow q}{p \Rightarrow p \wedge q}$$

Aquesta regla d'inferència és en realitat una equivalència.

$$p \Rightarrow p \wedge q \overset{(1)}{\equiv} \neg p \vee (p \wedge q) \overset{(2)}{\equiv} (\neg p \vee p) \wedge (\neg p \vee q) \overset{(3)}{\equiv} V \wedge (p \Rightarrow q) \overset{(4)}{\equiv} p \Rightarrow q$$

En cada pas s'ha aplicat la **definició de la implicació** (1), la **propietat (llei) distributiva** (2), la **llei del tercer exclòs** i la **definició de la implicació** (3), i la **llei d'identitat de la conjunció** (4). Totes elles equivalències notables enumerades en la secció 2.1.1, pàgina 42.
Alternativament, podem aplicar la **distributivitat per l'esquerra de la implicació** (a), la **definició de la implicació** (b), la **llei del tercer exclòs** (c), i la **llei d'identitat de la conjunció** (d).

$$p \Rightarrow p \wedge q \overset{(a)}{\equiv} (p \Rightarrow p) \wedge (p \Rightarrow q) \overset{(b)}{\equiv} (\neg p \vee p) \wedge p \Rightarrow q \overset{(c)}{\equiv} V \wedge p \Rightarrow q \overset{(d)}{\equiv} p \Rightarrow q$$

Llei d'exportació

$$p \Rightarrow (q \Rightarrow r) \equiv (p \wedge q) \Rightarrow r$$

És una equivalència que es pot aplicar com a regla d'inferència en els dos sentits. La demostració es deixa com a exercici.

dilema constructiu

$$\frac{\begin{array}{c} p \Rightarrow r \\ q \Rightarrow s \\ p \vee q \end{array}}{r \vee s}$$

Aquesta regla d'inferència és una versió disjuntiva del modus ponens. La demostració es deixa com a exercici.

dilema destructiu

$$\frac{\begin{array}{c} p \Rightarrow r \\ q \Rightarrow s \\ \neg r \vee \neg s \end{array}}{\neg p \vee \neg q}$$

Aquesta regla d'inferència és una versió disjuntiva del modus tollens. La demostració es deixa com a exercici.

resolució

$$\frac{\begin{array}{c} p \vee q \\ \neg p \vee r \end{array}}{q \vee r}$$

Aquesta regla d'inferència relaciona i simplifica clàusules. La demostració es deixa com a exercici.

2.3 Lògica de predicats

Un **predicat** k-ari o d'ordre k és una aplicació del producte cartesià de k dominis o universos (contextos) al conjunt $\{V, F\}$,

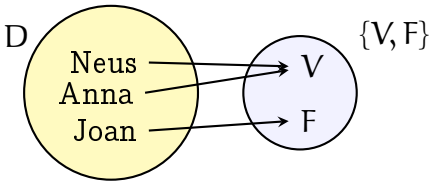
$$P : D_1 \times D_2 \times \cdots \times D_k \longrightarrow \{V, F\},$$

de manera que $P(a_1, a_2, \ldots, a_k)$, una vegada fixats els $a_i \in D_i$, ha de ser o vertader o fals.

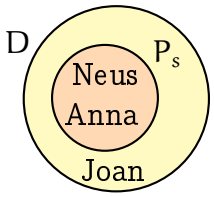
Alternativament, un predicat es pot veure també com una relació k-ària en els conjunts $D_1, D_2, \ldots, D_k$ de manera que podem escriure (abusant de la notació) que

$$P = \{(a_1, a_2, \ldots, a_k) \mid P(a_1, a_2, \ldots, a_k) \equiv V\} \subseteq D_1 \times D_2 \times \cdots \times D_k.$$

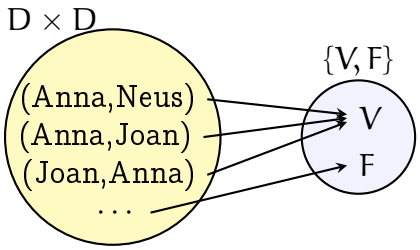
Exemples de dominis poden ser: persones, animals, nombres, símbols, etc. Exemples de predicats amb $k = 1$ i $k = 2$ sobre un mateix domini format per tres persones es mostren a continuació com a aplicació i com a relació (i com a correspondència en el cas $k = 2$).



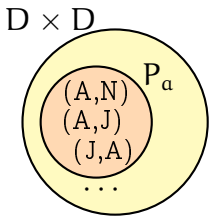
predicat P_s com a aplicació



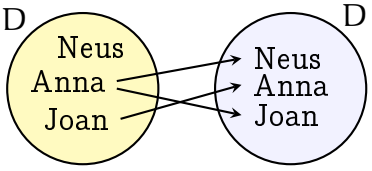
predicat P_s com a relació



P_a com a aplicació



P_a com a relació



P_a com a correspondència

Cada element concret de cada un dels dominis implicats són **constants**. A més a més podem utilitzar **variables** per a referir-nos a elements **genèrics** o **indeterminats** de cada un dels dominis. Normalment escriurem les variables com a x_i o y_i.

Les constants i/o les variables es poden combinar per a donar lloc a nous elements (de nous dominis) mitjançant les anomenades **funcions** o **functors**. Alguns exemples de funcions que poden servir com a arguments de predicats serien

$$f(a_1, x_2) \qquad g(a_1, f(x_1, x_2)) \qquad h(a_1, h(x_1, x_2))$$

Tant les constants com les variables i les funcions poden ser arguments dels predicats i s'anomenen en general **termes**.

Una **fórmula atòmica** és un predicat k-ari aplicat sobre k termes.

$$P(t_1, t_2, \ldots, t_k)$$

Fórmula ben formada (fbf)

1. Si A és una fórmula atòmica, aleshores és fbf.
2. Si A i B són fbfs, aleshores també ho seran

$$\neg A, A \wedge B. A \vee B, A \Rightarrow B, A \Leftrightarrow B$$

3. Si $A(.., v, ..)$ és una fbf i $v \in D$ és una de les variables que conté, aleshores també ho seran

$$\forall v : v \in D', A(.., v, ..) \qquad \exists v, v \in D' : A(.., v, ..)$$

La fórmula

$$\forall v : v \in D', A(.., v, ..)$$

es llegeix "per a tot element v tal que v pertany al conjunt D', $A(.., v, ..)$". I significa que el predicat A és cert per a **tots** els valors de v dins del subdomini D' que pot ser qualsevol subconjunt del domini corresponent a l'argument del predicat on apareix v. Aquest nou operador s'anomena **quantificador universal** o simplement **generalitzador**.

La fórmula

$$\exists v, v \in D' : A(.., v, ..)$$

es llegeix "existeix un element v, que pertany al conjunt D', tal que $A(.., v, ..)$". I significa que el predicat A és cert per a **algun** dels valors de v en D'. Aquest nou operador s'anomena **quantificador existencial** o simplement **particularitzador**.

Quan el conjunt D' associat als quantificadors siga el mateix domini D associat a l'argument corresponent del predicat i/o aquest siga prou clar pel context, escriurem simplement

$$\forall v, A(.., v, ..) \qquad \exists v : A(.., v, ..)$$

Els quantificadors universal i existencial es poden escriure com a conjuncions i disjuncions, respectivament. Si $D = \{a_1, \ldots, a_n\}$ tenim que

$$\forall v : v \in D', A(.., v, ..) \equiv \bigwedge_{v \in D'} A(.., v, ..) \equiv A(.., a_1, ..) \wedge \cdots \wedge A(.., a_n, ..)$$
$$\exists v, v \in D' : A(.., v, ..) \equiv \bigvee_{v \in D'} A(.., v, ..) \equiv A(.., a_1, ..) \vee \cdots \vee A(.., a_n, ..)$$

amb l'única particularitat que aquestes conjuncions o disjuncions poden ser infinites.

Si una fórmula ben formada **no** conté variables o si les que conté estan associades a quantificadors, s'anomena **fórmula tancada**. En cas contrari s'anomena **fórmula oberta**. Les variables no associades a quantificadors en una fórmula s'anomenen **variables lliures** mentre que les associades són **variables lligades**.

2.4 Inferència amb predicats

A una fórmula tancada li correspon un valor de veritat: cert o fals. En canvi, el valor de veritat d'una fórmula oberta dependrà dels valors que puguen prendre les seues variables que no estiguen lligades a quantificadors.

Amb fórmules tancades o obertes però sense quantificadors es pot raonar de la mateixa manera que amb proposicions. En particular, es poden aplicar totes les regles

d'inferència de la lògica proposicional. L'única particularitat de les fórmules obertes és que el raonament depèn dels possibles valors de les variables lliures. Però per a raonar amb quantificadors, necessitarem introduir dos nous parells de regles d'inferència.

EG. Eliminació del generalitzador

$$\frac{\forall x, P(x)}{P(a)}$$

En la fórmula $P(a)$, a és un element qualsevol del domini que ens puga interessar. Pot ser una variable o una constant que haja aparegut anteriorment. O pot ser una nova variable sense cap mena de restricció. Normalment usarem les lletres $a, b, c, \ldots$

IG. Introducció del generalitzador

$$\frac{P(a)}{\forall x, P(x)}$$

Només en el cas que a siga una variable sobre la qual no hi haja cap mena de restricció, podem substituir la fórmula $P(a)$ pel corresponent generalitzador.

IP. Introducció del particularitzador

$$\frac{P(a)}{\exists x : P(x)}$$

La regla és formalment idèntica a l'anterior. Però hi ha una diferència fonamental: ara es pot aplicar independentment del que siga a.

EP. Eliminació del particularitzador

$$\frac{\begin{array}{c}\exists x : P(x) \\ [P(a_i)] \\ \vdots \\ Q\end{array}}{Q}$$

Podem eliminar l'i-èsim particularitzador si introduïm una **nova** variable, a_i. A partir d'aquest moment continuem raonant fins que arribem a una fórmula, Q, que no depèn de a_i.

2.4.1 Algunes propietats interessants amb quantificadors

Lleis de De Morgan amb quantificadors

$$\neg(\forall x \in D, P(x)) \equiv \exists x \in D : \neg P(x)$$
$$\neg(\exists x \in D, P(x)) \equiv \forall x \in D : \neg P(x)$$

Reordenació de quantificadors

$$\forall x, \forall y, P(x,y) \equiv \forall y, \forall x, P(x,y)$$
$$\exists x : \exists y : P(x,y) \equiv \exists y : \exists x : P(x,y)$$

Normalment i per simplificar, escriurem

$$\forall x, y, z, \ldots, P(x,y,z,\ldots) \qquad \exists x, y, z, \ldots : \ P(x,y,z,\ldots)$$

en lloc de

$$\forall x, \forall y, \forall z, \ldots P(x,y,z,\ldots) \qquad \exists x : \exists y : \exists z : \cdots : \ P(x,y,z,\ldots)$$

Un detall important és que quantificadors diferents no es poden intercanviar.

$$\forall x, \exists y : P(x,y) \not\equiv \exists y : \forall x, P(x,y)$$

Exemple: existeix per a tot?

És cert que

$$\exists y : \forall x, \ P(x,y) \vdash \forall x, \exists y \ : \ P(x,y)?$$

Mitjançant regles estàndard a partir de la premissa es té que

1)	$[\forall x \ : \ P(x, a_1)]$	**EP**, introduïm variable particular ...
2)	$P(a, a_1)$	... **EG**, introduïm variable genèrica
3)	$\exists y : P(a,y)$	**IP** en 2. Acaba **EP** de 1.
4)	$\forall x, \exists y \ : \ P(x,y)$	**IG** en 3, s'obté la conclusió

Exemple: per a tot existeix?

És cert que

$$\forall x, \exists y \ : \ P(x,y) \vdash \exists y : \forall x, \ P(x,y)?$$

Si intentem aplicar regles estàndard ara,

1)	$\exists y \ : \ P(a,y)$	**EG**, introduïm variable genèrica
2)	$[P(a,a_1)]$	**EP**, introduïm variable particular ...
3)		

El següent pas que ens interessaria aplicar és la introducció del generalitzador sobre l'expressió de la línia 2, que és una suposició.
Però, el fet important és que aquesta suposició introdueix una **restricció** entre les variables a i a_1. En particular, per a que l'expressió siga certa, la variable a_1 ha de dependre de la variable a.
En altres paraules, $P(a,a_1)$ no pot ser certa per a **qualsevol** valor de a y **el mateix** valor de a_1.
Com que hi ha una restricció sobre la variable genèrica, a, no és possible aplicar la regla **IG** i no es pot completar la regla **EP** anterior.
Podem aleshores afirmar que

$$\forall x, \exists y \ : \ P(x,y) \nvdash \exists y : \forall x, \ P(x,y)?$$

Lamentablement no. Per a poder negar la pregunta inicial, el més fàcil és trobar un contraexemple que en el present cas és molt senzill.
Considerem un univers amb 2 elements, $\{a_1, a_2\}$, de manera que

$$P(a_1,a_2) = P(a_2,a_1) = V, \quad P(a_1,a_1) = P(a_2,a_2) = F.$$

La deducció inicial en aquest cas particular esdevé

$$(P(a_1,a_1) \vee P(a_1,a_2)) \wedge (P(a_1,a_1) \vee P(a_1,a_2)) \vdash (P(a_1,a_1) \wedge P(a_1,a_2)) \vee (P(a_1,a_1) \wedge P(a_1,a_2))$$

d'on directament s'obté

$$V \vdash F$$

la qual cosa és impossible.

2.5 Problemes resolts i comentats

Problema 2.1:[equivalencies]
Considera les expressions següents, calcula les corresponents taules de veritat i digues quines són equivalents.

$(p \Rightarrow q) \Rightarrow r$	$p \Rightarrow (q \Rightarrow r)$	$(p \Rightarrow q) \wedge (q \Rightarrow r)$	$(p \Rightarrow r) \vee (q \Rightarrow r)$

Deixarem les taules de veritat per al final i analitzarem primer les diferents expressions.

Observem primer que en l'última expressió podem aplicar la **pseudodistributivitat per la dreta** de la implicació (podem traure factor comú per la dreta però canviant disjunció per conjunció),

$$(p \Rightarrow r) \vee (q \Rightarrow r) \overset{(\text{pdistr.})}{\equiv} (p \wedge q) \Rightarrow r,$$

amb la qual cosa arribem a una implicació que tindria el mateix conseqüent que la primera expressió. No obstant això, els antecedents són molt diferents i no sembla que puguen ser equivalents la primera i la quarta expressions.

Una altra possibilitat és aplicar la **distributivitat per l'esquerra** a la segona expressió després d'haver aplicat la definició de la implicació al conseqüent (en aquest cas ja tenim l'expressió factoritzada i el que fem és introduir-hi el factor),

$$p \Rightarrow (q \Rightarrow r) \overset{(\text{def.})}{\equiv} p \Rightarrow (\neg q \vee r) \overset{(\text{distr.})}{\equiv} (p \Rightarrow \neg q) \vee (p \Rightarrow r).$$

D'aquesta manera arribem a una disjunció d'implicacions com en el cas de la quarta expressió.

Encara que les implicacions són diferents, és fàcil veure que si apliquem la definició de la implicació i la llei d'idempotència tant en el cas de la segona com en el de la quarta expressió s'arriba a l'expressió equivalent (que és per cert, una **clàusula de Horn**),

$$\neg p \vee \neg q \vee r,$$

la qual cosa significa que ambdues expressions són equivalents.

Com a curiositat, i com a conseqüència de la "simetria" respecte de les variables p i q, també s'ha de complir que

$$p \Rightarrow (q \Rightarrow r) \equiv q \Rightarrow (p \Rightarrow r).$$

Les taules de veritat de les quatre expressions que ens demanen (marcades en la primera fila) són

p	q	r	$p \Rightarrow q$	$(p \Rightarrow q) \Rightarrow r$	$q \Rightarrow r$	$p \Rightarrow (q \Rightarrow r)$	$(p \Rightarrow q) \wedge (q \Rightarrow r)$	$q \Rightarrow r$	$(p \Rightarrow r) \vee (q \Rightarrow r)$
V	V	V	V	V	V	V	V	V	V
V	V	F	V	**F**	F	**F**	**F**	F	**F**
V	F	V	F	V	V	V	**F**	V	V
V	F	F	F	V	V	V	**F**	F	V
F	V	V	V	V	V	V	V	V	V
F	V	F	V	**F**	F	V	**F**	V	V
F	F	V	F	V	V	V	V	V	V
F	F	F	F	**F**	V	V	V	V	V

I a partir d'aquestes es veu com efectivament, les úniques expressions que són equivalents entre elles són la segona i la quarta. (S'han marcat en les quatre columnes els valors F per a facilitar la comparació.)

Problema 2.2:[implicaForallDistr] A partir de l'expressió e_1, es pot deduir l'expressió e_2? I e_1 a partir de e_2? Justifica la resposta.

$$e_1 : \forall x,\ (P(x) \Rightarrow Q(x))$$
$$e_2 : \forall x,\ P(x) \Rightarrow \forall x,\ Q(x)$$

Primer farem una deducció informal mentre intentem entendre què signifiquen les expressions.

La primera fórmula ens diu que existeix una implicació entre els predicats P i Q **per a cada element** del domini. O expressat d'una altra manera,

$$P(x_1) \Rightarrow Q(x_1),\ P(x_2) \Rightarrow Q(x_2),\ \ldots$$

En canvi, la segona és una **única** implicació l'antecedent de la qual afirma P per a tots els elements del domini. Aquest únic antecedent el podem expressar de manera equivalent com un conjunt de premisses,

$$P(x_1),\ P(x_2),\ \ldots$$

Per això, si són certes **totes** les implicacions de què consta e_1, i aplicant el **modus ponens** tantes vegades com elements continga el domini, a partir de les premisses es dedueix el conseqüent que afirma Q per a tot el domini.

Més formalment, com que es tracta de deduir e_2 que és una implicació, podem usar la regla **II** que consisteix a suposar l'antecedent, $\forall x,\ P(x)$, i intentar arribar al conseqüent, $\forall x,\ Q(x)$. Esquemàticament escrivim

1)	e_1	introduïm la premissa
2)	$[\forall x,\ P(x)]$	suposem cert l'antecedent (**II** sobre la conclusió) ...
3)	$P(a) \Rightarrow Q(a)$	eliminem el generalitzador (**EG**) en 1 (elegim $x = a$)
4)	$P(a)$	eliminem el generalitzador (**EG**) en 2 (usem $x = a$)
5)	$Q(a)$	modus ponens (regla **EI**) en 3,4
6)	$\forall x,\ Q(x)$	introduïm el generalitzador (regla **IG**) en 5
7)	$\forall x,\ P(x) \Rightarrow \forall x,\ Q(x)$	... acaba la regla **II** aplicada en 2

Per tant, es compleix que

$$e_1 \vdash e_2.$$

Si ens plantegem la implicació e_2 com a premissa, resulta que aquesta significa que ha de ser cert P **per a tot** el domini perquè siga cert Q (també per a tot el domini). I això no implica necessàriament cap regla separada per als elements del domini. Açò ens fa sospitar que tal vegada $e_2 \nvdash e_1$. Però, com podem demostrar-ho?

La forma més senzilla de demostrar que una expressió **no** es pot deduir a partir d'una altra consisteix a trobar un **contraexemple**. És a dir, un cas concret per al domini de manera que existisca una assignació de veritat que faça certa e_2 i falsa e_1 al mateix temps.

Aquest domini ha de tenir com a mínim 2 elements, ja que per a dominis d'un o de cap element, les dues expressions coincideixen. En canvi, per a dominis de 2 elements tenim que $e_2 \vdash e_1$ esdevé

$$(P(x_1) \wedge P(x_2)) \Rightarrow (Q(x_1) \wedge Q(x_2)) \ \vdash (P(x_1) \Rightarrow Q(x_1)) \wedge (P(x_2) \Rightarrow Q(x_2)),$$

que en el cas concret en què

$$P(x_1) = Q(x_2) = F, \quad P(x_2) = Q(x_1) = V,$$

ens dona

$$\underbrace{F \wedge V}_{F} \Rightarrow \underbrace{V \wedge F}_{F} \vdash \underbrace{(F \Rightarrow V)}_{V} \wedge \underbrace{(V \Rightarrow F)}_{F},$$

d'on s'obté que $V \vdash F$ la qual cosa és òbviament falsa.

Com que hem trobat un cas concret on no es compleix, aleshores **no** és cert que $e_2 \vdash e_1$ i per tant

$$e_2 \nvdash e_1.$$

Problema 2.3:[iiAlternativa] Demostra que les dues regles següents són equivalents:

$A \vdash B \Rightarrow C$ $\qquad$ $A, B \vdash C$

Per a demostrar que les dues regles d'inferència són equivalents, podem deduir cada una d'elles assumint l'altra com a vàlida. Suposem primer que la primera és una regla d'inferència vàlida i l'anomenem **regla AA**. Aleshores,

0)		$A \vdash B \Rightarrow C$ (regla AA)
1)	A	premissa 1
2)	B	premissa 2
3)	$B \Rightarrow C$	regla AA en 1
4)	C	eliminació de la implicació (**EI**) en 2,3

És a dir, és possible deduir la conclusió C a partir de les premisses A i B si la regla AA és certa.

Si suposem ara que la segona és vàlida i l'anomenem **regla BB** tenim també que

0)		$A, B \vdash C$ (regla BB)
1)	A	premissa
2)	[B]	introducció de la implicació (**II**). Suposem l'antecedent ...
3)	C	regla BB en 1,2
4)	$B \Rightarrow C$	... acaba la regla **II** de 2

O siga, que hem pogut deduir $B \Rightarrow C$ a partir de A tot suposant certa la regla BB.

En lloc d'utilitzar les regles d'inferència, podem raonar directament a partir de la definició.

Si suposem que la regla AA és certa, això vol dir que si $A \equiv V$, aleshores $B \Rightarrow C$ també ha de ser vertader segons la definició. Si volem demostrar que la regla BB és certa, hem de mostrar que quan $A \equiv B \equiv V$ s'obté que $C \equiv V$.

En particular, si $B \equiv V$ i $(B \Rightarrow C) \equiv V$ (perquè suposem certa la regla AA), podem deduir la veritat de C (usant **modus ponens** o a partir de la taula de veritat de les dues expressions), per la qual cosa queda demostrat que es pot deduir C a partir de A i B (regla BB).

En sentit contrari, suposem certa la regla BB (a partir de $A \equiv B \equiv V$ es dedueix C), partim del fet que $A \equiv V$, i ens preguntem si és cert o no que $B \Rightarrow C$.

Sabem que $A \equiv V$ però pot ser que B siga vertader o fals. En el primer cas, podem deduir la veritat de C (per la regla BB), per la qual cosa la implicació és certa. En el segon cas ($B \equiv F$), resulta que també la implicació és certa independentment del valor de veritat de B. Per tant, com que podem deduir la implicació en qualsevol cas, la regla AA és certa.

Problema 2.4:[lesDuesAfirmacions]
Intenta establir la veritat o falsedat de cada una de les dues afirmacions següents:

a) El que vaig a dir a continuació és **fals**.

b) Tot el que dic és **fals**.

Considerarem les variables proposicionals a i b per representar cada una de les dues afirmacions (proposicions) anteriors.

La primera, a, diu que la següent, b, és falsa. Mentre que la segona, b, diu que tot (tant a com b) és fals. És a dir,

$$a \stackrel{\text{def}}{\equiv} \neg b$$

$$b \stackrel{\text{def}}{\equiv} \neg a \wedge \neg b$$

En substituir la definició de a en la de b s'obté

$$b \overset{\text{def}}{\equiv} \neg(\neg b) \wedge \neg b \equiv b \wedge \neg b \equiv F$$

per la qual cosa arribem a que b és **fals**. I en substituir açò en la primera definició s'obté de la mateixa manera que a és **certa**. Per tant,

$$a \equiv V \qquad b \equiv F$$

Alternativament, podem escriure les definicions de les dues afirmacions com un conjunt de premisses:

$$\begin{aligned} a &\Leftrightarrow \neg b \\ b &\Leftrightarrow \neg(a \vee b) \end{aligned}$$

on hem aplicat la llei de De Morgan. Si despleguem ara les coimplicacions i les manipulem,

$$\begin{aligned} P_1 &\equiv a \Rightarrow \neg b && \equiv b \Rightarrow \neg a \\ P_2 &\equiv \neg b \Rightarrow a && \equiv \neg a \Rightarrow b \\ P_3' &\equiv b \Rightarrow \neg a \wedge \neg b && \equiv \underbrace{(b \Rightarrow \neg a)}_{P_1} \wedge (b \Rightarrow \neg b) \equiv P_1 \wedge \underbrace{\neg b}_{\equiv P_3} \\ P_4' &\equiv \neg(a \vee b) \Rightarrow a && \equiv a \vee b \equiv P_2 \end{aligned}$$

vegem que podem resumir les anteriors afirmacions només amb 3 premisses simples una de les quals és ja la falsedat de b.

$$\begin{aligned} P_1 &\equiv a \Rightarrow \neg b \equiv b \Rightarrow \neg a \\ P_2 &\equiv \neg b \Rightarrow a \equiv \neg a \Rightarrow b \\ P_3 &\equiv \neg b \end{aligned}$$

De la premissa P_3 i de la P_2 es dedueix la veracitat de a en aplicar la regla **EI**. Si escrivim la deducció completa pas a pas tenim

$P_2)$	$\neg b \Rightarrow a$	premissa 2
$P_3)$	$\neg b$	premissa 3
4)	a	regla **EI** en P_2, P_3
5)	$a \wedge \neg b$	regla **IC** en P_3, 4

En aquesta deducció es veu que no s'ha necessitat la primera premissa per a arribar al resultat.

Si analitzem un poc més la deducció vegem que es tracta en realitat d'una equivalència entre les premisses P_2 i P_3 i el resultat.

$$P_2 \wedge P_3 \equiv (\neg b \Rightarrow a) \wedge \neg b \equiv (b \vee a) \wedge \neg b \equiv \cancelto{F}{(b \wedge \neg b)} \vee (a \wedge \neg b) \equiv a \wedge \neg b$$

Problema 2.5:[lesDuesAfirmacionsBis]
Intenta establir la veritat o falsedat de cada una de les dues afirmacions següents:

a) El que vaig a dir a continuació és **fals**.

b) El que he dit abans és **fals**.

Considerarem les variables proposicionals a i b per representar cada una de les dues afirmacions (proposicions) anteriors.

La primera, a, diu que la següent, b, és falsa. Mentre que la segona, b, diu que la primera, a, és falsa. És a dir,

$$a \stackrel{\text{def}}{\equiv} \neg b$$
$$b \stackrel{\text{def}}{\equiv} \neg a$$

Contràriament al que passava en l'exercici 2.4, la substitució de la definició de a en la de de b, o al revés, dóna lloc a una expressió de la forma,

$$p \stackrel{\text{def}}{\equiv} \neg p$$

que es correspon amb una paradoxa del tipus "aquesta afirmació és falsa" a partir de la qual no se'n pot establir ni la veritat ni la fasedat, per la qual cosa haurem de considerar un raonament alternatiu.

Suposem que a és certa, la qual cosa vol dir que b ha de ser falsa. b afirma la falsedat de a, però com que és falsa, a és efectivament vertadera.

Si suposem en canvi que a és falsa, aleshores b hauria de ser vertadera. I b efectivament afirma la falsedat de a.

En resum, a partir de les dues afirmacions només podem arribar a què a és vertadera i b falsa, o a que a és falsa i b vertadera. I això ho podem expressar simplement com $a \not\equiv b$ la qual cosa és equivalent a

$$(a \equiv V \wedge b \equiv F) \vee (a \equiv F \wedge b \equiv V)$$

O be fent servir expressions purament proposicionals com a

$$(a \wedge \neg b) \vee (\neg a \wedge b) \equiv \neg(a \Leftrightarrow b) \equiv a \not\Leftrightarrow b$$

si acceptem el símbol $\not\Leftrightarrow$ per representar la **disjunció exclusiva**.

Si considerem les dues afirmacions com a premisses les podem escriure com a

$$a \Leftrightarrow \neg b$$
$$b \Leftrightarrow \neg a$$

I una vegada desplegades les coimplicacions donen lloc només a

$$P_1 \equiv \neg a \Rightarrow b \equiv a \vee b$$
$$P_2 \equiv b \Rightarrow \neg a \equiv \neg b \vee \neg a \equiv \neg(a \wedge b)$$

La conjunció de les premisses és exactament la disjunció exclusiva obtinguda abans. I les úniques coses que es poden deduir a partir d'aquestes premisses serien les que s'il·lustren en la següent taula de veritat.

a	b	$P_1 \wedge P_2 \equiv a \not\Leftrightarrow b$	$a \vee b$	$\neg(a \wedge b)$	V
V	V	F	V	F	V
V	F	**V**	**V**	**V**	**V**
F	V	**V**	**V**	**V**	**V**
F	F	F	F	V	V

Com a conclusió i resultat final, no és possible de cap manera establir la veritat o la falsedat de cap de les dues afirmacions.

Problema 2.6:[tollendoPonens] Demostra que és cert que

$$P \vee Q, \neg P \vdash Q$$

Aquesta regla que s'utilitza en lògica clàssica, s'anomena **modus tollendo ponens** o també **sil·logisme disjuntiu** (pàgina 52). Un exemple d'aquesta regla el trobem en la coneguda frase atribuïda al personatge més famós de Conan Doyle: *"Una vegada que es descarta l'impossible, el que queda és la veritat per improbable que semble"*.

Per a demostrar-la podem usar les regles d'inferència estàndard.

1)	$P \vee Q$	premissa 1
2)	$\neg P$	premissa 2
3)	$[\neg Q]$	suposem el contrari del que volem (**IN**) ...
4)	$[P]$	... Comencem demostració per casos (**ED**) en 1. Primer cas ...
5)	$P \wedge \neg P \equiv F$	... ... introducció de la conjunció (**IC**) en 2, 4
6)	$[Q]$	... ... Segon cas ...
7)	$Q \wedge \neg Q \equiv F$	... ... introducció de la conjunció (**IC**) en 3, 6
8)	F	... acaba la regla **ED** en 1. Contradicció en qualsevol cas
9)	Q	acaba la regla **IN**

L'anterior és una demostració per reducció a l'absurd: comencem negant el que volem demostrar i comprovem que s'arriba necessàriament a una contradicció.

Una alternativa a l'anterior és convertir la disjunció de la primera premissa en una implicació per a poder aplicar el **modus ponens** o regla **EI**.

1)	$P \vee Q$	premissa 1
2)	$\neg P$	premissa 2
3)	$\neg P \Rightarrow Q$	**definició de la implicació** en 1
4)	Q	eliminem la implicació (**EI**) en 2,3.

Alternativament, podem veure que la conjunció de les dues premisses és

$$(P \vee Q) \wedge \neg P \equiv (P \wedge \neg P) \vee (Q \wedge \neg P) \equiv F \vee (Q \wedge \neg P) \equiv Q \wedge \neg P$$

I d'aquesta conjunció es dedueix Q mitjançant la regla d'eliminació de la conjunció (**EC**). El mateix en forma de taula seria

1)	$P \vee Q$	premissa 1
2)	$\neg P$	premissa 2
3)	$(P \vee Q) \wedge \neg P$	introducció de la conjunció (**IC**) en 1,2
4)	$Q \wedge \neg P$	equivalent a 3
5)	Q	eliminació de la conjunció (**EC**) en 4

Problema 2.7:[tollendoPonensDisj] Explica què significa la següent deducció i demostra-la a partir de les taules de veritat.

$$P \vee Q, \neg P \vdash Q \vee R$$

Tenim dues premisses. Una disjunció de dues proposicions i la negació d'una d'elles. La conclusió és una altra disjunció de la proposició no negada en les premisses i una altra proposició que no apareix en cap premissa.

Formalment, és com el **modus tollendo ponens** o **sil·logisme disjuntiu** (pàgina 52), però amb la introducció d'una disjunció en la conclusió. Per això mateix, és relativament fàcil demostrar-la aplicant *modus tollendo ponens* primer (com en l'exercici 2.6) i la regla d'**introducció de la disjunció**, **ID**, després.

Deixem aquesta opció de demostració com a exercici.

Per a demostrar-ho mitjançant taules de veritat necessitarem primer calcular-les per a la conjunció de les premisses d'una banda, i per a la conclusió de l'altra.

De les files de la taula de veritat, marcarem aquelles que es corresponen amb la veritat de les premisses que són les crítiques a l'hora de decidir si es pot deduir o no la conclusió.

					premisses	conclusió
P	Q	R	$P \vee Q$	$\neg P$	$(P \vee Q) \wedge \neg P$	$Q \vee R$
V	V	V	V	F	F	V
V	V	F	V	F	F	V
V	F	V	V	F	F	V
V	F	F	V	F	F	F
F	V	V	V	V	**V**	**V**
F	V	F	V	V	**V**	**V**
F	F	V	F	V	F	V
F	F	F	F	V	F	F

Podem observar que en els dos casos en què les premisses són certes, la conclusió també ho és. I per tant, la deducció queda demostrada. Observem, finalment, que la resta de la taula de veritat de la conclusió (en gris) no calia calcular-la.

Problema 2.8:[quantificadorsInvertits] Analitza la deducció següent i demostra-la utilitzant les regles d'inferència estàndard.

$$\forall X, \exists Y : (P(X,Y) \wedge Q(X)) \vdash \forall Y, \exists X : (P(X,Y) \vee Q(X))$$

Es tracta de dues fórmules, una premissa i una conclusió, amb quantificadors niats amb l'ordre invertit en cada cas.

Tenim un predicat binari i un altre unari, que apareixen connectats mitjançant conjunció i disjunció en la premissa i la conclusió, respectivament.

Però la part més important la formen els quantificadors invertits. La premissa diu que per a tot element del domini, X, n'hi ha d'haver un altre amb el qual es relacione, $P(X,\cdot)$, **i a més a més** s'ha de complir $Q(X)$.

Com que la fórmula oberta sense els quantificadors és una conjunció, es pot deduir separadament

$$\forall X, \exists Y : P(X,Y) \qquad \forall X, Q(X)$$

com a subconclusions a partir de la premissa. (De fet, la conjunció d'aquestes és equivalent a la premissa.)

En canvi, la conclusió diu que per a tot element, Y, ha d'haver-n'hi un altre que, o bé es relacione, $P(\cdot,Y)$, o bé complisca Q.

Afortunadament, com que es tracta d'una disjunció, podrem deduir la conclusió si arribem a una de les dues fórmules següents

$$\forall Y, \exists X : P(X,Y) \qquad \exists X : Q(X)$$

A la primera no hi ha manera d'arribar llevat que P siga commutatiu. Però la segona sí que és fàcilment deduïble a partir de la segona subconclusió abans esmentada.

Per tant, l'estratègia de deducció de la conclusió a partir de la premissa, haurà de passar per la segona subconclusió per a arribar a l'última de les fórmules, amb la qual cosa deduiríem la conclusió.

La deducció formal usant regles d'inferència reprodueix aquesta estratègia d'una manera una mica més directa.

1)	$\forall X, \exists Y : (P(X,Y) \wedge Q(X))$	premissa
2)	$\exists Y : (P(a,Y) \wedge Q(a))$	**EG** en 1 (elegim) $X = a$
3)	$[P(a,b_1) \wedge Q(a)]$	**EP** en 2 (introduïm) b_1 ...
4)	$Q(a)$	... **EC** en 3 (b_1 ja ha desaparegut) ... acaba **EP**
5)	$P(a,c) \vee Q(a)$	**ID** en 4 (introduïm c, vegeu text)
6)	$\exists X : P(X,c) \vee Q(X)$	**IP** en 5
7)	$\forall Y, \exists X : P(X,Y) \vee Q(X)$	**IG** en 6

En la línia 5, hem introduït una variable, c, que pot ser qualsevol element del domini, ja que no és en absolut important que la fórmula $P(a,c)$ siga certa perquè ho siga la disjunció. Aquesta matisació és important perquè després (línia 7) voldrem introduir un generalitzador associat a aquesta variable.

Problema 2.9:[quadratParell] Demostra l'afirmació: **Si un enter parell és el quadrat d'un altre, aleshores aquest altre és també parell**. Per a això, pots seguir el següent esquema informal: imagina que el quadrat és parell però el nombre no, i intenta arribar a una contradicció. Fes primer una demostració informal utilitzant llenguatge natural i després expressa tant l'enunciat com la demostració fent servir la lògica de primer ordre.

L'enunciat ens suggereix una demostració per reducció a l'absurd. I l'estratègia consisteix a suposar que un enter senar al quadrat dóna com a resultat un nombre parell.

Però si ho pensem un poc, un enter no parell multiplicat per ell mateix, mai no podrà donar com a resultat un nombre parell.

Podem per exemple imaginar la descomposició en factors primers del quadrat i la del nombre. Si en el quadrat no pot aparèixer el factor 2, és impossible que en el nombre hi haguera cap factor 2 (Perquè aleshores hauria d'aparèixer també en el quadrat).

Ara expressarem formalment l'enunciat del que volem demostrar i després la seua demostració.

$$\forall n \in \mathbb{Z}^+,\ (\exists k \geq 0\ :\ n^2 = 2k) \Rightarrow (\exists \ell \geq 0\ :\ n = 2\ell)$$

Des del punt de vista de la lògica de predicats, podem eliminar el generalitzador i escriure

$$\underbrace{\exists k \geq 0 \ : \ n^2 = 2k}_{P_1} \vdash \underbrace{\exists \ell \geq 0 \ : \ n = 2\ell}_{Q},$$

de manera que si som capaços de demostrar aquesta deducció per al valor genèric, n, quedarà demostrat el resultat general.

1)	$\exists k \geq 0 \ : \ n^2 = 2k$	premissa 1
2)	$[n^2 = 2k_1]$	**EP** en 1 (introduïm k_1) ...
3)	$[\neg(\exists \ell \geq 0 \ : \ n = 2\ell)]$	... **IN** ...
3')	$\forall \ell \geq 0, \ n \neq 2\ell$	... ... De Morgan en 3
4)	$\exists \ell \geq 0 \ : \ (n = 2\ell \vee n = 2\ell + 1)$	... ... introduïm una tautologia
5)	$[n = 2k_2 \vee n = 2k_2 + 1]$	... ... **EP** en 4 (introduïm k_2) ...
6)	$n \neq 2k_2$	... ... **EG** en 3'
7)	$n = 2k_2 + 1$	... ... modus tollendo ponens en 5, 6
8)	$n^2 = (2k_2+1)^2 = 2\underbrace{(k_2+1)2k_2}_{k_3} + 1$	... ... elevem al quadrat
9)	$n^2 = 2k_1 \wedge n^2 = 2k_3 + 1$	... ... **IC** en 2, 8
10)	F	... ... equivalent a 9 si $n, k_1, k_3 \in \mathbb{Z}^+$
11)	$\exists \ell \geq 0 \ : \ n = 2\ell$	... acaba **EP** 4 (k_2), acaba **IN**
12)	$\exists \ell \geq 0 \ : \ n = 2\ell$	acaba **EP**1 (k_1)

Problema 2.10:[inferenciaDosF] En la següent expressió, identifica premisses i conclusió i fes la demostració pas a pas especificant totes les regles d'inferència usades.

$$\exists Y : (\forall X, p(X,Y) \Rightarrow \neg q(X) \ ) \quad \vdash \quad \exists Y : \ q(Y) \Rightarrow \neg p(Y,Y).$$

Explica què canviaria en la demostració si en la premissa intercanviàrem l'ordre dels quantificadors, És a dir, si la premissa fóra

$$\forall X, \ \exists Y : \ (\ p(X,Y) \Rightarrow \neg q(X) \ ).$$

Es tracta d'una única premissa amb dos quantificadors niats,

$$P_1 \equiv \exists Y : \ (\forall X, p(X,Y) \Rightarrow \neg q(X) \ ),$$

i una conclusió amb un únic particularitzador,

$$\exists Y : q(Y) \Rightarrow \neg p(Y, Y).$$

Quantificadors a banda, la premissa és una implicació i la conclusió també. Per tant, en algun moment haurem d'aplicar la regla **II** i haurem de suposar el predicat q per a algun valor del domini corresponent.

Aquest predicat q s'haurà de combinar amb la implicació de la premissa per a arribar al conseqüent de la conclusió. Més detalladament i en forma de taula,

1)	$\exists Y : (\forall X,\ p(X, Y) \Rightarrow \neg q(X)\ )$	premissa
2)	$\forall X,\ p(X, a_1) \Rightarrow \neg q(X)$	**EP** en 1 (introduïm a_1)
3)	$p(a_1, a_1) \Rightarrow \neg q(a_1)$	... **EG** en 2 (elegim $X = a_1$)
4)	$[q(a_1)]$	... **II** ...
5)	$\neg p(a_1, a_1)$	... ... *modus tollens* en 3, 4
6)	$q(a_1) \Rightarrow \neg p(a_1, a_1)$	... acaba **II** de 4
7)	$\exists Y : q(Y) \Rightarrow \neg p(Y, Y)$	**IP** en 6, acaba **EP** en 1 (a_1)

La deducció anterior seria un poc més llarga si en lloc d'usar el **modus tollens** en la línia 5 s'hagueren d'usar regles estàndar.

En el cas que en la premissa s'intercanviaren els quantificadors com s'especifica en l'enunciat tindríem que

1)	$\forall X : (\exists Y,\ p(X, Y) \Rightarrow \neg q(X)\ )$	premissa
2)	$\exists Y :\ p(a, Y) \Rightarrow \neg q(a)$	**EG** en 1 (elegim $X = a$)
3)	$p(a, a_1) \Rightarrow \neg q(a)$	**EP** en 2 (introduïm a_1) ...
4)	$[q(a)]$	... **II** ...
5)	$\neg p(a, a_1)$	... ... modus tollens en 3, 4
6)	$q(a) \Rightarrow \neg p(a, a_1)$	... acaba **II** de 4
7)	$\exists Y, Z : q(Y) \Rightarrow \neg p(Y, Z)$	**IP** en 6 (x2), acaba **EP** en 2

Com que ara estem obligats a eliminar primer el generalitzador (i introduir un valor genèric, a), el valor que s'introdueix en eliminar el particularitzador no es pot elegir de manera que els dos coincidisquen, com abans. De fet, el valor particular a_1 depén del valor de a.

En altres paraules, no es pot arribar a la conclusió amb la mateixa extratègia que abans.

No obstant això, si volem **demostrar** que, efectivament la conclusió no és deduïble a partir de la premissa, hem de mostrar almenys un **contraexemple**. És a dir, un cas particular en què la premissa siga certa i la conclusió falsa.

Aquest contraexemple el podem construir sobre un domini amb dos elements, $\{1,2\}$ de manera que els valors de veritat siguen

$p(1,2)$	$p(2,1)$	$q(1)$	$q(2)$	$p(1,1)$	$p(2,2)$
F	F	V	V	V	V

Segons l'assignació de veritat elegida tenim que la premissa és certa,

$$\overbrace{\overbrace{(\overbrace{p(1,1)}^{V} \Rightarrow \overbrace{\neg q(1)}^{F})}^{F} \vee \overbrace{(p(1,2) \Rightarrow \neg q(1))}^{V}}^{V} \wedge \overbrace{\overbrace{(\overbrace{p(2,1)}^{F} \Rightarrow \neg q(2))}^{V} \vee \overbrace{(p(2,2) \Rightarrow \overbrace{\neg q(2)}^{F})}^{F}}^{V} \equiv V,$$

però la conclusió és falsa,

$$\overbrace{(\overbrace{q(1)}^{V} \Rightarrow \overbrace{\neg p(1,1)}^{F})}^{F} \vee \overbrace{(\overbrace{q(2)}^{V} \Rightarrow \overbrace{\neg p(2,2)}^{F})}^{F} \equiv F.$$

La deducció a partir de la segona premissa és falsa per que, segons l'enunciat, es tracta de dos quantificadors niats aplicats sobre una implicació. Però si haguérem escrit la premissa com

$$\forall X,\ (\exists Y:\ p(X,Y)\ ) \Rightarrow \neg q(X).$$

de manera que el particularitzador només actua sobre el predicat, p, i no sobre la implicació, aleshores la deducció si que seria certa. La demostració d'aquesta nova deducció sobre la segona premissa modificada es deixa com a exercici.

Es pot comentar que aquesta doble interpretació quant a l'abast dels quantificadors no té sentit en el cas de la primera premissa ja que la variable del predicat unari, q, està associada al quantificador extern.

Problema 2.11:[tresFormules] Considera les tres fórmules que es donen a continuació i raona si alguna es pot o no deduir a partir de les altres dues. Si és el cas, explicita la deducció corresponent fent servir les regles d'inferència estàndard.

$f_1 \equiv \boxed{\exists x :\ R(x)}$ $f_2 \equiv \boxed{\exists y :\ \forall x,\ ((P(x) \vee Q(y)) \Rightarrow R(y))}$ $f_3 \equiv \boxed{P(p)}$

Dues de les fórmules són fets o afirmacions de veracitat dels predicats R (per a alguns valors del domini) i P (per a un element concret, p). Mentre que l'altra fórmula combina implicacions per a determinats parells d'elements del domini.

A partir de fets no sembla fàcil deduir implicacions per la qual cosa descartarem en principi f_2 com a conclusió.

Això vol dir que f_2 és una premissa i que l'altra ha de ser un fet o conjunt de fets que es puguen combinar amb les implicacions de f_2.

Però com que totes les implicacions de f_2 tenen R com a conseqüent, això vol dir que l'única fórmula que té sentit plantejar-se com a conclusió és f_1.

Demostrarem, per tant, que

$$\underbrace{P(p)}_{\text{premissa 1}} \ ,\ \underbrace{\exists y :\ \forall x,\ ((P(x) \vee Q(y)) \Rightarrow R(y))}_{\text{premissa 2}} \vdash \underbrace{\exists x :\ R(x)}_{\text{conclusió}}$$

1)	$\exists y :\ \forall x,\ ((P(x) \vee Q(y)) \Rightarrow R(y))$	premissa 2
2)	$[\forall x,\ ((P(x) \vee Q(a_1)) \Rightarrow R(a_1))]$	**EP** ... (introduïm a_1)
3)	$(P(p) \vee Q(a_1)) \Rightarrow R(a_1)$	... **EG** (elegim $x = p$)
4)	$P(p)$	... premissa 1
5)	$P(p) \vee Q(a_1)$	... **ID** en 4
6)	$R(a_1)$	... **EI** en 3, 5
7)	$\exists x :\ R(x)$	**IP** en 6, acaba **EP** (a_1 ha desaparegut)

Si repensem ara f_2 com a conclusió, i ens fixem en què el generalitzador només abarca la disjunció, la podem aleshores reescriure com a

$$\exists y, x :\ (P(x) \vee Q(y) \Rightarrow R(y))$$

I per a que siga certa, només cal trobar una combinació de valors que facen cert algun $R(y)$. I sabem que això pot passar si f_1 és certa.

Anem doncs a intentar la deducció

$$f_1, f_3 \vdash f_2$$

1)	$\exists x : \ R(x)$	premissa 1
2)	$P(p)$	premissa 2
3)	$R(a_1)$	**EP** ... (introduïm a_1)
4)	$[\forall x, \ ((P(x) \vee Q(a_1))]$	... **II** (suposem antecedent a_1)...
5)	$(\forall x, \ ((P(x) \vee Q(a_1))) \Rightarrow R(a_1)$	... acaba **II**
6)	$\exists y : \ \forall x, \ ((P(x) \vee Q(y)) \Rightarrow R(y))$	**IP** en 5, acaba **EP** (a_1 ha desaparegut)

Si considerem finalment f_3 com a conclusió no es veu cap manera de deduir-la a partir de les altres fórmules. Però per a estar segurs de que no és deduïble, cal trobar un contraexemple.

Considerem el cas més senzill en què el domini de P només conté l'element p i el domini de Q i R (que han de ser el mateix) només tenen un element. En eixe cas, totes les fórmules esdevenen proposicionals i la nostra deducció es pot escriure com a

$$R, P \vee Q \Rightarrow R \vdash P$$

Si suposem que la primera premissa, R, és certa, tenim que

$$(P \vee Q \Rightarrow V \equiv V) \vdash P$$

la qual cosa és falsa ja que P pot no ser cert.

2.6 Problemes proposats

Problema 2.12:[absorcions]
Considera les lleis d'absorció de la conjunció i de la disjunció (pàgina 43),

$$(p \vee q) \wedge p \equiv p \qquad (p \wedge q) \vee p \equiv p$$

i demostra-les,

a) usant taules de veritat.

b) usant altres lleis d'equivalència.

c) usant exclusivament regles d'inferència.

d) En resum, quina és la forma més fàcil de demostrar les dues lleis d'absorció?

Problema 2.13:[lesTresAfirmacions]
Intenta establir la veritat o falsedat de cada una de les tres afirmacions següents:

a) La següent afirmació és **falsa**.

b) El que vaig a dir a continuació és **cert**.

c) No sé si certes o falses, però les dues afirmacions anteriors són **el mateix**.

■

Problema 2.14:[lesSisAfirmacions]
Sabries dir quines de les següents afirmacions són certes?

a) Totes les posteriors (a aquesta).

b) Cap de les que segueixen.

c) Només una de les anteriors.

d) Totes les anteriors.

e) Cap de les anteriors.

f) Cap de les anteriors.

Resol primer el problema amb la suposició addicional que **només una** de les afirmacions és certa.

■

Problema 2.15:[tresPredicats]
En la següent expressió, identifica premisses i conclusió i fes la demostració pas a pas especificant totes les regles d'inferència usades.

$$\forall X : ((\neg P(X) \vee Q(X)) \Rightarrow R(X)) \wedge \exists X : Q(X) \vdash \exists X : (P(X) \vee R(X))$$

Problema 2.16:[germans] Considera les relacions de parentiu familiars normals de primer i segon grau (pares, fills, avis, etc.), i tradueix les afirmacions a la lògica de primer ordre. Digues si constitueixen o no una contradicció i si es pot deduir alguna cosa sobre la família concreta.
Bohigues tenia un germà. El germà de Bohigues va morir. No obstant això, l'home que va morir no va tenir mai cap germà.

Problema 2.17:[irracional] Una demostració senzilla de la irracionalitat de $\sqrt{2}$ consisteix a suposar el contrari, és a dir que podem escriure

$$\sqrt{2} = \frac{a}{b}$$

i que, a més a més, aquesta fracció no es pot simplificar més (és irreductible).
Si això fos cert, en elevar al quadrat i aïllar, tindríem que

$$a^2 = 2b^2$$

la qual cosa implica que a^2 és parell.
Si a^2 és parell, també ho ha de ser a (problema 2.9),
la qual cosa implica que b no pot ser parell, perquè en aquest cas $\frac{a}{b}$ es podria simplificar.
Però si a és parell s'ha de poder escriure $a = 2k$ per a algun k, amb la qual cosa

$$a^2 = 2k \; 2k = 2b^2$$

I, per tant, b^2 és parell i b també.
O siga, que b ha de ser parell i senar alhora, la qual cosa és impossible. Per tant, la suposició inicial que $\sqrt{2}$ és racional ha de ser falsa.

Descriu aquesta demostració, incloent-hi l'enunciat, usant la lògica de predicats i les corresponents regles d'inferència. El resultat del problema 2.9 el pots fer servir com una regla d'inferència més.

Problema 2.18:[enTirant]
En la següent expressió, i en el text a continuació, identifica premisses i conclusió i fes la corresponent deducció fent servir les regles d'inferència estàndard.

$$\forall X : \exists Y : (P(X) \Rightarrow (Q(X,Y) \vee R(Y))),\ P(k),\ \forall X : \neg R(X) \vdash \exists X, Y : Q(X,Y)$$

Tot cavaller errant té la seua amada, de manera que o bé aquest l'estima bojament o és que ella és monja (o totes dues coses). En Tirant és un cavaller errant. I no hi ha monges en el regne. Com que l'estimada d'en Tirant ha d'existir (com la de qualsevol cavaller errant), i com que resulta que no hi ha monges en aquest regne, aleshores hi ha almenys un cavaller i una dama de manera que el primer estima bojament la segona.

3. Inducció i recursió

3.1 Definicions i predicats recursius

Una definició es pot entendre com una manera d'especificar un concepte nou a partir d'altres conceptes més bàsics o ja coneguts. En el context de la lògica, la definició d'un predicat és una doble implicació amb una fórmula, de manera que podem establir unívocament el valor de veritat o falsedat del predicat en funció dels seus arguments. Per exemple, els predicats $P(n)$ i $I(n)$ que són certs si n és parell/senar i falsos en cas contrari es podrien definir com a

$$\boxed{P(n)} \Leftrightarrow \boxed{\exists k \in \mathbb{Z}^+ \ : \ n = 2 \cdot k}$$
$$\boxed{I(n)} \Leftrightarrow \boxed{\neg P(n)}$$

Un altre exemple en matemàtiques és la definició de límit d'una successió

$$\boxed{\lim_{n \to \infty} s_n = a} \Leftrightarrow \boxed{\forall \varepsilon \in \mathbb{R},\ \varepsilon > 0,\ \exists n_0 \in \mathbb{N} \ : \ \forall n > n_0,\ |s_n - a| \leq \varepsilon}$$

Parlem de **definició recursiva** quan el concepte l'especifiquem en funció d'ell mateix. Per exemple i de manera informal, "un nombre enter és parell si (i solament si) el seu immediat anterior no ho és".

Tota definició recursiva ha de tenir un o més **casos base** de manera que quan s'aplica la definició recursiva a qualsevol element del domini sempre s'arribe a algun cas base.

Per exemple, el cas base de l'exemple anterior podria ser que el nombre zero és parell. De manera que la definició recursiva expressada formalment seria

$$P(n) = \begin{cases} V & \textbf{si } n = 0 \quad (\text{CB}, \text{cas base}) \\ \neg P(n-1) & \textbf{si } n > 0 \quad (\text{CR}, \text{cas recursiu}) \end{cases}$$

i si l'aplicàrem a l'enter 3 tindríem una cosa com ara

$$P(3) = \neg P(2) = \neg\neg P(1) = P(1) = \neg P(0) = \neg V = F.$$

La definició anterior que és una igualtat amb condicions és el que anomenem **relació de recurrència**.

Una forma alternativa en lògica per a la mateixa definició seria una conjunció (implícita) entre el(s) cas(os) base i el cas recursiu on tindríem una coimplicació, ja que la relació entre un enter i el següent o l'anterior és la mateixa.

$$\begin{cases} P(0) & (CB) \\ \forall n > 0,\ P(n) \Leftrightarrow \neg P(n-1) & (CR) \end{cases}$$

i calcularíem $P(3)$ de la manera següent. Primer obtindríem un total de 4 premisses en aplicar la regla CR per a $n = 1, 2, 3$ i la regla CB.

$$\underbrace{(P(3) \Leftrightarrow \neg P(2))}_{p_1} \wedge \underbrace{(P(2) \Leftrightarrow \neg P(1))}_{p_2} \wedge \underbrace{(P(1) \Leftrightarrow \neg P(0))}_{p_3} \wedge \underbrace{P(0)}_{p_4}$$

I a partir d'aquestes podríem fer la següent deducció

1)	$P(0)$	p_4
2)	$P(0) \Rightarrow \neg P(1)$	Def. coimp + **EC**+ Contrap. lòg. en p_3
3)	$\neg P(1)$	**EI** en 1) 2)
4)	$\neg P(1) \Rightarrow P(2)$	Def. coimp + **EC** en p_2
5)	$P(2)$	**EI** en 3) 4)
6)	$P(2) \Rightarrow \neg P(3)$	Def. coimp + **EC**+ Contrap. lòg. en p_1
7)	$\neg P(3)$	**EI** en 5) 6)

És important adonar-se que per a poder afirmar alguna cosa a partir d'una definició recursiva s'està fent un raonament que connecta el que es vol demostrar amb un cas base per a poder encadenar una seqüència d'implicacions (línies 2, 4 i 6) **des del** cas base **fins a** la conclusió corresponent (línia 7).

En lògica són particularment interessants les definicions que fan servir implicacions en el mateix sentit en què després seran aplicades en les demostracions de veritat corresponents. És a dir, definicions de predicats, Q, en principi sobre $\mathbb{Z}^+$ com ara

$$\begin{cases} \forall x \leq b,\ \boxed{Q(x)} & (CB,\ \text{un o més casos base}) \\ \forall x\ :\ \boxed{x > b,}\ \boxed{\mathcal{C}_i(Q(\alpha_i(x))) \Rightarrow Q(x)} & (CR_i,\ \text{cas recursiu i-èsim}) \end{cases}$$

on α_i és un functor que a partir d'un element del domini, x, ens dona un element "anterior", $\alpha_i(x) < x$, i $\mathcal{C}_i$ és una **fbf** que conté Q i que normalment és una conjunció de predicats.

Per a obtenir una definició d'aquest tipus en el cas de l'exemple anterior podem transformar-la de manera que en la definició apareguen les implicacions en el sentit en què seran usades en les deduccions.

$$\begin{cases} & P(0) & & (CB) \\ \forall n > 0, & P(n-1) & \Rightarrow \neg P(n) & (CR_1) \\ \forall n > 1, & \neg P(n-1) & \Rightarrow \ P(n) & (CR_2) \end{cases}$$

Amb aquesta definició els raonaments acabarien en CR_1 o en CR_2 segons es vulga demostrar la falsedat o la veritat de P, respectivament.

També podem combinar els dos casos recursius anteriors (i fer algunes manipulacions més) per a arribar a la següent definició equivalent.

$$\begin{cases} & P(0) & & (CB_1) \\ & \neg P(1) & & (CB_2) \\ \forall n > 1, & P(n-2) & \Rightarrow P(n) & (CR_1) \\ \forall n > 2, & \neg P(n-2) & \Rightarrow \neg P(n) & (CR_2) \end{cases}$$

on ara relacionem la veritat (falsedat) de dos nombres parells (senars) consecutius.

Sobretot si ens plantegem la possibilitat d'automatitzar raonaments, és encara més important restringir les definicions recursives a implicacions on el conseqüent no aparega negat.

Això planteja un problema a l'hora de demostrar la falsedat. Per això, en alguns sistemes de raonament automàtics i en el llenguatge de programació Prolog en particular, se segueix la convenció anomenada **negació per fallada**. Això vol dir, amb poques paraules, que si no podem deduir la **veritat** d'un predicat a partir de la seua "definició", aleshores és fals.

Per exemple, el predicat P anterior es podria "definir" també com a

$$\begin{cases} & P(0) & & (CB) \\ \forall n > 1, & P(n-2) & \Rightarrow P(n) & (CR) \end{cases}$$

la qual cosa es traduiria en Prolog com a

```
parell(0).        % CB
parell(N) :-
   N>1,           % CR
   M is N-2,      % avaluació del functor corresponent
   parell(M).
```

Ara considerarem un altre exemple de definició recursiva usant lògica de predicats:

$$\begin{cases} S(1,1) & (CB) \\ \forall n > 1, \forall y \in Z^+,\ S(y, n-1) \Rightarrow S(y+n, n) & (CR) \end{cases}$$

El predicat $S(y,n)$ és cert si y és la suma dels n primers enters.
Aquesta última "definició" podem modificar-la lleugerament

$$\begin{cases} S(1,1) \\ \forall n > 1, \forall x \in Z^+,\ x \geq n,\ S(x-n, n-1) \Rightarrow S(x,n) \end{cases}$$

I fins i tot introduir noves variables per a l'antecedent de la implicació

$$\begin{cases} S(1,1) \\ \forall n > 1, \forall x \in Z^+,\ x \geq n,\ \exists y, m\ :\ y = x - n \wedge m = n-1 \wedge S(y,m) \Rightarrow S(x,n) \end{cases}$$

la qual cosa admet una traducció quasi literal al llenguatge de programació Prolog.

```
suma(1,1).
suma(X,N) :-
   N>1,
   M is N-1,
   suma(Y,M),
   X is Y+N.
```

No obstant això, el predicat anterior (convenientment estés per al valor zero) es correspon en realitat amb una funció,

$$s : \mathbb{Z}^+ \longrightarrow \mathbb{Z}^+,$$

que podríem representar alternativament mitjançant la recurrència

$$s(n) = \begin{cases} 0 & \textbf{si } n = 0 \\ s(n-1) + n & \textbf{si } n > 0 \end{cases}$$

Una relació de recurrència és en realitat un sistema d'equacions, que també es pot escriure com a

$$\begin{cases} s(0) &= 0 \\ s(n) &= s(n-1) + n, \quad \forall\ n > 0. \end{cases}$$

No obstant això, les relacions de recurrència se solen interpretar com una operació o un càlcul:

$$s(n) \leftarrow \begin{cases} 0 & \textbf{si } n = 0 \\ s(n-1) + n & \textbf{si } n > 0 \end{cases}$$

on el símbol $\leftarrow$ significa "es calcula com a" o "es reescriu com a" segons que estiguem en un context de programació imperativa o funcional.

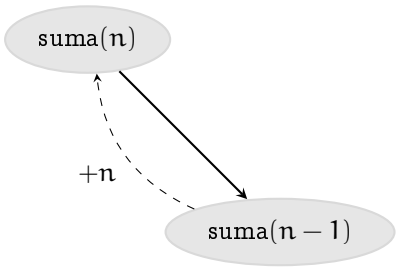
Per exemple, el predicat $s(n)$ de l'últim exemple es podria implementar en Haskell i en Python de la següent manera:

```haskell
suma ::  Integer -> Integer
suma 0 = 0
suma n | n > 0 = n + suma (n-1)
```

```python
def suma (n):
    if n==0:
        return 0
    else:
        return n + suma(n-1)
```

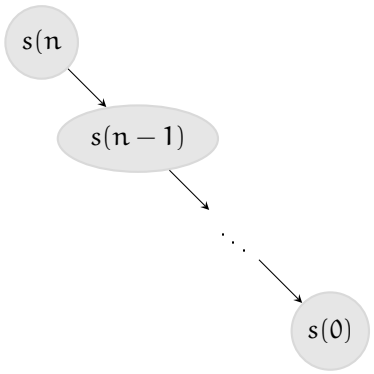
És molt important observar que totes les definicions recursives dels predicats anteriors juntament amb l'última recurrència així com totes les implementacions es corresponen amb la mateixa **idea recursiva** que en aquest cas consisteix a relacionar la suma dels n primers enters amb la suma dels $n-1$ primers enters.

Aquesta idea també es pot representar de manera gràfica:



En aquest diagrama la fletxa descendent indica la relació recursiva entre el problema, $s(n)$, i el subproblema, $s(n-1)$, que es representen com a nodes. La fletxa puntejada ascendent indica que la subsolució o resultat del subproblema es fa servir per a obtenir la solució del problema per a n.

Com que el subproblema (o subproblemes) es poden tornar a relacionar amb altres subproblemes recursivament, en expandir el diagrama anterior s'obté el que anomenarem **arbre de recursió**, que en el cas concret de l'exemple donaria un arbre amb una sola branca que representaríem simplificadament com a



En general serà convenient separar la idea recursiva d'una particular implementació o tipus de fórmula per a expressar-la.

Més concretament, per a qualsevol funció, f,

$$f : D \longrightarrow C$$

sempre existirà el corresponent predicat, P_f,

$$P_f : D \times C \longrightarrow \{V, F\}$$

de manera que

$$\forall c \in C \ \forall d \in D, \ \boxed{c = f(d)} \Leftrightarrow \boxed{P_f(c, d)}$$

A partir d'una mateixa idea recursiva sempre podrem donar definicions recursives que utilitzen f o P_f de manera equivalent.

Normalment sol ser més senzill escriure una relació recursiva per a f que no una definició lògica per a P_f. I més encara si la fórmula lògica ha de ser "executable" en, o traduïble a sistemes com ara Prolog.

3.1.1 Estructures recursives

De vegades interessa representar informació usant estructures definides recursivament com per exemple el que ací anomenarem **llista** la definició de la qual donem a continuació.

- La llista buida, $\emptyset$, és una llista.
- Si L és una llista, la mateixa amb l'addició a l'esquerra d'un element e qualsevol, que anomenarem $\mathtt{constr}(e, L)$, també és una llista.

Si $\mathbb{L}$ és el domini format per totes les possibles llistes, i $\mathcal{U}$ és qualsevol domini, la funció constr la descriurem com

$$\mathtt{constr} \ : \ \mathcal{U} \times \mathbb{L} \longrightarrow \mathbb{L}.$$

Aquesta funció s'anomena en altres contextos **constructor** de la corresponent estructura per que permet obtindre qualsevol llista del domini a partir del cas base, $\emptyset$. Per comoditat, abreviarem l'expressió $constr(e, L)$ com a $e|L$.

A partir de la definició, una llista no és més que una seqüència d'elements de qualsevol tipus. No obstant això, les escriurem usant claudàtors per que quede clar que estem parlant de llistes definides recursivament. Exemples de llistes serien

$$[] = \emptyset \qquad [1, 2, a, pep] \qquad [a, b, [1, c, 2], 0] \qquad [[][[]]]$$

La segona d'aquestes llistes es pot escriure també com

$$1|[2, a, pep] = 1|2|[a, pep] = 1|2|a|[pep] = 1|2|a|pep|\emptyset$$

Associades a les llistes definirem també les funcions bàsiques:

$$cap : \mathbb{L} \longrightarrow U,$$

que torna l'element que es troba en el primer lloc de la llista, i

$$cua : \mathbb{L} \longrightarrow \mathbb{L},$$

que torna la mateixa llista sense l'element que es trobava en el primer lloc de la llista.

Per a qualsevol llista no buida s'ha de complir necessàriament que

$$L = constr(cap(L), cua(L)) = cap(L)|cua(L)$$

Una vegada definida una estructura recursiva, es poden plantejar funcions recursives que les utilitzen. Per exemple, la funció recursiva que a partir d'una llista d'enters calcula la seua suma es podria especificar com

$$s(L) = \begin{cases} 0 & \textbf{si } L = \emptyset \\ s(cua(L)) + cap(L) & \textbf{si } L \neq \emptyset \end{cases}$$

Aquest tipus de funcions sobre llistes ademeten una traducció quasi directa a llenguatges com Prolog:

```
suma([],0).
suma([E|L],S) :-
   suma(L,SS),
   S is SS+E.
```

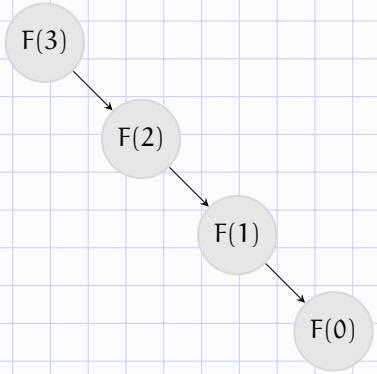
3.1.2 Tipus de recursió i exemples

Diem que una recursió és **lineal** quan la relació és només amb **un sol** cas anterior. S'anomena lineal perquè podem dibuixar el procés de càlcul per a un valor donat com una única cadena que acaba en un cas base.

Factorial

$$F(n) = \begin{cases} 1 & \textbf{si } n = 0 \\ n \cdot F(n-1) & \textbf{si } n > 0 \end{cases}$$

Aquest exemple és exactament igual que el de la suma dels n primers enters només canviant la suma per la multiplicació i tenint en compte que el cas base es correspon amb l'element neutre de cada operació.
L'arbre de recursió per a $n = 3$ seria



Aquesta representació arborescent es correspon amb la següent seqüència de substitucions, reescriptures o càlculs.

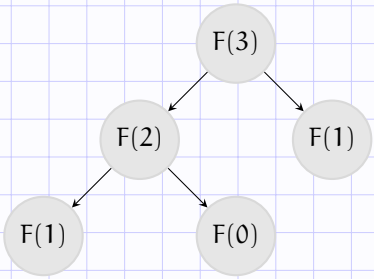
$$F(3) = 3 \cdot F(2) = 3 \cdot 2 \cdot F(1) = 3 \cdot 2 \cdot 1 \cdot F(0) = 3 \cdot 2 \cdot 1 \cdot 1 = 3! = 6.$$

Una recursió és **no lineal** o també **múltiple** si la relació és al mateix temps amb dos o més casos anteriors. En aquest cas el procés de càlcul o **arbre de recursió** té més d'una branca.

Successió de Fibonacci

$$F(n) = \begin{cases} n & \textbf{si } n \leq 1 \\ F(n-1) + F(n-2) & \textbf{si } n > 1 \end{cases}$$

L'arbre de recursió per a $n = 3$ seria



El càlcul de $F(3)$ representat per l'arbre anterior seria

$$F(3) = F(2) + F(1) = F(1) + F(0) + F(1) = 1 + 0 + 1 = 2.$$

La recurrència anterior la podem escriure com un predicat, $P_F(n, x)$, que siga cert quan $x = F(n)$.

$$\begin{cases} P_F(0,0) \wedge P_F(1,1) \\ \forall n > 1, \forall x_1, x_2 \in Z^+,\ P_F(n-1, x_1) \wedge P_F(n-2, x_2) \Rightarrow P_F(n, x_1 + x_2) \end{cases}$$

I aquest predicat es pot escriure també com un conjunt de regles en Prolog. Escrivim també el programa en Haskell que calcularia la funció F.

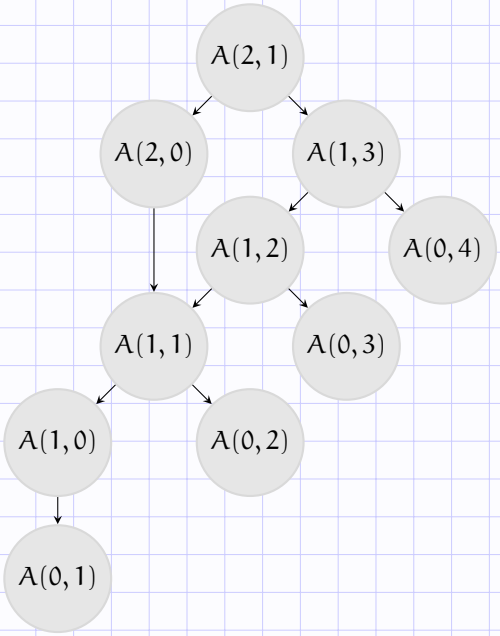
```
fib(0,0).
fib(1,1).
fib(N,R) :-
   N>1,
   M1 is N-1,
   M2 is N-2,
   fib(M1,X1),
   fib(M2,X2),
   R is X1+X2.
```

```
fib ::  Integer -> Integer
fib 0 = 0
fib 1 = 1
fib n |n>1 = fib (n-1) + fib (n-2)
```

Anomenem **recursió niada** quan es fa referència a algun cas anterior com a **argument** d'un altre cas anterior.

Funció d'Ackermann

$$A(m,n) = \begin{cases} n+1 & \textbf{si } m = 0 \\ A(m-1,1) & \textbf{si } m > 0 \wedge n = 0 \\ A(m-1,A(m,n-1)) & \textbf{si } m > 0 \wedge n > 0 \end{cases}$$



El càlcul de $A(2,1)$ seria

$$A(2,1) = A(1,A(2,0)) = A(1,\underline{A(1,1)}) = A(1,A(0,A(1,0))) = A(1,A(0,\underbrace{A(0,1)}_{2})) =$$

$$= A(1,\underbrace{A(0,2)}_{3}) = A(1,3) = A(0,A(1,2)) = A(0,A(0,\underline{A(1,1)})) =$$

Observem que en aquest punt arribem al càlcul de $A(1,1)$ per segona vegada. I com que ja sabem que $A(1,1) = 3$, continuem directament.

$$= A(0,\underbrace{A(0,3)}_{4}) = A(0,4) = 5.$$

L'especificació d'aquesta recursió en lògica de predicats i en Prolog es deixa com a exercici.
El programa recursiu en Haskell que calcula la funció A és pràcticament una traducció literal de la relació de recurrència.

```
ack ::  Integer -> Integer -> Integer
ack 0 n | n>=0 = n+1
ack m 0 | m>0 = ack (m-1) 1
ack m n | m>0,n>0 = ack (m-1) (ack m (n-1))
```

3.2 Principis d'inducció

El principi d'inducció és un esquema de raonament per a deduir fórmules que han de ser certes per a un domini infinit numerable.

Normalment el que es vol demostrar és de la forma $\forall n,\ P(n)$. I aquesta alternativa a utilitzar les regles d'inferència en lògica de predicats es pot enunciar informalment de la següent manera.

Principi d'inducció (informalment)

1. Un predicat, P, és cert per a un primer valor.
2. Suposem aquest predicat, P, cert per a un valor genèric.
3. Si a partir de l'anterior podem demostrar que P és cert per al valor següent,
4. aleshores es pot assegurar que P és cert per a tots els valors a partir del primer.

L'ítem 1 és el que anomenem cas base o **base d'inducció** (**BI**), i normalment s'identifica amb el valor 0 o el valor 1 o, en general, amb el valor enter més baix per al qual es puga complir P.

L'ítem 2 és l'anomenada **hipòtesi d'inducció** (**HI**), que consisteix a suposar cert el que es vol demostrar.

El **pas d'inducció** (**PI**) de l'ítem 3 és la demostració de la veracitat de P per al valor següent, la qual cosa (per la regla **II**) demostra la veracitat d'una implicació entre qualssevol valors consecutius.

I són precisament aquestes implicacions encadenades les que ens permeten arribar a la conclusió (ítem 4) gràcies a l'aplicació múltiple de la regla **EI** començant pel cas base.

Més formalment,

Principi d'inducció (bàsica)

$$\frac{\begin{array}{ll} P(n_0) & \textbf{BI} \\ & (*)\ \textbf{HI} \\ \forall n \geq n_0,\ \overbrace{P(n)}^{(*)} \Rightarrow P(n+1) & \textbf{PI} \end{array}}{\forall n \geq n_0,\ P(n)}$$

Exemple: És cert que

$$S_1(n) = \sum_{i=1}^{n} i = \frac{n}{2}(n+1)\ \forall n?$$

Primer comprovarem el valor de la fórmula per a alguns valors petits de n.

n	$S_1(n)$	$\frac{n}{2}(n+1)$
0	0	0
1	1	$\frac{1}{2}(2) = 1$
2	$1+2=3$	$\frac{2}{2}(3) = 3$
3	$1+2+3=6$	$\frac{3}{2}(4) = 6$

Com veiem, es pot agafar el cas $n = 0$ com a **base d'inducció**.
Suposem com a **hipòtesi d'inducció** que per a un **valor genèric** que anomenem n es compleix que

$$\left[S_1(n) = \frac{n}{2}(n+1)\right].$$

Escrivim la **HI** entre claudàtors per a indicar que es tracta d'una suposició.
El **pas d'inducció** consisteix a arribar a la fórmula per a l'enter següent, és a dir a

$$S_1(n+1) = \frac{n+1}{2}(n+2),$$

a partir de la hipòtesi, per a la qual cosa necessitarem relacionar dos casos consecutius.
En el cas de sumatoris quasi sempre resulta molt fàcil relacionar dos casos consecutius mitjançant la separació de l'últim terme,

$$S_1(n+1) = \sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1) = S_1(n) + (n+1).$$

És interessant adonar-se que la relació entre dos termes consecutius del sumatori juntament amb el fet que $S_1(0) = 0$ es correspon amb la **definició recursiva** del

mateix sumatori,

$$S_1(n) = \begin{cases} 0 & \text{si } n = 0 \\ S_1(n-1) + n & \text{si } n > 0. \end{cases}$$

Podem ara encadenar la relació recursiva amb la hipòtesi d'inducció i aleshores

$$S_1(n+1) \overset{(def)}{=} S_1(n) + (n+1) \overset{(HI)}{=} \frac{n}{2}(n+1) + (n+1) = (\frac{n}{2}+1)(n+1) = \frac{n+2}{2}(n+1),$$

que és, precisament, on es volia arribar.
Podem resumir la demostració esquemàticament de la següent manera:
BI: $S_1(0) = 0$,
HI: $\left[S_1(n) = \frac{n}{2}(n+1)\right]$,
PI: $S_1(n+1) \overset{(def)}{=} S_1(n) + (n+1) \overset{(HI)}{=} \frac{n}{2}(n+1) + (n+1) = \frac{n+2}{2}(n+1)$.

3.2.1 Principi d'inducció forta

En alguns casos, la relació entre un cas i el següent és difícil d'especificar però en canvi és fàcil la relació amb casos anteriors que no siguen l'immediatament anterior.

En aquests casos, es pot utilitzar una versió més forta (amb una suposició més àmplia) del principi d'inducció.

Principi d'inducció forta

$$\frac{\begin{array}{ll} P(n_0) & \textbf{BI} \\ & (*)\ \textbf{HI} \\ \forall n > n_0\ (\overbrace{\forall k < n\ P(k)}^{(*)}) \Rightarrow P(n) & \textbf{PI} \end{array}}{\forall n \geq n_0\ P(n)}$$

Exemple: l'n-èsim terme de la successió de Fibonacci, f_n, compleix que

$$f_n < 2^n.$$

Recordem primer la definició de la successió de Fibonacci,

$$f_0 = 0,\ f_1 = 1,\ \ \forall n > 1,\ f_n = f_{n-1} + f_{n-2},$$

i comprovem els primers valors de n,

n	f_n	2^n
0	0	1
1	1	2
2	1	4
3	2	8

La demostració usant el principi d'inducció forta es podria resumir com a

BI: $f_0 = 0 < 2^0 = 1$
$f_1 = 1 < 2^1 = 2$

HI: $[\forall k < n,\ f_k < 2^k]$

PI: $f_n \overset{(def)}{=} f_{n-1} + f_{n-2} \overset{(HI)}{<} 2^{n-1} + 2^{n-2} = (2+1)2^{n-2} < 4 \cdot 2^{n-2} = 2^n$

És important remarcar que en aquest cas són necessaris **dos** casos base perquè cada cas (recursiu) està sempre relacionat amb els dos casos immediatament anteriors.

3.2.2 Principi d'inducció general o Noetheriana

El principi d'inducció forta es pot generalitzar per a dominis més enllà dels enters en els quals es puga definir una relació d'ordre o fins i tot una relació encara més simple.

Preordre ben fundat

Una relació binària, $\preceq$, és un **preordre ben fundat** si compleix les propietats

a) **reflexiva** $(a \preceq a,\ \forall a)$,

b) **transitiva** $(a \preceq b \wedge b \preceq c \Rightarrow a \preceq c,\ \forall a, b, c)$,

c) i en el domini no hi ha successions infinites estrictament decreixents de la forma

$$a_1 \succ a_2 \succ \cdots$$

on la relació $\succ$ es defineix com a

$$a_i \succ a_{i+1} \Leftrightarrow (a_{i+1} \preceq a_i \wedge a_{i+1} \neq a_i)$$

Una conseqüència immediata d'un preordre ben fundat és l'existència d'un conjunt no buit, $\mathcal{M}$, d'elements **minimals** que constituiran els casos base d'inducció.

La inducció Noetheriana és exactament igual que la inducció forta només canviant la relació d'ordre usual sobre els enters pel preordre ben fundat definit sobre el domini corresponent.

Principi d'inducció general o Noetheriana

$$\frac{\begin{array}{ll}\forall z_0 \in \mathcal{M},\ P(z_0) & \textbf{BI}\\ & (*)\ \textbf{HI}\\ \forall z \notin \mathcal{M}\ (\overbrace{\forall t \prec z\ P(t)}^{(*)}) \Rightarrow P(z) & \textbf{PI}\end{array}}{\forall z\ P(z)}$$

Exemple: El valor de la funció d'Ackermann, $A(m,n)$, compleix que

$$A(m,n) > 2^{m+n-1}?$$

¿A partir de quins valors de m i n?

El primer que cal fer és definir un preordre ben fundat sobre parells d'enters de manera que aquesta relació siga compatible amb la relació de recurrència de la funció d'Ackermann en la pàgina 89.

Com que la referència niada a un cas anterior apareix com a segon argument d'una altra, basarem el preordre en el primer element del parell d'enters.

En particular, definim la relació, $\preceq$, entre parells d'enters com a

$$(k,\ell) \preceq (m,n) \Leftrightarrow (k < m) \vee (k = m \wedge \ell \geq n).$$

D'aquesta manera, els casos anteriors a què fa referència la definició recursiva són també anteriors segons $\preceq$.

Posposarem intencionadament la base d'inducció i enunciarem directament la hipòtesi.

HI: $$\left[\forall (k,\ell) \prec (m,n),\ A(k,\ell) > 2^{k+\ell-1}\right].$$

I a continuació el pas d'inducció.

PI:

Considerem primer el cas en què n és igual a zero.

$$A(m,n) \overset{(n=0)}{=} A(m-1,1) \overset{(HI)}{>} 2^{m-1} = 2^{m+0-1} = 2^{m+n-1}.$$

Vegem que el **PI** es compleix per a valors arbitraris de $m > 0$.

En el cas alternatiu, $n > 0$, es té que

$$A(m,n) \overset{(n>0)}{=} A(m-1,A(m,n-1)) \overset{(HI)}{>} A(m-1,2^{m+n-2}) \overset{(HI)}{>} 2^{m-2+2^{m+n-2}} \geq 2^{m+n-1}.$$

En l'última desigualtat hem considerat que

$$2^{m+n-2} = 2^{m-2} \cdot 2^n \overset{(m-2\geq 0)}{\geq} 2^0 \cdot 2^n = 2^n \overset{(n\geq 0)}{\geq} n+1,$$

per la qual cosa el pas d'inducció en aquest segon cas només serà cert si

$$m \geq 2.$$

Com veiem, la demostració és correcta llevat de la base d'inducció. Analitzem ara els valors més petits de la funció per a esbrinar quins són els valors associats a la base d'inducció.
En el cas $m = 0$ tenim

$$A_0(n) = A(0,n) = n+1,$$

i és clar que **no** es compleix que

$$A_0(n) > 2^{n-1}, \quad \forall n \geq 0.$$

En el cas $m = 1$,

$$A_1(n) = A(1,n) = \begin{cases} A(0,1) = 2 & \text{si } n = 0 \\ A(0,A(1,n-1))) = A_1(n-1)+1 & \text{si } n > 0, \end{cases}$$

o siga que $A_1(n) = n+2$. I tampoc no es compleix que

$$A_1(n) > 2^n, \quad \forall n \geq 0.$$

Si $m = 2$ tenim que

$$A_2(n) = A(2,n) = \begin{cases} A(1,1) = 3 & \text{si } n = 0 \\ A(1,A(2,n-1))) = A_2(n-1)+2 & \text{si } n > 0, \end{cases}$$

amb la qual cosa $A_2(n) = 2n+3$. I tampoc no es compleix que

$$A_2(n) > 2^{n+1}, \quad \forall n \geq 0.$$

Si considerem ara $m = 3$ obtenim la recurrència

$$A_3(n) = A(3,n) = \begin{cases} A(2,1) = 5 & \text{si } n = 0 \\ A(2,A(3,n-1))) = 2A_3(n-1)+3 & \text{si } n > 0, \end{cases}$$

que, a diferència de les anteriors, no és gens fàcil de calcular. Però en realitat només necessitem saber que la funció $f(n) = 2^{n+2}$ ve determinada per la recurrència

$$f(n) = \begin{cases} 4 & \text{si } n = 0 \\ 2f(n-1) & \text{si } n > 0, \end{cases}$$

per a concloure fàcilment que

$$A_3(n) > 2^{n+2} \quad \forall n \geq 0.$$

Per tant i finalment, podem dir que es compleix que $A(m,n) \geq 2^{m+n-1}$ per a parells de valors (m,n) majors o iguals (segons $\preceq$) que $(3,0)$.
En altres paraules, podem enunciar la **base d'inducció** com a

BI: $$A(3,0) = 5 > 2^{3-1} = 4,$$

i el que finalment s'ha demostrat com a

$$A(m,n) > 2^{m+n-1} \quad \forall m \geq 3, \ \forall n \geq 0.$$

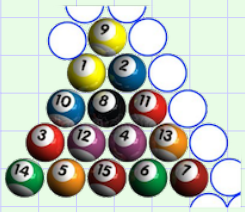
3.3 Problemes resolts i comentats

Problema 3.1:[billarREC] Les 15 boles de billar americà a l'inici formen un triangle equilàter de costat 5. I les 3 boles més internes en formen un de costat 2. De la mateixa manera, podríem afegir 21 boles més per a formar un triangle de costat 8. Si anomenem $B(n)$ el nombre de boles en tot el triangle de costat n,

a) Escriu la relació anterior entre els casos $B(2) = 3$, $B(5) = 15$ i $B(8) = 36$ com una definició recursiva vàlida en general.

b) Raona quin seria el valor de $B(n)$ per a tot n.

c) Defineix domini, codomini i imatge per a l'aplicació B i digues si és injectiva, exhaustiva i/o bijectiva.



a) La relació entre casos consecutius que es comenta és

$$B(5) = B(2) + 12, \qquad B(8) = B(5) + 21.$$

En general, si tenim un triangle de $B(n)$ boles de costat n, el següent triangle que podem formar en afegir una fila de boles que envolte totalment el triangle tindrà costat $n + 3$.

I el nombre de boles afegides serà tres vegades el costat menys tres (ja que les tres boles en els vèrtexs s'hauran comptat dues vegades. En resum,

$$B(n + 3) = B(n) + 3(n + 3) - 3.$$

O alternativament

$$B(n) = B(n - 3) + 3n - 3.$$

Perquè aquesta relació siga una definició recursiva vàlida cal afegir-hi els casos base. En aquest cas, caldran 3 casos base perquè la definició siga vàlida per a tot enter n (major o igual a 0).

$$\begin{cases} B(0) = 0,\ B(1) = 1,\ B(2) = 3, \\ B(n) = B(n-3) + 3n - 3, \qquad \text{si } n \geq 3. \end{cases}$$

b) Podem intentar trobar el valor de $B(n)$ per inspecció. Si apliquem la definició recursiva diverses vegades, arribem a

$$\begin{aligned} B(n) = B(n-3) + 3(n-1) &= B(n-6) + 3(n-4) + 3(n-1) = \\ &= B(n-9) + 3(n-7) + 3(n-4) + 3(n-1) = \cdots \end{aligned}$$

O en general a

$$B(n) = B(n-3k) + 3\sum_{\ell=1}^{k}(n - 3\ell + 2) = B(n-3k) + 3k(n+2) - 3^2\frac{k}{2}(k+1).$$

Ara podríem estudiar en quins casos $B(n-3k)$ arriba a ser un cas base, substituir-ho en l'expressió anterior i intentar arribar a una fórmula general per a $B(n)$.

En lloc d'això, deduirem relacions de recurrència equivalents però més senzilles per a $B(n)$.

Si tenim un triangle de boles de costat n i li llevem les boles de **dos costats**, i no tres com en la recurrència anterior, podem arribar a la recurrència equivalent,

$$\begin{cases} B(0) = 0,\ B(1) = 1, \\ B(n) = B(n-2) + 2n - 1, \quad \text{si } n \geq 2. \end{cases}$$

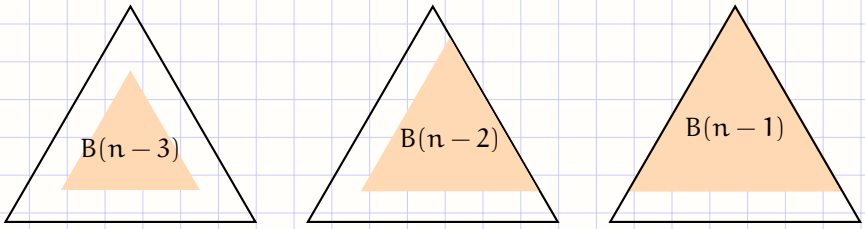
Alternativament, podem considerar un triangle de boles de costat n i llevar-li les boles de només **un costat**, amb la qual cosa tenim que

$$\begin{cases} B(0) = 0, \\ B(n) = B(n-1) + n, \quad \text{si } n \geq 1. \end{cases}$$

I aquesta última recurrència sí que es correspon trivialment amb el sumatori $\sum_{i=1}^{n} i$, per la qual cosa

$$B(n) = \frac{n}{2}(n+1).$$

Podem resumir les tres recurrències obtingudes per a $B(n)$ gràficament de la següent manera.



c) Ara que coneixem els valors que pren B, podem descriure-la com una aplicació de la forma

$$B : \mathbb{Z}^+ \longrightarrow \mathbb{Z}^+,$$

que així definida és injectiva i no exhaustiva. Injectiva perquè per a tot parell de valors del domini, (n, m), tals que $n \neq m$ necessàriament $B(n) \neq B(m)$. I no exhaustiva perquè valors del codomini com 4 o 5 no tenen antiimatge segons B.

Conseqüentment, B tampoc no és bijectiva.

Finalment, el conjunt imatge seria,

$$\mathrm{Im}(B) = \{m \in \mathbb{Z}^+ \mid \exists n \in \mathbb{Z}^+ \ : \ m = \frac{n}{2}(n+1)\} = \{0, 1, 3, 6, 10, 15, 21, 28, 36, \dots\}$$

Problema 3.2:[duesRecurrencies] Considera les relacions de recurrència que defineixen les funcions B i C, i demostra per inducció que $B(n) = C(n)$, per a tot $n \geq 0$.

$$\begin{cases} B(0) = 0, \\ B(n) = B(n-1) + n, \quad \text{si } n \geq 1 \end{cases} \qquad \begin{cases} C(0) = 0, \ C(1) = 1, \ C(2) = 3, \\ C(n) = C(n-3) + 3n - 3, \quad \text{si } n \geq 3 \end{cases}$$

Aquestes recurrències són dues de les tres que hem deduït informalment en l'exercici sobre les boles de billar americà (pàgina 100). Però ara es tracta de demostrar formalment per inducció que les dues són exactament equivalents.

Utilitzarem inducció forta sobre els enters (no negatius) guiats per la re-

currència de B. Per tant, tenim com a base d'inducció que

BI: $$B(0) = 0 = C(0).$$

Ara establim la hipòtesi.

HI: $$[B(k) = C(k),\ \forall k < n].$$

I a continuació ja podem demostrar el pas d'inducció.

PI:

$$B(n) \overset{(n\geqslant 1)}{=} B(n-1) + n \overset{(n\geqslant 2)}{=} B(n-2) + (n-1) + n \overset{(n\geqslant 3)}{=}$$
$$= B(n-3) + (n-2) + (n-1) + n = B(n-3) + 3n - 3 \overset{(n\geqslant 3)}{=} C(n).$$

Veiem que el pas d'inducció queda demostrat, però només si $n \geq 3$.

Aleshores, per a completar la demostració necessitem o bé ampliar la base d'inducció amb els casos $n = 1$ i $n = 2$, o equivalentment comprovar aquests dos casos separadament dins del pas d'inducció.

En qualsevol cas, tenim que

$$B(1) \overset{(def)}{=} B(0) + 1 = 1 = C(1),$$

$$B(2) \overset{(def)}{=} B(1) + 2 = 3 = C(2),$$

amb la qual cosa cloem la demostració.

Problema 3.3:[imparellParell] Considera la definició recursiva de la funció f i demostra per inducció que $f(n) > \frac{n-1}{2}$. Justifica el tipus d'inducció i raona clarament els diferents passos.

$$f(n) = \begin{cases} 1 & \text{si } n = 1, \\ f(n-1) + 1 & \text{si } n = 2\ell \text{ per a algun } \ell \geq 1, \\ f(n-2) + 1 & \text{si no.} \end{cases}$$

Calculem la funció per als primers valors de n i comprovem la desigualtat.

n	$f(n)$		$\frac{n-1}{2}$
1	1	$>$	0
2	2	$>$	$\frac{1}{2}$
3	2	$>$	1
4	3	$>$	$\frac{3}{2}$
5	3	$>$	2

A la vista de la taula podem fixar com a cas base $n = 1$.

BI: $$f(1) = 1 > \tfrac{1-1}{2} = 0.$$

Com que la recurrència no relaciona sols valors de n consecutius, farem servir inducció forta.

HI: $$\left[f(k) > \tfrac{k-1}{2},\ \forall k < n\right].$$

I a continuació estudiarem què passa amb el valor n. Però haurem de distingir entre el cas que n siga parell o senar.

PI:

$$[n = 2\ell] \quad f(n) = f(n-1) + 1 \overset{(HI)}{>} \frac{n-2}{2} + 1 = \frac{n}{2} > \frac{n-1}{2}$$
$$[n \neq 2\ell] \quad f(n) = f(n-2) + 1 \overset{(HI)}{>} \frac{n-3}{2} + 1 = \frac{n-1}{2}$$

Com que en els dos casos arribem al mateix resultat, podem concloure que es compleix el pas d'inducció en general.

Com a alternativa a la demostració anterior, podem transformar la recurrència original en una altra d'equivalent.

Primer reescrivim la recurrència,

$$\begin{cases} f(1) &= 1 \\ f(n) &= f(n-1) + 1 \quad \text{si } \exists \ell \geq 1 \ : \ n = 2\ell, \\ f(n) &= f(n-2) + 1 \quad \text{si } \exists \ell \geq 1 \ : \ n = 2\ell + 1. \end{cases}$$

I a continuació fem la substitució $f(n-1) = f(n-2) + 1$ en l'última equació, ja que $n = 2\ell + 1$ implica que $n - 1 = 2\ell$ per a $\ell \geq 1$.

$$\begin{cases} f(1) &= 1 \\ f(n) &= f(n-1) + 1 \quad \text{si } \exists \ell \geq 1 \ : \ n = 2\ell, \\ f(n) &= f(n-1) \quad\quad\ \ \text{si } \exists \ell \geq 1 \ : \ n = 2\ell + 1. \end{cases}$$

Aquesta última recurrència la podem escriure usant la funció $\mathrm{mod}(m,n)$ que dona el residu de la divisió entera entre m i n.

$$\begin{cases} f(1) &= 1 \\ f(n) &= f(n-1) + \mathrm{mod}(n-1,2) \quad \text{si } n \geq 2, \end{cases}$$

A partir d'aquesta recurrència equivalent es pot fer una demostració per inducció (bàsica) perquè la relació es dona només entre enters consecutius. Això es deixa com a exercici.

Alternativament, podem resoldre la recurrència, ja que es correspon trivialment amb el sumatori,

$$f(n) = f(1) + \sum_{i=1}^{n-1} \mathrm{mod}(i,2) = 1 + \underbrace{1+0+1+0+\cdots}_{n-1}$$

la solució del qual és

$$1 + \underbrace{1+0+1+0+\cdots+0}_{n-2} + 1 = 2 + \frac{n-2}{2} = 1 + \frac{n}{2},$$

si n és parell (tant el primer com l'últim terme del sumatori són 1), i

$$1 + \underbrace{1+0+1+0+\cdots+0}_{n-1} = 1 + \frac{n-1}{2},$$

en cas que n siga senar i per tant exactament la meitat dels termes del sumatori són 1.

A partir de l'anterior podem escriure la solució de la recurrència com a

$$f(n) = 1 + \lfloor \frac{n}{2} \rfloor = \lceil \frac{n+1}{2} \rceil,$$

d'on es dedueix trivialment que

$$f(n) > \frac{n-1}{2}.$$

Problema 3.4:[teSentit]
Per a cada una de les funcions, f i g, definides més avall, contesta:

a) Si la definició té sentit, calcula els valors de la funció fins $n = 4$ i, si pots, una expressió $\forall n$. Si alguna de les definicions no té sentit, explica clarament per què.

b) Quin tipus d'inducció hauries d'usar per a demostrar coses relacionades amb la funció?

c) Identifica domini, codomini i imatge, i digues si la funció és injectiva, exhaustiva i/o bijectiva.

$$\begin{cases} f(n) &= 2f(n+1) - f(n) - 1 & \text{si } n \geq 0 \\ f(0) &= 0 \end{cases} \qquad \begin{cases} g(n) &= g(n-1) - g(n-2) + 1 & \text{si } n > 0 \\ g(n) &= 0 & \text{si no} \end{cases}$$

a) Si entenem el cas recursiu de la primera recurrència com una fórmula per a calcular o "reescriure" el valor de $f(n)$, aleshores aquesta definició recursiva no té sentit perquè estaríem definint el cas n en funció d'ell mateix (definició circular) i d'un cas posterior, $n+1$, la qual cosa donaria lloc a una seqüència infinita de càlculs que no acabarien en cap cas base.

No obstant això, el que ens dóna la definició no és més que una equació. I se'ns diu que és vàlida per a tot $n \geq 0$.

Si ens hi fixem bé, el cas $n = 0$ està cobert tant per a la part recursiva com per al cas base. Si escrivim les dues equacions per a $n = 0$, tenim que

$$\begin{cases} f(0) &= 2f(1) - f(0) - 1 & , \\ f(0) &= 0 & , \end{cases}$$

la qual cosa significa que

$$f(1) = \frac{1}{2}.$$

De la mateixa manera podem ara aplicar la mateixa equació als casos $n = 1, 2, \ldots$ i calcular els primers valors de la funció:

n	$f(n)$
0	0
1	$\frac{1}{2}$
2	1
3	$\frac{3}{2}$
4	2

En resum, estem interpretant el cas recursiu com una fórmula per a calcular $f(n+1)$ a partir de $f(n)$. De fet, ho podem reescriure com a

$$2f(n+1) = 2f(n) + 1, \text{ si } n \geq 0.$$

Si ara dividim les dues parts de l'equació per 2 i fem un canvi de variable, arribem a la següent relació de recurrència equivalent:

$$\begin{cases} f(n) = f(n-1) + \frac{1}{2}, & \text{si } n \geq 1, \\ f(0) = 0 & . \end{cases}$$

A partir d'aquesta recurrència es veu que el valor de $f(n)$ és determinat per un sumatori,

$$f(n) = \cancel{f(0)}^{0} + \sum_{i=1}^{n} \frac{1}{2} = \frac{n}{2}.$$

La recurrència que defineix la funció g és una fórmula que ens permet calcular-la per a tot valor de n a partir de dos valors anteriors.

L'única cosa que té d'especial és que el cas base inclou el valor zero i **tots els negatius**. De fet, necessitem almenys el valor -1 per a poder calcular la funció a partir de $n = 1$.

Els valors de la funció fins a $n = 5$ es mostren en la taula següent.

n	$g(n)$	$\text{mod}(n,6)$
$\vdots$	0	
-1	0	
0	0	0
1	1	1
2	2	2
3	2	3
4	1	4
5	0	5

Si continuem calculant, veiem que $g(6) = 0$, amb la qual cosa podem deduir que es compleix que

$$g(n) = g(n+6), \quad \forall n \geq -1.$$

O en altres paraules, es tracta d'una funció periòdica el període de la qual és 6.

Per a poder escriure una expressió vàlida per a tot $n \geq 0$ podem fer servir la funció $\text{mod}(n,6)$ que també és periòdica i amb el mateix període. De fet, es compleix que

$$g(n) = \text{mod}(n,6) \quad \text{sempre que} \quad \text{mod}(n,6) \leq 2$$

En canvi quan $3 \leq \text{mod}(n,6) \leq 5$ els valors decreixen i es compleix que

$$g(n) = 5 - \text{mod}(n,6) \quad \text{quan} \quad 3 \leq \text{mod}(n,6) \leq 5$$

Per tant, podem expressar el valor de la funció g per a tot $n \geq 0$ com a

$$g(n) = \begin{cases} 0, & \text{si } n < 0, \\ \text{mod}(n,6), & \text{si } n \geq 0 \wedge 0 \leq \text{mod}(n,6) \leq 2, \\ 5 - \text{mod}(n,6), & \text{si } n \geq 0 \wedge 3 \leq \text{mod}(n,6) \leq 5. \end{cases}$$

O també com a

$$g(n) = \begin{cases} 0, & \text{si } n < 0, \\ \min(\text{mod}(n,6), 5 - \text{mod}(n,6)), & \text{si no.} \end{cases}$$

b) En el cas de la primera recurrència, com que el que s'estableix és una relació entre el valor de la funció per a dos enters consecutius, es podrà utilitzar el principi d'inducció bàsica.

A títol d'exemple, podem demostrar que la funció que defineix la recurrència és $f(n) = \frac{n}{2}$, de la següent manera.

BI: $f(0) = 0 = \frac{0}{2}$.

HI: $\left[f(n) = \frac{n}{2}\right]$.

PI: Escrivim primer el cas recursiu

$$f(n) = 2f(n+1) - f(n) - 1.$$

I ara apliquem la hipòtesi d'inducció,

$$\frac{n}{2} = 2f(n+1) - \frac{n}{2} - 1,$$

d'on s'obté que

$$2f(n+1) = n + 1,$$

i d'ací el resultat $f(n+1) = \frac{n+1}{2}$ que demostra el pas d'inducció.

En el cas de la segona recurrència la relació s'estableix entre un enter i els dos anteriors, per la qual cosa necessitarem utilitzar el principi d'inducció forta.

Com a exemple, demostrarem per inducció que $g(n) \leq 2$ per a tot $n \geq 0$.

Enunciem primer la hipòtesi i posposem la base d'inducció.

HI: $[g(k) \leq 2,\ \forall k < n]$.

En el pas d'inducció intentarem desfer-nos dels termes negatius que ens donaran problemes a l'hora d'aplicar la hipòtesi, ja que en aquest cas es tracta d'una fita superior.

PI:

$$g(n) \overset{(n\geq 1)}{=} g(n-1) - g(n-2) + 1 \overset{(n\geq 2)}{=} \cancel{g(n-2)} - g(n-3) - \cancel{g(n-2)} + 2.$$

Hem arribat a una relació directa entre $g(n)$ i $g(n-3)$,

$$g(n) = 2 - g(n-3) \quad \text{si} \quad n \geq 2$$

que podem tornar a aplicar perquè aparega només $g(n-6)$ en positiu i poder aplicar la hipòtesi. Aleshores,

$$g(n) \overset{(n\geqslant 5)}{=} g(n-6) - \not{2} + \not{2} \overset{(HI)}{\leq} 2,$$

amb la qual cosa conclou el pas d'inducció.

Com que el pas d'inducció només és cert per a $n \geq 5$, els valors menors o iguals a 4 s'hauran d'incloure en la base d'inducció.

BI:

$$\begin{aligned} g(n) &= 0 \leq 2 \quad \text{si} \quad n \leq 0 \\ g(1) &= g(0) - g(-1) + 1 = 1 \leq 2 \\ g(2) &= g(1) - g(0) + 1 = 2 \leq 2 \\ g(3) &= g(2) - g(1) + 1 = 2 \leq 2 \\ g(4) &= g(3) - g(2) + 1 = 1 \leq 2 \end{aligned}$$

c) La funció f la podem descriure com a

$$f : \mathbb{Z}^+ \longrightarrow \mathbb{R},$$

i, segons hem vist, és injectiva perquè

$$f(n) = \frac{n}{2} \neq \frac{m}{2} = f(m) \text{ si } n \neq m.$$

A més a més, és no exhaustiva i, per tant, no bijectiva. El conjunt imatge de f és

$$\mathrm{Im}(f) = \{x \in \mathbb{R} \mid \exists n \in \mathbb{Z}^+ \ : \ x = \frac{n}{2}\} = \{x \in \mathbb{R} \mid 2x \in \mathbb{Z}^+\}.$$

Quant a la funció g la descrivim com a

$$g : \mathbb{Z} \longrightarrow \{0, 1, 2\},$$

i no és injectiva (ja que, per exemple, $g(0) = g(6)$) i per tant no bijectiva. Però sí que és exhaustiva tal com l'hem definida, ja que els conjunts codomini i imatge de g coincideixen.

Problema 3.5:[fibonacci] Considera la definició recursiva de la successió de Fibonacci, $F(n)$, i demostra per inducció que $F(n) \leq n!$.

Com que en la recurrència de Fibonacci, s'estableix una relació entre tres termes consecutius, necessitarem utilitzar inducció forta.

A més a més, pel tipus de relació, necessitarem dos casos base consecutius que complisquen el que es vol demostrar.

BI:

$$F(0) = 0 \leq 1 = 0!$$
$$F(1) = 1 \leq 1 = 1!$$

La hipòtesi d'inducció serà

HI: $$[F(k) \leq k!,\ \forall k < n]$$

I en el pas d'inducció estudiem què passa per al valor n.

PI:

$$F(n) \overset{(n>1)}{=} F(n-1) + F(n-2) \overset{(HI)}{\leq} (n-1)! + (n-2)! = (n-1+1)(n-2)! < n!$$

Com que hem arribat al resultat correcte per a n fent servir la hipòtesi d'inducció, podem assegurar que la desigualtat és certa per a tot n.

Problema 3.6:[pseudoFibonacci] Considera la definició recursiva de la successió de Fibonacci, $F(n)$, i la definició recursiva de la funció $G(n)$. És possible expressar $G(n)$ en funció de $F(n)$ de manera no recursiva? Com? Raona la resposta.

$$G(n) = \begin{cases} n & \text{si } 0 \leq n \leq 1 \\ G(n-1) + G(n-2) + 1 & \text{si } n \geq 2 \end{cases}$$

Calcularem els primers valors de les dues funcions per a il·lustrar-ne el comportament.

n	$F(n)$	$G(n)$
0	0	0
1	1	1
2	1	2
3	2	4
4	3	7
5	5	12

Ara podríem cercar alguna relació entre els valors de les dues funcions, per a proposar una fórmula i demostrar-la per inducció.

En lloc d'això raonarem a partir de les dues definicions recursives. Escriurem primer la definició recursiva de G per als valors n i $n-1$,

$$G(n) = G(n-1) + G(n-2) + 1,$$
$$G(n-1) = G(n-2) + G(n-3) + 1.$$

Si ara restem les dues equacions anteriors, podem arribar a una nova equació que haurà de ser vàlida per a tot $n \geq 3$.

$$G(n) - G(n-1) = \underbrace{G(n-1) - G(n-2)} + \underbrace{G(n-2) - G(n-3)}.$$

Podem veure que l'equació anterior encara es pot simplificar més. Però no ho hem fet amb tota la intenció, perquè ara definirem una nova funció, H, com a

$$H(n) = G(n) - G(n-1),\ \forall n \geq 1$$

Independentment del(s) cas(os) base de la funció H, s'haurà de complir que

$$H(n) = H(n-1) + H(n-2),\ \forall n \geq 3.$$

Aquesta relació és exactament la mateixa que la que defineix F llevat dels casos base. Però es compleix que

$$H(2) = G(2) - G(1) = 1 = F(2)$$
$$H(1) = G(1) - G(0) = 1 = F(1)$$

i podem fixar $H(0) = 0 = F(0)$ en la definició de H.

Com a conseqüència, podem assegurar que es compleix que

$$F(n) = G(n) - G(n-1), \ \forall n \geq 1.$$

Com veiem, hem expressat F en funció de G (Tot i que caldria afegir que $F(0) = G(0)$). Però l'enunciat de l'exercici ens demana exactament el contrari.

Podem aleshores escriure la relació anterior, junt amb el fet que $G(0) = 0$, alternativament com a

$$G(n) = \begin{cases} 0 & \text{si } n = 0, \\ G(n-1) + F(n) & \text{si } n \geq 1, \end{cases}$$

la qual cosa no és més que una definició recursiva per a G en funció de F la solució de la qual és

$$G(n) = \sum_{i=1}^{n} F(i), \ \forall n \geq 0,$$

que és la definició de G en funció de F que se'ns demanava.

En el cas que haguérem arribat a l'expressió anterior per inspecció a partir de la primera taula de valors, caldria aleshores demostrar el resultat per inducció, tal i com es resumeix a continuació.

BI: $G(0) = 0 = \sum_{i=1}^{0} F(i)$, $G(1) = 1 = \sum_{i=1}^{1} F(i) = F(1)$.

HI: $\left[G(k) = \sum_{i=1}^{k} F(i), \ \forall k < n \right]$.

PI:

$$G(n) \overset{(n\leq 2)}{=} G(n-1) + G(n-2) + 1 \overset{(HI)}{=} \sum_{i=1}^{n-1} F(i) + \sum_{i=1}^{n-2} F(i) + 1 =$$

$$= \sum_{j=2}^{n} F(j-1) + \sum_{j=3}^{n} F(j-2) + 1 = \left(F(1) + \sum_{j=3}^{n} F(j) \right) + \underbrace{1}_{F(2)} = \sum_{j=1}^{n} F(j).$$

Problema 3.7:[sumaGeometrica] Demostra per inducció:

$$S(n) = \sum_{i=0}^{n} \frac{1}{2^i} = 1 + \frac{1}{2} + \frac{1}{4} + \cdots < 2.$$

Normalment quan es tracta de demostrar coses per inducció relacionades amb un sumatori se sol descompondre aquest deixant de banda l'últim terme del sumatori.

En aquest cas concret tindríem

$$S(n) = S(n-1) + \frac{1}{2^n}.$$

Però ocorre que si intentem aplicar la corresponent hipòtesi d'inducció en aquest cas concret arribaríem a

$$S(n) = S(n-1) + \frac{1}{2^n} \overset{(HI)}{<} 2 + \frac{1}{2^n}.$$

I resulta que la quantitat per la qual podem fitar el valor de $S(n)$ fent servir la **HI** és estrictament superior a 2 per a tot n.

La conclusió és que la relació recursiva amb què hem caracteritzat el nostre sumatori no ens permet demostrar aquest resultat directament mitjançant inducció.

Una alternativa podria consistir a demostrar algun resultat més general a partir del qual es puga deduir el nostre, com per exemple

$$S(n) = 2 - \frac{1}{2^n} < 2.$$

Una altra alternativa més interessant consisteix a obtenir una caracterització recursiva diferent del sumatori. En el nostre cas tenim que, si $n \geq 1$,

$$S(n) = 1 + \frac{1}{2} + \frac{1}{4} + \cdots \frac{1}{2^n} = 1 + \frac{1}{2}\left(1 + \frac{1}{2} + \cdots \frac{1}{2^{n-1}}\right) = 1 + \frac{1}{2}S(n-1).$$

Amb aquesta caracterització alternativa la inducció bàsica ens serveix perfectament per a demostrar el resultat com es mostra a continuació.

BI: $S(0) = 1 < 2$,

HI: $[S(n) < 2]$,

PI: $S(n+1) = 1 + \frac{1}{2}S(n) \overset{(HI)}{<} 1 + \frac{2}{2} = 2.$

3.4 Problemes proposats

Problema 3.8:[exponenciacio]
Considera la funció $G(n)$ definida més avall. Demostra per inducció que $G(n) \geq n!$

$G(n) = (((2^{\overbrace{2)^2)^{\cdots})^2}^{n-1}}$, $G(3) = (2^2)^2 = 16$, $G(4) = ((2^2)^2)^2 = 256$.

Problema 3.9:[binomials] Considera la següent propietat dels coeficients binomials

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}.$$

a) Digues quin(s) cas(os) base cal afegir perquè tinga sentit com a definició recursiva.

b) Anomena la funció com a F i defineix-la formalment explicitant els conjunts domini, codomini i imatge.

c) Digues si és injectiva, exhaustiva i/o bijectiva.

d) Demostra per inducció que el valor de la funció (a partir de la definició recursiva) és igual a

$$\frac{n!}{m!(n-m)!}.$$

Problema 3.10:[induccioMultiple] A partir de la definició recursiva de la funció f_n,

a) Calcula els seus valors fins a $n = 4$.

b) Demostra per inducció que $f_n > n!$

c) Demostra per inducció que $f_n < (n+1)!$

$$f_n = \begin{cases} \sum_{i=1}^{n} i f_{i-1} & \text{si } n > 0 \\ 1 & \text{si no} \end{cases}$$

Problema 3.11:[multiinduccio] Donada la definició recursiva de N_n,

$$N_n = \begin{cases} 1 & \text{si } n = 0 \\ \sum_{k=0}^{n-1} N_{n-k-1} N_k & \text{si } n > 0 \end{cases}$$

a) demostra per inducció que

$$N_n \geq 2^{n-1}.$$

b) ¿Es podria demostrar que

$$N_n \geq 2^n?$$

Explica què i com hauries de canviar en la primera demostració o explica per què no es pot.

4. Grafs i arbres

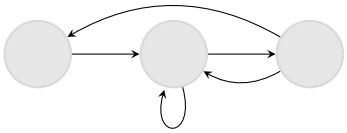
4.1 Grafs: definicions i propietats

Un **graf** és un parell de conjunts, (V, A), de manera que

a) el conjunt de **nodes** o **vèrtexs**, V, és un conjunt d'elements arbitrari, i

b) el conjunt d'**arcs** o **arestes**, A, és una **relació binària** entre elements de V.

Sovint representem els nodes com a punts o cercles i els arcs com a fletxes que uneixen parells de nodes.



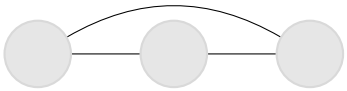
Un **graf no dirigit** (moltes vegades **graf**, simplement) és un graf el conjunt de nodes del qual és **no buit** i on el conjunt d'arcs és format per **parells no ordenats** de nodes. És a dir,

$$A \subseteq \{\{u, v\} \ : \ u, v \in V\}.$$

Alternativament es pot dir que en un graf no dirigit, A és una **relació binària simètrica** i **antireflexiva** ($a \not R a, \ \forall a$) en V. És a dir,

$$((u, v) \in A \Leftrightarrow (v, u) \in A) \ \wedge \ ((u, v) \in A \Rightarrow u \neq v)\,.$$

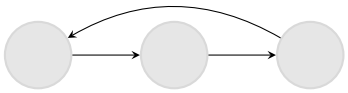
Quan un graf és no dirigit, se'l sol representar mitjançant cercles units per línies.



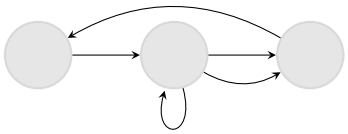
Un **graf dirigit** o també **digraf** és un graf el conjunt de nodes del qual és **no buit** i el conjunt d'arcs del qual és format per **parells ordenats** de nodes **diferents**. És a dir,

$$A \subseteq \{(u, v) \in V \times V \ : \ u \neq v\}.$$

També es pot dir que en un graf dirigit, A és una **relació binària antireflexiva**. Els grafs dirigits es representen mitjançant cercles units per fletxes i, per definició, no poden contenir ni **arcs paral·lels** (arcs entre un mateix parell ordenat de nodes) ni **bucles** (arcs entre un node i ell mateix).



Un **multigraf** o **graf amb arcs paral·lels** és un graf, (V, A), (del tipus que siga) on A és un **multiconjunt** en lloc d'un conjunt d'arcs.



Un **graf amb bucles** és un graf, (V, A), (del tipus que siga) on admetem relacions entre un node i ell mateix com a elements de A.

En alguns casos podrem definir el que anomenarem **graf nul**, o **graf buit**, $\emptyset = (\emptyset, \emptyset)$, que és un graf especial que no conté cap node i, per tant, cap arc.

Un **graf buit de n nodes** o **d'ordre n**, $\emptyset_n = \emptyset_V = (V, \emptyset)$, és un graf el conjunt de nodes del qual, V, té n nodes i no conté cap arc. Es té que

$$\emptyset = \emptyset_0 = \emptyset_\emptyset.$$

Dos nodes són **adjacents** si existeix un arc entre ells. Dos arcs són **adjacents** si tenen un node en comú.

Diem que un arc **incideix** o és **incident** en cada un dels nodes que el formen.

El **grau** d'un node u, $gr(u)$, és el nombre d'arcs que incideixen en u. En els grafs dirigits es pot distingir entre **grau d'entrada** o **igrau**, $gi(u)$ (nombre d'arcs que arriben a u), i **grau d'eixida** o **ograu**, $go(u)$ (nombre d'arcs que ixen de u).

Proposició

$$\forall G = (V, A), \ \exists k \in \mathbb{Z}^+ \ : \ \sum_{u \in V} gr(u) = 2k$$

En altres paraules, la suma dels graus dels nodes de qualsevol graf és parell.

La demostració és evident, ja que cada arc contribueix en 1 al grau de cada un dels dos nodes als quals incideix. Per tant, quan sumem per a tots els nodes tenim que

$$\sum_{u \in V} gr(u) = 2 \cdot |A|.$$

Corol·lari

En tot graf hi ha d'haver necessàriament un nombre **parell** de **nodes amb grau imparell**.

4.1.1 Connexió i descomposició

Diem que un graf, $G' = (V', A')$, és **subgraf** d'un altre, $G = (V, A)$, i ho escrivim com a $G' \subseteq G$, si

$$V' \subseteq V \wedge A' \subseteq A.$$

Donat un graf, $G = (V, A)$, i un subconjunt, $V' \subseteq V$, anomenem **subgraf induït per V'** el subgraf (V', A') on

$$A' = \{(u, v) \in A \ : \ u, v \in V'\}.$$

És a dir, és aquell subgraf de G que conté **tots** els arcs de G que enllacen nodes de V'. De vegades, si G' té $m \leq |V|$ nodes, direm que és un **subgraf d'ordre m** de G.

Donats dos grafs, $G_1 = (V_1, A_1)$ i $G_2 = (V_2, A_2)$, definim la seua **unió** com a

$$G_1 \cup G_2 = (V_1 \cup V_2, A_1 \cup A_2).$$

Diem que un graf, $G = (V, A)$, és un **graf complet** o que és un **clique**, si hi ha un arc en A per a tot parell de nodes en V.

Un graf, $G = (V, A)$, és **graf bipartit** respecte d'una bipartició (U, W) de V si tots els seus arcs relacionen nodes de U amb nodes de W o al revés. Equivalentment, el seu conjunt d'arcs ha de complir

$$A = \{(u, v) \ : \ u \in U \Leftrightarrow v \in W\}.$$

Donat un graf, $G = (V, A)$, anomenem **recorregut** tota successió de nodes i/o arcs adjacents que podem representar com a

$$(x_1, (x_1, x_2), x_2, (x_2, x_3), x_3, \ldots, x_{n-1}, (x_{n-1}, x_n), x_n),$$
$$((x_1, x_2), (x_2, x_3), \ldots, (x_{n-1}, x_n)),$$
$$(x_1, x_2, x_3, \ldots, x_{n-1}),$$

sempre que

$$\forall i,\ x_i \in V,\ \forall i < n,\ (x_i, x_{i+1}) \in A.$$

Diem que el recorregut comença en x_1 i acaba en x_n. Un exemple de recorregut que comença en v_2 i acaba en v_4 per al graf de la figura de la pàgina 121 seria

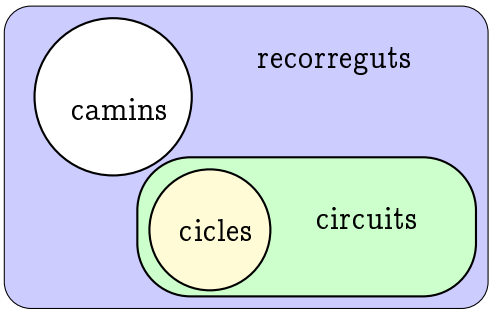
$$(v_2, v_3, v_1, v_2, v_4), \quad \text{o també} \quad (a_2, a_4, a_1, a_3).$$

Un **camí** és un recorregut on tots els nodes que conté són **diferents**. Un exemple de camí per al graf de la mateixa figura seria

$$(v_3, v_1, v_2, v_4) \quad \text{o també} \quad (a_4, a_1, a_3)$$

Un **recorregut circular** o **circuit** és un recorregut que comença i acaba en un mateix node. Un **cicle** és un circuit on l'únic node que es repeteix és l'inicial.

La relació entre els diferents tipus de recorreguts es mostra gràficament en la figura.

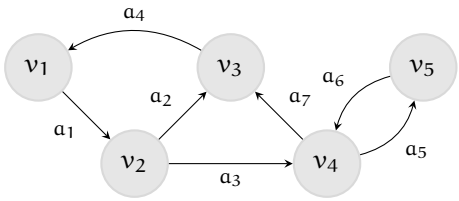


Exemples de circuit i cicle per al graf de la figura que es mostra després, són, respectivament,

$$(v_2, v_3, v_1, v_2, v_4, v_3), \quad \text{o també} \quad (a_2, a_4, a_1, a_3, a_7),$$

i

$$(v_3, v_1, v_2, v_4, v_3), \quad \text{o també} \quad (a_4, a_1, a_3, a_7).$$



Un recorregut/circuit en un graf $G = (V, A)$, s'anomena **eulerià** si recorre tots els **arcs** en A però només una vegada.

Un recorregut (circuit) en un graf $G = (V, A)$, s'anomena **camí (cicle) hamiltonià** si recorre tots els **nodes** en V però només una vegada (llevat del primer/últim en el cas de circuits).

4.1.2 Representació

Donat un graf, $G = (V, A)$, i $V = \{u_1, \ldots, u_n\}$, es defineix la seua **matriu d'adjacència** com una matriu, W, cada un dels coeficients de la qual val

$$w_{ij} = \begin{cases} 1 & \text{si } (u_i, u_j) \in A, \\ 0 & \text{si no.} \end{cases}$$

La matriu d'adjacència d'un graf no dirigit és simètrica. I els valors de la diagonal són zero si parlem de grafs sense bucles.

Per exemple, la matriu d'adjacència del graf de l'última figura seria:

$$W = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

4.1.3 Connexió

Un graf s'anomena **graf acíclic** o **sense cicles** si no conté cap cicle d'un o més arcs.

Un graf $G = (V, A)$ és **connex** si entre qualsevol parell dels seus nodes hi ha un **camí** que els uneix. És a dir, si

$$\forall u, v \in V,\ \exists v_1, \ldots, v_k \in V,\ k \geq 0\ :\ C_{u,v} = (u, v_1, \ldots, v_k, v),$$

i $C_{u,v}$ és un camí en G.

Un graf connex i acíclic s'anomena **arbre**.

Teorema (d'Euler)

Un graf (no dirigit) connex conté un **circuit** eulerià **sii** tots els seus nodes tenen grau parell.

Corol·lari

Un graf (no dirigit) connex conté un **recorregut no circular** eulerià **sii** només dos dels seus nodes tenen grau imparell.

En el graf de l'últim exemple, el recorregut

$$(v_2, v_3, v_1, v_3, v_4, v_5, v_4, v_3),$$

és eulerià. Sabem que no hi pot haver un circuit eulerià perquè els nodes v_2 i v_3 tenen grau 3.

En el mateix graf, el recorregut

$$(v_5, v_4, v_3, v_1, v_2),$$

és un camí hamiltonià i no existeix cap cicle hamiltonià. En canvi, sí que es pot trobar un cicle hamiltonià en el subgraf induït per $\{v_1, v_2, v_3, v_4\}$, per exemple:

$$(v_4, v_3, v_1, v_2, v_4).$$

4.1.4 Potències i clausures

La **potència n-èsima d'un graf** $G = (V, A)$, és un graf, $G^n = (V, A^n)$, on

$$(u, v) \in A^n \Leftrightarrow \exists v_1, \ldots, v_k \in V,\ k \geq n-1\ :\ C_{u,v} = (u, v_1, \ldots, v_k, v),$$

sent $C_{u,v}$ un camí en G. En altres paraules, els arcs de G^n són **camins** en G de longitud n.

Per a qualsevol relació binària, R, R^n és la **potència n-èsima de la relació** i es defineix com a

$$aR^n b \Leftrightarrow \exists x_1, \ldots, x_{n-1}\ :\ a = x_0, b = x_n, \forall 0 \leq i < n,\ x_i R x_{i+1}.$$

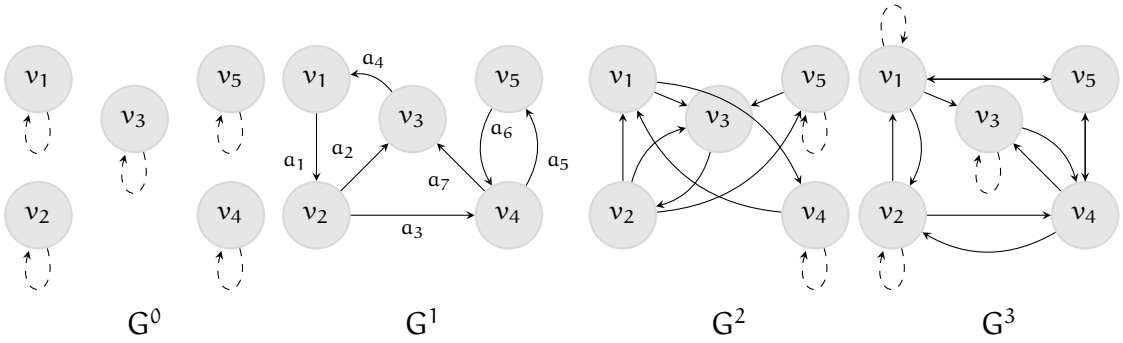
Es compleix que

$$G^0 = \emptyset_V, \qquad G^1 = G.$$

La **clausura transitiva** o de vegades simplement **clausura d'un graf** és el graf $G^+ = (V, A^+)$, on

$$A^+ = \bigcup_{n \geq 1} A^n.$$

Per a qualsevol relació binària, R, R^+ és la **clausura transitiva de la relació** que es defineix també com la menor relació transitiva que conté R.



En la figura es mostren les primeres potències del graf de la figura anterior (que coincideix amb G_1). En alguns d'aquests grafs es mostren bucles, ja que en aquests casos poden tenir interès. Per exemple, el bucle sobre v_3 en G_3 significa que hi ha un **cicle** que comença i acaba en v_3 de longitud 3, que és

$$(v_3, v_1, v_2, v_3).$$

La clausura transitiva del graf anterior es correspon amb un graf complet, ja que es tracta d'un graf connex.

Proposició

La clausura transitiva de tot graf **connex** és un graf **complet**.

A partir de la definició de connex, és evident que hi ha d'haver un camí de longitud k, $1 \leq k \leq |V|$ entre tot parell de nodes. I aquest camí apareixerà com un arc en la corresponent potència del graf.

En el cas que considerem grafs amb bucles, o relacions binàries en general, té sentit definir la **clausura reflexiva**. En el cas d'una relació, R, es defineix com la relació reflexiva més petita que la conté. O equivalentment, la mateixa relació però on a més a més tot element del domini està relacionat amb ell mateix. O siga,

$$R^0 \cup R.$$

En el cas de grafs, això es correspon amb afegir bucles en tots els nodes (que no els tingueren ja).

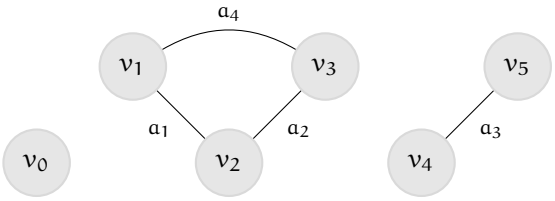
La **clausura transitiva i reflexiva** d'una relació, R, és la relació transitiva i reflexiva més petita que la conté i és equivalent a

$$R^* = \bigcup_{i \geq 0} R^i.$$

Donat un graf $G = (V, A)$, anomenem **component connexa** de G qualsevol subgraf **connex** de G que siga **maximal** respecte de la relació d'ordre induïda per la inclusió entre grafs.

En la figura es mostra un graf d'ordre 6 que té 3 components connexes que són determinades pels subgrafs induïts per

$$\{v_0\}, \quad \{v_1, v_2, v_3\}, \quad \{v_4, v_5\}.$$



Proposició

Tot graf no dirigit, $G = (V, A)$, té almenys $|V| - |A|$ components connexes.

La demostració és per inducció sobre el nombre d'arcs.
Siga $A = \{a_1, \ldots, a_{|A|}\}$ i

$$G_n = (V, \{a_1, \ldots, a_n\}), \ \forall n, \ 0 \leq n \leq |A|.$$

Anomenem c_n el nombre de components connexes del subgraf G_n.

BI: La base d'inducció es correspon amb $n = 0$ arcs. En aquest cas G_0 té tantes components connexes com nodes i per tant

$$c_0 = |V| \geq |V| - 0 = |V| - |A|.$$

HI:

$$[c_n \geq |V| - n].$$

PI: Tenim el subgraf G_n que té c_n components connexes i volem caracteritzar el nombre de components connexes del subgraf G_{n+1}, c_{n+1}.
Però l'única diferència entre G_n i G_{n+1} és l'arc a_{n+1}. I hi ha dos casos possibles:

a) a_{n+1} uneix dos nodes **dins** de la mateixa component connexa. Aleshores

$$c_{n+1} = c_n \overset{(HI)}{\geq} |V| - n > |V| - (n+1).$$

b) a_{n+1} uneix dos nodes en **diferents** components connexes. Aleshores

$$c_{n+1} = c_n - 1 \overset{(HI)}{\geq} |V| - n - 1 = |V| - (n+1).$$

Com que en els dos casos arribem al mateix, podem concloure finalment que

$$\forall G = (V, A) \ : \ |A| = n, \ c_n \geq |V| - n = |V| - |A|.$$

Corol·lari

Tot graf no dirigit connex té almenys $|V| - 1$ arcs. És a dir,

$$|A| \geq |V| - 1.$$

4.1.5 Demostració del teorema d'Euler

La demostració del teorema d'Euler és senzilla en un sentit però significativament més complexa en l'altre. La formulació més general amb multigrafs amb bucles no només no afegeix complexitat si no que fa una part de la demostració més natural.

Per conveniència demostrarem primer un lema que estableix l'existència de cicles en un multigraf (sense bucles).

Lema

Si G és un multigraf no dirigit i connex on el grau de tots els seus nodes és major o igual que 2, aleshores ha d'existir algun cicle en G.

La demostració fa servir el **principi de les caixes**. Comencem per un node arbitrari, u_1, dels $n \geq 2$ nodes de G i construim una seqüència de nodes diferents, cada un adjacent a l'anterior, fins que arribem a un node u_k que siga adjacent a algun node u_ℓ anterior en la seqüència ($\ell < k$).

$$(u_1, u_2, \ldots, u_k)$$

Aquesta seqüència sempre es podrà construir ja que tots els nodes tenen com a mínim altres 2 nodes adjacents. A més a més, el valor de k complirà necessàriament,

$$2 \leq k \leq n$$

i en tots els casos el node u_k tindrà (almenys) un node adjacent, $u_\ell, \ell < k$, a banda de u_{k-1} per tindre grau major o igual a 2.

Com a conseqüència podem afirmar que

$$(u_\ell, u_{\ell+1}, \ldots, u_{k-1}, u_k, u_\ell)$$

és un cicle en G.

Teorema d'Euler

Siga $G = \langle V, A\rangle$ un multigraf amb bucles no dirigit i connex. G conté un **circuit** Eulerià **sii** tots els seus nodes tenen grau parell major o igual que 2.

Organitzarem la demostració en dues parts, una pera cada sentit de la coimplicació. També, per comoditat, expressarem com a

$$C_{CE}(G)$$

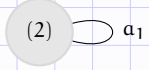
el fet que G conté un cicle Eulerià.

$C_{CE}(G) \Rightarrow \forall u \in V, \exists k \geq 1 : gr(u) = 2k$ La demostració és per reducció a l'absurd. I el que suposem és que hi ha en G un node, v, el grau del qual és senar.
Però resulta que el circuit Eulerià que ha d'existir ha de passar per tots els arcs (una vegada) i per tots els nodes (per ser connex) una o més vegades.
Això vol dir que haurà d'entrar i eixir, una o més vegades de cada node. Però en el cas de v, per tindre grau senar, arribaria un moment en què no es podria eixir. La qual cosa és impossible.

$\forall u \in V, \exists k \geq 1 : gr(u) = 2k \Rightarrow C_{CE}(G)$
La demostració és per inducció sobre el nombre d'arcs, n, en el multiconjunt d'arcs de G. Primer que res, establim una numeració arbitrària de tots els arcs.

$$A = \{a_1, \ldots, a_n\}$$

BI: Per a un sol arc, l'única possibilitat és un únic node amb un bucle (per tant amb grau 2, mostrat entre parèntesis). En eixe cas, el bucle constitueix un circuit Eulerià.

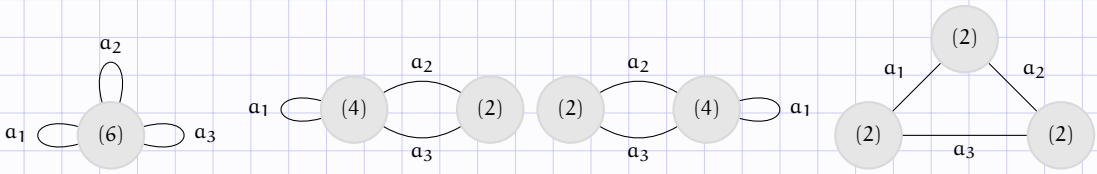


circuit: (a_1)

Amb 2 arcs hi ha dos casos. Un node amb dos bucles (grau 4) , o dos nodes amb dos arcs paral·lels (grau 2 cada un). En els 2 casos, la seqüència formada pels 2 arcs és un circuit Eulerià.

circuit: (a_1, a_2)

Entre els casos que hi ha amb 3 arcs, apareix el graf més simple (no multigraf i sense bucles) que estaria format per tres nodes totalment connectats entre ells i que conté també un circuit Eulerià.



circuit: (a_1, a_2, a_3)

Estrictament només necessitem el cas $n = 1$ com a base d'inducció.

HI: Com a hipòtesi d'inducció suposarem que tot multigraf amb bucles connex amb nodes de grau parell major que zero, i amb estrictament menys de n arcs, conté un circuit Eulerià.

Altrament dit, necessitarem **inducció forta**. I podem expressar formalment la hipòtesi com

$$\left[\forall G' = \langle V', A' \rangle : |A'| = \ell < n, \left(\forall u \in V',\ gr(u) \geq 2\right) \Rightarrow C_{CE}(G')\right]$$

PI: Considerarem ara un multigraf ,G, amb exactament n arcs i m nodes, tots ells amb grau parell major o igual que 2. És a dir,

$$G = \langle V, A \rangle, \quad V = \{v_1, \ldots, v_m\}, \quad A = \{a_1, \ldots, a_n\}$$

El multigraf G ha de contindre un cicle. Be per l'existència de bucles, be com a conseqüència del lema anterior. Ja que fins i tot sense comptar els bucles tots els nodes tenen graus majors o iguals que 2 per ser G connex.

Si expressem aquest cicle com a seqüència de nodes i arcs,

$$\phi = (u_1, b_1, u_2, b_2, \ldots, u_k, b_k, u_1), \quad \begin{array}{ll} 1 \leq k \leq n & \\ \forall i, & u_i \in V \wedge b_i \in A, \\ \forall i, j : i \neq j, & u_i \neq u_j \wedge b_i \neq b_j \end{array}$$

Si eliminem ara el conjunt d'arcs

$$B = \{b_1, \ldots, b_k\},$$

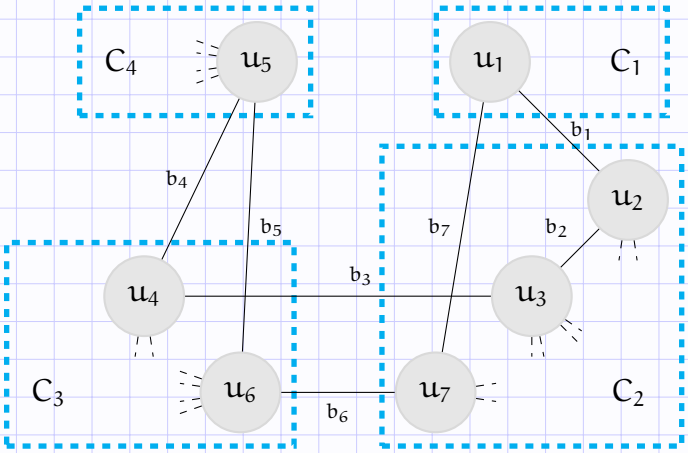
del multigraf G, el resultat serà un multigraf G' amb els mateixos nodes i amb $n' = n - k$ arcs on

$$0 \leq n' \leq n - 1$$

El multigraf G' no serà necessàriament connex i tindrà entre 1 i m components connexes,

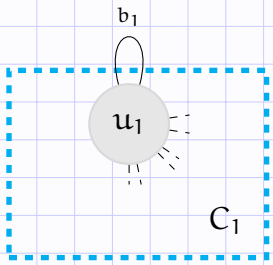
$$C_1, \ldots, C_p, \quad 1 \leq p \leq m,$$

que contindran tots els nodes de G.
El grau dels nodes de ϕ haurà decrescut en 2 i per tant tots els nodes de G' tindran grau parell només que alguns (dels de ϕ) podran tindre grau zero.
Quant a les components connexes, n'hi haurà de dos tipus: a) les formades per un únic node de ϕ de grau zero, i b) les formades per un o més nodes de ϕ i possiblement altres, tots ells amb grau major o igual a 2.
No pot haver cap component sense nodes de ϕ si el multigraf original era connex.
En la figura es mostra un exemple amb 4 components connexes i un cicle ϕ amb 7 arcs.



En el cas que totes les components connexes són nodes de grau zero, el circuit ϕ ja és un circuit Eulerià per al multigraf G.
En cas contrari es pot construir un circuit Eulerià de la següent manera. Comencem en u_1 i anem recorreguent el cicle ϕ i afegint els seus arcs al circuit en construcció començant per l'arc b_1.
Quan arribem per primera vegada a una component connexa, si el corresponent node, u_i, té grau zero continuem (la component només té el node u_i) i si no, la component connexa és un multigraf connex amb estrictament menys de n arcs.
Per **hipòtesi d'inducció**, la component contindrà un circuit Eulerià que comença i acaba en u_i i que podrem insertar en el circuit. A continuació afegim l'arc b_i i continuem.
Si en algun moment tornem a una component connexa ja visitada, ens limitarem a entrar i eixir del corresponent node u_j usant els arcs b_{j-1} i b_j ja que tots els altres arcs d'aquesta component ja s'hauran insertat en el circuit la primera vegada que s'ha visitat.
Després d'afegir l'últim arc de ϕ, b_k, arribarem de nou al node u_1 i haurem completat un circuit Eulerià.
El cas més senzill és quan el circuit és un bucle, $\phi = (u_1, b_1, u_1)$. Aleshores només hi hauria una única component connexa que seria tot el multigraf sense el bucle ϕ

i per tant amb exactament $n-1$ arcs. Aquesta component contindria un circuit Eulerià per **hipòtesi d'inducció** i només caldria afegir el bucle per tal d'obtindre un circuit Eulerià de tot G.



4.2 Arbres: definicions i propietats

Ja hem definit un arbre com un graf connex i acíclic. Però aquesta definició no respon a la idea de jerarquia que normalment associem als arbres. Per això, aquests arbres s'anomenen també **arbres sense arrel**.

En un arbre sense arrel no hi ha cap node que siga origen de cap jerarquia. Tampoc no podem parlar de cap relació jeràrquica entre nodes (pare, fill, ancestre, etc.).

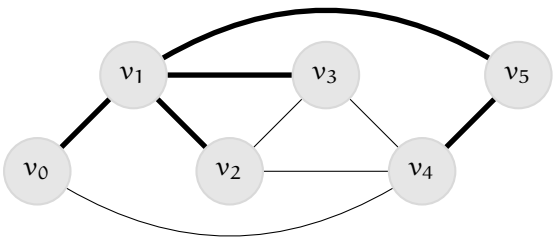
Entre qualssevol dos nodes d'un arbre sense arrel hi ha un únic camí.

A qualsevol subgraf d'un arbre que siga arbre l'anomenem **subarbre**.

Si suprimim un arc en un arbre sense arrel, el resultat és un graf no connex amb dues components connexes cada una de les quals és un subarbre.

Donat un graf connex $G = (V, A)$, diem que $G' = (V, A')$ és un **arbre generador** o **arbre d'extensió** de G **sii** $G' \subseteq G$ i G' és un arbre.

En la figura es mostra un graf i un dels seus arbres generadors (marcat amb línies més grosses).



Proposició

Tot graf connex conté almenys un arbre generador.

Demostració (informal):
Suposem que el graf **no** és arbre (altrament ja estaria demostrat). En aquest cas contindrà almenys un cicle.
Si eliminem qualsevol arc del cicle tenim un nou graf connex amb un arc menys i podem tornar al principi del raonament.
En repetir aquest raonament arribarem necessàriament a un graf connex i sense cicles.

4.2.1 Arbres amb arrel

Un **arbre amb arrel** és un arbre (graf connex i acíclic) en què distingim un node especial que anomenem **arrel**.

Com a conseqüència de la definició anterior tenim que:

a) Els nodes que no són arrel s'organitzen en $m \geq 0$ conjunts disjunts que indueixen arbres.

b) Els $m \geq 0$ nodes adjacents a l'arrel, r, se'ls anomena **fills** de r i se'ls distingeix com a arrels dels corresponents $m \geq 0$ subarbres. També es diu que r és **pare** dels seus nodes adjacents.

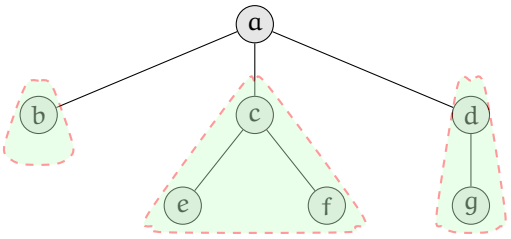
c) Els nodes d'un arbre amb arrel s'organitzen també en nivells. El **nivell** o **profunditat d'un node** és la longitud (en nodes) del camí des de l'arrel fins al node. L'arrel està a profunditat 1, els fills a profunditat 2, etc.

d) La **profunditat d'un arbre** es defineix com el nivell màxim dels seus nodes. I la **talla** o **grandària** de l'arbre és el nombre de nodes que conté.

e) Els nodes que no tenen fills s'anomenen **fulles**.

Ens referirem a l'arbre que té per arrel r com a $T(r)$ o moltes vegades simplement com a r. La profunditat d'aquest arbre l'escriurem com a $h(T(r))$ o $h(r)$ i la seua grandària com a $|T(r)|$ o $|r|$.

En la figura es mostra un arbre amb arrel de grandària 7 i profunditat 3 que té per arrel el node a. S'indiquen a més a més els 3 subarbres de a que tenen per arrels b, c i d. L'arbre té 4 fulles en total: b, e, f i g que són a profunditat (o nivell) 2, 3, 3 i 3 respecte de l'arbre a.



Els arbres amb arrel es poden veure com a estructures de dades que representen diferents classes de jerarquies, per la qual cosa admeten també i de manera natural definicions recursives com la que esbossem a continuació.

a) Un graf d'un node és un arbre amb arrel (l'arrel és l'únic node).

b) Un arbre amb arrel, $T(r)$, de grandària major que 1 és format per $k > 1$ subarbres amb arrels

$$T(r_1), \ldots, T(r_k),$$

que estan units amb r mitjançant arcs.

4.2.2 Arbres ordenats o m-aris

Un **arbre ordenat** d'ordre m o **arbre m-ari** és un arbre amb arrel on **tot** node té exactament m subarbres **distingibles** que poden ser buits.

L'**arbre buit** (cap node) es considera un cas especial d'arbre ordenat.

Es pot dir que en els arbres ordenats m-aris cada node té una **seqüència** de fills de longitud m, mentre que en els arbres amb arrel cada node té un **conjunt** de fills.

En un arbre m-ari hi ha un màxim de m^{h-1} nodes a profunditat h.

Un arbre m-ari està o és un **arbre ple fins a profunditat h sii** té exactament

$$\sum_{i=1}^{h} m^{i-1} = \frac{m^h - 1}{m - 1}$$

nodes a profunditat menor o igual que h.

Un arbre m-ari està o és un **arbre ple** si està ple fins a la seua profunditat.

En un arbre m-ari **ple** de n nodes es compleix que

$$n = \frac{m^h - 1}{m - 1},$$

o alternativament

$$h = \log_m (1 + n(m-1)) \overset{(n\geq 1)}{\leq} 1 + \log_m n.$$

Una característica fonamental dels arbres ordenats com a estructura de dades és la seua profunditat. I en molts casos és clau que aquesta siga logarítmica respecte al nombre de nodes.

L'exigència de profunditats logarítmiques motiva diverses definicions que afegeixen condicions a l'estructura particular dels arbres ordenats.

Un arbre m-ari de profunditat h està o és un **arbre complet** **sii**

a) està **ple fins a profunditat** $h-1$,

b) per als m fills de qualsevol node, $s_1, \ldots, s_m$ es compleix que

$$|s_1| \geq \ldots \geq |s_m|.$$

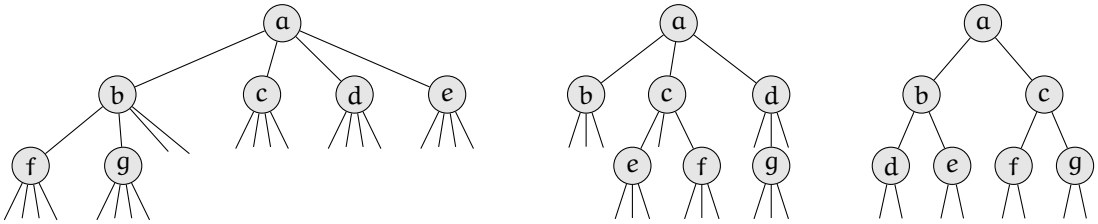
Un arbre m-ari està o és un **arbre equilibrat** **sii**

a) la màxima diferència entre qualsevol parell dels m subarbres és de 1,

b) els m subarbres són **equilibrats**.

Els **arbres binaris** són el cas particular dels m-aris quan $m = 2$. Però són especialment importants tant per motius teòrics com, especialment, pràctics i d'implementació.

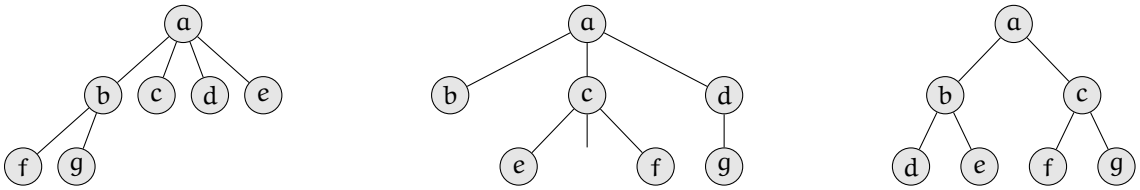
Quan $m = 2$ els dos fills de cada node se solen anomenar **esquerre** i **dret**, respectivament.



En la figura es mostren 3 arbres ordenats d'ordres 4, 3 i 2 respectivament. Els arbres contenen 22, 15 i 8 subarbres buits, 5, 4 i 4 fulles, encara que tots tres tenen 7 nodes i són de profunditat 3.

L'arbre de l'esquerra, que és 4-ari, és un arbre **ple fins al nivell 2** i **complet**, mentre que l'arbre de la dreta és **ple** (la qual cosa implica **ple fins al nivell 3** i **complet**).

Llevat que es tinga un interès especial, no serà normal representar tots els subarbres buits com s'ha fet en la figura anterior. No obstant això, si un node té menys fills que l'ordre de l'arbre ha de quedar clar quin dels m fills és cada un. En particular, els arbres anteriors els dibuixarem també com en la figura següent.



4.3 Grafs i arbres amb contingut

Fins ara els grafs i arbres són estructures que representen diferents tipus de relacions dins d'un conjunt de nodes el nom dels quals no és important. De fet, sovint els nodes es representen com a cercles que no tenen cap nom.

Però de vegades és interessant emmagatzemar o assignar alguna informació als nodes o als arcs. Aquesta informació pot ser de qualsevol mena, però els casos més importants i interessants són quan es tracta d'un **pes** (valor numèric normalment positiu que indica una quantitat d'alguna cosa: diners, distància, cost, etc.), una **clau numèrica** (valor numèric enter que pot ser únic o no: normalment un número d'ordre, una prioritat, etc.) o una **clau alfanumèrica** (cadena de caràcters que representa un identificador normalment únic: un DNI, un nom d'algú, un codi, etc.).

Un **graf amb nodes ponderats** és un graf $G = (V, A)$ i una aplicació,

$$p : V \longrightarrow \mathbb{R}^+.$$

Un **graf amb arcs ponderats** o simplement **graf ponderat** és un graf $G = (V, A)$ i una aplicació,

$$p : A \longrightarrow \mathbb{R}^+.$$

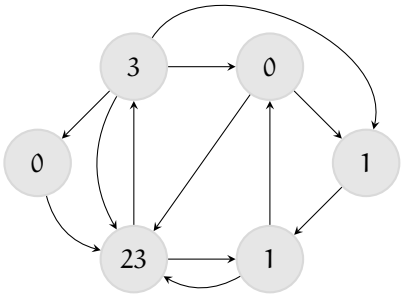
A vegades representarem ambdós tipus de grafs ponderats com a

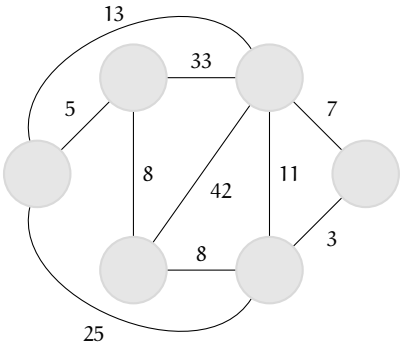
$$G = (V, A, p).$$

En els grafs ponderats es pot associar un pes a cada camí i també a tot el graf. Alguns exemples de problemes amb grafs ponderats són l'obtenció de camins de pes mínim, de l'arbre d'extensió minimal o de la bipartició de cost mínim (o màxim).

En la figura següent a l'esquerra es mostra un graf (dirigit) amb nodes ponderats que podria correspondre a relacions d'amistat en una minixarxa social on el pes de

cada node significa el nivell d'influència o popularitat de cada membre.





En la mateixa figura a la dreta es mostra un graf ponderat que podria correspondre a un conjunt de ciutats i les connexions per carretera entre elles. En aquest cas el pes de cada arc podria correspondre a la distància en quilòmetres de cada tram de carretera.

Un **arbre amb claus** és un **arbre** (del tipus que siga) format per un conjunt de nodes, V, un conjunt d'arcs, A, i una aplicació $c : V \longrightarrow \mathcal{C}$, de manera que $c(u)$ és la clau associada al node u.

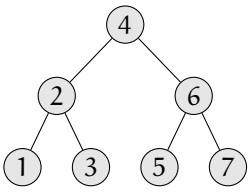
Quan no puga haver-hi confusió, escriurem u per a referir-nos a $c(u)$. És més, moltes vegades quan els nodes només continguen claus i aquestes siguen úniques definirem directament

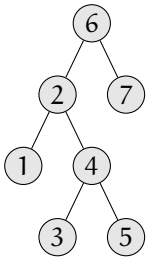
$$\forall u \in V,\ c(u) = u.$$

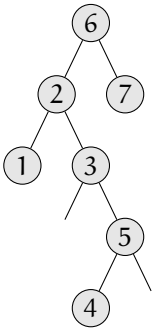
Un **arbre binari de cerca** és un arbre binari amb claus de manera que en $\mathcal{C}$ hi ha una relació d'ordre tal que per a tot node, a, es compleix que

$$\forall x, y \in V\ :\ x \in esq(a) \wedge y \in dre(a),\ x < a < y.$$

En la figura següent es mostren tres arbres binaris de cerca de 7 nodes i de profunditats 3, 4 i 5 respectivament.







Un **monticle (de mínims)** és un arbre amb claus numèriques de manera que en $\mathcal{C}$ hi ha una relació d'ordre tal que per a tot node, a, es compleix que

$$\forall x \in V \ : \ a = pare(x), \ a \leq x.$$

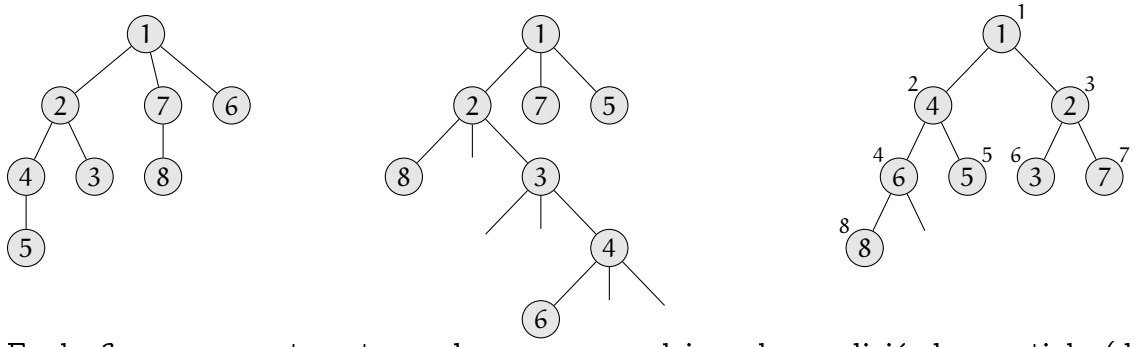
A vegades es poden definir monticles que són **conjunts** d'arbres. És a dir, **boscos**.

Els monticles són en general estructures arborescents que són ideals per a implementar un tipus de dades que s'anomena **cua de prioritat** i que suporta operacions com obtenció/eliminació del mínim, inserció de nous elements, canvis de prioritat o eliminacions.

Perquè unes operacions o altres puguen ser eficients, cal restringir fortament l'estructura arborescent, i això dona lloc a diferents tipus de monticles.

Un **monticle binari (de mínims)** és un arbre binari **complet** que és monticle.

El fet de ser complet permet identificar un monticle binari d'ordre n amb una n-tupla de valors, la qual cosa fa que moltes operacions es puguen fer de manera molt eficient.



En la figura es mostren tres arbres que compleixen la condició de monticle (de mínims). El de l'esquerra és un arbre amb arrel, el del mig és un arbre ternari i el de la dreta és un arbre binari complet i, per tant és un **monticle binari.**

La representació d'aquest monticle binari com a tupla seria

1	2	3	4	5	6	7	8
1	4	2	6	5	3	7	8

4.4 Problemes resolts i comentats

Problema 4.1:[minmaxArcs] Si un graf dirigit té n vèrtexs, quin és el nombre mínim i màxim d'arestes que pot tenir? I en el cas particular que siga connex? Raona les respostes.

El nombre mínim d'arestes o arcs de qualsevol graf és clarament zero i es correspon amb el graf buit d'ordre n, $\emptyset_V$.

Com que es tracta d'un graf dirigit (sense bucles, ni arcs paral·lels) tot arc ha d'unir dos dels n nodes. El nombre màxim d'arestes es correspondrà, doncs, al nombre de possibles parells (ordenats) sense repeticions, que és

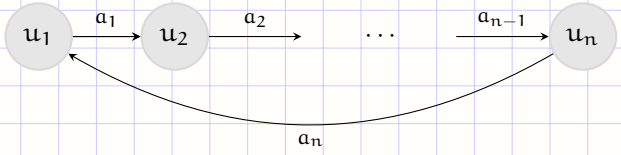
$$|V_n^2| = n^{\underline{2}} = n(n-1).$$

També es pot raonar que cada un dels n nodes pot estar unit amb tots els $n-1$ restants. I la quantitat anterior s'obté en aplicar el principi del producte.

En el cas particular que el graf siga connex el nombre màxim **no canvia**, ja que es correspon amb un graf on tots els nodes estan connectats amb tots (o graf complet) que serà, doncs, connex.

Quant al nombre mínim en grafs connexos, cal tenir en compte que hi ha d'haver camins entre els $n(n-1)$ parells de nodes, tot i que aquests camins compartisquen arcs.

Una manera de connectar tots els parells és la següent: Podem recórrer els n nodes (en un ordre donat) mitjançant $n-1$ arcs dirigits i afegir un altre arc per a tornar a l'origen. És a dir, podem construir un cicle hamiltonià i eulerià de n arcs que contindrà de fet camins entre tots els parells (ordenats) de nodes.



De l'anterior es dedueix que el nombre mínim d'arcs en un graf dirigit connex de n nodes ha de ser menor o igual que n. Però pot ser inferior? Ho demostrarem per reducció a l'absurd.

Suposem que es pogueren connectar n nodes amb $k < n$ arcs. En aquest cas, com que la suma total de graus és $2k$, es pot assegurar que hi ha d'haver nodes amb

grau menor que 2. (Si tots els nodes tingueren grau major o igual que 2 la seua suma seria major o igual que $2n$).

Això implica que a aquests nodes, o bé no es pot arribar, o bé no se'n pot eixir, i per això hi ha d'haver parells de nodes sense connectar i el graf no pot ser connex.

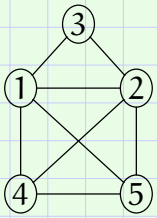
Com a conseqüència, podem estar segurs que el nombre mínim d'arcs en un graf dirigit connex és n

Problema 4.2:[caseta] Considera el graf $G = (V, A)$, de la figura.

a) Es pot trobar un recorregut eulerià en aquest graf? Per què? Escriu-ne un si és possible.

b) Es pot trobar un circuit que siga eulerià i hamiltonià al mateix temps? Per què? I en algun subgraf de G de 5 nodes? Digues en quants i en quins.

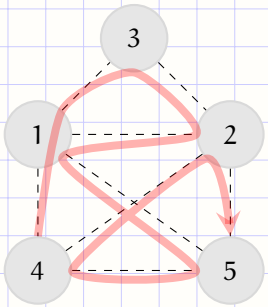
c) Hi ha algun arbre generador de G que es puga veure com a arbre binari? Dibuixa'n alguns o digues per què no n'hi ha.



a) Sí que es pot trobar un recorregut eulerià a causa del teorema d'Euler (pàgina 122). O més concretament del seu corol·lari que afirma que si hi ha exactament dos nodes amb grau imparell, ha d'haver-hi un recorregut eulerià (no circular), que començarà i acabarà en aquests dos nodes.

Un d'aquests recorreguts eulerians seria

$(4, 1, 3, 2, 1, 5, 4, 2, 5)$



b) Segons també el teorema d'Euler només hi pot haver un circuit eulerià si tots els nodes tenen grau parell. Per tant, menys encara pot haver-n'hi un que siga eulerià i hamiltonià.

La mateixa pregunta sobre subgrafs d'ordre 5 requereix primer calcular **tots** els subgrafs connexos de 5 nodes.

Afortunadament, dels 256 subgrafs d'ordre 5 (ja que hi ha 8 arcs i el nombre total de subconjunts serà $|\mathcal{VR}_2^8| = 2^8$), només cal tenir en compte els que són connexos.

Però això tampoc no és tan fàcil. En el seu lloc considerarem directament grafs connexos que tinguen exactament 5 arcs, que és la longitud (en arcs) que haurà de tenir qualsevol circuit eulerià i hamiltonià (i per tant, cicle).

En aquest cas tampoc cal considerar tots els subconjunts de 8 arcs de cardinalitat 5, ja que algunes configuracions donen lloc a grafs no connexos. Efectivament, si el cicle ha de passar pel node 3, això implica que els seus 2 arcs incidents hi han d'estar. I com a conseqüència, l'arc $(1,2)$ no cal considerar-lo perquè es formaria un cicle de 3 nodes (i el recorregut resultant no podria ser hamiltonià).

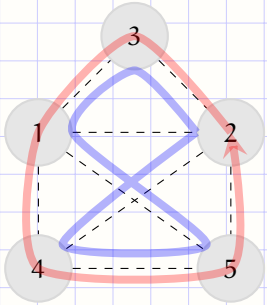
En altres paraules, qualsevol cicle hamiltonià haurà d'estar format d'una banda pels arcs $(1,3)$ i $(3,2)$. I de l'altra per un camí que vaja (indirectament) des de 1 a 2 o al revés. I les úniques possibilitats són:

$$(1,4,5,2) \qquad (1,5,4,2).$$

Per tant, els únics cicles hamiltonians que hi podria haver en algun subgraf (de 5 nodes) de G serien

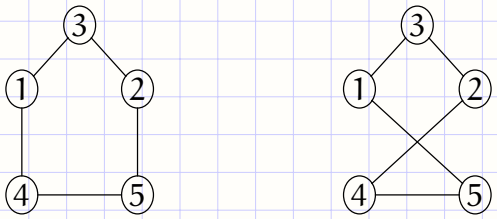
$(4,1,3,2,1,5,4,2,5)$

$(4,1,3,2,1,5,4,2,5)$



Com que ens demanen que els cicles siguen hamiltonians i **eulerians**, això vol dir que els subgrafs que els continguen no poden tenir cap arc més a banda dels que formen el cicle.

Per tant, els únics subgrafs de G d'ordre 5 que contenen un circuit (cicle) hamiltonià i eulerià són:

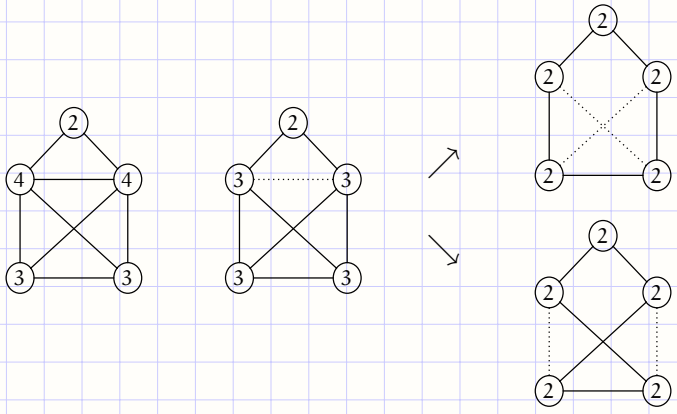


En realitat, hauríem pogut arribar a la mateixa conclusió més directament de la següent manera. Els nodes d'un subgraf que continga un circuit eulerià han de tenir grau parell. Però si a més a més ha de ser hamiltonià, el grau de **tots** els nodes ha de ser 2 (perquè només es pot entrar i eixir de cada node una vegada).

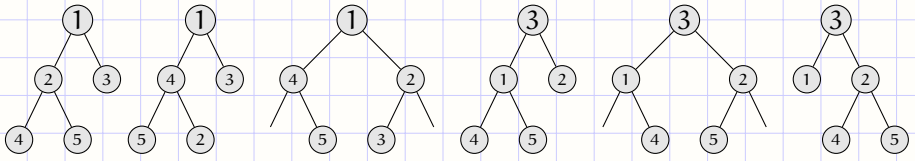
Per a obtenir subgrafs de G on tots els nodes tinguen grau 2, caldrà anar eliminant arcs considerant totes les possibilitats. Ja hem raonat que l'arc $(1,2)$ s'ha d'eliminar per a desfer el cicle no desitjat $(1,3,2,1)$.

A partir d'ací tenim els nodes $1,2,4$ i 5 amb grau igual a 3. Per a aconseguir que els quatre acaben amb grau igual a 2 només hi ha dues possibilitats. O eliminem $(1,5)$ i $(2,4)$, o eliminem $(1,4)$ i $(2,5)$. I això porta a les dues solucions d'abans.

Les seqüències de subgrafs que s'obtindrien amb indicació del grau de cada node seria:



c) Sí que n'hi ha arbres generadors que es poden veure com a binaris. I són molts.

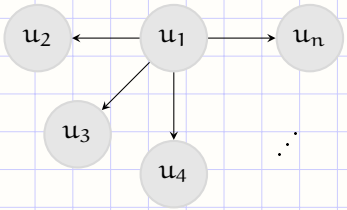


Problema 4.3:[grausSpan] És possible que en un arbre generador d'un graf de n nodes els graus de tots els nodes siguen parells? I que siguen tots imparells? Justifica les respostes i posa'n exemples si es pot.

Un arbre generador d'un graf de n nodes és un arbre dels mateixos n nodes. Per tant, ha de ser **connex** i **acíclic**.

Raonarem per **reducció a l'absurd**. Suposem que tots els nodes tenen grau parell. Aleshores, pel teorema d'Euler, hauria d'existir un circuit eulerià, cosa que implicaria que el graf (arbre) és **cíclic**. I això és impossible. Per tant, **no** poden ser tots els nodes de grau imparell.

La pregunta de si poden ser tots els nodes de grau imparell es pot reformular com a: Existeix un graf connex i acíclic de n nodes tots ells de grau imparell? I aquesta pregunta la podem contestar amb un exemple.

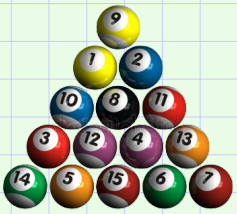


A partir de l'exemple podem dir que **sí** que poden ser tots els nodes de grau imparell, però sempre que n siga **parell** (i major que zero, clar). Això és perquè en l'exemple el grau del node u_1 és exactament $n-1$.

I què passa si n és imparell? En aquest cas és impossible que tots els nodes tinguen grau imparell perquè aleshores la suma dels graus de tots els nodes seria un nombre imparell. I sabem que aquesta suma ha de ser igual a dues vegades el nombre d'arcs ($2n-2$ en el nostre cas), que és un nombre parell.

De fet, hauríem pogut simplement aplicar la proposició de la pàgina 118 i el seu corol·lari que diu que no pot haver-hi un nombre imparell de nodes amb grau imparell.

Problema 4.4:[billar] Considera el graf $G = (V, A)$, on els nodes són les boles (numerades de la 1 a la 15) i els arcs es corresponen amb parells de boles que es toquen entre elles, com es mostra en la figura.



a) Defineix formalment el graf G.

b) Calcula el grau de cada node i la suma dels graus de tot el graf.

c) Dona, si existeix, un recorregut eulerià. Justifica la resposta.

d) Dona, si existeix, un camí **i/o** cicle hamiltonià. Justifica la resposta.

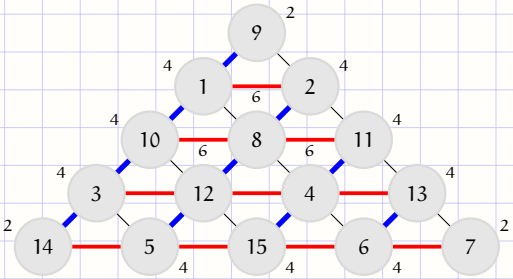
e) Què podem deduir del teorema d'Euler aplicat a G?

f) Siguen els subgrafs $G_i = (V_i, A_i)$, $i = 1, 2$, de manera que V_1 són les boles de l'1 a la 7 i V_2 les boles de la 9 a la 15. I els corresponents A_i estan formats per les arestes de A que relacionen els corresponents nodes. Descriu clarament els dos subgrafs i digues quantes i quines components connexes té cada un.

a) Es tracta d'un graf no dirigit, $G = (V, A)$, on el conjunt de nodes és $V = \{1, 2, \ldots, 15\}$, i el conjunt d'arcs el separarem en 3 subconjunts disjunts, $A = H \cup D \cup B$, per a distingir entre arcs que connecten boles en horitzontal (H), en diagonal (D) i en antidiagonal (B).

$$H = \{(1,2), (10,8), (8,11), (3,12), (12,4), (4,13), (14,5), (5,15), (15,6), (6,7)\}$$
$$D = \{(14,3), (3,10), (10,1), (1,9), (5,12), (12,8), (8,2), (15,4), (4,11), (6,13)\}$$
$$B = \{(3,5), (10,12), (12,15), (1,8), (8,4), (4,6), (9,2), (2,11), (11,13), (13,7)\}.$$

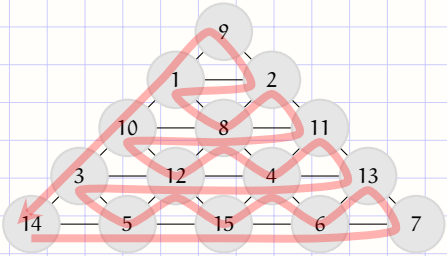
Gràficament,



b) El grau de cada node s'ha indicat en la figura anterior mitjançant nombres petits prop de cada node. La suma dels graus serà

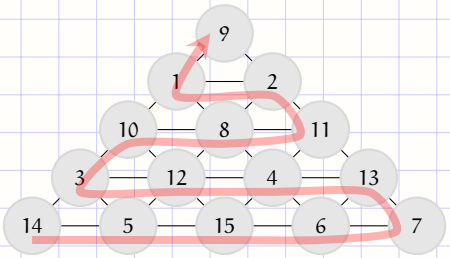
$$S = 3 \cdot 3 \cdot 4 + 3 \cdot 2 + 3 \cdot 6 = 3(12 + 2 + 6) = 60.$$

c) Segons el teorema d'Euler, com que tots els nodes tenen grau parell, necessàriament hi ha d'haver un circuit eulerià. Per tant, la resposta és que sí. Existeix un recorregut (circular) eulerià. Per a trobar-ne un cal procedir per inspecció. Se'n mostra un en la figura següent.

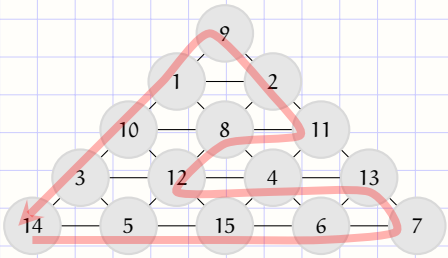


c) Per a veure si existeix o no algun recorregut hamiltonià només podem procedir per inspecció. És relativament fàcil trobar un recorregut hamiltonià si recorrem el graf per files (en sentits alternats) començant pel node 14 i acabant en el 9. També podem recórrer aquest mateix camí de manera simètrica (començant pel 7), i cada un dels dos de manera inversa (començant pel 9). I també podríem intercanviar el costat horitzontal del triangle per qualsevol dels altres dos en tots els recorreguts anteriors.

Trobar un cicle hamiltonià és un poc més entretingut. En la figura següent es mostren un camí i un cicle hamiltonians sobre el graf.



Camí hamiltonià:
(14, 5, 15, 6, 7, 13, 4, 12, 3, 10, 8, 11, 2, 1, 9)



Ciclo hamiltonià:
(14, 5, 15, 6, 7, 13, 4, 12, 8, 11, 2, 9, 1, 10, 3, 14)

Problema 4.5:[maxFulles] Demostra per inducció que el nombre màxim de fulles en un arbre binari de n nodes és com a molt $\lceil \frac{n}{2} \rceil$. Quin seria el mínim?

BI Un arbre binari buit té zero fulles i un arbre d'un node en té una. Es compleix, doncs, que

$$f_0 \leq \left\lceil \frac{0}{2} \right\rceil = 0,$$
$$f_1 \leq \left\lceil \frac{1}{2} \right\rceil = 1,$$

on f_n és el nombre màxim de fulles d'un arbre binari de n nodes.

HI

$$\left[f_k \leq \left\lceil \frac{k}{2} \right\rceil, \ \forall k < n \right].$$

PI

Qualsevol arbre binari de n nodes ($n > 0$), A_n, és format per dos subarbres de talles estrictament menors que n de manera que entre tots dos sumen $n-1$ nodes.

Siga $f(A)$ el nombre de fulles de qualsevol arbre A. Aleshores,

$$f_n = \max_{\forall A_n} f(A_n) = \max_{\forall A_n} \left(f(esq(A_n)) + f(dre(A_n)) \right) \leq$$
$$\leq \max_{0 \leq \ell \leq n-1} \left(f_\ell + f_{n-\ell-1} \right) \overset{(HI)}{\leq} \max_{0 \leq \ell \leq n-1} \left(\left\lceil \frac{\ell}{2} \right\rceil + \left\lceil \frac{n-\ell-1}{2} \right\rceil \right).$$

En el cas que ℓ siga parell tenim que

$$\left\lceil \frac{\ell}{2} \right\rceil + \left\lceil \frac{n-\ell-1}{2} \right\rceil = \frac{\ell}{2} + \left\lceil \frac{n-1}{2} \right\rceil - \frac{\ell}{2} \leq \left\lceil \frac{n}{2} \right\rceil .$$

I si és senar tenim també que

$$\left\lceil \frac{\ell}{2} \right\rceil + \left\lceil \frac{n-\ell-1}{2} \right\rceil = \left\lceil \frac{\ell}{2} \right\rceil + \left\lceil \frac{n}{2} \right\rceil - \frac{\ell+1}{2} = \left\lceil \frac{n}{2} \right\rceil .$$

O siga, que tenim una fita superior per a f_n independent de ℓ, i per això podem assegurar que

$$f_n \leq \left\lceil \frac{n}{2} \right\rceil .$$

Una vegada completat el pas d'inducció on s'ha considerat qualsevol arbre binari d'un o més nodes, podem concloure que l'únic cas base estrictament necessari en aquesta demostració és $n = 0$.

El nombre mínim de fulles en un arbre binari de n nodes és 1, que es correspon amb un arbre binari degenerat on cap dels nodes té dos fills.

Problema 4.6:[arbresCoixos] Un arbre binari és o està **coix** si és format per una arrel d'on pengen dos arbres coixos de manera que la profunditat de l'esquerre és la meitat de la del dret.

a) Converteix l'anterior descripció informal en una definició recursiva formal utilitzant lògica i la notació que cregues oportuna.

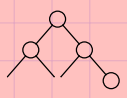
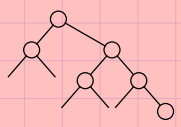
b) Posa exemples d'arbres coixos d'almenys 3 profunditats diferents.

Primer contestarem a) de manera informal (però precisa), donant els exemples que es demanen en b). I després donarem les definicions formals que es demanen en a).

b) Ens ajudarà a decidir primer quin és l'arbre més menut que pot ser coix. L'opció més lògica és considerar l'**arbre buit**, $\emptyset$, com a coix, perquè aleshores un arbre d'un sol node (del qual pengen 2 subarbres buits, ja que són arbres binaris) serà evidentment coix, ja que 0 és la meitat de 0. Amb això ja tindríem 2 dels exemples que demana b):

$A_0 \equiv \emptyset$ $A_1 \equiv$ 

Per a poder continuar cal decidir si interpretem **meitat** com a divisió entera, $\lfloor \frac{p}{2} \rfloor$, o com a divisió arredonida per dalt, $\lceil \frac{p}{2} \rceil$. En el primer cas, els següents arbres coixos que podríem formar serien

$A'_2 \equiv$  $A'_3 \equiv$  $A'_4 \equiv$  ...

que complirien la condició de ser coixos, ja que les profunditats dels subarbres de les arrels compleixen que

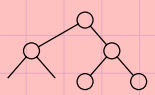
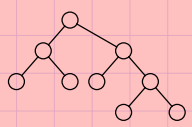
$$0 = \lfloor \tfrac{1}{2} \rfloor, \qquad 1 = \lfloor \tfrac{2}{2} \rfloor, \qquad 1 = \lfloor \tfrac{3}{2} \rfloor,$$

i tots els subarbres implicats són coixos.

La definició recursiva en aquest primer cas només requereix un cas base i es podria escriure com

$$\begin{cases} A'_0 \equiv \emptyset \\ A'_{n+1} \equiv \quad \bullet \quad \text{si } n \geq 1 \\ \qquad\qquad A'_{\lfloor \frac{n}{2} \rfloor} \quad A'_n \end{cases}$$

En canvi, en el segon cas en què la meitat s'arredoneix per dalt els següents arbres serien

$A''_2 \equiv$  $A''_3 \equiv$  $A''_4 \equiv$  ...

i en les seues arrels es compliria ara

$$1 = \lceil \tfrac{1}{2} \rceil, \qquad 1 = \lceil \tfrac{2}{2} \rceil, \qquad 2 = \lceil \tfrac{3}{2} \rceil.$$

En el segon cas la definició recursiva seria aleshores

$$\begin{cases} A''_0 \equiv \emptyset \\ A''_{n+1} \equiv \quad \overset{A''_{\lceil \frac{n}{2} \rceil}}{\bullet} \quad \overset{A''_n}{\bullet} \quad \text{si } n \geq 1 \end{cases}$$

Tant en un cas com en l'altre hi ha un únic arbre coix per cada enter no negatiu: $A_0, A_1, A_2, \ldots$. I resulta que el subíndex es correspon amb la profunditat de cada arbre.

Què passaria si començàrem amb A_1 com a cas base?

En el segon cas, res. Només caldria explicitar com a (únic) cas base

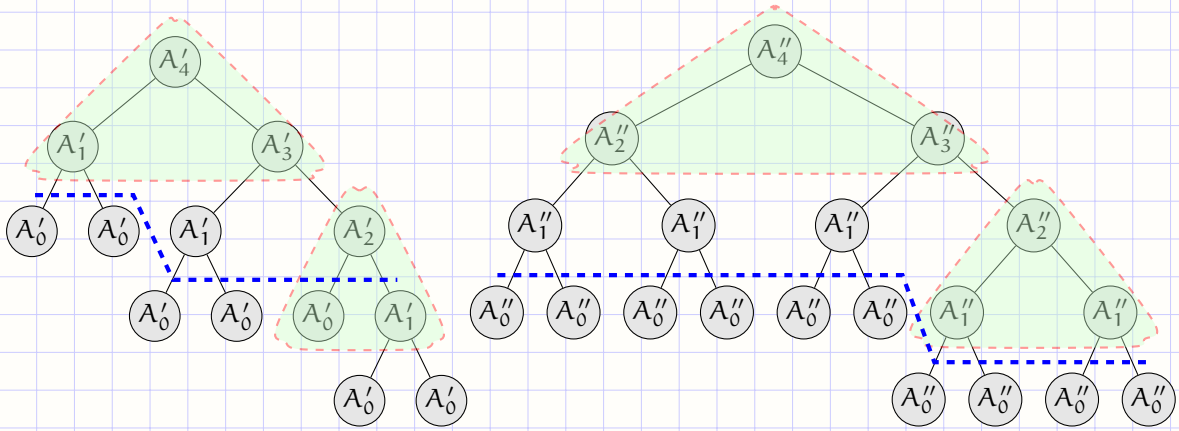
$$A''_1 \equiv \bullet$$

i la seqüència d'arbres seria ara $A''_1, A''_2, \ldots$

En canvi, en el primer cas necessitaríem definir **tant A'_1 com A'_2 com a casos base**. Això és perquè en el primer cas A'_2 depèn de A'_0 que ara no seria coix.

Aleshores tenim 2 opcions per a definir el cas base. I altres 2 per al cas recursiu. En total (variacions amb repetició de 2 elements agafats de 2 en 2), 4 definicions possibles lleugerament diferents.

Podem visualitzar gràficament les diferències entre les diferents definicions recursives si dibuixem els corresponents arbres de recursió per al cas d'arbres coixos de 4 nodes.



En la figura es mostren els dos arbres de recursió segons cada una de les definicions recursives en funció de com s'interprete la meitat d'un enter. S'han marcat les parts dels arbres que són diferents.

També s'ha indicat per on s'hauria de retallar cada arbre en el cas que no es considerara el cas $n = 0$. En el cas de la dreta totes les fulles es correspondrien amb l'arbre A''_1. Però en el de l'esquerra, tenim fulles amb A_1 i amb A_2, els dos possibles casos base.

Considerarem a partir d'ací només la primera definició que té en compte l'arbre buit ($n = 0$) i arredoneix per baix. És a dir, arbres de recursió com el de l'esquerra de la figura anterior al complet.

La definició formal podria constar de dues afirmacions (cas base i cas recursiu):

C.B. $A_0 \equiv \emptyset$ és coix,

C.R. $\forall A$, A és coix **sii** $\begin{cases} A \text{ és de la forma } \overset{A_e}{\circ}\!\!\diagup\!\!\diagdown\!\!\overset{A_d}{\circ}, \\ A_e, A_d \text{ són coixos}, \\ p(A_e) = \lfloor \frac{p(A_d)}{2} \rfloor, \end{cases}$

on $p : \mathcal{A} \to \mathbb{Z}^+$ és una funció que ens dona la profunditat de qualsevol arbre, $A \in \mathcal{A}$.

Observem que no es diu explícitament en cap lloc que A siga binari. Però és que l'anterior definició sense la condició de les profunditats **és en realitat una definició recursiva d'arbre binari!**

Per això, es pot donar també una definició alternativa sabent que tot arbre binari (llevat del buit) ha de tenir un subarbre esquerre i un subarbre dret:

C.B. $A_0 \equiv \emptyset$ és coix,

C.R. $\forall A \in \mathcal{A}_B \backslash \{\emptyset\}$, A és coix **sii** $\begin{cases} e(A), d(A) \text{ són coixos,} \\ p(e(A)) = \lfloor \frac{p(d(A))}{2} \rfloor, \end{cases}$

on $\mathcal{A}_B$ és el conjunt de tots els arbres binaris i

$$e, d : \mathcal{A}_B \to \mathcal{A}_B$$

són funcions que ens donen cada subarbre de qualsevol arbre binari no buit, $A \in \mathcal{A}_B \backslash \{\emptyset\}$.

Si volguérem utilitzar explícitament predicats, podríem escriure:

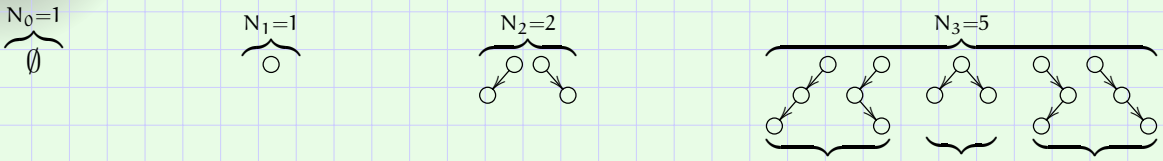
$$\begin{aligned} &\mathrm{coix}(\emptyset) \ \wedge \\ &\forall A \neq \emptyset, \ \ \mathrm{coix}(A) \ \ \textbf{sii} \ \Big[\mathrm{binari}(A) \ \wedge \\ &\qquad \Big(\exists E, D, P_d, \ \mathrm{esq}(A, E) \wedge \mathrm{dre}(A, E) \wedge \mathrm{pr}(D, P_d) \wedge \mathrm{pr}(E, \lfloor \frac{P_d}{2} \rfloor) \Big) \Big], \end{aligned}$$

on ara esq, dre i pr són predicats equivalents a les funcions anteriors. I A, E, D, P_d són variables que representaran arbres i enters, respectivament.

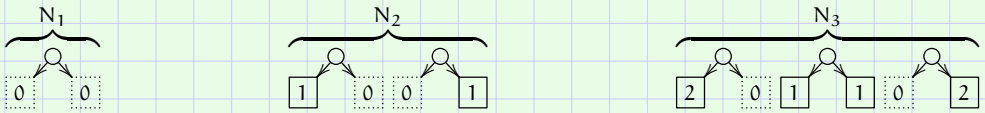
Alternativament, també podem expressar la mateixa definició amb molt poca notació però igual de formalment de la següent manera:

- L'arbre buit és coix.
- Tot arbre no buit és coix **sii**
  - és binari,
  - la profunditat del seu subarbre esquerre és $\lfloor \frac{p}{2} \rfloor$, on p és la profunditat del seu subarbre dret,
  - I a més a més tots dos subarbres són coixos.

Problema 4.7:[arbresDiferents] Siga N_n el nombre d'arbres binaris diferents de $n \geq 0$ nodes. Tenim per a $n \leq 3$ que

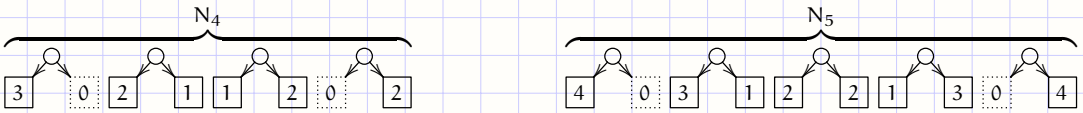


I podem establir les següents relacions on $\boxed{n}$ representa tots els arbres de n nodes.



a) Descriu i explica el raonament recursiu anterior i arriba a una definició recursiva per a N_n, b) Calcula els valors N_4 i N_5.

Només observant la sèrie hauria de ser fàcil arribar a:



i fins i tot arribar a calcular els corresponents valors com a

$$N_4 = 5 + 2 + 2 + 5 = 14,$$
$$N_5 = 14 + 5 + 2 \cdot 2 + 5 + 14 = 42.$$

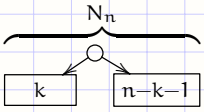
Però és molt més important descobrir per què:

> Tot arbre binari de $n \geq 1$ nodes (llevat del cas $n = 0$) és necessàriament format per **una** arrel i **dos** subarbres que, entre tots dos, han de contenir els restants $n - 1$ nodes.

I de quantes maneres poden els 2 subarbres (que són **distingibles** en un arbre binari) repartir-se els $n - 1$ nodes? Això són parells ordenats d'enters diferents menors que n (incloent-hi el zero). En total, n.

$$(n-1, 0), (n-2, 1), (n-3, 2), \ldots, (1, n-2), (0, n-1).$$

O, en general



$$k = 0, 1, \ldots, n-1.$$

I quants arbres corresponen a cada parell d'enters?

Si el subarbre esquerre té k nodes, hi haurà N_k opcions per a completar-lo. I altres N_{n-k-1} opcions per al dret. Pel **principi del producte**, per a cada valor de k tindrem aleshores $N_k \times N_{n-k-1}$ opcions. I el total serà la suma per a tot valor de k (pel **principi de la suma**),

$$N_n = \sum_{k=0}^{n-1} N_k N_{n-k-1},$$

que juntament amb el cas base, $N_0 = 1$, ens duu a la definició recursiva,

$$\begin{cases} N_0 = 0, \\ N_n = \sum_{k=0}^{n-1} N_k N_{n-k-1}, \ \forall\, n \geq 1. \end{cases}$$

En aquest cas, la definició recursiva la podem expressar també simplement com a

$$N_n = \sum_{k=0}^{n-1} N_k N_{n-k-1}, \ \forall\, n \geq 0,$$

ja que el cas base ($n = 0$) es correspondria amb un sumatori buit que és igual a zero.

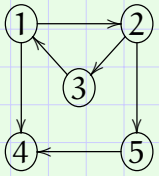
Els successius valors de la funció es poden calcular com a:

$$
\begin{aligned}
N_0 &= 0,\\
N_1 &= \cancel{N_0}^{1} \cdot \cancel{N_0}^{1} = 1,\\
N_2 &= \cancel{N_1}^{1} \cdot \cancel{N_0}^{1} + \cancel{N_0}^{1} \cdot \cancel{N_1}^{1} = 2,\\
N_3 &= \cancel{N_2}^{2} \cdot 1 + 1 \cdot 1 + 1 \cdot \cancel{N_2}^{2} = 5,\\
N_4 &= \cancel{N_3}^{5} \cdot 1 + \cancel{N_2}^{2} \cdot 1 + 1 \cdot \cancel{N_2}^{2} + 1 \cdot \cancel{N_3}^{5} = 14,\\
N_5 &= \cancel{N_4}^{14} \cdot 1 + \cancel{N_3}^{5} \cdot 1 + 2 \cdot 2 + 1 \cdot \cancel{N_3}^{5} + 1 \cdot \cancel{N_4}^{14} = 42.
\end{aligned}
$$

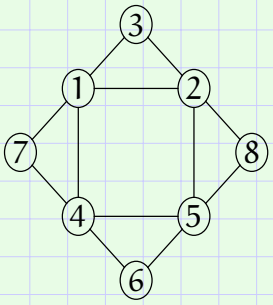
Els valors d'aquesta successió es coneixen en la literatura com els **nombres de Catalan**.

4.5 Problemes proposats

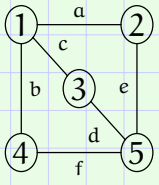
Problema 4.8:[grafSobre] Considera el graf de la figura i calcula: a) els graus de tots els nodes, b) la matriu d'adjacència, c) un recorregut que passe pel màxim d'arcs sense repetir-ne cap, d) un recorregut que passe pel màxim de nodes sense repetir-ne cap, e) un arbre d'extensió que siga binari i un altre que siga ternari, i f) digues quants cicles diferents conté el graf i quins són.



Problema 4.9:[grafRomb] Siga G el graf de la figura. a) Digues quants cliques (grafs totalment connexos) diferents conté G. b) Calcula un recorregut hamiltonià en G (o explica per què no n'hi ha). c) Calcula un recorregut eulerià en G (o explica per què no n'hi ha). d) Digues quants cicles té el recorregut anterior. e) Calcula un recorregut eulerià sense cicles en G (o explica per què no n'hi ha).



Problema 4.10:[grafZ] Considera el graf de la figura i respon a les qüestions: a) La matriu d'adjacència. b) Un recorregut eulerià, si existeix; i si no, digues per què. c) Un camí eulerià, si existeix; i si no, digues per què. d) Hi ha algun subgraf que continga algun cicle eulerià? Quants? Quins?



Problema 4.11:[components]
Considera la proposició de la pàgina 124 que relaciona els nombres de nodes, arcs i components connexes d'un graf i demostra-la per inducció **sobre el nombre de nodes**.

Problema 4.12:[sumaGrausInd] Demostra per inducció que la suma dels graus d'un arbre ternari és un nombre parell. Enuncia formalment el que es vol demostrar (usant lògica de predicats), justifica el tipus d'inducció i raona clarament els diferents passos.

Problema 4.13:[arbresEquilibrats] Demostra per inducció que en un arbre binari equilibrat de n nodes, la profunditat, h, és logarítmica i com a molt $2 \lg n$. Equivalentment,

$$h \leq 2 \lg n \quad \equiv \quad n \geq 2^{\frac{h}{2}}.$$

A. El llenguatge PROLOG--

A.1 Definicions

Per tal d'il·lustrar conceptes relacionats amb la lògica i la recursió considerarem un llenguatge de programació que és un subconjunt (estricte) del llenguatge de programació Prolog. Per aquest motiu anomenarem aquest llenguatge PROLOG--.

És important subratllar que no es tracta d'un llenguatge de programació complet ja que només es pretén utilitzar la part purament lògica de Prolog. Per això exclourem explícitament aquells aspectes no lògics de Prolog com ara l'entrada/sortida i sobretot l'operador **tall** (*cut*).

La implementació que s'ha fet servir per als exemples i problemes i que es recomana és SWI-Prolog[†]. En particular s'ha usat la versió 7.6.4. No obstant això, tots els exemples de PROLOG-- haurien de funcionar perfectament amb qualsevol altra implementació de Prolog.

A.1.1 Elements

Els components d'un programa en PROLOG-- són bàsicament els mateixos que s'han definit en la lògica de predicats.

D'una banda tenim les **constants** que poden ser de tipus **numèric** (enter o real) o **atòmic**. Els nombres enters o reals s'escriuen usant les notacions habituals.

```
1024          3.14          2.4e-3
```

Els **àtoms** representen elements de qualsevol domini i són cadenes de caràcters alfanumèrics (incloent-hi el símbol subratllat, "_") que comencen amb una lletra minúscula.

[†] Aquesta implementació és de llicència pública i es pot trobar en https://www.swi-prolog.org

```
a                aaaa              aB_c123_Ab
```

D'altra banda, les **variables** en PROLOG-- s'escriuran com a cadenes de caràcters alfanumèrics (incloent "_") que comencen amb una lletra majúscula.

```
A                X1                Ab_102_X
```

Existeix una variable especial que és la **variable anònima**, "_", que representarà una variable diferent cada vegada que aparega en una expressió.

Els àtoms, nombres i variables són els **termes** bàsics o simples en PROLOG--. Però a banda hi ha dues maneres d'agrupar termes per a formar-ne de nous. D'una banda els **functors** ja introduïts en definir les fórmules ben formades en lògica de predicats (pàg. 55), i d'altra les **llistes**, també introduïdes en la secció 3.1.1 (pàg. 86).

Els functors tenen un nom com el dels àtoms i un nombre variable d'arguments que han de ser necessàriament **termes**. També és possible agrupar termes formant tuples la qual cosa es pot entendre com un functor sense nom.

```
a(b,X)           f(a,X,_)          f(a,g(X,b))       (a,g(X,b),c)
```

Una **llista** és una seqüència de zero o més termes separats per comes i delimitat per claudàtors.

```
[b,X]            [a,X,_]           [a,[X,b],[]]
```

Una variable o un functor o llista que conté variables és un terme **genèric**, que pot representar diversos elements d'un determinat domini. En un moment donat una variable es pot instanciar a qualsevol altre terme, fins i tot a una altra variable.

A banda dels termes, l'altre component bàsic dels programes en PROLOG-- són els **predicats**. Un **predicat** té associat un nom (cadena de caràcters alfanumèrics que comença amb minúscula) i una **aritat** o nombre d'arguments associats. Ho escriurem resumidament com nom/#, on # és l'aritat del predicat nom. En el següent exemple es mostren els predicats aa/3, aa/2 i aa/1.

```
aa(a,X,_)        aa(a,g(X,b))      aa(a)
```

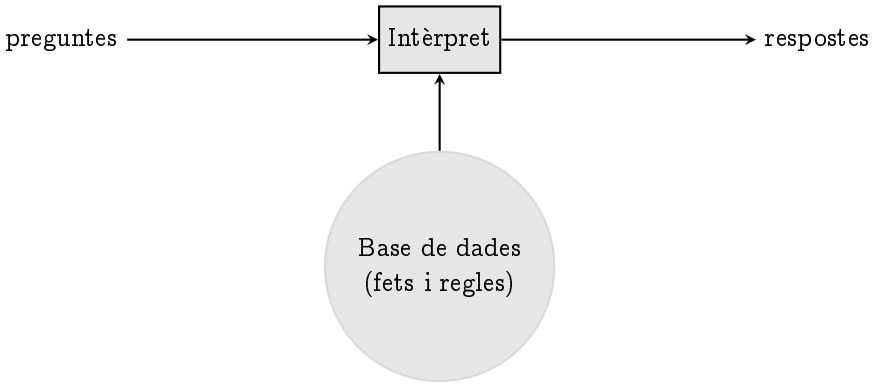
En lògica de primer ordre, els arguments d'un predicat han de ser necessàriament termes per la qual cosa no pot haver cap confusió en expressions com

```
aa(aa,aa(_),aa(_,_))
```

on el mateix identificador, aa, és alhora predicat amb aritat 3 , àtom, functor amb aritat 1 i functor amb aritat 2. De la mateixa manera, predicats amb idèntic nom i aritat diferent poden coexistir sense confusió.

A.1.2 Estructura

Un programa en PROLOG-- consisteix en la definició de zero o més predicats juntament amb una o més **qüestions** o **preguntes**. El resultat del programa és format aleshores per les respostes a les qüestions, donats els predicats definits. Un esquema general es mostra en la següent figura.



Cada predicat es defineix mitjançant una o més clàusules (de Horn, pàg. 44) que poden ser de dos tipus, **fets** o **regles**, i totes dues han d'acabar amb el símbol "." (punt).

Un **fet** és una afirmació sobre la veracitat d'alguna cosa i que pot contenir o no variables.

En PROLOG-- els fets només poden ser **fórmules atòmiques**. És a dir, han d'estar formats per un únic predicat (de qualsevol aritat) juntament amb tots els seus arguments i sense cap connectiva lògica. Això vol dir que en PROLOG-- no podem afirmar coses com

$$P \vee Q, \quad \neg(P \wedge Q)$$

Si que és possible en canvi afirmar la conjunció de diverses coses simplement en afirmar separadament cada una d'elles.

Una **regla** en PROLOG-- és una implicació l'antecedent de la qual és una expressió formada a partir de predicats usant les connectives "," (conjunció) i ";" (disjunció). El conseqüent en canvi ha de ser una fórmula atòmica.

En les implicacions s'escriu primer el conseqüent seguit dels símbols ":-" que es llegeixen "si" i a continuació l'antecedent, normalment en línies diferents i indentat respecte del conseqüent. A continuació es mostren exemples tant de fets com de regles en PROLOG--.

```
1.  plou.

2.  color(roig).
3.  color(gris).

4.  trist :-      color(roig).
5.  trist :-      plou.

6.  trist :-      color(roig); plou.
```

En la línia 1 es mostra la definició del predicat `plou/0`. Les línies 2 i 3 il·lustren la definició de `color/1`. En els 3 casos es té una fórmula atòmica. En canvi, les línies 4 i 5 constitueixen una definició de `trist/0` mitjançant regles. En realitat, l'enumeració de regles (o fets) és equivalent a la seua conjunció per la qual cosa (i en aplicar la pseudodistributivitat per l'esquerra de la implicació) les línies 4 i 5 són equivalents a la regla de la línia 6 on l'antecedent és format per una disjunció.

En PROLOG-- sempre evitarem l'ús de la disjunció fins on siga possible i sempre serà preferible definir predicats usant (la conjunció de) diverses regles en lloc d'utilitzar disjuncions en l'antecedent.

En PROLOG-- es pot escriure la negació d'un predicat o expressió amb predicats mitjançant el pseudopredicat `not/1` de manera que l'expressió `not(expressioLogica)` és certa si la veritat de `expressioLogica` no es pot demostrar. Cal tindre la precaució d'envoltar les expressions que continguen connectives amb parèntesis per tal que el compilador interprete correctament l'únic argument de `not/1`.

En PROLOG-- també evitarem l'ús explícit de la negació ja que aquesta no es correspon exactament amb la negació lògica.

Una **qüestió** o **pregunta** en PROLOG-- és qualsevol expressió formada amb predicats, parèntesis, conjuncions i disjuncions.

Si l'expressió és **tancada**, la resposta serà vertader o fals. Si l'expressió conté variables (fórmula **oberta**) la resposta, que podrà ser única o no, vindrà donada pels valors de les variables que fan vertadera l'expressió.

En qualsevol cas, quan la contestació és fals, no significa necessàriament que l'expressió corresponent a la pregunta siga lògicament falsa, sinó que PROLOG-- no ha sigut capaç de demostrar la seua veracitat. Això és el que s'anomena **negació per fallada**.

Sintàcticament, les preguntes en PROLOG-- es poden afegir a les definicions dels predicats o es poden introduir de manera interactiva. Només en el primer cas s'escriuen precedides dels símbols ":-". És a dir, com una regla sense conseqüent.

A.1.3 Unificació i comparació de termes

La igualtat de dos termes qualssevol es pot comprovar usant l'operador "==". L'expressió `terme1 == terme2` constitueix un predicat i és cert sii els 2 termes són lògicament idèntics.

La **unificació** de dos termes que en general contenen variables s'escriu com a `terme1 = terme2`. Aquesta expressió és certa sii existeixen valors per a les variables dels termes que facen que els dos termes siguen idèntics.

L'anomenat algorisme d'unificació és el que s'encarrega de comprovar-la i també de calcular els conjunts de valors per a les variables. De la mateixa manera que en el cas de la comparació de termes, la expressió `terme1 = terme2` constitueix un predicat.

Existeixen també versións negades tant per a la comparació de termes com per a la unificació que s'escriuen com a "\==" i "\=", respectivament.

A continuació es mostren alguns exemples de unificació i comparació de termes.

```
?- f(a,b)==f(a,b).
true.

?- f(X,b)==f(a,b).
false.

?- f(X,b)=f(a,b).
X = a.

?- f(X,Y,Z)=f(Y,X,X).
X = Y, Y = Z.

?- f(X,b,Z)=f(a,b,1);f(a,Y,Z)=f(a,b,2).
X = a, Z = 1;
Y = b, Z = 2.
```

Si hi ha més d'un conjunt de valors per a les variables, l'intèrpret donarà diverses respostes que caldrà obtindre en introduir el caràcter ";" o també polsant la barra d'espai del teclat (segons implementacions). Aquest caràcter es mostra en pantalla o no també segons implementacions.

En l'últim exemple del quadre anterior, la primera resposta diu que la variable `X` ha de valdre (o s'ha d'instanciar a) a i la variable `Z` a 1, i que la variable `Y` pot valdre qualsevol cosa (o pot no estar instanciada a cap valor) ja que no es diu res d'ella. En la segona resposta la variable `Z` ha d'estar instanciada a 2 i les variables `X` i `Y` han intercanviat els seus papers.

A.1.4 Resolució

En els exemples anteriors hem vist programes (interactius) sense cap predicat definit, on les preguntes es donen com a entrada a l'intèrpret (identificat pel *prompt* "?-"). No obstant això, la situació més normal és que el programa continga diversos predicats la definició dels quals es trobe en un arxiu amb extensió ".pl", com per exemple

```
    % Arxiu ``prova.pl''
1.  cotxe(bmv).
2.  moto(ducati).
3.  vehicle(reliant).
4.  vehicle(X) :-
5.          cotxe(X).
6.  vehicle(X) :-
7.          moto(X).
```

Aquest arxiu es pot carregar en PROLOG-- equivalentment mitjançant qualsevol de les quatre preguntes següents.

```
?-  [prova].
    true.
?-  ['prova.pl'].
    true.
?-  consult(prova).
    true.
?-  consult('prova.pl').
    true.
```

La contestació és la mateixa encara que pot dependre de diferents implementacions. Si el nom de l'arxiu (sense extensió) no és (sintàcticament) un àtom, aleshores és necessari envoltar-lo amb cometes simples.

A banda del predicat `consult/1`, també es poden fer servir els predicats predefinits `ls/0`, `pwd/0` i `cd/1` per llistar, comprovar i canviar de directori.

En el cas que l'arxiu continga també preguntes, aquestes no han de contindre variables no lligades i el seu resultat ha de ser vertader. Altrament, l'intèrpret emitirà un avís (*warning*)

Al conjunt de definicions format per fets i regles que es carrega en l'intèrpret se'l coneix com a base de dades. Una vegada carregats, fets i regles esdevenen axiomes del sistema i es poden fer preguntes com, per exemple

```
?-  cotxe(bmv).
    true.
?-  moto(bmv).
    false.
?-  vehicle(Y).
    Y=reliant ;
    Y=bmv ;
    Y=ducati.
?-  vehicle(X), not(moto(X);cotxe(X)).
    X=reliant
    false.
```

Quan les preguntes consisteixen en predicats, PROLOG-- recorre la base de dades de manera seqüencial i intenta unificar el predicat de la pregunta amb algun fet o amb el cap (conseqüent) d'alguna regla. Si es troba primer un fet, PROLOG-- respon cert, o amb els valors de les variables que fan cert el predicat, com en els exemples anteriors sobre unificació.

Si el que es troba és (el conseqüent o cap d') una regla, aleshores tot l'antecedent esdevé una nova pregunta i el procés continua recursivament fins que no queden possibilitats. Cada vegada que ocorre una unificació, algunes variables poden ser instanciades a diferents elements. Si hi ha més d'una possibilitat, l'intèrpret anirà provant-les totes si s'obté com a resultat fals o si s'introdueix la tecla ";" després d'una contestació positiva.

Per exemple, la pregunta "`vehicle(Y).`" donaria lloc a una primera unificació amb el fet de la línia 3 que faria que la variable s'instanciara al valor "`reliant`" i provocaria la primera resposta.

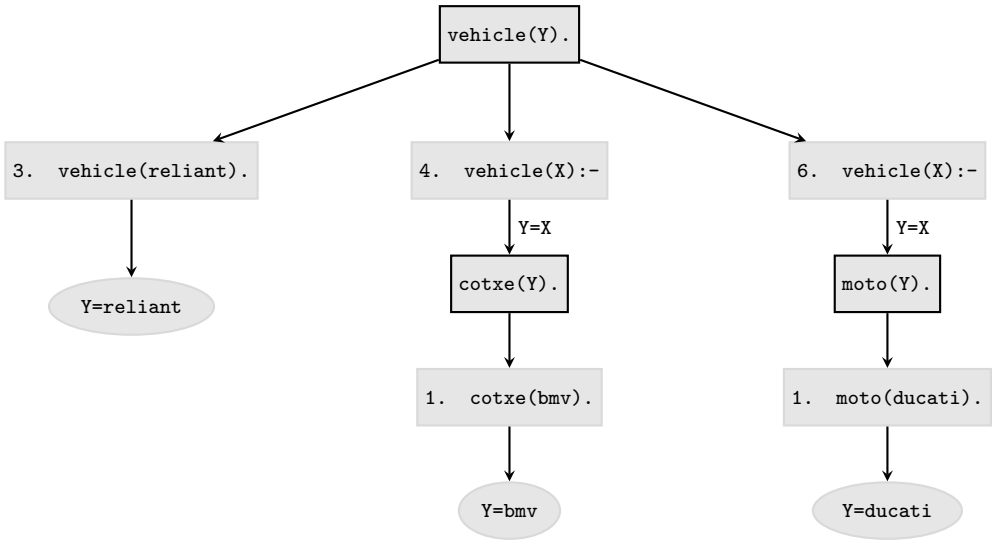
Si aleshores polsem ";" l'intèrpret torna enrere (desfà la instanciació de la variable) i troba una nova unificació en la línia 3 (la variable `Y` s'instanciaria a la variable `X`, que continuaria sent una variable lliure).

Això donaria lloc a la nova pregunta "`cotxe(X).`" que trobaria una única unificació en la línia 1 que donaria lloc al seu temps a la segona resposta.

Si tornem a polsar ";" l'intèrpret tornarà enrere i aquesta vegada trobarà una unificació en la línia 5 que de la mateixa manera que abans dispararà una nova pregunta que donarà lloc a una darrera unificació en la línia 2 i a la tercera i última resposta.

Aquest procés és el que es coneix com resolució que és en realitat un algorisme de retrocés el funcionament del qual es pot representar de manera arborescent.

Per exemple, el càlcul anterior es pot representar mitjançant el següent arbre on les preguntes es mostren en rectangles emmarcats amb línies i les respostes en caixes ovalades.



A.2 Predicats recursius en PROLOG--

La potència de PROLOG-- com a llenguatge de programació rau en la unificació, la resolució i en la recursió. Considerem el següent predicat i les contestacions de l'intèrpret a algunes preguntes.

```
1.  simple(a).
2.  simple(X-a) :-
3.     simple(X).
```

```
?-  simple(a).
    true.
?-  simple(b).
    false.
?-  simple(a-a-a).
    true.
?-  simple(Y).
    Y = a ;
    Y = a-a ;
    Y = a-a-a
    ...
```

Es tracta d'un predicat recursiu unari l'argument del qual són expressions formades amb l'àtom a i l'operador "-". Quan es fa la pregunta "`simple(a-a-a).`", l'intèrpret la unificarà amb la regla de la línia 2 la qual cosa porta a que `X=a-a` i a que es dispare la nova pregunta "`simple(a-a).`". A continuació es tornaria a repetir el mateix i

aleshores la nova pregunta seria "`simple(a).`", que ja no s'unificaria amb la regla sinó amb el fet de la línia 1.

En canvi si li fem a l'intèrpret la pregunta "`simple(Y).`", es trobaria la unificació en la línia 1 la qual cosa provocaria la primera resposta: `Y=a`. Si polsem ";" l'intèrpret trobarà a continuació la unificació amb la regla de la línia 2 que donarà lloc a que `Y=X-a` i a que es llance l'antecedent, "`simple(X).`", com a (nova) pregunta la resposta de la qual serà `X=a`. Combinant la unificació amb la resposta es té que `Y=a-a`, que seria la segona resposta. De la mateixa manera es produiria la tercera resposta, la quarta, etc.

A.2.1 Predicats recursius sobre enters

L'exemple anterior funcionaria exactament igual si canviàrem l'àtom a per l'enter 1 (o qualsevol altre). I també si canviem el símbol "-" per "+", per exemple. Aquestes expressions no signifiquen en principi res per a PROLOG--. Si volem utilitzar expressions aritmètiques i fer càlculs amb nombres cal **avaluar-los**.

El mecanisme bàsic per avaluar expressions en PROLOG-- és l'operador "`is`", que avalua l'expressió de la part dreta a un valor (enter o real) i després intenta **unificar** aquest valor amb la part esquerra.

```
?- 2 is 1+1.
   true.
?- X is 1+1.
   X = 2.
?- 2 is X+1.
   ERROR: Arguments are not sufficiently instantiated ...
?- X=1,2 is X+1.
   X = 1.
?- X=a,2 is X+1.
   ERROR: Arithmetic: 'a/0' is not a function
```

Com es dedueix dels exemples, la part dreta no pot contindre variables sense instanciar (a un valor numèric). Considerem el predicat `simple1/3` com una extensió de l'últim exemple.

```
1. simple1(a,1,a+a).
2. simple1(X-a,N,Y+a+a) :-
3.    simple1(X,N1,Y),
4.    N is N1+1.
```

```
?- simple1(a-a,N,R).
   N = 2,
   R = a+a+a+a.
?- simple1(Z,N,a+a+a+a+a+a).
   Z = a-a-a,
   N = 3.
?- simple1(Z,2,R).
   Z = a-a,
   R = a+a+a+a ;
   ERROR!!!!
```

L'esquema recursiu és el mateix que abans però ara després de cada "crida" recursiva s'incrementa en 1 el valor del segon argument. En la primera pregunta, el primer argument és una entrada i PROLOG-- calcula com a sortida els valors de N i R. En la segona pregunta els arguments segon i tercer intercanvien els seus papers.

En la tercera pregunta, només el segon argument té un valor. Com es pot veure, PROLOG-- contesta i roman a l'espera. I si polsem ";" es produeix un bucle infinit. Això és per que les preguntes que dispara la regla sempre poden unificar-se amb la mateixa regla, "X=X-a, N1=N,Y=Y+a+a.". Per tant, no s'instancia cap variable i el procés continua indefinidament. La primera resposta es produeix per que la primera vegada que es dispara, s'obté la unificació "X=a, N1=1, R=a+a." en la línia 1. Si en l'arxiu escriguérem primer la regla i després el fet obtindríem directament el bucle infinit.

És possible modificar el predicat per a que responga correctament a preguntes com l'anterior.

```
1. simple2(a,1,a+a).
2. simple2(X-a,N,Y+a+a) :-
3.    N>1.
4.    N1 is N-1.
5.    simple2(X,N1,Y),
```

```
?- simple2(Z,2,R).
   Z = a-a,
   R = a+a+a+a.
```

La diferència entre els dos predicats és subtil. La identificació entre les variables N i N1 és exactament la mateixa. Però abans es calculava N en funció de N1 després de la recursió, i ara es calcula N1 en funció de N abans de la recursió. D'aquesta manera,

ara és el segon argument el que controla la recursió la qual cosa obliga a que aquest sempre estiga instanciat (si no, es produirà sempre un error en la línia 4). L'altra diferència important és que ara cal afegir la condició de la línia 3 per a que el fet i la regla siguen casos mútuament exclusius.

És interessant comentar que aquesta necessitat d'instanciació no s'aplica al predicat `simple1/3` ja que una pregunta com "`simple(Z,N,R).`" és perfectament vàlida com ho era la pregunta "`simple(Y).`". Totes dues donen lloc a infinites respostes però no a un bucle infinit.

Hi ha molts exemples de predicats recursius on la recursió depén d'un argument enter. Sempre s'han d'afegir les condicions que calga per a assegurar que els diferents casos, recursius i no recursius siguen mútuament exclusius. També s'ha de parar atenció a fer els càlculs relacionats amb la variable de la que depén la recursió abans de qualsevol referència recursiva.

Tots els exemples del capítol 3 corresponen a aquest cas. Normalment, hi ha un argument enter que controla la recursió, i els altres arguments poden ser d'entrada (dades que es necessiten per poder fer els càlculs) o de sortida (resultats del procediment recursiu que o bé s'instanciaran a una variable o bé es compararan mitjançant unificació amb un valor donat.

A.2.2 Predicats recursius sobre llistes

Les llistes són la manera en què PROLOG-- pot representar informació estructurada. D'entrada les llistes tenen ja una definició recursiva:

1. La llista buida, `[]`, és una llista.
2. Qualsevol terme `t` seguit d'una llista `L` és una llista. Això s'escriu com a `[t|L]`.

Les notacions següents són equivalents per a tot terme t_k on $1 \leq k \leq n$.

$$[t_1, t_2, \ldots, t_k, \ldots, t_n] \qquad [t_1, t_2, \ldots, t_k | [t_{k+1}, \ldots, t_n]] \qquad [t_1 | [t_2 | [\ \cdots [t_n | [\,]\ ]]]]$$

A mode d'exemple s'inclouen a continuació i sense comentaris, versions dels predicats anteriors `simple/1` i `simple1/3` amb llistes.

```
1. lsimple([]).
2. lsimple([a|L]) :-
3.    lsimple(L).

4. lsimple1([],0,[]).
5. lsimple1([a|L],N,[a,a|R]) :-
6.    lsimple1(L,N1,R),
7.    N is N1+1.
```

És relativament fàcil plantejar recursions sobre llistes si el cas recursiu es correspon amb la llista d'entrada menys un nombre fix dels seus **primers** elements. En el cas d'un sol element l'esquema seria alguna cosa com

```
pred([],...).       % Cas Base (normalment)
pred([C|L],...) :-
   ...
   pred(L,...),
   ...
```

Però també podríem pensar en una relació recursiva entre la llista d'entrada i algun element de la llista que no fóra el cap. En aquestos casos es pot utilitzar el predicat predefinit `select/3` juntament amb alguna altra condició la qual cosa donaria lloc a un esquema com

```
pred([],...).       % Cas Base (normalment)
pred(L,...) :-
   select(X,L,R),
   ...
   pred(R,...),
   ...
```

El predicat `select(E,L,R)` és cert si `E` és un element de la llista `L` i `R` és el resultat d'eliminar `E` en la llista `L`. Qualsevol dels arguments de `select` pot ser d'entrada o de sortida.

Una altra possibilitat és que la relació recursiva no siga amb la llista d'entrada sense un element (o sense un nombre fix d'elements), sinó amb el resultat de dividir la llista en dues subllistes. En aquestos casos es pot utilitzar el predicat predefinit `append/3` juntament amb altres condicions la qual cosa donaria lloc a un esquema com

```
pred([],...).          % Cas Base (ara es pot complicar!)
pred(L,...) :-
   append(L1,L2,L),
   ...
   pred(L1,...),       % o L2 o les dues
   ...
```

El predicat `append(L1,L2,L12)` és cert si `L12` és la concatenació de les llistes `L1` i `L2`. Qualsevol dels arguments de append pot ser d'entrada o de sortida.

A banda de poder-se usar per a plantejar relacions recursives, molts problemes sobre llistes es poden resoldre de manera fàcil i compacta usant els predicats append i/o `select`.

A.3 Operadors, funcions i predicats predefinits

A.3.1 Operadors lògics i pseudo-predicats

Els operadors lògics es corresponen amb les connectives principals de la lògica proposicional i els seus arguments són per tant predicats. Anomenem pseudo-predicats aquells predicats que poden tindre algun argument que també és un predicat. Els operadors lògics són també pseudo-predicats ja que tot operador es pot utilitzar també com a predicat o com a functor. Aquesta forma que s'anomena canònica es pot visualitzar mitjançant el predicat `write_canonical/1` l'argument del qual és qualsevol expressió. El resultat d'aquestes expressions és sempre un predicat.

nom/aritat	**descripció**	**associatiu**
`:-/2`	definició de regles	no
`:-/1`	pregunta	no
`,/2`	conjunció lògica	si
`;/2`	disjunció lògica	si
`not/1`	negació	
`findall/3`	totes les respostes a una pregunta	

El pseudo-predicat `findall/3` permet recollir totes les respostes a una pregunta utilitzant una llista. En particular, `findall(X,P,L)` és cert si `X` és una variable, `P` és una pregunta (que depén de `X`) i `L` és una llista que conté tots els valors de `X` per als quals la resposta a `P` és certa.

A.3.2 Operadors de comparació i unificació

Aquests operadors ja s'han vist abans. Els seus arguments són termes i el resultat és un predicat.

nom/aritat	descripció	associatiu
=/2, \=/2	unificació de termes	no
==/2, \==/2	comparació de termes	no

A.3.3 Operadors de comparació i unificació aritmètica

L'operador "is/2" ja s'ha vist abans. Tots dos arguments de la resta d'operadors aritmètics han de ser expressions que puguen ser avaluades a un valor numèric. En altres paraules, si les expressions contenen variables aquestes hauran d'estar instanciades. Els valors numèrics són unificats o comparats fent els corresponents canvis de tipus si escau.

nom/aritat	descripció	associatiu
is/2	avaluació/unificació	no
=:=/2, =\=/2	comparació aritmètica	no
>=/2, =</2	comparació aritmètica	no

A.3.4 Operadors aritmètics

Es tracta dels operadors que també estan presents en la majoria de llenguatges de programació. Tant els seus arguments com el seu resultat són expressions aritmètiques que poden contindre variables. És a dir, a diferència dels anteriors, les expressions resultants són termes qualssevol.

nom/aritat	descripció	associatiu
+/2, -/2	addició/substracció	esquerra
+/1, -/1	signe	
*/2, //2	multiplicació/divisió	esquerra
///2	divisió entera	esquerra
mod/2	residu de la divisió entera	esquerra
^/2	potència	dreta

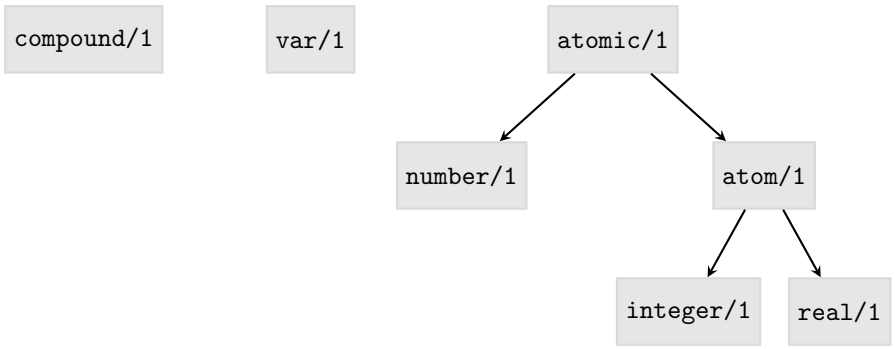
A.3.5 Funcions aritmètiques

Les funcions aritmètiques són functors predefinits que avaluen els seus arguments i donen com a resultat un valor numèric. Per tant, els seus arguments han de ser expressions que puguen ser avaluades.

abs/1	valor absolut	sign/1	funció signe
floor/1	funció sol	ceiling/1	funció sostre
exp/1	exponenciació	log/1	logaritme natural
mod/2	funció mòdul	round/1	enter més pròxim
max/2	funció màxim	min/2	funció mínim
e/0, pi/0	nombres e i π	nan/0	no és un nombre
inf/0	infinit	epsilon/0	valor real més petit

A.3.6 Predicats sobre termes

compound/1	és compost (no atòmic)	ground/1	sense variables lliures
var/1	és variable	atomic/1	és atòmic: àtom o nombre
atom/1	és àtom	number/1	és nombre: enter o real
integer/1	és enter	real/1	és real



A.3.7 Predicats sobre llistes

append/3	append(L1,L2,L12)	L12 és la concatenació de L1 i L2
select/3	select(E,L,R)	R és la llista L sense l'element E
member/2	member(E,L)	E és un element de L
is_list/1	is_list(L)	L és una llista
length/2	length(L,N)	la llista L té N elements

A.3.8 Predicats sobre enters

between/3	between(N,M,K)	N≤K≤M
succ/2	succ(N,M)	N, M consecutius i no negatius
plus/3	plus(N,M,S)	S=N+M
divmod/4	divmod(N,M,Q,R)	N=M∗Q+R

A.4 Algunes relacions entre lògica i PROLOG--

A.4.1 Fets

Els fets en PROLOG-- són fórmules atòmiques, encara que es pot afirmar la conjunció de diverses coses simplement en afirmar separadament cada una d'elles.

Els fets també es poden veure com (conseqüents de) regles l'antecedent de les quals és cert. De fet es poden escriure així en PROLOG-- encara que no té cap sentit fer-ho.

`plou.`	P	`plou :- true`	$T \Rightarrow P \equiv P$
`plou.` `neva.`	$P \wedge N$	`p(f(a,f(b))).`	$P(f(a, f(b)))$
`animal(nemo).`	$A(w)$	`oblida(dorothy,nemo).`	$O(d, n)$

Els fets que contenen variables duen un quantificador universal implícit per cada variable diferent.

`animal(X).`	$\forall x, A(x)$	`oblida(dorothy,X).`	$\forall x, O(d, x)$
`oblida(X,Y).`	$\forall x, \forall y, O(x, y)$	`oblida(X,X).`	$\forall x, O(x, x)$

Els fets anteriors són perfectament vàlids en PROLOG--. Però si s'inclou en un programa algun dels tres primers, l'intèrpret emetrà un avís (*warning*) per que hi ha variables que apareixen una única vegada i per tant no és necessari donar-los un nom. La forma en què caldria expressar aquests fets en PROLOG-- requereix l'ús de la variable anònima. Per exemple, els fets anteriors s'escriurien com

`animal(_).`	$\forall x, A(x)$	`oblida(dorothy,_).`	$\forall x, O(d, x)$
`oblida(_,_).`	$\forall x, \forall y, O(x, y)$	`oblida(X,X).`	$\forall x, O(x, x)$

A.4.2 Regles

Les regles en PROLOG-- s'han de poder escriure com a clàusules de Horn. En la pràctica sempre s'ha d'intentar escriure regles amb un conseqüent i un antecedent format per

la conjunció d'una o més fórmules atòmiques fent servir preferentment línies diferents convenientment indentades.

```
oblida(dorothy,X) :-
   peix(X),
   amic(X,dorothy),
   neva.
```

$$\forall x, \Big(P(x) \wedge A(x,d) \wedge N\Big) \Rightarrow O(d,x)$$

En les regles s'intenta minimitzar l'ús de la disjunció. Per això se solen descomposar antecedents formats per disjuncions en diverses regles fent servir la pseudodistributivitat de la implicació per la dreta.

```
oblida(dorothy,X) :-
   peix(X);
   animal(X).
```

$$\forall x, (P(x) \vee A(x)) \Rightarrow O(d,x)$$

```
oblida(dorothy,X) :-
   peix(X).
oblida(dorothy,X) :-
   animal(X).
```

$$\Big(\forall x, P(x) \Rightarrow O(d,x)\Big) \wedge \Big(\forall x, A(x) \Rightarrow O(d,x)\Big)$$

De fet, en els antecedents de les regles es poden combinar arbitràriament conjuncions i disjuncions ja que sempre es pot trobar una fórmula equivalent formada per clàsules de Horn. No obstant això, en la pràctica habitual es prefereix utilitzar més regles més senzilles i evitar si es pot la disjunció.

```
oblida(dorothy,X) :-
   peix(X),
   ( amic(X,dorothy);
      neva ),
   amic(dorothy,X).
```

$$\forall x, \Big(P(x) \wedge (A(x,d) \vee N) \wedge A(d,x)\Big) \Rightarrow O(d,x) \equiv$$

$$\equiv \Big(\forall x, \big(P(x) \wedge A(x,d) \wedge A(d,x)\big) \Rightarrow O(d,x)\Big) \wedge \Big(\forall x, \big(P(x) \wedge N \wedge A(d,x)\big) \Rightarrow O(d,x)\Big)$$

Com ja s'ha explicat, per cada variable diferent que apareix en el conseqüent hi ha un quantificador universal sobre tota la regla. Per cada variable diferent que apareix en l'antecedent i no en el conseqüent, es pot considerar que hi ha un quantificador existencial que s'aplica sobre l'antecedent. Per exemple,

```
oblida(dorothy,X) :-
   peix(Y),
   ( amic(X,Y);
      neva ).
```

$$\forall x, \Big(\exists y : \big(P(y) \wedge (A(x,y) \vee N)\big)\Big) \Rightarrow O(d,x)$$

Alternativament i de manera equivalentment, es pot considerar que les variables que apareixen només en l'antecedent tenen associat un quantificador universal aplicat sobre tota la regla. Aquesta equivalència és deguda a la pseudodistributivitat de la implicació per la dreta respecte dels quantificadors (conjuncions/disjuncions).

```
oblida(dorothy,X) :-
   peix(Y),
   ( amic(X,Y);
      neva ).
```

$$\forall x, \forall y, \Big(P(y) \wedge \big(A(x,y) \vee N\big)\Big) \Rightarrow O(d,x)$$

A.4.3 Preguntes

Les preguntes en PROLOG-- són expressions formades mitjançant predicats, conjuncions i disjuncions exactament igual que en els antecedents de les regles.

`?- plou,neva.`	$P \wedge N$	`?- plou;neva.`	$P \vee N$
`?- animal(nemo).`	$A(w)$	`?- oblida(dorothy,nemo).`	$O(d,n)$

Per cada variable que apareix en una pregunta es pot considerar que hi ha un quantificador existencial.

`?- animal(X),peix(X).`	$\exists x : \big(A(x) \wedge P(x)\big)$
`?- oblida(X,Y);peix(Y).`	$\exists x, \exists y : \big(O(x,y) \vee P(y)\big)$
`?- oblida(_,_);peix(_).`	$\exists x, \exists y, \exists z : \big(O(x,y) \vee P(z)\big)$

A.4.4 La negació en PROLOG--

La diferència més notable entre el que es pot expressar en PROLOG-- en relació a la lògica és la negació ja que aquesta no existeix en PROLOG--. El més pròxim que es té és el pseudopredicat `not/1` que es pot usar en preguntes i en antecedents de regles de manera que el seu resultat és cert o fals en funció de que PROLOG-- puga demostrar o no la seua veracitat. D'aquesta manera es poden expressar algunes fórmules amb negacions.

```
oblida(dorothy,X) :-
   peix(X),
   not((
      amic(X,dorothy),
      neva )).
```

$$\forall x, (P(x) \wedge \neg(A(x,d) \wedge N)) \Rightarrow O(d,x)$$

En la pràctica és convenient evitar la negació tant com siga possible ja siga re-estructurant la fórmula o fins i tot definint predicats explícits per a alguna cosa i la contraria.

A.4.5 Alguns exemples

Germanastres: Defineix la relació `esGermanastreDe/2` en funció de la relació `esFillDe/2`.

La relació/predicat que suposem donat, `esFillDe/2`, l'escriurem en lògica com un predicat binari, F, de manera que $F(x,y)$ siga cert si x és fill de y i fals en cas contrari.
Dues persones són germanes si comparteixen els dos progenitors. I són germanastres si només en comparteixen un dels dos. El més fàcil és escriure una fórmula que (per a tot x i y) siga certa **sii** dues persones son germanes o germanastres. És a dir,

$$P_1(x,y) \equiv \Big(\exists z : \ F(x,z) \wedge F(y,z) \wedge (x \neq y)\Big)$$

La desigualtat és necessària per que normalment no diem mai que algú és germà o germanastre d'ell mateix.
De la mateixa manera que abans, es pot escriure una fórmula que siga certa **sii** dues persones tenen almenys un progenitor diferent.

$$P_2(x,y) \equiv \Big(\exists z : \ F(x,z) \wedge \neg F(y,z)\Big)$$

Una primera solució per a la relació germanastre seria definir el nostre predicat, G, com la conjunció dels dos anteriors.

$$G(x,y) \equiv P_1(x,y) \wedge P_2(x,y)$$

Però també podem simplificar l'expressió anterior si canviem el nom de la variable en P_2 i apliquem la propietat distributiva sobre els quantificadors existencials (disjuncions) i les conjuncions. Al final podem arribar a

$$G(x,y) \equiv \exists z, \exists t : \quad \Big(F(x,z) \wedge F(y,z)\Big) \wedge \Big(F(x,t) \wedge \neg F(y,t)\Big)$$

on hem eliminat a més a més la desigualtat de P_1 per que no pot passar a la vegada que $x = y$ i que $P_2(x,x)$ siga cert.
Com a conclusió, la definició anterior en forma de regla en lògica i PROLOG-- podria ser

```
esGermanastreDe(X,Y) :-
   esFillDe(X,Z),
   esFillDe(Y,Z),
   esFillDe(X,T),
   not((
      esFillDe(Y,T))).
```

$$\forall x, \forall y, \quad \Big(\exists z, \exists t : \quad \big(F(x,z) \wedge F(y,z) \wedge F(x,t) \wedge \neg F(y,t)\big)\Big) \Rightarrow G(x,y)$$

La fórmula lògica anterior és també equivalent, com ja s'ha explicat abans, a

$$\forall x, \forall y, \forall z, \forall t, \quad \Big(\big(F(x,z) \wedge F(y,z) \wedge F(x,t) \wedge \neg F(y,t)\big) \Rightarrow G(x,y)\Big)$$

Com a exercici, raona si la següent definició alternativa del predicat G és o no correcta.

$$G'(x,y) \equiv \exists z, \exists t, \exists v : \quad \Big(F(x,z) \wedge F(x,t) \wedge \neg F(x,v) \wedge F(y,z) \wedge F(y,v) \wedge \neg F(y,t)\Big)$$

A.5 Problemes resolts i comentats

Problema A.1:[kami] En Takamagahara viuen els kamis. En particular, els amatsukamis són deus bons que sempre diuen la veritat. En canvi, els shinigamis són els deus de la mort i sempre diuen mentides. Tres kamis tenen la següent conversació:

a) Agyo: Bepo menteix!

b) Bepo: Calicarcha és un amatsukami.

c) Calicarcha: Agyo y Bepo són la mateixa cosa.

Quin tipus de kami són Agyo, Bepo i Calicarcha, respectivament? Podries resoldre el problema usant PROLOG--? Fes-ho.

Hi ha diverses maneres de representar aquest problema mitjançant lògica. D'entrada identifiquem el fet de ser amatsukami o shinigami amb el fet de dir sempre la veritat o no. Aleshores el més senzill és definir variables proposicionals, a, b i c, per a cada un dels 3 kamis.

Les 3 afirmacions es podrien escriure aleshores usant lògica proposicional com:

$$a \Leftrightarrow \neg b$$

$$b \Leftrightarrow c$$

$$c \Leftrightarrow (a \Leftrightarrow b)$$

Però també podríem definir un predicat `amatsukami/1` que siga cert o fals per a cada un dels kamis i tindríem una representació equivalent en lògica de predicats. Encara que es poden aplicar regles d'inferència estàndar prenent com a premisses aquestes representacions, no es pot arribar a un programa en PROLOG-- que reproduesca el raonament corresponent.

Una alternativa senzilla consisteix a definir un predicat ternari, `conversa(A,B,C)`, de manera que els seus tres arguments sobre el domini $\mathcal{K} = \{amatsukami, shinegami\}$, representen els kamis que tenen la conversa. El predicat serà cert si els arguments prenen valors compatibles amb el que s'afirma.

Cada afirmació, es representarà també mitjançant un predicat ternari `diuquees(X,Y,Z)` amb variables també sobre $\mathcal{K}$, de manera que `X` representa el (tipus de) kami que parla, `Y` representa el (tipus de) kami a qui es refereix, i `Z` representa el (tipus de) kami amb qui s'equipara `Y`.

Per fer que les variables prenguen els (s'instancien als) possibles valors dins de $\mathcal{K}$ es requereix també un predicat `kami/1` que es definirà amb dos fets, un per cada valor del domini. El resultat es mostra a continuació.

```
1.  kami(amatsukami).
2.  kami(shinigami).

    % diuquees(X,Y,Z) : X diu que Y és com (o és) Z
3.  diuquees(amatsukami ,K,K).
4.  diuquees(shinigami , K,Q) :-
5.     K≡Q.

6.  conversa(X,Y,Z) :-   % X,Y,Z representen a Agyo,Bepo,Calicarcha
7.     kami(X),kami(Y),kami(Z),
8.     diuquees(X,Y,shinigami),                    % a)
9.     diuquees(Y,Z,amatsukami),                   % b)
10.    diuquees(Z,X,Y).                            % c)
```

```
?-  conversa(A,B,C).
    A = amatsukami, B = C, C = shinigami.
```

Problema A.2:[bessones] Anna, Bea i Carme són tres bessones idèntiques. Però Anna sempre diu la veritat, Bea sempre menteix, i Carme de vegades diu la veritat i de vegades no. Sa mare les fa seure en tres cadires una al costat de l'altra i comença a preguntar d'esquerra a dreta ...

a) Qui és la que seu al mig? Anna, contesta la primera.

b) Qui eres tu? Sóc Carme, contesta la segona.

c) Qui seu a la teua esquerra? Bea, diu la tercera

Sabries endevinar qui és la que seu en cada cadira? Podries resoldre el problema usant PROLOG--? Fes-ho.

Representarem el domini format per les tres bessones mitjançant una llista amb l'ajuda del predicat `germanes/1` definit com

```
1. germanes([anna,bea,carme]).
```

El comportament a l'hora de contestar cada germana el representarem mitjançant el predicat `contesta/3` definit com a

```
2. contesta(anna,X,X).
3. contesta (bea,X,Y) :-
4.    X\==Y.
5. contesta(carme,_,_).
```

Representarem també la manera en què seuen les tres bessones mitjançant una llista de tres variables que podran seleccionar el seu valor del domini anterior sense repeticions.

Per a això, farem servir el predicat predefinit `select/3` de manera que quan una variable s'instancie a un valor, podrem obtindre el subdomini romanent per a poder seleccionar valors no repetits per a les variables.

El predicat `bessones/1` que resol el problema es mostra a continuació juntament amb la pregunta i la contestació corresponent que indica com seuen les tres bessones.

```
 6. bessones([Esq,Centr,Dre]) :-
 7.    domini(L),
 8.    select(Esq,L,L1), select(Centr,L1,[Dre]),
 9.    contesta(Esq, Centr,anna),
10.    contesta(Centr,Centr,carme),
11.    contesta(Dre, Centr,bea).
```

```
?- bessones(L).
   L = [carme, bea, anna] .
```

Problema A.3:[llop] Un pagès ha comprat en un mercat, un llop, una cabra i una col, i per a tornar a casa ha de travessar un riu amb una barca en què només cap ell i una de les tres coses que ha comprat. El llop es menjarà la cabra si es queda amb ella en qualsevol de les vores del riu sense la supervisió del pagès. I el mateix passarà amb la cabra i la col. De quina manera podrà el pobre home tornar

a casa amb tota la seua compra? Quants viatges en barca necessitarà? És possible obtindre la solució al problema usant PROLOG--? Com?

Representarem cada vora del riu mitjançant una llista que podrà contindre elements del domini

$$\{\texttt{barca},\texttt{llop},\texttt{cabra},\texttt{col}\},$$

on l'àtom `barca` representa tant la barca com el pagès. Inicialment tots els elements estan en una vora i es tracta de que tots passen a l'altra.

Definirem aleshores un predicat, `segur/1`, que siga cert si ningú es pot menjar a ningú **quan no està la barca** (i el pagès). Una possibilitat seria,

```
1. menja(L) :- member(cabra,L), member(col,L).
2. menja(L) :- member(llop,L), member(cabra,L).
3. segur(L) :-
4.    not(menja(L)).
```

I una altra possibilitat equivalent seria

```
1. segur([]).
2. segur([_]).
3. segur([llop,col]).
4. segur([col,llop]).
```

Definirem ara un predicat, `viatge/5`, de manera que

`viatge(L,R,LL,RR,Elem)`

siga cert si, donat el contingut de les dues vores en `L` i `R`, `Elem` és un element en la mateixa vora on està la barca i `LL` i `RR` és el contingut de les dues vores una vegada tant la barca com `Elem` passen d'una vora a l'altra.

Alternativament, la barca sempre pot passar d'una vora a l'altra de buit. En eixe cas, el cinquè argument s'instanciarà a l'àtom "o".

Suposarem, sense pèrdua de generalitat, que en la llista que representa el contingut de cada vora, la barca, si hi és, es troba en la primera posició. El ordre dels altres elements és en canvi irrelevant.

Aleshores, per a definir el predicat `viatge/5`, hi ha cuatre casos en funció d'on es troba la barca i si el viatge és de buit o no.

```
 5. viatge([barca|E],D,EE,[barca,A|D],A) :-
 6.    select(A,E,EE),
 7.    segur(EE).
 8. viatge(E,[barca|D],[barca,A|E],DD,A) :-
 9.    select(A,D,DD),
10.    segur(DD).
11. viatge([barca|E],D,E,[barca|D],o) :-
12.    segur(E).
13. viatge(E,[barca|D],[barca|E],D,o) :-
14.    segur(D).
```

Finalment, definirem un altre predicat recursiu, `viatges/6`, de manera que

`viatges(N,L,R,LL,RR,L_Elem)`

siga cert si, després de `N` viatges els continguts indicats per `L` i `R` han passat a ser `LL` i `RR` i `L_Elems` és una llista amb tots els elements que s'han mogut i en quin ordre, incloent-hi viatges buits.

Si considerem que la variable `N` ha d'estar instanciada, una definició possible seria

```
15. viatges(N,E,D,EEE,DDD,[Mov|Movs]) :-
16.    N>0,
17.    N1 is N-1,
18.    viatge(E,D,EE,DD,Mov),
19.    viatges(N1,EE,DD,EEE,DDD,Movs).

20. viatges(0,E,D,E,D,[]).
```

Considerant totes les definicions anteriors, podríem definir un últim predicat `solucio/2` que inicialitzara la vora esquerra amb tots els elements, exigira que aquesta vora esquerre acabe buida i intentara diferents nombres de moviments dins d'un rang, per exemple fins a 8. El resultat es mostra a continuació.

```
21. solucio(N, Movs) :-
22.    between(1,8,N),
23.    Ini = [barca,llop,cabra,col],
24.    viatges(N,Ini,[],[],_,Movs).
```

```
?- solucio(N,Movs).
   N = 7,
   Movs = [cabra, o, llop, cabra, col, o, cabra]
   N = 7,
   Movs = [cabra, o, col, cabra, llop, o, cabra]
   false.
```

El pagès necessita per tant fer almenys set viatges dos dels quals correspondran a tornades de buit.

En el primer i últim viatge transportarà la cabra, i per als tres viatges centrals té les dues possibilitats que ilustren les anteriors respostes de l'intèrpret.

Problema A.4:[minizebra] Hi ha 3 cases, en línia, de 3 colors diferents on viuen persones de 3 diferents nacionalitats, que els agraden 3 tipus d'animals diferents i juguen a 3 esports diferents. Considera les pistes següents:

a) El brasiler no viu en la segona casa.

b) A qui li agraden els gossos juga a bàsquet.

c) Hi ha una casa entre la casa on es juga a futbol i la casa roja.

d) A qui li agraden els peixos viu a l'esquerra de qui li agraden els gats.

e) La casa on els agraden els gossos està a la dreta de la verda.

f) L'alemà viu en la tercera casa.

Quin animal li agrada a l'angles? Qui juga a golf? Qui viu en la casa blava? Resol el problema també mitjançant PROLOG--.

Com que cada casa té associats quatre atributs (nacionalitat, color, animals i esport), considerarem un functor casa/4,

```
casa(Nacionalitat, Color, Animal, Esport)
```

per representar cada casa. I com que hi ha 3 cases alineades, considerarem una terna de functors com l'anterior,

$$\texttt{Cases=(C1,C2,C3)} \quad : \quad \texttt{Ci=casa(Ni,Ci,Ai,Ei)} \quad \texttt{i} \in \{1,2,3\}$$

Per resoldre el nostre problema definirem aleshores un predicat `problema/1` que prenga com a argument la terna de cases i comprove que es compleixen totes i cada una de les sis afirmacions.

```
1.  problema(CASES) :-
2.     pista1(CasaBrasiler,CASES),
3.     pista2(CasaGossosBasquet,CASES),
4.     pista3(CasaFutbol,CasaRoja,CASES),
5.     pista4(CasaPeixos,CasaGats,CASES),
6.     pista5(CasaGossos,CasaVerda,CASES),
7.     pista6(CasaAlema,CASES),

8.     CasaBrasiler =casa(brasiler,_,_,_),
9.     CasaGossosBasquet=casa(_,_,gossos,basquet),
10.    CasaFutbol =casa(_,_,_,futbol),
11.    CasaRoja =casa(_,roja,_,_),
12.    CasaPeixos =casa(_,_,peixos,_),
13.    CasaGats =casa(_,_,gats,_),
14.    CasaGossos =casa(_,_,gossos,_),
15.    CasaVerda =casa(_,verda,_,_),
16.    CasaAlema =casa(alema ,_,_,_).
```

En l'anterior codi s'haguera pogut canviar la variable `CasaGossos` per la variable `CasaGossosBasquet` ja que és evident que es tracta de la mateixa casa. Però s'ha preferit escriure una traducció literal de l'enunciat.

Per a convertir les sis afirmacions o pistes en codi observem que totes elles fan referència a les posicions de les cases dins la terna. Per això definirem un predicat general `estala/3` de manera que

```
estala(N,C,Cs)
```

se satisfà si una casa C està en una posició N en la terna Cs.

```
17. estala(1,A,(A,_,_)).
18. estala(2,A,(_,A,_)).
19. estala(3,A,(_,_,A)).
```

A partir d'aquest i per millorar la legibilitat definirem també el predicat esta/2 que se satisfà si una casa està en qualsevol posició.

```
20. esta(C,Cs) :-
21.    between(1,3,N),
22.    estala(N,C,Cs).
```

I dos nous predicats que consideren la posició relativa (a la dreta o a l'esquerra) de dues cases.

```
23. aladreta(A,B,(B,A,_)).
24. aladreta(A,B,(_,B,A)).
25. alesquerra(A,B,Cs) :- aladreta(B,A,Cs).
```

Amb els predicats anteriors, les definicions dels sis predicats corresponents a les afirmacions podrien ser

```
26. pista1(C,Cs) :-                          // No està la 2a
27.    member(N,[1,3]), estala(N,C,Cs).

28. pista2(C,Cs) :-                          // Hi ha una casa que...
29.    esta(C,Cs).

30. pista3(C1,C2,Cs) :-                      // No estan al mig.
31.    estala(1,C1,Cs),estala(3,C2,Cs).
32. pista3(C1,C2,Cs) :-
33.    estala(3,C1,Cs),estala(1,C2,Cs).

34. pista4(C1,C2,Cs) :-                      // A l'esquerra
35.    alesquerra(C1,C2,Cs).

36. pista5(C1,C2,Cs) :-                      // A la dreta
37.    aladreta(C1,C2,Cs).

38. pista6(C,Cs) :-                          // Està la 3a
39.    estala(3,C,Cs).
```

En realitat, la tercera s'haguera pogut expressar com la primera (per a cada una de les cases implicades), però s'ha fet de manera diferent.

Si fem ara la pegunta `problema(CASES).` observarem quina és la disposició de les cases que compleix totes les condicions imposades per les sis afirmacions.

```
?- problema(CASES).
   CASES = (casa(brasiler, _16890, peixos, futbol),
      casa(_16938, verda, gats, _16944),
      casa(alema, roja, gossos, basquet)) ;
   false.
```

Vegem que en la resposta no apareix res sobre l'anglès ni sobre el golf ni sobre la casa blava. Això és perque aquesta informació només es troba en les preguntes. La forma correcta de contestar la primera pregunta de l'enunciat seria

```
?- problema(CASES),esta(casa(angles,_,X,_),CASES).
   CASES = (casa(brasiler, _18390, peixos, futbol),
      casa(angles, verda, gats, _17984),
      casa(alema, roja, gossos, basquet)),
   X = gats ;
   false.
```

En canvi si intentem obtenir la contestació de la segona de la mateixa manera (i independentment de la primera) veurem que no obtenim resposta.

En altres paraules, no podem respondre (correctament) la segona pregunta sense saber que hi ha algú que és anglès. Tenim aleshores dues opcions, o fem les dues preguntes al mateix temps, o indiquem explicitament que en alguna casa viu un anglès.

En canvi, la tercera pregunta es pot fer independentment de les altres. No obstant això, el més natural seria fer les tres preguntes alhora.

```
?- problema(CASES),esta(casa(angles,_,X,_),CASES),
      esta(casa(Y,_,_,golf),CASES),
      esta(casa(Z,blava,_,_),CASES).
   CASES = (casa(brasiler, blava, peixos, futbol),
      casa(angles, verda, gats, golf),
      casa(alema, roja, gossos, basquet)),
   X = gats,
   Y = angles,
   Z = brasiler ;
   false.
```

Problema A.5:[ultim] Dissenya un predicat recursiu en PROLOG--, `ultim/3`, de manera que `ultim(L,E,R)` siga cert si la llista `L` és igual a la llista `R` afegint l'element `E` al final.

Com que ens demanen un predicat recursiu especificarem primer la idea recursiva. Considerarem el primer argument com a entrada, i es tracta de calcular quin és l'últim element de `L` i quina és la llista romanent quan aquest s'elimina.

La idea recursiva per calcular l'últim element és molt senzilla: l'últim d'una

llista (de dos o més elements) és el mateix que l´últim de la subllista sense el seu cap. En el cas base, l'últim d'una llista d'un element és eixe element. Això mateix, en PROLOG--,

```
1. ultim([_|L],E) :-
2.    ultim(L,E).
3. ultim([H],H).
```

```
?- ultim([a,b,c],E).
   E=c.
```

Tal i com s'ha escrit en el codi anterior, els casos recursiu i base no són estrictament exclusius ja que la cua L en la línia 1 pot ser buida. No obstant això, en eixe cas tindríem en la segona línia `ultim([],E)` que seria fals. O siga que la línia 2 està fent el mateix paper que la condició que caldria afegir per a comprovar que L **no** siga buida.

Per completar el disseny de `ultim/3` afegirem el tercer argument que haurà de contindre tots els elements del primer argument (i en el mateix ordre) llevat de l'últim. És a dir,

```
1. ultim([H|L],E,[H|R]) :-
2.    ultim(L,E,R).
3. ultim([H],H,[]).
```

```
?- ultim([a,b,c],E,R).
   E=c, R=[a,b].
```

Podem ara comparar aquesta versió recursiva amb la no recursiva, més natural, que faria ús del predicat estàndar append/3.

```
1. ultim(L,E,R) :-
2.    append(R,[E],L).
```

```
?- ultim([a,b,c],E,R).
   E=c, R=[a,b].
```

Problema A.6:[longitudLlista] a) Dissenya una funció recursiva que donada una llista i un element qualsevol torne el nombre de vegades que apareix aquest element en la llista. b) Expressa mitjançant lògica de predicats un predicat recursiu equivalent. c) Dissenya el corresponent predicat recursiu en PROLOG--.

La funció que se'ns demana és clarament una aplicació exhaustiva entre el conjunt de tots els possibles parells de llistes i elements, $\mathbb{L} \times \mathcal{U}$ (domini), i els enters no negatius, $\mathbb{Z}^+$ (codomini).

$$p : \mathbb{L} \times \mathcal{U} \longrightarrow \mathbb{Z}^+$$

Primer caldrà relacionar la funció per a un cas general (una llista qualsevol) i un cas més senzill (la mateixa llista sense el seu cap) la qual cosa és relativament directa en aquest cas.

Si l'element apareix n vegades en la llista sense el cap, a aquesta quantitat caldrà sumar 1 o 0 en funció de què el cap de la llista siga igual o no l'element.

I com en tota definició recursiva caldrà també definir els casos base que ara només serà un i molt senzill: en la llista buida qualsevol element apareix zero vegades.

Aleshores la definició recursiva que se'ns demana es pot escriure com a

$$p(L, e) = \begin{cases} p(cua(L), e), & \text{si } L \neq \emptyset \wedge cap(L) \neq e \\ 1 + p(cua(L), e), & \text{si } L \neq \emptyset \wedge cap(L) = e \\ 0, & \text{si } L = \emptyset \end{cases}$$

O simplement com a

$$p(L, e) = \begin{cases} 0, & \text{si } L = \emptyset \\ p(cua(L), e) + (cap(L) = e), & \text{si no} \end{cases}$$

si entenem $(cap(L) = e)$ com a funció característica.

Per al predicat corresponent considerarem només la implicació en un sentit (deducció a partir de casos més senzills). Aleshores podem escriure,

$$\forall L \in \mathbb{L}, \forall e \in U, \forall v \in \mathbb{Z}^+,\ p(L,e,v) \Leftarrow \begin{cases} (L \neq \emptyset \wedge (cap(L) \neq e) \wedge (p(cua(L),e,v)) \\ \vee \\ (L \neq \emptyset \wedge (cap(L) = e) \wedge (p(cua(L),e,v-1)) \\ \vee \\ (L = \emptyset \wedge v = 0) \end{cases}$$

O equivalentment d'una manera més usual en lògica de predicats,

$$\begin{cases} \forall L \in \mathbb{L}, \forall e \in U, \forall v \in \mathbb{Z}^+, (L \neq \emptyset \wedge (cap(L) \neq e) \wedge (p(cua(L),e,v)) \Rightarrow p(L,e,v) \\ \wedge \\ \forall L \in \mathbb{L}, \forall e \in U, \forall v \in \mathbb{Z}^+, (L \neq \emptyset \wedge (cap(L) = e) \wedge (p(cua(L),e,v-1)) \Rightarrow p(L,e,v) \\ \wedge \\ \forall e \in U, p(\emptyset,e,0) \end{cases}$$

I ara podem fer un canvi de variable per a escriure

$$\begin{cases} \forall L \in \mathbb{L}, \forall e,h \in U, \forall v \in \mathbb{Z}^+, (h \neq e) \wedge (p(cua(L),e,v)) \Rightarrow p(h|L,e,v) \\ \wedge \\ \forall L \in \mathbb{L}, \forall e \in U, \forall v \in \mathbb{Z}^+, p(cua(L),e,v-1) \Rightarrow p(e|L,e,v) \\ \wedge \\ \forall e \in U, p(\emptyset,e,0) \end{cases}$$

Aquesta darrera definició es tradueix a PROLOG-- d'una manera quasi literal.

```
p([H|L],E,V) :-  % Cas recursiu 1
   H\==E,
   p(L,E,V).

p([E|L],E,V) :-  % Cas recursiu 2
   p(L,E,V1),
   V is 1+V1.

p([],_,0).       % Cas base
```

Problema A.7:[esMembre] Dissenya un predicat recursiu que donat un element qualsevol i una llista siga cert o fals si l'element està o no està en la llista. Escriu diferents definicions si les trobes i les corresponents traduccions a PROLOG--.

La relació entre el predicat per a una llista qualsevol i la mateixa sense el seu cap depén de si el cap de la llista és o no igual a l'element. En particular, podem escriure

$$\forall e \in \mathcal{U}, \forall L \in \mathbb{L},\ m(e,L) \Leftarrow \begin{cases} ((L \neq \emptyset) \wedge (cap(L) = e)) \\ \qquad \vee \\ ((L \neq \emptyset) \wedge (cap(L) \neq e) \wedge (m(e, cua(L)))) \end{cases}$$

ja que només interessa poder demostrar quan el predicat és cert. En cas que es volguera demostrar també quan el predicat és fals caldria afegir

$$\forall e \in \mathcal{U}, \forall L \in \mathbb{L},\ \neg m(e,L) \Leftarrow \begin{cases} (L = \emptyset) \\ \qquad \vee \\ ((L \neq \emptyset) \wedge (cap(L) \neq e) \wedge (\neg m(e, cua(L)))) \end{cases}$$

Si ens restringim només al primer cas com és usual, podem reescriure la primera definició com a

$$\begin{cases} \forall e \in \mathcal{U}, \forall L \in \mathbb{L}, & m(e, e|L) \\ & \wedge \\ \forall e, h \in \mathcal{U}, \forall L \in \mathbb{L}, & ((h \neq e) \wedge m(e, cua(L))) \Rightarrow m(e, h|L) \end{cases}$$

En totes les definicions anteriors, les regles (el dos casos de la definició) són mútuament exclusives. És a dir, només un dels dos casos és cert al mateix temps.

Però si ho pensem be, la implicació

$$m(e, cua(L)) \Rightarrow m(e, h|L)$$

és certa independentment de què el cap de la llista, h, siga o no igual a e. En altres paraules, podem escriure una definició equivalent més senzilla on les regles no són mútuament exclusives.

$$\begin{cases} \forall e \in U, \forall L \in \mathbb{L}, & m(e, e|L) \\ & \wedge \\ \forall e, h \in U, \forall L \in \mathbb{L}, & m(e, cua(L)) \Rightarrow m(e, h|L) \end{cases}$$

La traducció a PROLOG-- d'aquestes definicions és la mateixa.

```
m(E,[H|L]) :-     % Cas recursiu
   H\==E,         % (*)
   m(E,L).

m(E,[E|_]).       % Cas base
```

L'única diferència entre les dues versions consistiria a incloure o no la línia marcada amb (*). Tot i que les dues definicions són lògicament equivalents, el funcionament dels corresponents programes en PROLOG-- seria diferent.

En la segona definició (sense la línia (*)), la recursió sempre esgotaria la llista d'entrada, i l'intèrpret de PROLOG-- contestaria per cada vegada que l'element apareguera en la llista.

En canvi, en la primera on les regles són excloents, l'intèrpret només contestaria la primera vegada que es trobara l'element en la llista i la recursió acabaria.

Per cert, la implementació del predicat predefinit `member/2` es correspon amb la segona de les definicions.

Problema A.8:[esMembreBis] Trobar una definició no recursiva en PROLOG--, a) per al predicat `member/2` en funció dels predicats `select/3` i/o `append/3`. b) per al predicat `select/3` en funció de `append/3`.

La definició en funció de `select/3` és molt senzilla ja que els dos predicats són molt similars.

```
m(E,L) :-
   select(E,L,_).
```

El mateix fent servir append/3 és relativament similar.

```
m(E,L) :-
   append(_,[E|_],L).
```

I finalment, la definició de `select/3` a partir de `append/3` és una generalització de l'anterior.

```
s(E,L,R) :-
   append(L1,[E|L2],L),
   append(L1,L2,R).
```

En realitat, la implementació anterior no és exactament equivalent a la implementació del predicat estàndar `select/3` ja que quan la primera de les dues llistes no està instanciada es produeix una recursió infinita.

Per a obtenir un comportament idèntic en aquest cas, caldria invertir les dues línies de què consta la definició anterior.

Des del punt de vista lògic, les dues definicions són la mateixa ja que la conjunció és commutativa.

A.6 Problemes proposats

Problema A.9:[bufons] En un regne molt, molt llunyà, tothom és cavaller, bufó o espia. Els cavallers sempre diuen la veritat, els bufons sempre diuen mentides, i els espies diuen el que els interessa.
Un cavaller, un bufó i un espia tenen una conversació:

a) Alleyn: Carvalho és un bufó.

b) Bond: Alleyn es un cavaller.

c) Carvalho: Jo sóc un espia.

Sabries endevinar qui és qui en la conversa? Podries resoldre el problema usant PROLOG--? Fes-ho.

Problema A.10:[paraigua] Quatre amics arriben a una festa un dia plujós i han de travessar el pati.
El primer problema és que només tenen un paraigua que com a molt tapa a dues persones.
L'altre problema és que, per diferents problemes físics, no tots ells són igual de ràpids i per travessar el patí necessiten 1, 2, 5 i 8 minuts, respectivament. Òbviament, quan dos van junts han d'anar a la velocitat del més lent.
El més ràpid proposa immediatament una solució: jo, que sóc el més ràpid, aniré i tornaré les vegades que faça falta i vos acompanyaré a tots. En total tardarem 17 minuts en 5 viatges: 2, 5 i 8 minuts per acompanyar-vos als tres i 2 viatges de tornada de 1 minut.
És possible que entren tots a la festa sense mullar-se en menys de 17 minuts? Com?
És possible obtindre la solució al problema usant PROLOG--? Com?

Problema A.11:[llopbis] Considera el problema del llop, la cabra i la col de la pàgina 177 i imagina que el pagès també ha comprat un lleó que només li agrada menjar-se llops. De quantes i quines maneres diferents pot arribar el pagès a casa? Amb quantes i quines coses arribaria en el pitjor cas? Amb quants viatges?
Per exemple, si transporta primer la col, el llop es podria menjar la cabra. I si en

tornar transportara el lleó i després el llop, arribaria a casa amb el lleó, el llop i la col.
Suposa, com en l'exemple, que en absència del pagès només pot ocórrer una consumició.

Problema A.12:[zebra] Hi ha 5 cases, en línia, de 5 colors diferents on viuen persones de 5 diferents nacionalitats, que tenen 5 tipus d'animals diferents, beuen 5 begudes diferents i fumen 5 marques diferents de cigarrets. Considera les pistes següents:

a) L'anglès viu a la casa roja.

b) L'espanyol té un gos.

c) En la casa verda es beu cafè.

d) L'ucraïnès beu te.

e) La casa verda està al costat de la blanca.

f) Qui fuma *Old Gold* té caragols.

g) En la casa groga es fuma *Kools*.

h) En la casa del mig beuen llet.

i) El noruec viu en la primera casa.

j) Qui fuma *Chesterfields* viu al costat de qui té una rabosa.

k) Fumen *Kools* al costat d'on tenen un cavall.

l) Qui fuma *Lucky Strike* beu suc de taronja.

m) El japonès fuma *Parliaments*.

n) El noruec viu al costat de la casa blava.

Contesta: Qui beu aigua? Qui té una zebra? Resol el problema també mitjançant PROLOG--.

Problema A.13:[invertir] Dissenya un predicat recursiu en PROLOG-- de 2 arguments que siguen llistes i que siga cert si una és igual a l'altra en ordre invers dels seus elements.

Problema A.14:[selectAppendRec] Dissenya una versió recursiva en PROLOG-- per als predicats `select/3` i `append/3`.

```
?- select(2,[1,2,3,2],X). --> X=[1,3,2]
?- append([1,2,3,2],[4,3,2],X). --> X=[1,2,3,2,4,3,2]
```

Problema A.15:[unioRec] Dissenya un predicat recursiu en PROLOG-- que donades tres llistes, siga cert si la tercera llista conté els elements que estiguen en alguna de les dues primeres sense cap repetició.

```
?- unio([1,2,3,2],[4,3,2],X). --> X=[1,2,3,4]
```

Problema A.16:[interseccioRec] Dissenya un predicat recursiu en PROLOG-- que donades tres llistes, siga cert si la tercera llista conté només els elements que estan en les dues primeres sense cap repetició.

```
?- interseccio([1,2,3,2],[4,3,2],X). --> X=[2,3]
```

Índex analític

Historial de versions

1.0 (15 de gener, 2020): versió enviada al Servei de Política Lingüística.

1.1 (25 de febrer, 2020): versió revisada i acabada.

2.0 (8 de març, 2020): primera versió bilingüe.

2.1 (24 de març, 2020): publicació en Roderic[†].

2.2 (18 de gener, 2021): afegit apèndix sobre Prolog

2.3 (19 d'agost, 2021): versió bilingüe completa

2.4 (4 d'agost de 2022): codificació UTF, errades, addicions, més problemes

[†] https://roderic.uv.es/handle/10550/73645