Universitat de Lleida

TESI DOCTORAL

Un Llenguatge Visual de Programació Lògica

Lleida, Juny de 2002

Memòria presentada per en **Jordi Puigsegur Figueras** per optar al títol de Doctor en Informàtica per la Universitat de Lleida sota la direcció del Dr. **Jaume Agustí Cullell**. El treball contingut en aquesta memòria ha estat realitzat a l'Institut d'Investigació en Intel·ligència Artificial (IIIA) del Consell Superior d'Investigacions Científiques (CSIC).

Als meus pares
i a la Cati.

Resum

El treball central d'aquesta tesi ha consistit en dissenyar una sintaxi visual per a la lògica de primer ordre inspirada i dirigida per la semàntica dels predicats, és a dir, basada en la interpretació dels predicats com a conjunts de tuples. Per aconseguir-ho proposem una representació visual on els dos punts clau són: 1) representar gràficament els predicats utilitzant les seves abstraccions conjuntistes o conjunts extensió i 2) la utilització de la inclusió gràfica per a denotar la implicació lògica. Després d'haver explorat totes les possibilitats d'aquest apropament ens hem centrat en el disseny d'un llenguatge de programació lògica, fent èmfasi en la seva semàntica operacional. Proposem una regla d'inferència visual similar a la resolució SLD habitual a la programació lògica, basada en transformacions de diagrames que capturen implícitament la transitivitat de la implicació lògica mitjançant la inclusió gràfica. Una aportació del nostre apropament visual a la programació lògica és la possibilitat de visualitzar en un únic diagrama diferents solucions juntament amb el camí seguit per a trobarles, és a dir, la seva traça o demostració. Finalment proposem una aplicació a la representació visual d'esquemes de bases de dades deductives i mostrem la implementació d'un prototip d'entorn de programació amb un editor dirigit per la sintaxi que ens permet prescindir d'un analitzador sintàctic.

Abstract

The main contribution of this thesis is a visual syntax for first order logic that was inspired and guided by the semantics of the predicates, i.e., the interpretation of the predicates as sets of tuples. The two basic principles used in the design of this visual syntax were: to represent predicates graphically using their set abstractions, or extension sets, and to use graphical inclusion to denote logical implication. After exploring many features of this approach, we have focused our work on the design of a visual language for logic programming, emphasizing its operational semantics. We propose a visual inference rule, similar to SLD resolution (commonly used in logic programming), based on diagram transformations that implicitly captures the transitivity of logical implication through graphical inclusion. An interesting result of our visual approach is the ability to visualize different solutions in a single diagram, together with their paths followed, i.e., their traces or proofs. Finally we propose an application of this approach to the visual representation of deductive database schemas, and we present the implementation of a programming environment prototype in the form of a syntax-directed editor that allows us to skip diagram parsing.

Resumen

El trabajo central en esta tesis ha consistido en diseñar una sintaxis visual para la lógica de primer orden, inspirada y dirigida por la semántica de los predicados, es decir, basada en la interpretación de los predicados como conjuntos de tuplas. Con el fin de conseguirlo proponemos una representación visual en la que los dos puntos claves son: 1) representar gráficamente los predicados a través de sus abstracciones conjuntistas o conjuntos extensión, y 2) la utilización de la inclusión gráfica para denotar la implicación lógica. Después de haber explorado todas las posibilidades de este enfoque nos hemos centrado en el diseño de un lenguaje de programación lógica, incidiendo especialmente en su semántica operacional. Proponemos una regla de inferencia visual similar a la resolucion SLD habitual en la programación lógica, basada en transformaciones de diagramas que capturan implícitamente la transitividad de la implicación lógica mediante la inclusión gráfica. Una aportación de nuestro enfoque visual a la programación lógica es la posibilidad de visualizar en un único diagrama diferentes soluciones, junto con el camino seguido para encontrarlas, es decir, su traza o demostración. Finalmente proponemos una aplicación a la representación visual de esquemas de bases de datos deductivas y mostramos la implementación de un prototipo de entorno de programación con un editor de diagramas dirigido por la sintaxis que nos permite prescindir de un analizador sintáctico.

Agraïments

Aquesta tesi conté el treball d'investigació realitzat durant més de quatre anys a l'Institut d'Investigació en Intel·ligència Artificial, on hi vaig trobar un entorn humà i científic de primer ordre i em van donar totes les facilitats materials per tal de poder realitzar aquest treball. Tot plegat ha estat possible gràcies al finançament ofert per la CICYT a través dels projectes DISCOR i MODELOGOS (TIC 94-0847-C02-01 i TIC 97-0579-C02-01), així com per la CIRIT mitjançant una beca FI. Part de la investigació s'ha dut a terme durant una estada al Visual Inference Laboratory, de la Universitat d'Indiana, on hi vaig estar convidat per Jon Barwise. En aquest darrer any el departament d'informàtica de la UdL m'ha proporcionat tots els mitjans necessaris per a poder completar aquesta tesi doctoral. I també ha estat important la correcció gramatical i d'estil realitzada per la meva germana, Marta Puigsegur.

He d'agrair molt sincerament a tots aquells que han col·laborat en aquest treball, durant els diversos anys que ha durat la seva realització i sense els quals no hauria estat possible. Entre ells destaco Dave Robertson, que va proporcionar la inspiració inicial de GraSp. Joan Antoni Pastor, de la Universitat Internacional de Catalunya, juntament amb el grup de l'Antoni Olivé de la Universitat Politècnica de Catalunya, pel suport donat i la col·laboració en l'aplicació a les bases de dades deductives. I Felip Manyà, de la Universitat de Lleida, amb Joan Antoni Pastor, per l'encoratjament final a acabar aquesta tesi. També vull agrair a Marco Schorlemmer tota la col·laboració donada durant els quatre anys a l'IIIA, i especialment pels seus valiosos comentaris sobre la versió final de la tesi, que han estat de gran utilitat. I finalment al meu director, Jaume Agustí, li he d'agrair l'encoratjament i guiatge constants, així com la paciència que en més d'una ocasió m'ha demostrat. Ben segur que sense ell aquesta tesi no hauria arribat a bon terme.

Índex

Índex de figures

Part I

Introducció

A major concern to the founders of modern logic—Frege, Peirce, Russell, and Hilbert—was to give an account of the logical structure of valid reasoning. Taking valid reasoning in mathematics as paradigmatic, these pioneers led the way in developing the accounts of logic which we teach today and that underwrite the work in model theory, proof theory, and definability theory. The resulting notions of proof, model, formal system, soundness and completeness are things that no one claiming familiarity with logic can fail to understand, and they have also played an enormous role in the revolution known as computer science.

The success of this model of inference led to an explosion of results and applications. But it also led most logicians—and those computer scientists most influenced by the logic tradition—to neglect forms of reasoning that did not fit well within this model. We are thinking, of course, of reasoning that uses devices like diagrams, graphs, charts, frames, nets, maps, and pictures.

Jon Barwise i John Etchemendy, *Heterogeneous Logic*, 1996 ([12])

Capítol 1

Introducció

En les dues darreres dècades del segle XX hem pogut constatar un augment sense precedents en la potència de càlcul dels ordinadors i en les capacitats de les seves interfícies gràfiques. Això ha propiciat l'estudi de formes de representació no textual en molts àmbits del coneixement. En concret ha donat lloc a dues línies d'investigació : el raonament diagramàtic i la programació visual. Els llenguatges visuals ens ofereixen una plataforma sintàctica diferent, encara desconeguda en gran mesura, per a desenvolupar llenguatges declaratius. Creiem que la pragmàtica i la comprensió de la lògica formal i dels llenguatges de programació declarativa són sensibles al tipus de sintaxi utilitzada. El nostre interès es centra en l'estudi d'aquestes noves formes d'expressió visual en el marc de la lògica, i concretament en la programació lògica.

Creiem que una bona sintaxi, visual o textual, hauria de ser propera a allò que vol representar, és a dir, la seva semàntica. Tal com argumenten Barwise i Etchemendy [12] una propietat bàsica de la sintaxi emprada en un sistema de raonament és l'existència d'una correspondència amb la seva semàntica. Si analitzem els llenguatges de programació lògica, i la lògica de primer ordre en general, podem veure que la sintaxi textual dels predicats no ens informa del seu significat pretès: un conjunt de tuples. La sintaxi estàndard de la lògica va ser dissenyada per matemàtics, inspirada en l'àlgebra de Boole. Aquesta sintaxi textual afavoreix la interpretació dels predicats com a funcions booleanes i no ens suggereix la seva interpretació com a conjunts de tuples. El nostre objectiu és trobar noves formes d'expressió que permetin reduir la distància entre la sintaxi i la semàntica, aprofitant aquesta interpretació natural dels predicats com a conjunts i utilitzant intuïcions familiars per la majoria de nosaltres: la representació gràfica de conjunts com a diagrames de Venn / Euler, i la utilització de grafs en la representació de termes estructurats.

En aquest treball proposem una sintaxi visual alternativa per a la lògica de primer ordre basada en l'ús de diagrames topològics, concretament en una adaptació dels higraphs (un formalisme topològic combinació de grafs i diagrames de Venn / Euler). L'eix central de la nostra proposta de representació és l'abstracció gràfica dels predicats. Tal com quedarà clar al capitol 3, es repre-

senten gràficament els predicats com els conjunts d'elements que els compleixen, i s'utilitza la inclusió gràfica per a denotar la implicació lògica. Aquest apropament original a la programació lògica ens permet obtenir diagrames altament expressius a partir d'unes poques primitives gràfiques. Basant-nos en aquesta sintaxi, proposem un llenguatge visual de programació lògica del qual en definim la sintaxi, la semàntica denotacional i una semàntica operacional completament visual similar a la resolució SLD. El nostre pla de treball inicial consistia en explorar tots els àmbits possibles de la nostra proposta de llenguatge visual de programació lògica, incloent-hi l'estudi de la pragmàtica i l'estudi empíric de la usabilitat del llenguatge. No obstant aquestes parts no s'han completat a causa de la envergadura del problema que per si mateix podria constituir una altra tesi doctoral.

El contingut d'aquesta tesi està dividit en quatre parts. La primera conté els capítols introductoris. Al capítol 2 fem un repàs a l'estat de l'art en la utilització de formalismes visuals al raonament diagramàtic i a la programació visual, fent èmfasi en aquells sistemes i/o llenguatges que considerem més rellevants per aquesta tesi. A continuació, al capítol 3 donem les intuïcions en què es basa la representació visual proposada i introduïm una sintaxi visual per a la lògica de primer ordre que és el treball previ al disseny del llenguatge visual de programació lògica.

A la segona part s'introdueix el llenguatge visual de programació lògica. Aquesta és la part central de la tesi i comprèn els capítols 4 i 5. En el primer definim formalment la sintaxi i la semàntica denotacional del llenguatge visual de programació lògica proposat, que utilitza un subconjunt de la sintaxi introduïda al capítol 3. Aquest llenguatge ha estat dissenyat pensant en la seva utilització com a llenguatge de programació lògica i té una potència expressiva similar a les clàusules de Horn. Al capítol 5 definim la seva semàntica operacional, que és similar a la resolució SLD i està basada en una regla d'inferència completament visual. Una aportació important d'aquesta semàntica operacional és la seva expressivitat en mostrar les solucions. Mentre que en la programació lògica és habitual mostrar les solucions una a una, mitjançant el llenguatge que proposem és possible mostrar diferents solucions en un mateix diagrama resposta, juntament amb el camí seguit per a obtenir-les, és a dir, la traça.

A la tercera part proposem una aplicació d'aquesta notació visual i s'adreça la implementació d'un entorn de programació lògica visual. Al capítol 6 estudiem la representació visual d'esquemes de bases de dades deductives utilitzant la sintaxi visual proposada anteriorment. A continuació, al capítol 7 estudiem la implementació d'un entorn de programació per al llenguatge visual proposat. En concret estudiem el disseny d'un editor de diagrames dirigit per la sintaxi, i se'n mostra el primer prototipus. Finalment, en la darrera part, al capítol 8 exposem les conclusions d'aquest treball.

Capítol 2

El marc de la programació visual declarativa

Des de temps immemorials els humans ens hem comunicat entre nosaltres utilitzant imatges. Els homes de les cavernes pintaven les parets de les coves, els egipcis utilitzaven jeroglífics i, per citar un exemple actual, la majoria de senyals de tràfic són icones amb un significat sovint fàcil d'entendre per a un neòfit. L'objectiu d'aquest capítol és contextualitzar el treball de definició del llenguatge visual per a la programació lògica proposat en aquesta tesi. Les idees de visualització en què es basa aquesta tesi provenen principalment de dues àrees:

- el raonament diagramàtic
- la programació visual

Dels llenguatges de programació visual prové l'experiència en la utilització d'una sintaxi visual en el disseny d'un llenguatge de programació. El raonament diagramàtic és la llavor que ens va empènyer a representar la programació declarativa de forma diagramàtica. Tal com veurem en aquest capítol i els següents, la idea fonamental del raonament diagramàtic —la proximitat de la sintaxi i la semàntica; el que Jon Barwise anomena homomorfisme [14]— és el motor que ens ha portat a dissenyar un llenguatge visual de programació lògica.

Al llarg d'aquest treball utilitzem el terme *visual* en contraposició a textual, és a dir, per referir-nos a tots aquells llenguatges o representacions lingüístiques amb elements no textuals. També utilitzem sovint el terme *diagramàtic*, el qual per a nosaltres té un significat més concret, tot i que també discutible. Entenem per *diagrama* una representació visual acotada a l'espai i que utilitza uns elements gràfics o diagramàtics prèviament definits.

Aquest capítol està estructurat de la següent forma: primer, a la secció 2.1 repassem els primers casos d'utilització de representacions visuals, és a dir, els seus precursors. Tot seguit, a la secció 2.2 estudiem el treball realitzat en l'àrea del raonament diagramàtic i a la secció 2.3 el realitzat en l'àrea de la programació visual.

2.1. Els precursors

L'ús de diagrames o representacions gràfiques en el discurs matemàtic no és cap descobriment del s.XX, sinó que ja estava present a les civilitzacions antigues. Quan hi havia molt menys coneixement matemàtic i científic i encara no s'havien desenvolupat les formalitzacions textuals actuals era comú l'explicació del coneixement a través de dibuixos, esquemes, etc. Es creu que a la Grècia clàssica era habitual la utilització de diagrames per part dels filòsofs. En moltes explicacions matemàtiques, especialment aquelles de geometria, es fa referència, sovint explícita, a diagrames. També hi ha referències de la utilització de representacions diagramàtiques en demostracions matemàtiques a la cultura xinesa (Kulpa [44]) i de representacions visuals a les cultures americanes pre-hispàniques (Noriega [58]).

Acostant-nos més als nostres dies, hi ha evidències en l'ús de diagrames en la il·lustració de raonaments matemàtics al s.XVI, però és amb Leibniz, al s.XVII, quan es van estudiar d'una forma metòdica. De fet Leibniz és considerat el pare de la matemàtica moderna, donat que va assentar les bases de la seva formalització textual actual i va introduir una metodologia de treball. Hi ha però un altre treball menys conegut de Leibniz: la seva "characteristica universalis" o l´intent de crear un llenguatge universal basat en un alfabet del pensament.

Leibniz va aparèixer en el moment adequat per tal de poder aprofitar moltes idees anteriors. No obstant ni la idea d'un llenguatge artificial ni la de reduir el raonament a un simple càlcul automatitzable eren completament noves. Remuntant-nos al S.XIII trobem Ramon Llull. Els més agosarats sostenen que Llull va dissenyar l'any 1275 el que es podria considerar com el primer ordinador de la història (Bexte i Künzel [17]). Si bé aquesta afirmació és, com a mínim, força arriscada, no deixa de tenir el seu fonament. L'objectiu de Ramon Llull era convertir els infidels a la religió catòlica i amb aquesta finalitat va dissenyar vàries màquines combinatòries a base de símbols, cercles i taules. Mitjançant aquestes màquines Llull pretenia demostrar la racionalitat de la creença en la fe cristiana i convèncer així els musulmans —que en aquells temps encara poblaven l'illa de Mallorca— que havien de convertir-se al cristianisme. L'enfocament de Ramon Llull al dissenyar aquestes màquines introdueix conceptes que s'avancen 700 anys al seu temps (Sales [63, 64]). Pel que respecta als diagrames i a l'ús d'informació visual Llull destaca per dues aportacions:

- Utilitza diagrames per tal de representar conceptes, i utilitza les relacions gràfiques entre els diferents elements diagramàtics per tal de representar les relacions entre aquests conceptes. En moltes màquines lul·lianes els cercles representaven conceptes i segons la ubicació dels cercles, tenint en compte la forma en que estan units i si es superposen o no, hom podia deduir l'afinitat dels conceptes representats.

- Utilitza mecanismes diagramàtics mecànics per donar una dinàmica a les seves màquines i utilitzar-les com a eina per a calcular (computar), en el seu cas, la veritat de l'existència de Déu. Són transformacions (moviments)

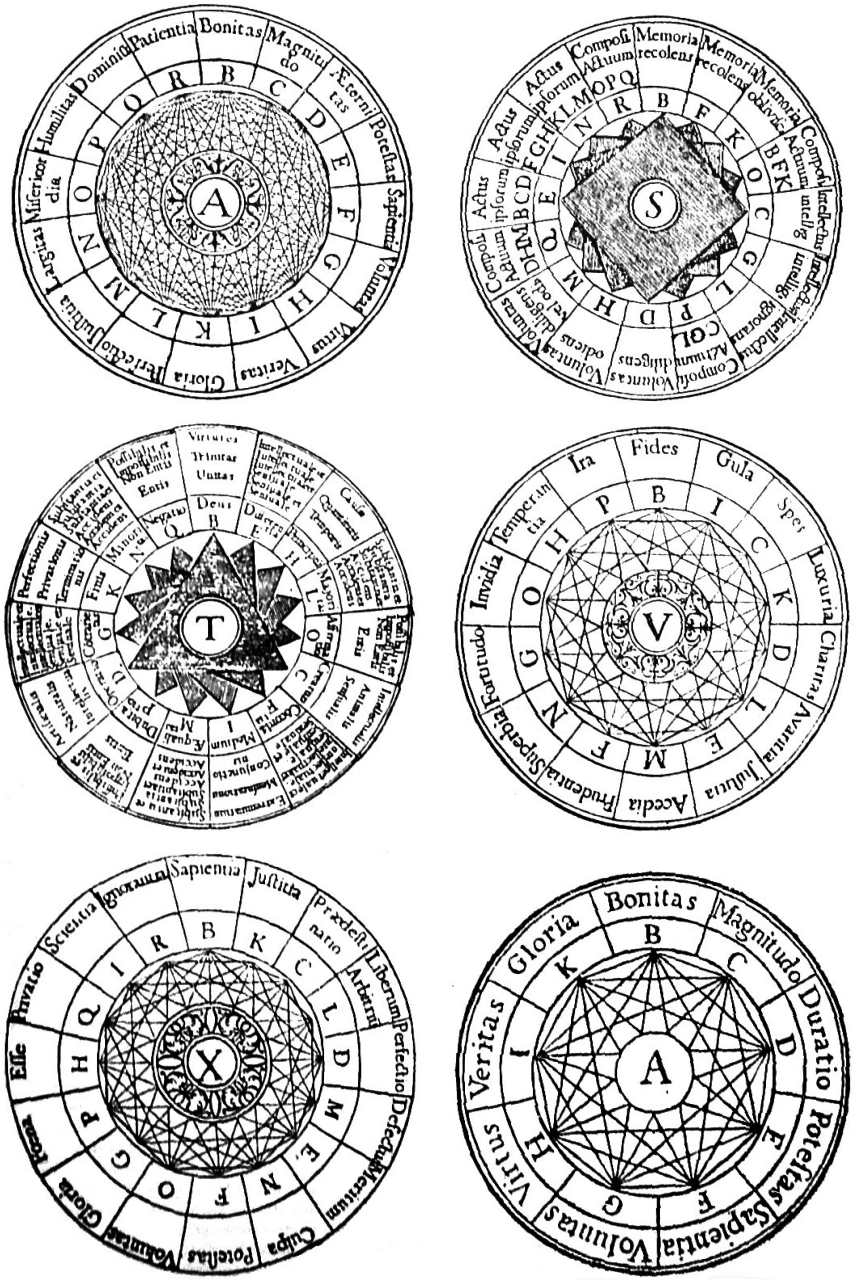


Figura 2.1: Figures de l'Ars Magna de Ramon Llull

regides per regles gràfiques que intenten demostrar propietats o fets dels conceptes representats pels elements gràfics de les màquines.

A la figura 2.1 trobem exemples de màquines lul·lianes del llibre Ars Magna[1]. Llull creia que a cada branca de coneixement hi ha un nombre petit de principis bàsics, acceptats per tothom. Mitjançant la combinació de tots aquest principis o categories hom pot explorar tot el coneixement. Aquests cercles no són més que màquines combinatòries que ens permeten obtenir aquestes combinacions (Gardner [33]).

El raonament diagramàtic té un clar exponent en el S.XVIII, el matemàtic suís Leonhard Euler. Euler introdueix les bases de la teoria de grafs per tal de solucionar el problema dels ponts de Königsberg. Així mateix, en les seves *Cartes a una princesa Alemanya* utilitza per primer cop els diagrames que actualment coneixem com a diagrames de Venn (Hammer i Shin [40],[2]). Euler utilitza cercles per a representar conjunts i les relacions que s'estableixen entre ells, a partir d'una representació diagramàtica de proposicions com "Tot A és B" o "Cap A és B" mitjançant la inclusió —i la no inclusió— gràfica. A la figura 2.2 veiem exemples dels *cercles d'Euler*. Per tal d'augmentar-ne la capacitat expressiva Venn proposa la noció de diagrames primaris, on tots els conjunts tenen intersecció amb tots els altres. Més endavant Peirce proposa —abans de dissenyar els grafs existencials— una nova modificació als diagrames d'Euler i Venn, tot introduint mecanismes per a la representació d'informació existencial i disjuntiva. Aquesta versió dels diagrames de Venn és la que més s'ha utilitzat. Els diagrames d'Euler són un dels fonaments del llenguatge visual proposat en aquesta tesi. A la secció 2.2.2 d'aquest capítol veurem sistemes moderns de raonament basats en diagrames de Venn.

[1]No es conserven manuscrits originals de Ramon Llull i per tant tot el que coneixem ens ha arribat a través del que s'ha anat transmetent entre els seguidors de Llull, moltes vegades de forma oral. Aquestes il·lustracions han estat extretes de [1].

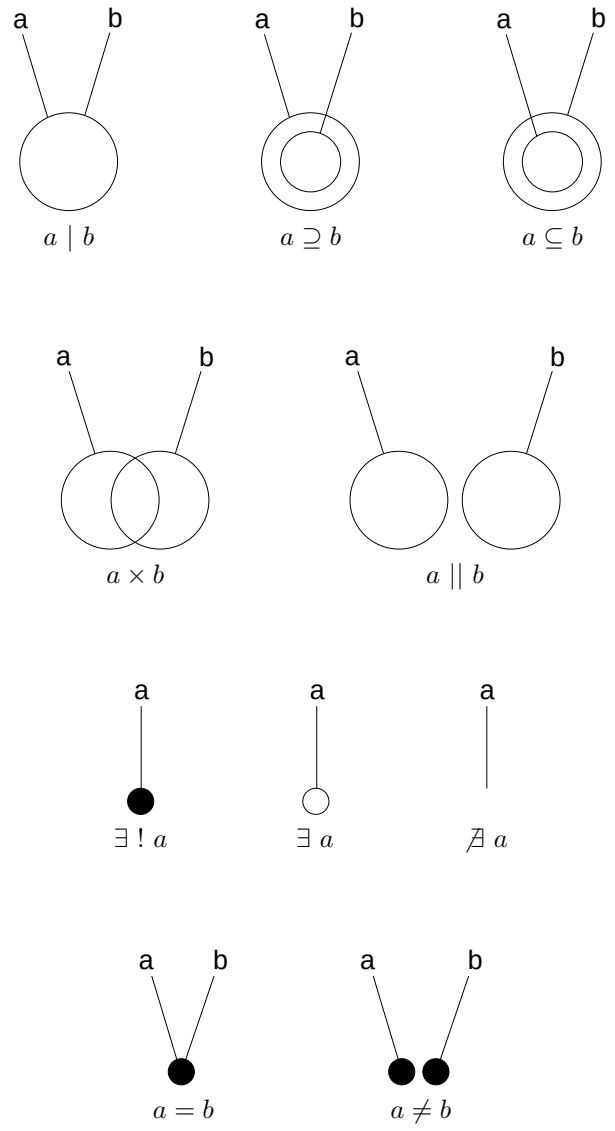


Figura 2.2: *Cercles* d'Euler

2.2. El raonament diagramàtic

El raonament visual i les intuïcions visuals són sovint un dels factors més rellevants en la investigació matemàtica—així com també en la informàtica—, però tot i així, encara hi ha la convicció generalitzada que els mètodes visuals només serveixen com a font d'inspiració i com a il·lustració de demostracions i raonaments, però mai com a arguments vàlids per ells mateixos. Sortosament, en els darrers temps hi ha un interès creixent en l'estudi de la utilització formal de diagrames en el raonament matemàtic, i hi ha resultats que mostren com la utilització de diagrames i/o altres fons d'informació visuals en el raonament matemàtic no implica una pèrdua de formalitat. En aquest capítol analitzarem les diferents propostes de sistemes de raonament diagramàtic que considerem rellevants per a aquesta tesi.

Una motivació d'aquesta tesi és trobar mecanismes d'expressió que ens acostin la sintaxi a la semàntica en el marc de la programació lògica. Aquest objectiu és comú amb la gent que treballa en el món del raonament diagramàtic. Una característica d'un bon diagrama és la possibilitat de transmetre molta informació d'una forma compacta, clara i senzilla. La propietat que fa que això sigui possible és la proximitat o semblança entre el que representen i la sintaxi del diagrama. Aquesta semblança ha estat anomenada de forma diferent per diferents autors: Kulpa [44] ens parla de representacions analògiques i representacions proposicionals. En aquest cas el terme analògic fa referència al paral·lelisme entre la sintaxi i la semàntica de la representació, mentre que el terme proposicional es refereix a representacions on no hi ha una correspondència clara entre aquests dos elements. Barwise i Etchemendy [12] argumenten que un bon diagrama ha de ser isomorf —o com a mínim homomorf— amb allò que vol representar, és a dir, que ha d'existir una correspondència entre la sintaxi i la semàntica que conservi l'estructura. El treball de Shimojima [69, 70] defineix de forma acurada aquesta idea d'homomorfisme, estudiant els avantatges i inconvenients de diversos sistemes de representació per a un domini comú, en termes dels tipus de restricccions que aquests sistemes projecten des de les representacions cap als dominis. Dit això podem veure que en el cas de les representacions textuals de la lògica aquesta correspondència entre la representació i el domini no existeix, o en tot cas està molt lluny de ser un homomorfisme.

L'objectiu d'aquesta secció és fer referència a tot aquell treball sobre raonament diagramàtic rellevant a aquesta tesi i, que en part, l'ha inspirat. Tot seguit, a la secció 2.2.1 veiem els grafs existencials (*existential graphs*), una de les primeres representacions visuals de la lògica de predicats desenvolupada per Charles Peirce. A continuació, a la secció 2.2.2 mostrem sistemes de raonament basats en diagrames d'Euler / Venn i a la secció 2.2.3 veiem els Higraphs, un dels fonaments d'aquest treball. Finalment a la secció 2.2.4 fem referència als sistemes de raonament heterogenis. Per a un estat de l'art complet sobre el raonament diagramàtic remetem el lector a [7, 9, 34, 44].

$P \wedge \neg Q$ $\quad\quad$ $\neg(P \wedge \neg Q)$
$\neg P \vee Q$
$P \Rightarrow Q$

Figura 2.3: Els *existential graphs* de Peirce: *Sistema* α

2.2.1. Grafs existencials

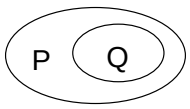
Charles S. Peirce va ser el primer a investigar l'ús d'una representació diagramàtica en la lògica de predicats (N.Houser et al. [57], Roberts [62]). Els seus grafs existencials (*existential graphs*) són tres sistemes de raonament diagramàtic, cadascun amb diferent poder expressiu, que han arribat a generar escola i que el mateix Peirce considera com la seva contribució més important a la lògica:

- α: sistema diagramàtic equivalent a la la lògica proposicional. Hammer [37, 38] fa un apropament modern a aquest sistema demostrant la solidesa i la completesa del seu sistema deductiu.
- β: sistema diagramàtic equivalent al càlcul de predicats o lògica de primer ordre.
- γ: sistema diagramàtic equivalent a la lògica de segon ordre i d'ordre superior.

A les figures 2.3 i 2.4 trobem exemples dels sistemes diagramàtics α i β. En una primera observació sembla que ens trobem davant d'una sintaxi basada en diagrames de Venn i, per tant, que utilitza un enfocament molt similar al nostre, però no és el cas. La sintaxi de Peirce no incorpora cap concepte conjuntista[2] sinó que es basa en el que ell anomena talls, i, al sistema β, punts i línies. Els talls s'utilitzen per a indicar negació i els punts i les línies representen l'existència d'un element.

Noteu que la representació diagramàtica de la implicació en els EG de Peirce és exactament al revés de com seria en una representació conjuntista d'una relació:

- Implicació al sistema α de Peirce



[2]Abans de proposar els grafs existencials Peirce va proposar modificacions als diagrames de Venn (veure secció 2.2.2).

$\exists x \;\; lleig(x)$ $\quad\quad$ $\neg(\exists x \;\; lleig(x))$
$\forall x \;\; \neg lleig(x)$

$\exists x \;\; \neg lleig(x)$ $\quad\quad$ $\neg(\exists x \;\; \neg lleig(x))$
$\forall x \;\; lleig(x)$

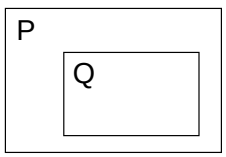
$\exists x \;\; lleig(x) \;\wedge\; bo(x)$ $\quad\quad$ $\neg(\exists x \;\; lleig(x)) \;\wedge\; bo(x)$
$\forall x \;\; \neg lleig(x) \;\vee\; \neg bo(x)$

$\exists x \;\; lleig(x) \;\wedge\; \neg bo(x)$ $\quad\quad$ $\neg(\exists x \;\; lleig(x) \;\wedge\; \neg bo(x))$
$\forall x \;\; \neg lleig(x) \;\vee\; bo(x)$
$\forall x \;\; lleig(x) \;\Rightarrow\; bo(x)$

Figura 2.4: Els grafs existencials de Peirce: *Sistema* β

$$\neg P \vee Q$$
$$P \Rightarrow Q$$

- Implicació en la notació conjuntista proposada en aquesta tesi



$$Q \subseteq P$$

$$\forall x \;\; x \in Q \;\Rightarrow\; x \in P$$

$$Q \Rightarrow P$$

Segons Shin [73] hi ha situacions en les quals els EG poden ser més eficients que altres representacions textuals. Per exemple, amb el sistema α és possible veure intuïtivament l'equivalència lògica: és més eficient veure que dos diagrames resultants constitueixen el mateix diagrama, que trobar una seqüència deductiva entre dues fórmules textuals. L'eficiència dels EG es troba en el fet que amb només dues eines sintàctiques —la inclusió gràfica (també anomenada "talls") i l'adjacència— ens permeten expressar tota la lògica proposicional. No obstant, l'autora també ens fa notar que la lectura dels EG no és senzilla i que les regles d'inferència dels EG no són tan intuïtives com les regles d'inferència de la deducció natural.

Més recentment John Sowa ha desenvolupat els grafs conceptuals, inspirats en els grafs existencials de Peirce (Sowa [75, 76]). Els grafs conceptuals són un sistema diagramàtic per a l'especificació de sistemes i la representació del coneixement. És un formalisme molt utilitzat, però que s'allunya de l'objectiu central d'aquesta tesi: la programació lògica visual.

2.2.2. Sistemes de raonament basats en diagrames d'Euler / Venn

Al Visual Inference Laboratory (Indiana University) sota la direcció del Dr. Jon Barwise es van dissenyar diferents sistemes de raonament diagramàtic basats en les representacions proposades per Euler i Venn, i en la modificació dels diagrames de Venn proposada per Peirce (veure figura 2.5). En tots els casos es proposen sistemes de raonament sòlids i complets, amb regles d'inferència que s'apliquen directament als diagrames.

Shin [71, 72] proposa un sistema de raonament basat en la modificació dels diagrames de Venn proposada per Peirce. Dóna la definició recursiva per a la construcció de diagrames ben formats i en defineix la semàntica des d'un punt de vista de teoria de situacions. A continuació proposa un sistema d'inferència sòlid i complet (finitament) per aquests diagrames. Més tard Hammer [37] modifica aquest sistema utilitzant una semàntica basada en models i demostra un resultat més general de completesa del sistema d'inferència proposat (Hammer i Danner [39]).

En treballar amb diagrames de Venn amb més de 3 conjunts, la construcció i comprensió dels diagrames es complica ja que han d'estar representades totes les possibles interseccions. Per tal d'obtenir un sistema més senzill i escalable, Hammer [37] també proposa un segon sistema diagramàtic basat en els diagrames d'Euler. La diferència fonamental és que no cal que totes les interseccions entre conjunts estiguin presents. A diferència del nostre treball s'entén que dos conjunts disjunts gràficament són disjunts. Swoboda [79, 80] proposa un apropament híbrid, on es combinen propietats dels sistemes basats en diagrames de Venn i dels sistemes basats en diagrames d'Euler.

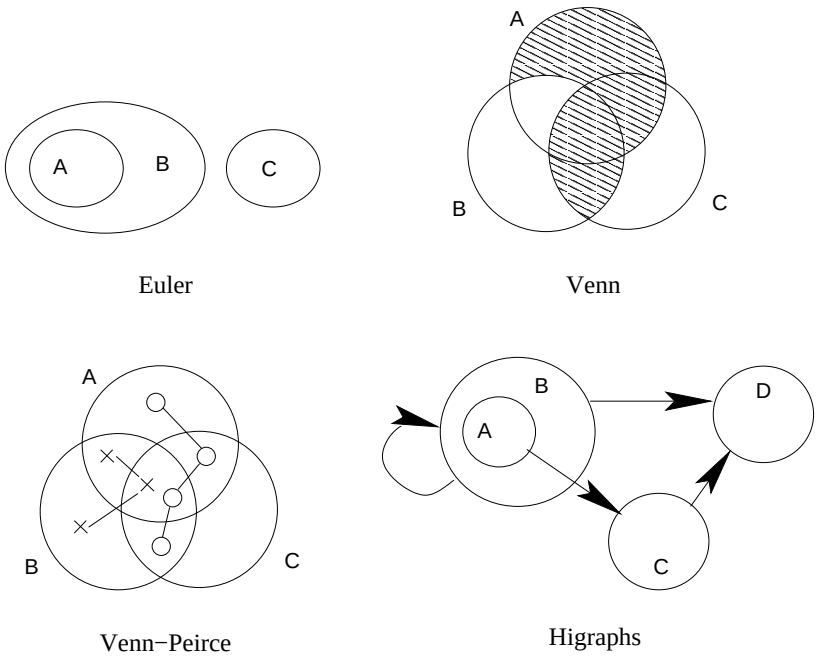


Figura 2.5: Diagrames de Venn, Euler, Venn-Peirce i Higraphs

2.2.3. Higraphs

Els higraphs són un formalisme diagramàtic combinació de grafs i diagrames de Venn (veure figura 2.5) introduït per Harel [41]. És a dir, són diagrames topològics formats per grafs on els nodes estan relacionats entre ells mitjançant la inclusió gràfica. Com veurem en els propers capítols, la inspiració directa de la sintaxi del llenguatge visual proposat en aquesta tesi parteix d'aquest formalisme. La motivació de Harel en dissenyar els higraphs és el fet de poder combinar en un mateix diagrama topològic la relació d'ordre parcial, donada per la inclusió gràfica, amb d'altres relacions que aquests objectes (conjunts) també han de complir en funció del domini d'aplicació tractat.

Els higraphs, igual que els grafs, són un formalisme matemàtic general, independent del domini d'aplicació. Harel ens proposa una aplicació com a diagrames d'estat. A la figura 2.6 veiem el diagrama d'estats corresponent a un rellotge digital.

Hammer [37] construeix un sistema de raonament modern basat en els higraphs. Formalitza la sintaxi donada per Harel, dóna regles d'inferència i defineix una semàntica similar a la donada als cercles d'Euler, però on també es té en compte el significat dels vèrtexs. Finalment demostra que les regles d'inferència són sòlides i completes.

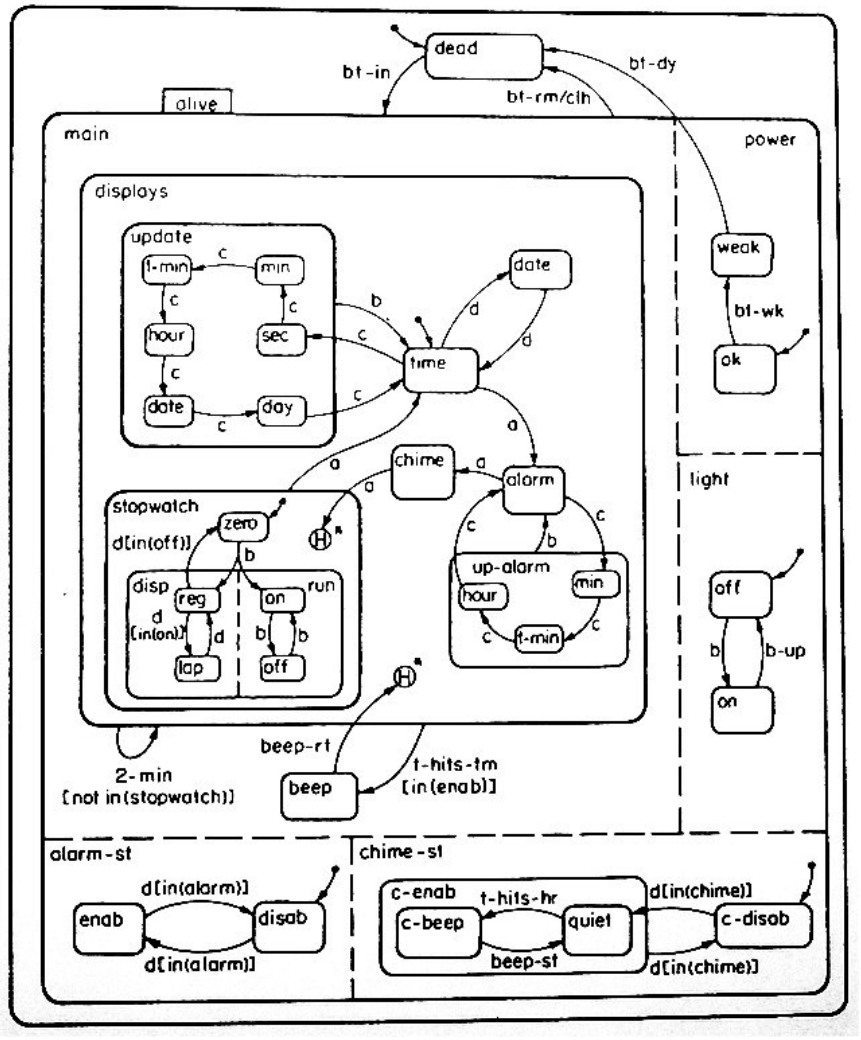


Figura 2.6: Higraphs com a diagrames d'estat: un rellotge digital

2.2.4. Lògiques heterogènies

Una lògica heterogènia és aquella en què s'utilitza més d'un tipus de representació en el procés de raonament. Segons Barwise i Etchemendy [12] buscar representacions *universals* —textuals com la lògica de primer ordre o diagramàtiques— és anar en la direcció equivocada. Consideren que el raonament és intrínsecament heterogeni (o híbrid) i que cal buscar formes de raonament que englobin representacions diagramàtiques, textuals o de la natura que siguin, cadascuna adequada per a la tasca requerida. Hyperproof (Barwise i Etchemendy

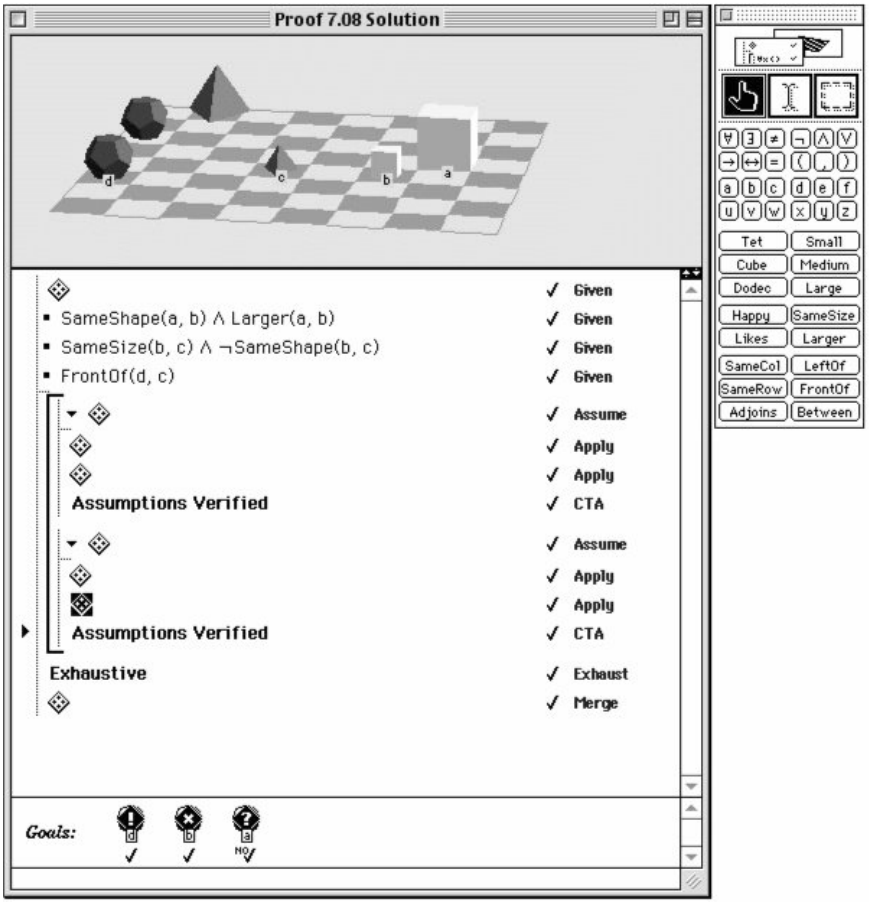


Figura 2.7: Hyperproof

[11, 14]) és un exemple de sistema de raonament heterogeni que a més està implementat per a ser utilitzat en l'ensenyament de la lògica, i que va constituir la base i motivació de gran part del treball realitzat al Visual Inference Laboratory (Indiana University) per Jon Barwise. Hyperproof combina representacions visuals d'un món de blocs amb sentències de lògica de primer ordre que donen informació sobre aquest món. Mitjançant aquest sistema s'obtenen demostracions que són intrínsecament heterogènies. És a dir, extreuen informació de les dues representacions per tal d'arribar al resultat desitjat. A la figura 2.7 veiem una pantalla de Hyperproof. La darrera versió de Hyperproof es troba a [15] en combinació amb altres sistemes de demostració.

Altres sistemes heterogenis són els sistemes de raonament sobre circuits electrònics (Barwise i Hammer [16]) o les combinacions de lògica de primer ordre i taules d'informació (Fisler [31]). Hammer [37] també defineix un sistema

de raonament heterogeni basat en diagrames de Venn i lògica de primer ordre, utilitzant com a vincle entre els dos sistemes les abstraccions de predicats, de forma similar a com es fa en el treball presentat en aquesta tesi.

2.3. La programació visual

Els termes "programació visual" o "llenguatges visuals" engloben un camp molt ampli. La programació visual es pot veure com una nova manera d'enfocar el disseny de llenguatges de programació en general. Podem trobar llenguatges visuals en pràcticament tots els paradigmes tradicionals dels llenguatges de programació: procedurals, declaratius, etc. Trobar una definició exacta del que és un llenguatge visual de programació pot ser difícil si no volem excloure'n cap. De fet, actualment en el món de la informàtica quan s'utilitza la paraula *visual* hom ho associa als entorns de programació de Microsoft Visual Studio. Aquests llenguatges o entorns de programació no són programació visual tal i com s'entén en aquest treball. Creiem que s'haurien de catalogar com a llenguatges de programació amb entorn visual. També hauríem de descartar tot el que són llenguatges per a processar informació visual.

Les dues darreres dècades del segle passat han estat testimonis d'un augment sense precedents de la capacitat de càlcul i de representació gràfica dels ordinadors. Això ha permès que es pugui dedicar molt més esforç d'investigació i experimentació en el camp dels llenguatges visuals. Alguns dels noms més rellevants de la comunitat científica dedicada als llenguatges visuals han treballat per tal d'identificar-ne les seves característiques i tractar de classificar-los (Burnett i Baker [18], Chang [24]). Una possible classificació no exclusiva proposada a [18] és:

1. Llenguatges visuals purs
2. Sistemes o llenguatges híbrids (textuals i visuals)
3. Sistemes de programació per exemple
4. Sistemes orientats a restriccions
5. Sistemes basats en formularis

El treball presentat en aquesta tesi es centra en la primera categoria de llenguatges, corresponent als llenguatges que depenen de tècniques visuals durant tot el procés de programació. Els llenguatges completament visuals es compilen o interpreten directament de la representació visual sense passar per cap representació intermèdia textual. A aquesta categoria de llenguatges hi podríem aplicar moltes altres subclassificacions en funció del tipus d'elements diagramàtics utilitzats (llenguatges icònics, topològics, etc.), del paradigma (orientats a objecte, funcionals, imperatius, de programació lògica, etc.) o d'altres propietats del llenguatge. Un estat de l'art detallat de tots els apropaments a la programació visual queda fora de l'àmbit d'aquesta tesi. En els propers apartats examinarem alguns dels llenguatges visuals existents més rellevants al treball portat a terme en aquesta tesi. Per aprofundir més en els llenguatges i la programació visuals remetem el lector a [20, 25, 35, 36, 52, 74].

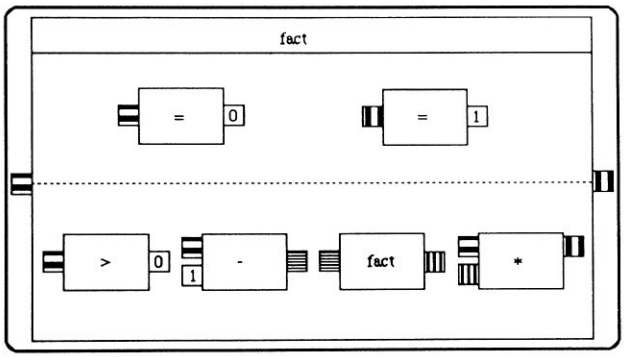


Figura 2.8: VLP: càlcul del factorial

2.3.1. Llenguatges visuals de programació lògica

Hi ha diferents apropaments per dissenyar un llenguatge visual de programació lògica. Cap dels que nosaltres coneixem utilitza la metàfora conjuntista (representar les relacions mitjançant conjunts) com a base del disseny del llenguatge. Un apropament habitual és utilitzar arbres AND-OR, que representen molt bé el procés de resolució SLD habitual en la programació lògica. Pau i Olason [59] proposen un apropament visual (sistema VPP) al Prolog per tal de visualitzar-ne l'execució. El sistema proposat per Senay i Lazzeri [67] també s'engloba dins del mateix grup. Aquest tipus de tècniques són molt útils per a mostrar gràficament l'execució d'un programa Prolog però no serveixen per a representar (ni definir) el programa estàtic. Al capítol 5 veurem com utilitzem un arbre AND-OR per tal de seguir l'execució del llenguatge proposat en aquesta tesi.

VLP (Ladret i Rueher [45]) és un llenguatge de programació lògica basat en diagrames de flux de dades. Clàusules i literals es representen com a capses. L'arranjament horitzontal de caixes equival a la conjunció, mentre que la disposició vertical de caixes equival a la disjunció. La inclusió espacial de capses s'utilitza per a la definició de predicats. VLP utilitza textures per tal d'indicar les variables compartides per diferents predicats (capses), cosa que al nostre entendre dificulta la tasca de localització de les diferents ocurrències d'una mateixa variable. A la figura 2.8 trobem el predicat factorial expressat en VLP.

CUBE (Najork [55, 56]) és el primer llenguatge visual que utilitza una sintaxi tridimensional i el primer que té un sistema de tipus polimorf. És un llenguatge que comparteix fonaments amb VLP, concretament la representació de predicats com capses (cubs), la utilització de diferents dimensions per a denotar la disjunció i la conjunció (a CUBE diferents plans impliquen disjunció), i la utilització de la inclusió de capses (cubs) per a definir nous predicats. Com a mancances cal destacar que CUBE no té ni una semàntica denotacional ni una semàntica operacional definides directament a partir de la sintaxi visual, és a dir, no és un

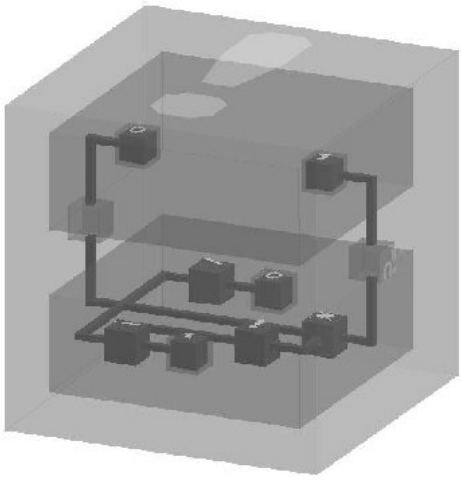


Figura 2.9: Cube: càlcul del factorial

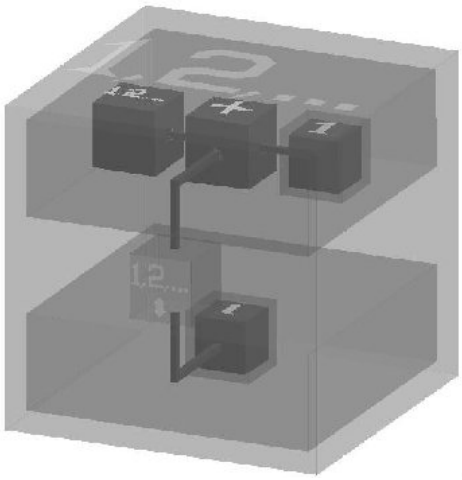


Figura 2.10: Cube: càlcul dels nombres naturals

llenguatge completament visual. Totes les definicions i computacions es realitzen a partir d'un llenguatge textual intermedi. D'altra banda considerem molt interessant l'ús pioner d'una sintaxi tridimensional, almenys des d'un punt de vista acadèmic. Encara queda, però, molta feina d'anàlisi i valoració dels seus avantatges enfront dels possibles (i probables) inconvenients.

SPARCL (Spratt [77], Spratt i Ambler [78]) és un llenguatge visual de programació lògica basat en conjunts. Tot i aquesta descripció el seu apropament

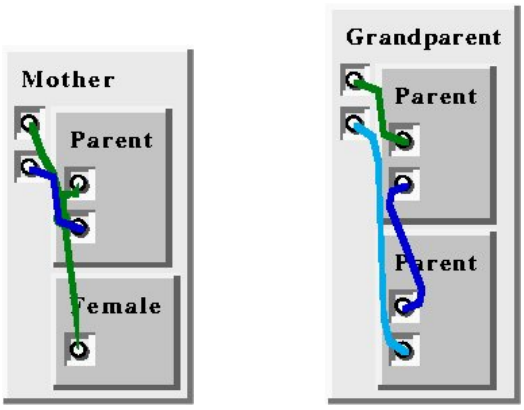


Figura 2.11: SPARCL: definició dels predicats *Mother* i *Grandparent*

és molt diferent al nostre. A SPARCL els conjunts són el constructor bàsic per a la manipulació de dades, de forma similar a les llistes i els termes al Prolog. SPARCL utilitza la notació visual conjuntista al representar aquests conjunts, però no en la resta de constructors: combina elements diagramàtics amb propietats topològiques amb altres elements diagramàtics més similars a VPP o CUBE. Mentre que en el primer cas la inclusió gràfica implica pertinença als conjunts—només s'utilitza la inclusió gràfica entre elements i conjunts, mai entre conjunts— en la resta la inclusió s'utilitza en la definició de predicats.

2.3.2. Llenguatges visuals funcionals

La major part d'investigació en llenguatges visuals funcionals s'ha dirigit a la visualització de LISP. Les estructures de dades de LISP (llistes) són fàcils de representar com a diagrames en arbre. Una excepció la trobem en el llenguatge funcional proposat per Cardelli [22], inspirat en llenguatges funcionals moderns com el ML. Aquesta proposta utilitza moltes idees que més endavant apareixen a altres llenguatges visuals declaratius, com la utilització de la juxtaposició i la inclusió gràfica com a elements constructius i d'abstracció procedural del llenguatge respectivament. A la figura 2.12 trobem la funció *split*, que donada una llista la parteix en un punt donat, expressada de forma visual mitjançant aquesta sintaxi.

Més recentment trobem VEX (Citrin et al. [28]), una representació gràfica del λ-càlcul que neix com a component funcional de VIPR (Citrin et al. [26, 27]), un llenguatge visual imperatiu orientat a objectes i similar sintàcticament al VEX. Segons els autors mitjançant aquesta sintaxi s'aconsegueix simplificar la representació de les expressions del λ-càlcul així com una millor codificació de les regles de transformació. L'aplicació de funcions es representa mitjançant adjacència, amb fletxes que van de les funcions aplicades als seus arguments. L'ús d'una sintaxi visual permet eliminar la necessitat de donar noms a les variables i

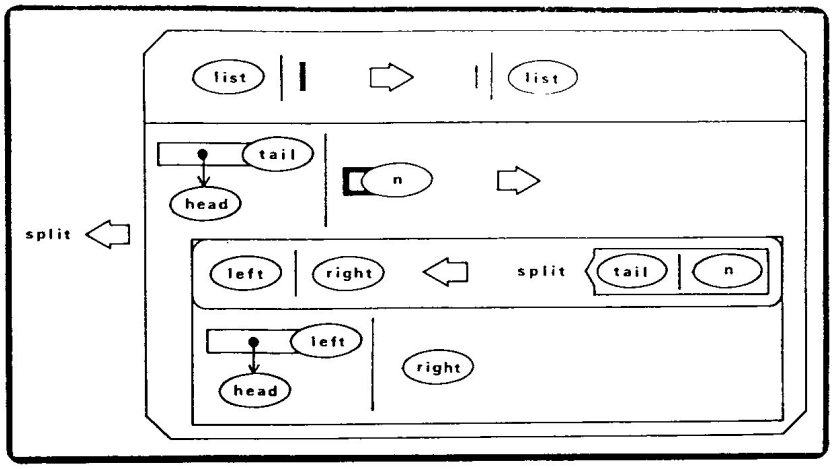


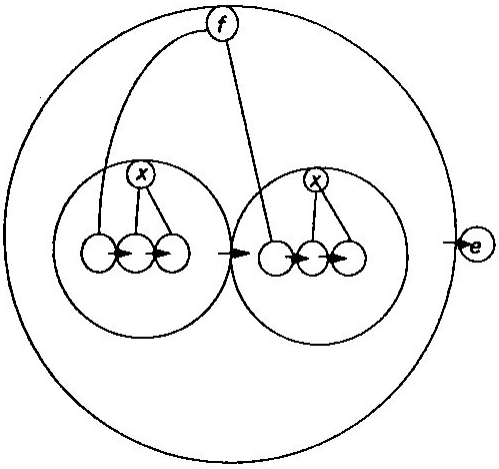
Figura 2.12: Llenguatge visual funcional proposat per Cardelli

d'aquesta forma simplifica l'abstracció i el procediment de substitució. Les regles de transformació s'expressen directament en la sintaxi visual i segons els autors són més senzilles que les regles del λ-càlcul textual. VEX incorpora mecanismes per tal de poder visualitzar el procés dinàmic d'avaluació d'expressions típic del λ-càlcul .

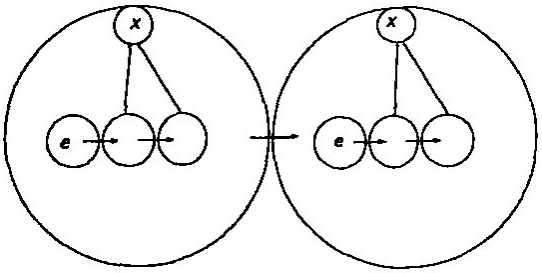
2.3.3. Altres llenguatges visuals

Pictorial Janus (Kahn i Saraswat [43]) és un llenguatge de programació lògica concurrent amb restriccions. Totes les definicions, clàusules i consultes es representen com a cercles (o contorns tancats) de forma molt similar a VEX i VIPR. Aquest llenguatge disposa d'una semàntica operacional completament visual basada en re-escriptura de diagrames, de forma semblant al mecanisme utilitzat a VEX i a la semàntica operacional que nosaltres utilitzem en aquesta tesi (capítol 5). A la figura 2.14 trobem un exemple que mostra la implementació d'una cua en Pictorial Janus.

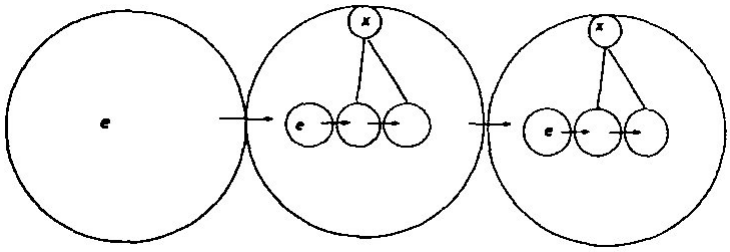
També trobem exemples de llenguatges visuals aplicats al món de les bases de dades (Cruz i Leveille [29]), però cap que faci referència a la representació visual d'esquemes de bases deductives.



(Ye)



$(\lambda x.e(xx))(\lambda x.e(xx))$



$e(\lambda x.e(xx))(\lambda x.e(xx)) = e(Ye)$

Figura 2.13: VEX: aplicació del combinador Y

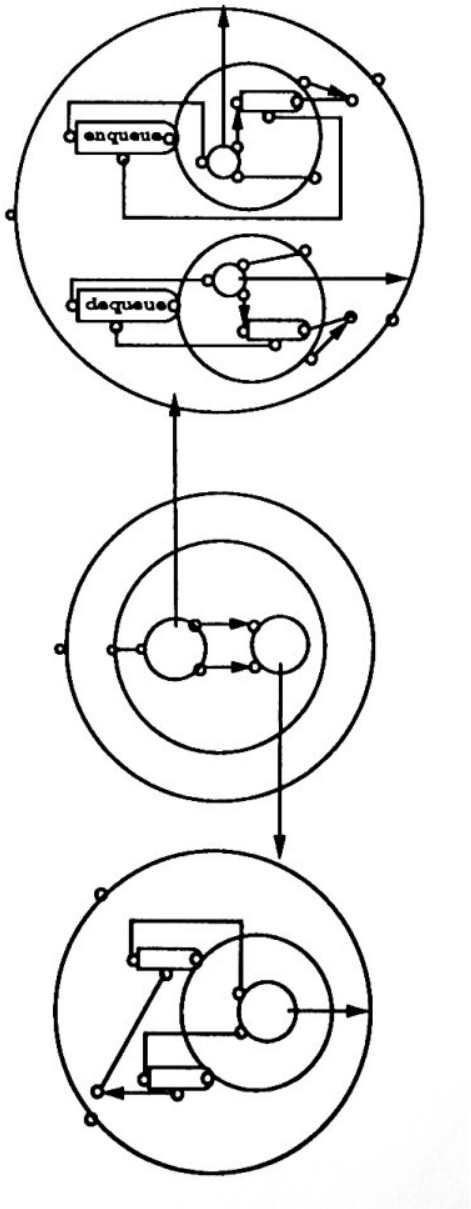


Figura 2.14: Pictorial Janus: implementació d'una cua

Capítol 3

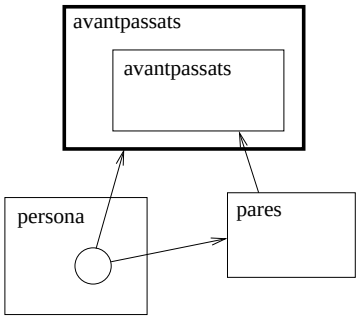
El nostre apropament

En aquest capítol mostrem les motivacions que ens van portar a desenvolupar un llenguatge visual de programació lògica, així com el treball previ en el qual ens hem basat. A la secció 3.1 veiem GraSp, un llenguatge visual per a l'especificació preliminar. A continuació, a la secció 3.2 aprofundim en el desenvolupament d'un llenguatge visual per a la lògica de primer ordre que ens serveix per a introduir el nostre apropament a la representació visual topològica de la lògica utilitzada en aquesta tesi. Aquesta sintaxi visual per a la lògica de primer ordre és la que ens serveix de base del llenguatge de programació lògica visual que introduïm en el proper capítol.

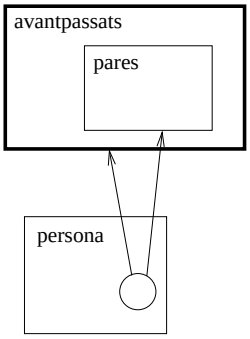
3.1. GraSp: Un llenguatge gràfic per a l'especificació preliminar

El llenguatge GraSp (Agustí et al. [4, 6]) va néixer amb l'objectiu de servir de pont entre l'especificació informal d'un sistema i la primera especificació formal del mateix en un llenguatge basat en la lògica (p.e. un llenguatge de programació lògica com ara el Prolog). L'objectiu era adreçar els problemes que hi ha en l'enginyeria dels requeriments, és a dir, el procés de crear especificacions completes i adequades a partir de descripcions informals dels requeriments d'un sistema. Els llenguatges de programació lògica es poden utilitzar com a llenguatges d'especificació, però són encara excessivament difícils d'utilitzar i d'interpretar per neòfits, alhora que són massa distants conceptualment del domini d'aplicació. GraSp és un intent de dissenyar un llenguatge de més alt nivell, independent del domini d'aplicació, que tot i ser formal sigui més senzill d'utilitzar i faciliti el pas de requeriments informals a especificacions formals.

La base formal de GraSp és COR (*Calculus of Refinements*), un llenguatge d'especificació formal basat en la relació d'inclusió entre conjunts. L'estudi computacional de la relació d'inclusió ha donat lloc a dues tesis doctorals. En la primera (Jordi Levy [42]) es donen les bases de la semàntica, tan matemàtica com operacional, de les relacions d'inclusió utilitzant la re-escriptura. En la



$$avantpassats(pares(X)) \subseteq avantpassats(X) \Leftarrow X \subseteq persona$$



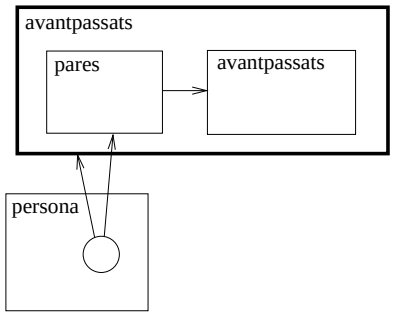
$$pares(X) \subseteq avantpassats(X) \Leftarrow X \subseteq persona$$

Figura 3.1: Recursió en GraSp (1)

segona (Schorlemmer [65]) es generalitza l'estudi de les relacions d'inclusió a un conjunt més ampli de relacions binàries que anomenem relacions especials. Aquesta tesi ha tingut sempre present aquesta base tècnica de les relacions en donar semàntica als diferents llenguatges proposats. En concret la definició d'una semàntica operacional (veure capítol 5) ha estat directament inspirada per aquest treball previ (Agustí et al. [4, 5], Schorlemmer et al. [66]).

La sintaxi visual de GraSp es basa en una personalització dels *higraphs* (Harel [41]). Els higraphs, tal com hem comentat al capítol anterior, són un formalisme topològic que combina diagrames de Venn (o cercles d'Euler) amb grafs (veure figures 2.5 i 2.6) i que cal no confondre amb els hipergrafs.

El llenguatge COR es basa en conjunts i funcions entre conjunts, funcions que tenen com argument un conjunt i retornen un conjunt. Tots els elements de GraSp són conjunts i les fletxes s'utilitzen per a representar les dependències funcionals entre aquests conjunts. A la figura 3.1 trobem un exemple d'utilització de GraSp per a modelitzar un concepte recursiu: els *avantpassats* d'una *persona*.



$$pares(X) \subseteq avantpassats(X) \Leftarrow X \subseteq persona$$

$$avantpassats(pares(X)) \subseteq avantpassats(X) \Leftarrow X \subseteq persona$$

Figura 3.2: Recursió en GraSp (2)

Les variables es representen com a cercles, mentre que els conjunts de persones i d'avantpassats d'una persona es representen mitjançant rectangles etiquetats. Trobem dos diagrames: un primer per a representar el cas recursiu i un altre per al cas base. La intuïció que hi ha darrera aquests dos diagrames és: 1) els avantpassats dels pares d'un conjunt de persones són avantpassats d'aquest conjunt de persones, i 2) els pares d'un conjunt de persones són avantpassats d'aquest conjunt de persones. La seva formalització es veu clarament mitjançant la representació textual de les inclusions gràfiques dels diagrames. Mitjançant la notació diagramàtica és fàcil unificar els dos casos i presentar-los en un sol diagrama. A la figura 3.2 trobem un diagrama equivalent als dos anteriors.

A GraSp ja apareixen algunes de les convencions visuals que utilitzarem en treballs posteriors, i concretament en el llenguatge visual de programació lògica proposat en aquesta tesi:

- La *implicació* en un diagrama s'expressa mitjançant la inclusió gràfica. El conjunt que volem definir, és a dir la conclusió de clàusula, apareix distingit de la resta de conjunts. Mitjançant la inclusió n'especifiquem alguns dels seus elements.

- Del diagrama s'extreu només *la informació inclusional positiva*, és a dir, el fet que dos conjunts (capses o cercles) no estiguin inclosos un dins l'altre no aporta cap informació. En aquest aspecte el nostre sistema visual difereix dels diagrames de Venn / Euler i de les seves formalitzacions com a sistemes de raonament (Hammer [37], Shin [71]).

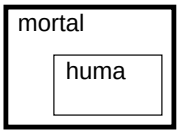
3.2. Un llenguatge visual per la lògica

Basat en les idees de visualització de GraSp a [3, 61] introduïm una sintaxi visual per a la lògica de primer ordre. Igual que a GraSp, la sintaxi està inspirada en els higraphs, si bé amb vàries diferències. Aquest treball ens servirà per a introduir els principis de la representació visual que utilitzarem en el proper capítol en la definició del llenguatge visual de programació lògica.

3.2.1. Els elements bàsics de la sintaxi visual

Un objectiu bàsic en el disseny de la sintaxi visual ha estat mantenir la proximitat de la sintaxi i la semàntica. Per aconseguir-ho, la combinació de grafs i diagrames topològics ens permeten utilitzar dues metàfores visuals bàsiques en tot el treball d'aquesta tesi:

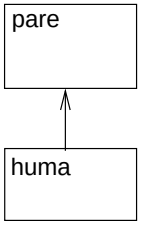
1. Representem els predicats (o relacions) mitjançant conjunts gràfics (rectangles etiquetats) que denoten les abstraccions conjuntistes dels predicats. El significat pretès del predicat és representat mitjançant la inclusió gràfica de conjunts (inclusió de rectangles), que denoten la inclusió de les abstraccions conjuntistes respectives. Per exemple, el següent diagrama descriu parcialment el predicat mortal tot afirmant que tots els humans són mortals:



   El diagrama mostra la inclusió dels conjunts abstracció corresponents a cada capsa etiquetada:

$$\{x \,|\, huma(x)\} \subseteq \{y \,|\, mortal(y)\}$$

2. Representem l'aplicació de predicats mitjançant fletxes, que juntament amb les capses i els altres elements diagramàtics del llenguatge formen grafs acíclics dirigits (DAGs). Aquests DAGs corresponen als termes estructurats de la lògica. Per exemple, el següent DAG:



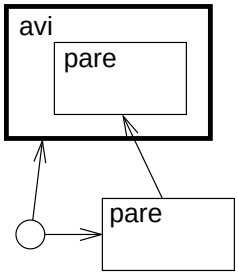
que es llegeix com "els pares[1] d'algun humà" i denota el següent conjunt abstracció:

$$\{y \mid \exists x\ pare(x, y) \wedge huma(x)\}$$

on els pares de y estan representats visualment pel contingut del rectangle *pare*.

En informàtica els grafs s'han utilitzat sovint com a medi de visualització de termes. Permeten compartir subtermes i donen lloc a representacions més compactes, i que a tots ens resulten familiars.

Aquestes dues metàfores visuals es poden combinar, com per exemple en el següent diagrama que defineix el predicat *avi*:



Aquest diagrama formalitza que els pares d'algú (un element variable es representa com a cercle) són els seus avis, i denota la següent inclusió:

$$\{z \mid \exists y\ pare(x, y) \wedge pare(y, z)\} \subseteq \{w \mid avi(x, w)\}$$

Noteu que aquesta sintaxi visual està basada en un formalisme diagramàtic topològic. En aquests diagrames cada capsa etiquetada amb un símbol de predicat representa un conjunt d'elements que satisfà el predicat. No és una sintaxi icònica de flux de dades amb operacions de transformació de dades, com és habitual en molts llenguatges visuals.

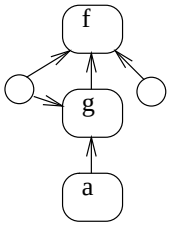
Tal com ja hem dit, el punt clau del nostre apropament és representar visualment els predicats com els conjunts d'elements que els satisfan, és a dir, la seva abstracció conjuntista o extensió del predicat. Llavors la inclusió d'aquests conjunts equival a la implicació lògica entre les fórmules dels conjunts abstracció, permetent així una representació visual elegant i intuïtiva de la implicació. Per continuar amb l'exemple anterior, el diagrama que defineix el predicat *avi* equival a la següent implicació lògica:

$$avi(x, z) \ \Leftarrow\ pare(x, y)\ \wedge\ pare(y, z)$$

Una diferència important respecte GraSp és que en aquesta sintaxi diferenciem entre elements i conjunts, mentre que a GraSp tots els elements diagramàtics representaven conjunts. En aquesta sintaxi visual hi ha termes conjunt i termes

[1] En aquesta tesi i per facilitar-ne la lectura utilitzarem la paraula *pare* per denotar indistintament pare o mare, és a dir progenitor.

element, que es combinen per a formar els diagrames. Tots els exemples que hem vist fins ara són combinacions de termes conjunt, excepte la variable del diagrama anterior que és un terme element. Per a representar constants i termes estructurats utilitzarem capses arrodonides etiquetades organitzades com a DAGs. Per exemple, si volem representar $f(x, g(x, a), y)$ de forma visual ho farem mitjançant el següent graf:

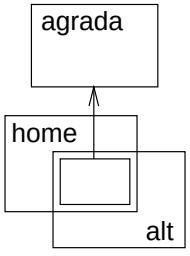


3.2.2. La composició de predicats

Anem ara a considerar amb més detall com funciona la composició de predicats dels termes conjunt. De fet, hi ha dos tipus de composició:

- composició existencial
- composició universal

En els exemples vistos fins ara només s'ha utilitzat la composició existencial, que es representa mitjançant una fletxa simple. El següent terme conjunt ens mostra un exemple de composició existencial:



Aquest terme denota el conjunt de coses que agraden a algun home alt. S'anomena composició existencial perquè ens interessa agrupar els elements relacionats amb *algun* element de la intersecció dels conjunts *home* i *alt*. La fórmula del conjunt representat per aquest terme és:

$$\{y \mid \exists x \; agrada(x, y) \; \wedge \; home(x) \; \wedge \; alt(x)\}$$

La composició universal és aquella on volem els elements relacionats amb *tots* els elements del conjunt argument. Per exemple, si volem el conjunt de coses que agraden a *tots* els homes alts utilitzarem la doble fletxa per indicar la composició universal i obtindrem el següent terme visual de conjunt:

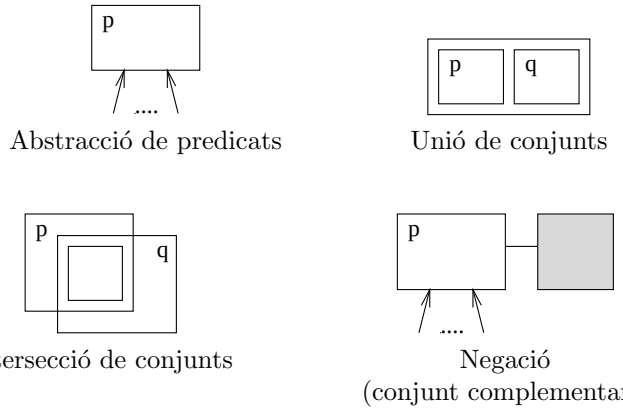
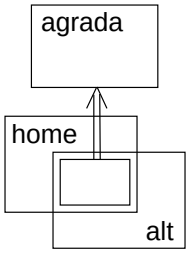


Figura 3.3: Diferents constructors de termes visuals conjunt



La fórmula del conjunt representat per aquest terme visual és:

$$\{x \,|\, \forall y \; (\, agrada(y, x) \;\Leftarrow\; home(y) \;\wedge\; alt(x)\,)\}$$

Noteu que en aquests dos termes hem introduït un element diagramàtic nou: la intersecció. En aquesta sintaxi visual es permeten diferents constructors de termes conjunts a més de l'abstracció de predicats, com la intersecció, la unió i la negació. A la figura 3.3 veiem exemples d'aquests constructors.

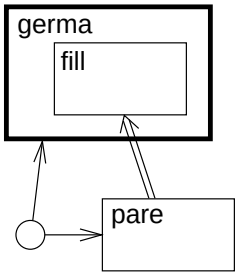
3.2.3. Diagrames

Les unitats bàsiques del nostre llenguatge visual —equivalent a les fórmules al llenguatge textual de la lògica— són els diagrames. Un diagrama és la unitat de descripció completa més petita, i es composa de diferents termes visuals (termes conjunt i termes element) relacionats entre ells mitjançant la inclusió gràfica. Un exemple el trobem en la definició del predicat *avi* a la secció 3.2.1.

L'objectiu dels diagrames és definir predicats. A cada diagrama hi haurà un terme conjunt que distingirem dibuixant el rectangle en negreta i anomenarem terme objectiu. Aquest terme correspon al predicat que volem definir, és a dir, el predicat objectiu del diagrama, i ho fem indicant quins són els elements que pertanyen al seu conjunt abstracció a través de la inclusió gràfica. Vist d'una

altra forma, un diagrama és una inclusió visual condicional on la conclusió és la inclusió principal i les altres inclusions són les condicions. A [3] es formalitza la sintaxi i la semàntica d'aquesta sintaxi visual per a la lògica de primer ordre.

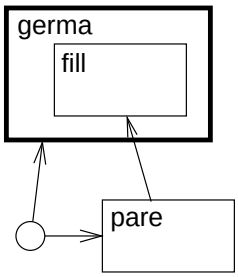
Utilitzant aquests diagrames podem representar fórmules més complexes que les clàusules de Horn. Suposem que volem definir el predicat *germa* de manera que representi els germans dels mateixos pares (el mateix pare i la mateixa mare alhora). Utilitzarem la composició universal obtenint el següent diagrama:



que es llegeix com: "els fills d'ambdós pares d'algú són els seus germans."[2] Aquest diagrama equival a la següent fórmula en lògica de primer ordre (que no es pot expressar com a clàusula de Horn):

$$\forall x, y\ (brother(x, y)\ \Leftarrow\ (\forall z\ son(z, y)\ \Leftarrow\ parent(x, z)))$$

També podem definir *germa* de forma menys restringida, incloent com a germans aquells que comparteixen només un dels pares:



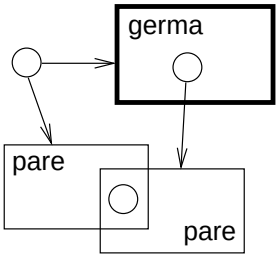
i es llegeix com: "els germans d'algú són els fills d'un dels pares". Mitjançant l'ús no restringit d'inclusions i de la composició de predicats es pot cobrir gran part de la lògica de primer ordre, encara que, com veurem més endavant, el nostre objectiu en aquesta tesi és ajustar-nos a la potència expressiva de les clàusules de Horn.

3.2.4. Flexibilitat i riquesa de la notació

Tot seguit explorarem altres possibilitats expressives del llenguatge visual presentat. Els diagrames de la secció anterior que definien el predicat *germa*

[2] Noteu que en aquestes definicions de germà hom és germà de si mateix, però no hem volgut complicar la definició.

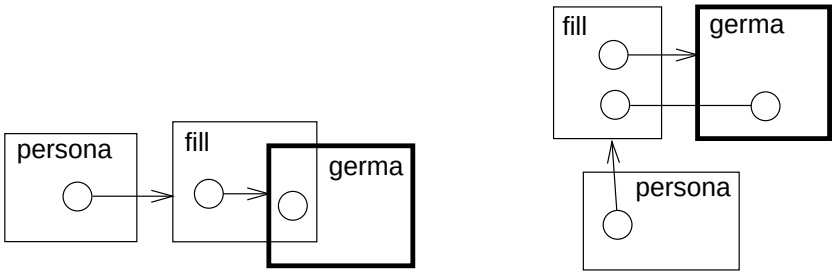
feien èmfasi en l'ús dels predicats com a funcions no-deterministes, i utilitzaven la composició per tal d'obtenir diagrames molt compactes. No obstant, també es possible dibuixar diagrames en un estil més relacional. El predicat *germa* també el podem definir de la següent forma:



que es pot llegir com: "dos persones són germans si comparteixen algun pare". Aquest diagrama equival a la fórmula:

$$germa(x, y) \Leftarrow pare(x, z) \wedge pare(y, z)$$

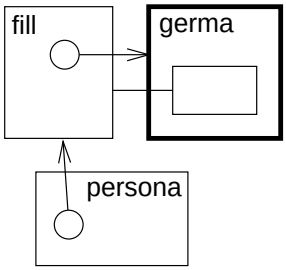
Aquest exemple ens permet mostrar com aquest llenguatge visual ofereix la possibilitat d'acomodar diferents estils de descripció: funcional, relacional o una combinació dels dos. En general aquesta sintaxi visual és molt flexible i permet expressar un mateix concepte de formes molt diferents. És important poder organitzar les capses i cercles del diagrama de forma que la intuïció que es vol transmetre sigui més fàcil de captar. Això és el que s'anomena notació secundària (Petre [60]), o elements sintàctics que no varien el significat formal de l'expressió però ajuden a comprendre-la. La notació secundària ha de permetre al lector del diagrama localitzar la informació rellevant, i és un element molt important d'un llenguatge de programació[3]. Per exemple, si examinem aquests dos diagrames:



veurem que són formalment equivalents, però que d'entrada és més senzill captar el significat en el de la dreta. En el diagrama de l'esquerra el solapament de dues caixes que comparteixen una variable distreu l'atenció mentre que, en el

[3]La notació secundària no és exclusiva dels llenguatges visuals. Als llenguatges textuals s'aconsegueix principalment mitjançant la indentació i la distribució del text de forma que aquest sigui més entenedor.

diagrama de la dreta, el mecanisme d'ubicar una mateixa variable a dos llocs del diagrama unint-la mitjançant una línia ens dóna informació secundària sobre el que es vol transmetre: "els fills del mateix pare són germans". Aquest mecanisme també el podem aplicar a un conjunt i obtindrem el següent diagrama, també equivalent:



Ara hem vist la flexibilitat i la potència expressiva d'aquesta sintaxi visual per a la lògica de primer ordre. L'objectiu d'aquesta tesi és aconseguir un llenguatge visual similar a aquest apropament topològic a la lògica de predicats, però que restringeixi la seva potència expressiva a la del llenguatge de les clàusules de Horn per tal d'heretar-ne les seves propietats en el moment de definir la semàntica operacional.

En els propers capítols definirem la sintaxi i la semàntica d'aquest llenguatge i en donarem una semàntica operacional. Per mantenir el poder expressiu dels llenguatge visual resultant dins les clàusules de Horn, prescindirem de la composició de predicats i restringirem el tipus d'inclusions que es poden donar en un diagrama. A més, limitarem el nombre de constructors utilitzats per tal de simplificar la semàntica operacional visual.

Part II

El llenguatge visual de programació lògica

[...] the intricate nature of a variety of computer-related systems and situations can, and in our opinion should, be represented by *visual formalisms*: visual, because they are to be generated, comprehended, and communicated by humans; and formal, because they are to be manipulated, maintained, and analyzed by computers.

David Harel, *On Visual Formalisms*, 1988. ([41])

Capítol 4

El llenguatge visual de programació lògica: sintaxi i semàntica

L'objectiu central d'aquesta tesi és proposar un llenguatge de programació visual lògica on els conceptes clau de la programació lògica —predicats, implicació, etc.— quedin expressats d'una forma clara i intuïtiva a través de la sintaxi visual. En aquest capítol en definim formalment la sintaxi i la semàntica basant-nos en les idees introduïdes al capítol 3, però reduïm el nombre de constructors utilitzats i limitem el poder expressiu del llenguatge per tal que sigui equivalent a les clàusules de Horn. El llenguatge visual proposat és un llenguatge completament visual, és a dir, totes les definicions de la sintaxi, la semàntica denotacional i la semàntica operacional es porten a terme directament sobre la sintaxi visual.

El capítol està organitzat de la següent forma: a la secció 4.1 definim la sintaxi del llenguatge visual de programació lògica i a la secció 4.2 tractem la semàntica denotacional. Finalment a la secció 4.3 mostrem de forma intuïtiva que la potència expressiva del llenguatge visual proposat és equivalent al llenguatge de les clàusules de Horn.

4.1. Sintaxi

Els elements bàsics del llenguatge visual són rectangles, cercles, fletxes, línies, símbols (textuals) de funció i símbols (textuals) de predicats. Els cercles són de mida fixa mentre que els rectangles poden ser de diferents mides, però sempre més grans que els cercles. Tots aquests elements es combinen seguint les normes sintàctiques del llenguatge i formen termes visuals i predicats visuals. Els primers corresponen a elements i els segons a conjunts d'elements. Aquests termes i predicats es combinen mitjançant la relació d'inclusió gràfica i a partir d'ells s'obtenen literals, que són els elements que constitueixen els diagrames,

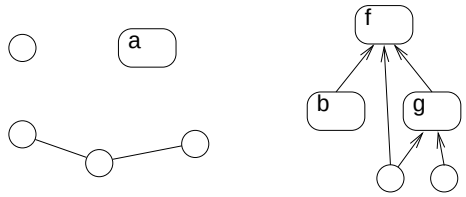


Figura 4.1: Exemples de termes visuals

que venen a ser en el llenguatge visual proposat l'equivalent a les fórmules dels llenguatges textuals.

4.1.1. Termes Visuals

Un terme visual és un graf acíclic dirigit (DAG) construït mitjançant cercles, línies i fletxes. Els cercles sense cap símbol són variables. Això és una característica molt important del llenguatge: no és necessari donar nom a les variables. La co-referència de variables es denota mitjançant seqüències de cercles units mitjançant línies. Aquesta facilitat també l'anomenem compartició de termes visuals.

Definició. Un **terme visual** ben format és:

1. un cercle.
2. una seqüència de cercles units mitjançant línies.
3. un cercle amb un símbol de constant a dins (per facilitat dibuixarem els cercles com a caixes arrodonides adaptant-los així a la llargada del símbol que els identifica).
4. un terme visual compost (DAG) construït mitjançant:
   - un cercle
   - un símbol de funció d'aritat m situat dins del cercle
   - n termes visuals t_i $(n \geq 1)$
   - m fletxes $(m \geq n)$ que opcionalment es poden etiquetar i cadascuna va de t_i al cercle, de forma que no hi han cicles i de cada t_i surt com a mínim una fletxa.
5. res més no és un terme visual ben format.

□

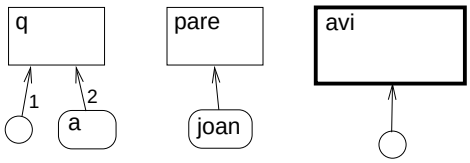


Figura 4.2: Exemples de predicats visuals

4.1.2. Predicats Visuals

Els predicats es representen com a rectangles etiquetats, és a dir, amb un símbol que els identifica. La interpretació intuïtiva és que els rectangles representen visualment el conjunt d'elements que satisfà aquests predicats, és a dir, el que matemàticament es coneix com a *abstracció d'un predicat*. Això funciona per als predicats unaris. Quan el predicat té dos o més arguments, un d'ells es selecciona (per convenció sempre és el darrer) com a rang del conjunt, mentre que la resta són representats explícitament mitjançant fletxes. Donat que la representació visual no determina l'ordre entre els arguments, les fletxes es poden etiquetar per tal d'identificar els arguments. A la secció 4.2 aportem una definició precisa de la semàntica d'un predicat visual. Tot seguit definim la sintaxi d'un predicat visual:

Definició. Un **predicat visual** ben format es composa de:

- un rectangle dibuixat amb línies primes o gruixudes indistintament.
- un símbol de predicat d'aritat m ($m \geq 1$) situat en una de les cantonades del rectangle.
- n termes visuals t_i ($n \geq 0$).
- $m-1$ fletxes ($m-1 \geq n$) opcionalment etiquetades, cadascuna anant de un t_i al rectangle, de forma que de cada t_i surti com a mínim una fletxa.

□

A la Figura 4.2 hi podem trobar exemples de predicats visuals. El primer predicat visual representa el predicat q d'aritat 3. Intuïtivament, el rectangle representa el conjunt $\{w|q(x, a, w)\}$. Els altres dos predicats visuals representen els *pares* d'en *Joan* i el conjunt d'*avis* d'*algú*.

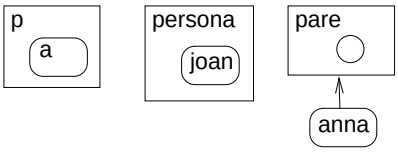


Figura 4.3: Exemples de literals visuals de condició

4.1.3. Literals Visuals

En aquesta secció definirem els elements, que tot i no tenir encara entitat sintàctica i semàntica per si mateixos són els elements bàsics que s'utilitzen per a construir els diagrames. Aquests elements, seguint l'analogia del cas textual, s'anomenen literals visuals.

Primer de tot definirem la relació d'inclusió entre un terme visual i un predicat visual, i entre predicats visuals. Aquesta relació la utilitzarem després en la definició de literal visual.

Definició. Sigui VP el conjunt de predicats visuals i VT el conjunt de termes visuals. La **relació d'inclusió** ($\sqsubseteq$) es defineix com $\sqsubseteq \ = \ \sqsubseteq_1 \ \cup \ \sqsubseteq_2$ on $\sqsubseteq_1$ i $\sqsubseteq_2$ són:

1. $\sqsubseteq_1 \ \subseteq \ VP \times VP$
   $v_1 \in VP$ està inclòs en $v_2 \in VP$ ($v_1 \ \sqsubseteq_1 \ v_2$) si i només si el rectangle del predicat v_1 està gràficament inclòs dins del rectangle de v_2.
2. $\sqsubseteq_2 \ \subseteq \ VT \times VP$
   $t \in VT$ està inclòs en $v \in VP$ ($t \ \sqsubseteq_2 \ v$) si i només si
   - quan t és un cercle, t està gràficament contingut al rectangle de v.
   - quan t és una seqüència de cercles, un dels cercles està gràficament contingut al rectangle de v.
   - quan t és un terme compost o una constant, el cercle principal o arrel està contingut al rectangle de v.

□

Els literals representen visualment el fet bàsic en aquest llenguatge: la inclusió d'un element o conjunt dins un altre conjunt. Aquest fet s'expressa directament mitjançant la inclusió gràfica. Distingim dos tipus de literals visuals: literals visuals de condició i literals visuals de conclusió, en funció del rol que tenen en la definició d'un predicat. Els literals visuals de condició estan formats per la inclusió d'un únic terme visual dins d'un predicat visual, que ha estat dibuixat utilitzant línies fines.

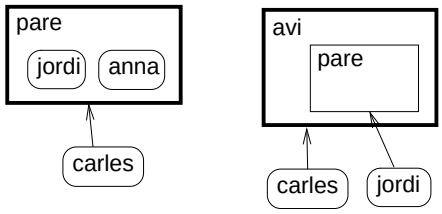


Figura 4.4: Exemples de literals visuals de conclusió

Definició. Un **literal visual de condició** ben format l està composat de:

- un predicat visual v tal que el seu recuadre està dibuixat amb línies primes.
- Un terme visual t gràficament inclòs a v ($t \sqsubseteq v$).

□

A la Figura 4.3 trobem exemples de literals visuals de condició. El primer literal representa el fet que el terme a satisfà el predicat p. El segon literal representa que *joan* és una *persona* i el tercer literal que *algú* és el *pare* de l'*anna*.

Els literals visuals de conclusió estan formats per la inclusió gràfica d'un o més termes visuals i/o predicats visuals dins d'un altre predicat visual (dibuixat mitjançant línies gruixudes).

Definició. Un **literal visual de conclusió** l està composat de:

- un predicat visual g tal que té el rectangle dibuixat amb línies gruixudes i que s'anomena el predicat objectiu.
- n termes visuals t_i gràficament inclosos dins de g ($t_i \sqsubseteq g$).
- m predicats visuals v_i gràficament inclosos dins de g ($v_i \sqsubseteq g$).

tal que $n + m \geq 1$ i no hi hagi cap més altre inclusió gràfica apart de les esmentades en la definició.

□

A la Figura 4.4 trobem exemples de literals visuals de conclusió. El primer literal visual expressa el fet que en *jordi* i l'*anna* són *pares* d'en *carles*, mentre que el segon representa el fet que els *pares* d'en *jordi* són *avis* d'en *carles*.

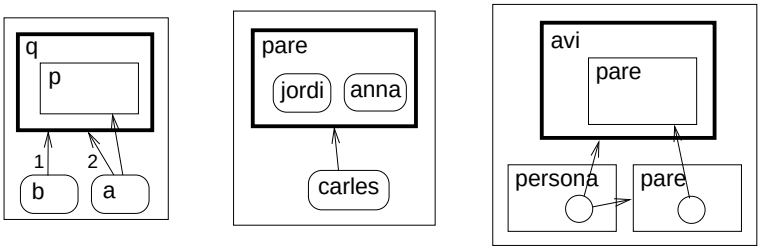


Figura 4.5: Exemples de diagrames de definició

4.1.4. Diagrames

Els diagrames són els elements bàsics que constitueixen un programa visual. Són les sentències del llenguatge i equivalen al que en lògica és una fórmula o que en programació lògica textual és una clàusula.

Un diagrama es defineix com una col·lecció de literals visuals continguts gràficament dins un rectangle que en delimita l'abast sintàctic. Cal destacar que diferents literals d'un mateix diagrama poden compartir subtermes. De fet aquesta notació hereta dels DAGs la facilitat de compartir subtermes (sub-DAGs).

Hi ha dos tipus de diagrames: diagrames de definició i diagrames consulta. Un programa visual es defineix com una col·lecció de diagrames definició. Els diagrames consulta s'utilitzen per a posar preguntes als programes.

Diagrames de definició

Un diagrama de definició està format per un literal visual de conclusió i (de forma opcional) un conjunt de literals visuals de condició. El literal visual de conclusió conté el predicat visual que es defineix, mentre que els literals visuals de condició especifiquen les condicions sota les quals la definició és vàlida. A un diagrama definició es pot identificar fàcilment el predicat definit o predicat objectiu, ja que és el que està dibuixat amb línies gruixudes. A continuació formalitzem la sintaxi d'un diagrama de definició (A la secció 4.2 en definim formalment la semàntica):

Definició. Un **diagrama de definició** ben format (d) es composa de:

1. un rectangle que delimita l'abast sintàctic del diagrama. Tots els literals visuals del diagrama han d'estar completament inclosos dins d'aquest rectangle.
2. un i només un literal visual de conclusió
3. $n \geq 0$ literals visuals de conclusió.

de forma que no hi ha inclusions gràfiques entre predicats de diferents literals visuals.

□

A la figura 4.5 trobem exemples de diagrames de definició. En el primer diagrama definim el predicat q mitjançant la inclusió de conjunts. Aquest diagrama expressa visualment la següent inclusió de conjunts:

$$\{w|p(a,w)\} \subseteq \{z|q(b,a,z)\}$$

que equival a la següent fórmula en lògica de primer ordre:

$$p(a,y) \rightarrow q(b,a,y)$$

Els altres dos diagrames defineixen els predicats *pare* i *avi*. Els predicat *pare* es defineix donant dos elements (*jordi* i *anna*) que pertanyen al conjunt dels pares d'en *carles*. El predicat *avi* es defineix especificant un subconjunt d'ell, és a dir, es diu que els pares dels pares d'algú (una persona) són els avis d'aquesta persona.

Diagrames consulta

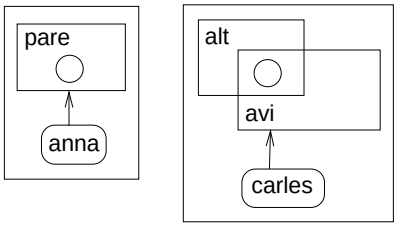


Figura 4.6: Exemples de diagrames consulta

Un diagrama consulta es defineix de forma idèntica que un diagrama de definició, excepte que aquest cop no hi ha cap literal visual de conclusió. A continuació definim formalment la sintaxi d'un diagrama consulta:

Definició. Un **diagrama consulta** ben format (q) està composat per:

1. un rectangle que delimita l'abast sintàctic del diagrama. Tots els literals visuals del diagrama han d'estar completament inclosos dins d'aquest rectangle.

2. $n \geq 0$ literals visuals de conclusió.

de forma que no hi ha inclusions gràfiques entre predicats de diferents literals visuals.

□

Un diagrama consulta es pot veure com una fórmula existencial, és a dir, com una conjunció d'inclusions on les variables estan quantificades existencialment. A la figura 4.6 trobem exemples de diagrames consulta.

En el primer diagrama consulta volem saber qui són els *pares* de l'*anna*, el que en lògica de primer ordre és: $\exists x\ pare(ana, x)$?. En el segon diagrama consulta es busquen qui són els *avis* d'en *carles* que són *alts*. Textualment, en lògica de primer ordre, s'expressa com:

$$\exists x\ alt(x) \wedge avi(carles, x)$$

.

4.2. Semàntica denotacional

Primer de tot definim els models del llenguatge visual proposat:

Definició. Un **model de primer ordre** $\mathcal{A}$ està composat per:

1. Un domini semàntic $|\mathcal{A}|$: el conjunt d'elements sobre els quals es projecten els termes visuals.

2. Una funció d'interpretació de termes visuals $\phi^{\rho}_{\mathcal{A}}$, que projecta cada terme visual a un element del domini. $\rho : \mathcal{V} \rightarrow |\mathcal{A}|$ és una projecció de cada variable (cercle) a un element del domini semàntic (de forma que tots els cercles d'una seqüència de cercles són projectats als mateix element). $\mathcal{V}$ és el conjunt de possibles variables del llenguatge visual.

3. Una funció d'interpretació de símbols de predicat $\varphi_{\mathcal{A}}$ tal, que donat un símbol de predicat p d'aritat n retorna el conjunt de tuples d'elements del domini semàntic que satisfà el predicat $(\varphi_{\mathcal{A}}(p) \subseteq |\mathcal{A}|^n)$.

□

Definim ara una funció d'interpretació de predicats visuals basada en la funció d'interpretació de símbols de predicat $(\varphi_{\mathcal{A}})$:

Definició. La funció d'**interpretació de predicats visuals** que projecta cada predicat sobre un subconjunt de $|\mathcal{A}|$, es defineix com:

$$\psi^{\rho}_{\mathcal{A}}(v) \;=\; \{z \mid \langle \phi^{\rho}_{\mathcal{A}}(t_1), ..., \phi^{\rho}_{\mathcal{A}}(t_{n-1}), z\rangle \in \varphi_{\mathcal{A}}(p)\}$$

on el predicat visual v correspon al símbol de predicat p d'aritat n ($n \geq 1$), i tots els seus arguments són termes visuals $t_1, ..., t_{n-1}$.

□

4.2.1. Semàntica dels literals visuals

Definim sota quines condicions $\mathcal{A}$ és un model d'un literal visual:

Definició. $\mathcal{A}$ és un **model d'un literal visual de condició** l ($\mathcal{A} \models_{\rho} l$) si i només si

$$\phi^{\rho}_{\mathcal{A}}(t) \in \psi^{\rho}_{\mathcal{A}}(v)$$

on v és el predicat visual i t és el terme visual contingut gràficament a v ($t \sqsubseteq v$).

□

Definició. $\mathcal{A}$ és un **model d'un literal visual de conclusió** l ($\mathcal{A} \models_{\rho} l$) si i només si

$$\begin{array}{ll} \forall t_i & \phi^{\rho}_{\mathcal{A}}(t_i) \in \psi^{\rho}_{\mathcal{A}}(g) \\ \forall v_j & \psi^{\rho}_{\mathcal{A}}(v_j) \subseteq \psi^{\rho}_{\mathcal{A}}(g) \end{array}$$

on g és el predicat visual que es vol definir (dibuixat mitjançant línies gruixudes), t_i són els termes visuals continguts gràficament a g ($t_i \sqsubseteq g$) i v_j són els predicats visuals continguts gràficament a g ($v_j \sqsubseteq g$).

□

$\mathcal{A}$ serà un model d'un literal visual si i només si després d'interpretar els termes visuals i els predicats visuals del literal visual, les inclusions originades per la inclusió gràfica es compleixen.

4.2.2. Semàntica d'un diagrama de definició

A continuació definim quan un model de primer ordre $\mathcal{A}$ és un model d'un diagrama de definició.

Definició. $\mathcal{A}$ és un **model d'un diagrama de definició** d ($\mathcal{A} \models d$) si i només sí

$$\forall \rho \ \ (\mathcal{A} \models_\rho h \ \vee \ \mathcal{A} \not\models_\rho l_1 \ \vee \ ... \ \vee \ \mathcal{A} \not\models_\rho l_n)$$

on h és el literal visual de conclusió, $l_1 ... l_n$ ($n \geq 0$) són els literals visuals de condició i ρ és la funció de substitució de variables.

□

La intuïció que hi ha darrera aquesta definició és que un model $\mathcal{A}$ és un model d'un diagrama definició si, per totes les possibles substitucions de variables, quan és model de tots els literals visuals de condició llavors també és model del literal visual de conclusió.

4.2.3. Semàntica d'un diagrama consulta

Finalment definim quan $\mathcal{A}$ és un model d'un diagrama consulta.

Definició. $\mathcal{A}$ és un **model d'un diagrama de consulta** q ($\mathcal{A} \models q$) si i només si

$$\exists \rho \ \ (\mathcal{A} \models_\rho l_1 \ \wedge \ ... \ \wedge \ \mathcal{A} \models_\rho l_n)$$

on $l_1 ... l_n$ ($n \geq 0$) són els literals visuals de condició i ρ és la funció de substitució de variables.

□

Diem que $\mathcal{A}$ modela un diagrama consulta si existeix una substitució de variables tal, que sigui model de tots els literals visuals de condició.

4.3. Correspondència expressiva amb les clàusules de Horn

L'objectiu d'aquesta secció és mostrar intuïtivament que la potència expressiva del llenguatge visual és similar al llenguatge de les clàusules de Horn. De

fet veurem que qualsevol clàusula de Horn sense predicats d'aritat zero es pot expressar com a diagrama del llenguatge visual i viceversa:

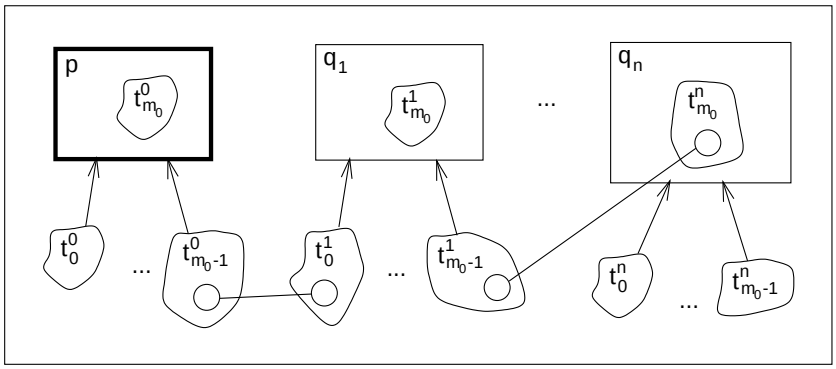
- Donada qualsevol clàusula de Horn on tots els predicats tenen aritat igual o més gran a 1 (amb o sense literal positiu p):

  $$p(t_1^0, \ldots\ t_{m_0}^0)\ \vee\ \neg q_1(t_1^1, \ldots\ t_{m_1}^1)\ \vee\ \ldots\ \vee\ \neg q_n(t_1^n, \ldots\ t_{m_n}^n)$$

  podem expressar-la com a un diagrama definició o un diagrama consulta, en funció de si el literal positiu apareix o no, seguint els següents passos:

  1. Representem cada predicat com una capsa, sense que hi hagi cap inclusió gràfica entre capses de diferents predicats. Si hi ha literal positiu el representarem com una capsa amb les línies gruixudes.
  2. S'estableix que el darrer argument serà el rang del conjunt que utilitzarem per a representar gràficament l'abstracció del predicat. Representem el darrer terme de cada predicat inclòs gràficament dins la seva capsa.
  3. Per cada predicat representem tots els altres arguments de forma visual. No hi hauran inclusions d'aquests termes amb cap capsa.
  4. Unim mitjançant línies totes les ocurrències d'una mateixa variable.

  El resultat que obtenim és sempre un diagrama definició o un diagrama consulta correcte equivalent a la clàusula de Horn inicial:

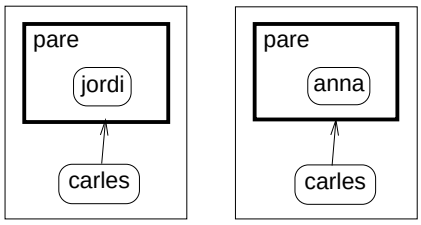


- Si anem seguint les definicions veurem que qualsevol diagrama ben format es pot traduir a una clàusula de Horn equivalent seguint un procés de traducció similar al descrit a la secció 6.5.

  Cada literal visual de condició equival a un literal negatiu de la clàusula de Horn. En el cas dels literals visuals de conclusió podem distingir dos casos. Quan el literal prové d'una inclusió d'un terme visual a un predicat visual, llavors equival directament a un literal positiu. Si el literal prové de

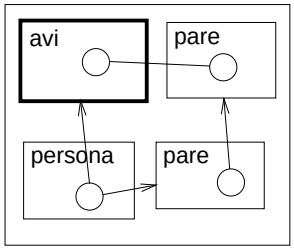
la inclusió de dues capses es pot transformar el diagrama en un diagrama equivalent, on la inclusió de capses ha estat substituïda per dues inclusions de variables dins de capses. En aquest cas el literal visual de condició haurà donat lloc a dos literals textuals, un de positiu i un de negatiu.

A la secció 6.5 mostrem pas a pas com funciona un algorisme de traducció d'una sintaxi visual propera a la definida en aquest capítol a clàusules de Horn. A continuació mostrem el procés de traducció a clàusules de Horn de dos diagrames de la figura 4.5. El primer diagrama conté dos literals, si bé primer cal separar-los ja que donaran lloc a dues clàusules de Horn diferents.



$$pare(carles, jordi)$$
$$pare(carles, maria)$$

En l'altre diagrama, que mostra la definició del predicat *avi*, primer cal transformar la inclusió de capses i després traduir-lo:



$$avi(x, y) \ \vee \ \neg persona(x) \ \vee \ \neg pare(x, w) \ \vee \ \neg pare(w, y)$$

Per tant hem vist intuïtivament que el llenguatge visual proposat té una potència expressiva equivalent al llenguatge de les clàusules de Horn sense predicats d'aritat zero[1]. A causa de la metàfora conjuntista utilitzada en la representació visual dels predicats, no hi ha una representació visual *directa* dels

[1]La intuïció ens porta a creure que ha d'existir algun isomorfisme de models que permeti demostrar que el llenguatge és equivalent a les clàusules de Horn. Trobar i demostrar l'existència d'aquest isomorfisme no és evident i no és l'objectiu de la tesi.

predicats d'aritat zero. Per incloure'ls en la representació hauria calgut utilitzar algun mecanisme simbòlic menys proper a la semàntica que l'utilitzat. Creiem que per als objectius d'aquesta tesi és millor prescindir d'aquests predicats nul·laris i mantenir la simplicitat del llenguatge.

Capítol 5

Semàntica operacional visual

En aquest capítol mostrem com es poden portar a terme inferències directament amb els diagrames. Definim un sistema d'inferència completament visual proper a la resolució SLD. De fet el llenguatge visual formalitzat al capítol anterior ha estat considerat de forma que representa el mateix subconjunt de la lògica que les clàusules de Horn. El sistema d'inferència visual que definim en aquest capítol és similar a la resolució SLD.

Volem tenir una semàntica operacional simple d'entendre i, sobretot, propera a la semàntica intuïtiva dels diagrames. Per tal d'aconseguir-ho, el sistema d'inferència visual que presentem està basat en transformacions de diagrames que utilitzen la inclusió gràfica, de forma que la seva transitivitat intrínseca s'obté de forma implícita. Creiem que el nostre llenguatge visual permet una semàntica operacional (o teoria de la demostració) propera a la semàntica del llenguatge, fet que no es dóna en el cas convencional (resolució SLD textual). Una altra millora del nostre apropament és la possibilitat de visualitzar en un únic diagrama diferents solucions juntament amb el camí seguit per a trobar-les, és a dir, la seva demostració.

A la secció 5.1 definim el procediment d'unificació visual. A la secció 5.2 formalitzem els conceptes de diagrama consulta estès i diagrama resposta, necessaris per a portar a terme inferències visuals, i definim la regla d'inferència visual. A la secció 5.3 veiem com es porta a terme un pas d'inferència i estudiem el mecanisme de duplicació, que ens permetrà mostrar múltiples possibles solucions en un mateix diagrama. Finalment, a la secció 5.5 mostrem el funcionament de la inferència visual mitjançant un exemple pràctic.

5.1. Unificació

La unificació és molt important ja que cada pas d'inferència implica la unificació de dos literals visuals: un literal visual de condició i un literal visual de

conclusió.

Definició. [Unificació de termes visuals] Dos termes visuals unifiquen si i només si es dóna un dels següents casos:

- Dos variables (cercles) unifiquen sempre i el resultat és un altre cercle.
- Un cercle i un altre terme visual unifiquen sempre i el resultat és el terme visual. Si la variable pertany a una seqüència de cercles, llavors totes les variables s'unifiquen amb el terme visual i les línies que unien les variables de la seqüència desapareixen.
- Dos constants unifiquen si el seu símbol de constant és el mateix.
- Dos termes visuals compostos unifiquen si i només si els seus símbols de funció (a l'arrel) són els mateixos i tots els parells de termes visuals corresponents als arguments unifiquen. Quan hi ha més d'un argument, les etiquetes de les fletxes s'utilitzen per a aparellar els arguments. Les fletxes que no estan etiquetades s'ordenen seguint el sentit de les agulles del rellotge, a partir de la cantonada superior dreta de l'arrel del terme.

□

Cal remarcar que no és necessari aplicar substitucions ja que les variables estan compartides explícitament per diferents literals visuals. Això es produeix pel fet que els termes visuals estan basats en DAGs.

Definició. [Unificació de predicats visuals] Dos predicats visuals unifiquen si i només si el seu símbol de predicat és el mateix i tots els parells de termes visuals corresponents als arguments dels dos predicats unifiquen. A l'igual que a l'unificació dels termes visuals, quan hi ha més d'un argument, les etiquetes de les fletxes s'utilitzen per a aparellar els arguments. Les fletxes que no estan etiquetades s'ordenen seguint el sentit de les agulles del rellotge , a partir de la cantonada superior dreta de l'arrel del terme.

□

Definició. [Unificació de literals visuals] Sigui l_1 un literal visual de condició (representant la inclusió $t \sqsubseteq v$), i l_2 un literal visual de conclusió (representant la inclusió $t_i \sqsubseteq g, v_j \sqsubseteq g$ on $1 \geq i \geq n$, $1 \geq j \geq m$ i $n + m \geq 1$). l_1 i l_2 unifiquen si i només si:

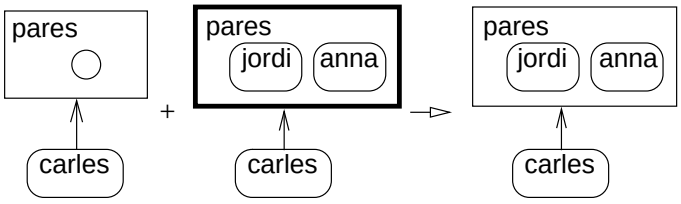
1. Els predicats visuals dels dos literals, v i g, unifiquen.

2. Es dóna almenys un dels següents casos:
   - $m \geq 1$. En aquest cas es formen nous literals visuals de conclusió ($t \sqsubseteq v_j$).
   - t unifica amb algun $t_1, ..., t_m$.

□

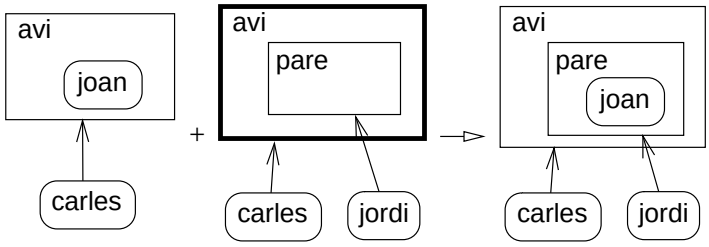
La unificació de dos literals visuals comprèn dos tipus diferents de inferència, en funció del tipus d'inclusió en el literal visual de conclusió:

1. Inclusió d'elements



En aquest exemple el predicat visual corresponent a *pare d'en Carles* es defineix donant dos elements que el compleixen (*jordi*, *anna*). D'aquesta forma la unificació del literal visual de conclusió que defineix aquest predicat i el literal visual de condició es fa unificant directament els termes continguts en ambdós literals visuals. La unificació de dos literals visuals pot produir instanciacions múltiples de variables. Això succeeix quan el literal visual de conclusió té una inclusió múltiple i existeix més d'una possibilitat alhora d'instanciar la variable. En aquesta situació totes les possibles instanciacions es poden representar alhora en el literal visual resultant mitjançant un procés anomenat duplicació —tal com veurem a la secció 5.3.1. En el nostre cas es produeix una instanciació múltiple d'una variable amb dos constants: *jordi* i *anna*.

2. Inclusió de conjunts



Aquest segon cas es dóna quan el predicat visual corresponent al literal de conclusió està definit amb un subconjunt d'elements que el compleixen, és a dir, mitjançant la inclusió d'un altre predicat visual. Veiem com el predicat *avis d'en carles* es defineix mitjançant la inclusió del predicat *pares d'en jordi*. És a dir, es diu que tots els pares d'en Pere són avis d'en Carles. En unificar els dos literals visuals es crea un nou literal visual: és a dir, reduïm el problema de demostrar que en *joan* és un *avi d'en carles*, ja que demostrem primer que és *pare d'en jordi*. Aquest raonament captura la transitivitat intrínseca de la inclusió de conjunts (i també la de la implicació lògica)

5.2. Inferència visual

Tot seguit formalitzarem el nucli del sistema d'inferència visual o semàntica operacional del nostre llenguatge. La regla d'inferència visual que proposem és similar a la resolució convencional. En realitat un diagrama, a l'igual que una clàusula textual, és un conjunt de literals visuals. Com veurem, la regla d'inferència s'aplica de forma similar a la resolució textual i es basa en la unificació de literals visuals tractada anteriorment.

A la secció 5.2.1 introduïm la regla d'inferència i mostrem com funciona mitjançant un exemple. Tot i que la regla d'inferència es pot aplicar als diagrames consulta, tal i com han estat definits al capítol 4 secció 5.2.3, mostrem que aquests diagrames consulta es poden millorar (obtenint diagrames consulta estesos) per tal de poder portar a terme instanciacions múltiples i aprofitar al màxim les possibilitats del llenguatge. Finalment, a la secció 5.2.4 formalitzem els diagrames resposta.

5.2.1. Regla d'inferència visual

Primer de tot definim formalment la regla d'inferència visual.

Definició. [Regla d'inferència visual] Sigui q un diagrama consulta amb els següents literals visuals de condició: $l_1, ..., l_n$ $(n \geq 1)$, i sigui d un diagrama definició tal que el literal visual de conclusió és g i $l'_1, ..., l'_m$ $(m \geq 0)$ són els literals visuals de condició.

Es podrà realitzar un pas d'inferència si i només si existeix i tal que l_i i g unifiquin. Llavors obtenim un nou diagrama consulta q', resultat de la inferència, i que contindrà els següents literals:

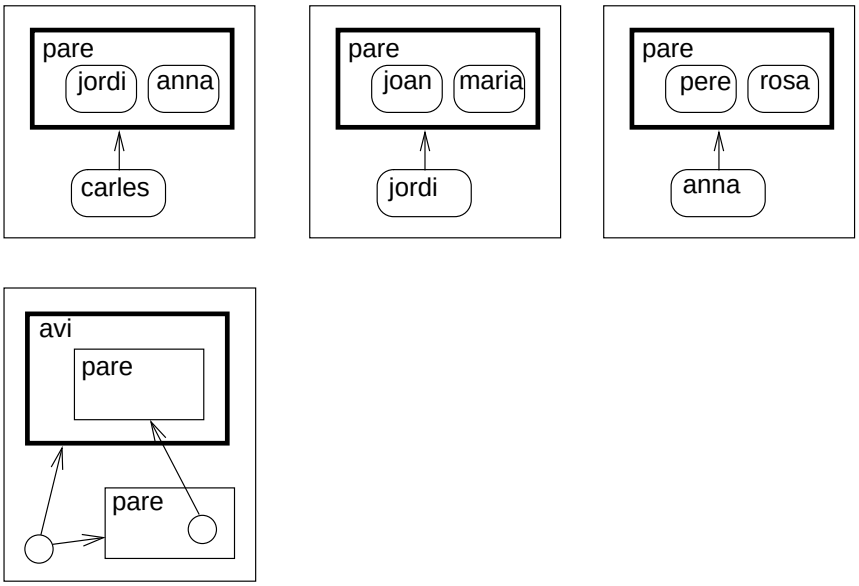
- $l_1, ..., l_{i-1}, l_{i+1}, ..., l_n$
- $l'_1, ..., l'_m$

- Qualsevol nou literal obtingut de la unificació de l_i i g.

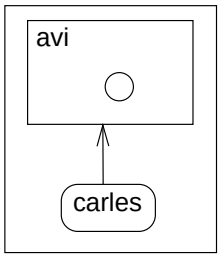
□

5.2.2. Exemple d'inferència visual

En aquest exemple disposem d'un petit programa lògic que ens defineix els predicats *pare* i *avi* i ens servirà per a mostrar l'aplicació de la regla d'inferència proposada en la secció anterior:



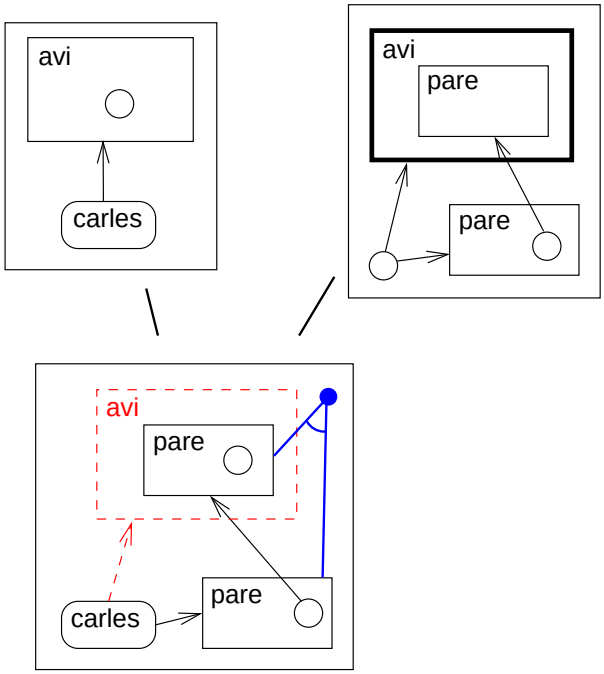
A continuació ens plantegen una pregunta *Qui són els avis d'en Carles?* en forma de diagrama consulta:



Aquest diagrama consulta equival a la següent fórmula lògica:

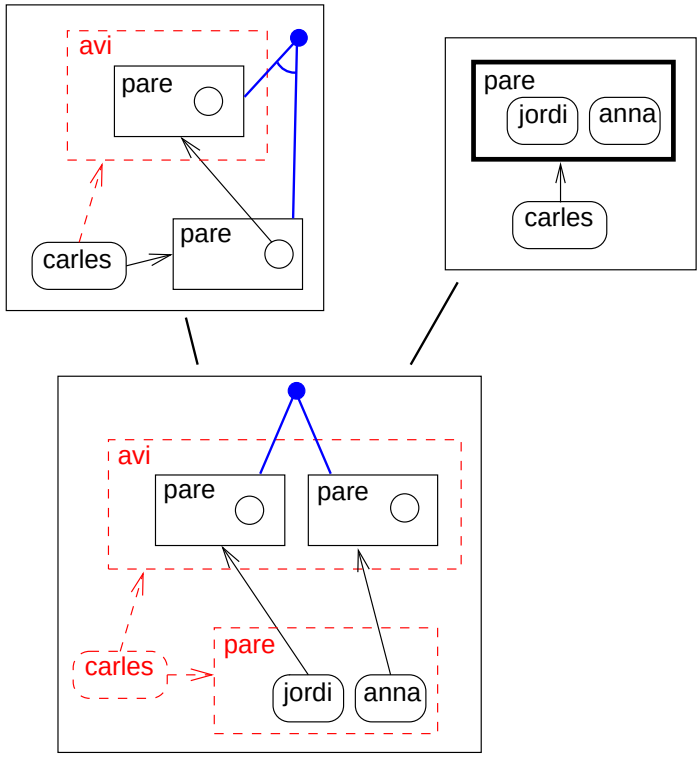
$$\exists x \;\; avi(Carles, x)$$

Apliquem la regla d'inferència visual unificant el literal *avi* del diagrama consulta amb el seu equivalent al diagrama definició. Aquest pas captura visualment la transitivitat de la relació d'inclusió i aporta nova informació de forma 'gratuïta'. És el que A.Shimojina anomena *free rides*, o l'obtenció de nova informació gràcies a la utilització d'una sintaxi visual. En el nostre cas en utilitzar la notació diagramàtica obtenim de forma natural la nova inclusió de la variable, dins del predicat visual *pare* que constitueix un nou literal a resoldre.



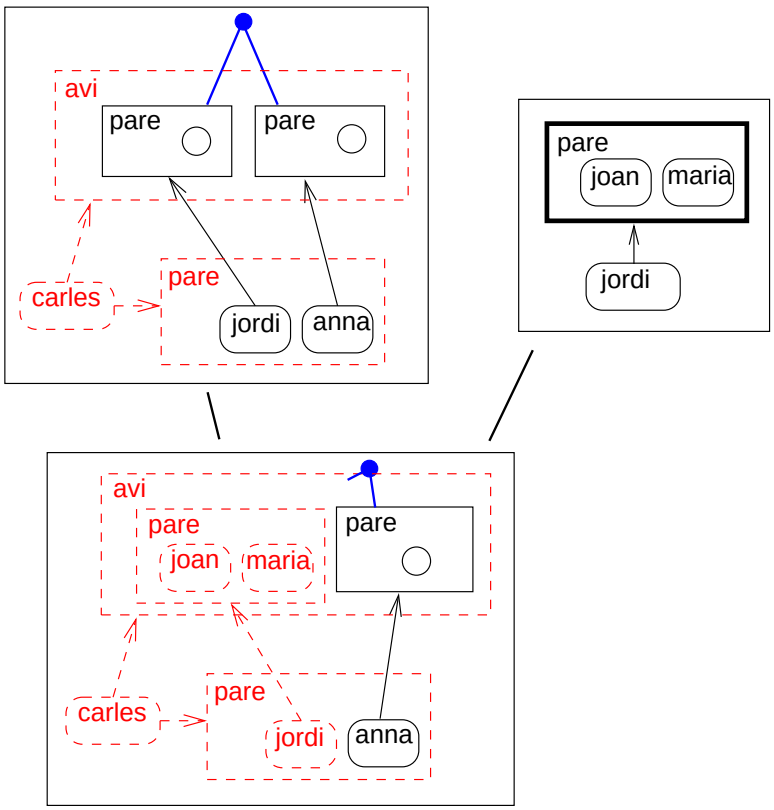
A més, per tal de deixar constància dels passos seguits no esborrem ni els predicats visuals ni els termes visuals dels literals que ja han estat resolts. Els mantenim al diagrama amb línies discontínues per tal que no es confonguin amb els literals pendents. Quan un terme visual és compartit per dos o més literals visuals només es marca com a resolt (amb línies discontínues) un cop han estat resolts tots els literals on apareix.

En aquest punt tenim dues possibilitats en funció de quin sigui el literal visual del diagrama consulta que escollim. En primer lloc optem pel literal corresponent al predicat *pares d'en carles* i portem a terme un nou pas d'inferència visual.

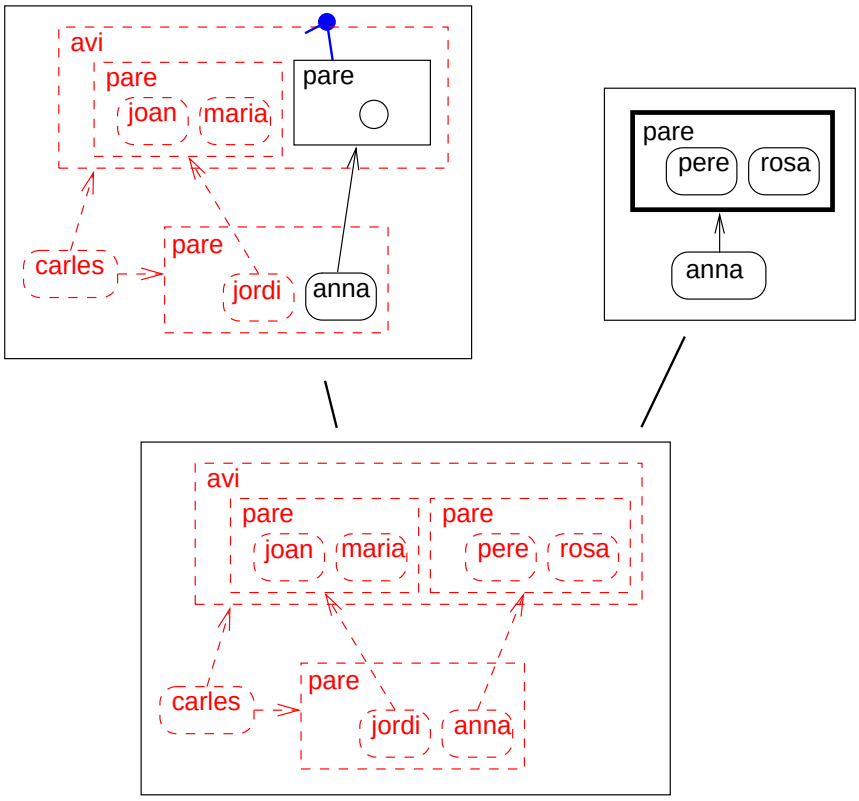


En portar a terme la inferència visual ens trobem que són possibles dos instanciacions diferents, ja que el literal visual de conclusió (al diagrama definició) conté dos termes visuals. Aquí observem un altre dels avantatges de la nostra notació: en aquests casos es pot portar a terme una instanciació múltiple —tal com hem comentat abans a la secció 5.1— i representar les dues solucions al mateix diagrama. A causa d'aquesta instanciació múltiple alguns literals s'han de duplicar i s'ha d'anotar el diagrama amb un arbre AND-OR, que reflecteixi correctament les diferents opcions. A la secció 5.3.1 formalitzem aquesta duplicació.

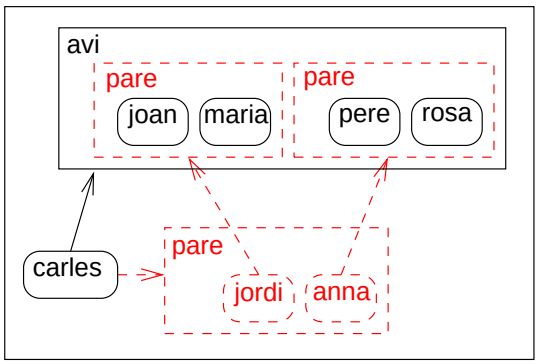
Finalment només ens falta resoldre els dos literals visuals pendents. N'escollim un i portem a terme un altre pas d'inferència visual.



Portem a terme el darrer pas d'inferència resolent el darrer literal visual pendent.



I finalment obtenim el diagrama resposta marcant al diagrama consulta els predicats i termes visuals originals, i els que provenen de la instanciació d'un terme visual present al diagrama consulta.



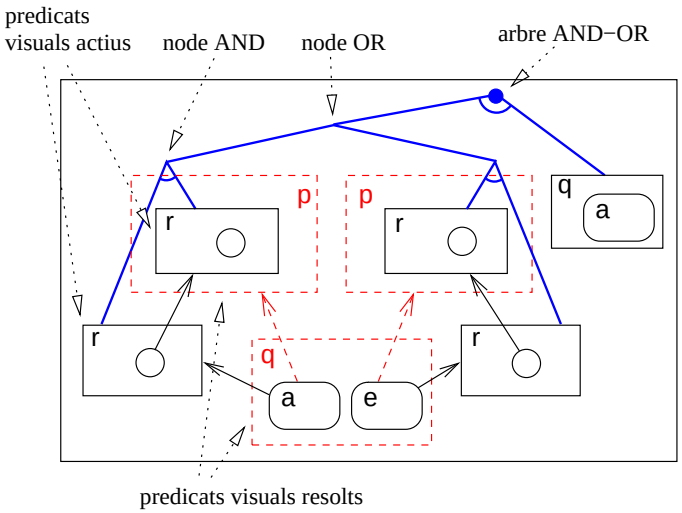


Figura 5.1: Un exemple de diagrama consulta estès

5.2.3. Diagrames consulta estesos

Tal com hem vist, el resultat d'una inferència visual és un diagrama consulta ampliat que pot emmagatzemar informació sobre els literals que ja han estat resolts així com representar també les relacions de conjunció i disjunció entre els literals que quedin per resoldre. Aquest tipus de diagrames s'anomenen *diagrames consulta estesos* i es defineixen de la següent forma:

Definició. [Diagrames consulta estesos] Un diagrama consulta ampliat és un diagrama consulta amb les següents característiques addicionals:

- Els literals que ja han estat resolts es dibuixen utilitzant línies de punts per tal de distingir-los dels literals pendents de resoldre.
- Un arbre AND-OR que representa l'estructura lògica de la consulta a resoldre representat pel diagrama. Les fulles de l'arbre AND-OR són tots els literals pendents de resoldre. Els nodes AND de l'arbre es diferencien dels nodes OR mitjançant una línia circular que uneix totes les branques del node (similar a la representació d'un angle).

□

Normalment ens referirem als literals visuals pendents de resoldre com a *literals visuals actius*. A la figura 5.1 trobem un exemple d'un diagrama consulta estès, amb les seves diferents parts indicades.

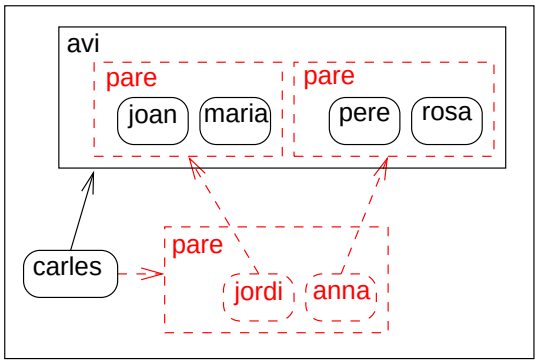


Figura 5.2: Un exemple de diagrama resposta

Aquest arbre AND-OR és un element allunyat de la semàntica dels diagrames, però necessari per tal de poder representar la disjunció entre diferents literals visuals actius. Tal com han observat Barwise i Etchemendy [13] la representació diagramàtica de la disjunció és difícil i sovint cal recòrrer a convencions i mecanismes simbòlics.

5.2.4. Diagrames resposta

L'equivalent a una clàusula buida en resolució SLD textual és un diagrama consulta estès sense literals actius, és a dir, un diagrama consulta estès tal que tots els literals han estat resolts[1]. Al no haver-hi literals actius tampoc hi ha l'arbre AND-OR. Quan s'obté un diagrama amb aquestes característiques la consulta ha estat resolta, i el mateix diagrama consulta estès conté les solucions que es presenten mitjançant un diagrama resposta.

Definició. [Diagrama resposta] Un diagrama resposta és un diagrama consulta estès tal que tots els literals han estat resolts. Per tal de ressaltar la solució, tots els predicats i termes visuals presents al diagrama consulta original es dibuixen amb línies normals enlloc de línies de punts.

□

El procés d'inferència es pot veure com un procés consistent en completar un diagrama consulta fins a aconseguir un diagrama resposta. La solució es

[1] Ja que la consulta conté diferents alternatives és possible tenir una solució de la consulta tot i quedar literals actius. Nosaltres considerarem sempre el cas en què no queden més literals actius, és a dir, quan totes les branques OR han estat recorregudes (veure secció 5.3).

va emmagatzemant al diagrama consulta estès fins que tots els objectius s'han assolit i obtenim un diagrama resposta. A la figura 5.2 trobem un exemple de diagrama resposta.

5.3. L´algorisme: un pas d'inferència visual

Un pas d'inferència visual (resoldre un literal l_k de un diagrama consulta estès) es porta a terme en tres passos:

1. Aplicar la regla d'inferència tal com s'ha definit a la secció 5.2. Pot succeir una instanciació múltiple i que nous literals provinents del diagrama definició s'afegeixin al diagrama consulta estès. Aquests nous literals no estan lligats a l'arbre AND-OR actual.

2. Reconstruir l'arbre AND-OR: els nous literals s'afegeixen al node AND on el literal resolt l_k estava. Si l_k estava lligat directament a un node OR llavors es crea un nou node AND.

3. Si s'ha produït una instanciació múltiple llavors cal duplicar els literals visuals que comparteixen variables amb el literal resolt. Definirem formalment aquest procés de duplicació a la següent secció.

A les següents seccions també adreçarem el procés de duplicació, les modificacions de l'arbre AND-OR i altres aspectes relacionats amb el backtracking.

5.3.1. Duplicació

Quan es produeix una instanciació múltiple una part del diagrama es duplica. En aquesta secció definirem formalment com es produeix aquesta duplicació.

Duplicació dels literals visuals

Definim una funció de duplicació ($\Phi()$) tal que en passar-li com a argument el literal que es vol resoldre (és a dir, el literal en el qual s'ha produït la duplicació) retorna el conjunt de literals visuals que s'hauran de duplicar. Aquest conjunt constarà de 1) tots els literals visuals que continguin una variable instanciada de forma múltiple i 2) qualsevol altre literal visual que comparteixi una variable amb algun literal visual seleccionat per duplicació.

Definició. [Funció de duplicació]

La funció de duplicació es defineix en dos passes. Primer calculem el conjunt de literals visuals relacionats amb la variable que ha estat instanciada de forma múltiple. Seguidament afegim recursivament tots els altres literals visuals que també s'han de duplicar.

Sigui l_m el literal on una instanciació múltiple ha ocorregut. Aquest literal correspon al predicat visual amb símbol r i t_m és el terme que conté la variable duplicada.

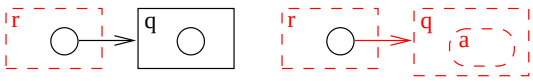
Pas 1:
El conjunt inicial de literals visuals es defineix com el conjunt de literals que està relacionat amb t_m:

$$\Phi(l_m) := \{l | l \ \ relacionat\ amb \ \ t_m\}$$

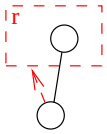
Hi ha tres casos en què un literal visual l (corresponent al símbol de predicat q) està relacionat amb el terme duplicat t_m (Noteu que en aquest cas es poden produir duplicacions de predicats que ja han estat resolts):

1. Literal amb una variable inclosa que es duplica:
   - $l_m \neq l$
   - l és un literal actiu
   - Els dos literals l i l_m tenen la mateixa variable inclosa en els respectius predicats visuals.

2. Literal amb un terme visual duplicat com a argument:
   - $l_m \neq l$
   - Un dels arguments de l és un terme visual que conté una variable que pertany al terme visual duplicat t_m.



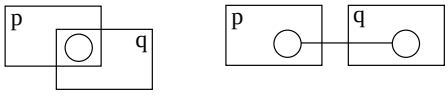
3. Literal resolt on la variable duplicada també pertany a un terme visual argument del predicat visual:
   - $l_m = l$ (i per tant $r = q$)
   - La variable duplicada està inclosa al predicat visual i alhora és part d'un dels arguments del predicat visual.



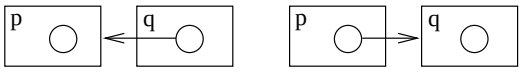
Pas 2:
Repetir $\Phi(l_m) := \Phi(l_m) \cup \{l_1\}$ fins que no quedi cap literal visual per afegir, de forma que:

- Un literal visual l_1 (corresponent al símbol de predicat p) està relacionat amb un literal visual $l_2 \in \Phi(l_m)$ (corresponent al símbol de predicat p)
- Un dels quatre casos exposats a continuació es compleix:

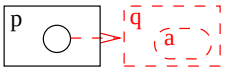
1. - $l_1 \neq l_2$
   - l_1 i l_2 són actius
   - Els dos literals comparteixen la mateixa variable (o terme que conté una variable) inclosa en els respectius predicats visuals.



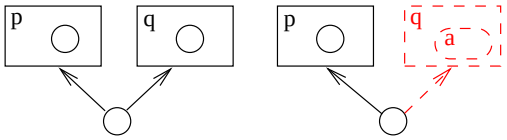
2. - $l_1 \neq l_2$
   - l_1 i l_2 són actius
   - Un dels literals conté una variable (o terme que conté una variable) com a argument del respectiu predicat visual i aquesta variable està inclosa en l'altre predicat visual.



3. - $l_1 \neq l_2$
   - l_2 és actiu
   - l_1 és un literal resolt
   - l_2 conté una variable (o terme amb una variable) que és argument del predicat visual inclòs al predicat visual de l_2.



4.
   - $l_1 \neq l_2$
   - l_2 és actiu
   - Els dos literals visuals contenen la mateixa variable com a argument dels seus predicats visuals.



□

Per definició de Φ, el conjunt de literals visuals per duplicar com a conseqüència d'una instanciació múltiple ocorreguda al literal l_m, és a dir el conjunt $\Phi(l_m)$, no comparteixen cap variable amb cap altre literal que no pertanyi a $\Phi(l_m)$. Per tant el procés de duplicació es pot portar a terme simplement duplicant tots els literals visuals de $\Phi(l_m)$.

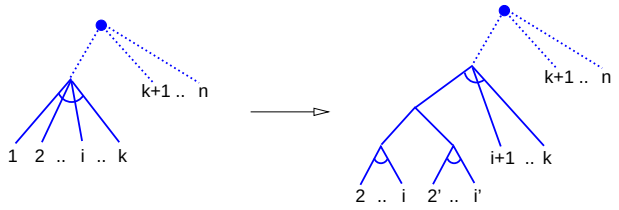
Duplicació de l'arbre AND-OR

Per tal de formalitzar la transformació de l'arbre AND-OR després de la duplicació, ens cal primer provar que el conjunt de literals visuals a duplicar sempre pertanyen a la mateix branca OR, és a dir, a un arbre AND comú:

Lema. *Dos literals l_1 i l_2 tals que $l_1, l_2 \in \Phi(l_m)$ (per algun l_m) no estaran mai sota branques diferents d'un node OR.*
Demostració. *La única forma en què es poden introduir nodes OR és a través d'una instanciació múltiple d'un literal visual l_m. Quan dupliquem una part del diagrama dupliquem tots els literals que pertanyen a $\Phi(l_m)$, per tant és impossible que dos literals estiguin relacionats i alhora pertanyin a branques diferents d'un node OR.*

□

A continuació veiem com s'obté un nou arbre AND-OR després d'una duplicació. Suposem que tenim un diagrama amb literals $l_1, ...l_n$, de forma que s'ha donat una instanciació múltiple a l_1. Suposem també que $\Phi(l_1) = \{l_2, ...l_i\}$. La transformació de l'arbre AND-OR és la següent:



Cal notar que abans de la duplicació l'arbre conté tots els literals de $\Phi(l_1)$ sota el mateix node, i l'arbre resultant s'ha obtingut mitjançant la duplicació dels literals $l_2, ...l_i$ (obtenint $l'_2, ...l'_i$) i marcant el literal l_1 com a resolt. El procés de duplicació també ha de respectar totes les inclusions gràfiques, incloses aquelles que involucren literals ja resolts.

5.4. Estratègia d'inferència

La estratègia d'inferència que apliquem és molt similar a la resolució SLD (Lloyd [47]), amb la particularitat que el nostre apropament diagramàtic permet mostrar en un mateix diagrama resposta diferents solucions i el camí que s'ha utilitzat per arribar-hi. De fet el nom SLD és un acrònim per ***S****election*, ***L****inear* i ***D****efinite*. La nostra estratègia és *lineal* ja que sempre apliquem la regla d'inferència a un diagrama definició i un diagrama consulta estès, per a obtenir un nou diagrama consulta estès. També és *definida*, ja que els diagrames definició tenen, a l'igual que les clàusules de Horn, un sol literal conclusió sobre el qual aplicar la regla d'inferència. Finalment cal dir que en aquest treball no hi ha una funció de selecció definida, donat que això queda pendent de l'estudi de la completesa. A priori creiem que es pot definir una funció de selecció sintàctica, com es fa habitualment en la programació lògica (p.e. ordenar els literals en funció de la seva proximitat a la cantonada superior esquerra del rectangle que delimita el diagrama).

5.4.1. Backtraking

Una de les diferències més importants entre el procés d'inferència visual proposat i un procés d'inferència basat en la resolució SLD, és la possibilitat de tenir més d'un camí a resoldre representat explícitament i alhora en un mateix diagrama. Per tant necessitem considerar les transformacions del diagrama quan un literal visual no es pot resoldre, és a dir, quan s'ha de portar a terme el que habitualment s'anomena backtraking.

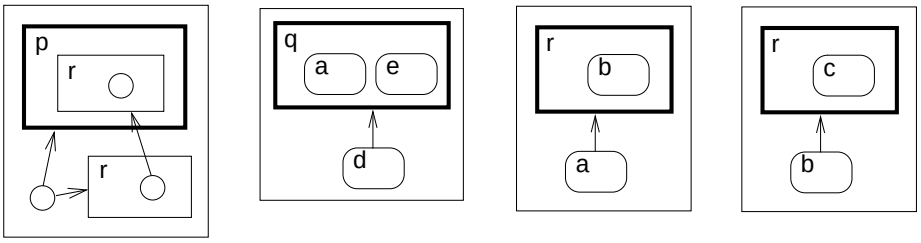
Cada vegada que es resol un literal visual, els seus predicats visuals, ara marcats com a resolts mitjançant línies discontínues, s'associen al node de l'arbre

AND-OR on el literal visual estava associat. Si es troba una solució per al node, és a dir, tots els seus literals visuals estan resolts, llavors tots els predicats visuals (amb línies discontínues) es tornen a associar a l'ancestre del node de l'arbre AND-OR.

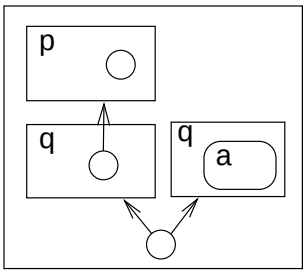
- Quan una branca d'un *node AND* falla, tot el *node AND* falla. Tots els seus literals actius i predicats visuals associats a l'arbre s'esborren del diagrama.
- Quan una branca d'un *node OR* té èxit, el *node OR* es marca com a resolt. Això es fa per tal de recordar, en el cas que les altres branques fallin, que la disjunció representada pel node ja havia estat resolta.
- Quan una branca d'un *node OR* falla i el *node OR* té altres literals actius o ja ha estat marcat com a resolt, només s'esborra la branca que ha fallat. Si es dóna el cas que és la darrera branca del node i el node encara no havia estat marcat com a resolt llavors tot el *node OR* falla.
- Quan l'arrel de l'arbre AND-OR falla, ja sigui un node OR o un node AND, llavors tot el diagrama consulta falla. No s'ha pogut trobar cap solució.

5.5. Un exemple

Per tal d'il·lustrar millor el funcionament del sistema deductiu introduït en aquest capítol utilitzarem l'exemple següent. Suposem que tenim el següent programa lògic:



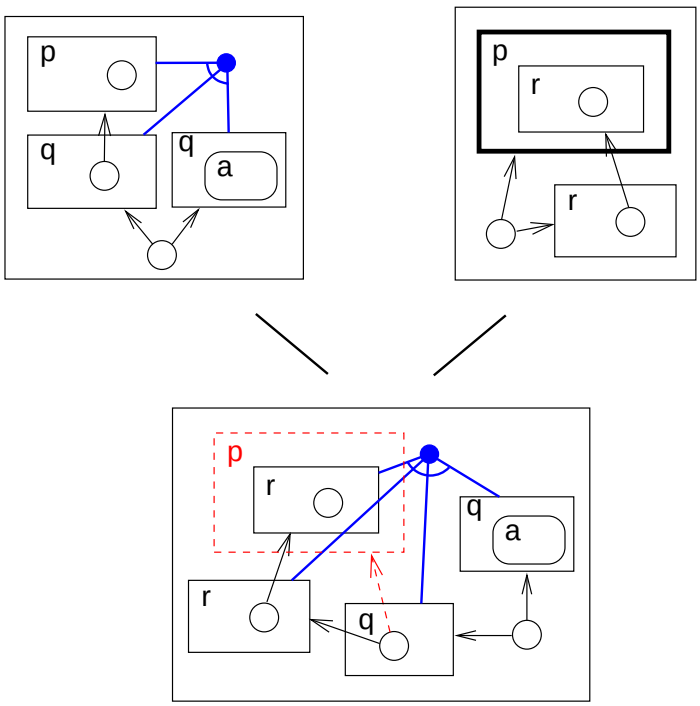
I la consulta inicial:



que expressada en lògica proposicional textual de primer ordre equival a:

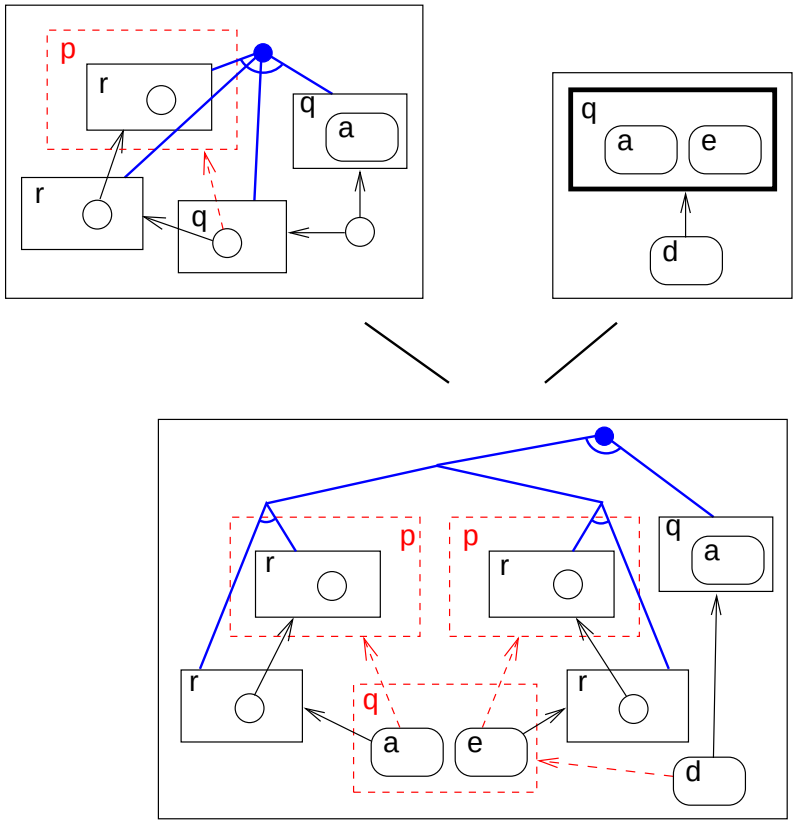
$$\exists x, y, z \;\; p(x, y) \wedge q(z, x) \wedge q(z, a)?$$

Primer de tot s'afegeix un arbre AND-OR amb un únic node AND amb tots els literals actius del diagrama. Aquest arbre mostra l'estructura lògica de la consulta.



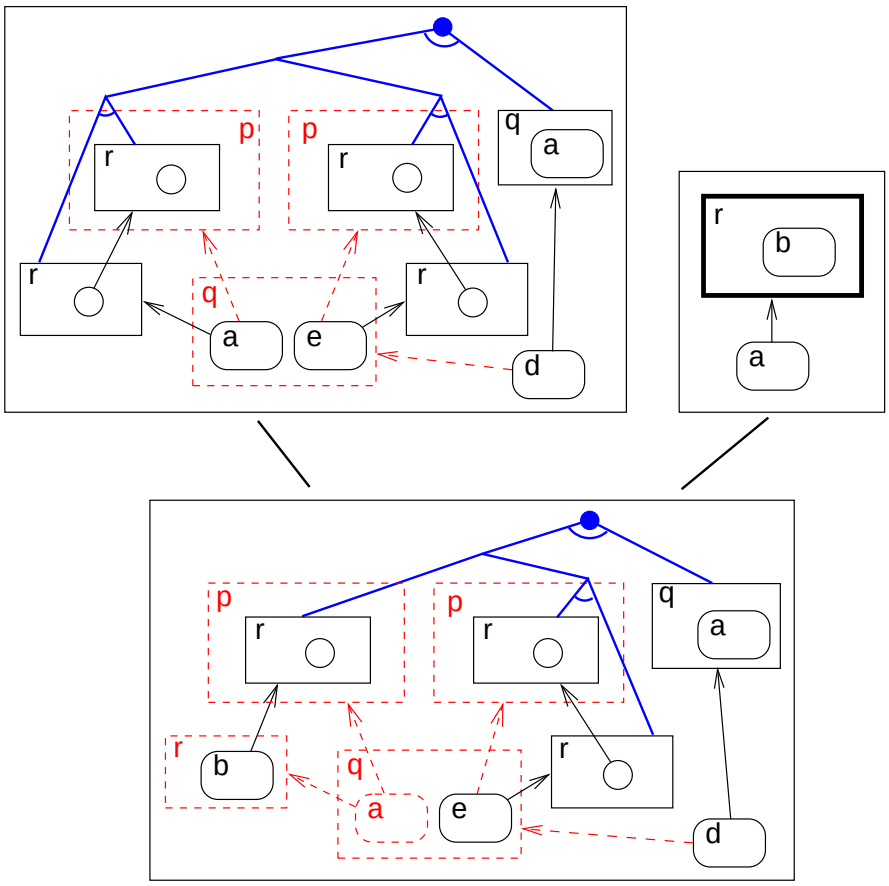
El primer pas d'inferència l'efectuem sobre el literal que correspon al predicat p. Noteu que els nous literals introduïts pel pas d'inferència pertanyen al mateix node AND. No s'ha generat cap node OR ja que no hi ha hagut cap duplicació.

Tot seguit portem a terme el segon pas d'inferència, on s'hi produirà la primera instanciació múltiple que introduirà diferents alternatives dins la consulta.

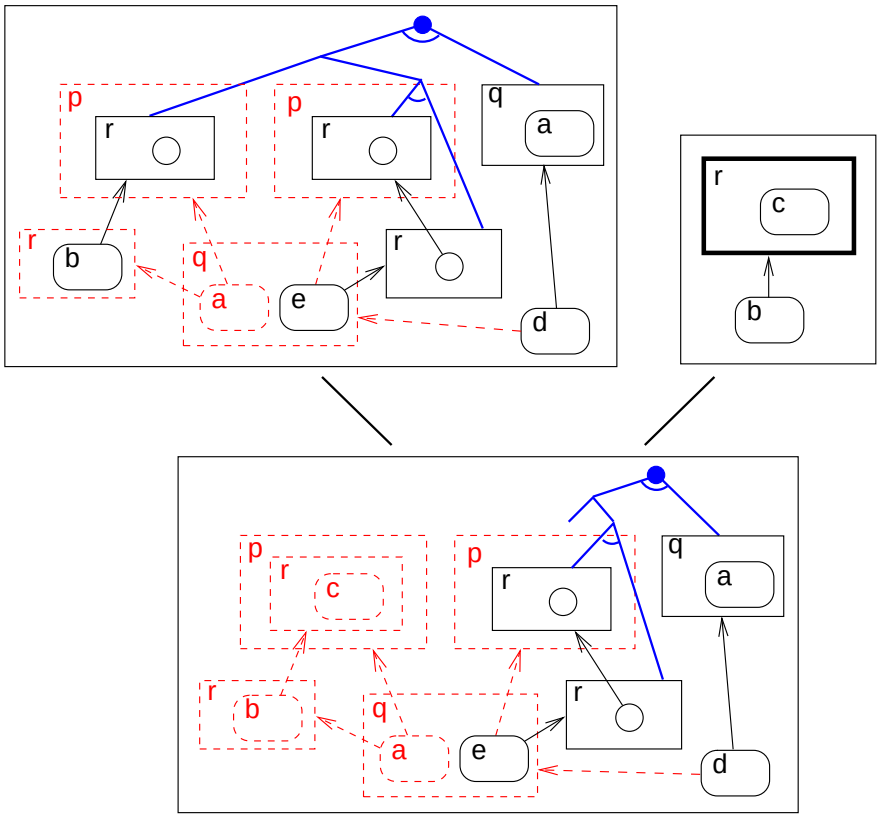


Com a resultat d'aquesta instanciació múltiple, part dels literals visuals del diagrama s'han duplicat i s'ha introduït un node OR. Aquest node OR representa les dues alternatives possibles a l'hora de resoldre la consulta, el que en una procediment deductiu textual equivaldria a dues branques de l'arbre de backtracking.

Escollim una de les dues branques i intentem resoldre una de les alternatives:



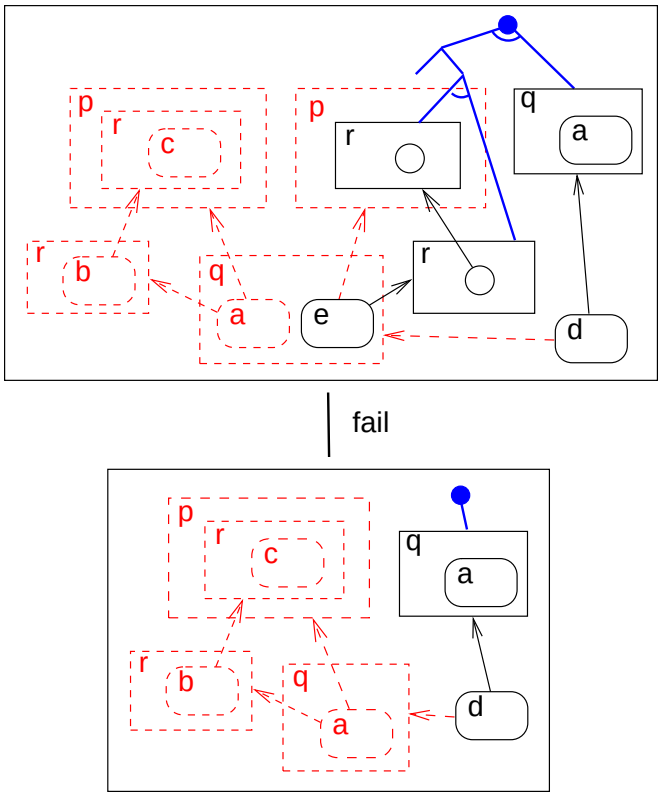
I resolem el darrer literal actiu de la branca del node OR:



En aquest punt hem aconseguit resoldre completament una de les branques del node OR. Per tant cal marcar com a resolt el node OR, ja que encara que l'altra branca falli ja hi ha una possible solució. Per tal de marcar que el node ja està resolt ho fem deixant una branca sense literals.

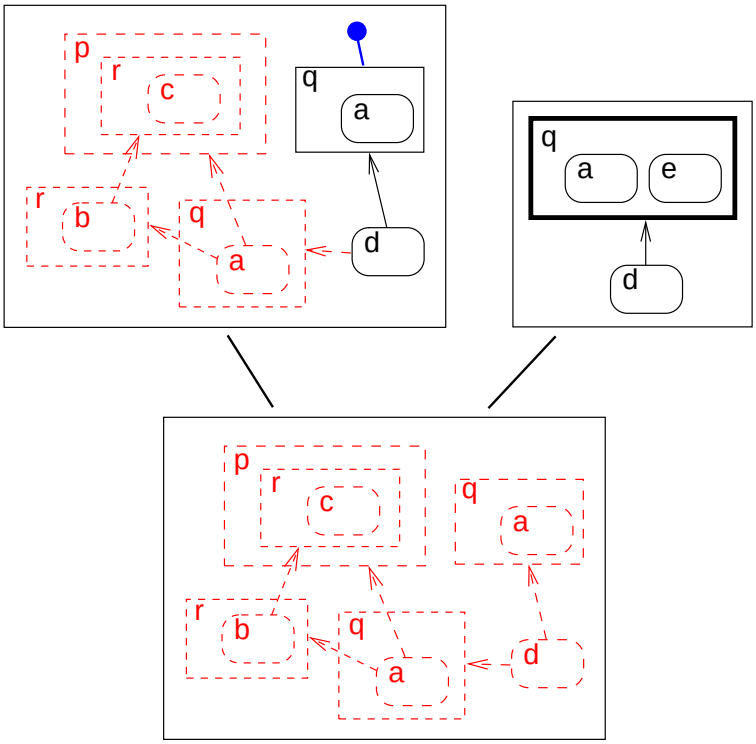
Tot i així encara no podem assegurar que la consulta tingui solució, ja que queda tota una branca del node principal de l'arbre AND-OR per resoldre.

Ens adonem que els literals de la segona branca del node OR no es poden resoldre:

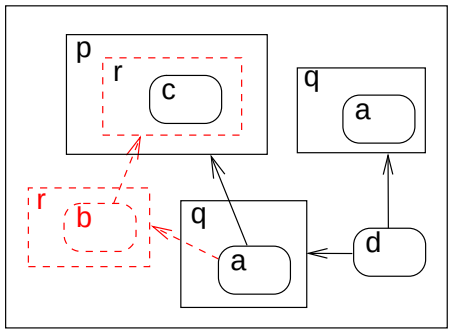


Tots els literals visuals, actius i no actius, associats a aquesta branca de l'arbre AND-OR són esborrats. Aquest pas equivaldria a un pas de *backtracking* en un sistema deductiu textual basat en la regla de resolució.

Resolem el darrer literal visual que queda en diagrama consulta i obtenim una solució:



Finalment obtenim un diagrama resposta ressaltant els predicats i termes visuals presents a la consulta original, o que s'han format per instanciació de variables també presents a la consulta original:



Part III

Aplicacions i implementació

It is essential to move away from the superlativist claims in the literature and toward a recognition that graphics requires readership—and production—skills in the same way that text does, and to provide support in the environment and in the culture for the acquisition and exercise of those skills. Graphical representations, especially those used in notation, far from being intuitively obvious, may require more training from both the originator and the reader to achieve the most effective communication—but may prove richer for expert users. The challenge for the designer is to make good use of secondary notation, "good" in supporting access to relevant information, "good" in avoiding mis-cueing, "good" in matching the secondary notational conventions used by experts in the domain under study, and "good" in nurturing effective graphical readership.

Marian Petre, *Why Looking isn't Always Seeing*, 1995 ([60])

Capítol 6

Esquemes visuals a les bases de dades deductives

En el món de les bases de dades deductives s'utilitza una nomenclatura similar a la utilitzada en la programació lògica, però amb algunes particularitats. El nostre objectiu és aplicar la sintaxi visual proposada en aquesta tesi a la representació visual de l'esquema de la base de dades, tot aprofitant aquesta similitud. En aquest sentit, creiem que la visualització d'esquemes de base de dades és un camp que pot beneficiar-se de l'ús d'una sintaxi visual.

En aquest capítol proposem una variant del llenguatge visual introduït als capítols anteriors per a ser utilitzada en la definició i representació d'esquemes de bases de dades deductives. A la secció 6.1 definim el tipus d'esquemes de bases de dades adreçats, així com la nomenclatura textual utilitzada habitualment en el món de les bases de dades deductives. A continuació, a la secció 6.2 definim el llenguatge visual per a esquemes de bases de dades deductives proposat. A la secció 6.3 desenvolupem un exemple d'esquema visual basat en una base de dades d'un departament de recursos humans. Tot seguit, a la secció 6.4 introduïm noves notacions visuals que ens permetran ampliar la potència expressiva del llenguatge. Finalment, a la secció 6.5 presentem un algorisme de traducció d'esquemes visuals a esquemes textuals.

6.1. Esquemes de bases de dades deductives

En aquesta secció recordem el llenguatge textual i la nomenclatura utilitzats habitualment en les bases de dades deductives (veure [32, 47]) i definim el tipus d'esquemes de bases de dades deductives considerats.

6.1.1. El llenguatge

Considerem un llenguatge de primer ordre amb un conjunt de constants, un conjunt de variables, un conjunt de símbols de predicat i sense símbols de funció.

Un *terme* és un símbol de variable o de constant (assumim que el rang de valors d'un terme és un domini finit). Un *àtom* és de la forma $P(t_1, ..., t_m)$ on P és un símbol de predicat i $t_1, ..., t_m$ són termes. Un *literal* és un àtom positiu o negatiu, és a dir, un literal és o bé un àtom o bé un àtom negat. Un *fet* és una fórmula del tipus $P(t_1, ..., t_m) \leftarrow$ on $P(t1, ..., tm)$ és un àtom instanciat. Quan P és un predicat base, els fets es consideren emmagatzemats explícitament a la base de dades i formen la seva part *extensional* o *extensió de la base de dades*, en contraposició a l'esquema que constitueix la seva part *intensional*.

6.1.2. Predicats base

Cada fet està modelat a l'esquema de la base de dades a través d'un predicat base (o per ser més exactes d'un esquema de predicat base). Els predicats base descriuen la base de dades extensional, i prenen la forma de:

$$B(t_1, ..., t_m) \qquad on \;\; m \geq 0$$

on cada terme $t_1, ..., t_m$ s'interpreta com una variable diferent, modelant així totes les possibles instanciacions que constitueixen la part extensional de la base de dades. Els esquemes de predicats base apareixen només en la seva pròpia definició i en el cos de les regles deductives i d'integritat, on pot estar (parcialment) instanciat.

6.1.3. Predicats derivats i regles deductives

Un *predicat derivat*, també anomenat *vista*, és un esquema que representa informació que no està emmagatzemada explícitament a la base de dades sinó que es pot obtenir aplicant regles deductives. Un predicat derivat és de la forma:

$$D(t_1, ..., t_m) \qquad on \;\; m \geq 0$$

on els termes $t_1, ..., t_m$ són variables diferents. Un predicat derivat apareixerà com a cap de les regles deductives i, probablement, als cossos de les regles deductives i d'integritat, on podrà estar (parcialment) instanciat. Formalment una *regla deductiva* és una fórmula del tipus:

$$D(t_1, ..., t_m) \leftarrow L_1 \wedge ... \wedge L_n \qquad on \;\; m \geq 0 \wedge n \geq 1$$

$D(t_1, ..., t_m)$ és la *conclusió*, és a dir, el predicat que volem definir, i $L_1, ..., L_n$ són literals que en defineixen les condicions, que poden ser:

- Predicats base
- Predicats derivats
- Predicats avaluables: són predicats del sistema, com ara la comparació, o predicats aritmètics que s'apliquen a termes d'altres condicions. Aquests predicats podran ser avaluats sense accedir a la base de dades.

Igual que en les fórmules lògiques (o diagrames) dels capítols anteriors, s'assumeix que totes les variables estan quantificades universalment respecte tota la fórmula (o diagrama). Així les variables que no apareixen en la conclusió s'anomenen *variables existencials* o variables locals. També es pot parlar de cap i cos de la regla, fent referència a la conclusió i les condicions respectivament.

6.1.4. Actualitzacions de la base de dades

A les bases de dades és important poder reflectir les actualitzacions o canvis que es fan a la seva part extensional. Bàsicament podem distingir tres tipus d'actualitzacions: les insercions, els esborrats i les modificacions. Es denoten anteposant al predicat base (o derivat si es permeten actualitzacions de predicats derivats per les peticions d'actualització de vista) les lletres gregues iota (ι), delta (δ) i mu (μ) respectivament i prenen les següents formes:

- insercions

$$\iota B(t_1, ..., t_m) \qquad on \quad m \geq 0$$

- esborrats

$$\delta B(t_1, ..., t_m) \qquad on \quad m \geq 0$$

- modificacions

$$\mu B(t_1, ..., [t_i \rightarrow t_i'], ..., [t_l \rightarrow t_l'], ..., t_m) \qquad on \quad \begin{array}{l} m \geq 0 \\ 1 \leq i \leq m \\ 1 \leq l \leq m \end{array}$$

6.1.5. Restriccions d'integritat i regles d'integritat

Les restriccions d'integritat són condicions que s'han de satisfer en tot moment. S'utilitzen per tal d'especificar els estats no desitjats de la base de dades així com aquells canvis no admesos. Així distingim dos tipus de restriccions d'integritat: d'*estat* (o estàtiques) quan s'han de complir en qualsevol estat de la base de dades, i *dinàmiques*, quan involucren l'evolució entre dos o més estats de la base de dades. Les restriccions d'integritat dinàmiques que especifiquen només una transició entre dos estats successius s'anomenen restriccions d'integritat de *transició*.

Les restriccions d'integritat d'estat es poden expressar utilitzant predicats base i/o derivats. Formalment una *restricció d'integritat d'estat* és una fórmula de primer ordre que la base de dades ha de complir, i està expressada en forma de negació:

$$\leftarrow L_1 \wedge ... \wedge L_n \qquad amb \quad n \geq 1$$

on L_i són literals i s'assumeix que les variables estan quantificades universalment respecte tota la fórmula. Restriccions d'estat més complexes o generals poden

ser transformades en fórmules que s'ajusten a aquest esquema, aplicant primer la transformació descrita per Decker [30], i utilitzant després el procediment descrit per Lloyd i Topor [48]. Les restriccions d'estat són les més habituals.

Les restriccions d'integritat de transició restringeixen les actualitzacions acceptables en la transició entre dos estats consecutius de la base de dades, en funció d'una condició que depèn de l'estat que es vol actualitzar i de les actualitzacions proposades. Formalment una *restricció d'integritat de transició* s'expressa igual que una restricció d'integritat d'estat on els literals poden representar actualitzacions.

6.1.6. Esquemes de bases de dades deductives

Un esquema d'una base de dades deductiva consisteix en tres conjunts:

- Els predicats base.
- Els predicats derivats amb les seves corresponents regles deductives.
- Les restriccions d'integritat amb les seves corresponents regles d'integritat.

Assumim que un predicat és o bé un predicat base o bé un predicat derivat, però no les dues coses alhora. Qualsevol base de dades deductiva es pot definir seguint aquestes normes (veure [10, 23]).

6.2. Esquemes visuals: notació bàsica

En aquesta secció definim el llenguatge visual per a la representació d'esquemes de bases de dades deductives que proposem. Està basat en el llenguatge visual introduït al capítol 4 amb algunes particularitats per tal d'adaptar-lo a les bases de dades deductives.

Qualsevol predicat, ja sigui un predicat base o un predicat derivat, es pot interpretar com un conjunt de n-tuples on n és l'aritat del predicat. Igual que al capítol 4, representem visualment els predicats com a capses i els definim per mitjà de tuples incloses gràficament dins dels predicats. Tot seguit definim els elements del llenguatge, fent especial èmfasi en les diferències respecte la sintaxi visual utilitzada en la resta de la tesi.

Considerarem un llenguatge visual equivalent semànticament al llenguatge textual definit a la secció 6.1. Els termes són variables, constants o funcions:

- *Variables*

  Una variable es representa com un cercle sense cap símbol específic que li doni nom.

- *Constants*

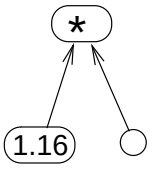
  Les constants també es representen com a cercles amb el nom o símbol que les identifica contingut gràficament dins el cercle o capsa arrodonida.

  Seguim la convenció, habitual a les bases de dades deductives, d'utilitzar noms que comencen per lletres minúscules per símbols de constant i noms que comencen amb lletres majúscules per símbols de predicat. [1]

- *Funcions*

  De vegades és necessari representar un terme com a resultat d'aplicar una funció a un o més termes. En un diagrama, una aplicació d'una funció es representa com una capsa arrodonida amb el símbol de funció a dins i una fletxa provinent dels terme corresponent per a cada argument.

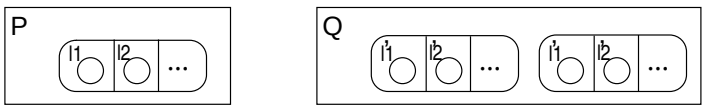
  Per exemple, si volem multiplicar una variable per 1,16 (per tal de calcular l'import amb I.V.A.) ho representaríem com:



- *Literals visuals positius*

  Un *literal visual positiu* és una capsa amb el seu símbol de predicat P en una cantonada (dins del rectangle), i una o vàries tuples d'aritat n gràficament incloses dins la capsa, on n és l'aritat del símbol de predicat P. Variables i constants poden estar incloses directament dins la capsa quan el predicat té aritat 1.

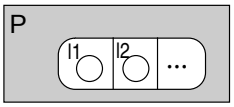
  Aquesta és una diferència important respecte del llenguatge visual introduït al capítol 4, on no hi havien tuples. Una tupla es representa com una capsa arrodonida compartimentada, amb tants de compartiments com l'aritat de la tupla. Cada compartiment conté una etiqueta i un terme, és a dir, una variable o una constant. Una diferència important respecte els llenguatges textuals és que els elements d'una tupla poden etiquetar-se, fent que l'ordre en què apareixen no sigui rellevant. Això fa que la sintaxi sigui més flexible i clara:



[1] Cal notar que als altres capítols no es fa aquesta distinció, sinó que s'utilitzen lletres minúscules i majúscules indistintament.

- *Literals visuals negatius*

  Un *literal visual negatiu* es defineix com un literal visual positiu amb la capsa del predicat ombrejada.

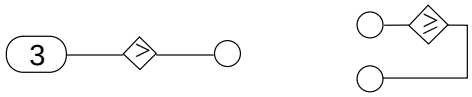


  Un literal negatiu representa la no pertinença de la tupla al conjunt de la intensió del predicat.

- *Literals visuals avaluables*

  Finalment ens cal definir el tercer tipus de literals, que no existien en el llenguatge formalitzat al capítol 4, els *literals visuals avaluables*. S'utilitzen per tal d'indicar la relació entre dos termes, com ara la igualtat o la desigualtat, comparacions, etc. Per exemple, per representar la igualtat entre dos termes els unim mitjançant una línia recta, mentre que per representar la desigualtat marquem la línia amb una creu:

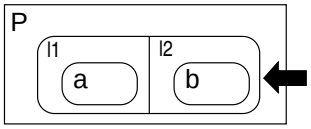
  Altres literals avaluables, com la comparació, es defineixen mitjançant una línia recta que uneix els dos termes amb l'operador de comparació dins d'un rombe que etiqueta la línia:



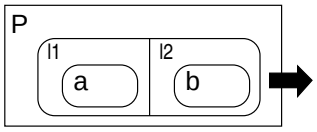
- *Actualitzacions de la base de dades*

  Així mateix també és possible representar visualment les operacions d'actualització de la base de dades, és a dir, les insercions, esborrats i modificacions de valors. En els tres casos aquestes actualitzacions es representen visualment mitjançant una fletxa sòlida que denota el tipus d'operació a realitzar.

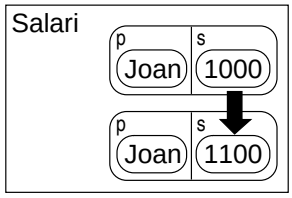
  - *Insercions*: la fletxa negra apuntada envers una tupla indica que aquesta és la tupla que s'insereix a la base de dades.



- *Esborrats*: la fletxa negra provinent d'una tupla i apuntada envers l'exterior de la capsa del predicat indica l'esborrat de la tupla de la base de dades.



- *Modificacions*: en una modificació d'una tupla la fletxa negra identifica el canvi que s'ha produït a la base de dades i uneix la tupla inicial a la tupla final. La fletxa pot unir dos tuples o centrar-se en dos valors concrets de les tuples que són els que es modifiquen .



6.3. Exemple: un departament de recursos humans

En aquesta secció desenvolupem un exemple d'esquema visual de base de dades, corresponent a un departament de recursos humans d'una empresa estructurada de forma jeràrquica. A més de formalitzar aquesta estructura la base de dades ha de recollir la informació referent als treballadors i directors de les diferents unitats, així com totes les entrevistes entre persones que sol·liciten un lloc de treball i les respectives unitats. També recull informació sobre l'estat legal i de la existència de registre criminal de totes les persones. L'objectiu d'aquest exemple no és representar un domini d'aplicació real, sinó un mitjà que mostri les situacions que es poden representar mitjançant el llenguatge visual proposat.

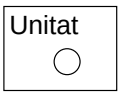
Tal com hem dit amb anterioritat el nostre objectiu es representar visualment l'esquema de la base de dades deductiva. Ens centrarem en els dos components que habitualment tenen definicions més complexes: els *predicats derivats* i les *restriccions d'integritat*. Intentarem obtenir representacions visuals d'aquests components que siguin més intuïtives visualment que els seus equivalents textuals, però mantenint tota la seva formalitat.

6.3.1. Predicats base

Els predicats base visuals es representen utilitzant un literal visual amb una tupla on tots els elements són variables. A continuació veiem els predicats base del nostre exemple i la seva representació visual:

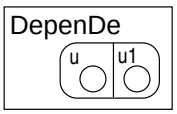
- 'u' és una unitat de l'organització

$$Unitat(u)$$



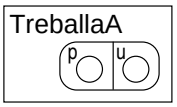
- La unitat 'u' depèn directament de 'u1'

$$DepenDe(u, u1)$$



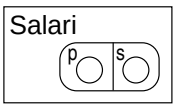
- 'p' és un treballador a la unitat 'u'

$$TreballaA(p, u)$$



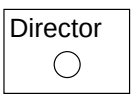
- El treballador 'p' té salari 's'

$$Salari(p, s)$$



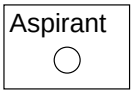
- 'p' és un director

$$Director(p)$$



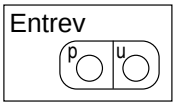
- 'p' és un aspirant a un lloc de treball

$$Aspirant(p)$$



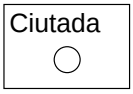
- 'p' té una entrevista a la unitat 'u'

$$Entrev(p, u)$$



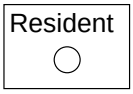
- 'p' és un ciutadà nacional

$$Ciutada(p)$$



- 'p' és un ciutadà estranger amb permís de residència

$$Resident(p)$$



- 'p' té registre criminal

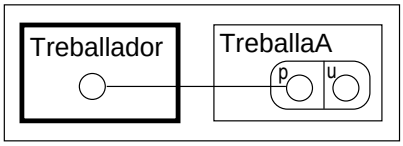
$$RegCrim(p)$$

6.3.2. Predicats derivats i regles deductives

Els *predicats derivats visuals* es defineixen utilitzant *regles deductives visuals*, que també anomenem diagrames. Una regla deductiva visual, igual que el seu equivalent textual, conté un únic predicat visual conclusió i un o més literals de condició. El literal conclusió és sempre un literal visual positiu que conté una i només una tupla, i es distingeix dels altres dibuixant-lo amb línies gruixudes. A més a més, cada diagrama està inclòs gràficament dins d'una capsa que en delimita el seu abast sintàctic.

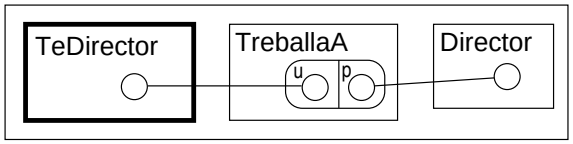
- 'p' és un treballador ja que treballa a la unitat 'u'

$$Treballador(p) \leftarrow TreballaA(p, u)$$



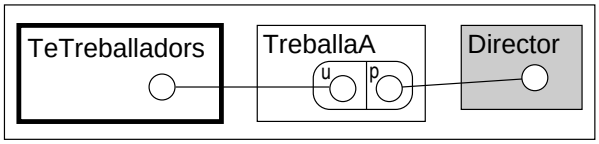
- La unitat 'u' té director ja que 'p' hi treballa i és director

$$TeDirector(u) \leftarrow TreballaA(p, u) \wedge Director(p)$$



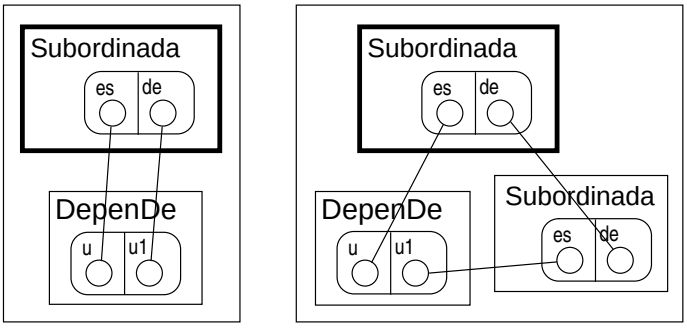
- La unitat 'u' té treballadors no directors ja que 'p' hi treballa i no és director

$$TeTreballadors(u) \leftarrow TreballaA(p, u) \wedge \neg Director(p)$$



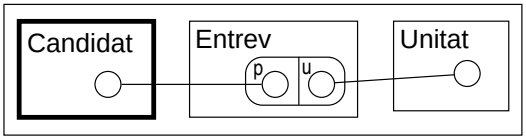
- La unitat 'u' està subordinada a la unitat 'u1' ja que o bé depèn directament de 'u1' o depèn de una altra unitat 'u2' que està subordinada a 'u1'

$$Subordinada(u, u1) \leftarrow DepenDe(u, u1)$$
$$Subordinada(u, u1) \leftarrow DepenDe(u, u2) \wedge Subordinada(u2, u1)$$



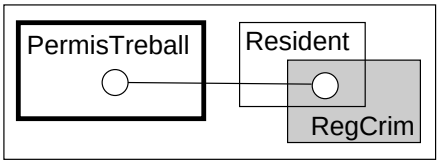
- 'p' és un candidat a un lloc de treball quan té una entrevista amb alguna unitat 'u'

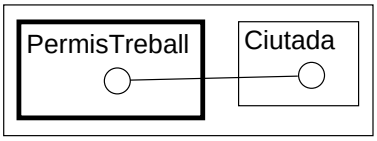
$$Candidat(p) \leftarrow Entrev(p, u) \wedge Unitat(u)$$



- 'p' té permís de treball si és un resident estranger sense registre criminal o bé un ciutadà nacional.

$$PermisTreball(p) \leftarrow Resident(p) \wedge \neg RegCrim(p)$$
$$PermisTreball(p) \leftarrow Ciutada(p)$$



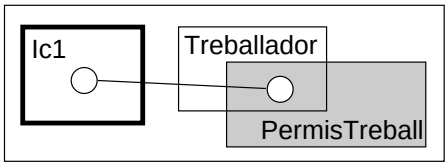


6.3.3. Regles d'integritat d'estat

Tractarem les *restriccions d'integritat d'estat* expressades com a negacions, o sigui, que si algun predicat Ic_n es compleix vol dir que la consistència de la base de dades deductiva s'ha violat. La capsa que correspon al literal conclusió de la regla visual d'integritat representa el conjunt d'elements que viola la restricció d'integritat. La base de dades és consistent quan totes les caixes Ic_n representen conjunts buits.

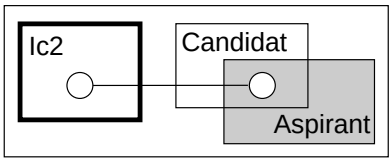
- Tots els treballadors han de tenir permís de treball. És inconsistent que hi hagi treballadors sense permís de treball.

$$Ic_1(p) \leftarrow Treballador(p) \wedge \neg PermisTreball(p)$$



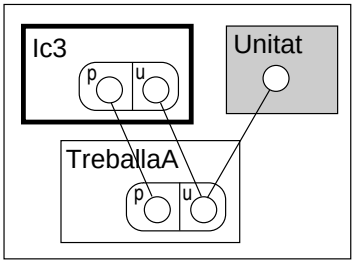
- Tots els candidats han de ser aspirants. No hi pot haver un candidat que no sigui aspirant.

$$Ic_2(p) \leftarrow Candidat(p) \wedge \neg Aspirant(p)$$



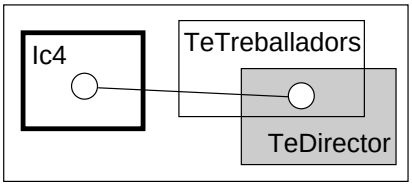
- Els treballadors han de treballar per les unitats. És inconsistent que hi hagi algú treballant per alguna àrea que no es consideri una àrea de la organització.

$$Ic_3(p, u) \leftarrow TreballaA(p, u) \wedge \neg Unitat(u)$$



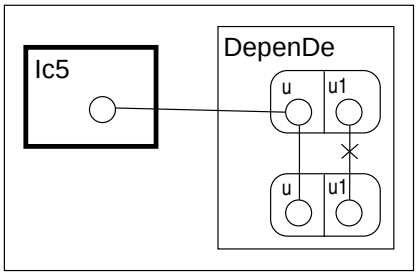
- Cada unitat que tingui treballadors també ha de tenir un director. És inconsistent que hi hagi una unitat amb treballadors assignats però sense director.

$$Ic_4(u) \leftarrow TeTreballadors(u) \wedge \neg TeDirector(u)$$



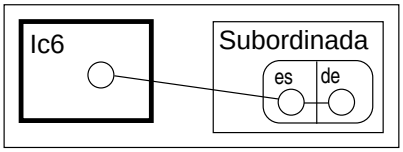
- Una unitat no pot dependre alhora de dues unitats diferents. És inconsistent que una unitat depengui alhora de dues unitats.

$$Ic_5(u) \leftarrow DepenDe(u, u1) \wedge Depen(u, u2) \wedge u1 \neq u2$$



- Una unitat no pot estar subordinada a ella mateixa.

$$Ic_6(u) \leftarrow Subordinada(u, u)$$



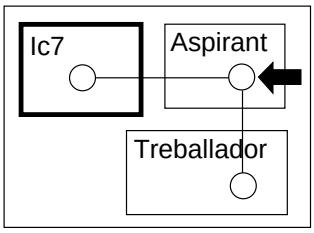
Regles complexes, amb diferents literals que corresponen al mateix predicat (com per exemple la que defineix Ic_6) es representen de forma senzilla utilitzant conjunts. En altres esquemes del món real, més complexos, els avantatges d'utilitzar una sintaxi visual i la metàfora conjuntista creiem que serien encara més evidents.

6.3.4. Regles d'integritat de transició

Les *regles d'estat de transició* les tractarem igual que les regles d'estat i les expressarem com a negacions. A diferència de les regles d'integritat d'estat els literals poden representar actualitzacions.

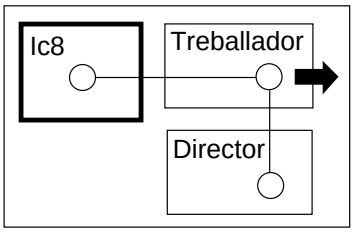
- Un aspirant no es pot inserir si és un treballador. És inconsistent inserir a la base de dades un aspirant si ja és treballador.

$$Ic_7(p) \leftarrow \iota aspirant(p) \wedge treballador(p)$$



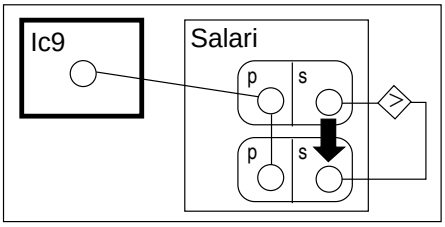
- Els directors no es poden esborrar. És inconsistent esborrar un treballador que és director.

$$Ic_8(p) \leftarrow \delta treballador(p) \wedge director(p)$$



- El salari de un treballador no pot disminuir.

$$Ic_9(p) \leftarrow \mu salari(p, [s_1 \rightarrow s_2]) \wedge s_1 > s_2$$



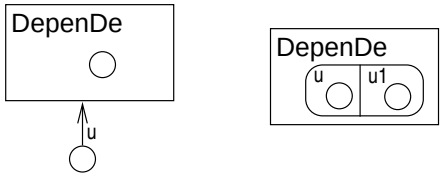
6.4. Esquemes visuals: notació funcional i inclusional

En aquesta secció introduïm dos nous elements del llenguatge visual: la notació funcional i la notació inclusional, que ens permetran ampliar la potència expressiva del llenguatge. Mitjançant la notació funcional podrem representar fàcilment les funcions d'agregació, mentre que la notació inclusional ens permetrà en molts casos simplificar els esquemes visuals.

6.4.1. Notació funcional

A les bases de dades és habitual disposar de *funcions d'agregació*, que ens permeten portar a terme operacions sobre seleccions de les relacions donades. En aquesta secció mostrarem com definir visualment aquestes funcions utilitzant notació visual. Amb aquesta finalitat utilitzarem la notació funcional tractada als capítols 4 i 5 en la representació dels predicats.

Alguns predicats són més fàcils de comprendre com a funcions no deterministes que com a relacions sense cap direccionalitat predefinida. Una funció no determinista és aquella que, donada una entrada, pot tenir un o més resultats. Per exemple, el predicat *DepenDe*, utilitzat a l'exemple d'aquest capítol, és més fàcil d'entendre com una funció que ens dóna les unitats (sortida) que depenen d'una determinada unitat (entrada). Sembla natural anomenar el conjunt de resultats pel nom del predicat i distingir-lo així de l'entrada. Podem comparar les dues notacions alternatives i equivalents als següents diagrames:



El diagrama de l'esquerre utilitza notació funcional. La caixa representa gràficament el conjunt d'unitats que depenen d'una unitat. Textualment equivaldria a la següent fórmula:

$$x \in DepenDe(y)$$

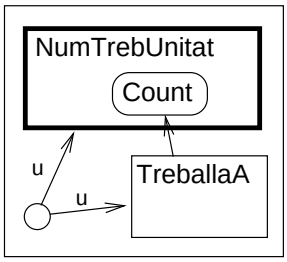
6.4.2. Funcions visuals d'agregació

La notació funcional introduïda a la secció anterior ens permet representar fàcilment les funcions d'agregació de forma molt similar a com es representen les funcions sobre termes. En el llenguatge visual representarem una *funció d'agregació* ($Count$,Min, Max, Sum, etc.) com una funció que té per argument una caixa, és a dir, un conjunt d'elements o tuples. Gràficament es representen com a funcions tals que el seu argument és una capsa (o predicat).

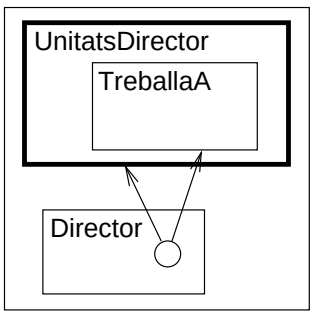
Per exemple, volem calcular el nombre d'empleats de cada unitat i definir un predicat derivat nou $NumTrebUnitat$ de la següent forma:

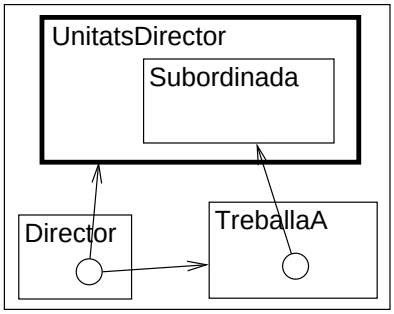
$$NumTrebUnitat(x, y) \leftarrow Unitat(x) \ \wedge \ Count(\{w|TreballaA(w, x)\}, y)$$

que podem representar mitjançant el següent diagrama equivalent:

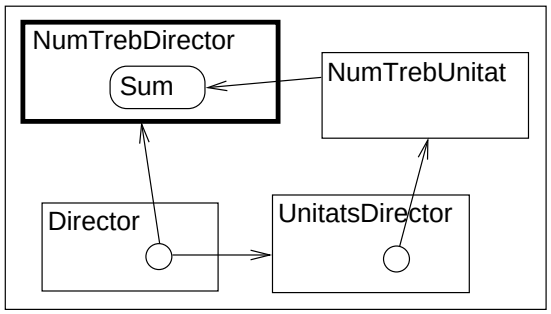


Un altre exemple de la utilització de funcions d'agregació seria definir un predicat derivat que ens doni el número de treballadors que depenen d'un director. Primer definim un predicat derivat que ens dóna les unitats que depenen d'un director, que són la unitat on treballa el director i totes les seves subordinades:





I després definim el predicat $NumTrebDirector$:



Utilitzem el predicat $NumTrebUnitat$ definit anteriorment per tal de calcular el nombre de treballadors de cada unitat dirigida pel director en qüestió. Després utilitzem una altra funció d'agregació, *Sum*, per a calcular el total de treballadors.

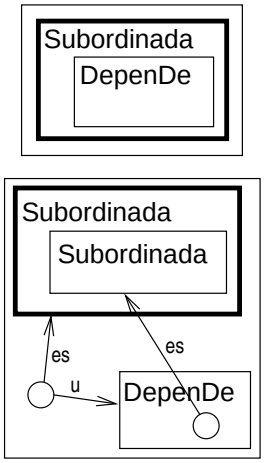
6.4.3. Notació alternativa amb inclusions

Una altra característica del llenguatge visual utilitzada als capítols 4 i 5 i que fins ara no havíem utilitzat en els esquemes visuals és la inclusió de capses (inclusió de conjunts). La representació funcional dels predicats introduïda a la secció anterior afavoreix l'ús de la inclusió de conjunts. D'igual forma que en el llenguatge visual declaratiu restringirem l'ús de la inclusió de conjunts als literals conclusió, per tal de mantenir la complexitat del llenguatge visual dins els límits dels esquemes visuals que volem adreçar.

Per exemple, recordem les regles deductives utilitzades en la definició del predicat derivat *Subordinada*:

$$Subordinada(u, u1) \leftarrow DepenDe(u, u1)$$
$$Subordinada(u, u1) \leftarrow DepenDe(u, u2) \wedge Subordinada(u2, u1)$$

Aquestes dues regles segueixen l'esquema habitual de la recursivitat utilitzat en diverses ocasions en els capítols anteriors. Aquestes regles es poden representar de la següent forma:



Aquests diagrames són equivalents als presentats a la secció 6.3 d'aquest capítol però utilitzant ara un estil funcional. El llenguatge visual proposat permet combinar apropaments relacionals amb apropament funcionals segons l'objectiu i les preferències de l'usuari.

6.5. Traducció d'esquemes visuals a esquemes textuals

En aquesta secció presentem les pautes per traduir el llenguatge visual presentat a les seccions anteriors a la sintaxi textual habitual de les bases de dades deductives, és a dir, a lògica de primer ordre. Disposar d'un algorisme de traducció a sintaxi textual és important a l'hora de considerar l'aplicació dels esquemes

visuals al món real, ja que tots els sistemes gestors actuals de bases de dades utilitzen una sintaxi estàndard textual. Primer introduïm les regles de traducció i a continuació mostrem alguns exemples de traduccions.

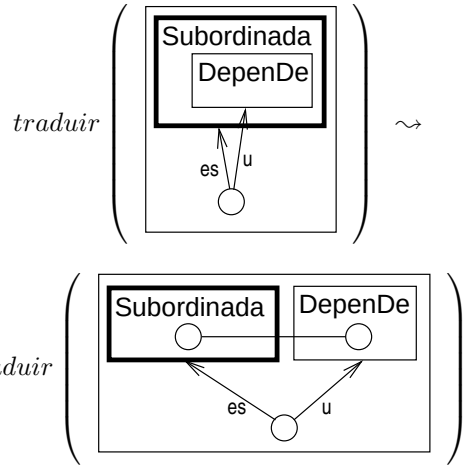
6.5.1. Regles de traducció

Una de les diferències entre la notació visual i la notació textual és l'absència d'un ordre explícit dels arguments en la primera. Abans de la traducció caldrà establir la correspondència entre les etiquetes dels termes dels literals visuals i l'ordre dels termes en els literals visuals. En el nostre exemple només hi ha tres predicats amb més d'un terme. Aquesta correspondència s'estableix de la següent forma:

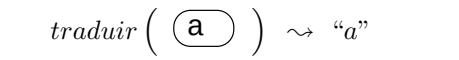
$DepenDe(\mathrm{u} : x, \mathrm{u1} : y) \longmapsto DepenDe(x, y)$
$TreballaA(\mathrm{p} : x, \mathrm{u} : y) \longmapsto TreballaA(x, y)$
$Subordinada(\mathrm{es} : x, \mathrm{de} : y) \longmapsto Subordinada(x, y)$

Com que probablement estarem utilitzant algun editor de diagrames dirigit per la sintaxi (veure capítol 7) podem assumir que l'entrada de l'algorisme és sempre un esquema visual de base de dades sintàcticament correcte. Cada regla visual deductiva correspon a una regla deductiva textual, la traducció de la qual es porta a terme aplicant els següents passos:

1. *Eliminació de la inclusió de capses*: si hi ha inclusions de capses el diagrama es transforma en un d'equivalent on les inclusions s'hauran substituït mitjançant l'ús de variables.



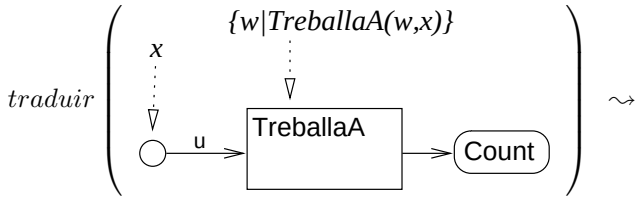
2. Cada *constant* es tradueix com el símbol inclòs dins la capsa rodona (Assumim que els símbols utilitzats dins les capses rodones són símbols textuals).



3. Cada *variable* es tradueix com a una variable textual. Caldrà anar assignant noms diferents a cadascuna de les variables.

$$traduir\left(\ \bigcirc\ \right) \rightsquigarrow \text{“}x, y, ...\text{”}$$

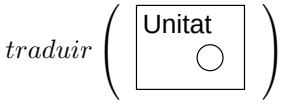
4. Les *funcions d'agregació* es tradueixen com l'aplicació del símbol de funció a la fórmula textual del conjunt denotat pel seu argument.

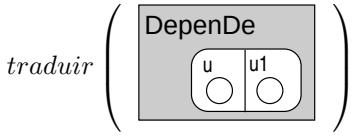


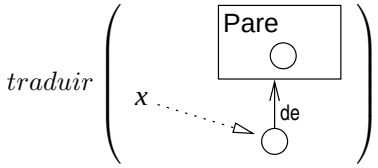
$$\text{“}Count(\{w|TreballaA(w,x)\})\text{”}$$

5. *Literals Visuals*: Cada inclusió d'una *tupla* a una *capsa* es tradueix com un literal textual tal que:

   - El seu símbol de predicat és el símbol que hi ha indicat dins la capsa corresponent al literal que estem traduint.
   - L'aritat del símbol de predicat és l'aritat de la tupla inclosa a la caixa més el nombre de fletxes que li arriben[2]. Pel que ens ocupa ara, les variables o constants es poden considerar com a tuples d'aritat 1.
   - El literal serà un àtom negat si i només si la capsa està ombrejada.
   - Els termes del predicat són les traduccions textuals dels termes visuals (constants i variables) de la tupla i dels termes visuals associats a les fletxes que arriben a la capsa.







[2]Després d'aplicar les regles anteriors no hi pot haver cap capsa sense una tupla (o una variable o constant) inclosa.

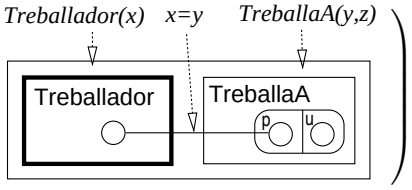
6. *Literals visuals avaluables*: cada literal visual avaluable es tradueix com el literal textual equivalent. Per exemple la igualtat i la desigualtat de variables es traduirà com '$x = y$' i '$x \neq y$' respectivament, on 'x' i 'y' són les traduccions textuals de les dues variables obtingudes al pas núm. 2.

$$traduir\left(\ \right) \rightsquigarrow \text{“}x = y\text{”}$$

$$traduir\left(\ \right) \rightsquigarrow \text{“}x \neq y\text{”}$$

7. *Regles deductives visuals* (diagrames): cada regla deductiva visual es tradueix com una regla deductiva textual equivalent tal que:

   - La conclusió de la regla és la traducció corresponent a la capsa conclusió (dibuixada amb línies gruixudes), que s'ha obtingut al pas núm. 5.
   - Les condicions de la regla són les traduccions de totes les capses de condició (tal com s'han obtingut al pas núm. 5) més les traduccions dels literals visuals avaluables (tal com s'han obtingut al pas núm. 6), units per connectives $\wedge$.

$traduir$ 

$\rightsquigarrow$

$$\text{“}Treballador(x) \leftarrow TreballaA(y, z) \wedge x = y\text{”}$$

   Les regles d'integritat es tracten com regles deductives visuals normals.

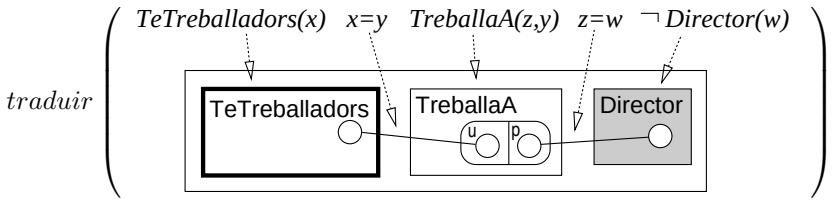
8. Opcionalment els *literals textuals* que denoten la igualtat de variables es poden substituir mitjançant la substitució de les variables iguals per un únic nom nou.

$$\text{“}Treballador(x) \leftarrow TreballaA(y, z) \wedge x = y\text{”} \rightsquigarrow$$

$$\text{“}Treballador(w) \leftarrow TreballaA(w, z)\text{”}$$

6.5.2. Exemples de traduccions de diagrames a fórmules textuals

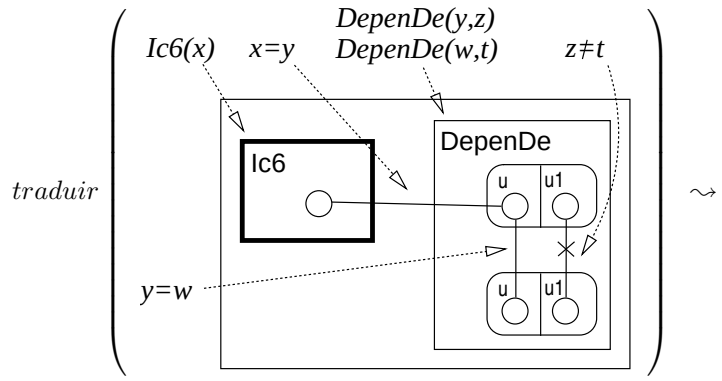
Primer traduïm la regla deductiva que defineix el predicat $TeTreballadors$ com les unitats que tenen algun treballador que no és director:



"$TeTreballadors(x) \leftarrow TreballaA(y,z) \wedge \neg Director(w) \wedge x = y \wedge z = w$" $\rightsquigarrow$

"$TeTreballadors(u) \leftarrow TreballaA(t,u) \wedge \neg Director(t)$"

A continuació traduïm una regla d'integritat: una unitat no pot dependre de dos unitats diferents.

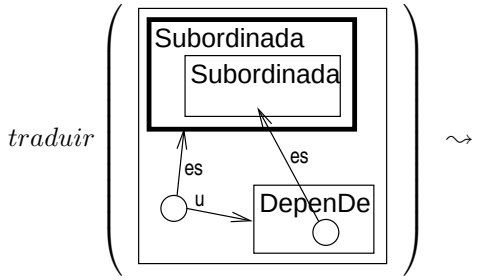


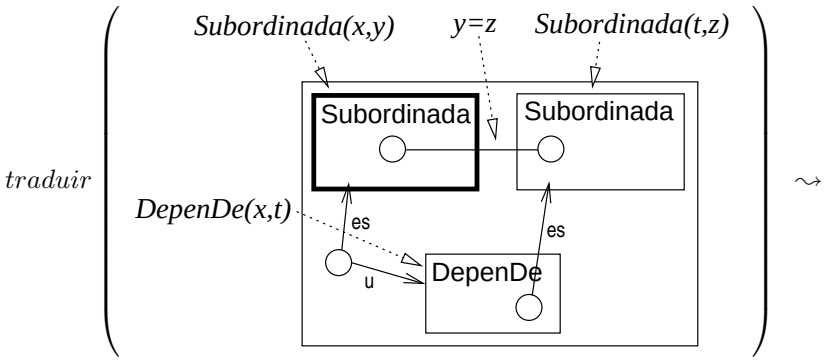
"$Ic_6(x) \leftarrow DepenDe(y,z) \wedge DepenDe(w,t) \wedge x = y \wedge y = w \wedge z \neq t$" $\rightsquigarrow$

"$Ic_6(v) \leftarrow DepenDe(v,z) \wedge DepenDe(v,t) \wedge z \neq t$"

Noteu que en aquest exemple la capsa *DepenDe* conté dues tuples i per tant el resultat de la seva traducció són dos literals textuals diferents.

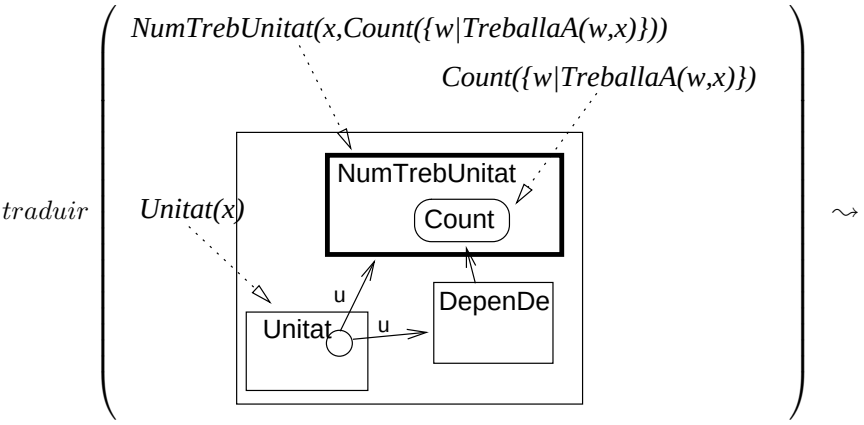
Ara traduïm el cas recursiu de la definició de *Subordinada* segons la definició de la secció 6.4.3, utilitzant la notació alternativa amb inclusions.





"$Subordinada(x, y) \leftarrow DepenDe(x, t) \wedge Subordinada(t, z) \wedge y = z$" $\rightsquigarrow$
"$Subordinada(x, w) \leftarrow DepenDe(x, t) \wedge Subordinada(t, w)$"

Finalment traduïm un diagrama d'exemple on s'ha utilitzat una funció d'agregació:



"$NumTrebUnitat(x, y) \leftarrow Unitat(x) \wedge Count(\{w|TreballaA(w, x)\}, y)$"

Capítol 7

Un entorn de programació visual lògica

En aquest capítol definim les directrius per tal de dissenyar un editor de diagrames dirigit per la sintaxi i orientat al tipus de llenguatges visuals que proposem en aquesta tesi. L'objectiu és mostrar la viabilitat d'un entorn de programació adaptat als llenguatges visuals proposats. Aquest aspecte és fonamental si, almenys en certs àmbits, es vol considerar els llenguatges visuals com una alternativa real a la programació textual.

Primer de tot veurem com l'edició dirigida per la sintaxi ens permet eliminar la necessitat de construir un analitzador diagramàtic (*diagrammatic parser*) i alhora proporciona a l'usuari una eina per a crear i editar diagrames de forma senzilla. Tot seguit, a la secció 7.2, formalitzarem el nostre llenguatge visual utilitzant una gramàtica generativa (al capítol 4 vam definir la sintaxi del llenguatge de forma matemàtica). En utilitzar una gramàtica generativa podrem determinar un conjunt d'operacions d'edició de diagrames d'alt nivell que ens permetin crear i editar diagrames d'acord amb les regles sintàctiques del llenguatge. Finalment a la secció 7.4 mostrem un prototip d'editor dirigit per la sintaxi i construït per tal de provar aquestes idees.

7.1. Edició dirigida per la sintaxi

Mentre que l'anàlisi sintàctica (*parsing*) de llenguatges textuals és un tema molt estudiat, els llenguatges visuals ofereixen un camp nou, poc estudiat i que presenta noves dificultats. Les tècniques d'anàlisi de llenguatges visuals estan subdesenvolupades —tot i que hi ha diversos estudis sobre aquest aspecte dels llenguatges visuals (Lakin [46])— i alguns estudis recents (Marriott i Meyer [50, 51]) ens fan pensar que la complexitat intrínseca de l'anàlisi sintàctica de llenguatges visuals és molt més complexa que en el cas textual.

Per aquestes raons, si bé no cal menysprear la recerca enfocada a l'anàlisi sintàctica de llenguatges visuals, considerem molt important desenvolupar tèc-

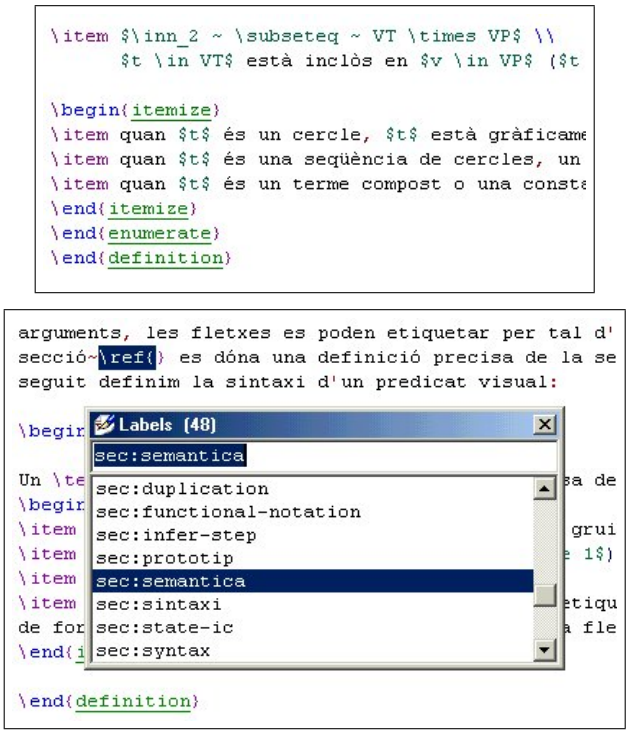


Figura 7.1: WinEdt: editor de LaTeXdirigit per la sintaxi

niques d'edició dirigides per la sintaxi per tal de facilitar l'ús dels llenguatges visuals, i en concret del que nosaltres proposem.

La majoria d'entorns de programació textual més populars actualment incorporen alguna mena d'editor dirigit per la sintaxi que contenen, entre d'altres, algunes de les següents característiques:

- Detecció de paraules clau del llenguatge —variables, constants, etc.— marcant-les de forma que el codi sigui més entenedor i fàcil de llegir. Una forma habitual de marcar les diferents partícules sintàctiques és mitjançant diferents colors, negretes, diferents fonts, etc.
- Eines per a completar automàticament sentències de codi. Per exemple proposar una llista de tipus predefinits a l'escriure una expressió.
- Comprovació de la sintaxi a mesura que l'usuari escriu el codi. Normalment es porta a terme una anàlisi local de la línia que s'està escrivint. Es poden detectar errors bàsics com ara assignacions incorrectes, mancança d'un operador o una paraula clau, etc.
- Utilització de plantilles predefinides amb preservació d'aquestes estructures. Tot allò que s'ha creat a través d'una plantilla no es pot modificar,

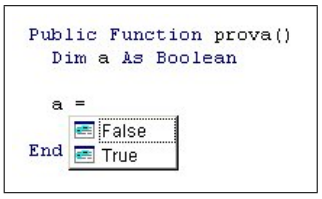


Figura 7.2: Editor de Microsoft Visual Basic

sinó és a través de les regles que la plantilla proporciona.

- Integració visual de l'execució del codi en mode de prova (*debugging*).

La mateixa tecnologia visual que ha nascut amb els llenguatges visuals ha influït enormement en els entorns de programació (Ambler i Burnett [8]). A les figures 7.1 i 7.2 hi trobem dos exemples d'editors dirigits per la sintaxi. El primer és WinEdt, un editor amb un mòdul especialitzat per LaTeX (l'exemple s'ha extret del codi LaTeX d'aquesta tesi). Veiem com totes les paraules clau del llenguatge estan ressaltades amb colors i com l'editor ens proposa les diferents marques o etiquetes a l'hora d'inserir una referència. La figura 7.2 ens mostra com l'editor de Microsoft Visual Basic ens proposa completar una expressió en funció del tipus de dades de la part esquerra. Hi ha altres estudis dins l'àmbit dels llenguatges visuals amb objectius similars (Calder [21], Meyer et al. [53], Minas [54], Serrano [68]).

El cicle d'escriure, analitzar, compilar i provar dels primers llenguatges de programació s'ha trencat. Si l'editor és la única forma d'alimentar el codi al compilador i aquest pot garantir que certs errors sintàctics ja han estat filtrats, això permet simplificar l'analitzador del compilador. De fet l'analitzador no tindrà com a entrada un arxiu (o varis) ASCII, sinó una estructura sintàctica o arbre sintàctic on certs elements ja estaran analitzats. En això ens basarem a l'hora de dissenyar el nostre entorn de programació.

7.2. Formalització del llenguatge mitjançant una gramàtica generativa CMG

A continuació definirem la sintaxi del llenguatge visual mitjançant una *Constraint Multiset Grammar* (CMG) lliure de context. Una gramàtica CMG és una gramàtica generativa desenvolupada per Marriott [49] basada en regles de producció i que permet descriure llenguatges visuals utilitzant vàries relacions espacials entre els elements sintàctics. Mentre que en una *Context Free Grammar* (CFG) s'utilitza de forma implícita la relació espacial d'adjacència (o seqüencialitat en la disposició dels elements), en una CMG es pot determinar quin és el conjunt de relacions que caracteritzaran el nostre llenguatge.

Símbol Terminal	**Repr. Visual**	**Punts de Connexió**
<rectangle>		perímetre rectangle
<rectangle–negreta>		perímetre rectangle
<cercle–terme>		perímetre cercle
<cercle–variable>		perímetre cercle
<fletxa>		Els dos extrems de la fletxa (punta i cua) i el punt mig: $\{p, c, m\}$
<linia>		Els dos extrems de la línia: $\{a, b\}$
<nom>	p,q,r,.. a,b,c,.. f,g,h,..	

Taula 7.1: Símbols terminals

Cada regla de producció d'una CMG té a la seva part dreta un multiconjunt de símbols terminals i no terminals que corresponen a elements diagramàtics. A la part dreta de la gramàtica s'utilitzen els atributs d'aquests elements per tal d'especificar les relacions espacials que s'estableixen entre ells. A la taula 7.1 podem veure els símbols terminals del nostre llenguatge visual.

El nostre llenguatge visual declaratiu es basa en les connexions entre els elements diagramàtics i en la seva inclusió gràfica. Per tal de capturar aquestes dues relacions espacials i simplificar la gramàtica hem definit operadors i atributs específics per aquestes dues relacions, tal com s'observa a les taules 7.3 i 7.2 respectivament.

Cada element diagramàtic té un conjunt de possibles punts de connexió, i hi ha un operador especial ($\leftrightarrow$) per tal d'indicar que dos elements estan connectats. Cada terme està representat visualment mitjançant un graf acíclic dirigit (DAG), però per treballar amb la relació d'inclusió gràfica hem de focalitzar l'atenció en l'element arrel que representa topològicament el terme.

S'assumeix que totes les relacions gràfiques presents en un diagrama han de quedar reflectides en la gramàtica. L'àmbit sintàctic està compost per tots els elements continguts dins d'un rectangle. El rectangle en sí no està contemplat per aquesta gramàtica, però s'hi suposa de forma implícita.

A continuació definim formalment mitjançant regles CMG la sintaxi del llenguatge introduït formalment (però no mitjançant una gramàtica generativa) al

Atribut	**Explicació**
root	element diagramàtic que representa l'element o conjunt d'elements associat al terme
ulc	*upper left corner*: cantonada superior esquerra
con	conjunt de punts de connexió

Taula 7.2: Atributs utilitzats.

Operadors	**Explicació**
$A \subseteq B$	A està gràficament inclòs a B
$A \leftrightarrow B$	A i B estan conectats ($A.con \cap B.con \neq \emptyset$)
$A \leftrightarrow B[s]$	A està connectat al punt s de B ($s \in A.con$)

Taula 7.3: Operadors específics.

capítol 4:

Termes visuals

Un terme visual és o bé una variable (cercle o seqüència de cercles):

[**1**] T :<terme> ::= C :<cercle–variable>

[**2**]
$$\begin{array}{lll} T:\text{<terme>} & ::= & C_i:\text{<cercle–variable>}*, \\ & & L_j:\text{<linia>}* \\ & on & \\ & & 1 \leq i \leq n \\ & & 0 \leq j \leq n-1 \\ & & n \geq 1 \\ & & \forall_{k\in[1..n-1]} \begin{array}{l}(C_k \longleftrightarrow L_k[a]) \wedge \\ (C_{k+1} \longleftrightarrow L_k[b])\end{array} \\ & & T.con = \bigcup_{\forall i} C_i.con \end{array}$$

o bé una constant o una funció:

[**3**]
$$\begin{array}{lll} T:\text{<terme>} & ::= & C:\text{<cercle–terme>}, \\ & & N:\text{<nom>}, \\ & & A_i:\text{<argument>}* \\ & on & \\ & & C.ulc = N.ulc \\ & & 1 \leq i \leq n \\ & & n \geq 0 \\ & & \forall j \quad A_i.con \leftrightarrow C.con \\ & & T.con = C.con \end{array}$$

Predicats visuals

Un predicat visual és una capsa amb un nom i, opcionalment, un o més arguments:

[**4**]
$$\begin{array}{lll} T:\text{<predicat>} & ::= & R:\text{<rectangle>}, \\ & & N:\text{<nom>}, \\ & & A_i:\text{<argument>}* \\ & on & \\ & & R.ulc = N.ulc \\ & & 1 \leq i \leq n \\ & & n \geq 0 \\ & & \forall j \quad A_i.con \leftrightarrow C.con \\ & & T.con = R.con \end{array}$$

Alternativament un predicat visual és un "predicat definició" si la capsa està dibuixada amb línies gruixudes (en negreta):

[**5**]
$$\begin{array}{lll} T:\text{<predicat-definicio>} & ::= & R:\text{<rectangle-negreta>}, \\ & & N:\text{<nom>}, \\ & & A_i:\text{<argument>}* \\ & on & \\ & & R.ulc = N.ulc \\ & & 1 \leq i \leq n \\ & & n \geq 0 \\ & & \forall j \quad A_i.con \leftrightarrow C.con \\ & & T.con = R.con \end{array}$$

Arguments de termes i predicats visuals

Un argument, ja sigui d'un terme o d'un predicat visual, sempre és un terme visual amb una fletxa cap a l'element diagramàtic del qual n'és argument:

[**6**]
$$\begin{array}{lll} T:\text{<argument>} & ::= & T:\text{<terme>}, \\ & & F:\text{<fletxa>} \\ & on & \\ & & T \leftrightarrow F[c] \\ & & T.con = F[p] \end{array}$$

Literals

Distingim dos tipus de literals: "literals condició" i "literals definició". Un literal condició és una inclusió entre un predicat i un terme:

[**7**]
$$\begin{array}{lll} T:\text{<literal-condicio>} & ::= & P:\text{<predicat>}, \\ & & T:\text{<terme>} \\ & on & \\ & & T \subseteq P \end{array}$$

Un literal definició és aquell que està format a partir d'un predicat definició:

$$[\mathbf{8}] \quad \begin{array}{lcl} T : \text{<literal-definicio>} & ::= & P : \text{<predicat-definicio>}, \\ & & T_i : \text{<terme>}, \\ & & V_j : \text{<predicat>} \\ & on & \\ & & T \subseteq P \\ & & 0 \leq i \leq n \\ & & 0 \leq j \leq m \\ & & n + m \geq 1 \end{array}$$

Diagrames

Un diagrama definició, o simplement diagrama, consta d'un sol literal definició i, opcionalment, d'n literals condició:

$$[\mathbf{9}] \quad \begin{array}{lcl} T : \text{<diagrama>} & ::= & D : \text{<literal-definicio>}, \\ & & L_i : \text{<literal-condicio>} * \\ & on & \\ & & 0 \leq i \leq n \\ & & n \geq 0 \end{array}$$

Un diagrama consulta, o simplement una consulta, consta d'un o més literals condició:

$$[\mathbf{10}] \quad \begin{array}{lcl} T : \text{<consulta>} & ::= & L_i : \text{<literal-condicio>} * \\ & on & \\ & & 0 \leq i \leq n \\ & & n \geq 0 \end{array}$$

7.3. Operacions o primitives d'edició de diagrames

Distingim dos tipus d'operacions gràfiques a tenir en compte en un editor de diagrames dirigit per la sintaxi: "operacions de creació d'elements diagramàtics" i "operacions d'edició". A continuació detallarem les diferents primitives que un editor de diagrames haurà de tenir.

Operacions de creació d'elements diagramàtics

Les operacions s'extreuen directament de les regles de la gramàtica de l'anterior secció:

- Crear una variable (regla 1)
- Afegir un nou cercle a una seqüència de cercles (o a un cercle sol) (regla 2)

- Crear una constant o arrel d'una funció (regla 3)
- Crear un predicat (regla 4)
- Crear un predicat definició (regla 5)
- Afegir un argument a una constant, un predicat o un predicat definició (regla 6)

La forma d'implementar aquestes operacions bàsiques de creació d'elements ha de garantir que el resultat serà sintàcticament correcte. Donat que cada operació correspon en gran mesura a una regla de la gramàtica aquesta comprovació no és complexa.

Operacions bàsiques d'edició

Són les operacions habituals en qualsevol editor gràfic:

- moure elements
- modificar tamany
- esborrar elements

La implementació d'aquestes operacions ha de seguir garantint que el resultat serà sintàcticament correcte. Per exemple, si s'esborra un predicat o una funció també caldrà esborrar els arguments, o com a mínim esborrar les fletxes, però deixant els termes. A més caldrà tenir en compte altres detalls, com ara no permetre que dos elements estiguin superposats i per tant puguin donar lloc a un dubte en el moment de dir si un està inclòs en un altre o no.

Construint un editor d'aquesta forma, les comprovacions sintàctiques a fer per veure si el diagrama és sintàcticament correcte es redueixen a:

- Identificar els diferents literals corresponents a les regles 7 i 8. Equival a buscar totes les inclusions gràfiques entre predicats i termes.
- Comprovar que no hi han termes ni predicats que no pertanyin a cap literal, ja que això voldria dir que aquests termes no "encaixen" en la definició gramatical del llenguatge. És a dir, que el diagrama és incorrecte sintàcticament.
- I finalment comprovar que efectivament es tracta d'un diagrama definició (regla 9) o d'un diagrama consulta (regla 10).

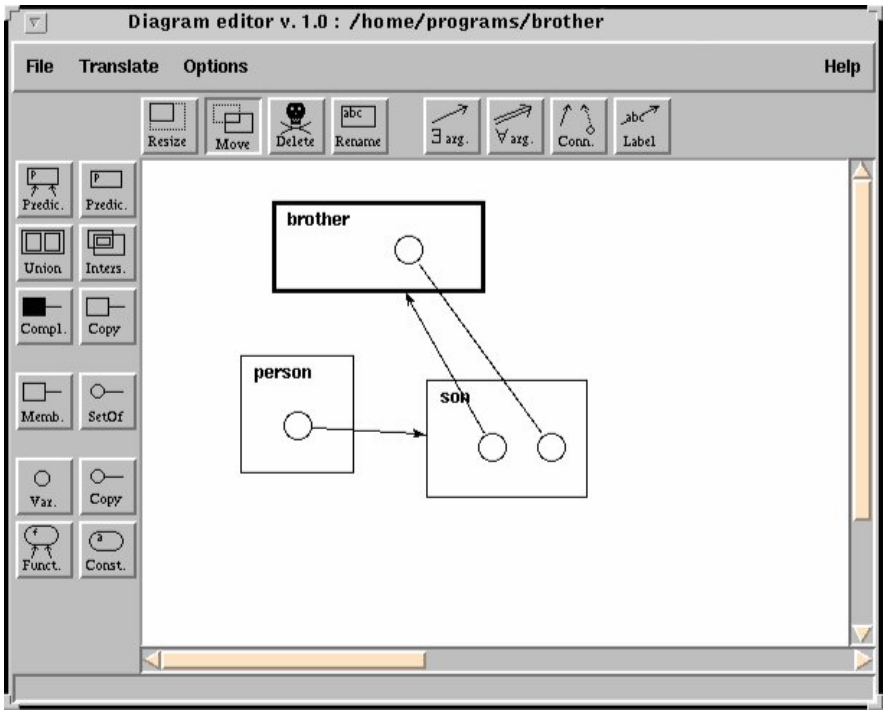


Figura 7.3: Una mostra de l'editor dirigit per la sintaxi

7.4. El prototipus

S'ha dissenyat i implementat un prototip d'editor de diagrames dirigit per la sintaxi mitjançant Tcl/Tk sota un entorn X-Windows / Solaris. Aquest editor està adaptat a la sintaxi visual per a la lògica introduïda al capítol 2. A la figura 7.3 podem veure una instantània de l'editor de diagrames amb un dels exemples de la definició del predicat *germa* del capítol 2. A la dreta podem veure tots els botons corresponents a les operacions de creació d'elements diagramàtics, mentre que a la part superior hi trobem les operacions bàsiques d'edició (esborrar, moure, etc.) i les operacions de connexió d'elements.

Aquesta implementació ens ha servit per a constatar que és possible implementar un editor de diagrames dirigit per la sintaxi, de forma que garanteixi el grau de correctesa sintàctica dels diagrames necessari per a eliminar la necessitat d'un analitzador sintàctic. Queda pendent implementar el llenguatge visual proposat als capítols 4 i 5 per tal de poder portar a terme un estudi empíric que adreci la usabilitat del llenguatge proposat.

Part IV

Conclusions

We are entirely convinced the future is "visual". We believe that in the next few years many more of our daily technical and scientific chores will be carried out visually, and graphical facilities will be far better and cheaper than today's. These languages and approaches we shall be using in doing so will not be merely iconic in nature (e.g. using the picture of a trash can to denote gargabage collection), but inherently diagrammatic in a conceptual way, perhaps also three-dimensional and/or animated. They will be designed to encourage visual modes of thinking when tackling systems of ever-increasing complexity, and will exploit and extend the use of our own wonderful visual system in many of our intellectual activities.

David Harel, *On Visual Formalisms*, 1998. ([41])

Capítol 8

Conclusions

El treball central d'aquesta tesi ha consistit en dissenyar una nova sintaxi visual per a la lògica de primer ordre, i en concret per a la programació lògica, inspirida i dirigida per la semàntica dels predicats, és a dir, basada en la interpretació dels predicats com a conjunts de tuples. Per aconseguir-ho hem proposat una representació visual on els dos punts clau són: 1) representar gràficament els predicats utilitzant les seves abstraccions conjuntistes o conjunts extensió i 2) la utilització de la inclusió gràfica per a denotar la implicació lògica. Aquesta representació combinada amb grafs per a representar els termes estructurats ens ha permès obtenir una sintaxi propera a la semàntica matemàtica de la lògica, contràriament al que succeeix als llenguatges textuals convencionals. Aquest apropament no havia estat utilitzat abans en cap llenguatge visual de programació declarativa i ha estat molt ben acollit a la comunitat de programació visual, com ho demostren les diferents publicacions en revistes i congresos internacionals a què ha donat lloc (veure pàg. 123).

El llenguatge visual proposat és completament visual, és a dir, tota la definició de la sintaxi, la semàntica denotacional i la semàntica operacional s'ha fet directament sobre la sintaxi visual, sense utilitzar cap notació textual intermèdia. Aquest apropament innovador ens ha suposat un repte a l'hora de poder treballar en igualtat de condicions que amb una sintaxi textual convencional. La majoria dels mètodes formals per a definir un llenguatge, les notacions i convencions i les tècniques d'implementació que existeixen estan orientats a les sintaxis textuals. Ha estat necessari adaptar formalismes i mètodes ben coneguts al nostre llenguatge visual, i en alguns casos ha calgut utilitzar nous mecanismes dissenyats específicament per a sintaxis visuals. A les definicions de la sintaxi i la semàntica del llenguatge hem vist com la utilització de diagrames té cabuda en la notació matemàtica habitual. Així hem definit el concepte de diagrama ben format, equivalent a la fórmula en un llenguatge textual, i hem definit la semàntica denotacional del llenguatge proposat de forma similar a com es fa en la lògica de primer ordre. També hem utilitzat un mecanisme nou, una gramàtica CMG, per tal de donar una definició generativa del llenguatge. Finalment hem vist com cal adaptar les tècniques d'implementació del llenguatge a la sintaxi visual

mitjançant un editor dirigit per la sintaxi, que ens simplifiqui la fase d'anàlisi sintàctic.

El capítol 3 reflecteix l'intent de donar la màxima amplitud expressiva a la nostra proposta d'una sintaxi visual per a la lògica de primer ordre. Després d'haver explorat totes les seves possibilitats ens hem centrat en el disseny d'un llenguatge de programació lògica, fent èmfasi en la seva semàntica operacional. El nostre objectiu era trobar una regla d'inferència visual, similar a la resolució SLD, i que capturés visualment les intuïcions del seu funcionament. Per a aconseguir-ho hem seleccionat un subconjunt del llenguatge, definit al capítol 3, de potència expressiva similar a les clàusules de Horn (clàusules de Horn sense predicats d'aritat zero) i un nombre més reduït de constructors. La regla d'inferència visual proposada es basa en l'aplicació de la unificació de literals visuals, de forma semblant a la resolució textual i a la unificació de termes textuals. Una diferència respecte la unificació textual és la possibilitat de portar a terme unificacions múltiples quan la definició del predicat dóna diferents elements. D'aquesta forma, mentre que en la programació lògica és habitual obtenir les solucions una a una, mitjançant aquest llenguatge és possible mostrar diferents solucions en un sol diagrama resposta. També és possible representar en el diagrama resposta la traça de cadascuna de les solucions. Creiem que aquesta capacitat de representació dels diagrames resposta és una de les aportacions més significatives d'aquesta sintaxi visual. Així mateix, aquest llenguatge visual i concretament la seva semàntica operacional tenen un grau de proximitat molt elevat a la semàntica matemàtica del llenguatge. Com a treball futur queda pendent demostrar la completesa del sistema d'inferència proposat. Mentre que la solidesa és evident, creiem que la demostració de la completesa és particularment complexa i no és necessària demostrar-la dins els objectius d'aquesta tesi.

Una altra aportació d'aquesta tesi és l'aplicació d'aquesta sintaxi visual a d'altres camps. Els formalismes basats en el llenguatge de la lògica de primer ordre s'utilitzen en diferents àmbits. Creiem que la nostra proposta de sintaxi visual per a un subconjunt de la lògica de primer ordre es pot aplicar a alguns d'aquests formalismes, i obtenir representacions visuals que poden ser útils en aquests altres àmbits. Al capítol 6 hem mostrat com es pot aplicar el llenguatge visual de programació lògica en la representació visual d'esquemes de bases de dades deductives. Hem definit una variant de la notació visual proposada per a la programació lògica introduint-hi nous constructors per adaptar-la a les necessitats de les bases de dades deductives. Creiem que hem aportat una representació alternativa complementària als llenguatges textuals habituals que, essent totalment formal, permet definir esquemes de bases de dades deductives d'una forma senzilla i elegant. Altres treballs portats a terme (Agustí et al. [4, 6]) fan pensar en la viabilitat d'utilitzar una sintaxi visual similar en un llenguatge d'especificació formal.

Encara que aquesta tesi es presenta amb un cert retràs respecte la seva elaboració els resultats són encara plenament vigents. Per això té sentit parlar de futures línies d'investigació que aquesta tesi obre, tot aprofundint en els resultats obtinguts. Una primera línia oberta seria la utilització de diferents sintaxis

visuals. Creiem que el llenguatge visual proposat en aquesta tesi pot servir com a base d'un llenguatge visual heterogeni de programació lògica. Segons Barwise i Etchemendy [12] el raonament visual és intrínsicament heterogeni: el fet d'utilitzar representacions properes a la semàntica ens porta necessàriament a la utilització de diferents sintaxis, cadascuna adequada al que vol representar. Així, els sistemes de raonament i els llenguatges de programació haurien de ser heterogenis, és a dir, haurien de permetre la utilització simultània de vàries sintaxis, textuals o visuals. En el nostre cas, els diferents tipus d'elements estructurats com taules, matrius, tuples, etc. es podrien representar utilitzant les notacions diagramàtiques habituals. Un exemple és la representació visual de les tuples al capítol 6. Aquestes representacions s'utilitzarien dins el llenguatge visual com a elements especials dels conjunts. Creiem que la principal dificultat es troba en dissenyar aquestes sintaxis específiques de forma que la unificació de termes sigui possible i eficient.

Pel que fa a l'aplicació del llenguatge visual als esquemes de bases de dades deductives, el nostre treball de definició de la semàntica operacional al capítol 5 ens fa pensar que és factible utilitzar el llenguatge visual per a representar consultes a una base de dades deductiva i mostrar-ne els resultats visualment. Igual que en el llenguatge visual de programació lògica creiem que ens permetria mostrar diferents solucions alternatives juntament amb el camí seguit per a obtenir-les (la traça) en un sol diagrama resposta. També creiem interessant estudiar de quina forma les diferents propietats dels esquemes de bases de dades deductives, com ser permeses (*allowed*) o estratificades, es poden representar o fins i tot definir de forma visual. Un inconvenient de les bases de dades deductives és el caràcter pla de la definició de l'esquema; la falta de conexions explícites entre les diferents ocurrències d'un mateix predicat en diferents regles. Aquest problema també és present en certa manera en la programació lògica. En alguns entorns, com per exemple el LPA MacProlog, s'ha intentat solucionar mitjançant grafs de dependències. Creiem que seria interessant estudiar l'ús de mecanismes visuals dins el llenguatge proposat que tractin aquest problema, i que també poden ajudar a subsanar la manca d'escalabilitat que en alguns casos presenten els llenguatges visuals (Burnett et al. [19]).

Un altre treball important complementari al presentat en aquesta tesi és la visualització de programes lògics textuals (o esquemes textuals de bases de dades deductives) mitjançant el llenguatge visual. Això permetria visualitzar molts programes i esquemes existents alhora que ajudaria a l'usuari en la familiarització amb la sintaxi visual. La principal dificultat és trobar una distribució (layout) del diagrama resultant amb la notació secundària necessària per tal que resulti entenedor. S'han portat a terme moltes investigacions en el disseny d'algoritmes de dibuix de grafs (*graph layout*), però encara és un problema obert en el cas general.

La implementació del llenguatge visual mostrada al capítol 7 es troba encara a les fases inicials, d'aquí que les proves a individus de fora de l'equip de desenvolupament han estat molt limitades. No obstant s'ha pogut constatar que és possible implementar un llenguatge visual d'aquestes característiques de forma

eficient. Un dels principals problemes que es deriven de la utilització d'una sintaxi visual és la complexitat de l'anàlisi sintàctica. Hem vist com aquest problema s'ha resolt mitjançant un editor de diagrames dirigit per la sintaxi. Al capítol 7 dissenyem i implementem un editor dirigit per la sintaxi basat en la definició generativa del llenguatge visual. Aquest editor ens garanteix un grau suficient de correctesa sintàctica dels diagrames obtinguts, fet que elimina la necessitat de portar a terme l'anàlisi sintàctica d'aquests diagrames, alhora que proporciona a l'usuari una eina per a crear i editar diagrames de forma senzilla. Queda pendent dur a terme un estudi empíric que permeti contrastar la usabilitat del llenguatge visual enfront de les seves alternatives textuals, així com també un estudi de la pragmàtica del llenguatge. S'han dut a terme petites proves amb el prototip exposat al capítol 7, les quals han donat lloc a un projecte fi de carrera. No obstant, un estudi empíric ampli de la usabilitat i la pragmàtica del llenguatge proposat és d'una envergadura tal, que podria constituir una altra tesi doctoral per si mateix.

Publicacions de la Tesi

- Jaume Agustí, Jordi Puigsegur i Dave Robertson. A Visual Syntax for Logic and Logic Programming. *Journal of Visual Languages and Computing*, 9:399–427, 1998.

  `http://www.iiia.csic.es/~jpf/JVLC98.pdf`

- Jaume Agustí, Jordi Puigsegur, Dave Robertson i W. Marco Schorlemmer. Visual logic programming through set inclusion and chaining. In *CADE-13 Workshop on Visual Reasoning*, Agost 1996.

  `http://www.iiia.csic.es/~jpf/VR-CADE13.pdf`

- Jaume Agustí, Jordi Puigsegur i W. Marco Schorlemmer. Towards Specifying with Inclusions. *Mathware and Soft Computing*, 4(3):281–297, 1997.

  `http://www.iiia.csic.es/~jpf/Mathware97.pdf`

- Jaume Agustí, Jordi Puigsegur i W. Marco Schorlemmer. Query Answering by means of Diagram Transformations. In *Proceedings of the Conference on Flexible Query Answering Systems*, Roskilde, Denmark, May 1998. In Springer Lecture Notes in Artificial Intelligence, núm. 1495.

  `http://www.iiia.csic.es/~jpf/FQAS98.pdf`

- Jaume Agustí, Dave Robertson i Jordi Puigsegur. GraSp: A GRAphical SPecification Language for the Preliminary Specification of Logic Programs. Research Report 95/13, Institut d'Investigació en Intel·ligència Artificial, Bellaterra, Catalunya, 1995.

  `http://www.iiia.csic.es/~jpf/IIIA-95-13.pdf`

- Jordi Puigsegur i Jaume Agustí. Visual logic programming by means of diagram transformations. In *Proceedings of the APPIA-GULP-PRODE Joint Conference on Declarative Programming*, A Coruña, Spain, Juliol 1998.

  `http://www.iiia.csic.es/~jpf/AGP98.pdf`

- Jordi Puigsegur, Jaume Agustí i Joan-Antoni Pastor. Towards Visual Schemas in Deductive Databases. In *Proceedings of the 9th International*

Conference and Workshop on Database and Expert System Applications, August 1998. In Springer Lecture Notes in Computer Science, núm. 1460.

`http://www.iiia.csic.es/~jpf/DEXA98.pdf`

- Jordi Puigsegur, Jaume Agustí i Dave Robertson. A Visual Logic Programming Language. In *Proceedings of the 12th IEEE Symposium on Visual Languages*, Boulder, Colorado, USA, Setembre 1996.

  `http://www.iiia.csic.es/~jpf/IEEE-VL96.pdf`

- Jordi Puigsegur, Joan-Antoni Pastor i Jaume Agustí. Defining and Translating Visual Schemas for Deductive Databases. Research Report 98-31-R, Departament de Llenguatges i Sistemes d'Informació, Universitat Politècnica de Catalunya, Barcelona, Catalunya, Juny 1998.

  `http://www.iiia.csic.es/~jpf/LSI-98-31-R.pdf`

- Jordi Puigsegur, W. Marco Schorlemmer i Jaume Agustí. From Queries to Answers in Visual Logic Programming. In *Proceedings of the 13th IEEE Symposium on Visual Languages*, Capri, Itàlia, Setembre 1997.

  `http://www.iiia.csic.es/~jpf/IEEE-VL97.pdf`

Fonts d'informació

- Michael Anderson (`anderson@hartford.edu`) manté una plana Web molt completa amb informació i links relacionats amb el raonament diagramàtic:

  `http://uhavax.hartford.edu/diagrams/`
  `http://www.hcrc.ed.ac.uk/gal/Diagrams/`

- Dave Barker-Plummer (`dbp@csli.stanford.edu`) és el moderador de la *Diagrams mailing list*, una llista de distribució de correu electrònic, molt activa, sobre temes de raonament diagramàtic:

Arxius:	`ftp://ftp-csli.stanford.edu/pub/diagrams`
Subscripcions:	`diagrams-request@csli.stanford.edu`
Participació:	`diagrams@csli.stanford.edu`

- Zenon Kulpa manté una plana sobre raonament diagramàtic:

  `http://www.ippt.gov.pl/~zkulpa/diagrams/zkdiagr.html`

- Bertrand Ibrahim (`Bertrand.Ibrahim@cui.unige.ch`) manté una plana Web amb molta informació sobre llenguatges de programació visual i una base de dades amb bibliografia de l'àrea:

  `http://cuisung.unige.ch/Visual/`

- Margaret Burnett (`burnett@cs.orst.edu`) i altres col·laboradors mantenen una completa base de dades de bibliografia sobre llenguatges de programació visual:

  `http://www.cs.orst.edu/~burnett/vpl.html`

- Un dels congressos més rellevants de l'àrea dels llenguatges visuals és el *IEEE Symposia on Human Centric Computing Languages and Environments*. Substitueix a l'*IEEE Symposium on Visual Languages* i cada any comprèn diferents symposiums de temàtiques variades:

- 2002 - Arlington, VA, EUA `http://www2.cs.fau.de/HCC02/`
  - Visual/Multimedia Programming and Software Engineering
  - Empirical Studies of Programmers
  - End-User and Domain-Specific Programming
- 2001 - Stresa, Itàlia `http://cui.unige.ch/eao/www/HCC01/`
  - Symposium on End User Programming
  - Symposium on Visual/Multimedia Approaches to Programming and Software Engineering
  - Symposium on Visual Languages and Formal Methods

Les darreres edicions del *IEEE Symposium on Visual Languages* van ser:

- 2000 - Seatle, EUA `http://www.cs.orst.edu/ burnett/vl2000/`
- 1999 - Tokyo, Japó `http://www.isl.hiroshima-u.ac.jp/vl99.html`
- 1998 - Halifax, Canada `http://www.pictorius.com/vl98/index.html`

- La revista *Journal of Visual Languages and Computing* és un dels referents en publicacions sobre llenguatges visuals i computació visual.

`http://www.cs.pitt.edu/~chang/ijvlc.html`
`http://www.academicpress.com/jvlc/`

Bibliografia

[1] Ramon Llull, *Enciclopedia Universal Ilustrada*, vol. 49, pàgs. 551–559, Espasa-Calpe, Barcelona, 1923.

[2] The History and Kinds of Logic, *The New Encyclopaedia Britannica (15a edició)*, vol. 23, 1985.

[3] Jaume Agustí, Jordi Puigsegur, i Dave Robertson. A Visual Syntax for Logic and Logic Programming. *Journal of Visual Languages and Computing*, 9: 399–427, 1998.

[4] Jaume Agustí, Jordi Puigsegur, i W. Marco Schorlemmer. Towards Specifying with Inclusions. *Mathware and Soft Computing*, 4(3):281–297, 1997.

[5] Jaume Agustí, Jordi Puigsegur, i W. Marco Schorlemmer. Query Answering by means of Diagram Transformations. A Troels Andreasen, Henning Christiansen, i Henrik L. Larsen, editors, *Flexible Query Answering Systems (FQAS'98)*, núm. 1495, Lecture Notes in Artificial Intelligence. Springer, Roskilde, Denmark, 1998.

[6] Jaume Agustí, Dave Robertson, i Jordi Puigsegur. GraSp: A GRAphical SPecification Language for the Preliminary Specification of Logic Programs. Research Report 95/13, Institut d'Investigació en Intel·ligència Artificial, Bellaterra, Catalonia, 1995.

[7] Gerard Allwein i Jon Barwise, editors. *Logical Reasoning with Diagrams*. Studies in Logic and Computation. Oxford University Press, New York, USA, 1997.

[8] Allen Ambler i Margaret Burnett. Influence of Visual Technology on the Evolution of Language Environments. *IEEE Computer*, pàgs. 9–22, 1989.

[9] Michael Anderson, Bernd Meyer, i Patrick Olivier, editors. *Diagrammatic Representation and Reasoning*. Springer Verlag, 2002.

[10] F. Bancilhon i R. Ramakrishnan. An amateur's introduction to recursive query processing strategies. A *Proc. of ACM Int. Conf. on Management of Data*, pàgs. 16–52, Washington D.C., 1986.

[11] Jon Barwise i John Etchemendy. *Hyperproof*. Núm. 42, CSLI Lecture Notes. Center for the Study of Language and Information, 1994.

[12] Jon Barwise i John Etchemendy. *Heterogeneous Logic*, cap. 8, pàgs. 170–200. A Allwein i Barwise [7], 1997.

[13] Jon Barwise i John Etchemendy. *Visual Information and Valid Reasoning*, cap. 1, pàgs. 3–26. A Allwein i Barwise [7], 1997.

[14] Jon Barwise i John Etchemendy. *Computers, Visualization and the Nature of Reasoning*, pàgs. 93–116. Blackwell, London, 1998.

[15] Jon Barwise, John Etchemendy, Gerard Allwein, Dave Barker-Plummer, i Albert Liu. *Language Proof and Logic*. CSLI and Seven Bridges Press, Stanford, 1999.

[16] Jon Barwise i Eric Hammer. *Diagrams and the Concept of a Logical System*, cap. 3, pàgs. 49–78. A Allwein i Barwise [7], 1997.

[17] Peter Bexte i Werner Künzel. Llullus, oder was der Computer im Mittelalter konnte. *Frankfurter Allgemeine Magazin*, (452), 1988.

[18] M. Burnett i M. Baker. A Classification System for Visual Programming Languages. *Journal of Visual Languages and Computing*, 5:287–300, 1994.

[19] Margaret M. Burnett, Marla J. Baker, Carisa Bohus, Paul Carlson, Sherry Yang, i Pieter van Zee. Scaling Up Visual Programming Languages. *IEEE Computer*, pàgs. 45–54, 1995.

[20] Margaret M. Burnett, Adele Goldberg, i Ted G. Lewis, editors. *Visual Object-Oriented Programming: Concepts and Environments*. Manning Publications Co., Greenwich, Connecticut, USA, 1995.

[21] Jo Calder. *How to Build a (Quite General) Linguistic Diagram Editor*, cap. 28. A Anderson et al. [9], 2002.

[22] Luca Cardelli. Two-dimensional Syntax for Functional Languages. A *Proceedings of Integrated Interactive Computing Systems*, pàgs. 107–109. Elsevier Science Publishers B.V., 1993.

[23] U.S. Chakravarthy, J. Grant, i J. Minker. Foundations of semantic query optimization for deductive databases. A J. Minker, editor, *Foundations of deductive databases and logic programming*, pàgs. 243–273. Morgan Kaufmann Publ., 1988.

[24] Shi-Kuo Chang. Visual Languages: A Tutorial and Survey. *IEEE Software*, pàgs. 29–39, 1987.

[25] Shi-Kuo Chang, Tadao Ichikawa, i Panos A. Ligomenides, editors. *Visual Languages*. Plenum Press, 1986.

[26] W. Citrin, M. Doherty, i B. Zorn. Formal Semantics of Control in a Completely Visual Programming Language. A *Proceedings of the 10th IEEE Symposium on Visual Languages*, St. Louis, Missouri, 1994. IEEE Computer Society Press.

[27] W. Citrin, M. Doherty, i B. Zorn. *The Design of a Completely Visual OOP Language*. Manning Publications Co., Greenwich, Connecticut, USA, 1995.

[28] Wayne Citrin, Richard Hall, i Benjamin Zorn. Programming with Visual Expressions. A *Proceedings of the 11th IEEE Symposium on Visual Languages*. IEEE Computer Society Press, 1995.

[29] Isabel F. Cruz i Peter S. Leveille. As You Like It: Personalized Database Visualization Using a Visual Language. *Journal of Visual Languages and Computing*, 12:525–549, 2001.

[30] Hendrik Decker. The range form of databases or: How to avoid floundering. A *Proc. of 5th. GAI*, Innsbruk, Austria, 1989.

[31] Kathi D. Fisler. *Exploiting the Potential of Diagrams in Guiding Hardware Reasoning*, cap. 10, pàgs. 225–256. A Allwein i Barwise [7], 1997.

[32] H. Gallaire, J. Minker, i J.M. Nicolas. Logic and databases: A deductive approach. *ACM Computing Surveys*, 16(2):153–185, 1984.

[33] Martin Gardner. *Logic Machines and Diagrams (2a edició)*. The University of Chicago Press, Chicago, Illinois, USA, 1982.

[34] Janice Glasgow, N. Hari Narayanan, i B. Chandrasekaran, editors. *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. AAAI Press, Menlo Park, California, USA, 1996.

[35] Ephraim P. Glinert, editor. *Visual Programming Environments: Applications and Issues*. IEEE Computer Society Press, Los Alamitos, California, USA, 1990.

[36] Ephraim P. Glinert, editor. *Visual Programming Environments: Paradigms and Systems*. IEEE Computer Society Press, Los Alamitos, California, USA, 1990.

[37] Eric Hammer. *Logic and Visual Information*. Studies in Logic, Language and Computation. CSLI and FoLLI, Stanford, California, USA, 1995.

[38] Eric Hammer. *Peircean Graphs for Propositional Logic*, cap. 6, pàgs. 129–147. A Allwein i Barwise [7], 1997.

[39] Eric Hammer i Norman Danner. *Towards a Model Theory of Venn Diagrams*, cap. 4, pàgs. 81–108. A Allwein i Barwise [7], 1997.

[40] Eric Hammer i Sun-Joo Shin. Euler's Visual Logic. *History and Philosophy and Logic*, (29):1–29, 1998.

[41] David Harel. On Visual Formalisms. *Communications of the ACM*, 31(5): 514–530, May 1988.

[42] Jordi Levy. *The Calculus of Refinements: a Formal Specificacion Model Based on Inclusions*. Tesi doctoral, Universitat Politècnica de Catalunya, 1994.

[43] Kenneth M. Kahn i Vijay A. Saraswat. Complete Visualizations of Concurrent Programs and their Executions. A *Proceedings of the 6th IEEE Symposium on Visual Languages*, Skokie, Illinois, 1990. IEEE Computer Society Press.

[44] Zenon Kulpa. Diagrammatic Representation and Reasoning. *Machine GRAPHICS & VISION*, 3(1/2):77–103, 1994.

[45] Didier Ladret i Michel Rueher. VLP: a Visual Logic Programming Language. *Journal of Visual Languages and Computing*, 2:163–168, 1991.

[46] Fred Lakin. *Spatial Parsing for Visual Languages*, pàgs. 35–85. A Chang et al. [25], 1986.

[47] J.W. Lloyd. *Foundations on Logic Programming (2a edició)*. Springer, 1987.

[48] J.W. Lloyd i R.W. Topor. Making prolog more expressive. *Journal of Logic Programming*, (3):225–240, 1984.

[49] Kim Marriott. Constraint Multiset Grammars. A *Proceedings of the 10th IEEE Symposium on Visual Languages*, St. Louis, Missouri, 1994. IEEE Computer Society Press.

[50] Kim Marriott i Bernd Meyer. Towards a Hierarchy of Visual Languages. A *Proceedings of the 12th IEEE Symposium on Visual Languages*, Boulder, Colorado, 1996. IEEE Computer Society Press.

[51] Kim Marriott i Bernd Meyer. On the Classification of Visual Languages by Grammar Hierarchies. *Journal of Visual Languages and Computing*, 8(4): 374–402, 1997.

[52] Kim Marriott i Bernd Meyer, editors. *Visual Language Theory*. Springer, 1998.

[53] Bernd Meyer, Kim Marriott, i Gerard Allwein. *Intelligent Diagrammatic Interfaces: State of the Art*, cap. 23. A Anderson et al. [9], 2002.

[54] Mark Minas. *Specifying Diagram Languages by Means of Hypergraph Grammars*, cap. 32. A Anderson et al. [9], 2002.

[55] Marc A. Najork. Programming in Three Dimensions. *Journal of Visual Languages and Computing*, 7:219–242, 1996.

[56] Marc-Alexander Najork. *Programming in Three Dimensions*. Tesi doctoral, University of Illinois at Urbana-Champaign, 1994.

[57] N.Houser, Don D. Roberts, i J. van Evra, editors. *Studies in the Logic of C. S. Peirce*. Indiana University Press, Bloomington, Indiana, USA, 1997.

[58] Pablo Noriega. Reflexiones Pragmáticas en torno al desciframiento del Códice de Santa María Asunción. *Butlletí de l'ACIA*, (6), 1996.

[59] L.F. Pau i H. Olason. Visual Logic Programming. *Journal of Visual Languages and Computing*, 2:3–15, 1991.

[60] Marian Petre. Why Looking isn't Always Seeing: Readership Skills and Graphical Programming. *Communications of the ACM*, 38(6):33–44, 1995.

[61] Jordi Puigsegur, Jaume Agustí, i Dave Robertson. A Visual Logic Programming Language. A *Proceedings of the 12th IEEE Symposium on Visual Languages*, Boulder, Colorado, USA, 1996.

[62] Don D. Roberts. *The Existential Graphs of Charles S. Peirce*. Mouton, The Hague, Netherlands, 1973.

[63] Ton Sales. Llull as Computer Scientist or why Llull was One of Us. A Miquel Bertran i Teoror Rus, editors, *Transformation-based Reactive Systems Development*, núm. 1231, Lecture Notes in Computer Science. Springer Verlag, Berlin, 1997.

[64] Ton Sales. Llull com a informàtic-avant-la-lettre. *Butlletí de l'ACIA*, (10-11), 1997.

[65] W. Marco Schorlemmer. *On Specifying and Reasoning with Special Relations*, volum 10, *Monografíes de l'IIIA*. Institut d'Investigació en Intel·ligència Artificial, 1999.

[66] W. Marco Schorlemmer, Jaume Agustí, i Jordi Puigsegur. On reasoning in category theory by diagram chasing. A *LICS'98 Workshop on Logic and Diagrammatic Information*, 1998.

[67] Hikmet Senay i Santos G. Lazzeri. Graphical Representation of Logic Programs and their Behaviour. A *Proceedings of the 7th IEEE Symposium on Visual Languages*, Kobe, Japan, 1991. IEEE Computer Society Press.

[68] J. Artur Serrano. The Use of Semantic Constraints on Diagram Editors. A *Proceedings of the 11th IEEE Symposium on Visual Languages*. IEEE Computer Society Press, 1995.

[69] Atsushi Shimojima. *On the Efficacy of Representation*. PhD thesis, Indiana University, 1996.

[70] Atsushi Shimojima. The Graphic-Linguistic Distinction. *ÇArtificial Intelligence Review*, 23:313–335, 1999.

[71] Sun-Joo Shin. *The logical Status of Diagrams*. Cambridge University Press, Cambridge, Massachusets, USA, 1994.

[72] Sun-Joo Shin. *Situation-Theoretic Account of Valid Reasoning with Venn Diagrams*, cap. 4, pàgs. 81–108. A Allwein i Barwise [7], 1997.

[73] Sun-Joo Shin. *Multiple Readings of Peirce's Alpha Graphs*, cap. 17. A Anderson et al. [9], 2002.

[74] Nan C. Shu. *Visual Programming*. Van Nostrand Reinhold Company, New York, USA, 1988.

[75] John F. Sowa. *Conceptual Structures. Information Processing in Mind and Machine*. Addison Wesley, 1984.

[76] John F. Sowa. Relating Diagrams to Logic. A Guy W. Mineau, Bernard Moulin, i John F. Sowa, editors, *Conceptual Graphs for Knowledge Representation, Proc. of the First Int. Conf. on Conceptual Structures, ICCS'93, Quebec City, Canada*, Lecture Notes in Artificial Intelligence (699). Springer Verlag, Berlin, 1993.

[77] Lindsey Spratt. *Seeing the Logic of Programming with Sets*. Tesi doctoral, University of Kansas, 1997.

[78] Lindsey Spratt i Allen Ambler. A Visual Logic Programming Language Based on Sets and Partitioning Constraints. A *Proceedings of the 9th IEEE Symposium on Visual Languages*, Bergen, Noruega, 1993. IEEE Computer Society Press.

[79] Nik Swoboda. *Designing Heterogeneous Reasoning Systems with a Case Study on FOL and Euler/Venn Reasoning*. Tesi doctoral, Indiana University, 2001.

[80] Nik Swoboda. *Implementing Euler / Venn Reasoning Systems*, cap. 21. A Anderson et al. [9], 2002.